Distributed Consensus in Wireless Network

Dachao Yu

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

James Watt School of Engineering

College of Science and Engineering

University of Glasgow

March 2023

Abstract

Connected autonomous systems, which are powered by the synergistic integration of the Internet of Things (IoT), Artificial Intelligence (AI), and 5G technologies, predominantly rely on a central node for making mission-critical decisions. This reliance poses a significant challenge that the condition and capability of the central node largely determine the reliability and effectiveness of decision-making. Maintaining such a centralized system, especially in large-scale wireless networks, can be prohibitively expensive and encounters scalability challenges. In light of these limitations, there's a compelling need for innovative methods to address the increasing demands of reliability and latency, especially in mission-critical networks where cooperative decision-making is paramount. One promising avenue lies in the distributed consensus protocol, a mechanism intrinsic to distributed computing systems. These protocols offer enhanced robustness, ensuring continued functionality and responsiveness in decision-making even in the face of potential node or communication failures.

This thesis pivots on the idea of leveraging distributed consensus to bolster the reliability of mission-critical decision-making within wireless networks, which delves deep into the performance characteristics of wireless distributed consensus, analyzing and subsequently optimizing its attributes, specifically focusing on reliability and latency. The research begins with a fundamental model of consensus reliability in an crash fault tolerance protocol Raft. A novel metric termed *ReliabilityGain* is introduced to analyze the performance of distributed consensus in wireless network. This innovative concept elucidates the linear correlation between the reliability inherent to consensus-driven decision-making and the

reliability of communication link transmission. An intriguing discovery made in my study is the inherent trade-off between the time latency of achieving consensus and its reliability. These two variables appear to be in contradiction, which brings further performance optimization issues.

The performance of the Crash and Byzantine fault tolerance protocol is scrutinized and they are compared with original centralized consensus. This exploration becomes particularly pertinent when communication failures occur in wireless distributed consensus. The analytical results are juxtaposed with performance metrics derived from a centralized consensus mechanism. This comparative analysis illuminates the relative merits and demerits of these consensus strategies, evaluated from the dual perspectives of comprehensive consensus reliability and communication latency.

In light of the insights gained from the detailed analysis of the Raft and Hotstuff BFT protocols, my thesis further ventures into the realm of optimization strategies for wireless distributed consensus. A central facet of this exploration is the introduction of a tailored communication resource allocation scheme. This scheme, rooted in maximizing the performance of consensus mechanisms, dynamically assesses the network conditions and allocates communication resources such as transmit power and bandwidth to ensure efficient and timely decision-making, which ensures that even in varied and unpredictable network conditions, consensus can be achieved with minimized latency and maximized reliability.

The research introduces an adaptive protocol of distributed consensus in wireless network. This proposed adaptive protocol's strength lies in its ability to autonomously construct consensus-enabled network even if node failures or communication disruptions occur, which ensures that the network's decision-making process remains uninterrupted and efficient, irrespective of external challenges. The sharding mechanism, which is regarded as an effective solution to scalability issues in distributed system, does not only aid in managing vast networks more efficiently but also ensure that any disruption in one shard

cannot compromise the functionality of the entire network. Therefore, this thesis shows the reliability and security analysis of sharding that implemented in wireless distributed system. In essence, these intertwined strategies, rooted in the intricate dance of communication resource allocation, adaptability, and sharding, together form the bedrock of my contributions to enhancing the performance of wireless distributed consensus.

Contents

List of Tables

List of Figures

Acknowledgements

Firstly, I would like to express the deepest thanks to my primary supervisor, Prof. Lei Zhang, who contributes time and efforts to supervise my research, which helps me get through the harsh during pandemic. Moreover, I must also thank Prof. Muhammad Ali Imran and my colleagues from the CSI group for their supports on my research and life in the University of Glasgow. Finally, I appreciate my family for their suggestions, support and helps.

Abbreviations

- 5G - Fifth-generation technology standard for broadband cellular networks
- ACDA - Average Consensus-based Distributed Algorithm
- AI - Artificial Intelligence
- AODV - Ad hoc On-Demand Distance Vector Routing
- BFT - Byzantine Fault Tolerance
- BGP - Byzantine General Problem
- BTC - Bitcoin
- CFT - Crash Fault Tolerance
- CN - Central Node
- CR - Consensus Reliability
- CV - Coefficient of Variation
- DApp - Decentralized Application
- DLT - Distributed ledger technology
- DSDV - Destination-Sequenced Distance Vector Routing
- DSR - Dynamic Source Routing
- ETH - Ethereum
- FC - Full Consensus
- FN - Full Function Node
- IIoT - Industrial Internet of Things
- IP - Internet Protocol
- IoT - Internet of Things
- LM - Lagrangian Multiplier
- LR - Link reliablity

- `MANETs` - Mobile Ad Hoc Networks
- `P2P` - Peer to Peer
- `PBFT` - Practical Byzantine Fault Tolerance
- `PCDA` - Perception-Collection-Decision-Action
- `PICA` - Perception-Initiative-Consensus-Action
- `PoW` - Proof of Work
- `PoS` - Proof of Stake
- `PSO` - Practical Swarm Optimization
- `QC` - Quorum Certificate
- `RCC` - Reliable Cloud Connectivity
- `RG` - Reliability Gain
- `RTV` - Real-time Virtualization
- `SINR` - Signal-to-interference-plus-noise ratio
- `SNR` - Signal-to-noise ratio
- `SQP` - Sequential Quadratic Programming
- `TP` - Log Throughput
- `TS` - Taylor Series
- `URC` - Ultra Reliable Communication
- `URLLC` - Ultra Reliable Low Latency Communications
- `V2V` - Vehicle to Vehicle
- `V2X` - Vehicle to Everything
- `VRF` - Verifiable Random Function
- `WDC` - Wireless Distributed Consensus

List of Pulications

Journal Paper

- **D. Yu**, Y. Sun, Y. Li, L. Zhang, M. A. Imran ***Communication Resource Allocation of Raft in Wireless Network***
  in IEEE Sensors Journal, doi: 10.1109/JSEN.2023.3293715.
- **D. Yu**, W. Li, H. Xu and L. Zhang, ***Low Reliable and Low Latency Communications for Mission Critical Distributed Industrial Internet of Things***
  in IEEE Communications Letters, doi: 10.1109/LCOMM.2020.3021367.
- **D. Yu**, H. Wu, Y. Sun, L. Zhang, M. A. Imran ***Adaptive Protocol of Raft in Wireless Network***, Submitted to Ad Hoc Networks

Conference Paper

- **D. Yu**, L. Zhang ***Centralized and Distributed Consensus in Wireless Network: An Analytical Comparison*** 2022 IEEE 20th International Conference on Embedded and Ubiquitous Computing (EUC), doi: 10.1109/EUC57774.2022.00022.
- **D. Yu**, H. Xu, L. Zhang, B. Cao and M. A. Imran, ***Security Analysis of Sharding in the Blockchain System*** 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Helsinki, Finland, 2021, pp. 1030-1035, doi: 10.1109/PIMRC50174.2021.9569351.

List of Symbols

Table 1: Common parameters used in Chapter 2 and 3

Notation	Definition
N	Number of nodes in the network
r	Total rounds of synchronization
f	Number of links failed replicas in a distributed consensus
M_t	Mean of the number of time slots cost
t	Length of the time slot for one link transmission
T	Overall time latency for consensus
P_l	link reliability
$P_{ls}(r)$	link reliability in the r_{th} round of synchronization
P_C	consensus reliability
P_F	Full consensus reliability
TP	Log throughput
S_k	Large Scale Effect of the k^{th} channel
H_k	Rayleigh Fading Gain of the k^{th} channel
P_{sum}	The overall transmit power
B_{sum}	The overall bandwidth
P_{tk}	Transmit Power allocated to the k^{th} channel
B_k	Bandwidth allocated to the k^{th} channel
P_{lk}	Link reliability of the k^{th} channel
t_k	Transmission time of the k^{th} channel
t_c	Transmission time cost by consensus
N_{max}	Number of node with maximized consensus reliability

Table 2: Common parameters used in Chapter 4

Notation	Definition
TC	Current term of consensus
VF	Candidate ID that this node vote for
LOG	Log entries that contain the commands and corresponding term
CI	Index of highest Log entry that is committed
AI	Index of highest Log entry applied to state
NI	Index of next log entry send send to nodes
MI	Index of highest log entry replicated on server (matched)
CT	Candidate Term
CA	Candidate Address
LLI	Index of Last Log Entry
LLT	Term of Last Log Entry
FT	Candidate Term
$VOTE$	Candidate Address
LA	Leader Address
NLI	Index of New Log Entry
NLE	Term of New Log Entry
LC	Leader Commit Index
FT	Follower Term
SR	Success Response
SNS	Sequence Number of Source Node
HN	Hopping Number
$RRID$	RREQ Identity
SND	Sequence Number of Destination Node
HN	Hopping Number
SR	Success Response
$RPID$	RREP Identity
M	Number of shards
H	Total number of malicious nodes in the network
R	The ratio of malicious nodes in total nodes
k	The number of nodes distributed in a shard
h	The number of malicious nodes distributed in any shards
$P(k)$	Probability of k nodes distributed in a shard
$P(m)$	Probability of m nodes distributed in validating set
P_h	Probability of h malicious nodes in a shard
P_c	Probability of secure transaction in a shard
$P_H(k)$	Probability of h malicious nodes in a k nodes shard
$P_s(k)$	Probability of secure transaction in a k nodes shard

Chapter 1

Introduction

1.1 Background of Distributed Consensus

Distributed consensus lies in the field of distributed systems, where multiple independent nodes or computers work together to achieve a common goal. In such systems, achieving consensus or agreement among the nodes is crucial for ensuring the correct functioning of the system [1]. The concept of distributed consensus emerged as a solution to the problem of achieving agreement among the nodes in critical aircraft control applications of Software Implemented Fault Tolerance (SIFT) despite the possibility of nodes failing or behaving maliciously [2]. The problem is commonly known as the Byzantine Generals Problem, which was first proposed in 1982 by Leslie Lamport, Robert Shostak, and Marshall Pease [3].

This problem is named after the hypothetical scenario where a group of Byzantine generals is trying to coordinate an attack on a common enemy. The generals can only communicate with each other by sending messages, but some of them may be traitors who will deliberately send false messages to disrupt the coordination. The challenge is to come up with a protocol that ensures that the loyal generals agree on a common plan of action, even if some of the generals are traitors.

Over the years, researchers have proposed several algorithms and protocols for achieving distributed consensus, such as the Paxos algorithm [4], the Raft algorithm [5], and the Practical Byzantine Fault Tolerance (PBFT) protocol [6]. These algorithms and protocols have been applied to various distributed systems, including databases [7], blockchain [8], and cloud computing [9] [10] [11], to ensure the consistency and reliability of the system, and they can work as an interior algorithm that regulates the decision based on the collected information by nodes. In the protocol of a distributed consensus, every participant is capable of transmitting and receiving the command to switch the state of replicas if it follows specific fault-tolerant protocols [12]. The majority of distributed consensus protocols aim to solve two types of fault: Crash fault and Byzantine fault.

Crash fault refers to a sudden and unexpected failure of a node, which may be caused by various factors such as hardware failure, software bugs, or network problems. Crash fault tolerance is a concept in distributed systems that refers to the ability of a system to continue functioning even if one or more nodes in the system experience a crash or failure. To achieve crash fault tolerance, distributed systems often use replication, where multiple copies of the same data or process are stored on different nodes in the system. If one node fails, the other nodes can continue to operate and provide the required services. The failed node can be replaced or restarted without affecting the overall system functionality [13]. Crash fault tolerance (CFT) protocols, such as Raft and Paxos, are designed to follow these coteries to manage reliable state duplication and prevent system breakdown from node crash failure.

Paxos is the first CFT consensus algorithm in the field of distributed systems, proposed by Leslie Lamport in 1989. The main goal of Paxos is to ensure that a network of distributed, potentially unreliable (in the sense of crashing or going offline), agents can agree on a single value (a consensus). This is particularly crucial for databases and other distributed storage systems, where they need to ensure data consistency across all nodes. Paxos operates through a series of proposals, where each proposal is issued by a proposer, voted on by acceptors, and disseminated by learners. The fundamental guarantee of Paxos is

that if a value is chosen, then every future proposal will also suggest that value. The protocol is structured such that it ensures that only one value is selected and that all participants can eventually be aware of that value, despite the potential for failures and message losses [14]. However, Paxos can be quite complex to implement correctly due to the intricate interplay of its components. There are simpler versions like Multi-Paxos [15] and variations [16] [17] developed to handle more complex real-world scenarios, such as Google's Chubby [18] and Apache Zookeeper [19]. These derivatives aim to improve upon the efficiency and simplicity of the base Paxos algorithm.

CFT assumes that a node that fails will stop functioning completely but it will not behave maliciously or provide incorrect information to other nodes in the system. However, in real-world scenarios, nodes may fail in more complex ways, such as by sending conflicting information, withholding information, or launching attacks on other nodes in the network [20]. In such cases, CFT may not be sufficient to ensure the correctness and consistency of the system. This is because CFT does not provide mechanisms to detect or isolate malicious nodes, and relies on the assumption that a majority of the nodes in the system are honest and will provide correct information. To address these limitations, more advanced fault tolerance mechanisms such as Byzantine Fault Tolerance (BFT) are used in distributed systems where there is a risk of malicious behavior. BFT provides stronger guarantees of correctness and consistency in the presence of arbitrary and malicious faults but may require more computational resources and communication overhead than CFT.

Byzantine failure represents the malicious behaviors given by an adversary, including contradictory commands to the progress, communication abort, and lengthy intentional delay to critical messages [21], which are more disruptive to the system than crash failures. Such failures may be caused by software bugs, hardware failures, or malicious attacks, and may result in the node providing incorrect information or behaving in a way that disrupts the system's operation. Therefore, Byzantine fault tolerance is proposed as a mechanism to let the whole system consistently respond correctly and reliably even in the presence of faulty nodes that may behave maliciously or send equivocation to other nodes. To achieve

Byzantine fault tolerance, distributed systems use BFT protocols like PBFT and Hotstuff BFT [22] to keep a unique sequence in the state of nodes through quorum intersection, and to ensure that the correct information is propagated throughout the system even in the presence of faulty nodes [23]. These algorithms and protocols often involve redundancy, such as replicating data or processes across multiple nodes, and consensus mechanisms that allow nodes to agree on the correct state of the system.

Byzantine fault tolerance is particularly important for critical systems, such as financial systems including blockchain [24] and cryptocurrency [25] [26], where the consequences of faulty behavior can be severe. Some real-time aircraft systems, such as the Boeing 777 Aircraft Information Management System and flight control system, has used low latency Byzantine fault tolerance solution to ensure their robustness [27].

In a distributed system, a quorum typically refers to the minimum number of nodes that must participate to make a decision or commit an action. If a system wants to remain consistent and avoid conflicting decisions, any two quorums in this system should have an intersection, which means they should share some nodes. This mechanism ensures that the situation where one quorum decides on one action and another quorum decides on a conflicting action cannot happen, since they both always have at least one node in common that would notice and prevent such a conflict. In CFT, two quorums only need to share one node in a quorum intersection because all nodes are assumed trusted. However, in BFT this intersection needs to have more nodes than the malicious nodes f in the system, which means the number of nodes in the network should be more than $2f$ to achieve consensus. Therefore, BFT can be computationally expensive and require significant communication overhead. The protocol of BFT usually requires multiple rounds of communication and computation to achieve consensus among nodes in a distributed system, which can lead to increased latency and reduced throughput. Furthermore, BFT assumes that all nodes in the system have equal computing power and communication capabilities and there is no distinction between nodes that are more or less trusted. In real-world scenarios, nodes may have different capabilities and trust levels, and the system may

require more nuanced mechanisms for achieving consensus and managing node behavior. Overall, while BFT provides stronger guarantees of correctness and security than CFT, it requires careful design and tuning to ensure that the benefits outweigh the drawbacks of a particular system.

1.2 Literature Review of Common Distributed Consensus

Distributed consensus algorithms employ several communication schemes, including leader-based [5], peer-to-peer [28] [29], and gossip-based protocols [30]. Our research primarily centers on distributed consensus protocols using leader-based communication, such as Raft and Hotstuff. In these consensus mechanisms, communication adopts a star topology, which signifies a network configuration where a central or coordinator node is directly connected to all other peripheral nodes. The central node serves as the primary communication and coordination point, effectively managing interactions among the peripheral nodes [31].

In the star communication topology for distributed consensus, each peripheral node only needs to communicate with the central node, rather than with every other node in the network. This topology simplifies the communication process and reduces the complexity of maintaining connections. Since all peripheral nodes send their messages directly to the central node, reaching consensus can be faster compared to decentralized topologies, as there are fewer message hops between nodes. The central node can swiftly aggregate information and disseminate the consensus to all connected nodes and detect and handle faulty nodes or network issues more efficiently because it has a direct connection to each peripheral node. Moreover, because the new node needs to establish a connection with the central node, the engagement of new nodes to a network with star topology is relatively straightforward, which allows the network to scale up easily.

However, there are also some notable drawbacks to using a star topology for distributed consensus. If the central node suffers from a single point of failure, it will be compromised and the entire network can be broken down. The reliance on a central node for communication and decision-making can make the network more susceptible to censorship, control, or manipulation, which goes against the principles of decentralization that distributed systems aim to achieve. Therefore, the protocol of distributed consensus with star topology needs to be optimized when it is implemented in a wireless network to overcome these flaws. This section will introduce the protocol of several common distributed consensus for different scenarios.

1.2.1 Protocols of Paxos and Raft

The protocol of distributed consensus has been deployed in many decentralized systems to keep the consistency of the state in nodes. In a system that requires a trusted authority to access (i.e., private blockchain [32]), the possibility that the system suffers from Byzantine fault can be negligible [3]. The crash of nodes and link transmission failure are the main threats to these trusted systems. Therefore, it is appropriate to deploy the CFT protocol in these scenarios. Two common CFT protocols are introduced in the following paragraphs:

The Paxos algorithm, a consensus algorithm named after a fictional legislative consensus system used on the Paxos island in Greece, was introduced by Leslie Lamport in 1989 [33]. Its primary objective is to ensure that in a distributed system, a single value can be agreed upon by participants, even in the face of failures. The origin of Paxos is deeply intertwined with Leslie Lamport's endeavors to understand and simplify the problem of achieving consensus in distributed systems. Lamport initially presented the algorithm in a parable form, in which he described an imaginary parliament on a Greek island, Paxos. The parliament's operation was described as a protocol, which later became the foundation for the Paxos algorithm. While the parable presented a unique way to think about

the problem, many researchers found it confusing. Hence, Lamport later wrote a more traditional, direct explanation of the algorithm. Regardless of its original presentation style, Paxos became a fundamental algorithm in distributed systems, due to its promise of ensuring system consensus in the presence of node failures.

The Paxos algorithm comprises three primary roles: the Proposer, which initiates the consensus process by suggesting a value; the Acceptor, which either accepts or declines the proposed values; and the Learner, which acknowledges the agreed-upon value. In the first phase, known as the Prepare Phase, the Proposer selects a unique proposal number n and transmits a prepare request with this number to a majority of the Acceptors. When an Acceptor gets a prepare request with n that is higher than any it has seen before, it sends back two types of information to the Proposer: a commitment that it will not accept proposals with a number less than n in the future, and details of the highest-numbered proposal it has previously accepted. If the Proposer hears back from a majority of Acceptors, it will issue a new proposal with the number n and a value v. v is typically chosen based on the highest-numbered proposals received from the Acceptors. If the Proposer doesn't get any previously accepted proposals in the responses, it can freely choose the value for v.

The second phase called the Accept Phase, begins when the Proposer sends out an accept request, containing its proposal with number n and value v, to a majority of Acceptors. An Acceptor, upon receiving this request, will agree to the proposal if it cannot commit to any higher-numbered proposal during the Prepare Phase. The moment a majority of Acceptors accept a particular proposal, the associated value is deemed chosen. Then the Learners are informed of this decision and can proceed with the necessary actions, such as committing a transaction or updating a database. It is worth noting that while Paxos guarantees a consensus, it does not always ensure rapid progress in systems where communication is asynchronous because Proposers might continually override each other's proposals. Practical implementations often introduce additional mechanisms, like designating a leading Proposer, to ensure decisions are made in a timely fashion.

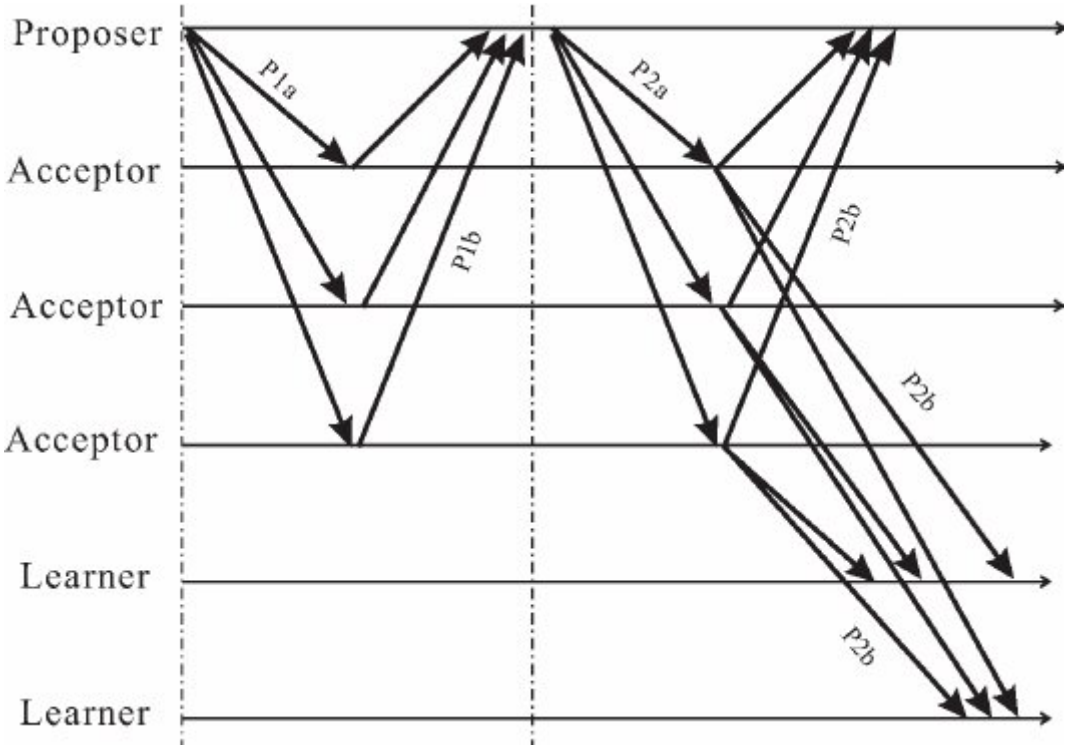

Figure 1.1: Normal operation of the Paxos algorithm [34]

Raft, as a CFT algorithm, is prevalently implemented in a private, trustworthy, distributed system to tolerate the breakdown of replicas [5]. Because Raft is simpler and understandable than the traditional CFT protocol Paxos [4], this CFT consensus protocol has drawn attention to the applications [35], [36] and [37]. In wireless communication networks, link transmission failure can be regarded as a type of breakdown, which means Raft is practical in this situation. The Raft-enabled distributed network is composed of consensus replicas, including the leader and followers. The protocol of Raft contains two stages: Leader election and log replication, which is shown in Fig. 1.2.

Leader election begins with the situation that a follower node transitions to a candidate state when it cannot receive any heartbeat messages from the current leader within a predefined timeout period. The candidate then starts a new election term, votes for itself, and sends a *RequestVote* message to all other nodes in the network. Upon receiving the *RequestVote* message from this candidate, followers decide whether to vote for the candidate or not based on its state and the content of the messages. They need to check

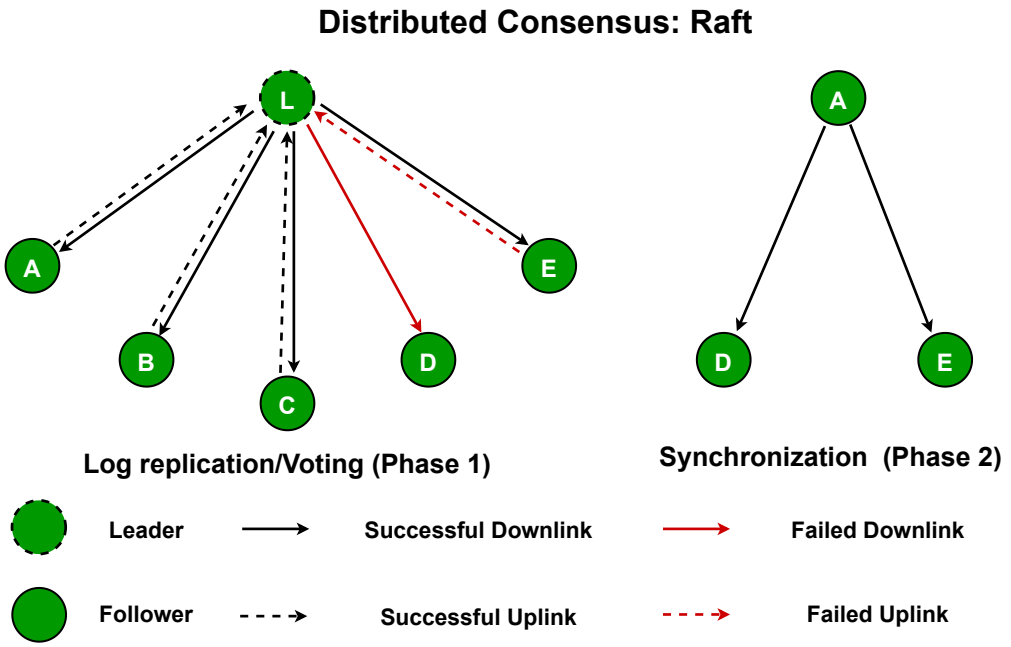

Figure 1.2: Communication scheme of distributed consensus with Raft

if they have already voted in the current term and if the candidate's log is at least as up-to-date as their own. If both conditions are satisfied, the follower grants its vote to the candidate. If the candidate receives votes from over half of the nodes in the whole network, it transitions to the leader state and starts sending heartbeat messages to maintain its leadership. If the candidate fails to win votes from the majority, it reverts to follower state, and the process may repeat until a new leader is elected.

In the stage of log replication, the leader needs to pack the commands in log entries and replicate the entries to all followers ceaselessly through downlink transmission. Depending on the successful reception of entries replication, the followers reply confirmation packets to the leader through uplink unicast and start to execute the confirmed commands or synchronize the state of leader in the current term. A successful Raft consensus represents that more than 50% overall followers can receive the log entries from the leader and send the confirmation back to the leader successfully in one term. The selection of the leader may follow the criteria of maximizing performance because the node with better wireless connections is more likely to become the leader.

In some scenarios, the states of all nodes from the network need to be synchronized in a time interval, which requires an extra synchronization stage after the stage of log replication. This synchronization stage may improve the consensus reliability but cause longer time latency [38].

1.2.2 Protocols of PBFT and Hotstuff BFT

Practical Byzantine Fault Tolerance (PBFT) is a consensus algorithm designed to work efficiently in asynchronous systems and cope with Byzantine faults. The Byzantine Generals Problem, which it addresses, is a situation in which actors in a distributed system must agree on a strategy, but some of these actors may be unreliable or malicious. The term Byzantine or Byzantine Generals Problem refers to a classic problem in distributed computing where actors must reach a consensus even if some of them are traitors who might lie. Practical in PBFT emphasizes its aim to provide a solution that is efficient enough for real-world use.

Practical Byzantine Fault Tolerance (PBFT) differentiates nodes into two primary roles: a single primary node and multiple backup nodes. The primary node's task is to propose the order of logs, whereas the backup nodes are responsible for agreeing or disagreeing with this proposed order. The algorithm's operation can be categorized broadly into three phases: Pre-prepare, Prepare, and Commit. In each phase, the nodes exchange messages to achieve consensus. Initially, in the Pre-prepare phase, the primary node, after receiving a client's request, broadcasts a Pre-prepare message containing the log details to all backup nodes. Following this, during the Prepare phase, backup nodes, after validating the received pre-prepare message, send a prepare message to all nodes, signifying their readiness to process the log. Finally, in the Commit phase, a backup node, having received $2f$ valid prepare messages, where f denotes the maximum number of nodes that can be faulty, sends a commit message to all nodes. This action is an assertion that more than $\frac{2}{3}$ of the nodes have acknowledged the log and are poised to finalize it.

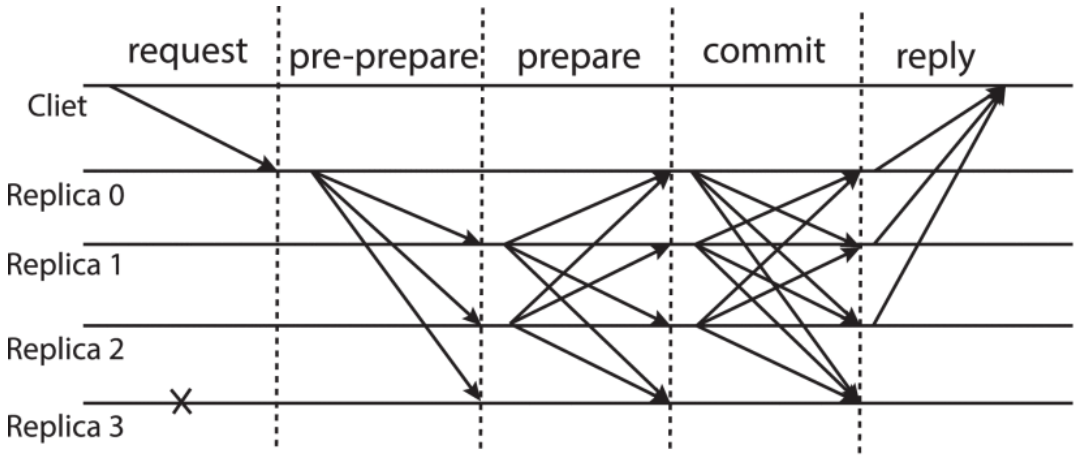

Figure 1.3: Communication scheme of PBFT [6]

For the actual execution, once a node gathers $2f+1$ commit messages for a particular log, including its own, it carries out the log and then communicates the result to the client. An integral component of PBFT is its ability to initiate view changes. This mechanism comes into play if a primary node is perceived to be malfunctioning or is dormant. In such instances, backup nodes can trigger a shift in view if they discern an absence of timely communication from the primary. Successful execution of this change results in the election of a new primary.

