Best Effort Measurement Based Congestion Control

Alan Henry David Miller
Faculty of Computing Science, Mathematics and Statistics
University of Glasgow

A dissertation submitted for the degree of
Doctor of Philosophy
September 2001

Abstract

Congestion is caused when the load presented to a communication infrastructure exceeds its optimum capacity and the measures in place to control it are inadequate. The effect of congestion is that resources that would otherwise be available to facilitate communication are wasted and the effectiveness of the infrastructure impaired.

Packet switching networks, of which the Internet can be considered an example, offer the benefit of high levels of resource utilisation through their ability to multiplex together data streams. This advantage carries with it the danger of congestion. The subject of this dissertation is congestion control on the Internet. Specifically it is advocated that the addition of explicit mechanisms for controlling congestion at the time-scale of a session would increase the control exercised over traffic and increase the quality of the service that is provided to users.

The original design of the Internet deliberately omited the inclusion of congestion control measures. Consequently when faced with the phenomena of congestion collapse, which threatened the very utility of the network, it became important to add congestion control onto the existing architecture. This was achieved by embedding congestion control algorithms in to the Transmission Control Protocol (TCP). The design assumed long lived connections and low bandwidth wide area links. Neither of these assumptions hold today. Consequently the congestion control mechanisms often unnecessarily limit access to the network increasing the latency experienced by users. They also discriminate in favour of large transfers at the expensive of the short interactive traffic that is typical of the World Wide Web. Measurements of web traffic and simulations of TCP's congestion control algorithms are used to demonstrate these assertions.

The addition of best effort congestion control at the session time-scale can address these problems resulting in traffic that is more responsive to the congestion signal generated by the network and in a better service to users. A significant proportion of this dissertation is devoted to demonstrating the benefits of such session level congestion control and in developing a practical approach to its implementation. The design of a Location Information Server (LIS), which passively monitors TCP data streams to discover the network characteristics of the paths to aggregates of destinations, and makes estimates of these characteristics available to hosts in a timely manner is presented. Significant development work has been invested in building an implementation of an LIS. The effectiveness of the LIS was then evaluated using live Internet experiments.

The results of these experiments demonstrate that: i) estimates of congestion remain valid for significants periods of time, ii) can be communicated to hosts in a timely fashion and iii) can be used to derive appropriate starting values for TCP control variables. The consequence of this dynamic initialisation is that Slow Start is made largely redundant and TCP can move directly to the Congestion Avoidance state. This in turn results in average window sizes that are appropriate for the level of congestion and in a reduction in the latency attributable to the data transfer phase of TCP connections.

Whilst TCP accounts for around 90% of Internet traffic there are important traffic classes for which the reliable byte-stream service it provides are not appropriate. These can be an important source of congestion information and may also benefit from the addition of session level congestion control. Consequently it would be desirable to extend the LIS to cater for such traffic types. The design for such an extension, which integrates QoS feedback from Real Time Control Protocol packets with that collected from TCP streams is presented. The design for a group based conferencing application that utilises the feedback provided by an LIS to configure itself at the start of a session thereby providing a more predictable level of service to users is also given.

The approach to congestion control advocated in this dissertation is not intended to relace the existing Internet congestion control mechanisms, which typically operate within the confines of connections. Rather it is intended to address limitations in the existing approach by adding control at the session time-scale.

Acknowledgments

There are numerous people who have in one way or another made it possible for me to complete this dissertation. I would like to thank Derek Mcauley for giving me the opportunity to embark on this research, for posing many of the questions that provoked it, for offering valuable guidance at important junctures and for his comments on drafts of this document. I would also like to thank Richard Black for his insistence on accuracy and intellectual rigour and Peter Dickman for encouraging me to see this work through to its completion and for his comments on several revisions of the dissertation. I would also like to thank Colin Allison who heads my research group at St Andrews University for his patience and guidance and Pete Lindsay for proof reading large chunks of the dissertation.

Horst Mayerdinks, Ian McDonald and Rolf Neugebar were contempories of mine at Glasgow University and have been valuable sounding boards for my ideas. Ian Pratt has made facilities available to me at Cambridge Computer Laboratory and together with Andrew Moore has offered valuable advice.

On a personal note I would also like to thank Lynn my partner for all her support during the gestation of this project, without which I would undoubtedly have given up, long ago. Finally, it remains for me to thank my mother and father, Sue Johnston and Henry Miller for all their support and help over the years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

At the start of the 1980's the Internet was an experimental network supporting a few thousand users, mostly engaged in scientific research, and was funded almost entirely by the American Department of Defence. The next two decades saw the Internet grow until it became an important component of millions of peoples professional and social lives, and draws its funding from a diverse set of institutions. With this expansion has come a diversification in how the Internet is utilised. Businesses advertise and sell over it, sports events are broadcast over it and friends use it to communicate over thousands of miles.

Yet in the mid to late 1980's the phenomenon of congestion collapse threatened to make the Internet unusable. This provided a stimulus for systematic research into congestion control for packet switched networks in general and the Internet in particular. Of course the problem of congestion is not confined to the Internet, it arises in one manifestation or another whenever a communications infrastructure or part therefore is put under strain by the load presented to it.

Throughout its existence the Internet has used the Transmission Control Protocol (TCP) to ensure that a message which arrives at its destination is the same as the one that was sent and to regulate the rate at which data is sent. Whilst alternative Transport Protocols are available and are increasingly being utilised, over 90% of traffic has and continues to be carried by TCP [Sch01]. TCP provides a reliable, connection oriented, byte stream service on top of the connection-less, unreliable Internet Protocol (IP). It forms part of the collection of protocols collectively known as TCP/IP [Ste94, Ste96]. TCP uses a sliding window protocol [Tan81] to allow multiple packets to be outstanding in the network at any point in time and limit the number of such packets.

In 1988, Van Jacobson's classic paper Congestion Control and Avoidance [Jac88] was published. This paper drew upon preceding work [CJ89, Edg83] and provided the blueprint that was to cure the Internet's susceptibility to congestion collapse. Congestion was to be tackled on an end-to-end basis, by algorithms embedded in the source code of TCP implementations. In particular two algorithms, which Van Jacobson named Slow Start (SS) and Congestion Avoidance (CA), would force connections to adapt to the bandwidth available on a path. These congestion control mechanisms rely upon inferred feedback about the state of the network rather than explicit feedback or congestion control implemented in the routers.

The success of these measures is demonstrated by the size and growth rate of the Internet. This growth however has been accompanied by significant changes, which in turn mean that the assumptions against which the algorithms were formulated — low bandwidth and bulk transfers — often no longer hold. The variation in available bandwidth is now much higher; on some paths there may be gigabits per second available, on others only a few kilobits. Internet usage is dominated by web browsing, which usually generates small connections, many of which have insufficient

time to adapt to network conditions.

In addition there is an expansion in the volume of Internet traffic which is not carried by TCP and an increasing demand for the ability to provide different levels of service. These considerations have led to much debate about how to improve upon the Internet's existing and somewhat primitive traffic control mechanisms. This dissertation is a contribution to that debate.

The central thesis of this dissertation is that:

> *the addition of measurement based best effort congestion control to the Internet at the session time-scale would result in traffic that is more responsive to congestion and facilitate a significantly improved service to users. Furthermore such an addition is technically feasible.*

1.1.1 TCP Flow and Congestion Control

In this section an overview of TCP's congestion and flow control mechanims are given in order to provide background to the following discussion. TCP provides a duplex reliable byte stream service to applications. This service is particularly appropriate for the communication of data as a single bit error may corrupt the data. It is the transport protocol over which the Hyper Text Transfer Protocol (HTTP) is layered.

Central to the design of TCP is a combined, window flow control and reliability mechanism. Each byte sent, is associated with a sequence number. The receiver communicates to the sender the next byte that it is expecting to receive and the size of the window that it has available. Each end of the connection maintains variables which track the sequence numbers of the last byte acknowledged, the last byte sent and the highest sequence number that is allowed to be sent.

As acknowledgments are received the sender is able to transmit more packets, thus, allowing a steady flow of data from the sender to the receiver to be maintained. Multiple packets of data may be unacknowledged but more data cannot be sent than the receiver can accommodate in its buffer.

If a packet is dropped that packet's lowest sequence number and higher sequence numbers will be unacknowledged thereby preventing further data from being sent until it is retransmitted. The loss of a packet may be detected when a retransmit timer goes off or through the receipt of multiple duplicate acknowledgments. In either case the dropped packet is retransmitted.

TCP congestion control assumes that the dropping of a packet indicates the existence of congestion and the successful acknolwedgement of a packet indicates the absence of congestion. Whilst for flow control the receiver is responsible for regulating the transmission, of data for congestion control it is the sender that is responsible. This involves two variables maintained by the sender, the Congestion Window `cwnd` and the Slow Start Threshold `ssthresh`. Packets may only be sent if there is space available in both the flow control and the congestion control window. Two algorithms Slow Start and Congestion Avoidance regulate the size of `cwnd`.

At the start of a connection the congestion window is initialised to a small number usually the size of a single packet. When that connection enters the ESTABLISHED state a single segment is sent. When that packet is acknolweged the congestion window is increased in size by one and two more segments are sent. Whilst the connection is in the Slow Start state the congestion window is increased by the number of acknowledgments received, thus it grows exponentially.

The initial Slow Start phase of a connection ends when a packet loss is detected. If the loss was detected through a retransmit timeout then `ssthresh` is set to half of the current value of `cwnd` which is in turn set to one. A new Slow Start phase is then entered. If the packet loss is detected through duplicate acknowledgments `cwnd` and `ssthresh` are both set to half of the current value of `cwnd` and Congestion Avoidance is entered.

During congestion avoidance the congestion window is increased, by one over the maximum seg-

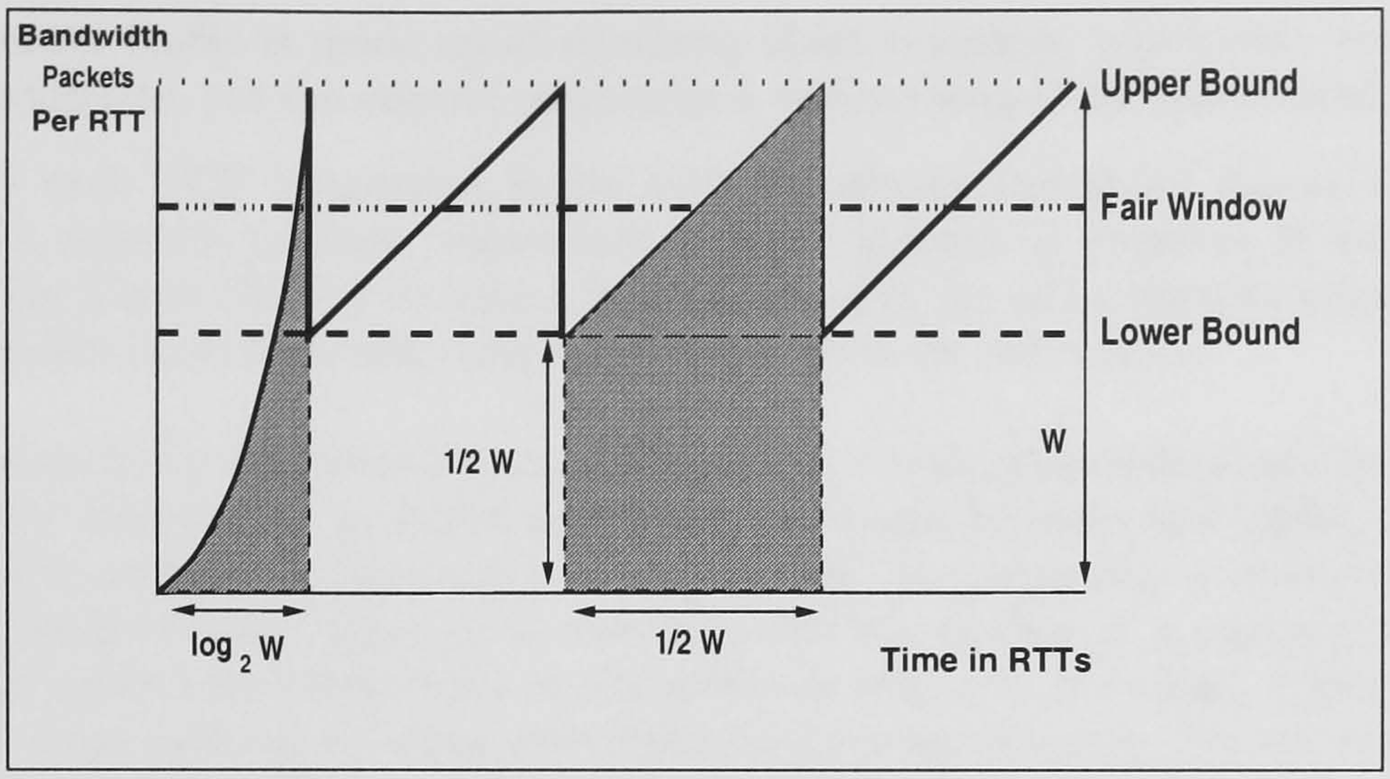


Figure 1.1: **Slow Start and Congestion Avoidance: This graph shows the evolution of TCP congestion window under deterministic loss assumptions. W is the maximimum window size during steady state.**

ment size, for each acknowledgment received, or equivalently by one packet for each window worth of acknowledgments. This amounts to an additive increase in the congestion window and represents the steady state behaviour of TCP. Behaviour upon the detection of loss is the same as during Slow Start.

Figure 1.1 illustrates the evolution of a connection's congestion window under deterministic loss conditions and where packet loss is detected through duplicate acknowledgements. After the initial Slow Start phase the congestion window oscillates in a saw tooth pattern around the fair window size.

Typically when the server has transmitted all its data a FIN is sent. The client then acknowledges that FIN. After communication with the application the client will then send a FIN of its own, which the server in turn acknowledges.

1.1.2 Observations about Congestion Control

The thesis statement flows from five observations which are central to this work.

1. Congestion occurs when the load presented to a communications infrastructure rises to such a level that resources, which would previously have been used in transportation, are no longer available. The result is a reduction in efficiency.

2. Effective congestion control allows a communication infrastructure to continue to operate efficiently, when the load presented by sources would otherwise cause a deterioration in service. This goal is achieved without unduly restricting access to the infrastructure at times when the presented load is at or below optimum levels.

3. A long stated aim of computer networks is to make it appear as though the services being offered by the network are actually being provided by a single host. The network should be transparent. For graphical user interfaces, low response times are critical in achieving this.

4. There are two fundamental tensions at the heart of Internet congestion control. The first is between the Internet Architecture and the Congestion Control mechanisms: the Internet Architecture is connection-less yet congestion control is located within the connection oriented TCP. The second is between the nature of Internet traffic and the control mechanisms:

most Internet traffic is made up of relatively short transfers, which only take a few round trips to complete, yet the control mechanisms assume long lived connections.

5. Currently each TCP connection starts with no information about the network, initialises its control variables to static values and uses the absence or presence of loss over multiple Round Trip Times (RTTs) to infer a fair window size. Its adaptation to network conditions is based solely upon feedback received in response to its own traffic.

In this dissertation it is argued that increasing the scope — both geographical and temporal — from which congestion information is drawn allows the mismatch between the traffic on the Internet and its congestion control mechanisms to be addressed. Consequently a connection's access to bandwidth is increased when there is no shortage, and the danger of a connection overshooting its fair share of bandwidth when there is contention is reduced. For many transfers this will in turn lessen the delay suffered by users and therefore move the Internet towards realising the goal of transparency.

1.2 The Original Contributions of this Dissertation

In validating the thesis statement of this dissertation a number of original contributions have been made. These lie in four areas: i) the analysis of TCP congestion control algorithms using a combination of measurement and simulation, ii) the development of a functional decomposition of congestion control [IMG+79], iii) the design implementation and evaluation of a Location Information Server (LIS) for the Internet and iv) the drawing of lessons relevant to protocol design.

1. Experimental and Investigative: Surprisingly, given the volume of research publications on the Internet there have been few studies that combine measurment of actual traffic with consideration of the implications for TCP's congestion control algorithms. Much of the literature lies in the realm of analysis, simulation and specially generated or probe traffic. Of those that are based on real traffic most are located solely or primarily in the US [Mog92, CaGP93, Pax94b, CBP94, BPS+98]. In addition there is a tendency to focus upon one aspect of the traffic being measured such as the RTT, the dynamics of Fast Retransmissions or Ack compression.

   The study presented in Chapter 4 combines simulation and the analasis of traffic measurements. The traces were captured in the UK providing an additional perspective on the Internet to the literature. Considering measurments of; connection sizes, levels of congestion, offered window size the gaps between connections and their relationship to the dynamics of TCP congestion and flow control allows an evaluation of TCP traffic management to be made. A deterministic simulator was developed, which allows the behaviour of TCP algorithms to be explored across a large parameter space and nouvelle techniques for the visualisation of the comparative behaviour of TCP connections were utilised.

   Whilst it has long been suggested that TCP does not work well with short connections, this is often treated as the exceptional case. This study suggests that it is, in fact, the common case. Furthermore it was found that a significant proportion of connections rely upon flow control mechanisms to prevent aggressive behaviour resulting from the exponential increase of Slow Start. It is observed that this dependeny may be dangerous in the future as increasing the flow control window which, is desirable on paths with a large bandwidth delay product, would remove the constraint.

   The distribution of average levels of congestion on network paths was investigated. It was found that there is a wide variation across different paths. This leads to the conclusion that sharing of state between connections that share a path may be advantageous.

   The temporal separation between connections, is measured. It is found that a large proportion of connections are separated by time scales orders of magnitude larger than RTTs. This

places a limit on the effectiveness of sharing congestion information between connections. Fortunately, experiments that are presented in Chapter 6 show that the value of congestion information is sufficiently long lived to bridge the gap between connections.

Finally, although most Internet connections are short the majority of packets are carried in large longer-lived connections. This suggests that the Internet may be resilient to changes in start-up regimes.

2. Analytical: Malek [IMG+79] presents a functional decomposition of congestion control, where the congested router is responsible for detecting congestion and signaling it to the source host, which is in turn responsible for adjusting its rate of send. This decomposition is developed and applied to the session time-scale in Chapter 5.

   An explicit signal from which a host could derive a fair starting rate or window size would allow short connections to utilised available bandwidth without causing congestion. On the Internet congestion is signaled by dropped packets. In the future some form of Explicit Congestion Notification (ECN) is likely to be introduced. In both cases the signal is binary in nature and a fair window size can only be derived after multiple rounds of feedback.

   This contradiction between the need for an explicit congestion estimate and the binary nature of Internet congestion signaling can be resolved by deploying resources at the edge of the network. These resources provide the function of monitoring the congestion signal, detecting the start of connections and supplying hosts with an explicit estimate of congestion. The host may then use this estimate to determine a fair window size or rate.

   This approach makes it possible to get the main benefits of explicit rate feedback without paying the penalty of increased complexity in the core of the network.

3. Practical Design and Implementation: Chapter 5 presents the architecture, design and implementation of a Location Information Server, which can be used to facilitate measurement based congestion control. Chapter 6 presents an evaluation.

   The LIS is located on the WAN/LAN interface, and uses passive monitoring of TCP traffic to extract network level information such as congestion events and round trip times. This information extraction occurs within a connection layer where per connection state is maintained. Each reading is then passed to a location layer where statistics are maintained on a per location basis. A location is defined as an aggregation of hosts, where traffic to each is likely to experience similar network conditions.

   Upon the detection of a new connection, a Location Information Packet (LIP), which contains predictions of network conditions, is sent to the local host participating in the connection.

   The local host makes use of the information it is supplied to dynamically initialise the Congestion Window (`cwnd`) and the Slow Start Threshold (`ssthresh`) to values that are appropriate for the predicted level of congestion. After initialisation the normal algorithms would apply. This can be thought of as facilitating session or connection level measurement based congestion control.

   The evaluation of the LIS shows that, statistically, dynamic initialisation results in an increase in the strength of the negative correlation between congestion and window sizes, a reduction in the latency suffered by most users and no increase in congestion.

   The design of the LIS is extended in Chapter 7 to facilitate the extraction of congestion information from real time and multicast data. The issue of how to integrate such QoS feedback with existing data and the use to which it can be put by real-time applications are also addressed.

4. Lessons for Protocol Design: The main draw back of the Location Information Server flows from deficiencies in the Architecture of the Internet. The absence of network level signaling of congestion information forces the LIS to extract network level QoS feedback from transport level packet headers. Consequently in order to extract QoS feedback from non TCP

traffic explicit extension of the LIS is required. An approach to extracting QoS information from RTP traffic and integrating it with that extracted from streams is presented. This is fortunate as streamed video and sound is likely to continue increasing on the Internet and is a potentially rich source of congestion information.

These observations lead to the conclusion that there is a need for a generic, protocol independent, congestion signaling mechanism that is meaningful to intermediate points. This requirement could be met by designing an explicit congestion notification mechanism. Unfortunately the current ECN assumes that signaling in the return path will be protocol dependent. One beneficial effect of this dissertation might be a reconsideration of the requirements for Internet explicit congestion notification.

1.3 Historical Background Material

This dissertation starts by discussing the link between the fundamental advantage of packet switching and the problem of congestion. The multiplexing together of multiple data streams allows higher levels of resource utilisation, but also means that shortages may occur resulting in dropped packets and wasted network resources. This is accompanied by a concrete discussion of the issues facing early packet switched networks. In particular the control of congestion and the monitoring of traffic was recognised as a central priority in the design of early packet switched networks.

Congestion control, traffic management and monitoring were however lower priorities when the Internet architecture was being designed. There were two reasons for this contrast. Firstly, it was hoped that traffic management mechanisms installed in existing networks would render Internet congestion control unnecessary [VGK78]. Secondly, external aggression was identified as the primary threat to network integrity; consequently seamless recovery from physical failures in the network infrastructure was adopted as a central aim, at the expense of efficient traffic management [Cer80a, Cla88]. This reflected the military mind-set of DARPA.

In the early eighties the phenomena of congestion collapse began to be reported on TCP/IP networks [Nag84b]. During a period of congestion collapse retransmissions dominate network traffic and the amount of useful work shrinks to a fraction of the network's capacity. By the late 1980's congestion collapse had become so common that it threatened the future expansion and basic functionality of the Internet.

In particular the combination of high bandwidth Local Area Networks being inter-connected by low bandwidth Wide Area Links was identified as a major contributing factor to the problem [Jai86].

These problems with congestion sparked a period of intense research which culminated in the addition of the Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery algorithms to TCP [Ste97]. For the next decade the Internet relied almost exclusively upon these algorithms for explicit congestion control.

Van Jacobson's classic paper [JB90], where these algorithms were advocated, focused on preventing TCP from massively overshooting the physical bandwidth available to a Wide Area connection. The paper assumed that the steady state would dominate during a connection's life time. The transfers used to demonstrate the effectiveness of Slow Start and Congestion Avoidance were around 20,000 segments long.

The congestion control mechanisms introduced to the Internet were shaped by the need for them to be added to an already existing architecture, rather than being incorporated as part of the original design process. This resulted in a concentration of functionality in end hosts, which limits the effectiveness of the mechanisms introduced. They have however significantly reduced the occurrence of congestion and contributed to the avoidance of congestion collapse. Thus the central aims have been realized.

The Internet has continued to grow and evolve since the introduction of congestion control. There has however been relatively little change in the congestion control mechanisms themselves. When

it is considered how much has changed in computing between 1988 and 2000 this is remarkable in itself.

Slow Start limits the degree that a connection can overshoot the physical bandwidth to a factor of two. The absolute limit of physical bandwidth may be important in a significant number of cases; for example where there is a low bandwidth connection linking a source or destination to the Internet. The growth in bandwidth means that at a route's bottleneck router there is often sufficient bandwidth for each connection, but congestion is caused when the aggregate demands of a large number of connections multiplexed together take the network beyond its optimum operating point. Here the fair distribution of available bandwidth between competing connections and the limitation of the bandwidth that each connection can utilize are the issues that need to be addressed.

Traffic on the Internet was often categorized as being either low bandwidth and interactive or high bandwidth bulk transfers. The traffic management algorithms in TCP are optimized for these two classes of traffic. Karn's [KP87, Ste94] algorithm prevents the Internet from becoming dominated by a large number of "tiny-grams" and Slow Start and Congestion Avoidance limit the bandwidth available to a connection during each round trip time.

In the mid nineties this categorization was challenged by the phenomenal growth of the World Wide Web which is today responsible for over 70% of traffic [OC01, CAI01]. This in part reflected the transition from textual to graphical user interfaces. The resulting traffic contains a significant graphical element, the bandwidth requirements of which are higher than for traditional telnet traffic and the timeliness constraints more severe than for traditional bulk transfers.

One result is a large number of small connections, which can be seen as the graphical representation of the tiny-gram problem. Each connection has the full set up and tear down overhead, but may consist of only one data packet. Neither Karn's algorithm nor Slow Start and Congestion Avoidance regulate the bandwidth used by these connections.

A second consequence and the one addressed by this dissertation is that a significant number of WWW connections are large enough to interact with TCP's congestion control mechanisms but are not large enough to do so in the way intended. Although a high proportion of connections are short, long transfers still account for a large proportion of packets. For this reason it is still necessary for the bandwidth a connection receives to be controlled throughout its lifetime.

In the last decade the Internet has made the transition from what was primarily a research and government network to one which is public and open to the pressures of commerce. This has put pressure on the way in which congestion control algorithms are developed.

In part because of the perceived slowness of the Internet a number of products claiming to boost performance have come on to the market. Some of these products achieve an increase in performance by circumventing congestion control mechanisms and pay little regard to the impact on competing traffic [Flo01].

The paradigm of controlling the resources allocated by embedding algorithms in source code has been undermined. This has resulted in increased interest in putting enforcement mechanisms in Internet routers. It also poses the question of how should the development of congestion control algorithms be controlled. One strand of research draws upon game theory [Axe85, KM99] and claims that if a source is made aware of the implications of its resource usage, and a cost is attached to that resource usage then an incentive is created for the development of algorithms that are reactive to congestion feedback. This approach has the added advantage of allowing users to buy increased resources if they require it and thereby enabling differentiated resource provision, without requiring explicit signaling between the network and end hosts.

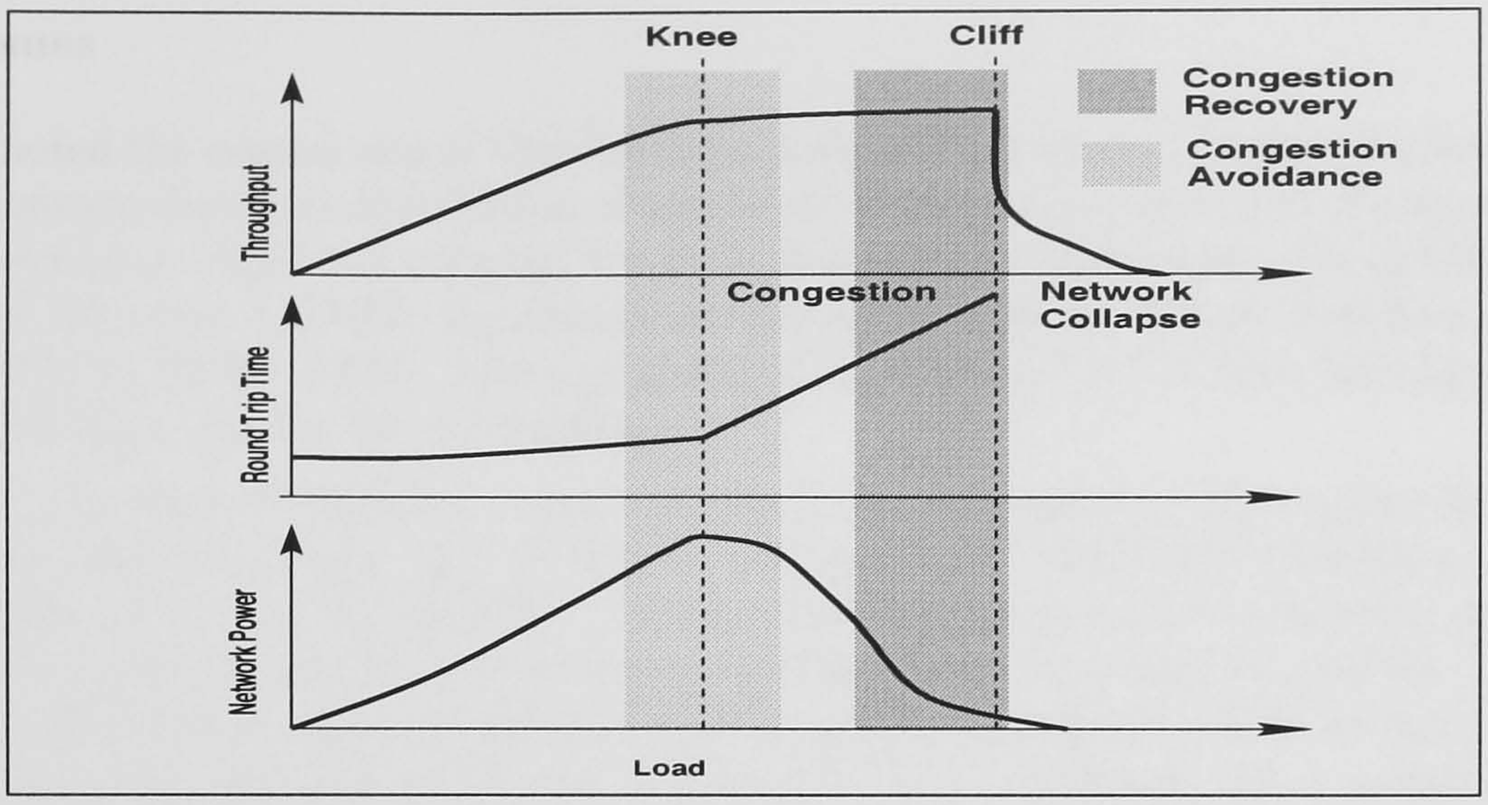


Figure 1.2: **Congestion Avoidance and Recovery: These graphs illustrate the different regions of operation that correspond to congestion avoidance and congestion recovery. Congestion avoidance is preferable because it allows the network to be operated at an optimal load**

1.4 Technical Background

The introduction of congestion control to TCP resulted in the concentration of congestion control functionality within end hosts, within a particular transport protocol and at the intra-connection time scale. This was largely a consequence of the need to add congestion control to an already existing Internet Architecture, which had adopted the datagram paradigm at the (inter)network layer.

Research contemporary to and preceding 1988 however, suggested that a functional division of labour between different network elements would be more effective in preventing congestion and promoting fairness between concurrent connections. This division of labour was first identified by Malek [IMG+79] and formed the starting point for much of Jain's [JR88a, RJC87, JR88b, CJ89] work and the approach to congestion control taken in the PUP inter-network [BSTM80].

1. Detection - The congested router is in the best position to detect the onset of congestion, as it has a global view of traffic passing through it.
2. Communication - Special packets [BSTM80, Nag84b] or setting bits in packet headers [RJ88] have been used to communicate the onset of congestion, before router queues became saturated and packets were dropped.
3. Reaction - End hosts should adjust their rate of send in response to the receipt of congestion notifications with the aim of preventing packet loss.
4. Enforcement - Boundary routers would be responsible for ensuring that hosts reacted correctly to congestion notifications.

Jain draws an important distinction between congestion avoidance and congestion control [JR88a]. In a congestion avoidance scheme the system is kept operating at the knee of the delay load graph shown in Figure 1.2, in congestion control the system is allowed to reach the cliff where packet loss occurs. By Jain's definitions TCP congestion avoidance is actually congestion control.

Following the adoption of Van Jacobson's algorithms and a significant increase in backbone bandwidth, the problem of congestion collapse was effectively eliminated and that of congestion diminished.

Fairness Issues

As has been noted the central aim of Van Jacobson's algorithms was to prevent congestion collapse, it was not to ensure the equal distribution of bandwidth between concurrent TCP connections. The congestion avoidance algorithm adopted for TCP was however based upon the Additive Increase Multiplicative Decrease (AIMD) algorithms developed by Jain for Binary Feedback Congestion Avoidance (BFCA) [RJ88, RJ90]. Jain's proposals did achieve a form of Max-Min fairness [Kes97] between connections sharing the same path.

The way that the network signaled congestion and hosts responded to that signal were both key features of the BFCA scheme. Specifically it was necessary that hosts sharing a path receive similar feedback. This was achieved by routers indicating the presence or absence of congestion by setting a bit in the packet header. Hosts would take an average over two RTTs. If congestion was indicated the window would be reduced by one eighth otherwise it would be increased by one. For connections using different paths, but sharing the same bottleneck link a variation on Max - Min fairness is achieved, as the increase and decrease algorithms would result in a similar window size. The throughput of a connection is a function of the window size and the RTT, consequently hosts communicating over a shorter path receive a larger bandwidth.

$$Throughput = \frac{Window}{RTT} \tag{1.1}$$

When the AIMD algorithms were put into TCP, the signal used to indicate the presence of congestion was packet loss, conversely the lack of packet loss indicates the absence of congestion. The consequence of this is that concurrent connections do not receive the same feedback; only the connections which happen to lose a packet in a congestion event will detect congestion, all others will continue increasing their congestion window. One consequence of this with respect to fairness is that the convergence of congestion windows, even for a long connection may not occur. Thus TCP does not achieve the approximation to Max-Min fairness that BFCA does, but may achieve a weaker form of fairness which may be called *statistical* fairnesss [WC92]. Each connection (on a shared path) has an equal probability of utilising bandwidth. A consequence of this is that in order to improve upon TCP's statistical fairness it is necessary to change the way that congestion is signaled by the network.

Fairness is not achieved between long and short connections because the low bandwidth utilization achieved while a connection is ramping up to its steady state dominates for short connections while the higher bandwidth achieved in the steady state dominates for long connections.

To summarize, congestion avoidance is based upon algorithms that can achieve a window version of Max-Min fairness. The combination of algorithm and signal used by TCP means that at best statistical window Max-Min fairness is achieved. These observations assume that it is the steady state behaviour which dominates during a connection's lifetime.

Binary Feedback and Start Up

The normal procedure for TCP is to initialize variables to default values. These values may then be updated in the light of feedback from the network. The main variables for congestion control are `cwnd` which controls the size of the congestion window and the Slow Start Threshold `ssthresh` which defines the window size that the transition between Slow Start and Congestion Avoidance occurs at.

Conditions on different paths vary by several orders of magnitude. Round Trip Times vary from several milli-seconds to several seconds (Figure 4.3). The level of packet re-ordering ranges from thousandths of a percent to 20% percent and more [Pax96a]. The physical bottleneck bandwidth ranges from a few kilobits/sec to gigabits per second. The probability of a connection receiving an implicit congestion notification ranges from practically zero to twenty or thirty percent

(Figure 4.2).

For all connections to start off with the same default values, means that to achieve optimal performance the process of adjustment must be rapid. Secondly, it is likely that the procedure of Initialisation and convergence must be repeated again and again every time a connection is started. A significant drawback of using a binary signal to adjust the congestion window, is that it takes multiple cycles for the equilibrium operating position to be reached. This is efficient when connections are long, the period between start-up and reaching the steady state is relatively short and when network conditions are relatively stable.

Motivated by the desire to allow a connection to rapidly adjust to changing network conditions, a class of algorithms that calculate an explicit rate were developed. These have mainly been targeted at ATM, although a similar approach has been proposed for the Internet. There are drawbacks to explicit rate feedback: it increases complexity and load on routers and it requires either the generation of special control packets or change in packet headers.

At the router, resources need to be allocated to allow the fair rate to be calculated. It is precisely at this point that resources may be scarce. Secondly the rate needs to be communicated, either in a control packet or in the header of a data packet. In this case it will also need to be echoed back to the source. It can be argued that the benefits of this approach outweigh the costs, nevertheless where it has not been implemented the benefits do not accrue. Recent years have seen the advancement of proposals to spread the functionality of congestion control from being located in end hosts to assigning roles to other network elements in a similar way to that proposed by Malek in the 70's and Jain in the 80's.

Random Early Detection (RED) Gateways [FJ93a] aim to detect the onset of congestion by monitoring queue lengths they randomly drop packets if the queue goes beyond a minimum threshold and drop all incoming packets if the Queue length goes beyond a second threshold. There are two problems with this approach. Firstly, the use of random feedback again limits the fairness that can be achieved between connections. Secondly, the effect of the feedback on producing a reduction in the offered traffic depends upon the level of multiplexing at the host. If there is only one connection using the link, a dropped packet will result in a 50% reduction in the rate at which traffic is presented. If there are 1000 connections using the bottleneck the drop will result in only a 0.05% reduction in traffic.

If a RED router is using Explicit Congestion Notification (ECN) [RF99], instead of dropping a packet to signal congestion the packet is marked, and the mark is reflected back to the source by the destination in the transport header. Thus the randomness of feedback is maintained. ECNs received during the last window of a connection will have no effect on the behaviour of that connection. Due to the exponential increase of Slow Start up to half of the traffic in a connection may be contained in the last window, consequently this may be seen as a serious problem.

In general the use of marking instead of packet dropping opens up the possibility of congestion notifications being handed off from one data stream to another by the source, thereby enabling intelligent choices based upon the value of a data stream to the source (or destination). These proposals continue to assume long connections, where feedback during the lifetime of a connection is adequate. Whilst RED and ECN offer certain benefits to the Internet they do not address the effects of the mismatch between congestion control and traffic distribution as the congestion signal remains binary. In addition it is tied to the transport layer which obstructs the sharing of congestion feedback and remains random which limits the fairness of resource distribution that can be achieved.

1.5 Goals and Constraints

This section provides the link between an assessment of Internet Congestion Control and the design for enabling information sharing and introducing feedback at the connection and session

time scales. The discussion is structured into two subsections. The first describes the constraints that have been adopted for the design. The second summarises the positive goals that are to be achieved.

Constraints

Four main constraints were adopted; no extra processing should be required at core routers, there should not be a requirement for extra WAN bandwidth, no changes to TCP/IP headers should be necessary and the amount of extra local traffic generated should be kept to a minimum.

The congestion control scheme should not require WAN resources which include bandwidth on WAN links and the processing capabilities of routers. This follow from the observation that it is desirable that a congestion control scheme does not increase the resource usage at the point of congestion. Secondly, by adopting these constraints a design which does not require changes at the core of the Internet for successful deployment results. This is desirable because it allows changes to be implemented in a network attached to the Internet without requiring changes in other parts of the Internet. Administrative cooperation between different network facilitators is not therefore required.

No change in the TCP/IP headers is desirable because meeting it would also allows incremental deployment of the design and avoid a possibly protracted standardisation process. Limiting LAN traffic may be important because whilst the LAN is not likely to be the bottleneck for most connections it may be for some.

Goals

The central aim of the design is to address the mismatch between the needs of traffic carried by TCP and the effect of its congestion control mechanisms. The effectiveness of a congestion control scheme may be evaluated against three criteria; fairness, efficiency and optimization.

Fairness: some of the features of TCP congestion control and its relationship to fairness have already been discussed. It remains to identify areas that will be targeted for change.

The constraint of not requiring changes in core routers, TCP/IP headers and not requiring the generation of extra traffic rules out introducing an explicit mechanism for signaling network state from the core of the network to hosts or other local network elements. This curtails the possibility of improving upon the statistical character of fairness that is used by TCP.

The broad framework of what we have called statistical, max-min fairness is therefore accepted. Within this framework the aim is to improve fairness by reducing the inequality of resource allocation between different sized transfers. A secondary aim is that the design should be easily expandable to support proportional fairness.

Efficiency: a congestion control scheme should use the minimum possible amount of the congested resource. In this case the aim is met by minimise the usage of WAN bandwidth and the load on routers.

Optimisation: As has been discussed it is desirable for a congestion control scheme to allow a heavily loaded network to operate at the knee of the delay load curve and not to prevent access to the network if the offered load is less than this level. The central aim here is to improve optimisation by reducing limitations imposed on traffic when the load is below the optimal level. Initiatives such as RED move the operating point towards the knee in the presence of heavy load, the design should not act as a barrier to these developments.

1.6 Implementation Overview

An introduction to the implementation is presented in this section. An implementation which applies the principles of the thesis to the control of connections on the Internet has been built and evaluated. The implementation has three main elements. A monitor observes traffic and measures network characteristics. The monitor also observes when a connection is being set up and sends a control packet to relevant local host. Upon receipt of a control packet the host uses the information it contains to initialize start-up variables to appropriate values. Each of the three functional units of the implementation; monitoring, communication and host reaction are considered below.

In building an implementation of the LIS the design space was constrained by the desire to demonstrate its functionality on a real network (the Internet). This in turn led to a focus upon TCP, because the information in TCP headers allows inferences to be made about the state of network, such as the level of packet loss and the size of RTTs on the path. It will however be shown that the conclusions drawn also have relevance for future protocol designers.

Active Monitoring

TCP packet headers are captured using the Berkeley Packet Filter. TCP was chosen for two reasons; firstly it carries the bulk of Internet traffic, secondly network characteristics such as RTT, level of congestion and packet re-ordering cannot be deduced by passive observation at the IP layer due to the lack of an associations between packets and the absence of explicit signaling. The connection oriented and reliable nature of TCP gives rise to protocol features which allow the deduction of network conditions by observing a stream of TCP packet headers.

Upon receipt of a packet header it is passed to the monitor for processing. State is maintained by the monitor for each open connection and for the *path* between the local network and a set of destination networks.

The state which is maintained for each connection allows a number of measurements to be made. Repeat packets are detected and filtered so that a burst of dropped packets only counts as a single Implicit Congestion Notification (ICN). The RTT between the monitor and foreign host is measured once for each window of data[1]. A separate layer aggregates together readings to produce estimates of network conditions for particular network paths.

Communication

The information that the monitor extracts from TCP packet streams needs to be delivered to hosts before the transfer of data commences and should not delay the data transfer. There is a gap between the receipt of a SYN in a passive open or the sending of a SYN for an active open and the start of data transfer. This gap is utilised to allow the monitor to communicate with the local host.

The monitor detects the start of a connection by observing the arrival of a SYN packet. It then dispatches a Location Information Packet (LIP) packet, which is an extension to the existing ICMP source quench packet and contains QoS feedback. This approach has the advantage that only one packet is required per connection and that packet only needs to be transferred across the

[1]The state for connections and the state for paths are each maintained in two data structures. A hash table is used to enable quick access, and an ordered queue is maintained to enable the removal of stale connections and lightly used paths. When a connection is set up on a path either a new path data structure is placed to the head of the path queue or the relevant path is placed at the head of the queue. A Location is removed from the tail of the queue if a threshold number of paths is exceeded. When a connection is set up it is associated with a path. This arrangement enables efficient updating of state at both the connection and path layers. It also allows arbitrary limits on the number of connections and paths for which state will be maintained. In the absence of these limits the number of both would grow without limit over time causing memory requirements of the program to become too large. With them the monitor can run indefinitely within defined memory limits.

local network. No WAN resources are required.

Host Reaction

Upon receipt of an LIP by the local host it is passed to TCP. The congestion information, which is expressed as the proportion of data packets that result in the receipt of an ICN, is then used to calculate a starting value for the congestion window size. The RTT and variance in RTT are used to calculate starting values for the SRTT and the RTO. The calculation of start-up values from path advice allows connections to bypass Slow Start and start transmission at a fair window size.

1.7 Synopsis of the Dissertation

Congestion is a common feature of communication infrastructures and arises whenever too heavy a load is presented and effective congestion control mechanisms are not in place. The fact that volume and nature of Internet traffic and the capacity of the Internet is bound up with developments in computer technology and human communication means that even if congestion is eliminated at a particular historical point new circumstances will contrive to enable it to re-emerge.

The control of congestion was central to the design of early packet networks, the design of the Internet was exceptional in that it contained no explicit congesiton control scheme. This was in part a consequence of the design priorities for TCP/IP which were in turn influenced by the military priorities of DARPA. Namely the main threat to the integrity of the network was external.

The success and growth of the Internet meant that congestion quickly became a problem. This was in part associated with the mismatch between high bandwidth local area networks and low bandwidth wide area networks meaning that a single source could saturate a Path with traffic. As a consequence it became necessary to add congestion control post hoc to the internet architecture.

The mechanism used was to utilise packet loss to signal the presence of congestion. A congestion window was added to the sending TCP implementation which provided an upper limit on the amount of data and therefore the number of packets that could be outstanding without an acknowledgment. The size of the congestion window is adjusted in response to the congestion signal.

During the Slow Start stage the congestion window is increased exponentially in the absence of congestion. When loss is detected or the congestion window reaches the size of the Slow Start Threshold the Congestion Window switches out of the Slow Start phase and into Congestion Avoidance. During Congestion Avoidance the congestion window is additively increased for each window's worth of acknowledgments. Each time a packet loss is received the Slow Start Threshold is halved in size. If the packet loss is detected through a Retransmit Timeout (RTO) the congestion window is reduced in size to one and Slow Start is re-entered. If congestion is detected through the receipt of duplicate acknowledgments the congestion window is cut in size by half and Congestion Avoidance is continued. In this way during the lifetime of a connection the window size is able to adapt to the prevalent network conditions.

Whilst the congestion control algorithms that are embedded in TCP have been important in facilitating the continued growth of the Internet, there are a number of weaknesses. The central one is that information that is gained about the state of the network by one connection is thrown away at the end of that connection's lifetime and is consequently not available to be made use of by subsequent transfers. This problem may not be considered serious where connections are long lived, as the start up transients may be amortacised over the length of the connection with the consequence that the average window size is largely determined by TCP's steady state behaviour.

The emergence of the World Wide Web as the dominant form of traffic on the Internet and the use of web browsers as the main way in which users interact with WWW resources has meant that a large proportion of transfers are large enough to be governed by TCP's congestion control

algorithms but not large enough for Congestion Avoidance to be reached. Consequently even when the bandwidth is consistently available on a path such connections are unable to utilise it, thereby leading to the network being run at a sub optimal level and unnecessarily increasing latency. A second effect is that the size of the transfer becomes a significant factor in determining the bandwidth that it will receive. Large transfers receiving a greater allocation of bandwidth than small transfers, thus undermining the fairness characteristics of the network.

If at the start of a connection sufficient information were to be made available to enable the congestion window to be initialised to a value consistent with the level of congestion that exists on the path that is to be used this would reduce the bias in bandwidth allocation against short connections, reduce the latency that users experience and increase the responsiveness of traffic to the network's congestion signal. This amounts to adding congestion control at the session time-scale to TCP, how can it be achieved? One solution would be for the congested router to produce an explicit indication of the level of congestion. This solution has the draw back of increasing complexity at the core of the network and using resources that might otherwise be deployed in forwarding packets. A second solution, and the one advocated here, is to introduce an entity at the edge of the wide area network that converts the existing binary signal into an explicit congestion estimate and communicates this to hosts at the start of a connection. This approach has the added advantage of being compatible with the Internet as it now is. Only minor changes are required in the end hosts to enable them to take advantage of enhanced congestion feedback.

This dissertation reports on the design, implementation and evaluation of such an entity, a Location Information Server (LIS). The approach adopted is to passively monitor TCP data streams and to extract from them feedback about the state of the network, from this estimates about the likely network conditions on paths to aggregates of destinations are made and communicated to hosts in a timely fashion at the start of a connection. Passive monitoring has the advantage over generating probe traffic in that no extra wide area traffic is generated and that estimates to the most frequently used destinations are automatically based upon more data.

The LIS is evaluated by generating experimental traffic to a number of destinations. The traffic includes connections that use the normal static initialisation, and those that use input from an LIS to determine initial values for the size of the congestion window and the slow start threshold. It is shown that using the LIS results in a stronger negative correlation between the level of congestion and the average window size for a range of transfer sizes. For larger transfers there is little difference. There is little difference between the peak window size during the start up phase for static and dynamically initialised connections indicating that dynamic initialisation does not result in aggressive behaviour. In addition the duration of the transfer is cut significantly for a range of transfer sizes and congestion conditions resulting in a reduction of latency experienced by the user. In short LIS informed start up results in the required changes to TCP behaviour.

Whilst TCP accounts for a high proportion of network traffic there are significant traffic types for which the service it provides is not suited. In particular real time traffic is not well served by the delays introduced by TCP's loss recovery mechanisms and multicast traffic can not be accommodated by TCP's unicast service. This dissertation also addresses the application of session level best effort congestion control to these traffic types. It is argued that extending the LIS would enable it to extract QoS information from RTCP packets and integrate this with information extracted from TCP data streams. Furthermore a group based conferencing application which would be able to use such information to configure itself at the start of a session and thereby improve the QoS as perceived by users is presented.

In this way the thesis that the addition of best effort session level best effort congestion control is technically feasible and would significantly improve Internet congestion control is validated.

1.8 Structure of the Dissertation

The discussion in the dissertation is structured into three parts. The first part surveys the development of congestion control in packet switched networks; from Baran's original vision [Bar64a], to the introduction of congestion control to TCP and consists of Chapters 2 and 3.

In the second part (Chapter 4) an analysis of TCP dynamics which is based upon simulation of TCP's congestion control algorithms and analysis of traffic traces captured at the University of Glasgow is presented. These measurements show that most connections are indeed too short to adapt to network conditions. Consequently even if the network is being under-utilised a new connection will not be able to make use of the available bandwidth. This in turn results in many connections taking several RTTs more than necessary to complete, which adversely effects the utility a user receives from the network.

In the third part the design, implementation and evaluation of a Location Information Server (LIS) [Rud00], which facilitates the sharing of congestion information between hosts, is presented. Chapter 5 presents the LIS design. Chapter 6 presents an evaluation and Chapter 7 extends the design to facilitate the integration of congestion information extracted from RTP [AS96] data streams with that from TCP.

1.9 Summary

This chapter has introduced a number of themes that will be developed in greater detail in the body of the dissertation. It opened by presenting some background to congestion control on the Internet and the thesis of the dissertation that session level congestion control can improve the responsiveness to congestion and the service provided to users. The relevance of the thesis statement was then motivated by describing the workings of TCP's congestion control mechanisms and making a number of observations about Internet congestion control.

This was followed by the introduction of some historical and technical material realated to congestion control, which provides a broad brush description of the context against which the work of this disertation was conducted. The goals and constraints adopted for the work were presented to allow the reader to judge whether the project has been succesful in its own terms. In addition an overview of the main devlopemental work is provided so that the reader is aware of the technical approach taken to addressing the above stated goals. Next a precis or synopsis of the line of argument that runs through the dissertation, which is intended to guide the reader through the subsequent chapters, is proffered. In the final section an outline of the dissertation's structure is provided, so that the reader interested in particular aspects of the work can move directly to them.

Chapter 2

Congestion Control in Packet Switched Networks

2.1 Introduction

The TCP/IP protocol suite was designed to interconnect heterogeneous packet switched networks. This chapter discusses congestion control in such networks. The period covered is from the pioneering work of Baran at the RAND corporation in 1962, to the start of the eighties by which time significant experience had accumulated.

The discussion is structured broadly chronologically, so that developments can be traced and key advances placed in the context of previous work. At each stage both the way that congestion manifests itself and the schemes that are developed to control it are related to the networking architecture. The discussion is organised into four sections:

1. The consequences of pre and dynamic resource allocation in digital networks for congestion control
2. The origins of packet switching
3. The ARPANET
4. Datagram and virtual circuit networks

Circuit switching allows bandwidth and routes to be allocated for each call whilst packet switching allows them to be dynamically allocated at the granularity of individual packets. The fundamental difference in the form that congestion takes in circuit and packet switched networks is discussed in the first section. In the second part the main contributers to the discovery of packet switching are identified and their ideas on congestion control discussed. The ARPANET was the most important early packet network and is discussed in the third section. The success of the ARPANET helped stimulate the spread of packet switching. The fourth section focuses on congestion control in store and forward packet switched networks. A differentiation between networks that provide a datagram and a virtual circuit is identified and the implications of each approach for congestion control investigated. The chapter concludes with a summary of the main trends in congestion control up to 1981.

2.2 Pre vs Dynamic Resource Allocation

> *There have always been two fundamental and competing approaches to communications: pre-allocation and dynamic allocation of transmission bandwidth* [Rob78].

The postal system is an early example of dynamic allocation of bandwidth. When a letter arrives at a post office it is sorted according to address and placed in a bag to be forwarded. No bandwidth (space in the bag) is reserved in advance. When the system is overloaded a backlog of mail builds up in sorting offices causing mail to be delayed.

The telephone network is an example of the pre-allocation of resources, when a call is established a circuit with a fixed bandwidth is created between the two communicating entities. When the load on a telephone network increases, first the call takes longer to set up and then some calls may be refused.

There are two main reasons why dynamic resource allocation is desirable.

1. *Survivability* If a failure occurs in the communication infrastructure it is possible to allocate new resources to circumvent the problem.

2. *Resource Sharing* If bandwidth is reserved, it is unavailable for use by other sources, whether it is actually in use or not. If bandwidth is dynamically allocated, all the unused capacity is available. It is therefore possible to achieve higher levels of utilisation.

The dynamic allocation of bandwidth requires routing and sorting decisions to be made at each hop. The time taken to perform manual routing and sorting operations however would preclude the use of dynamic allocation in real time communication. Consequently up until the late 1960s real time remote communication was dominated by techniques that utilised the preallocation of resources (telephone, radio, television).

The advent of computers made possible the automation of routing and sorting operations, which widen the practicable application of dynamic resource allocation. The relationship between the cost of computing facilities and the cost of a communication infrastructure determines whether it is economically feasible to utilise computers to achieve higher levels of bandwidth utilisation. In the late sixties the cost of computing had fallen sufficiently to make the deployment of computers, to switch packets of data, economically beneficial in some circumstances [Rob74]. In networking, fixed bandwidth allocation is realised through circuit switched networks and dynamic allocation through packet switched networks.

2.2.1 Circuit Switching

In a digital circuit switched network, synchronous time division multiplexing is employed to allow multiple communication channels to utilise a single physical link. This is realised by a frame being repeatedly transmitted at regular intervals. Each frame has multiple sub channels multiplexed into it. The bits belonging to each sub-channel are inserted into predetermined positions within a frame.

At a switch the link on which each sub channel should be forwarded on is set up with the circuit. The sub-channel is identified by its position within a frame. Switching is made simple and no bandwidth is required for addressing information. The bandwidth available to each circuit is fixed for the duration of the call, and can not be be used by other channels if it is underutilised.

To establish a call a route needs to be chosen, bandwidth allocated and switches configured. This results in a significant overhead when a connection is made. If a failure prevents communication on a particular route, calls using that route will fail.

Under high loads congestion can occur. When network resources are insufficient to meet demand, some call set-ups will fail. The network resources expended in failed set ups are wasted, which results in a fall in the resources available for successful calls and a decline in the throughput achieved by the system. Once a call is set up, the resources it requires will have been allocated and congestion will not normally occur.

Congestion control in a circuit switched network needs to be applied whilst calls are being set up and is unnecessary within a connection.

2.2.2 Packet Switching

Whereas circuit switching takes no notice of the activity or the lack of activity within a particular call, the use of packet switching can allow bandwidth allocation to vary dynamically.

In a packet switched network multiple channels are asynchronously multiplexed together so that each channel receives a variable bandwidth, predetermined positions within a frame can no longer be used to identify channels and so some bandwidth needs to be used to provide an address for each block of data. The combination of data block and address together with control information make up a packet, from which the term packet switching is derived.

> *A packet of information is a finite sequence of bits, divided into a control header part and a data part. The header will contain enough information for the packet to be routed to its destination* [VGK78].

The use of addressed data blocks allows packets to be independently routed through the network. It is not necessary for a route to be set up and so the connection set up overhead is reduced. A route may also be changed during a connection. This may allow the network to recover from a node failure without dropping connections. Packets from the same connection traveling along different routes may however introduce jitter into the data stream if the delay characteristics of the routes are different.

The dynamic allocation of bandwidth introduces the problem of contention for bandwidth and buffers during the lifetime of a connection. Packets may have to be discarded when they try to use buffers that are not free. The dropped packets will have used and therefore wasted network resources. As network load rises, contention and therefore useful throughput may decrease resulting in congestion.

The advantages of packet switching are bound to the problem of congestion arising during a connections lifetime. The high speed at which a computer handles data internally means that it is capable of producing traffic which is characterised by short periods of demand for a large bandwidth. These bursts may be interspersed with long periods of low or zero demand [Kle78]. Computer traffic is bursty. The reservation of bandwidth to cope with the peaks in demand is expensive and packet switching is therefore an appropriate technology for the transport of computer generated traffic.

2.2.3 Summary

Packet switching became economically possible as a result of the decreasing cost of computing and was necessary for the economically viable interconnection of remote computing facilities. Packet switching also enables the building of communication networks that are resilient to failure.

In order to use packet switching efficiently, problems resulting from the contention for resources during a connection need to be overcome. These problems include, packet loss, packet duplication, resequencing of data and deadlock.

Inherent in the multiplexing together of bursty traffic sources is both the possibility of high levels of resource utilisation and the danger of congestion of those resources. Packet switching requires mechanisms to control traffic within a connection.

2.3 Origins of Packet Switching

The end of the Second World War ushered in an era of world economic expansion. This was accompanied by peacetime military budgets in the UK, USA and USSR that were unprecedented in human history. The race for space, the arms race and advances in computing were all intertwined. In particular the US defense department had large resources that were directed towards research and development.

It has been observed [Rob74] that by the 1960s the cost of computing had fallen sufficiently to contemplate the deployment of computers in communication infrastructures. Given the background at the time it is not surprising that the first realisation of this took place in a military context.

2.3.1 RAND

The earliest plan for a packet switching network was developed by Paul Baran of the RAND Corporation. In 1962 he carried out work commissioned by the US Air force that was published in 1964 [Bar64a, Bar64c]. The aim of the project was to design a command and control system that would continue to function even if large sections of it were destroyed in a nuclear or conventional war. The system was to be capable of carrying both voice and data and would deploy computers as switches in the network infrastructure but was not aimed at interconnecting computers.

The solution proposed was called The Distribute Adaptive Message Block Network and contained four central elements.

1. The communication infrastructure was to be a decentralised mesh with each node capable of making intelligent routing decisions.
2. Messages were to be split up into packets each of which contained a global address and could be independently routed to its destination.
3. Digital lines, television transmitters, microwave broadcasts and satellite were to be used to enable communication in different domains to be effectively integrated.
4. The network would be protected from congestion.

Congestion

The control of overload was seen as an essential element in the system. In particular Baran argued the systems performance during a conflict was likely to be compromised by two phenomena which could lead to overload: sections of the infrastructure may be destroyed, and large numbers of people would want to start communicating at the same time. The combination could lead to congestion even in an over-provisioned network.

Two traffic control mechanisms were proposed [Bar64b], one would allow the network to control its acceptance of traffic and the second would allow sites to manage their output of traffic. Control by the network would be exercised by choking the entry of new traffic whilst the processing of transit traffic would continue normally if overload was detected. Each node would monitor the use of input and output lines. If output lines usage rose above some threshold input traffic would be choked off. The reduction in input is achieved by the node alternating between accepting and rejecting new traffic at a fine granularity. To the users this would appear as a smoothed variation in bandwidth availability.

The second traffic control mechanism was concerned with allocating the total allowable data rate among local users. A traffic management device called a Communications Control Console was to be inserted on the interface between each site and the network. The amount of gross bandwidth that each establishment could use would be controlled by a communications officer utilising the

console. The gross allocation could be adjusted following negotiation with other sites if a temporary increase in bandwidth was required. The console would also partition a sites bandwidth between different classes of traffic. The partitioning could again be manually adjusted. Feedback would be provided by a series of dials.

2.3.2 National Physical Laboratory (NPL)

A second independent source of ideas about packet switching was the National Physical Laboratory (NPL) in the UK. A small single node packet switch network was built at the NPL and extensive simulations of larger networks were run to investigating network congestion.

Isarithmic Congestion Control

Davies found that datagram networks were prone to congestion if the load presented exceeded the network capacity. He observed that as load increased so to did throughput until a maximum value was reached, subsequent increases in load would lead to a decrease in throughput. An Isarithmic Congestion Control scheme that allowed the network to continue to operate at its optimum throughput level when the level of offered traffic would have produced congestion in an uncontrolled network was proposed.

The scheme limited the total number of packets in circulation. This was achieved by using a permit system, there existed in the network a fixed finite number of permits and small stores of permits were held at each node. When a packet was transmitted by a source node it acquired a permit which it carried to the destination. If a node had no permits it could not transmit a packet and the number of packets in circulation were thus limited. If a node was inactive and had a store of permits it would distribute some of them to neighbours preventing a node suffering from permanent starvation [Dav72].

Isarithmic congestion control was found to be effective in controlling global congestion, but if the load was heavy and unbalanced local congestion could occur. It also required the global management of permits because loss of permits within the system would throttle throughput and duplication of permits could lead to congestion. The management of permits would be a difficult problem in large networks.

2.3.3 Advanced Research Projects Agency (ARPA)

The ARPANET was the most important early packet switching network built.

In 1961 ARPA hired J. C. R. Licklider to oversee its new command and control initiative and changed the name of his office from the Command and Control Research Office to the broader Information Processing Techniques Office (IPTO). Licklider was interested in how individuals interact with computers. He was convinced that by linking together computers, communication and cooperation between researchers involved in projects funded by his office would be facilitated. Connecting every computer to every other computer by leased lines was not economical, because the addition of a new computer would involve interconnections to every other computer. The fixed bandwidth and dialing overhead on a telephone network was not suitable for bursty computer generated data traffic [Kle78]. Packet switching was the technology that needed to be created in order to facilitate economically viable communication between large numbers of computers.

Robert Taylor took over as head of the IPTO and continued the interest in computer networking. Larry Roberts joined IPTO in 1967 and headed the network development that became ARPANET. Roberts drew on the work by Davies and Baran on packet switching. In November 1967 at the Association for Computer Machinery (ACM) Symposium in Gatlinburg, Roger Scantlebury presented a paper on a small packet switching network planned by Donald Davies at the NPL.

Robert Taylor was present and was influenced by Scantleburies paper [O'N95]. Davies is credited with inventing the term packet switching [Rob78].

After the Gatlinburg conference Roberts found

> *this huge collection of reports ... which were sitting around the ARPA office, and suddenly I learned how to route packets. So we talked to Paul and used all of his concepts to put together the [ARPANET] proposal...* [O'N95]

Although ARPANET was funded, conceived and managed by ARPA it was built by outside contractors located in Universities and Industry. An immediate motivation for the creation of ARPANET was to allow the large one-of-a kind computers that were located in different parts of the US to be shared by ARPA research projects.

The computers that ARPA wanted to interconnect were not compatible with each other. Wesley Clark [O'N95, Rob78] proposed the attachment of an interface computer to each of the existing systems, which would pass messages between each host and the network. These interface computers became known as Interface Message Processors (IMPs). Host computers would only connect with the network through an IMP, allowing any part of the network, except the IMP-Host interface, to be modified without requiring changes in the host computers [HKO+70].

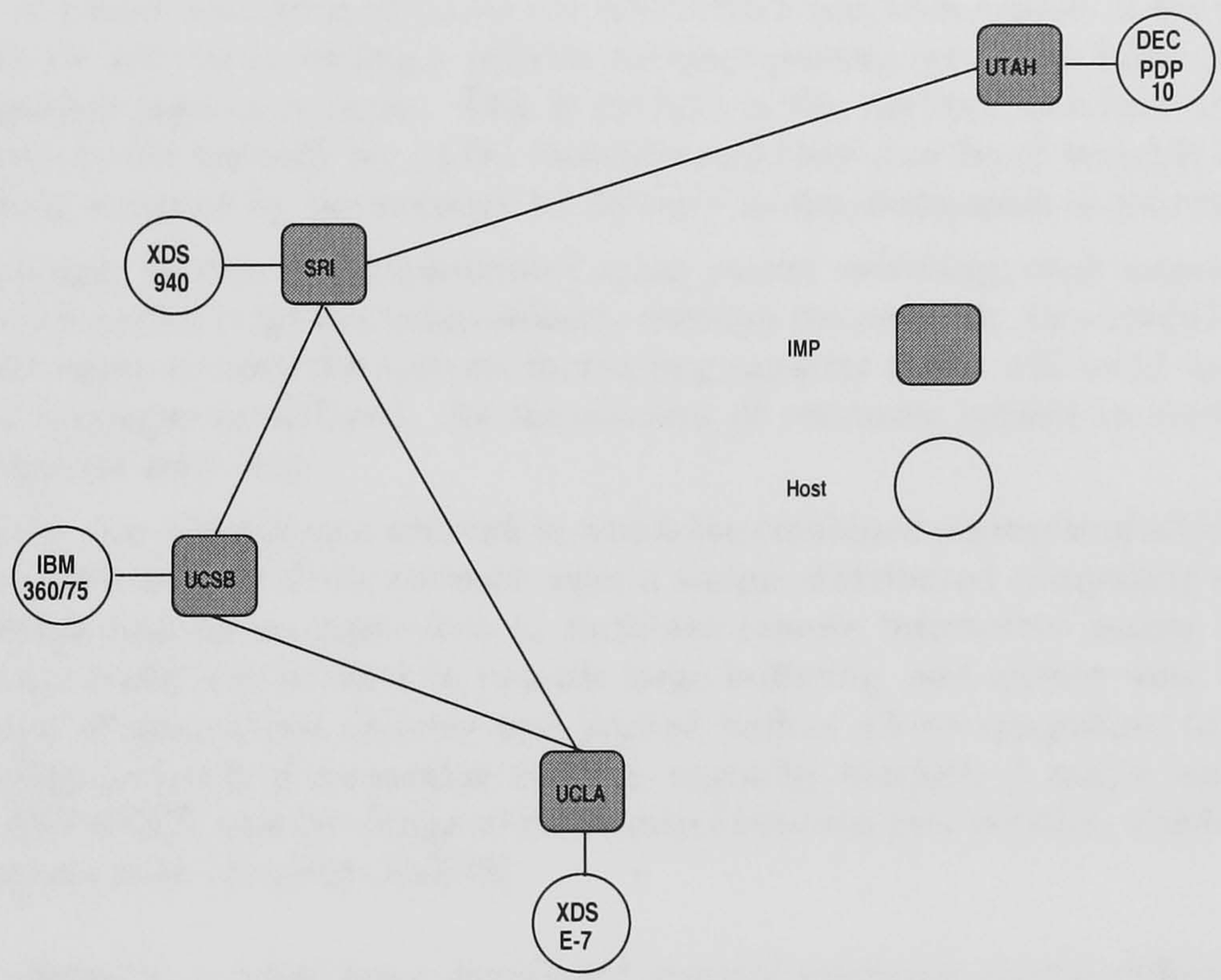


Figure 2.1: The Initial ARPANET configuration: IMPs and HOSTs

In 1969 the installation the ARPANET [Cro69] began. The first node was installed at the University of California which was to house the Network Measurement Center headed by L. Kleinrock. Each node could support up to four hosts. In December 1971 4 nodes and hosts were operating, by April 1972 this had risen to 23 hosts, by June 1974 62 hosts and by March 1977 there were 111 hosts [Rob78]. The IMPs were connected together using 50-kbit/s leased lines.

2.3.4 Summary

Congestion control was at the center of the packet networks designed by Baran and Davies. Baran proposed choking off input to the system when local network load rose above a predefined threshold.

Davies proposed limiting the global amount of packets allowed in the system. The design of the ARPANET drew heavily on the packet switching ideas developed by Baran and Davies. In the next section the nature of congestion on the ARPANET and the policies used to control it will be discussed.

2.4 The ARPANET

This section discusses the development of the ARPANET. The discussion is structured into three sections:

1. Architecture
2. The ARPANET protocols
3. Congestion and flow control

2.4.1 ARPANET Architecture

The influence of packet switching ideas on the ARPANET has been traced in Section 2.3.3. The ARPANET can be seen as providing a reliable message passing service to hosts [RW70], similar to previous message passing systems. This is evident in the network interface used. The units which are passed to the network are called messages and they may be of variable length. Once a message has been accepted by the network its delivery to the destination is guaranteed.

The reliable message service was implemented using packet switching; each message is split into packets [1] and each packet is routed independently through the network. In a traditional messaging system, if traffic input exceeds the systems forwarding capacity traffic will build up in queues and the delivery of messages be delayed. As the amount of resources needed to service the queues increases throughput may drop.

In ARPANET the aim of building a network in which the combined resources of all hosts computers are available to each host as if the network were a simple distributed computing system [RW70] meant that delays had to be controlled to facilitate remote interactive access. Consequently secondary storage could not be used to provide large buffering, and queues were limited in size. The combination of guaranteed delivery and limited buffers allows congestion to take the form of deadlock, where progress of competing traffic is mutually blocked. A major aim of congestion control in the ARPANET was the design of flow control between peer entities, which would prevent deadlock situations from occuring [Kah72].

> *...flow control is a multi layer distributed protocol involving several different levels. At each level the flow control implementation must be consistent and compatible with other protocol functions at the same level* [GK80]

It is therefore helpful to provide a breakdown of the layering in ARPANET before discussing the flow control at each layer.

2.4.2 Protocols

Like future computer networks the ARPANET needed protocols to enable communication between different hosts. The protocol stack is divided into a number of layers each of which is traversed by a message before reaching its destination. The main layers in the ARPANET protocol are shown in figure 2.2 [VGK78].

[1] If a message is small enough to fit in one packet it was not split

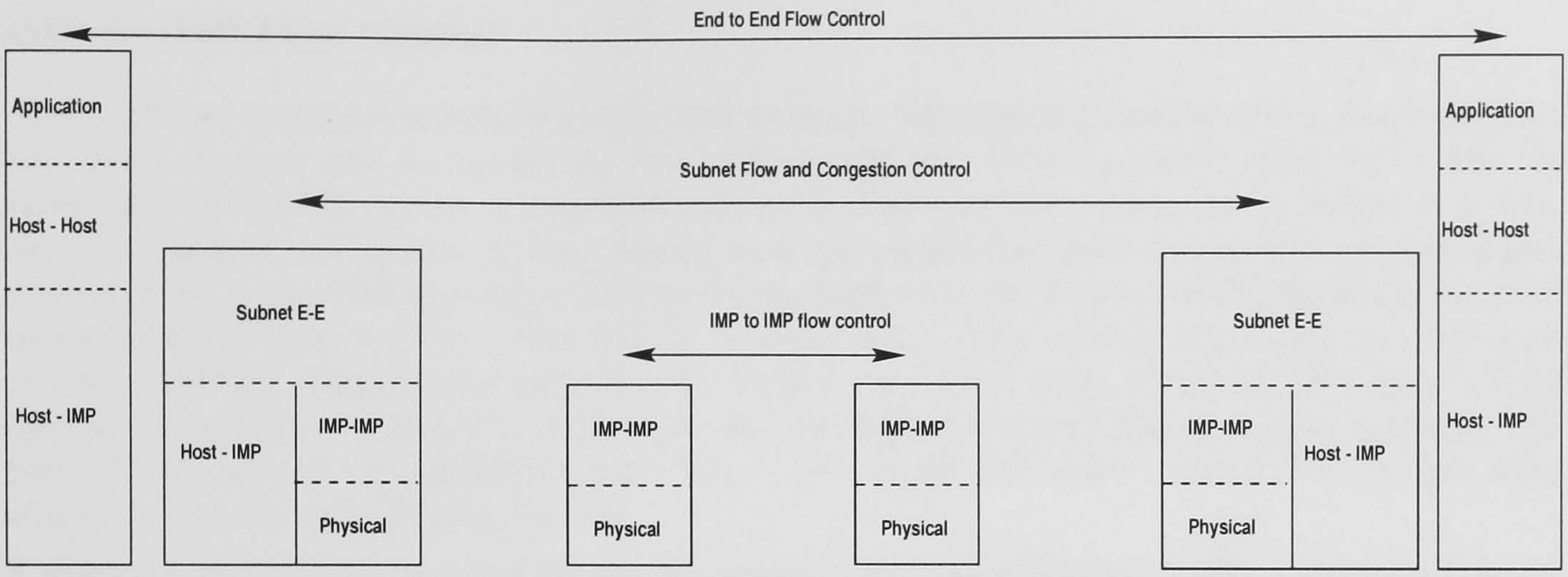


Figure 2.2: The main Layers in ARPANET protocol stack

The Host-to-Host [HKC70] protocol was called the Network Control Program (NCP). Its main functions were the setting up of a connection, control of the flow of data across the connection and connection termination. It provided a sequenced, flow controlled and error free service to applications. To realise a duplex connection two simplex connections were required.

The Host - IMP protocol maintained 32 virtual links between each pair of hosts on the network. These were utilised by NCP to set up a connection and to pass data to the subnet. Once a connection had been established a logical link number was used to address messages consequently a global address did not have to be passed to the subnet. Messages of a maximum 8064 data bits were passed across the Host to IMP interface. Once a message was accepted its delivery was guaranteed.

The subnet End-to-End protocol is responsible for splitting a message into packets with no more than 1008 bits. Outgoing packets have the address of the destination IMP placed in a header, they are then independently routed to the destination IMP. Upon receipt of a packet by the destination IMP it is stored until all the packets in its message are received. The message is then reassembled and sent to the destination host and a Request For Next Message (RFNM) dispatched to the sending IMP, which will then delete its copy of the message.

The IMP to IMP protocol is responsible for the transfer of packets between IMPs. Each IMP stores a packet and then uses a routing algorithm to forward it to a neighboring IMP. Packets are transferred reliably between IMPs. Reliability is provided at the IMP-IMP and subnet End-to-End levels.

2.4.3 Flow and Congestion Control

Flow control is performed at three levels:

1. IMP to IMP
2. Subnet End-to-End
3. Host to Host

A number of deadlock conditions were identified in ARPANET [Kah72]. These included reassembly lock up, direct store and forward lock up, indirect store and forward lock up and Christmas Day lock up. Each of the flow control levels will be discussed in turn below.

IMP-to-IMP Flow Control

IMP software multiplexes eight logically independent, full duplex channels onto a single physical line. Each channel uses a stop and go flow control protocol. When a packet needs to be sent the lowest idle channel is chosen, a sequence number is placed in the packet header before it is sent. No more packets can be sent on the channel until the packet has been successfully acknowledged. A copy of the packet is kept until an acknowledgment is received. If no acknowledgment is received before a 125ms time out the packet will be retransmitted. This scheme was based on work done at NPL [BSW69]. Direct store and forward deadlock is caused when a pair of IMPs have a large number of packets to forward to each other and no buffers are available to receive packets. This form of lock up can be avoided by each input and output line always being able to have some minimum amount of buffering [Kah72].

A more general solution to deadlock was investigated at Gesellschaft fur Mathematik und Dataenvarbeitung (GMD) which had an experimental network [GJMH81]. The method was based on buffer classes. A minimum number of buffers were reserved for each class of traffic. A packet was placed in a class according to the number of hops it had travelled with one class for each number of hops. In this way both direct and indirect lock up could be avoided. This method does not scale well, because the more hops that exist in the network the more classes that are needed, and was not implemented in the ARPANET.

Subnet End-to-End Congestion Control

The degree to which a source can cause congestion was limited by the End-to-End subnet protocol locally controlling the amount of data that could enter the network. This provides protection from a host flooding the network. Within the first years of the ARPANET three different granularities of control were used. In the original implementation

> *...no HOST can send two successive messages over the same link before the IMP at the destination has sent back a special message called an RFNM (Request for Next Message)* [Cro69]

Each pair of hosts had 32 logical links between them. In 1972 a replacement scheme was introduced which limited the number of outstanding messages between a pair of hosts to four [Wal74]. The third scheme was introduced in 1974. It replaced the limit of four outstanding packets per pair of IMPS with a limit of 8 outstanding messages between a pair of Hosts [Wal74].

In all three schemes the control is imposed locally and independently of the demand from other sources. If network load is low it may be unnecessarily restrictive and congestion can still occur if a high traffic load is offered at multiple IMPS. Reassembly deadlock is caused when all the buffers at an IMP are taken up by partially reassembled messages. The buffers cannot be released until the rest of the packets belonging to a message arrive. If these packets are blocked in the network be packets from other messages waiting to be transmitted to the IMP deadlock will occur.

The subnet end-to-end protocol was amended in 1972 to prevent reassembly lock up by requiring an IMP to reserve buffers at the destination IMP before starting to send a message. The reservation packet could carry data allowing single packet messages to avoid waiting for a reservation. Once a buffer was reserved a second and subsequent messages could utilise it without making a reservation, provided the message was transmitted promptly.

Host-to-Host Flow Control

NCP used an explicit buffer allocation mechanism to control flow between hosts. When a connection is made no data could be sent until a notification of the number of messages and bits

the receiving host is prepared to accept is received. As messages are sent these allocations are decremented. The receiving host may notify an increase in the allocation if needed [CPNK70].

2.4.4 Summary

The ARPANET successfully demonstrated that packet switching was an economical and effective form of communication. Within a few years of the ARPANET there were many packet switched networks in operation across the globe. A host to host reliable message passing service is combined with a datagram oriented subnet. Congestion control on the ARPANET had two main threads. The network protected itself from overload by the source IMP limiting the entry of traffic into the network. This method does not provide global protection nor is it able to adapt control to changing network conditions. A second strand of congestion control was deadlock avoidance. This was implemented by designing the End-to-End subnet protocol and the IMP-IMP protocol to prevent lock ups.

2.5 The Spread of Packet Switching

The growth of the ARPANET showed the utility of packet switching networks and provided a stimulus for their spread. ARPA funded the development of a packet radio network (PRNET) [KGBK78] and a satellite network (SATNET) [JBH78]. Research that would enable these networks to be interconnected was initiated in 1973 [CK74]. The consistency between these developments and Paul Baran's vision of interconnected line, microwave satellite and TV broadcast networks is remarkable.

A number of experimental networks were also developed including the one at GMD which has been discussed (section 2.4.3) in relation to deadlock avoidance. In France the CYCLADES [IMG+79], network was developed by Louis Pouzan and Herbert Zimmerman and was the source of several influential innovations.

Commercial packet networks were developed by Digital with the Digital Network Architecture (DNA) [Wec80] and IBM with the Systems Network Architecture (SNA) [Dha80]. These networks employed packet switching techniques to interconnect the respective companies computer products. National carriers that had previously provided telephone services began to utilise packet switching to provide Public Data Networks. An interface to these networks called X.25 [Ryb80] was developed and standardised by the CCITT. Examples of X.25 networks included DATAPAC in Canada and TRANSPAC in France.

This section focuses on store and forward packet switched networks with particular reference to the influence of the virtual circuit and datagram paradigms on the development of congestion control. First virtual circuit and datagram services and switching are treated generally, then specific networks are discussed, finally the discussion is summarised.

2.5.1 Virtual Circuits and Datagrams

The type of service provided by a network became an important point of differentiation within packet switching. On the one hand developers of Public Data Networks were concerned with the provision of a tangible resource to customers which could be managed and charged for. Connections provided a convenient unit and virtual circuits a defined service. This lead to the adoption of a Virtual Circuit Service. On the other hand some researchers preferred the provision of a minimal datagram service by the network which could be supplemented by hosts.

A Virtual Circuit Service involves the network emulating desirable characteristics of a circuit switched network, whilst preserving the increased network utilisation resulting from dynamic

bandwidth allocation. A virtual circuit was set up by requesting a connection to a destination, subsequent data that was presented would be guaranteed delivery to the destination in the sequence it was transmitted. The circuit features emulated are the guaranteed sequenced delivery of data, features not emulated include low levels of jitter.

A datagram is defined as:

> *a packet of data which contains sufficient addressing information for it to be routed independently, from other packets, from its source to destination* [VGK78]

A datagram service offers no guarantees about reliable delivery or maintenance of the sequence of data and requires no connection setup phase.

A virtual circuit service may be realised in two fundamental ways; either by being built on top of a datagram layer by virtual circuit switching. With virtual circuit switching a physical path down which all packets on a connection will travel is set up in the call set up phase. When datagrams are used internally no path need be set up but a connection between the network end points must be established. A datagram service may also be offered by a network that uses virtual circuit switching.

The CYCLADES and X.25 networks provided a Datagram and Virtual Circuit Service respectively and will be discussed below.

2.5.2 CYCLADES - A Datagram Network

CYCLADES was an experimental packet switched network developed in France from 1973. The main characteristic of CYCLADES was the migration of functionality previously associated with the network into hosts. Reliability, sequencing and flow control were all provided to applications on an end-to-end basis by the transport protocol and were not offered as a service by the network. The underlying network was called CIGALA and provided a simple datagram service. The result was a simplification of the communication infrastructure and several innovations which have out-lived the network. Here the discussion will focus on flow control, reliability and congestion avoidance.

Sliding Window Flow Control

Flow control and reliability are both provided by a sliding window flow control scheme. The reliable subnet in ARPANET meant that the Host-to-Host flow control scheme did not have to cope with lost or duplicated credit updates. With an unreliable subnet a lost update could result in the strangulation of a flow and duplicated updates in the flooding of a receivers buffers.

In CYCLADES consecutive segments are allocated monotonically incremented sequence numbers. The highest sequence number the receiver is prepared to accept is sent to the sender. Duplication is rendered harmless and loss is corrected by the receipt of the next credit update. If the credit update is sent as the highest sequence number of consecutive packets seen, plus a window size [Pou81] error control is integrated into the scheme, by the sender retransmitting unacknowledged packets after a time out. The combined error control and flow control mechanisms [LRCI79] pioneered in the CYCLADES network were to prove influential in future networks.

Congestion Control

Deadlock is avoided by relaxing the reliability requirement between nodes and within the network. This is acceptable because reliability is provided between hosts. Normally a copy of a packet is stored until an acknowledgment from the neighboring node is received. Buffer occupancy is monitored and if it rises above a threshold packets are discarded [Irl78]. In this way the manifestation of congestion as deadlock is avoided.

In the original CYCLADES design congestion control was confined to nodes dropping packets. It was found [2] that under heavy load the effective throughput achieved began to fall as an increasing amount of bandwidth was taken up with packets that were later dropped. Effective throughput could continue to decrease and spread until the whole network was paralyzed and congestion collapse occured. This lead to the observation that the control of congestion by discarding traffic

> *may be used as a last resort to prevent deadlocks, but it cannot be considered an efficient tool* [PZ78].

The development of a congestion control for the CYCLADES network took a different direction to existing mechanisms. Global Isarithmic control and Baran's proposal for local congestion control have been discussed. A second variant of local control called an Input Buffer Limits (IBL) scheme had been developed by W. Price [Pri79] as a means of controlling the local manifestations of congestion not prevented by Isarithmic control.

In an IBL scheme a proportion of buffers at each node were reserved for transit traffic, which can use any buffer but new traffic can only use a subset of the buffers at a node. This has the effect of strangling the entry of new traffic into the network when load is high and allowing transit traffic to continue to circulate. The scheme was shown to be effective in controlling local congestion and was followed by several variants, which differed in the proportion of buffers reserved for transit traffic [aMR81, Kam81].

The main weakness in the scheme is that congestion which is caused by the interaction of traffic from distant hosts will not be directly controlled. A secondary problem is that it could discriminate against a stream between two local hosts if it is competing with a stream between two distant hosts.

Channel Load Limitation (CLL)

The solution adopted was to reinstate the ability of the subnet to limit the flow of traffic from hosts, which existed in the ARPANET, but to allow the dynamic adaption of this control to measured network conditions. The detection of congestion was communicated to possibly distant hosts which were required to reduce their input to the network. The scheme was called Channel Load Limitation (CLL).

Central to CLL was the decomposition of congestion control [IMG+79] into the following functions.

1. monitoring of network state
2. communication of network state
3. user reaction
4. network enforcement.

Different functions are assigned to the most appropriate network element. The CLL scheme involves the interaction of different network elements; namely source hosts, congested nodes and network entry nodes. This is in contrast with the ARPANET where congestion is controlled by limiting the flow between peer network entities.

Network nodes monitor the bandwidth utilisation on links [Pou76], if a threshold is passed (70%)[3], a warning state is entered. The warning is communicated to hosts by sending a choke packet and to nodes through a routing update. Each node maintains a warning variable associated with the path between each pair of hosts. When a host receives a choke packet it should respond by slowing

[2] as Baran and Davies had found before

[3] Whether the link is in use is sampled periodically and a low pass filter is used to smooth load measurements

down its rate of send, resulting in a reduction in traffic at the congested link and exit from the warning state.

If congestion remains for an extended time a trip mechanism is activated[4] at the source node and all traffic entering the network which would be routed through the congested node is dropped. This scheme relies on the host knowing its current transfer rate so that it can slow down and on the first switch knowing whether packets will traverse the congested path. Both of these conditions involve treating packets as part of a stream rather than as independent individual entities, as such they conflict with the original datagram philosophy of the CYCLADES design.

The CLL congestion control system had the property of being able to react to local manifestations of congestion, unlike Isarithmic control, and being able to choke the source of the congestion if it is remote.

The essential feature of the CYCLADES network was the development of a pure datagram network by migrating functionality from the network to the hosts. This was found to conflict with the need to control congestion and a mechanism, which allowed the network to protect itself from misbehaving hosts was introduced. The scheme had the novel property of allowing hosts and nodes to adapt the load they placed on the network in response to distant and local measured network conditions. Other contributions included the demonstration that deadlock could be avoided by nodes dropping packets, and a combined end-to-end flow and error control scheme which was to be a model for future networks.

2.5.3 X.25 - A Virtual Circuit Example

In the mid seventies Public Data Networks often run by Telecom authorities began to emerge. X.25 was developed to provide a standard device-independent interface between a PDN and user equipment.

The X.25 interface is shaped by the needs of the telecom companies, in particular a reliable service, with mechanisms for charging and for the regulation of traffic was required. X.25 defines a standard interface between the network and users equipment, and a set of services that are associated with a virtual circuit, which also provides a useful unit for charging, congestion control and flow control. X.25 specifies a physical, frame and packet level interfaces which correspond to the physical, data-link and network levels in the ISO OSI reference model.

The physical layer specifies a duplex, point-to point synchronous circuit which provides a physical path between the Data Terminal Equipment (DTE) and the Data Circuit Terminating Equipment (DCE) or network. The frame level provides the packet level with an error free, variable delay link between the DTE and the network. This is achieved by providing error detection and recovery of transmission errors. Unrecoverable errors are reported to the packet level. The packet level specifies the way that data and control information is structured into packets. The control is placed in a packet header and includes addressing information.

In order to communicate, a DTE will set up a virtual circuit by sending a CALL REQUEST packet. This will include the calling and called addresses, control information and possibly data. During the call set up phase facility negotiation may take place. Facilities may include throughput priority and reverse charging.

If the network is suffering from congestion the network may block the call and signal to the caller the reason. Other reasons for blocking a call include an unobtainable address. Further data cannot be sent until the call is accepted. Once a call has been accepted a packet is identified with a virtual circuit by a logical channel number. An accepted call enters the data transfer phase. If congestion occurs the call may be cleared. Other reasons for clearing a call include a temporary failure in the network.

[4] In an alternative implementation the trip mechanism is employed when load at the congested link goes above a second threshold

Flow control is implemented for each virtual circuit which is made up of a concatenation of three virtual circuits as shown in figure 2.3. Once a packet reaches the far DCE an acknowledgment is sent to the originating DTE. An acknowledgment does not therefore guarantee that the packet has reached the destination DTE.

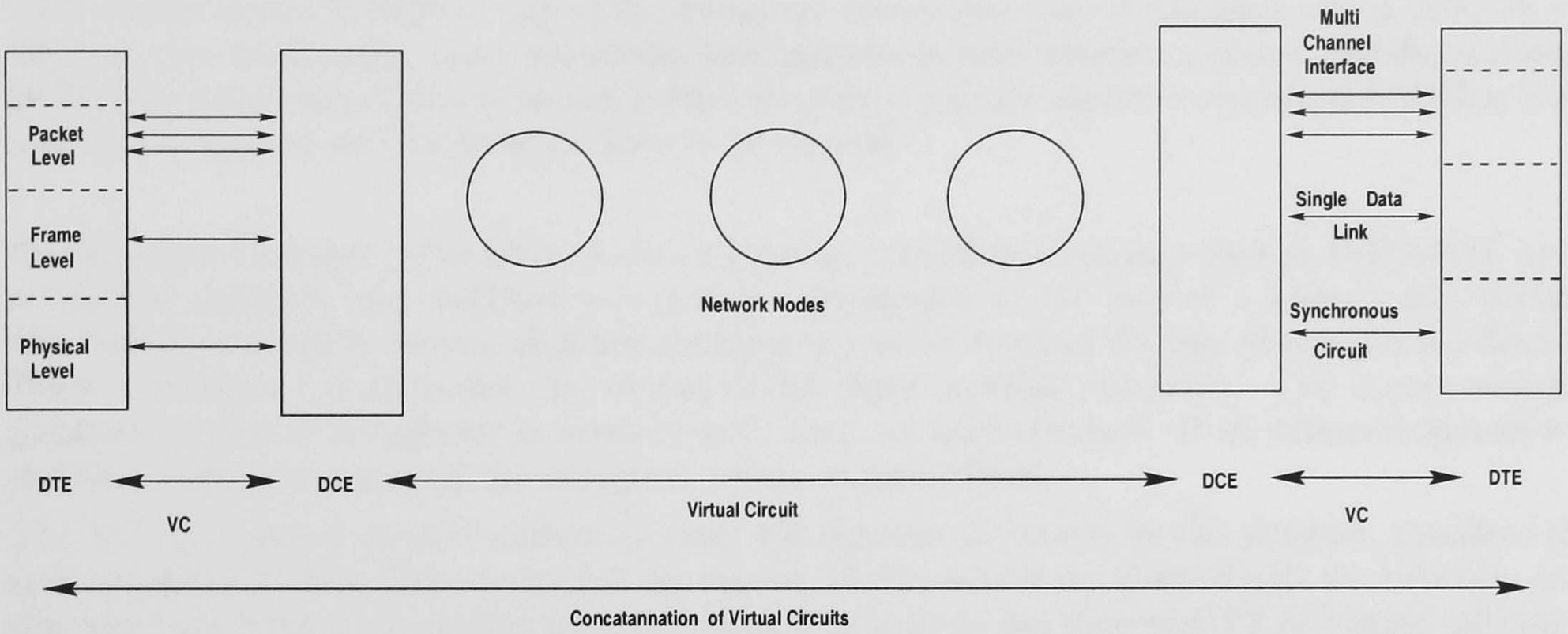


Figure 2.3: **X.25 Interfaces and Virtual Circuits Making a connection in X.25 requires multiple virtual circuits to be concatenated together**

A maximum number of packets are allowed to be in the network at once, this maximum is set when the call is established.

DATAPAC - Datagram Switching

DATAPAC was the Canadian public data network. In 1989 it was made up of 14 nodes and 24 trunk connections with 2000 users. The network access layer provided an X.25 interface and interconnection to other PDNs was provided using X.75 gateways [SM81]. The underlying network was provided by the Northern Telecom SL-10 Packet Switching System, a virtual circuit service is built on top of a datagram subnet and is utilised by the network access layer to provide X25 virtual circuits.

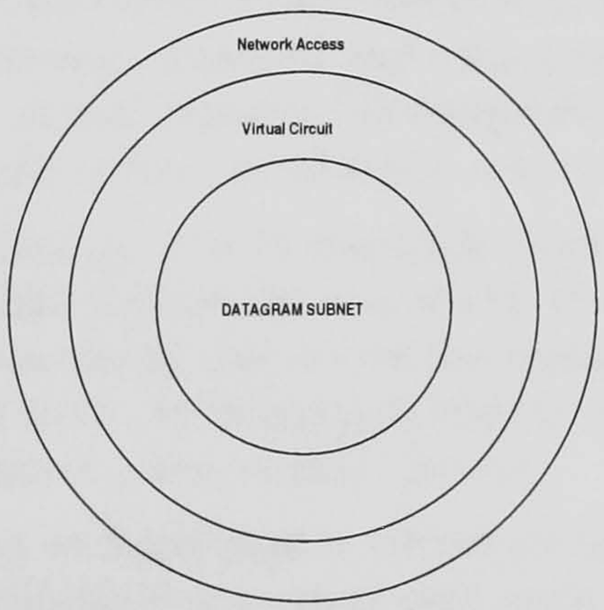


Figure 2.4: **Datapac Network Layers A virtual circuit service is provided to the network access layer, and is built on top of a datagram subnet**

The datagram service provided by the subnet is lossy, non sequenced and may include duplicates. Each datagram contains the address of the destination network node and is routed through the

network with the aid of routing tables. Changing network conditions may result in consecutive datagrams following different paths, resulting in a change in the sequence of data. Datagrams may be discarded to prevent buffer deadlock. The timeout and retransmit mechanisms may result in the duplication of datagrams.

Each virtual circuit is built on top of the datagram subnet and once it has been set up provides a loss free, non duplicating, order preserving communication path between a source and destination DCE. One advantage of this approach is that the subnet can dynamically reconfigure routing if a node is lost and so the call does not have to be cleared.

Congestion Control The aims of the congestion control mechanisms used in DATAPAC are to prevent deadlock and maintain a reasonable throughput in the face of a heavy load. If the free buffers at a switch become depleted datagrams may be dropped thereby preventing deadlock. When a datagram is discarded the virtual circuit layer provides reliability. The source switch maintains copies of datagrams in memory until they are acknowledged. If no acknowledgment is received within three second the datagram will be retransmitted.

The VC layer uses a window scheme to limit the number of packets in the network, therefore if acknowledgments slow down so to will the rate at which packets are allowed into the network. In this way back pressure is applied from the datagrams layer to the sources DTE and input into the network is throttled when congestion is experienced. The failure of four retransmissions is taken as an indication of severe congestion and will result in the connection being be cleared. Packets which are at their first node are dropped in preference to those in transit. Thus packets which have already used network resources are less likely to be discarded.

DATAPAC uses a datagram subnet and provides a virtual circuit service. Deadlock is avoided by allowing switches to drop packets and severe congestion is combatted by clearing calls.

TRANSPAC - Virtual Circuit Switching

TRANSPAC was the French public data network. It used virtual circuit switching to provide a virtual circuit service [Gie79]. When a call was received by the network it carried with it a throughput class, which indicated the maximum data rate that the call would require over a short time scale. Each node kept a record of the data rate required by its existing calls. In addition it measured the actual level of throughput and buffer use.

Each node on a path would decide whether to accept or reject a call based on its maximum commitment and actual use. The measured traffic levels would be below the capacity commited to and, depending on the ratio between measured and contracted data rates, nodes could commit to higher levels of traffic than the actual capacity. Consequently, it was possible for congestion to occur during a call and a congestion control mechanism was required.

Each virtual circuit had buffers allocated to it dynamically from a common pool. Three thresholds were operated on this pool. If buffer utilisation rose above the lowest threshold the node would not accept any more calls. If the threshold rose above the middle threshold the throughput on all current virtual circuits was slowed down, by delaying acknowledgments and if the highest threshold was passed the node would disconnect some virtual circuits.

The use of virtual circuit switching to implement a virtual circuit service allows admission control to be conducted on the basis of available resources at each node on the path. The dynamic sharing of buffers at each node allows a higher level of resource allocation than statically allocated buffers but requires the implementation of dynamic congestion control.

X.25 virtual circuits are a service which is provided to the DTE. Networks which utilise datagram or virtual circuit switching can both be used to implement X.25 virtual circuits. An advantage of a datagram switching implementation is the ability to reconfigure a route if a switch fails. Virtual circuit switching allows tighter control of resource management at each node on a path. A

major attraction of the virtual circuit service is that it provides a useful granularity for congestion control, flow control and charging.

2.5.4 Summary

The contrast between the development in congestion control in virtual circuit and datagram networks is marked. The absence of a connection in the datagram model means that no decision as to whether a connection should be accepted or rejected can be made. Congestion control is therefore exercised at the RTT time-scale (CLL) and packet time scales (IBL). The presence of a connection in networks that provide a virtual circuit service means that connection can be refused and terminated if congestion occurs. This does not exclude the use of methods at other time scales. Where a virtual circuit service is implemented using virtual circuit switching a known path is established and resource reservation is possible.

2.6 Issues in Packet-Network Interconnection

Packet switch networks connect computers, thereby making the resources of each computer accessible from all others on a network. Internets connect networks, thereby making the resource of every computer physically accessible to all others[5] on connected networks. The Internet has been defined as the sum of networks and computers that can be reached using the Internet Protocol (IP) [Com00]. In 1999 tens of thousands of networks and millions of hosts are included by this definition, and the numbers are continuing to grow.

The success of the Internet has made it almost synonymous with internetworking, however its design occupies a specific location within the potential internetworking design space. As with packet switched networks the design priorities and decisions have consequences for the control of congestion.

This section discusses the fundamental problems that need to be solved in order to interconnect networks. This is developed with an analysis of alternative approaches and a comparison of the relative merits of different designs.

The characteristics of a network connecting computers separated by distances in the order of meters will be different those connecting computers separated by hundreds of miles. Important differences include, the physical media, delay and reliability as wall as levels the of trust that can be assumed within the network. Different types of network are therefore necessary to interconnect computers separated by distances of varying orders of magnitude. The protocols in these networks will have to overcome different problems and a degree of heterogeneity will therefore be necessary to achieve optimum performance.

Internetworking allows the development of heterogeneous network technologies, whilst allowing computers to communicate across them, by interconnecting networks. To achieve internetwork communication data must be propagated between networks and translation must occur between peer protocols [VGK78]. These data propagation and protocol translation functions are fulfilled by a Gateway on the interface between the networks [CK74].

Where there is a semantic difference between protocols, the translation process is made more difficult if not impossible. For all protocols to be the same would however, inhibit the optimisation of networks to take advantage of the specific properties of the domain they operate in. A common internet architecture can enable such optimisations whilst maintaining compatibility between protocols [Mca89].

The central issues that are addressed by an internet architecture are:

[5] There may be a desire and mechanisms to restrict this access

1. protocol translation
2. naming, addressing and routing
3. congestion control

2.6.1 Inter-Connection Level

A common and effective technique in the design of internet architectures is to establish a layer of protocol which is common across the internet [Sun90]. Below the common layer protocols are generally heterogeneous and Gateways provide mappings from one networks protocol to the next. Above the common layers protocols may be homogeneous or else care must be taken to ensure an easy mapping.

In a layered protocol stack each layer adds functionality to the services provided by lower layers [Hub80]. Consequently the higher up the protocol stack a common layer is established the greater the amount of functionality that needs to be mapped from one network to the next. This leads to greater gateway complexity [Pos80]. Conversely the lower down the protocol stack that a common layer is created, the less flexibility there is to accommodate heterogeneous networks. Where in a protocols stack a common layer is situated has important implications for the development of other aspects of the internet architecture. Four possibilities will be considered:

1. The physical layer
2. The data link layer
3. A new internet layer
4. The transport layer

The Physical Layer

Bridges connecting together Ethernet segments are an example of the extension of networks with common physical media. Different media are appropriate in different domains, therefore the scope of an internet utilising a common physical layer is seriously limited.

Common Data-Link

Paul Baran advocated a common data block across different media.

> *As we move to the future, there appears to be an increasing need for a standardized message block for all-digital communications networks. ... The standardized message block simplifies construction of very high speed switches* [Bar64b]

The provision of a common data link layer allows for straightforward propagation of data between networks. This leads to the possibility of designing gateways that are simple, efficient and cheap. Not all internetworking functions can be effectively conducted in the data-link layer. The setting up of routes, accounting and congestion control are best performed in a higher inter-network layer. This leads to a natural division in the inter-network architecture between the data forwarding functions performed in the data-link layer and control functions carried out in an internet layer [Mca89].

The provision of a common data-link layer does not exclude the adaption of protocols to take advantage of the specific properties of the domain within which they are operating. Different media can be supported and control functions can be tailored to support a specific environment.

This has been demonstrated for Desk Area Networks (DANS) [HM91, M.93, BaDMP95], Local Area Networks (LANS) [ID91], Metropolitan Area Networks (MANS) [Gre89] and Wide Area Networks (WANS) [TL89].

Networks with different data-link layers can be connected by providing data-link protocol translation in the Gateway attaching the network to the internet. This has the cost of adding complexity at the specific network internet interface and the attached network may not be able to take advantage of all the properties of the internet. The data-link mapping however, only has to be undertaken at one network interface rather than at every gateway [Mca89].

ATM has the potential of providing a common data-link level interconnection [LTM83] between networks.

Internet Layer

Networks can be interconnected by creating a new internetwork layer directly above the intra network layer. The early experimental inter-network PUP [BSTM80] and the Internet [VGK78] are examples of internets linked by a new internet layer [Pos80].

Layers below the internet layer are mapped from one network to another by encapsulating an internet datagram in a network level packet. Gateways switch packets between networks by performing a number of operations. When a packet is received the gateway would unwrap it, decide which output to transmit it on and then wrap it in a packet appropriate for the next local network before forwarding. This process creates a significant overhead in processing for each gateway. In this architecture even packets that are not destined to cross network boundaries may still need to include an internet datagram. This means overheads are also paid by local traffic. With an internet layer the data forwarding and control functions can both be located in the same protocol layer resulting in a tight binding between them.

Transport Layer

Different types of traffic require different services from a transport layer. To accommodate different traffic types it is therefore necessary to provide different transport protocols. The DoD internet architecture started as a Transport level common layer and was later changed to providing a common internet layer. The provision of internetworking at the transport level is impracticable because of the complexity that is introduced.

To change the common network layer is difficult as it implies simultaneous change across multiple network types. The development of new traffic types are likely to require the development of new transport protocols, consequently transport layer internetworking does not provide the flexibility to accommodate new traffic types easily. Cole describes an attempt to map TCP to the ISO Class 4 transport protocol [Col90] which suffers from similar problems.

In summary the establishment of a common network layer at the physical, data-link, network and transport layers has been considered. Of these options interconnection at the physical and transport layers are not practicable for general internetworking solutions. The establishment of a common data-link layer has the benefit of enabling gateway simplicity and naturally supporting the separation of data forwarding and control functions. A common network layer leads to a closer binding between control and data forwarding, greater gateway complexity, but it naturally accommodates the interconnection of heterogeneous networks.

2.6.2 Routes, Addresses and Names

To allow a process to communicate with a peer on another network the internetwork architecture must support the translation of names into addresses, and be able to use an address to find a

route to the destination. If a host knows the name of the resource that it wants to use, then it can reach that resource if there is a way of mapping from the name to an address and if the address can be used to find a route to the resource.

> *A name tells what an object is; an address tells where it is and a route tells how to get there.* [Sho78, Coh78]

There are three common approaches to resolving the routing, addressing and naming bindings in a inter-networking environment; source routing, virtual circuit and datagrams switching. The three approaches will be discussed below. In source routing [Sal80] each packet contain an ordered list of the addresses for each gateway on the route to the destination. As the packet progresses through the network a gateway removes its address from from the front of the list, places it on the end and forwards the packet to the Gateway with the next address. When the packet arrives at the destination the list of Gateways is used to determine a reverse path. Gateways do not need to maintain routing tables and only need to know the addresses of neighbours. The larger the number of hops on a route the greater the space consumed by addressing information, consequently this approach does not scale well.

Virtual Circuit and Datagram Switching

When virtual circuit switching is used a route is set up at the start of a connection. All subsequent packets will travel down that path. It is therefore only necessary for the first packet to contain a global address all subsequent packets carry a virtual circuit identifier (VCI), which each gateway uses to decide where to forward the packet. With datagrams each packet contains a global address for its destination. The different bindings between services (names), routes and addresses for datagram and virtual circuit switching is described in reference [Les83].

For a datagram:

> *... a service is bound to an address by the source. An address is bound to a route by the gateways and the source, which must pick the first hop on the route. This binding is made every time a block passes through the network ...*

For a virtual circuit:

> *... the binding of a service to a route is made when a connection to that service is established*

Virtual circuit switching results in a larger connection set up overhead than datagram switches and there is more state in the switches that needs to be initialised, but this is compensated for by faster switching and a lower addressing overhead for subsequent packets [Mca89].

The location of error control has implications for the service resulting from datagram and virtual circuit switching. If a network guarantees the reliable sequential delivery of packets, one dropped packet may result in the delay of many as its retransmission is awaited before subsequent packets are forwarded. The jitter on a connection is therefore increased. Where an end-to-end virtual circuit is created by concatenating virtual circuits the problem is made worse as each gateway may introduce jitter. In addition the gateway itself is significantly complicated.

For applications that require a smooth delivery of data but are tolerant of the loss of small amounts of the provision of reliability by the network is unnecessary, and if it is require it con be supplied on an end-to-end[SRC84] basis. When virtual circuit switching is utilised and reliability is not provided by the network the properties of a circuit that are virtualised in the service provided are: a low addressing overhead, low jitter, fast switching and sequential delivery of data. These properties flow from the switching mode adopted. These have been called Light Weight Virtual Circuits [Les83].

There is a natural fit between virtual circuit switching and internetworking with a common data link layer, because the switching enables the required clean division between the between the control and data paths. The main control functions can be performed out of band when a connection is set up thereby simplifying the data path. In a datagram the data and control paths are bound together. This makes them inappropriate for internetworks which use the data-link layer as the common layer. They can be used in networks where the (inter) network layer forms the common layer. The independence of each datagram means that if a node on a route is lost a new route can be used without having to reset the connection. Datagrams are therefore more tolerant of network failures, however datagrams may follow different paths through the network increasing the likelihood of re-ordering and the amount of jitter.

2.6.3 Congestion Control

In discussing internet congestion control its fundamental cause and how this relates to the packet networking experience will be considered. Next the relationship between datagrams, virtual circuit, paths and connections switching and the exercise of congestion control at the session and intra connection time scales are discussed. Thirdly some of the ways in which Internet congestion posed new challenges are identified.

An Internet can be conceptualised as a packet switched network built out of gateways and other lower level packet switched networks. In this model internet gateways correspond to packet switches and lower level networks to links [HHS83]. Gateway congestion results from contention for resources in a similar way to congestion in a packet network. The dynamic allocation of buffers and bandwidth in the course of a connection allows high levels of resource utilisation but risks demand exceeding the supply for a resource. Consequently, even if there is effective internal congestion control in the packet networks that constitute an internet congestion can still occur at gateways.

Connections and Paths

Virtual circuit switching requires the establishment of a connection between hosts and a path between them. With datagram switching a known path is not created, but communication may be either connection oriented or connection-less. Whether the network is aware of connections and whether a fixed path is established between communicating hosts has similar implications for internet as it has for other packet switched networks.

Session Control A connection is needed for the implementation of congestion control on the session time scale. When a connection is being set up it can either be accepted or refused depending on a negotiation between the host and network. The network may be aware of current network usage and the host of the demands it is likely to make on the network. If the network is experiencing congestion, refusing calls will help bring it under control. The presence of a known route allows resource reservation and negotiation with, and if necessary call rejection by, intermediary nodes.

Intra Connection Adaptive Control Inherent in many approaches to congestion control is the network indicating to the source that it should slow the rate at which it is transmitting data. Within a connection the notion of speeding up and slowing down is natural because of the association between packets. In a connection-less network each packet is treated independently and there is no natural association between them, consequently slowing down does not make sense. To utilise adaptive congestion control some other association between packets must be created in a connection-less network.

New Challenges

Congestion control on internets is made more intractable as a result of several factors. Different networks may have different capacities as a result of the technology they employ. A packet radio network may have a larger capacity than a network based on land lines[6]. The mismatch in capacity between networks types can lead to congestion of the gateway.

A greater range of RTTs between pairs of hosts is likely to exist on an Internet than on a single network. This is caused by a combination of a larger range of distances between hosts and a wider range of bandwidth capacities. The variation in RTTs may also be greater between any one pair of hosts. Variation in RTT makes it harder to derive an accurate estimate of the RTT on a connection and consequently makes it harder to estimate when a packet should be retransmitted. An overly conservative retransmission strategy would unnecessarily reduce the connections throughput, an aggressive strategy will result in unnecessary retransmissions thereby increasing network load and contributing to congestion.

The inter-connection of networks aggravate problems of scale. The sources of congestion may be further removed from the point that congestion occurs and the number of hosts on the interconnected networks will be larger than on each individual network. A simulation study of congestion between and within interconnected networks was conducted at NPL. A two level hierarchical network structure of a high speed backbone network interconnecting local area networks was used to investigate the effectiveness of IBL and Isarithmic congestion control [Pri79]. The study found that the IBL scheme is most effective when congestion is caused by a host directly connected to a switch experiencing congestion. When networks are interconnected, traffic may concentrate at the interconnections between networks giving rise to congestion at a point removed from the source.

Isarithmic congestion control becomes ineffective and unmanageable as network scale increases. The management of permits becomes intractable in large networks and manifestations of local congestion more serious. A solution proposed in reference [Pri79] was the use of Isarithmic control in the backbone and IBL in the local networks. A degree of success was achieved suggesting that a hierarchical internet structure may simplify congestion control problems. The combination of IBL and Isarithmic control was however only demonstrated to be successful within the limited context of homogeneously distributed traffic and a small internet[7] with only two levels. Its general applicability was not demonstrated.

A large number of hosts has implications for the choke scheme developed in the CYCLADES network. In that scheme each switch maintained a list of all pairs of hosts connected to the network. Associated with each pair were variables that indicated the congestion state of the path between them, these variables were used by entry switches to choke off input from hosts that did not respond to congestion notifications. If there are a large number of hosts it becomes inpracticable for each switch or gateway to maintain a list of each pair.

An internet may span several administrative domains. This complicates the process of capacity planning and ineffective capacity planning can aggravate congestion.

In summary congestion control in an internet is fundamentally the same problem as on a packet switched network, because fundamentally an internet is a packet switched network. This was significant to the development of internetting because it meant that experience gained in controlling congestion like the ARPANET, X.25 and CYCLADES was of relevance. The greater variability of RTTS, capacities and levels of reliability are some of the characteristics of internets that posed new or aggravated problems and contribute to making internet congestion control a hard problem. The interconnection of packet networks extends the resource sharing across network boundaries. The technology employed is similar to that within computer networks in that packets are switched in gateways. This allows the dynamic utilisation of bandwidth and brings with it the same congestion

[6]circa 1980 ARPANET was connected with 50kbps terrestrial lines; the Packet Radio Network(PRNET with dual rate 400/10kbps radios and the Packet Satellite Net (SATNET - used a 64kbps shared channel on Intelstat IV [Kah94]

[7]A total of fifty hosts

problems associated with computer networks.

2.7 Summary

The nineteen seventies saw a massive growth in the number of packet switched networks. Inherent in packet switching is both the ability for bursty sources to efficiently share a path and the danger of congestion arrising as a result of contention during communication.

The way in which congestion manifests itself is shaped by the design of the networks. For networks such as the ARPANET, where the subnet guarantees message delivery, the avoidance of deadlock is a major priority. For networks, such as CYCLADES, where reliability is provided by hosts, the decline in throughput caused by the retransmission of dropped packets is the major manifestation of congestion. Just as the architecture of packet networks shapes the nature of congestion, so the network structure provides a context within which congestion is combatted.

Where each packet is treated independently there exists a tension between the network architecture and congestion control. Packet-wise congestion control is flawed. Isarithmic control does not prevent local congestion and requires global management of permits, which is impracticable in large networks. Input buffer limits, can lead to the starvation of some sources. Its effectives in limiting congestion, caused by the interaction of distant sources, is limited. Other congestion control schemes developed in datagram networks rely upon an association between packets. The CCL scheme, requires the sources on a congested link to slow down, and on the network being aware of the route that packets will take. Other schemes [Eli79] create temporary paths for packets to follow and depend upon nodes recording information about streams of packets.

To provide an effective congestion control scheme it is necessary for the network to be aware of associations between packets even if such an association is not provided explicitly by the network architecture. The provision of a virtual circuit service by the network does facilitate a natural association between packets. A virtual circuit service can be implemented in two ways; on top of a datagram subnet or by using virtual circuit switching. In either case control can be exercised at two levels. When a connection is requested by the network it can be accepted or rejected based upon an assessment of current network conditions. The flow of traffic on the virtual circuit can be controlled to provide a finer grain of reaction to congestion.

The concatenation of separate virtual circuits to provide an end-to-end circuit can result in ambiguity about the cause of delay and unnecessary retransmissions. Where the service is implemented using virtual circuit switching a fixed route is established. Whilst the route is being established resources at each node may be reserved, and each node may veto the connection if sufficient buffers or bandwidth are not available. Greater control and flexibility are bought at the expense of more complexity in each node.

The development of packet networks and congestion control, went hand in hand in the seventies. Whilst the problem of congestion was not solved a wealth of experience certainly existed. It is against this background and with this experience to draw on that the challenge of the interconnection of packet networks took place. In drawing his survey of flow control to a close, Kleinrock observes that internetwork flow control should:

> *... be designed to prevent the congestion of gateways along the path and should be supported by explicit gateway-gateway protocols for the exchange of information.*

The extent to which these aims were met is the subject of the next chapter.

Chapter 3

TCP Congestion Control

This chapter presents an account of the development of the TCP/IP protocols. It is argued that the post hoc addition of congestion control to an Internet Architecture designed around the datagram paradigm both shaped and limited the effectiveness of the congestion control mechanisms introduced.

Firstly, the design priorities which shaped TCP/IP's and therefore the Internet's architecture are discussed. These design priorities shaped the development of TCP/IP from the start of the research effort in 1973 to standardisation in 1981. The consequences for traffic management are explored with particular reference to the absence of congestion control from the initial Internet Architecture.

Early growth in the Internet resulted in the phenomena of congestion collapse which threatened it's basic functionality. This issue is discussed in the second section.

In section three the development of mechanisms to combat congestion collapse are discussed; from the use by Nagle of Source Quench packets, to the development by Jain of Congestion Avoidance and Recovery algorithms, to the host based procedures developed by Van Jacobson and introduced into TCP.

3.1 The Design of the Internet Protocols

ARPA funded the development of a range of packet network technologies in addition to the ARPANET; these included Packet Radio [KGBK78], Satellite Networks [JBH78] and later Local Area Networks (LANS) [CPR78].

> *Each of the different networks is best suited to different military requirements: local coaxial nets for intra-vehicle (ship, plane, etc.) digital communications, packet satellite for long haul and ship/shore (for instance), terrestrial packet switching for communications within CONUS and packet radio for ground and air mobile computer communications. ...* [Cer80a]

The Department of Defence (DoD) wanted to be able to interconnect these networks to build an integrated control and command structured. The resulting top level design aim for the TCP/IP protocol was:

> *... an effective technique for multiplexed utilization of existing interconnected networks* [Cla88]

It would have been possible for ARPA to adopt a common data-link layer across these networks, but this was not pursued. First the individual networks were designed and then the question of how to interconnect them was addressed. An early requirement placed on the internetworking design was that constituent networks would require no internal changes [LCC+97] before being connected.

The combined goals of supporting heterogeneous protocols, without modifying the individual networks lead to the decision to establish a common internet layer above the existing network layers. This architecture can be layered across many different networks.

An early suggestion was to use the ARPANET transport protocol (NCP) as the internet common layer, however NCP assumed that it received an error free sequence of data. This made it inappropriate for networks that did introduce errors (such as packet satellite networks) and so in 1973 a research effort to produce an internet transport protocol, called TCP, was started by Cerf and Kahn [CK74].

3.1.1 The Hierarchy of Design Aims

In this section the design aims for the replacement of NCP are discussed. A hierarchy of design aims, below the general aim of interconnection of networks, is defined by Clark [Cla88] this hierarchy is itemised below and then discussed with reference to the motivation for the priorities selected and the resulting design choices that were made.

The hierarchy of design aims was defined as:

1. Internet communication must continue despite loss of networks or gateways
2. the Internet must support multiple types of communication service
3. the Internet architecture must accommodate a variety of networks
4. support for distributed resource management
5. the design should be cost effectiveness
6. easy host attachment should be facilitated
7. resource accountability

This hierarchy is shaped by military requirements. To illustrate, under normal conditions the loss of a network or node would be an exceptional event, yet efficient recovery from node failure is made the most important design aim. This is understandable when it is recognised that loss of gateways is a likely event in a battlefield situation.

> *... For military networks, robustness and rapid adjustment/recovery from failures are essential to meet the requirements of operation under hostile conditions. The datagram orientation of IP promotes these goals ...* [Cer80a]

The control of congestion is omitted from, and accounting for resource usage is explicitly placed at the bottom of, the list of design aims. Baran's arguments for the importance of congestion control in a conflict situation are ignored. The reason for de-prioritising resource monitoring is given in terms of a military analogy [Cla88]. It is argued that in war the mobilisation and deployment of resources is more important than the accounting of such resources. The monitoring of traffic is however useful not only for the generation of billing, which may not be important in a military context but also for the regulation of the flow of, materials in war, and data in networks. The absence of this from the overall design of TCP/IP means that the information was not available to enable effective regulation of traffic flows.

Reliability

The main threat to the reliable delivery of data was identified as the failure of intermediary nodes or networks. TCP was therefore designed to cope with node failure. The aim was to ensure, as far as possible, that if two entities were communicating over the internet they should be able to continue communicating, without resetting state or reconnecting, unless one of the entities was destroyed.

This leads to a design where state and trust are as far as possible concentrated in the end nodes. In this way fate sharing is accomplished; if an end entity is destroyed so is the state that enables communication, otherwise the state is maintained and communication can continue. In this model there is only one type of failure and that is complete. All other failures can be recovered by the end entities. The fate sharing model is consistent with a traffic control approach, which allows congestion to occur but places enough state in hosts to allow them to recover from resulting packet loss.

In the first paper on TCP [CK74] the idea of a connection having a fixed path through the network was rejected in favour of the establishment of an association between two entities which made no assumption about the path. During an association each packet would be independently routed through the network. The dynamic routing of packets allows an alternative path to be used in the event of a Gateway failure, therefore communication can continue without the need to reset end state or reinitiate a connection, but resource reservation is prevented.

A three way handshake was used to set up an association and the exchange of release packets and acknowledgments to close a connection. TCP and therefore at this point the internet layer provided a connection oriented datagram service.

Multiple Services

The desire to support multiple traffic types, in particular data and real-time voice, was the prime motivation for the division of TCP into separate internet and transport layers. Work by Danny Cohen on packet voice showed that real time communication required different flow control mechanisms to traditional data transfers[Coh80]. He suggested to Cerf[Cer93] that the reliable sequenced delivery of data provided by NCP and now TCP is unsuitable for real time voice communications. The retransmissions required for reliability in the face of loss can introduce high levels of delay that inhibit real time communication.

To accommodate these different reliability requirements in early 1979 [Cer80b] TCP was split into two layers. A lower IP layer, was created which handles the basic internetting[1] functions (of addressing routing and fragmentation) and a higher TCP layer which handles the end-to-end functions (sequencing, retransmission, flow control and port selection).

At the same time the ability for Applications to build their own transport services by accessing the basic IP datagram service through a new protocol UDP was introduced. If new traffic types required new transport services parallel protocols to TCP could also be developed. An important side effect of this change was the removal from the internet layer of knowledge about connections. For IP there would be no association between separate packets even if they formed a stream in a higher protocol. The implications for congestion control of this change appear not to have been considered.

The need to support different applications, which may require either the reliable or unreliable transportation of data resulted in the splitting of TCP into the separate IP and TCP protocols and in the internet layer providing a connection-less datagram service.

[1] This layer could be considered a new Internet layer [Les83] or as a sub-layer of the network layer. The later definition was adopted by the ISO in 1978 [Sun90] in IP networks it forms the network layer.

Interconnection

The desire to be able to interconnect heterogeneous networks lead to a minimum set of explicit assumptions about the underlying network. It is assumed that the network will be able to; transport a packet and support addressing.

This allowed IP to be deployed on a large number of networks. It also means that features that the underlying network may support like: knowledge of speeds or delays, are not available to higher level protocols. The insulation acts to protect transport protocols from the evils of unreliable networks and the advantages of more intelligent network designs. IP is therefore an exception to the general protocol design approach of each layer building upon services provided by lower layers, IP may discard some of those services.

If knowledge about reliability, sequencing, priority ranking. delays, speeds, losses etc are required by protocols above IP they must be re-engineered at the transport or application level even if the network below IP has the relevant information. This design acts to ease the deployment of IP across rudimentary network technologies, but it may also acts to discourage the introduction of intelligence in networks and encourages the development of sophisticated (complicated) transport protocols.

3.1.2 The TCP/IP Standards

In this section the main features of, and interrelationships between the protocols that make up TCP/IP are presented. The internetworking research effort lead to the standardisation of TCP/IP in 1981.

In September 1981 RFCs for the main components of the TCP/IP protocol suite were published.

1. Assigned Network Numbers (RFC790) [Pos81a].
2. The Internet Protocol (RFC791) [Pos81b].
3. The Internet Control Message Protocol (RFC792) [Pos81c].
4. The Transmission Control Protocol (RFC793) [Pos81d].

These were followed in November 1981 by NCP/TCP Transition Plan (RFC801) [Pos81e], which proposed a structure 18 month plan to replace the old NCP with TCP on the 200 hosts which then made up the ARPANET network. The transition on the ARPANET was completed in January 1983 when NCP was switched off.

IP

IP is the main network layer protocol. It provides two services. First it provides every endpoint with a network address. Secondly packets which are injected into the network are forwarded to the destination address which is in the packets header. IP hides the details of lower layers from endpoints and provides the abstraction of an unreliable, best effort, end to end link. IP does not guarantee that a packet will reach its destination. It offers no quality of service guarantees, but it will attempt to carry a packet from its source to destination. In doing so it provides facilities for fragmenting packets if they are to large to be carried over intermediary networks.

ICMP

The Internet Control Message Protocol carries error messages from the network to the IP source. It is layered directly above IP and is carried in the data portions of an IP packet. One of the ICMP messages is called the Source Quench message which is intended for use in congestion control.

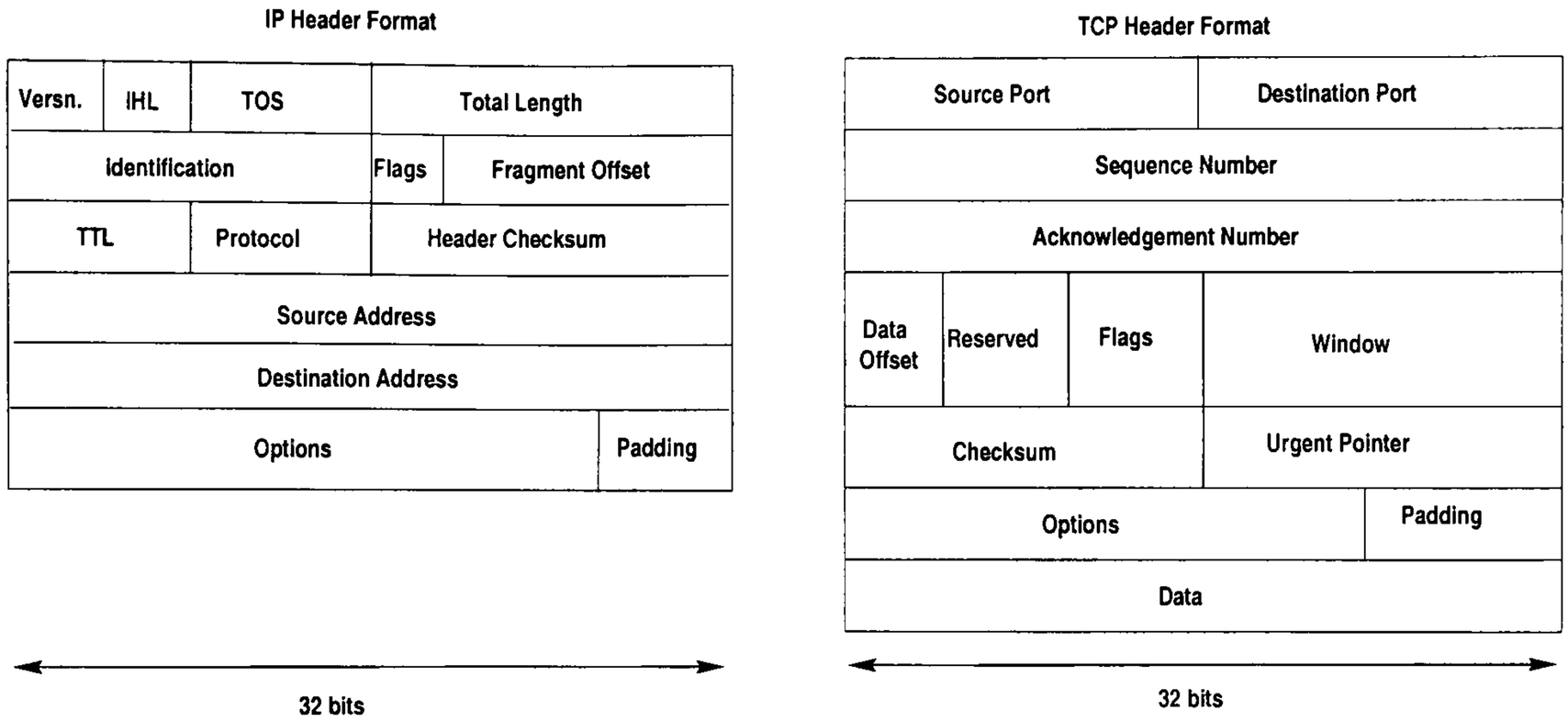


Figure 3.1: **IP and TCP headers: These diagrams show the different fields in the IP and TCP headers. each row represents 32 bits.**

Transmission Control Protocol

The Transmission Control Protocol provides a multiplexed, duplex, connection-oriented, reliable, flow-controlled, byte-stream service, which supports the reliable start up and graceful close down of connections, to applications. It builds this upon the services provided by IP.

A sliding window flow control scheme is utilised to enable multiple packets to be outstanding in the network and to limit the number of such packets. The maximum number of packets allowed to be outstanding is defined by the receiver and advertised to the sender in the TCP header.

Recovery from packet loss, possibly caused by congestion, is provided by the sending host retransmitting a packet if an acknowledgment has not been received before a timeout occurs. The receiving TCP is responsible for resequencing data so that it is passed to the application in the correct order.

UDP

UDP provides an application level interface to the services provided by IP and provides port numbers for demultiplexing.

3.1.3 TCP/IP Traffic Management

In this subsection an assessment is made of the congestion control in the original TCP/IP standards. The traffic control mechanisms themselves which were touched upon in the previous section are examined in more depth. The three main features of TCP/IP's traffic management are discussed below; host to host flow control, recovery from packet loss and ICMP Source Quench messages.

Flow Control

The IP layer provides no flow control facilities, nor does it make any explicit assumptions about the flow control conducted at lower layers. Flow control within is provided within TCP connections. This enables a host to control the quantity of data transmitted to it and to ensure that it has

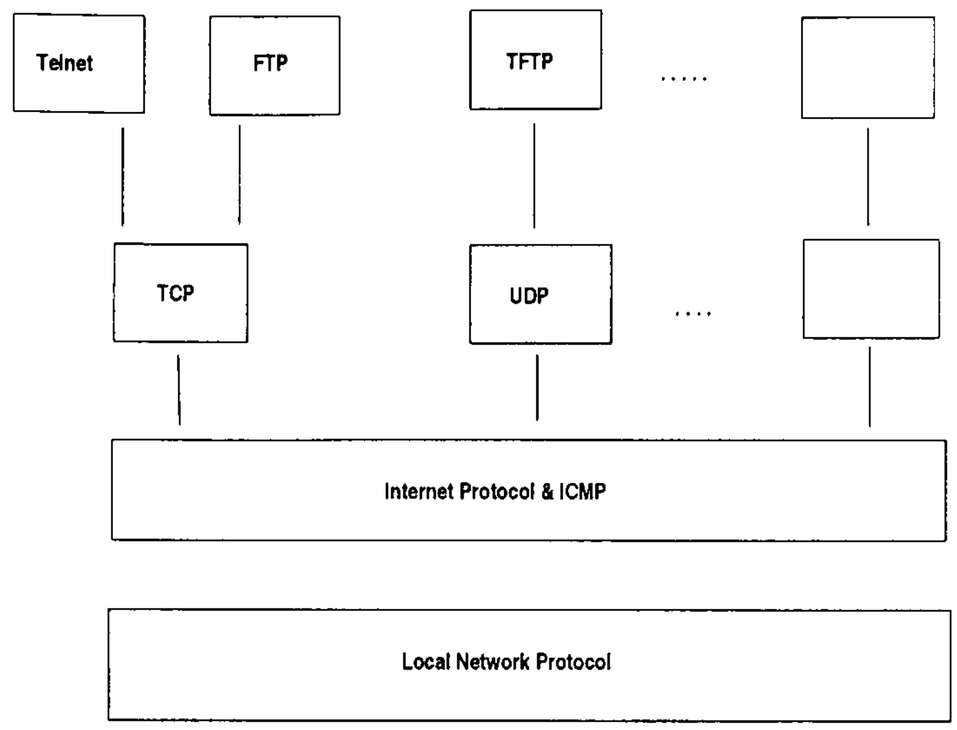


Figure 3.2: **TCP/IP Protocol Relations: This diagram illustrates the relations between protocols in the TCP/IP protocol suite. There are multiple application level protocols, a small number of Transport protocols, one Internet protocol and multiple local network protocols.**

sufficient buffers allocated to receive the data in. A sliding window scheme based on the flow control in the CYCLADES network is used.

When a connection is established a starting sequence number is chosen for each direction of flow. Data is received from an application by TCP on the sending host and is passed to an application on the receiving host in the same order as the originating TCP received it. Each data byte has a monotonically incremented sequence number associated with it.

When a packet is sent the TCP header has a 32 bit sequence number, which corresponds to the first byte in the packet. This number allows the receiving TCP to correct any reordering that has occured before passing data to the application. The sequence numbers are also used to control the flow of data. Each packet header also contains a 32 bit acknowledgment number, which indicates the next byte that is expected to be received. If several packets have been received and a packet has been dropped, there may be a gap in the sequence space. In this case the first byte in the gap will be acknowledged.

Each packet also contains a 16 bit window size. The size of window advertised is likely to correspond to the unused space in a connection's receive buffer. The sum of the acknowledgment and window size indicates to the sender the highest sequence number that the receiver is currently prepared to accept and therefore the data that it is legitimate to send. As a connection progresses the window of sequence numbers that can be sent slides across the sequence space until all data has been successfully transferred. If an acknowledgment is duplicated extra credit will not be allocated to the sender and if an acknowledgment is lost it will merely delay a window update which will be corrected upon receipt of the next acknowledgment.

A further characteristic of this scheme is that as load on the internet increases, so too will the number of packets held in queues and the delay a packet experiences will increase. This results in a corresponding delay in the receipt of an acknowledgment and a slowing of the rate at which packets are transmitted, however the number of packets outstanding in the network will not be reduced. A sender can only transmit one windows worth of data within one round trip time. This results in important side effect in that the maximum burst of a connection is limited to one window of data.

Whilst TCP's sliding window flow control provides a mechanism for the matching of data flow to the receivers buffers it does not provide protection from congestion of intermediary gateways. A host is not prevented from flooding the network, it may open multiple connections or may bypass

TCP and use UDP to transmit data. Multiple hosts may also create congestion by presenting a combined load which is in excess of the networks capacity.

Recovery From Packet Loss

An IP router will not accept a packet if its buffers are full, and when a packet is transmitted no copy is kept or inter gateway acknowledgment system provided. This prevents deadlock situations from arising. In order to provide a reliable data stream TCP needs to both detect and retransmit packets that have been dropped. When a packet is sent a retransmit time out (RTO) is set and if it has been not acknowledged when the timer goes off it will be retransmitted.

> *Because of the variability on the networks that compose an internetwork system ... the retransmission timeout must be determined dynamically* [Pos81b]

The RTT between two hosts can also vary due to a number of factors; a change of route, a change in network load, variable packet size, delay in the sending host or delayed acknowledgment from the receiving host.

If the RTO had an optimum value it would cause a retransmission to occur after a minimum delay and would cause no unnecessary retransmissions. If the RTO value was to high, a larger than necessary delay would occur and throughput on the connection would be reduced. If the RTO was too small packets which had not been dropped would be retransmitted and network resources wasted, reducing the achievable aggregate network throughput. The approach taken was to measure the time between a packet being sent and the arrival of its acknowledgment and to calculate a smoothed round trip time SRTT using equation 3.1 where ALPHA is 0.8 or 0.9.

$$SRTT = (ALPA * SRTT) + ((1 - ALPHA) * RTT) \tag{3.1}$$

The SRTT is then used to calculate the RTO, which must be between an upper bound (UBOUND) and a lower bound (LBOUND). BETA is a fixed delay variance factor.

$$RTO = min(UBOUND, max(LBOUND, (BETA * SRTT))) \tag{3.2}$$

If these algorithms are successful unnecessary retransmissions and delays will be avoided, but they will not prevent packet loss due to overload. A retransmit mechanism was found to be inadequate for controlling congestion in the CYCLADES network. When load exceeded the capacity of the network traffic became dominated by packets that were later dropped, resulting in a collapse in throughput. The same phenomena was observed whilst TCP/IP was being tested over a 9.6 kb/s London to U.S. satellite circuit, only low throughput was achieved and the following was observed:

> *We found a significant improvement in performance to be possible upon using ARPANET* type 3 *packets which are not limited to 8 outstanding messages as is normal ARPANET traffic. However this mode of operation if stressed, caused serious network congestion problems.* [Cer80b]

ARPANET type 3 packets are a datagram service which bypasses the control on the number of outstanding messages imposed by IMPs. Using type 3 packets is therefore akin to testing TCP over a network which does not contain its own congestion control mechanisms. It can be concluded [PZ78] that detection and retransmission of dropped packets does not constitute an effective congestion control measure.

Source Quench

TCP/IP contains a third traffic control mechanism called Source Quench. The basic approach of Source Quench is similar to the Channel Load Limits scheme used in the CYCLADES network. A gateway monitors some resource usage and when it detects congestion communicates this to the host generating the traffic which should respond by slowing down.

> *The source Quench message is a request to the host to cut back the rate at which it is sending traffic* ... [Pos81b]

In section 2.5.2 the functional decomposition of the CLL scheme was discussed. The four functions identified were; monitoring of network state, communication of network state, user reaction and network enforcement. A comparison of how Source Quench and CLL deal with each functions will highlight important differences between the two schemes.

The CLL scheme tries to detect the onset of congestion in order to keep the network operating at its optimum throughput level. It does this by sending a slow down request when line usage reaches 70%. Source Quench monitors buffer usage and a message is only generated when congestion occurs. Both schemes are similar in that congestion is notified to source hosts by choke packets. The ICMP standard specifies that a host MAY generate a Source Quench message when a packet is discarded, this is therefore an optional feature. In the CLL scheme source switches are notified through routing updates, Source Quench does not first hop routers.

The required response of a host in both systems is similar. A host should slow down its rate of transmission until it stops receiving Source Quench messages whereupon it should slowly increase its rate of send. The CYCLADES network protects itself from hosts which do not respond by the source switch dropping all packets if congestion persists. There is no such enforcement mechanism in Source Quench. Thus although the two schemes are similar the CLL scheme is superior in that it detects the onset of congestion and provides a mechanism for the network to enforce control.

There are difficulties in the implementation of Source Quench. The lack of a connection and minimal service provided by the IP datagram service means that the current rate at which traffic is being produced is not known and consequently it is not clear how it would be possible for a host to slow down. Source Quench is not mentioned in the IP or UDP standards.

The connection orientation of TCP is more amenable to controlling the rate at which traffic is produced, however the Source Quench message is not mentioned in the TCP standard either and so the exact response to a message is not specified. Early implementations of TCP did not respond to Source Quench messages [SE80, Hin81, Gur81] and Rosen observes that *generally they are just ignored* [cR81]. The BSD UNIX reference implementation of TCP did not respond to Source Quench messages before 1984 [KM84, KM86].

A similar picture emerges with the generation of Source Quench messages. Nagle [Nag84a] observes that many gateways could not be relied upon to produce a Source Quench packet when a packet is dropped, some did so for only one in twenty dropped packets. Whilst a Gateway MAY[2] produce a source Quench message, the lack of specification in the Standards and its absence from TCP implementations means that it could not be considered and in fact was not an active congestion control mechanism.

3.1.4 Conclusion

The decision to build heterogeneous packet networks and interconnect them combined with design aims influenced by military priorities lead to a number of decisions being made about the architecture for internetworking developed under the supervision of ARPA. The main elements of this

[2]MAY is the key word used to define the level of requirement for a router to support Source Quench, in RFCs it is used to mean that *an item is truly optional* [Bra97]

architecture are the introduction of a new internetworking layer which provides a common minimal datagram service across multiple heterogeneous packet networks. Gateways between networks provide termination of lower level protocols and the propagation of data.

As far as is practicable state and trust are concentrated in the end nodes. The internet layer provides the minimum functions of addressing and routing. It is assumed that the underlying networks provide routing and addressing facilities but no explicit assumptions about reliability or congestion control are made. The lack of a connection in IP came about as a side effect of the splitting of TCP into two layers to accommodate the need for different transport services for real time and traditional data traffic.

The Internet Protocol has achieved its main design aim of connecting heterogeneous networks. It can be layered over a wide variety of packet switched networks. IP is the defining protocol of the Internet which interconnects millions of hosts and serves millions of users. The main threat to communication on the internet was identified as node failure. Congestion was not seen as a major problem, consequently TCP/IP contains no effective congestion control mechanisms and congestion was not a major consideration in the design of the architecture. A number of key decisions which limit the options available for congestion control have been identified.

1. The adoption of datagram switching means that no fixed path exists for a connection, consequently resource negotiation and reservation are obstructed.

2. The Internet Protocol is connection-less, consequently congestion control implemented at the session time scale is obstructed.

3. The internet layer IP provides a minimum service confined to addressing, routing and fragmentation IP and the internetwork do not control congestion. The feedback mechanism which is provided is optional and enforcement mechanisms are absent.

4. The lack of an association between packets, in IP and UDP leads to traffic control mechanisms being concentrated in TCP. These are confined to flow control and error recovery.

In [VGK78] an unresolved question is posed

> *What are the relative effects of just discarding packets in gateways, and relying on the end-to-end protocol to detect and compensate for this?*

The next decade would answer this question and demonstrate how significant a problem congestion would be for the Internet. It would also show how TCP/IP's architecture would shape the introduction and development of congestion control.

3.2 Problems from Growth

Van Jacobson's classic paper on Congestion Control and Avoidance opens with a description of congestion on the Internet.

> *In October of '86 the Internet had the first of a series of* congestion collapses. *During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and three IMP hops) dropped from 32 kbps to 40 bps* [Jac88].

Practice had shown that allowing gateways to drop packets and relying on hosts to retransmit them was not adequate to prevent congestion in internets.

This section discusses the growth of the Internet in the eighties and the problem of congestion which accompanied it. This is followed by a discussion of developments in traffic control techniques.

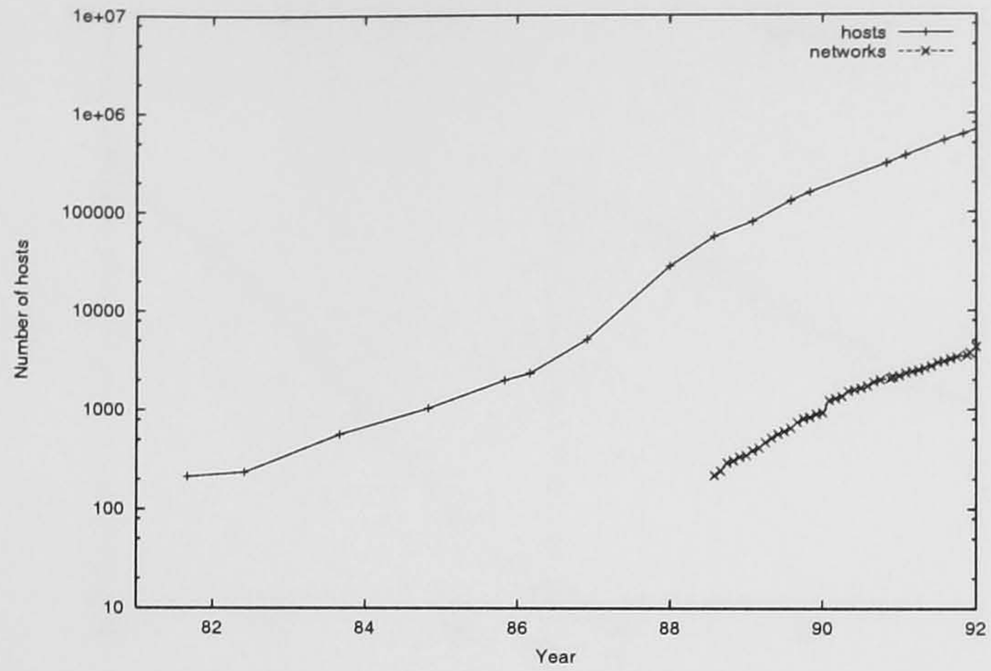


Figure 3.3: **Growth in Hosts and Networks: the 1980s saw exponential growth in the number of Internet hosts and networks.**

Internet traffic mainly fell into two categories, interactive textual traffic generated by remote terminals and bulk transfer both of these were carried by TCP, but had different requirements from the network.

Remote terminal traffic involves the transport of small quantities of data, but is highly interactive and so requires a quick response time. The increase in network RTT caused by congestion is particularly serious for this sort of traffic. Remote terminal traffic could aggravate congestion by inefficiently utilising bandwidth, Section 3.2.2 on small packet discusses this problem.

Bulk transfer traffic is less interactive consequently delay is not as important, however it had to be reliably[3] delivered and could monopolize bandwidth, thereby delaying interactive traffic. Reliability was achieved by retransmitting dropped packets, but variations in RTT meant that detecting when a packet should be transmitted was not easy. This problem is discussed in Section 3.2.3 on retransmissions and timers.

Even when small packets and retransmissions are handled well the load presented to the network may exceed capacity and congestion may result. The next two sections discuss ways of adapting the load presented to the network, by a connection, to the available bandwidth. The ARPANET had a number of characteristics which compensated for the lack of congestion control in TCP/IP. It was under the control of a single administration, which allowed for effective capacity planning so that bandwidths between links were matched and there were no obvious long term bottlenecks in the system. It also contained its own congestion control mechanisms and had excess bandwidth capacity.

The success of TCP/IP in allowing the interconnection of diverse networks enabled a tremendous growth of the Internet. When the old NCP protocol was switched off in 1983 there were approximately 235 hosts attached to ARPANET, by the end of the decade the Internet connected 300,000. The original TCP/IP tests took place over three networks by the end of 1990 the Internet encompassed over 2000.

All of the ARPANET's attributes, which helped control congestion, were lost to the Internet in this period of growth, leaving TCP/IP's traffic control mechanisms exposed in a hostile environment.

3.2.1 Congestion Collapse

The growth in the number of hosts attached to the network was in part caused by a change in the computing facilities at each site. The old model of a small number of large time shared main

[3] As did remote terminal traffic

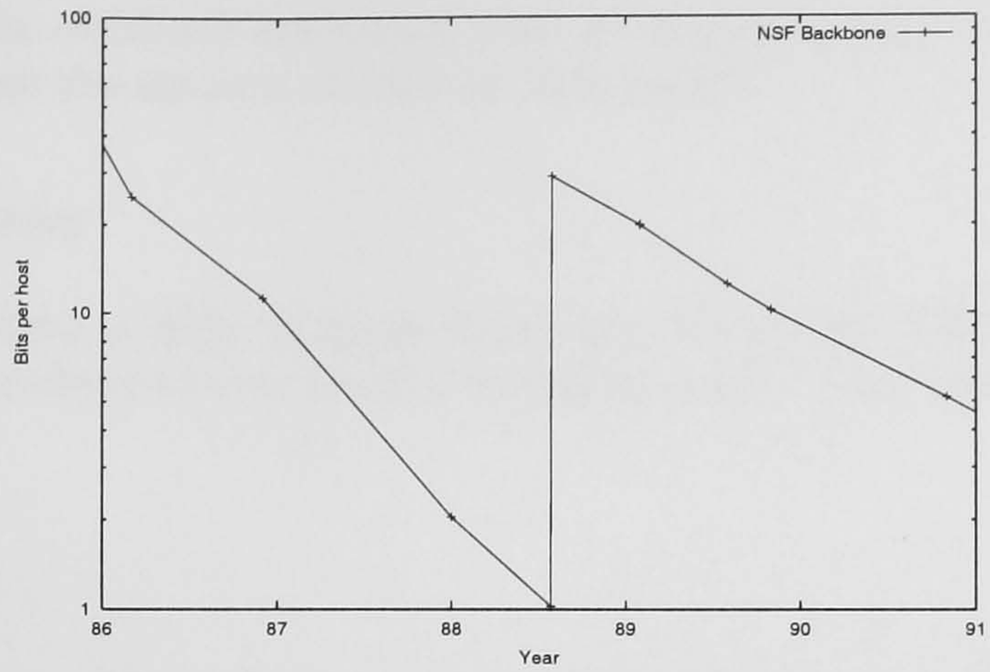


Figure 3.4: Gross Bits per Host: Backbone bandwidth did not keep up with the growth in hosts.

frame computers was replaced by distributed computing environments where a large number of desktop workstations are interconnected by high speed local area networks.

Typically LANs would have capacities of 10 mbps and would be connected to WANs with much smaller bandwidth. When the NFS backbone was first introduced in 1986 it had a capacity of 56 kbps, which was soon found to be inadequate.

Jain reports an experiment which demonstrates that the introduction of a high bandwidth link can greatly degrade performance in an uncontrolled network.

> *Four nodes serially interconnected by three 19.2 kbs lines were used in the experiment ... All the intermediary nodes were configured with very few buffers. The time to transfer a particular file was measured as five minutes. After the line between the first two nodes was replaced by a fast mbs line, the transfer time increased to seven hours!* [Jai86]

An example of congestion collapse in a real network is described by John Nagle [Nag84b].

Ford ran the first wide area private IP network. It had no internal flow control mechanisms and so depended on TCP/IP for traffic control. Leased lines of 9.6kbs/sec and a UK-US satellite link connected four sites some of which had 10mbs LANs, so there was a wide variation in link bandwidths. The cost of wide area connections meant that excess WAN capacity was not feasible. This network was cross tied to the ARPANET network.

Under heavy load a drop in network throughput to a fraction of its normal level was observed. Traffic was dominated by retransmissions, with each packet having to be retransmitted many times before it was delivered. Once entered congestion collapse was observed to be a stable state for the network.

As the Internet continued to grow the Ford experience would become more common and would act as a stimulus to establish effectual traffic control mechanisms.

3.2.2 Small Packets

The large bandwidth and processing overhead for datagrams, means that if each packet carries only a small amount of data waste will occur and congestion may be needlessly aggravated.

It had long been the case that much ARPANET traffic was made up of tinygrams [KN74]. When congestion became a serious problem, it became necessary to utilise resources more efficiently and

find ways of reducing the overhead associated with transmitting each byte of data. One way of doing this was to increase the amount of data in each packet.

Silly Window Syndrome

One cause of small packets is Silly Window Syndrome where the TCP window space becomes fragmented and large numbers of very small datagrams result. This was identified and solved in reference [Cla82].

Terminal Interaction

Telnet sessions could give rise to large numbers of small datagrams each being generated by one keystroke and each with only one data byte but all with at least 40 bytes of header. This represented a 4000% overhead.

Nagle identified this as a contributory cause of congestion in reference [Nag84a] and proposed an algorithm which addressed it. Each TCP connection is only allowed one outstanding packet that is smaller than the MSS of the connection. For connections on a LAN the ACK is likely to return before the next keystroke generates a packet. This preserves a quick response, but wastes bandwidth. On a LAN bandwidth is probably not a scarce resource and so the overhead does not matter. For WAN connections the ACK will take longer to return, causing data from many key strokes to coalesce and be sent in one packet. The longer the delay the more data is stored and therefore the smaller the per packet overhead. The algorithm therefore gives the required performance on both LANs and WANs.

The Nagle's algorithm was introduced to the 4.3 BSD implementation of TCP [KM84].

3.2.3 Retransmissions and Timers

As discussed in section 3.1.3 IP and UDP contained no mechanism for the retransmission of lost packets. TCP measured the RTT between a packets transmission and its acknowledgment and estimated when the absence of an ACK would indicate a gateway had dropped the packet.

Measurement and observation of Internet traffic [Mil83, Zha86, Nag84b] showed that significant numbers of packets were being retransmitted unnecessarily, and that this was contributing to congestion. Consequently the mechanism for packet loss detection needed revision.

Intrinsic Limitations of Timers

The TCP RTO timer aims to infer the loss of a packet, by the absence of an acknowledgment over a period of time. As such it is a specific instance of a more general phenomena in a distributed systems; using the passage of time to detect a failure.

In a distributed system the opportunity for communicating elements to directly observe changes in state and failures does not exist. There are two ways of detecting failure. An external report may communicate the error, or it may be detected locally. Time is the only tool that can be used to estimate remote failures locally. If some expected event does not occur within a time limit a failure can be assumed.

Lixia Zhang argues that:

> *Timers have intrinsic limitation in offering optimal performance. Any timeout based action is a guess based upon incomplete information and as such is bound to be non optimal. We can conclude that, if we aim at high performance, we should use external*

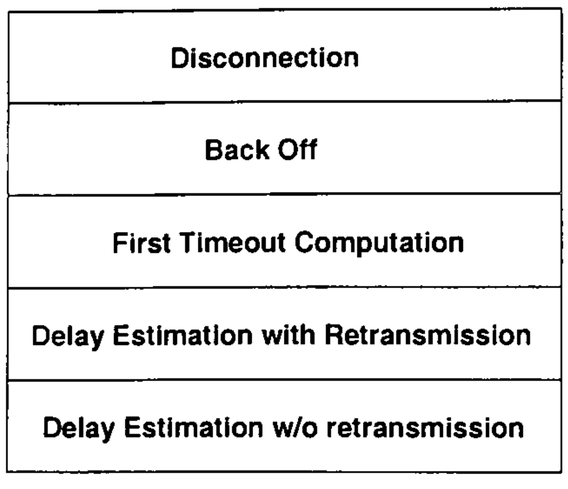


Figure 3.5: **Timing Hierarchy: Jain proposes a timing hierarchy as a layered hierarchy of functions, each of which is independent of higher layers and depends upon the services of lower layers.**

> *events as a first line of defence against failures, and depend on timers only in cases where external notification has failed* [Zha86].

In TCP an acknowledgment may be delayed or not arrive at all for a number of reasons:

1. A change in traffic conditions may have increased the RTT
2. The acknowledgment may have been dropped
3. The receiver may have delayed returning an acknowledgment
4. The network may have partitioned
5. The host may have crashed
6. The original packet may have been dropped by a router
7. A transmission error may have occured
8. The original packet may have been delayed in a lower level host protocol
9. The route to the destination or returning from the destination may have changed

In only case six and seven it is appropriate for the packet to be retransmitted, yet the use of a timer does not distinguish between the nine cases.

For optimal performance the TCP retransmit timer would go off as soon as a packet is dropped and only when a packet is dropped[4]. To optimise for fast retransmissions increases the likelihood of false alarms, which in turn will waste network resources and may contribute to congestion. To minimise false alarms will increase the delay before a retransmission occurs and reduce the throughput achieved on a connection.

The use of external events to deduce packet loss may allow retransmissions to occur quicker and avoid false retransmissions, however this method may fail even if an explicit report is used. Consequently local timer based detection of dropped packets would still be necessary. The following section outlines developments in using timeouts to accurately detect packet loss.

Hierarchy of Services

Jain [Jai86] analyses timing mechanisms as a layered hierarchy of functions, each of which is independent of higher layers and depends upon the services of lower layers.

[4]Unfortunately optimal performance is not achievable, because there is an unavoidable latency between a packet being dropped and the host becoming aware of it even with explicit notification.

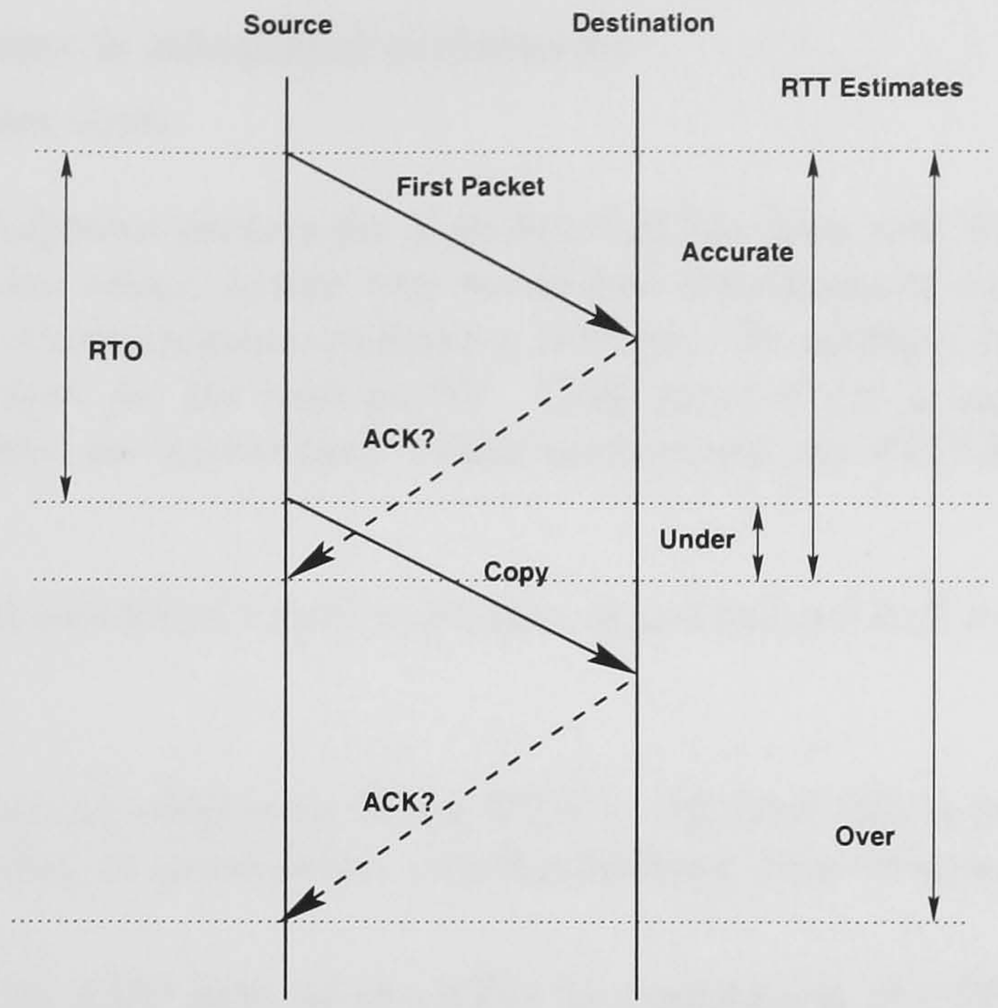


Figure 3.6: **Retransmission Ambiguity: When an Ack is received for a retransmitted packet it is not known whether the Ack was generated by the original or repeat packet.**

RTT estimation without loss

An accurate estimation of the RTT is important because it is the foundation upon which the retransmit algorithms are built. The exponentially weighted averaging adopted in the 1981 standards remains as the way of estimating RTT in the absence of loss.

An important question is what value should be adopted at the start of a connection. At this point no measurements have been made. If a low value is chosen false retransmissions are likely. If the value is high it may take along time to converge to the actual RTT and to detect an error, but the network is protected from retransmissions. The second option is generally adopted by TCP implementations.

RTT estimation with loss - Karn's Algorithm For acknowledgments of retransmissions it is ambiguous which copy of the packet generated the acknowledgment. Consequently it is ambiguous where the RTT should be measured from.

If the RTT is taken from the original packet but the ACK was generated by the copy an overestimate will probably result see figure 3.6. There is positive feedback, in that the overestimate of the RTT will drive up the RTO, which will in turn lead to a further increase in ambiguous RTT measurements. For small levels of loss this will be corrected by the unambiguous readings, if loss is persistent the SRTT diverges upward from the actual RTT [Jai86]. Slow recovery of lost packets and an overly sharp fall in throughput results.

When the RTT is estimated, from the transmission time of the latest packet sent to the receipt of its acknowledgment, the new RTT measurement will be an underestimate if the ACK was generated by the original packet; as shown in figure 3.6. If this underestimate is used to update the SRTT a further reduction in the SRTT and RTO will occur resulting in an increase in the probability of further false retransmissions.

A circle of reducing SRTT and false retransmissions is entered which results in unnecessary network load for all and a reduction in throughput for the specific conversation.

Either way the consequence is suboptimal performance.

Karn's algorithm proposes that:

> *When an acknowledgment arrives for a packet that has been sent more than once (i.e. retransmitted at least once), ignore any round-trip measurement based on this packet, thus avoiding the retransmission ambiguity problem. In addition the backed-off RTO for this packet is kept for the next packet. Only when it (or a succeeding packet) is acknowledged without an intervening retransmission will the RTO be recalculated from SRTT* [KP87].

Thus the integrity of the smoothed round trip timer is maintained and a divergence towards high and low values avoided.

RTO Estimation Once an estimation of the RTT is obtained this is used to calculate an RTO value. Significant numbers of unnecessary retransmissions were observed using this method of setting the RTO.

The original algorithm for TCP derived the RTO by multiplying the SRTT by a fixed factor of 2. This did not allow the RTO to always keep pace with increases in the RTT. Under high load the variation in RTT increases above 2, so the RTO was likely to be an underestimate, when the network was congested. The consequence of to low a value for the RTO is false retransmissions, so when the network was highly loaded, the RTO would often be to low and false retransmissions would be common.

Mills proposed using two values for the multiplier, a larger value would be used if the SRTT was going up and a lower one if it is going down.

Edge [Edg83] proposes dynamically measuring the variation in RTT and using this to determine a multiplication factor for the RTO. When the RTT is relatively stable an RTO which tightly fits will be achieved. When the RTT is increasing rapidly a loose overestimate will result.

Van Jacobson [Jac88] developed algorithms which allowed efficient estimation of the variation in RTT. The mean deviation which is a close conservative estimation of the standard deviation but does not require multiply and divide operations to calculate is used.

$$Err = RTT - SRTT \tag{3.3}$$

The algorithm for estimating the SRTT is unchanged:

$$SRTT = gain * SRTT + (1 - gain) * RTT \tag{3.4}$$

The same equation rearranged.

$$SRTT = SRTT + gain * (RTT - SRTT) \tag{3.5}$$

The mean deviation in RTT is also estimated:

$$Dev = gain * Dev + gain(|(RTT - SRTT)| - Dev) \tag{3.6}$$

or equivalently:

$$Dev = Dev + gain(|Err| - Dev) \tag{3.7}$$

The SRTT and the mean deviation are used to estimate the RTO.

$$RTO = Dev * SRTT \tag{3.8}$$

The use of an absolute for the Err term in equation 3.7, means that the Deviation will go up faster than it comes down.

Back Off Van Jacobson argues that the back off for consecutive retransmissions should be exponential:

> *a network is, to a very good approximation, a linear system, That is, it is composed of elements that behave like linear operators - integrators, delays, gain stages etc. Linear system theory says that if a system is stable, the stability is exponential. This suggests that an unstable system (a network subject to random load shocks and prone to congestive collapse) can be stabilized by adding exponential damping (exponential timer back off) to its primary excitation (senders, traffic sources)* [Jac88]

3.2.4 Summary

The observation of bandwidth being wasted by the unnecessary retransmission, lead to a re-evaluation of TCP timers and in particular the RTO.

Zhang shows that using timers to detect packet loss cannot achieve optimal performance, that explicit notification is more effective but that timers are necessary for when notification fails.

Khan develops an algorithm for preventing divergence of the SRTT estimate from the actual RTT when levels of loss are high.

Edge proposes measuring the variation of RTT and using it to allow the RTO to keep up with wide and narrow variations in the RTT. Van Jacobson advocates using the mean deviation to allow efficient implementation.

3.3 Connection-less Datagram Congestion Control

The load presented to a network may exceed capacity even if false retransmissions are avoided and reasonable packet sizes are used. A mechanism is required to control the load put on the network if congestion is to be controlled.

As has been discussed the Internet has a connection-less network layer, but the bulk of traffic was carried by the connection oriented transport protocol TCP. These two features were to shape the development of congestion control. The absence of knowledge about a connection in the network layer, prevented feedback of network conditions at the connection granularity and therefore the information needed to exercise session level control was not available in the network or to the endpoints[5].

The size and diversity of the Internet made the monitoring of permits necessary for Isarithmic congestion control unworkable. It also meant the traffic source may be many hops removed from the location of congestion, conditions which curtailed the effectiveness of an Input Buffer Scheme. The two main strands of congestion control which did not rely upon an association between packets were not appropriate for the Internet.

Marek Irland had proposed congestion control techniques which combined local detection, signaling to sources, adaption to traffic condition by the sources and enforcement by the network. This scheme was the reference point from which congestion control for the Internet developed.

A common assumption was that congestion control would be added to the Internet layer IP [Zha86, PP87]. The existence of state; including timers, sequence numbers, flow control windows and the association of packets into connections however, provided much of the infrastructure needed to adjust the flow of packets in response to signals from the network. For this reason the main thrust of research assumed and the schemes developed required a window flow controlled connection

[5]The experience of congestion on the Internet lead some to question its connection-less architecture [Gro85, Jr.85, Ros85], but the main response was to investigate congestion control solutions for connection-less networks

oriented transport protocol[6].

Between 1984 and 1988 three major congestion control schemes for connection-less networks with connection oriented transport protocols were developed; Nagle's Source Quench (NSQ), Congestion control Using Timeouts at the End-to-end layer (CUTE) and Binary Feedback Congestion Avoidance (BFCA), each is discussed in turn below.

3.3.1 Source Quench

In Section 3.2.1 the Ford Aerospace IP network and the congestion problems it faced are described. Here we discuss the congestion control mechanisms that were used to overcome the problem. In essence the Source Quench message was made effective by alterations to the hosts and router software controlled by Ford Aerospace.

The solution proposed and implemented by Nagle involved the interaction of routers and TCP implementations.

The scheme was motivated by the desire to prevent congestion collapse, and packet loss.

Congestion detection and notification: a router detects the onset of congestion by monitoring queue lengths. When a packet arrives at a Gateway, the buffer queue is examined and if it is over half full a choke packet is returned to source host.

Host response: when a host received a Source Quench packet it would be passed to the responsible TCP connection. Transmission of new packets would then be restricted to one packet per RTT for the duration of ten RTTs. The generation of retransmissions and acknowledgments would not be affected.

Enforcement: it is also observed that one uncontrolled host could and frequently did, deny service by flooding the network. It is therefore important for the network to be able to protect itself from misbehaving hosts. One approach is to selectively drop packets when queues become full three options are enumerated; drop a packet from the connection with the most segments in the queue, preferentially drop a duplicate packet and identify an unresponsive host and drop its packet

Implementation: throttling of traffic on receipt of a Source Quench packet was introduced to the now reference BSD implementation of TCP. This work strongly influenced congestion control in the 4.3BSD implementation [KM84]. The previous version (4.2BSD) Did nothing with Source Quench packets. Most router implementations generated a Source Quench packet when a packet was dropped or after a threshold number of packets had been dropped, commonly 20. This reduced the effectiveness of Source Quench.

To summarise Nagle's solution: the router monitors the level of congestion and explicitly passes signals back to the offending TCP which is required to reduce its rate of send. Mechanisms for detecting and controlling TCP implementations that do not cut back their rate of send are discussed but not developed.

[6]An exception to this rule was the proposal by Postel; *Source QUench Induced Delay (SQUID)* [PP87], which proposed altering the IP module to adapt the rate at which datagrams were sent between a pair of hosts, in response to the receipt of Source Quench messages. This would have had the advantage of allowing the control of all traffic produced by a host, rather than just that produced by TCP.

3.3.2 CUTE

CUTE was a congestion control [Jai86] scheme designed for the Digital Network Architecture by Raj Jain, and is appropriate for networks with a connection-less network layer and a window flow controlled transport layer like the Internet.

The aim is to prevent congestion collapse without using explicit notifications from routers. This has a double advantage; bandwidth is not consumed by choke packets and the scheme will work with dumb network elements [7].

Detection

When a network is overloaded router queue lengths build causing increased delay until packets are dropped. Packet loss is therefore a symptom of congestion and its detection can be used as an implicit signal from the network [8]. If an end-to-end reliable service is being provided the host will already detect packet loss to facilitate retransmissions.

In the CUTE scheme timeouts are used to detect loss and therefore congestion. A similar scheme was concurrently developed by Werner Bux and Davide Grillo [BG85] where packet loss is detected by negative acknowledgments from the receiver. The advantage of a time out scheme is that it is more robust, but there may be a greater delay between packet loss and detection.

Increase Decrease Policies

The reliance on packet loss to detect congestion means that congestion has reached an advanced stage a severe decrease in the rate at which packets are sent is therefore required. Adjusting the size of the flow control window is used to adapt the number of packets transmitted per round trip time to network conditions. In the absence of congestion the window is increased to probe for spare bandwidth and when packet loss is detected it is reduced.

The increase policy determines the speed at which the window is opened. The window is increased by one packet for each window of packets successfully acknowledged this gives a linear increase of one packet per RTT. The decrease policy determines the degree to which the window should be closed in the presence of a congestion signal. Closing the window to the minimum size is advocated, because of the advanced stage of congestion indicated by a packet drop. A minimal window size of one segment is proposed to enable support for the maximum number of users.

If false congestion signals are generated by too short a timeout period the window will be needlessly closed and throughput retarded. If the timeout period is to long the flow control window will fill as packets are not acknowledged and an unnecessary pause in transmission occur.

Initialisation

Jain argues that the initialization policy is the least important because over a long period it has a minimal impact on network performance. For short transfers this is not however true and if a high proportion of total transfers are short then the importance of a good initialisation policy increases. Jain does not define an initialisation policy but instead enumerates the alternatives

> *In a heavily loaded network it is safer to start at WS min. In a lightly loaded network it is more efficient to start at WS max. Other alternatives are to start at the WS used during the last connection to the same node, or to start halfway between the minimum and maximum.* [Jai86]

[7] It may be that congestion occurs in a bridge which would not be expected to engage in network level signaling.
[8] This assumes that congestion is the main cause of packet loss

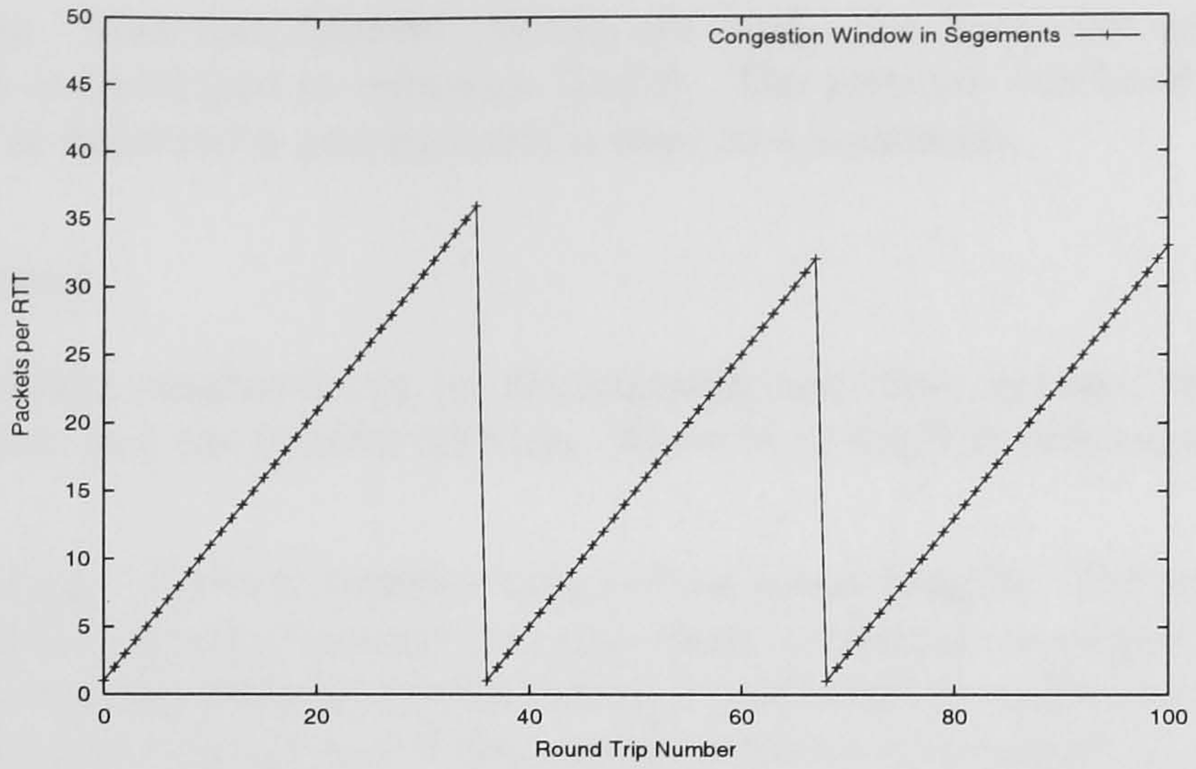


Figure 3.7: **Cute Window Adaption: Window starts at one and increases by one for each window without loss. When a loss is detected the window is reduced to one**

Over time the adaptive policies for increasing and decreasing the window will bring a long-lived connection to the same state whatever the starting policy.

Summary The chief advantages of using loss as an indication of congestion are that there is no need to generate control traffic and if congestion occurs at dumb network elements it can still be detected. By increasing the window size in the absence of congestion and decreasing it in the presence of congestion the amount of data that can be sent over 1 RTT is adapted to network conditions. The chief disadvantages are the late stage at which congestion is detected and the reliance on timers to detect loss.

3.3.3 Binary Feedback Congestion Avoidance

Jain proposes a congestion control scheme which aims to achieve optimum global performance and fair distribution of resources [JR88a, Jai86, RJC87]. It is intended for networks (like the Internet) with a connection-less network layer and assumes that all traffic is carried by a connection oriented, window flow controlled transport protocol.

There are three design aims:

1. Optimal global resource utilisation
2. Fair distribution of resource between users
3. Low resource consumption by the scheme

Design Aims

A distinction is made between congestion avoidance, which is the aim of this scheme and congestion recovery. A congestion avoidance scheme aims to keep the network operating in the region of the knee of the load, delay curve and thus maximise the global power of the network. If the operating point is to the left of the knee throughput is reduced and if it is to the right delay increases rapidly, therefore divergence in either direction reduces network performance.

Congestion recovery aims at operating the network to the left of the cliff, see Section 5.3.3 and figure 5.6. Fair distribution is interpreted as throughput at a resource being distributed equally

between connections. Max min fairness criteria are used. An index for quantifying the level of fairness achieved is developed in reference [Jai84]. The resource overhead needed to run the scheme, at routers, in bandwidth and by hosts is kept to a minimum.

Control Mechanisms

The control mechanisms employed can be decomposed into three groups; resource monitoring, signaling to the source and the sources reaction. There is no explicit enforcement policy.

Resource Monitoring Routers monitor their output queue lengths. The instantaneous results are not communicated to sources, because this may result in premature congestion indications and in separate sources receiving different signals, which would result in unfairness. Instead the queue lengths are averaged over a period and if the average is above a threshold (1) a decrease signal is sent otherwise an increase signal is used. Queue lengths are averaged from the start of the previous busy period, through the previous idle period to current time. Other periods were rejected because they resulted in different sources receiving different signals and therefore in unfairness.

Signaling Previous congestion control schemes utilised special packets or relied upon implicit signals to convey congestion information from the network to the source. Innovatively here the signal is conveyed to the source by the router setting a bit in packet headers upon reaching the destination the signal is returned in the transport layer acknowledgments. Only one bit in each header is required and no extra traffic is generated thereby realising the aim of minimising resource consumption.

User Reaction The transport layers flow control window is adjusted in size in response to increase and decrease signals from routers. Three issues are addressed; the frequency with which increase decrease decisions are made, the way in which the signal is filtered and the increase decrease policy.

There is a delay between a source changing its window size and the result of the change being reflected in the feedback the source receives from the network. For this reason the adjustment upon the receipt of every acknowledgment was found to cause large oscillations in window size. A decision to adjust the window is only taken after a full window of new data has been acknowledged since the previous window and only the congestion indications from the new data are used.

If 50% or more packets had the congestion indication bit set the window will be decreased, otherwise it is increased. In reference [CJ89] it was shown that a multiplicative decrease and additive increase policy result in the system converging to optimum utilisation and fair distribution, if all users use the same path and receive the same signal. This result holds for both rate and window based pacing and convergence is achieved from any starting point.

Adjusting the increase and decrease parameters effects the speed with which the system converges to equilibrium and the size of the oscillations around equilibrium. Jain recommends an increase of 1 and a decrease factor of 0.875 (7/8), which results in low oscillation around the knee.

Summary

All three congestion control schemes focus upon the steady state behaviour of a bulk transfer connection. It is assumed that over the life time of an association, the start-up behaviour will have little effect upon overall performance.

A distinction can be drawn between the aims of the three schemes. CUTE and Source quench aim to detect the occurrence of congestion and recover from it thereby preventing the network

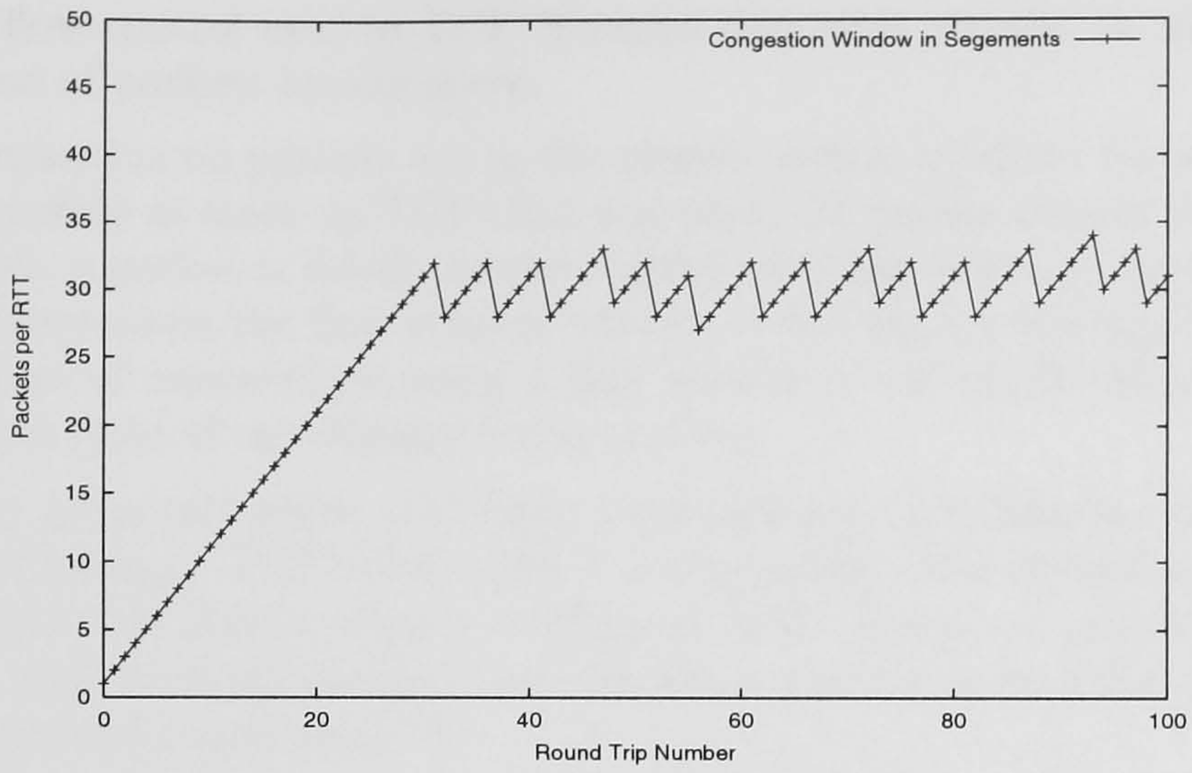


Figure 3.8: **BFCA Window Adaption: The window is initialised to one and increased by one for each window without loss. In the event of loss the window is reduced by an eighth. This is an Additative Increase Muliplicative Decrease (AIMD) algorithm.**

descending into a state of congestion collapse. CUTE demonstrates that congestion recovery can be facilitated by hosts detecting packet loss and using it as an implicit congestion signal.

BFCA is more ambitious in that it aims to minimise packet loss, allow the network to operate around its optimal load and distribute bandwidth equally between connections sharing a bottleneck. To achieve these aims the scheme requires resources to monitor traffic and transport a binary increase decrease signal to the source.

The multiplicative decrease and additive increase algorithms are shown to result in stability and fairness between connections, if each source receives equivalent signals and the gateway correctly determine whether to indicate an increase or decrease.

The existence of a congestion avoidance scheme utilising explicit congestion notifications, does not render congestion detection by implicit means unnecessary, because the congestion may occur in a network element not capable of issuing a notification or the notification may fail.

3.3.4 Slow Start and Congestion Avoidance

In 1988 the National Science Foundation upgraded its backbone from 56 kbps to 1.544 mbps and Van Jacobson presented his paper *Congestion Control and Avoidance.* The combination of increased backbone bandwidth and the implementation of Jacobson's algorithms in TCP would help bring congestion down to manageable proportions.

The congestion control mechanisms proposed for TCP were aimed primary at controlling bulk data transfers. Starting from the idea[9] that if the number of elements making up a flow remained constant congestion should not occur.

> *The flow on a TCP connection should obey a conservation of packets principle. And if this principle were obeyed, congestion collapse would become the exception rather than the rule.* [Jac88]

The conservation of packet principle states that when a connection is in a state of equilibrium a packet should only be injected into the system when one leaves, which is the normal behaviour of

[9]Similar to Davies Isarithmic ideas but applied to a connection rather than the whole system

the sliding window flow control used in TCP. Starting from this premise he identified three ways in which conservation of packets breaks down.

At the start of a connection no packets are in the system, and a windows worth of packets can be transmitted, consequently at start up TCP does not obey the packet conservation rule. A second violation occurs when a packet is falsely retransmitted, this has been discussed in Section 3.2.3. The third failure occurs when the flow control window is too big for the bandwidth available, or if the sum of windows of connections using a link results in too much demand for bandwidth, therefore preventing a state of equilibrium being reached.

Two algorithms were proposed which addressed these failures. The first is called Slow Start and is used to get data flowing if TCP's self-clock is not running. The second is called Congestion Avoidance and controls the steady state behaviour of TCP. These are discussed in the separate sections which deal with start-up, steady state behaviour and the interaction between Slow Start and Congestion Avoidance respectively [10].

Finally a mechanism for detecting lost packets, without having to wait for an RTO time out is proposed and Fast Recovery and Retransmit algorithms which define how a TCP connection should behave when congestion is detected using this method are discussed.

Slow Start

Slow Start addresses two problems; how to get a connection quickly to an efficient utilisation of bandwidth per RTT without seriously overshooting network capacity and how to ensure that packets within a round trip time are evenly distributed.

Prior to the introduction of Slow Start a connection could put a full window of packets onto the network at start-up. This could result in a load several times higher than intermediary routers could support.

A connection using Slow Start first transmits one packet, when the acknowledgment for that packet is received two packets are sent and for each subsequent acknowledgment of new data two packets are transmitted. There is therefore an exponential increase in the number of packets sent per RTT. A connection in Slow Start can only double the number of packets it has in the system each RTT, it can therefore overshoot the total bandwidth by at most a factor of two.

To open the window to a given window size takes:

$$T = Rlog_2W \tag{3.9}$$

where T is the time to open the window, R is the RTT and W is the window size in segments. This compares favourably to CUTE and BFCA where;

$$T = RW \tag{3.10}$$

It may still take several RTTs to reach high levels of bandwidth utilisation, this is particularly a problem for paths with large bandwidth delay products (for example satellite links [AHKO97]) and short lived connections.

Slow Start addresses the distribution of packets within a RTT as well as controlling the number of packets transmitted per RTT. Previously a full window of packets would be transmitted at the speed that the local network supported. The send one wait for an ACK and then send two behaviour of Slow Start would allow the network to shape traffic. The hope was that the spacing established on the bottleneck link on the outward journey, would be preserved in the returning

[10]Congestion Avoidance is the name given by Van Jacobson to the algorithm which controls the steady state behavior of TCP, whether the algorithm aims for congestion recovery or congestion avoidance as defined by Jain is discussed later.

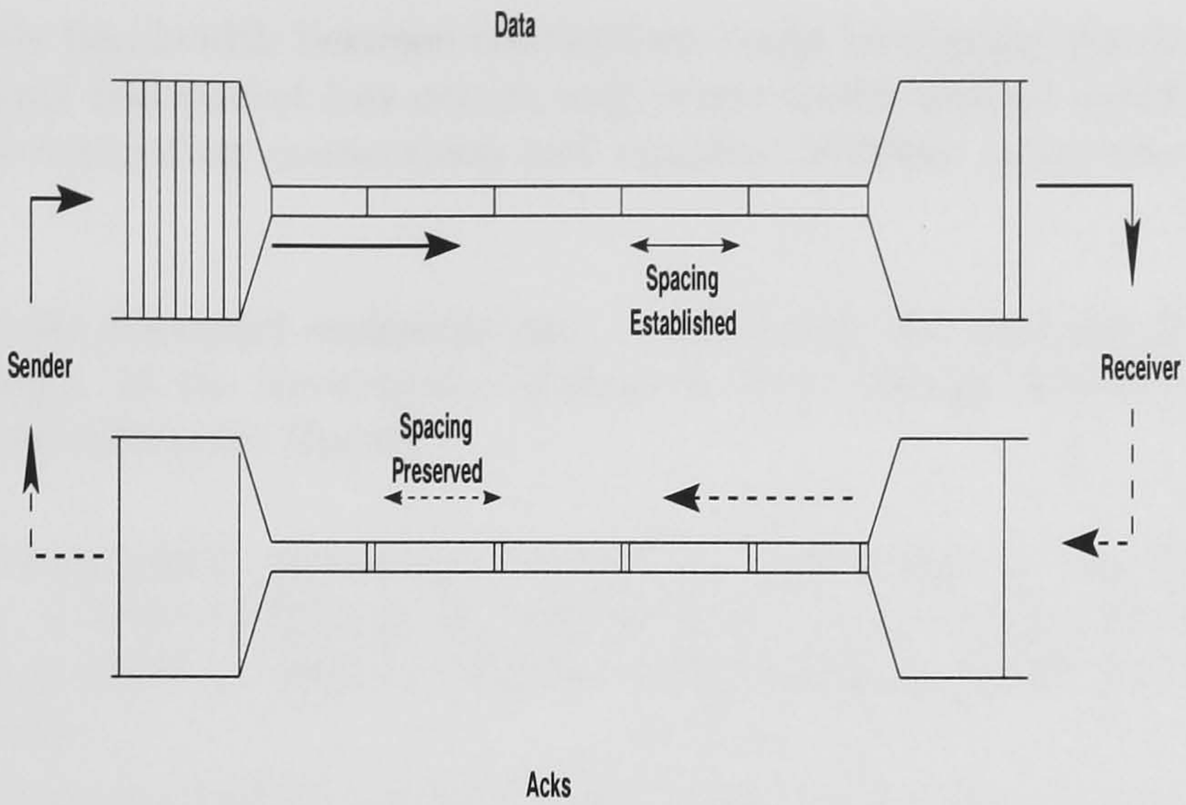


Figure 3.9: **Self Shaping Traffic: Acks come back at the rate of the outward bottleneck thereby shaping the transmission rate within a window of data.**

acknowledgments (as illustrated in figure 3.9[11]). Consequently the maximum overshoot of capacity would by by a factor of two on the packet as well as the RTT timescales and no explicit traffic shaping by the sender would be necessary[12] and none was recommended.

Implementation requires that a congestion window variable is maintained by the sender, it is called `cwnd` and holds the number of segments Slow Start allows to be sent but unacknowledged[13]. The sender can have up to the minimum of the `cwnd` and the receiver's advertised window packets unacknowledged by the receiver.

To summarise Slow Start is a mechanism for getting traffic flowing on a connection. It limits a connection's ability to overshoot total bandwidth to a factor of two and relies upon implicit network traffic shaping to avoid bursty traffic.

Congestion Avoidance

Detection Packet loss was proposed as the method for detecting connection, which was essentially the same as that used in the CUTE scheme 3.3.2.

The alteration of the TCP timer algorithms discussed in Section 3.2.3 lead to greater confidence that the RTO estimate could keep up with rapid increases in RTT and a time out would not be a false alarm but would indicate a dropped packet. As discussed in Section 3.2.3 using timers to detect error conditions is a guess based upon incomplete information, consequently false alarms and delays between packet loss and detection are inevitable in any system which depends solely on timers.

It has been argued that packet loss is a particularly extreme form of congestion notification, because it occurs late in the development of congestion and it would be desirable for packet loss to be avoided by rather than relied upon in a congestion control scheme [GK98].

Reaction TCP's reaction to the absence and presence of congestion is based on Jain's Additive Increase and Multiplicative Decrease (AIMD) algorithm. The congestion signal in BFCA and TCP is not however the same, this leads to different fairness properties and points of equilibrium. BFCA was designed so that each connection would receive the same feedback signals from the

[11] Based on a diagram in [Jac88]

[12] This assumes that the return and outward paths are symmetrical, which may not be the case.

[13] The congestion window is actually maintained in bytes, we will use segments in this discussion for simplicity.

network consequently bandwidth between connections could be equally distributed. When packet loss is used as a signal and packet loss occurs only when router queues overflow, the same signal is not generated to competing connections and equality between connections is unlikely to be achieved.

> *Algorithms at the transport endpoints can ... not insure fair sharing of that capacity. Only in gateways, at the convergence of flows is there enough information to control sharing and fair allocation* [Jac88].

It could be argued that whilst competing connections achieving equal bandwidth allocations is difficult to achieve, a looser definition of fairness where each connection has a fair chance of utilising bandwidth is easier to achieve. This has been called statistical fairness and is achieved over many connections.

A second difference between BFCA and TCP is the point around which the systems oscillates. For congestion avoidance equilibrium is around the knee of the delay load curve. The TCP scheme operates to the right of the cliff, it is therefore more accurately described as congestion recovery rather than congestion avoidance. The increase parameters used in the two schemes are the same; the congestion window is increased by one segment per RTT. In TCP this is achieved by increasing `cwnd` by `1/cwnd` segments per ACK. This allows the connection to probe the network for more bandwidth. Van Jacobson recommends a decrease parameter of 1/2 compared to 1/8 in BFCA.

There are three reasons for using a half rather than an 1/8. If packet loss was caused by Slow Start overshooting, then it is known that the last good window size was at least half the current size. If packet loss was caused by a new connection coming in and taking bandwidth, the maximum reduction in the existing connections fair share will be by 50%, in the case where there was previously one connection and now there are two sharing the bottleneck. A third consideration is that congestion is detected in a more advanced state and therefore a more aggressive shutdown is sensible.