PBFT's robustness is underscored by two pivotal attributes: safety and liveness. While safety guarantees that any pair of nodes decide on identical transaction orders, liveness ensures that each request from a correct node will inevitably be committed. This dual assurance stems from the algorithm's mandate that a transaction can only be committed after being validated by $2f+1$ nodes. Lastly, PBFT's fault tolerance capacity enables it to manage up to $\frac{n-1}{3}$ malicious nodes (with n as the total nodes). Therefore, for the system to accommodate f corrupt nodes, the network must consist of at least $3f+1$ nodes. In practical applications, the principles of PBFT find relevance in several contemporary blockchain platforms, especially where rapid transaction throughput and irrevocability are prioritized over a completely permissionless and decentralized environment. The definitive nature of PBFT, wherein a committed transaction is irreversible, makes it stand out from many BFT protocols.

Hotstuff, as a BFT state machine replication protocol, aims to ensure that non-faulty replicas agree on the order of execution for client-initiated service commands in a decentralized network with $N \geq 3f+1$ replicas, despite the efforts of f Byzantine replicas [22]. The Libra blockchain project has chosen the Hotstuff BFT as its consensus protocol because of the responsiveness and linearity in the protocol [23]. The view, as a round of consensus of Hotstuff BFT, contains three phases: prepare, pre-commit, and commit. The basic Hotstuff protocol works in a succession of view numbers with monotonically increasing view numbers. Each view number has a unique dedicated leader for other replicas. The leader must collect votes from a quorum of $N-f$ replicas in three phases. The collection of $N-f$ votes to one leader is referred to as a quorum certificate (QC), which is associated with a particular node and a view number [39]. In the prepare phase, the leader should

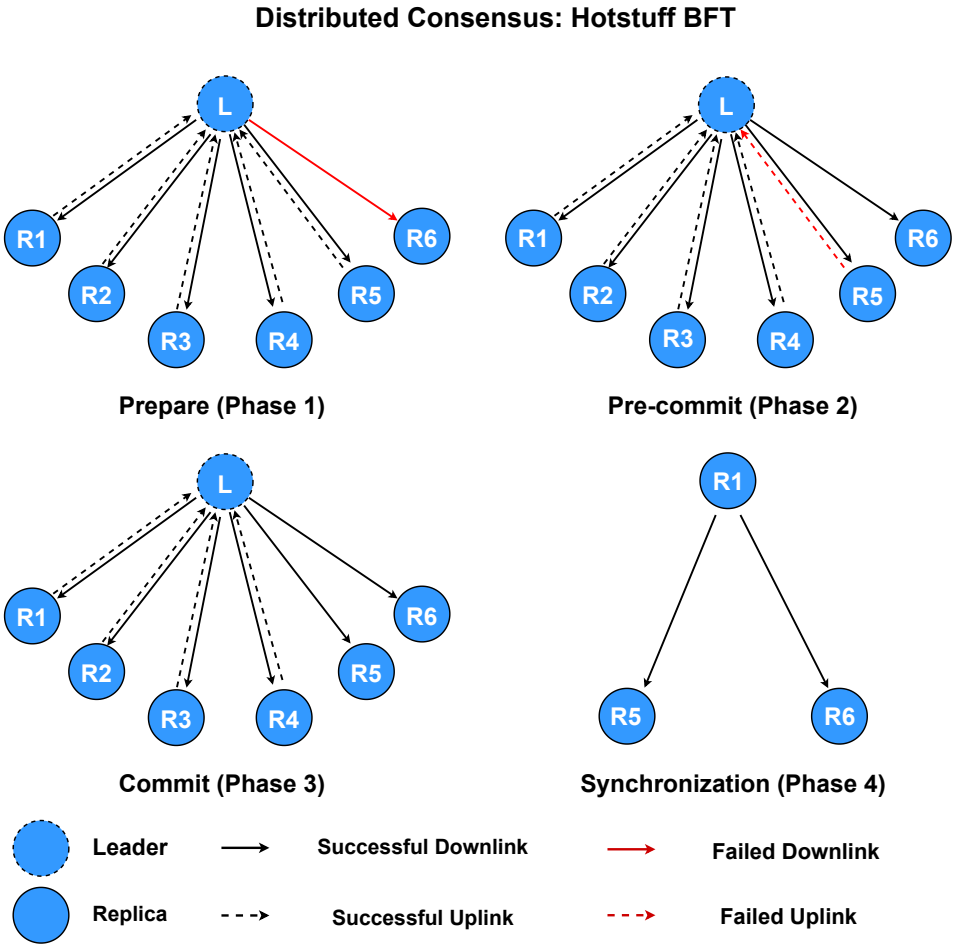

Figure 1.4: Communication scheme of distributed consensus with Hotstuff BFT

collect new-view messages from replicas and process these messages to select a branch

that has the highest preceding view. If over $N-f$ replicas have sent their messages to the leader successfully, a valid prepare QC can be formed through a threshold signature scheme. Then the leader broadcasts the prepare QC in pre-commit messages to replicas, and replicas respond to the leader with the pre-commit vote if they receive and verify the prepare QC. Similarly, while receiving more than $N-f$ pre-commit votes, the leader combines them into a pre-commit QC and broadcasts it in commit messages to replicas. Over $N-f$ replicas should send the commit vote back to the leader for the final commit QC combination. After a successful commit QC assembly, the leader needs to send in a decided message to all other replicas to notify them the consensus protocol in this view is completed, and the replicas will update their state according to the decided message.

Compared with the high communication complexity $O(N^2)$ in the phases of other BFT consensus protocols [6], [40], the start topology and threshold signature scheme ensure Hotstuff BFT can have a linear communication complexity to the number of nodes, which is $O(N)$ in all three phases of the consensus protocol and lower than other common BFT protocols like PBFT. In the scenario of wireless communication, this low communication-complexity feature can cause less interference and bandwidth cost [41]. However, because the basic Hotstuff BFT has three phases in its protocol, the communications between the leader and replicas in Hotstuff will cause longer time latency than a BFT protocol with fewer consensus phases.

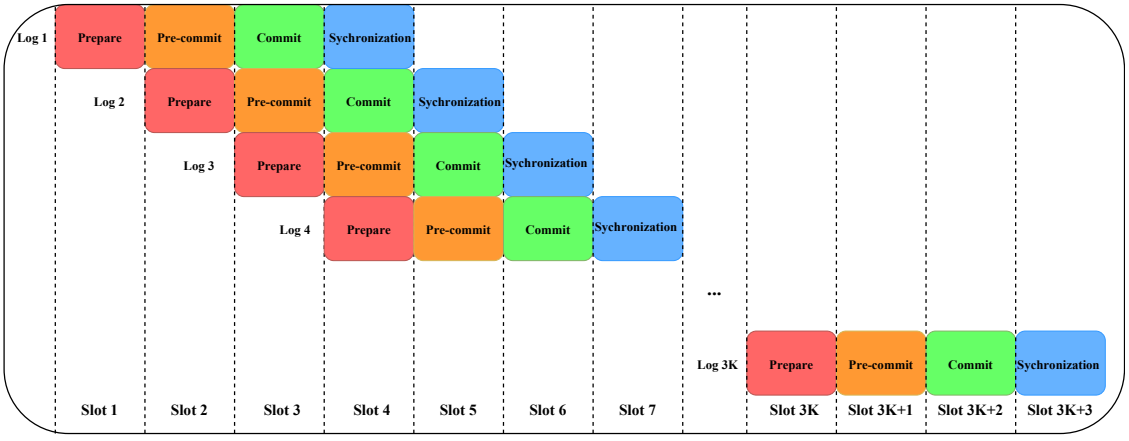

Figure 1.5: Chain Hotstuff BFT protocol

According to the statement of basic Hotstuff BFT, the leader only needs to collect the vote message in the form of a threshold signature through all three phases. This feature ensures that changing the view on every prepare phase of different proposals cannot actively influence the reliability of the consensus. Fig. 1.5 indicates such a pipeline structure of decision-making in a type of more effective Hotstuff BFT protocol, which is called chained Hotstuff [22]. The prepare message of the next view is bound with the pre-commit message of the former view in the same package. Every new view of chained Hotstuff BFT needs to start automatically after the prepare phase of the last view. This approach aims to significantly improve throughputs of the views change in the Hotstuff BFT consensus protocol, which means the average latency for every view in chained Hotstuff will be shorter than the basic Hotstuff protocol if the success rate of the consensus is reasonably high.

1.3 Distributed Consensus in Wireless Network

The exploration of distributed consensus in wireless networks spans a wide array of applications and challenges. This section discusses the current research about wireless distributed consensus in wireless sensor networks, wireless blockchain systems, and independent models or frameworks for connected critical decision-making applications.

1.3.1 Distributed Consensus in Wireless Sensor Network

As the value of distributed consensus becomes increasingly evident in wireless communication networks, a growing number of researchers are turning their attention to exploring its underlying theories and practical applications. Pioneering research efforts have sought to apply distributed consensus mechanisms to wireless sensor networks, specifically for achieving accurate time synchronization among the network's nodes [42]. They have found that distributed consensus can play a crucial role in time synchronization for wire-

less sensor networks by enabling all nodes to agree on a common time reference, thereby ensuring consistent timing across the network. By handling multi-hop communication and compensating for unpredictable time delays caused by factors such as interference, backoff, and operating system scheduling, distributed consensus algorithms help maintain synchronization and improve overall network performance. In essence, the function of distributed consensus in time synchronization is to provide a robust and accurate mechanism for achieving and maintaining synchronized time across a wireless sensor network. Another critical research about distributed consensus in wireless sensor networks aims to maximize total network utility [43]. As for the cases without a centralized coordinator among all sensors, the researchers propose an average consensus-based distributed algorithm (ACDA) to distributively schedule the work modes of all sensors using only local information.

1.3.2 Distributed Consensus in Wireless Blockchain System

In recent years, many critical decision-making scenarios and applications such as wireless blockchain [44] [45], Vehicle-to-Everything (V2X) communication [46] [47], and Industrial Internet of Things (IIoT) [48] [49] emerge due to the rapid development of advanced communication technology. Various distributed consensus mechanisms have been adapted to these critical applications and researchers have recognized that the quality of wireless links can significantly influence the performance of distributed consensus and decision-making in these new scenarios, while simultaneously presenting challenges related to power and spectrum efficiency. Therefore, the focus of researchers has shifted towards performance analysis and optimization of the distributed consensus in wireless applications.

To tackle these issues, [50] introduces a comprehensive framework for wireless blockchain networks that takes into account various consensus mechanisms, network topologies, and communication protocols. This research delves into the four primary stages of the blockchain process, laying the groundwork for future optimization efforts. By examining

the connection between blockchain performance and communication resource allocation, factors such as spectrum usage, transmission power, receiver sensitivity, interference, and intentional radio jamming are considered. The analytical and numerical results provided in this study highlight the extent to which communication resource provisioning can affect the overall performance of wireless blockchain networks.

Moreover, [51] and [52] make efforts to improve the performance of distributed consensus in wireless blockchain systems. [51] present a new network model for blockchain-enabled IoT systems, along with a theoretical analysis of its performance and an optimal full function node (FN) deployment strategy. The model addresses key concepts such as blockchain transaction throughput and communication throughput and establishes their mathematical relationship. Performance is analyzed using spatiotemporal domain Poisson point process modeling, and the transaction success rate and overall communication throughput are calculated. They also propose an algorithm for optimal FN deployment under given IoT node density and transaction throughput and analyze the security performance under three typical attacks, with solutions like physical layer security presented to ensure system security.

1.3.3 Independent Model and Framework of Wireless Distributed Consensus

The problem addressed in [52] shifts from the success of individual communication links to ensuring that at least half of communication links (both uplink and downlink) in a multiple-link network are successful, regardless of malicious jamming. The authors first map and model transaction processing into wireless downlink and uplink transmissions, then investigate the transaction success rate of wireless Raft-enabled blockchain. Analytical results demonstrate the relationship between the success probability of consensus, node location, and transmission power.

[53] introduces a probabilistic failure model for CFT protocol, presents analytical reliability results, and proposes the Tolerance Gain indicators to guide wireless distributed consensus deployment, with findings extendable to other consensus mechanisms. [54] presents a novel framework for Connected critical autonomous systems (C-CASs) using a perception-initiative-consensus-action protocol, integrating mechanisms like PBFT and Raft, offering enhanced system reliability and design insights for growing node networks. [55] explores multi-valued fault-tolerant distributed consensus, introducing voting validity for exact consensus output, presents impossibility results and system tolerance bounds, and proposes practical algorithms to prioritize voting validity, with optimizations to boost operation speed and fault tolerance.

1.4 Challenges and Motivations

Driven by advances in the fifth generation (5G) mobile network, industry 4.0, cloud computing and artificial intelligence, etc., the application of cooperative intelligence clusters is expected to grow rapidly in the scenarios of critical industrial sectors within the next decade. According to [56], in such critical application scenarios, the system needs to obtain an end-to-end decision delivery that has latency lower than 1 ms and exceedingly high reliability less than 10^{-9}. Up to now, the majority of wireless communication networks are in centralized schemes, which require the connected nodes to transmit their data to a central control station, where the critical decisions will be made and sent back to actuators for processing. However, a large number of mobile applications will be discretely located in the topology, which means the scheme of centralized systems may hardly be implemented in these applications.

Additionally, the centralized system suffers from an ever-present single point of failure issues [57]. In a centralized communication network, the nodes can only synchronize the information with the central station, which means the system's reliability performance heavily relies on the central station, and the performance can be limited by the worst node connection with the central station. Any wireless communication link failure can cut off the synchronization, which may cause disaster or loss of human life in extreme cases.

The centralized communication system can be very costly since it is well-known that high communication reliability is contradictory to low time latency with constant communication resources. The cost can be unaffordable when the network scales up [58], e.g., on a busy road of autonomous driving scenarios or a smart factory with a large number of mobile robots. As the number of nodes in the network increases, achieving consensus becomes more challenging due to the increased communication overhead, complexity, and convergence time. Wireless nodes are often battery-powered, and energy conservation is a significant concern. The consensus algorithms need to be energy-efficient to prolong the network's lifetime. In a distributed environment, nodes may be malicious or faulty, affecting the consensus process. Robust security mechanisms and trust models are required to ensure the integrity and reliability of the consensus. Therefore, from algorithms and protocols perspectives, an alternative low-cost solution should be investigated on how to improve the overall network's critical decision reliability and latency with low individual link transmission reliability.

Moreover, critical real-time decisions are normally made and processed based on the local data collection from distributed sensors scattered along with the devices. For example, a decentralized approach has been proposed for decision-making in autonomous driving [46], which presents that the local nodes in distributed networks can collect data, make initial decisions, and send global consent to the joint nodes in the distributed network. However, the initiative has to be consented by other vehicles in proximity through a safe and secure consensus protocol since any nonalignment among the vehicles may cause a disaster. In such a distributed system, communication plays a pivotal role in the information

exchange among the connected components. Especially in mobile environments, where the connection among the nodes (e.g., cars, robots, or any other type of equipment) is wireless, the uncertainty of wireless channels and scarcity of communication resources can be critical factors to limit the performance in terms of decision reliability and latency.

Fortunately, such distributed systems can achieve stringent requirements with relaxed communication link reliability by using the distributed consensus to achieve the necessary agreement on a unified state of the network. As one of the most famous distributed systems, blockchain has applied multiple types of distributed consensus to ensure the synchronization among the nodes. Unlike the traditional centralized communication system that requires all communication links to be reliable under a time delay constraint to make correct decisions for the network, distributed consensus in the system can tolerate a certain ratio of link transmission failure, i.e., it can achieve a high-reliability critical decision with relatively low reliable communication links.

To fully leverage the potential of distributed consensus mechanisms in wireless networks, further research is needed to develop efficient algorithms and protocols that address communication overhead, complexity, and convergence time. Additionally, resource efficiency schemes should be considered for nodes in wireless networks to improve the performance of different distributed consensus. By addressing these challenges, distributed consensus mechanisms can provide a viable alternative to centralized systems for critical decision-making in wireless networks.

1.5 Original Contribution

In its quest to delve deep into the intricacies of distributed consensus within wireless networks, this thesis meticulously investigates the distributed consensus algorithm Raft that prominently mentioned in the literature review due to the low communication complexity of its star topology. The focus of research is primarily on achieving a robust consensus in

the face of mission-critical decision-making applications in wireless networks, which are characteristic of communication links that, while potentially offering low latency, might often present challenges in terms of reliability. The original contributions in this thesis can be summarized as follows:

- The research commences with the formulation of a comprehensive link failure model for Raft. This model's essence is to indicate the intricate mathematical interplay between the reliability metrics of communication links and that of consensus. Stemming from this analytical framework, the research introduces a pivotal concept called Reliability Gain. This concept is paramount, as it quantifies the mathematical correlation between the inherent reliability of the consensus and that of the communication link. Intriguingly, the findings suggest that the Reliability Gain follows an approximately linear trajectory concerning the number of nodes. Moreover, the derivation reveals that consensus reliability contradicts delay, providing valuable design guidance for incorporating consensus mechanisms into distributed systems.
- To bolster the efficiency and reliability of the Raft algorithm in wireless networks, my research moves on to the optimization of communication resource allocation through novel methodologies. An optimal power transmission strategy, hinged on the principles of Sequential Quadratic Programming (SQP), is devised to maximize the consensus reliability of Raft. Concurrently, the research dives deep into discerning the optimal bandwidth allocation methodology. Tapping into the power of Particle Swarm Optimization (PSO), the study identifies the perfect bandwidth allocation strategy. The research further extends to unveiling the ideal number of nodes that should be assimilated within the wireless network, ensuring that distributed consensus reliability can be optimized even with the constraints of a given communication resource. The inferences and findings are backed by rigorous analytical proof, cementing their validity.

- The thesis proposes some methods to solve the potential scalability issues of wireless distributed consensus in large-scale wireless networks. An adaptive protocol for distributed consensus Raft in wireless networks is designed to ensure nodes within the network can reach consensus on shared data, even in the presence of network latency and node failures. This adaptive protocol adds a distinct node counting procedure and node entry/exit mechanism to the original protocol of Raft. The proposed adaptive protocol leverages widely-used routing protocols Ad hoc On-Demand Distance Vector Routing (AODV) from Mobile Ad Hoc Networks (MANETs) to efficiently maintain the consistency of node states in Raft-enabled wireless networks. Moreover, the security analysis of sharding, which is considered a practical scheme to improve the throughput of distributed consensus, is present to indicate the security level can be affected by the rate of malicious nodes in both cross-shard transactions and non-cross-shard transactions.

1.6 Thesis Outline

The rest of this thesis is as follows. Chapter 2 starts with a fundamental reliability model of distributed consensus from the perspective of probability of wireless link transmission and the number of nodes. With this consensus reliability model, my research focuses on the performance comparison between centralized consensus and distributed consensus including Raft and Hotstuff BFT. These contents are relevant to "Low Reliable and Low Latency Communications for Mission Critical Distributed Industrial Internet of Things" (The first journal publication in List of Publications) and "Centralized and Distributed Consensus in Wireless Network: An Analytical Comparison" (the first conference publication in List of Publications).

Chapter 3 proposes two optimal communication resource allocation schemes to improve the consensus reliability and latency of Raft. It is produced on top of "Communication Resource Allocation of Raft in Wireless Network" (The second journal publication in List of Publications).

Chapter 4 focus on adapting the original protocol of Raft and security issues in the sharding scheme of distributed consensus in wireless network, which is based on the content in "Adaptive Protocol of Raft in Wireless Network" (the third journal publication in List of Publications) and "Security analysis of Sharding in the Blockchain System" (the second conference publication in List of Publications).

Chapter 5 summarizes the content from Chapter 2 to Chapter 4 and discusses the trend of future research about this thesis.

Chapter 2

Fundamental Model and Analysis of Consensus Reliability

In Chapter 1, we delved deep into the intricacies of mission-critical decision-making within wireless networks. The findings suggest that while making critical decisions, it is possible to meet stringent requirements even if the communication link is not completely reliable. This is achievable through the implementation of a distributed consensus mechanism, which ensures that the majority of nodes in the network converge to a consistent and agreeable state or view of the network. Distributed consensus algorithms, such as Raft and PBFT, have been the backbone of numerous distributed systems requiring a coherent and consistent state among their constituents.

It's vital to understand the essence of these consensus algorithms. The original protocols of Raft and PBFT were primarily developed and fine-tuned for wired communication environments. In such settings, the primary threats to the consensus process arise from node failures or potential malicious attacks. Thus, these protocols are inherently focused on ensuring nodes' robustness to against these challenges. In contrast, the world of wireless networks introduces a unique set of challenges. Even if all nodes in a wireless network function correctly and efficiently, the communication links or channels connecting these nodes can be inherently unstable due to various factors, including environmental disturbances, interference, and physical obstructions. This dynamic nature of wireless communication emphasizes the importance of deriving reliable consensus for critical decision-making

applications, even in the face of potential link failures. It is not just about achieving consensus among normal nodes but ensuring it can be reliably reached even when the communication among nodes is sporadically interrupted. Furthermore, while we know how consensus protocols work in wired networks, the implications of using them in a wireless context remain uncertain, especially concerning communication delays. Any delay can be critical, especially in mission-critical applications, and understanding these nuances becomes vital. Before people try to adopt distributed consensus in real-world wireless scenarios, these nuances and concerns need to be comprehensively addressed.

This chapter aims to pave the way for a more profound understanding of distributed consensus in wireless environments. My research starts with introducing a specialized link failure model for the distributed consensus. This model establishes a mathematical relationship, mapping the correlation between the reliability of the communication link and the reliability of the consensus process itself. Leveraging this relationship, a pivotal concept, termed Reliability Gain, is introduced. The findings suggest that the Reliability Gain is approximately linear in its relationship to the number of nodes in the network. After that, I analytically compare the performance of Raft and Hotstuff BFT with a centralized consensus from the perspective of consensus reliability and latency. This chapter will elucidate these concepts and analysis to provide deeper insights into their implications.

2.1 Reliability of Raft-enabled Wireless Network

Considering there are N nodes in a distributed system with Raft, theoretically, a reliable critical decision requires that over $\frac{N-1}{2}$ followers can receive log entries from the leader and send the confirmed messages back to the leader to achieve the commitment of log replication, which means the number of nodes with both successful downlink and uplink transmissions should be more than half nodes (i.e., $\frac{N-1}{2}$ followers and the leader) to accomplish the consensus progress. We assume that the communication link success rate

is P_l. Mathematically, the consensus success rate of the system P_C is accumulated by the probability of every case in the successful consensus progress, which is in the form of two summations of probability in a binomial distribution. The accurate probability of success consensus P_C in the distributed communication system can be derived in the following equation,

$$P_C = \sum_{i=\frac{N-1}{2}}^{N-1} \binom{N-1}{i} {P_l}^i (1-P_l)^{N-1-i} \sum_{j=\frac{N-1}{2}}^{i} \binom{i}{j} {P_l}^j (1-P_l)^{i-j}. \tag{2.1}$$

The first summation represents the probability that the majority of followers can download the log entry from the leader. The second summation equals the probability that the majority of followers can upload their confirmation back to the leader. Because the downlink transmission happens before uplink transmission in Raft, the number of successful uplink transmissions is never larger than the number of successful downlink transmissions. Therefore, the probability of a successful consensus term is the product of these two summations. It is worth mentioning that consensus success rate P_C increases monotonically with the number of nodes N. Though this property cannot be revealed by (2.1) straightforwardly, however, our following simplification and the simulation result can show this property explicitly.

Remark 1. *According to (2.1), to satisfy the most stringent reliability requirement, i.e., the consensus failure rate* $1-P_C$ *is less than* 10^{-9}*, the nodes number* N *should not be less than* 69, 31, 12, 5*, when the link success rate* P_l *is 90%, 95%, 99% and 99.9%, respectively.*

The remark. 1 indicates the fact that even if the link success rate P_l is undesirable, the consensus success rate of Raft can still be improved to the requirement, and the number of nodes N can influence this reliability improvement. Therefore, we introduce a parameter called *Reliability Gain* (also can be interpreted as reliability amplification factor)

Definition 1. *The Reliability Gain is the ratio of consensus failure rate* $1-P_c$ *and link failure rate* $1-P_l$ *in logarithm*

to represent the quantitative relationship between the reliabilities of consensus and communication link.

Theorem 1. *When the link success rate P_l is reasonably large*[1]*, it has a linear relationship with consensus failure rate $1-P_C$ in logarithm*

$$log(1-P_C) = k \cdot log(1-P_l) + h, \tag{2.2}$$

where the Reliability Gain $k = \frac{N+1}{2}$ and the intercept $h = log(\binom{N-1}{\frac{N-3}{2}}) + \Delta h$, with Δh being given in Table 2.1.

Proof. See Appendix A □

Theorem 1 indicates the linear relationship between consensus failure rate $log(1-P_C)$ and link failure rate $log(1-P_l)$ after the simplification. With fixed link reliability, the increasing N rises up the consensus reliability, which proves the increasing monotonicity of consensus success rate P_C with the nodes number N. Compared to (2.1), this equation shows a simple relationship between link reliability and consensus reliability. Thus, this linear relationship can provide practical methods to estimate the maximum value of consensus reliability with given link reliability. The simulation result shows that the consensus failure rate $log(1-P_C)$ satisfies the linear relationship in (2.2) when P_l is as low as 90%.

Δh is derived by the gap between the intercept h from the simulation and the item $log(\binom{N-1}{\frac{N-3}{2}})$ from the simplification of (2.1). Table 2.1 shows the estimated Δh when nodes

Table 2.1: Estimated Δh

N	Δh	N	Δh	N	Δh
5	0.472	6	0.704	7	0.797
8	1.081	9	1.214	10	1.509
11	1.668	12	1.965	13	2.144
14	2.441	15	2.635	16	2.931
17	3.136	18	3.431	19	3.645

1. Our results show that $P_l = 90\%$ is large enough to make the conclusion, while this communication link reliability can be achieved in most of the communication environments.

number N increases from 5 to 19. Δh remains constant with a fixed nodes number N. To increase the reliability gain in a consensus communication system, more nodes can be added to participate in the consensus network.

A wireless communication model [2], which aims to analyze the packet error probability of the wireless short package transmissions in URLLC [60], is used to define the link transmission failure and find out the relationship between consensus success rate P_C and the consensus latency T. We assume it is caused by downlink and uplink transmission delay. I.e., Raft consensus latency T is only composed of communication transmission delay to show the communication impacts on the overall consensus latency. The link failure rate $1-P_l$ used in (2.1) and (2.2) can be written as a function of T as follow,

$$1-P_l = f_Q\left(\frac{B\frac{T}{2N}(C-R)+\frac{\log_2(B\frac{T}{2N})}{2}}{(B\frac{T}{2N})^{\frac{1}{2}}\log_2(e)}\right), \tag{2.3}$$

where B is the available spectrum bandwidth. R and C are the uplink or downlink transmission rate and channel capacity, respectively. f_Q refers to the Q-function, which is the tail distribution function of the standard normal distribution [61]. Note that here both uplink and downlink transmissions are assumed to be time divisioned, i.e., given the overall consensus delay, T, each transmission can have $t=\frac{T}{2N}$ transmission internal since there are N transmissions in both uplink and downlink. Therefore, with a constant N, the increasing consensus delay T can provide more time t for each link transmission, which intuitively can reduce the link failure rate $1-P_l$. By substituting equation (2.3) into equation (2.1) or (2.2), the relationship of consensus failure rate $1-P_C$ with the latency T can be obtained. The contradiction of consensus failure rate $1-P_C$ and time delay T can be proved in mathematics. By calculating the derivative of the variable $Q=\frac{B\frac{T}{2N}(C-R)+\frac{\log_2(B\frac{T}{2N})}{2}}{(B\frac{T}{2N})^{\frac{1}{2}}\log_2(e)}$ in Q-function,

$$\frac{\partial Q}{\partial T}=\frac{\frac{B}{2N}(C-R)-\frac{1}{2T}\log_2(\frac{T}{2N}B)+\frac{1}{T}(ln2)}{2\sqrt{\frac{T}{2N}B}\log_2(e)}. \tag{2.4}$$

2. With the constraints on high reliability and low latency of distributed consensus, the block length is short and small packet size is adopted [59]. In addition, the size of log entries transmitted by node is usually small in Raft. In such a scenario, the impact of decoding error cannot be ignored.

the derivative $\frac{\partial Q}{\partial T}$ in equation always keeps positive, which means the variable Q increases monotonically along with T. Based on the decreasing monotonicity of Q-function $f_Q(*)$ along with Q and the increasing monotonicity of P_C along with P_l, the time delay of consensus T and consensus reliability $1-P_C$ are contradictory. [3]

According to the conclusion of Reliability Gain, the consensus reliability P_C increases monotonically with N. However, given fixed consensus delay T, increasing node number will also result in a shorter transmission time $t = \frac{T}{2N}$ for each link, thus causing a smaller P_l, which may turn out a less reliable consensus according to (2.1) or (2.2). Thus, it is expected that there is an optimal N to achieve maximum consensus reliability.

2.2 Performance Comparison between Centralized and Distributed Consensus

Based on the characteristics of stability and instantaneity in the mission-critical decision, the reliability and time of decision delivery are the most conspicuous factors in the performance of the centralized and distributed consensuses. For an illustration, paradigms in the IoT network's decision-making are presented in Fig. 2.1. Compared with the Perception-Collection-Decision-Action (PCDA) paradigm deployed in the critical decision execution of centralized consensus, a novel Perception-Initiative-Consensus-Action (PICA) scheme that the local nodes collect data, make initial decisions, and accomplish the consent among the majority of joint nodes before the execution is purposed. The initial decision can be determined based on the local informative sensing and potentially combined with some machine learning algorithms [62], [63]. A request is made by the node

3. The scheme of wireless transmission in this chapter is assumed in time-division. This assumption will switch to bandwidth-division in the analysis of following chapters. Yet, the conclusion about the contradictory between consensus reliability and latency is still valid.

based on the initial local decision and then sent to the network for a joint global decision (i.e., Consensus), where only the consented joint decision will be sent to the actuators for execution. For example, the robots that collaboratively coordinate movements to perform tasks can use the PICA paradigm through emergent action [64].