The Combined Algorithm

To decide whether a connection should be using the Slow Start or the Congestion Avoidance algorithm another variable is required to mark the slow start threshold and is called `ssthresh`. If the congestion window is smaller than `ssthresh` the connection will use Slow Start otherwise congestion avoidance will be used. When congestion is indicated by a Retransmit Time Out (RTO) `cwnd` is set to one segment and `ssthresh` is set to half of the old congestion window. The window therefore opens exponentially until it reaches half of the old size and then the Congestion Avoidance algorithm is used.

At start-up `ssthresh` is initialised to a default value. This default is usually the maximum window size allowed by the host. Slow Start will therefore continue until either packet loss occurs or `cwnd` reaches the maximum offered window. The advertised window, which is intended to regulate flow control between hosts and communicate the buffer space at the receiver, actually plays an important role in congestion avoidance. If the advertised window is greater than a connection's fair share then Slow Start is guaranteed to overshoot that fair share or encounter packet loss or both. If the advertised window is less than the connections fair share then fair utilisation can never be reached.

```
if (cwnd < ssthresh) cwnd += 1
else cwnd += 1/cwnd
```

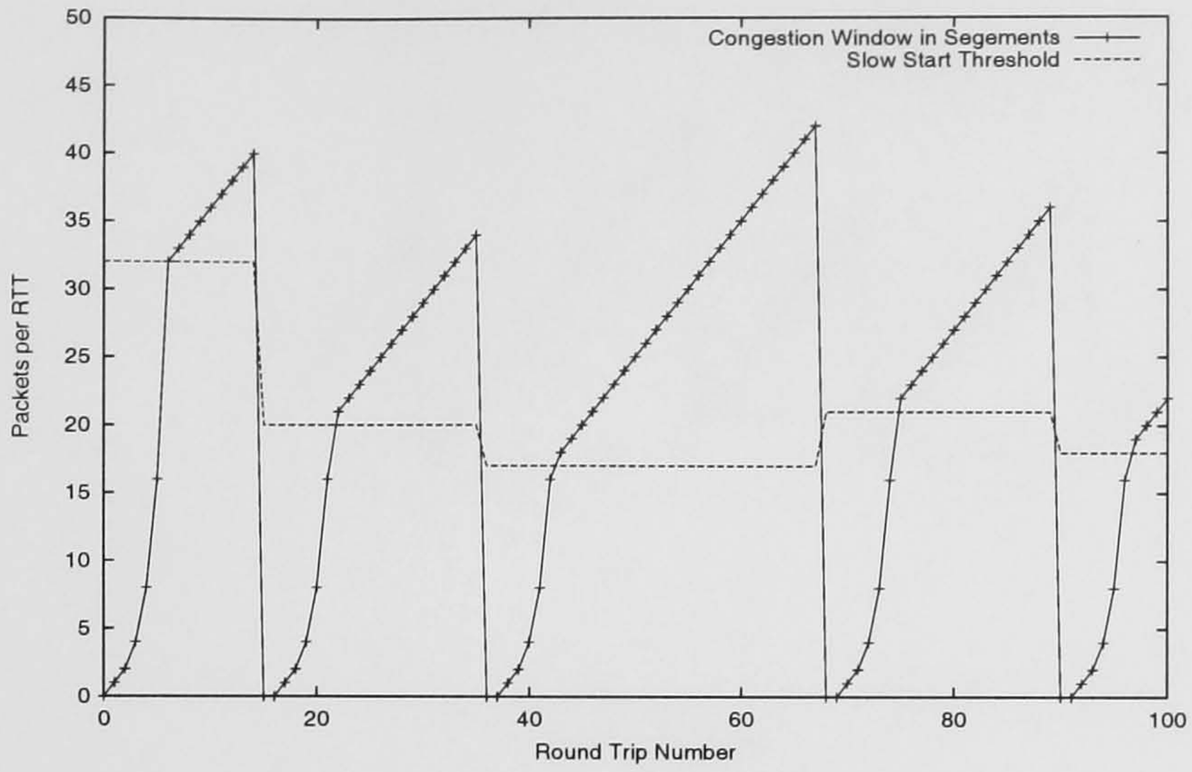


Figure 3.10: **Slow Start and Congestion Avoidance: The congestion window is increased exponentially for each window of data without a loss whilst the congestion window is smaller than the slow start threshold. If the congestion window is larger than the slow start threshold it is increased additatively for each window without loss. When a loss is detected the congestion window is reduced to one.**

Fast Recovery and Fast Retransmit

In 1990 Van Jacobson proposed the addition of Fast Recovery and Fast Retransmit algorithms to TCP. This was motivated by the poor performance on large bandwidth delay paths [Jac90, JB90]. When TCP receives an out of order segment, it generates an immediate duplicate acknowledgment. This ACK tells the sender, that a segment was received out of order and what the starting sequence number of the first missing packet is. If several consecutive duplicate ACKs are received this indicates that the out of order segment was due to packet loss and not packet reordering or duplication. Duplicate ACKs can therefore be used as a signal of congestion thus avoiding the wait for a timeout. Duplicate ACKs also show that data is continuing to flow across the network, indicating that congestion collapse has not occured. In the case where a packet has been lost but data is continuing to flow, it is desirable to cut back but continue the flow of data by avoiding the wait for a timeout and the cost of falling back into Slow Start [Jac90, JB90].

Fast Retransmit and Fast Recovery are the algorithms that are triggered by congestion signaled by duplicate ACKs. They are usually implemented together as follows.

In Reno TCP when 3 duplicate ACKs have been received `ssthresh` is set to half of the current congestion window. Each subsequent duplicate is taken to indicate that a packet has left the network, the senders congestion window is therefore increased by one. The inflation of the congestion window allows the sender to keep transmitting. This is fast retransmit.

When an ACK is received that acknowledges new data Fast Recovery takes place. The inflated window is reduced to the size of `ssthresh` and the normal Congestion Avoidance algorithm is used. In this way packet loss is detected without a timeout, the multiplicative decrease in the window size is maintained, traffic is kept flowing across the connection and Slow Start is avoided.

Efficiency is improved by using a more direct congestion signal, therefore the importance of the RTO is reduced. Consider the twin requirements for an RTO, that it occur as soon as possible after loss has occured and that no false alarms occur. With the use of duplicate ACKs the need for a quick RTO is lessened, but the requirement for no false alarms remains. The main requirement for the RTO is that it be a conservative estimate, provided duplicate ACKs provide an efficient packet loss detection mechanism.

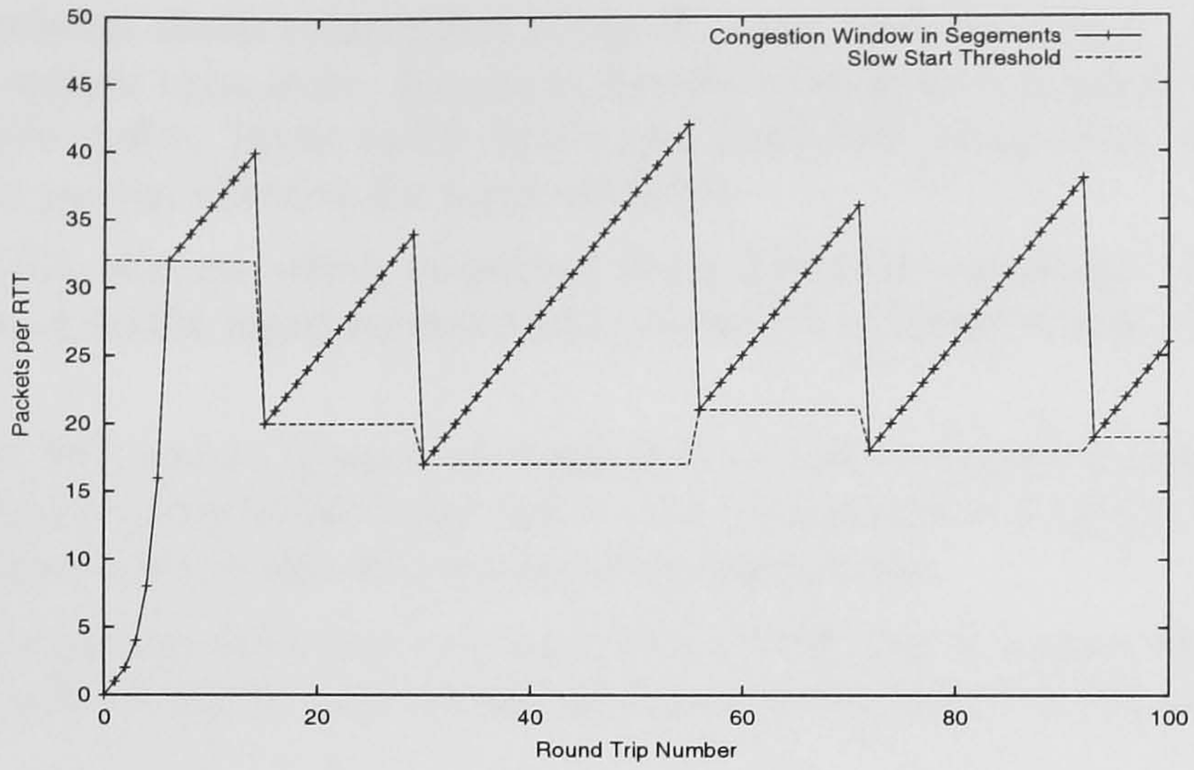


Figure 3.11: **Fast Retransmit and Recovery: When packet loss is detected through the receipt of duplicate acknowledgements the congestion window is cut in size by half.**

3.4 Conclusion

The only explicit assumptions made by IP about the networks it interconnected, were that they supported addressing and could transport a packet. On the early Internet there was a balance between the capacities of the constituent networks and flow control implemented in the ARPANET was important in limiting congestion.

When networks without their own congestion limitation schemes and with large mismatches in capacity were interconnected, congestion collapse resulted in greatly inhibited communication.

This phenomena became more common as a high capacity Local Area Networks were connected by low capacity Wide Area Networks. Capacity planning could limit congestion control be developing a balanced network architecture, yet the mismatches in bandwidth were a consequence of technological and economical considerations. To provide WANs that matched the capacity of LANs would be to inhibitively expensive and technicly difficult, to limit LAN capacity to WAN proportions would unnecessarily constrain the bandwidth between local nodes. Thus congestion control by capacity planning was rendered ineffective.

Flow control on the ARPANET limited the amount of data that could be outstanding between a pair of hosts. With a small number of hosts and excess capacity this could be effective. With the growth of the number of LANs connected to the ARPA Internet and the number of hosts connected to those LANs congestion it became less effective. In addition congestion could occur in non ARPANET gateways.

The two implicit assumptions which IP depended on for traffic control ceased to be valid and congestion became a serious problem on the Internet. The growth of congestion was accompanied by renewed interest in optimising use of network resources and in congestion control.

Traffic on the Internet was dominated by small bandwidth interactive traffic and bulk transfers. The interactive traffic generated a large numbers of packets with only one byte of data. There was a large overhead of 40 bytes headers for each byte. This was improved by adopting Nagle's algorithm.

Bulk transfers wasted bandwidth when unnecessary retransmissions took place. Improvements in the estimation of the RTO timeout greatly reduced unnecessary retransmissions. If the small packet and false retransmits problems are solved the central problem of controlling the load presented to the network remains.

The absence of knowledge about connections in the IP layer, inhibited the development of congestion control at the session time scale. Research therefore focused on congestion controlled which operated at finer time scales. Input buffer limits and Isarithmic congestion control proved to be dead ends as neither proved effective for large networks.

The CYCLADES CCL scheme which combined local detection, signaling of congestion to end points and adaption of traffic input by end points became a significant reference point for future developments.

Nagle developed the IMP source Quench message into an active congestion control mechanism in the portion of the Internet run be Ford Aerospace, but the absence of a consistent implementation in routers across the rest of the Internet inhibited its general use.

Jain developed a Congestion Recovery scheme called CUTE and a Binary Feedback Congestion Avoidance scheme, both of which were influential in the development of the algorithms that were adopted for TCP.

All these schemes depended upon a connection oriented transport protocol and therefore only regulate traffic that was part of a connection. In addition congestion control became closely bound to the transport services offered by the transport protocol.

In the Internet this meant that congestion control was located in TCP, but not in UDP. As TCP offers only a reliable service, traffic that requires a connection but is damaged by a reliable transport service, such as real time voice is not controlled.

The absence of an explicit signal of traffic condition in the network and the desire not to have to alter all routers in order to generate one lead to the adoption of packet loss as an implicit congestion signal. This was despite the advantages in fairness, stability and efficiency that could be gained by using explicit congestion notification, which were demonstrated by the Binary Feedback Congestion Avoidance scheme developed by Jain.

The Slow Start and Congestion Avoidance mechanisms proposed by Van Jacobson and implemented in TCP have been the Internet's congestion control mechanism throughout the 1990's. They succeeded in the central aim of preventing congestion collapse. TCP congestion control takes place within each connection. The primary motivation for its introduction were the episodes of congestion collapse experienced by the Internet and its central aim was to prevent recurrences of those collapses.

Although the controls were designed for bulk transfers, and it was assumed that start-up would be a transient state that if handled correctly would have little impact on the overall performance achieved by a connection, and unlike previous similar schemes; a specific start up algorithm called Slow Start was proposed. Compared to previous TCP behaviour this reduced the overshoot of bandwidth that could occur in one RTT at the start of a connection and compared to CUTE it increased the speed that a connection could reach equilibrium.

Slow start also addressed the distribution of packets within one RTT. It was hoped that the network would space returning ACKs so that the burstiness of outgoing traffic would also be controlled. Like CUTE packet loss is used as a signal for congestion unlike CUTE a timeout is not the only way of detecting a dropped packet. The receipt of duplicate ACKs is the first line of congestion detection.

The steady state behaviour of TCP draws upon the BFCA scheme, in that it uses an AIMD algorithms to adapt output in response to congestion signals. It is different in that the signal does not allow the scheme to oscillate around the knee of the delay load curve, but instead equilibrium occurs to the right of the load throughput cliff. It is therefore a congestion recovery rather than a congestion avoidance scheme. In addition the signal does not allow equal distribution of throughput between connections to be achieved.

The reliance of the Internet on congestion control located within TCP connections, means that it in turn relies upon the bulk of traffic being transported by TCP. The absence of explicit traffic shaping and congestion control at session time scales means there is only one time scale over which

explicit control of traffic is exercised.

Across the dimensions of times, network element and protocol layer explicit control is only exercised at one point, which can be seen as a consequence of adding congestion control post hoc to an existing architecture

Chapter 4

An Evaluation of TCP Congestion Control

The combination of initialising the congestion window to a single segment, and the exponential increase of Slow Start mean that a TCP connection could overshoot the physical bandwidth, but that such an overshoot is limited to a factor of two; less if delayed acknowledgments are being used. In the circumstance where there are high bandwidth Local Area Networks interconnected by low bandwidth Wide Area Network links with a low level of multiplexing, this provides an important upper limit on the aggressiveness of TCP's start up. Changes in wide area technology and expansion of the Internet's user base however, mean that end points are often interconnected by high bandwidth Wide Area Networks where there is also a high level of multiplexing. In this environment the extent to which TCP overshoots fair bandwidth utilisation is more relevant than its potential to overshoot the physical bandwidth. As new access technologies are deployed and more users acquire high bandwidth access to the Internet, the limitation imposed by the needs of competing traffic will reinforce the importance of the fair distribution of network resources.

The phenomenal growth of the World Wide Web means that for many, it is the main way that the Internet is used. Consequently users frequently interact with the Internet using point and click interfaces that benefit from short response times. The latency between a user clicking on an URL or an action button and the response being displayed is therefore important in determining user satisfaction.

This chapter provides a re-evaluation of TCP's congestion control algorithms within the context of high bandwidth WANs, a high level of multiplexing and with a particular focus on the needs of web traffic. The evaluation focuses on the contribution made by TCP to the latency experienced by users and the extent to which it achieves the goal of the fair distributing of constrained network resources between competing demands.

In conducting this evaluation, a number of methodologies are deployed; the automated modelling of the behaviour of TCP algorithms is used to explore the parameter space, measurements of web traffic are used to provide points of reference in the discussion of the effects characterised by the modelling and controlled generation of experimental traffic is used to determine the significance of client and server limitation.

The methodology of trace collection and the design of the tool used to obtain measurements is given in Appendix A, discussion of the automated modelling and experimental techniques are located within the body of this chapter.

Context is provided by considering the degree of tolerance that users have to delay and the point at which further reductions are no longer beneficial. This is followed by discussion of the limitation imposed during the data transfer phase of a connection by the network infrastructure

and competing network traffic, which collectively is referred to as network limitation. This section includes the presentation and discussion of the average level of congestion by location and the distribution of network round trip times.

Next, contributions to the latency attributable to the transport protocol are analysed. Three constituent elements are identified; error recovery, start-up and flow control. By considering the contribution of each in turn it is possible to identify areas for improvement and quantify the bounds which limit such progress. The significance of connection confined adaptation is analysed first by modelling the interaction between Slow Start and Congestion Avoidance for a range of different connection sizes and congestion regimes. Packet traces of Internet traffic with the University of Glasgow were then analysed to determine the distribution of Internet connection sizes, thereby facilitating a concrete discussion of the identified effects.

Having discussed TCP's congestion control its relationship to flow control is treated. It is shown that for long connections, where it might be expected that congestion feedback would determine average window sizes, a significant role is played by window limitation. This effect is easily underestimated by measurement studies which consider all connections, because short connections can never become window limited. The chapter is rounded off by discussion of the significance of the contribution of Client and Server limitation to latency.

4.1 Latency and Human Perception

The delay between clicking on a web page and the result being displayed is shown in [RAP98] to be an important factor in determining a user's perception of the quality of the page itself, with significant deterioration in perceived quality occuring for delays in the order of seconds. Furthermore these perceptions are shown to be dependent upon the expectations of the users.

There is however, a limit beyond which further reductions in latency are no longer perceptible. When a bright light is flashed on and off there is a certain frequency at which it appears as a single light. The perception interval varies from person to person but has been established as being between 60ms and 140ms [CMN83]. The implication is that delays which are smaller than the perception interval are not noticeable. This in turn defines a hard limit below which there is no benefit to the user, in further reducing the latency of web traffic. It will become clear that we are a long way from reaching this limit.

4.2 Network Limitation

A connection can be said to be network limited when the fair rate at which the network is capable of delivering data is not fast enough to keep up with the rate at which the data is presented by the server or consumed by the client. How then is it possible to determine this rate?

For a given network path it has been shown that Additive Increase Multiplicative Decrease (AIMD) [CJ89] algorithms lead to the fair and stable distribution of bandwidth between competing connections. TCP's steady state behaviour is governed by the particular AIMD called Congestion Avoidance [JB90]. The dominance of TCP traffic on the Internet has lead to the adoption of Congestion Avoidance as a benchmark against, which the fairness of other traffic is judged [MF97]. This leads us to the conclusion that analysis of TCP's Congestion Avoidance algorithm can be used to determine the fair rate at which the network delivers data, and therefore the rate at which network limitation becomes active.

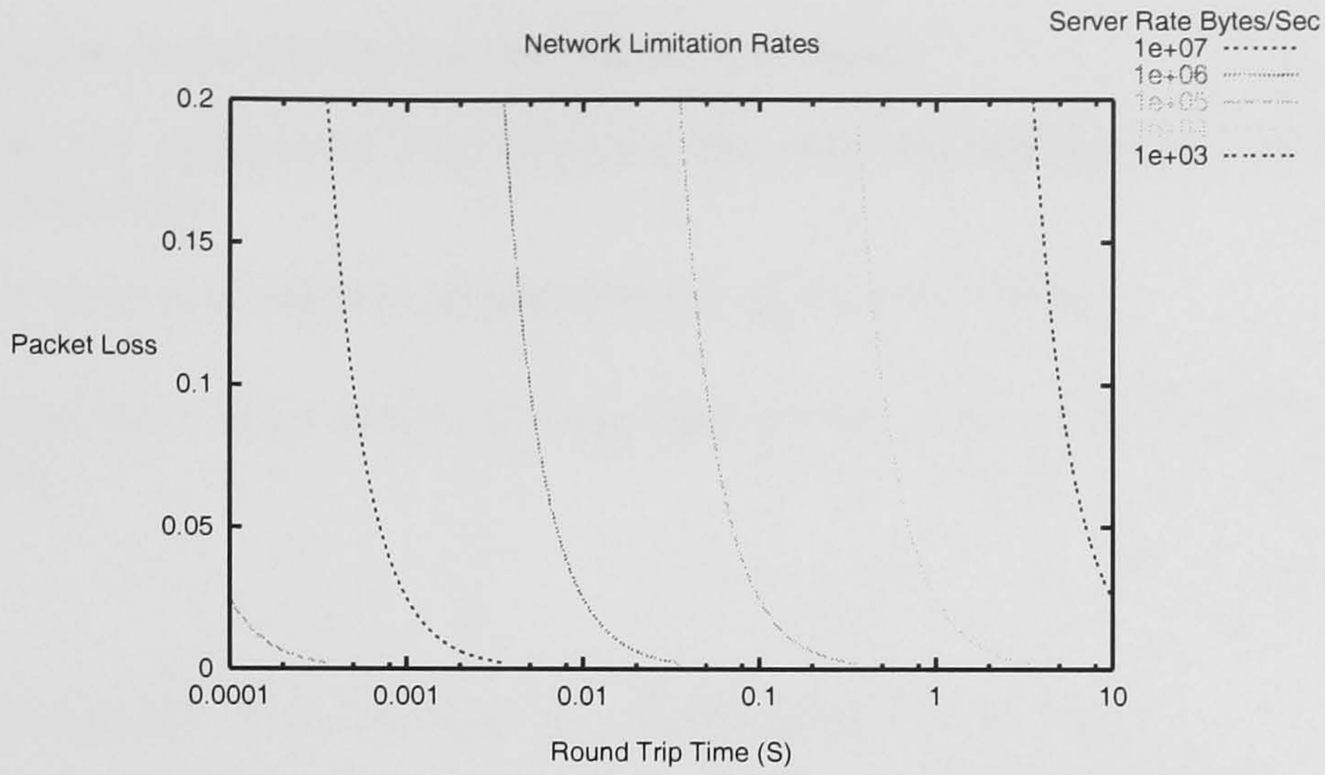


Figure 4.1: **Network Limitation Boundaries: Each line represents a rate of data transfer supported by a server. To the right of each line that server will experience network limitation.**

4.2.1 TCP Friendly Models

In recent years several models of TCP behaviour have been developed. These have been motivated in part by the growth of non TCP Internet traffic, which has lead to concerns of a return of congestion collapse [MF97]. Applications which use the Internet for real-time voice or video communication have demands which are not suited to the reliable service provided by TCP. Consequently the Internet's congestion control mechanisms are bypassed by such applications. By modeling TCP's behaviour it is hoped to be able to detect and penalise streams which are unresponsive to network conditions and to provide a benchmark that developers will follow in building congestion control into applications. Data streams which conform to the model of TCP congestion behaviour, but do not utilise the same algorithms are often described as being *TCP Friendly*. Indeed protocols have been designed which use these models to validate TCP Friendly behaviour [WS99, RHE99] or use the model directly to set transfer rates [PKTK98].

Any model of a system is an abstraction which captures the essential behaviour and discards some inconsequential or undesirable features. The model proposed in [MSM97, OKM99, MF97] has been shown to closely approximate TCP performance for long connections under moderate loss, where the congestion avoidance algorithm governs transmission.

Assumptions

The following equation is based on the assumptions below:

1. The connection is not application limited.
2. There is always data to send.
3. Loss is deterministic.
4. Loss is shared equally between concurrent connections.
5. Network conditions remain stable for the duration of a connection,
6. The receiving application can keep up with the rate at which data is delivered.

7. The connection is not limited by the receivers window.

8. Connections are sufficiently long lived for the start-up transients to have little effect on overall performance.

9. Retransmit time outs will not be necessary to recover from loss.

These assumptions have been shown to approximate the behaviour of TCP steady state behaviour [OKM99].

The Model

In this model throughput (T) is given as a function of loss (P) and RTT.

$$T = \frac{C * MSS}{RTT * \sqrt{P}} \tag{4.1}$$

In practice a number of factors mean that TCP connections often do not reach the throughput implied by the model described above. Firstly, the way in which TCP's loss recovery mechanisms interact with congestion avoidance may cause throughput to be severely reduced. Secondly, the Slow Start algorithm, which is used to quickly probe the network for available bandwidth may have a significant effect on the average behaviour of a connection. Thirdly, the offered window size, which the receiver uses to limit the number of unacknowledged packets, may also depress bandwidth utilisation. Collectively these three factors can be thought of as Protocol Limitation in that they are all effects of the TCP protocol, which influence the extent to which a connection's bandwidth utilisation deviates from its fair share.

A model proposed by Padhye [PFTK98, PFT99] more accurately captures the behaviour of the Reno variant of TCP [PFTK98, PFT99]. It includes RTOs and receiver window limitations but does not account for start-up transients. Cardwell et al. define a model for short TCP connections [CSA88]: they assume that the connection never leaves Slow Start. Such an approach is useful for short connections, but is not useful for comparing long with short.

The different functions that a model may perform will influence the features that are incorporated. There are three common reasons for wanting to model TCP. 1) To detect non *TCP Friendly* flows [FF97, FF99]. 2) To determine whether new congestion control algorithms are TCP friendly [RHE99], for example where rate based flow control is being used, or to use the model to control traffic streams [PKTK98]. 3) To accurately predict the behaviour of TCP.

Padhye's model is certainly useful in predicting TCP's behaviour. The problem with using TCP as a benchmark for detecting non-conformant flows or to judge new congestion control algorithms is that it has features that it is better not to reproduce. At the same time as TCP behaviour is being used as a model for new applications, research is being conducted into various ways of improving TCP itself, for example preventing the overshoot of slow start, reducing the frequency and impact of RTOs and dynamically adjusting buffer and therefore flow control window sizes. Consequently, the first model described above is better for this function.

As equation 4.1 captures the behaviour of TCP's congestion avoidance algorithm and abstracts away other factors, it provides an accurate estimate of the network limitation on throughput. Figure 4.1 shows the boundary of network limitation for a range of round trip times and levels of congestion. Each line represents a rate in bytes per second. Co-ordinates to the left of the line will not be network limited at that rate and points to the right of the line will be network limited.

From 4.1 to determine the network limitation for a particular path it is necessary to know the level of congestion, the round trip time, the maximum segment size and the constant C. In [Rud00] network measurement was used to estimate a value for C of 1.079, this result is slightly larger than the value obtained analytically under deterministic and random loss assumptions [WC91]. The

most common maximum segment size on the Internet is 1460 data bytes, which is a consequence of the Ethernet segment size, the next most common is 536 bytes, which is simply an old default value. Methods for determining packet loss rates and the RTT are discussed next.

4.2.2 Congestion Measurements

The Internet implicitly signals congestion to hosts by dropping packets. When congestion occurs at a router it takes at least one round trip time for the signal to reach the source, for the source to adjust its rate of send and for that adjustment to be reflected in the load at the congested router. Packets which are dropped in this period simply indicate congestion at the unmodified transfer rate and do not show that a further cut in the congestion window is required. It takes a further RTT for the effect of that change to be reflected in the congestion signal received by the host.

Furthermore the bursty nature of TCP traffic, when combined with drop tail routers, often results in more than one packet being dropped from a source during a single congestion event. For these reasons there is a consensus that it is acceptable for a TCP connection to only adjust its congestion window in response to a congestion signal once per RTT. This is reflected in the RFC for Explicit Congestion Notification [RF99], which only requires TCP to react to one ECN per window of traffic.

When TCP detects a dropped packet, either through the receipt of duplicate acknowledgments or after the expiration of an RTO, the segment is retransmitted. The detection of retransmissions can therefore be used to determine the level of loss on a path. In order to accurately determine the level of congestion it is however necessary to filter the loss signal. Specifically once a retransmission has been detected further retransmissions are not counted as indicating congestion until new data has been sent and the retransmission has been acknowledged. This will take at least one RTT. The exception is when a retransmission is itself retransmitted, this is counted as a congestion signal.

To reflect the distinction introduced between the level of loss and the level of congestion a new term Implicit Congestion Notification (ICN) is used in this dissertation which reflects the use of implicit congestion signalling on the Internet, but refers to the filtered rather than raw level of loss. Typically the congestion signal is presented as a Proportion of ICNs (PICN), which may simply be referred to as an ICN; context will make it clear whether a proportion is being referred to.

$$PICN = \frac{\sum ICN}{\sum DataPackets} \tag{4.2}$$

Expected Levels of ICNs

In this subsection some measurements of levels of congestion are discussed. For a meaningful proportion of ICNs to be obtained the proportion should be calculated over a sufficient quantity of data. To achieve this it is often necessary to aggregate readings for a number of connections, as many connections are too small to yield a meaningful average and each connection only provides one temporal reference point.

The aggregation used here is by foreign location, an average of the ICN proportion was calculated for all traffic from the local network for each location. A location was defined as consisting of all the hosts which share the same higher order three bytes in their network address. For the period under study transfers occurred to some 21,000 locations, this compares with 28,000 unique foreign IP addresses. The level of aggregation could be increased by aggregating on foreign network addresses.

When there is only one data packet transferred, the ICN proportion can either be one or zero. As the number of packets increases the number of values that PICN can take increases and the likelihood that it approaches a true value increases. If the readings are taken from only one

% Locations	Min. Cons.	% Cons.
100	0	100
99	0	100
90	1	99.72
80	1	99.05
50	3	95.19
20	11	84.07
10	26	73.77
5	55	62.33
1	207	39.14

Table 4.1: **Minimum Number of Connections: This table shows that 5% of locations accounted for 62% of all locations. In addition each of these locations had a minimum of 55 connections.**

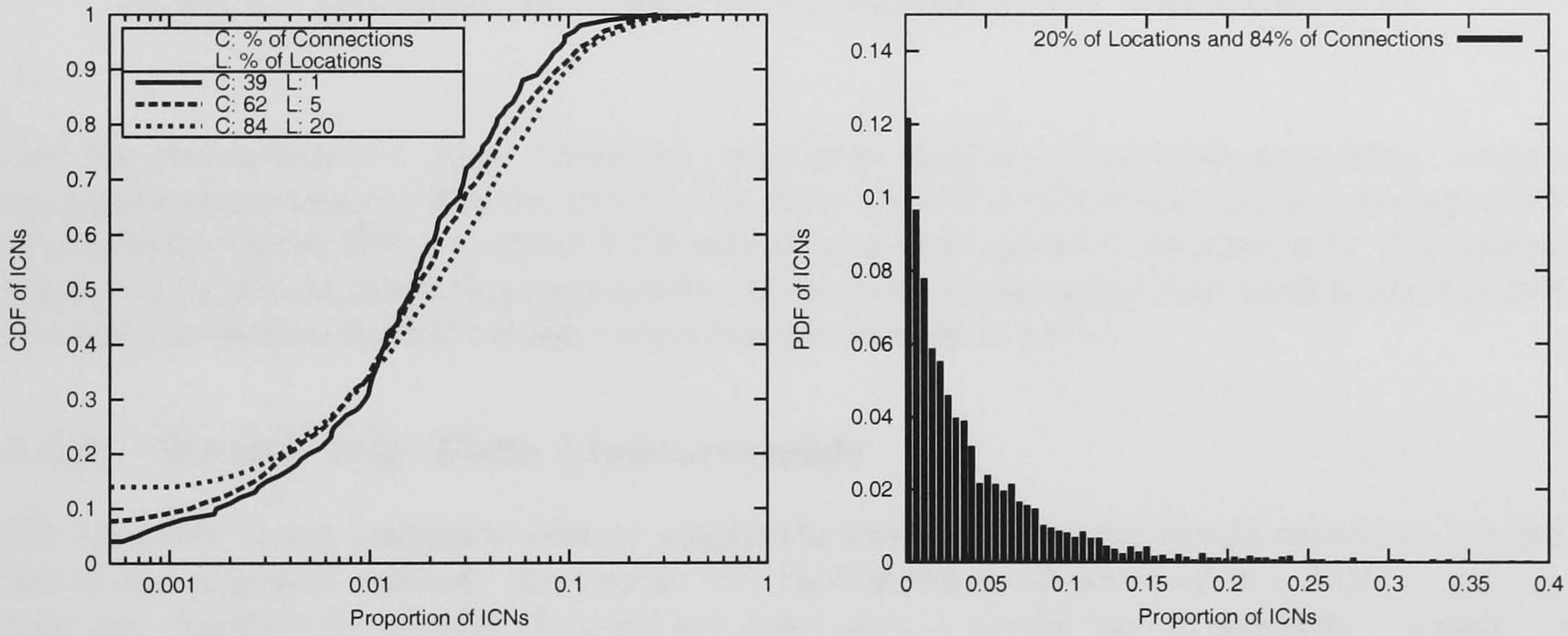


Figure 4.2: **PDF and CDF of ICNs by Location: The average proportion of Implicit Congestion Notifications to packets was calculated for each location. These graphs show the resulting probability distributions.**

connection the state of feedback during that connection will dominate, giving rise to the potential for atypical values that are likely to diverge from a true expected value. Using locations with only a small number of connections or packets may therefore bias the results, by flattening the Cumulative Distribution and exaggerating the number of locations that low and high values of PICN can be expected.

One solution is to only include locations, which have a certain amount of traffic. There is however, a danger in excluding a large proportion of connections, in that a bias is introduced into the results against locations with a high PICN as they can be expected to have a lower usage. Fortunately a large proportion of packets and connections are made with a relatively small proportion of Locations, as shown in Figure 4.1. Therefore it is possible both to include a large proportion of all packets and connections in the sample and ensure a reasonable number of samples for each location.

Figure 4.2 shows the cumulative and probability distributions for the average level of congestion by destination. The cumulative distribution curve approximates to a log-normal distribution. There are no obvious clusters of destinations with a common level of congestion. This suggests

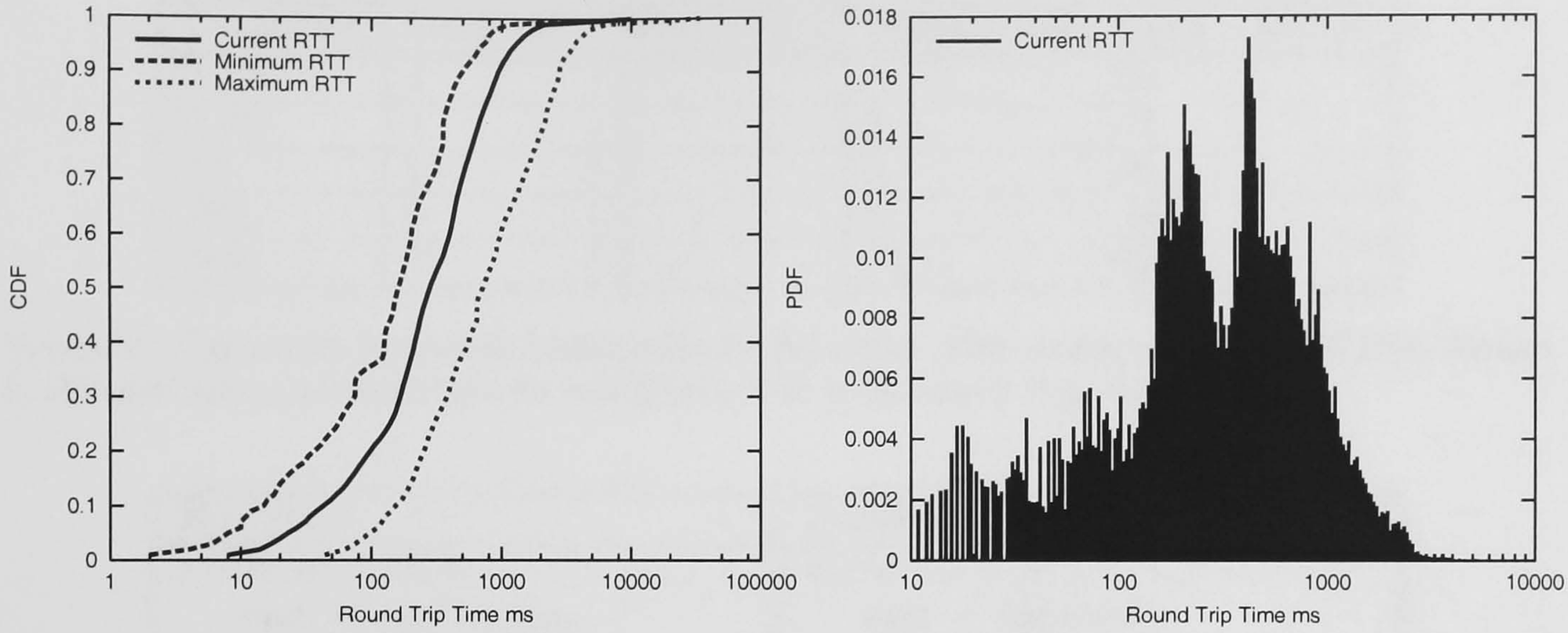


Figure 4.3: **Distributions of Smoothed, Maximum and Minimum RTTs.**

that the abstraction of a single bottleneck, with large numbers of destinations sharing common congestion characteristics does not fit with this data. Over 90% of destinations have average levels of congestion below 10% and above 0.1% indicating a wide spread of congestion by destination. There is a significant tail with a large number of locations experiencing very small levels of packet loss and the median level of average congestion per location is 1.6%.

4.2.3 Round Trip Time Measurements

For a window based congestion control scheme the level of congestion should determine the fair size of the congestion window, the amount of data that can be transmitted in a single round trip time and therefore the number of round trip times that it should take to complete a transfer of a certain size. To determine the delay between the start and finish of transmission it is therefore necessary to know the round trip time between the source and destinations.

Figure 4.3 shows the cumulative distribution and probability density for readings of the current RTT to locations, which has been calculated using the standard TCP low pass filter. The initial reading is used for the starting point of evolution rather than some arbitrary static initialisation. The cumulative distribution of location Maximum and Minimum RTTs are also shown in the left hand graph. These distributions demonstrate a wide variation in the RTTs for Wide Area Network Traffic, with over 90% of current RTT readings being between 10 ms and 1 second. There are significant peaks visible in the probability distribution, which correlate with UK, European and US destinations. South East Asian, Eastern European and African destinations are located progressively further out in the tail of the RTT distributions. This reflects the influence of geography and development in determining RTTs.

The key point to be drawn from this distribution is however that the majority of destinations are in the order of magnitude of RTT where when the number of rounds that a connection takes to complete is increased the lengthening of the delay will be clearly perceptible to a human observer.

4.2.4 Summary

It has been argued that an abstract model of TCP's congestion avoidance algorithm provides a good estimate of the limitation on throughput that is attributable to the network, where the network is defined as consisting of the physical infrastructure and competing traffic. Table 4.2

Congestion	Small 10K	Medium 26K	Medium 51K	Large 1.5Mbyte
0.01%	1	1	1	10
0.1%	1	1	2	31
1%	1	2	4	95
10%	2	6	11	302
20%	3	8	15	426

Table 4.2: **Expected Network Limitation in Rounds: The number of Round Trip Times it should take a connection to complete if it is network limited.**

Slow Start	Congestion Avoidance

`while (n > 0) {`
`  sent = cwnd / mss;`
`  last += sent*mss;`
`  n -= sent;`
`  r++;`
`  if ( last >= separation ) {`
`  cwnd = max(cwnd/2,mss);`
`  last -= separation;`
`  break;`
`  } else {`
`  cwnd += cwnd/ackspace;`
`  }`
`  cwnd = min(cwnd,maxwin);`
`} //end while`
	`while (n > 0) {`
	`  sent = cwnd/mss;`
	`  last += sent*mss;`
	`  n -= sent;`
	`  r++;`
	`  if ( last >= separation ) {`
	`  cwnd = max(cwnd/2,mss);`
	`  last -= separation;`
	`  break;`
	`  } else {`
	`  cwnd += mss/ackspace;`
	`}`
	`  cwnd = min(cwnd,maxwin);`
	`} // end while`

Table 4.3: **Slow Start and Congestion Avoidance: Core code for a deterministic simulation.**

gives the expected network component of delay, which covers the range of transfer sizes shown in 4.4 and for the levels of congestion found in Figure 4.2, an MSS value of 1460 is assumed. The figures shown are for the latency attributable to the network in RTTs. They can be scaled for a particular RTT by multiplying the figure in the table by the RTT, a distribution of RTTs for Glasgow University traffic has been presented in Figure 4.3..

Estimates of network limitation for a number of transfer types and congestion regimes have been provided. These figures suggest that for many network paths the portion of latency that is contributed by the network is considerable. This suggests that there are benefits in reducing this component of delay, by increasing bandwidth and reducing network queues. It also suggests that it is important that the transport protocol does not unnecessarily add to the latency attributable to the network.

4.3 Transport Protocol Limitation

In this section a method for visualising the interactions between network and protocol limitations is presented, which allows the extent to which TCP is able to utilise available network bandwidth to be determined. The effects of three factors are considered; that of congestion level, connection size and flow control window size. The analysis allows clarification of what can be achieved by addressing shortcomings in TCP's congestion control architecture. It was not possible to utilise existing models as Equation 4.1 does not capture some of the phenomena we are interested in, and

Padhye's and Cardwell's models conflate together the different elements that contribute to delay. Our aim is to separate them out in order to clarify their relative impacts for different parameter values.

In the previous section three factors were identified as contributing to protocol limitation; error recovery, start-up transients and window limitation. The interaction between congestion avoidance and error recovery is addressed by a number of efforts to reduce the occurrence of RTOs. Random Early Detection [FF97] gateways seek to control the length of queues and prevent packets from being dropped in bursts; this increases the likelihood of Fast Recovery being successful and an RTO being avoided. Selective Acknowledgments [MMFR96b] and New Reno [FH99] seek to enable TCP connections to recover from multiple packet losses without an RTO. Explicit Congestion Notification [RF99] seeks to replace packet loss as an indication of congestion with packet marking.

The two remaining factors Slow Start and (offered) Window Limitation are analysed across the dimensions of congestion regime and connection size. First the methodology for modelling TCP performance is described; this is then applied to produce an evaluation of the effect of static initialisation and Slow Start. Measurements of the distribution of connection sizes for web traffic are then presented and their implications discussed. This is followed by an evaluation of the effect of Window Limitation. Measurements of the distribution of offered window sizes and of distributions of utilised window sizes are then presented. The section is concluded by drawing together the dynamics of initialisation, Slow Start and Window Limitation with the measurements of web traffic to make statements about the significance of Protocol Limitation for the latency experienced by users and fairness.

4.3.1 Methodology

A program, which takes as input a grid of congestion probabilities and file size and traces the evolution of a connection's congestion window forms the basis of the analysis, the core of which is shown in Table 4.3. This program embodies the same assumptions as described in Section 4.2.1 with two important differences. The assumption that:

1. the connection is not application limited is maintained.
2. there is always data to send is maintained.
3. loss is deterministic is maintained.
4. loss is shared equally between concurrent connections is maintained.
5. network conditions remain stable for the duration of a connection is maintained.
6. the receiving application can keep up with the rate at which data is delivered is maintained.
7. the connection is not limited by the receivers window is relaxed. Simulations are carried out for both when this assumption holds and when it is relaxed. In this way it is possible to investigate the effect of receiver window limitation on the throughput achieved by window limitation, for a range of levels of loss and connection sizes.
8. connections are long lived is relaxed. In order to achieve this the simulator models the start up behaviour of TCP. The simulator therefore allows the effect of start up transients on the average behaviour of connections for the range of congestion levels and connection sizes under investigation.
9. Retransmit Time Outs will not be necessary to recover from loss is maintained.

The relaxation of assumption 7 allows the effect of TCP's flow control window to be evaluated. The relaxation of assumption 8 allows the effect of TCP's start up to be evaluated.

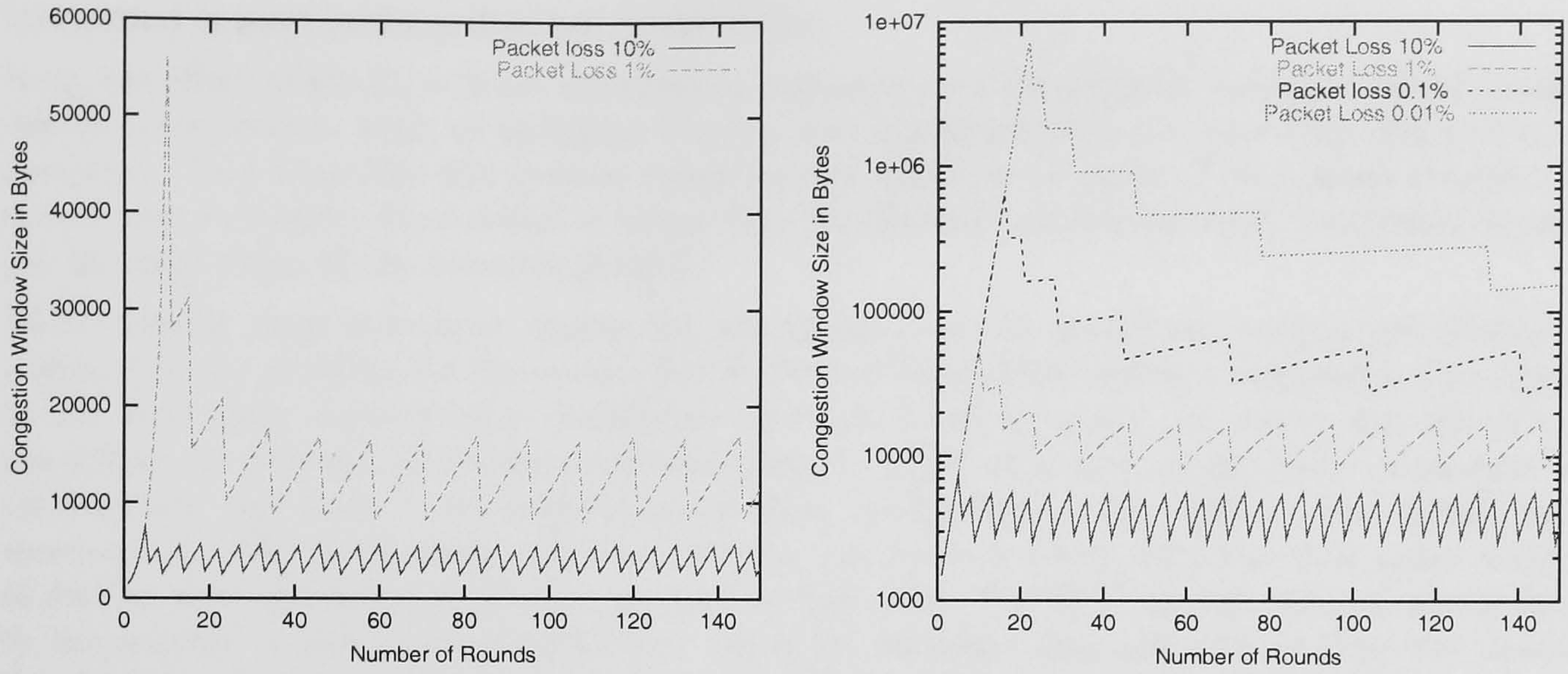


Figure 4.4: **Window Evolution: Traces of congestion window evolution for a range of congestion levels.**

The central loop of the program calculates the window evolution in a round, during which it is assumed that a full window of packets are sent and acknowledged. A number of variables are shown in Figure 4.3 `n` is the number of packets that remain to be sent. `cwnd` is the size of the congestion window maintained, `maxwin` is the maximum window size offered by the receiver, `ackspace` is the data packets to acknowledgments ratio, `separation` is the number of bytes between congestion events, `last` is the amount of data that can be sent between packet drops. The congestion window, maximum window and separation between congestion even4s are all maintained in bytes. In this program at the start of a connection the Slow Start period ends when the first loss occurs and the program moves into the Congestion Avoidance mode, which continues until all data has been transmitted and acknowledged. The rate of increase during both Slow Start and Congestion Avoidance is controlled by the `ackspace` parameter, which corresponds to the number of data packets that are required to arrive at a destination before an acknowledgment is generated.

As the evolution of the congestion window is traced count is kept of the number of packets remaining and the number of rounds that have passed. The window evolution can be controlled by setting a number of parameters. The maximum segment size and the maximum offered window size may be set to arbitrary values. This allows investigation of the effect of different maximum flow control windows. The amount of data and the gap between congestion events can also be set, these allow investigation of the interaction between Slow Start and Congestion Avoidance for a range of transfer sizes and levels of congestion.

4.3.2 Initialisation and Window Evolution

The left hand graph in Figure 4.4 shows the evolution of TCP's congestion window for loss rates of 1% and 10% and packet sizes of 1460 bytes. The right hand graph shows loss levels of 0.01%, 0.1%, 1% and 10% with the congestion window size set to a log scale. A number of features are of interest. Static initialisation means that there is a period at the start of each where the window size is below the steady state average. The exponential increase of Slow Start means that there is then a significant overshoot beyond this average, consequently for a significant period in a connection's life bandwidth utilisation is greater than its fair share. This phenomenon is insignificant at loss levels of around 10%, significant at 1% and extreme at 0.1% and lower. After the congestion avoidance phase has been entered there is a period of adjustment as the AIMD algorithm causes window size oscillations to move towards the steady state region of operation,

this period is most prolonged at low levels of loss.

Next, the effect of the Slow Start algorithm is evaluated by comparing the average utilised window size of a connection, with an optimum window size consistent with the Max-Min distribution of resources. In a Max-Min fair system resources are allocated in order of increasing demand, no source gets a resource share which is larger than its demand and sources with unsatisfied demand get an equal share of the resource [Kes97].

TCP's steady state behaviour (under the assumptions of our model and within the context of a shared path) provides for the equal distribution of bandwidth between competing connections in the case where sources have unsatisfied demands (data to send). A source can not receive more than its demand as resources network resources are allocated dynamically as packets are transmitted, and there is no resource reservation mechanism in the system. The allocation of resources in order of increasing demands implies that sources which have less than a fair window of data to send should be allowed to send all of their data. For TCP the fair window size is equal to the average steady state window size. Thus for transfers that are smaller than the average steady state window size the optimum average utilised window is equal to the size of the transfer. The optimum window size is therefore given as the minimum of the file size and the fair window size.

The ratio between the average utilised and optimum window size is a metric which can be used to determine the influence of Slow Start and Window limitation on a connection's average behaviour. A value of 1 indicates fair utilisation, the closer the ratio is to zero the greater the impediment TCP's congestion control is to the realisation of optimal bandwidth usage. If the ratio is greater than one this shows that TCP would use more than the fair share of bandwidth for the particular congestion level and transfer size. As the ratio indicates the extent to which a TCP connection is able to utilise its fair share of network resource we refer to it as the Resource Utilisation Ratio (RUR) or simply the ratio.

$$ResourceUtilisationRatio = \frac{AverageWindowSize}{min(TransferSize, AverageSteadyStateWindow)} \tag{4.3}$$

In discussing calculation of a fair window size it is useful to define a number of terms. An epoch contains a number of rounds and lasts from the first packet transmitted after a congestion event until the next congestion event, it contains one Local Minimum and one Local Maximum. A Local Maximum is the window size immediately prior to a congestion event and a Local Minimum the window size immediately after a congestion event. The Lower Bound is the line defined by interconnecting the local minima and the Upper Bound is the line created by interconnecting the local maxima. Steady State commences when the local minimum ceases decreasing and continues until the end of transmission.

The Fair Window size is taken as the average size of the congestion window during steady state. To calculate it accurately it is necessary to first identify where the Local Minima stop decreasing, this marks the start of steady state behaviour. It is also desirable to calculate over a sufficient number of epochs to enable the marginal effect of third-order harmonics to cancel out, 20 epochs are used in these calculations. Thirdly it is necessary to ensure that the average is calculated over complete epochs.

At the end of the transfer the average window size is obtained by dividing the size of the transfer by the number of rounds that it takes to complete.

Figures 4.5, 4.6 and 4.7 give different views of how the Resource Utilisation Ratio of a connection changes as a function of the proportion of packet loss and the size of the transfer.

Figure 4.5 plots the ratio against the number of segments for packet loss rates of 10%, 1%, 0.1%, 0.01% and 0.001%. The left hand graph shows the ratio to a linear scale and the right hand graph shows it to a log scale, this allows a greater range of congestion levels to be included on the same graph.

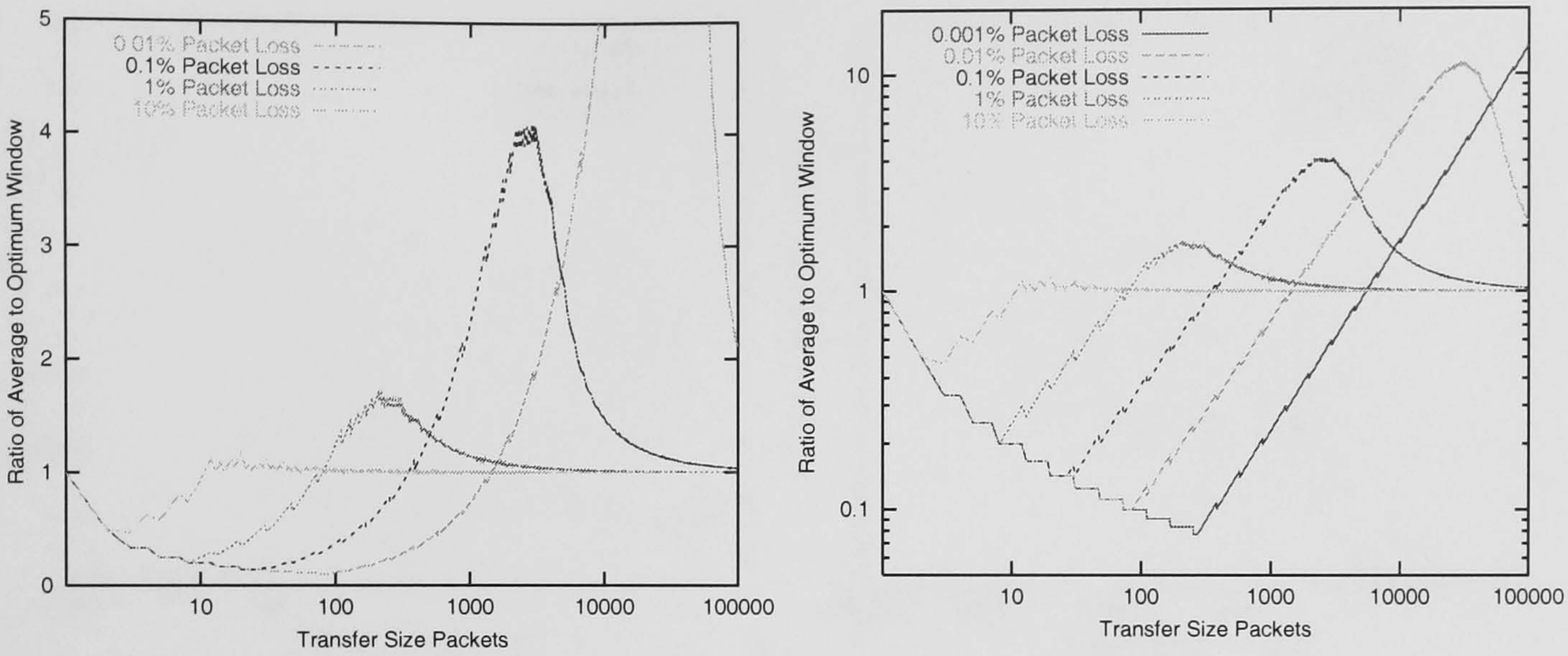


Figure 4.5: **Resource Utilisation Ratio against Transfer Size: These graphs show how close to fair utilisation TCP can be expected to approach, for transfers of 1 to 100,000 packets.**

It can be observed that for connections that can complete within a single segment a ratio of one is achieved. As the transfer size increases however the ratio declines exponentially. This reflects the static initialisation of Slow Start resulting in the connection taking multiple RTTs to complete, if the initial window size had been initialised to the average steady state window size data transfer would have been completed in one window. Once the file size becomes larger than the steady state window the number of rounds that it would take to complete increases, even with optimal initialisation, and therefore the ratio begins to climb. The exponential increase of Slow Start means that given enough data the congestion window will open beyond the steady state window average. There is therefore a rising slope which represents the average behaviour of TCP connections of those sizes being more aggressive than the steady state for a given congestion level. The peak occurs when the transfer is completed immediately before the switch to Congestion Avoidance would occur. For connections that last beyond the initial Slow Start stage, the average window size reduces until the oscillations center on the fair window size, the longer the transfer size allows the connection to remain in congestion avoidance, the more the average utilised window size will tend towards the steady state average window. This is reflected in the downward curve to the right of each the peak.

Comparing the curves for different levels of congestion it can be observed that as the level of loss decreases the depth of the trough increases. This is a consequence of the trough base being located at the point where the file size equals the steady state window size. At high levels of loss the peak is minimal but it increases in size significantly for small levels of loss. Reaching a peak of over four times the steady state bandwidth for 0.1% loss and over 10 times for 0.01% loss. The range of transfer sizes included in the peak widens and shifts to the right as congestion decreases. Thus at 1% loss transfers of between 100 and 1000 packets would be firmly within the overshoot peak. At 0.1% loss transfers requiring between 700 and 12000 packets would be within the overshoot peak. This in turn means that as loss increases larger transfer sizes are required for the average behaviour to approximate to steady state behaviour. At 10% loss average window sizes approach steady state for transfers of 10 packets and more, for 1% loss the size required is over 1000 packets and for 0.1% loss it is nearer to 100,000 packets.

Figure 4.6 plots the ratio against the probability of packet loss for transfers of 10, 100, 1000, 10000 and 100000 packets. The graph on the left has the ratio on a linear scale and the plot to the right has it on a log scale. For a given size of transfer, at high levels of congestion average and steady

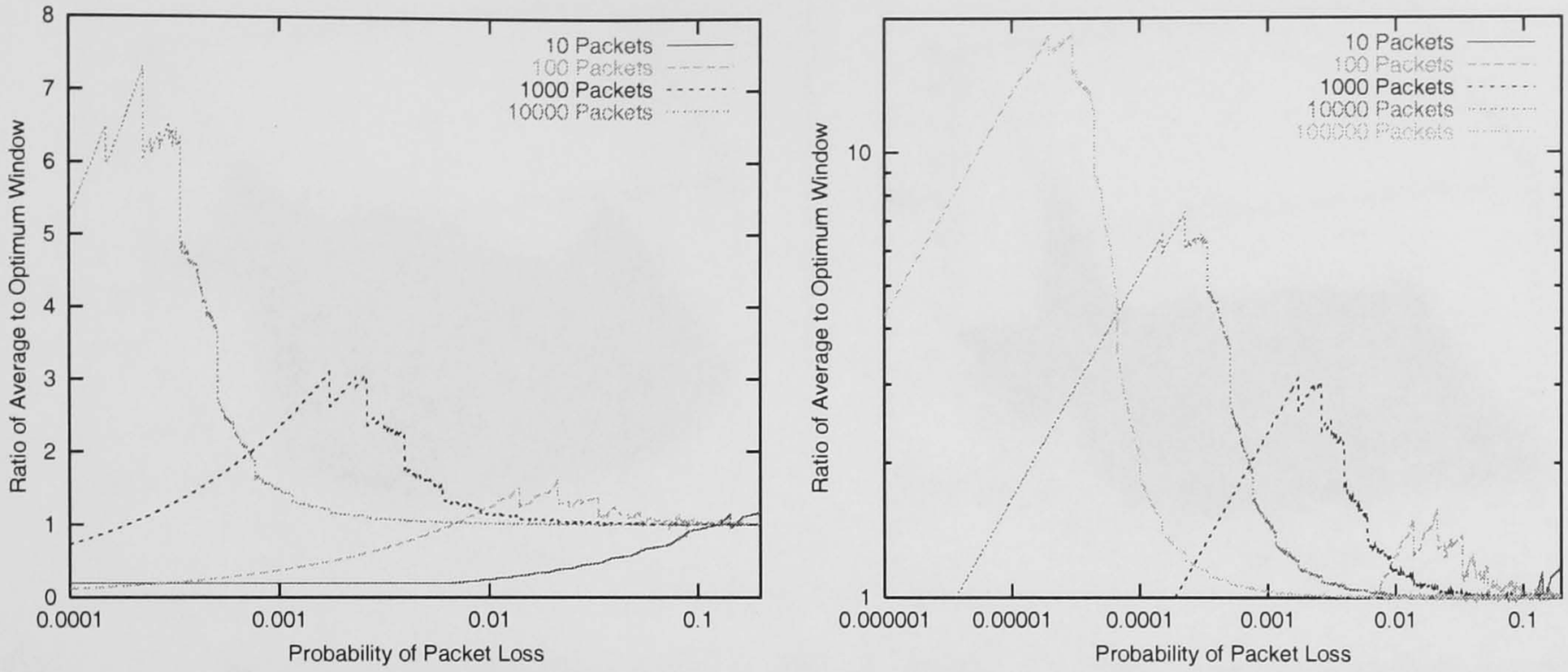


Figure 4.6: **Resource Utilisation Ratio against Loss Probability: These graphs show how close to fair utilisation TCP can be expected to approach for levels of packet loss between 10 and 0.1%.**

state behaviour are similar. As congestion decreases TCP's average behaviour becomes more aggressive in comparison to the steady state. At a certain point as congestion continues to decrease TCP's aggressiveness begins to drop off when there are insufficient packets for congestion window evolution to complete the Slow Start stage. The larger the transfer the lower the level of congestion at which aggressive behaviour commences and the higher the peak of that aggressiveness. For example for transfers of 100 packets departure from steady state occurs at around 1% packet loss and reaches a peak ratio of approximately 1.2, for 100,000 packets departure occurs around 0.1% packet loss and reaches a peak ratio of over 20.

Figure 4.7 provides a 3D plot which facilitates visualization of the regions within which TCP's average window size respectively approximates to, is larger than or is less than the previously defined optimum window size. The X-axis is the log of the transfer size in packets. The Y-axis is the log of the Fair Window size. The Z-axis is the ratio described above. The left hand graph plots congestion levels from 10% to 0.01% and transfer sizes from 1 to 1000 packets, which focuses this graph is dominated by the trough in performance where start-up transients prevent fair bandwidth utilisation. The right hand graph plots loss levels from 10% to 0.0001% and transfer sizes from 1 to 100000 packets. This graph illustrates the ridge of TCP's overshoot running from the top right hand corner of small transfers and high congestion to the bottom left hand corner of small congestion and large transfers. The height of the ridge increases exponentially from top right to bottom left. To the left of the ridge is the area within which TCP's average behaviour approximates to the steady state. To the left of the ridge TCP's average window size is smaller than the steady state average.

4.3.3 Measurements of Connection Sizes

In this section distributions of connection sizes and related measurements are presented and discussed with particular reference to the performance regions identified in Section 4.3.2. The quantity of data that is transferred in each connection was measured by recording the starting sequence number and subtracting it from the highest sequence number. The procedure was followed for all connections which transfered data, connections which failed to transfer data were not logged. A typical connection consisted of the exchange of the three way handshake, the receipt by the server of a small data packet, which corresponds to an HTTP get request, followed by the transfer of

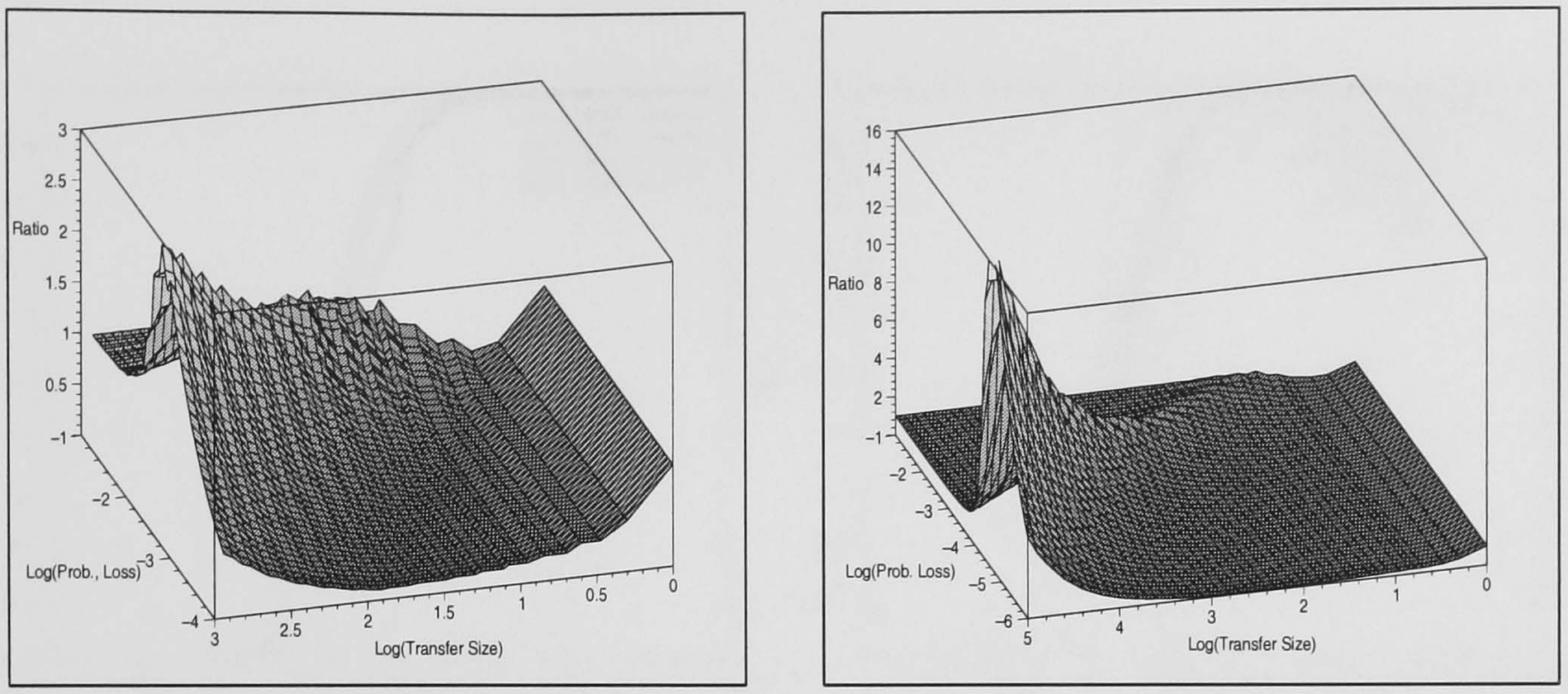


Figure 4.7: **Resource Utilisation Ratio: 3D Graphs showing approximation to fair resource utilisation for range of congestion levels and transfer sizes.**

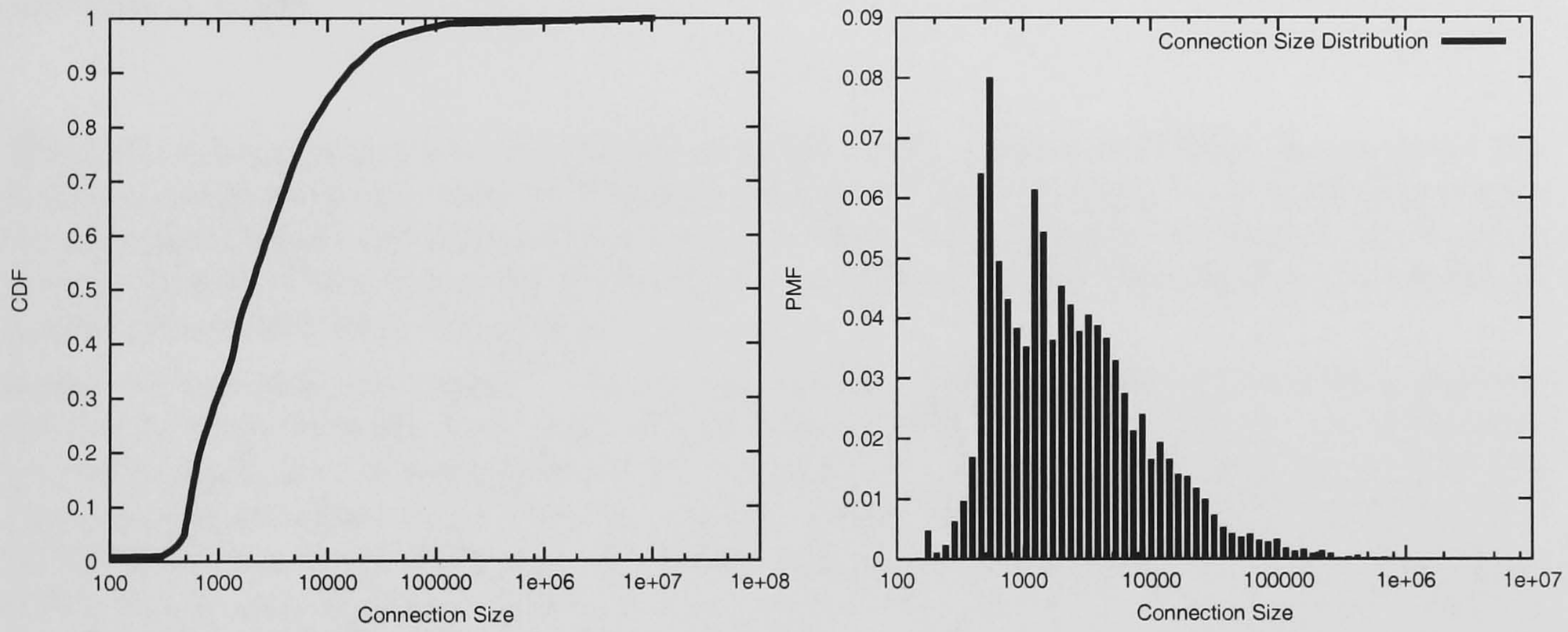


Figure 4.8: **CDF and PMF Connection Size, Log x-axis: show the distribution of connection sizes for WAN HTTP traffic with the University of Glasgow Scotland.**

requested data in a sequence of rounds.

The connection size data is discussed from three perspectives; firstly connection size distribution are considered directly, this enables statements to be made about the proportion of connections that will complete within a particular number of rounds which is important from the user perspective, secondly the proportion of data that is carried in connections of a certain size is considered which is important from the perspective of the network, thirdly quantiles of congestion levels are related to fair window sizes and the distribution of connection sizes.

Distribution of Connection Sizes

Figure 4.8 shows the cumulative distributions and probability masses for all HTTP web traffic. Connections which are smaller than 1460 bytes account for 42% of all connections but only 2.87% of all data. There is a high overhead for each of these connections as the three way handshake and four way connection close is carried out for all connections irrespective of size.

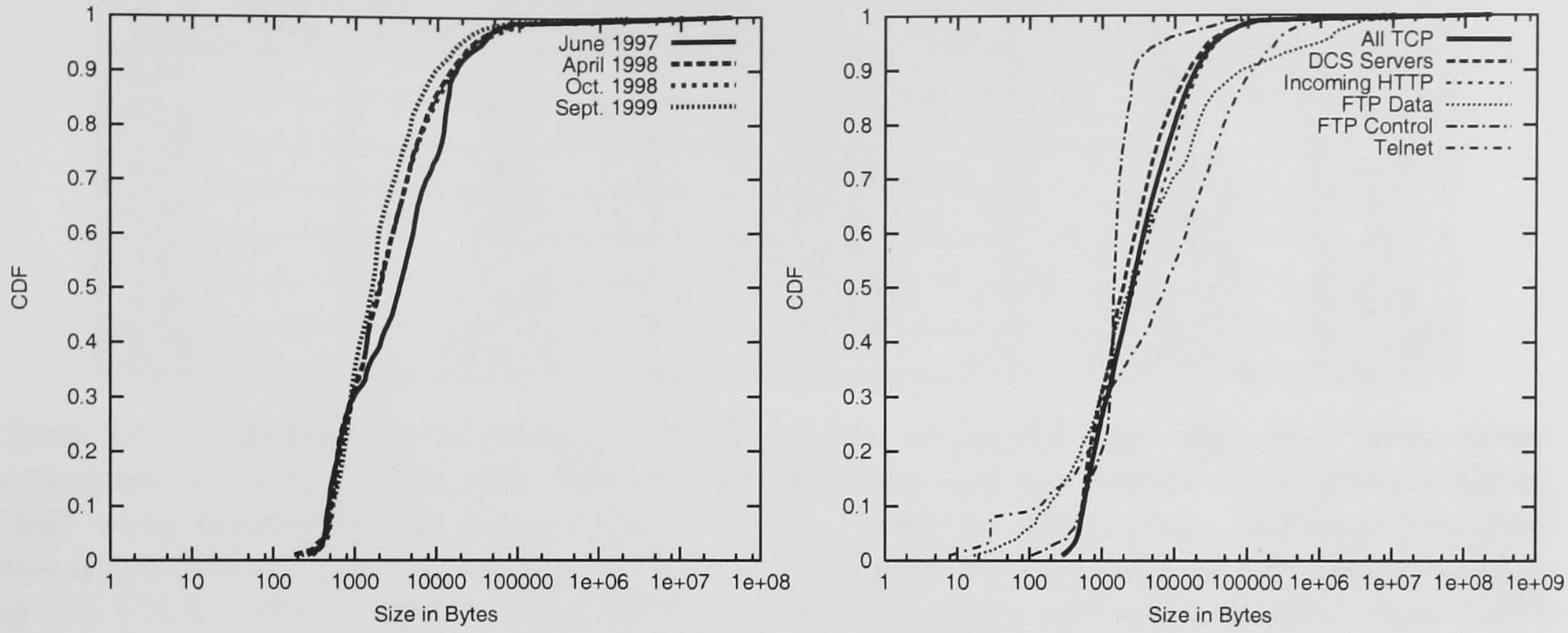


Figure 4.9: **Long Term Trends and Comparison with other Traffic: Changes in distribution of HTTP connection sizes between 1997 and comparison between HTTP, FTP and Telnet traffic.**

There are a large proportion of connections (47%), which require more than one segment but could be completed in less than 10 1460 byte packets. These connections have little opportunity to probe the network and therefore the extent to which they can adapt to network conditions is severely limited. There is a strong tail to the distribution indicating the existence of a significant number of relatively large connections.

Figure 4.9 contains two graphs. The one on the left shows the CDF of connection sizes for four traces taken between April 1997 and September 1999. They approximate to a log-normal distribution and show a slight tendency for connection sizes to get smaller over this period. The right hand graph shows CDFs from the October 1998 trace for; all traffic with the DCS, WAN HTTP traffic from DCS web servers, WAN HTTP traffic with DCS clients, FTP Data connections, FTP Control connections and Telnet. The comparison of outgoing and incoming HTTP traffic is significant; as incoming traffic came from a thousands of different servers, the closeness of fit with traffic from local web servers suggests that conclusions derived from the distribution of outgoing traffic has wider applicability.

To summarise, the Internet appears dominated by large numbers of short TCP connections for which connection confined adaptation seems badly suited.

Distribution of Data in Connections

In this section the way that data is distributed within connections is considered. This provides a different perspective on Internet traffic, from the distribution of connection sizes. The figures are presented in Table 4.4. The first column shows quantiles of connections and describes the second two columns so for example the first three columns of the first row describe the smallest one percent of connections. The second column shows the size of connection at the quantile, thus 50% of connections were 1,958 bytes or smaller. The third column shows the proportion of data carried in connections of the indicated size or smaller. Thus 52% of the data was carried in the 99% of connections that are equal to or smaller than 130,651 bytes. Columns four and five should be read together the fifth column gives the quantiles for the way that data is distributed within connections and the fourth column gives the connection size at each quantile level. So 90% of data was carried in connections that were smaller than or equal to 3,934,413 bytes.

The central contrast illustrated by this table is that although the mean connection size is only

Connections	Connection Size	Proportion of Data	Connection Size	Data
0.01	301	0.000187	709	0.01
0.05	486	0.001597	2,371	0.05
0.20	667	0.008941	12,662	0.20
0.50	1,958	0.040122	114,453	0.50
0.80	7,072	0.137894	2,272,222	0.80
0.90	15,549	0.228057	3,934,413	0.90
0.99	130,651	0.518746	10,782,101	0.99

Table 4.4: **Distribution of Data in Connections: Columns one, two and three show quantiles for connection size, the connection size and the proportion of data bytes that were transferred in connections smaller than the given size. Column five give the quantiles for the proportion of data and column four the size of connection which is the UB for the quantile. Thus fifty percent of connections were smaller than 1,958 bytes and these connections accounted for 4% of all data transfered. Fifty percent of the data was transfered in connections smaller than 114,453 bytes.**

slightly larger than a single segment almost half of the was carried in the largest 1% of connections. Consequently 50% of data was carried in connections larger than 114,453 bytes or 78 segments and a significant proportion in connections requiring thousands of segments.

The significance of this contrast is two fold. From a user's perspective the average connection size is most relevant, as each connection or set of connections will correspond to delivering some response to a user action. From the point of view of network stability the average type of connection that a data packet is likely to be carried in is more significant. At the network level there is no concept of connections and packets are treated atomically. Therefore, if, as a result of most packets belonging to large connections the flows correspond to steady state behaviour the transient behaviour that dominates small connections may have little impact on the stability of the network. In short the distribution of connection sizes is important from a users perspective and the distribution of data within connections is important from the perspective of the network.

Congestion Level, Connection Size and Time to complete

Table 4.5 links together the distributions of congestion level, connection and fair window sizes. Column one shows quantiles for congestion level, column two the proportion of ICNs for that quantile boundary. Column three shows the fair window size that corresponds to the congestion level and column four the percentage of connections that would complete within the first round of data transfer, if the initial window was set to the fair window size with the assumption that congestion level and connection size are independent.

These measurements suggest that most connections did not approach their fair share of bandwidth, when competing against TCP fair congestion control schemes or bulk transfers. In addition they were not able to utilise bandwidth even when it is unused. As a result the users experienced unnecessary increases in the delay for the display of web pages.

A large proportion (42%) of connections were too small to be affected by TCP's Slow Start and Congestion Avoidance algorithms as all data could have been transmitted in a single 1460 byte segment. For these connections measures such as; the widespread implementation and use of T/TCP would cut this overhead and the piggy backing of a GET request with the first SYN would respectively reduce the overhead of connection set up and increase the speed of response. Special measures for small messages similar to this were included in the ARPANET.

The trough is the second region of operation within which a large proportion of connections are located. For the traces measured (47%) of transfers would require more than one and up to ten

Quantile	Proportion of ICN	Fair Window Size	% Connections
1	0		
5	0.00069	41.19	97
10	0.0017	26.07	96
20	0.0050	15.33	92
50	0.016	8.42	86
20	0.043	5.19	80
10	0.073	4.00	76
5	0.091	3.51	70
1	0.16	2.7	60

Table 4.5: **Distribution of Congestion Notifications: Table links together the distributions of congestion level, connection and fair window sizes. Column one shows quantiles for congestion level, column two the proportion of ICNs for that quantile boundary. Column three shows the fair window size that corresponds to the congestion level and column four the percentage of connections that would complete within the first round of data transfer, if the initial window was set to the fair window size with the assumption that congestion level and connection size are independent.**

packets to complete, which for congestion levels of a few percentage points and less would be located firmly in the trough region.

The largest 1% of connections account for more than 50% of the data transfered. At loss rates of 1% approximately 1% of connections would have an average behaviour that approximates to TCP's steady state; at 0.1% loss this 99th quantile would be firmly located within the aggressive region of TCP's behaviour. Although this is only a small proportion of connections it accounts for more than 50% of all data packets in the study. Thus there is a danger that a significant proportion of TCP packets will be carried in connections whose behaviour is more aggressive than is warranted by TCP Friendly criteria; this danger increases as congestion levels decrease.

4.3.4 Flow Control Window Limitation

Both end points of a connection maintain send and receive buffers, the space available in the receive buffer is communicated in the TCP header's window field. This places an upper limit on the amount of new data that may be sent before the receipt of further acknowledgments and ensures there is sufficient space for packets to be received. The size of the receive buffer therefore places an upper limit on the amount of data that may be unacknowledged at any time.

The existence of a flow control window prevents the network from committing resources to transporting a packet only for those resources to be waisted when the packet is dropped by the receiving host due to lack of buffer space. There is a trade off in determining the size of a connection's buffers. If the buffers are too large then resources that may have been usefully utilised elsewhere within the end system will have been waisted. If the buffers are too small they may prevent the connection utilising bandwidth that would otherwise have been available to it. What then is an appropriate buffer size?

If the receive buffer is the size of the bandwidth-RTT product it will allow the pipe between two hosts to be filled and will not limit the throughput achieved or prevent network resources being utilised. In the situation where there is a high bandwidth path and a high level of multiplexing the maximum Fair Window size is likely to be smaller than the bandwidth delay product, thereby requiring less resources to be committed to buffers, but it is sufficiently large to prevent buffer allocation from acting as a barrier to network resource allocation. Thus an estimation of the maximum expected fair window size would be more useful than the bandwidth delay product for

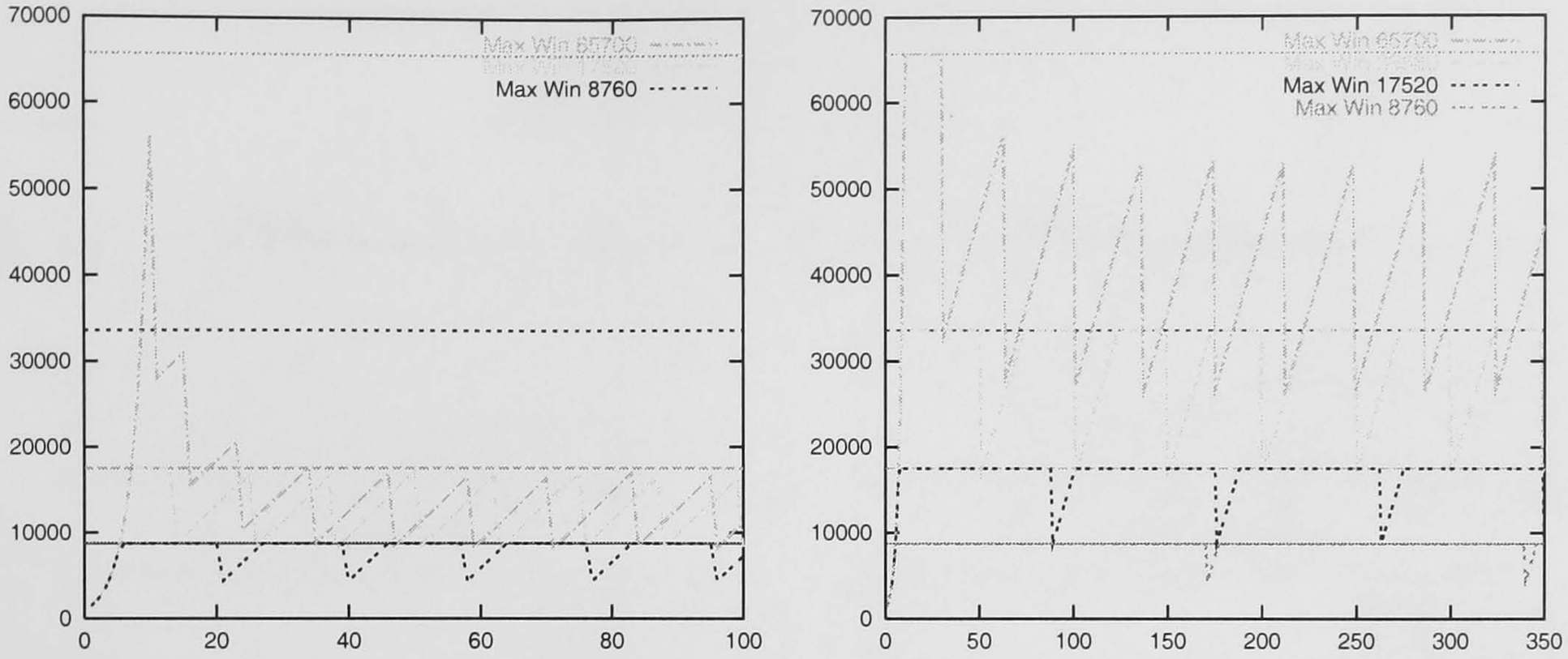


Figure 4.10: **Receiver Limited Window Evolution: shows the evolution of the congestion window for different sizes of flow control windows offered by the receiver.**

estimating estimating appropriate buffer sizes.

It would be undesirable to reduce the size of a buffer during a connection, as this would run counter to the design principle which states that implementations should be conservative in what they send but liberal in what they are prepared to receive, consequently mean connections moving the right edge of their flow control window leftwards, is discouraged by the RFC 793. Consequently the finest granularity at which buffer size should be determined is once per connection at the time of connection set up. Currently most buffer and therefore maximum window sizes are determined by the default values set when the operating system was distributed.

In the rest of this section the way in which TCP's congestion control and flow control mechanisms interact is discussed; first analysis is used to evaluate the effect different maximum window sizes have upon the Resource Utilisation Ratio, this is followed by the presentation and discussion of measurements of maximum advertised window sizes and the maximum utilised window, which enable judgments to be made about the prevalence of window limitation.

Window Limitation and Network Resource Utilisation

Figure 4.10 shows the effect of window limitation on the size of the utilised window during the lifetime of a connection. The left hand graph shows window evolution plots were the packet loss rate is 1% and where the maximum offered window is 8760, 17520 and 65700 KBytes. The right hand graph gives plots for a packet loss rate of 0.1% and window sizes of 8760, 17520, 33580 and 65700 KBytes. From the left hand graph it can be seen that at this level of loss a 65 KByte window does not effect effect the size of the utilised window, a 33 KByte window will limit the effect of Slow Start, a 17Byte window will effectively prevent Slow Start from overshooting the region of steady state oscillation and a window size of 8760 Kbytes will significantly interfere with the steady state window evolution thereby reducing the long term average window size. From the right hand graph it can be seen that at a packet loss rate of 0.1% all the window sizes shown prevent Slow Start from causing significant departure from the steady state region of window oscillation. The offered window sizes of 50 KBytes and below significantly reduce the long term average window size.

Figure 4.11 is similar to Figure 4.6 in that it illustrates the average behaviour of transfers ranging in size from 1 to 100,000 packets at specific levels of loss but differs in that it takes into account the effect of window limitations for maximum window sizes of reading from left to right and top to

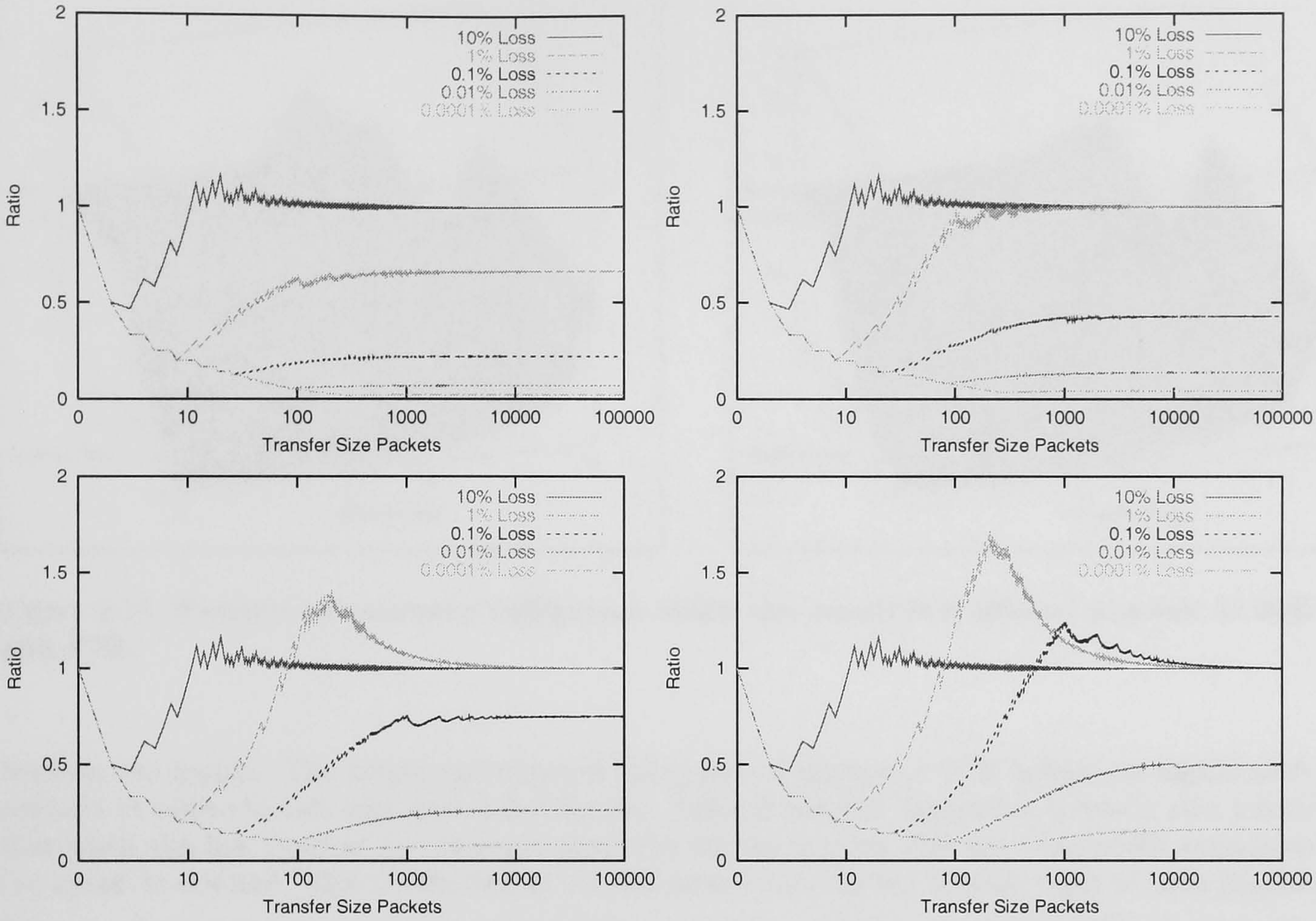


Figure 4.11: **Effect of Window Limitation on Resource Utilisation Ratio, with from top left to bottom right, maximum window sizes of top left 8 Kbytes, top right 17 KBytes, bottom left 33 KBytes, and bottom right 65 KBytes.**

bottom, 8, 17, 33 and 65. Comparing Figures 4.11 and 4.6 it can be seen that window limitation has no effect on small connections of a single packet and little effect on connections whose average behaviour positions them in the trough, with the exception that the climb out of the trough can be retarded and prematurely terminated. Window limitation can have the effect of reducing the range of transfer sizes that are effected by overshoot and reducing the size of the overshoot. Window limitation can also prevent long connections' average window size from approaching the Fair Window size and cause it to stabalise at lower ratio level.

For an 8 KByte window when congestion is at 10% window limitation has no noticeable effect. When congestion level is at 1% and lower the overshoot phase is prevented from occuring and the average behaviour of long connections is depressed. For a 16 KByte window the overshoot is still depressed for congestion levels of 1% and lower, but the steady state behaviour at 1% is not depressed. At 33KByte windows there is some overshoot at 1% congestion, although it is less severe and covers a smaller range of connection sizes when compared to non window limited regimes. When the maximum window size is set to 65 KBytes, the overshoot at 1% is similar to when there is no window limitation in scale and breadth. At 0.1% congestion there is some overshoot but it is severely retarded in scale and depth, and not only does the steady state behaviour of long connections approach fair bandwidth utilisation but the convergence occurs sooner than in the absence of window limitation.

Figure 4.12 provides a 3D visualisation of the effect of window limitation on TCP's bandwidth utilisation. In the left hand graph the maximum window size is set to 65K or 45 1460 byte segments, in the right hand graph the maximum segment size is set to 17K or 12 1460 byte segments. A comparison of these graphs with Figure 4.7 shows that for small transfers there is little difference

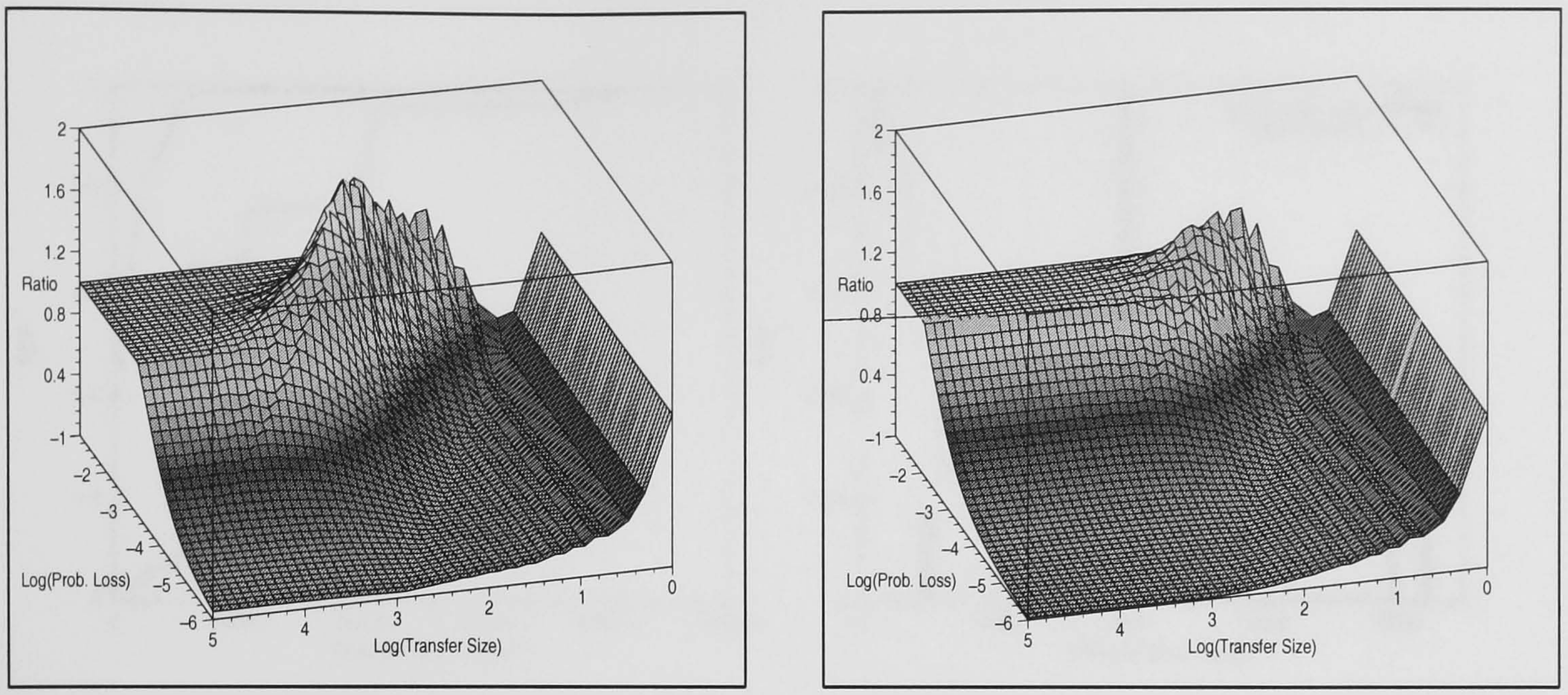


Figure 4.12: **Pattern of resource utilisation when the receiver's offered window is 65K and 17K.**

between the graphs. The height and scope of the region of aggressive TCP behaviour significantly reduced in both the left and right hand figures. Offered window limitation however also means that when the fair window size is larger than the offered window size fair bandwidth utilisation can never be reached. This is reflected in the downward curve in the bottom right of both graphs.

When congestion levels are greater than 0.1% a maximum window size of 65 KBytes, will offer some protection against the overshoot caused by Slow Start and will not prevent the average behaviour of long connections approximating to the Fair Window Size.

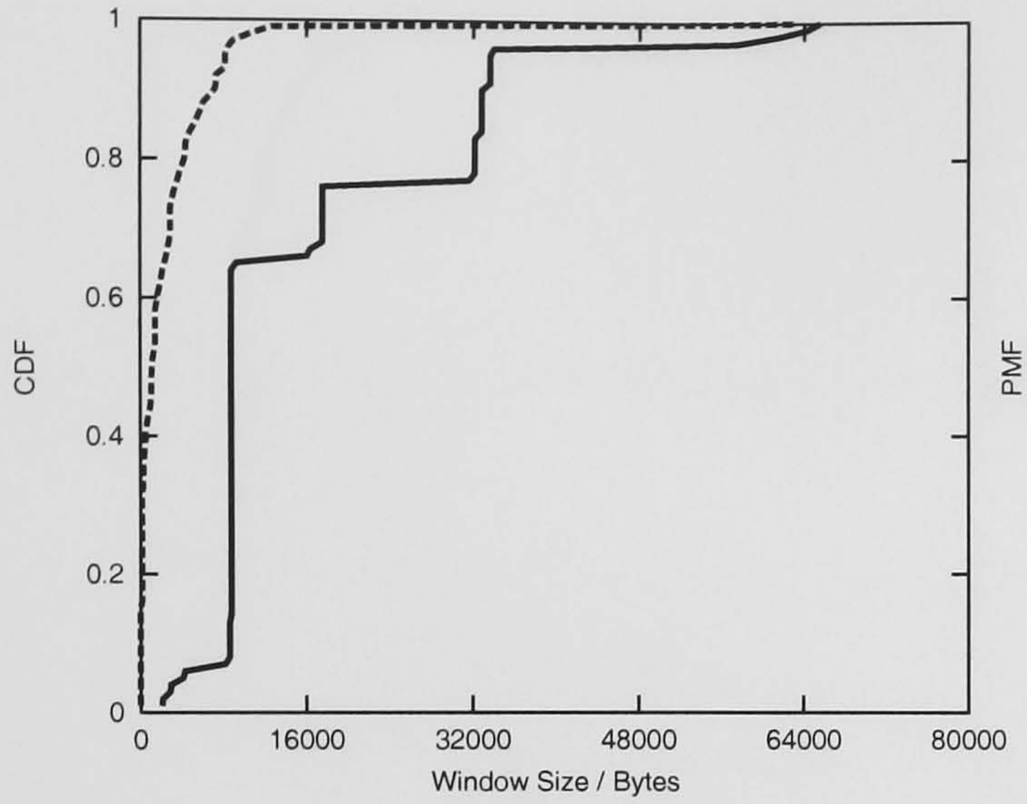
Window Limitation Measurements

The maximum offered window during the data transfer phase is recorded for each connection. This approximates to the size of the receivers buffer. At the start of each round the highest sequence number of the previous round is recorded, this is subtracted from the highest sequence number sent at the end of the round. The difference is taken as the size of window used. For each connection the maximum window used is logged.

Figure 4.13 displays two graphs. The first shows the Cumulative distribution for both the maximum used and maximum offered windows for all connections. The second shows a close up of the most significant features of the Probability Mass for the same connections and windows. The distributions are calculated at 1000 byte granularity.

A number of the graphs features are of interest. The large jumps in the offered window reflects the widespread use of default values for setting window sizes. For over 60% of the connections offered, windows of around 8K and smaller were offered. Two common window sizes ar included in the 8K bin; 8192 bytes and 8760 bytes. For 90% of connections this size of window does not limit throughput. The peak in used window size approaching 8000 and the sharp drop off after, indicates a significant number of connections were window limited. This finding is consistent with previous studies [BPS+98].

Simply considering the proportion of connections which are window limited at a particular window size can however seriously underestimate the significance of window limiting behaviour on the Internet. A large number of connections cannot reach a large window size because they do not have enough data to transfer. With delayed acknowledgments and a 1460 byte MSS a transfer would have to be at least 24820 bytes long for the congestion window to reach 8760 bytes.



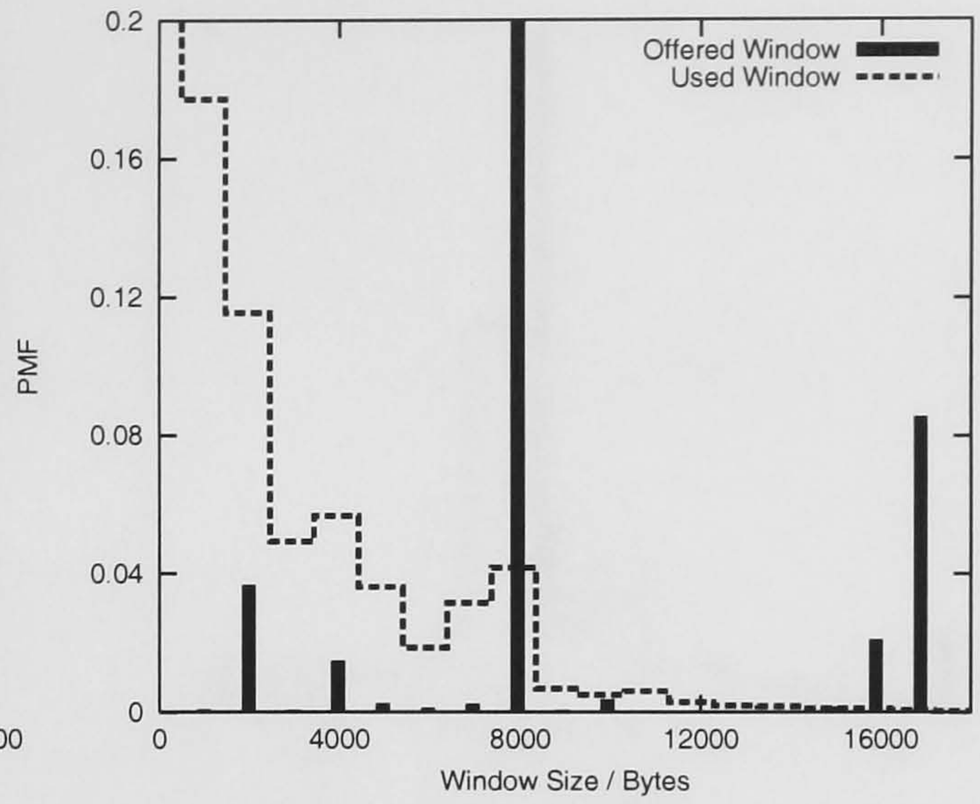


Figure 4.13: **Cumulative distributions of measured maximum offered and used window sizes.**

If connections which are long enough to become window limited are considered then a very different picture emerges. Figure 4.14 shows cumulative and probability distributions for transfers larger than 25,000 bytes. Here the peak in the vicinity of the 8K window is much more pronounced. Fully 78% of connections of this size would experience some degree of window limitations at this window size.

With current window sizes used on the Internet window limitation is a significant factor in caping the throughput achieved by connections. Considering all connections underestimates the significance of window limitation because , although the offered window size only limits the throughput achieved by a small proportion of connections, it caps a large proportion of connections that are larger than 25K-bytes.

4.4 Client and Server Limitation

In this section the delay attributable to network and protocol limitation is placed in context by looking at the contribution to the total latency of the client and the server. The approach taken here is the experimental measurement of transfers under controlled conditions.

4.4.1 Client Limitation

In this subsection the method used to make measurements related to client limitation are discussed. There are two senses in which client limitation is important. The first is the extent to which the rate at which the client is prepared to accept data limits the speed the server and network can deliver data. An easily detected symptom of the existence of client rate limitation is the closing of the offered window. The second is the absolute amount of time, in the absence of server and network limitation that the client takes to present data to the user. This is important because it places a limit on the reduction in latency that can be achieved by network and server side optimisations. We will call this absolute client limitation.

Absolute client limitation can be decomposed into two elements; one at the start of the connection (the resolve time) and one at the end of the connection (the presentation time).

Consider the time taken to resolve the URL and form the HTTP GET request at the start of the connection. To determine upper and lower bounds on the clients contribution to this delay four

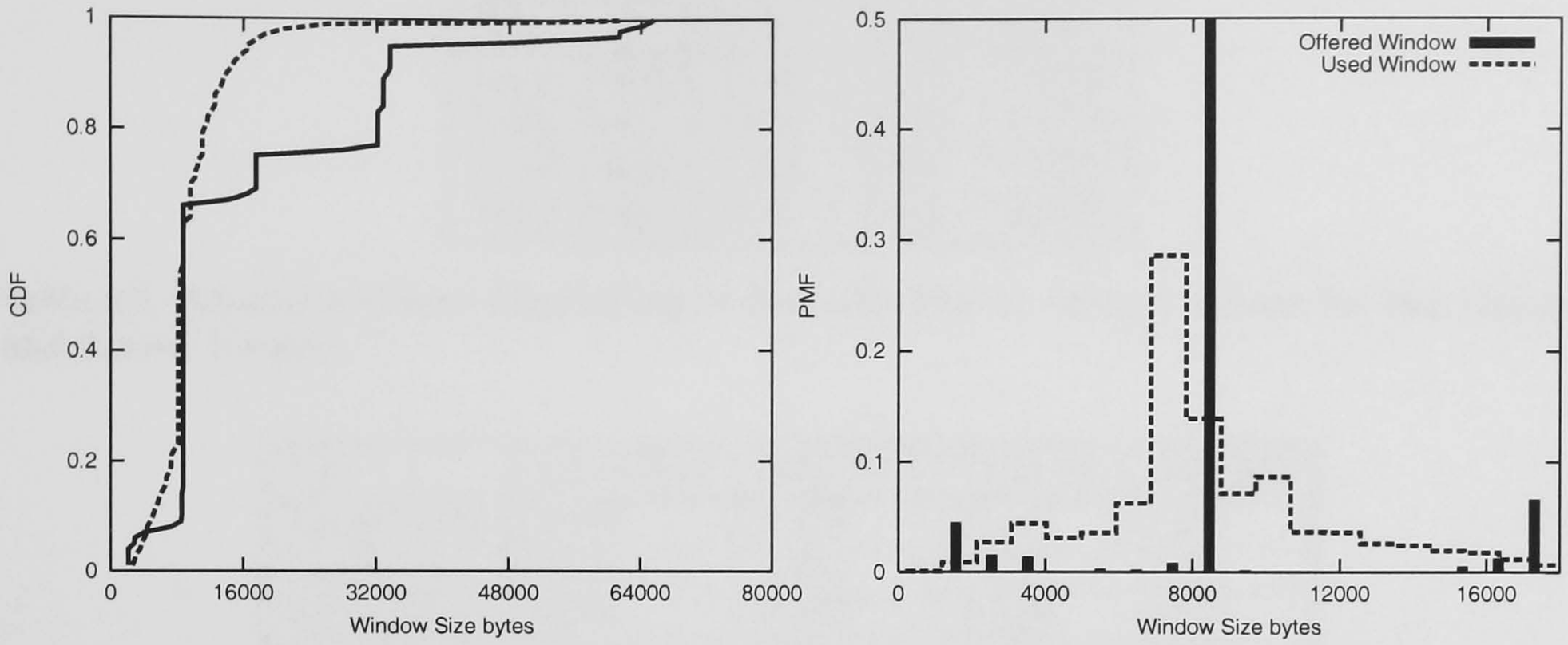


Figure 4.14: **Window Limitations for Large Files: Cumulative distributions of measured offered and used windows for large files.**

measurement points are required; the time the user clicks on the page (C), the transmission of the client's TCP SYN segment (S), the receipt of the server's SYN/ACK segment and the transmission of the HTTP GET request (G). A lower bound on the resolve time is given by:

$$ResolveTime = (S - C) + (G - A) \tag{4.4}$$

An upper bound on the resolve time is given by G-C, although this measurement includes network delay the request may have been being formed during this time. Next consider the presentation time. The upper bound is given by the time to display the web page from the arrival of the first data. The lower is the time required to display a page once all the data has been received.

The gap between the lower and upper bounds can be minimised by arranging for server and network limitation to be minimised. The was achieved by connecting the server and client to the same Ethernet switch and by generating the HTML for normally dynamically generated pages in advance.

A small harness web application that consists of a server script, which dynamically generates a multi-framed set of web pages and records measurements in a server side data repository was developed. Links to the page being measured are included in a presentation frame. When a user clicks on one of these links the system time is read using the JavaScript onClick() method and temporarily stored in a data frame. When the new page finishes loading the onLoad() JavaScript method is used to record the time in the data frame. These pairs of readings are periodically down-loaded to the server pending analysis. Taken together each pair defines the duration of a CURL. To obtain measurements of the checkpoints intermediary to the CURL's end points passive packet level monitoring of the client was utilised.

These techniques allow statements to be made about the client limitation, which occurs for specific web pages and specific client configurations. By carefully chosing the web pages measured and the client configurations it is possible to make general statements about the clients contribution to total latency.

The contribution of client limitation is in the order of 100s of milliseconds. As client limitation by itself contributes enough latency to be perceptible, but not so much as to drown out other sources it suggests a strong motivation for reducing protocol latency.

File	Lower	Upper	Conf
Large Table	0.108	0.136	0.0069
Large Text	0.096	0.124	0.0025
Small Table	0.080	0.090	0.0072
Small Form	0.099	0.113	0.0099

Table 4.6: **Absolute Client Limitation in Seconds Linux: Giving values for the Upper and Lower bounds.**

File	Linux N4.7	2000 N4.7	2000 N6	2000 IE5
Large Table	0.136	0.470	0.451	0.201
Large Text	0.124	0.060	0.220	0.070
Small Table	0.090	0.100	0.240	0.150
Small Form	0.113	0.381	0.350	0.100

Table 4.7: **Absolute Client Limitation in Seconds: for a selection of operating system and browser combinations.**

4.4.2 Server Limitation

As with client limitation there are two elements to server limitation; an absolute amount, which contributes to the latency of each connection and a rate, which places a bound on the speed with which data can be transferred. Absolute server limitation is the time taken for the server to process an HTTP GET request and start delivering data. This is the server's contribution to phase A in our structured timing model. It may be determined by reading packet level traces captured at the server and is the difference between the arrival time of the get request and the departure time of the of the first data packet.

Whether a connection is server limited can be verified by determining the relationship between the window size utilised during a connection and comparing it with the congestion and advertised window. If the utilised window is smaller than the minimum of the congestion window and the advertised window then at that point in the connections lifetime it can be said to be server limited.

The advertised window can be obtained directly from TCP packet headers. The evolution of TCP's congestion window can be calculated as TCP's slow start and congestion avoidance algorithms are well known and packet losses can be detected by the retransmission of dropped packets. Samples of the utilised window may be obtained by the following procedure. First a packet is selected; next the number of data bytes transmitted between the sending of the packet and the receipt of its acknowledgment are counted. This value is the utilised window. The traces of window sizes for dynamically generated web pages in Figure 6 illustrate server limitation.

Once the existence of server limitation is established it can be quantified. This was achieved by taking readings of the times the first and last data packets were transmitted for each connection. The size of each transfer is known so the rate can be calculated as quantity of data over the data transfer time. To measure the size of server limitation for a particular server, load and web page it is simplest to arrange that client, protocol and network limitation are likely to be smaller than the server limitation. This can be achieved by using a powerful lightly loaded client, in close proximity to the server, which a high bandwidth lightly used network path connects it to.

In this section it has been shown that it is possible to detect server limitation using passive monitoring of packet trances. Furthermore it has been argued that for statically generated HTML server limitation under reasonable loads is not significant. The extent of server limitation for dynamically generated web pages depends upon a number of factors which includes the technologies

used, the coding methodology and the load, but can be a significant factor.

4.5 Conclusion

In this chapter it has been argued that latency and fairness are critical metrics for determining the operation of transport protocols. It has been shown that whilst the network, client and server contributions to latency in many situations are considerable the operation of the transport protocol can unnecessarily increases the delay for time sensitive traffic.

Four regions of operation for TCP have been identified; small connection where only one packet is transmitted and congestion control has no affect, trough connections where static initialisation inhibits the average window size, aggressive connections, where the exponential increase of Slow Starts allows the average window size to be larger than the connection's fair share and steady state connections, which are sufficiently long lived for average window sizes to converge with steady state behaviour. It is interesting to note that the small and steady state regions correspond to the network conditions and traffic patterns against which Slow Start [Jac88] was validated and which were thought to be prevalent on the Internet when congestion control was introduced to TCP.

From the study of Glasgow University web traffic short connections accounted for 42% of all transfers. They are not affected by congestion control and only account for 2.7% of all data packets. The overhead of connection set up and tear down is however particularly high for these connections, the deployment T/TCP would considerably decrease this overhead.

A large proportion of Web connections fall into the trough category, where connections are small the static initialisation of TCP limits utilisation of the available bandwidth. For many connections average window size will be less than one fifth of the fair window size, which given the RTTs on many paths will introduce delays that are noticeable by users. Given the importance of latency in web browsing this leads to the conclusion that it is desirable to design mechanisms that allow short connections to approach fair window sizes and thereby eliminate this unnecessary contribution to latency.

When transfer sizes are large and congestion is low the exponential increase of Slow Start will make TCP overly aggressive. Thus for bulk transfers which are less insensitive to delay a bandwidth share greater than that implied by fair distribution may be obtained. Although only a small proportion of Internet connections are large enough for this phenomenon to manifest itself they account for a disproportionate proportion of data packets and may have serious consequences for congestion on the Internet in the future.

Fortunately this phenomenon is suppressed by TCP's flow control window. The maximum sizes of the windows offered by receivers plays a critical role in capping the aggressiveness of TCP's congestion control algorithms. It is important to recognise this dependency as it means that the widespread use of larger flow control windows, facilitated by the ever falling cost of memory, could have significant consequences for Internet congestion control. It would therefore be beneficial to address the aggressiveness of TCP and thereby remove the dependency upon flow control mechanisms in curbing congestion.

The interaction between TCP's flow control and congestion control has been discussed. It has been shown that only a small proportion of connections are effected by window limitation, even when the window is as small as 8K. This is largely because most connections do not transfer enough data for the congestion window to become larger than the offered window. If longer connections are considered a different picture emerges. For long connections, with an 8K offered window up to 80% can be expected to be window limited. Thus there is a region of operation, where congestion is low and transfer size large where window limitation governs the allocation of network resources.

A consequence of the importance of connection length and window limitation in determining a connections average window size is to reduce the responsiveness of TCP to congestion notifications. There is only a small proportion of traffic on the Internet, which is governed by TCP's AIMD

algorithms, most connections will be governed by static initialisation, exponential increase and flow control window limitation. The key distortion caused by connection confined adaptation on the Internet today is that TCP's congestion control algorithms result in overly aggressive bandwidth utilisation for bulk transfers which are typically insensitive to delay and penalise short connections which are often interactive and therefore sensitive to delay.

In a future Internet however, where users have high bandwidth access, where the network is well engineered and congestion is low and where music, video and software is routinely down loaded the problems of connection confined adaptation will manifest themselves in other ways. There will be an incentive to increase flow control windows to speed bulk transfers, combined with low congestion fair bandwidth overshoot by a factor of 12 or more will be probable. This may result in bandwidth being stolen from streaming applications that are designed to be TCP friendly or the interactive graphical traffic that has insufficient time to adapt.

These problem are rooted in the confinement of traffic management within the lifetime of individual connections. In short, connections do not have time to adapt to network conditions. In order to increase the sensitivity to network conditions ways of broadening the geographic and temporal scope of congestion control need to be found. Connections should start transmission at a fair bandwidth and be able to adapt if network conditions change. The next chapter derives a design which aims to increase the responsiveness of TCP to congestion notifications[1].

[1]It has been argued that each connection is important from a users perspective, as it maps to one or more elements in the interface that the web browser presents. For the IP network it is not the number of connections, but the number of packets that is important. Each packet requires forwarding and routing decisions. From this perspective the Internet is dominated by a small number of large connections. This is significant, because the start-up transients of TCP have less effect on the average behaviour of large connections. This suggests that the Internet would be resilient to different start-up regimes.

Chapter 5

Addressing Protocol Limitation

5.1 Introduction

The goal of this chapter is to motivate and present an approach to best effort congestion control at the session time scale which addresses three deficiencies identified within TCP, namely: the extra delay for small connections caused by static initialisation to a conservative Initial Window size, the overshoot in window size caused by the exponential increase of Slow Start and the dependence on the flow control window for regulating the effect of TCP's congestion control algorithms.

The first section discusses aspects of connection confined adaptation; the advantages of intelligent initialisation are demonstrated using an experimentally adapted TCP implementation, measurements of Web traffic are presented which shed light on the limits of extending TCP's congestion control from intra connection to intra session granularity, and there is a discussion of traffic management time scales. The second section presents a functional decomposition of congestion control and applies it to today's Internet to derive an architecture which will support inter-session congestion control.

The design and implementation of a Location Information Server (LIS), which extracts Quality of Service (QoS) feedback from TCP packet headers and facilitates the sharing of such feedback between end hosts is then presented. The way in which communication between such a server and end hosts can be efficiently supported is discussed. An end host TCP implementation is presented, which is capable of utilising the LIS supplied QoS feedback.

5.2 Addressing the Limitations of Connection Confined Adaptation

TCP's control variables, such as the congestion window and round trip time estimate, are initialised to default values with the assumption that they will converge to values that are appropriate to the operating environment at the start of a connection. They will then adapt during the lifetime of the connection as network conditions change. In Chapter 4 it was shown that the average length of connection often precludes effective adaptation, making the initial value important in determining performance. It was also shown that the average level of congestion and network round trip may vary by several orders of magnitude for different paths, making it difficult to find good default initial values.

This section discusses the issues that arise in trying to address the limitations imposed by connection confined adaptation. First some methodological issues are discussed including the introduction of an experimental TCP stack used to explore issues relating to initialisation. One proposal for

reducing delay is to simply increase the size of the Initial Window for all connections, the merit of such an approach is evaluated before using experimental traffic to demonstrate the control that can be achieved by initialising the Slow Start Threshold (`ssthresh`) and Congestion Window (`cwnd`) to the same value, effectively by passing Slow Start.

The advantage of basing such initialisation of round trip estimates and the congestion window on past traffic is then illustrated, by comparing the behaviour of multiple short serial connections with a single long connection and introducing the possibility of using the state at the end of one short connection to initialise the next. The limitation of such an approach is explored by returning to the traces of web traffic at the University of Glasgow. The section is concluded by discussing the range of time scales over which traffic management mechanisms may be applied.

5.2.1 An Experimental TCP Implementation

The Net/3 implementation of TCP/IP contained within the FreeBSD 2.2.7 kernel was modified to provide control over TCP start-up dynamics. In particular three facilities were added; the ability to set TCP variables to arbitrary values, for state to be saved at the end of a connection and for that state to be utilised in the initiation of new connections.

The state associated with a TCP connection is stored in the Transport Control Block. This state can be divided into two categories, local process state and per connection state: the local process state binds the connection to its socket and the IP layer; the per connection state contains the variables required internally by TCP and can be further divided into macro and micro state. Macro state describes the TCP state machine, whilst the micro state consists of the variables required to manage the data flow across a TCP connection.

The variables `rtt`, `srtt`, `cwnd`, `ssthresh`, `vrtt` and `mss` are all part of the micro state, and are all derived from the characteristics of the path between the two hosts involved in the connection. A logical location for a common repository of information relating to the path between two hosts is the kernel's routing table, this approach is utilised by the Net/3 implementation of TCP/IP. The experimental implementation developed here refines these facilities, to allow the congestion window to be initialised through the sockets interface and to allow the saving of state variables to occur at an appropriate point in the lifetime of a connection thereby facilitating the use of such values for the initialisation of new connections.

A number of changes to the Net/3 implementation of TCP were made to allow the congestion window to be saved. A new member was added to the rt_metrics structure, which is attached to the routing table entry. Tcp_mss was edited to allow the initialise of `cwnd` if there was a non zero value in the rt_metrics structure and the new function tcp_update added to update the rt_metrics structure with `cwnd` if it is not locked and has a positive value.

There is a problem with the point in a connection's life that Net/3 saves values to the routing table. When an active close is done the routing table is only updated after the connection exits the TIME_WAIT State. Upon entry of the TIME_WAIT State the 2MSL timer is set to sixty seconds, when it goes off tcp_close will be called and the connection will move from TIME_WAIT to CLOSED.

The tcp_close function performs two tasks. It frees up memory resources and updates the routing table. The sixty-second delay between the last transfer of data and updating the routing table causes three problems. Information that may have been of value to other connections has not been made available, when the update of the routing table does take place network conditions may have changed considerable, and thirdly it may overwrite more recent information from a connection that did a passive close.

Changing the point at which the routing table is updated when an active close is done, can solve the above problems. However tcp_close is called at the correct time to free memory resources, therefore tcp_close has been split into two functions; one function frees resources and a second

called tcp_update updates the routing table.

When an Ack of the FIN has been received all the data has been transferred from the host and all the ACKs returned. This is therefore a reasonable time to update the routing table as `cwnd` and `ssthresh` have reached their final values. The receipt of a FIN ACK causes one of the following transitions, from FIN_WAIT_1 to CLOSING or to FIN_WAIT_2 for an active close and from LAST_ACK to CLOSED for a passive close. The routing table is now updated by calling tcp_update from tcp_output when the specified transitions occur.

Thus modifications to the socket layer, routing table and several TCP functions were made in the FreeBSD kernel to provide facilities required for experimentation with the initialisation of TCP connections.

5.2.2 Changing the Default Initialisation

A simple method of reducing the depth of the performance trough at start-up is to use a larger default Initial Window size. For example if the Initial Window was set to four segments, for the Glasgow University study 76% of connections would be able to complete within the first window of data transfer as compared to 42% with an initial window size of one. With any default Initial Window size however, for some paths it will be too large and for others it will be too small. For the Glasgow University data over 80% of locations have an average level of congestion which would support a fair window size greater than five segments and for over 9% of locations four would on average be too large.

There is a further problem with increasing the default Initial Window size which becomes clear when the evolution of the congestion window for longer connections is considered. Whilst increasing the size of all initial windows will have the effect of reducing the trough of underutilisation it will also increase the spread and height of the overshoot caused by Slow Start for longer connections. Consequently, it will also increase the prevailance of reliance on window limitation in controlling this overshoot.

Thus changing the default size of the Initial Window, reduces the problem of underutilisation for short connections but at the cost of increasing the aggressiveness of longer connections. This further suggests that the problem lies with static initialisation itself rather than with the particular default value chosen.

5.2.3 Bypassing Slow Start

Next, the control that can be exercised over the behaviour of a TCP connection, by setting the initial values of `cwnd` and `ssthresh`, is demonstrated. A number of long transfers were made from Glasgow DCS to the University of Pennsylvania with varying starting sizes for `ssthresh` and `cwnd` and with `cwnd` equal to `ssthresh` in each case. In effect this means a connection bypasses the initial SS phase of network probing and goes straight into Congestion Avoidance. These were compared with the default TCP behaviour of Reno TCP.

Figure 5.1 is a selection of those transfers where the underlying network conditions for each connection are the same. This was determined by measuring the network RTT for each connection and comparing only connections with the same average, to an accuracy of 10ms, and a small difference between the minimum and maximum RTTs. Only connections without loss were compared. The graphs are therefore applicable to two classes of transfer. For the case when the offered window is less than the fair window size and for the case when the size of transfer means that the fair window size is never reached. When loss occurs the behaviour of the different configurations will be the same for a given window size.

Two graphs are presented, the left is a sequence plot of the transfers. The right hand graph plots the time at which an acknowledgment was received for a particular sequence number over the time

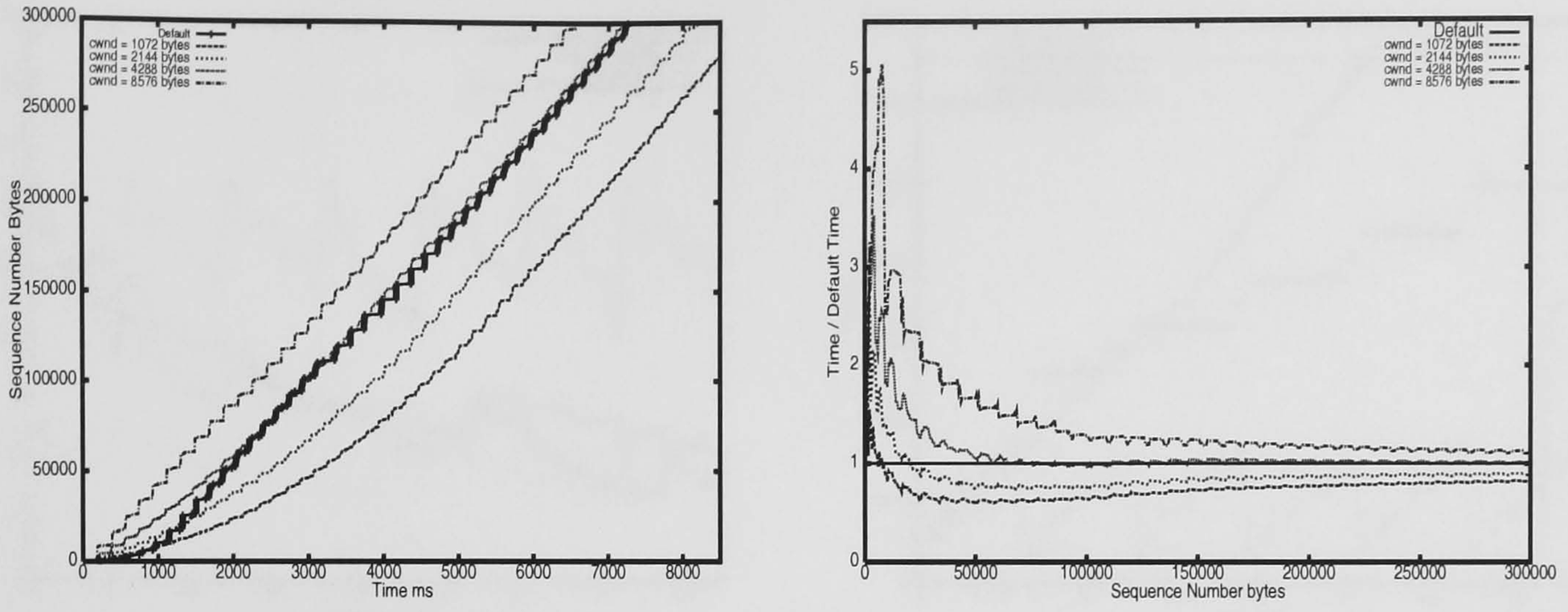


Figure 5.1: **Window and Slow Start Threshold Sizes: These graphs illustrate changes in TCP behaviour when the initial window size is set to a value larger than one packet and Slow Start is bypassed. Two graphs are presented, the left is a sequence plot and the right shows the throughput relative to Reno TCP (transfer size is shown on the y axis.**

an acknowledgment was received for the same sequence number of the default configuration. It shows the throughput relative to Reno TCP, that would be achieved for a connection the same size as the sequence number plotted.

These graphs demonstrate three things. Setting the congestion window and `ssthresh` to appropriate values can result in behaviour that is less or more aggressive than the normal TCP start up, thus if the correct values can be determined then behaviour that is appropriate to the current network conditions can be achieved. The peak in the difference in behaviour occurs when the congestion window is set to the size of the offered window and the transfer contains one window worth of data. This peak corrects the divergence from the TCP Fair equations observed in normal TCP behaviour. In the long term there is a convergence between the behaviour of all connections largely due to the effect of the flow control window. As the majority of packets on the Internet belong to long connections, this demonstrates the limited effect of different start up regimes on Internet stability.

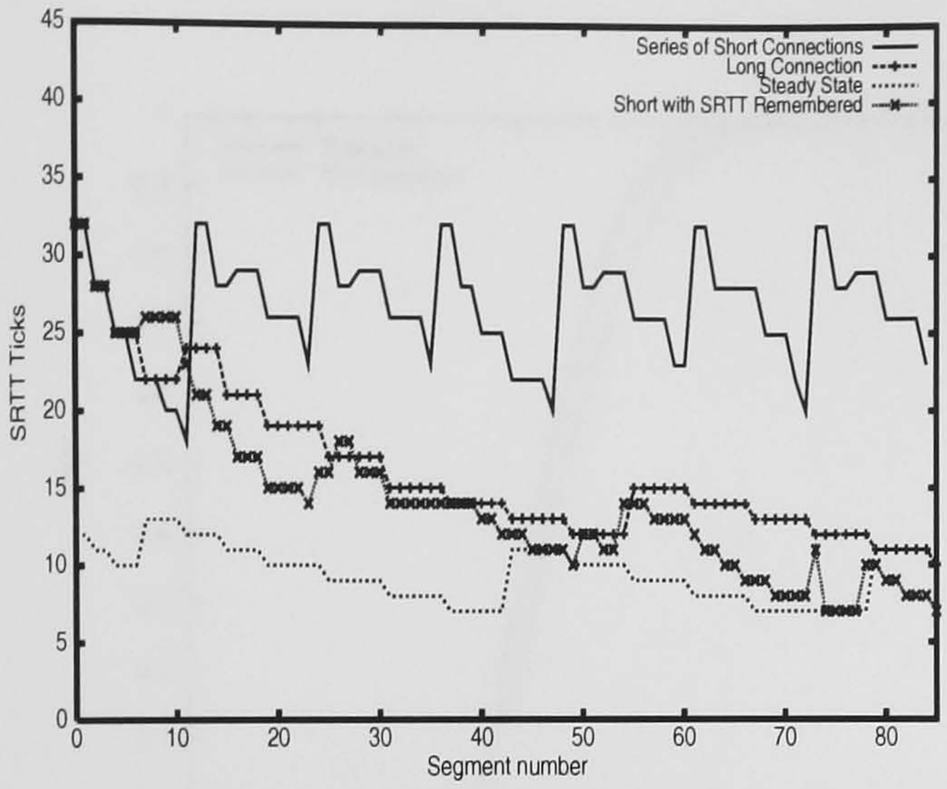
5.2.4 Saving State between Multiple Serial Connections

Figure 5.2, illustrates the evolution of `srtt` and `cwnd` for the cases where values are remembered between short connections and where static initialisation to a default value is used for all connections.

The left hand graph shows the evolution of `srtt` for various connections. Four series of connections are depicted. There are two long connections one shows the `srtt` estimate homing in on the true round trip time, the other shows the steady state behaviour of oscillation around the true RTT. There are also two series of short connections. In the first `srtt` is reset to a default value at the start of each connection. In the second the value of `srtt` is carried over from one connection to the next.

It can be observed that static initialisation prevents the value of `srtt` from approaching a true RTT for a series of short connections, but that carrying over `rtt` between connections allows adaption to take place, as though the series of short connections was one long connection.

The right hand graph shows the evolution of the congestion window for three sets of connections. There is one long connection and two sets of short connections. For one set of short connections the congestion window is reset to one at the start of each connection, for the other set `cwnd` is



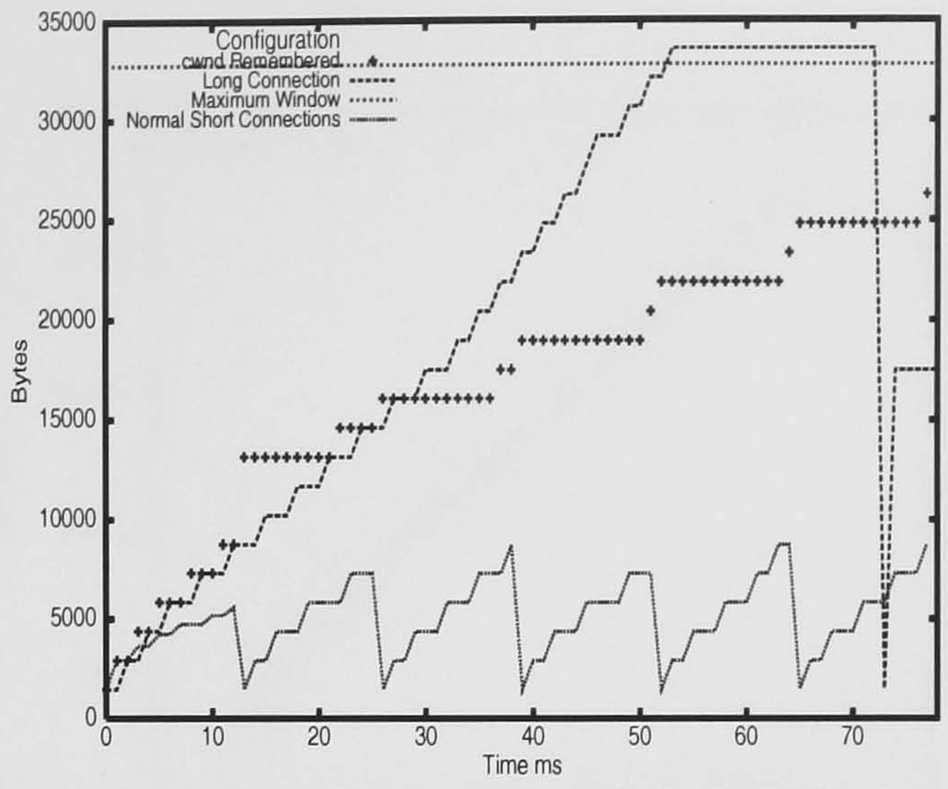


Figure 5.2: **RTT and Congestion Window Evolution: The left hand graph shows the differences in the evolution of** `srtt` for long and short connections. The right hand graph shows the evolution of the congestion window for three sets of connections. There is one long connection and two sets of short connections, one where the congestion window is set to one at the star, and one where `cwnd` is carried over from one connection to the next.

carried over from one connection to the next. Remembering the congestion window between connections allows the congestion window to evolve during a series of short connections in a way which approximates the evolution of the congestion window during one long connection[1].

This section has shown that setting the initial value of the congestion window and the smoothed RTT, to a value that is derived from past traffic, allows a series of short connections to approach the behaviour of a single long connection. It can be concluded that carrying over variables between connections, illustrates the advantages that accrue from information sharing, and demonstrates that the critical time to share information is at the start of a connection.

While the initialisation of control variables to values that are derived from past traffic can allow adaptation to network conditions to be more effective, there are problems with simply remembering the values of control variables and utilising them for the next connection. In particular if there is a large delay between connections the instantaneous window size at the end of a connection may no longer be valid [MJS99]. Consequently the degree to which connections cluster together into sessions and the separation between such sessions are important in determining the effectiveness of such an approach.

5.2.5 Measurements of Web Sessions

The application of congestion control to groups of connections that form a session may form a way of overcoming the difficulties caused by the small size of most Internet connections. This sub section discusses the potential for reducing the effects of start-up transients on TCP behaviour by extending the scope of TCP's congestion control algorithms from a connection to a session of concurrent sessions. In so doing two questions are addressed; firstly would extending the scope significantly increase the volume of data over which TCP's algorithms operate and is the gap between sessions sufficiently small to justify the carrying over of congestion window sizes from one session to the next.

[1]There are two important differences. The absence of an ACK clock means that, when there is a large initial congestion window a correspondingly large burst of packets may be transmitted at once. This has two effects. The congestion window opens more slowly for the series of short connections than for the long connection, because only one acknowledgment is received for each burst of packets and the peak load on the network is increased because a full window of packets is transmitted at the start of each connection.

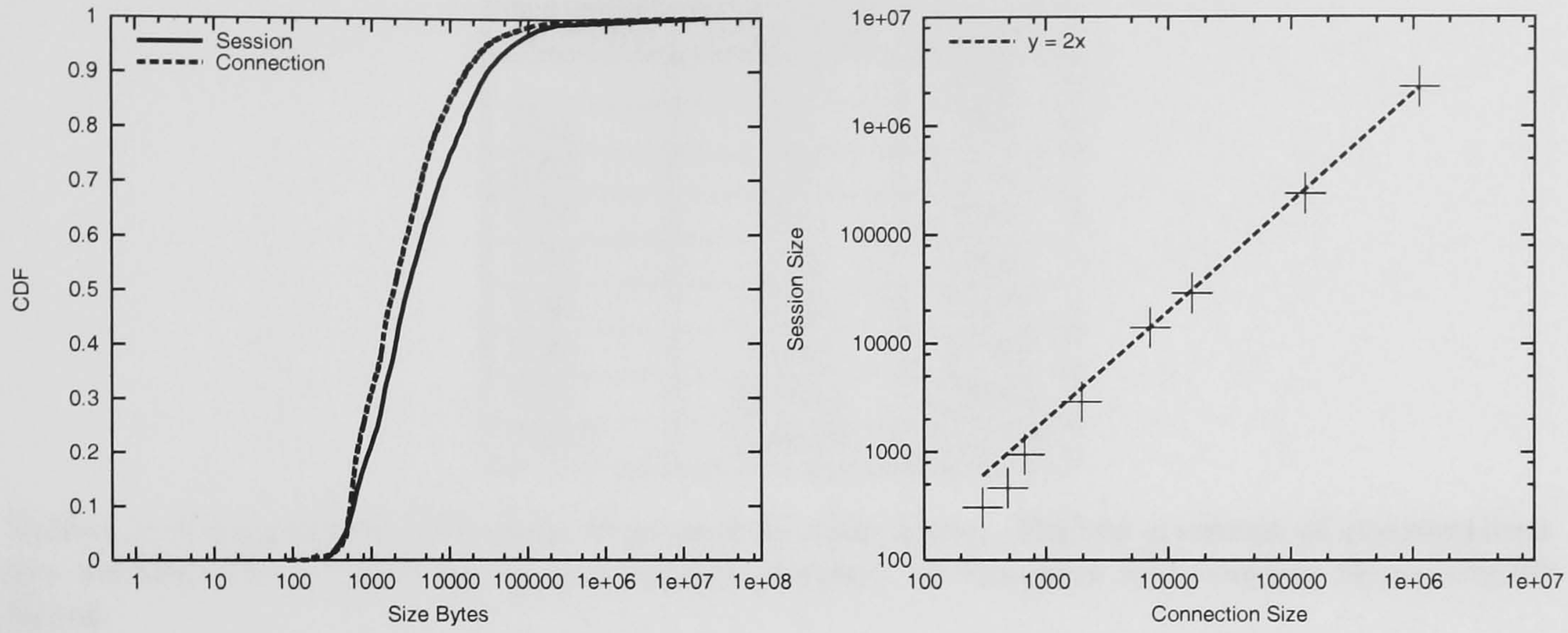


Figure 5.3: **Comparison of Session and Connection Sizes: If a session is defined as existing between the host network and a group of hosts whilst there is at least one connection that is open and in the data transfer stage, then these measurements suggest a session can be expected to contain approximately twice as much data as a connection.**

A session is defined as existing between the host network and a group of hosts whilst there is at least one connection that is open and in the data transfer stage, which is in turn defined as starting when the first data packet is sent and ending when the server's FIN is acknowledged or a RESET sent. A number of alternatives exist for defining a session. From a users perspective a session may correspond to the series of web pages down loaded from a single location at one sitting. All the connections that are required to down load a single page might also be considered a session. These three types of a session might be considered to correspond to the network, user and server perspectives respectively.

At the end of each session the current RTTs the elapsed time between the start of the present and previous session (inter-session arrival time), the duration of the session, the time between the end of the previous session and the start of the current session (inter-session time) and the number of data bytes sent and received in each session are all logged. The inter-session and inter-arrival times are for sessions from the web servers shown in Figure A.1 and groups of hosts sharing the same higher order three bytes in their IP address. Measurements of sessions drawn from the October 1998 Glasgow University web trace are presented and their implications discussed.

Intra-Session Congestion Control

Extending the scope of TCP congestion control from the connection to the session potentially has two advantages. Firstly, the congestion control algorithms operate over a larger number of packets, giving a greater opportunity for adaptation to network conditions and thereby reducing the impact of start up transients and increasing the significance of steady state behaviour. Secondly, the granularity of fairness is shifted from the connection to the session, so an unfair advantage can no longer be gained by a client opening multiple consecutive connections.

A comparison of the distribution of connection and session sizes for the Glasgow University Web traces, makes it possible to determine the impact of such an extension. The left hand graph in Figure 5.3 shows the cumulative distributions of connection size and session size and the right hand graph shows a quantile quantile plot of connection size against session size, with a line which corresponds to sessions being twice the size of connections. Table 5.1 shows the sizes for connection

Quantile	Connection	Session
	4556	8717
0.01	301	306
0.05	486	464
0.20	667	946
0.50	1958	2904
0.80	7072	14072
0.90	15549	29160
0.99	130651	239248
0.999	1122990	2268599

Table 5.1: **Comparison of Connection and Session Sizes: Eighty percent of connections are smaller than 7,000 bytes and eighty percent of Sessions are smaller than 14,000 bytes**

and sessions at different quantile levels.

Forty two percent of connections could have been completed in a single data packet compared to the 30% of sessions which were smaller than 1460 bytes in size. Reading the quantile plot from bottom left to top right the gap between sessions and connections sizes starts off small and widens in the higher quantiles. The small difference at lower quantiles in part reflects the number of sessions that are made up of only one connection. At higher quantiles the gap increases until at around the 60% the ratio between connection and session sizes stabilises to around 2.

These figures suggest that shifting the scope of congestion control from intra-connection to intra-session granularity would significantly increase the number of packets over which the congestion control algorithms operate. Most sessions would however still be dominated by the start-up rather than the steady state behaviour.

Inter-Session Congestion Control

The scope of TCP's congestion control algorithms could be further extended by carrying state over from one session to the next. The effectiveness of this approach depends upon the time-scale over which the congestion window remains valid and the separation between sessions.

One proposal [MJS99] for determining the validity of the congestion window is to assume that the congestion window should decay by half for each round trip time of inactivity. This conservative approach is justified on the grounds that in the presence of congestion `cwnd` would be halved each round trip time, and that connections in the Slow Start phase may double their window size each RTT. Alternatives strategies are possible but this is a mainstream proposal and so the boundaries of what it can achieve are worth examining.

Table 5.2 shows the number of round trip times between sessions, Figure 5.4 gives the distributions in milli seconds of the inter-session gap and the inter-session arrival times. From Table 5.2 it can be seen that about 13% of sessions would be able to continue at the old level. Whilst a further 20% are separated by between one and 10 RTTs and would therefore benefit in some way from a decayed congestion window.

Figure 5.4 shows that 78% of sessions are separated by less than 100 seconds. If it could be shown that network conditions were relatively stable over this time span, then an alternative to the decay policy could be formulated and used. Figure 5.4 also has a heavy tail, with a second group of peaks starting at around 24 hours. This indicates a diurnal pattern in web usage.

A distinction can be drawn between pauses between requests for new pages and those caused by delays between components of a page being down loaded. The first peak of the time between

Gap in RTTs	Number of Sessions	Proportion of Sessions
0 - 1	18916	0.128
1 - 2	15670	0.106
2 - 3	6384	0.043
3 - 4	4258	0.029
4 - 5	3619	0.024
5 - 6	3151	0.021
6 - 7	2942	0.020
7 - 8	2789	0.019
8 - 9	2515	0.017
9 - 10	2119	0.014
More	85490	0.578
Total	147853	1

Table 5.2: **Gap Between Sessions in RTTs: The gap between over 50% of consecutive sessions to the same location is greater than ten RTTs.**

sessions distribution occurs at around 100 ms, this time span is too short to reflect user requests, we therefore allocate it to the time between sessions, which together would make up a web page. This class of inter-session gap would benefit from a policy of exponential window decay, adding an estimated 26-32% more aggregation.

Next, the session inter-arrival time is considered. Here an estimation of the distribution of requests to the network are made. All inter-arrival times which have a gap between sessions that is less than three round trip times are removed, on the grounds that these are likely to represent pauses introduced by browser TCP interaction rather than separate requests.

The resulting distribution is bimodal with peeks at around five and thirty seconds. There is a strong tail, with further peaks starting at the 24 hour interval. 84% percent of sessions are separated by a gap of less than 10 minutes. The gap in requests are of the order of at least seconds, orders of magnitude larger than round trip times.

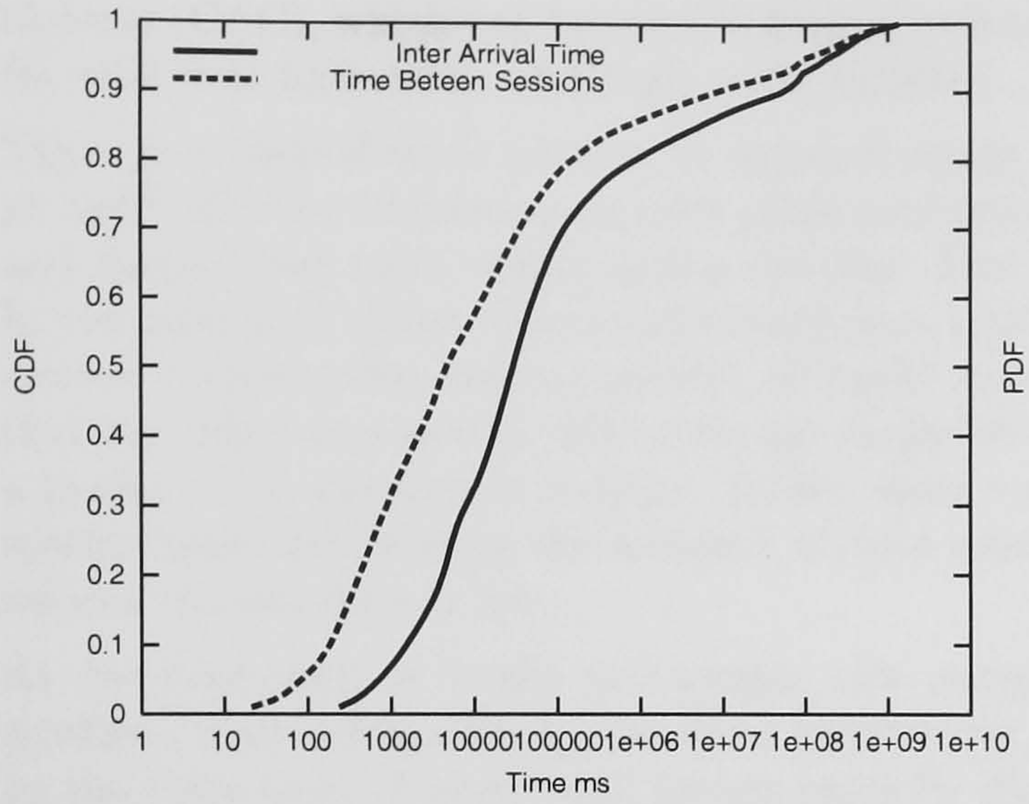
Summary

The benefits of applying congestion control to sessions as compared to connections are twofold.

1. A common congestion window between concurrent connections, would prevent applications stealing more bandwidth by opening multiple connections and controls the bursts that can be caused by multiple connections simultaneously exponentially opening their congestion windows.

2. The penalty paid by a transfer as it ramps up to its fair share of bandwidth is amortized over a larger volume of data.

The measurements presented in this section suggest that whilst these benefits are real, their effect is often likely to be marginal, because sessions are on average only larger than connections by a factor of two. The volume of data over which start-up is amortized can be further increased if information about network state obtained during one session is used to set the start-up rate of the next session. The effectiveness of such a strategy depends crucially on the time between sessions and the stability of network conditions. Measurements have been presented which suggest that halving the transfer rate for each RTT separating sessions would leave a large proportion of sessions having to probe the network from scratch.



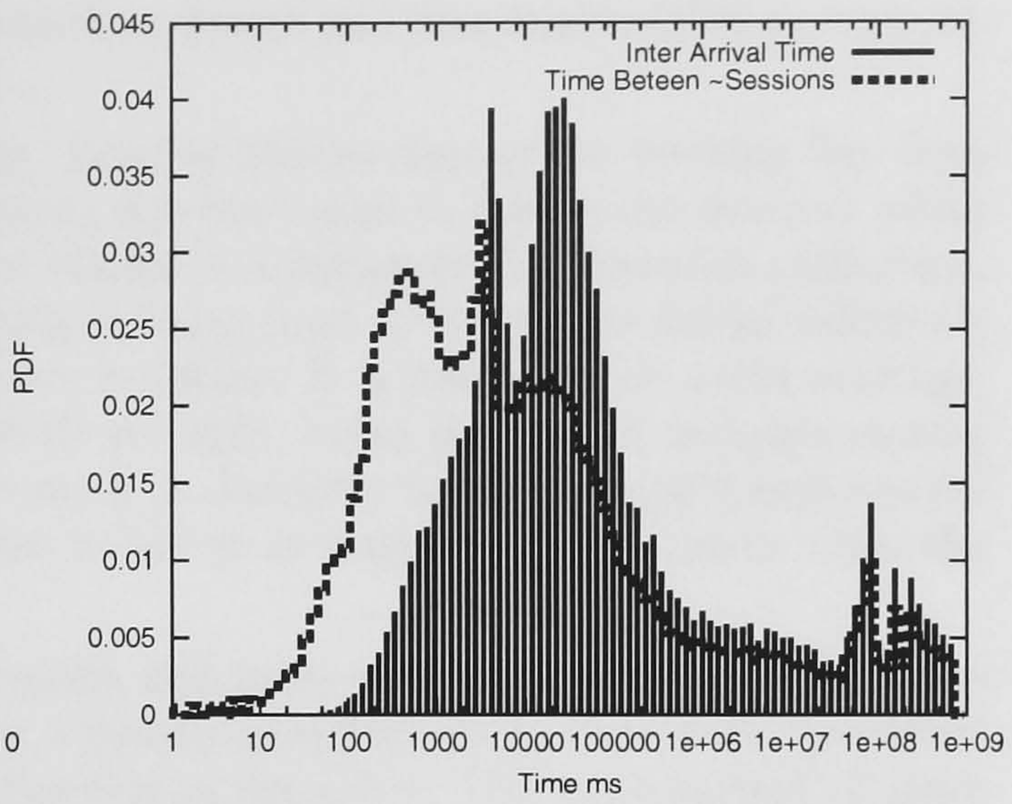


Figure 5.4: **Inter Session Arrival Times and Time Between Sessions: These graphs show the distributions for the gap between sessions and the gap between the start of the last and the start of the current session.**

Time Scale	Mechanism
Less than one RTT	Traffic Shaping
Multiple RTT	Flow & Congestion Control
Session	Connection Admission Control (CAC)
Day	Peak Load Pricing
Weeks or Longer	Capacity Planning

Table 5.3: **Traditional Traffic Management Mechanisms and Time Scales**

5.2.6 Time scale of Control

Congestion may occur at a number of different times scales and for each time scale different control mechanisms are appropriate [Kes91]. A summary of the main time scales is shown in Table 5.3 and each is discussed below.

Bursts of traffic occur at time scales of less than a RTT. At this time scale scheduling and buffer management can be used to shape traffic to meet network needs. TCP utilises the ACK clock to implicitly shape traffic at this granularity. Starting a connection with a large initial window prevents the ACK clock from smoothing the transmission of packets and results in the bursts of packets seen in Figure 5.2. There are a number of other ways in which the ACK clock may fail, one example of which is given in [Mog92], and [ZQK99] proposed solutions to them, which are discussed in detail in Section 8.5. Here, following [MJS99], it is sufficient to note that this is a distinct problem from traffic control at other time scales.

A RTT is a fundamental unit of time for congestion control, that utilises forward feedback from a network, because it is the minimum amount of time that is needed for the host's reaction to feedback to be manifested at the congested router. At this time scale a host can adjust its rate of send during a session depending on the current state of the network. Changing the way in which a connection is initialised does not alter the long term behaviour of this type of congestion control.

A Session is the period of time between a connection or set of connections being initiated and finished. Examples of a session include; a user joining a video conference, down loading a web page and transfering a file. An example of congestion control at this time scale is Call Admission

Control (CAC), which can be used to limit access to the network and thus enable QOS guarantees, for calls that have been admitted, to be satisfied.

There is a clear diurnal pattern to network usage. Load is heavier during the working day than at night. On the telephone network peak load pricing has been used to flatten the demand peaks and spread load more evenly across the day. This results in a higher level of network utilisation. In the situation where the experience of past traffic is being used to determine initial values for control variables the diurnal pattern of traffic raises problems. It is important to avoid readings, that are taken in a period when average traffic levels are light, being used to set variables during a period when congestion is high. At the same time it is desirable to avoid complicated control mechanisms and limiting the amount of data used to arrive at estimates, particularly when the volume of such data is low.

At the time scale of weeks and longer, new networks and hosts may join an Internet and new working relationships develop between institutions, resulting in changed long term traffic patterns. In the time scale of weeks and longer capacity planning is necessary. The introduction of more bandwidth is necessary but not sufficient to satisfy growing demand. Here a distinction should be drawn between the local domain and the wide area network. The local domain is likely to be under the administrative control of a single agent. Here problems of congestion can be easily resolved by properly deploying more resources. This observation allows the assumption that local congestion and latency is limited. On the WAN various agents are responsible for capacity planning. This leads to the situation where some routes have plenty of capacity and little problem with congestion, whilst others are under provisioned and have high levels of congestion. Therefore congestion management at other time scales becomes important for WAN traffic.

In recent years the problem of the evolution of congestion control algorithms has been identified as an important question [GK98]. Over a period of years the nature of the traffic itself may change. With the growth of faster processors and cheaper memory, the processing of first graphical and now multi-media data has become more common. For networks to support these new data types, new protocols and congestion schemes are developed. It may take years for them to be created, tested, implemented and distributed. Game theory [Axe85, KM99] has been applied to the problem of encouraging the development of responsible algorithms [KMBL99]. These observations mean that the approach adopted to session level congestion control, should not inhibit the development of algorithms at other time scales or locations.

Explicit congestion control in the Internet rests almost exclusively upon the time-scale of multiple RTTs. One way of addressing the problems with TCP's congestion control discussed in this section is to explore different ways of extending the scope of such congestion control. A second related approach is to recognise that the problem is caused by the absence of congestion control at the session time-scale and that the addition of congestion control at this time-scale could be used to facilitate the intelligent initialisation of control variables. This suggests the need for calculating a longer term average level of congestion which would remain valid for qualitatively longer periods of time than a particular level of the congestion window. It is the diurnal patterns of network usage that is likely to invalidate such an average, giving a lifespan of minutes to hours. At these longer time scales, benefits are more likely to accrue from sharing information between hosts.

5.2.7 Summary

In this section it has been argued that the combination of static initialisation and the confinement of adaptation within the lifetime of a connection is inappropriate for today's Internet where levels of congestion and round trip times vary by several orders of magnitude.

Increasing the size of the initial window has been considered as a means of reducing the delay incurred by short connections but rejected as it carries with it the disadvantage of further increasing the aggressiveness of long connections. It has been shown that setting the Initial Window and Slow Start threshold to the same value, allows Slow Start to be bypassed and more control to be exercised

over the average window size of a TCP connection. Carrying over the state between consecutive connection has been used to demonstrate that using past traffic to dynamically initialise TCP control variables allows average behaviour to approximate to the steady state behaviour of a long connection.

Analysis of Glasgow Web traffic suggests that extending the scope of TCP's adaptation from connections to sessions would only increase the size of data by a factor of two thereby only marginally increasing the opportunity for adaptation. Furthermore the average gap between sessions is sufficiently large to nullify the benefits in bandwidth utilisation by saving congestion information between sessions if the congestion window is exponentially decayed for idle round trip times between data transfers.

Approaching the problem from the point of view of time scales and dealing with the question of initialisation as belonging to a separate time scale from that of adaptation however, suggests that averages calculated over longer time scales may be utilised for initialisation even when there is significant delay between connections. Thus from this section we conclude that the addition of complementary session level congestion control would address the shortcomings of TCP's congestion control algorithms by rendering Slow Start largely redundant.

5.3 An Architecture for Best Effort Session Level Congestion Control

In this section an architecture, which addresses the mismatch between control mechanisms and traffic identified in Chapter 4 is proposed. The discussion opens with consideration of the functional decomposition of congestion control; a number of discrete functions are identified and associated with appropriate network elements. This division of labour is then related to the architecture of today's Internet: in particular the implications of the bindings between names, addressees and routes and the lack of network level congestion signalling are considered. Congestion control schemes aim to be fair, facilitate optimal network utilisation and to do so in an efficient manner. Each of these aims is discussed to establish what are desirable and reasonable expectations for introducing best effort session level congestion control. The section is closed by the presentation of a Location Information Server Architecture, which incorporates the functional decomposition outlined at the start of the section and is practicable to implement.

5.3.1 Functional Decomposition of Congestion Control

As early as 1979 [LRCI79] a functional division of labour, which allows different network elements to co-operate in implementing a congestion control scheme, was proposed. Future developments built on this division of labour [RJ88], and the widespread deployment of RED and ECN would see it applied to the Internet. This section discusses the functional decomposition of congestion control and how such a decomposition could be applied at the session time scale. Four main functions were identified.

1. Congestion Detection: The ability of a router to obtain a global view of traffic that may contribute to its congestion, makes it well placed for detecting the onset of congestion.

2. Communication: A mechanism is required for the router to be able to communicate the state of the network to the traffic source.

3. Host Reaction: Each source is responsible for interpreting and responding to notifications received from the network.

4. Enforcement: This is identified as the responsibility of the network, preferably the first hop router.

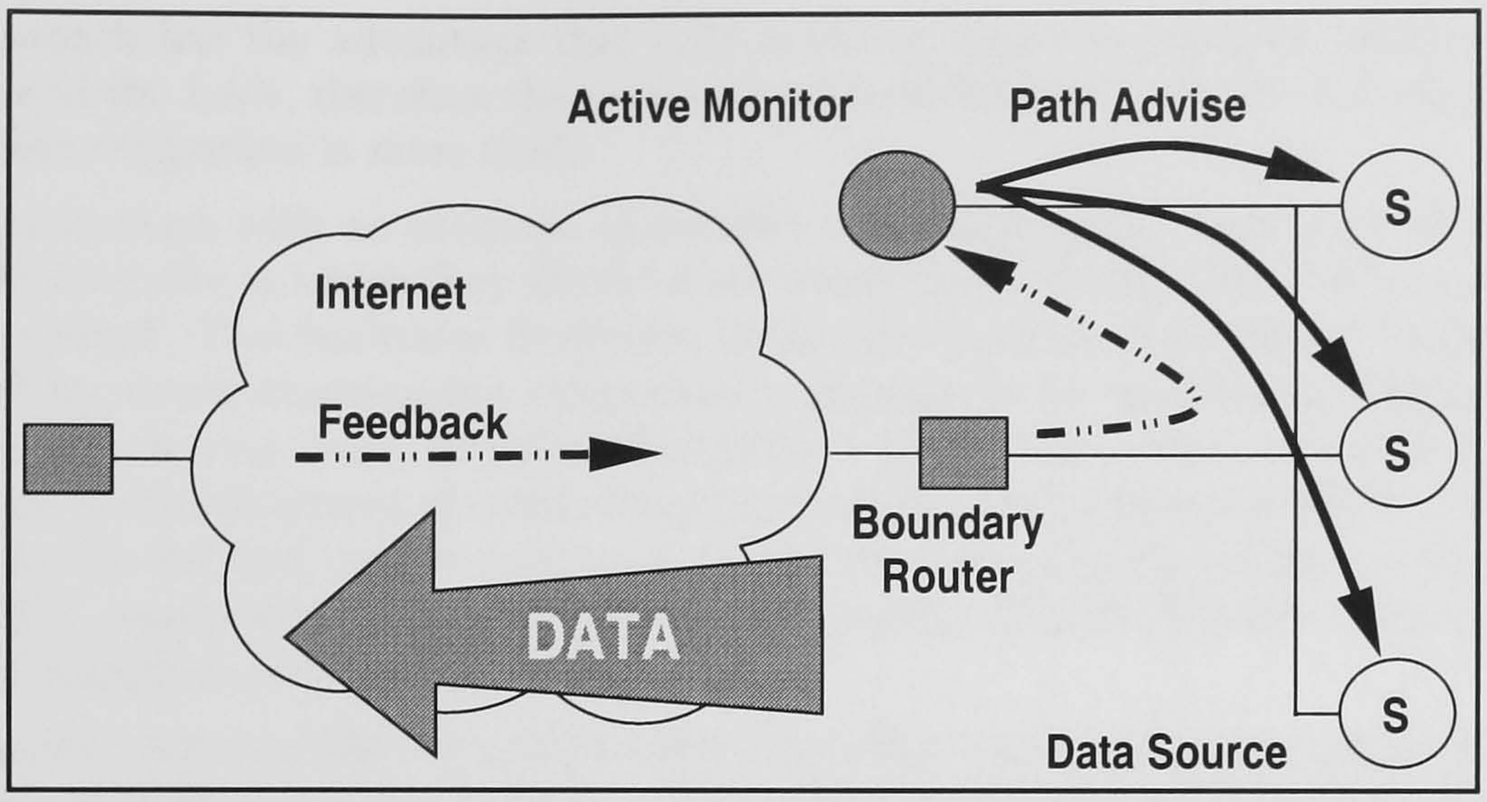


Figure 5.5: **Location Information Server Overview: The LIS monitors feedback from the network which enables it to supply hosts with predictions of likely network conditions.**

The congestion signal received by a host is binary in nature, either one or more packets in a window of data are dropped or there is no congestion notification. If the congested router is drop tail and the queue is full, arriving packets will be lost. Subsequent packets, which arrive at their destination will generate duplicate acknowledgments which may be interpreted as indicating that a congestion event has occurred. If there are insufficient subsequent packets, the absence of an acknowledgment signals a congestion event. In the absence of loss, successfully acknowledged data is interpreted as indicating the absence of congestion.

Over multiple windows of data, using additive increase and multiplicative decrease algorithms [CJ89] hosts react to the congestion signal, and can be expected to arrive at statistically [WC92] fair window sizes. The nature of the congestion signal therefore precludes a connection from moving straight to a fair window size at the start of the connection unless the signal is supplemented in some way.

The use of RED changes the criteria a router uses to define a congestion event and generate a congestion signal. The introduction of a form of Explicit Congestion Notification[2] would change the way that network conditions are conveyed but not the binary nature of the signal and the importance of the passage of time, in establishing an average operating window size around which the window size oscillates.

The introduction of Explicit Rate Indications (ERI) from routers would allow connections to quickly move to their equilibrium operating point [CCJ95, Cha94]. ERI have the advantage that the router, at the point of congestion, has a global view of traffic passing through and can therefore calculate a fair rate. The disadvantages of Explicit Rate Indications are that they require more network resources to communicate with end nodes, and they introduce more complexity into switches, at the core of the network. Explicit Rate Indications are unlikely to be introduced in the forceable future on the Internet.

If the network were to provide a quantitive estimate of the state of congestion on the path, connections could by pass their start-up transients and move straight to their steady state behaviour. This result can also be achieved by a Location Information Server converting the binary congestion signal into a quantitive estimate.

[2]With the introduction of Explicit Congestion Notifications, if a notification is received after a connection has finished it will have an effect on future traffic. Currently ECN notifications for the last window of a connection will have no effect. For average web traffic the last window is likely to contain half of the traffic and impose the most stress on the network.

The LIS approach has the advantage that only a binary signal is required between the core of the network and the LAN, therefore the overhead required for communication is minimized in the locations where congestion is most likely.

Supplying connections with an estimate of network conditions rather than a recommendation of the rate or window size at which they should start allows the host to decide how to react to the advise that is received. This facilitates flexibility, in the development of end point congestion control algorithms. The router experiencing congestion is responsible for generating congestion notifications, how it does so may vary, therefore flexibility is allowed in router responses to congestion. This facilitates the development of router based algorithms. Enforcement is left as the responsibility of the network and may take a number of forms, monitoring and penalization of unresponsive streams [FF97], attaching a cost to congestion notifications [KMBL99], fair queuing [DKS95] or some form of admission control [MPCC00].

Once a connection has started, its usual congestion control mechanisms are employed. Therefore if it starts at too high or too low a level, it can utilize the binary feedback from the network to adapt its rate of send in the normal way. In this way the congestion control mechanisms, that exist within connections, are supplemented by congestion control at the session level.

5.3.2 Internet Architecture: Names, Addresses and Routes

It is instructive to consider the proposal and how it relates to the bindings between Names, Addresses and Routes. Elsewhere [Coh78] it has been observed that a Name identifies the service required, an Address identifies the location of the name and a Route is how the address can be reached. The Internet, being a datagram network, binds an address to a route, as each packet traverses the network [Les83].

Congestion control on the Internet, involves attempting to associate characteristics, of the as yet unbound route, to a packet before it is sent. This is a contradiction, which lies at the heart of controlling congestion on datagram networks. In the next section some of the tensions, which derive from this contradiction are discussed.

It would be desirable to utilise all WAN Internet traffic in determining estimates of network conditions. This would be facilitated by network level signaling of RTTs and congestion. In practice the datagram architecture of the Internet means that these are not available. Even the proposal for ECN ties the backward notification to the transport layer.

These considerations make it difficult to design *session* level congestion control that utilises information from all traffic. Specific coding for different transport protocols and applications would be necessary. Therefore in this chapter the approach adopted is to focus on TCP traffic. The reliable byte stream nature of a TCP connection means that an association between packets is established, which can be utilised to infer network conditions. In the past and for the foreseeable future the majority of Internet traffic will be carried by TCP, making the approach of practicable value. There is however nothing to stop applications making use of the information extracted, even if the traffic they generated is not carried by TCP.

5.3.3 Aims Objectives and Constraints

In order to be able to design and evaluate the effectiveness of a congestion control scheme it is necessary first to define the criteria by which it should be judged. In this section three criteria are proposed; fairness, efficiency and optimisation. The limits of what can be achieved are also considered in relation to each of the criteria.

Fairness

A broad class of congestion control schemes utilise co-operation between routers, which generate a congestion signal, and end hosts to control the aggregate of demands for a network resource. This model implies the need for a policy on how access to the resource should be distributed between the competing demands.

The hegemonic approach is to aim at achieving Max-Min fairness. In a Max-Min system the users whose demand is smaller than the total resources divided by the number of users is met in full. The surplus is distributed evenly among the other users. In this system:

1. Resources are allocated in order of increasing demand.
2. No source gets a resource share larger than its demand.
3. Sources with unsatisfied demands get an equal share of the resource.

The Max-Min allocation of resource can be considered fair, if all user share an equal right to or derive an equal utility from the resource[3] [Kes97].

A Max-Min allocation has the advantage of being intuitively fair and easy to derive, at the point where congestion occurs. The number of connections, the rate at which each connection uses the resource and the rate at which the resource is available, are all that needs to be known. There is therefore a good fit between Max-Min fairness and deciding on resource allocation at the scarce resource.

There is a tension between sharing out resources, in a fair manner and maximising the Utility, that users receive from the network. If there are two flows; one of which contains very valuable data and the other contains less valuable data, an equal distribution of resource between them will not maximise the utility that is derived from the network. Instead preferential treatment should be given to the more valuable data.

It is easy for the user to make a decision about the utility they obtain from the network, it is much harder for a switch or router to come to a similar judgment. From this it follows that, the maximisation of utility fits well with the end system deciding how to respond to congestion notifications.

TCP and Max-Min Fairness

The fairness that is likely to be achieved between competing TCP connections can be compared to the Max-Min model. Three points of deviation are identified and discussed within the context of:

1. Window Congestion Control
2. Statistical Fairness
3. Connection Size

TCP uses window size, rather than rate, to control its bandwidth usage; the size of a connection's congestion window is adjusted in response to the presence or absence of congestion signals. In a window based scheme two connections, which receive the same feedback from the network but have different round trip times, will not take up the same bandwidth allocation.

These observations have lead to a distinction between rate based fairness, which approximates to the Max-Min model and Window fairness. It can be argued that Window Fairness is preferable, because it accounts for the aggregate resources consumed by a connection, rather than those at a single router.

[3] In practice resources are usually allocated to flows or connections rather than users. Consequently if there is not a one to one mapping of connections to users the fairness begins to break down

There is a second way in which TCP departs from the Max-Min model. The use of packet loss, as the congestion signal, makes it impractical for all connections to receive the same feedback. This leads to the observation that, TCP achieves at best statistical fairness, which has been defined to mean:

> *during the demand adjustment all users start increasing their traffic demands at the same time from the same level and with the same algorithm until the path is saturated. The final share that each user has may not be absolutely equal due to the statistical nature of traffic and network. But all users have the equal opportunity. In other words, such an approach is statistically fair, i.e. as over many runs of operation, each user on average has an equal share.* [WC91]

An important consequence is that an increase in loss may not signal an increase in congestion, but merely indicate that, the connection concerned is receiving an increased share of the total congestion notifications.

In Chapter 4 it was shown that an important factor in determining the bandwidth a connection receives is the number of RTTs, that it has to probe the network. Connections, which have a large amount of data, are likely to receive a greater share of the bandwidth than connections, with a smaller amount of data.

An accurate characterization of TCP's fairness regime is important in establishing the bounds of fairness that can be expected when session level congestion control is added, that utilises the same feedback mechanisms as TCP.

Fairness and Session Congestion Control The fairness achieved when session based congestion control is added to TCP can be compared against two benchmarks. The fairness that is really achieved by TCP on the Internet, and an ideal where competing streams sharing a route receive an equal bandwidth share. When comparison is made with the actual performance of TCP, it is easier to show, that dynamically initialising the congestion window will decrease the disparity between bulk transfer and interactive graphic users.

The comparison with the ideal situation, where very long or infinite connections are assumed is a tougher criteria. The information, that is available before a connection is started, will be inferior to the information available in the middle of a long connection. It is likely to be older and so network conditions may have changed, and it is also likely to come from a number of small connections rather than the long connection of the ideal model.

These observations lead to the conclusion that, an estimate of the fair bandwidth a connection should start with can only approach the bandwidth, that would have been allocated in the middle of a long connection. Thus whilst staying within the TCP paradigm, and utilising the same feedback signal as TCP connections, the fairness goal is to improve upon the fairness actually achieved by TCP connections and to as closely as possible approach equal distribution.

Efficiency

An efficient congestion control scheme will use the minimum amount of network resources possible. In particular the use of scarce resources in a time of congestion should be avoided. There is a trade off between efficiency and optimisation, the allocation of more resources to the management of congestion, if properly deployed should allow a higher level of optimisation.

This trade off lies at the heart of debates between those advocating special control packets or more complex router algorithms which would allow greater optimisation and those arguing for simplicity and more efficient congestion control.

It can be observed that not all network resources have equal value. It is an assumption of this work that the utilisation of Local Area Network resources to improve congestion control of Wide

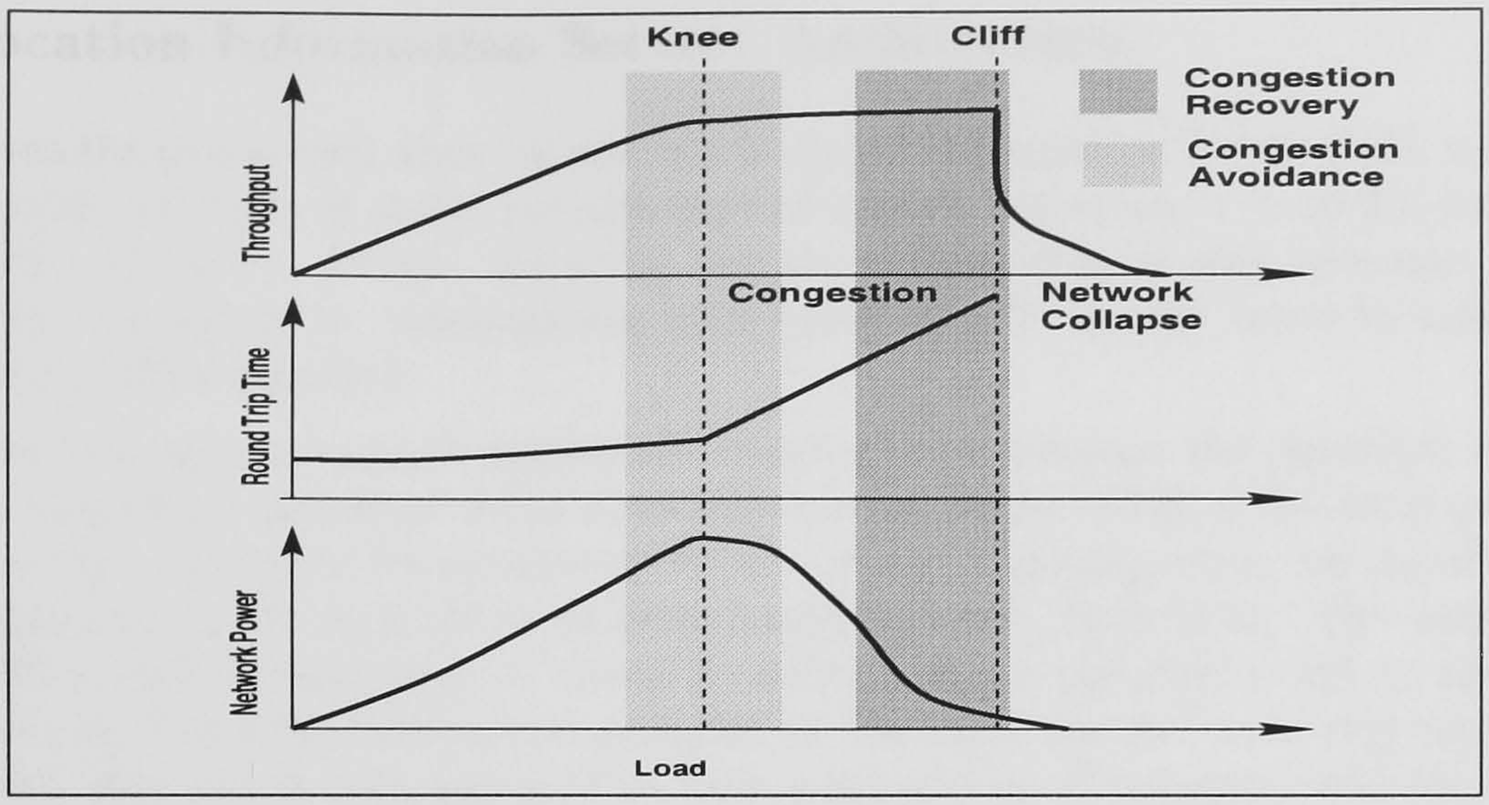


Figure 5.6: **Congestion Avoidance and Recovery: It is desirable to operate the network at the Knee of the delay load curve (Congestion Avoidance). it is necessary to prevent the network from operating in the area of network collapse (Congestion Recovery).**

Area Network traffic is a sensible trade off, because it is possible for as local administrator to improve their local resources through network engineering.

Optimisation

An optimal congestion control scheme will allow the maximum utilisation of network resources, when there is excess demand and not prevent their utilisation, when demand is lower than the optimal level. Network Power has been advocated as a measure of the utilisation routers [GK80]

$$Power = \frac{Throughput}{ResponseTime} \tag{5.1}$$

Figure 5.6 shows the behaviour of Throughput, Delay and Network Power as presented load is increased for a network without congestion control. By the definition given above a congestion control scheme can be considered as optimal, if it allows the network to operate in the region of the delay load graph or around the peak of the power load graph.

In Chapter 8 a contrast was drawn between BFCA and TCP. It was shown that BFCA attempts to operate the network in the optimal region, whilst the use of Drop Tail routers and loss as a congestion signal mean that TCP operates to the left of the cliff in Figure 5.6.

It has also been shown that, the need for a TCP connection to ramp up its share of bandwidth over several RTTs limits its bandwidth utilisation, even if there is sufficient network resources for all demand to be met.

The function of the LIS has been defined as monitoring traffic and converting the binary congestion signal into a probability. The monitor then communicates this probability to hosts when a connection is set up. This does not address the problem of at what level the network should be operated at, it is left to the router to decide its optimal operating point and generate an appropriate signal. This has the advantage that improvements in algorithms at the router will feed through LIS to the hosts, without modifications in the LIS being required.

The use of a Location Information Server does however address the problem of utilisation of resources when there is no shortage, as connections are not slowed down, by the need to ramp up their bandwidth usage over several RTTs.

5.3.4 Location Information Server: Architecture

In order to test the hypothesis, that the use of a Location Information Server (LIS) would improve upon the current practice of static initialisation of control variables, it is useful to evaluate an implementation of such a server. To carry out an evaluation it is also necessary to design a mechanism for the server to communicate with hosts, and to modify hosts to enable them to utilise the information supplied.

The proposed architecture aims to resolve the contradiction between the character of web traffic and Internet congestion signaling. A Location Information Server (LIS), under local administrative control, on a well engineered local network where latency and congestion are minimal, provides the information that hosts need about network conditions to a destination. The interpretation of that information and the mapping to values for initial control variables is left to the host. This division of labour allows the information supplied by the LIS to be put to a wide range of uses by the local hosts, that are in turn assumed to have a knowledge of transport and application level behaviour.

What information should the LIS supply? A prediction of the proportion of congestion notifications is necessary for the host to calculate a TCP Fair window size. This may be scaled to provide the Initial Window size, as the basis for some proportionally fair starting window [CO98], or in combination with the network Round Trip Time(RTT) to calculate an initial fair rate. The RTT estimate may also be used by TCP to calculate an initial Retransmit Time Out (RTO), if factors such as the variance in RTT and delayed acknowledgments are accounted for.

The quality of the estimates is assessed, by a host, using information about the data stream(s), from which the information was extracted. This includes the number of packets, the age of the data used and the recent maximum window.

The existence of network level signaling of connections, congestion and RTT would allow the extraction of estimates from all traffic, however the nature of the IP service precludes this. Instead the implementation used to evaluate the above architecture uses passive monitoring of TCP traffic, to detect the start of connections, determine the proportion of Implicit Congestion Notifications (ICNs) and measure network RTTs.

Passive monitoring has the advantage over techniques that use probe traffic, such as the Network Probe Demon [Pax96b], in that no extra traffic, which may contribute to congestion, is generated. A second advantage is that the information available for estimates automatically increases on the busiest routes and at the busiest times of the day. This means that the best estimates are achieved on the routes with the most traffic and that misleading readings, such as those taken in the middle of the night when there is little real traffic, have a minimal effect on estimations.

End Host Reporting re-uses the results of calculations conducted by TCP to estimate such values as the Smoothed Round Trip time and the correct congestion window size. This implies that a mechanism needs to be defined to enable communication between the Information Server and hosts, the obvious solution is to use control packets. Upon receipt of the reports the Server performs some level of aggregation and makes the results available to hosts.

This approach has a number of pitfalls: TCP estimates may be inaccurate; the granularity of reporting needs to be defined; and end host modification is required. The values calculated by TCP may be unreliable. Two examples will suffice. Many TCP implementations allow the congestion window to increase even when the connection is application or window limited. In these cases, the size of the congestion window may bear little resemblance to the amount of traffic, that is actually transmitted each RTT. Secondly `srtt` is typically initialised to a large value, if enough readings are not taken to allow it to converge to the true value, then the reported value may be several orders of magnitude greater than the actual RTT.

The granularity at which reports are generated needs to be resolved. Here there is a trade off. If reporting takes place at a fine granularity, the information available to the server will be better, but the overhead in bandwidth, used in communicating the information, will be high. If a courser

granularity is used, the overhead will be reduced, but the data will have aged, before it is made available for sharing. Finally End Host Reporting requires that each host's operating system be altered, to allow information on network conditions to be gathered.

The centralisation of estimation in the LIS has a number of advantages[4] over using the LIS to share estimates that are made at hosts [Ste99, ZQK99]. The question of whether a host's estimates are accurate is not posed, the deployment of new algorithms is facilitated and it is easy to ensures that estimates are derived in a uniform manner. In addition the update of estimates can take place within the LIS and require no communication from a host thereby reducing the load placed on the local network.

The LIS passively monitors TCP data streams. It therefore has no direct knowledge of host based sessions but can easily observe the ACKs that mark the start of a connection. Consequently, updates to hosts are made on a per connection basis. For the LIS a session therefore lasts for precisely one connection.

There is however an interface for hosts to request information and the automatic updating can be turned off for a particular host. Information can then be sent only at the start of a session as defined by the host. It would be for the host to determine the definition of a session. Some heuristic such as a certain sized gap between connections might be used to signal the end of a session.

5.3.5 Summary

In this section it has been argued that the initialisation of control variables to values that reflect current network conditions would allow available bandwidth to be utilised more effectively, whilst reducing the number of false time outs and preserving the fairness semantics that exist on the Internet.

An architecture for introducing best effort congestion at the session time-scale has been introduced. This can be facilitated by a monitor passively observing passing traffic and communicating its estimation of network conditions to a host during the establishment of a connection. Whilst this approach does not fit well with the underlying datagram paradigm, which lies at the heart of the Internet, the connection oriented nature of TCP allows its application to a significant proportion of Internet traffic.

5.4 The Location Information Server: Design and Implementation

The design and implementation of a Location Information Server is presented in this section. The top level function of the server is to detect the setting up of a connection and to supply the local host with the best possible prediction of the network conditions, that will be experienced by the ensuing traffic. The goal here is to demonstrate that the construction of an LIS is a practical proposition. An evaluation of the success of the design is the subject of Chapter 6.

In order to further refine the operational goals of the server the functionality that the LIS is required to fulfill is examined with particular reference to the timeliness constraints and content of the QoS feedback that is to be supplied. In order to deliver the feedback there are four main sets

[4]Applying techniques developed for Passive Monitoring of network traffic, allows network information to be gathered, whilst avoiding the difficulties associated with end host reporting.

There is no need to rely upon the potentially inaccurate estimates of TCP host implementations, the problem of communication reduces from the transfer of data across the network, to the transfer of data between objects in the server and there is no need for host modifications to facilitate information gathering.

The centralisation of information extraction and the processing of that information in one place, also allows algorithms and parameters to be easily modified.

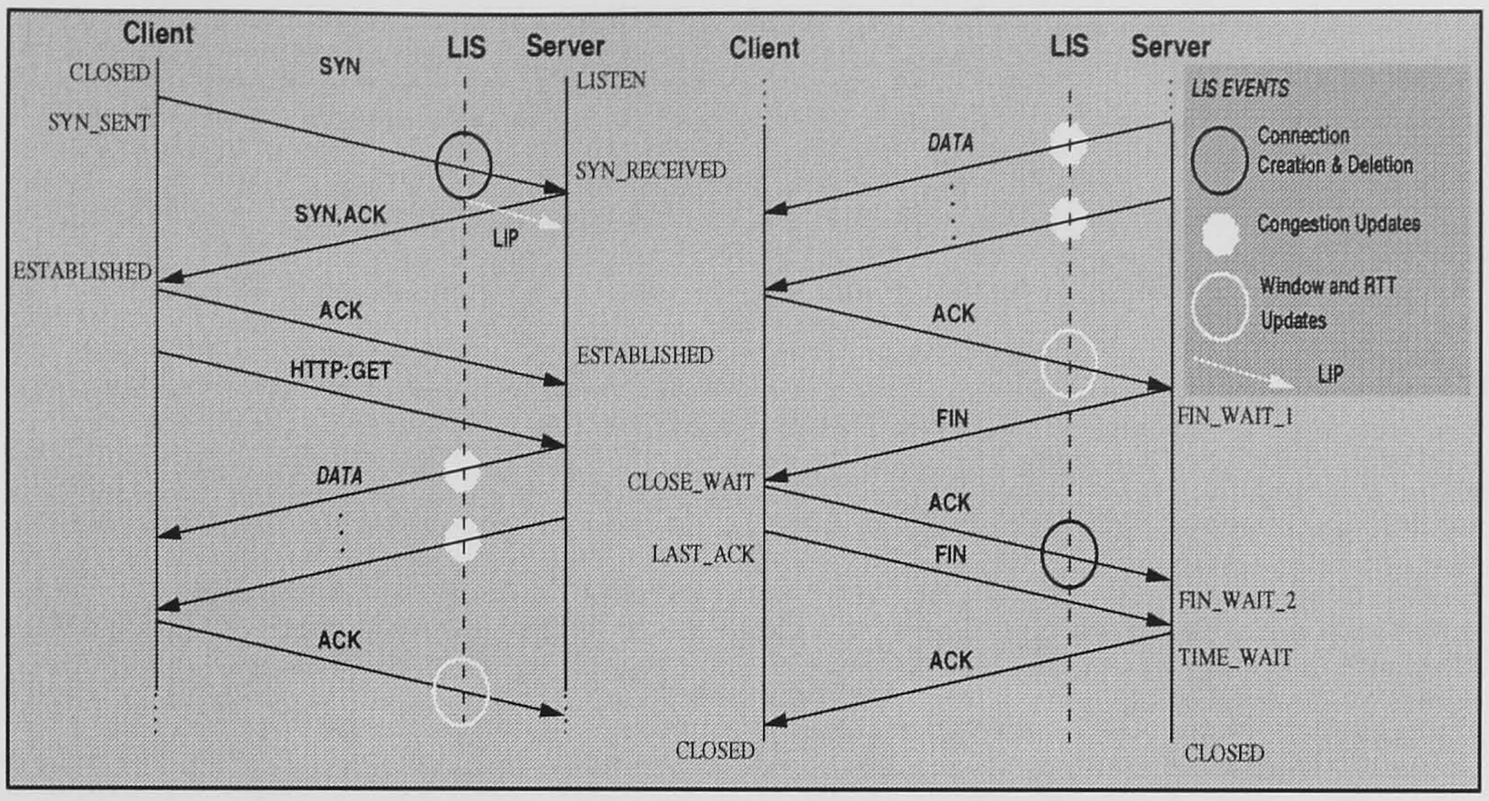


Figure 5.7: **LIS Signal Sequence Diagram: Which illustrates when the LIS detects a connection, communicates with a server, detects congestion events and measures Window sizes and RTTs.**

of operations that need to be performed; a mechanism is required for capturing and demultiplexing information from passing TCP headers, readings need to be extracted from the streams of packets that make up connections, statistics on the QoS experienced need to be maintained for each location, and these statistics need to be communicated to relevant hosts in a timely manner. This division of functionality corresponds to the four main modules in the system.

The remainder of this section defines the top level structure and interactions between each module and then presents the design of each module in more detail. The section concludes with the examination of the communication module which sets the scene for consideration of an end host implementation that utilises the feedback from an LIS.

The server has been implemented on a platform running FreeBSD 2.2.7, it utilises the Berkeley Packet Filter (BPF) to capture the headers of TCP packets. Structurally the design is object oriented and the implementation is in C++.

5.4.1 Definition of Server Functionality

The top level functionality of the LIS is to monitor TCP traffic at the WAN LAN boundary, to extract information about traffic conditions from TCP headers, using techniques developed for passive monitoring of traffic, to use these observations to make predictions of likely network conditions and to estimate confidence in these predictions.

This information is then communicated to hosts, at the time that it is needed, whilst the communication overhead is kept to a minimum. TCP's three way handshake, illustrated in Figure 5.7 offers a period of time between the detection of a new connection by the LIS and the commencement of data transfer. During this period of time the LIS needs to communicate to the host the expected conditions and the host needs to initialise its control variables.

Here the nature of the information, that can usefully be supplied by the server to hosts is considered. This information can be divided into two types; that which relates directly to the path between a pair of hosts and that which relates to the data stream(s), from which the path information was extracted. The first is useful for setting control parameters and the second is useful for assessing the confidence in the path information.