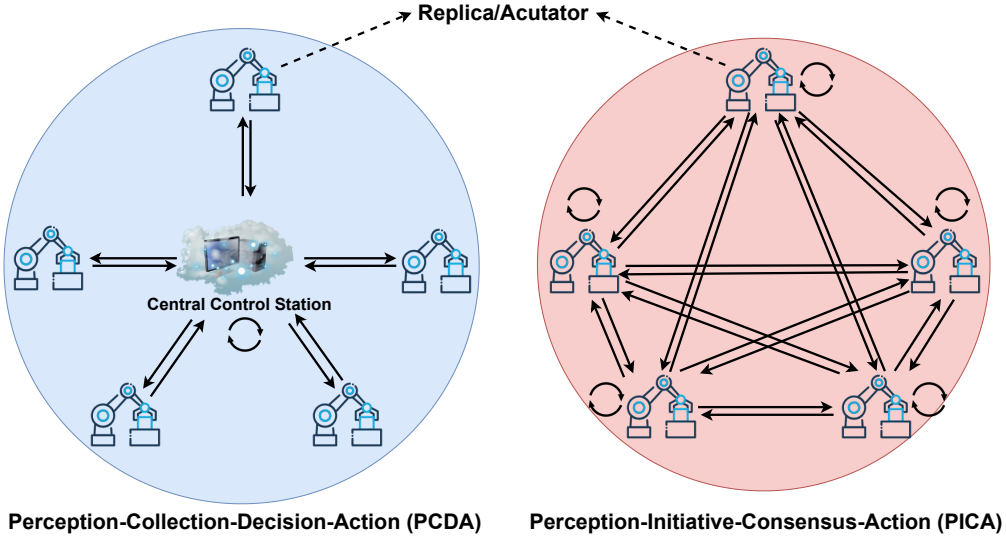

Figure 2.1: Communication topology for centralized and distributed consensus networks

To fairly compare the performance of centralized and distributed consensus, the analysis in this chapter focuses on the cost of time and consensus reliability caused by the wireless connection during the consensus and synchronization. The link failure in wireless communication channels can cause breakdowns that are identical to crashing faults and Byzantine faults in functionality. Therefore, the link reliability, which is influenced by the allocation of communication resources, can determine the consensus reliability [65]. Specifically, a longer transmission time can improve the reliability of link transmission in a single channel and eventually improve the reliability of the consensus and state synchronization of all replicas in the network. Meanwhile, the higher link reliability may cause the overall latency to reach the threshold of requirement.

Unlike the original consensus reliability noted in (2.1), a concept called full consensus reliability is implemented in the analysis in this chapter, which is defined as the probability P_F of all replicas in the network having completed successful link transmissions to update their state in one term of consensus. The performance analysis of all consensus protocols

in this article is related to the overall time latency T and the probability of successful full consensus P_F. We assume the length of time slot in a downlink broadcasting is equivalent to the length of time slot in an uplink unicasting, which is unified as t. Table 2.2 shows all common parameters used in this chapter.

Table 2.2: Common parameters used in centralized and distributed consensus

Notation	Definition
N	Number of nodes in the centralized / distributed consensus
r	Total rounds of synchronization
f	Number of links failed replicas in a distributed consensus
P_C	Probability of successful consensus
M_t	Mean of the number of time slots cost
P_F	Probability of successful full consensus
t	Length of the time slot for one link transmission
T	Overall time latency for a consensus view
P_l	Probability of successful link transmission in consensus
TP	Log throughput
$P_{ls}(r)$	Probability of successful link transmission in the r_{th} round synchronization

2.2.1 Full Reliability and Latency of Centralized Consensus

In the centralized consensus, to achieve full replicas communication success, the central base station should receive the acknowledgment packet from all replicas to confirm that the replicas have acquired the successful state synchronization messages. The probability $P_{TD}(0)$ that i_0 of total N replicas can successfully receive the message from CN during the centralized consensus can be derived by the binomial distribution of link reliability P_l

$$P_{TD}(0) = \binom{N-1}{i_0} {P_l}^{i_0}(1-P_l)^{N-1-i_0}. \tag{2.5}$$

Similarly, the probability that the CN successfully receives j_0 of i_0 messages from replicas in the centralized consensus with link reliability P_l

$$P_{TU}(0) = \binom{i_0}{j_0} {P_l}^{j_0}(1-P_l)^{i_0-j_0}. \tag{2.6}$$

The number of nodes that have not been synchronized successfully after h rounds of synchronization can be denoted as $j_F(h)$,

$$j_F(h) = N - \sum_{k=0}^{h} j_k. \tag{2.7}$$

The equation (2.8) and (2.9) indicates the probability $P_{TD}(r-1)$ that i_{r-1} of $N-1-j_F$ successfully receive the downlink message and the probability $P_{TU}(r-1)$ that the CN successfully receive j_{r-1} of i_{r-1} acknowledgement messages after the $(r-1)_{th}$ round of synchronization.

$$P_{TD}(r-1) = \binom{N-1-j_F(r-2)}{i_{r-1}} P_{ls}(r-1)^{i_{r-1}} (1-P_{ls}(r-1))^{N-1-j_F(r-2)}, \tag{2.8}$$

$$P_{TU}(r-1) = \binom{i_{r-1}}{j_{r-1}} P_{ls}(r-1)^{j_{r-1}} (1-P_{ls}(r-1))^{i_{r-1}-j_{r-1}}. \tag{2.9}$$

In the final round of synchronization, $P_T(r)$ represents the probability that the CN receives the acknowledgment messages from all of $N-1-j_F(r-1)$ replicas that failed in previous rounds, which can ensure all replicas states have been updated successfully by the CN after r rounds of synchronization.

$$P_T(r) = (P_{ls}(r))^{2(N-1-j_F(r-1))}. \tag{2.10}$$

Therefore, the probability P_F that the states of all replicas have been correctly updated by the CN after the centralized consensus and r rounds of synchronization is accumulated by the probability P_{TD} and P_{TU} in every round of synchronization, which represents the full consensus reliability of the centralized consensus.

$$P_F = \sum_{i_0=0}^{N-1} (P_{TD}(0) \sum_{j_0=0}^{i_0} (P_{TU}(0) \cdots \sum_{i_{r-1}=0}^{N-1-j_F(r-2)} (P_{TD}(r-1) \sum_{j_{r-1}=0}^{i_{r-1}} (P_{TU(r-1)}(P_T(r)))) \cdots)). \tag{2.11}$$

Because of the unified length of time slot t, the overall time latency T is consequently proportional to the number of time slots M_t that have been spent in protocols. M_t can be accumulated by the number of time slots cost in the consensus and r rounds of synchronizations.

$$\begin{aligned} M_{t-b} = & \sum_{i_0=0}^{N-1} P_{TD}(0) i_0 + \sum_{i_0=0}^{N-1} (P_{TD}(0) \sum_{j_0=0}^{i_0} (P_{TU}(0) \sum_{i_1=0}^{N-1-j_0} (P_{TD}(1) i_1))) + \cdots + \\ & \sum_{i_0=0}^{N-1} (P_{TD}(0) \sum_{j_0=0}^{i_0} (P_{TU}(0) \cdots \sum_{i_r=0}^{N-1-j_F} (P_{TD}(r) i_r) \cdots)) \\ & + N - 1 + \sum_{i_0=0}^{N-1} (P_{TD}(0) \sum_{j_0=0}^{i_0} (P_{TU}(0)(N-1-j_0))) \\ & + \sum_{i_0=0}^{N-1} (P_{TD}(0) \sum_{j_0=0}^{i_0} (P_{TU}(0) \cdots \sum_{j_{r-1}=0}^{i_{r-1}} (P_{TU}(r)(N-1-j_F(r-1)))) \cdots)), \end{aligned} \tag{2.12}$$

The overall latency T is the length of all time slots cost by broadcasting and unicasting in the protocol and each time slot has the constant length t

$$T = tM_t. \tag{2.13}$$

The equation (2.12) and (2.13) show that the overall latency T depends on the round of synchronization r, link reliability P_l and P_{ls}.

2.2.2 Full Reliability and Latency of Raft and Hotstuff BFT

The communication scheme of Raft is identical to the centralized consensus, so the probabilities derivation of Raft consensus are comparable to the equation (2.5) and (2.6), which are derived in equation (2.14) and (2.15)

$$P_{RD} = \binom{N-1}{i_0} {P_l}^{i_0} (1-P_l)^{N-1-i_0}, \tag{2.14}$$

$$P_{RU} = \binom{i_0}{j_0} {P_l}^{j_0} (1-P_l)^{i_0-j_0}. \tag{2.15}$$

According to the link failure model in (2.1), the probability of successful consensus P_C represents that the leader can receive over $\frac{N-1}{2}$ uplink messages from followers through the consensus, which can be accumulated by P_{RD} and P_{RU} as i_0 and j_0 is not less than $\frac{N-1}{2}$

$$P_C = \sum_{i_0=\frac{N-1}{2}}^{N-1} (\binom{N-1}{i_0} {P_l}^{i_0}(1-P_l)^{N-1-i_0} \sum_{j_0=\frac{N-1}{2}}^{i_0} (\binom{i_0}{j_0} {P_l}^{j_0}(1-P_l)^{i_0-j_0})). \tag{2.16}$$

The followers who fail in consensus can be modified through the synchronization provided by the followers that attain the successful consensus. The probability P_{SF} that after r rounds of synchronization, there are still i_1 of $N-j_0-1$ followers have not received the downlink synchronization messages from other followers is performed in (2.17)

$$P_{SF} = \binom{N-j_0-1}{i_1} (\prod_{k=1}^{r}(1-P_l(k)))^{i_1})(1-(\prod_{k=1}^{r}(1-P_l(k))))^{N-j_0-i_1-1}. \tag{2.17}$$

The probability P_F represents that all $N-j_0-1$ consensus-failed followers receive the broadcasting state message from other followers with successful consensus after certain rounds of synchronization, which can be derived as

$$P_F = P_C - \sum_{i_0=\frac{N-1}{2}}^{N-1} (P_{RD} \sum_{j_0=\frac{N-1}{2}}^{i_0} (P_{RU} \sum_{i_1=1}^{N-j_0-1} (P_{SF})). \tag{2.18}$$

In the broadcasting downlink scheme, the number of time slot M_t cost in Raft protocol is accumulated by the number of unicasting response messages and the number of times to broadcast,

$$M_t = \sum_{i_0=\frac{N-1}{2}}^{N-1} P_{RD} i_0 + r + 1. \tag{2.19}$$

In the unicasting downlink scheme, the number of time slots costs in each round of downlink is equivalent to the number of channels used in downlink transmissions. Therefore, the total time slots cost M_t in the protocol can be presented in (2.20)

$$M_t = \sum_{i_0=\frac{N-1}{2}}^{N-1} P_{RD} i_0 + N - 1 + r \sum_{i_0=\frac{N-1}{2}}^{N-1} (P_{RD} \sum_{j_0=\frac{N-1}{2}}^{i_0} (P_{RU}(N-1-j_0))). \tag{2.20}$$

where $N-1-j_0$ is the number of channels in r rounds of synchronizations. The overall latency T of broadcasting and unicasting downlink schemes in Raft can be both expressed in (2.13).

2.2.3 Reliability and Latency of Hotstuff BFT

Since the number of new-view messages in one view of Hotstuff BFT is identical to the number of backup replicas $N-1$, it is reasonable to assume that the link transmission for new-view messages in the prepare phase is reliable. The successful consensus rate should concern the successful link rate P_l after new-view messages. We assume the numbers of replicas that receive the downlink message are i_p, i_{pc}, and i_c in the prepare phase, pre-commit phase, and commit phase. The numbers of response messages that the leader receives in the prepare phase, pre-commit phase, and commit phase are j_p, j_{pc}, and j_c. The probability P_{pd} that i_p replicas receive the downlink prepare message from the leader in Hotstuff BFT should be

$$P_{pd} = \binom{N-1}{i_p} P_l^{i_p} (1-P_l)^{N-1-i_p}. \tag{2.21}$$

The probability P_{pu} that the leader receives j_p uplink responses from a backup replica in the prepare phase should be

$$P_{pu} = \binom{i_p}{j_p} P_l^{j_p} (1-P_l)^{i_p-j_p}. \tag{2.22}$$

The pre-commit phase and commit phase have the same downlink and uplink scheme as prepare phase in Hotstuff BFT protocol, which means the probability P_{pcd} and P_{cd} that i_{pc} and i_c replicas receive the downlink message in pre-commit and commit phase is in the same formation as P_{pd}. The formation of the probability P_{pcu} and P_{cu} that j_{pc} and j_c response messages are received by the leader also follows this property.

The link failure model of Hotstuff indicates that $N - j_c$ replicas fail in the consensus progress, the probability P_S that every replica from this group should receive at least one synchronization message after the consensus to achieve the full consensus success.

$$P_S = 1 - \sum_{i=1}^{N-j_c} \binom{N-j_c}{i} (\prod_{k=1}^{r} (1 - P_{ls}(k)))^i)(1 - (\prod_{k=1}^{r} (1 - P_{ls}(k))))^{N-j_c-i}). \tag{2.23}$$

The probability of successful full consensus P_F can be derived by all the deduced probabilities in the subsection 2.2.3 with the condition that the number of correct response messages received by the leader is not less than $2f$ in every phase.

$$P_F = \sum_{i_p=2f}^{N-1} (\sum_{j_p=2f}^{i_p} (\sum_{i_{pc}=2f}^{j_p} (\sum_{j_{pc}=2f}^{i_{pc}} (\sum_{i_c=2f}^{j_{pc}} (\sum_{j_c=2f}^{i_c} P_{cu} P_S) P_{cd}) P_{pcu}) P_{pcd}) P_{pu}) P_{pd}. \tag{2.24}$$

The minimum number $2f$ of replicas that achieve the consensus should be less than $\frac{2(N-1)}{3}$ in the link failed model of Hotstuff BFT.

The communication scheme in every phase of Hotstuff BFT is identical to the scheme in the Raft consensus protocol. Therefore, the time slots cost in three phases M_{t-p}, M_{t-pc} and M_{t-c} of Hotstuff BFT with broadcasting downlink scheme can be calculated in the same way of (2.19), which are indicated in (2.25), (2.26) and (2.27)

$$M_{t-p} = 1 + \sum_{i_p=0}^{N-1} P_{pd} i_p, \tag{2.25}$$

$$M_{t-pc} = 1 + \sum_{i_p=2f}^{N-1} (\sum_{j_p=2f}^{i_p} (\sum_{i_{pc}=0}^{j_p} P_{pcd} i_{pc}) P_{pu}) P_{pd}, \tag{2.26}$$

$$M_{t-c} = \sum_{i_p=2f}^{N-1} (\sum_{j_p=2f}^{i_p} (\sum_{i_{pc}=2f}^{j_p} (\sum_{j_{pc}=2f}^{i_{pc}} (\sum_{i_c=0}^{j_{pc}} P_{cd} i_c) P_{pcu}) P_{pcd}) P_{pu}) P_{pd} + 1. \tag{2.27}$$

The overall time latency T in Hotstuff BFT is the sum of all time slots that cost in three consensus phases and the stage of synchronization,

$$T = (M_{t-p} + M_{t-pc} + M_{t-c} + r)t. \tag{2.28}$$

When the downlink scheme of Hotstuff is unicasting, the time slot cost is the sum of the channels used in both downlink and uplink through the whole view, which is more than the cost of the broadcasting scheme in this consensus protocol.

$$M_{tp} = N - 1 + \sum_{i_p=0}^{N-1} P_{pd} i_p, \tag{2.29}$$

$$M_{tpc} = \sum_{i_p=2f}^{N-1} \left(\sum_{j_p=0}^{i_p} P_{pu} j_p \right) + \sum_{i_p=2f}^{N-1} \left(\sum_{j_p=2f}^{i_p} \left(\sum_{i_{pc}=0}^{j_p} P_{pcd} i_{pc} \right) \right), \tag{2.30}$$

$$\begin{aligned} M_{tc} = & \sum_{i_p=2f}^{N-1} \left(\sum_{j_p=2f}^{i_p} \left(\sum_{i_{pc}=2f}^{j_p} \left(\sum_{j_{pc}=0}^{i_{pc}} (P_{pcu} j_{pc}) P_{pcd} \right) P_{pu} \right) P_{pd} + \right. \\ & \left. \sum_{i_p=2f}^{N-1} \left(\sum_{j_p=2f}^{i_p} \left(\sum_{i_{pc}=2f}^{j_p} \left(\sum_{j_{pc}=2f}^{i_{pc}} \left(\sum_{i_c=0}^{j_{pc}} P_{cd} i_c \right) P_{pcu} \right) P_{pcd} \right) P_{pu} \right) P_{pd}. \end{aligned} \tag{2.31}$$

The link failed model of Hotstuff consensus reveals that there are $N-1-j_c$ replicas that have failed in the consensus. The number of channels used in one round of synchronization should correspond to this number of failed replicas. Thus, the number of time slots cost $M_t s$ in one round of synchronization can be presented in (2.32)

$$M_{ts} = \sum_{i_p=2f}^{N-1} \left(\sum_{j_p=2f}^{i_p} \left(\sum_{i_{pc}=2f}^{j_p} \left(\sum_{j_{pc}=2f}^{i_{pc}} \left(\sum_{i_c=2f}^{j_{pc}} \left(\sum_{j_c=2f}^{i_c} P_{cu} (N-1-j_c) \right) P_{cd} \right) P_{pcu} \right) P_{pcd} \right) P_{pu} \right) P_{pd}. \tag{2.32}$$

The overall time slots cost M_t through the Hotstuff BFT consensus with a downlink unicasting scheme and the stage of synchronization will be

$$T = (M_{tp} + M_{tpc} + M_{tc} + r M_{ts}) t. \tag{2.33}$$

The comparison of the overall latency of Hotstuff BFT protocol in equation (2.28) and the overall latency of Raft protocol in equation (2.19) shows that basic Hotstuff protocol costs much more time slots than Raft because of the two additional consensus phases in the Hotstuff BFT. Such long latency limits the throughput of the whole communication network. Yet, the chain Hotstuff protocol with pipeline structure is presented to solve

this throughput issue. In Fig. 1.5, if there are $3K$ logs processed by the chain Hotstuff protocol and all of them successfully achieve the consensus in three phases, the overall latency can cost $3K+3$ slots, and each slot represents the average time latency in one phase of Hotstuff. Therefore, the log throughput of chain Hotstuff can be derived

$$TP = \frac{K}{K+1}. \tag{2.34}$$

When the number of successful logs processed $3K$ is large, the log throughput TP in (2.34) will converge to 1, which is equivalent to the throughput of Raft protocol with a single phase. Therefore, the chain Hotstuff can solve this long latency issue in Hotstuff BFT.

2.3 Simulation Results

Simulations are conducted to validate the proposed consensus communication model and its derivations. The location of nodes are set to be evenly distributed in a circle with constant radius and density. The given bandwidth for link transmission B is set as 18 kHz, and the SINR (signal-to-interference-plus-noise ratio) is set to 10 dB [66]. The jammer sends interference signals continuously at both uplink and downlink frequency bands. The uplink and downlink capacity R is assumed 50% of the channel capacity, which is calculated by $C = \log(1+SINR)$.

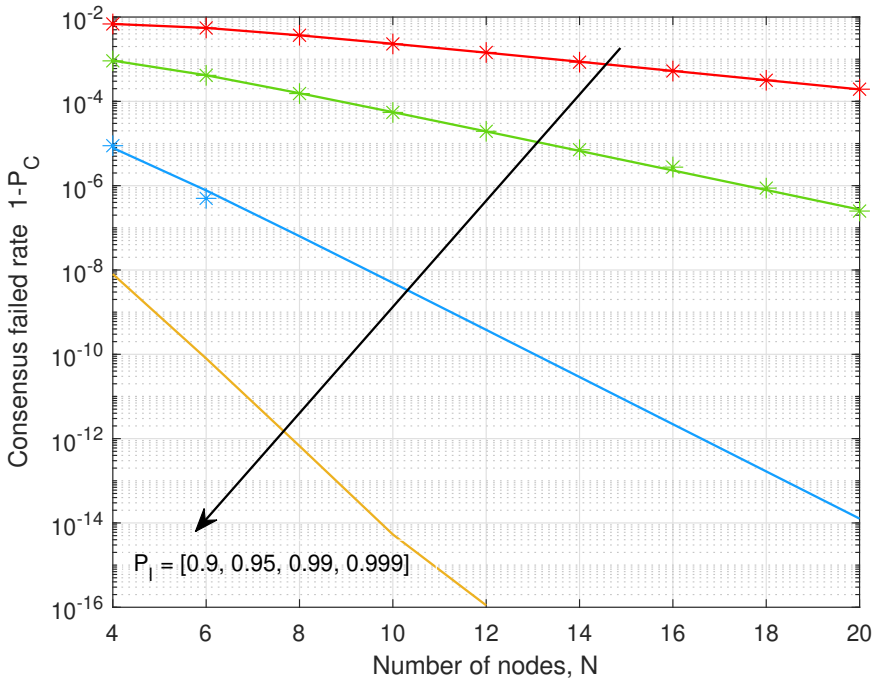

Figure 2.2: Consensus failure rate $1-P_C$ vs. Nodes number N [Lines: analytical result from equation (2.1), Asterisks: simulated]

Fig. 2.2 indicates the consensus failure rate with several constant link reliability P_l and an increasing number of nodes N in Raft-enabled wireless network. The consensus failure rate $1-P_C$ declines as N increases with relatively low communication link success rate $P_l = 90\%, 95\%, 99\%, 99.9\%$ respectively. The simulated results (in asterisks) of the consensus failure rate $1-P_C$ is overlapped to their analytical curves (in lines) when the link success rate $P_l = 90\%$ and 95%, which proves the correctness of the equation (2.1). The analytical curves show the property that the consensus success rate P_C increases monotonically with N. Because the consensus failure rate is extremely low for a larger P_l and MATLAB compute power is limited, the simulated result of consensus failure rate $1-P_C$ cannot be completely presented in Fig. 1 when P_l is 99% and 99.9%.

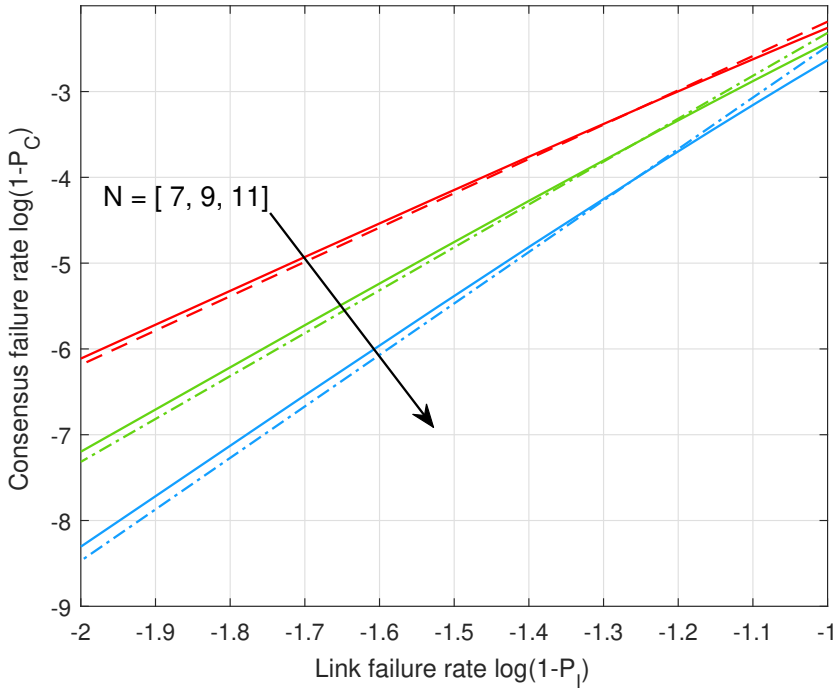

Figure 2.3: Consensus failure rate $\log(1-P_C)$ vs. Link failure rate $\log(1-P_l)$ [Solid lines: analytical results from equation (2.1), Broken lines: simplified analytical results from equation (2.2)]

Fig. 2.3 shows the consensus reliability tendency along with the link success rate P_l. The analytical result represents the original consensus reliability $1-P_C$ in logarithm in (2.1). The simplified result represents the consensus failure rate $\log(1-P_C)$ in (2.2). Analytical lines and simplified lines are highly matched, which supports the accuracy of the linear relation in (2.2). The slopes of lines are equivalent to the value of *Reliability Gain* $k = \frac{N+1}{2}$, which become steeper when the nodes number N rises up. The result shown here suggests that we can use a simplified model to guide the real deployment of Raft for distributed systems.

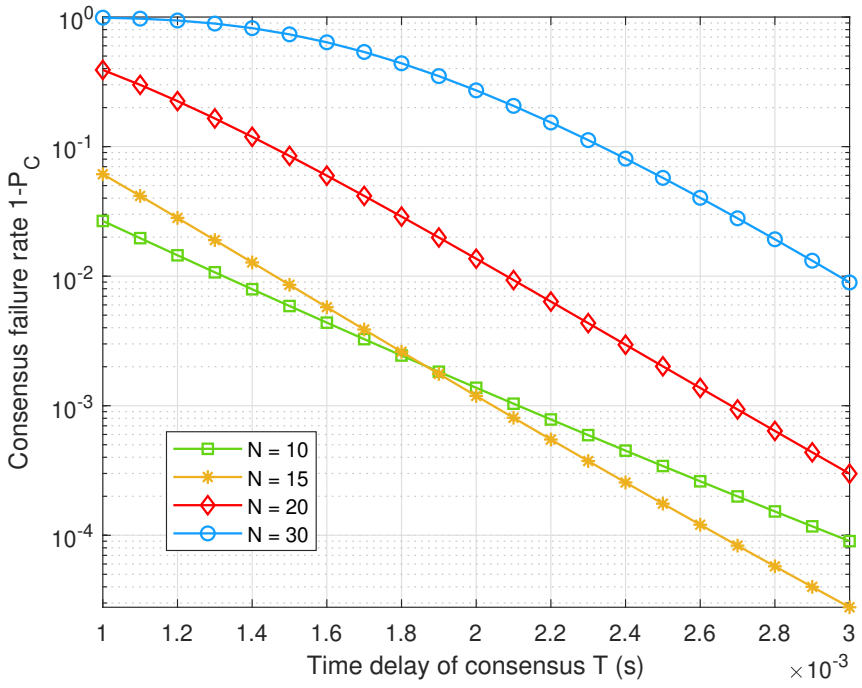

Figure 2.4: Consensus failure rate $(1-P_C)$ vs. Consensus delay T

The simulation in Fig. 2.4 reveals the contradiction between consensus reliability $1-P_C$ and consensus delay T. Four curves correspond to different nodes number $N = 10, 15, 20, 30$ respectively. All curves show that with the constant N, the consensus reliability $1-P_C$ reduces when the time delay T rises, which proves the contradiction of the consensus reliability and time latency. The tendency of the consensus failure rate at $N = 15$ drops more dramatically than the consensus failure rate at $N = 10$, along with the increase of time delay, which causes the interception of two curves. It implies that the consensus reliability does not have monotonicity along with N if consensus delay T's effect on the link transmission reliability is considered. Therefore, further investigation of this phenomenon is performed.

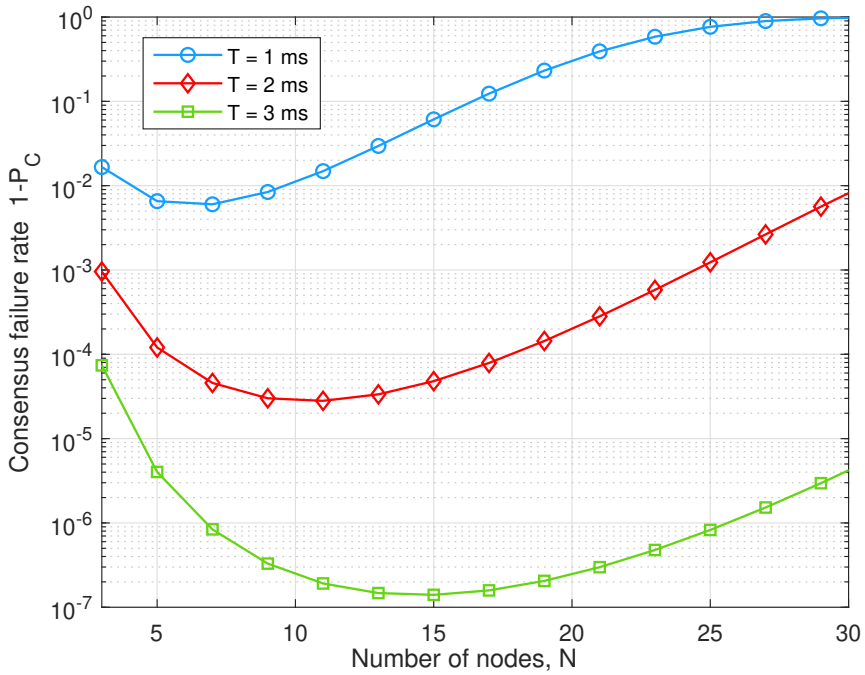

Figure 2.5: Consensus failure rate $(1-P_C)$ vs. Nodes number (N)

Fig. 2.5 indicates the change of consensus failure rate $1-P_C$ in the wireless link model of 2.3 along with N and a constant consensus time latency T. The curves show that the consensus reliability $1-P_C$ fluctuates when nodes number increases, and there is maximum consensus reliability. By modifying the time delay in the consensus system, the maximum value of the reliability curve will be shifted to a higher value with the larger corresponding N. The reason for this phenomenon is that the consensus failure rate follows the monotonicity in (2.1) when N is small; when N becomes large, given the fact that communication resource (i.e., the communication time T) is limited, the time latency in each link transmission will be reduced, and $1-P_l$ will increase dramatically along N based on the property of Q function in (2), which causes the rise of $1-P_C$. Therefore, N has both positive and negative effects on consensus reliability. The shifting of curves indicates that the optimization of consensus reliability by allocating communication resources can be implemented to reach the requirements of different scenarios.