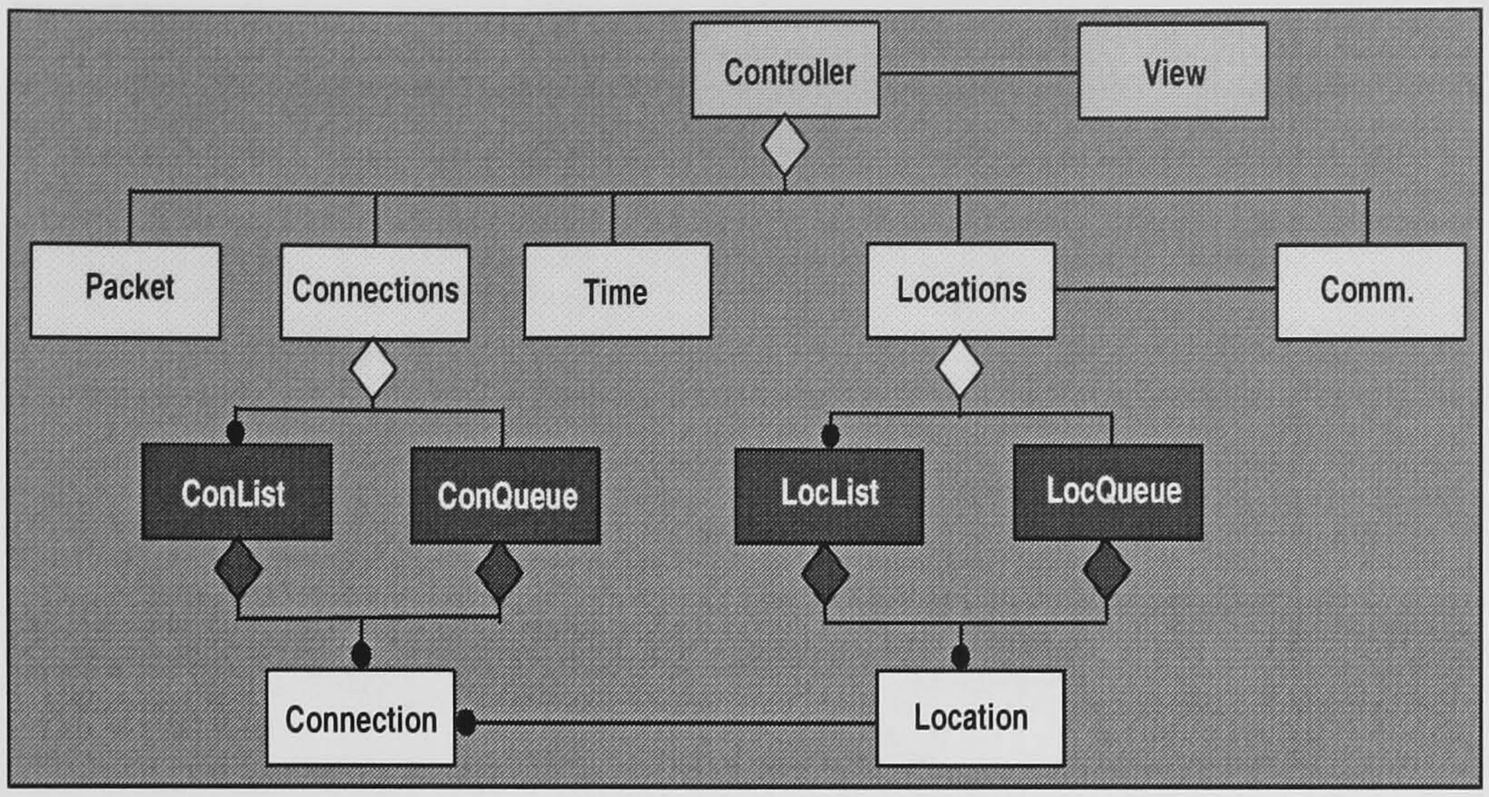


Figure 5.8: **Top Level Object Model of LIS: Each Connection object holds data while a connection is live and feeds readings to a Location object, where averages are maintained.**

Path Characteristics There are a number of network characteristics that it would be useful for a host to know before the start of a connection. The most significant ones are discussed below:

1. Implicit Congestion Notification Probability — From the ICN probability a TCP connection can calculate the fair window size.
2. Round Trip Time — RTT estimates can be used to set initial RTO value to suit the location. Consequently on paths where the current default results in a large proportion of false retransmissions a higher value can be set, which will reduce the amount of network resources that are waisted. Where the predicted RTT is low a smaller RTO value can be used, facilitating quicker recovery from packet loss.

   A second function for RTT estimates relates to Rate Based flow control. If the RTT is known it can be used to convert a Fair Window size into a rate. This is useful for rate based congestion control, which is aiming to be *TCP friendly*, or to combine Rate Based Pacing with window flow control.

Data Stream Characteristics The characteristics of the data streams, which were used to estimate network characteristics, is important in determining the level of confidence in the results.

1. Used Window Size — If there is a very low proportion of congestion notifications, but the connection is application or window limited, the projected congestion window may be much larger than the actual congestion window used. This could result in the initial congestion window being larger than the path will support. If the peak used window size is known this can be utilised to prevent too large an initial window.
2. Age of data — If the data used to estimate an Initial Window size is old, it may not reflect current network conditions. The same argument holds for the estimated RTT, but the Maximum and Minimum RTTs are likely to maintain their value over larger time spans.

5.4.2 LIS Structure

This section provides an overview of the interactions between the main modules in the Location Information Server. The discussion is illustrated by Figure 5.7, which shows a signal sequence

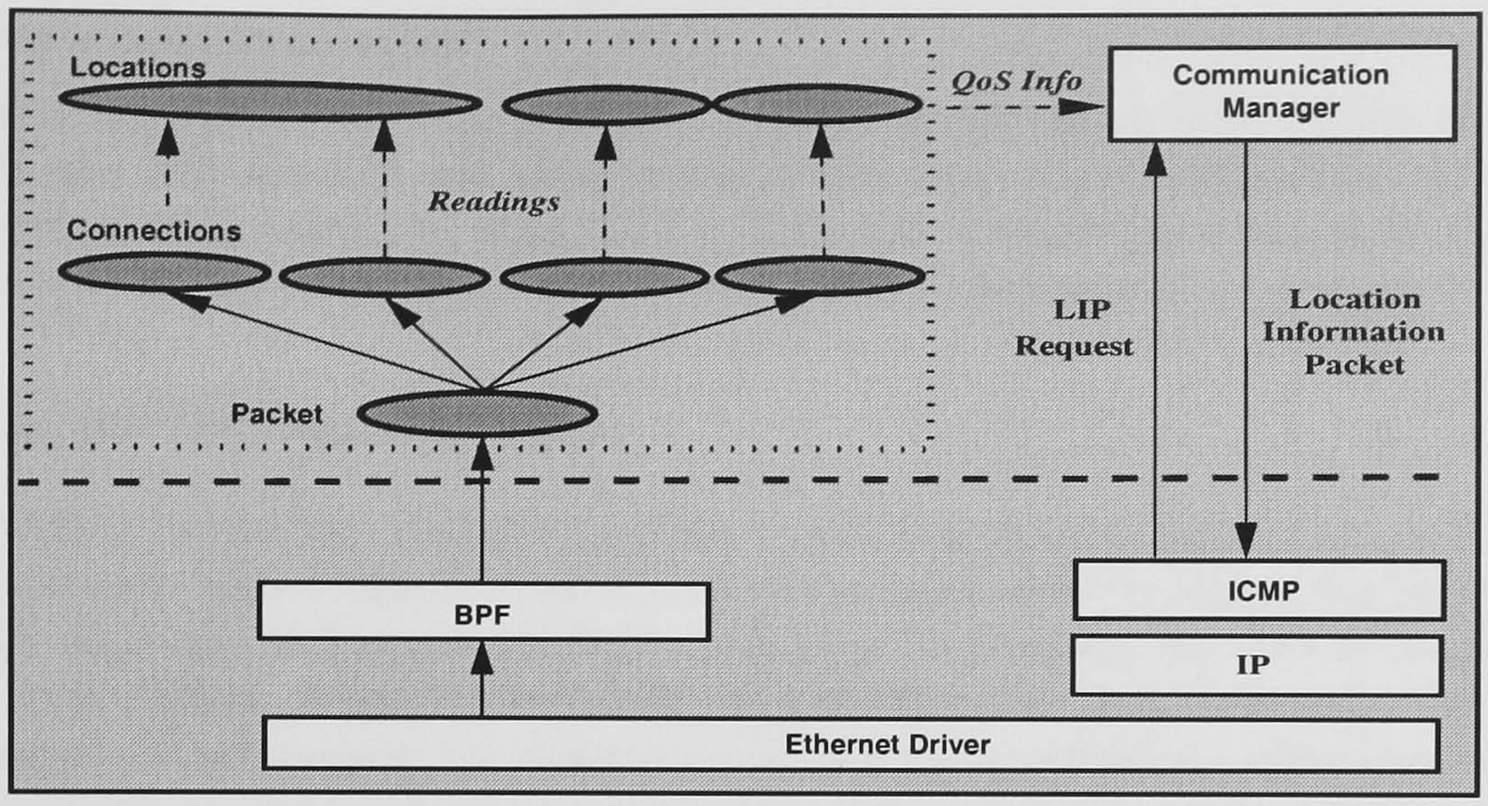


Figure 5.9: **Location Information Server Data Flows: Packet headers are captured by the Berkeley Packet Filter encapsulated in a Packet object by the LIS. The correct connection object is identified and appropriate readings taken. These readings are then passed to a Location object which maintains long term averages.**

diagram for a simple web request, Figure 5.8, which shows a top level object model for the LIS and Figure 5.9, which shows the main data flows.

A simple Web Request

First consider the signal sequence diagram which showed the interactions between a web client, a web server and an LIS. To initiate a connection the client sends a SYN packet. When the SYN is observed by the LIS it performs a number of actions. First the packet header is captured by the BPF which prepends a timestamp. The LIS then updates the packet object, which encapsulates the current packet header and provides methods for accessing the headers data fields.

The LIS then checks to see whether it holds statistics (in a Location object) for the client. If so the statistics are retrieved and communicated to the web server by the Communication Manager, which controls all between the server and hosts. The statistics supplied by the LIS may be used them to set the initial sizes of the following variables `cwnd`, `ssthresh`, `srtt` and `vrtt`. This communication and initialisation should be completed before the commencement of data transfer initiated shortly after receipt of the clients HTTP GET request.

A new connection object is then created within the LIS and associated with the appropriate location. For each active connection a separate object is maintained, once the connection enters the ESTABLISHED state readings are taken at the connection layer within the LIS and passed to the Location layer where statistics are maintained.

Once for each window of data readings of the RTT and utilised window size are made by a connection object and communicated to the relevant Location, these events are shown as white rimmed circles in Figure 5.7. Each time a data packet is observed, shown as small white discs, a message is sent to the Location layer, the connection object determines whether this packet represents an Implicit Congestion Notification in which case a congestion event notification is sent to the Location layer.

Each location object corresponds to a collection of IP addresses that are assumed to share a common location in this implementation the group of addresses that share the same higher order three bytes. Each Location object maintains a data structure that contains estimates of path and data flow characteristics that may be queried by hosts, which are updated upon the receipt of

messages from a connection object.

Once the server has sent all the requested data a FIN segment will be transmitted, when the client acknowledges the FIN the LIS will close down its tracking of the connection. This somewhat more aggressive closure behaviour than TCP is intended to reduce to a minimum the number of connections that are not closed down and thereby control the memory requirements of the LIS.

Data Access and Garbage Collection

From Figure 5.8, it can be seen that the controller contains a separate data structure for connection and location objects. There may be many Connections and Locations maintained at any one time, it is therefore important to have an efficient mechanism for locating the correct connection which corresponds to a packet. A one behind cache is maintained so that consecutive packets for the same connection do not require the connection object to be retrieved, otherwise data access is facilitated by holding connections in a hash table, each entry has a linked list of zero or more connections attached. The sum of the port number and second byte of the IP address moded with a prime number, which is the size of the table, have been found to give a good distribution [Mog92] and is used as the hash key.

Location objects are also held in a hash table, which uses the same key as connections. When a connection is being set up the relevant location is found by querying the hash table. Once it has been found a pointer to its Location is maintained by each connection, which is used for sending messages during the lifetime of the connection.

Normally, when a connection is closed its state is removed from the system, the state for a location is maintained so that it is available the next time a connection is set up. It is possible however that the connection will not close properly, either because of a malfunction at the host or because the monitor failed to detect a Reset or FIN packet. Over time these can cause the number of Connections in the system to grow without bound. For this reason it is necessary to collect and dispose of the state associated with stale connections.

For one month's traffic with the Glasgow University DCS servers there were connections from over 20,000 separate locations, whilst this poses no problem for the server it can be expected that over time this number would increase to the point where it would be necessary to remove some locations from the system. The distribution of connections among locations suggests that many Locations could be removed without detriment. Using the Glasgow University traffic as an example Table 4.1 shows that 80% of Connections were with 20% of Locations and 50% of Locations had only 5% of the connections. Thus a large proportion of Locations have a small amount of traffic.

The need to collect stale state is met by holding Connection and Location objects in data structures designed to give easy access to the Connection or Location that is to be deleted. Locations and Connections are each held in queues for this purpose. A separate limit on the number of each is defined, once the limit has been reached a new object cannot be constructed without deleting an existing object of that type.

Each time a Connection is associated with a Location, the Connection and is added to the front of its queues and the Location is moved to the front of its queue. When a Location is to be removed, first a check is performed to see if there are any live connections associated with it. If so, it is moved to the front of its Queue and the next Location is checked. If there are no live connections it is removed. This approximates to removing the Least Recently Used Location and the Oldest Connection. To ensure that a Location may always be removed, the number of connections allowed is less than the number of Locations.

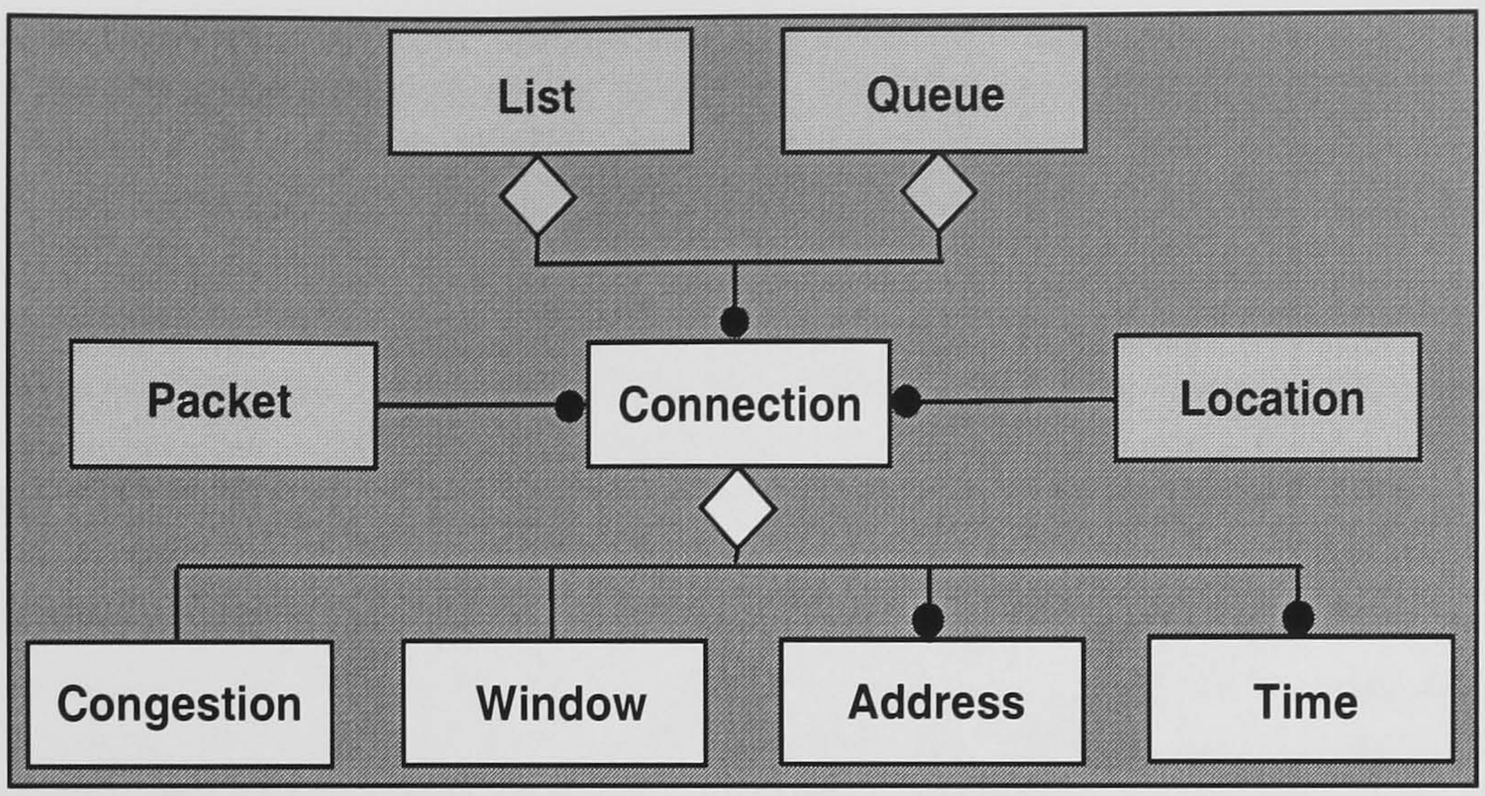


Figure 5.10: **Object Model for the Connection Layer: Each connection object contains objects for tracking congestion events, utilised window size and time.**

5.4.3 Packet Capture

The Berkeley Packet Filter (BPF) was used to capture packet headers, which is configured to pass packet headers up after a maximum delay of 10ms. Each packet is then processed in turn by LIS. The BPF prepends a timestamp to the front of the captured headers, these timestamps are used by the LIS to update a global clock, which the relevant objects are initialised.

5.4.4 Connection Layer

Figure 5.10 shows the object model for the connection layer, each connection object corresponds to a single live TCP connection, which is being monitored. It contains source and destination IP addresses and port numbers which uniquely identify each connection, time objects, for the time the object was created, the time the connection became established and the time the last packet was observed. The connection object also contains state for tracking sequence numbers and state transitions. The Congestion and Window objects are used to extract QoS readings.

Upon receipt of a packet the controller through a hash table look up retrieves the appropriate connection, which in turn processes the packet. Three sets of operations are carried out. The evolution of each connection through its state transitions are tracked to enable appropriate actions to be taken at the correct points. Data transfer takes place within the ESTABLISHED state, during this stage the evolution of the local hosts send and receive windows are tracked, which mirrors end host flow control window evolution. The tracking of sequence numbers provides the framework which facilitates the extraction of QoS information. Each time a data packet from the local host is observed it is evaluated to determine whether it represents an Implicit Congestion Notification or not. Once per round readings of the RTT and the size of the offered window are made and passed to the Location layer.

Macro State - TCP State Machine

The tracking of state transitions provides a set of reference points, which allow actions to be taken at appropriate points. For example; it allows the ending of connections to be systematically detected and memory reclaimed, it allows the point at which an LIP packet is sent to be defined for all connections and allows the time at which a connection enters the established state to be recorded.

The state transitions that a TCP connection goes through are defined in RFC 793 [Pos81d]. Figure 5.7 shows the state transitions that both the client and server would be likely to go through during the download of a simple HTML page. The LIS however is situated between the client and server in this case near the server. From this perspective the transitions undergone by each connection do not correspond directly to those undergone by the end points.

Firstly, state transitions that are not caused by the receipt of a packet may not be detected by the LIS. For example the LIS will not be able to detect the transition from CLOSED to LISTEN caused by an application setting up a listening socket. Neither will the LIS be able to detect the transition from TIME_WAIT to CLOSED, which should occur after a 2MSL timeout. Fortunately the inability to detect the TIME_WAIT/CLOSED transition is inconsequential and the LISTEN and CLOSED states can be merged for the LIS without loss of functionality.

Secondly, a number of transitions which are atomic at the host appear as two separate transitions for the LIS. To take one example, the transitions from the LISTEN to SYN RECEIVED states are caused by the receipt of a SYN and result in the transmission of a SYN,ACK. To a monitor there is a gap between the observation of the SYN and the SYN,ACK this is accounted for by introducing a transitory intermediary state.

Thirdly, it is possible that some packets would not be observed, allowing for example a direct transition from the ESTABLISHED to the TIME_WAIT states.

To account for these difficulties it was necessary to define the transitions as observed from an intermediary point between the client and server. The approach adopted was to take the starting point as the State Transitions defined in RFC 793, to merge the LISTEN and CLOSED states and to introduce a number of intermediate transitions. The resulting state transition diagram is shown in Figure 5.11.

There are two methods, which process state transitions, one for incoming and one for outgoing packets.

Micro State - Tracking of Sequence Numbers

Once a connection enters the ESTABLISHED state no transitions occur until a FIN or RESET is observed. In order to track the progress of a connection during this data transfer phase, the progress of data and acknowledgments through their respective sequence spaces is tracked, for both sent and received data. Per connection state for data sent by the local host includes; the oldest unacknowledged sent sequence number, the highest sequence number sent and the size of the window offered by the receiver. Per connection state for data received by the local host includes the next sequence number expected, the highest unacknowledged sequence number and the size of the receive window. The maximum acknowledged segment size, the maximum utilised window size the time the connection became established and the time that the last packet was sent are also maintained.

Window and Congestion objects also utilise the tracking of sequence numbers to identify congestion events, and allow the size of the utilised window and the RTT to be estimated.

ICNs: Drop tail routers implicitly signal congestion to hosts by dropping packets. The fact that TCP provides a reliable service means that when it detects a dropped packet the data is retransmitted. The LIS may in turn detect a dropped packet in two ways; through the detection of duplicate acknowledgments and through the observation of a repeat transmission. If the lowest sequence number in a packet is smaller than the highest sequence number sent then the LIS classifies it as a retransmission. Keep alive packets are identified as such and ignored.

The Fast Recovery and Fast Retransmit algorithms can result in TCP exhibiting go back N and retransmit behaviour, with packets being unnecessarily retransmitted. These segments should not be counted as ICNs as they are a result of TCP protocol behaviour rather than a congestion

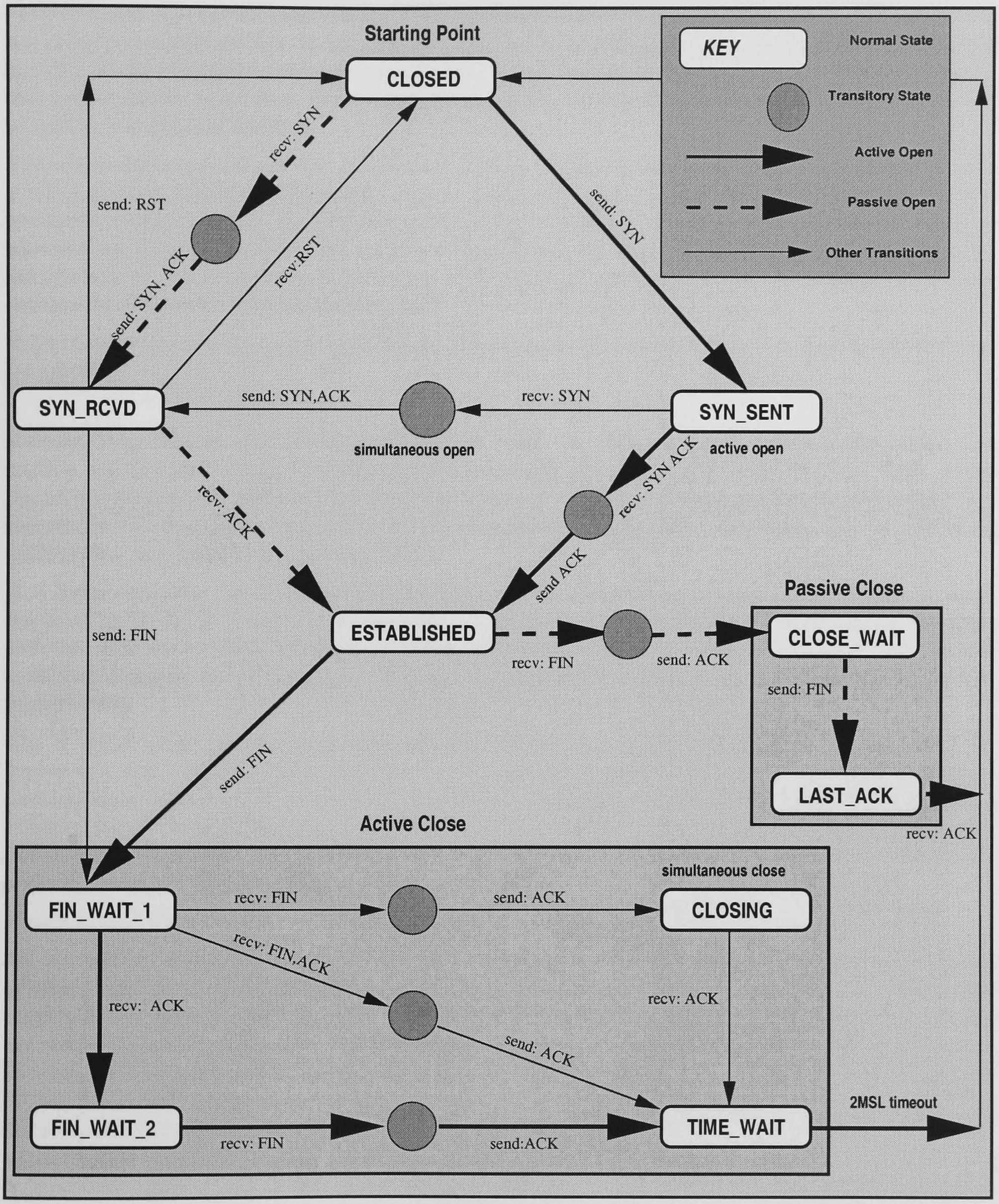


Figure 5.11: **TCP State Transition Diagram for a Monitor: The LIS is positioned between TCP endpoints, consequently a number of transitions which appear atomic to TCP have transitory states which are observed by the monitor.**

notification. The bursty patterns of TCP traffic mean that often more than one packet belonging to a connection is dropped during a single congestion event, in this scenario only one of the multiple retransmissions that are necessary should count as a retransmission. Once a packet has been dropped it takes at least one RTT for the congestion event to be detected by the source, for the congestion window size to be adjusted and for the change in the rate at which packets are arriving at the congested router to be reduced. Congestion notifications that are generated during this period simply show that there is congestion at the old window size and do not indicate a need to reduce the window further.

The above issues are addressed by filtering which retransmissions are interpreted as ICNs; once a loss has been detected subsequent repeat packets are not counted as retransmissions for one window's worth of data. Specifically once a retransmission has been detected further retransmissions are not counted as ICNs until new data has been sent and the retransmission has been acknowledged. This will take at least one RTT. The exception is when a retransmission is itself retransmitted, which is counted as an ICN.

Each time a data packet is observed and each time an ICN is detected the relevant location object is updated.

Round Trip Times and Utilised Window Size In order to track the evolution of the congestion window and calculate RTTs the data transfer phase is divided into rounds. Each round lasts from the transfer of a *distinguished* segment until its acknowledgment. The first distinguished segment is the first data segment sent and subsequent distinguished segments are the first data packet after the end of the previous round.

It is important that the readings for the RTT are as accurate as possible. There may be large fluctuations in RTT caused by delayed acknowledgments, for example NET/3 implementations wait up to 200m/s before sending an acknowledgment if there is no data to piggy back on and a further segment is not received. If a second segment is received an acknowledgment is sent immediately.

The Window class has two main methods, one for processing received packets and the other for processing sent packets. The segment that is transmitted at the start of a round is called a distinguished segment. It must carry data that needs to be acknowledged and must not be a retransmission. When a distinguished segment is observed the time and start and finish sequence numbers are recorded. The time and end sequence number are also recorded for the next data segment observed after the distinguished segment.

Count is kept of the number of bytes and number of packets transmitted during the round.

All the acknowledgments are checked, when a packet is received that acknowledges the distinguished segment a check is performed to determine whether it is for at least two MSS sized segments. This ensures that it is not a delayed acknowledgment. Next the highest sequence number acknowledged is evaluated. If it is one higher than the distinguished segment this implies that the acknowledgments transmission was triggered by receipt of the distinguished segment and the RTT is taken as the time between the transmission of the distinguished segment and receipt of the acknowledgment. If it is one higher than the second segment this implies that it was the second segment that triggered transmission of the acknowledgment and the RTT is taken as the difference between the observation of the second segment and the current time. If the sequence number is higher still it is not known which segment triggered its transmission and the RTT is therefore not known. If there has been a retransmission during the round due to Karn's law the segment to which the acknowledgment applies may not be known and so the RTT reading for that round is discarded.

One sample of the utilised window size is taken every round by counting the number of data bytes that are transmitted between the sending of a distinguished segment and the receipt of its acknowledgment. If the connection is neither client or server limited the utilised window will approximate to the size of the congestion window.

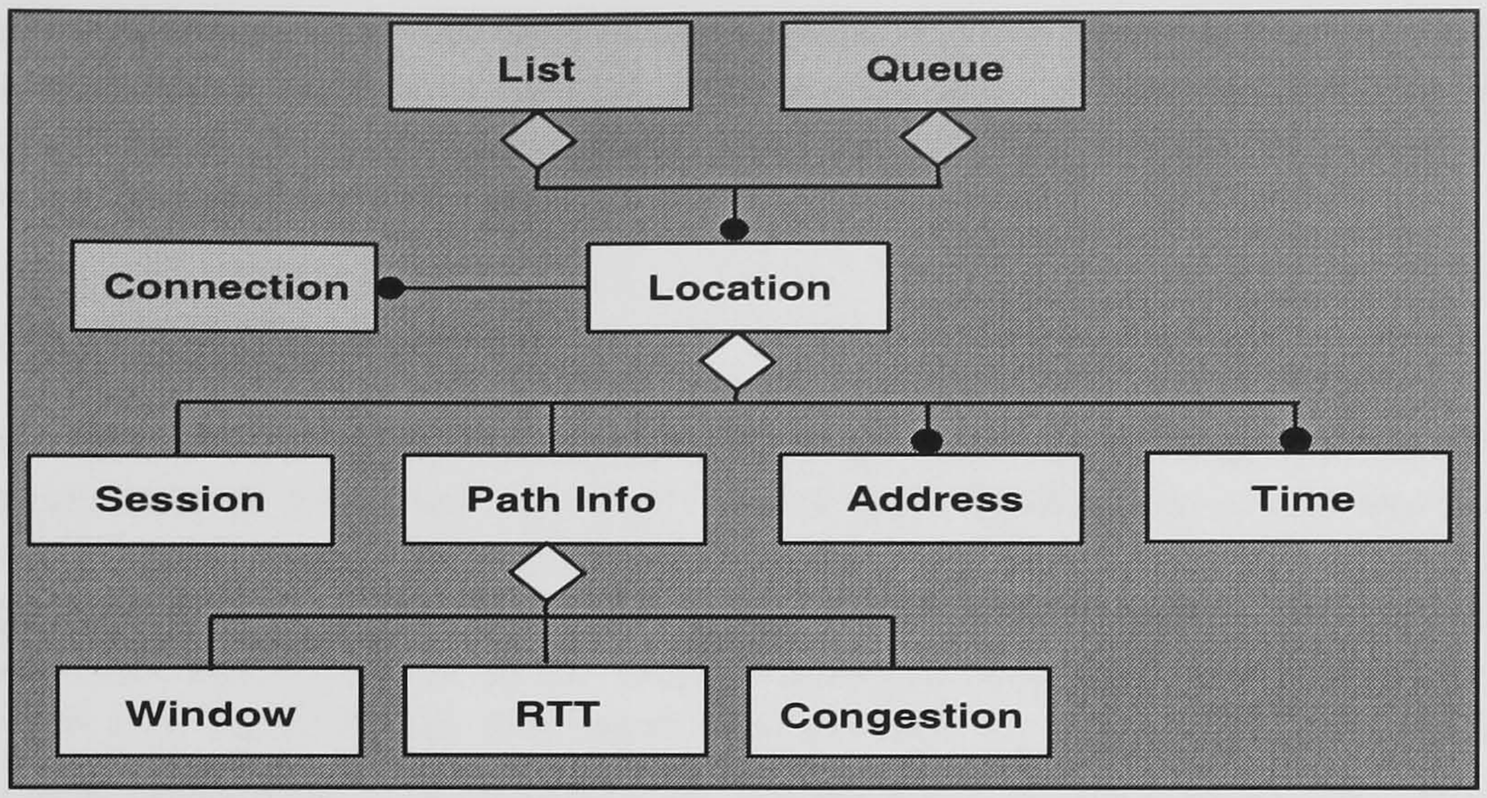


Figure 5.12: **Object Model for the Location Layer: Each Location object maintains statistics on the utilised window size, the RTT and congestion events.**

A round lasts approximately one RTT. During a round if the connection is not application limited, the amount of data sent is determined by the minimum of the congestion window, offered window and send socket size, allowing estimation of the congestion window from packet traces. From the RTT and the number of bytes transferred the throughput during a round can also be calculated.

5.4.5 Location Layer

The Location Layer builds upon the services provided by the Connection Layer so that Quality of Service predictions for aggregates of destinations can be made available to the Communication Module in a timely manner. To achieve this four interconnected functions are performed; the Location Layer provides a place where the averaging of readings can be conducted, it aggregates readings from hosts that are situated close to each other, it allows the QoS information extracted during one connection to be aggregated with the QoS information from previous connections and it makes the resulting statistics available for inclusion in Location Information Packets.

Figure 5.12 presents the object model of a Location. It situates a Location object within the data structures that are used to access it and shows the main classes of the objects that a Location contains. Access to a Location is obtained, via a hash table of linked lists, through an ordered queue or via a reference held by a connection object. At the start of each connection a hash table is used to efficiently retrieve the appropriate Location Object for the connection, a reference from the connection to the location object is then established and subsequently used to pass QoS readings from the Connection to the Location. The queue of locations is maintained to allow the least recently used location to be removed, when a new location is required and the limit on the number of locations the system will support has been reached.

Contained within each Location are four main classes of objects. The session object keeps track of the number of active connections, and start and finish times of sessions. Multiple time objects are available for various housekeeping functions. The address object contains an IP address and netmask, which together define the aggregation of IP addresses for which traffic statistics are maintained. The Path Information object performs the aggregation and averaging of readings and maintains the actual statistics. Each Path Information object itself contains three further objects the Window, RTT and Congestion objects, which respectively maintain statistics on the utilised window size of contributing streams, the round trip time to be expected on the path and the expected level of congestion.

In the remainder of this section, the level of aggregation and the methods used to convert readings taken by the connection layer into estimates of network and connection characteristics are discussed. Methods for maintaining appropriate measures of ICNs, Round Trip Times and utilised window sizes are all considered in turn.

Level of Geographic Aggregation

There are two directions in which the Location layer performs spatial aggregation of connections. Local sources are treated as a single aggregate and remote destinations are grouped together into multiple distinct aggregations. It is assumed that the local network is well engineered and that the LIS is positioned so that that there is minimal delay and congestion between the local data sources and the LIS. Consequently all the local sources can be treated as a single aggregation. For example the LIS may facilitate the sharing of QoS between a cluster of WWW servers.

The level of aggregation that is appropriate for remote destinations is more difficult to determine. There are several approaches that it is possible to adopt which include; a single global estimate, aggregation of IP addresses whose traffic share common statistical properties, aggregation by network address, aggregation by some properties of the IP address or simply aggregation by the host.

These options form a sliding scale: at one extreme there is a single remote aggregate with a high level of aggregation and consequently a large volume of traffic available for the calculation of averages, but the cost is likely differences in the path characteristics from which the readings were extracted. At the other extreme there is a single host resulting in less traffic to calculate the statistics over but greater likelihood of the results being valid.

There is a wide variability in the expected level of congestion for different paths on the Internet. This suggests that simply retaining a global estimate of the congestion level will consistently provide underestimates to some paths and overestimates to others. For this reason it is desirable to have separate estimates for different destinations.

It is likely that for many groups of hosts it is the connection between the local and wide area network that is a congestion bottleneck. Thus aggregation by RTT, which offers potential in grouping together networks situated geographically close together would cluster together networks with very different congestion regimes. The large size of class A and B networks suggests that they contain groups of hosts that connect to metropolitan networks using links with widely varying bandwidths again resulting in different congestion regimes within the network.

The approach adopted in this work is to aggregate measurements of those hosts which share the common three higher order bytes, this is a conservative level of aggregation which is justified on the grounds that experimentation into higher levels of aggregation did not produce measurements which result in confidence that congestion information sharing is practical.

Implicit Congestion Notifications

The proportion of congestion notifications received over a specified period of time is the metric used to indicate the level of congestion that can be expected on the path to a location. Ideally the proportion that is communicated to a host at the start of a session will be the same as the proportion received during that session.

A number of factors limit the extent to which this ideal can be approached. The course granularity of congestion feedback and bursty nature of TCP traffic means that for a given level of congestion the feedback to separate hosts sharing the same congested route will be different. Secondly, network conditions may change between the collection of statistics and the commencement of the connection to which they are applied. Indeed network conditions may change during the lifetime of the connection.

Of particular significance for the variation of network conditions over time is that there is a strong diurnal pattern of network usage, consequently statistics gathered at one time of day may not be directly applicable at another.

It is however envisaged that the congestion estimates supplied by the LIS will be used as a supplement to the current connection confined congestion control schemes, rather than as a replacement to them. Measurements presented in Section 4 suggest that satisfactory results may be obtained, even if the estimates themselves are poor.

Consider four cases:

1. A small connection with an underestimate of available bandwidth — doing congestion avoidance with a starting window of one, is not seriously less aggressive than doing Slow Start with a congestion window of one until several rounds into a connection.

2. A small connection with an overestimate of available bandwidth — will do little damage to the network as only a small amount of data is transmitted.

3. A large connection with an underestimate of available bandwidth — at the start of the connection the window may be too large. Feedback should however force the connection back to its fair share.

4. A large connection with an underestimate of available bandwidth — whilst throughput at the start of the connection will be depressed, the average effect on the connection is reduced the longer the connection is.

There are numerous approaches that are available for maintaining an average level of congestion. The approach adapted here flows from consideration of a number of characteristics that it is desirable for the average to have. It desirable to be able to control the time span over which the average is calculated, too short a time span would make the average susceptible to short term network transients, too long a time span might invalidate the applicability of the average by including data from a time of day with very different traffic characteristics.

It is also desirable that the average be calculated over a sufficient number of readings. For example if a 1% level of loss is to be registered the average must be calculated over at least 100 packets. It is therefore desirable to maximise the number of packets over which the average is calculated.

Thirdly, it is desirable that the start time, end time and amount of data used to calculate the average be known so that the statistics may be stored in a long term traffic data repository, which is further discussed in Chapter 6.

There are a number of alternative ways an average could be kept; for example an exponentially weighted average may be maintained. This would have the advantage of cheap updates and only requiring the maintenance of one number. Unfortunately the number of packets over which the statistic had been calculated would not be clearly defined nor would the time span for which it applied. The maintenance of a collective congestion window mirroring TCP's congestion control mechanism would also be possible. Each congestion notification received would result in a multiplicative decrease of the congestion window and each packet transmitted in an additative increase. This has the disadvantage of calculating a fair window size rather than the level of congestion, thereby limiting the ability of the host to determine how to respond for a given level of congestion, The simplest approach is to keep count of the number of data packets and the number of congestion notifications to a destination. The expected level of congestion can then simply be expressed as the ration of congestion notifications to data packets. This has the disadvantage of not supporting the removal of old readings from the calculation of the average.

The approach adapted is to calculate the proportion of congestion notifications over a known period of time and for a known number of data packets. Two slots are used. For each slot the start and end time, number of data packets and number of congestion notifications are recorded.

The average level of congestion is calculated by dividing the sum of congestion notifications for both slots by the sum of data packets.

A minimum number of packets and a minimum time span are defined. When both have been reached the most recent slot is retired, the data being transferred to a the long term Traffic Data Repository (TDR) discussed in Chapter 7. A new slot is then started. This overcomes the problem of averages being calculated over too little data when an average is taken at the start of a slot.

The approach adopted has a number of advantages. The number of packets over which the average was calculated are known, as are the number of congestion notifications. The start and finish time of the measurements are both known. It is possible to specify a minimum time span over which the statistics are calculated. It is also possible to stipulate a minimum number of packets over which they are calculated. The effectiveness of this method of measuring the average level of congestion is evaluated using experimentation in Chapter 6.

Round Trip Times

Unlike congestion where a single notification indicates a congestion event but by itself does not indicate the level of congestion a round trip time reading with a timestamp attached to it communicates a quantified sample. Individual readings are stored in the Traffic Data Repository (TDR) and an exponentially weighted moving average is used for maintaining the average level of congestion. The algorithm used in maintaining this smoothed round trip time is the same as that used in TCP as discussed in Chapter 3. The variance in RTT is also maintained using the TCP algorithms.

The main difference between the way the TCP and the LIS maintain smoothed RTTs is in the way that the measurements are made and the aggregation of measurements from concurrent and consecutive connections. Using an accurate timestamp and filtering readings to exclude delayed acknowledgments increases the accuracy of estimation.

The variation in network Round Trip Time between two locations, should be bounded by a maximum and minimum value. Even when the current round trip estimate is out of date, the minimum and maximum round trip times may still be of interest. The maximum can provide the basis for setting an upper bound on the initial RTO value. The minimum gives the minimum amount of time that will be available to pace out a window of packets. Either may provide useful points to start the evolution of round trip time estimates.

A problem with absolute minimums and maximums is that extreme values remain even if they are a freak rather than the usual maximum or minimum. The maximum RTT is slowly aged by reducing it by a small fraction when the RTT reading is smaller than the current maximum. When the reading is greater than the current maximum the maximum is set to the current reading. The minimum RTT is aged slightly for larger readings and set to the current reading when the reading is smaller than the current minimum.

Connection Throughput

The maximum utilised window size achieved while an estimate is being made is useful for hosts to know, as it indicates the stress that was put on the network. Where bandwidth and the level of multiplexing are low it is possible for connections that are bandwidth or application limited to receive very few congestion notifications, which could be interpreted as indicating a large initial window is appropriate, when in fact it is not.

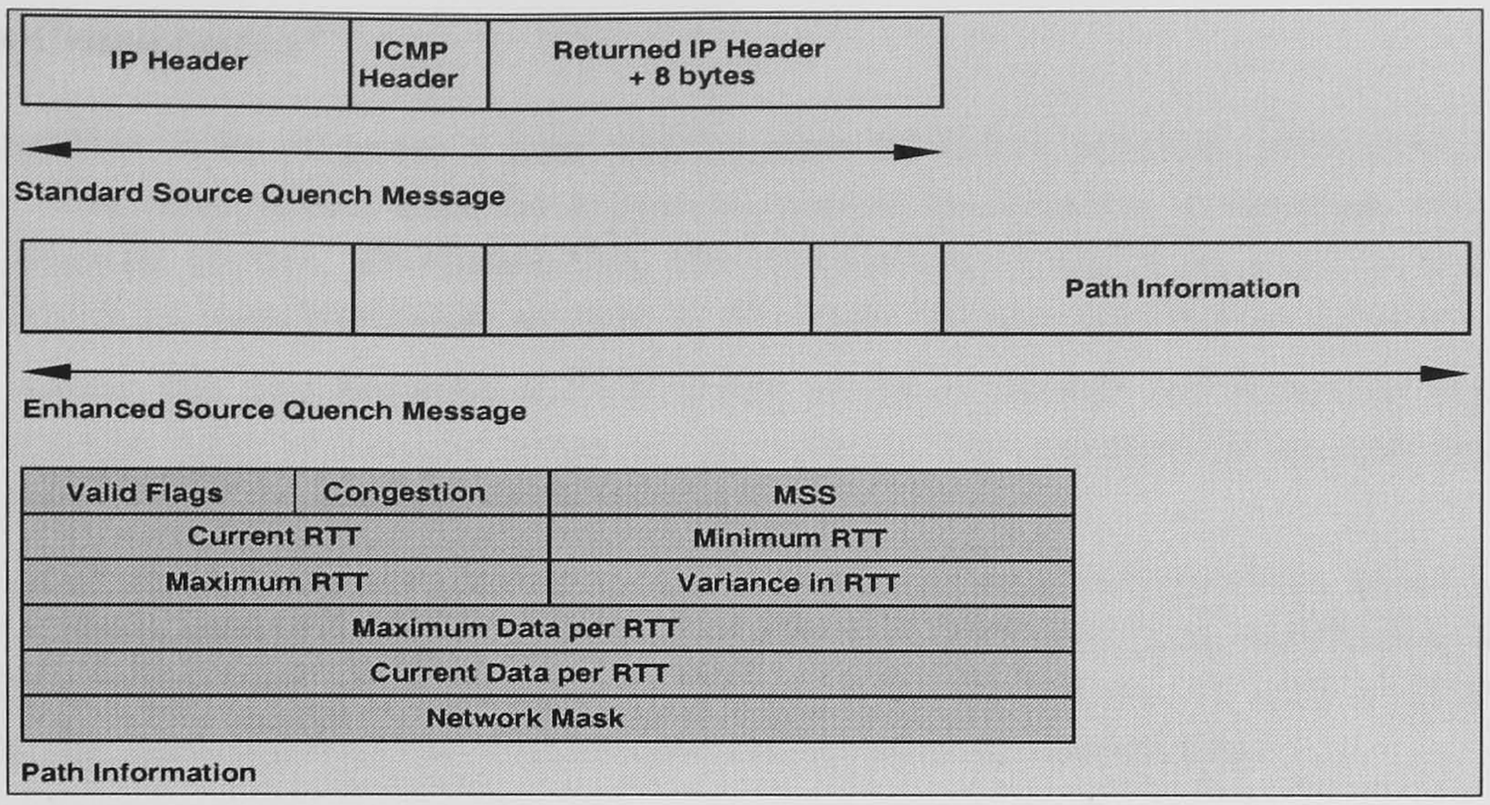


Figure 5.13: **Location Information Packet: Information is communicated from the LIS in information packets, which are simply as extension of the ICMP Source Quench Packet.**

5.4.6 Location Information Dissemination

To be utilised in controlling traffic, the information gathered by the server needs to be communicated to hosts. It is desirable that this communication, does not introduce delay into packet forwarding, uses the minimum network resources possible and arrives at hosts when it can be most usefully deployed.

This section discusses how the goals outlined above have been met and presents an implementation. The discussion is structured into the following subsections. The first section addresses the content of what needs to be communicated, next the temporal constraints on communication are identified, this is followed by consideration of the types of destination and how the communication should be conducted.

Content of Communication

From the functional decomposition defined in Section 5.3 and the data a host needs to decide on initialisation discussed in Section 5.4.1 it follows that the actual values for `cwnd`, `ssthresh`, the SRTT and other parameters should not be calculated in the server and communicated to hosts but that the raw network and data stream characteristics should be communicated. Figure 5.13 shows the fields the content of each is defined below.

1. Flags: These indicate which fields are valid.
2. CNP: Is the probability of receiving a congestion notification.
3. SRTT: The smoothed round trip time.
4. MinRTT: the minimum round trip time.
5. MaxRTT: the maximum round trip time.
6. Var: The current variance in RTT.
7. The maximum window size observed on the path.
8. The recent average throughput observed on the path.

Timeliness Constraints?

Once a connection is in progress a host will be able to derive network characteristics from the data stream. The most useful time for a host to receive an estimate of network characteristics is before data transfer occurs. To ensure that the information received by hosts is as up to date as possible it should be communicated as near to the start of data transfer as possible.

For a TCP server the gap between a SYN being received, which indicates the setting up of a connection and the start of data provides an opportunity for communication and initialisation to take place close to but before data transfer starts.

The LIS needs to monitor passing traffic to determine network characteristics, when a SYN packet is observed a check is performed to see if it is the first SYN for that connection and to see if the local host is registered as requiring information from the server. If both of these checks are positive then the host is informed of the expected network characteristics.

What Should Communication be with?

It is the entity, which is responsible for controlling the flow of data from the Local to the Wide area network that can make use of information about expected network conditions. The focus of this dissertation is on congestion control in TCP and in this case it is the routines that initialise TCP start-up parameters to which the information needs to be delivered.

The recognition that the Internet can no longer solely rely upon TCP for its congestion control, has lead to a number of proposals for congestion control located outside of the TCP stack.

There have been proposals for Multicast congestion control, for congestion managers which co-ordinate congestion control across protocols and connections and for congestion control to be embedded in applications that provide video and sound streams.

Communication with a range of congestion control entities is facilitated, by leaving the mapping from network characteristics to control parameters to the host. Unfortunately the benefits of this design cannot be fully realised, whilst network characteristics are only extracted from TCP streams. The introduction of network level signaling of Congestion, RTT and connection set up would allow network characteristics to be derived from non TCP streams and facilitate the timely communication of such characteristics to non TCP hosts.

Communication Mechanism

Information could be carried from the Server to a host in two ways. It could be piggy backed on an existing packet or a control packet could be generated by the server.

1. The advantage of piggy backing is that no extra packets are generated. The disadvantage is that delay may be introduced to all SYN packets as checks are performed and control information is inserted. Modification to the packet headers will also be necessary.
2. Whilst, control packets have the disadvantage of requiring new packets, no interference with the existing packet streams is required nor is there any need to alter packet header formats.

The use of congestion control packets on the Internet has been criticised in the context of Source Quench for using network resources at the point and at the time that congestion occurs. Similar criticisms have been made of PPT, which proposes using control packets to probe the network for bandwidth. It has been shown in Section 3.2.1, that in the past Source Quench was used to effectively control congestion, however problems of scale and the need for a consistent host and server implementation remain.

The use of control packets proposed here does not suffer from the problems associated with Source Quench, for two reasons. Their generation does not synchronise with congestion and only one packet is required for each connection. Secondly it is envisaged that their scope would be limited[5] to the Local Area Network environment, therefore WAN resources would not be used at all.

Summary

A new connection is detected by observing SYN packets; if the local host is registered with the LIS and the LIS has information on the remote host a LIP is sent to the local host. This procedure reduces latency and LAN traffic when compared to an explicit request response protocol. In the event that location information is required during a connection, or by a non TCP data stream, an interface for explicitly requesting LIPs is provided.

The LIP is implemented as an extension to the Source Quench Packet. Applications can access LIP packets by listening on a raw socket and in the Host Implementation LIP packets are passed to the TCP connection where their information is used to dynamically initialise control variables.

5.4.7 Host Interpretation and Actions

This section discusses the use to which the information contained in an LIP packet could be put by a host. The discussion is split into three sections; first the constraints on when it would be useful to maintain an LIP are discussed, next adaptations to TCP's start-up algorithms which are designed to improve TCP's congestion control mechanisms are introduced, finally issues relating to a host implementation which allows the proposed modifications to be evaluated are described.

Time Constraints

Figure 5.7 shows the signal sequence diagram for a client requesting a simple web transfer. There is over one RTT between the LIS observing the first SYN request and the server beginning the transmission of data. This time period can be utilised by the LIS to retrieve statistics for the clients location, dispatch an LIP and for the server to use the LIP's information to intelligently initialise appropriate start-up variables.

It is undesirable for the LIP to arrive before the TCP control block has been created as this would complicate the initialisation process. In the eventuality of it occuring TCP would revert to its normal static initialisation and Slow Start consequently there is no negative effect. In practice it was found that this was not a problem,

If the LIP arrives after data transfer has begun there are two possible effects. That the transfer takes longer to complete than if the LIP had arrived on time due to unnecessarily small initial windows. Validation of the LIS Host interactions found that the time from observation to initialisation was between 1-10ms on 10 MBytes per second Ethernet, With the bulk of this time being taken waiting in the BPF for a timeout. At this time-scale it would be connections with a small RTT that would be affected by late reception of an LIS. The additional delay for these connections is not significant. The second danger is that the LIP arrives after congestion notifications have started arriving at the host and initialisation by the LIP results in overly aggressive behaviour. On many TCP implementations the default initial RTO is set around a second, which is orders of magnitude greater than the delay expected for an LIP. Consequently this source of congestion notification is unlikely to be a problem. For a congestion notification to be received as three duplicate acknowledgments the connection would have had to have lasted for at least three RTTs. By this definition the probability of a late LIP is significantly reduced. A simple way to guarantee a late LIP does not increase the congestion window in the face of congestion notifications is to

[5] By setting the time to live field.

introduce a congestion notification received bit to the TCP control block, which if set prevents the LIP being utilised to increase the congestion window.

Congestion Window Initialisation

The two parameters which control the evolution of the congestion window are `cwnd` and `ssthresh`. This subsection addresses the value that each should be initialised to given that the likely proportion of ICNs is known.

The TCP fair equation, where W is the fair window size C a constant term and p the ICN probability, is used to calculate an Initial Window size (IW).

$$W = \frac{C}{\sqrt{p}}$$

C has been estimated analytically as lying between 0.87 and 1.31 [MSM97]. Under deterministic loss conditions TCP's window size will be bounded by $\frac{2}{3}W$ and $\frac{4}{3}W$, which shall be referred to as the Lower and Upper Bounds respectively. The IW is set to the Lower Bound giving a conservative starting size, which [ZQK99] shows is optimal for minimising the number of rounds it takes to complete a connection. The `ssthresh`is also set to the Lower Bound, meaning that the first SS phase of the connection is bypassed and Congestion Avoidance is applied from the start.

The proportion of ICNs is not sufficient for determining the initial window size for the following reasons. Loss maybe low, but the data stream may be less than the number indicated by the connections fair share. This can occur when the connection is application limited, window limited or the size of the connection is not large enough for the fair window size to be reached. The number of packets lost is not independent of the size of the data stream.

One solution is to set the initial window size to the minimum of the value derived from the ICN and the recent maximum window size, and set `ssthresh` to the maximum of the fair window size and the largest recent window size. With this policy the initial window is set to a conservative value but is allowed to increase exponentially to the larger value. If there is a lot of data flowing the values for `ssthresh` and `cwnd` should converge.

Implementation

The above procedures have been implemented by modifying the network stack on the FreeBSD 2.2.7 kernel, which used the BSD reference implementation for Reno TCP. Location Information Packets that are received, by the IP layer are passed to icmp_input, which in turn calls the ctl_input function. Finally a new function is called, which locates the TCPCB for the connection whose SYN caused the LIP packet to be sent and uses the path information to initialise the micro state.

The LIP is also passed to rip_input and from there is made available to applications listening on raw sockets. This allows congestion aware applications to make use of the information contained in a LIP packet.

Summary

In the case where the calculated IW is larger than the recent maximum used window size, which may occur if the connection was application or receiver window limited the congestion window is set to the recent maximum window size. However, `ssthresh` is still set using the ICN information. Thus `cwnd` is set to a safe size but is allowed to increase exponentially as there is no positive information suggesting that the congestion information is incorrect.

5.5 Conclusion

This chapter opened by drawing upon the findings in Chapter 4 to identify areas where Internet Congestion Control can be improved. It then proposed measures that address these issues, discusses how these proposals relate to the datagram paradigm that lies at the heart of the Internet's architecture and introduces an implementation of a Location Information Server.

It has been observed that the small size of many TCP connections means that there is little opportunity to adapt to network conditions. This makes it important to have a good initial value for control variables. Given the wide range in levels of packet loss and Round Trip Times, that can be expected on different network paths, it is impossible to have a default initial window or RTO value that suits all paths. This leads to the observation that initialisation to values based on past traffic may be beneficial[6].

From these observations it can be concluded that it is important to address the question of initialisation, for connections performance to be improved. This can be characterised as the addition of Congestion Control at the *Session Time-scale* .

To realise QoS guarantees, congestion control at the session time scale has usually taken the form of admission control combined with resource reservation. A problem with the application of admission control at the local wide area network interface is that the point of congestion may occur somewhere else in the core of the network. In the absence of such facilities best effort session level congestion control could be effectively implemented by following a functional division of labour which allowed congestion to be detected at the point were it occured and for congestion information to be communicated to the data sources, which were responsible for regulating their behaviour in response to the signal.

The confinement of adaptation to the congestion within a connections lifetime limits the effectiveness of the above division of labour. This chapter advocates the introduction of a Location Information Server, which passively monitors network traffic and remembers the signal received by past connections, to enable the dynamic initialisation of new connections. It has been argued that the underlying datagram architecture of the Internet has lead to an absence of network level signaling, which limits the effectiveness of such a proposal. Nonetheless it has been shown that sufficient information can be extracted from TCP headers to allow the demonstration of the effectiveness of the Location Information Server.

The second half of this chapter presented the design and implementation of a LIS, a mechanism for facilitating communication between the server and hosts and a host implementation that allows experimentation to be supported.

The Location Information Server is structured into four main components as illustrated in Figure 5.9. An input module captures packets and passes them to the connection layer. The connection layer maintains Macro and Micro TCP state for each active connection, takes readings and detects the initiation of new connections. These readings are passed to the Location layer, where estimates of congestion, RTT and bandwidth utilisation are maintained for destinations. When a new connection is detected, a Location Information Packet (LIP) containing predictions of network conditions is sent to the local host by the communication module.

TCP's Macro State defines the state transitions that a connection passes through during its lifetime. Maintaining Macro state, at the monitor, allows memory to be reclaimed, when a connection closes and tasks to be scheduled, for particular points in a connections lifetime. The monitor is situated between the communicating hosts, meaning a different set of state transitions are seen compared to the endpoints. These modified state transitions are maintained by the monitor.

TCP's Micro state refers to the variables necessary to track the progress of data transfers. For example the last acknowledged byte and the next byte to send. The monitor uses the abstraction

[6]A large proportion of packets belong to long connections, for which the initial value is less important, this suggests that the stability of the network is unlikely to be sensitive to initial control parameter values

of a round, which lasts from a packet being sent, until its acknowledgment is received, to obtain readings for the used window size, the number of bytes sent in a round and the duration of a round[7]. If a round contains one or more retransmitted packet this is counted as an Implicit Congestion Notification.

The destination layer maintains estimates for each location, where a Location is defined as the collection of hosts that share the same three higher order bytes in their IP address. The standard TCP low pass filter is used to maintain a `srtt` estimate and the TCP algorithm is used to maintain a `vrtt`. In addition the maximum and minimum round trip times to a destination are maintained.

The proportion of ICNs is calculated as I/D where I is the number of ICNs and D the number of Data Packets. A current and recent past total is maintained for the sum of ICNs and the sum of data packets. Once a threshold number of packets have been observed the recent totals are made equal to the current totals and the current are reset, provided the totals have been accumulated over a minimum time span. The time span and threshold are both design parameters.

A mechanism has been defined which enables the timely delivery of the above predictions to hosts. Bandwidth utilisation is kept to a minimum, whilst the communication of detailed information is facilitated.

A host implementation has been presented that facilitates the initialisation of TCP control variables by applications, a Location Information Server and defines a mechanism for variable values to be carried over from one connection to the next.

The server and host implementation form the platforms that are used for the live Internet experiments presented in the next chapter.

[7]Delayed acknowledgments and the retransmit ambiguity are both accounted for.

Chapter 6

Evaluation of Congestion Window Initialisation

6.1 Introduction

From 1988, the Internet has relied upon the congestion control mechanisms in TCP [Ste97] to prevent a recurrence of the congestion collapse experienced in the mid [Nag84b] to late eighties [Jac88]. Since then assumptions that were made, about the nature of Internet traffic, have been undermined by the tremendous growth of the World Wide Web. It used to be the case that traffic could be categorised as either bulk transfer, which was insensitive to delay, or interactive, which had low bandwidth requirements but was sensitive to delay. Today the Internet is dominated by web traffic, which is sensitive to delay and has bandwidth requirements that lie between the old bulk transfer and interactive categories.

Chapter 5 presented the design and implementation of a Location Information Server (LIS), which is capable of monitoring traffic and providing feedback of the expected level of congestion to hosts. The congestion information can then be used by a host, to allow it to start transfers with a window size appropriate for the available network resources. For the approach to be effective it is necessary to be able to predict the level of congestion and translate that prediction into initial values for the appropriate control variables.

This chapter presents an assessment of the feasibility of such predictions and translations. It opens by presenting a methodology, which utilises live Internet experiments. This is followed by an analysis of the accuracy with which congestion can be predicted using past traffic. Having established the strengths and weaknesses of congestion prediction the issue of how TCP should map from a congestion prediction to an Initial Window (IW) size is discussed. The consequences of dynamic window initialisation are then explored, before concluding the chapter.

6.2 Methodology

This section presents the methodology used to conduct the evaluation. It opens with a discussion of the service provided by TCP and the metrics that are appropriate for the experiment. This is followed by a consideration of the parameters to the system and the factors that need to be accounted for. Having established the metrics and factors, possible evaluation techniques are discussed and motivation is given for the ones that are deployed. The experimental apparatus is also specified in this section. Next a discussion of issues relating to and a specification of the workload used and a description of the data generated is presented.

6.2.1 Services and Metrics

TCP provides a reliable byte stream service: if all the bytes of a file are received the transfer is successful, otherwise it is not completed. The proportion of successful completions is a possible metric, however this is unlikely to successfully detect small differences between configurations, as the robustness of TCP means that only a very small proportion of connections do not complete.

The amount of time that it takes to complete data transfer is a significant component of the delay experienced by the user, it therefore presents itself as a metric for assessing the effectiveness of a configuration from the perspective of an individual user and will be discussed next. Following on from the discussion of delay, window size, burstiness and congestion feedback are considered as possible metrics.

Delay

The transfer time of a TCP connection is made up of two phases, connection set up (the time for SYNs to be exchanged and acknowledged) and data transfer. The time taken to complete the data transfer phase can in turn be divided into two components. The time spent waiting for RTOs and the time spent transfering data. Research [BPS+98] has shown that a large proportion of packet losses (as high as 46%) result in an RTO, which often lasts a second. This component of data transfer, even under moderate levels of loss can dominate the total delay. Furthermore the effect of retransmit timeouts can be attributed to TCP's error recovery mechanism rather than the congestion control algorithms. Numerous techniques have been proposed, which reduce the delay caused by RTOs; Implicit Admission Control [MPCC00, Flo96, GT95], Forward and Selective Acknowledgments [MM96, FF96], New Reno [FH99] and Fast Recovery Strategies [LK98], Random Early Detection Gateways [FJ93a, LM97] and Explicit Congestion Notifications [Flo95, Kri98, RF99, CKLT97, FF98] and will be discussed further in Chapter 8. A reduction in the frequency of RTOs, will increase the significance of congestion window regulation as a factor in determining the delay suffered by connections in the future. For these reasons, although delay is of interest, the absolute delay suffered by a connection is not an appropriate metric for accessing the effectiveness of TCP's congestion control algorithms or the effect that dynamic initialisation of the congestion window would have on them.

The remaining component of delay is the amount of time that a connection spends actually transfering data. For a given round trip time, this will be determined by the number of rounds that a connection takes to complete. From an individual user's perspective the fewer rounds the better. The number of rounds to complete is therefore a metric which relates to the utility a user receives from the network.

When demand for bandwidth exceeds availability a mechanism for fairly distributing the available bandwidth amongst competing users is necessary. The current Internet consensus is that the *average window size* of a TCP connection during its steady state and at a given level of loss represents such a fair share [MF97]. Whilst it has been questioned whether this leads to an optimal use of resources [Wis99], or is even the best benchmark of fairness [HS97], since this investigation operates within the paradigm of TCP congestion control it is the one that is adopted.

The data transfer phase of a TCP connection may be subdivided into start-up and steady state periods with some transitional period between the two. The focus of this dissertation on initialisation of control parameters, leads to the desire for a metric which quantifies the aggressiveness of the start-up phase. The peak window size before a loss occurs, gives the maximum load that is placed on the network during start-up and is used for this purpose.

Burstiness

It has been shown analytically that the variability in throughput is important in determining whether packet loss occurs [Wis99].

Ideally the packets within a window of data would be evenly distributed over the entire window. If they are clumped together loss is more likely and network resources will be wasted by retransmissions. This is the motivation for the self clocking of packets using acknowledgments [JB90]. Although initialising the congestion window to a larger value will prevent the ACK clock from working at start-up, it has been shown that there are other problems with relying on the ACK clock to shape traffic[1] which taken together can be characterised as the failure of Implicit Traffic Shaping. This has led to the investigation of supplementary [KKP00, aRHK98] and alternative [ZQK99] ways of shaping traffic that would work with larger IWs [AFP98a]. These issues are discussed further in Chapter 8, and are not addressed by this evaluation.

Congestion Feedback

Packet loss is the main way the network signals congestion to hosts. In assessing the global fairness of different set ups the effect on other traffic is important. If there is a systematic increase in the Implicit Congestion Notifications (ICNs) received for one configuration against another it can be concluded that congestion has been caused and it is likely that other traffic is suffering. Conversely if the level of congestion notifications is significantly less than for other configurations, it can be concluded that less congestion is being caused.

The comparison of ICNs received by different configurations is therefore used as a metric for assessing their congestion friendliness.

Assumptions for Experimental Work

During the experimental work the use of the Internet for the conduct of the experiments means that a number of the assumptions made during the simulation work could be relaxed. The assumption that:

1. the application is not application limited is maintained
2. there is always data to send is maintained
3. loss is deterministic is relaxed. No assumption about the distribution of loss or model of traffic is made or required as the experiments were run over real service networks and therefore with real traffic in the background.
4. loss is shared equally between concurrent connections is relaxed
5. network conditions remain stable for the duration of a connection is maintained in calculating averages.
6. It is no longer assumed that the connection is not limited by the receivers window. In fact the inter-relationship between Flow and Congestion control algorithms is an important subject of this study.
7. It is no longer assumed that start up behaviour has little effect on the average behaviour of a connection.

[1] ACK Compression [ZSC91, Mog92], bursts during Fast Retransmit [Hoe96], Slow Start After Restart (SSR) [Hei97] and failure to fill the pipe

8. It is no longer assumed that RTOs will not be necessary to recover from loss, as under normal conditions a signicant proportion of loss events result in RTOs. However here we are interested in congestion and flow control but do not wish to focus on the effect of error recovery. In these experiments measuring the duration of a connection in rounds to complete rather than seconds removes the effect of error recovery from the results.

Summary

This section has discussed metrics relating to delay, window size, burstiness and congestion feedback. Four metrics have been selected which are summarised below.

Number of Rounds to Complete Summarises the component of the delay that is attributable to congestion window evolution.

Fair Window Size Is the socially acceptable window size and provides a benchmark with which comparisons can be made.

Peak Window Before Loss Provides a measure of the aggressiveness of start-up.

Proportion of ICNS to Data Packets Measures the congestion experienced by a connection and systematic differences between configurations indicate their congestion friendliness.

6.2.2 Parameters and Factors

There are a wide number of factors that affect the behaviour of TCP, all of which need to be considered when designing a performance evaluation [All99] experiment. Each can be placed into one of three categories: end to end, network issues and the parameters of the monitoring system itself.

End to End parameters include: the Maximum Segment Size [All00], the offered window, whether delayed acknowledgments [Joh95] are being used and the algorithms that are used to control the congestion window [Ste97]. In addition there are several experimental implementations of TCP, such as SACK [FF96], FACK [MM96], New Reno [FH99], Vegas [BOP94, DLY95] and ECN [RF99] enabled TCP.

Network considerations include: the distribution and volume of competing traffic, whether the routers employ a variant of active queue management [FJ93a] or are drop tail, and the level of congestion feedback received by hosts.

The parameters for the monitoring system include the functions for arriving at a prediction of ICNs and for translating from predicted congestion to the IW size. Finally the volume and age of data from which predictions are made is likely to be significant in affecting accuracy.

The number of parameters makes it necessary to identify the most important ones. To this end the parameter list is divided into two, those parameters that are fixed for the duration of the experiments and those for which different levels are used.

Of the experimental TCP implementations listed above, only Vegas [BOP94] specifically addresses start-up behaviour and it is primarily of historic interest. Consequently the experiment uses the Reno reference implementation of TCP, both as a benchmark and as the basis for an experimental implementation. TCP Reno employs the Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery algorithms [Ste97] for controlling the congestion window.

A maximum segment size of 1460 bytes is used, this is the most common MSS used on the Internet today[2]. An offered window size of 32768 bytes is used, this is larger than the average window size used on the Internet, as shown in Figure 4.13. Using an atypical value is justified because it

[2]The alternative of using 536 byte sized segments is declining in popularity

will allow more room for the dynamics of congestion window evolution to be explored. Delayed acknowledgments are also used, this is the default behaviour of most TCP implementations.

There are two reasons why it is possible to estimate the behaviour of small connections from measurements of long connections.

1. Future traffic will not have an effect on current traffic. Consequently it can be assumed that the behaviour of a connection for the first n bytes will be the same whether there is a further megabyte to transfer or whether there is no further data to transfer.
2. TCP algorithms operate on a RTT cycle. If there is no congestion in a RTT the congestion window is increased by one segment. If there is congestion the window is reduced by half. The gaps between connections are designed to allow sufficient time for these algorithms to stabalise between connections.

Deducing the behaviour of short connections from long connections has the advantage of allowing sufficient feedback to be received to be able to make some statements about the state of the network. Whilst experimental traffic based upon lots of short connections may more closely approximate actual traffic it is not clear that deductions about the underlying state of the network could be made. Consequently it would be harder to compare actual to desired behaviour.

This is the cost of moving the experimental work out of Simulations and the Laboratory and utilising the Internet. The benefit is that the measurements are made against a background or real traffic, real levels of multiplexing and across real networks.

The focus of the research is the evaluation of the Location Information Server, for this reason it is important to be able to evaluate different mechanisms for predicting levels of ICNs and for mapping predictions to IW sizes. The age and volume of data with which these predictions are made and the level of congestion are both likely to have a significant effect on the system and the metrics used to evaluate it. These observations lead to three main factors; the level of congestion feedback, the age of traffic used to make predictions, the method of window initialisation (Static or Dynamic).

6.2.3 Evaluation Techniques

This section discusses the evaluation techniques used, describes the experimental apparatus and specifies the initial configuration.

Choice of Techniques

The implementation of an experimental Location Information Server, as presented in Chapter 5 makes evaluation through the measurement of experimentally controlled traffic an obvious choice of technique. The availability of the Discard Service on many Internet hosts, facilitates the use of the Internet as the test bed. This choice has a number of advantages and dangers.

The difficulties that arise when trying to simulate the Internet are well document [PF97a]. In particular the failure of Poisson [PF97b] modeling and the absence of a convenient alternative method of generating workloads for simulations means that the significance of results is not always clear. The size of the Internet means that very large scale simulations are required to emulate the level of multiplexing that occurs in the core of the network. Many simulations therefore use topologies that are unrealistic, yet [Wis99] suggests that assumptions about multiplexing can greatly simplify traffic management problems.

The use of the Internet as the test bed avoids both these problems. The background traffic used is the actual traffic on the Internet, consequently there is no need to model it. Similarly the multiplexing experienced by the experimental traffic will be the same as that experienced by real

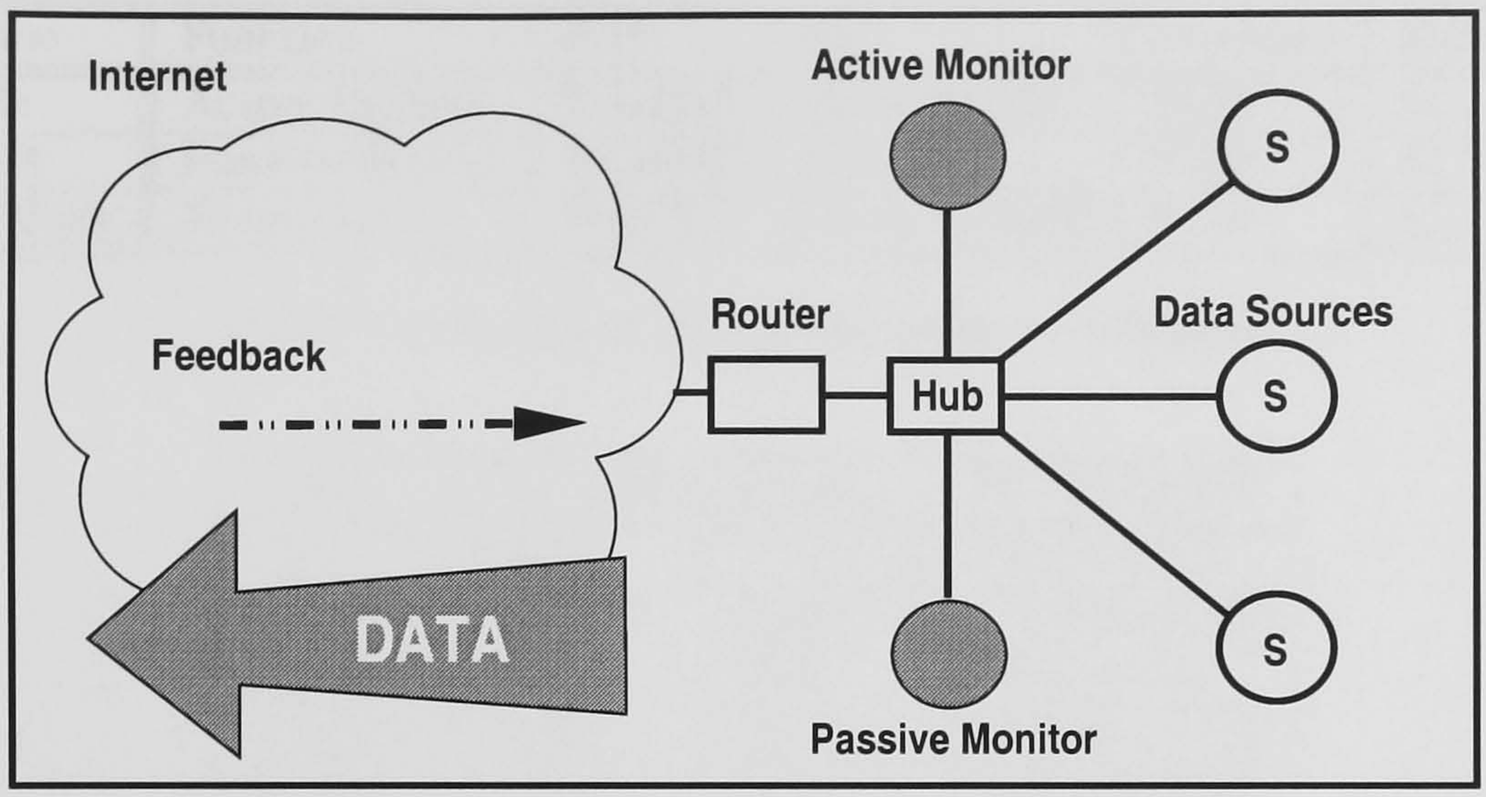


Figure 6.1: **Experimental Apparatus: An active monitor was used to passively measure network conditions and actively provide feedback to the data sources. A passive monitor was also used to collect traces of experimental traffic for latter analysis.**

traffic. Thus the need for very large scale simulations and for simplifying assumptions about traffic distributions are both avoided.

There are however limitations and difficulties associated with the technique. Firstly, the generality of results obtained by experiments across a limited number of paths is open to question. In part this is offset by the choice of metrics, which are independent of RTT.

Secondly, the number of factors under study would require a large number of runs for a full factorial analysis. This is compounded at low levels of loss by the need for a relatively large transfer size to be able to calculate a meaningful average for ICNs received. The use of a service network for experimentation brings with it a responsibility on the experimenter to control the amount of traffic generated, so that the normal operation of the network is not disrupted.

The above considerations were reconciled by supplementing live experiments with trace analysis. Traces of traffic captured during the experiments were rerun against different settings for the monitor and host. In addition the traces are filtered to provide different levels for the age and volume of the traffic used to make predictions.

Using trace analysis, the accuracy of prediction and initial window size that would result from various configurations can be reconstructed. The number of windows it takes to complete a connection and the peak window size are more difficult to recreate with confidence.

In summary, use of experimental measurement, trace analysis and a simple analytical model, when combined with appropriate metrics allows results to be obtained from a controlled amount of data for a sufficient number of factors and levels.

Apparatus

This section presents the apparatus used. Figure 6.1 is a diagrammatic representation of the main elements, the function of each is defined below.

Location Information Server: The LIS monitored the experimental traffic, which it used to derive estimates of congestion for each destination. When the LIS detected a new connection a Location Information Packet, which contains predictions of congestion levels, was sent to the local data source.

Name	Function	O/S	NIC	Processor	Memory
siple	Active Monitor	FreeBSD	3Com 3C900	P133	64
bylot	Passive Monitor	FreeBSD	Intel 100	P133	32
chatham	Data Source	FreeBSD	3Com 3C589C	P120	32

Table 6.1: **Hosts at the University of Glasgow**

Name	Location	Domain	IP ADDRESS
vega	Spain	cc.opu.es	158.42.49.57
pearl	US	UMDNJ	130.219.69.100
montigny	US	MIT	18.43.0.152
dibemail	Italy	dibe	130.251.51.200
mobis	Czech	snl	193.135.174.3
IVY	US	Cornell	128.253.77.201

Table 6.2: **The experimental data was transferred to six destinations across Europe and America**

Passive Monitor: The passive monitor used `tcpdump` to capture the headers of all experimental traffic, which were then be used for trace analysis.

The Data Source was used to generate traffic. A kernel based upon FreeBSD was utilised, the Net/3 networking code was modified to accept Location Information Packets and use the information they contain to initialise the congestion window. The `sock` [Ste94] program was used to initiate transfers and `traceroute` was used to periodically record the routes to the destinations under study.

Destination The details of the hosts to which the experimental traffic was sent are given in Table 6.2. These were selected from traces of web traffic with the University of Glasgow to obtain a wide geographical spread and range of congestion regimes.

Initial Configuration

An important comparison is made between results from connections that are *dynamically initialised* and those that use the normal *static initialisation.* For static transfers the congestion window is initialised to one segment and Slow Start (SS) is performed in the usual way. For dynamic connections the congestion window is initialised to an estimated fair size and starts using the CA algorithm.

Fields in a Location Information Packet are used to initialise `ssthresh` and `cwnd`. Two fields are utilised, one indicates the level of congestion and the other gives the recent maximum used window size.

The host converts the congestion prediction into a window size using the equation:

$$W = \frac{1}{\sqrt{p}} * C \tag{6.1}$$

Where C is the constant 0.93 defined in [MSM97]. Upon the receipt of three duplicate ACKs the congestion window will be halved in value. This gives $\frac{2}{3}W$ as the Lower Bound (LB) and $\frac{4}{3}W$ as

the Upper Bound (UB) [3].

The Slow Start Threshold (`ssthresh`) was set to the Lower Bound (LB) and the congestion window to the minimum of LB and the largest recent window size. The absence of application limiting behaviour from the experiments means that congestion and the offered window size are the two factors that determine the size of the initial window[4].

The Location Information Server was set to estimate the level of congestion and the recent maximum window size in the following ways:

Congestion: A long term average of the probability of receiving an ICN was maintained, by recording the sum of data packets and the sum of congestion notifications. A current and old total were kept. At the start of each connection the current totals were checked to see if a minimum number of packets had been counted and if the total had been maintained for a minimum amount of time. If both checks are positive the old totals are replaced with the current and the current reset to zero.

For this experiment the minimum number of packets was set to 500 and the time to four minutes. These parameters were motivated by the desire to obtain a long term average, which nonetheless was sensitive to the diurnal changes in congestion levels. It not claimed that the algorithm or parameters were optimal and trace analysis could be used to explore alternatives.

Maximum WS: The recent maximum window sized is maintained by the location information server for each destination and is updated once per round. If the current window size is larger than the maximum, then the maximum is updated to the current window size otherwise it is decremented by a small amount.

6.2.4 How Self Similarity might effect averages

In determining how to engineer networks it is important to have an understanding of the load that is presented to the network. The traditional way of modeling such traffic involves using Poisson distributions. When multiple Poisson sources are multiplexed together the bursts from each source tend to cancel each other out leading to a smooth aggregate flow.

Unfortunately in the nineties a number of studies [PF97b, PF97a, CB97, LTWW93] showed that Poisson distributions fail to accurately capture the behaviour of traffic on computer networks. Instead it has been shown that compmuter traffic shows strong Self Similar characteristics such as:

> *there is no natural length of a "burst": at every time scale ranging from a few milliseconds to minutes and hours, similar-looking traffic bursts are evident.* [SV01]

The evidence that wide area computer network traffic displays self similar characteristics is strong. In [LTWW93] a long term study of Ethernet traffic is presented which shows that local area network Ethernet traffic is Self Similar and that aggregating streams of such traffic, as may occur on a wide area network, usually increases the burstiness rather than leading to the traffic being smoothed. In [PF97a] it is shown that Internet traffic shows Self Similar characteristics between the timescales of hundreds of milli-seconds to some tens of minutes. For timescales larger than tens of minutes the diurnal pattern of network usage becomes significant and at smaller timescales structure imposed by network protocols dominates [PF97a].

In [CB97] it is shown that World Wide Web traffic shows characteristics that are consistent with Self Similarity. This is explained by a number of factors including; the distribution of document

[3] In practice losses will not be deterministic and the congestion window may oscillate above and below the bounds given

[4] These issues are discussed further in Section 6.4.

sizes, the effects of caching and the users "think time". In [SV01] it is shown that the load created by a single connection often shows Self Similar characteristics. This is caused by the relatively long silent periods caused by RTOs. Consequently when levels of loss are low (0.1%) and below long range dependence is greatly reduced.

A major advantage of Packet Switching over circuit switching is that the statistical multiplexing of streams of traffic together may result in higher levels of, and therefore more economic, resource utilisation [Rob74], whilst still enabling Quality of Service requirements to be met. It was expected that statistically multiplexing traffic together would lead to a smoothing of the bursts and consequently that it would be possible to achieve high levels of utilisation without bursts leading to dropped packets, jitter and increased delay.

Self Similarity has a number of consequences for congesetion control. These are most significanat in the situation where one is both trying to uses network resources efficiently and trying to provide a well defined Quality of Service, such as might be required for an important video conference [JDSZ95]. If network traffic is Self Similar this implies that a burst can occur on any timescale, therefore long lived correlated bursts are possible making the combination of efficient network utilisation and avoidance of packet loss more difficult to achieve.

Paxson [PF97a] argues that self similarity can lead to drastic reductions in the effectiveness of deploying buffers in Internet routers in order to absorb transient increases in traffic load. This is likely to have an effect on increasing the level of congestion and meaning that congestion events of different size and duration will occur. When predictions are made about the average level of congestion, there will be periods of time when levels of congestion are significantly higher than predicted. Furthermore, whatever timescale an average is calculated over there will be bursts of traffic within that timescale that exceed the average.

This suggests that it would be beneficial for the Location Information Server to provide some estimate of the likely vaiation in congestion level as well as an average value. However, there are a number of problems associated with predicting variation in congestion which are rooted in the incomplete congestion signal that the LIS receives. (It only receives feedback about a particular location when traffic from its LAN is travelling through it). Consequently the LIS only provides average estimates, the effectiveness of this approach is evaluated in this chapter.

In the simulation work in Chapter 4 a deterministic model of loss is assumed. No attempt is made to generate realistic workloads and distributions of loss. This is justified in that by holding this assumption to be true the effect of varying the flow control window and connection size could be investigated and comparisons with previous studies facilitated [OKM99, MF97].

In the experimental work presented in this Chapter the use of a service network for the conduct of the experiments, means that the experimental traffic will be competing with traffic that displays self-similar characteristics. Consequently, the results and conclusions presented in this chapter address the issue of Self Similarity.

Self Similarity means that a small number of unusually large congestion events can be expected. Consequently, means calculated from measurements of the accuracy of prediction can be expected to be overly influenced by a few outliers in the tail. When determining the effectiveness of the LIS in predicting levels of congestion it is not sufficient to determine the average behaviour but it is also necessary to look at the variation from that average.

Furthermore in analysing the measurements a normal distribution can not be assumed. Consequently when evaluating the distribution of for example the error in predictions we need to use robust statistics [aSATVF93] in describing our measurements rather than making assumptions about the underlying distributions. Thus wide use is made in this chapter of percentiles as a measure of spread, Kolmogorov Smirnov statistic as a comparison of distributions and a non-parametric measure of correlation (The Spearman Rank Correlation Coefficient).

6.2.5 Workload and Overview of Data

In designing the workload for the experiment a number of considerations are important.

To obtain an estimate of the probability of loss at any point in time, a significant amount of data has to be transferred, to enable a meaningful average to be calculated. This is compounded by the desire to obtain measurements of steady state behaviour, for which each connection has to be long enough for averages to be calculated over the data transfered, after the start-up phase has been completed. These considerations led to the need for relatively long transfers.

The focus of the investigation is on start-up behaviour, and it is therefore desirable to have a large number of connections as each only starts once. In addition it is desirable to limit the amount of data that is used to obtain results. In balancing the above considerations, a transfer size of 1000 1460 byte segments was arrived at. When there is no loss this means that over 700 packets remain to be transferred when the offered window size is reached, giving sufficient data for calculating steady state averages.

To be able to measure the affect of aging on the accuracy of prediction, it is desirable to have a large number of samples which employ data of the same age to predict congestion and to have different ages available. To achieve this the transfers to a single destination are organised into groups, the structure of each is shown below:

1. Dynamic Transfer
2. Pause five minutes
3. Static Transfer
4. Pause One minute
5. Dynamic Transfer
6. Pause 1 second
7. Dynamic Transfer
8. Pause One Minute

This gives a large number of readings where the gap between transfers is 1 second, 1 minute and 5 minutes, as multiple sets of groups are transfered. In addition during each run only three destinations are used, giving a minimum separation of 21 minutes between each transfer in a round and its corresponding transfer in the subsequent round to the same destination.

The workload configuration was not chosen to be representative of Web traffic the main consideration. Rather it is conceived of as probe traffic which allows readings of the state of the network to be taken at a particular point in time. The implicit assumption is that the traffic generated in the experiment will not be sufficient to significantly alter the state of the network and that between probes the network will have sufficient time to return to its normal state. In short it is assumed that if a long transfer is observed the deductions about the behaviour of connections of size N can be made by observing the first N packets.

Trace Analysis

In order to control the quantity and age of data used in making congestion predictions the traces were rerun through the LIS. Each estimate is formed from the proportion of congestion notifications received by a single connection and the accuracy of a prediction is evaluated against the proportion of congestion notifications in a subsequent connection. For each connection the predicted and actual number of Implicit Congestion Notifications are logged along with the time, the destination host and local port number.

Description of Data

Throughout the following sections results are presented based upon the data generated using the techniques discussed above. Table 6.3 gives the number of connections and the mean proportion

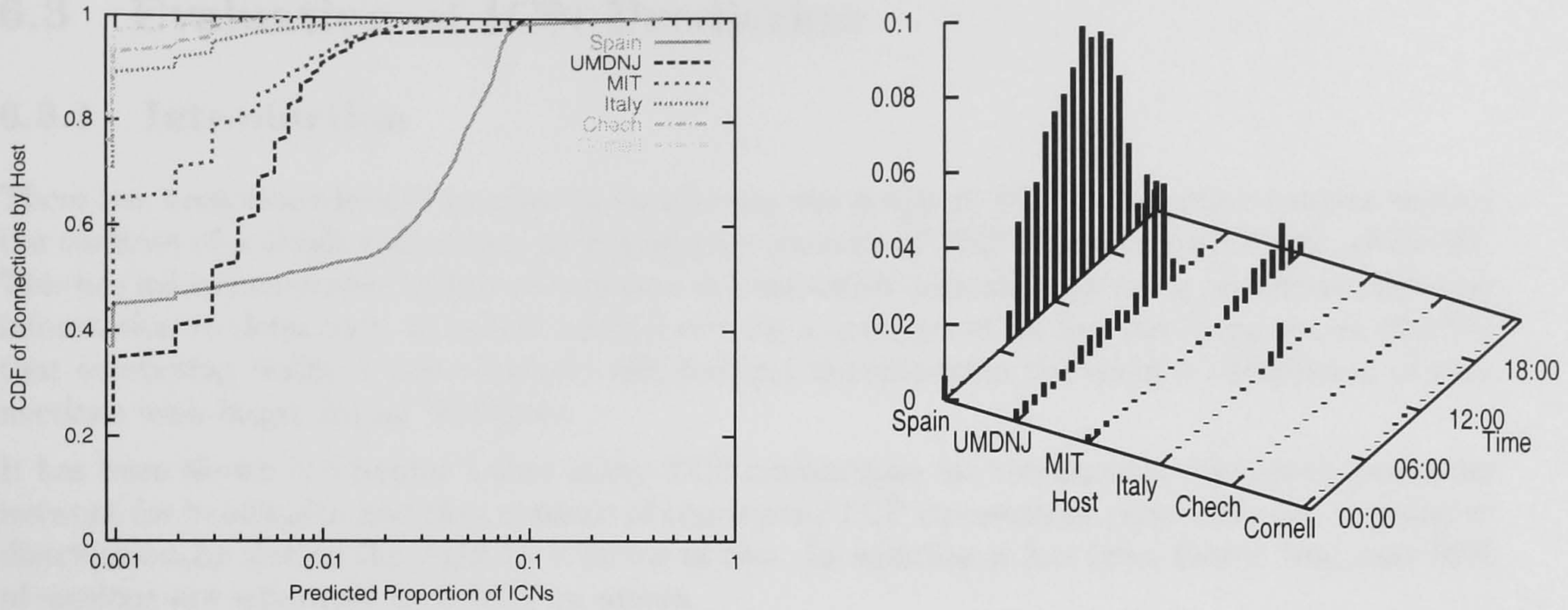


Figure 6.2: **Left: CDF of Implicit Notifications for each destination. Right: Average ICNs for each Hour of the Day (GMT)**

	All Cons		1 Sec.		1 Min.		5 min		20 Min	
Name	NC	ICN	NC	ICN	NC	ICN	NC	ICN	NC	ICN
All	4728	0.56	1181	0.54	1181	0.54	1182	0.57	1184	0.57
Spain	740	2.46	185	2.38	185	2.45	185	2.50	185	2.53
UMDNJ	822	0.58	205	0.58	205	0.56	206	0.60	206	0.58
MIT	820	0.26	205	0.25	205	0.26	205	0.27	205	0.27
Italy	740	0.09	185	0.09	185	0.06	185	0.09	185	0.11
Czech	785	0.03	196	0.04	196	0.03	196	0.04	197	0.04
Cornell	821	0.02	205	0.03	205	0.03	205	0.03	206	0.02

Table 6.3: **Number of Connections and Mean Percentage of Congestion Notifications by Destination**

of ICNs for the following categories; all connections, all connections to a host, all connections with a common inter connection gap, and all connections that are to the same destination and have a common interconnection gap.

The smallest number of connections upon which results are shown is 185 for connections to the Spanish Host for particular interconnection gaps. There is a wide range of congestion experienced on paths to different hosts, and the average congestion varies little for different interconnection gaps.

The left hand graph in Figure 6.2 shows the Cumulative Distributions for the proportion of ICNs received for each connection by destination host. The x-axis is to a log scale. This shows a wide variation in the level of congestion experienced by the different destinations. The right hand graph in Figure 6.2 shows the average proportion of ICNs received for each host for each hour of the day. This graph shows a strong diurnal pattern.

6.2.6 Summary

A methodology for investigating the use of a Location Information Server to control the initialisation of TCP's congestion window using long term averages has been described. The following sections present an analysis of the results.

6.3 Evaluation of ICN Prediction

6.3.1 Introduction

There has been considerable interest in broadening the scope of TCP's congestion control beyond the confines of a single connection to incorporate sessions of TCP connections [oET00, aRHK98]. This has led to favourable reports of increases in bandwidth utilisation by using previous congestion information to determine an initial window size for a connection. It has also been shown [Pad98] that competing traffic is not adversely affected and benefits from the quicker completion of connections with larger Initial Windows.

It has been shown in Chapter 4 that many TCP connections are too small to effectively probe the network for bandwidth and that sessions of concurrent TCP connections only shifts the cumulative distribution by size to the right by a factor of two. In addition it has been shown that over 60% of sessions are separated by 8 RTT or mores.

There is concern that if old congestion information is applied to a new connection, the IW may be too large for current conditions. This has led to a proposal for congestion window validation [MJS99], where it is recommended that the size of a congestion window is halved for every RTT[5]. It is argued that, in the absence of activity, competing flows will ramp up their bandwidth to utilise the free resources, making old congestion information invalid.

The model of long lived connections, using congestion avoidance to fairly distribute resources, does not however fit well with the reality of today's Internet. Most connections are too short lived. This observation leads to the idea of measuring congestion feedback over relatively long periods of time and applying equation based congestion control to estimate a statistically fair starting size for a connection's IW.

For this approach to be successful it is necessary to predict, with a reasonable degree of accuracy congestion conditions over time spans that are several orders of magnitude greater than a network RTT. The strong diurnal pattern to congestion notifications evident in Figure 6.2 suggests that such an approach is feasible. It will be further investigated in this section, the aim of which is, to establish how well past congestion feedback predicts the congestion that a future connection will experience.

The discussion is structured into the following sections; firstly, global, local and temporal prediction are compared, next the accuracy of predictions under different congestion regimes are compared and finally how the accuracy of prediction decays as the congestion information ages is discussed.

6.3.2 Comparison of Global Local and Temporal Prediction

Throughout this Section results are presented and discussed for predictions of congestion made from recent traffic to a distinct destination. To set the context for this discussion two other methods are considered, namely to derive a single estimate from all connections to all destinations and an estimate for each destination from all connections to that destination.

The simplest approach discussed is to maintain a single congestion estimate. For the traces under consideration this would yield a mean of 0.56% ICNs. For 10% of connections the actual percentage of ICNs would be 0.51% above the mean and for 1% of connections 6.9%. If the errors to individual destinations are considered the results would be worse. For destinations on congested paths ICNs are massively underestimated (by a factor of five for the Spanish host) and for destinations on uncongested paths they are overestimated (by a factor of 28 for the Cornell host).

A congestion estimate calculated over all connections to a common destination[6] will result in a

[5] In practice approximated by an RTO.

[6] Where a destination is a group of hosts known to share the same location: this could be defined by IP number, hosts with a common higher order three bytes in their IP address, or by network address.

	Absolute			Global Mean		Local Mean		Prediction	
Name	Mean	90	99	90	99	90	99	90	99
All	0.56	1.07	7.52	0.51	6.96	—	—	0.29	1.61
Spain	2.46	6.73	8.87	6.17	8.31	4.27	6.41	0.98	2.58
UMDNJ	0.58	0.87	8.51	0.31	7.95	0.29	7.93	0.39	1.14
MIT	0.26	0.77	2.39	0.21	1.83	0.51	2.13	0.39	1.43
Italy	0.09	0.19	1.52	-0.37	0.96	0.10	1.43	0.10	0.57
Czech	0.03	0.10	0.39	-0.46	-0.17	0.07	0.36	0.10	0.29
Cornell	0.02	0.10	0.29	-0.46	-0.27	0.08	0.29	0.10	0.29
Average	0.56	1.45	3.66	0.89	3.10	0.89	3.10	0.34	1.05

Table 6.4: **Comparison of Global, Local and Temporal Prediction: The first three columns give the mean, 90^{th} and 99^{th} percentiles of congestion notifications. The columns labeled *Global Mean* give the distance between the mean, of ICNs for all connections, and the 90^{th} and 99^{th} percentiles for the row entity. The columns labeled *Local Mean* give the distance between the mean of ICNs, for all connections *to the row entity* , and the 90^{th} and 99^{th} percentiles. The columns labeled *Prediction* give the values for the 90th and 99th percentiles of error, between a prediction based upon recent traffic and the proportion of ICNs actually received.**

smaller spread of error, but the average of all estimates calculated in this way will be the same as for a global mean.

For each destination the distance between the upper congestion percentiles and the local mean may be more or less than the distance from the global mean. To illustrate this three cases will be considered using the data in Table 6.4.

If the destination mean is higher than the global mean, using the local mean reduces the distance between the estimate and the upper percentiles. For example with the Spanish Hosts 90^{th} percentile the distance will reduce from 6.17% to 4.27%. Thus when congestion is high on a route the proportion of connections where congestion is underestimated is reduced by using a local mean.

If the destination mean is lower than the global mean, but the upper percentile is higher than the global the distance between mean and local percentile will be increased by using a local mean. For example the 99^{th} percentile of the MIT host will be 2.13% greater than the local mean but only 1.83% greater than the aggregate mean. For percentiles that are smaller than the global average an overestimate will be changed to an underestimate.

The sum of the distance, to a percentile, for all hosts is the same, whether the distance is measured from the local mean or from the global mean. The variance of these distances is however reduced by using a local mean. These observations simply reflect a local average being more responsive to local network conditions. An aggregate mean underestimates congestion when it is high and overestimates it when it is low.

The effectiveness of using recent rather than all traffic to produce a congestion estimate is discussed next. If the prediction of congestion is based only on recent traffic it reduces the distance between a prediction and the upper percentiles. This reduces the average error for the 90^{th} and 99^{th} percentiles by a factor of 3 when compared with the global and local mean methods. Figure 6.3 plots the predicted and actual proportion of ICNs for all connections and graphically illustrates the accuracy with which congestion is predicted using recent traffic.

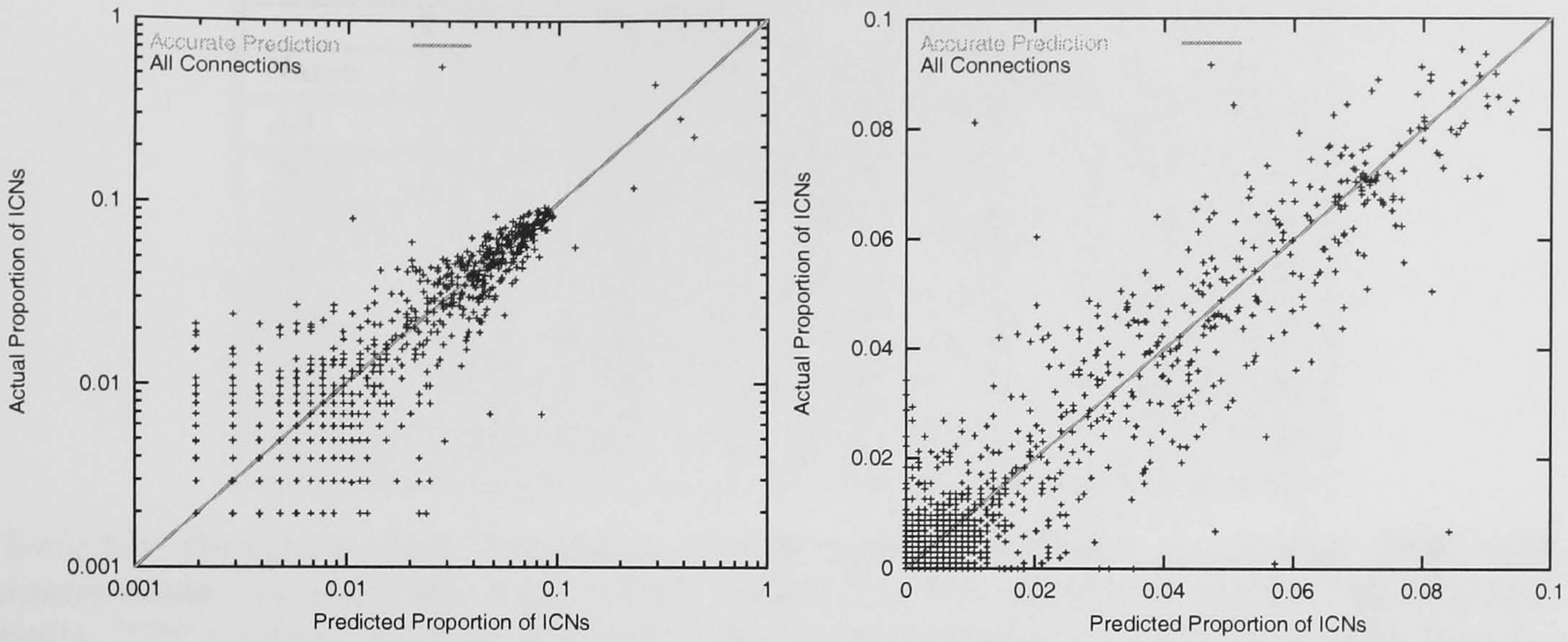


Figure 6.3: **Predicted and Actual Connection Congestion Proportions: On these scatter graph each point represents one connection. The left hand graph is to a log and the right to a linear scale. The x-axis is the proportion of congestion notifications predicted for a connection and the y-axis the proportion of congestion notifications that it received.**

6.3.3 Comparison of Effectiveness of Prediction by Destination

This section discusses the effectiveness of congestion prediction and compares the results for each destination. Firstly, the correlation between predicted and experienced congestion is discussed. Secondly, the Absolute Mean Deviation of Error (AMDE) is used to quantify the accuracy with which the predictions are made. Thirdly, the 90^{th} and 99^{th} percentiles of error are used to discuss the distribution of error. The results are presented in Table 6.5.

Two measures of correlation were used, the parametric Pearson Correlation Coefficient and the non parametric Spearman Correlation Coefficient. For both methods a value of one indicates complete positive correlation, a value of zero shows there is no correlation and minus one shows complete negative correlation.

As the name suggests Spearman's Rank method first ranks the data point in each pair and then calculates the correlation of ranks. If there is more than one point with the same value, they are counted as one data point and the mean of their ranks is used. Consequently all the connections with predictions and outcomes of no congestion notifications are counted as one data point. The strength of correlation is therefore an underestimate, particularly for paths with a low level of congestion.

The visual strength of the correlation in Figure 6.3 is supported by the Pearson Correlation Coefficient of 0.88 and Spearman Rank Correlation Coefficient of 0.62 for all connections.

Next consider the correlation for separate destinations. Coefficients of 0.84 and 0.89 for the host in Spain contrast with a correlation of only 0.02 for the host in Italy and the rank method yields a negative correlation of -0.04, albeit with a low significance level.

The negative correlation for traffic to Cornell reveals a real effect, which could be termed the *stable door* effect. If a congestion notification is very unlikely, it will arrive when the congestion prediction is zero. This will cause a prediction of congestion for future connections, yet the unlikeliness of congestion means that they probably will not receive a notification. For the connection that received a congestion notification none was predicted and for the connection that a congestion notification was predicted none were received, as the double zero connections count as only one reading there is a negative correlation.

	ICNs	Percentile		AMDE		Correlation	
Name	Mean	90	99	AMD	PAMD	P	R
All	0.56	0.29	1.61	0.22	0.39	0.88	0.62
Spain	2.46	0.98	2.58	0.65	0.27	0.84	0.88
UMDNJ	0.58	0.39	1.14	0.26	0.45	0.91	0.59
MIT	0.26	0.39	1.43	0.22	0.85	0.66	0.47
Italy	0.09	0.10	0.57	0.10	1.06	0.69	0.23
Czech	0.03	0.10	0.29	0.05	1.77	0.11	0.12
Cornell	0.02	0.10	0.29	0.04	2.15	0.09	-0.04
Average	0.56	0.34	1.05	0.22	1.09	0.55	0.37

Table 6.5: **Comparison of Congestion Predictions to Different Locations: This table summarises the accuracy with which congestion was predicted to the various locations. The columns labeled ICNs give the mean percentage of ICNs to each location and the 90^{th} and 99^{th} percentile of error in predictions. Subsequent columns give the Absolute Mean Deviation in prediction, the Proportion of the Absolute Mean Deviation to the Mean number of Implicit Congestion Notifications, the Pearson Correlation Coefficient and the Spearman Rank Correlation Coefficient respectively.**

These results show that there is a stronger correlation on paths with a high level of congestion than on paths with a low level of congestion. When the correlation coefficient was calculated over connections to all destinations there was a stronger correlation than when the paths are considered individually.

Having discussed the correlation between predictions and congestion outcomes the accuracy of prediction will be examined. The Absolute Mean Deviation of Error (AMDE) provides a measure of the average distance between a prediction and the result. From the figures presented in Table 6.5 a number of observations can be made.

1. The AMDE for all connections of 0.217% is around half of the average level of congestion 0.56%.

2. There is a positive relationship between the level of congestion on a route and the AMDE. Thus despite the weaker correlation between predictions and outcomes at lower levels of congestion the absolute error is less.

3. The proportion of AMDE to average ICN decreases as the level of congestion increases. This is consistent with the stronger correlation on paths with more congestion.

Whilst the AMDE provides a measure of the accuracy of prediction, it is also useful to estimate the error in prediction that only a small percentage of connections will exceed. This is important because, when a congestion prediction is used as an input to congestion window initialisation the consequences of over estimation and underestimation differ. If the actual level of congestion is larger than predicted, there may be global consequences for the stability of the network. If congestion is smaller than expected the consequences are local, in that the connection may take longer to complete than necessary.

This in turn means that a reliance on the symmetry of error to ensure that under and overestimation cancel out is inadequate. It is useful to know the range of underestimation that can be expected. Table 6.5 suggests that this varies significantly by destination, but that 90% of connections will have an error less than the mean level of congestion for the path. The implications of this will be discussed further in Section 6.4.

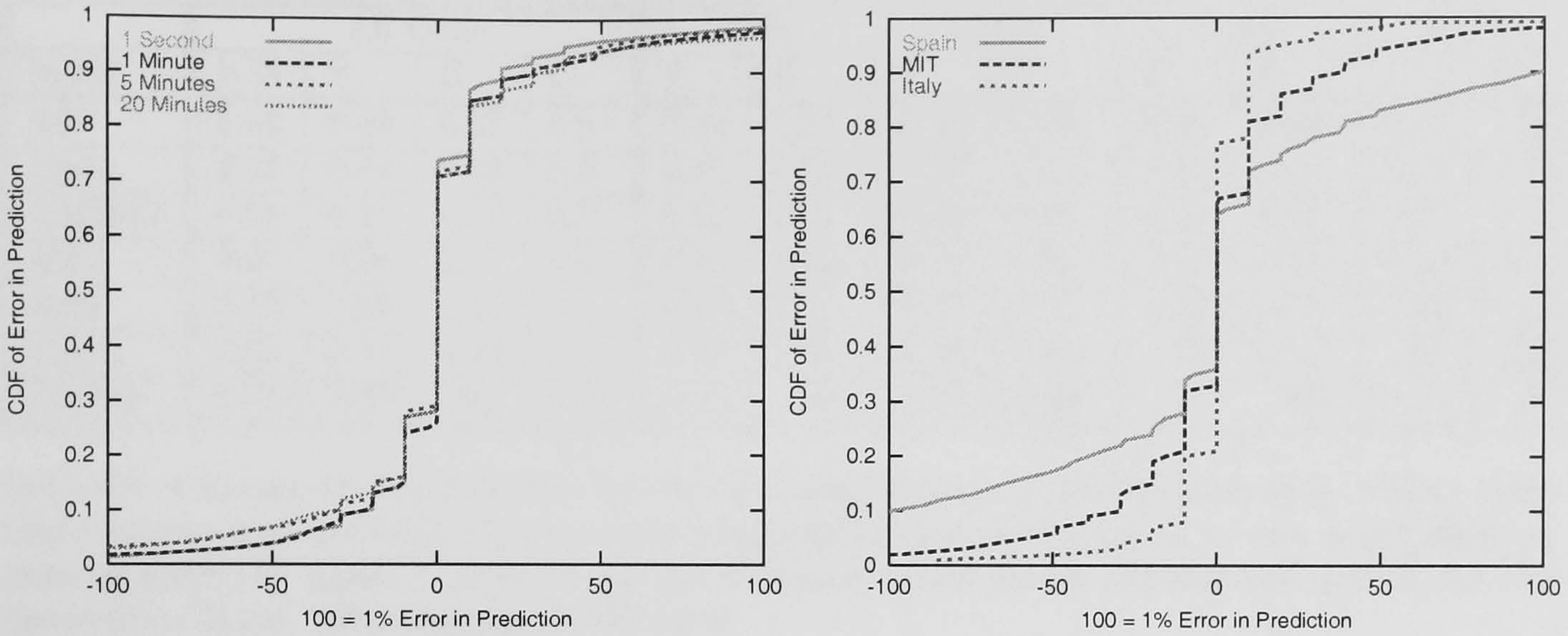


Figure 6.4: **Right: CDF of Error in congestion prediction by Destination. Left: CDF of Error in congestion prediction by Age of Data**

6.3.4 Decay of Accuracy over Time

This section discusses how congestion predictions are affected by the aging of the data upon which they are based. To achieve this the Correlation Coefficients, AMDE and Upper Percentiles of Error for interconnection gaps ranging from one second to twenty minutes are compared. The results are presented in Table 6.6 for the correlations and Table 6.7 for the AMDE and percentiles.

Context is provided by Figure 6.4, which plots the cumulative distribution function for the four categories of interconnection gap and for three hosts. This figure shows that the error in congestion estimates varies more widely by location than be the delay between connections, which in turn reflects earlier findings of the importance of the absolute level of congestion in influencing the size of prediction error.

Next consider the relationship between the age of data and the strength of correlation. For delays between 1 second and five minutes there is a slight decline in the strength of correlation. For the Pearson correlation coefficient there is a marked decline between five and twenty minutes, but there remains a strong positive correlation. The Rank method gives a less sharp decline. For the All Connection category the Pearson Coefficient drops from .095 to 0.71 and the Spearman from 0.67 to 0.54.

The variation in correlation results for timescales between 1 second and 5 minutes is consistent with the results for accuracy and spread of error in predictions. The AMDE for all connections being 0.170% for a one second delay and 0.219% for a five minute delay. For the 99^{th} percentile there is a similar increase from 1.47% to 2.36%.

There is a significant decline in the accuracy of prediction when the 1 Second and 20 minute categories are compared. The AMDE increases from 0.170% to 0.288%, a factor of 1.7. The 99^{th} percentile of error also increases by just under a factor of 2. Similar results apply to individual destinations, although the differentiation is most marked for routes where the level of congestion is high.

6.3.5 Summary

This section has discussed the accuracy with which congestion can be predicted over time scales ranging from one second to twenty minutes.

	All Cons				1 Sec.		1 Min.		5 min		20 Min	
Name	ICN	P	R	C	P	R	P	R	P	R	P	R
All	0.56	0.88	0.62	0.0	0.94	0.67	0.95	0.65	0.95	0.60	0.71	0.54
Spain	2.46	0.84	0.88	0.0	0.92	0.91	0.95	0.91	0.95	0.88	0.61	0.82
UMDNJ	0.58	0.91	0.59	0.0	0.98	0.77	0.97	0.61	0.94	0.51	0.77	0.47
MIT	0.26	0.66	0.47	0.0	0.82	0.63	0.76	0.56	0.58	0.41	0.50	0.32
Italy	0.09	0.69	0.23	0.0	0.89	0.22	0.79	0.20	0.80	0.32	0.43	0.24
Czech	0.03	0.11	0.12	0.01	0.15	0.20	0.28	0.19	0.10	0.12	-.07	.004
Cornell	0.02	0.09	-.04	0.26	0.23	-.058	0.06	0.046	-.07	-.074	0.11	-.06

Table 6.6: **Change in Correlation between Congestion and Prediction over Time: This table shows the correlation between congestion and predictions which are based on data of different ages. P stands for the Pearson Correlation Coefficient and R for the Spearman Rank Correlation Coefficient**

	All			1 Second			1 Minute			20 Minutes		
Name	AMD	90	99	AMD	90	99	AMD	90	99	AMD	90	99
All	0.217	0.19	1.17	0.170	0.29	1.47	0.219	0.29	1.58	0.288	0.29	2.36
Spain	0.652	0.88	1.81	0.527	1.03	1.89	0.610	0.79	2.15	0.901	1.34	3.41
UMDNJ	0.262	0.19	0.39	0.179	0.48	0.77	0.289	0.48	0.97	0.361	0.48	1.35
MIT	0.221	0.29	0.68	0.170	0.39	0.97	0.240	0.38	0.96	0.281	0.38	1.63
Italy	0.095	0.10	0.29	0.069	0.10	0.48	0.095	0.10	0.49	0.121	0.19	1.53
Czech	0.053	0.10	0.19	0.049	0.10	0.29	0.057	0.10	0.29	0.057	0.10	0.38
Cornell	0.043	0.10	0.29	0.046	0.10	0.29	0.041	0.10	0.19	0.044	0.10	0.10

Table 6.7: **Change in accuracy and spread of prediction over time: This table presents figures that show how the accuracy of prediction changes as the data used for the prediction ages. The Absolute Mean Deviation of Error and the 90^{th} and 99^{th} percentiles of error are used to measure accuracy and spread respectively.**

When predictions are based upon recent traffic to the same destination they are more accurate than when made from averages of all traffic or on averages of all traffic to the same destination. This is reflected in requiring a smaller overestimation to ensure that 90% and 99% of connections experience less congestion than the prediction. This result holds for periods of up twenty minutes.

The strength of correlation, mean deviation of error and size of the upper error percentiles have been compared for paths with differing congestion regimes. It has been found that the correlations are strongest and the AMDE is largest for paths with a high level of congestion. The correlation coefficient when all paths are considered is 0.88. This suggests that congestion information is of value for the time spans under consideration.

The decline in correlation and accuracy of prediction as the age of congestion information increases, has been considered. It has been found that there is little difference in time scales from one second to five minutes and that there is a significant weakening of the correlation between the five minute and twenty minute categories. The accuracy of predictions degrades by a factor of two over the same time scale.

This leads to the conclusion that the combination of equation based congestion control and medium term congestion prediction is a reasonable proposition. Subsequent sections will investigate the issues that arise when this is applied to the initialisation of TCP's congestion window and the implications of such a policy.

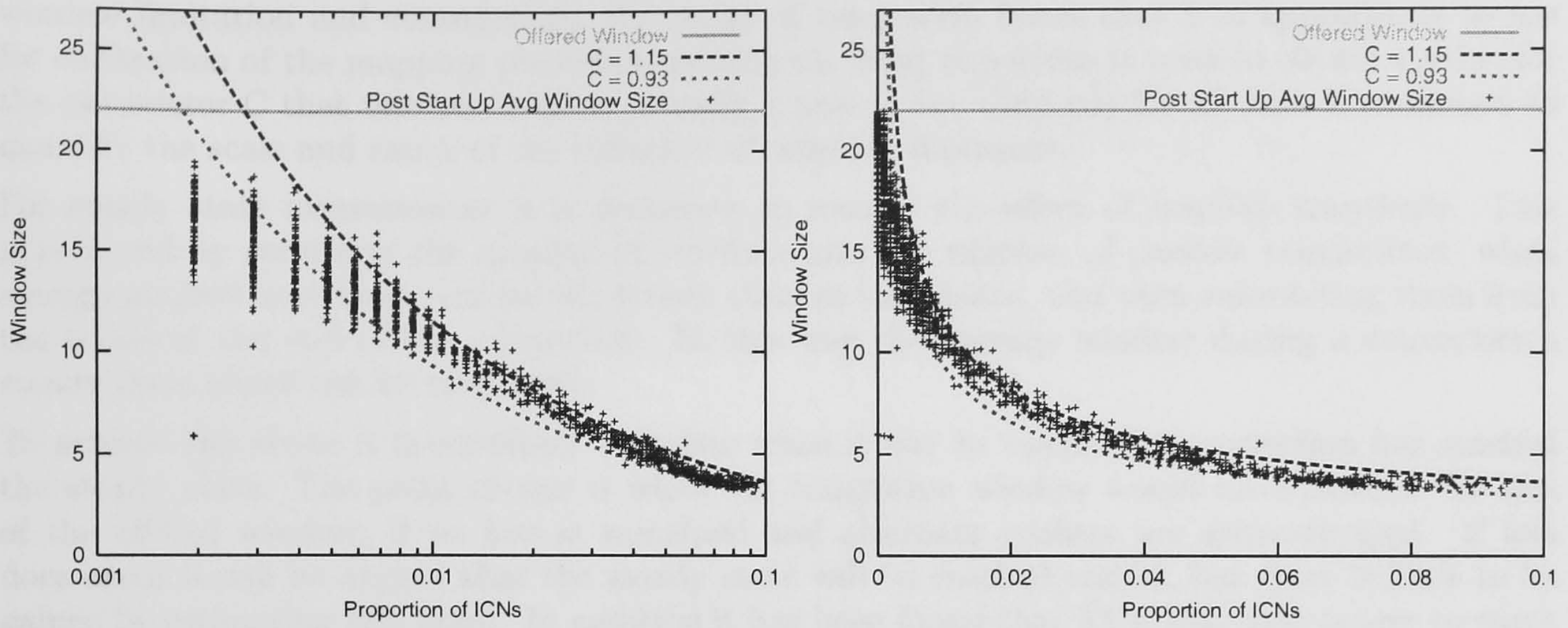


Figure 6.5: **Post Start-Up Average Window Size: the average steady state window size is proportional to 1 over the square route of loss. However at low levels of loss window limitation depresses the steady state average window size.**

6.4 Host Utilisation of Congestion Notifications

The late 1990's have seen a growth in the development of new End-to-End congestion control schemes for Internet Traffic. This has in part been driven by the need to support data types that are poorly served by TCP's reliable byte stream, connection oriented service. Examples include congestion control schemes appropriate for rate based flow control [MF97, RHE99], Multicast traffic [GS99, KC98] and End-to-End differentiated service [Gev98, CO98, KMT98][7].

Section 6.3 focused on the accuracy with which congestion can be predicted, over time scales that are relatively long when compared with a RTT. The discussion did not consider the characteristics of congestion control schemes that might utilise the congestion information. The techniques presented were intended to be orthogonal to a large subset of such schemes[8].

The aim of this section is to discuss the issues that arise, when TCP is modified to use congestion prediction to set a connection's initial window size. The argument is developed in three stages; first measurements are used to arrive at a mapping from a congestion measurement to a fair window size, next the interaction between flow[9] and congestion control is considered and thirdly trace analysis is used to analyse how closely IWs approach their target sizes.

6.4.1 Mapping from Congestion to Steady State Window Size

In this section measurements are used to explore the relationship between the steady state window size of a connection and the proportion of ICNs that it receives. The mapping from *measured congestion* to *steady state window* size is calibrated to minimise error. It will then be shown that the LIS's measurements of ICNs can be used to accurately predict the fair window size of connections. In Section 6.4.2 similar techniques will be applied to map from a *congestion prediction* to an IW size.

First the procedure used to obtain the steady state window size is described. This is followed by a comparison of the Ideal Model with a plot of the data. Conclusions about the influence of

[7] As distinct from network negotiated Differentiated Service [BBC+98]

[8] This is particularly so when the congestion estimate is accompanied by a RTT measurement, which makes the mapping from a fair window to a fair rate trivial.

[9] In the form of window limitation

window limitation and consequently the range of congestion levels that it is appropriate to use for calibration of the mapping procedure are drawn. Next regression is used to obtain a value for the parameter C that minimises error. Finally a plot of the residuals for all data is examined to quantify the scale and range of the influence of window limitation.

For steady state measurement it is necessary to remove the effect of start-up transients. This is achieved by recording the number of windows and the number of packets transmitted, when enough packets have been sent for the steady state to be reached, and then subtracting these from the totals at the end of the connection. In this way the average window during a connection's steady state phase can be estimated.

To achieve the above it is necessary to define when it can be assumed a connection has reached the steady state. The point chosen is when the congestion window would have reached the size of the offered window, if no loss is sustained and alternate packets are acknowledged. If loss does occur it can be argued that the steady state will be reached earlier, but there is little to be gained by estimating this point. In addition it has been found that TCP may take longer to reach equilibrium, because during SS the congestion window opens exponentially the expected size of the window when loss occurs will be proportional[10] to $\frac{1}{p}$ rather than $\frac{1}{\sqrt{p}}$.

In reference [MSM97] a value for C under random loss conditions and delayed acknowledgments[11] is derived as 0.93. This is plotted along with the value *1.15*. The actual value for C appears to lie between these two and regression will be used to determine it. The maximum offered window size for all connections was 32768 bytes, which rounds down to 22 segments. In addition to the model and offered window sizes Figure 6.5 also presents a scatter plot of the proportion of ICNs against average steady state window size. Each point corresponds to one connection.

The fit between the model and the results appears good between levels of loss of 1% and 10% . There is a strong divergence at lower levels of congestion, where the effects of window limitation reduce the average window size. This is emphasised in the left hand graph where a log scale is used on the x-axis. Consequently when regression was used to derive a value for C from the measurements only connections with greater than 1% and less than 10% ICNs were utilised.

To allow linear regression to be applied first the proportion of ICNs is transformed by taking one over its square root. This is then used as the predictor variable. It is assumed that the Y axis will be intercepted at (0,0). This leaves C as the slope of the line which best fits the data.

The regression yields a value for C of 1.079, using 95% confidence intervals the upper interval is at 1.086 and the lower at 1.073. This results are slightly larger than the value obtained analytically under deterministic and random loss assumptions [MSM97].

Figure 6.6 plots the predictor and predicted variables for the data used in the regression and for the rest of the data, as well as the line of best fit. The right hand graph plots the residuals.

Three observations can be made from these graphs; at high levels of loss SS slightly depresses the steady state window size, at low levels of loss window limitation greatly depresses the steady state window size and there is a good fit in the intermediate range. To conclude, the value estimated for C is reasonable and will be used in future sections, when assessing the mapping from predicted congestion levels to IW Sizes.

6.4.2 Mapping From Congestion Prediction to Initial Window Size

The previous Section, used measurements of congestion notifications and steady state window sizes to obtain a value for the parameter C in the Ideal TCP model. This section will discuss using *predictions of congestion* to determine an *IW* size. The argument will be developed in three stages. First issues relating to how to determine, which value `cwnd` should be initialised with are considered. Next two means for evaluating how the IW size relates to congestion experienced by

[10] This issue is revisited in Section 6.5.2.

[11] In Equation 6.1

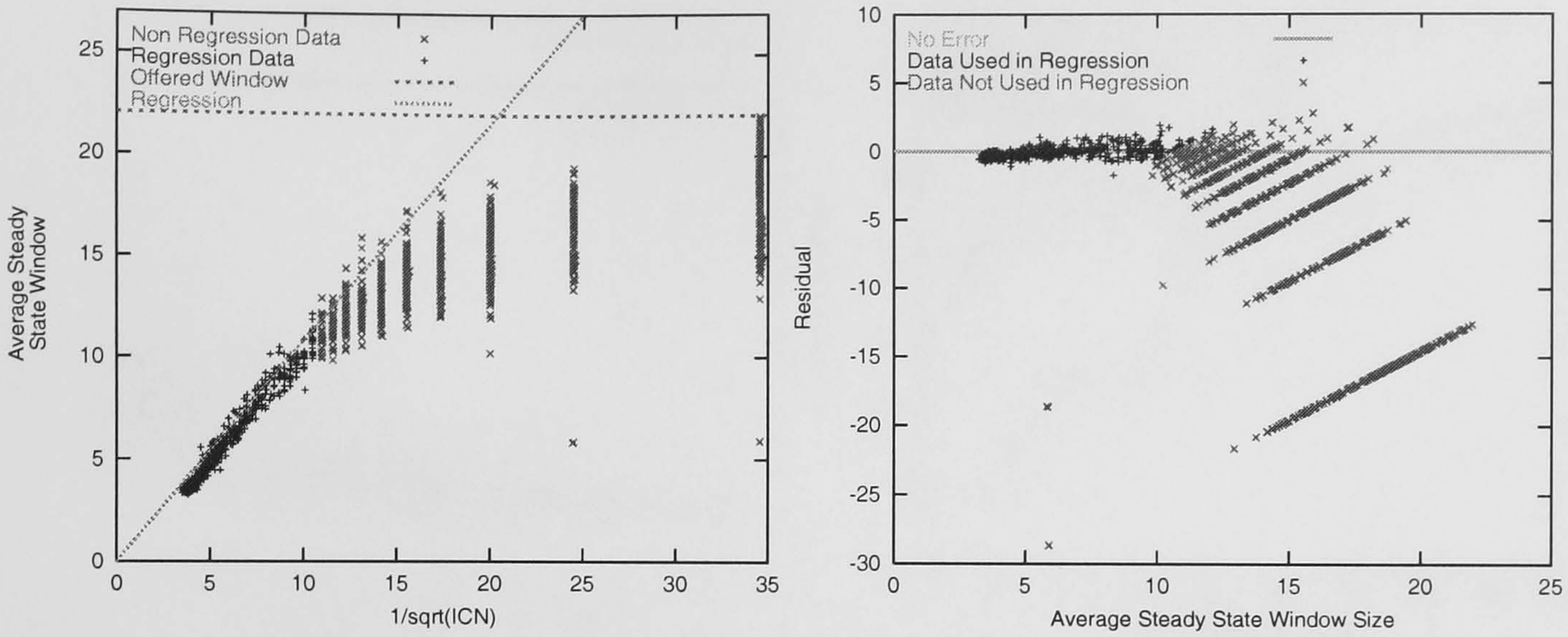


Figure 6.6: **Left: Regression for ICNs and Window Size. Right: Residuals from the Regression**

the connection are described. This is followed by the presentation of results and a discussion of their implications.

In using a congestion prediction to determine an IW it is hoped that it will be possible for the connection to commence in CA and avoid SS, without prejudicing the service received from the network of competing connections. In trying to achieve this three issues arise. Given that a fair window size can be determined from a congestion prediction in the same way that it can be derived from a measurement of congestion, how should the initial window size relate to the predicted fair window size. This will be considered after two other issues: how to handle the case when no congestion is predicted, and how to react when the flow(s) from which congestion information has been extracted are application or window limited.

The window size for the Ideal Model is undefined when there is no congestion. The simple solution adopted is to convert a zero congestion prediction to a small value. In this experiment zero prediction is converted to one congestion notification in 1024 data packets, which yields a fair window size of 35 segments.

It cannot be assumed that the congestion feedback a flow receives is independent of the bandwidth that it utilises. Whilst TCP's congestion control mechanisms will normally result in adaption to the current congestion regime resulting in a convergence between feedback and bandwidth, when the connection is window or application limited this convergence will not occur. In this case the connection throughput may be significantly smaller than is implied by the congestion feedback, consequently it is not safe to infer that congestion notifications alone are sufficient to determine fair bandwidth. In this situation it is better to limit the IW to the bandwidth utilised by the connection(s) from which the congestion information was extracted. This is achieved by the Location Information Server monitoring connection window sizes and communicating the largest recent window size alongside a congestion prediction in a Location Information Packet. The host then uses this window size as an upper limit on the size of the initial window. There is however no indication that the congestion information is wrong, therefor the SS threshold is still set to the window size indicated by the congestion information using the procedure described below. Consequently available bandwidth can be probed using SS.

With a prediction of congestion conditions it is possible to avoid the exponential increase of SS and go straight into Congestion Avoidance. Under deterministic loss conditions a TCP connection will oscillate around its fair window size between an upper and lower bound, which are respectively $\frac{4}{3}$ and $\frac{2}{3}$ of the fair window. The lower bound provides an intuitively attractive starting point. It is

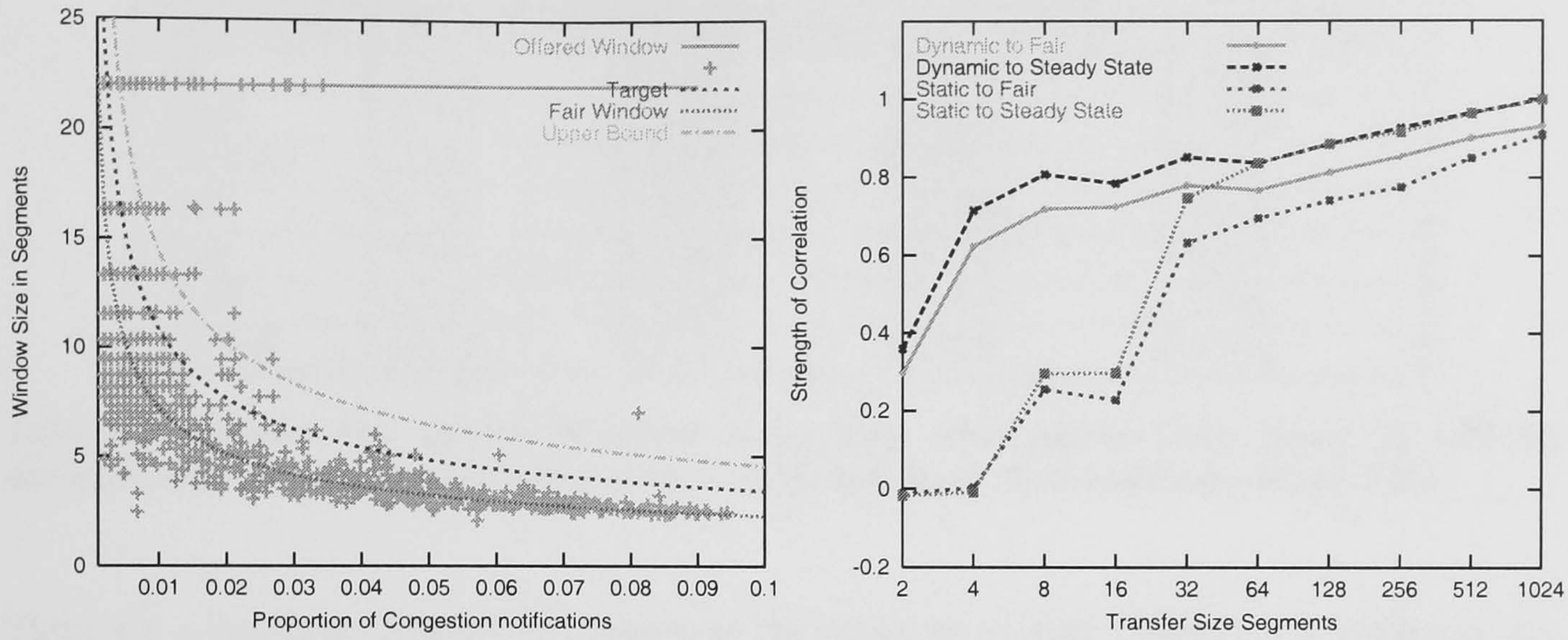


Figure 6.7: **Left: The Initial Window Sizes of connections are plotted against the current proportion of congestion notifications. The guideline shows an equation calculated fair window size and target window size. Right: Shows the correlation between Steady State and Average window sizes. For small to medium connections using the LIS increases the strength of the correlation indicating that responsiveness to network conditions is increased.**

conservative in that the IW is at the lower end of the normal region of oscillation. Yet it has been shown [ZQK99] that under certain conditions it can lead to optimal performance. For the results presented in this section the IW was set to the lower bound. This will allow an assessment of the intuition.

Having discussed procedures for determining an IW size, the results are evaluated by comparing the size of the start-up window with the congestion actually experienced by each connection. Two methods of evaluation are used, a guided scatter plot (Figure 6.7 and a table of proportional categories (Table 6.8. The scatter plot is described first.

The x and y axis are respectively, the proportion of congestion notifications and the IW size in segments. In order to aid visual assessment a number of lines are plotted: the offered window size provides an upper limit on the IW size; in addition the Lower and Upper Bounds and the Fair Window size are also drawn. The Lower Bound is also the target window size. These lines provide a visual framework for the assessment of the connection's window sizes. Each connection is plotted as a single point.

Whilst the scatter plot allows identification of trends it needs to be supplemented to allow quantification. To this end the results are also provided in tabular format. Each connection is placed in one of four categories. Below Lower Bound for connections with IWs up to and including the target window size. Lower to Fair for connections greater than the target window but less than the Fair Window size. It would be desirable for all connections to fall into these two categories. Connections with a slight overshoot are in the Fair to Upper category and those with a more serious overshoot are in the Above Upper category. This method of categorisation is based on a similar scheme used in [AP99].

Having specified the elements of Figure 6.7 and Table 6.8 the results will now be discussed.

Firstly, there is a clear clustering of connections around the target window size. This is most discernible at the higher levels of congestion. In part this effect is exaggerated, because at low levels of congestion a large number of connections share the same window size and congestion level and appear as one point. This effect is most extreme for connections with no congestion predicted and non experienced, which form a majority of all connections yet appear as only one point.

Destination	Below Lower	Lower to Fair	Fair to Upper	Above Upper
All	0.822	0.107	0.044	0.027
Spain	0.701	0.236	0.032	0.029
UMDNJ	0.631	0.225	0.087	0.057
MIT	0.764	0.117	0.068	0.051
Italy	0.919	0.034	0.032	0.015
Czech	0.940	0.022	0.033	0.005
Cornell	0.975	0.010	0.012	0.002

Table 6.8: **Where the Initial Window Sizes Fall: this shows that using an LIS to dynamically set the initial window size does not lead to excessively large IWs.**

There are a significant number of connections for which no or little congestion is predicted, yet which experience significant congestion. This is in part exaggerated by the large number of connections for which congestion predictions are low, but appears as a real effect.

Next consider Table 6.8 Between 63% and 98% of connections to each location were at or below the lower bound. Only 2.7% of connection were above the upper bound[12]. A strong tail to the distribution is indicated by the small drop in the proportion of connections between the Fair to Upper and Above Upper categories. This suggests that these outliers are insensitive to adjusting the target window size[13].

6.4.3 Summary

In this section it has been argued that predictions of congestion can be utilised by a wide range of congestion control schemes, but that the concrete issues that arise with each may differ. TCP has been used as a vehicle to illustrate the effectiveness of congestion prediction and to explore concrete issues that arise when it used to determine an initial value for the congestion window.

Measurements of congestion and steady state window sizes were used to derive a value for the parameter C in the ideal TCP model of 1.07. In the process of deriving this parameter it was shown that window limitation has a significant effect when congestion is below 1%[14].

These findings were then used to enable a comparison of IW Sizes, derived from congestion predictions, with the fair window sizes for the congestion experienced by each connection. These results were presented graphically and as a proportion of connections which fell into defined categories. The IW sizes cluster around the Target window size. This demonstrates that dynamic initialisation is effective. A significant minority of connections (2.7%) however had IWs larger than the Upper Bound. This suggests that there are difficulties in using long term predictions to forecast transient congestion.

6.5 Global and Local Implications

This section looks at how dynamic window initialisation changes the subsequent behaviour of a connection, with particular reference to the start-up phase. The consequences are discussed with

[12] An inference that can be drawn by the effects of window limitation starting before the upper bound reaches the offered window, is that under normal conditions TCP connections window sizes will stray above the upper bound.

[13] This leads to speculation that the spread of time over which measurements are taken may be important. It may just be that transient congestion is difficult to predict and there will be a small but significant proportion of outliers. In this case it can be argued that it is no worse than TCP's current behaviour.

[14] For a window size of 32K and 1460 byte segments. For smaller window sizes the effects will start at lower levels of congestion and for smaller segments at higher levels of congestion

respect to; the utility received by the user, the peak load placed upon the network and the effect upon competing traffic.

Firstly, the response time as experienced by a user is discussed. To measure the effect of window initialisation on delay the number of windows that it takes to complete transfers of different sizes are used. For this metric, if the number of windows is reduced, the user will receive a better service. If it is achieved by simply increasing the load on the network at the expense of other users it can however be argued that one users utility is being increased at the expense of others.

This question is addressed in three ways. First it is demonstrated that dynamic initialisation increases the responsiveness of connections to network feedback[15]. Secondly, the peak load that is placed upon the network during start-up is compared to TCP's steady state and connections using SS. Thirdly, the congestion feedback received by connections, that are dynamically and statically configured, is compared.

In discussing the above issues the interaction with the network as a connection develops is important and the result of many factors. For this reason trace analysis is not used to predict values but all the measurements are taken directly from actual connections.

6.5.1 Number of Windows required

A long standing design aim of network architects has been to make the use of the network transparent to users [RW70]. That is interaction with a remote computer should be indistinguishable from local interactions. Yet delays of around 60 of milliseconds are perceivable to users and delays of over 100ms can degrades interactions with a graphical user interface [CMN83]. The web experience certainly falls significantly short of this goal most of the time.

There are physical limits which prevent the ideal being reached. The speed of light prevents a RTT across the US from dropping much below 60ms. If the network is under provisioned and there is no admission control significant, queuing delays will increase the response time.

There are however two situations in which TCP's current congestion control mechanisms may themselves contribute to the delay experienced by users. When network resources are unused TCP's congestion control will limit a connection's bandwidth until the network has been probed. Secondly, when there is light contention for network resources a new connection will not receive its fair share until several round trip times into its life.

This section will show that dynamic initialisation reduces the number of round trip times that it takes to complete many connections and that the bandwidth utilisation is responsive to the level of congestion. The discussion proceeds in the following manner. First a definition of what is being measured is given. This is followed by a description of how the measurements are made. Next the results are presented and their implications discussed.

In measuring the number of rounds that it takes to complete a connection, only the influence of the evolution of the congestion window on delay is accounted for. The effect of RTOs is excluded[16].

All the transfers used are of a fixed size, however this does not prevent measurements of the number of rounds it would take varying sizes to complete being made. The focus on start-up makes this fortunate. The number of rounds that occur before a particular segment is acknowledged are recorded for 2, 4, 8 ... 1024 segments. Retransmissions are not counted.

The results for dynamic and static configurations are compared using two metrics. The mean difference in the number of rounds between each configuration and the mean ratio of Static to

[15]The right hand graph in Figure 6.7 shows the correlation between the average window size for the first X segments transmitted and the fair and steady state window sizes. There is a stronger correlation for connections that are dynamically initialised

[16]Although there is evidence to suggest that dynamic window initialisation when combined with Rate Based Pacing will reduce the number of RTOs. This is because when the congestion window is smaller than four segments there are insufficient packets in flight for duplicate acknowledgments to trigger fast recovery and fast retransmit. Any loss will result in a timeout

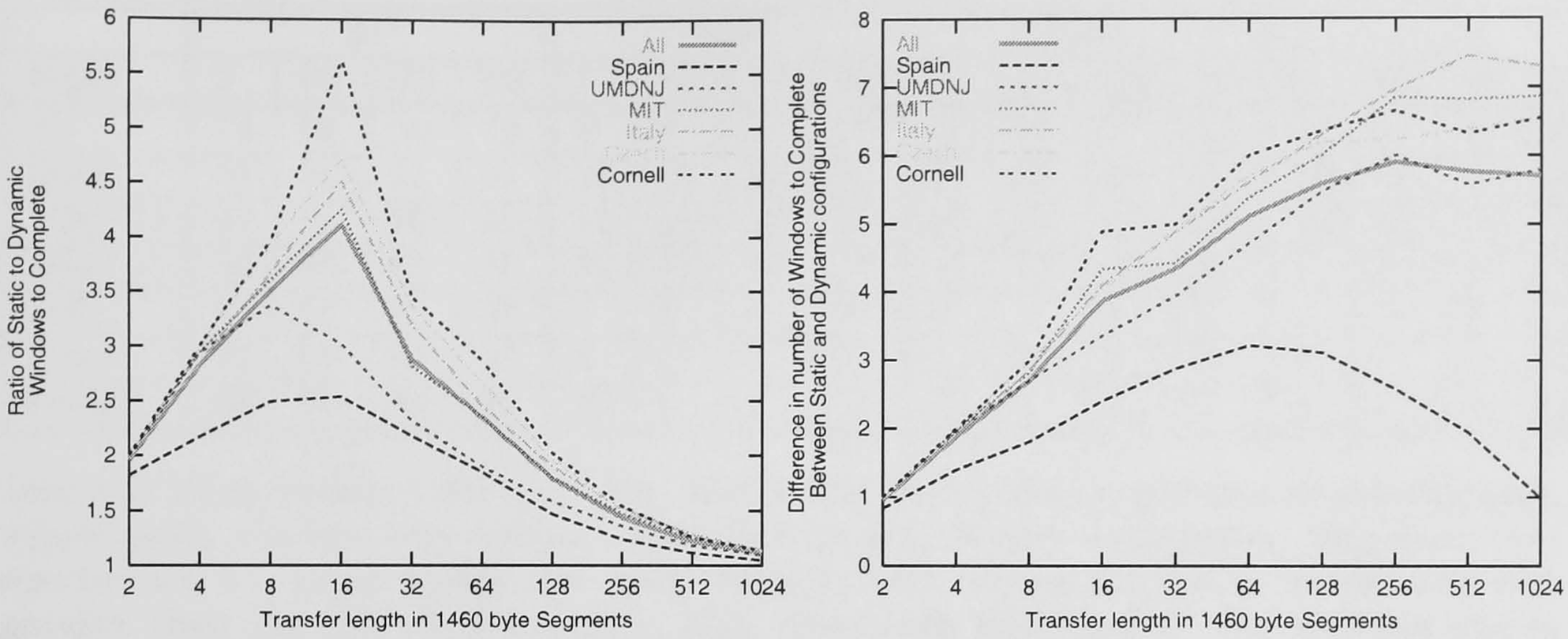


Figure 6.8: **Number of Windows required to complete a connection**

Dynamic rounds. These are calculated using the following procedure: the static connection is compared with the average for the three dynamic connections in each group[17]. Thus the comparisons are made between connections that are likely to experience similar network conditions.

Means are then computed for different transfer sizes for connections to each destination and for all destinations. These are presented graphically in Figure 6.8. A number of features of these graphs are of interest:

1. When the transfer is only one packet the configurations are indistinguishable.

2. The peak in the ratio is reached at 22 packets for most destinations[18]. A this transfer length all the packets may be transfered in one window (if the congestion prediction allows). For larger transfer sizes window limitation determines that multiple windows are required to complete the transfer. Using Slow Start with delayed acknowledgments six windows are required even when there is no congestion.

3. For long transfers the performance of the two configurations converge.

Thus the performance for normal web traffic is improved, but the performance for bulk transfers remains unchanged. Dynamic window configuration solves the problem of average sized connections not being able to utilise their fair share of bandwidth.

There is also a clear differentiation, at all transfer sizes, by destination, and as different destinations have different congestion regimes, by the level of congestion. This contrasts with TCP where short connections are unlikely to receive congestion notifications and their performance is determined primarily by size.

This section has shown that dynamic window initialisation, reduces the response time for users and increases the responsiveness of connections to the prevailing network conditions.

6.5.2 Network Load at Start-Up

The IW size, describes the behaviour at the start of a connection and the steady state behaviour describes a connection after the start-up phase has finished. It is also of interest to look at the effect

[17]The details of each group of connections is given in Section 6.2
[18]This point is not actually plotted the nearest point plotted being 16 segments

	L–				L2F				F2U				U++			
Name	S	D1	D2	D3	S	D1	D2	D3	S	D1	D2	D3	S	D1	D2	D3
All	74	73	74	73	8	8	9	7	15	16	17	17	8	9	8	9
Spain	57	54	52	55	15	18	19	16	27	26	25	25	14	16	16	20
UMDNJ	40	37	36	35	11	12	10	11	42	46	47	46	24	25	26	25
MIT	69	68	69	67	13	12	12	11	11	15	16	18	7	9	10	8
Italy	90	91	91	88	4	3	4	2	5	5	3	8	1	2	3	3
Czech	92	94	94	94	4	2	2	2	4	3	4	3	1	1	0	1
Cornell	98	98	96	99	15	1	2	1	0	1	2	1	1	1	0	1

Table 6.9: **Peak Window Before Loss: This table shows the proportion of connections, whose peak window size before a loss falls in one of four categories. The four categories are L–, at or below the lower bound, L2F, above the lower bound and not greater than the Fair Window Size, F2U above the fair window size and not above the upper bound and U++, above the Upper Bound. The connections are organised into groups by the type of initialisation and the time since the previous connection. S is statically initialised and starts five minutes after the previous connection finished. D stands for Dynamic Initialisation, The previous connection predates S1 by one second, S2 by one minute and S3 by approximately 20 minutes.**

of dynamic window initialisation on the peak window achieved by a connection before loss occurs. This is the point were the most stress is placed on the network, over a RTT, by a connection and can be characterised as the end of the start-up phase.

During start-up a linear increase in the congestion window is employed for dynamically initialised connections as opposed to the exponential increase used by SS.

A comparison of the peak window sizes for dynamic and static connections will allow an estimate of which procedure produces the most aggressive start-up.

It has been observed analytically [MSM97] and [Hoe95, Hoe96] and in simulations that on average the exponential increase of Slow Start leads to the congestion window opening beyond the Upper Bound before a loss occurs. It may then take many epochs[19] before equilibrium is reached. It would be expected that starting a connection at the lower bound and moving straight into CA would avoid this problem. Thus the peak load placed upon the network would be less even though the IW size is smaller.

These hypothesis are tested by measuring the largest window before a loss occurs. The results are presented in Table 6.9, using the same method of categorisation as was employed in the analysis of IW sizes.

If the hypotheses was born out by measurements the peak window sizes for the SS connections would be clustered above the Upper Bound. For the Dynamic connections they would be clustered around the Upper Bound. In discussing the results first connections that employ normal SS are considered. This is followed by discussion of the results for dynamically configured connections and a comparison of the two configurations.

Contrary to expectation it appears that in practice SS does not result in an overly aggressive start-up. The majority of peak window sizes are below or at the Lower Bound. The proportion of connections that are above the Upper Bound is smaller than in the Fair to Upper category. Even when connections for which no congestion is experienced and predicted are discounted the start-up still appears surprisingly conservative.

There are two interconnected explanations for this result. Firstly, delayed acknowledgments dampen the exponential increase of SS, which only takes off at larger window sizes. Secondly,

[19] Where an epoch lasts from one congestion notification to the next.

these large window sizes are never reached because of the effect of window limitation. It can therefore be expected that for large flow control windows dynamic initialisation and by passing Slow Start will result in peak windows before loss that are clearly smaller than for TCP's normal start-up.

The picture for dynamically configured connections is similar. The majority are below the lower bound. There are approximately twice as many in the fair to upper than the above upper category. This leads to the conclusion that for both SS and the dynamic configuration, start-up is less aggressive than expected or than the steady state behaviour of TCP.

When the static and dynamic configurations are compared, the dynamic configuration appears marginally more aggressive. The margin is however so small that it is not considered a problem. In conclusion, in practice peak start-up windows of both configurations are less aggressive than the behaviour of established connections.

6.5.3 Level of Congestion Feedback

This section compares the congestion feedback received by the different classes of connection. A systematic difference between classes that use the same path, would indicate that either a particular class is more prone to receiving congestion notifications, or that it receives a similar share but actually creates more congestion. In the first case there are implications for performance and in the second for network stability. Conversely a lack of systematic difference is consistent with neither class being more likely to receive or cause others to receive congestion notifications.

For each host, the congestion feedback received by the statically initialised connections is compared with the different dynamic classes, namely connections that start approximately one second, one minute and twenty minutes after the previous connection completed.

Two complementary statistics are used to compare the congestion notifications received, the mean difference and the Kolmogorov-Smirnov D which is defined as the maximum value of the absolute difference between the cumulative distribution functions of the variables being compared.

The mean difference is obtained by calculating the mean for the dynamic class and subtracting the mean for the corresponding static connections. A well known problem with the mean is its sensitivity to outliers[20], consequently it is desirable to also compare the distributions of ICNs.

If the null hypotheses that the two sets of ICN readings are drawn from the same distribution is disproved it follows that the configurations have different behaviour, if there is a consistent difference between configurations of different types it would increase certainty in the result. If the K-S D is sufficiently large the difference can be considered important for the network. Conversely if there is no difference or if the difference is small it is consistent with the readings being drawn from the same distribution or the choice between configurations having little significance for the network.

No assumptions about the nature of the distributions themselves are made and the scale chosen to represent ICNs does not affect the K-S D statistic.

The results are presented in Table 6.10. Consider first the mean difference. The differences between configurations are very small, ranging from one part in a thousand to four parts in one hundred thousand. Each of the inter connection gaps have rates of ICN smaller than and larger than the static configuration. Within this context a slight trend towards more loss when there is a larger gap between connections is discernible and the likelihood of dynamically configured windows having more ICNs increases slightly with the level of congestion.

Now consider the K-S D. For the majority of connection types there is a small difference.

[20]The mode or median are often used as measures of location in place of the mean because of their greater robustness and resistance to outliers. In this case however for most of the hosts the median value would be 0% ICNs as would the mode.

	Static	*Dynamic*					
Gap	5 Minutes	1 Sec.		1 Min.		20 Min.	
Statistic	SM	SM-DM	K-S D	SM-DM	K-S D	SM-DM	K-S D
All	0.5408	0.0091	0.1187	0.0283	0.1125	0.0259	0.1052
Spain	2.3828	0.0687	0.0471	0.115	0.0408	0.1442	0.0432
UMDNJ	0.5694	-0.0053	0.0656	0.0285	0.0936	0.0057	0.0631
MIT	0.2524	0.0054	0.1512	0.0224	0.1610	0.0166	0.1432
Italy	0.0913	-0.0267	0.2216	-0.0034	0.1891	0.0203	0.1698
Czech	0.0398	0.0085	0.1632	-0.0025	0.1581	-0.0044	0.1487
Cornell	0.0228	0.0043	0.1122	0.0043	0.1024	0.0240	0.1174

Table 6.10: **K-S Comparison of Congestion Notification Distributions: Two comparisons of congestion notifications are presented here. The Kolmogorov-Smirnov test tests to see if the distributions are the same, the result is given in the columns marked K-S D. The columns marked SM-DM give the mean of the Static minus the Dynamic percentage of congestion notifications for each category of connection.**

The lack of a clear result and the small scale of differences is consistent with there being no important difference in the level of congestion notifications received by the different configurations. This is surprising, because the initialisation of the congestion window to a large value would prevent TCP's Ack clock from smoothing packets out across a round trip time.

It is possible that this dos not matter at low levels of congestion which is when the congestion window should be initialised to a large value and when there is a high level of congestion the IW is small thereby avoiding large bursts[21].

6.6 Conclusion

This chapter has presented experiments, in which the LIS is used to generate estimates of network conditions and these estimates have been used to set the IW of certain connections. The results show that congestion information retains is value over many minutes. That the degree of accuracy with which congestion can be predicted over these time scales facilitates the use of equation based congestion control to predict fair bandwidths.

It has been further shown that the dynamic initialisation of TCP's congestion window, results in a reduction of delay for users and an increase in the responsiveness to congestion. This is achieved without increasing the peak load on the network or unfairly penalising traffic with which the network is being shared. Using dynamic initialisation and the LIS means that the load placed upon the network is not overly aggressive and the congestion feedback received by dynamic and static configurations is similar.

On balance it can be concluded that dynamic window initialisation, increases the utility that users receive from the network, increases the responsiveness of connections to congestion and does not prejudice the stability of the network.

[21] In addition it may be that a number of factors prevent TCP's Ack clock from effectively smoothing packets in any case and so bursty traffic patterns dominate in both instances.

Chapter 7

An Architecture for Sharing Quality of Service Feedback

7.1 Forward

The work presented in this chapter attempts to generalise the approach adopted to facilitate the sharing of congestion information between TCP connections to include realtime multimedia traffic. It is not supported by serious experimental validation or a rigorous analytical approach. It is more analogous to a position paper, in that various technologies are discussed, issues that may arise when trying to integrate congestion information are identified and a proposed architecture is presented. It is expected that other issues will arise when an implementation is attempted that may require substantial review of the arguments presented in this chapter. Nonetheless it is felt that the inclusion of this chapter is justified as a marker against which future work can be judged.

7.2 Introduction

This dissertation has demonstrated that improvements in the fairness and efficiency with which network resources are utilised may be achieved by measuring data streams at the edge of the network. So far the focus has been on TCP traffic as this is the transport protocol over which most Internet traffic is carried. The unicast, reliable, byte stream service that TCP provides is not however appropriate for all traffic. In particular real time video and voice streams, such as those generated by a video conference, have timeliness requirements that TCP's loss recovery mechanisms prevent it from meeting.

This chapter broadens the discussion to investigate the feasibility of extending congestion information sharing to include such traffic types. Three questions are addressed. 1) Is it practical for a third party monitor to obtain congestion information from real time data traffic? 2) How can such information be integrated with traffic statistics obtained by passively monitoring TCP connections? 3) How may such information be productively utilised by applications?

In addressing the above questions the discussion focuses on the Real Time Protocol (RTP). The choice of RTP is motivated by two considerations. Firstly, it is the main Internet protocol that is used for real time traffic. Secondly, it was designed to be able to accommodate applications with a broad range of requirements. Whilst a Location Information Server, which was capable of monitoring TCP and RTP traffic would by no means cover all Internet applications it is suggested that the issues that arise when examining such an extension will illuminate to what extent the LIS [Rud00] architecture has general applicability.

The discussion is structured into four sections. In the first section an analysis of RTP is presented, which discusses; the design goals of the protocol, the protocol structure and the methods used to provide QoS feedback. In the second section RTP extensions to the LIS are presented, these include mechanisms for obtaining information from RTP sessions and for making QoS feedback available to RTP applications. The third section presents the design of a specific synchronous conferencing groupware application that utilises an LIS to provide a predictable QoS to its users. In the fourth section issues in multicast congestion control are discussed from the perspective of sharing QoS feedback.

7.3 Real Time Protocol

Improvements in fiber optics and switching technology meant that by the start of the nineties the possibility of networks that integrated services such as video, voice, images and text appeared as a practical possibility for the first time. There was however much about existing networks that needed to be addressed for the vision of a multi-service internet to be realised.

One approach was to design an internet architecture that met the needs of both data and real time traffic. The Multi Service Network (MSN) was such a project. It provided a protocol family that exploited the ATM compromise between the needs of real-time and data traffic by facilitating the interconnection of local [ID91], metropolitan [Gre89], wide area [TL89] and desk area [BaDMP95] ATM networks, thereby facilitating the creation of multi service internets [Mca89, LTM83]. It represents a new internet architecture, which could efficiently and effectively support the sharing of the network by real time and data traffic.

A second approach was to develop the existing Internet architecture to meet the needs of real time traffic. The Resource ReserVation Protocol (RSVP) allowed resources to be reserved therefore enabling QoS guarantees [BZB+97] to be made and met. IP multicast [Dee88] allows real time (and other data) to be multicast to multiple receivers thereby saving bandwidth on links close to the source.

RTP [ASC+96] builds upon the IP multicast and unicast datagram services and provides QoS feedback that enables the creation of adaptive real-time applications. In part this explains the breadth of deployment achieved by RTP. The design was strongly influenced by two concepts: Application Level Framing (ALF) and Integrated Level Processing (ILP) [CT90].

The remainder of this section provides a description and analysis of RTP, which prepares the ground for the presentation of techniques for the sharing of congestion information between TCP connections and RTP sessions. First the central ideas that shaped the design of RTP are discussed. Next an overview of the functionality and structure of the protocol itself is provided. This is followed by a detailed discussion of RTCP, which sets out the different types of control packet, the functions they fulfill and the information they contain.

7.3.1 Application Level Framing

Although today the concept of Application Level Framing (ALF) is often imbued with general meaning the original formulation was quite specific. The motivation for ALF stemmed from the observation that the best way to respond to packet loss is application specific. For example TCP's strategy of suspending transmission and retransmitting from the transport layer's copy may not be appropriate for all applications. Clarke and Tennenhouse [CT90] observe that some applications may be able to accept some data loss and continue without a pause, for others it may be appropriate for retransmission to occur, but for the application to provide the data.

> *A general purpose data transfer protocol ought to permit any of these options to be selected: buffering by the sender transport, recomputation by the sending application, or proceeding without retransmission.* [CT90]

For the application to be able to respond to loss, the nature of the loss must be meaningful to the application. A consequence of TCP's stream abstraction is that even if the sequence numbers contained in a lost packet could be communicated to an application they would have no meaning, apart from the application being made aware of the existence of loss and its volume.

This is the problem that ALF addresses directly. If the application breaks the data that is to be transmitted into Application Data Units (ADUs) and lower layer protocols preserve the framing established by the application, then if an ADU loss is reported to the application it will be in a position to respond appropriately. Thus ALF means that ADUs take the place of packets as the unit of data manipulation.

The architectural principle of ALF also supports the engineering approach called Integrated Layer Processing (ILP). This permits the processing of all data manipulation steps in integrated loops. It is possible, as multiple protocol layers terminate in the same location; the end host. It is desirable because a naive approach of processing each packet sequentially as it passes through each layer is inefficient. It is facilitated by ALF as each layer works on the same unit of data, the ADU.

7.3.2 Overview of RTP

> *RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services. The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner scalable to large multi-cast networks, and to provide minimal control and identification functionality* [ASC+96].

The services provided include: definition of the payload type, time-stamping of packets which facilitates timed play back, a monotonically increasing sequence number, identification of the source of the packet and feedback on the quality of service received by the stream [CP00]. In the following discussion the main components of the RTP protocol are considered. There is a discussion of the context within which an RTP session may be expected to exist, the scope of the session and the components that make up a session are all discussed. This is followed by discussion of the structure of the protocol and how this relates to ALF and the generality of QoS feedback.

RTP Sessions

The practical stimulation for the development of the Real Time Protocol (RTP) arose from the experience of designing multi-point video conferencing facilities for the Internet [MJ95]. Rather than designing a centralised video conferencing application individual tools were developed that could be combined together in appropriate ways to provide the particular functionality required by each specific conference.

This approach to Internet video conferencing is best summarised by the concept of Light Weight Sessions (LWS) [Jac94, McC99]. Each session consists of a collection of multicast senders and receivers, who communicate using one or more multicast groups and a selection of (Mbone) tools. The approach is loosely coupled and decentralised. It is built upon an "announce/listen" model and the maintenance of soft state by receiver's [SEFJ97]. Each member of an LWS periodically announces their presence, receiver's listen for announcements and maintain state for each participant in the conference. In the absence of fresh announcements such state is timed out. With this approach the need for an explicit conference set up or centralised conference controller is avoided which makes the architecture scalable.

Within the context of an LWS several RTP sessions may exist, for example in a multi-media conference each media would be transported in a separate RTP session. An RTP session provides

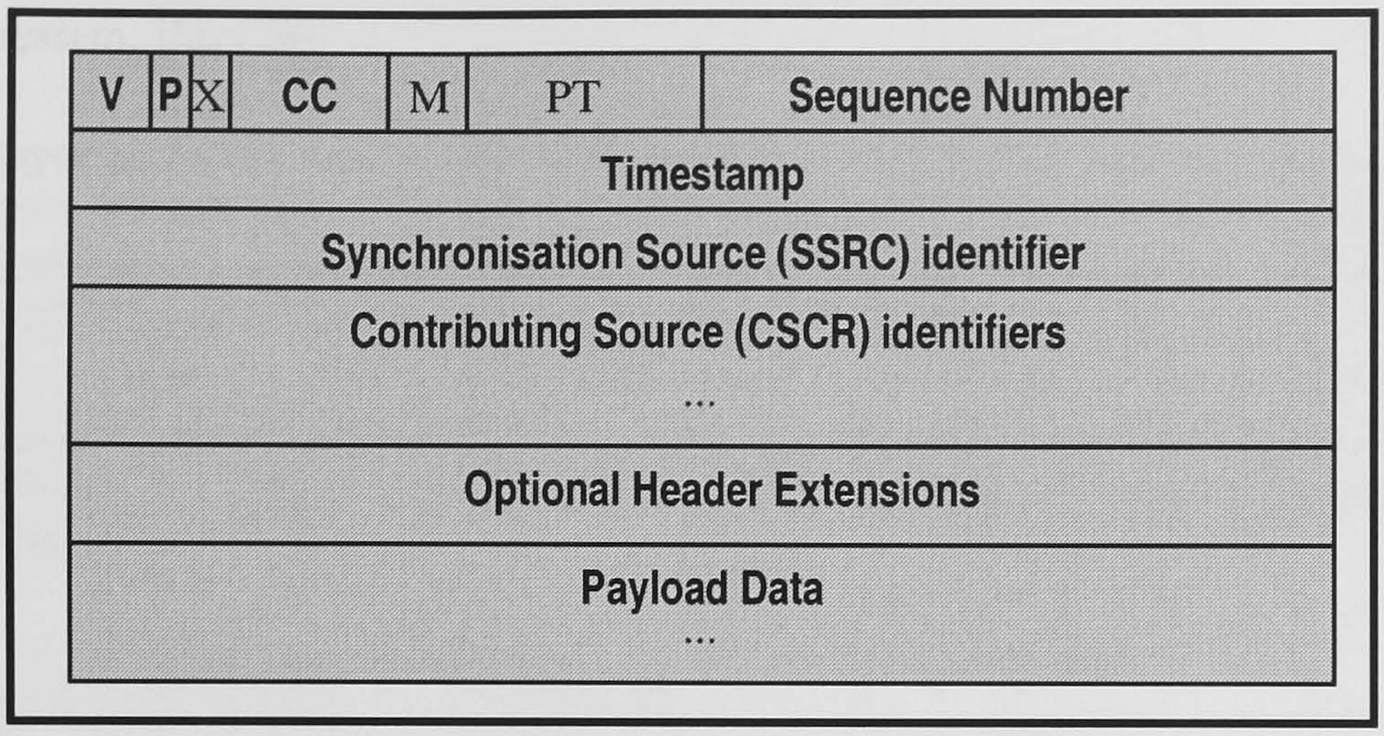


Figure 7.1: **RTP Data Packet Header: the header can be easily extended to facilitate Application Level Framing.**

the association between a set of participants that are communicating using RTP. The session also defines the scope within which information about the state of the network is collected and disseminated to participants by the Real Time Control Protocol (RTCP). In this respect an RTP session is analogous to a TCP connection. A session is uniquely identified by an IP multicast address and a pair of port numbers, one for data and one for control traffic.

A session may contain multiple data sources and receivers. Timing information and sequence numbers need to be supplied to every receiver for each source to allow it to detect loss and pace out the playback of packets. A Synchronization Source is the origin of a stream of RTP packets, and is identified by a 32 bit Synchronisation Source (SSRC) identifier, which is carried in the RTP header. All packets from a synchronization source form part of the same sequence and timing number spaces. A receiver therefore groups packets by synchronization source for playback. An SSRC is globally unique within a particular RTP session, however different SSRC identifiers may be used for the same source in separate RTP sessions. If a participant generates multiple streams (e.g. from separate cameras) in a single RTP session then each must have a separate SSRC.

RTP Protocol Structure and Application Level Framing

Whilst the motivation for ALF arose within the context of a single general purpose data transport protocol and the term itself applied specifically to the application framing of data to facilitate efficient recovery from packet loss neither of these ideas are embodied in the design of RTP.

RTP is not the general purpose protocol, which Clark and Tennenhouse argued that ALF could facilitate. Rather it is designed to meet the needs of the specific class of applications, which can be characterised as requiring soft real time delivery of data and being resilient to small levels of loss. These applications respond to loss by "proceeding without retransmission". Consequently RTP is complementary to and not a replacement for TCP and there exists classes of traffic for which neither RTP nor TCP are appropriate. For example interactive real time traffic such as that generated by a distributed white-board [McC92]. For this type of traffic protocols which support reliable transport are required for example [FJM95, FJL+97].

Within the confines of the class of traffic for which RTP was intended flexibility was achieved by designing it as an *extensible protocol framework* , which may be tailored to the needs of specific applications. Thus there is a real sense in which RTP is influenced by ALF.

> *The central idea of ALF is that an application's semantics should be reflected in the design of its network protocol to optimise its performance across the network and into*

the end system. [McC99]

RTP is not a layer as in the traditional OSI model [Hub80] but is rather a hierarchy of interdependent protocols. At the base is the RTP specification [ASC+96], which provides the framework. At the next level exists profiles the main example being the audio/visual profile [AS96]. Here elements of the RTP header may be defined, see Figure 7.1. For example; for audio the marker bit indicates the first packet in a talk burst and for video it represents the last packet in a frame, the profile also defines the clock frequency to be used. At the final layer payload formats define the precise way in which bit streams are framed into RTP packets, examples of payload definitions include [KNF+00].

7.3.3 Real Time Control Protocol

Introduction

RTCP provides three functions; it is the mechanism whereby feedback on the Quality of Service being experienced by data streams is distributed to session participants; it makes available sufficient timing information to allow streams from separate sessions to by synchronised, and it allows SSRCs to be mapped to names that are meaningful outside of the context of a particular RTP session.

In fulfilling the above functions five types of RTCP packets; Sender's Reports (SR), Receiver's Reports (SR), Source Description packets (SDES), BYE packets and APP packets, are utilised. Sender Reports (SR) are used to distribute information about received and transmitted packets from participants in a conference who are actively sending packets. If a conference participant is not an active sender then, Receiver Reports (RR) will be used to carry reception information to other participants. SDES packets provide a description of a source. The most important piece of information is the Canonical Name (CNAME), which allows a participant to be identified across RTP sessions. A BYE packet is used to indicate the end of a participants involvement in a session and APP packets are reserved for application specific functions.

In the remainder of this section three aspects of RTCP are discussed: the QoS information that is available, the mechanism for mapping from SSRCs to IP addresses and the effect of controlling the volume of control traffic on the timeliness of reporting.

Sender and Receiver Reports

Sender and receiver reports are the RTCP mechanism that is used to distribute Quality of Service information to the participants in a session. The format for a SR is given in Figure 7.2; it is made up of three sections, the header, the senders information and reception information. There is one block of receiver information for each source heard since the previous report. A receiver's report differs from a senders report in that it does not contain the second group of fields, which describe transmissions.

1. Header: The header contains; the version number of RTP, a padding bit, a count of the number of reception reports, a packet type field, the length of the packet and the Synchronization Source identifier (SSRC), which identifies the packet source.

2. Sender Information: this section provides information about the volume of data that has been transmitted by the source and the time the report was sent. The NTP time stamp indicates the wall clock time a report was generated at, the accuracy of the time stamp is not communicated. The RTP time stamp corresponds to the same time as the NTP time stamp but is compatible with the time stamp used in RTP data packets. The NTP and RTP timestamps allow a mapping between wall clock and RTP session time, which in turn

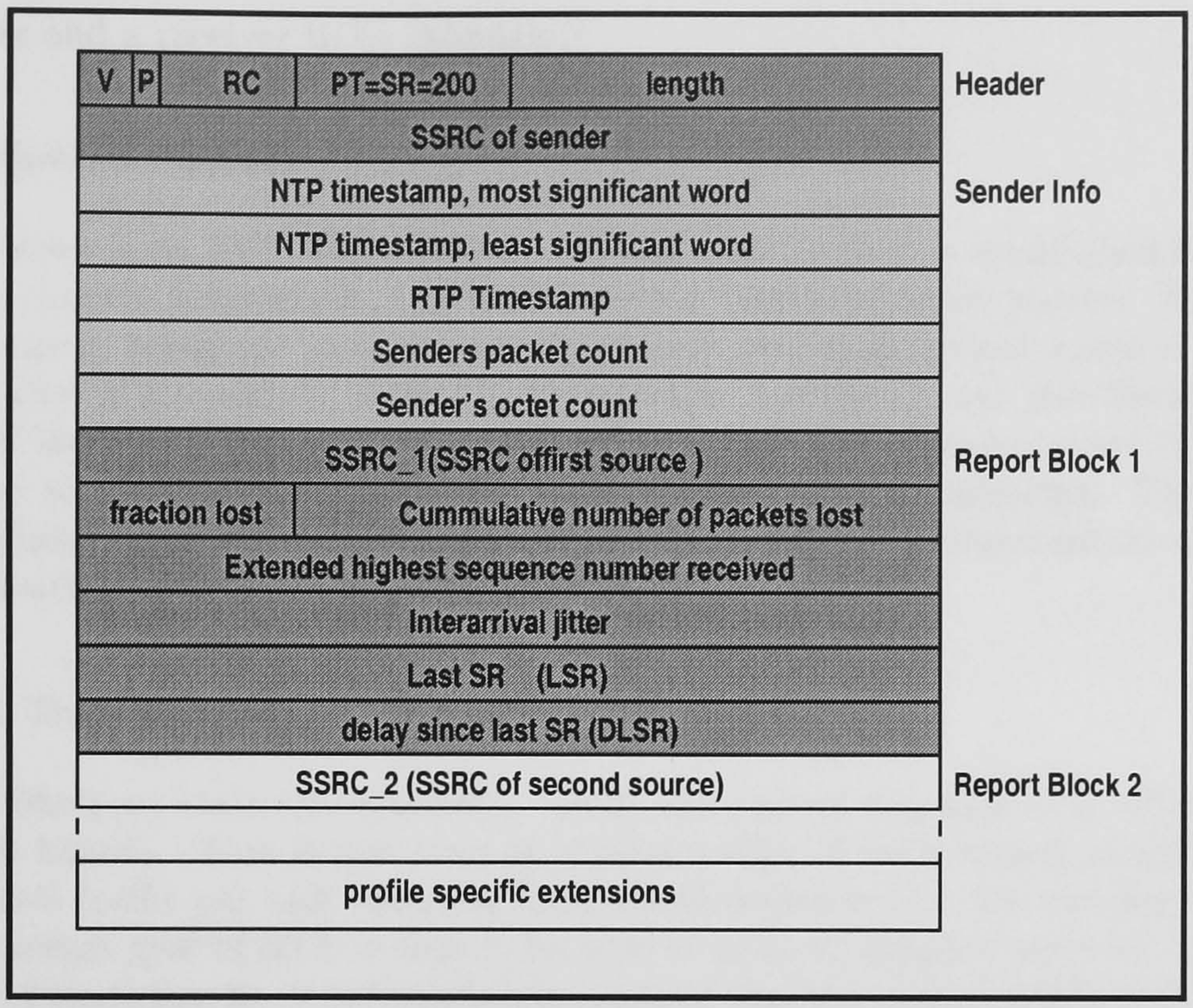


Figure 7.2: **RTCP SR Format: Real Time Control Protocol header format for a Source Report packet.**

allows an application to synchronise the play out of streams carried in separate sessions. The senders packet count gives the number of data packets that have been transmitted by the sender since since the start of transmission or the last time the SSRC was changed. The senders octet count gives the total number of payload bytes transmitted. Together with the timestamps, these may be used to determine the rate of transmission and the average data packet size.

3. Reception Report Blocks: The number of RR blocks depends on the number of sources that have been heard by the receiver since the last report. Each block contains statistics on packets received from a single synchronization source. The SSRC identifies the source that the reception information describes. The fraction lost field is an estimate of the proportion of RTP data packets that were lost since the last SR. This is defined as the number of packets lost over the number of packets that were expected and is an 8 bit value.

   The cumulative number of lost packets is the difference between the number of packets expected and the number of packets received since the start of transmission. Packets that arrive late are not counted as lost. The number of packets expected is defined by the initial and last sequence number received. The extended highest sequence number gives the highest sequence number that has been received and the number of sequence number cycles that have been completed.

   The inter-arrival jitter field contains an estimate of the variance of the RTP data packet inter arrival time. It is measured in time stamp units. The estimate of inter-arrival jitter allows the proportion of packets that would arrive too late for display given a particular play out buffer size to be calculated. To be valid the packets must have been evenly spaced when sent.

   The LSR field echoes the middle 32 bits of the NTP time-stamp from the last received senders report. The DLSR field gives the delay between the LSR and the dispatch of this report. Together with the reception time, the LSR and DLSR fields allow the RTT between

the sender and a receiver to be calculated.

Source Description Packets

The mapping between an RTP SSRC and a conference participant is established by RTCP SDES packets, which contain information associated with a synchronisation source. This information may include a name, telephone number, e-mail address and geographical location. The one piece of information that is mandatory is the CNAME, which has the format user@host, where user is a username and host is a fully qualified domainname or the ascii representation of the IP address, and is intended to allow an association to be established between streams. Together with the timing information contained in SRs this facilitates cross media synchronisation of data streams; for example allowing audio to be synched with video.

Frequency of Transmission

Data traffic is likely to scale automatically. Only one person can talk in a video conference at a time (and be heard). This is not true of control traffic, if each source generates a constant amount of control traffic per unit time the total would increase with the number of participants. An important design goal of RTP is that it be able to scale to support sessions that involve very large numbers of participants. It is therefore important that the rate at which RTCP packets may be generated by an individual source is inversely proportional to the number of participants in a session.

RTCP is designed so that control traffic will take up a small known volume of the total session bandwidth, thereby ensuring that RTP's primary function of data transport is not impaired. If RTP was to be used with RSVP the proportion of control traffic would need to be known in advance so that it may be included in any traffic specification. The existence of a known proportion also allows individual sources to independently calculate their share of the control traffic bandwidth. The figure of 5% was chosen to fulfill both these criteria and a recommended algorithm to calculate the frequency of RTCP packets is included in the appendix A of RFC 1889 [ASC+96].

Bandwidth is limited by sources calculating a minimal interval between sending compound RTCP packets. When there are a small number of sources this interval will be small. When there are a large number of sources the amount of control traffic will be limited. A number of additional factors are included in determining the actual interval between packets. Senders are allocated a minimum of 25% of the total control traffic. A minimum interval of five seconds is specified. The actual interval is randomly varied, between 0.5 and 1.5, of the interval calculated by each source, to avoid synchronization between sources [FJ93b]. Similarly the first RTCP packet is delayed randomly by between 0 and 0.5 of the calculated interval to avoid bursts resulting as a consequence of multiple sources simultaneously joining a session.

It is possible to limit the rate at which hosts generate RTCP traffic and not adversely impact upon the efficacy of the protocol because of the type of control information that is carried and assumptions about the characteristics of the data being reported upon. In particular RTCP is not used to report dropped packets in a way that would enable a data source to retransmit that packet in a timely manner. Consequently whether a report arrives every ten seconds or every minute may limit the speed with which an application can adapt to changing network conditions, it will not affect the time taken to recover from errors as it is assumed that no such recovery is necessary.

Thus the way in which RTCP meets the scalability requirement for RTP is only appropriate if timely error reporting is not required. Making RTP scalable depends upon assuming that the correct response to loss is to ignore it, which in turn limits RTP to traffic types that do not require reliable transport.

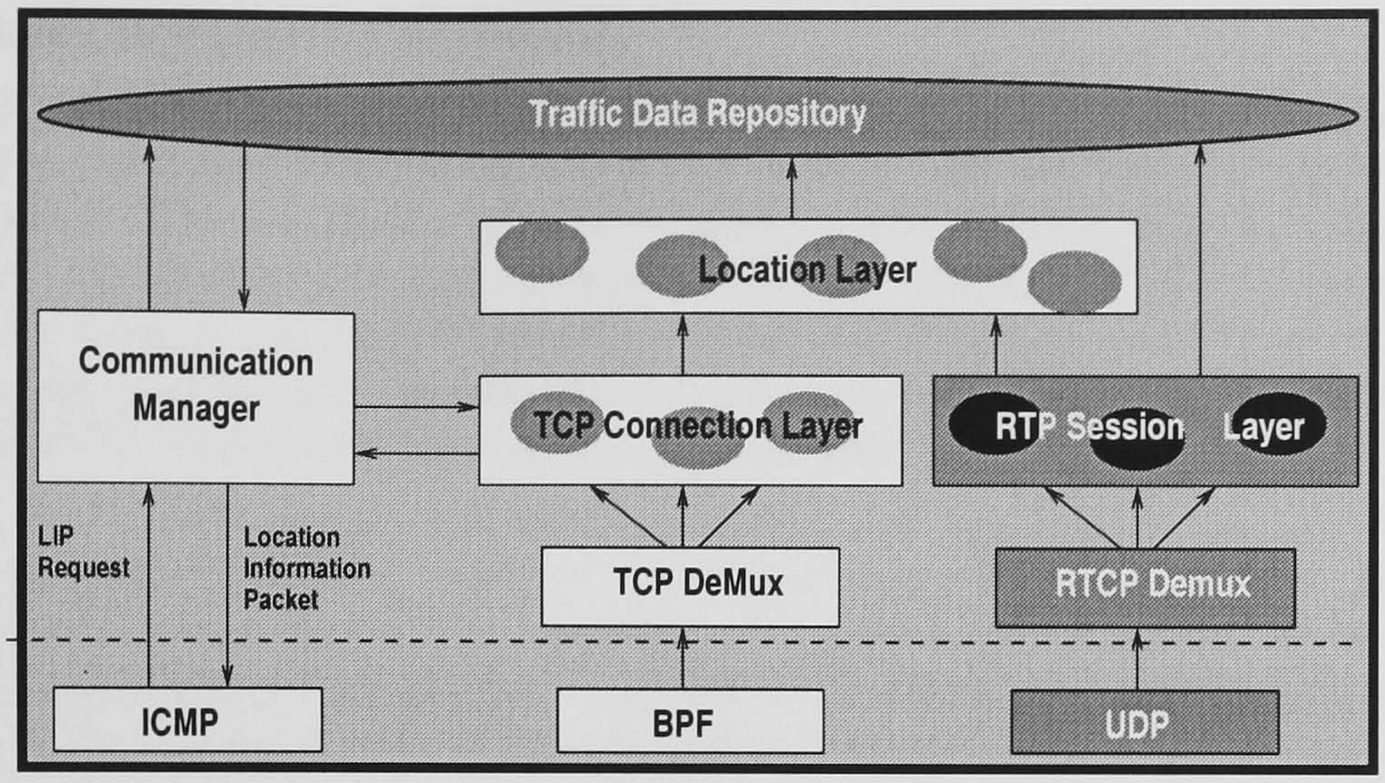


Figure 7.3: **LIS extension for RTP traffic: A separate mechanism is required for extracting readings, but integration can occur at the location level. In addition long term statistics can be held in a database called the Traffic Data Repository.**

7.3.4 Summary

RTP is complementary to TCP in that it provides the end-to-end network functions that are suitable for applications transmitting real time data such as audio, video or simulation data over multicast or unicast network services. There are other traffic types for which neither TCP nor RTP are appropriate [MHKE99, CP00].

The design of RTP was strongly influenced by the design philosophy of ALF. Consequently RTP is structured as an interdependent hierarchy of extensible components. Payloads framing may be specified for particular applications. The implementation of RTP as a layer within applications means that applications get direct access to the framing of ADUs. A reporting mechanism is also provided whereby applications can learn about the QoS experienced by data streams and react in an appropriate application specific way.

This RTCP feedback allows monitoring of data delivery in a manner appropriate for a small conference, and is scalable to large multi-media events. RTCP also provides a mechanism, which can be used to map between identifiers that are internal to a particular session and a conference participant.

The way in which RTCP provides each of these services has been specified in some detail. Of particular significance is that the provision of QoS information is independent of the application that is to make use of that information. This facilitates the sharing of congestion information between RTP sessions and is made possible, as the QoS information itself is a function of the state of the network rather than the type of application. This in turn means that a third party monitor may make sense of the QoS feedback without being aware of the applications semantics.

Although RTCP is nominally extensible, in that applications may define APP packets, the generic nature of feedback and the detail with which that feedback is specified stands in contrast to the framework approach adopted for RTP. In addition RTP's application of ALF is applicable to the data transport and not the control plane.

7.4 Quality of Service Feedback Integration

The architecture used in extracting congestion information from RTP streams is similar to but differs in significant ways from that used by the LIS for TCP. In this section an overview of the

architecture shown in Figure 7.3 is provided.

The absence of acknowledgments means that it is not practical to determine packet loss rates, round trip times, or other information about the state of the network by passively monitoring RTP data streams. The possibility of a third party monitor being able to make use of RTCP reports was however also foreseen and incorporated in the design of RTP. It is possible for a third party monitor to subscribe to an RTP session, receive RTCP reports but not receive or send data. For this reason the possibility of extending the traffic that an LIS extracts congestion information from to include RTP traffic exists as a practical proposition.

It is proposed that RTCP sender and receiver reports will be used by the LIS to obtain traffic statistics and integrate them with statistics extracted from TCP data streams. In this approach the network retains responsibility for detecting and signalling congestion through packet loss. RTP receiver's keep count of the number of packets received and the number of packets expected, and multicasts this information to all participants in a session. The Location Information Server acts as a third party monitor, it receives RTCP packets but does not receive data. From the RTCP reports the LIS receives it extracts the QoS information, associates it with end-point IP addresses in the LIS's RTP session layer. This information is then passed to the location layer where it is integrated with congestion information obtained from other relevant RTP sessions and TCP connections. In this way RTCP reports are used to help maintain running estimates of the network state on the paths between two endpoints.

The assumption made earlier that an LIS need only provide information for WAN traffic is weakened when its functionality is extended to provide services for RTP applications. The higher bandwidth requirements of video traffic may stress even a well engineered collection of LANs, such as a campus network. The extraction of QoS information from RTCP reports is not dependent on the LIS being able to directly monitor data streams. This means that the LIS can compile information from traffic, which its location precludes it from directly monitoring. Therefore the situation of the LIS on the gateway to the WAN, which facilitates monitoring of wide area TCP traffic, does not prevent collection of relevant statistics on local RTP traffic.

The LIS detects new TCP connections by observing SYN packets. A similar mechanism is not appropriate for RTP. Instead an Session DiRectory (SDR) [Jac] server's announcements are monitored to detect new sessions and the LIS queried directly be RTP applications that require estimates of network conditions. The likely longer lifespan of an RTP session, when compared to an HTTP connection, relaxes the timeliness constraint on delivering QoS data and therefore makes this approach more suitable to RTP traffic.

In this section the issues are discussed that arise in designing the extension of the LIS to integrate RTP traffic statistics. First, the design of an RTP session layer which extracts QoS information from RTCP packets is presented. This is followed by discussion of the integration of TCP and RTCP readings in the LIS location layer. Finally a Traffic Data Repository, which stores past QoS information, is motivated and described.

7.4.1 RTP Session Layer

The RTP session layer in the LIS plays the same role in relation to RTP as the connection layer does in relation to TCP. The session layer receives and processes RTCP packets. To achieve this state is maintained, for each active session, that will enable appropriate QoS feedback to be extracted from RTCP packets, which is in turn then communicated to the location layer.

When the start of a session is detected an object is created, which will contain all the state specific to that session. Participants in the session are identified through the receipt of SDES packets, which contain the mapping between CNAME and SSRCs. From the CNAME the LIS establish a mapping between SSRCs and IP addresses. This may in turn be used to facilitate the information reported by RTCP about one session being associated with network endpoints. A list of mappings from SSRCs to IP addresses is kept, for each relevant pair of IP addresses. A pair of IP addresses is

considered to be relevant if they both belong to the session, at least one of them is a local IP address and at least one of them is a sender. A list of all relevant participants is also maintained with a timestamp for the last activity recorded for each participant. This allows inactive participants who have left the session to be identified and their corresponding state removed [SEFJ97].

Congestion Estimation

Each receiver's report contains the fraction of packets lost since the previous report, which has been calculated by dividing the number of packets received by the number of packets expected since the last report. This is an obvious candidate for determining the current level of congestion and is an 8 bit value giving 256 levels. A reasonably engineered network will have low levels of loss, thus most of the levels will never be used as the step size between each level is approximately 0.4%. It is therefore desirable to have a finer grained measurement of packet loss. By maintaining state for each pair of IP addresses within a session it is possible to obtain both a finer grained proportion and the number of packets over which the proportion is calculated, which in turn enables a weight to be associated. To achieve this two values for each direction need to be maintained between reports; the cumulative number of lost packets, and the extended highest sequence number received.

The number of packets lost since the last Sender's Report is given by the current minus the previous cumulative number of packets lost. The number of expected packets is the difference between the current and previous highest sequence number. Thus the fraction of packets lost can be calculated to a precision which is defined by the number of packets received. The number of packets expected enables the weight that should be attached to the statistic to be calculated. This loss-fraction should agree with the loss-fraction in the senders report once rounding is taken into account and if no intermediary reports were lost.

There remain three issues with the accuracy yielded by this method of calculating the probability of loss. 1) The number of packets received includes all packets so if duplication of a packet occurs, either by the source or the network, it is counted as one of the packets received. The number of packets expected is however defined by the highest sequence number so far received. Thus duplicate packets are not counted in the total for packets expected. As the number of duplicates is not communicated in a receiver's report it is not possible to tell from the report whether and to what extent losses are being underestimated. The estimate of packet loss and therefore of congestion may consequently be an underestimate. 2) Difficulty arises when the meaning of a lost packet is considered. When determining the level of congestion on a path by observing TCP traffic the LIS counts only some dropped packets as indications of congestion. A burst of packets is counted as only a single congestion notification. In calculating the loss statistic for RRs RTP does not however distinguish between losses; all are counted. This may lead to the loss statistic in the RR being an overestimate of the level of congestion on a route, but the rate-based pacing of packets by RTP applications will also mean that the data is less bursty and therefore less suceptable to multiple losses. 3) By observing sequence numbers at the receiver it is not possible to identify packets that have been dropped since the last packet was received.

Round Trip Time Estimation

A host that is participating in an RTP session is able to use the information contained within RTCP packets to calculate the RTT to other participating hosts. Assuming that the current time is known, the round trip time can be obtained by determining the difference between the current time and the LSR field and subtracting the DLSR field.

If no senders reports have been received from the source being described by the originator of the SR, then the DLSR and LSR fields are set to zero, consequently the RTT cannot be determined. As the LIS will be acting as a third party monitor it will not be generating traffic and so the LSR and DLSR field of incoming senders reports will be set to zero. The LIS will not be able to determine RTT using this methods.

There are alternative approaches, both of which have drawbacks associated with them. The NTP timestamp on the incoming SR could be compared to the current time and the delay between the LIS and the source determined. This has two problems. Firstly, the route may be asymmetrical with a significant difference in the delay encountered in each direction. Secondly, issues of clock synchronization arise which may not be easy to resolve. Thus it would be difficult to determine whether the RTT estimate was accurate. In addition, unlike the loss statistics where the congestion estimate is based upon many readings, each senders report would yield only one RTT reading. When the course granularity of report generation is considered, this limits the usefulness of RTT estimation from senders reports and makes the estimation of the variance in RTT is impractical. A second approach would be for RTP end-points to do the RTT calculation and communicate this to the LIS. This requires modification to existing application, but is feasible where a new application is being designed which wishes to make use of the services an LIS can provide.

Rate

The rate of packet generation and data transmission can both be determined from the receipt of RTCP packets. The difference between the previous and the current NTP timestamp, gives the time period that the report covers. Consequently time stamp state needs to be maintained between reports in order to calculate the data and packet generation rates from the senders packet and octet counts. In addition the average packet payload can be obtained by dividing the octet by the packet counts. It is useful to know the characteristics of the data stream, as a low bandwidth stream which suffers little congestion does not necessarily imply a high fair bandwidth.

Jitter

Determining the amount of jitter a stream will experience allows the play-out buffer for real-time media to be properly configured. Too large a buffer will increase delay and too small a buffer will result in packets arriving after they should have been played thereby reducing the quality of audio and video playback. A measurement of inter arrival jitter is included in RRs. This is an estimate of the statistical variance of the RTP data packet inter-arrival time, measured in time stamp units and expressed as an unsigned integer. As the statistics generated by the LIS have been primarily aimed at non real-time traffic no estimation of jitter is currently made. It may however be appropriate to add this facility and make it available to applications that utilise real-time communication.

Summary

This section has discussed the extraction of Quality of Service information from RTCP senders and receiver's reports. Appropriate mechanisms for detecting the start of a session and determining the IP addresses of RTP synchronisation sources has been outlined. It has been shown that the LIS acting as a third party monitor can obtain estimates of congestion, jitter and the data rate. It has also been shown that obtaining reliable RTT readings is more circumspect. In the next section the integration of this Quality of Service information with readings from TCP data will be discussed.