The analysis of consensus reliability in the Raft-enabled wireless network concludes that with the constant communication link reliability, the consensus reliability in Raft increases monotonically along with the number of nodes. The relationship of consensus reliability with communication link reliability is interpreted in a linear form for simplicity. Meanwhile, the simulation results show that the time latency is contradictory to the consensus reliability in Raft.

Numerical results are conducted to compare the full consensus reliability P_F and the overall time latency T among the centralized consensus and two distributed consensuses mentioned in this chapter. The link reliability P_l of consensus is assumed as 99.9%. The synchronization is a type of retransmission strategy given by replicas, which can enhance the signal-to-noise ratio (SNR) of link reliability P_l in the consensus. We assume the link reliability of the synchronization P_{ls} increases by $2dB$ after every round of synchronization. It should be noted that the value affects not only the communication results but the derivations.

Fig. 2.6 indicates the failure rate $1-P_F$ of full consensus and the overall latency T caused by the link transmission within one view of the centralized consensus and distributed consensus Raft. The downlink scheme is broadcasting and the number of nodes N ranges from 4 to 32 with an interval of 4. Without any rounds of synchronization, the curve that represents the reliability and latency of Raft exactly coincides with the curve of the centralized consensus because the communication scheme in the consensus of Raft is identical to the scheme of the centralized consensus.

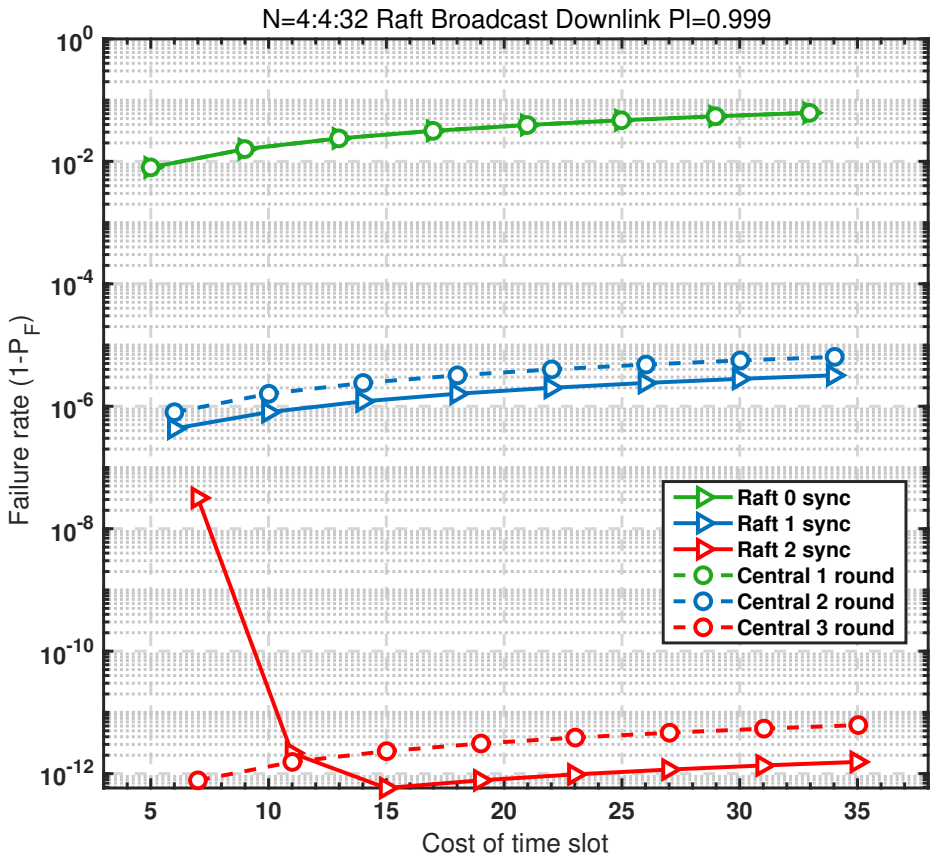

Figure 2.6: Performance of centralized consensus vs Performance of distributed consensus Hotstuff (Broadcasting downlink)

The curves in Fig 2.7 show that when the network has a constant number of nodes, the reliability P_F is identical to the rate in broadcasting the downlink scheme when the downlink scheme is unicasting. Yet, the unicasting downlink scheme will take more time slots to complete the log replication than the broadcasting downlink scheme. Without any rounds of retransmission and synchronization, the curve that represents the reliability and latency in the Raft protocol exactly coincides with the curve of the centralized network because the communication scheme in the consensus of Raft is identical to the scheme of the centralized communication network.

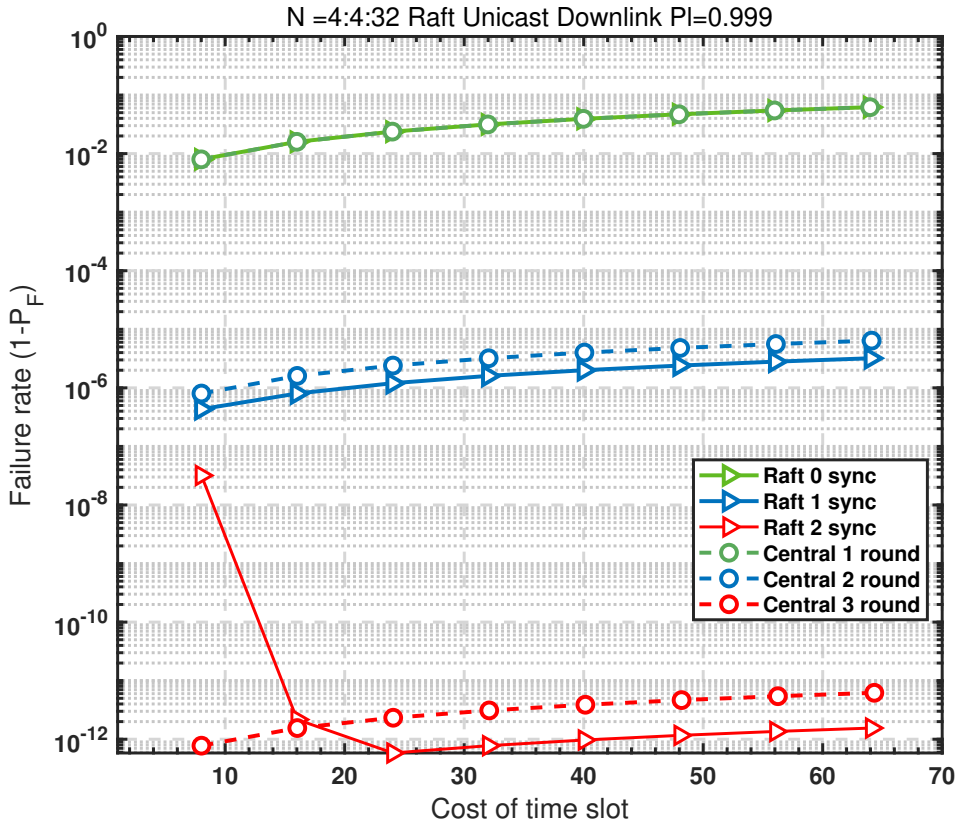

Figure 2.7: Performance of centralized network vs Performance of decentralized network with Raft (Unicasting downlink)

When one or more rounds of synchronization are implemented, there is a considerable drop in the failure rate $1-P_F$ of both protocols. When two consensuses have the same number of replicas, the distributed consensus Raft tends to cost fewer time slots and has a lower failure rate $1-P_F$. When two rounds of synchronization are implemented, the result shows a conspicuous result that the failure rate $1-P_F$ of Raft in the small-scale network ($N < 12$) is higher than the centralized consensus. This result is caused by the high failure rate of the consensus $1-P_C$ when the number of replicas N is small.

Fig. 2.8 presents the failure rate $1-P_F$ and latency T comparison between the distributed consensus Hotstuff BFT and the centralized consensus in one view. The downlink scheme is broadcasting and the number of nodes N in the network ranges from 4 to 28 with an interval of 3.

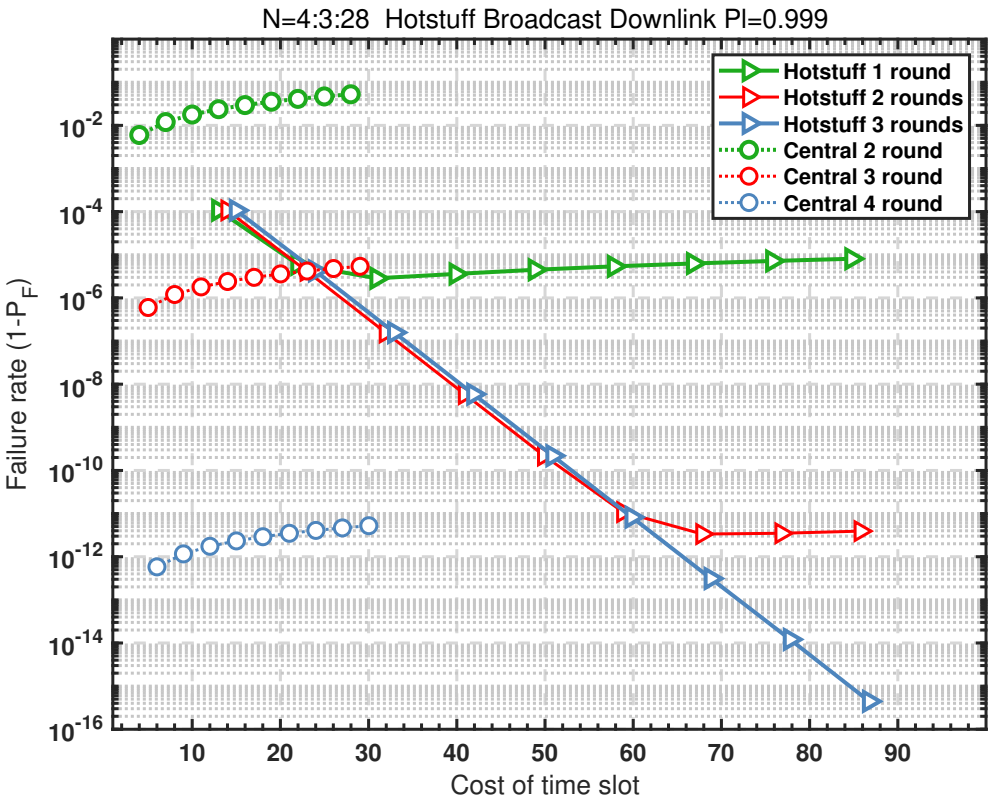

Figure 2.8: Performance of centralized consensus vs Performance of distributed consensus Hotstuff (Broadcasting downlink)

In Fig. 2.9, the failure rate $1 - P_F$ and latency T comparison between a decentralized network with Hotstuff BFT and a centralized network in one term of view are shown when the downlink scheme is unicasting. The number of nodes N engaged in the communications is identical to the setting as the downlink scheme is broadcasting.

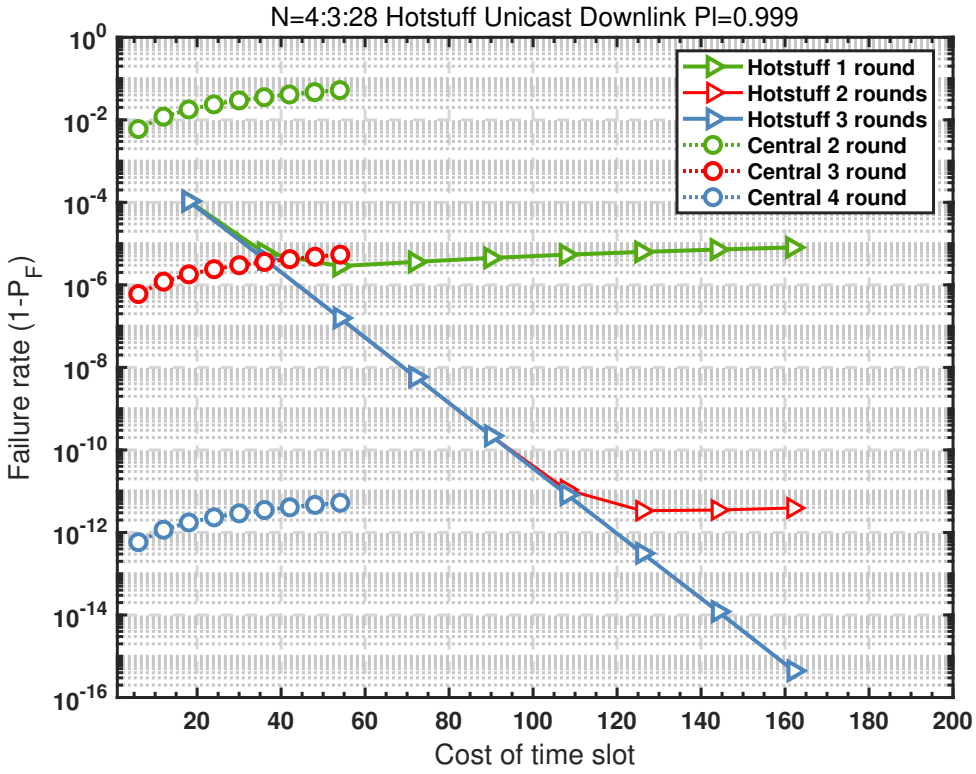

Figure 2.9: Performance of centralized network vs. Performance of decentralized network with Hotstuff protocol (Unicasting downlink)

When one round of synchronization is implemented ($r = 1$) and link failure rate $1 - P_l$ remains 10^{-3}, the failure rate of the full consensus $1 - P_F$ in Hotstuff BFT is always far lower than the centralized consensus. With two or more rounds of synchronization, the Hotstuff BFT may have the same reliability issue as Raft that its reliability P_F is worse than the reliability in the centralized consensus when the number of nodes N is small. This abnormal condition is also caused by the limit of low consensus reliability P_C. When over $f = \frac{N-1}{3}$ replicas suffer from link transmission failures, the centralized network can still step into the synchronization stage and complete multiple rounds of synchronization. However, the protocol of Hotstuff BFT regards this case as a failure of consensus and terminates the state of synchronization. The probability of such cases monotonically de-

creases as the number of nodes N rises. Therefore, the centralized consensus can have higher reliability than the consensus reliability P_C of Hotstuff BFT when a small-scale network is implemented. However, the distributed consensus will become more reliable than the centralized consensus as the size of the network grows.

Unlike the comparison of latency between the centralized consensus and the Raft in Fig. 2.6, Fig. 2.8 indicates that the corresponding latency T to the basic Hotstuff BFT is longer than the centralized consensus because of multiple phases in the basic Hotstuff protocol. The numerical result of M_t in Hotstuff BFT approaches three times of M_t that cost in the centralized consensus. Yet, the discussion about the log throughput TP in every view of chain Hotstuff proves that the average time cost of one view in the chain Hotstuff BFT protocol can approach the average time latency of one-phase CFT protocol like Raft and the centralized consensus when the number of successively successful consensus views K is reasonably large.

2.4 Conclusion

In this chapter, the analysis of consensus reliability in the Raft-enabled wireless network concludes that with the constant communication link reliability, the consensus reliability in Raft increases monotonically along with the number of nodes. The relationship of consensus reliability with communication link reliability is interpreted in a linear form for simplicity. Meanwhile, the results show that the time latency is contradictory to the consensus reliability in Raft.

The link failed models of two distributed consensuses Raft and Hotstuff BFT, are analyzed to compare their performances to the centralized consensus in terms of full consensus reliability and time latency. The comparison of performance in centralized and distributed consensus indicates that the large-scale network with CFT protocol Raft will have higher full consensus reliability and lower latency than a centralized consensus when multiple

rounds of synchronization are implemented. The large-scale network with Hotstuff BFT protocol has higher full consensus reliability than the centralized consensus when synchronization is implemented. However, the network with basic Hotstuff BFT needs to consume more time slots than the centralized consensus to complete the consensus because of the three phases in the Hotstuff consensus protocol. The pipeline construction implemented in the chain Hotstuff BFT may provide a practical solution to improve the throughput of the Hotstuff BFT protocol.

Chapter 3

Communication Resource Allocation for Distributed Consensus

In both CFT and BFT protocols, communication acts as a critical enabler to ensure that every node can exchange its state information with others in the distributed consensus. Currently, most of the distributed consensus usually is deployed through stable wired communication [67]. However, the majority of the upcoming generation of IoT networks have the trend to become wireless systems. For example, The protocol of distributed consensus can be deployed in DLT-enabled wireless networks [51]. Unlike the reliable link transmission in a wired network, wireless channels are more stochastic and dynamic. The link transmission failure that occurs in the wireless channel can have the same influence on the state synchronization as the node that has crash or byzantine faults within it. This influence should be addressed when distributed consensus is implemented in the wireless network.

Some researchers have investigated the impact of wireless transmission on distributed consensus performance. A consensus-enabled industrial IoT network based on PBFT protocol has been presented to prove that the consensus mechanism is feasible for critical decision-making in the distributed wireless communication system. However, nodes in the wireless network may generally undertake the risk of link transmission errors and state synchronization loss [68]. The authors in [65] conclude the relationship between the reli-

ability of Raft and the reliability of wireless link transmissions. The excessive nodes can intensively occupy the limited wireless communication resource, which can cause the attenuation of the link reliability and consensus reliability. This problem usually occurs in the massive IoT network with wireless connection [69].

The conclusions from the above research indicate that the limited communication resources (e.g., transmit power, spectral bandwidth, etc.) can cause inadequate reliability of link connections, which reduces the reliability of distributed consensus through these link connections. The decline of consensus reliability can increase the frequency of the primary nodes changing, which may cause a longer latency to complete the consensus and the state synchronization among nodes in the network. Therefore, reasonable and practical communication resource allocation methods should be investigated to achieve high reliability and low latency of the distributed consensus.

3.1 Wireless Link Model

Reliability and latency are the important performance metrics for the distributed consensus in wireless networks [68]. The consensus reliability P_C refers to the probability that most trusted nodes complete vote or log replication in a term and the latency of Raft, which includes the time consumed by one round of downlink and uplink transmissions between the leader and all followers and the time of message verification [70]. When the number of nodes in the network is constant, P_C only depends on the link reliability of channels, which refers to the probability of successful link transmission between the leader and followers [65]. Different resource allocation methods and stochastic fading gains may cause variations in the link reliability and transmission time among the chan-

nels between the leader and followers. Therefore, varied link reliabilities and latency of wireless channels are determined by a derived wireless link model in this chapter. Relevant optimization problems of resource allocation are solved based on the proposed link reliability and latency.

The protocol of Raft is deployed on the considered wireless network that has $N+1$ static nodes, including a leader and N followers. The communication scheme in the protocol of Raft is assumed to be frequency division in this paper. The $2N$ channels, which include N downlink channels and N uplink channels that connect the leader and followers, are characterized by the Rayleigh fading model [71]. Rayleigh Fading is a statistical model for the effect of a propagation environment on a radio signal, such as that used by wireless devices. This model assumes that the magnitude of a signal that has passed through a communication channel will vary randomly, or fade, according to a Rayleigh distribution. It is viewed as a reasonable model in situations where the communication signal may bounce off objects from many directions before reaching the receiver, resulting in a large number of signal paths that can destructively interfere with each other. Rayleigh Fading Model simulates the worst-case scenario for signal distortion by a propagation environment. Therefore it is used extensively in designing wireless networks even if the channels are in terrible conditions. H_k denotes the Rayleigh fading gain of the k^{th} channel that $k \in [1, 2N]$, which follows the complex normal distribution, i.e., $H_k \sim \mathscr{CN}(0,1)$. The channel gains are assumed to be independent and identically distributed (i.i.d.). Therefore, $|H_k|^2$ follows the exponential distribution. When a package is sent through the k^{th} channel with a given transmit power P_{tk}, the signal-to-noise ratio (SNR) in this channel can be indicated as γ_k

$$\gamma_k = \frac{S_k |H_k|^2 P_{tk}}{P_{noise}}, \tag{3.1}$$

where P_{noise} refers to the white Gaussian noise power, S_k represents the large-scale effect on the k^{th} channel from the environment, such as the path loss and shadowing, and ρ is the SNR threshold. If γ_k is below the threshold ρ, the SNR outage occurs in the k^{th} channel. Consequently, the link reliability P_{lk} of the k^{th} channel can be calculated by the

SNR outage probability in this channel [72]

$$P_{lk} = 1 - Pr(\gamma_k < \rho) = exp(-\frac{\rho P_{noise}}{S_k P_{tk}}), \tag{3.2}$$

which reveals that the transmit power P_{tk} is the communication resource that can affect the link reliability P_{lk} when other parameters remain constant in the wireless link model. Meanwhile, the latency cost by transmission in the k^{th} channel can be represented as

$$t_k = \frac{M}{B_k log(1+\gamma_k)}, \tag{3.3}$$

where M is the average length of the package sent by the leader or followers and B_k is the bandwidth used in this channel. When the distributed consensus is implemented in the wireless network, the derived model of link reliability P_{lk} in (3.2) and time latency t_k in (3.3) can determine the critical parameters of the performance, such as consensus reliability P_C and the latency of consensus t_c. The derived model shows that these performance parameters can be improved by optimizing the power and bandwidth allocation.

3.2 Power Allocation Scheme for Consensus Reliability

The model of wireless channel in (3.2) is implemented as an example to demonstrate the influence in the consensus reliability P_C given by the allocated transmit power P_{tk}, which is a prevalent type of communication resource that can influence the link reliability in practice. Therefore, P_{tk} is regarded as a variable of the communication resource allocation scheme to pursue the maximum consensus reliability P_C. The procedure of analysis can be similar when other wireless communication models are selected.

With the proposed link reliability, the consensus reliability P_C can be represented as a function with the transmit power P_{tk}. The communication scheme of Raft shows that the successful follower needs to complete both the downlink and uplink transmission. Therefore, the consensus reliability P_C can be calculated as

$$P_C = \sum_{k=\frac{N}{2}+1}^{N} \sum_{Q_k \in \Omega_S} \prod_{w \in Q_k} P_w \prod_{v \in Q_k^C} (1 - P_v), \tag{3.4}$$

where Q_k refers to the set of k followers that successfully complete both the downlink and uplink transmission. Ω_S refers to the set that over $\frac{N}{2}$ followers have reached the consensus. W is a successful follower that belongs to Q_k and v is a failed follower that belongs to complement of set Q_k. P_w represents the probability that w belongs to the set Q_k

$$P_w = P_{lw}^{DL} P_{lw}^{UL}, \tag{3.5}$$

which is the product of the downlink reliability P_{lw}^{DL} and uplink reliability P_{lw}^{UL}. Similarly, P_v refers to the probability that nodes from v complete the downlink and uplink transmissions successfully

$$P_v = P_{lv}^{DL} P_{lv}^{UL}, \tag{3.6}$$

Other parameters in (3.4) are assumed constant for all $2N$ channels.

The scheme of power allocation aims to maximize the consensus reliability P_C when the overall transmit power P_{sum} is fixed. In the protocol of Raft, the overall transmit power P_{sum} is allocated to all $2N$ channels. Therefore, the problem of optimization for the power allocation scheme can be formulated as

$$\begin{aligned} \min_{P_t} \quad & 1 - P_C \\ \text{s.t.} \quad & \sum_{k=1}^{2N} P_{tk} \leq P_{sum}, \end{aligned} \tag{3.7}$$

This optimization problem has $2N$ variables of transmit power. The channels from 1 to N represent the downlink channel of N followers, and channels from $N+1$ to $2N$ are the corresponding uplink channel of N followers. Sequential quadratic programming (SQP) is implemented to solve the nonlinear programming in this resource allocation scheme, which aims to transform the original optimization problem into an optimal quadratic problem and find the appropriate descent direction d. The original objective function in 3.7 will be approximated to the first three items of taylor's series in SQP, which can be formulated as follow:

$$\begin{aligned} \min_{d} \quad & f(P_{tk}) + \nabla f(P_{tk})^T d + \frac{1}{2} d^T \nabla^2 L(P_{tk}, \lambda) d \\ & \text{s.t.} \nabla g(P_{tk}) d + g(P_{tk}) = 0, \end{aligned} \tag{3.8}$$

where $f(P_{tk})$ represents the objective function $1 - P_C$ with a vector of transmit power P_{tk} allocated to all $2N$ channels, and it is the first item of the Taylor's series in $1 - P_c$, $\nabla f(P_{tk})^T$ denotes the gradient of the transpose of $f(P_{tk})$ and it refers to the second item of the Taylor's series. $\frac{1}{2} d^T \nabla^2 L(P_{tk}, \lambda) d$ represents the third item and $\nabla^2 L(P_{tk}, \lambda) d$ represents the Hessian matrix of the Lagrangian function at $f(P_{tk})$. The Hessian matrix is a square matrix of second-order partial derivatives of a scalar-valued function and is used to describe the local curvature of the function. The summation of these three items can be the approximation of the objective function in 3.7. $g(P_{tk})$ denotes the constraint and $L(P_{tk}, \lambda)$ denotes the Lagrangian Function, which can be calculated as:

$$L(P_{tk}, \lambda) = f(P_{tk}) - \sum \lambda g(P_{tk}). \tag{3.9}$$

where λ represents the Lagrange multipliers, which are auxiliary variables introduced in the optimization problem to deal with constraints [73]. The remainder R_n of the Taylor series [74] can be calculated as:

$$R_n = \sum_{n=3}^{+\infty} \frac{\nabla^n f(P_{tk})}{n!} d^n. \tag{3.10}$$

If the descent direction d is small in each iteration, the remainder R_n will converge to zero, which means the transformed optimization problem in (3.9) is equal to the original nonlinear optimization problem. Therefore, the solution to the optimization problem (3.8) is identical to the convergence of the result from SQP. However, consensus reliability P_C from (3.4) shows that the overall probability is the summation of the product of link reliabilities from $2N$ channels, which can exponentially increase the complexity of nonlinear programming. The high complexity can be impractical to deploy the scheme of communication resource allocation in a large-scale wireless network.

Two power allocation methods, which can be practical to implement in reality, are proposed to compare with the performance of the optimal power allocation scheme from SQP. The first method is allocating the transmit power equally to each channel,

$$P_{tk}{}^1 = \frac{P_{sum}}{2N}. \tag{3.11}$$

With identical communication resources, the channel with better channel gain will have higher link reliability to complete transmission.

The second power allocation method aims to ensure all channels receive appropriate transmit power P_t to reach the same link reliability P_l, which follows the proportion of the channel fading gain S_k in each channel to the summation of channel fading gains from all $2N$ channels. The link reliability in (3.2) indicates that the transmit power P_{tk} is inversely proportional to S_k when link reliability P_{lk} is constant. Therefore, the link reliability in this power allocation method should be

$$P_{tk}{}^2 = \frac{P_{sum}\sum_{k=1}^{2N} S_k}{S_k}. \tag{3.12}$$

According to the inverse proportional relationship between the transmit power P_t and fading gain S_k when the link reliabilities of all channels tend to be identical with this allocation method, more transmit power should be compensated to the communication channel with lower S_k to keep the identical link reliability. These two power allocation methods have lower complexity than the result of SQP, which means they can replace the optimal power allocation method from the nonlinear optimization if the gap between their performances can be tolerated.

3.3 Bandwidth Allocation Scheme for Consensus Latency

Besides reliability, latency is also critical to the performance of distributed consensus. Consensus reliability and transmission time are two factors that can influence the overall latency of distributed consensus in a wireless network. Optimal consensus reliability indicates that the protocol of Raft has the maximum probability of preventing a new leader election and spending extra time on this stage. Therefore, an optimal consensus latency means the reliability of consensus needs to reach the maximum, which means the power allocation method in this condition should be optimal and follow the result of SQP, then the only factor that can change the consensus latency is the transmission time cost by nodes. Based on the model in (3.3), the consensus latency can be reduced by minimizing the transmission time through an optimal bandwidth allocation method. In this section, we aim to investigate this optimal bandwidth allocation scheme to pursue the minimum value of consensus latency.

The protocol of Raft indicates that each follower needs to receive a downlink message from the leader and respond with confirmation through uplink transmission in one term of consensus. The time that $\forall n \in 1,2...,N$ follower spends in completing the consensus can be represented as

$$\begin{aligned} t_n &= t_n^{DL} + t_n^{UL} + t_v \\ &= \frac{M^{DL}}{B_n^{DL} log(1+SNR_n^{DL})} + \frac{M^{UL}}{B_n^{UL} log(1+SNR_n^{UL})} + t_v, \end{aligned} \tag{3.13}$$

which is the summation of delays caused by the downlink t_n^{DL}, uplink transmissions t_n^{UL} and verification time t_v. M^{DL} and M^{UL} refer to the package length during downlink and uplink transmission. In the same round of communications, the protocol of Raft indicates that M^{DL} and M^{UL} are identical for all downlink and uplink channels, respectively. All nodes are assumed to have the same ability to handle the verification, so the verification time t_v of all N followers is the same. The derived model of latency in (3.13) shows the bandwidth allocated to n^{th} channel is the communication resource that can influence the transmission latency t_n besides the SNR of channels. The consensus ends the term when the last follower completes its transmission. Therefore, the longest latency cost by the follower can be considered as the latency t_c of distributed consensus.

$$t_c = \max\{t_1, t_2, ..., t_N\}, \tag{3.14}$$

which derives an optimization problem to solve the minimum value of t_c when the overall bandwidth B_{sum} is constant.

$$\begin{aligned} &\min_{B} \quad t_c \\ &\text{s.t.} \sum_{k=1}^{2N} B_k \leq B_{sum}. \end{aligned} \tag{3.15}$$

where SNR in all downlink and uplink channels of the followers are based on the result of SQP from transmit power allocation, which means the consensus reliability P_C converges to the theoretical maximum value in this scheme. The overall bandwidth B_{sum} is the constraint for this optimization problem. Table 3.1 shows the notations of major parameters used in the proposed resource allocation schemes.

Table 3.1: Notation used in resource allocation of Raft-enabled Network

Notation	Definition
N	Number of Nodes within network
S_k	Large Scale Effect of the k^{th} channel
H_k	Rayleigh Fading Gain of the k^{th} channel
P_{sum} (dBm)	The overall transmit power
B_{sum} (MHz)	The overall bandwidth
P_{tk} (dBm)	Transmit Power allocated to the k^{th} channel
B_k (MHz)	Bandwidth allocated to the k^{th} channel
P_{lk}	Link reliability of the k^{th} channel
P_C	Consensus reliability
t_k (s)	Transmission time of the k^{th} channel
t_c (s)	Transmission time cost by consensus
N_{max}	Number of node with maximized consensus reliability

The optimization problem presented in equation (3.15) is nonlinear, and its objective function lacks an explicit closed-form solution, implying that the solution is complex and cannot be obtained through straightforward mathematical methods. Thus, we have employed Particle Swarm Optimization (PSO) to iteratively resolve this optimization problem and find the minimum value of t_c. The PSO algorithm, renowned for its prowess in global optimization, enables us to evade suboptimal solutions [75]. In the context of our study, Algorithm.1 represents the application of PSO in bandwidth allocation within the Raft consensus algorithm. The position of the particles in this algorithm corresponds to the bandwidth distributed to the wireless channels. The PSO's inertia weight w, along with acceleration constants $c1$ and $c2$, guide the particle's movements and drive it towards the historically optimal and collective optimal position. The position of a particle gets updated

Algorithm 1 PSO algorithm for t_c

Initializepopulation
for $m = 1 : Iterations$ **do**
  for $i = 1 : n$ **do**
    $t_{i,m} = f(B_{i,m})$
    if $t_{i,m} < t_{i,h}$ **then**
      $t_{i,h} = t_{i,m}$
      $B_{i,h} = B_{i,m}$
    else
      $t_{i,h} = t_{i,h}$
      $B_{i,h} = B_{i,h}$
    end if
    $t_{i,opt} = min(t_{i,m})$
    $B_{i,opt} = B_{min(t_{i,m})}$
  end for
  for $i = 1 : n$ **do**
    $v_i(m+1) = wv_i(m) + c_1 r_1 (B_{i,opt} - B_i) + c_2 r_2 (B_{i,h} - B_i)$
    $B_i(m+1) = B_i(m) + V_i(m+1)$
    if $V_i(m+1) > V_{max}$ **then**
      $V_i(m+1) = V_{max}$
    else if $V_i(m+1) < V_{min}$ **then**
      $V_i(m+1) = V_{min}$
    end if
    if $B_i(m+1) > B_{i,max}$ **then**
      $B_i(m+1) = B_{i,max}$
    else if $B_i(m+1) < B_{i,min}$ **then**
      $B_i(m+1) = B_{i,min}$
    end if
  end for
end for

iteratively through the combination of its inertia weight and acceleration constants. After sufficient iterations, we are able to derive $t_{i,opt}$ as the maximum value of consensus latency t_c, a testament to PSO's effectiveness in exploring and converging towards an optimal solution in a complex problem space.

In the protocol of Raft, all followers need to occupy a constant overall bandwidth. A reasonable expectation of the optimization result is that most of the followers' latency t_n tends to be close when the optimal bandwidth allocation method is implemented because a non-optimal bandwidth allocation method can cause some followers to cost more time to

complete the transmission, which increases the overall latency of distributed consensus in wireless networks. However, the stochastic wireless channel between the leader and some followers may have extremely terrible conditions, which can occupy a large proportion of communication resources and limit the optimal performance of the distributed consensus.

Because Particle Swarm Optimization is a type of practical evolutionary algorithm for non-linear optimization. It needs to complete large number of iterations to let the fitness curve reach the optimal value of the objective function, which means the optimal result can be much better than the benchmarks that used in power allocation even if the Coefficient of Variation is small.

3.4 Limited Communication Resource and Optimal Number of Nodes

The algorithms of nonlinear optimization proposed in this chapter can solve the optimization problem of the communication resource allocation to achieve the maximum consensus reliability P_C and minimum consensus latency t_c. However, if the overall communication resources are not adequate, even the optimal consensus reliability and latency cannot reach the requirement of high reliability and low latency in specific scenarios. This section aims to investigate the solution to the problem of inadequate overall communication resources in resource allocation. Firstly, the criteria of adequate communication resources for the distributed consensus Raft is defined. Then we find out the solution based on the feature of fault tolerance in the distributed consensus to improve the performance of the optimized consensus reliability and latency from the perspective of network size.

3.4.1 Limited Overall Communication Resource for Raft

In the assumption of this article, the allocated communication resources to the wireless channels and channel gains are the parameters that can influence link reliability P_l and the consensus reliability P_C. Therefore, the link reliability P_l and consensus reliability P_C can be reasonable criteria to judge the condition of overall communication resources when the wireless channel gain is determined. The reliability of information delivery and synchronization changes in different applications. These reliability requirements correspond to the consensus reliability if the distributed consensus is implemented. The dotted lines in Fig. 3.1 denote the target consensus reliability in multiple 5G scenarios, including URC over the long term, V2V wireless coordination, Reliable cloud connectivity, and Real-time Virtualization [76] [77]. The optimization problem in (3.7) indicates that even though the

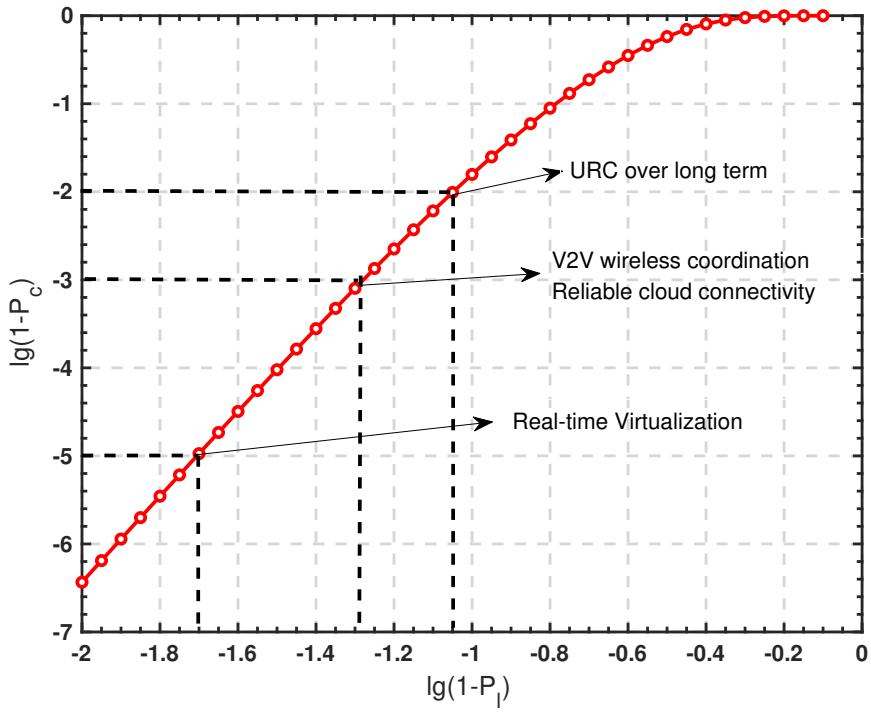

Figure 3.1: Reliability requirements in different scenarios

power allocation method is optimized by SQP, adequate overall transmit power should also be provided if the consensus reliability needs to be improved to reach the requirement of a specific scenario. Otherwise, an alternative solution should be implemented to improve the consensus reliability of Raft in the wireless network.

3.4.2 Optimal Number of Nodes

When the overall communication resource is constant, the number of nodes that participate in the distributed consensus can influence the performance of distributed consensus because more nodes should occupy the limited communication resources, and each node is expected to take fewer resources for the transmission. Specifically, the performance of the resource allocation method will be damaged when the overall communication resources are inadequate because some channels cannot gain enough resources to achieve the target performance.

A reasonable solution to this problem is eliminating the redundant consensus nodes that are linked with terrible communication channels. However, the increasing size of the network represents that the distributed consensus can tolerate more crash faults or byzantine fault nodes [78]. These two controversial characteristics can cause the maximum global value for the reliability of consensus P_C with a dynamic number of nodes but constant communication resources for a local wireless network. The corresponding number of nodes N to the maximum of P_C can be determined by Proposition 1.

Proposition 1 shows that when the overall communication resources are inadequate for a distributed network, the number of nodes engaged in this network should be less than the value of N_{max}. The maximum value of the consensus reliability P_C indicates that excessive consensus nodes can damage the reliability of Raft. Therefore, a large-scale network can abandon some nodes that have terrible communication channels to converge the number of nodes N to N_{max} if the overall communication resource is rare, which

can improve the consensus reliability of Raft. For example, In a multiple-layer consensus network [79], the network size in the consensus layers can be optimized based on the communication resource allocated to them, which helps the whole network achieve the highest performance.

Proposition 1. *If N_{max} is assumed as the number of followers that can reach the maximum of consensus reliability,*

$$N_{max} = \lceil M_a \rceil = \lfloor M_b \rfloor. \tag{3.16}$$

M_a and M_b correspond to the value of function

$$\begin{aligned} M_a &= \frac{\widetilde{P} - \sqrt{\widetilde{P}^2 - 4\widetilde{P} + 1}}{\frac{1}{2} - 2\widetilde{P}} \\ M_b &= \frac{1 - 3\widetilde{P} - \sqrt{\widetilde{P}^2 - 4\widetilde{P} + 1}}{\frac{1}{2} - 2\widetilde{P}}, \end{aligned} \tag{3.17}$$

where $\widetilde{P} = (1 - P_l^2)P_l^2$ and P_l denotes the average link reliability of channels

Proof. See Appendix B □

The computational complexity of the model revolves around the calculation of N_{max}, which is the optimal number of nodes that can reach the maximum consensus reliability. Calculating N_{max} involves solving the equation (3.17), which is the function of link reliability $P(N)$. $P(N)$ is a function with $2N$ variables, which means the calculation of $P(N)$ can involve iterating over all $2N$ variables at least once. Therefore, the computational complexity for $P(N)$ will be $O(N)$. Subsequently, N_{max} is calculated from $P(N)$ with the equation (3.17), which are operations with constant computational complexity. Therefore, the overall computational complexity of the model primarily depends on the calculation of $P(N)$ and is $O(N)$.

While the proposed model's computational complexity is linear in the size of network, the feasibility of real-time or near-real-time implementation of the proposed model depends on the number of nodes N and environmental effects. If N is large in the network, the calculation of link reliability $P(N)$ can be computationally intensive, which makes real-time implementation challenging. Moreover, the dynamic change of the communication environment causes a varied distribution of link reliability among nodes, and the Raft-enabled network has to frequently recalculate the optimal resource allocation scheme, which may pose influence the real-time implementation of the proposed model. Therefore, an ideal condition for the real-time deployment of the proposed model should contain an appropriate number of nodes within the network and a stable communication environment.

3.5 Simulation Results

In the simulation section, the proposed resource allocation schemes for Raft are simulated in MATLAB R2019b. Based on the Rayleigh Fading model, we assume the channel fading gain H_k and large-scale effect S_k of $2N$ channels from (3.1) are in the Gaussian distribution [71]. The nodes are set as static nodes, and the number of them N in the wireless network is set to 13. The overall power P_{sum} ranges from 20 dBm to 36 dBm for the transmit power allocation. The Coefficient of Variation (CV), which refers to the ratio of standard derivation to mean of channel fading gain H and large-scale effect S in the wireless model, is implemented in the simulation to represent the dispersion in the probability distribution of wireless channel fading gains and large-scale effect. A higher CV means that part of channels have more probabilities of suffering terrible fading gain H and large-scale effect S, which influence the performance of proposed resource allocation schemes.

In this section, the optimal reliability of the distributed consensus P_C from SQP is compared with the other two transmit power allocation methods. The numerical results of three transmit power allocation methods are presented in Fig. 3.2 when the channel gains S_k has a high coefficient of variation (CV=1.303). The consensus reliability given by the three allocation methods is significantly different. The output P_C from the equal power method in (3.11) is closer to the optimized result of SQP, which reveals that the equal power allocation method has better performance than the equal link reliability method when the variation of channel gains is large. Even though the complexity of SQP will rise when the size of the network increases, the transmit power allocation method derived by SQP is still the best allocation method to use in this case.

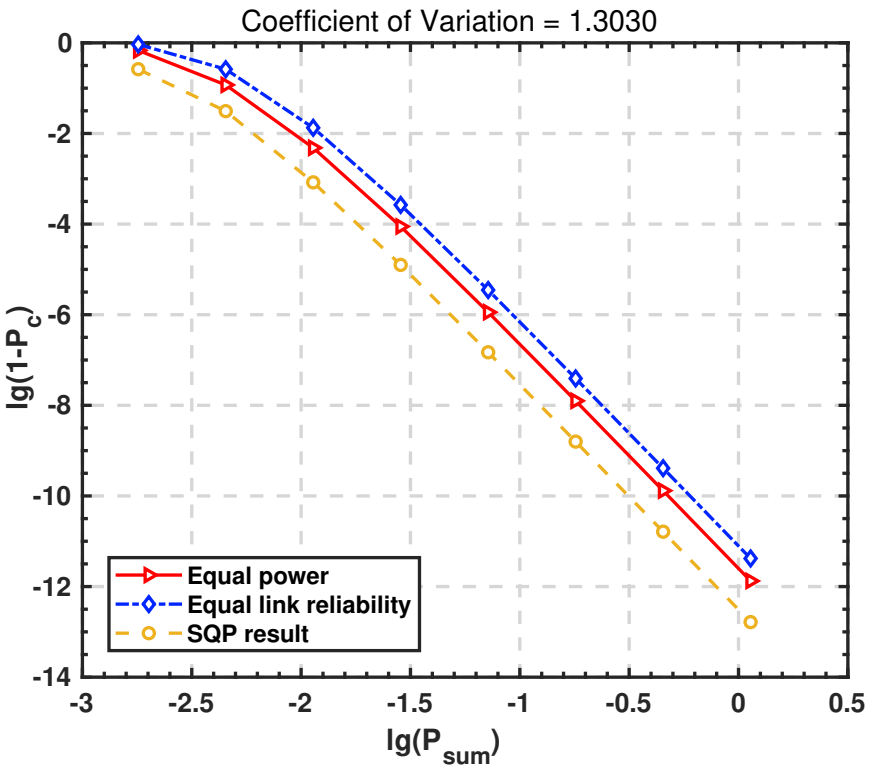

Figure 3.2: Performance of three power allocation methods with a high coefficient of variation in channel gains

Moreover, Fig. 3.3 shows that when the channel fading gain is more concentrated (CV=0.392), the curves of equal power and equal link reliability methods will converge to the optimized consensus failure rate $1-P_{Copt}$, which means three power allocation methods will have similar performances when the conditions of wireless channels are close. Therefore, two practical transmit power allocation methods in (3.11) and (3.12) can substitute the optimal power allocation method derived by SQP in this case.

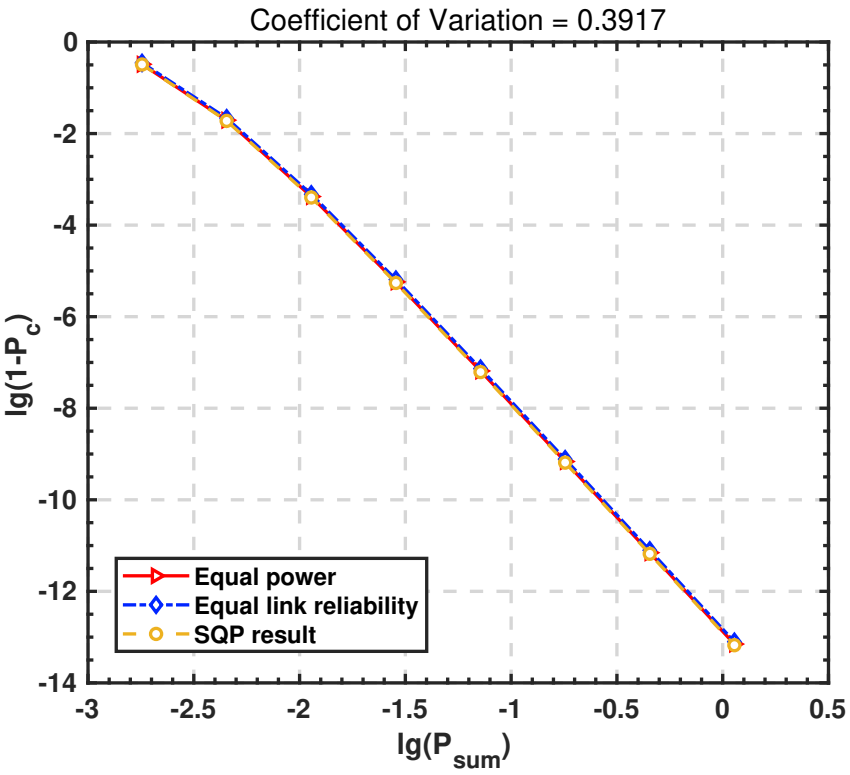

Figure 3.3: Performance of three power allocation methods with low coefficient of variation in channel gains

Fig. 3.4 illustrates the influence of the varied channel gains in consensus reliability where P_C denotes the consensus reliability derived by the two practical power allocation methods in (3.11) and (3.12), P_{Copt} is the optimal consensus reliability from SQP, and Reliability Gap (RG) represents the ratio of consensus failure rate between $1-P_C$ and $1-P_{Copt}$.

The difference among the three allocation methods gradually increases when the CV of channel gains is rising. All three methods have approximate results when the CV is less than 0.5, which means the other two power allocation methods can replace the optimal power allocation method derived by SQP with a small compromise performance.

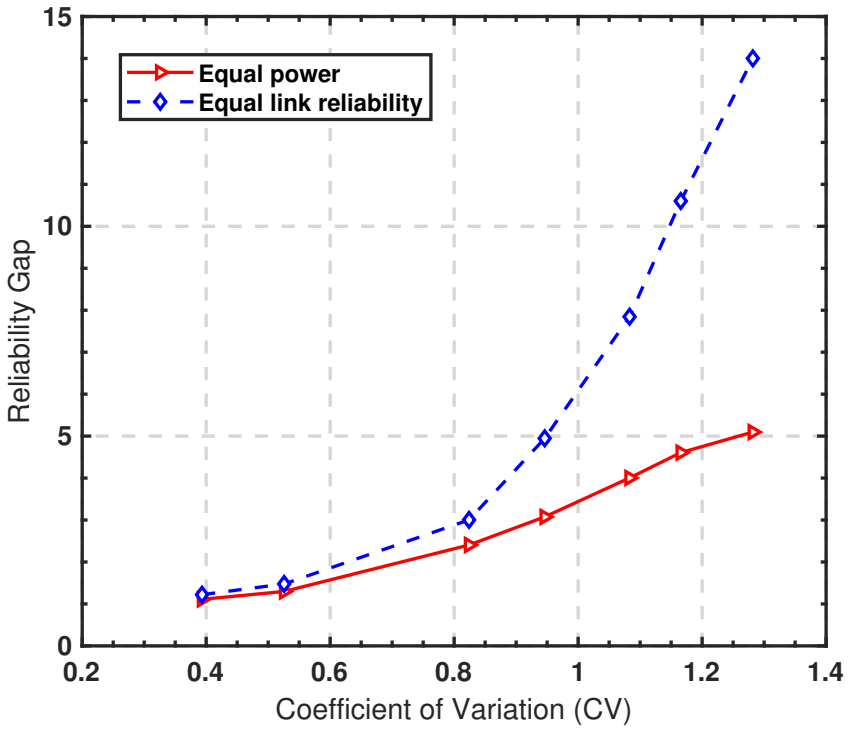

Figure 3.4: The performance comparison among optimal consensus reliability and other two methods with different CVs in wireless channel gains

In practice, the CV of wireless channel gain can be reduced by abandoning some nodes with bad channel conditions (e.g., low large-scale effect S,etc.) to achieve a near-optimal power allocation scheme, which is supported by the feature of fault tolerance in the distributed consensus.

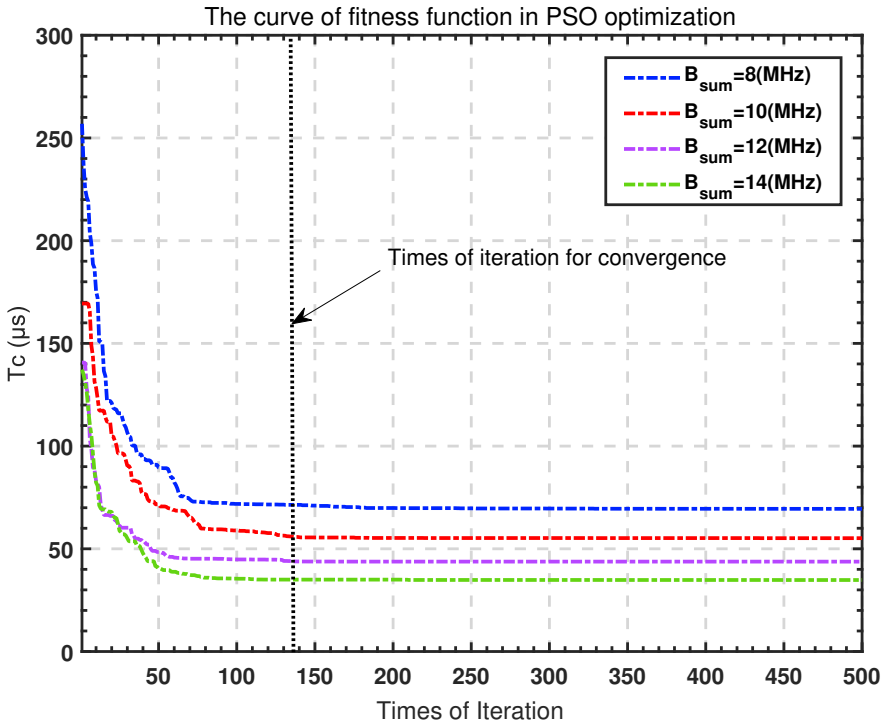

Figure 3.5: The curve of fitness function in PSO

The simulation of bandwidth allocation assumes that the amount of overall bandwidth B_{sum} ranges from 8 to 14 MHz, and the number of followers $N = 12$. The model of the wireless channel is the same as previous transmit power allocation and the SNRs of all channels are set based on the optimal result of transmit power allocation scheme from SQP. The iteration rounds are set to 500 in the PSO algorithm. The curve of the fitness function in the proposed optimization problem should be presented first. Fig. 3.5 shows the convergence of the optimal consensus latency when different overall bandwidths are used in the same wireless network. The convergence of consensus latency decreases when more overall bandwidth is provided for the communication. The number of iterations that the result of PSO converges to the minimum consensus reliability is between 100 to 150. The transmission time cost by all followers is evaluated in Fig. 3.6 when the optimal bandwidth

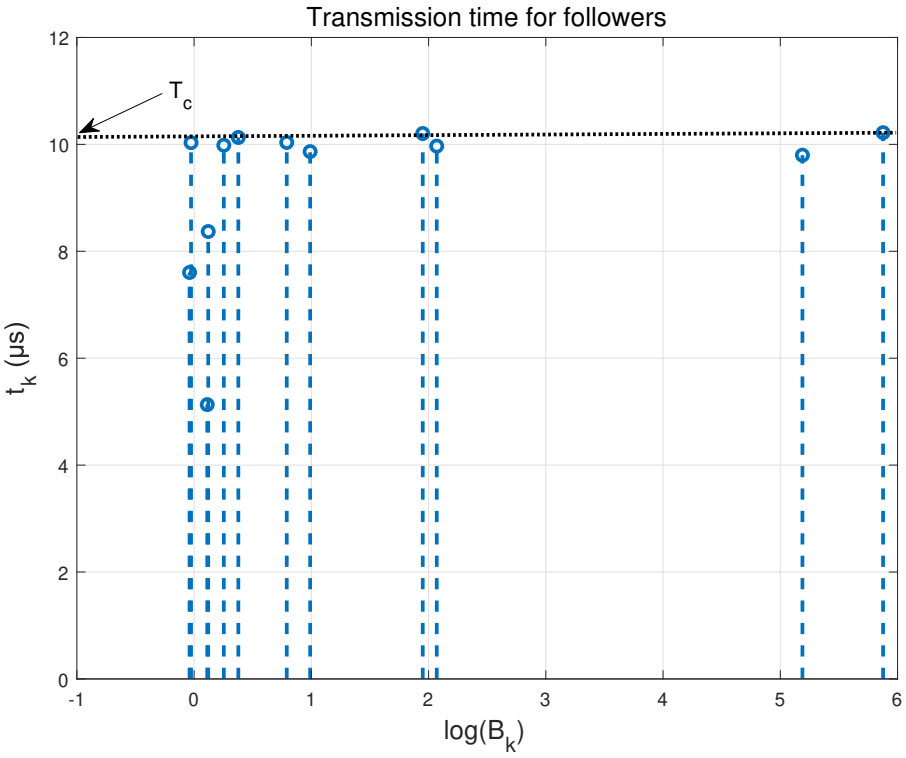

Figure 3.6: The transmission time used by followers with optimized bandwidth allocation scheme

allocation scheme is exploited. Because the definition of consensus latency refers to the longest time cost by the follower from the whole wireless network, the simulation result matches the expectation that the transmission time cost by most of the followers is close when the consensus latency t_c reaches a minimum value.

The stochastic wireless channels between the leader and followers have variable channel gains, which can have a significant influence on consensus latency. Fig. 3.7 aims to indicate the tendency of optimized consensus latency t_c with an increased coefficient of variation CV in channel gain S_k. The results show that when CV increases from 0.74 to 1.56, the optimal consensus latency t_c dramatically rises from 1 μs to 10^5 μs. This numerical result reveals a larger variation of channel gain can increase the optimal latency of Raft in the wireless network.

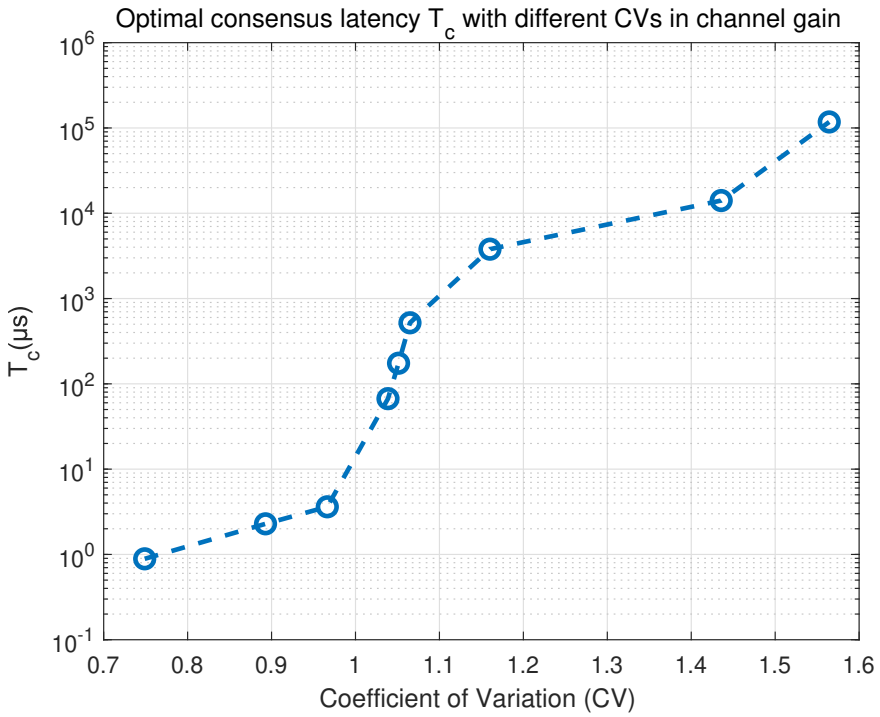

Figure 3.7: The optimal consensus latency with different CV in the channel gains

The simulation of the optimal number of nodes is presented in Fig. 3.8, which illustrates the change in the consensus reliability when the number of nodes in the network increases. The number of nodes is assumed to range from 4 to 40 and the overall communication resource keeps constant. The trend of consensus reliability increases first and then drops when the number of followers reaches the optimal network size and finally increases. The number of nodes that corresponds to the maximum consensus reliability matches the result of the optimal number of nodes in Proposition 1. S represents the rounds of synchronization processed during the Raft consensus protocol. When more rounds of synchronization S are implemented to the distributed consensus protocol, the maximum value of consensus reliability P_C will increase. But the eventual tendencies of all curves still remain the same.

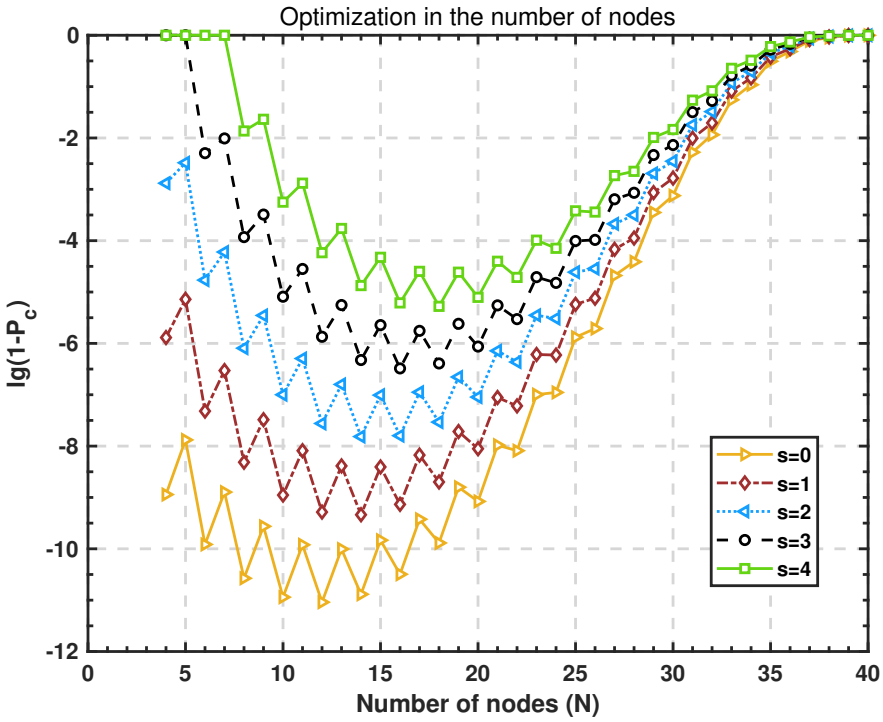

Figure 3.8: Optimized network size for Raft

The negative influence on consensus latency from varied wireless channel gains indicates that if the consensus latency needs to be improved, the node with terrible channel gain should be removed. Fig. 3.9 compares the numerical result of optimal consensus latency t_c before and after the followers with the worst channel gains are eliminated from the network. The number of followers $N = 8$ in the initial network. The channel gains S_k of all nodes follow the normal distribution. The convergence of optimized t_c is close to 2000 μs when no followers are removed. The convergence of t_c drops to the region between 300 and 400 μs when one follower with the worst channel gain is removed. And t_c will keep dropping to 10 μs after two followers are removed from the network, which proves this method is also efficient in reducing the consensus latency.