7.4.2 LIS Location Layer

The fundamental reason why it is possible to integrate congestion information derived from TCP and RTP flows is that they are measurements of the same thing; the state of the network. There is however an important difference in the timescales with which data is collected. RTCP reports from each source are separated by a minimum of five seconds and for sessions involving many

users the granularity increases considerably. This is considerably courser than when monitoring TCP traffic, where feedback on the absence or presence of congestion and estimates of RTTs can be obtained each RTT that data flows. Consequently the question of whether the information is useful needs to be addressed. Some authors have suggested discarding congestion information that is more than a few RTTs old [MJS99]. In Chapter 6 it was shown that congestion information remains valid for long periods of time. This suggests that the course granularity of reports may not be a barrier to utilising the information they contain.

The approach taken to integration is simple. Once the RTP session layer has finished processing a receiver's report, it returns to the location layer the following data; the IP addresses of the sender and receiver, the wall clock time that the report covers, the number of packets included in the report along with the probability of packet loss and the data and packet transmission rates. From the IP addresses it is possible to determine which foreign Location the readings should be associated with. Structure has to be added to the Location Layer to facilitate holding statistics on pairs of local IP addresses. Once the location has been determined it can be updated. The probability of packet loss is converted into the number of ICNs and number of packets; the current bin is then incremented for both these values. If the time stamps indicate the report spans more than one bin the ICNs and packets are distributed proportionally. A round trip time reading would be straightforward to integrate. Single RTT readings communicated to the location layer consequently the TCP algorithms for updating the smoothed round trip time and variance in RTT can be applied directly. The location layer holds the maximum recent window size. The rate of data transfer can be converted to window size if the current RTT is known [GS99]. In this case it can be used to update the recent maximum.

In this section the mechanism proposed for integrating TCP and RTP readings into the Location Layer of an LIS have been outlined. With this approach the path descriptions will be updated in a timely manner and estimates be available with a minimum of delay.

7.4.3 Traffic Data Repository

Three factors contribute to making a Traffic Data Repository (TDR), where timestamped data on network characteristics is stored in a database, a useful extension of the LIS architecture. With RTP traffic the timeliness constraint on the delivery of congestion information is likely to be less severe than TCP, where there is essentially one RTT between the detection of a connection and the start of data transfer. Consequently, holding data in a database and using this to calculate estimates is a feasible approach.

The lifespan of many TCP connections is very short and for longer connections intra-connection adaptation means that the effect of initialisation is short lived. A prediction based upon current information is therefore appropriate. RTP sessions are however likely to be long lived, parameters set at the beginning may determine for example: the playout buffer size for the duration of the session. In deciding the buffer size it would be important to know the likely variance in RTT over some lengthy timespan. A simple running total, as provided by the Location Layer, would not be appropriate.

The strong diurnal pattern of network usage, and therefore of RTT and congestion, means that if there have been no readings for a particular location for a period of time it may be better to use data from a previous day at the same time than the most recent data. A TDR could facilitate this.

The TDR provides a central repository for information about network paths. It has three main modules; an interface for receiving reports, a module for receiving queries and generating replies and a database for holding previous reports. Consequently information gathered from one session may be made available to future sessions.

The TDR presents two types of query interface. One allows simple queries to be made, such as the predicted level of congestion and variance in congestion between two end points, over some time

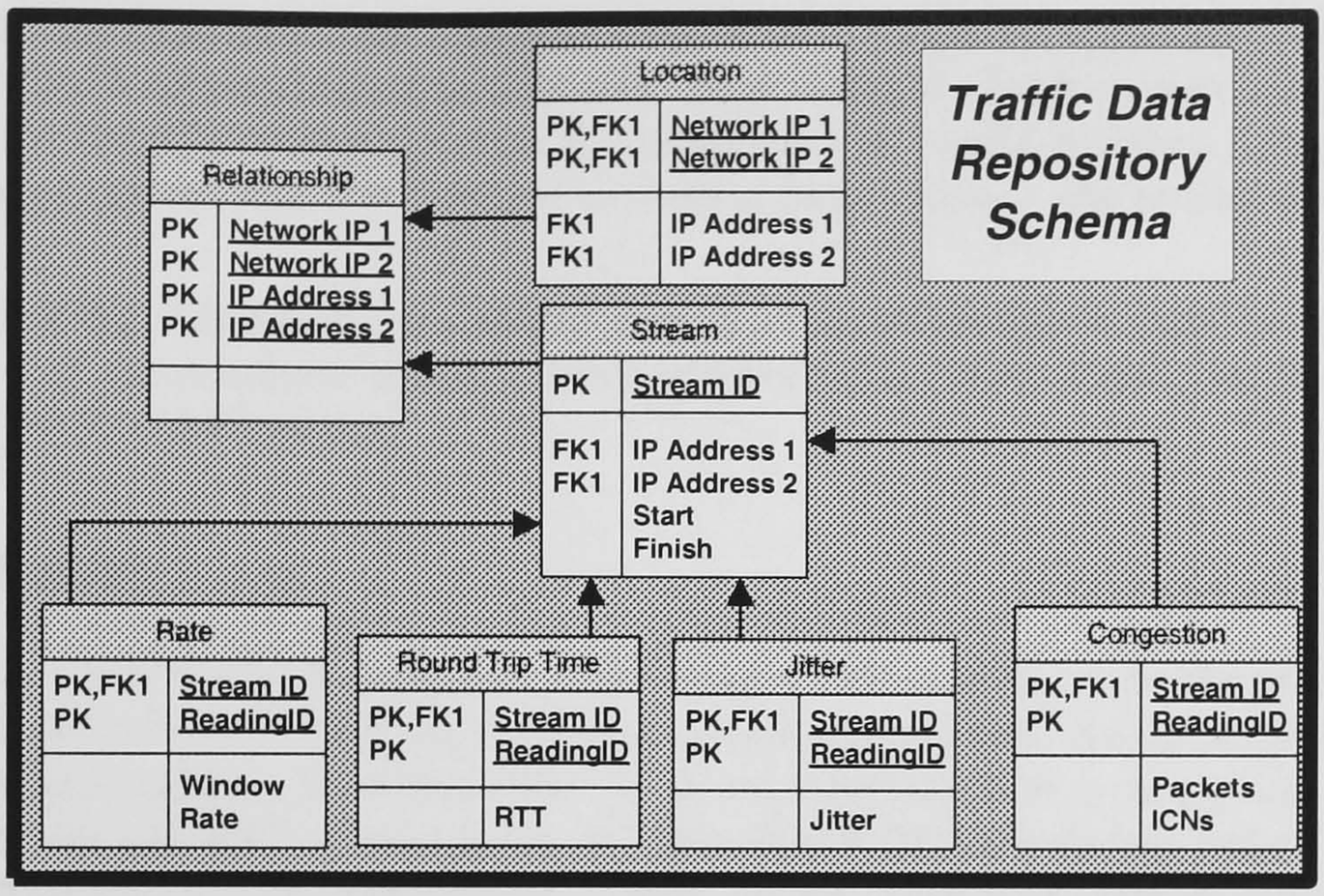


Figure 7.4: **Traffic Data Repository Schema: Readings for the RTT, data rate, jitter and congestion are all associated with a particular stream.**

period. The second allows the application to export an algorithm for determining the prediction. This allows the application to determine how the predictions are made but avoids the need to transport large amounts of raw data across the network.

Information may be given to the TDR as data is aged out of the TCP location layer. For example as a block of congestion information is aged it may be stored in the TDR for future reference. The TDR may also receive data from the LIS RTP session layer, as readings are passed to the Location Layer they may also be passed to the TDR. Validated applications may also insert data into the TDR, for example RTT estimates made by RTP end points may be obtained in this way.

The information held within the TDR is structured spatially, temporally and by type; see Figure 7.4. The structure is hierarchical. At the top level, a location is defined by an IP[1] address and network mask. All other data is associated with at least one location. At the next level a stream provides the temporal component; it is uniquely identified by a stream ID and contains a pair of end point host addresses and start and finish times. Each stream has associated with it a number of readings, which are structured into different types and provide information about the the quality of service experienced by the stream and about the stream itself. Congestion, jitter and round trip times describe the quality of service. Rate or window size describes the state of the network. For congestion the number of ICNs and the number of data packets are held, which allows the probability of a congestion event occuring to be calculated and for readings to be aggregated. Round Trip Times are held in two forms; a single RTT reading and a Smoothed RTT. The SRTT has the additional parameter of the Variance in RTT. Both the SRTT and VRTT are produced using the TCP algorithms given in [JB90]. It is important to know the rate of transfer, because it would be wrong to infer from say a Kilobit/sec stream, which suffered little congestion that the network path would support a high transfer rate. The data stream rate is expressed as both packets and data bytes per second, the average size of packet can therefore be obtained from it.

A stream may be associated with more than one location as a particular location may be a subset of another location. As multiple streams may be associated with each location there is a many-to-many relationship between locations and streams. Thus a table is required to maintain these

[1] A second IP address allows for differentiation between local hosts and for holding data on conditions between two remote locations.

associations. A long lived data flows representation will be partitioned into multiple stream objects in the database each corresponding to a specific time segment. Each stream has zero or more of each reading type linked to it and provides the temporal association between them. Consequently, for a given congestion reading it is possible to determine the rate of the stream that it was drawn from.

Issues relating to the volume of data stored may need to addresses; it may be necessary to age information out of the database whilst providing some higher level description of the aged data. This is left to future consideration.

7.4.4 Summary

This section has considered the extension of a Location Information server to make use of Sender and Receiver reports from RTCP packets. It has been shown that useful measurements can be extracted for estimating loss, jitter and rate. To do so, state needs to be maintained for individual receivers, within an LIS RTP session layer. A Data Traffic Repository has been described, which takes advantage of the relaxed timeliness constraint on the delivery of network information to RTP applications to provide applications with more flexibility in specifying the required data.

7.5 An example application: Synchronous Groupware

7.5.1 Introduction

In this section the design of an application, which utilises RTP and a Location Information Server, is described as an example of the usefulness of sharing congestion information. The application is an extension of TAGS (Tutors and Group Support) [ARML01], which is a framework for building Distributed Learning Environments (DLEs) and is used on degree programs in six Scottish Higher Education institutions.

The extensions utlises the Java Media Framework (JMF)[2] [BRA01, ABR01] to realise real-time synchronous web based conferencing. The design also achieves the automatic configuration of multimedia resource instances based on past traffic measurements and thus ensure fairness with competing traffic and a predictable level of service.

The discussion is structured into four sections. First issues that pertain to the allocation of resources in the TAGS environment and multicast groups are discussed. Next some of the central QoS issues relating to this application are identified. This is followed by the presentation of a conference control architecture which addresses these issues and a discussion of the adaptation strategy that it supports.

7.5.2 Groupware Resource Allocation

The user model for TAGS is based around the three simple components of users, groups and resources (see figure 7.5) [ARMM01]. Sets of users are brought together with the resources that support their collective work by the TAGS group mechanism.

[2]The Java Media Framework (JMF) is a collection of APIs which aim to provide a method for handling time based multimedia within Java. It does this by allowing the capture, transmission, storage and displaying of various formats of audio and video. There are two main distributions for the JMF, a pure Java implementation and a native library version. The pure Java implementation is quite limited in power since it is unable to capture or transmit video/audio. It can however, render audio/video streams that it receives and is intended for use in situations where the native version of JMF has not been installed but the user still wishes to receive the audio/video streams. In the native library version of the JMF the majority of the processing is performed in platform specific code. This has significant performance advantages over a pure Java implementation and offers greater capabilities for the programmer and user.

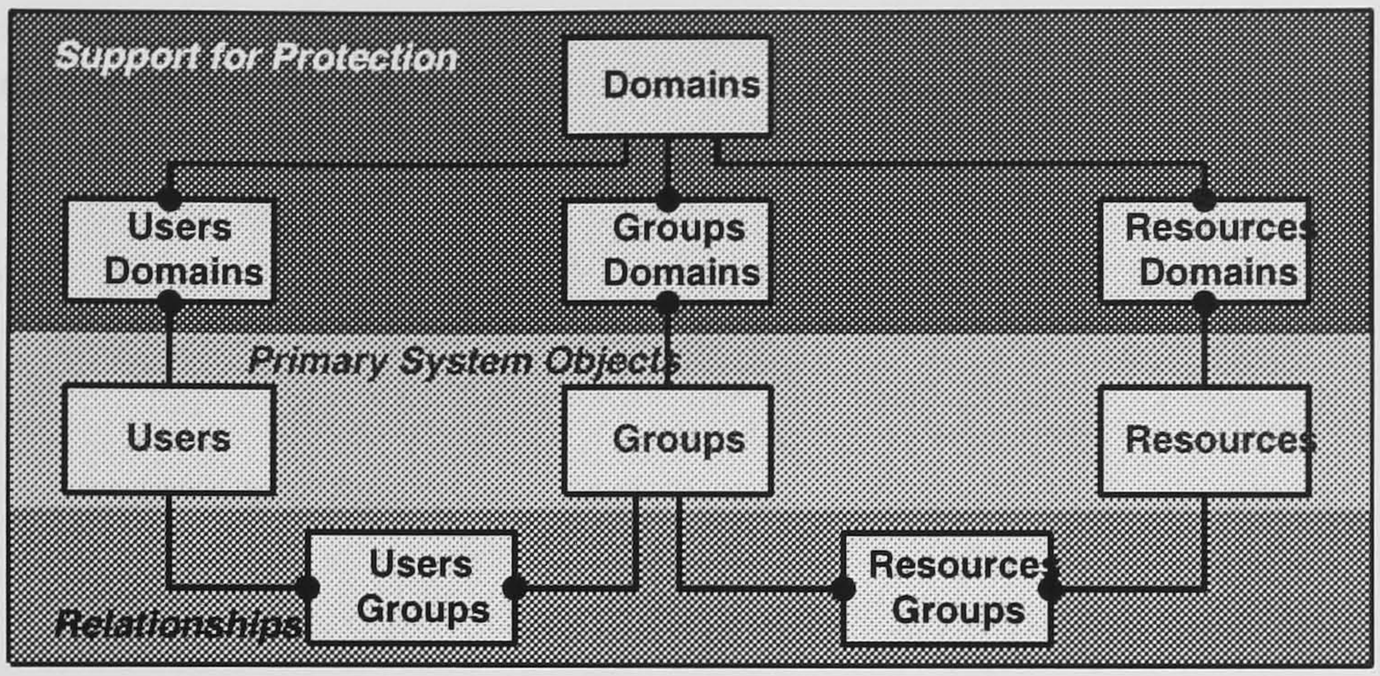


Figure 7.5: **TAGS Database Schema: Users access to resources is controlled by the group mechanism.**

Users and groups are unique by name; resources are unique by name and type. Groups form the basis of privileges and access, they provide the building blocks for the user's home page and they can also act as dissemination channels when a resource is involved. There is no restriction on the nature of a resource. It can be a simple timetable, an interactive multi-user spreadsheet or a live data feed. All a users allocated resources appear on their home page, which is generated from their group memberships. Access rights can be specified when a resource is allocated to a group.

When a resource is edited the changes are passed on to all members of the group(s) it has been allocated to. If it is deleted, it is removed from all the groups it was associated with. When a group is deleted, the mappings it has formed between individuals and resources are removed. If a group's membership is changed then only the updated membership will have access to the resources allocated to that group. There are three main levels of privilege in TAGS appropriate for students, tutors and lecturers respectively. Membership of a privilege groups defines the privilege level a user has, which in turn defines the view that a user receives. For example, in the Register resource a student can only view her own marks. A tutor is able to view and edit the marks for all members of their group, and a lecturer is allowed to view and edit all marks.

Figure 7.6 shows the main interactions in the system during the lifetime of a user session. When the user logs-on their username and password are authenticated. The portal generator retrieves information about the user from the TAGS database and generates a personalized portal to the system. The user may then request access to one of their resources. The status of the user is checked and control passed to resource specific code, which may in turn access the TAGS database through a safe API and retrieve resource specific data through its own API. The view of the resource that is created depends upon the status of the user. There are also special system resources, which have direct access to the TAGS library. These provide a number of utilities for System Administrators, which among other functions facilitate the allocation of users and resources to groups.

This allocation model can be used to manage video conferences, as shared resources and thereby benefit from the group allocation and domain protection facilities[3]. Multicast RTP supports

[3]Each instance of a TAGS server may service multiple courses. This reduces the cost to each course but brings with it the need for protection so that administrative rights granted to a user on one course are not global throughout the system. This is particular important for activities such as user creation, and the allocation of users and resources to groups.

From the point of view of a course administrator the requirement for protection is two-fold. On the one hand they do not want to have to deal with users, groups and resources that belong to other courses. On the other they require that tutors and system administrators who are not associated with their course should not be able to gain access to their work or educational materials. The TAGS architecture resolves this by constraining the rights of an administrator within the boundaries of an Administrative Domain of Protection (see Figure 7.5).

sessions and participants, where a session is a group of users. Sessions are usually short-lived entities and anyone is free to enter or leave at anytime whereas a TAGS group is a long term entity, typically lasting around 10 weeks.

In order to inter-operate with the TAGS framework, there must be functionality to allocate the sessions to groups from within TAGS. This would have the advantage that it would mean that multimedia sessions are easy to join, since the complexity of multicast addresses and port numbers is abstracted away from the user. The traditional method of joining multimedia conferencing session on the Mbone is via the Session Description Protocol (SDP) [HJ98], implemented by the SDR [Jac] tool. SDR displays a list of sessions which are either scheduled to take place or which are currently underway. The entries consist of a multicast IP address, a port number, the multicast scope in the form of a TTL, the RTP payload type and some textual information about the session. From this information any session advertised by SDR may be joined. The mechanism proposed replaces the need to use SDR when joining a multicast session from within the TAGS framework, but it does not eliminate the need for the SDP. As stated, many sessions are already advertised within SDR and this information can be used when allocating session addresses so that the probability of a clash is reduced.

7.5.3 Quality of Service Issues

Whilst the integration of TAGS allocation with conference control mechanisms helps make these technologies accessible to non-expert users, for widespread deployment to become a possibility it is necessary to address QoS issues [ALMR01, RAL01, ALMR00], both to ensure that users receive the best possible QoS and to protect the network from congestion. There are three characteristics of relevance to QoS; the envisaged synchronous communication will be between small groups of people, it may take place across heterogeneous networks and will utilise a selection of mediums. An example use might be a tutorial session including a tutor and half a dozen students. The small numbers involved in the conference means that centralised control is a realistic proposition.

The networks used to facilitate communication may have different characteristics for different groups. This can be illustrated by briefly considering a number of scenarios. Postgraduate students may engage in a small symposium, whilst being connected directly to 100 Mb/s Ethernet switches. An undergraduate tutorial might involve a shared connection to a 10 MBit/s hub for many of

---

When the possibility of a user being associated with more than one course is considered, there are further requirements: the need for a single user to have access to multiple domains, and that a single log in and password should allow access to all of the domains that a user has the right to access. The protection that Domains provide should not prevent members of separate administrative domains from engaging in collaborative learning. It follows that a mechanism to allow inter-domain communication is needed.

The problem of protection is one that has been explored in the context of operating system design. Silberschatz and Galvin [SG94] make the following observations: 1) A computer system can be thought of as a collection of processes and objects. The operations that are possible may depend upon the object. 2) A process should be allowed to access only those resources it has been authorised to access. Furthermore, at any time it should be able to access only those resources that it requires to complete its task. This "need-to-know" principle limits the damage that can be done by a faulty process.

These observations provided the starting point for the design of TAGS domains. In the TAGS system a user plays a role analogous to that of a process in the OS and Users, Groups and Resources that of operating system objects.

A domain is an area of protection, which maps to an administrative area of control. A user, resource or group may belong to one or to many domains of protection. Normally a user will only be active in one domain at a time. If they wish to manage objects in another domain to which they have management privileges, they must explicitly change the domain in which they are active. This helps support a weak version of the need to know principle. The user only has access to those objects they need for the tasks they are undertaking at a particular point in time.

As discussed above, group membership is used to structure collaborative learning. TAGS treats group membership as a tighter binding than domain membership. Consequently, assigning members of different domains to the same group will facilitate inter-domain communication. The group has to belong to multiple domains but the users may be in separate domains. Groups may also be used as a mechanism for allowing resources in one domain to be made available to users in other domains.

In this way domains provide the administrative protection that is required without unnecessarily constraining the freedom of communication that collaborative group work requires.

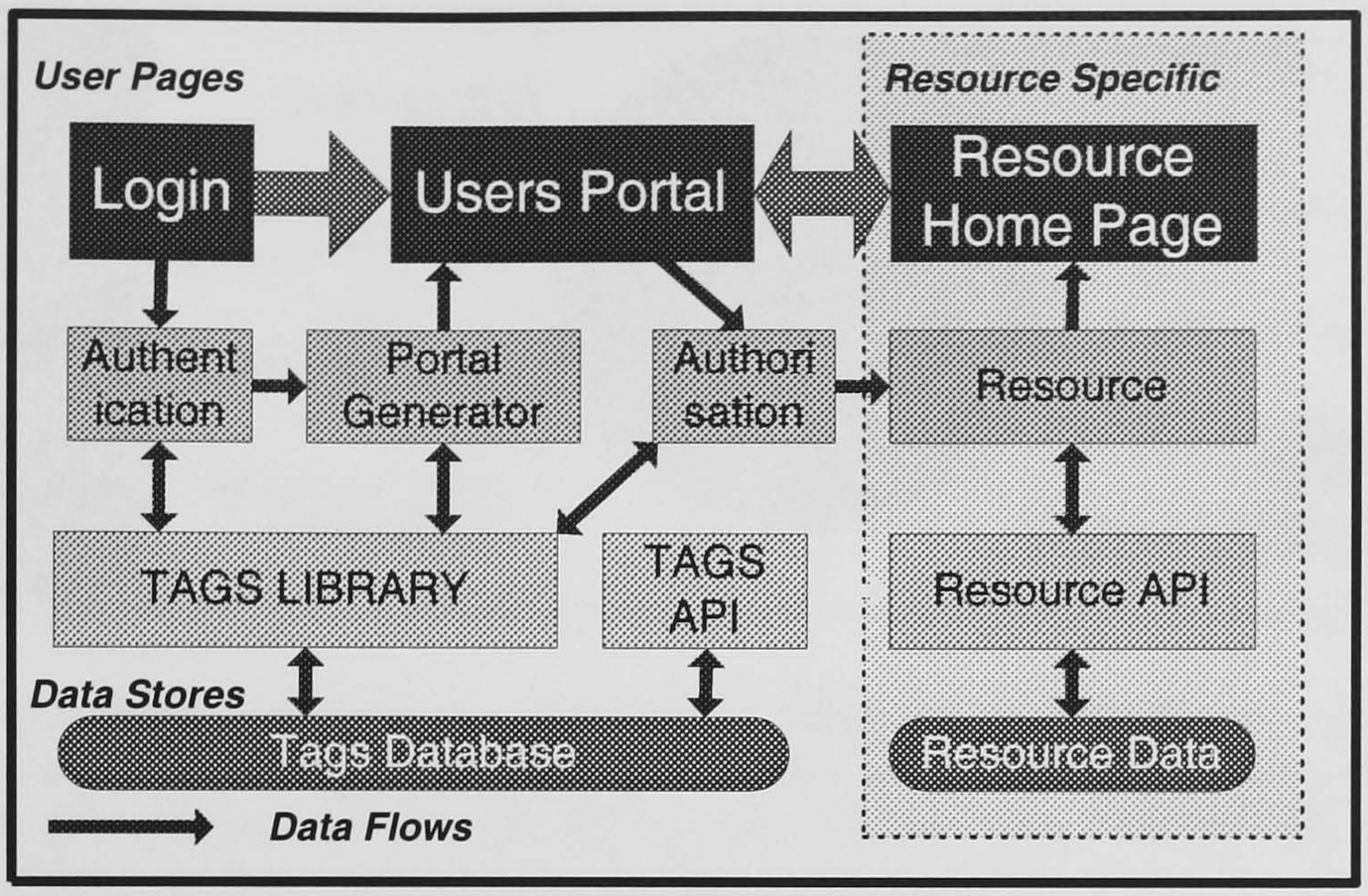


Figure 7.6: **TAGS System Interactions: When logging in the user is authenticated. Once a users ID is established access to resources can be determined. APIs are provided to the underlying databases to make ease integration of new resources into the system.**

the participants, as well as connections from student residences. The need may arise for cross-institutional conferences during which data must traverse each institutions connection to a wide area network, at which point competing traffic may severely constrain bandwidth availability and increase jitter. Another scenario may involve some participants logging on to the conference from a home computer. Here the technology used in the "last mile" will constrain bandwidth. ADSL allows up to 256 Kb/s upload; ISDN-2 128 Kb/s and a V90 modem 56 Kb/s.

Supported communication channels might include video, audio, and shared objects. A standard video connection using the H.263 video coded at 176x144 and 24 f/s requires 120Kb per second where as an audio connection using LPC encoding at 8 KHz mono requires 5.6Kb/s. per second. Putting these together means that for a reasonable audio/video conferencing requires a total of 126Kb/s per sender in a conference. So using our assumption that a tutorial group consists of 6 students and one tutor then the overall bandwidth required is 879kb/s.

In the scenarios where bandwidth is plentiful, the system should be able to make full use of the available resources, thereby ensuring a high Quality of Service for the participants. On the other hand, when bandwidth is seriously constrained, a workable solution is required that does not waste the constrained resource or act unfairly [Jai84] in relation to competing traffic.

7.5.4 A Conference Control Architecture

Significant attention has been paid to the development of conference control architectures, Light Weight Sessions are the dominant model on the Internet. The existence of a group allocation model within TAGS, the absence of a requirement for scalability and the desire to control participation leads to enhancement of the LWS model. Conferences take place at a particular time, much like tutorials. This means that an explicit conference set up phase, which is absent from the LWS model is possible. This phase is utilised to determine likely network conditions for the duration of the conference and to configure the conference appropriately.

There are three main components of the system. A Traffic Data Repository (TDR), which makes available to conferences traffic statistics and receives measurements of network statistics from a

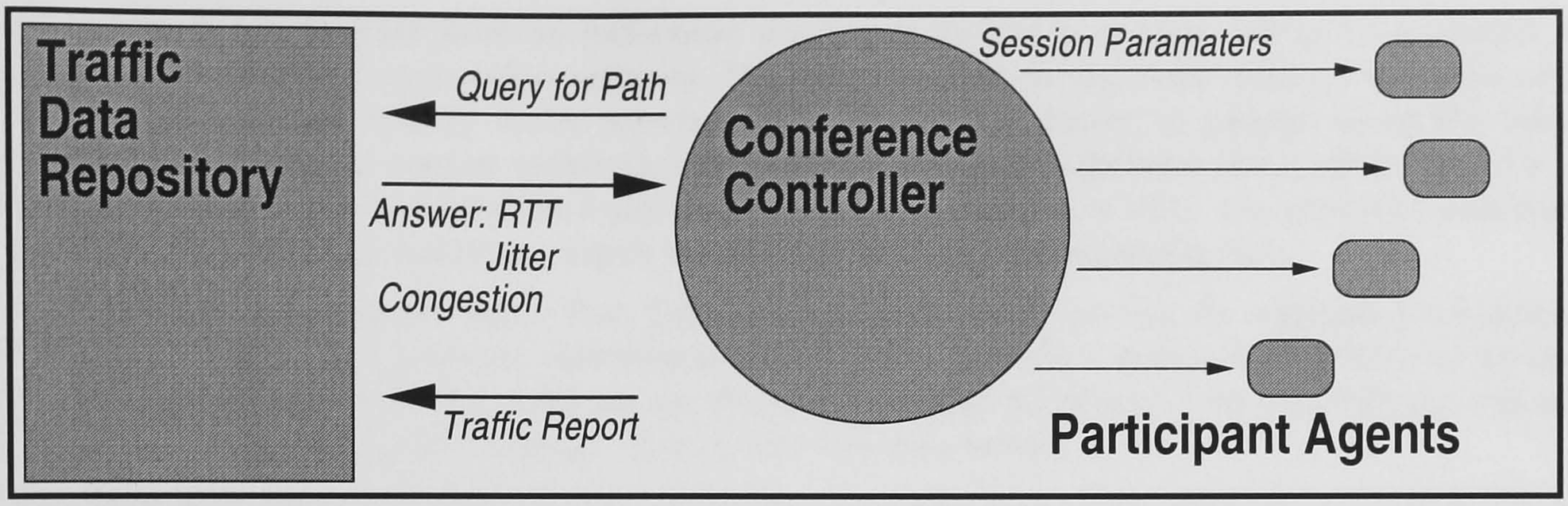


Figure 7.7: **Conference Control Architecture: When a conference is set up the conference controller retrieves information on network conditions from the TDR and provides each participant with a set of Session Parameters, which will correspond to a QoS appropriate to the predicted available resource.**

variety of sources such as TCP or an RTP application. A Conference Controller (CC), of which there is one instance per conference. The CC provides centralised coordination for the set up and running of the conference and interfaces with a TDR. Participant Agents (PAs), of which there is one for each participant in the conference. Each PA informs the CC of available local resources and configures those resources on the instructions of the CC.

The Conference Controller and Participant Agent combine the information provided by the individual RTCP sessions to build an accurate picture of the state of network the during a conference. This is reported back to the TDR for use by future sessions and is used by the Conference Controller to determine whether reconfiguration of the conference is necessary.

When a Conference Controller wishes to start a conference it queries the TDR for a prediction of traffic characteristics for each pair of IP addresses that are to participate in the conference. The reply contains an estimate of the level of congestion, the maximum sustained rate of transfer for the streams that generated the congestion readings, a likely worst case estimate of congestion, a RTT and Variation in RTT estimate and a jitter estimate. These estimates are calculated by the TDR using the best available information to it. Factors that are taken into account when determining the best available information are; time that the data was generated, volume of packets from which measurements were made and match between IP addresses of current query and IP addresses held in the repository.

7.5.5 Adaptation Strategy

There is a tension between maximizing individual and global value derived from a computer network. On the one hand it is desirable to maximize the value that individual users receive from a video conference. On the other it is sociable to control the cost to the network and therefore to other users of the network.

The value users receive from the network can be quantified using objective or subjective criteria. Bouch and Sasse [BS99, BBK00, BS00] advocate using the subjective criteria of perceived quality. They demonstrate that there is not a direct mapping from the technical characteristics of a data stream to the QoS perceived by users. In particular they demonstrate that a users expectations are important in determining their perceived quality of service. The range of timescales with which such expectations are updated is however in the order of tens of seconds, which leads to the conclusion that consistency of technical quality is of crucial importance in determining perceived quality.

From this it follows that from an individual user's perspective it is often better to constrain or limit the adaptation to network conditions. If during the lifetime of a video session, the achievable frame rate or sound quality varies between two bounds, it is better to present using the lower bound throughout the session rather than trying to dynamically optimise the quality. To achieve such consistency it is necessary to know at the start of a conference what the network conditions are likely to be for the duration. There are at least two ways of achieving this.

Previously it has been invisaged that these requirements could be met by combining admission control [JDSZ95] with resource reservation [BZB+97] to provide a guaranteed quality of service for the duration of a call. We make the assumption that such facilities are unavailable and explore what can be achieved in the absence of such resource reservation.

The approach advocated here involves collecting measurements of the conditions experienced by past sessions in a common repository and using them to make predictions at the start of a conference about the conditions that are likely prevail during its lifetime. These predictions can then be mapped to fair resource utilisation estimates that account for the cost to the network. These resource estimates can in turn be used to determine how the conference should be configured to maximise the utility participants receive from it. In particular relevant parameters need to be initialised, so that a Quality of Service appropriate to the available network resources can be achieved at the start of, and maintained for the duration of, the conference.

Why is it reasonable to expect such an approach to work? There should be strong locality of reference; conferences for a particular group are likely to consist of the same people, from the same machine over similar network paths. There is a strong diurnal pattern to network usage; therefore measurements made at a similar time of day are likely to be relevant in the future. The physical infrastructure of a path will limit the bandwidth available. The time scale over which the infrastructure is likely to change is significantly larger than that which separates conferences.

With a given set of predictions about network conditions for the set of pairs of locations that will participate in the conference it is possible to determine the resources that should be available for the conference to utilise. The majority of Internet traffic uses TCP as its transport protocol, consequently the steady state behaviour of TCP has been advocated as a model, which non TCP traffic should use when determining the bandwidth it should utilise for a given RTT and level of packet loss. Such behaviour is described by the equation, which is derived in [MF97]:

$$T = \frac{C * MSS}{RTT * \sqrt{p}} \tag{7.1}$$

Where T is the throughput MSS is the maximum segment size, RTT is the round trip time, p is the probability of a packet being lost and C is a constant derived analytically to be close to 1.

It is, however, arguable that a video conference could fairly receive a higher proportion of bandwidth than is implied by this equation. Firstly, the technologies involved may require more bandwidth to be effective. Secondly, the equation describes fair bandwidth for a single connection but a video conference will involve multiple sessions and users. Thirdly, the users may receive more utility from the conference than a simple web session. It can therefore be argued that bandwidth utilisation should be adaptive to network conditions in some way that is proportional to the TCP fair equation. Under a high level of congestion the conference will be configured to use less bandwidth than under a light congestion regime. Ultimately the ratio of TCP fair bandwidth to conference utilisation depends upon some subjective judgment about the relative utility of the conference.

Within a regime where charging is associated with the level of congestion [KMBL99], the charge can mediate between the conflicting local and global optimums. The conference organisers are constrained from simply trying for maximum resource by the charge that is likely to be associated with it. In the current regime they are constrained by their conscience.

The long lived TAGS groups that conferences are associated with here provides a convenient

mechanism for budget holders to express the utility that they associate with a conference by defining a mapping from TCP fair bandwidth to conference utilisation. This might take the form of a simple ratio; a particular group conference may receive say three times the TCP fair bandwidth. The consequence of this choice will be a level of invoice against the budget and a level of quality experienced by the users.

The likely upper and lower bounds for the bandwidth available to the conference can be obtained by the same mapping procedure from high and low bounds on the predicted levels of loss. Once the bandwidth available to the conference has been obtained it needs to be partitioned between the different mediums in a way that will maximise users utility.

7.5.6 Summary

This section has discussed the design of a synchronous conferencing application that makes use of information provided by a TDR intelligently how to set up a conference. It has demonstrated that it is possible to design applications that make effective use of the QoS information provided by a Location Information Server and Traffic Data Repository.

7.6 Comments on Multicast Congestion Control

There are a number of application types that may make use of multi-cast routing. The specific needs of and traffic dynamics associated with each application type may require different approaches to congestion control. For this reason a concrete analysis of different application types, with respect to their needs and the challenges they present is undertaken next. In this section a number of topics are discussed. Firstly, congestion control for real time applications of the type RTP was designed to service are discussed. This is followed by a discussion of interactive distributed applications. Finally, reliable multicast data transport with weak timeliness constraints are discussed.

7.6.1 Realtime Multicast Congestion Control

One requirement for multicast streaming of video and audio is that it should be able to scale to support large numbers of recipients from a single source. Receiver-based congestion control helps facilitate this. There are a number of elements which are common to schemes addressing this problem. The ability to join and leave multicast groups provides a router based mechanism to exercise on-off congestion control. By organising the transmission of data into layers, allocating each layer to a different multicast group and bandwidth, receivers can adjust their bandwidth utilisation to the prevailing network conditions by joining and leaving multicast groups.

The RLC multicast congestion algorithm [RVC00], TCP-like congestion control [VRC98] and receiver driven layered multicast [MJV96] adopt a similar approach.

RLC is designed to provide TCP Fair congestion control for Multi-cast traffic on the Mbone and RLC works by having a small number of rates, which a receiver may jump between, depending on network feedback. As receivers sharing a bottleneck need to synchronize, instability is avoided by limiting the frequency with which levels can be changed. Therefore it may take several seconds to reach equilibrium if a receiver joins at the wrong level. The small number of levels means that a fine grained mapping from measured congestion to rate is not required.

Receiver-based estimation of congestion and subscription to an appropriate number of layers means that congestion information need only be transported from the point of congestion to the receiver. It is not relevant to the sender. It may still however be useful for the receiver to have an estimate of the appropriate level at which to start subscription and thereby involve a long period of probing an LIS could provide this information.

7.6.2 Interactive Distributed Applications

In [MHKE99] an RTP profile is proposed that would provide services appropriate for interactive application. As has been discussed RTP is intended for use by real time media, with soft real time timing constraints that are tolerant of small amounts of loss. It soon became apparent that there are a class of applications for which multicast services are useful but that require reliable delivery. Perkins and Crowcroft draw a distinctions between the requirements of real time media such as voice and video and real time interactive media [CP00]. A real-time interactive distributed application is defined as one where the state of the system changes primarily in response to user interaction. Examples include a shared white-board or text editor, a distributed presentation tool or a networked multi-player game. They contend that the characteristics of interactive media make RTP or the extension of RTP, as described in [MHKE99] an inappropriate design choice as a transport protocol. Rather they see RTP as a starting point for the design of an appropriate protocol, the Real Time Framing Protocol (RMFP) [CVW+98] and Reliable Multicast Framework (RMF) [DBF+97] are cited as examples. Consequently it is argued that many applications would be best designed using multiple protocols for example realtime streams by RTP, interactive media by something akin to RMFP and bulk data by reliable multicast.

The three main reasons why the extension of RTP for use as a protocol for interactive multimedia applications relate to reliability and timeliness of transmission. i) Interactive media may require reliability, this implies that if a packet is dropped it needs to be retransmitted, which in turn means that a mechanisms for detecting loss and informing the sender are required. Whilst the RTP sequence numbers allows packet loss to be detected no mechanism exists for informing the sender. In addition the constraints on the sending of RTCP packets means that even if RTCP was extended timely signalling would not be possible without undermining the scalability of RTCP. ii) The retransmission of dropped packets implies the need to supplement RTP's sequence numbers, which increment for each packet sent and take no account of retransmissions. The jitter calculation assumes that packets are played out evenly spaced, this is unlikely to be true for applications whose output is generated by interactions from the user. iii) The chief advantage of an RTP profile for interactive applications from the point of view of congestion sharing is that it would extend the scope of RTCP to a whole new application class. the same benefit could be achieved by designing protocols that were appropriate to interactive applications but including in them an RTCP module that provided standardised QoS feedback.

7.6.3 Reliable Multicast

There has been considerable interest in the requirements of a class of applications that could make use of reliable multicast protocols. These may be built by adding reliability on top of RTP or by designing alternative to RTP.

Reliable multicast may be used for the distribution of software, the update of Internet mirrors or any case where there is the need to distribute data from a single source to multiple destinations[4]. Typically it is assumed that there are no time constraints on the distribution of data, that a large volume of data needs to be transferred and that reliability is important [HF].

The provision of reliable multicast services carries with it the danger of creating congestion for three reasons. The congestion control mechanisms embedded in TCP cannot be used, a single source may produce multiple copies of the same data, and the transfer of data is likely to be automated (for example the automatic update of software). For these reasons congestion control for reliable multicast data should be conservative. A commonly advocated approach is to synchronise all receivers to the same rate or window size and for this to be determined by the fair rate of the slowest receiver. The fair rate can be defined as the average rate that a bulk transfer using TCP would have received. TCP fair equations can be used to determine what the fair rate is.

[4] In many cases the distribution from multiple sources to multiple destinations may be treated as the multiple instances of distribution from a single source to multiple destinations

A reliable multicast protocol, which uses Forward Error Correction, suitable for wireless environments is presented in [RV98]. Criteria for evaluating reliable multicast transport and application protocols are outlined in [MRBP98]. Issues of congestion, fairness and freshness are discussed in [GC99]. Examples of congestion control schemes for reliable multicast include; SBMCC [NRK00], which is a generic source based congestion control algorithm for reliable multicast and PGMCC [Riz00], which is a single rate multicast congestion control scheme for reliable multicast. Scalability of the congestion control scheme to large groups is important and is achieved in PGMCC by the sender only receiving acknowledgments from one host.

The key problem addressed by reliable multicast is how to recover from loss in an efficient way given there may be a low level of loss correlation between recipients. The conservative requirements for reliable multicast congestion control and the likely non interactive nature of transfers means that whilst it is desirable to start transmission at an appropriate rate for network conditions static intialisation and probing is an acceptable solution. Reliable multicast may however be an important source of data for an LIS. QoS feedback provided by an LIS may be used to set the initial rate for a reliable multicast transmission. This is particularly the case where the set of recipients is likely to be stable over a large number of transmissions.

7.6.4 Summary

Despite the different concrete problems that confront distinct application classes we find that for each class there is a strong case for sharing QoS feedback. RTCP is designed to be scalable to large numbers of participants and provides feedback that would be valuable across application types. The granularity of feedback necessary to fulfill the scalability requirement limits the applicability of its feedback during the lifetime of a session, but is appropriate as the basis for longer term predictions. This leads us to the conclusion that it could provide the starting point for the design of a generic feedback mechanism which when combined with an LIS would provide applications with sufficient information to determine an efficient and fair starting point for initilial rate or window size, buffer size and/or subscription level.

7.7 Conclusion

This chapter has discussed the issues that arise when the LIS is extended to support the extraction of congestion information from RTP traffic and the provision of congestion information to applications that utilise RTP.

It has been argued that the passive monitoring of RTP traffic by an LIS would have a number of benefits. It would widen the base of traffic from which congestion information was derived thereby increasing the proportion of paths for which such information could be make available and increasing the the accuracy of some predictions. Secondly it would facilitate the sharing of information between RTP sessions. Thirdly it would facilitate the sharing of QoS feedback between applications using TCP and RTP transport protocols.

RTP does not provide a reliable service. Consequently, there is no need for acknowledgments of received packets or retransmissions of lost packets. The mechanisms used by the LIS to determine rate of loss RTTs and window size are therefore not available. The direct passive monitoring of RTP data streams cannot therefore be easily achieved.

RTCP packets however, provide feedback on the state of the network to participants in a session. The format of these packets is tightly specified, so it is possible for an LIS to use passive monitoring techniques to identify RTCP packets and extract the information they contain. There remains the question of how to integrate the information gathered by monitoring RTCP and TCP packets.

The underlying problem stems from a tension between the LIS architecture and the necessity of relying upon RTCP to calculate traffic statistics. With TCP, end-stations did not report measure-

ments to the LIS, the LIS took the measurements directly. This has the advantage of centralizing the processing of those measurements and increasing the accuracy with which congestion and RTTs could be estimated. This approach also removed the possibility of different methods of calculation employed by TCP implementations resulting in varying measurements for the same network conditions. This benefit is unobtainable for RTCP traffic and consequently a number of issues arise about how RTCP calculates traffic statistics.

For example RTCP's estimation of loss does not account for the possibility of duplicate packets so a negative loss rate is possible. Also bursts of packets may be lost in a single congestion event. This is not accounted for in RTCP's reporting of lost packets. Despite these difficulties it has been argued that the levels of loss reported by RTCP provides valuable information and that that information can be successfully integrated into a common repository with TCP derived information. To achieve the above, the interface to the location layer needs to be extended and an LIS RTP session layer added.

It has also been argued that the timescale over which an RTP session is likely to exist is significantly longer than many TCP connections. Consequently there is a looser timeliness constraint on delivering QoS feedback. There is however also a requirement to maintain a consistent Quality of Service during the lifetime of a session. In part because this consistency improves the users perception of quality and in part because some of the popular conferencing tools are not adaptive [MJ95]. Consequently if the QoS information is to be used to configure the session, it is desirable that it predicts the likely conditions for the duration of the session. To do this a historical record of congestion on the route is required rather than an instantaneous reading of current conditions.

The relaxation in timeliness constraints and the requirement for a long term prediction respectively facilitate and motivate the design of a Traffic Data repository, which holds historical data on network conditions and allows the application to specify the information that it requires. The design of such a repository and its relationship to the LIS have been described. The design of a realtime synchronous groupware application has been presented, which utilises a TDR to configure conferences during a distinct set up stage, has been presented. This illustrates the benefit to RTP applications of the architecture for the sharing of QoS information that has been presented in this chapter.

The needs of three classes of multicast applications have been discussed in relation to congestion control. It has been argued that while each confronts different problems during the lifetime of a session, each would benefit from the sharing of QoS feedback to facilitate intelligent initialisation. This suggests that an LIS could have generic applicability, particularly if combined with a generic mechanism for providing QoS feedback. It has been argued that RTCP would provide a strong starting point for the design of such a protocol.

Chapter 8

Related Work

8.1 Introduction

This chapter surveys a number of bodies of work that are relevant to the subject of this dissertation. The aim is to clarify the relationship between the work reported within this dissertation, the work whcih immediately predated it and that which ran concurrently with it.

TCP's Congestion Avoidance algorithm regulates window size during the lifetime of a connection in order to avert congestion collapse. An important strand of research seeked to improve upon TCP's approach by continuing to adjust transmission in response to an implicit congestion signal generated by the network in response to that connection's traffic. The aim of this strand however was to allow the network to operate around an optimal load rather than simply averting congestion collapse.

The second body of work discussed, developes mechanisms within the network to improve upon TCP's and potentially other end points reaction to congestion. These efforts fall into three categories, those that modify the criteria for dropping a packet and therefore the content of the congestion signal, those that address the way that congestion is signalled and those that aim to introduce inforcement mechanisms. This work is largely orthoganal to this dissertation.

The next two sections relate to work which directly addresses TCP's start-up behaviour. The first discusses work which attempts to prevent TCP from overshooting its fair share of bandwidth or facilitates reaching its fair share more promptly. The second deals with reasearch that addresses weaknesses in using TCP's ACK clock to implicitly shape the distribution of packets within a window of data. This is relevant to the subject of this disseration as starting with a larger initial window has the danger of increasing the burstiness of TCP's traffic.

8.2 End-to-End Congestion Avoidance

TCP's congestion control mechanisms treat the network as a black box, no explicit signaling is required from the network. The load that a source should present to the network is deduced from information available outside the network. This is particular useful for IP networks which produce no reliable explicit signal. The specific way that load is determined, by the detection of packet loss, results in a congestion recovery mechanism, where the network power is not maximised, and loss is created by the congestion control mechanisms.

A number of congestion control schemes accepted the lack of a reliable explicit signal, but aimed to operate the network at optimum load or at least to improve upon TCP's loss based congestion detection. These schemes replace TCP's reactive control strategy with a pro-active approach

which detects the onset of congestion and adjusts the send rate before packet loss occurs.

As a route becomes congested the number of packets in queues increases, this results in each packet spending more time queuing and the RTT going up. An increase in the RTT is therefore an indication of the onset of congestion. An increase in the RTT will also result in a slow down in the rate at which ACKs are returning to the sender. This results in a flattening in the send rate which can also be used to detect the onset of congestion.

Four schemes which use black box detection are itemised and discussed below:

1. Congestion Avoidance Using Round Trip Delay (CARD) [Jai89]
2. DUAL [WC92]
3. Slow Start and Search (TRI-S) [WC91]
4. Vegas [BOP94, DLY95]

CARD and DUAL both use changes in delay to locate an optimal operating point, TRI-S and Vegas use changes in throughput.

8.2.1 CARD

Jain starts from the observation that as network load increases so to does delay, and as most transport protocols track RTTs to support retransmissions this information can also be utilised to adjust the load applied to the network.

A distinction is drawn between a selfish optimum operating point and the social operating point. An expression for the socially optimum window size is derived as a function of the delay-window curve, that holds for deterministic networks.

The decision to increase or decrease the window is based on changes to the gradient of the delay window curve. This leverages upon the observation that at the knee there is a sharp increase in the slope of this curve. Once every two RTT the window is adjusted.

If

$$0 > (WS_{curr} - WS_{old}) * (RTT_{curr} - RTT_{old}) \tag{8.1}$$

the window is decreased by 1/8 and if not is increased by one. The congestion window therefore oscillates around an optimum operating point.

This scheme is not presented as a fully worked out solution for controlling congestion in real networks, but as a demonstration of the fruitfullness of black box congestion avoidance strategies. This scheme is the point of reference for DUAL, Tri-S and Vegas, which followed it

8.2.2 DUAL

Zheng proposes [Wan92] a Delay Threshold, which estimates the minimum and maximum RTTs and ensures that the maximum is never reached in order to avoid packet loss. The maximum RTT is at the point when packets are lost due to overflowing buffers, the minimum RTT is when the router queues are empty. An approximation for the minimum RTT is achieved by measuring the RTT of the first packet. At this point the buffers will at least be empty of the new stream's packets.

If a minimum and maximum RTT are known, they can be used to determine a safe RTT region to be operating in. This is assumed to be around the average of the minimum and maximum RTTs. If the RTT is greater than this average the window is decreased and if it is smaller the window is increased.

This approach uses RTT measurement to detect the onset of congestion and take pro-active action before packet loss occurs. Its effectiveness is supported by simulation results.

8.2.3 TRI-S

The TRI-S scheme attempts to establish an optimal and fair operating point at the start of a connection and each time there are major traffic changes. It does this by deducing the load from acknowledgments using a metric called *Normalized Throughput Gradient (NTG)*

There are three modes of operation.

1. Initialisation mode: At start-up the window size is increased exponentially until the maximum window size is reached.
2. Decrease mode: when a packet is lost decrease mode is entered. The window size is set to one and increased exponentially if the NTG is over a threshold, if it is below the threshold increase mode is entered.
3. Increase mode: the window is increased linearly. The NTG is periodically checked and if it is less than a second threshold the window size is decreased by one segment.

The NTG is used to decide to decrease the congestion window before packet loss occurs.

The average current throughput is estimated as the Number of packets outstanding in the network over the current RTT. Where T is the throughput and and W the window size the throughput gradient is defined as:

$$TG(W_n) = \frac{T(W_n) - T(W_{n-1})}{W_n - W_{n-1}} \tag{8.2}$$

NTG is defined as the current throughput gradient over the throughput gradient for the first window.

$$NTG(W_n) = \frac{TG(W_n)}{TG(W_1)} \tag{8.3}$$

The NTG is assumed to vary between 1 and 0. If more resources become available through connections leaving the throughput gradient can jump considerably, allowing the window to increase in size.

The Tri-S scheme attempts to avoid packet loss and get connections to operate around the optimum operating point. It does so by detecting a reduction in the rate of increase in throughput as the congestion window opens. It is not proven that the actual point of operation is optimal, but simulations indicate an improvement in performance when compared to Reno TCP.

8.2.4 Vegas

TCP Vegas, like Tri-S, utilises changes in throughput to detect the onset of congestion. It differs in two important respects. Throughput is measured differently and a notion of expected throughput is introduced. The actual throughput is measured and compared against expected throughput, if the difference is below a minimum threshold the window is increased and if it is above a second threshold it is decreased. Both the increase and decrease quantities are linear.

The minimum RTT measured is assumed to correspond to the RTT when there is no congestion and is called the Base RTT. The Expected throughput is defined as Window Size / BaseRTT. The actual sending rate is calculated by:

1. recording the time at which a segment is sent

2. recording the amount of data sent before an ACK for the recorded segment is received

3. dividing the number of bytes sent by the sample RTT

The difference between the actual and expected throughput is then obtained. $Diff = Expected - Actual$ If Diff is below the minimum threshold the window is increased and if it is above the maximum it is decreased. Simulation, live emulation and some experiments on the Internet indicate improvements in throughput and reductions in loss. Simulations also show that the performance of Reno TCP does not degrade when it is competing against Vegas implementations.

8.2.5 Summary

A number of schemes aimed to improve on TCP's loss based congestion control mechanisms, whilst maintaining a black box approach to congestion detection. Changes in the RTT and throughput can be used as implicit signals of changing network state.

When the bandwidth delay product is high a large amount of data may be in the pipe in this case waiting for a timeout and redoing Slow Start is particularly painful. The introduction of fast retransmit and recovery, the proposed Fast Recovery Phase, SACK and FACK TCPs are all aimed at avoiding this pain be recovering from packet loss quickly. It would be better to minimise the incidence of dropped packets.

Jain utilises changes in RTT to derive a socially optimum RTT size for the congestion window in deterministic networks. The Tri-S, CARD and Vegas schemes aim to approximate to this optimum in non deterministic networks, in each case a base measurement which corresponds to an uncongested network is used as a comparison with the current state. Tri-S assumes that the throughput gradient for the first packet corresponds to a uncongested network. DUAL assumes that the RTT of the first segment will correspond to the minimum RTT for the path. Vegas assumes that the minimum RTT for a connection will correspond to the uncongested state of a the Path.

It is however questionable whether any of these assumptions hold. In particular if the route is busy with other traffic at start-up or in the Vegas case for the duration of the connection values that reflect the uncongested state of the network may not be obtained. The central weakness with the above approaches is that they do not address the limitations imposed by confining adaptation to within the lifetime of a connection.

Using RTTs or a flattening in the rate of send to detect the onset of congestion attempts to avoid multiple packet loss. The reduction in the opportunity a connection has to gather implicit feedback caused by the reduction in the number of RTT in a connections life however limits the utility of both methods.

8.3 Router Support for Congestion Avoidance

In 1988 the role of routers in controlling congestion on the Internet was confined to dropping arriving packets when buffers were full,. The service discipline was first come first served (FCFS). One way of changing the behaviour of the TCP congestion control algorithms without modifications in the host was to change the events which generates a congestion signal. The proposal for Random Early Detection Gateways aims to control delay by randomly dropping packets if queues grow beyond some threshold. This aims to hold network load further to the left of the congestion collapse cliff.

Once intelligence is deployed in gateways to monitor queue lengths it is a small step from using loss to signal congestion to using Explicit Congestion Notifications (ECNs). Consequently several proposals to add ECN to the Internet have been made. It is possible for applications to bypass

Internet congestion control by using UDP as a transport protocol. The growth in non TCP Internet traffic has lead to concerns about the lack of fairness between Internet streams. This in turn has resulted in proposals to add enforcement mechanisms to routers, so that streams which do not respond to congestion notifications in a similar way to TCP[1] can be penalised.

To create TCP friendly congestion control mechanisms in non TCP streams and to be able to identify non TCP friendly streams, it is useful to have a model of how TCP should behave. Several such models varying in the level of abstraction have been developed.

The proposals for RED gateways, Explicit Congestion Notification and penalising traffic which is not TCP friendly, amount to an extension of congestion control functionality from the end host to the network. Alternative proposals have been made which address the same issues, but aim to allow greater freedom in the development of end node congestion control mechanisms.

A router has a number of advantages over an end node that can be utilised in traffic management. First it has a global view of all the traffic that is passing through. Therefore there is no need to infer what the traffic characteristics may be at a bottleneck they can be directly measured. Secondly the traffic can be directly controlled. A third advantage is that it is easier to change routers than it is host implementations, because there are less of them and they are under the control of a smaller number of entities.

8.3.1 Random Early Detection Gateways

The most common type of gateways on the Internet are Drop Tail Gateways. These work by dropping an arriving packet if the queue is full and do not actively aid congestion control. The central idea behind RED gateways is to generate congestion signals, requiring hosts to slow down before resources are saturated. This is not a knew idea, in fact Baran's papers advocated controlling input to the network when link utilisation rose above a threshold. Routers in the CYCLADES network signaled sources to slow down before congestion occurred. Nagle's Source Quench and Postel's SQUID both sent congestion notification messages if buffer queue lengths became to long. It was however Jain's Binary Feedback Congestion Avoidance (BFCA) scheme implemented for connection-less networks that represented the state of the art prior to RED Gateways.

The RED proposal is shaped by the objective of introducing mechanisms in routers that will help in controlling congestion, without requiring modification to the already established congestion control mechanisms in TCP/IP. This consideration required a number of design decisions to be made which distinguish RED from BFCA.

The mechanism TCP uses for detecting congestion is packet loss, RED routers drop packets instead of marking them[2]. An identified weakness in the TCP was traffic phase effects, where connections cut back and increase their windows in a synchronised manner. This resulted in periods of under-utilisation and periods of congestion. It is caused by the wait for a timeout and reduction in window size to 1 segment in Tahoe TCP. Whilst Reno may detect a packet loss without a timeout and cuts its congestion window by half the problem remains in a less severe form. Motivated by the need to avoid these phase effects, RED does not try to generate the same signal to all competing connections, it does not therefore improve upon TCP's statistical fairness.

The queue length is continually monitored and if it exceeds a maximum value all the new packets arriving are marked. If the queue length is between a minimum and maximum value then a proportion of the packets are marked to signal congestion. Packets to be marked are chosen randomly to avoid phase effects. The proportion of packets to be marked is a function of the queue length.

This contrasts with BFCA where the cut in window size is less drastic, phase effects are not a problem, loss is not the congestion indicator. Consequently a similar signal can be sent to each

[1] Implicit or explicit.
[2] It is however possible for marking to be used if end protocols understand the marking

connection and fairness between connections is aimed for.

The aim of both schemes is to hold the long-term average queue length low whilst having sufficient buffering to cope with large bursts of packets. This becomes particularly important in high speed networks were large amounts of buffering may be required, but the TCP algorithms may result in these buffers being fill with a consequent increase in RTTs.

8.3.2 Explicit Congestion Signaling

The use of an explicit congestion notification potentially increases the information that can be communicated from a point of congestion, which allows better adaption to network conditions. For example the replacement of packet dropping with marking potentially allows a host to distinguish between severe congestion, which is still indicated by a packet drop and on setting congestion which is indicated by the receipt of marked Acknowledgments. Being able to distinguish between the two cases would allow the appropriate response to each for example where loss indicates congestion recovery should be undertaken and marking indicates congestion avoidance.

Packet dropping as a mechanism for indicating congestion has obvious drawbacks; the dropped packet will have used network resources which are therefore waisted and TCP has problems in recovering efficiently from packet loss if its window is small. Duplicate acknowledgments will not work and the wait for an RTO may be in the order of a second.

There are however costs associated with generating an explicit congestion signal; routers need to expend resources monitoring and marking packets, some bandwidth may be consumed in transporting the marks through the network and end points need to be able to respond to explicit notifications. If routers already perform active queue management (e.g. RED routers) resources already need to be monitored and so the additional cost of introducing explicit signaling is reduced.

Onsetting congestion may be indicated by a router setting a bit in a data packet header, or by setting fields in or generating a special resource management packet. The advantage of a resource management packet is that more information may be carried, for example the rate at which a source should adjust its transmissions to, allowing quicker source convergence to its fair share than is achievable using a binary scheme. The disadvantage is that more intelligence is required in routers and more bandwidth will be consumed by the control information.

Three proposals to add explicit congestion notification to the Internet are discussed below. Explicit Congestion Notification (ECN) [RF99] simply replaces the RED behaviour of probabilisticly dropping packets with probabilisticly marking them. Backward Rate Explicit Congestion Notification would generate explicit rate indications on the back channel. Priced Explicit Congestion Notification [CKW97, KM99, KMBL99] proposes marking all packets that contribute to a congestion event.

When implicit congestion notification is used there is no way for an intermediary node to redirect notifications. With explicit notifications they may be erased or copied from one flow to another. This property may help in the realisation of differentiated service on the Internet. There are three levels of aggregation which are relevant here:

1. Application
2. Host
3. Network

Consider a conferencing application. It may be desirable to hand off congestion notifications from the whitebeard and voice streams to the video stream. At the host a large background FTP transfer may receive the notifications from a interactive web traffic. At the network level, congestion notifications for hosts identified as having important users may have their congestion

notifications handed off to less important hosts. This of cource implies monitoring entities being active at each of these levels.

This is simply an extension of Random Early Detection gateways; when average queue lengths are between the minimum and maximum thresholds packets for ECN compliant flows are marked instead of being dropped.

In the DECbit scheme all packets passing through a congested router are marked and the end node responds by reducing the congestion window by at most 1/8th every two RTTs. In the ECN scheme a proportion of packets are marked but the response at the end node is more severe; the congestion window may be reduced by half every RTT.

ECN preserves the statistical fairness properties of the TCP and RED algorithms, and does not improve on them by requiring a reduced and uniform response. The response of a TCP connection to an explicit congestion notification is the same as if a packet drop was discovered by duplicate acknowledgments. The `ssthresh` is set to half of `cwnd` and then the congestion window is halved. Slow Start is not performed and there is no need to wait for a timeout before continuing transmissions.

ECN is intended as a mechanism which can be used by any transport protocol, and it is designed to allow it to be utilised by TCP with the minimum of changes. When a connection is initiated the end transport protocols determine whether ECN is mutually supported. If it is in each subsequent packet sent an ECN supported bit is set in the IP header. A router may indicate congestion by modifying a Congestion Experienced (CE) bit. The CE bit is echoed in the transport header by the receiving transport protocol. All subsequent packets to the source will contain a congestion warning until the source replies by setting a Congestion Window Reduced bit in the transport header. The source should only respond to one congestion notification per RTT.

The consequence of this specification is that the notifications are tightly bound to a particular connection and the possibility of handing off notifications between streams is complicated. This maybe undesirable, because congestion is an attribute of a network path, the introduction of Explicit Congestion Notification involves routers generating a signal which indicates the level of congestion being experienced. If this signal is intelligible to intermediary entities it will allow it to be utilised in as yet unforeseen ways. By binding the signal to a particular connection the ways in which it can be used are restricted.

Comparisons of RED [FJ93a] gateways using packet drops and markings have yielded mixed results. Krishnan [Kri98] finds an increase in throughput of around 20%. The initial RTO timeout is however set to 6 seconds for most of the simulations. Floyd [Flo95] only finds an improvement if the flow control window is small and TCP can only recover from congestion. Taken together these results indicate that ECN can improve TCP performance if TCP is unable to quickly detect and recover from packet loss.

Backward Explicit Congestion Notification

Like ECN this proposal is premised on the existence of RED gateways that probabilisticly mark packets in the presence of incipient congestion. The congestion experienced and congestion capable bits in the IP header are treated in the same way. Congestion notifications are not however echoed back to the source in a transport header, rather they are carried in a Source Quench packet on the reverse path to the source directly from the router. It is claimed that this has a three of advantages. The source receives a congestion notification more quickly. Congestion signaling remains at the network layer. The level of congestion can be indicated. It is further argued that the overhead of generating and transporting a Source Quench packet, will be compensated for by a more timely response to congestion notifications.

Priced Explicit Congestion Notification

Feedback is not given to hosts in a probabilistic manner rather each packet that contributes to a congestion event is marked [KMBL99]. In practice packets contributing to a congested busy cycle that are serviced before the congestion event cannot be marked. This is compensated for by marking packets at the start of the next busy cycle, so that the total number of packets marked is correct. This has the advantage of increasing the amount of feedback that a host receives, and thereby reducing the size of response required for each congestion notification, which in turn addresses the problem of traffic phase effects and allows for greater fairness between streams. The precise definition of a congestion event is undefined, thereby allowing implementation flexibility.

8.3.3 Enforcement

From the start of packet switching networks the advantages of enforcing congestion control in the network have been understood. The Internet is one of the few such networks without enforcement mechanisms. Changes in the nature of the Internet are leading to a consensus that existing enforcement mechanisms are inadequate and need at least to be supplemented by mechanisms located in the network.

Today the Internet relies upon the implementation of standard algorithms in host TCP stacks to ensure fairness between connections. Traffic which is not carried by other transport protocols may not be regulated by the same algorithms. There is therefore an effort to capture the behaviour of TCP in models, so that developers of applications and transport protocols may design congestion control mechanisms which behave in a similar way to TCP's without constraining them to use the exact algorithms, thus maintaining fairness but introducing a degree of freedom in the design of congestion control mechanisms [3].

Whilst the development of TCP friendly congestion control makes it possible for a diverse set of applications to use a standard congestion control mechanism, it is still the case that selfish improvements in performance can be achieved by making applications unresponsive to congestion indications. This results in TCP streams receiving a reduced share of bandwidth and may also result in a global decline in efficiency and even congestion collapse.

These concerns lead to the conclusion that it necessary to deploy enforcement mechanisms in the network, which penalise unresponsive flows and therefore provide an incentive for developers to design and users to utilise applications which are response to congestion notifications from the network.

Scheduling

Scheduling of packets can be used to separate the flows at a router so that each receives an equal share of the available bandwidth. There are numerous such schemes which achieve similar results. Fair Queuing [DKS95] and Fair Queueing with Round Robin [SV95] are two examples.

This approach has been criticized for requiring per flow state to be maintained at each router and for not being able to prevent congestion collapse caused by too many flows and therefore not being an appropriate alternative to adaptive congestion control. The packet pair scheme is an example of a black box end-to-end congestion control mechanism that requires fair queuing and exhibits superior fairness and efficiency characteristics than the standard TCP congestion control mechanisms. It can ensure that bandwidth is distributed evenly among flows, usually according to max min criteria, although weighted versions exist which allow bandwidth to be distributed proportionally.

[3] If rate based flow control is used in preference to window flow control the TCP algorithms cannot be simply ported

The complexity of implementation in each router is the main barrier to the deployment of per flow packet scheduling mechanisms.

Measurement

The proportion of bandwidth that is consumed by a flow can be estimated from an RED routers packet drop/marking history. The random marking of packets means that the probability of packet in a particular flow being marked will be proportional to the amount of bandwidth that the flow is consuming. The idea is for a RED router to examine a flows packet drop history and employ enforcement measures against flows which are found to be non conformant.

RFC2309 [BCC+98] *Recommendation on Queue Management and Congestion Avoidance in the Internet.* Identifies non Responsive and Non-TCP compatible flows as possible causes of discrimination against TCP flows and future congestion collapse Floyd adds to these two categories flows which consume a large amount of bandwidth during congestion.

> *A flow that is not* TCP - friendly *is one whose long term arrival rate exceeds that of any conformant TCP in the same circumstances. ... An unresponsive flow is one failing to reduce its offered load at a router in response to an increased packet drop rate. ... A disproportionate bandwidth flow is one that uses considerably more bandwidth than other flows in a time of congestion.*

Pricing

Frank Kelly [KMT98, GK98] proposes a more flexible approach. The central idea is to make the resource implications of a stream's action known to it and allow the end node to decide what action to take. In this way information about the level of congestion is explicitly conveyed from the router to the source taking advantage of the routers global view of the traffic. But it is left to the source that knows the importance of the stream to decide what to do. Attached to the notification is a price so in the presence of congestion the end node pays a penalty for the use of the resource, thus providing the incentive to reduce traffic when congestion exists. It is then up to the end node or protocol implementers to devise ways of maximising utility whilst minimising the use of resource. Thus the problem is distributed.

A possible problem is that a large administrative load would be imposed upon the Internet in implementing such a stream. Certainly a charging mechanism for each stream or host would be complex. If the unit of charge was by autonomous system or institution then the number of units needing accounting would however be greatly reduced. Each institution would then take responsibility for the use of WAN resources and could introduce internal policies based upon its own priorities and policing of traffic going onto the WAN. Each institution connected to the network would have an incentive in minimising congestion and therefore cost. Allowing more resources for critical traffic.

A weighted proportional fair sharing TCP as proposed by Crowcroft [CO98] is an example of the sort of implementation that could take advantage of the evolution of congestion control proposed by Kelly. It is aptly named MulTCP and allows one TCP connection to receive the network share normally associated with several.

8.4 Improving TCP's Start-Up - Getting to Steady State

Slow Start performs the two roles of getting TCP's self clock going and quickly taking a connection to its operational bandwidth. In this section mechanisms for improving the second role are discussed. There are two ways in which Slow Start can be improved, by increasing the accuracy with

which the operational bandwidth is located and by increasing the speed with which a connection reaches that bandwidth. In both cases it is necessary to arrive at an estimate of the bandwidth that the connection should be using before a packet loss has occured.

These challenges arise against the background of increasing variation in the amount of bandwidth available and the level of multiplexing on Internet paths. In addition a large proportion of connections with sufficeint data to benefit from utilising the bandwidth but are prevented from doing so by small initial window sizes.

The work surveyed in this section directly addresses the same key challenges as this dissertation and provides evidence that a number of other researchers consider the issues addressed here to be important. As will be seen in the discussion the methods used to address TCP's Start-Up differ significantly from those advocated in this dissertation thereby demonstraing the originality of this work.

The work surveyed in this section is categorised by the scope over which congestion information is shared. First work which utilises the same scope as TCP, only congestion feedback which is generated in direct response to the connection's own traffic is considered. Next various approaches to sharing congestion information from connections between a pair of hosts is discussed. Finally work which facilitates the sharing of QOS feedback between aggregates of hosts is considered.

8.4.1 Connection Confined Adaptation

In this section ways of improving TCP start-up by inferring path characteristics from a single connections traffic are discussed. Research using this approach has focused on the problem of preventing the connection from overshooting its available bandwidth and does not directly address moving that level of bandwidth utilisation more quickly.

The lower bound on end to end delay is set by the transmission speed across the medium, which is constant. Therefore as bandwidth increases so too does the amount of traffic that can be in the network at any point in time. This in turn means that the minimum number of round trip times that it takes to transfer a fixed quantity of data decreases. Work has shown that the size of TCP connections is either not growing over time or increasing slowly [Cac92, Mog95a, Mog95b, BSSK97]. Each connection will on average take fewer and fewer RTTs. As one RTT is the minimum amount of time that a connection has to receive feedback the opportunity that exists for current TCP implementations to adapt to network conditions is decreasing [Pax94a].

As the lifetime of connections decreases the start-up behaviour increases in significance. The limit to this process is when the data phase of each connection can be completed in one RTT [Pax94a].

Overshooting Bottleneck Bandwidth

At the start of a TCP connection, the congestion window is initialized to one segment and increased by one segment for every acknowledgment received. Consequently in one RTT the bottleneck queue can be overflowed by half of the congestion window in one RTT[4]. It takes most TCP implementations multiple RTTs to recover from the ensuing multiple packet loss.

If the bottleneck queue capacity is reached whilst TCP is operating in its steady state however, the congestion window only increases by one per RTT and the connection will only overshoot the queue by one packet. TCP's Fast Retransmit and Recovery algorithms can efficiently recover from single packet drops. It follows that it is preferable for the connection to be in its steady state mode when packet loss occurs.

TCP infers characteristics of the PATH, including the available bandwidth, during a connection, at startup the bottleneck and available bandwidth are unknown. Most TCP implementations

[4] if delayed acknowledgments are being used this is reduced to 25% of `cwnd`.

therefore set `ssthresh`[5] to the maximum window size it therefore only becomes active after packet loss has occured.

On most connections there will be a number of RTTs between the start of data transfer and a connection saturating the bottleneck queue. A number of authors advocate utilising this time to derive an estimate of the window size at which steady state behaviour should be entered. Four such schemes are discussed below:

1. Vegas [BOP94]
2. Packet Pair Variant (PPV) [Hoe95, Hoe96]
3. Tracking Closely Spaced ACKs (TCSA) [AD98]
4. Receiver Sided Estimation (RSE) [AP99]

Vegas The method used in Vegas is to compare the expected throughput with the achieved throughput in the same way as is done for Vegas' steady state behaviour as described in Section 8.2.4. The congestion window is only increased every other RTT to allow estimates to be made and if the difference between actual and expected throughput falls below a threshold Slow Start exited.

Packet Pair Variant Hoe advocates estimating a value for `ssthresh` for each connection at start-up. By initialising `ssthresh` to 64 segments. The SYN segment is then timed to find the RTT. The intervals between the first three ACKs are measured to get the bandwidth. The calculated value is then used to set `ssthresh`.

Tracking Closely Spaced ACKs This is based on the Packet Pair Variant, but the result from the first three ACKs is not used to set `ssthresh`. Instead results are computed for subsequent groups of acknowledgments. When two sets of readings are found to be within 10% of each other the average of the two is used to estimate the bottleneck bandwidth.

Receiver Sided Estimation Vern Paxson observes that although the receiving TCP cannot directly manipulate the sending congestion window, it is in a better location to estimate the bottleneck bandwidth than the receiver. Consideration of how to communicate estimates is postponed and the accuracy of Receiver Sided Estimation investigated.

One reason for inaccuracy of sender sided estimates based on the spacing between ACKs is that ACK compression may occur on the return path. If the receiver estimates the bottleneck bandwidth from the separation between arriving data packets the ACK compression problem is eliminated. It is however necessary to know that the separation was introduced by the network and not by the host. If the sending application is not the bottleneck [6] the receiver can deduce which packets will be sent in response to the receipt of an Acknowledgment. These packets are likely to be sent back to back and therefore any separation between them at the receiver will have been introduced by the network. This method has the added advantage of allowing a reading to be made for every acknowledgment sent.

Performance If the available bandwidth estimate is too low Slow Start is exited earlier than necessary and it will take more round trip times than necessary for the connection to get its fair share of bandwidth. The length of the connection will therefore be increased. If the bandwidth estimate is too high then the congestion window will open exponentially beyond the connection's fair share and multiple packet loss is likely.

[5] Which marks the window size at which Slow Start should and TCP's steady state behaviour starts
[6] In which case packets will be smaller than the MSS

Vern Paxson [AP99] has used simulations driven by packet traces to estimate the effectiveness of each method, apart from Vegas. He found that in the traces where packet loss occured during Slow Start, it would have been prevented 12% of the time by PPV, 11% of the time by TCSA and 24% for RSE. In 4, 2 and 7% of the respective cases Slow Start would have been exited sufficiently early to significantly increase the transfer time.

Relying on loss as a signal for congestion does not work well at the start of a connection. This has lead several researchers to investigate ways of estimating the bandwidth that a connection should be operating at, prior to packet loss occuring. The aim is to use such an estimate to decide when to switch from slow start to steady state behaviour and thereby reduce the degree to which available bandwidth is exceeded to one packet per RTT.

Persistent - HTTP

With the growth of the World Wide Web HTTP accounts for a large proportion of Internet Traffic. HTTP traffic shares some of the characteristics of telnet traffic and some of the characteristics of FTP traffic. Much of it is generated interactively by people using a Web browsers; this interactivity means that response time is more important than for FTP traffic [Mog95a]. It however involves larger transfers. A web session will typically produce a large number of transfers groups most of which will be to and from the same host. The down load of a single page may involve the establishment of several different connections. These connections may run serially or in parallel. For example Netscape can be configured to open several parallel connections the default being four.

Whether the browser generates serial or parallel connections, problems ensue. By opening parallel connections a greater share of the available bandwidth is being asked for, this is in effect buying extra performance at the expense of other traffic. If serial connections are opened a high overhead for each connection is entailed by having to go through TCPs three-way handshake and then go through Slow Start before all the bandwidth is utilised. This adds delay which may be perceptible to the user.

One solution to these problem was proposed by Mogul [Mog95a, Mog95b] and is now incorporated into HTTP 1.1. The proposal is to allow many HTTP requests to be carried in-sequence by a single TCP stream. In this way the multiple handshakes can be avoided and the connection will have sufficiently long life to be able to adapt to network conditions and therefore utilise the available bandwidth. The way in which P-HTTP interacts with TCP's restart after idle policies has however proved a barrier to the potential increase in performance being reached [Hei97]. Between each HTTP request that is carried on a single TCP connection an idle state is likely to occur. TCP is idle when all data sent has been acknowledged.

Congestion Window Validation

In a paper entitled Congestion Window Validation [MJS99] it is argued that when a connection is idle for long periods of time or if a connection is application limited the size of the congestion window may be invalid in that it does not reflect network conditions.

A simple modification to TCP's congestion control algorithms is proposed. The congestion window should be decayed exponentially if the idle period is sufficiently long. Sufficiently long is defined as greater than a RTT. It is also proposed that the previous size of the congestion window be remembered using the Slow Start Threshold, which should be set to the maximum of its current value and 3/4 of the current `cwnd`. In addition it is argued that the congestion window should not be increased if the connection is application limited. That is the congestion window should not be increased it the previous utlised window was smaller than the existing window. Whilst the work is aimed at addressing idle periods within a connection, it can easily be extended as a strategy for decaying the size of a cached congestion window between connections.

Larger Initial Windows

A current proposal exists to allow a larger initial congestion window. This is the subject of an Internet draft [AFP98b, PN98a] and supporting research [AHO98, AHKO97, All97, PN98b, AFP98a]. The exact proposal is to allow an initial window of up to the minimum of four packets and four kilobytes unless the segment size is greater than 2190 bytes in which case the maximum initial window size will be two segments.

Min(4*MSS, max(2*MSS, 4380 bytes))

There is a strong motivation for changing the initial size of `cwnd`. The TCP delayed acknowledgment policy results in random delay usually between 0 and 200 ms if a single packet is received until an acknowledgment is generated. The sending of at least two segments will illuminate this delay. For small data transfers (up to 4k) the data transfer phase would be reduced to one RTT. For larger transfers up to three RTT and one delayed Acknowledgment wait would be removed from the data transfer phase thus increasing throughput and reducing the length of the connection.

These changes are particularly beneficial to high bandwidth and large propagation delay paths [AHKO97, All97]. Mark Allman shows that for a satellite link with a 112Kb window and a RTT of 585 ms it took 11 RTT or 6.5 seconds before full bandwidth utilisation is reached. The impact of this delay is most noticeable on short transfers.

In support of the proposed change it is suggested that the burstiness of traffic on the Internet would not be dramatically increased because traffic is already bursty. In particular bursts of two or three segments are already typical of TCP and are caused by Slow Start, delayed acknowledgments and dropped acknowledgments.

Burstiness of a TCP connection is not however simply caused by the number of packets released by a single acknowledgment. The receipt of closely spaced acknowledgments will result in a sequence of closely spaced packets being placed on the network. This can be aggravated by the phenomena of ACK compression [Mog92, ZSC91]. If TCP connections start with an initial window of four packets when the ACK for these come in eight packets will be introduced into the network in the second RTT of data transfer. Under normal Slow Start this would not occur until the fourth or fifth RTT of data transfer. If each passage through the network on average introduces a separation between packets, then this separation will be much lower for an initial window of 4 than for an initial window of 1 packet. Thus it can be expected that an increase in the initial window size will lead to burstier TCP traffic.

One experiment [AHO98] was designed to test the effect on a connection's performance by altering the starting size of the congestion window. Starting window sizes of between one and thirty-two segments were compared, using 20 Kb file sizes and 512 byte packets. All the transfers were all from one site to the discard service of a hundred hosts, each of which had an offered window greater than 16 Kb. A significant increase in throughput and a small increase in packet loss are reported for initial window sizes up to 16 Kb.

The result show (as described by the authors) a dramatic increase in throughput as the initial window size increases, accompanied by a modest increase in packet loss. A window size of four packets achieves a 25% reduction in transfer time and a lower increase in packet loss than 2 and 3 packet windows and is therefore recommended as a good starting size for TCPs initial congestion window. The small reduction in transfer time between 16 and 32 segment windows is caused by the increase in packet loss resulting in more RTO time outs.

A number of questions are left unresolved. Firstly, the increase in initial window size is tested in favourably circumstances. The authors establish that the common path between the test site and the rest of the Internet is lightly loaded. Secondly the packet size of 512 Kb is half of the maximum proposed for a four-packet minimum. A threshold at which improvement in transfer time falls off rapidly is identified at around a 32 packet initial window and is explained as resulting from packet loss. The effect of reaching this threshold in the second or third RTT is not investigated because the use of 20K transfers means that the 32 packet initial window is the only occasion when 32

Window Size	% Reduction in Transfer Time	Increase in Dropped Segments
1	0	0
2	3	0.08
3	16	0.08
4	25	0.04
5	31	0.15
8	44	0.4
16	59	1.3
32	61	2.8

Table 8.1: The effect of different Initial window Sizes
Read from graph in [AHO98]

packets will be transferred in one RTT.

K. Poduri and K. Nichols [PN98b] have also conducted a number of simulations that support the contention that increasing the initial window to 3 or 4 segments increases performance without harming background traffic.

Summary

Statically increasing the Initial Window (IW) size of TCP can increase throughput and need not result in significantly increased loss rates. The exact effect depends on the network conditions. The problem with moving from one fixed size to another fixed size is that one size of congestion window cannot be correct for all situations. The initial window size should never be greater than connections fair share so if the initial window is standardised at four segments this will be too large for some paths. On the other hand it may be possible to start with a congestion window larger than four in lightly loaded networks.

8.4.2 Host Based Sharing

Other researchers have taken the approach of extending the amount of data over which TCP's congestion control mechanisms operate. For example Padamanabhan [Pad98] proposes session level control, where the congestion window is maintained over concurrent connections and saved between connections. Integrated Congestion Management [BRS99] extends this approach to data transfered using different transport protocols. Floyd's work on Congestion Window Validation [MJS99] advocates exponential decay of the congestion window for each RTT of inactivity. Whilst this work addresses similar issues to this paper, it does not facilitate the sharing of information between hosts. In addition our measurements of interconnection gaps suggest that the decay policy would severely curtail the number of connections that would benefit and our measurements of the decay of correlation between predictions and results suggests that it is unnecessarily conservative.

Ensemble TCP

In TCP Control Block Interdependence [Tou97] and Ensemble TCP [Lar99] Touch et all discuss techniques for re-using information among connections in series and aggregating information among connection in parallel. This is motivated by observing that TCP carries over 90% of all Internet bytes and of the TCP connections a high proportion are Web Transfers. It is observed that Web Transfers are often too short to adapt to network conditions and that the practice of browsers opening multiple concurrent connections may contribute to congestion.

The aim is two fold. The first is to make a collection of concurrent TCP connections, between the

same endpoints, to behave as if they were a single connection. The second is to reuse information about the path from previous connections to initialise TCP control variables, thereby speeding the adaptation to current network conditions. This is achieved by moving the maintenance of certain variables from the TCP Control Block (TCPCB) to an Ensemble Control Block (ECB), which is used to cache values and enable sharing of state information. The ECB allows the Round Trip Time, the Smoothed Round Trip Time the Variance in Round Trip Time the Congestion Window and the Slow Start Threshold to be calculated across multiple connections.

In the event of a bundle of connections starting with a large initial window Rate Based Pacing is advocated for smoothing the first window of packets. Using the above techniques information may be shared between TCP connections that sharing the same end hosts. The location of the sharing mechanisms within the host allows this sharing to occur at the packet granularity. A novel feature of the work is the employment of scheduling algorithms, which give priority to the initial HTML page thereby improving interactivity. The work is supported by simulation studies that compare Reno TCP and Ensemble TCP. It is shown that there are significant increases in performance and reductions in delay.

TCP Fast Start

TCP Fast Start [aRHK98] predates Ensemble TCP, but takes a similar approach and motivation. It is observed that web browsing generates short bursts of data transfer interspersed by idle periods and that TCP yields poor performance for such a workload. In addition it is observed that Slow Start Restart after an idle period, usually requires several round trip times after an idle period to probe the network and consequently nullifies much of the advantage that could accrue from using P-HTTP.

The approach adopted is to cache the Congestion Window, Slow Start T@hreshold and the `srtt`, to enable a subsequent connection to avoid paying the Slow Start penalty for each down loaded page. It is argued that this approach carries the danger of too aggressive behaviour if the cached information is stale. This is addressed by assigning Fast Start packets a higher drop priority so that in the event of congestion they will be dropped first. TCP loss recovery is augmented to allow quick recovery from dropped Fast Start packets. The smoothed round trip time and congestion window are used to calculate a rate for pacing out the first window of packets.

This work is supported by simulation and some experimentation. The simulations show a factor of 2-3 reduction in latency for Fast Start and a small reduction in latency for competing TCP flows. The need to deploy modifications to routers to allow priority marking to work curtails the experimental evaluation and introduces an extra requirement for deployment. Fast Start focuses on state sharing at the start of a connection, but is accompanied by complementary research into Session level congestion control. Fast Start allows TCP connections within a host that share a common foreign end point to share state.

Integrated Congestion Manager

Whilst Ensemble TCP and Fast Start address the mismatch between Web transfers and TCP by advocating changes to the transport protocol the Integrated Congestion Manager (CM) [BRS99] starts from a broader problem definition and aims to produce a more generic solution.

It is suggested that the mismatch between the needs of applications and the service provided by TCP is leading to increasing numbers of applications that do not use TCP, and that these applications frequently do not implement effective congestion control. It is also observed that the desire to reuse TCP's congestion control leads to applications that are not well served by TCP's service using it non the less. It is argued that there is a need for an end system architecture centered around a Congestion Manager, which integrates congestion management across Transport Protocols and Applications.

An API is defined, which allows applications to learn about network conditions and to pass such information to the congestion manager. The application is allowed to decide which data should be transmitted and with what priority, whilst the actual scheduling is the responsibility of the congestion manager (CM).

It is envisaged that the congestion manager would use a Window Congestion Management scheme, but that packets within each window would be scheduled using some form of rate based pacing. A light weight protocol to elicit feedback from receivers is defined, but the Congestion Manager can still function if implemented at the server side only. Exponential ageing of cached data values is advocated.

The work describes how TCP and an Audio application could be implemented to work with the congestion manager. It is claimed that the CM provides a useful and pragmatic framework for building adaptive Internet applications. The application is left in charge of what to send and how to distribute sending opportunities between its flows. The congestion manager is situated between the Transport and IP layers, provides state sharing at a fine granularity between data flows that utilise different transport protocols but emanate from the same application.

A Web Transport protocol WebTP

A third host based approach to the mismatch between TCP and Web traffic is to design a new transport protocol an example of which is called WebTP [GCMW99]. This has the advantage of allowing the protocol to be designed with both the needs of Web traffic and the understanding of congestion control in mind.

WebTP is a receiver oriented request and response protocol, which enables Application Level Framing of data. Although only a prototype description and partial simulation level implementation is presented it has a number of novel features. A central design aim is the migration of complexity from the server to the client side of transactions.

The three way TCP connection set up and handshake is removed, thereby reducing the overhead for short connections. The responsibility for requesting retransmissions is left to the receiver and there is no requirement for the sender to maintain packets until they are acknowledged.

The abstraction of Application Data Units (ADU) is used for each unit that is to be retrieved from a web page. This allows the application to specify which element of a web page should be retrieved first, using the meta data contained in an HTML header. Each ADU may be fragmented into multiple packets if necessary.

Congestion control is located at the receiver and is based upon TCP's established Slow Start and Congestion Avoidance algorithms. The congestion window increases exponentially until a loss occurs and then the linear increase phase is entered. The size of the congestion window is used to define what Application Data Units and packets will be requested. A rate at which these should be transmitted is calculated by the receiver using the equation R = `cwnd`/`srtt`, and communicated to the sender, which paces out the packets. Thus a hybrid Window and Rate based scheme is proposed.

It is asserted that the values for the congestion window and `srtt` may be cached between page downloads and re-used and that the removal of the connection set up overhead means that it is not necessary for connections to be left open.

This work is supported by simulations in the ns simulator, using long lived connections. It is shown that high levels of link utilisation are achieved, that bandwidth is shared fairly between WebTP connections and that competing TCP connections are not starved of bandwidth[7].

[7]Although WebTP does obtain a higher proportion of bandwidth, this may in part be a result of the design decision not to halve the congestion window when WebTP equivalent of a multiple Duplicate ACK event occurs.

8.4.3 Inter Host Sharing

The ability to widen the information used to derive estimates of network conditions to all traffic on a local subnet with a common destination is desirable. In particular it would allow information to be shared between a cluster of web servers that have similar patterns of requests.

In this subsection four pieces of work which relate to the sharing of information between hosts on a common subnet are discussed. First the Network Probe Daemon, which actively probes the network to discover path conditions and was developed by Vern Paxson [Pax96a] is considered. Whilst it does not directly address sharing information between hosts it is the result of work which was motivated by the desire to achieve this end [Pax94a]. Secondly, SPAND is discussed. SPAND uses passive monitoring of traffic at the application level and active reporting to a centralised repository to enable applications to share information and adapt to network conditions. Thirdly NewSPAND is considered. This adaptation of the SPAND architecture again uses passive monitoring, but the aim is to allow the sharing of information at the Transport level. Finally techniques which aim to discover, which flows share a common bottleneck are discussed. If successful these techniques would further widen the number of destinations for which congestion information could be successfully shared.

Network Probe Daemon

Vern Paxson develops a Network Probe Daemon (NPD) [Pax96b] that creates TCP connections and captures packet traces for post processing. Whilst this approach is useful in determining underlying network characteristics it is not appropriate for providing feedback to hosts in real time. The use of dual pass algorithms requires the maintenance of to much data to be practicable and the need for probe traffic may increase congestion rather than avoid it. In addition it also poses the question of where probes should be sent, which is elegantly solved by passive monitoring of traffic. It should of course be noted that this is not the declared design aim of the NPD.

SPAND

Mark Stemm developed and implemented Shared PAssive Network Discovery (SPAND) [SSK97, Ste99]. This is characterised by three design principles, passive monitoring, application level metrics and sharing of information between applications on a host. It is argued that passive monitoring has a number of benefits over active probing. In particular the level of information feedback is automatically tuned to the time of day and paths over which the most traffic is generated. The absence of probe traffic means that additional load is not introduced to already congested paths.

There is no attempt to directly measure network characteristics, rather these are inferred from application level metrics, which are stored in a centralised repository, which may in turn be queried by hosts. The disadvantage of this approach is that a number of factors are included in the metrics that bear little relation to network characteristics. For example the network RTT is exaggerated by delayed acknowledgments, throughput is greatly reduced if a TCP connections happens to recover from a loss by an RTO rather than by duplicate acknowledgments. The evolution of the congestion window means that the size of a transfer is a prime determinant of the bandwidth which it receives and it is a factor that bears little relation to the prevailing state of network conditions. All these factors are included in an analytical category called network noise by the authors.

Upon the retrieval of a metric from the shared repository it may be used by the application to adapt its behaviour to perceived network conditions. Two applications have been built to demonstrate this. In the first case a web browser dynamically adjusts the fidelity of images depending on the level of congestion. This introduces the concept of changing the amount of data that is transmitted in response to congestion. In the second the least congested of a set of mirror sites is selected.

An addition drawback of the use of application level metrics is that they may inhibit the sharing of information between different applications, where one application may not understand the others metrics. Substantial experimental work using an implementation of SPAND and of the adaptive applications is presented in support of the work. It is demonstrated that substantial reductions in the proportion of connections that take more the 10 seconds to complete are achieved.

NewSPAND

NewSPAND [ZQK99] is based upon the SPAND architecture but develops it in important respects and has different design aims. The aim of NewSPAND is to share information about values for TCP control variables between hosts on a subnet that share the same receiver. The monitoring and sharing therefore takes place primarily at the Transport and not the Application level.

Rather than waiting for a client to request information from the server a new request to a Web server is detected and a report on the likely network conditions is sent. This reduces the delay between connection request and the receipt of congestion information and cuts the amount of local area network traffic.

This work is supported by the definition of an architecture, some analysis of the benefits of using a larger initial window size and some simulation work.

Shared Congestion Discovery

If the abstraction of a single bottleneck is adopted it follows that it may be desirable to share congestion information between all flows that have a common bottleneck link. A prerequisite to achieving this level of sharing is to be able to detect, which flows share a common bottlenest. This question is addressed in the paper Detecting Shared Congestion of Flows [RKT00a].

These techniques rely upon shaping the traffic in ways which may not be desirable, they are therefore a significant step forward in detecting shared congestion but do not show that it is possible for a practice implementation.

8.4.4 Summary

The start-up behaviour of TCP has been the focus of considerable research. In particular three separate but interconnected problems have been identified i) how to pace out the first window of packets, ii) how to prevent a connection from overshooting the available bandwidth and iii) how to reduce the cost of start-up.

TCP's Slow Start algorithm performs two functions, it allows a connection to probe for available bandwidth and starts TCP's Ack clock, which implicitly shapes traffic within a window. The use of a larger initial value for the congestion window poses the question of how to shape at least the first window of traffic. Rate Based Pacing has been proposed as a means of preventing large bursts after an idle period in a connection [VH97, HTH98] and at the start of a connection [Pad98] when a large initial window is used. It is discussed in the next section.

Hoe [Hoe95, Hoe96] showed that Reno TCP was unable to recover efficiently from multiple packet loss. This observation lead to the development of New Reno, SACK [MMFR96a] and FACK [MM96], which improve recovery from packet loss and to preventative measures, which aim to switch the congestion increase algorithm to congestion avoidance before loss occurs. This is achieved by using the first few packets of a connection, to measure the round trip time and available bandwidth. The product is then used to set `ssthresh`. Available bandwidth is determined from the spacing of acknowledgments, using techniques similar to those developed for Packet Pair Flow Control [Kes91], although important assumptions such as fair queuing [DKS95] do not apply. Refinements on the above scheme are proposed in [AD98], which uses several groups

of acknowledgments to estimate bandwidth, and Paxson [AP99], who advocates Receiver Side Estimation to avoid ACK compression [ZSC91, Mog92].

The above work does not address the high proportionate cost of ramping up to fair bandwidth for short connections. Mogul proposes Persistent HTTP [Mog95a, Mog95b], now incorporated in HTTP/1.1 [FGM+97], which enables multiple HTTP requests to be carried in a single TCP connection. In this way the cost of multiple handshakes are avoided and it is hoped that each connection will be sufficiently long lived to adapt to network conditions. The generality of the solution is however limited to HTTP and there are complications associated with the interaction between TCP's restart after idle policies [Hei97] and P-HTTP.

The alternative of simply increasing the default initial window size has been explored by Allman [AHO98, AHKO97, All97], with particular reference to satellite links.

Other researchers have taken the approach of extending the amount of data over which TCP's congestion control mechanisms operate. For example Padamanabhan [Pad98] proposes session level control, where the congestion window is maintained over concurrent connections and saved between connections. Ensemble TCP [Lar99] facilitates host based sharing of information between connections at a fine granularity. Integrated Congestion Management [BRS99, BS01] extends this approach to data transfered using different transport protocols. Floyd's work on Congestion Window Validation [MJS99] advocates exponential decay of the congestion window for each RTT of inactivity. Whilst this work addresses similar issues to this paper, it does not facilitate the sharing of information between hosts. In addition, exponential decay of the congestion window, at the rate suggested, will preclude information sharing between transfers associated with separate web pages.