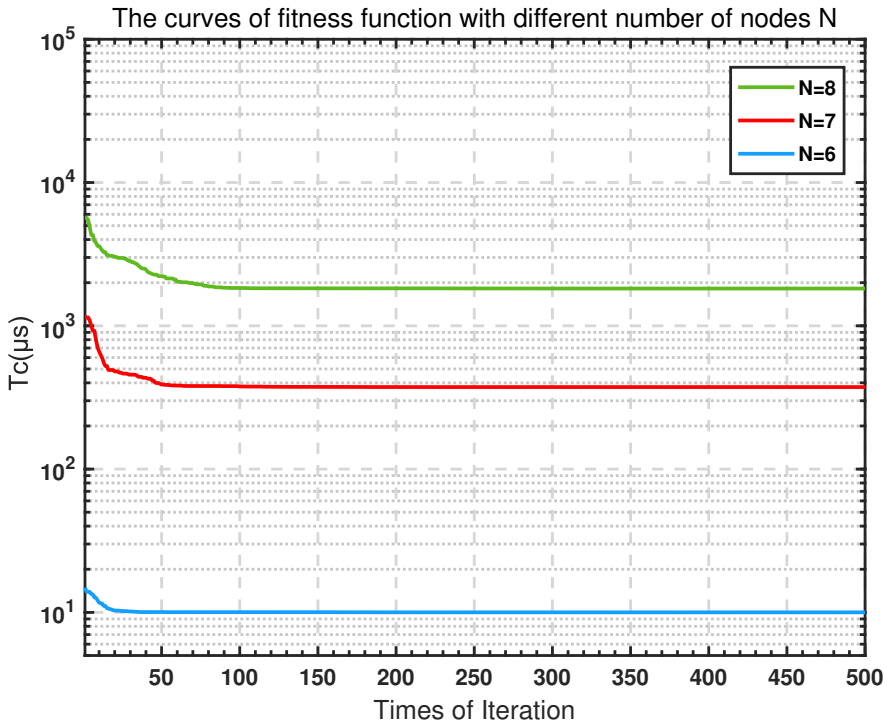

Figure 3.9: The convergence of consensus latency with different numbers of followers

The procedure of simulation indicates that the computational complexity of SQP and PSO heavily depends on the number of nodes N. The computational consumption if optimization algorithms is also verified during the simulation. When number of node $N > 50$, the time cost by the simulation will expands to hours. Therefore, some low computational complexity algorithms need to be implemented to optimize the communication resource allocation in wireless distributed consensus with large networks.

Another optimization method has been tried in the optimization problem of bandwidth allocation is bisection. But this method has a higher computational complexity $log(n)$ than PSO and it is more possible to achieve a local optimal value instead of a global one. Therefore, this bisection method is abandoned in the optimization problem of bandwidth allocation.

3.6 Conclusion

In this chapter, we delve into methodologies aimed at enhancing the efficiency of the distributed consensus Raft in wireless networks. One of the key challenges in such networks is ensuring effective and reliable data transmission. To address this problem, optimal power, and bandwidth allocation methods are explored to boost reliability and cut down latency in the wireless distributed consensus. Understanding the intricacies of these methods necessitates a deep dive into their foundational principles. The power and bandwidth allocation methods are meticulously derived by two distinct optimization algorithms. Each of these algorithms is implemented to extract the best possible performance from the network even when the overall communication resources available remain unchanged. In other words, they aim to reach near-optimal performance levels within the constraints of a constant resource pool.

However, real-world scenarios often throw curveballs, and it's conceivable that in certain situations, the available resources might be insufficient to meet the required performance standards. Anticipating such scenarios, we have delineated an optimized network size to help pinpoint the optimal number of nodes that can function efficiently given the available resources. It can also offer a tangible solution for scenarios where resources fall short of requirements. In essence, this chapter aims to provide a roadmap for the effective deployment of resource allocation schemes, ensuring the efficiency and reliability of wireless distributed consensus.

Chapter 4

Adaptive Protocol and Sharding Scheme for Distributed Consensus

The nature of wireless communication magnifies some of the core challenges of distributed systems, especially scalability. Scalability, the ability of a system to grow and manage increased demand effectively, encounters numerous obstacles in wireless distributed consensus. Some of the most pressing issues like bandwidth limitations and energy constraints were already tackled by optimizing communication resources and adjusting the number of nodes as detailed in [80]. However, these solutions are just the tip of the iceberg, and several underlying challenges persist. A quintessential example is the impact of wireless mobile nodes, which introduce a factor of dynamism to network topologies. Envision a constantly evolving network where nodes are not stationary; they frequently enter, exit, or relocate within the network. Establishing a consistent consensus under such fluid circumstances becomes an intricate task. The propensity of wireless networks to morph makes them susceptible to network partitioning. As the size and complexity of the network burgeon, it tends to fragment into multiple, disconnected sub-networks. This partitioning hinders communication and consensus achievement.

Furthermore, while most discussions around scalability gravitate towards performance, it's imperative to acknowledge that scalability extends beyond just throughput or latency. As a network scales, so does its vulnerability. The enlarged network can bring malevolent entities, be they malicious nodes or those attempting to jam signals, with more opportunities to disrupt the consensus process. Thus, the security of consensus mechanisms becomes increasingly imperative in expansive wireless networks.

Given these multifaceted challenges, this chapter introduces the adaptive protocol of distributed consensus and security analysis of the sharding scheme to alleviate the aforementioned scalability issues. The adaptive protocol of distributed consensus does not only contain a static set of rules but also provides a dynamic framework. It possesses the ability to recalibrate the consensus algorithm, adapting it to the real-time conditions prevalent in the wireless network. Such adaptability ensures the system remains resilient, even when nodes are involuntarily ejected due to issues like network partitioning or temporary disconnections.

Furthermore, sharding offers a structural solution. Originating from the domain of computer networks, sharding dissects a monolithic network or dataset into smaller, more manageable chunks known as shards. These shards are functional entities that can operate autonomously, processing their subset of data independently. Such a decentralized approach accelerates processes, paving the way for parallel operations and, consequently, amplifying the system's throughput and efficiency. This chapter will analyze the security level of sharding when it is implemented in wireless distributed networks, providing insights and strategies for highly effective wireless distributed consensus.

4.1 Adaptive Protocol of Raft in Wireless Network

The protocol of distributed consensus is primarily designed for wired networks, where the communication link is stable, predictable, and reliable [81]. Therefore, communication failure is not a major concern in this circumstance. However, the deployment of distributed consensus in wireless networks may suffer from several challenges. For example, wireless networks are typically subject to interference, signal fading, and environmental factors, all of which can lead to unpredictable delays and packet loss [82]. Most of the distributed consensus assumes that messages will be delivered in a timely manner and that failures are relatively rare. In a wireless network, these assumptions may not hold, leading to frequent leader changes or a large number of failed attempts to reach a consensus. Moreover, wireless networks generally have less bandwidth compared to wired networks. However, a high communication-complexity distributed consensus needs a significant amount of network traffic for leader election and state synchronization [83]. The high traffic could deplete the limited bandwidth available on a wireless network and damage the performance of distributed consensus [84]. The high mobility of nodes in wireless networks can cause frequent changes in the network topology and reduce the throughput of state synchronization [85], which cannot be handled by the original protocol of distributed consensus.

Efforts have been made by various researchers to address the challenges of distributed consensus in wireless networks. [50] shed light on the impact of communication resources on the CFT and BFT consensus protocols within wireless networks. To demonstrate the feasibility of consensus mechanisms in critical decision-making processes within distributed wireless communication systems, they introduced a consensus-enabled industrial IoT network based on the PBFT protocol. Further studies, such as the one conducted by [65], established a relationship between the reliability of Raft and the reliability of wireless link transmissions. They found that an excessive number of nodes can significantly occupy limited wireless communication resources. This occupation can lead to a reduction in both the reliability of the link and the consensus, particularly in large-scale IoT networks

that utilize wireless connections, as discussed in [69]. [80] proposes power and bandwidth allocation schemes to optimize the consensus reliability and latency of Raft for multiple 5G scenarios, and a blockchain-based mobile edge computing framework is presented for adaptive resource allocation and computation offloading in wireless networks [86]. The findings from these studies try to solve the network instability and resource allocation issue of the distributed consensus in wireless networks. However, they have not investigated any solutions that involve adapting the original protocol of distributed consensus to better address these potential issues in wireless circumstances.

A blockchain system for security-related data collection is designed in MANETs [87], which proves the compatibility of MANET'S routing scheme in distributed consensus. A lightweight Blockchain-based secure routing algorithm is proposed to improve the security level of swarm Unmanned Aircraft System (UAS) networking [88]. With the enlightenment from these solutions to an unreliable wireless connection in different routing scenarios, We adapt the original protocol of Raft to ensure the protocol can be compatible while it is deployed in wireless networks in this section.

In the protocol of Raft, a valid node should support broadcasting, multicasting, peer-to-peer communication, and the verification of request call [89]. If Raft is deployed in a wireless environment, these functionalities should be adapted to suit the unique characteristics of wireless networks, which include high variability in signal strength, dynamic network topology, and higher data failure rates. Therefore, the adaptive protocol of distributed consensus in wireless networks should consider several phases: the determination of consensus nodes, consensus achievement, and extra state synchronization [90]. Therefore, this section proposes the main procedure of Raft's adaptive protocol to show the completed progress of Raft when it is deployed in a wireless network. The key functions of the adaptive protocol include node counting, leader election, log replication, extra synchronization, and node entry or exit mechanism to ensure that information can be transmitted and received accurately and efficiently by every normal node within the network. The main procedure of the proposed adaptive protocol is shown in Fig. 4.1.

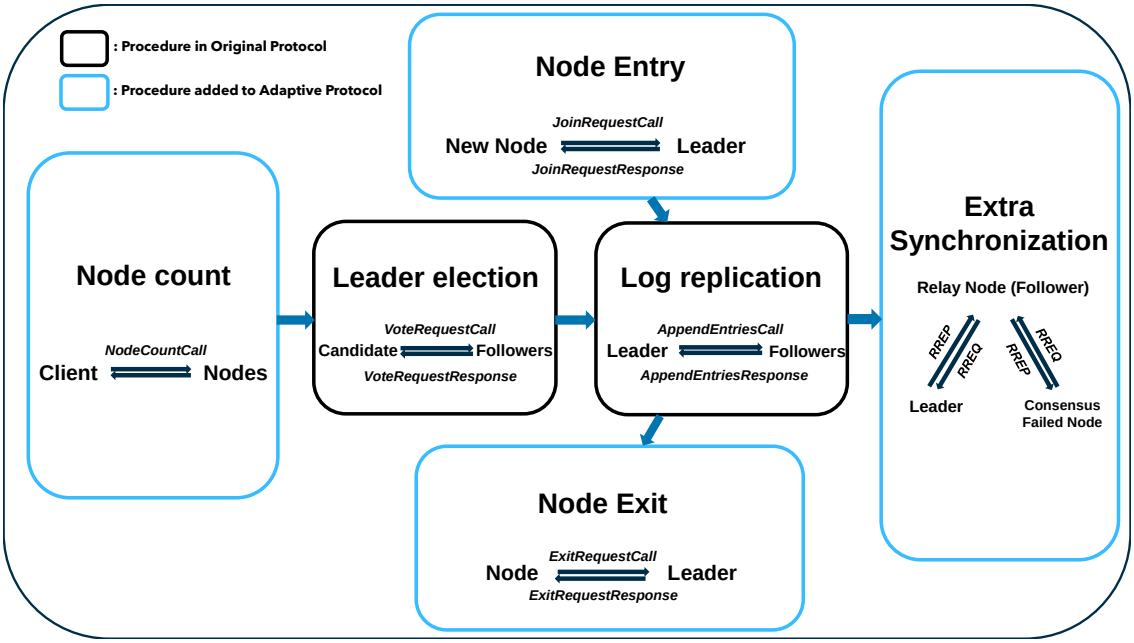

Figure 4.1: Main procedure of Raft's adaptive protocol in wireless network

4.1.1 Nodes Counting

Before initiating network construction, the total number of nodes in the network is unknown to each individual node. Therefore, the first phase of this adaptive protocol should involve determining the number of nodes participating in the consensus process. Once an accurate number of nodes is identified by the nodes that intend to join the consensus, the protocol can enter the stage of leader election. A client, which usually synchronizes the commands with the leader but cannot engage in consensus progress of Raft, can be used to estimate the number of nodes that are interested in constructing a new Raft-enabled wireless network.

The stage of node counting is initiated by broadcasting a *NodeCountCall* command to all nodes in a certain region. This *NodeCountCall* command contains information including the client's IP address, the type of consensus, and the network sequence. Nodes that support the specified consensus protocol and have not yet joined another network will re-

spond with an acknowledgment message when they receive the *NodeCountCall* command. The client node can repeat this node counting procedure several times and accumulate the number of acknowledgments to fully determine the total number of nodes that intend to join the consensus.

Algorithm 2 Node counting by client

```
1: Initialize client
2: RepeatCall ← R
3: N_count ← 0
4: for i ← 1 : RepeatCall do
5:    for j ← 1 : N_max do
6:       if NodeCountCall = 1 and NodeCountReply = 1 then
7:          N_count ← N_count + 1
8:       else
9:          N_count ← N_count
10:      end if
11:   end for
12: end for
```

Algorithm 2 presents a procedure of the nodes counting in this adaptive protocol of Raft. It shows that the individual node is counted if the client sends *NodeCountCall* to this node and receives *NodeCountReply* from it successfully in one round of communication. This node counting procedure can be repeated several times to fully count the number of nodes that intend to join distributed consensus until the total number of nodes reaches the maximum value of the network that can afford, and the counted nodes cannot respond to *NodeCountCall* when they receive it again. *RepeatCall* represents the times of *NodeCountCall* that broadcasts to uncounted nodes by the client. N_{max} is the maximum number of nodes that can join in the distributed consensus. Once the node counting is complete, the client node broadcasts a consensus permission message to the counted nodes. Upon receiving this permission, each node starts a random-length timer and transitions to the candidate state when the timer expires. Then these new candidates will start the leader election stage.

4.1.2 Leader Election

In a wired communication network, nodes start the leader election when they don't receive a heartbeat from the leader within a certain period. However, nodes may miss heartbeats more often due to issues like signal attenuation, interference, or congestion in a wireless network. Nodes in the wireless network usually have high mobility, which means they might come into contact with different subsets of the network at different times, which leads to issues like split votes or prolonged periods without a leader. Therefore, we increase the election timeout to improve the probability of a successful leader election and reduce the frequency of unnecessary elections.

In the adaptive protocol of Raft, the state of nodes is synchronized in stable storage before any type of call. Three types of states are assumed in the node for consensus, and they are shown in Table 4.1. T,VF, and LOG belong to the persistent state on all nodes, CI and AI are volatile states on all nodes, NI and MI are also volatile states but only function on the leader, which are reinitialized after the stage of leader election.

Table 4.1: States in the node

Notation	Definition
TC	Current term of consensus
VF	Candidate ID that this node vote for
LOG	Log entries that contain the commands and corresponding term
CI	Index of highest Log entry that is committed
AI	Index of highest Log entry applied to state
NI	Index of next log entry send send to nodes
MI	Index of highest log entry replicated on server (matched)

The commands from candidates in leader election refer to *VoteRequestCall*, and the commands from followers refer to *VoteRequestResponse*. Table 4.2 presents the types of commands in *VoteRequestCall* and *VoteRequestResponse*. Algorithm 3 presents the logic of leader election in this protocol. The logic of leader election in this stage basically follows the leader election from the original Raft protocol: The candidate broadcasts

the *VoteRequestCall* message to every follower in the network. Once a follower receives *VoteRequestCall*, it checks whether CT is smaller than the term of this follower FT. Otherwise, the follower will reply *VoteRequestResponse* with a *True* state in the $VOTE$ command. Otherwise, it will give a *False* state response, which means it cannot vote for this candidate.

Table 4.2: Commands of *VoteRequestCall* and *VoteRequestResponse* in leader election

Message	Command	Definition
VoteRequestCall	CT	Candidate Term
	CA	Candidate Address
	LLI	Index of Last Log Entry
	LLT	Term of Last Log Entry
VoteRequestResponse	FT	Candidate Term
	$VOTE$	Candidate Address

Algorithm 3 Leader Election Algorithm

```
Function LeaderElection(N_count, TT, TV)
TV ← Random(0, TT)
VOTE_C ← [0] * N_count
Candidates ← []
Followers ← []
for k ← 1 to Ncount do
    if TV(k) = 0 then
        Candidates.append(k)
        VoteRequest_k ← 1
        VoteResponse_k ← 0
    else
        Followers.append(k)
        VoteRequest_k ← 0
        VoteResponse_k ← 1
    end if
end for
for candidate in Candidates do
    for follower in Followers do
        backoff ← Random(0, MaxBackoff)
        sleep(backoff)
        Send VoteRequest(candidate, follower)
        ack_received ← VoteResponse(candidate, follower)
        if ack_received and VoteRequest[candidate] = 1 then
            VOTE_C[candidate] ← VOTE_C + 1
        end if
    end for
end for
Leader ← 1
MaxVotes ← −1
for i ← 1 to Ncount do
    if VOTE_C[i] > MaxVotes then
        MaxVotes ← VOTE_C[i]
        Leader ← i
    end if
end for
if MaxVotes ≥ (N_count − 1)/2 then
    print "Node", Leader, "is the leader"
else
    print "No leader elected"
    Leader Election(N_count, TT, TV)
end if
```

If the candidate can receive VOTEs from over $\frac{N_{count}-1}{2}$ nodes in the network before the timeout of its election, it can win the election and become the leader of the current consensus term. A countdown timer with a random time interval is set in candidates to ensure that they can start to broadcast the *VoteRequestCall* message at a different moment, which reduces the probability of conflicts in *VoteRequestResponse*.

4.1.3 Log Replication

The leader needs to synchronize its log entries with the majority of followers in the stage of log replication. In a wireless network, this stage could be affected by higher failure rates and variable latency. To reduce the impact of high consensus failed rate and latency, the leader needs to batch together multiple log entries with a pipeline approach and connect the failed node through a routing scheme. When the leader is successfully elected during the current term, Raft starts the log replication stage. The commands from the leader in the stage of log replication belong to *AppendEntriesCall*, and the commands from the followers in the stage of log replication refer to *AppendEntriesResponse*, which are presented in Table 4.3

Table 4.3: Commands in *AppendEntriesCall* and *AppendEntriesResponse*

Message	Command	Definition
AppendEntriesCall	***LT***	Leader Term
	LA	Leader Address
	NLI	Index of New Log Entry
	NLE	Term of New Log Entry
	LC	Leader Commit Index
AppendEntriesResponse	***FT***	Follower Term
	SR	Success Response

The logic of algorithm in log replication can be concluded that followers reply with a false status in the response if LT is less than the FT or if the NLI doesn't contain an entry with a matched term at the given NLT. If a conflict arises between existing and new entries, the existing entry is deleted, and all nodes follow the new one. New entries that have not been presented in the log should be appended, and if the leader's Commit Index LC is less than the follower's CI, the CommitIndex CI needs to be updated to the minimum value of LC. Algorithm 4 shows the completed program in the stage of log replication in Raft.

Algorithm 4 Log Replication

```
1: Term = 1
2: NLI[Leader] = NLI[Client]
3: AckCount = [0,...,0] of size (Ncount + 1)
4: for l = 1 to Ncount − 1 do
5:    if is_follower(l) then
6:       success, index = send_AppendEntriesCall(Leader, l)
7:       if success then
8:          AckCount[index] = AckCount[index] + 1
9:       else
10:         decrement_next_index(Leader, l)
11:      end if
12:   end if
13: end for
14: for i = 1 to Ncount do
15:   if AckCount[i] ≥ (Ncount − 1)/2 then
16:      Consensus = 1
17:      apply_log_entry(Leader, i)
18:      update_commit_index(Leader, i)
19:      break
20:   end if
21: end for
22: if Consensus ≠ 1 then
23:   leader_election()
24: end if
```

Leader election and log replication are two main stages of Raft. Nodes in two stages should follow the rules that all nodes should update their terms and convert to followers if they receive requests or responses with higher terms. Followers only respond to calls from candidates and a leader and become candidates if their election timeout elapses. Candidates initiate elections, increase their term, vote for themselves, reset their election

timer, and request votes from other nodes. They can become leaders only if they receive a majority of votes in one term. Otherwise, they need to revert to followers if they receive an *AppendEntriesCall* from a new leader. The leader sends initial *AppendEntriesCalls* to prevent election timeouts, append log entries upon receiving client commands, and synchronize entries with followers. The leader also needs to manage the next indices and matches indices for each follower and update its commit indices based on log entries' terms and majority acknowledgment from followers.

A consensus with high liveness represents the probability of log replication failure as negligible, ensuring that one leader node can consistently synchronize the log entries throughout multiple terms. In such a network with stable and robust distributed consensus, data is reliably replicated across nodes, maintaining data integrity and minimizing the risk of data loss or inconsistencies. Consequently, if we add necessary procedures in the adaptive protocol of Raft to improve the consensus reliability, the overall performance of the distributed consensus in the wireless network can be enhanced, allowing it to effectively manage and process transactions in challenging or dynamic network conditions.

4.1.4 Participant and Exit of Node

The distributed consensus in wireless networks may encounter the participant and exit of nodes more frequently than in wired networks, which can have a critical influence on the performance of consensus. Therefore, we present Algorithm 5 to show the procedure of node entry and exit in the adaptive protocol of Raft. When a new node intends to join the existing distributed consensus, it needs to send the *JoinRequest* command to the leader of the current term. If the leader receives *JoinRequest* and replies to it, this new node can participate in the state synchronization in the next round of log replication as other normal followers. Otherwise, it has to build the routing path to complete the state synchronization.

Algorithm 5 Node Participation and Exit in adaptive Raft protocol

```
FunctionHandleNodeJoin(newNode)
Send JoinRequest to leader
if leader replies then
    Participate in the next round of log replication
else
    Follow routing scheme
end if
FunctionHandleNodeExit(node)
nodeType ← DetermineNodeType(node)
if nodeType = Follower then
    Send ExitRequest to leader
    Leader stops sending AppendEntriesCall to node
    Leader deletes all routing paths involving node
    Update Routing Paths For State Synchronization
else if nodeType = Leader then
    Multicast ElectionCommand to all followers
    for f in followers do
        StartElectionTimer(f)
        ConvertToCandidate(f)
    end for
else if nodeType = ConsensusFailedNode then
    Send ExitRequest to leader through current routing path
    Leader deletes routing path
    Leader does not synchronize state of node in next round of log replication
end if
```

In a Raft-enabled wireless network, there are various scenarios in which different types of nodes may exit the network. If a follower wishes to leave the network, it sends *ExitRequest* to the leader, who then stops sending *AppendEntriesCall* to this follower and deletes all routing paths involving the follower. Destination nodes affected by these deleted paths must restart the routing protocol to update new routing paths for state synchronization. When a leader intends to leave the network, it multicasts an election command to all followers, prompting them to start an election timer. All existing routing paths expire, and new ones are established after the election. If a consensus failed node decides to leave the network, it sends a *ExitRequest* to the leader through the current routing path. The leader then deletes the routing path and does not synchronize the state of the failed node

in the next round of log replication. Moreover, the leader needs to eliminate the nodes that failed in state synchronization of several terms because they cannot build up any routing paths to the leader, which means these failed nodes have crashed and damaged the performance of distributed consensus.

Besides the scenarios in which nodes intend to exit the network, the case of nodes unintentionally exiting should also be considered in the adaptive protocol of Raft. A subset of nodes might be isolated from the rest of the network when the network partition happens. Although these nodes are still operational, they have essentially exited from the network's perspective because they can't communicate with the nodes in another partition. However, instead of halting the consensus process, the adaptive protocol aims to maintain consensus among the remaining nodes, which can be achieved by holding a mini-election within each partition and electing temporary leaders to keep the consensus process going. When the network partition is resolved, the system can trigger a new election to revote a single leader and reintegrate the partitions to preserve the consistency of the consensus.

Another case is when a node gets temporarily disconnected caused by interference or power loss, the adaptive protocol treats it as a temporary exit. During the disconnection period, the node is removed from the routing paths to prevent the leader from trying to communicate with an unavailable node, which can avoid unnecessary delays or failures in message transmissions. If the disconnected node rejoins the network, it should resume the original routing paths and can participate in the consensus process again. The node, upon rejoining, also needs to complete a state synchronization process to update any missed changes during its absence. All cases of node entry and exit can alter the ratio of followers and consensus failed nodes in the network. If the ratio of followers falls below half of the overall nodes, the protocol of Raft terminates the current term and initiates a new leader election.

4.1.5 Extra State Synchronization and Routing Scheme

In the original protocol of Raft, a successful consensus only requires the synchronization of states in over half of the nodes. However, in certain reliability-sensitive scenarios, it is crucial to eventually synchronize the states of failed nodes as well. These nodes may not have a reliable direct connection to the leader, necessitating a routing path to complete the extra state synchronization. The routing protocol plays a crucial role in maintaining the stability of the distributed consensus in two scenarios: when a new node intends to join or exit the network but cannot directly connect to the current leader; when a node's state must be synchronized after a failure during the log replication stage. In both cases, the routing protocol can establish connections between the affected nodes and the leader through a series of intermediary nodes. which ensures seamless communication and integration within the network to preserve system-wide consistency and reliability.

Table 4.4: Commands in *RREQ* and *RREP*

Message	Command	Definition
RREQ	*SNS*	Sequence Number of Source Node
	HN	Hopping Number
	RRID	RREQ Identity
RREP	*SND*	Sequence Number of Destination Node
	HN	Hopping Number
	SR	Success Response
	RPID	RREP Identity

Algorithm 6 Extra State Synchronization and Routing Scheme in Adaptive Raft Protocol

```
1: Function Handle_message(message)
2: if message.type is RREQ then
3:    Process_RREQ(message)
4: else if message.type is RREP then
5:    Process_RREP(message)
6: end if
7: Function Process_RREQ(RREQ)
8: Update_reverse_route(RREQ)
9: if node is the destination then
10:   Send_RREP(RREQ)
11: else
12:   Forward_RREQ(RREQ)
13: end if
14: FunctionProcess_RREP(RREP)
15: Update_forward_route(RREP)
16: if node is the source then
17:   Start_state_synchronization()
18: else
19:   Forward_RREP(RREP)
20: end if
21: Function Route_discovery(destination)
      Send_RREQ(destination)
      Create RREQ message with destination address
      Broadcast RREQ
22: Function Send_RREP(RREQ)
      Create RREP message with source address from RREQ
      Unicast RREP to the previous hop
23: Function Forward_RREQ(RREQ)
      Increment RREQ's hop count
      Broadcast RREQ
24: Function Forward_RREP(RREP)
      Unicast RREP to the previous hop
```

The routing scheme used in this adaptive protocol of Raft is derived from Ad hoc On-Demand Distance Vector (AODV) scheme from Mobile Ad Hoc Network (MANET). MANETs are wireless networks without a central controller, allowing nodes in the network to move freely and establish connections autonomously without the need for infrastructure support [91]. The AODV scheme is an on-demand routing strategy that establishes routes only when needed, which has good adaptability and scalability, effectively coping with

the high dynamics and topology changes of nodes in the network [92]. Other common routing protocols in MANET, such as Dynamic Destination-sequenced Distance-vector routing protocol (DSDV) [93] and Dynamic Source routing protocol (DSR) [94], can be implemented if they have satisfactory performance in the adaptive protocol of Raft. In the routing scheme of the adaptive protocol, it follows the rule that if the new node intends to join the network or the failure node in log replication cannot connect to the current leader, the leader node can act as the source node and broadcast RREQ (Routing Request message) to neighbor nodes to found an efficient routing path. Commands in the RREQ message are shown in Table. 4.4.

When a node receives an RREQ for the first time, it should create a reverse route entry for the source node in its routing table, with the previous hop, sequence number, and an incremented hop number (HN). If the node receiving the RREQ is not the destination node (the leader in the current term), it should rebroadcast the RREQ to its neighbors, updating the hop count in the process. Otherwise, it should initiate the RREP (Routing Response Message) process. When a node receives an RREP, it should update its routing table with a forward route entry for the destination node, including the next hop, destination sequence number, and hop count. This node should then forward the RREP towards the source node, following the reverse path established earlier. If a node receives a duplicate RREQ, it should discard the RREQ if it has already processed it or if the RREQ contains a lower or equal sequence number and hop count than the existing route entry. Intermediate nodes that have a valid route to the destination with a higher sequence number than the RREQ can send a gratuitous RREP to the source node, informing it of a better route. Route maintenance can be achieved through monitoring link breakages. When a node detects a link breakage, it should send a Route Error (RERR) message to the affected nodes. These nodes can then either re-initiate the route discovery process or use alternate routes if available.

4.1.6 Simulation Results

This subsection shows the simulation result of the proposed adaptive protocol of distributed consensus Raft. The simulation platform is MATLAB R2019B. The number of initial nodes is set to 20, and they are located randomly in the region of a square area with a side length of 10km and a center on the origin point. All nodes are assumed to be reliable when they process the commands, and they have the full function to complete the adaptive protocol, including broadcasting and unicasting messages, routing path storage, and timer activation. The position of nodes is assumed dynamic during the consensus, and they can enter and exit the consensus network until the number of nodes reaches the maximum value N_{max} that the protocol can support, which is set at 40 in the simulation. Nodes that exit the network are deleted in the simulation. The transmission failure rate of wireless links follows the Rayleigh Fading Model, where the distance between nodes acts as the large-scale effect [68]. The transmission time is set as 1 ms and the countdown timer of nodes ranges from 100 ms to 500 ms. The client, which is set at the original point as a static node, can only participate in the stage of node count but cannot act as any consensus node in the later stages of Raft.