The sharing of information between hosts could improve estimation by increasing the information available. This could be facilitated by using probe traffic, by end hosts passively monitoring existing traffic and reporting results to a server, or by the server monitoring traffic and calculating estimates itself. Paxson developed a Network Probe Daemon (NPD) [Pax96b] that creates TCP connections and captures packet traces for post processing. Mark Stemm developed and implemented Shared PAssive Network Discovery (SPAND) [SSK97, Ste99]. This is distinctive in that it does not attempt to determine network RTT or congestion levels but instead relies upon applications to determine their own metrics and store these in a centralised repository. The disadvantage is that the sharing of information between different applications may be inhibited. NewSPAND [ZQK99], facilitates the sharing of information between TCP connections on separate hosts. The server receives reports on window size from TCP connections, aggregates them and makes the results available at the start of new connections. A dependence on the accuracy of TCP's estimates is therefore built into the estimation process.

8.5 Improving TCP's Start-Up - Traffic Shaping

TCP's Slow Start algorithm performs two functions, it allows a connection to relatively quickly probe for available bandwidth and starts TCP's Ack clock, which implicitly shapes traffic within a window. Dynamic initialisation of the congestion window posses the question of how to shape at least the initial window of traffic. Rate Based Pacing has been proposed as a means of preventing large bursts after an idle [HTH98] period in a connection and at the start of a connection [Pad98] when a large initial window is used and is orthogonal to the work presented in this paper.

TCP has no explicit mechanism for shaping traffic. The size of the receiver's offered window however, limits the amount of unacknowledged data that can be outstanding in the network at any point in time. As memory available on workstations continues to increase so it becomes possible to have larger receive (and transmit) buffers. Consequently the opportunity exists to have larger offered windows. This is particularly beneficial on paths with a large delay bandwidth product.

The offered window also places an upper limit on the size of burst that can be placed on the network. As the window sizes increase then so to does the upper limit on the burstiness of a connection.

Window based flow control has a number of advantages; it is simple to implement, places low demands on end system, automatically adapts the rate of transmission with variations in the RTT and automatic limitation the size of bursts at the granularity of one RTT and over one window of data.

Rate based flow control also has a number of advantages. The Service needs of a wide class of applications are naturally expressed as a rate and the service capabilities of intermediary switches are naturally expressed as a rate. With rate based control an explicit decision is possible over the time scale or quantity of data bursts over which controlshould be exercised. Examples of rate based control include; Fair Queuing [DKS95], Virtual Clock [Zha90] and Leaky Bucket. For leaky bucket this equates to the maximum burst size.

8.5.1 The Limitations of Implicit Traffic Shaping

There are a number of ways in which using TCP's Ack clock to implicitly shape traffic can lead to busts of packets being transmitted onto the network. The phenomena of Ack compression, the Fast Recovery algorithm and the restart of data transmission after an idle period can all lead to packet bursts.

ACK compression

Van Jacobson showed (under certain conditions) that ACKs would arrive at the sending TCP at the rate that segments arrived at the receiving TCP. This was however only shown for traffic flowing in one direction and ACKs flowing in the other, for the more complex case where there is data flowing in both directions a phenomenon called ACK compression can occur. ACK compression has been identified by Shenker [ZSC91] using simulation and by Mogul [Mog92] using measurement of real traffic, both confirm the existence of ACK compression and both show that it undermines the packet conservation principle on which TCP's flow control is based. When ACKs are produced they retain the spacing of the packets they are acknowledging. If the traffic on the connection is one way then the ACKs will not encounter non-empty queues. If the traffic is two way then there may however be nonempty queues. If ACKs are held up in queues then the minimum time separating two ACKs will be their transmission time, as an ACK packet will probably contain no data, it will be considerably smaller than data packets and have a correspondingly smaller transmission time. This means that ACK packets will arrive with a closer separation than the data packets they are acknowledging.

Consequently the spacing of ACKs may not provide a reliable clock. In the presence of ACK compression packets may be generated in larger bursts than the service rate of the bottleneck link resulting in possible packet loss.

Fast recovery

TCP's fast recovery algorithm may result in a burst half the size of the receivers advertised window being transmitted at the speed of the local network. Consider the case of a single packet being dropped from the first half of a window sized flight of packets. Duplicate Acknowledgments will cause the sending TCP to detect the loss and retransmit the packet. When the retransmission arrives the hole in the receivers sequence space will be filled. All the unacknowledged packets will be acknowledged at once. When the acknowledgment arrives the minimum of `cwnd` and the receivers advertised window may be transmitted at once. The receipt of a non duplicate ACK will

cause `cwnd` to be cut in half and so its maximum will be half of the receivers maximum advertised window. This is also the maximum size of burst that can be generated in this circumstance.

Restart After Idle

The interaction between a sending application and TCP may also give rise to large bursts of packets. If TCP runs out of data to send, when acknowledgments are received the congestion window opens, but no packets can be sent. If the application later supplies more data a full window of segments may be transmitted at once. As the congestion window is designed to limit the amount of data sent over a full RTT this can be considerably more than the connection's fair share or than the network will tolerate without loss [Hei97].

In a revision of his 1988 paper on congestion control and avoidance [Jac88] Van Jacobson proposed the introduction of a send timer to detect idle periods. Upon the resumption of transmission Slow Start is to be performed, this can be called Slow Start Restart (SSR). Examples of implementations that do Slow Start Restart (SSR) after an idle period are 4.4BSD, Free BSD 2.1 and Linux 2.0 [VH97].

The use of a timer to detect an idle period is subject to the following tradeoff; if the time period is to long idle states may be missed and bursts allowed, if the period is to short Slow Start may be entered unnecessarily. The Net/3 / Reno implementation of TCP will back off to Slow Start if the idle period is at least as large as the current retransmit timeout value.

Other implementations neither detect idle periods nor do SSR, this can be called No Slow Start Restart (NSSR), an example implementation that does NSSR is Sun OS4.x [VH97].

The advantage of SSR is that it if an idle period is successfully detected less strain will be placed on the network. The advantage of NSSR is a faster resumption of transmission but at the cost of increased stress on network buffers.

A desirable solution would combine the fast resumption of transmission with a light strain on the network or provide a reasonable compromise between the two. Several alternative approaches to the existing SSR and NSSR are discussed in [HTH98] and supporting literature.

Window based sub RTT traffic Shaping and Bandwidth

Over time advances in technology have lead to increasing variation in the amount of bandwidth available to connections. As bandwidth for a connection increases the effectiveness of window flow control in limiting bursts decreases.

Consider two cases; bandwidth and window size increase and bandwidth increases but window size remains constant. In the first case, for a given data transfer, the file will be broken into a smaller number of windows. As the window size provides an upper limit on the burst size the maximum burst size will increase. At the limit the window size equals the file size and so the full file can be transmitted at once at the speed of the local network[8].

In the second case, as bandwidth increases the window will cover a decreasing proportion of the bandwidth delay product. Even if packets are evenly distributed within a window, they may be transmitted over a small section of the total RTT.

Whilst window based flow control does not explicitly control the burstiness of data within a window, TCP utilises Slow Start and implicit shaping from the network to try to achieve smooth traffic patterns. If effective this will limit the burstiness in case 1, but will do nothing to address case 2.

It has been shown that TCP's reliance on implicit network traffic shaping can break down in a number of circumstances; as a result of ACK Compression, during Fast Recovery and during a

[8] Even with slow start the upper limit is half of the file size

restart after an idle period.

TCP's self clock can break down in a number of ways, when this happens bursts are limited by the flow control window. As bandwidth increases the effectiveness of window based flow control in limiting the burstiness of traffic is decreased, whether the window size increases in line with bandwidth or not.

8.5.2 Explicit Traffic Shaping and Window Flow Control

Two approaches to addressing the bustiness of TCP connections are discussed below; tackling individual symptoms of the problem and combining window and rate based control.

Eliminate Symptoms

Maximum Burst Limitation (MBL) This has been proposed by Floyd and has been tested using the ns simulator. The proposal is to limit the size of bursts allowed on to the network by not allowing bursts greater than an imposed limit (recommended 4). The question of when a second burst of packets should be allowed on to the network is not addressed. If a burst of four packets is allowed for each acknowledgment received then ACK compression could result in large bursts of packets.

Congestion Window Monitoring (CWM) This is claimed to be analogous to a leaky bucket algorithm. It has the consequence that if sending opportunities are missed `cwnd` is shrunk. If `cwnd` is greater than the amount of outstanding data plus four packets it is reduced to outstanding data plus four packets. This aims to limit the burstiness of a TCP connection throughout its life and not just when recovering from idle. It does not address packet bursts caused by ACK compression.

Rate Based Pacing (RBP) The aim of rate based pacing is to obtain the throughput achievable with NSSR whilst preserving the lack of loss observable with SSR. RBP pacing is discussed in [VH97] and requires five implementation mechanisms.

- Idle Time Detection.
- An estimate of the available bandwidth.
- A limit on the number of packets that are to be sent in one burst.
- A mechanism to clock out the packets.
- A decision on when to stop RBP.

Idle time detection already exists in some implementations of TCP. An estimate of available bandwidth can be based on previous network conditions. Two approaches are outlined for bandwidth estimation, packet counting per RTT as used by Vegas TCP or using the current `cwnd` value scaled by some conservative factor. Packets can be clocked out by using a new timer. RBP stops when acknowledgments start to arrive from paced packets. From that point on TCP's normal acknowledgment based self-clocking takes over.

The results of simulations conducted suggest that RBP does produce the good initial throughput of NSSR with the resilience against loss of SSR. Padamanabhan [aRHK98] advocates using rate based pacing at the start of a connection to allow TCP to start with a larger initial window. If a burst larger than Maxburst is to be transmitted it is limited by the Maxburst parameter and

a new transmission is scheduled. This kind of pacing does not address bursts caused by ACK compression[9] consequently neither solutions provide a general solution.

General Solutions

In reference [BRS99] using *a rate-based traffic scheme to reduce bursts ... using a rate estimate that is the ratio of the window size to the smoothed round-trip time* is advocated. Optimizing TCP Start Up Performance [ZQK99] and [Rud98] both advocate using a variant of the leaky bucket algorithm to smooth traffic during the lifetime of a connection.

8.5.3 LIS and Explicit Traffic Shaping

Optimal

It is easy to translate from a rate to a window size and back again if the round-trip time is known.

$$Rate = \frac{WIN}{RTT} \tag{8.4}$$

This gives the optimal rate of transmission, which will not reduce throughput at larger time scales, and will result in an even spacing between each packet. There are however a number of factors which limit our ability to achieve this optimum; the next RTT can not be known a priori, TCP's timers are tuned for specific tasks, there is a limit on the granularity with which it is efficient to clock out packets, competing traffic may disrupt transmission.

RTT estimates The length of the next RTT can be estimated from previous traffic. Its actual value will however either be smaller or larger than the estimate. If the actual RTT is greater than the estimate packets will be bunched together more than is necessary. If the actual RTT is smaller than the estimate, a full windows worth of data will not be transmitted before the first packet in the window is acknowledged. This has the same effect as reducing the congestion window, and will lower throughput. Choosing an estimate must take into account the relative costs of an error in either direction.

TCP's current RTT estimation is used mainly as an input to the algorithm that calculates an RTO timeout. The cost of a false retransmission is higher than the cost of delaying a transmission. In addition timeouts are only a back up means of detecting loss and so an accurate SRTT estimate is not very important. Although it is important that it is not an underestimate. Consequently TCP's SRTT is tuned to avoid underestimates and generally overestimates the actual RTT. Consequently if we aim for an Inter Packet Departure Interval (IPDI) of:

$$IPDI = \frac{SRTT + MSS}{WIN} \tag{8.5}$$

the throughput the connection receives will be significantly restricted. It is not however necessary to further restrict throughput as congestion control operates effectively at the RTT time scale. An underestimate of the inter departure time will result in bunching of packets, but this currently occurs in TCP and so some bunching can be tolerated and the overall burstiness may be reduced. For these reasons we consider an underestimate of the RTT to be preferable to an overestimate. Possible choices include; using the minimum RTT, maintain a parallel SRTT or rework TCP RTT and RTO estimation to yield an accurate RTT estimate and a conservative RTO

[9] This is not surprising as it is intended only to pace out the first window of packets, and is switched off after the first ACK is received

Competing Traffic Once an IPDI has been calculated it may be disrupted for a number of reasons. The timeout may be locked out by a process operating at a higher or equal interrupt level. The local network may be busy, lock out is not uncommon in Ethernets. This results in a choice between allowing sending opportunities to accumulate, in which case bursts of traffic are likely, or allowing lost sending opportunities to be lost, which if the IPDI is optimal will result in a reduction in throughput.

Granularity of Transmission There is a limit in many systems on the granularity of timeouts. For many Unix's this is 10ms. If there is a hard limit on the size of burst allowed the throughput will be limited as a function if the transmission interval and RTT[10]. Let WIN be the amount of data transferred per RTT.

$$WIN = \frac{RTT}{Interval} * MAXBurst \tag{8.6}$$

For connections with small RTTs this is particularly nasty. Yet local transmissions are likely to be able to support the highest throughput and burstiness. If the maximum burst size is set to MAXBurst then larger bursts will be tolerated for smaller RTTs and throughput will not be limited by the smoothing process.

$$MAXBurst = \frac{WIN * Interval}{RTT} \tag{8.7}$$

For TCP to clock out packets a greater load will be placed on the host machine given the increased powers of today's processors this is not unreasonable. It does however raise issues as to whether todays operating systems are adequate to meet these new demands. Many Unix operating systems [MM96] provide a course scheduling facility. The high overhead of context switches leads to a desire to limit frequency. In addition under high load cross talk between applications using a common resource may delay scheduling.

For example a TCP window of 32000 bytes and RTT of 50 ms would have an upper limit on its throughput of 768000 bytes/sec. With a 1460 byte packet this would require scheduling every 1.9 m/s. This is a moderate sized window and RTT for larger windows and smaller RTT the scheduling requirement would be in the order of microseconds.

Work on the Nemesis [LMB+96] operating system addresses the issue of providing quality of service in a vertically structured single address space operating system. The ability to specify a fine grained share of the processor, to prevent cross talk between applications [Bar97, Bla95, BBDS97] and to multiplex data at once at the lowest possible level [Mca89] would all contribute to an efficient implementation of a TCP which supported rate based start-up.

8.5.4 Summary

Three causes of bursty traffic have been identified, the Slow Start algorithm itself, the Fast Recovery algorithm and ACK compression. These three symptoms are possible because a solely window based flow control method limits the rate of transfer of packets at a minimum granularity of one RTT. Within the RTT the interaction of congestion control algorithms, router queues and other factors can cause bursty traffic patterns.

When windows are small and each transfer takes many RTT then the size of these bursts is limited. Yet as bandwidths increase larger windows are needed to utilise them connections exist for less RTTs then the possible burstiness of the traffic increases. This leads to the proposition that window based flow control becomes less effective as bandwidth increases.

[10] In the absence of ACKs

Solutions, which are based upon changing the size of or limiting the amount of outstanding data in the congestion window, are trading off the utilisation of bandwidth against the burstiness of the traffic. An increase in the size of the initial window will allow greater throughput but produce burstier traffic. A smaller starting window size will produce smoother traffic but at the cost of lower network utilisation. At the heart of the dilemma is TCP's window scheme relying on the receipt of an ACK to trigger the sending of data. When there is a steady flow of data this is not as serious a problem but when the flow of packets breaks down another policy has to be grafted on to restart the ACK clock.

Rate Based Pacing does not suffer from these problems. It allows a graceful resumption of transmission without forcing a reduction in the utilisation of bandwidth over a full RTT.

It does however raise questions about the interaction between itself and the congestion window. If packets are clocked out at too slow a rate then when the first acknowledgment is received the congestion window will not be closed and a large burst of packets may be allowed onto the network. If the rate at which packets are transmitted is determined by TCP's current variables there is a danger that it will be miscalculated. In particular the SRTT is used inside TCP (along with the VRTT) to estimate when an RTO timeout and retransmission is to occur. Thus it is not tuned too achieve an accurate estimate of the actual network RTT and may be considerably larger than the real RTT if a few samples have been made.

Secondly a rate based policy places different demands upon the operating system the TCP implementation is located in. A far finer grained timing is required than either the 500ms slow timer or 200ms fast timer. These questions will be investigated further later in this paper.

8.6 Conclusion

This chapter surveys research related to best effort congestion control on the Internet. Much of this work takes as its starting point TCP Reno's Slow Start, Congestion Avoidance and associated algorithms and attempts to improve on them in a number of ways. Four strands of research were identified and discussed in separate sections.

The first strand accepts the constraints of connection confined adaptation and there being no explicit signal of congestion available from the network but aims to achieve, in Jain's terminology, congestion avoidance rather than congestion recovery. That is to allow the network to operate at the knee of the delay throughput curve. This direction of research has proved to be a dead end as the constraints adopted are too restrictive for the goal to be realised in practice, this is particularly so against the background of current Internet traffic patterns.

The second strand of research investigates modifications in routers, which would enable end points to improve their reaction to congestion and facilitate the control of misbehaving flows. Random Early Detection gateways explicitly drop packets to operate the network under the assumption that TCP will react to the loss as an implicit signal of congestion and reduce their rate of send. The aim is to control queue lengths and therefore operate the network nearer to an optimal level. The use of loss as the congestion signal precludes improvement on TCP's statistical fairness. The proposal for Explicit Congestion notification is a simple development of RED replacing packet loss with packet marking as the congestion signal, thereby changing the form but not the content of the congestion signal. This work is orthogonal to the LIS in that manipulation of the congestion signal by routers will be reflected in the congestion averages calculated by the LIS and therefore in the initial window sizes that end hosts derive from those averages.

The third strand of work addresses problems in TCP's start up behaviour; namely for long connections Slow Start‘ 1 tends to cause the peak start-up window size to overshoot the fair size as calculated by the TCP friendly equation and the tendency of initialisation to a small initial value to prevent short connections from utilising their fair share of bandwidth.

Connection confined approaches attempt to deduce a fair operating point from the first few packets

transfered and therefore intelligently switch from Slow Start to Congestion Avoidance before packet loss occurs. There is little evidence of significantly improved performance using these techniques.

The second approach is to share QoS feedback between consecutive and/or concurrent connections between a pair of hosts. Typically the congestion window at the start of a connection or session is used as the basis for initialising the congestion window at the start of a connection or session. The problem here is that the gap between sessions is often large enough for the validity of the last congestion window to be questioned. The sharing of congestion information between streams by a congestion manager allows application prioritisation of particular connections and prevents bandwidth grabbing by opening multiple concurrent connections. This work is complementary to the LIS the outputs of which could be used to initialise Congestion Manager control variables at the start of a session.

SPAND and NewSPAND facilitate the sharing of congestion information between aggregates of hosts, they differ from the LIS approach in that SPAND uses application metrics, where the LIS supplies network level information for interpretation by applications. This has the advantage of removing much of the noise that was found to limit SPAND's effectiveness. NewSPAND uses end host reporting rather than passive monitoring, this has the disadvantage of relying on end points for accurate estimation of network QoS thereby introducing an extra element source of error when compared to the LIS.

The use of a larger initial window has implications for the distribution of packets within a window as it undermines the use of Slow Start to establish TCP's ACK clock which in turn implicitly shapes the intra window packet distribution. As window sizes increase the reliance on implicit traffic shaping itself becomes more problematic as the maximum burst sizes caused by restarts after an idle period, ACK compression and Fast Retransmit increases. This has lead to research into combing rate and window based flow control to explicitly shape traffic within each window. Knowledge about likely maximum and minimum window sizes supplied by an LIS can be used to determine an appropriate rate. These areas of research are therefore complementary.

This chapter has surveyed four areas of work related to best effort congestion control. It has demonstrated that this dissertation addresses issues considered to be important by the the network research community and that it adopts a novel approach in so doing.

Chapter 9

Conclusion

9.1 Introduction

This chapter summarizes the argument that has been developed in the body of the dissertation, highlights the contributions that have been made and represents the thesis statement. Implications that flow from the findings in this dissertation for future work on the design of network protocols and congestion control are also discussed. The structure of this chapter mirrors the structure that has been adopted for the dissertation as a whole.

Firstly a number of background observations, drawn from the literature, which influenced the direction of the work are summarized. This is followed by highlighting the main conclusions drawn from the measurements of web traffic. The problem definition, which is drawn from the background material and measurements, is then rearticulated.

Next, an architecture which facilitates the temporal and geographic sharing of congestion information between hosts is presented. This is linked to the central thesis statement of the dissertation. The thesis statement was tested by designing and building building an implementation of a Location Information Server that is a realisation of the proposed architecture within the context of today's Internet.

Experiments using the LIS form the core of the dissertation's empirical evidence and were presented in Chapter 6. Here the main results are revisited and it is argued that these results validate the thesis statement. A limitation of the LIS's implementation however is that it only extracts information from TCP traffic. Different approaches, which would enable the LIS approach to be applied to more traffic have been evaluated and discussed here the main conclusions are presented.

The chapter is concluded with a restatement of the main finding of the dissertation.

9.2 Background

The Internet was designed to be resilient against the threat of external aggression. The facilitation of traffic management and the control of congestion were not seen as being priorities and are not well supported by the Internet's Datagram architecture.

Within a few years of TCP/IP being deployed however, congestion collapse became a serious problem. This lead to the introduction of Van Jacobson's Slow Start and Congestion Avoidance algorithms into TCP. Fortunately most IP traffic was carried by the connection oriented Transmission Control Protocol and these measures successfully prevented congestion collapse. There are however important weaknesses in these measures in that to be effective they rely upon the assumption that most traffic will be carried by TCP. Furthermore, for bandwidth utilisation to

be efficient it is assumed that the steady state behaviour of the Congestion Avoidance algorithm will dominate over the transient start up behaviour of Slow Start.

9.3 Web Measurements

When congestion control was introduced to TCP, most transfers could be characterised as being either low bandwidth interactive traffic generated by telnet sessions or high bandwidth bulk transfers typical of the File Transfer Protocol. The growth of the World Wide Web invalidated this characterisation.

Today most traffic is generated by Web browsers and servers. A typical web transfer is interactive, thereby placing a premium on low latency, and is relatively short lived. Consequently many connections have insufficient time to adapt to network conditions and bandwidth utilisation is often determined by the size of the transfer rather than network conditions. Long lived bulk transfers receive a better service from the network than short lived interactive connections. These observations are born out by simulations and measurements of Web traffic presented in Chapter 4. A high proportion of Internet traffic continue to be carried by the Transmission Control Protocol. Of this traffic a large proportion of connections are too small to interact with TCP's congestion control mechanisms. Of those connections which do interact the vast majority do not reach steady state behaviour.

Two further observations were made; the average level of congestion varies widely by route, suggesting that any single static starting size for the congestion window will be to aggressive for a significant proportion of connections or too small to allow bandwidth to be utilised effectively. This further suggests the need for the initial window size to be determined dynamically. Secondly, the gap between connections is often too large for a saved congestion window which is decayed exponentially each RTT of inactivity to be usefully used as a starting point for new connections.

Although most connections are too small to adapt to network conditions the majority of bytes are carried in large connections. This suggests that whilst there may be benefit in changing the start up behaviour of TCP the Internet is likely to be resilient to such changes. Furthermore a small proportion of destinations account for a high proportion of all connections further suggesting that using information from past traffic is a viable proposition.

9.4 Problem Definition

The binary nature of the congestion feedback from the network means that it is necessary for several rounds of data to be transfered before bandwidth utilisation stabilises around a fair value. Proposals to introduce Random Early Detection Gateways and Explicit Congestion Notification, whilst improving congestion control in some respects do not address this problem. The current Internet draft describing a Congestion Manager does not take advantage of information sharing between hosts and severely limits the effectiveness with which past traffic can be utilised.

Extending the scope of congestion control beyond a single connection would allow more time for adaptation to network conditions. The usual approach of taking TCP's congestion control algorithms as the starting point for determining how long congestion information should remain valid is however both questionable in logic and poor in results. It is questionable in logic, because it is not the case that most connections achieve steady state behaviour and are governed by multiplicative decrease and additative increase algorithms. The time that congestion information remains valid is likely to depend upon the statistical properties of the number of connections that are starting and ending.

The results are poor because the logic of starting from TCP's AIMD algorithms, leads to the conclusion that the congestion window size should be decreased exponentially for each RTT that

there is no activity. The gap between HTML pages is orders of magnitude larger than a RTT and therefore congestion control is only effectively extended over the connections that make up a single page.

What is required is for the network to supply to the host at the start of the connection sufficient information about the state of the network for the host to derive a far starting window or rate. The desire not to consume scarce resources in the core of the network means that for the foreseeable future the congestion signal is likely to remain binary in nature. Therefore to achieve the above goal it is necessary to deploy resources in the local area network that convert the binary signal into a congestion estimate and make this estimate available to hosts in a timely manner.

The novel approach proposed in this dissertation was to calculate an average level of congestion and communicate this to the host. The host could then use a TCP fair equation to calculate a fair initial window. By calculating an average over a larger time span it was hoped that the congestion information would have a longer lifespan.

9.5 Architecture

A number of considerations lead to the design decision to locate the information server in a separate machine. Firstly, this could be facilitated because the aim was to provide information for initialisation at the start of a connection and ongoing communication during a connection's lifetime was unnecessary. Whilst a connection was active a continual flow of binary congestion information would reach the host in the usual way. Secondly, locating the LIS on the WAN LAN interface would allow it to leverage the locality of reference for web connections. Information extracted from communication with one host could be used in deriving estimates for a nearby host.

The architecture proposed involves the addition of an entity called a Location Information Server. It is the job of the LIS to monitor the congestion signal for different Internet paths and maintain a congestion estimate for aggregates of destinations. The LIS takes advantage of the connection set up time to communicate this information to local hosts. The local hosts then makes use of equation based congestion control to convert the congestion information to a fair starting rate or window size to provide a starting point for the transfer. Once the transfer is underway congestion feedback is received in the normal way and the connection can thus continue to adapt to network conditions.

There is therefore a division of labour, longer term statistics are used to determine a connection's starting point, immediate feedback is used to adapt from that starting point. In this way relatively long lived congestion information can be utilised without compromising the connections ability to adapt to short term congestion problems.

9.6 Implementation

In order to evaluate the above architecture a realisation of a Location Information Server for the Internet was built. The absence of a generic congestion signaling mechanism on the Internet meant that this realisation focused upon the monitoring of TCP traffic to determine congestion levels.

Macro and micro state was maintained for each TCP connection. Macro state enabled connection set up, tear down and the detection of new connections. Micro state enabled readings of the instantaneous window size, round trip time to be taken and the presence of an Implicit Congestion Notification to be detected.

For each aggregate of destinations statistics for the maximum recent window size smooth round trip time and the proportion of ICNs were maintained. Upon the detection of a new connection these statistics were communicated to the local host using a Location Information Packet.

The local host used the information in the LIP to determine a conservative fair window size. It is envisaged that this information could also be utilised by applications in a number of ways.

These design decisions were tested by building an implementation of a Location Information Server, by modifying the FreeBSD TCP/IP network stack so that it could utilise information provided by the LIS and analysing the traces using custom built software.

We have demonstrated the design and implementation of a Location Information Server, which was capable of processing all TCP traffic with the University of Glasgow in real time. The platform used was a P133 with 64 megabytes of memory.

A design for the extension of the LIS to extract information from RTP data streams and a multimedia conferencing application that utilises the LIS to configure streams so appropriately use network resources has also been presented.

9.7 Comparison of LIS and static initialisation

The performance of connections which utilised information provided by an LIS to initialise `cwnd` and `ssthresh` have been compared, with the performance of those that use the traditional static initialisation, against a number of metrics.

It has been shown that the methods of dynamic initialisation deployed do not result in start up behaviour that is more aggressive than the normal steady state behaviour of TCP. Furthermore the utilisation of dynamic initialisation does not lead to an increase in congestion feedback, suggesting that these techniques do not lead to more congestion.

When the correlation between average window size and fair window size is considered for connections longer than 32 segments there is little difference between static and dynamic initialisation. This shows that changing the start up behaviour has little effect on the long term dynamics of TCP connections. This is significant, because although the majority of connections are small in size most packets are carried in the few long connections. Most connections are significant to a user, as they map to individual interface elements. For the stability of the network the load measured in packets is more important.

When the average to fair window correlation is considered for shorter connections it is evident that using dynamic initialisation results in a much stronger correlation. This reflects the fact that dynamic initialisation increases the responsiveness of network traffic to congestion notifications. The number of rounds to complete a connection has been used as a metric to estimate the difference in the delay suffered by users and it has been shown that application of the LIS will significantly reduce the latency experienced by a high proportion of users.

9.8 Lessons for the future

Neither the connection oriented nor the datagram model fit the traffic patterns on the Internet. It has been argued that the separation of congestion signaling and the policies to be applied upon the detection of congestion is necessary to support the flexibility that is required in a dynamically changing environment such as the Internet.

Furthermore as congestion is a property of the network layer, congestion signaling should be independent of higher protocol layers and transparent to intermediate nodes.

A central design goal of congestion schemes is to minimise the resources that they use. This is particularly important at a time of congestion when such resources are in short supply and the work required by the congestion control scheme is likely to increase. This leads to the observation that congestion signaling on the Internet is likely to be binary in nature.

It can no longer be safely assumed that the start up phase of a connection can be ignored in

determining average behaviour. There is therefore a need for hosts to be supplied with an explicit estimate of congestion at the start of a transfer. It has been shown that a Location Information Server can fulfill this function for TCP. For the approach to be extended so that information about network paths can be extracted from non TCP traffic, it is necessary to either include knowledge about each congestion signaling protocol in the LIS or for a common signaling mechanism to be developed and deployed.

9.8.1 Combing the LIS with Rate and Window based Flow Control

The setting of control paramaters to determine an initial size for the congestion window has been explored. However there are other control paramaters that are important, for example the RTT. Future work might also look at the issue of pacing out traffic. It was suggested in Chapter 4 that window based flow control allows large bursts of traffic when large windows are used. It would be usefull to be able to pace out at least the first window of data this involvees combining rate and window based flow control with predictions of fair windnow size and RTT gathered by the LIS.

9.9 Future Work

In this section five areas for future work are identified. The first two are a natural extension of the breadth of the work. The first identifies ways in which the evaluation conducted in Chapter 6 could be strengthened. The second area involves explaining why it will be adventageous to continuing to measure Internet traffic. The other three areas involve developing areas of work related to the central thrust of the dissertation. It is argued that development and experimental work based upon the architeture presented in Chapter 7, that investigating how the LIS could copoperate with some host based mechanism for sharing congestion information and devel 1 oping aproaches for aggregation of traffic statistics will all be fruitful areas of study.

9.9.1 Broadening the Scope of the Experimental Work

Support for the thesis of this dissertation would be strengthened by broadening the experimental work presented in Chapter 6. It would be interesting to see the results for transfers from a number of different sources. It would be desirable to use sources from a spread of geographical locations, perhaps situated in the East and West Coast of America, Europe, Japan, Latin America, Eastern Europe and Korea. A second dimension worthy of investigation is the results that would be obtained using sources in different domains. For example, military, educational, business and government. A comparison of intra and extra domain traffic could be carried out. It is anticipated that doing so would strengthen the empirical evidence for the arguments presented in this dissertation.

The analysis of TCP presented in Chapter 4 could be strengthened by relaxing the assumption of deterministic loss. For example by running simulations in a simulator such as ns, where it is possible to set a number of parameters, such as the network topology, buffer space in routers and the model used to generate traffic.

9.9.2 Developing a Passive Monitoring Measurement Infrastructure

The Internet is a developing entity [Bar64a, LCC+97, O'N90]. Three factors may be identified which stimulate change in the nature of the Internet and therefore of its traffic and the demands made upon traffic management mechanisms.

1. As technology advances changes in the basic infrastructure occur on two fronts [Cer93]. The bandwidth available in the core of the network has the potential to increase. The rate at

which data is manipulated and created at the edge also increases as processors become faster and memory cheaper.

2. As society develops the relationship of the Internet to that society changes. For example the move from a primarily military [Cla88] to an academic [Kah94] to a commercial network [Cla95]. The impact of the problems faced by dot coms on internet infrastructure, the tension between the need for a coordinated information infrastructure and the desire to stimulate competition.

3. People think up new ways of using the technology, for example the growth of peer to peer network and the distribution of mpeg music files across the Internet

All these factors mean that there is a strong case for continuing to measure Internet traffic so that the infrastructure can be managed [FRG97] in such a way as to best satisfy the needs of the users. This in turn implies the need to develop a generic Internet measurement infrastructure [Pax97]. Passive monitoring techniques will continue to have an important role to play in this effort [Pax96b].

9.9.3 Sharing congestion information between TCP and RTP data streams

As the processing power of end hosts increases it becomes more practical for them to be able to manipulate multimedia data in realtime. As the bandwidth at the core of the Internet also increases the possibility of it routinely carrying realtime multimedia traffic arises [LLSZ96].

For these reasons there is an increasing amount of real time data being carried on the Internet and significant interests in congestion control schemes which meet the needs of these data types [Cha94, DKS95, CO98, MPCC00, KMBL99] whilst maintaining some notion of fair resource allocation between TCP and non TCP data.

It would therefore be desirable for the LIS to be able to:

1. Extract congestion information from real time as well as TCP traffic.

2. To be able to provide real time applications with information that enabled them to more efficiently make use of the bandwidth that was available to them.

An approach to achieving these ends and a discussion of its practicality within the context of RTP traffic was presented in Chapter 7. Future work in this area could involve implementing these extensions to the LIS. In particular to develop a third party monitor for RTP traffic and to integrate the data collected with that collected from TCP data streams.

9.9.4 Combining the LIS with the Congestion Manager

There has been significant research into the sharing of the feedback a connection receives about the state of the network between concurrent and consecutive connections. The approaches adopted vary in sophistication between simply storing values in the routing table so that other connections to the same host can use them for initialisation [Ste96], to sharing information between concurrent TCP connections [Lar99] to the development of a congestion manager [BRS99, BS01] which is protocol independent, can provide relevant information to applications and can play a policing role within a host.

These developments can be seen as complementary to the LIS approach in that the LIS is well placed to facilitate the sharing of congestion information over longer timescales and between servers that are situated at a common location.

So for example proposals for a congestion manager advocate a common congestion window that is shared between all connections to a common destination. The evolution of the congestion window is controlled by AIMD algorithms based upon the Congestion Avoidance algorithm. In the absence of traffic the common congestion window is decayed exponentially. Thus after a few RTTs in the absence of traffic the information held by the congestion manager on the path to a location will be lost.

Due to the LIS's calculation of longer term averages the information it holds can be expected to be valid for longer periods of time. Congestion information from an LIS could be supplied to the Congestion Manager at the start of a session and used to initialise the Congestion Managers control variables.

As the LIS is capable of aggregating together information from a number of hosts, in the case where there is a cluster of web servers information collected from one web server could be used to initialise a congestion manager in another web server.

9.9.5 Automated Learning and Traffic Management

In the measurements presented in Chapter 4 and Chapter 5 hosts are considered to be situated in the same location if they share the same values for the three higher order bytes in their IP address. Future work could fruitfully be done in investigating other methods for aggregation.

There may be mileage in aggregating hosts by the network part of their IP address. With classless addressing it is no longer the case that a hosts network address can be determined simply from its IP Address. However it may be possible to use information held in the routing tables of routers to determine network membership and to in turn use this information to aggregate hosts into groups.

It may be possible to use statistical methods to detect patterns in the congestion er term averages the Information it holds can be expected to be valid for longer periods of time. Congestion information from an LIS could be supplied to the Congestion Manager at the start of a session and used to initialise the Congestion Managers control variables.

As the LIS is capable of aggregating together information from a number of hosts, in the case where there is a cluster of web servers information collected from one web server could be used to initialise a congestion manager in another web server.

9.9.6 Automated Learning and Traffic Management

In the measurements presented in Chapter 4 and Chapter 5 hosts are considered to be situated in the same location if they share the same values for the three higher order bytes in their IP address. Future work could fruitfully be done in investigating other methods for aggregation.

There may be mileage in aggregating hosts by the network part of their IP address. With classless addressing [Ste94] it is no longer the case that a hosts network address can be determined simply from its IP Address. However it may be possible to use information held in the routing tables of routers to determine network membership and to in turn use this information to aggregate hosts into groups.

It may be possible to use statistical methods to detect patterns in the congestion and other feedback received by hosts [RKT99, RKT00b]. Using these methods it may be possible for an LIS to dynamically aggregate hosts together into groups that can share congestion feedback information.

A further area open for investigation is the possibility of a layered hierarchy of aggregation. At the top level information from all destinations is aggregated together. This may be useful in determining conditions on the link between the local and wide area network. At the next level down statistical methods are used to group hosts into loose aggregates. These might correspond to quite wide geographical domains, such as Eastern Europe, America East Coast, America West Coast etc which could in turn be determined by, the diurnal pattern of traffic (giving a fix on time

zone), RTT giving a fix on distance and congestion. At the next level down network membership is used to share congestion between closely located hosts. At the lowest level IP addresses may be used to identify individual addresses.

Each layer could be thought of as corresponding to a different timescale and a different quality of information. At the start of a flow and in the absence of recent information about a host at a lower layer and timescale, the next layer up could be queried and used to initialise the lower layer.

9.10 Conclusion

The thesis of this dissertation has been validated on a number of fronts.

Simulation has been used to determine the range of congestion levels and transfer sizes that TCP congestion control mechanism operate as intended. It has been shown that there are important regions where they significantly and unnecessarily restrict access to the network and there is also a significant region where overly aggressive behaviour can be expected. Measurements of web traffic have been analysed and it has been shown that a large proportion of traffic falls into the region where access is unnecessarily restricted and that flow control limitation currently prevents the overly aggressive behaviour from becoming a serious problem.

It has also been shown that the increase in delay attributable to static initialisation has a noticeable effect on reducing the QoS experienced by users. Given the aggregate amount of time that is spent browsing the web it follows that this is an important problem area. An analysis of how this state of affairs came about which includes a survey of early congestion control schemes and analysis of the relationship between the design priorities for the Internet and the form that Internet congestion control took has been presented.

A significant development effort was successfully undertaken to build a Location Information Survey and an end host which can utilise the information provided by an LIS. It has been shown using live experimental traffic to a number of hosts that the use of an LIS has a number of beneficial effects.

The negative correlation between average window sizes and the level of congestion is significantly strengthened for a range of smaller connection sizes. The number of round trip times it takes to complete a connection is reduced by a ratio of up to 5:1 for certain connection sizes on un congested routes. The peak window size during start up is not increased indicating that overall the proposed changes do not make TCP's start up more aggressive. In short it has been shown that the use of an LIS increases the responsiveness to congestion and enhances the service provided to the user.

The application of best effort session level congestion control to real time traffic has been considered through an analysis of RTP. It has been shown that the extraction of QoS feedback is possible as is its integration with feedback collected from TCP traffic. The design of a group conferencing based application that will make use of such feedback has been presented.

It has further been shown by surveying concurrent and previous work that whilst the approach outlined in this dissertation is related to other work in the field it differs in significant ways. The clear identification of the session time-scale and the measurement of network characteristics centralised in the LIS being important examples.

The thesis statement has been formulated through an extensive reading and analysis of historical and contemporary literature in the field. The existence of the problem identified has been demonstrated through simulation of TCP congestion control algorithms and measurement of Internet traffic. It has been demonstrated using visualisation techniques that are innovative for the data presented. The architecture adopted has been validated by building and evaluating server and host implementations. The implementations themselves have been evaluated using experimental traffic, which has been systematically analysed and confirms the thesis of this dissertation.

Bibliography

[ABR01] C. Allison, M. Bateman, and A. Ruddle. Realising real time multimedia groupware on the web. In *6th Eurographics Workshop on Multimedia*, 2001.

[AD98] Mohit Aron and Peter Druschel. TCP: Improving startup dynamics by adaptive timers and congestion control. Technical Report TR98-318, Rice University Computer Science, 1998.

[AFP98a] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's Initial Window. *Internet Draft draft-floyd-incr-win-03.txt*, 1998.

[AFP98b] M. Allman, S. Floyd, and C. Partridge. RFC 2414: Increasing TCP's Initial Window, September 1998. Status: EXPERIMENTAL.

[AHKO97] Mark Allman, Chris Hayes, Hans Kruse, and Shawn Ostermann. TCP performance over satellite links. *Proceedings 5th International Conference on Telecommunication Systems Ohio University*, 1997.

[AHO98] Mark Allman, Chris Hayes, and Shawn Ostermann. An evaluation of TCP Slow Start with larger Initial Windows. *Computer Communications Review*, July 1998.

[All97] Mark Allman. Improving TCP performance over satellite channels. Master's thesis, College of Engineering and Technology Ohio University, JUN 1997.

[All99] Mark Allman. On the effective evaluation of TCP. *Computer Communication Review*, 29(3), July 1999.

[All00] Mark Allman. TCP byte counting refinements. *Computer Communications Review*, 29(3), 2000.

[ALMR00] C. Allison, H. Lawson, D. McKechan, and A. Ruddle. Quality of Service Issues in Distributed Learning Environments. In *Proceedings of Euromicro 2000*, pages 47–53, Maastricht, 2000. IEEE Press ISBN 0769507808.

[ALMR01] C. Allison, H. Lawson, D. MCkechan, and A. Ruddle. An holistic approach to Quality of Service (QoS). *Interactive Learning Environments*, 9(1), 2001. ISSN 1049-4820.

[aMR81] Simon S. Lam amd Martin Reiser. Input buffer limits an analysis. *IEEE Transactions on Communications*, COM-27(1):127, January 1981.

[AP99] Mark Allman and Vern Paxson. On estimating end-to-end network path properties. In *Proceedings of ACM SIGCOMM 99*, 1999.

[aRHK98] Venkata N. Padmanabhan annd Randy H. Katz. TCP Fast Start: A technique for speeding up web transfers. In *Proceedings IEEE Globecom '98 Internet Mini-Conference*, November 1998.

[ARML01] C. Allison, A. Ruddle, D. McKechan, and P. Lindsay. The architecture of a framework for building Distributed Learning Environments. In *IEEE International Conference of Advanced Learning Technologies (ICALT 2001)*, Madison Wisconsin USA, aug 2001. IEEE.

[ARMM01] C. Allison, A. Ruddle, D. McKechan, and R. Michaelson. A group based system for group based learning. In *European Pespectives on Computer Supported Collaborative Learning the proceedings of Euro CSCL 2001*, pages 43–50, Maastricht, 2001. CSCL, Maastricht McLuhan Institute ISBN 90-5681-097-9.

[AS96] Audio-Video Transport Working Group and H. Schulzrinne. RFC 1890: RTP profile for audio and video conferences with minimal control, January 1996. Status: PROPOSED STANDARD.

[aSATVF93] William H. Press anad Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical Recipes in C: The Art of Scientific Compmuting Second Edition.* Cambridge University Press, 1993.

[ASC+96] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889: RTP: A transport protocol for real-time applications, January 1996. Status: PROPOSED STANDARD.

[Axe85] Robert Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1985.

[BaDMP95] P. Barham, M. Hayter abd D. McAuley, and I. Pratt. Devices on the Desk Area Network. *IEEE Journal on Selected Areas in Communication*, 1995.

[Bar64a] Paul Baran. On distributed communication. Technical report, RAND Corporation, 1964.

[Bar64b] Paul Baran. On distributed communication:iv. priority, precedence and overload. Technical Report RM-3638-PR, RAND Corporation, August 1964.

[Bar64c] Paul Baran. On distributed communications networks. *IEEE Transactions on Communications Systems*, pages 1–9, March 1964.

[Bar97] P. R. Barham. A fresh approach to file system quality of service. In *Proceedings of NOSDAV 1997*, April 1997.

[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475: An architecture for differentiated services, December 1998. Status: PROPOSED STANDARD.

[BBDS97] Richard Black, Paul Barham, Austin Donnelly, and Neil Stratford. Protocol implementation in a vertically structured operating system. *Proceedings IEEE LCN 1997*, November 1997.

[BBK00] A. Bouch, N. Bhatti, and A. J. Kuchinsky. Quality is in the eye of the beholder: Meeting users' requirements for Internet Quality of Service. In *CHI'2000*, 2000.

[BCC+98] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. RFC 2309: Recommendations on queue management and congestion avoidance in the Internet, April 1998. Status: INFORMATIONAL.

[BG85] Werner Bux and Davide Grillo. Flow control in Local Area Networks of interconnected Token Rings. *IEEE Transactions on Communications*, COM-33(10):1058–1065, 1985.

[Bla95] Richard John Black. *Explicit Network Scheduling.* PhD thesis, University of Cambridge Computer Laboratory, 1995. TR 361.

[BOP94] Lawrence S. Brakmo, Sean W O'Malley, and Larry L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings of ACM SIGCOMM '94*, 1994.

[BPS+98] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, Mark Stemm, and Randy H. Katz. TCP behavior of a busy Internet server: Analysis and improvements. In *Proceedings of IEEE Infocmm*, March 1998.

[Bra97] S. Bradner. RFC 2119: Key words for use in RFCs to indicate requirement levels, March 1997.

[BRA01] M. Bateman, A. Ruddle, and C. Allison. Using the Java Media Framework to build adaptive multimedia groupware applications. In *Proceedigs of EPSRC PGNET 2001*, pages 258–263, Liverpool, oct 2001. 2nd Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting.

[BRS99] Hari Balakrishnan, Hariharan S. Rahul, and Srinivasan Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *Proceedings of ACM SIGCOMM 99*, 1999.

[BS99] A. Bouch and M. A. Sasse. Network Quality of Service: What do users need? In *4th International Distributed Conference*, pages 78–90, Madrid, 1999.

[BS00] A. Bouch and M. A. Sasse. The case for predicatable network service. In *Proc MMCN'2000*, pages 188–195, San Jose, January 2000.

[BS01] H. Balakrishnan and S. Seshan. The congestion manager. *Internet Draft draft-ietf-ecm-cm-04.txt*, 2001.

[BSSK97] Hari Balakrishnan, Srinivasan Seshan, Mark Stemm, and Randy H. Katz. Analyzing stability in wide-area network performance. In *Proceedings of SIGMETRICS Conference on Measurement and Modeling of Computer Systems, June 1997*, June 1997.

[BSTM80] David R. Boggs, John F. Shoch, Edward A. Taft, and Robert M. Metcalf. PUP an internetwork architecture. *IEEE Transactions on Communications*, 1980.

[BSW69] K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM*, 12(5), May 1969.

[BZB+97] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin. RFC 2205: Resource ReSerVation Protocol (RSVP) — version 1 functional specification, September 1997. Status: PROPOSED STANDARD.

[Cac92] Ramon Caceres. *Multiplexing Traffic at the Entrance to Wide-Area Networks.* PhD thesis, University of California, 1992.

[CaGP93] K. Claffy and H. Braun amd George Polyzos. Long term aspects of the NSFNET. Technical report, Computer Systems Laboratory, University of California, 1993.

[CAI01] CAIDA. Internet traffic workload. http://www.caida.org/outreach/resources/learn/trafficworkloa 2001.

[CB97] Mark E. Crovella and Azer Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. In *IEEE/ACM Transactions on Networking*, pages 835–846, 1997.

[CBP94] Kimberly C. Claffy, Hans-Werner Braun, and George C. Polyzos. Tracking long term growth of the NSFNET. *Communications of the ACM*, 17(8):34–45, August 1994.

[CCJ95] A. Charny, D. Clark, and R. Jain. Congestion Control with Explicit Rate Indication. In *Proceedings ICC'95*, pages 1954–1963, June 1995. charny.ps.

[Cer80a] Vinton Cerf. Protocols for interconnected packet networks. *Computer Communication Review*, 1980.

[Cer80b] Vinton G. Cerf. Final report of the stanford university TCP project. IEN 151, April 1980.

[Cer93] Vinton Cerf. How the Internet came to be. In Bernard Aboda, editor, *The Online Users Encyclopedia.* Addison Wesley, 1993.

[Cha94] Anna Charny. An algorithm for rate allocation in a packet switching network with feedback. Master's thesis, Massachusetts Institute of Technology, May 1994.

[CJ89] D.M. Chiu and R. Jain. Analysis of increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[CK74] Vinton G. Cerf and Robert E. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 1974.

[CKLT97] Chris Chen, Hariharan Krishnan, Steven Leung, and Nelson Tang. Implementing Explicit Congestion Notification (ECN) in TCP for ipv6. UCLA, December 1997.

[CKW97] Costat Courcoubetis, Frank Kelly, and Richard Weber. Measurement-based charging in communications networks. Technical report, University of Cambridge Statistical Laboratory, 1997.

[Cla82] D. D. Clark. RFC 813: Window and acknowledgement strategy in TCP, July 1982. Status: UNKNOWN.

[Cla88] David D. Clark. The design philosophy of the DARPA Internet. In *Proceedings of ACM SIGCOMM '98*, 1988.

[Cla95] David D. Clark. Adding service discrimination to the internet. Technical report, MIT Laboratory for Computer Science, September 1995. Version 2.

[CMN83] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale New Jersey, 1983.

[CO98] Jon Crowcroft and Philipe Oechslin. Differentiated end to end Internet services using a weighted proportional fair sharing TCP. Technical report, University College of London, April 1998.

[Coh78] Dan Cohen. On names, addresses and routing (ii). IEN 31, April 1978.

[Coh80] Danny Cohen. Flow control for real time communication. *Computer Communication Review*, 10:41–47, January 1980.

[Col90] Robert Cole. Experience and analysis of network interconnection. *IEEE Journal on Selected Areas of Communication*, January 1990.

[Com00] Douglas E Comer. *Internetworking with TCP/IP Principles, Protocols and Architecture*, volume 1. Prentice Hall, 4 edition, 2000.

[CP00] J. Crowcroft and C. S. Perkins. Notes on the use of RTP for shared workspace applications. *Computer Communications Review*, 30(2), April 2000.

[CPNK70] S. D. Crocker, J. Postel, J. Newkirk, and M. Kraley. RFC 54: Official protocol proffering, June 1970. Updated by RFC0057. Status: UNKNOWN.

[CPR78] David D. Clark, Kenneth Pogran, and David P. Reed. An introduction to local area networks. *Proceedings of the IEEE*, 1978.

[cR81] Eric c. Rosen. Issues in internetting: Part 2: Accessing the internet. Technical report, Bolt Beranek and Newman Inc., June 1981.

[Cro69] Steve Crocker. RFC 1: Host software, April 1969. Status: UNKNOWN.

[CSA88] Neal Cardwell, Stefan Savage, and Tom Anderson. Modeling the performance of short TCP connections. Technical report, Department of Computer Science and Engineering University of Washington, October 1988.

[CT90] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 200–208. IEEE, 1990.

[CVW+98] J. Crowcroft, L. Vivisano, Z. Wang, A. Ghosh, M. Fuchs, C. Diot, and T. Turletti. RMFP: A reliable multicast framing protocol. Internet Draft, March 1998. Work in progress.

[Dav72] Donald Watts Davies. The control of congestion in packet switched networks. *IEEE Transaction on Communications*, COM-20(3):546–550, June 1972.

[DBF+97] B. DeCleene, S. Bhattacharaya, T. Friedman, M. Keaton, J. Kurose, D. Rubenstein, and D. Towsley. Reliable Multicast Framework (RMF). A white paper, 1997.

[Dee88] Stephen E. Deering. Multicast routing in internetworks and extended LANS. In *Proceedings of SIGCOMM '88*, 1988.

[Dha80] C. Retna Dhas. Networks and flow control. *Computer Communication Review*, 1980. SNA.

[DKS95] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings of ACM SIGCOMM '95*, 1995.

[DLY95] Peter B. Danzig, Zhen Liu, and Limin Yan. An evaluation of TCP Vegas by live emulation. In *Proceeedings of ACM SIGMetrics '95*, 1995.

[Edg83] Stephen William Edge. An adaptive timeout algorithm for retransmission across a packet switching network. In *SIGCOMM '84*, 1983.

[Eli79] M. Elie. Traffic control by latency in a packet switching network. In Jean-Louis Grange and Michel Gien, editors, *Flow Control in Computer Networks: Proceedings of the International Symposium on Flow Control in Computer Networks*, pages 89–103. North Holland, 1979. Versailles.

[FF96] Kevin Fall and Sally Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. Technical report, Lawrence Berkeley National Laboratory, 1996.

[FF97] Sally Floyd and Kevin Fall. Router mechanism to support end-to-end congestion control. Network Research Group LBNL Berkeley CA, February 1997.

[FF98] Sally Floyd and Kevin Fall. ECN implementations in the ns simulator. LBNL, October 1998.

[FF99] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the internet. To appear in IEEE/ACM Transactions on Networking, May 1999.

[FGM+97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. RFC 2068: Hypertext Transfer Protocol — HTTP/1.1, January 1997. Status: PROPOSED STANDARD.

[FH99] S. Floyd and T. Henderson. RFC 2582 the New Reno modifications to TCP's fast recovery algorithm, April 1999.

[FJ93a] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *ACM/IEEE Transactions on Networking*, 3(4):397–423, August 1993. 1993.

[FJ93b] S. Floyd and V. Jacobson. The synchronisation of periodic routing messages. In *Proceedings of ACM SIGComm*, pages 33–44, San Francisco, September 1993. ACM.

[FJL+97] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, December 1997.

[FJM95] Sally Floyd, Van Jacobson, and Steven McCanne. A reliable framework for Light Weight Sessions and Application Level Framing. In *Proceedings of SIGCOMM*, pages 342–356, Boston Mass, September 1995. ACM.

[Flo95] Sally Floyd. TCP and Explicit Congestion Notification. *ACM Computer communications Review Vol. 24 Num. 5 October 199*, 1995.

[Flo96] Sally Floyd. Comments on measurement-based admissions control for controlled-load services. Draft, Lawrence Berkeley National Laboratory, July 9, 1996 Incomplete, July 1996.

[Flo01] Sally Floyd. Less conservative congestion control. http://www.aciri.org/floyd/tcp_unfriendly.html, 2001.

[FRG97] Will A. Foster, Anthiny M. Rutowski, and Seymour E. Goodman. Who governs the internet. *Communications of the ACM*, 40(8), 1997.

[GC99] Vijay Gupta and Roy Campbell. Reliable multicast of bulk data: congestion, fairness and freshness issues. Technical report, University of Illinois, 1999.

[GCMW99] Rajarshi Gupta, Mike Chen, Steven McCanne, and Jean Walrand. Webtp: A receiver-driven web transport protocol. Technical report, University of California Berkeley, 1999.

[Gev98] Panos Gevros. Experimental results on weighted proportional TCP throughput differentiation. In *Hipparch '98 Workshop Program*, June 1998.

[Gie79] Alfred Giessler. Throughput guarantee in x.25 networks. In Jean-Louis Grange and Michel Gien, editors, *Flow Control in Computer Networks: Proceedings of the International Symposium on Flow Control in Computer Networks*, pages 89–103. North Holland, 1979. Versailles.

[GJMH81] Alfred Giessler, Annemarie Jagemann, Ellen Maser, and Jurgen O Hanle. Flow control based on buffer classes. *IEEE Transactions on Communications*, COM-29(4):436–437, April 1981.

[GK80] Mario Gerla and Leonard Kleinrock. Flow control a comparative survey. *IEEE Transactions on Communications*, 28(4), April 1980.

[GK98] R.J. Gibbens and Frank Kelly. Resource pricing and the evolution of congestion control. Statistics laboratory, University of Cambridge, Preliminary Draft, 1998.

[Gre89] D.J. Greaves. *Multi Access Metropolitan Networks.* PhD thesis, Cambridge University Computer Laboratory, 1989.

[Gro85] Daniel Grossman. Comments on "Congestion control in IP/TCP internetworks". *Computer Communication Review*, 15(2), April 1985.

[GS99] S. Jamaloddin Golestani and Krishnan K. Sabnani. Fundamental observations on multicast congestion control in the internet. In *Proceedings of INFOCOM 99*, March 1999.

[GT95] Matthais Grossglauser and David Tse. A framework for robust measurement-based admission control. In *Proceedings ACM SIGCOMM '95.* University of California and INRIA, June 1995.

[Gur81] Robert F. Gurwitz. VAX-UNIX networking support project: Implementation description. Technical Report IEN 168, Bolt Beranek and Newman, Inc, January 1981.

[Hei97] John Heideman. Performance interactions between P-HTTP and TCP implementations. *ACM Computer Communications Review 27 April 1997*, 1997.

[HF] Mark Handley and Sally Floyd. Strawman specification for TCPfriendly (reliable) multicast congestion control (TFMCC. Reliable multicast Research Group.

[HHS83] Robert Hinden, Jack Haverty, and Alan Sheltzer. The DARPA Internet: Interconnecting heterogeneous computer networks with gateways. *Computer*, pages 38–48, September 1983.

[Hin81] Robert Hinden. Design of TCP/IP for the TAC. Technical Report IEN 166, Bolt Beranek and Newman, January 1981.

[HJ98] M. Handley and V. Jacobson. RFC 2327: SDP: Session Description Protocol, April 1998. Status: PROPOSED STANDARD.

[HKC70] F. E. Hart, R. E. Kahn, and V. G. Cerf. Host/host communication in the ARPA network. In *AFIPS Spring Joint Computer Conference*, May 1970.

[HKO+70] F. E. Hart, R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden. The interface message processor for the ARPA computer network. In *AFIPS Spring Joint Computer Conference*, May 1970.

[HM91] M. Hayter and D. McAuley. The Desk Area Network. *ACM Operating Systems Review 1991*, 1991.

[Hoe95] Janey C. Hoe. Start-up dynamics of TCP's congestion control and avoidance schemes. Master's thesis, Massachusetts Institute of Technology, June 1995.

[Hoe96] Janey C. Hoe. Improving the start-up behavior of congestion control scheme for TCP. In *Proceedings of ACM SIGCOMM '96*, June 1996.

[HS97] Tom Henderson and Emile Sahouria. Improving fairness of TCP Congestion Avoidance. Technical report, LBL, May 1997.

[HTH98] Amy Hughes, Joe Touch, and John Heidemann. Issues in TCP Slow Start after idle. draft-ietf-tcpiml-restart-00.txt, 1998.

[Hub80] Zimmermann Hubert. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, April 1980.

[ID91] I.M. Leslie and D.R. McAuley. Fairisle: an ATM network for the local area. *ACM Communication Review*, 19(4):327–336, September 1991.

[IMG+79] M. Irland, J. C. Majithia, J. L. Grange, N. Cohen, and C. O'Donnell. Experimants in congestion control techniques. In Jean-Louis Grange and Michel Gien, editors, *Flow Control in Computer Networks: Proceedings of the International Symposium on Flow Control in Computer Networks*, page 211. North Holland, 1979. Versailles.

[Irl78] Marek Irland. Buffer management in a packet switch. *IEEE Transactions on Communications*, COM-26(3), March 1978.

[Jac] Van Jacobson. Session directory. ftp://ftp.ee.lbl.gov/conferencing/sd.

[Jac88] Van Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM '88*, 1988.

[Jac90] Van Jacobson, April 1990. Communication to the End to End interest mailing list.

[Jac94] V. Jacobson. Tutorial: Multimedia conferencing on the Internet. In *Proceedings ACM SIGCom*. ACM, August 1994.

[Jai84] R. Jain. A quantitative measure of fairness and discrimination for resource allocation in shared computer sytems. Technical report, Digital Equipment Corporation, 1984.

[Jai86] R. Jain. Divergence of timeout algorithms for packet retransmissions. In *Proceedings Fith Annual International Phoenix Conference on Computers and Communications*, pages 174–179, 1986.

[Jai89] R. Jain. A delay based approach for congestion avoidance in interconnected computer networks. *Computer Communications Review, ACM SIGCOMM*, pages 56–71, October 1989.

[Jai86] R. Jain. A timeout based congestion control scheme for window flow-controlled networks. *IEEE Journal of Selected Areas of Communications*, SAC-4(7):1162–1167, 86. control.pdf.

[JB90] Van Jacobson and R.T. Braden. Congestion control and avoidance. *ACM Computer Communications Review,*, 18(4):314–329, August 1990.

[JBH78] Irwin Mark Jacobs, Richard Binder, and Estil V. Hoversten. General purpose packet satellite networks. *Proceedings of the IEEE*, 1978.

[JDSZ95] Sugih Jamin, Peter B. Danzig, Scott Shenker, and Lixia Zhang. A measurement - based admission control algorithm for integrated services packet networks. In *Proceedings of ACM SIGCOMM '95*, 1995.

[Joh95] Stacy R. Johnson. Increasing TCP throughput by using an extended acknowledgement interval. Master's thesis, Ohio University, 1995.

[Jr.85] Paul J. Santos Jr. (comments)2 on "Congestion Control in IP/TCP internetworks". *Computer Communication Review*, 15, 1985.

[JR88a] R. Jain and K. K. Ramakrishnan. Congestion avoidance and computer networks with a connecitonless network layer. part i: Concepts, goals and methodology. Technical report, Digital Equipment Corporation, 1988.

[JR88b] Raj Jain and K. K. Ramakrishnan. Congestion avoidance in computer networks with a connectionaless network layer: Concepts, goals and methodology. In *Proceedings IEEE Computer Networking Symposium*, pages 134–143, Washington D. C., April 1988.

[Kah72] Robert E. Kahn. Flow control in a resource-sharing computer network. *IEEE Transactions on Communications*, COM-20(3):538–546, June 1972.

[Kah94] Robert E. Kahn. The role of government in the evolution of the Internet. *Communications of the ACM*, 37(8):15–19, August 1994.

[Kam81] Farouk Kamoun. A drop and throttle flow control policy for computer networks. *IEEE Transactions on Communication*, COM-29(4):445, April 1981.

[KC98] Nadia Kausar and Jon Crowcroft. End to end reliable multicast transport protocol for floor control and other conference control functions. In *Hipparch '98 Workshop Program*, June 1998.

[Kes91] S. Keshav. *Congestion Control in Computer Networks.* PhD thesis, U. C. Berkely, August 1991.

[Kes97] S. Keshav. *An Engineering Approach to Computer Networking.* Addison Wesley, 1997.

[KGBK78] Robert E. Kahn, Steven A. Gronemeyer, Jerry Burchfiel, and Ronald C. Kunzelman. Advances in packet radio technology. *Proceedings of the IEEE*, 66(11):1468–1495, 1978.

[KKP00] Shrikrishna Karandikar, Shivkumar Kalyanaraman, and Prasad Pagal. TCP rate control. *Computer Communications Review*, 30(1), 2000.

[Kle78] Leonard Kleinrock. Principles and lessons in packet communication. *Proceedings of the IEEE*, 66(11):1320–1329, November 1978.

[KM84] Michael J. Karels and Marshall Kirk McKuisk. Network performance and management with 4.3 BSD and IP/TCP. Journal, 1984.

[KM86] Michael J. Karels and Marshall Kirk McKusick. Network performance and management with 4.3bsd and IP/TCP. In *USENIX Summer 1986*, pages 182–188, 1986.

[KM99] P. B. Key and D. R. McAuley. Differential qos and pricing in networks: where flow control meets game theory. In *IEEE Proceedings Software*, volume 146, pages 39–43, March 1999.

[KMBL99] P. B. Key, D. R. McAuley, P. Barham, and K. Laevens. Congestion pricing for congestion avoidance. Technical report, Microsoft Research, 1999.

[KMT98] F. P. Kelly, A. K. Maullo, and D.k.H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. Statistics Laboratory, University of Cambridge, 1998.

[KN74] L. Kleinrock and W. E. Naylor. On measured behavior of the ARPA network. In *AFIPS Spring Joint Computer Conference*, May 1974.

[KNF+00] Y. Kikuchi, T. Nomura, S. Fukungaga, Y. Matsui, and H. Kimita. RFC 3016: RTP Payload Format for MPEG-4 Audio/Visual Streams, 2000.

[KP87] Phil Karn and Craig Partridge. Improving round trip time estimates in reliable transport protocols. In *Proceedings of ACM SIGCOMM '87*, August 1987.

[Kri98] Harihan Krishnan. Analyzing Explicit Congestion Notification (ECN) benefits for TCP. Master's thesis, University of California, 1998.

[Lar99] Lars Eggert and John Heidemann and Joe Touch. Effects of Ensemble-TCP. *ACM Computer Communication Review*, 30(1), January 1999.

[LCC+97] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Khan, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Lawrence G. Roberts, and Stephen S. Wolf. The past and future history of the Internet. *Communications of the ACM*, 1997.

[Les83] Ian Malcolm Leslie. *Extending the Local Area Network.* PhD thesis, University of Cambridge Computer Lab, 1983.

[LK98] D. Lin and H. Kung. TCP Fast Recovery strategies: Analysis and improvements. In *INFOCOM '98*, March 1998.

[LLSZ96] Christopher Lefelhoez, Brian Lyles, Scott Shenker, and Lixia Zhang. Congestion control for best-effort service: Why we need a new paradigm. *IEEE Network*, 10(1), January 1996.

[LM97] D. Lin and R. Morris. Dynamics of Random Early Detection. In *SIGCOMM '97*, September 1997.

[LMB+96] Ian Leslie, Derek McAuley, Richard Black, Timothy Roscoe, Paul Barham, David Evers, Robin Fairbairns, and Eoin Hyden. The design and implementation of an operating system to support distributed multimedia applications. *IEEE Journal on Selected Areas in Communication*, 14(7):1280–1297, September 1996.

[LRCI79] Anthony Lauk, Robert Ryan, George Conant, and Marek Irland. A combined error and flow control algorithm for dynamic receive buffering. In Jean-Louis Grange and Michel Gien, editors, *Flow Control in Computer Networks: Proceedings of the International Symposium on Flow Control in Computer Networks*, pages 173–189. North Holland, 1979. Versailles.

[LTM83] Ian M. Leslie, David L. Tennenhouse, and Derek R. McAuley. ATM everywhere? *IEEE Network*, 1983.

[LTWW93] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of ethernet traffic. In *ACM SIGCOMM*, pages 183–193, 1993.

[M.93] Hayter D. M. *A Workstation Architecture to Support Multimedia.* PhD thesis, University of Cambridge Computer Laboratory, 1993.

[Mca89] Derek R Mcauley. *Protocol Design for High Speed Networks.* PhD thesis, University of Cambridge Computer Laboratory, September 1989. TR 186.

[McC92] Steven McCanne. A distributed whiteboard for network conferencing. Technical report, University of California, May 1992.

[McC99] Steve McCanne. Scalable multimedia communication - using IP multicast and lightweight sessions. *IEEE Internet Computing*, pages 33–45, March 1999.

[MF97] Jamshid Mahdavi and Sally Floyd. TCP-Friendly unicast rate-based flow control. Note sent to end2end-interest mailing list, January 1997.

[MHKE99] M. Mauve, V. Hilt, C. Kuhmunch, and W. Effelsberg. RTP/I - an application layer protocol for the transmission of interactive media with real time characteristics. In *Proc IEEE Multimedia Systems*, 1999.

[Mil83] D. L. Mills. RFC 889: Internet delay experiments, December 1983. Status: UNKNOWN.

[MJ95] S. McCanne and V. Jacobson. VIC: a flexible framework for packet video. In *Proc ACM Multimedia*, pages 511–522, San Fransisco, February 1995.

[MJS99] Handley M., Padhye J., and Floyd S. Congestion Window Validation. Technical report, UMASS CMPSCI Technical Report 99-77, September 1999.

[MJV96] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *ACM SIGCOMM '96*, Stanford CA, August 1996.

[MM96] M. Mathis and Jamshid Mahdavi. Forward ACKnowledgment: Refining TCP congestion control. In *Proceedings of ACM SIGCOMM '96*, pages 281–291, 1996.

[MMFR96a] M. Mathis, J Mahdavi, S. Floyd, and A Romanow. RFC 2018 TCP selective acknowledgement options, 1996.

[MMFR96b] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. RFC 2018: TCP selective acknowledgment options, October 1996. Status: PROPOSED STANDARD.

[Mog92] Jeff C. Mogul. Observing TCP dynamics in real networks. In *Proceedings of ACM SIGCOMM '92*, pages 305–317, October 1992.

[Mog95a] Jeff C. Mogul. The case for Persistent Connection HTTP. In *Proceedings of ACM SIGCOMM '95*, pages 299–313, 1995.

[Mog95b] Jeffrey C Mogul. The case for Persistent - Connection HTTP. Technical report, Western Research Laboratory, 1995.

[MPCC00] Richard Mortier, Ian Pratt, Christopher Clark, and Simon Crosby. Implicit admission control. *IEEE Journal on Selected Areas in Communication*, 2000.

[MRBP98] A. Mankin, A. Romanow, S. Bradner, and V. Paxson. RFC 2357: IETF criteria for evaluating reliable multicast transport and application protocols, June 1998. Status: INFORMATIONAL.

[MSM97] Mathew Mathis, Jeffrey Semke, and Jamshid Mahdavi. The macroscopic behaviour of the TCP Congestion Avoidance algorithm. *Computer Communication Review 27*, 1997.

[Nag84a] J. Nagle. RFC 896: Congestion control in IP/TCP internetworks, January 1984. Status: UNKNOWN.

[Nag84b] John Nagle. Congestion control in IP/TCP internetworks. *Computer Communication Review*, 14(4), October 1984.

[NRK00] Neelkanth Natu, Priya Rajagopal, and Shivkumar Kalyanaraman. SBMCC: A generic source-based congestion control algorithm for reliable maulticast. In *Journal of Computer Communications*, 2000. http://www.ecse.rpi.edu/Homepages/shivkuma/research/papersrpi. html.

[OC01] A.M. Odlyzko and K. G. Coffman. Growth of the internet. *Optical Fibre Telecommunications*, 2001.

[oET00] Effects of Ensemble TCP. Lars eggert and john heidemann and joe touch. *Computer Communications Review*, 30(1), 2000.

[OKM99] T. J. Ott, J.H.B. Kemperman, and Matt Mathis. The stationary behavior of ideal TCP Congestion Avoidance. In *IEEE INFOCOMM'99*, New York, March 1999.

[O'N90] Judy O'Neil. Excerpts from an oral history interview with vintin cerf. http://www.si.edu/resource/tours/comphist/vc1.html, 1990.

[O'N95] Judy O'Neill. The role of ARPA in the development of the ARPANET, 1961-1972. *IEEE Annals of the History of Computing*, 17(4), 1995.

[Pad98] V. N. Padmanabhan. *Addressing the Challenges of Web Data Transport.* PhD thesis, University of California Berkeley, 1998.

[Pax94a] Vern Paxson. Adapting to wide-area network dynamics. Qualifying Paper University of California Berkeley, 1994.

[Pax94b] Vern Paxson. Growth trends in wide area TCP connections. *IEEE/ACM Transactions on Networking*, July 1994.

[Pax96a] Vern Paxson. *Measurement and Analysis of End-to-End Internet Dynamics.* PhD thesis, University of California at Berkeley, 1996.

[Pax96b] Vern Paxson. Towards a framework for defining Internet performance metrics. In *Proceedings of INET 1996*, June 1996.

[Pax97] Vern Paxson. Automated packet trace analysis of TCP implementations. In *Proceedings of ACM SIGCOMM '97*, 1997.

[PF97a] Vern Paxson and Sally Floyd. Why we don't know how to simulate the internet. In *Proceedings of the 1997 Winter Simulation Conference*, 1997.

[PF97b] Vern Paxson and Sally Floyd. Wide-area traffic: The failure of poisson modelling, July 1997.

[PFT99] Jitendra Padhye, Victor Firoiu, and Don Towsley. A stochastic model of TCP Reno Congestion Avoidance and control. Technical report, University of Massachusetts, feb 1999.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its emperical validation. In *Proceedings SIGCOMM'98*, 1998.

[PKTK98] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A model based TCP-friendly rate control protocol. *UMass-CMPSCI Technical Report TR 98-04*, 1998.

[PN98a] K. Poduri and K. Nichols. RFC 2415: Simulation studies of increased initial TCP window size, September 1998. Status: INFORMATIONAL.

[PN98b] K. Poduri and K. Nichols. Simulation studies of increased initial TCP window size. draft-ietf-tcpimpl-poduri-01.txt, 1998.

[Pos80] Jonathan B. Postel. Internetwork protocol approaches. *IEEE Transactions on Communications*, 28(4):604–611, April 1980.

[Pos81a] J. Postel. RFC 790: Assigned numbers, September 1981. Obsoleted by RFC0820 Obsoletes RFC0776 Status: HISTORIC.

[Pos81b] J. Postel. RFC 791: Internet Protocol, September 1981. Obsoletes RFC0760. See also STD0005 Status: STANDARD.

[Pos81c] J. Postel. RFC 792: Internet Control Message Protocol, September 1981. Obsoletes RFC0777. Updated by RFC0950 See also STD0005. Status: STANDARD.

[Pos81d] J. Postel. RFC 793: Transmission Control Protocol, September 1981. See also STD0007. Status: STANDARD.

[Pos81e] J. Postel. RFC 801: NCP/TCP transition plan, November 1981. Status: UNKNOWN.

[Pou76] Louis Pouzin. Distributed congestion control in a packet switch network: the channel load limiter. In *Proceedings 6th Congress Gesellschaft fur Informatik*, 1976.

[Pou81] Louis Pouzin. Methods tools and observation on flow control in packet-switched networks. *IEEE Transactions on Communication*, COM-29(4), apr 1981.

[PP87] W. Prue and J. Postel. RFC 1016: Something a host could do with source quench: The Source Quench Introduced Delay (SQuID), July 1987. Status: UNKNOWN.

[Pri79] W. L. Price. A review of the flow control aspects of the network simulation studies at the national physical laboratory. In Jean-Louis Grange and Michel Gien, editors, *Flow Control in Computer Networks: Proceedings of the International Symposium on Flow Control in Computer Networks*, pages 89–103. North Holland, 1979. Versailles.

[PZ78] Louis Pouzin and Hubert Zimmerman. A tutorial on protocols. *Proceedings of the IEEE*, 66(11):1346–1370, November 1978.

[RAL01] A. Ruddle, C. Allison, and P. Lindsay. Determining the sources of delay in distributed learning environments. In *EPSRC PGNET 2001*, pages 71–78, Liverpool UK, 2001. 2nd Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting, EPSRC.

[RAP98] J. Ramsay, B. Alessandro, and J. Preece. A psychological investigation of long retrieval times on the World Wide Web. *Interqacting with computers*, 10:77–86, 1998.

[RF99] K. Ramakrishnan and S. Floyd. RFC 2481: A proposal to add Explicit Congestion Notification (ECN) to IP, January 1999.

[RHE99] Reza Rejaie, Mark Handley, and Deborah Estrin. Rap: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *IEEE INFOCOMM 99*, 1999.

[Riz00] Luigi Rizzo. pgmcc: a TCP-friendly single-rate multicast congestion control scheme. In *ACM SIGCOMM 2000*, 2000.

[RJ88] K. K. Ramakrishnan and Raj Jain. A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. In *Proceedings of ACM SIGCOMM '88*, 1988.

[RJ90] K. K. Ramakrishnan and R. Jain. A binary feedback scheme for for congestion avoidance in computer networks. *ACM Transactions on Computer Systems*, 8(2):158–181, 1990.

[RJC87] K. K. Ramakrishnan, Raj Jain, and Dah-Ming Chiu. Congestion avoidance in computer networks with a connectionless network layer part iv: A selective binary feedback scheme for general topologies. Technical Report DEC-TR-510, Digital Equipment Corporation, August 1987.

[RKT99] Dan Rubenstein, Jim Kurose, and Don Towsley. Detecting shared congestion of flows via end-to-end measurement. Technical Report 99-66, University of Massachusetts, 1999.

[RKT00a] Dan Rubenstein, Jim Kurose, and Don Towsley. Detecting shared congestion of flows via end-to-end measurement. *ACM Sigmetrics*, 2000.

[RKT00b] Dan Rubenstein, Jim Kurose, and Don Towsley. Detecting shared congestion of flows via end-to-end measurement. In *Proceedings of ACM Sigmetrics 2000*, 2000.

[Rob74] Lawrence G. Roberts. Data by the packet. *IEEE Spectrum*, February 1974.

[Rob78] Lawrence G. Roberts. The evolution of packet switching. *Proceedings of the IEEE*, 66(11):1307–1312, November 1978.

[Ros85] Marshall T. Rose. Comments on "comments on 'congestion control in IP/TCP internetworks' ". *Computer Communication Review*, 15, 1985.

[Rud98] Alan Ruddle. TCP: Learning from past traffic. Multi Service Networks, aug 1998.

[Rud00] Alan Ruddle. A Location Information Server for the internet. In *Proceedings of the 9th IEEE International Conference on Computer Communications and Networks (ICCCN 2000)*, Las Vegas Nevada USA, oct 2000. IEEE Computer Society Press.

[RV98] Luigi Rizzo and Lorenzo Vicisano. RMDP: an FEC-based reliable multicast protocol for wireless environments. *ACM Mobile Computing and Communications Review*, 2(2), April 1998.

[RVC00] Luigi Rizzo, Lorenzo Vicano, and Jon Crowcroft. The RLC multicast congestion control algorithm. *IEEE NETWORK - Special Issue*, 2000.

[RW70] L. G. Roberts and B. D. Wessler. Computer network development to achieve resource sharing. In *AFIPS Spring Joint Computer Conference*, May 1970.

[Ryb80] Antony Rybczynski. X.25 interface and end-to-end virtual circuit service characteristics. *IEEE Transactions on Communications*, 1980.

[Sal80] Jerome H. Saltzer. Source routing for campus-wide internet transport. IEN 144, March 1980.

[Sch01] Henning Schulzrinne. Long term traffic statistics. http://www.cs.columbia.edu/ hgs/internet/traffic.html, 2001.

[SE80] Sax and Edmond. HP3000 TCP design document. Technical Report IEN 167, Bolt Beranek and Newman Inc., July 1980.

[SEFJ97] Puneet Sharma, Deborah Estrin, Sally Floyd, and Van Jacobson. Scalable timers for soft state protocols. In *Proceedings IEEE Infocomm*, April 1997. Kobe Japan.

[SG94] A. Silberchatz and P. Galvin. *Operating System Concepts.* Adison and Wesley, 4th edition, 1994.

[Sho78] John Shoch. Internetworking naming, addressing and routing. IEN 19, 1978.

[SM81] Donald E. Sproule and Frank Mellor. Routing, flow, and congestion control in the datapac network. *IEEE Transactions on Communication*, COM-29(4), April 1981.

[SRC84] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(3):277–288, August 1984.

[SSK97] Srinian Seshan, Mark Stemm, and Randy H. Katz. SPAND: Shared Passive Network Performance Discovery. *Usenix Symp. Internet Technologies and Systems*, December 1997.

[Ste94] W. Stevens. *TCP/IP Illustrated, Vol. 1.* Addison-Wesley, 1994.

[Ste96] W. Richard Stevens. *TCP/IP Illustrated, Vol. III, TCP for Transactions.* Addison-Wesley, 1996.

[Ste97] W. Stevens. RFC 2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, January 1997.

[Ste99] Mark Stemm. *An Network Measurement Architecture for Adaptive Applications.* PhD thesis, University of California at Berkeley, 1999.

[Sun90] Carl A. Sunshine. Network interconnection and gateways. *IEEE Journal on Selected Areas of Communication*, 1990.

[SV95] M. Shreedhar and George Varghese. Efficient fair queuing using deficit round robin. In *Proceedings of ACM SIGCOMM '95*, 1995.

[SV01] Biplab Sikdar and Kenneth S. Vastola. The effect of tcp on the self-similarity of network traffic. In *Proceedings of the 2001 Conference on Information Sciences and Systems*, The John Hopkins Univeresity, mar 2001.

[Tan81] Andrew S. Tanenbaum. *Computer Networks.* Prentice Hall International Editors, 1981.

[TL89] David L. Tennenhouse and Ian M. Leslie. A testbed for wide area ATM research. In *Proceedings of SIGCOMM 1989*, 1989.

[Tou97] J. Touch. RFC 2140: TCP Control Block interdependence, April 1997. Status: INFORMATIONAL.

[VGK78] Cerf Vinton G and Peter T. Kirstein. Issues in packet-network interconnection. *Proceedings of the IEEE*, 66(11):1386–1409, 1978.

[VH97] Vikram Visweswaraiahm and John Heidemann. Improving restart of idle TCP connections. Technical report, University of Southern California, 1997.

[VRC98] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft. TCP-like congestion control for layered multicast data transfer. In *IEEE Infocom '98*, 1998.

[Wal74] D. C. Walden. RFC 660: Some changes to the IMP and the IMP/host interface, October 1974. Status: UNKNOWN. Format: TXT=4991 bytes.

[Wan92] Zheng Wang. *Routing and Congestion Control in Datagram Networks.* PhD thesis, University College London, JAN 1992.

[WC91] Z. Wang and J. Crowcroft. A new congestion control scheme: Slow Start and Search (Tri-S). *ACM Computer Communications Review*, 21(1):32–43, January 1991.

[WC92] Z. Wang and J. Crowcroft. Eliminating periodic packet losses in 4.3-Tahoe BSD TCP congestion control algorithm. *ACM Computer Communications Review*, 22(2):9–16, April 1992.

[Wec80] Start Wecker. DNA: the digital network architecture. *IEEE Transactions on Communications*, 28(4):510–526, April 1980.

[Wis99] Damon Wischik. *Large Deviations and Internet Congestion.* PhD thesis, University of Cambridge, 1999.

[WS99] Huayan Amy Wang and Mischa Schwartz. Achieving bounded fairness for multicast and TCP traffic in the internet. In *ACM SIGCOMM '98*, 1999.

[Zha86] Lixia Zhang. Why TCP timers don't work well. In *ACM SIGCOMM 1986*, 1986.

[Zha90] Lixia Zhang. Virtual Clock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM 1990*, 1990.

[ZQK99] Yin Zhang, Lili Qiu, and Srinivasan Keshav. Optimizing TCP start-up performance. Technical report, TR99-1742, 1999.

[ZSC91] Lixia Zhang, Scott Shenker, and David D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two way traffic. In *Proceedings of ACM SIGCOMM '91*, pages 133–148, 1991.

Appendix A

Appendix

A.1 Packet Trace Collection and Analysis Tool Design

A.1.1 Introduction

To facilitate the analysis of web traffic presented in this chapter passive monitoring techniques were used to capture and process packet level traces of connections with local web servers. The development of a tool to process the traces was required to reconstruct connection and network characteristics such as window size and round trip time.

This section opens with a description of the packet traces, the hardware and software used in their collection and the location of the web servers. The design goals, functionality and structure of the measurement tool are discussed in turn. The details of how particular measurements are made is left to be discussed at the relevant points in this Chapter.

A.1.2 Trace Collection

The traffic from a collection of Web Servers is studied from the configuration shown in Figure A.1 and details of the configuration of the different servers is given in Table A.1. Traffic from the above web servers was monitored from the eighth to the twenty fourth of October 1998. A total of 342,049 connections were logged to a total of 28,225 individual hosts.

The University of Glasgow Department of Computer Science was connected to the University network via a 10 mbit/sec Ethernet link. The department network consisted of several hundred workstations and servers. There were two main web servers, which hosted multiple web sites. The university is in turn connected to the JANET network via a duplex 20 mbit/sec link which is shared with the University of Strathclyde. JANET is the Joint Academic NETwork and is located in the UK.

The main web server was a Sparc Station 10 with dual 50 MHz processors, it hosted the major DCS web sites and a number of logical sites as specified in Table A.1. Mars in an important subsidiary server, it was an Ultra Sparc II with four 300 MHz processors. Both the Sparcs were running SUN OS 5.5.1, the TCP stacks supported the SS, CA, FR and FT algorithms and both use delayed acknowledgments. The maximum segment size is 1460 bytes and the default socket buffer sizes are 8760 bytes. The Time Stamp, PAWS and Window Scale options were not active. The web servers both use HTTP 1.0.

The workstation used to monitor traffic and capture packet headers was a dual homed Pentium 133 with 64 megabytes of RAM. One interface was dedicated to monitoring, whilst the other was used for normal network communication. The monitoring interface was an Intel Ether Express

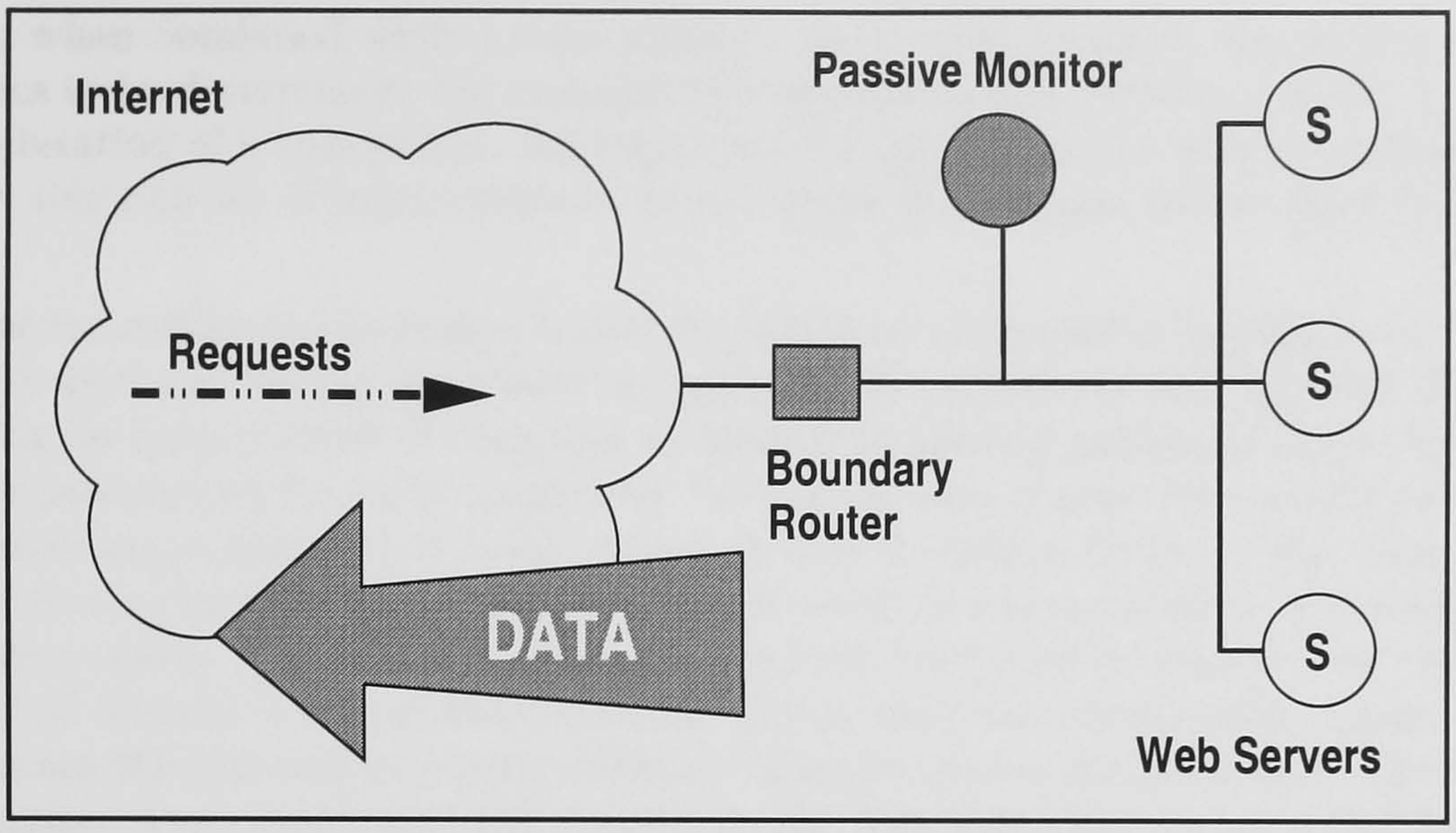


Figure A.1: Server Locations

Alias	Name	Num Cons	Hardware	OS	Web Server
crete	crete	233868	Dual Sparc Station 10	Sun OS 5.5.1	www and ftp
phoebe	mars	74086	Quad UltraSPARC II	Sun OS 5.5.1	Glasgow1999
sais	crete	12560			sais
students	crete	5084			
mars	mars	4689			
ir	crete	3329			
faraday	crete	2885			Faraday
hyperion	mars	2363			British School of Athens
local	crete	1889			
sumisu	sumiso	1283	PC	NT	
bressay	bressay	581	SPARC 20 in G121		

Table A.1: WWW Connections from DCS.GLA.AC.UK

Pro 100 B, which was attached to a 10 megabit per second Ethernet segment, which in turn was used by all traffic going in and out of the department. The operating system was FreeBSD 2.2.7. Time stamped packet headers were passed to `tcpdump` by the Berkeley Packet Filter. The time stamp resolution and accuracy was better than one millisecond. The packet trace was compressed on the fly and written to a local disk prior to processing.

A.1.3 Trace Processing - Design Aims

Numerous techniques have been developed to determine the characteristics of network conditions and traffic. The three main categories of such techniques are kernel instrumentation, network probing and passive monitoring. An advantage of passive monitoring is that it is unobtrusive as no alteration of the kernel or generation of network traffic is required. It can therefore beused to monitor existing web servers without interfering with the traffic being monitored. Kernel instrumentation and probing were both used in the validation of the tool to confirm estimates of window sizes and RTTs.

In the past, passive monitoring has been used to determine the correctness of TCP implementations and to identify phenomena such as ACK compression. The TCP header contains sufficient

information, when combined with a time stamp[1], for a wide range of connection and network characteristics to be determined. For example the connection size, window size, RTT, level of congestion and duration of a connection. An important design aim for the tool is to adopt a structure which allows this rich set of information to be extracted from traces, rather than focusing on one aspect.

An important constraint on the design is that the traces be processed in a single pass. This allowed the tool to be used and results generated in real time. In addition it was desirable for the tool to be able to run for long periods of time and to be able to process million of connections. As state needed to be maintained for each connection the end of each connection needed to be identified and the memory associated with it freed. Although only a small proportion of packets are dropped by the kernel during monitoring, over time this can result in a large number of connections, which do not appear to close. It is therefore necessary for long lived inactive connections to be identified and disposed of. Finally it is important that the tool is easily modifiable with respect to accepting different formats of input and supplying different views on the results generated. The functionality that was required was considered with respect to the logs that may be generated from a packet trace.

Functionality

Next, the functionality required of the tool is considered with respect to the logs that may be generated from a packet trace.The monitoring tool has the capability of generating timestamped logs at different timescales and aggregations. These include packet, window, connection, destination and sessions, each of which is specified below.

Packet: At this granularity a log entry can be generated for every packet. Each log entry includes; a connection identifier, the direction of travel, time since the start of the connection, the relative sequence number, the relative acknowledgment number and the size of the packet. This level of logging allows the evolution of a connection to be plotted at the packet granularity.

Window: A log can be generated for each window of data. Each log contains entries; which identifies the connection, represents the number of packets transfered, the number of data bytes sent and the round trip time. This logging level allows the evolution of the congestion window and the Round Trip time to be tracked.

Connection: Upon the termination of a connection a summary of activity may be logged. This information includes; the connection size in packets and data, the proportion of congestion notifications and the duration of the connection. In addition the time since the last connection to that destination is also logged.

Location: Periodic reports of the network conditions on the path to a location may be generated. Each entry contains the number of connections, the number of packets and bytes sent, the proportion of congestion events, minimum, maximum and the current round trip time. The report interval can be made arbitrarily large or small. This information allows the preparation of distributions for average levels of congestion and round trip times to be prepared.

Concurrent Sessions A log can also be generated for each concurrent session. Where a concurrent session consists of connections between a source and destination whose data transfer phases overlap. Information logged includes the gap since the previous session, the duration of the session the number of connection and the volume of data transfered.

[1] Added by the BPF

A.1.4 Structure

Next the tools structure is presented. The design was based upon the Model, View Controller architecture, where the control, display and model code are held in separate modules. In addition there is a separate input module which converts a packet trace into a packet object. This allows different input formats to be easily accommodated. The controller code sets up the logging levels, the inputs and any filtering that is used. As packets are processed a number of events may occur, for example the end of a connection. For each event the View module is notified and decides whether to create a log depending on the match between the log level and the event.

The model contains two main data structures, which in turn contain the state for connections and destinations. The connections are held in a hash table, with a linked list off each entry. The destinations are held in a similar table. Each connection contains a pointer to its destination, which facilitates the passing of readings from connections to destinations.

In addition to the data access data structures the connections and destinations are held in separate queues, which allow the oldest inactive connection or destination to be removed if predefined quotas are exceeded.

The distinction between TCP's macro state, which defines the state transitions a TCP connection goes through and the micro state which tracks the progress of data transfer is mirrored within the monitor.

A.1.5 Summary

This section has presented the methodology used in the collection of traces of web traffic. It has also discussed the design aims and implementation of a tool for obtaining network and connection characteristics from the traces. The analysis tool is capable of operating on or off line and generating logs at a number of granularities. To achieve this flexibility state is maintained for each connection and destination.

The first peak of the time between sessions distribution occurs at around 100 ms, this time span is too short to reflect user requests, we therefore allocate it to the time between sessions, which together would make up a web page. This class of inter-session gap would benefit from a policy of exponential window decay, adding an estimated 26-32% more aggregation.

Next the session inter-arrival time is considered. Here an estimation of the distribution of requests to the network are made. All inter-arrival times which have a gap between sessions that is less than three round trip times are removed, on the grounds that these are likely to represent pauses introduced by browser TCP interaction rather than separate requests.

The resulting distribution is bimodal with peeks at around five and thirty seconds. There is a strong tail, with further peaks starting at the 24 hour interval. 84% percent of sessions are separated by a gap of less than 10 minutes.

The gap in requests are of the order of at least seconds, orders of magnitude larger than round trip times. Exponentially Decaying a congestion window each RTT will not help in reducing delay across web pages.