The diagram depicted in Fig. 4.2 elucidates the comprehensive procedure of node counting that is managed by the client. It initiates the interaction by broadcasting a *NodeCountCall* message to all 20 nodes within the network. This message provokes a response from nodes to confirm their readiness to partake in consensus proceedings. However, dynamic networks mean that not all nodes may be responsive at all times. In this instance, Nodes 9 and 12 cannot send back the anticipated *NodeCountReply* message, which means the availability for the consensus process of Nodes 9 and 12 is ignored by the client node, and the total count of nodes that are participating in the consensus falls to 18. In a broader perspective, this procedure symbolizes the resilience provided by the adaptive protocol of Raft, showcasing their inherent ability to maintain functionality despite the

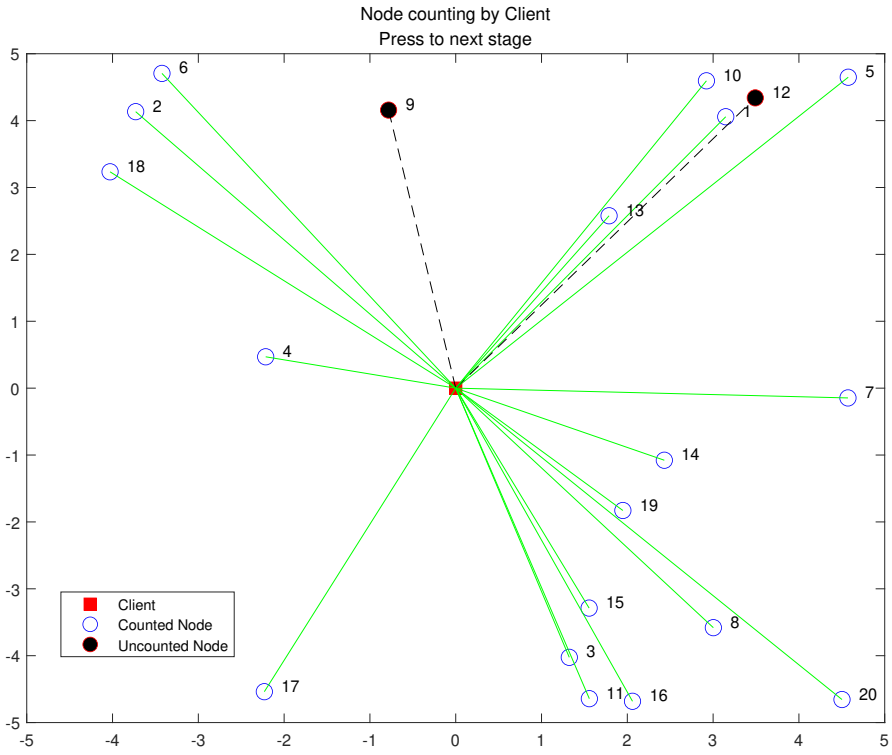

Figure 4.2: Node counting by client

occasional unresponsiveness of individual components. The consensus process proceeds with the remaining responsive nodes, ensuring the robustness and continuation of the network. The simulation result of node counting by the client indicates this stage can play a critical role in monitoring and managing the reliability of the entire system.

As the network protocol progresses to the stage of leader election, a distinctive shift in node behavior is observed in Fig. 4.3. Node 4 exhausts its pre-established timer and transforms its state to a candidate. Once this transformation occurs, Node 4 tries to win the vote by broadcasting a *VoteRequestCall* message to the other 17 active nodes within the network. Although communication link failures lead to Nodes 3, 5, 14, and 18 being unable to respond with their votes, Node 4 still successfully secures votes from other 76.47% nodes, which ensures Node 4 can work as the leader in this round of election.

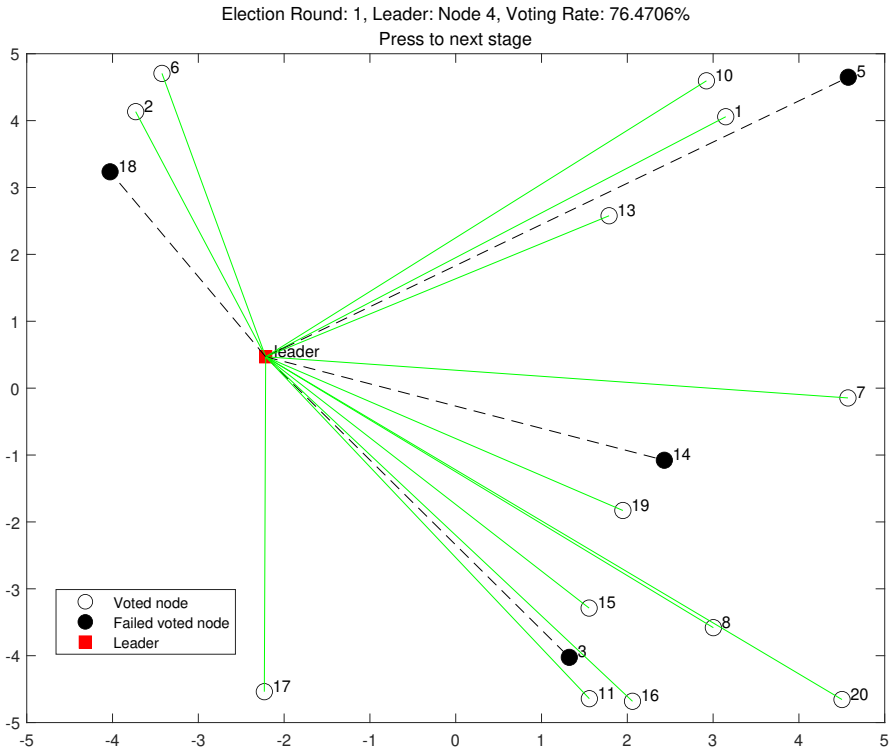

Figure 4.3: Election for leader

Figure. 4.4 clearly illustrates the procedure involving the entry and exit of nodes within a Raft-enabled wireless network. After the successful completion of the leader election stage, Node 21 and Node 22 emerge on the scene. They managed to join the existing network by establishing successful communication with the recently elected leader, Node 4. Their entrance not only adds to the network's size but also signifies their endorsement of Node 4's leadership, as they contribute their votes towards its commitment. This commitment indicates their states are identical to the overall state of the network, and they are prepared to maintain the continuity of consensus.

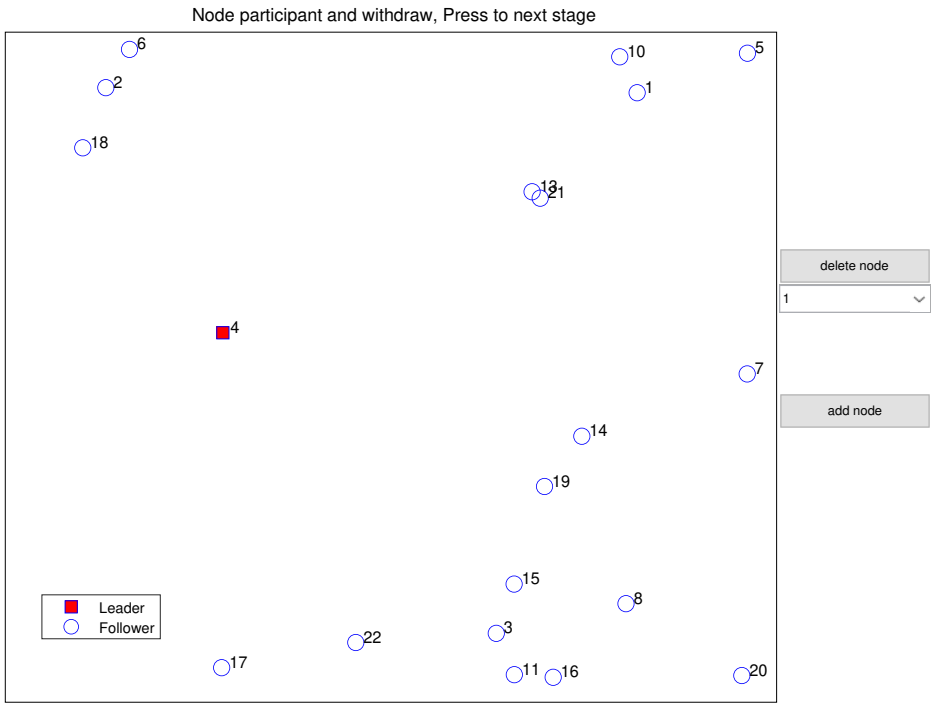

Figure 4.4: Node entry and exit

Following the acceptance into the network, Nodes 21 and 22 can then participate in subsequent crucial stages of the Raft protocol. They get actively involved in the log replication stage, which is essential for maintaining a consistent state across the majority of nodes in the network. They are also included in the state synchronization stage, a process that ensures that failed nodes can have an updated and accurate view of the network's state. This sequence of actions underlines the flexibility of the adaptive protocol of Raft in accommodating new nodes and integrating them into the existing network.

When new nodes enter the network, the total number of followers increases to 17. These nodes then proceed into the log replication phase, as shown in Figure 4.5. Unfortunately, only a disappointing 45% of the nodes successfully complete this process, and the rest of followers suffer communication failure. Node 4, despite its leadership role, struggles

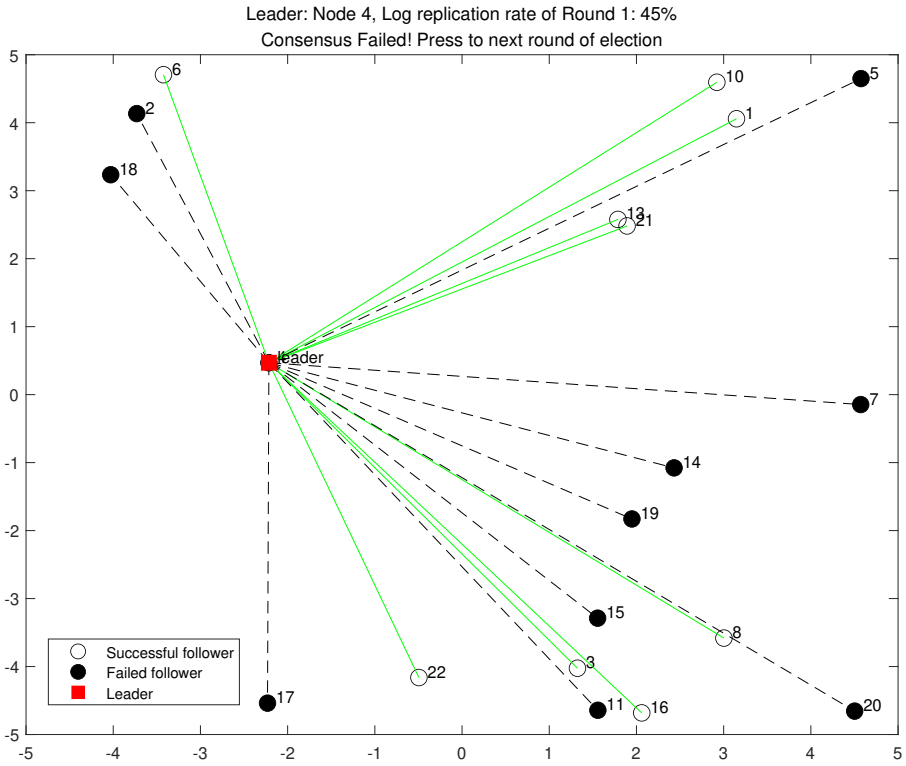

Figure 4.5: Failed log replication

to achieve state synchronization across the majority of the network nodes. This failure indicates the need for a significant adjustment within the network. Consequently, Node 4 relinquishes its leadership, converting into a follower instead, which triggers the initialization of a new leader election.

Figure 4.6 provides an overview of successful log replication after another round of leader election. It depicts that Node 2 has won the vote and assumes the mantle of leadership for the new term. The network experiences an improved level of success in the log replication stage, with 60% of the nodes successfully completing the task. This signifies that a consensus has been achieved within the network, allowing it to progress to the next round of log replication. Furthermore, additional state synchronization measures are instigated

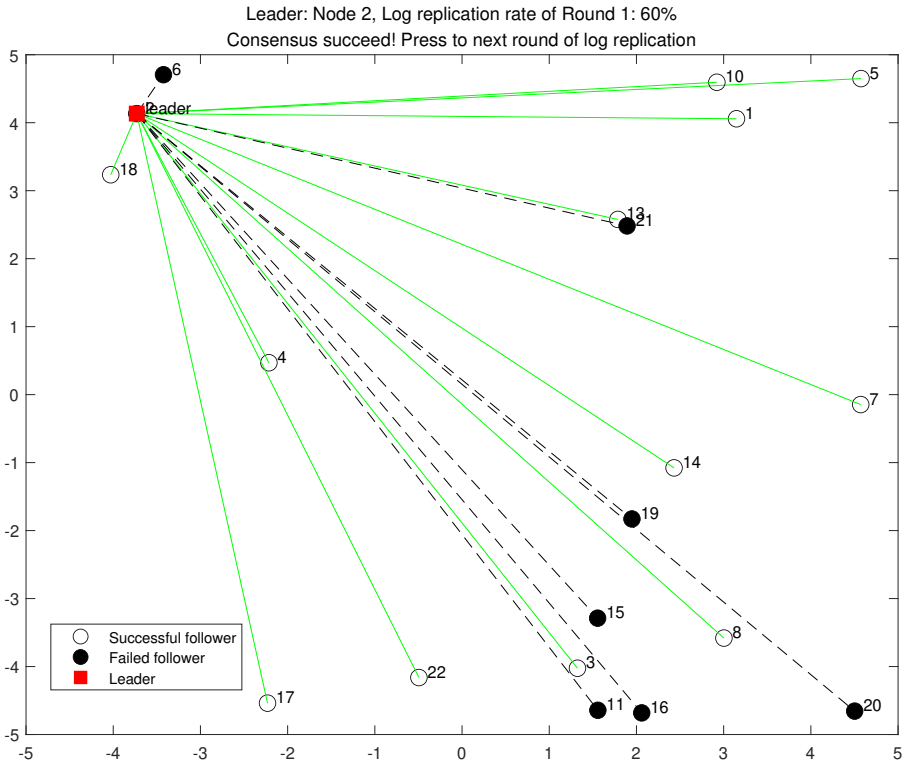

Figure 4.6: Successful log replication

for nodes that had failed in this process. This extra step ensures that the states of all nodes can be updated, thereby preserving the robustness and integrity of the system. It highlights the adaptive protocol's resilience and adaptability, capable of rectifying prior failures and ensuring functioning, even in the face of dynamic situations.

After a successful log replication, failed nodes need to synchronize their states, which can be achieved via a designated routing path and relay nodes. As illustrated in Figure 4.7, the routing path is determined using an Ad-hoc On-Demand Distance Vector (AODV)-like routing protocol, which is well-regarded for its efficiency. Three relay nodes 4, 12, and 18, are chosen from among the follower nodes that successfully completed the log replication. These nodes are selected due to their proximity to the leader, making them ideal for

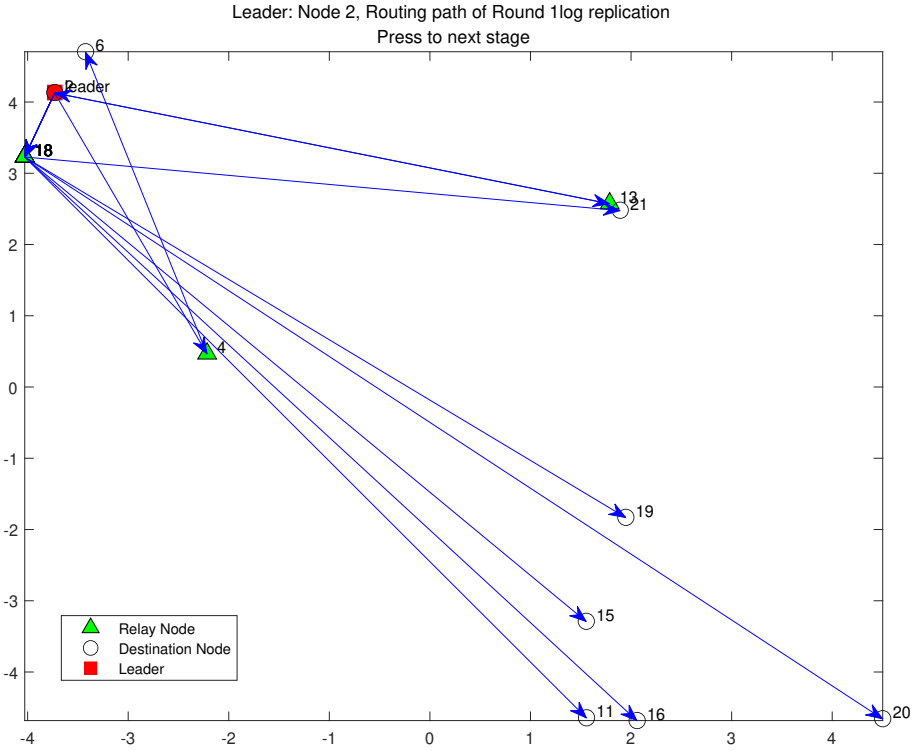

Figure 4.7: Routing Path for state synchronization

this crucial role. The selected relay nodes essentially bridge seven reliable links between the leader and failed nodes, thereby completing the process of state synchronization. This comprehensive routing scheme ensures that all nodes can keep a consistent state to maintain the integrity and functionality of the entire network.

The performance of adaptive protocol can also be analyzed from the perspective of consensus reliability and latency. And these two parameters are dominated by the routing scheme. [95], [96] indicates that the performance of routing scheme is mainly determined by the number of relay nodes. However, because every node that successfully achieve consensus has the same state, the routing path built by the adaptive protocol of Raft should have only one relay node, which means the performances are convergent even if multiple routing schemes of MANETs are implemented in the adaptive protocol.

Because Raft is a crash fault tolerance protocol, the security issue such as malicious attack tolerance is not a major concern under this circumstance. However, if the further work is about adaptive protocol of BFT, the relevant security analysis should be proposed as part of the protocol's performance.

4.2 Security analysis of Sharding

Most large-scale blockchain networks have a common problem with scalability and efficiency. In Proof of work (PoW) consensus, nodes need to consume time and computing power to solve the complex hash function [97]. The block broadcasting is also time-consuming because a completed transaction needs verification and state storage from every miner node. Therefore, the transaction throughput in a public blockchain network is much lower than traditional centralized transaction processors such as VISA, which can process thousands of transactions per second. In order to increase the transaction throughput in the blockchain system, the developers of public blockchain initially try to extend the size of the block to verify more transaction information in one block. However, the growing block size overloads the communication throughput and storage space in the database, which has less benefit in efficiency improvement.

To solve the scalability issue in blockchain systems, a concept called sharding that is from the distributed database has been implemented in new generation blockchain networks such as Ethereum 2.0 [98]. The sharding in the blockchain system aims to allocate transactions and validated nodes into different groups, which also refers to shards, which can process transactions in parallel. Sharding can be categorized in ways of transactions or ledger storage. The transactions in sharding include non-cross-shard or cross-shard transactions [99]. Non-cross-shard transaction corresponds to a transaction that happens in a single shard and the validated nodes of this transaction are all the nodes in this single shard. According to the principle of blockchain transaction verification, the size

of this single shard is closely related to the security level of non-cross-shard transactions in it. Cross-shard transaction refers to the transactions that happen between different shards, and the validating nodes are selected randomly from all shards even though most of them are unrelated to this transaction. Moreover, the type of sharding can be defined as transaction sharding or state sharding based on the way of ledger storage. Transaction sharding means every node of all shards will store a completed ledger that contains all verified transactions like traditional blockchain networks. In the state sharding, nodes in a shard will only store the verified transactions that have been processed by nodes in this shard instead of all shards. Therefore, it may require less storage space. Shards from the same blockchain network can implement different consensus mechanisms to fit the requirement of the decentralized application (dApp), which solves the scalability problem in a traditional blockchain system. In wireless blockchain system [100], the sharding can reduce the communication complexity of different consensus protocols [101], which saves the cost of corresponding communication resources in the blockchain network [102].

Several sharding protocols have been developed in a new blockchain network. They are making efforts to improve the scalability of systems even though there are still drawbacks to these protocols. Elastico [103] is the first sharding protocol design for a public blockchain network. It has combined Bitcoin PoW protocol and standard BFT but it only concerns about the way of transactions and network sharding. The formation of committees highly improves transaction throughput in the blockchain network, which is approximately proportional to the number of committees (shards). However, the transaction latency in Elastico is not affordable, even if the sharding number is small. Additionally, the normal committees in Elastico tend to have a limited number of nodes. This feature leads to the fact that after several transaction epochs, the probability of transaction failure could be tremendous. Even though Elastico has many drawbacks as a sharding protocol, it still has pointed out a direction to advanced sharding development for later design [104].

OmniLedger [105] is deployed as the Decentralized Ledger (DL) with a sharding structure. It is based on ByzCoin [106] and Hybrid consensus to select representative attesting nodes via scalable collective signing [107][108]. RandHound [109] is implemented in OmniLedger to distribute attesters to shards securely and ensure that shards are large enough to resist potential attacks. A two-phase client-driven lock/unlock protocol Atomix to let transactions commit or abort atomically during cross-sharding transactions. OmniLedger supports trust-but-verify validation to reduce transaction latency in low-value payment cases, and it allows the validators to switch between different shards securely and efficiently. However, the OmniLedger system epochs are time-consuming, and it requires advanced anti-censorship to detect unfairly censored transactions.

Ethereum, as the first decentralized Blockchain platform, implements a Turing-complete programming language for smart contracts development. The sharding in Ethereum 2.0 (ETH 2.0) aims to solve the severe issue of low scalability and transaction throughput in the ETH 1.0 network. Shards in ETH 2.0 may use different consensus mechanisms to reach the requirements of their own scenarios. The beacon chain, as the essential structure of ETH 2.0, will be implemented in Stage 0 of development. The primary function of the beacon chain is to assign an attesting committee randomly to verify the transactions or smart contracts in 1024 shards. The communication between the beacon chain and shards, which is through crosslinks, will generally be cross-shard communication. Beacon chain and other shards will use Casper FFG [110] to determine the canonical chain with Proof of Stake. Even though the development plan of Phase 0 in ETH 2.0 is explicit, the details of protocols have not been finalized [111].

All these references above only talk about the performance of their unique design even though they use similar random nodes distributed mechanisms. Therefore, it lacks a kind of general method to analyze the performance of sharding with random node distribution. The main contribution of this study is building a random nodes distributed sharding model to analyze the security performance with the conditions of cross-shard/non-cross-shard transactions and transaction/state sharding. The model initially calculates the probability

P of the assigned number of nodes in any shard. With this probability, the probability of a secure transaction in any shard can be determined, which represents the security level of the shard. Then the transaction throughput and communication throughput, which is related to the probability of secure transaction, will be conducted. The analysis is indicated in subsection 4.2.3. This shard analyzing model provides a general pattern to figure out the security performance of a specific sharding design and improve its reliability.

4.2.1 Security Model of Sharding

When link communication is stable among nodes, the reliability of distributed consensus is positively correlated to the number of validating nodes in it. However, the sharding divided the blockchain network into several smaller pieces and the number of validating nodes may decline, which can cause an inevitable decrease in security in transactions. In order to reduce the influence from less validating nodes, most sharding designs use Verifiable Random Function (VRF) [112] to provide randomness for validating node distribution. Before distributing the validating nodes randomly, the nodes will be categorized according to the specific bits of their hash value, which can be adjusted to change the difficulty of assigning functions. Some researchers have already optimized the node distribution methods in sharding to improve the secure transaction rate, such as Game-theoretic analysis [113] and Trust-Based shard distribution [114].

The analysis of this model depends on the types of sharding (transaction/state) and transaction (cross-shard/non-cross-shard). These types of sharding designs are presented in Fig. 4.8 and Fig. 4.9. The model assumes that N nodes with the same computing power are randomly distributed into M shards initially, and the total N nodes contain H malicious nodes in them, which will violate the transaction verification. If the number of malicious nodes h reaches 50% of the total nodes number k in a shard, the transaction validated by these nodes will fail, and it will not be recorded in the ledger. Normally, in distributed systems, the malicious node ratio R can influence the reliability of transaction

verification in blockchain, which is represented in (4.1)

$$R = H/N. \tag{4.1}$$

This section investigates the security performance in the sharding system when the malicious nodes ratio R changes. Table 4.5 shows all parameters set in this model.

Table 4.5: Parameter setting in sharding security analysis

Notation	Definition
N	Total number of nodes in the network
M	Number of shards
H	Total number of malicious nodes in the network
R	The ratio of malicious nodes in total nodes
k	The number of nodes distributed in any shards
h	The number of malicious nodes distributed in any shards
$P(k)$	Probability of k nodes distributed in a shard
$P(m)$	Probability of m nodes distributed in validating set
P_h	Probability of h malicious nodes in a shard
P_c	Probability of secure transaction in a shard
$P_H(k)$	Probability of h malicious nodes in a k nodes shard
$P_s(k)$	Probability of secure transaction in a k nodes shard

4.2.2 Non-cross-shard Transaction

Fig. 4.8 presents the main stages of the non-cross-shard transaction. The first stage is node distribution: All nodes from the original network are randomly distributed in several shards. During consensus, nodes cannot alter the shards they have been assigned to. The second stage is transaction verification: transactions can only happen between nodes that belong to the single shard, and the transaction can only be verified by nodes in this specific shard. The last stage is ledger storage, and it could be different in the transaction or state sharding. If all transaction records are still stored in every node like a traditional blockchain system, it will be defined as a transaction or network sharding. But if nodes in a shard only store the transaction processed in this specific shard, it will be state sharding. State sharding may require less memory space. However, it may conflict with the purpose of decentralization in the blockchain network. So this trade-off needs to be optimized in innovative sharding designs. Before analyzing the security performance of

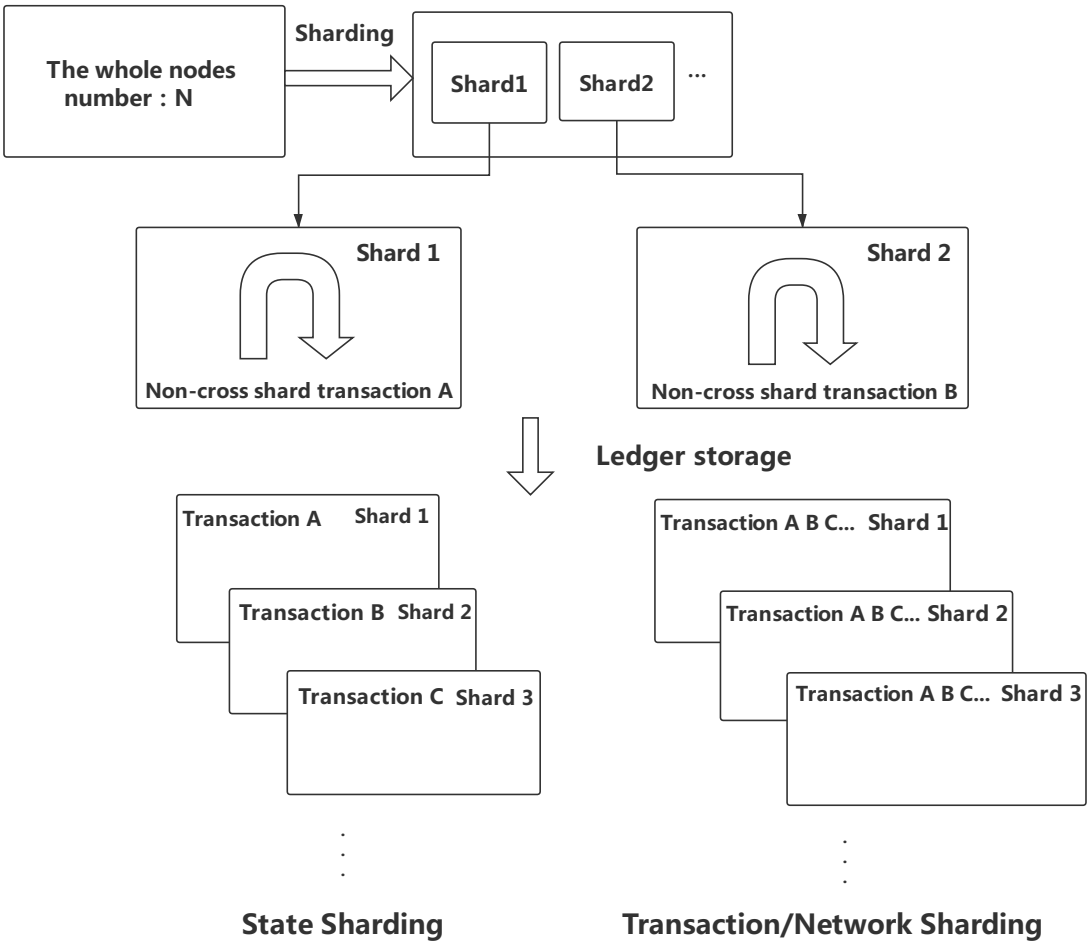

Figure 4.8: The non-cross-shard transaction

sharding, it is necessary to figure out the random process of node distribution to shards. The random distribution in sharding is similar to the dice-tossing problem. Each shard represents a side of the dice that has an equivalent probability of selecting an individual node. Therefore, the probability $P(k)$ that k nodes are randomly distributed to a shard can be given by a binomial distribution, which is implied in (4.2), which is influenced by shards number M and total nodes number N

$$P(k) = \binom{N}{k} (\frac{1}{M})^k (\frac{M-1}{M})^{N-k}. \tag{4.2}$$

With the theoretical probability $P(k)$ in every shard, the security level of sharding transactions can be analyzed. The probability $P_h(k)$ that N nodes within H malicious nodes have randomly distributed k nodes to a shard within h malicious nodes in this shard. $P_h(k)$ is similar to the problem of product sample check. There are total N products within H poor products. k products are selected from them, and $P_h(k)$ is the probability of h poor products from these k products, which follows the principle of Hyper-geometric Distribution. (4.3) indicates the probability distribution of P_h

$$P_h(k) = f(h,k,H,N) = \frac{\binom{H}{h}\binom{N-H}{k-h}}{\binom{N}{k}}. \tag{4.3}$$

Once $P_h(k)$ is determined, the successful transaction rate in this shard with k nodes could be calculated. With the consensus of PoW, if we assume the node number k in a shard is set and each node has identical computing power, the probability P_c of secure transaction processing in the shard can be accumulated by $P_h(k)$ until h reaches 50% of k

$$P_c(k) = \sum_{h=1}^{\frac{k}{2}} P_h(k) = \sum_{h=1}^{\frac{k}{2}} \frac{\binom{H}{h}\binom{N-H}{k-h}}{\binom{N}{k}}. \tag{4.4}$$

To obtain the eventual successful transaction rate $P_s(k)$, it's necessary to consider both the random nodes distribution k and malicious nodes distribution h in the shard. Therefore, for the probability that a shard has k nodes within h malicious nodes $P_H(k)$ is the production of $P(k)$ and $P_h(k)$ because these probabilities are independent to each other

$$P_H(k) = P(k) \cdot P_h(k). \tag{4.5}$$

If h is over 50% of k, the transaction verification will be controlled by malicious users, and the transaction in this shard will be insecure. The probability of secure transactions in one shard with specific nodes number k is $P_s(k)$, which is accumulated by the probability $P_H(k)$ that h is less than half of k while h and k are both uncertain

$$P_s(k) = \sum_{h=1}^{\frac{k}{2}} P_H(k) = P_c(k)P(k). \tag{4.6}$$

According to (4.4), (4.5), (4.6), $P_s(k)$ is only depends on the amount of k and ratio of malicious nodes R.

4.2.3 Cross-shard Transaction

The main stages of cross-shard transactions in Fig. 4.9 also include sharding and ledger storage. The difference from a non-cross-shard transaction is that the transaction can happen between nodes from different shards. It could be quite complicated in the stage of ledger storage if cross-shard transaction records are stored in the way of state sharding because the sharding system can hardly determine which parts of nodes are responsible for storing the records. The mainstream idea tends to let all nodes that are related to the transaction keep the ledger consistent, including transaction nodes and validating nodes. However, this mechanism may cause the issue that some nodes from the same shard may have different ledger contents, which means the blockchain system may lose state consistency.

In a cross-shard transaction, the probability $P(k)$ that k nodes are assigned to a shard is the same as (4.2) in the non-cross-shard transaction because the number of total nodes N and shards M do not change and the nodes are randomly assigned to shards. However, the security analysis in cross-shard is different from non-cross-shard. The nodes participating in transaction verification are not only from the transaction-relevant shards but also randomly assigned nodes from the whole blockchain network. In the case of validating,

it assumes the probability that any node participant validating is 20%. Therefore, the probability $P(m)$ that m nodes are chosen to be validators in a transaction is

$$P(m) = \binom{N}{m}(\frac{1}{5})^m(\frac{5-1}{5})^{N-m}. \tag{4.7}$$

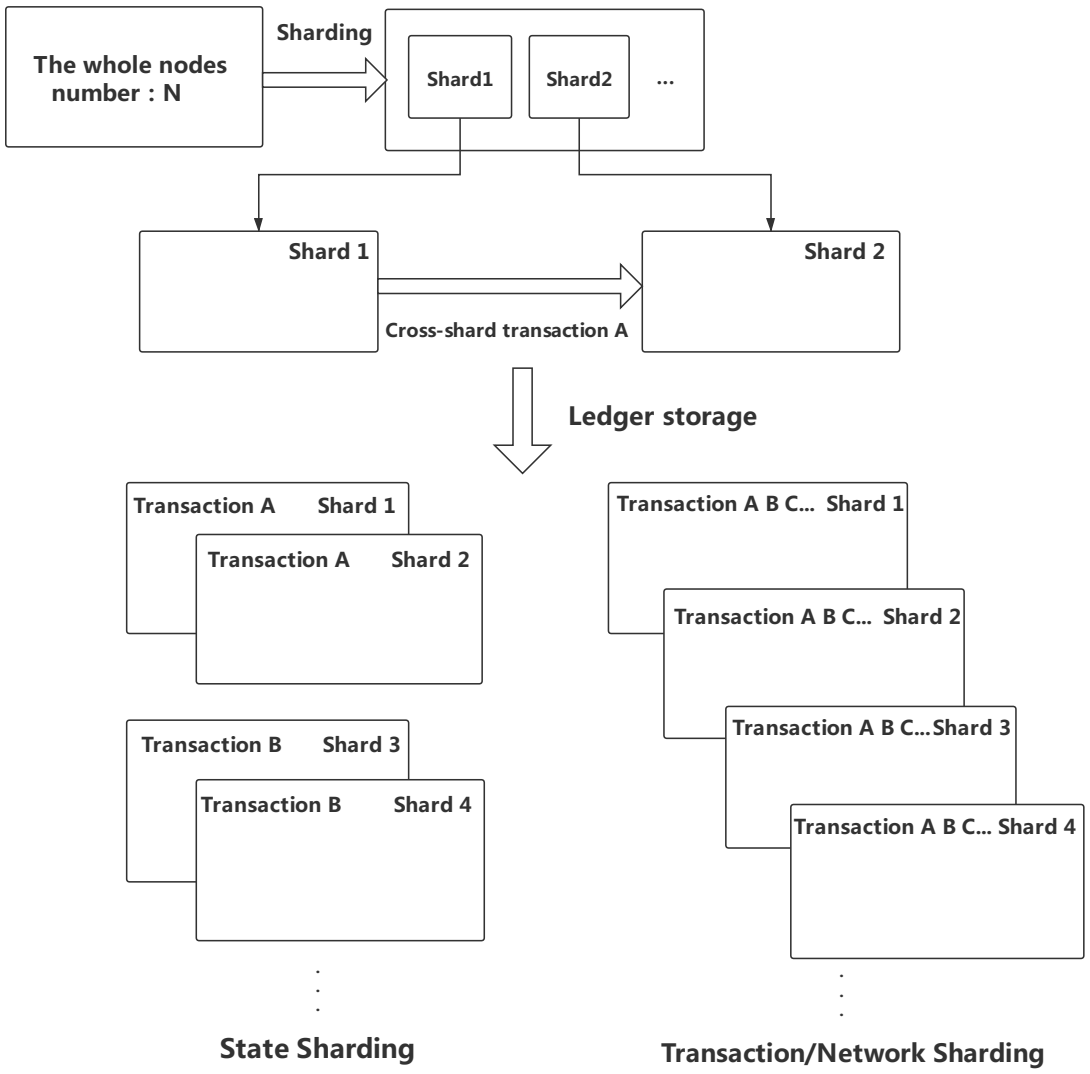

Figure 4.9: The cross-shard transaction

In cross-shard transaction, the probability that random x nodes within h malicious nodes engage in transaction validating from the blockchain network is P_H:

$$P_h(m) = f(h,m,H,N) = \frac{\binom{H}{m}\binom{N-H}{m-h}}{\binom{N}{m}}, \tag{4.8}$$

$$P_H(m) = P(m)P_h(m). \tag{4.9}$$

When h is less than 50% of m, the transaction will be secure. The probability of secured transaction with m validators P_s will be accumulated by P_H as malicious nodes number h is less than half of validating nodes number m

$$P_s(m) = \sum_{h=1}^{\frac{m}{2}} P_H(m) = \sum_{h=1}^{\frac{m}{2}} P(m)P_h(m). \tag{4.10}$$

Compared with the non-cross-shard transaction, the security of a cross-sharding transaction is influenced by the malicious nodes rate in randomly selected validating nodes set instead of the malicious nodes number k in the shard. If the size of validating nodes set is large enough, the difficulty in accomplishing Sybil attacks in a cross-shard transaction will be much more incredible than in a non-cross-shard transaction. Therefore, the security of cross-shard transactions can be improved. However, the cross-shard transaction may require more resources to support the communication for a massive validating network and solve the problem of ledger consistency.

The number of nodes N is set as 1000, and the number of shards M is set as 10 in the simulation. By comparing the tendency curves of $P(k)$ in theory and simulation while the nodes number k in a shard changing, the correctness of (4.1) will be determined. The random distribution progress in simulation is repeated 10^5 times to get a mean of probability $P(k)$ and then compared with the theoretical value of $P(k)$ in (4.2).

The comparison between analytical and simulated results is shown in Fig. 4.10, which concurs with the analytical value in equation 4.2 because the simulation points are overlapped with the analytical curve of $P(k)$. The result presents that $P(k)$ is mainly distributed at $k = \frac{N}{M}$, which means the size of shards will be close if the nodes are randomly distributed into the shards before the consensus progress.

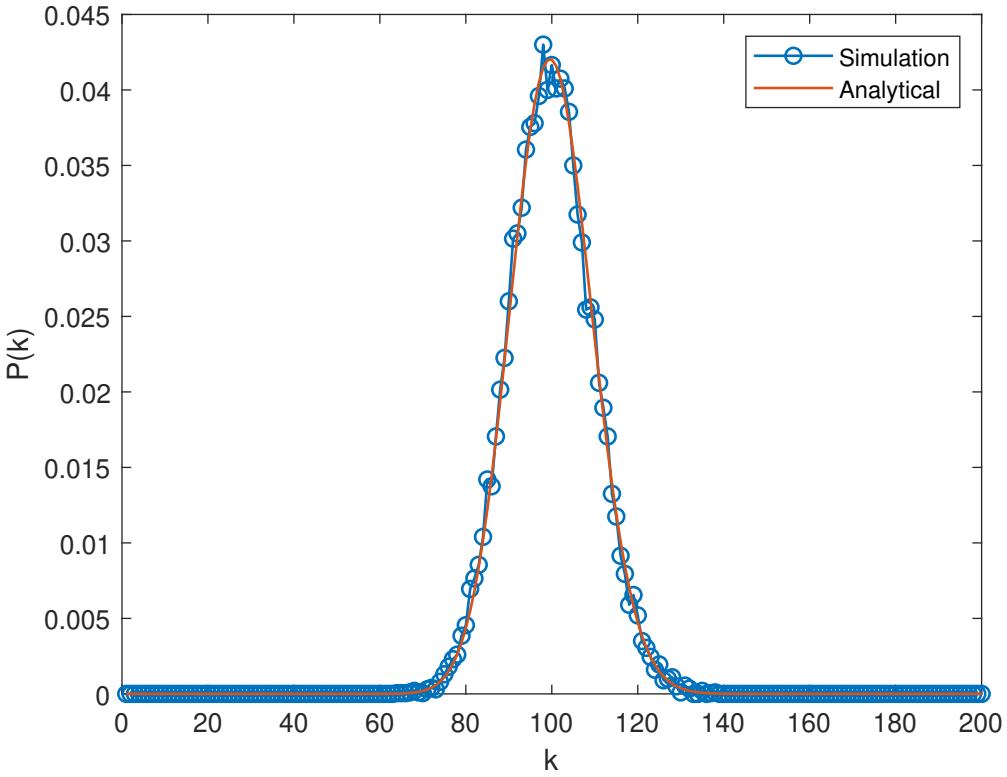

Figure 4.10: Probability $P(k)$ of nodes number k distributed in a shard

4.2.4 Simulation Result

Subsection 4.2.4 presents the crucial probabilities that represent the security level of non-cross-shard transactions in sharding. In Fig. 4.11, the probabilities P_c indicate that it can be influenced by R. When R is less than 50%, the probability curve of P_c will converge to 100% as h increasing.

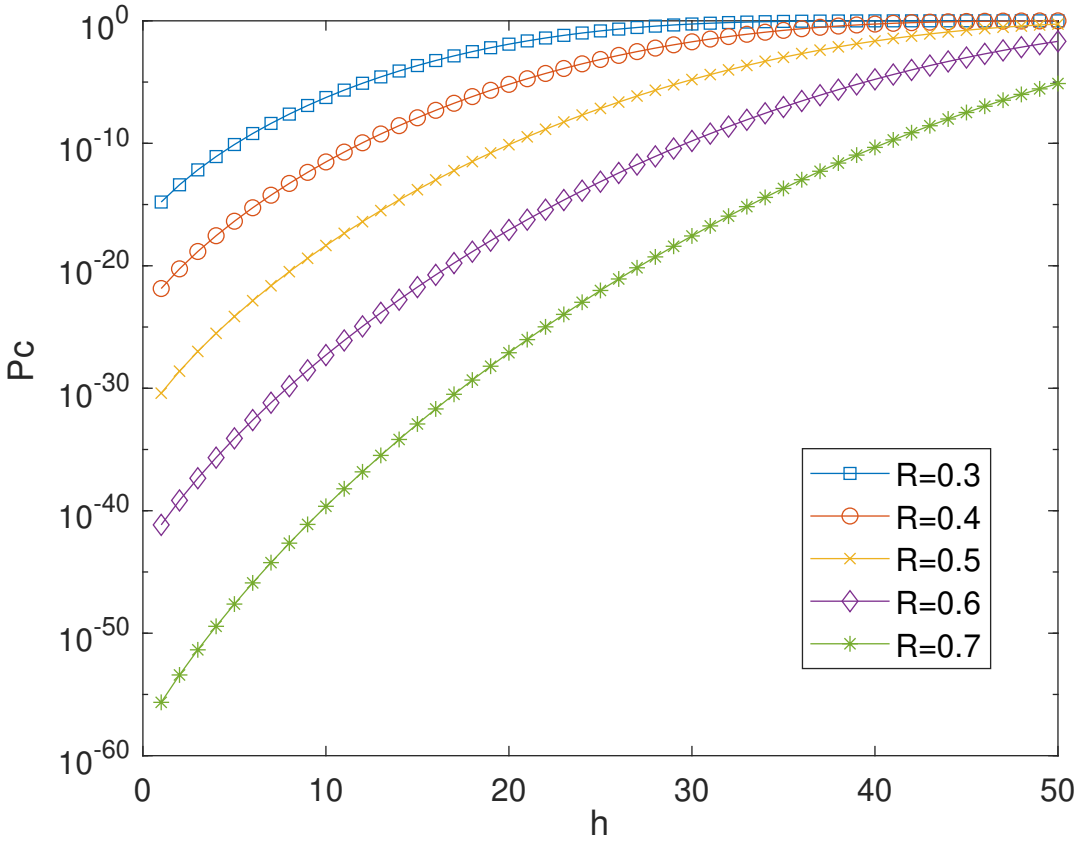

Figure 4.11: Probability P_c of secure transaction in a shard

The simulation result reveals that the probability P_H of the ratio of malicious nodes in a shard depends on the ratio of malicious nodes to all nodes in the whole blockchain network R. As R changes from 0.3 to 0.7, the corresponding number of malicious nodes to the peak value of $P_H(k)$ will rise up, which is indicated in Fig. 4.12.

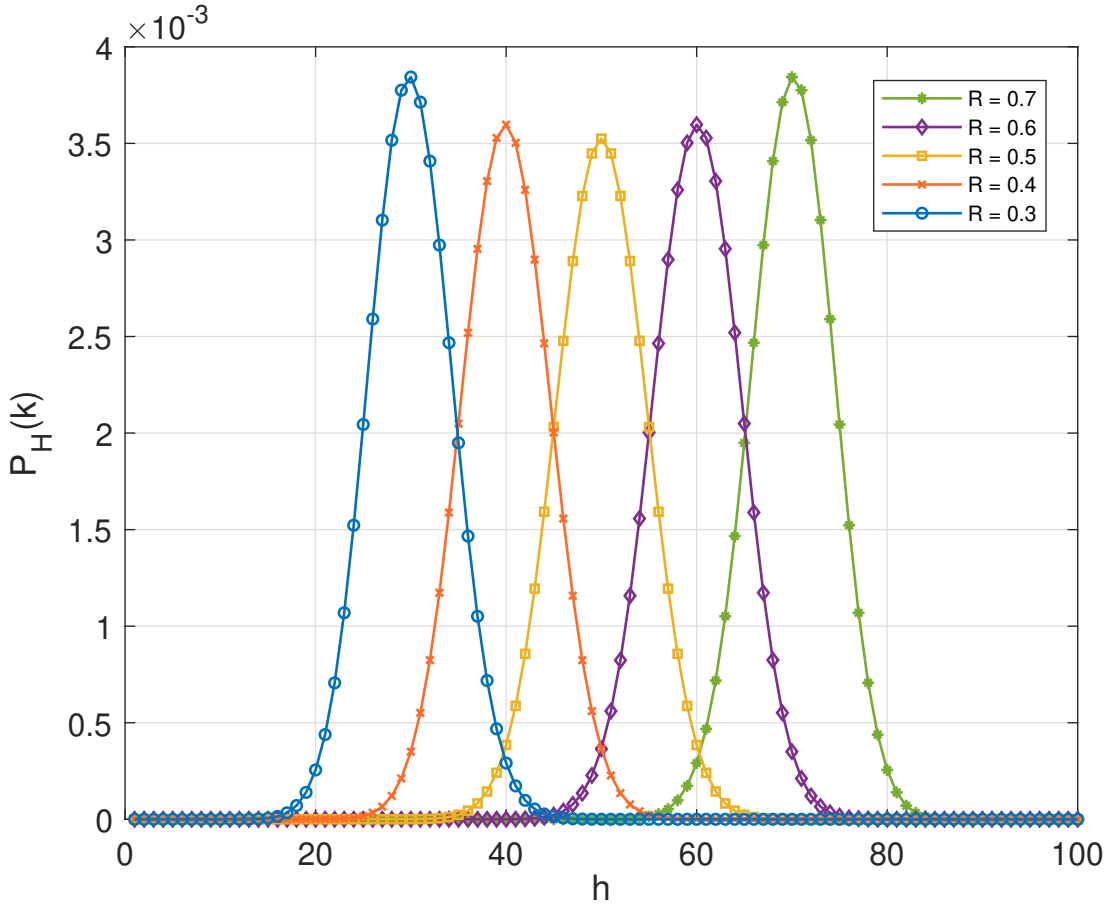

Figure 4.12: Probability of malicious nodes number h in a shard contains k nodes (P_H)

The result in Fig. 4.13 implies the dominated influence in $P_S(k)$ given by the rate of malicious nodes R. As R is less than 0.5, the peak value of $P_S(k)$ could be over 10^{-2}, which is considerable for a shard transaction when all values from the same curve are accumulated. However, if the number of malicious nodes M is over 50% of N, The successful transaction rate $P_S(k)$ will be less than 10^{-5}, which could be negligible from the perspective of transaction security.

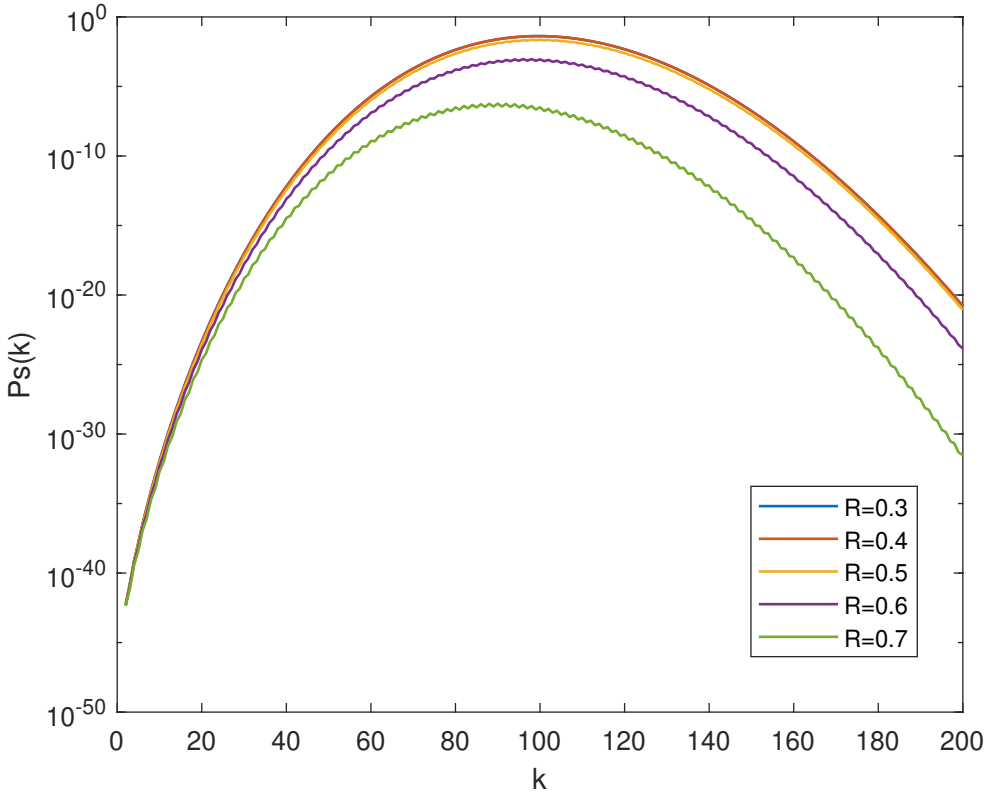

Figure 4.13: Probability of secure transaction in a shard with k nodes (P_S)

The simulation result of $P(m)$ in Fig. 4.14 is similar to $P(k)$ of non-cross-shard transaction in Fig. 4.11. But the peak value of the probability curve is changed because the number of validating nodes m in cross-shard transaction differs from shard's nodes number k in non-cross-shard.

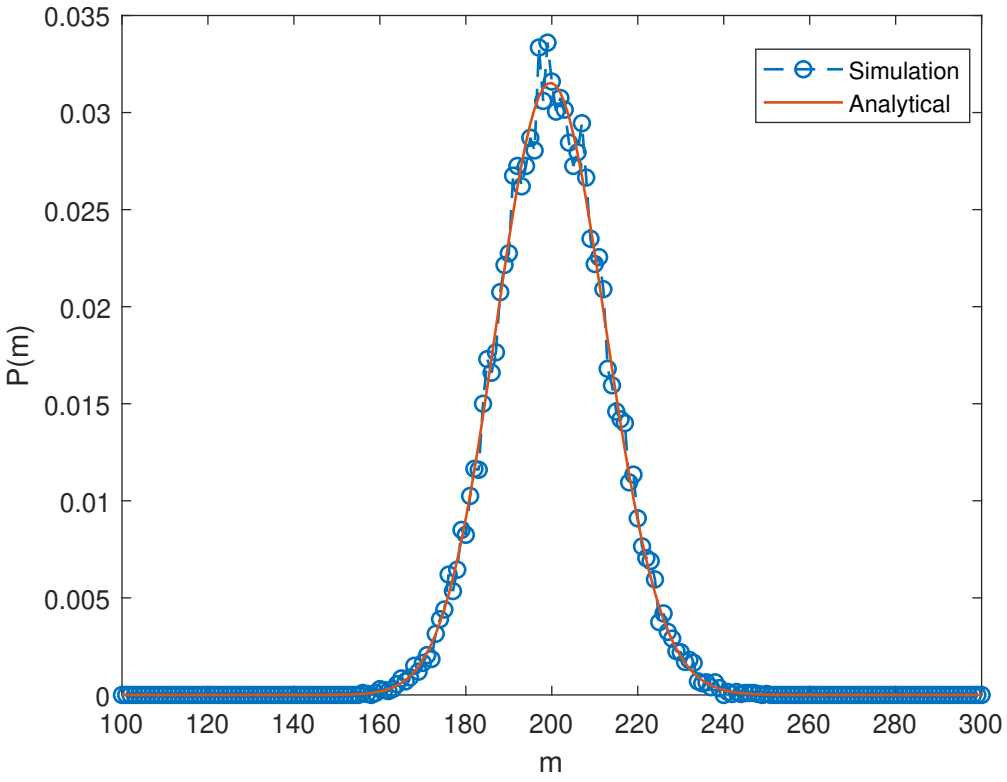

Figure 4.14: Probability of validating nodes number m in cross-shard transaction

As R is 0.3 and 0.4, the value of $P_S(m)$ is close in Fig. 4.15, which means R has less influence in the successful transaction rate of cross-shard transaction. But once malicious nodes occupy the majority of network ($R > 0.5$), the reliable transaction can hardly happen.

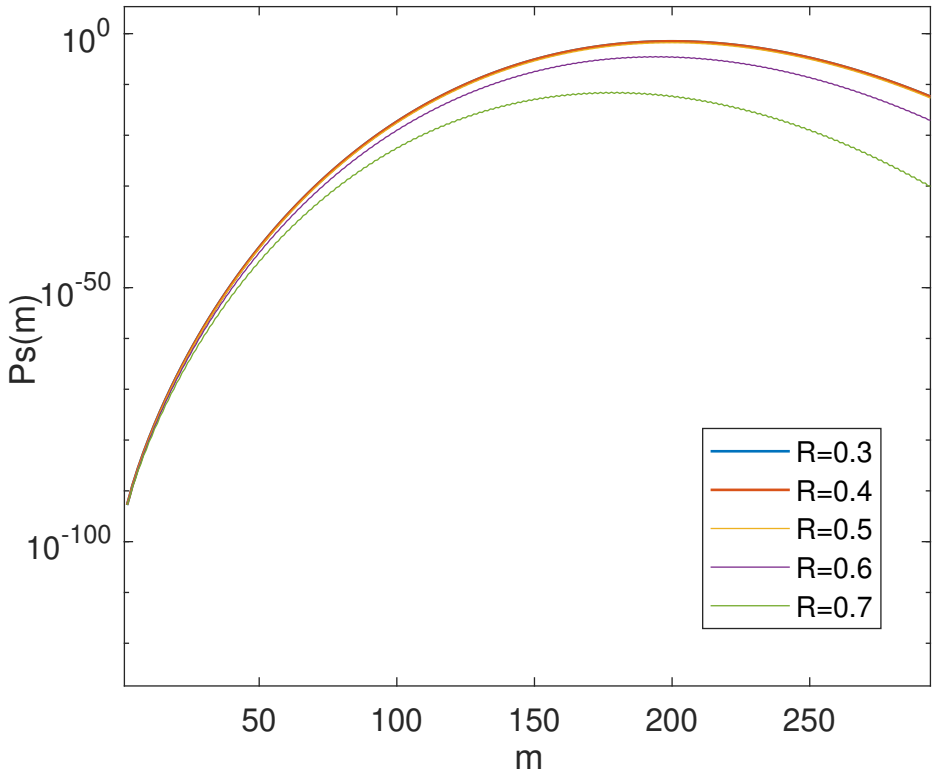

Figure 4.15: Probability of reliable cross-shard transaction (P_S)

The model assumes that the number of validating nodes m in the cross-shard transaction is larger than the size of shards k in the non-cross-shard transaction, which causes the corresponding vertical peak values of all curves to be right-shifted in the model of cross-shard transaction. It indicates that if attackers want to compromise the security of cross-shard transactions, they need to cost more computing power than non-cross transactions. In other words, the cross-shard transaction is normally more reliable than the non-cross-shard transaction in shards.

4.3 Conclusion

This section presents an adaptive protocol for the Raft consensus algorithm in wireless networks with dynamic network topology changes and potential data losses. It introduces stages including node counting, extra synchronization, and adaptive node entry/exit mechanism to the adaptive protocol of Raft. Simulation results validate the protocol's robustness and reliability in wireless environments, demonstrating its promise for real-world deployment in autonomous wireless systems, and opening new possibilities for distributed consensus in various wireless applications.

The content in this section also indicates the reliability and security analysis of sharding schemes in wireless distributed consensus. Most of the current sharding designs use a random distribution method to assign validating nodes from the whole network to complete the consensus. The analysis of the sharding model shows the security level can be affected by the rate of malicious nodes in both cross-shard transactions and non-cross-shard transactions. In the future, more advanced validating nodes distribution methods and consensus algorithms should be explored and applied in a wireless network to reduce the malicious node's influence on the shard's security performance.

Chapter 5

Conclusion and Future Trend

5.1 Conclusion

All models and experiment results that are shown in the chapters above focus on the reliability, latency, security, and scalability of wireless distributed consensus, which provides explicit performance analysis, optimization, and adaptive schemes for the practical application in wireless networks.

In Chapter 2, the analysis of consensus reliability of Raft concludes that with the constant communication link reliability, the consensus reliability in Raft increases monotonically along with the number of nodes. The relationship of consensus reliability with communication link reliability is interpreted in a linear form for simplicity. Meanwhile, the results show that the time latency is contradictory to the consensus reliability in Raft. After that, the link failed models of two distributed consensuses Raft and Hotstuff BFT, are analyzed to compare their performances to the centralized consensus in terms of full consensus reliability and time latency. The comparison of performance in centralized and distributed consensus indicates that the large-scale network with CFT protocol Raft can have higher full consensus reliability and lower latency than a centralized consensus when multiple rounds of synchronization are implemented. The large-scale network with Hotstuff BFT protocol has higher full consensus reliability than the centralized consensus when synchronization is implemented. However, a network with basic Hotstuff BFT needs to

consume more time slots in three phases to complete the synchronization when it is compared with the centralized consensus. Fortunately, the pipeline construction implemented in the chain Hotstuff BFT may provide a practical solution to this long latency issue in the Hotstuff BFT protocol.

Chapter 3 proposes optimal power and bandwidth allocation methods are proposed to improve reliability and reduce latency for the distributed consensus Raft in a wireless network. Both power and bandwidth allocation methods, which are derived through two different optimization algorithms, can reach near-optimal performance when the overall communication resource is constant. Besides the SQP and PSO algorithm that are used to solve the optimization problems, more state-of-art optimization methods with lower computational complexity can be implemented for more complicated optimization problems of resource allocation in wireless distributed consensus. Moreover, an optimized network size is defined to address the issue that the overall communication resources are inadequate to reach the required optimal performance of distributed consensus.

Chapter 4 brings attention to the scalability and applicability issue of distributed consensus in practical scenarios. The adaptive protocol of Raft is presented to improve the compatibility of distributed consensus in wireless networks through a routing scheme and node-exiting mechanism. Security analysis of sharding, which is a common scalability-improvement paradigm in blockchain systems, is investigated to find a reliable solution to deploy different distributed consensuses.

5.2 Future Trends

5.2.1 Extension of Current Researches

The advancement of wireless distributed consensus has opened up numerous avenues for research, particularly in areas where real-time decision-making and data synchronization are critical. The intersection of distributed consensus with practical applications in the realms of industry and transportation is poised to redefine how these systems operate.

Take, for instance, the evolving landscape of cooperative driving. As our roads become more populated with interconnected vehicles, the need for these vehicles to communicate seamlessly with each other becomes paramount. Whether it's for enhancing the flow of traffic or averting potential mishaps, the vehicles continuously exchange data about their respective speeds, positions, routes, and even potential obstacles. Each vehicle, in this context, isn't just a mode of transportation. It embodies the role of a network node, which not only sends and receives data but also processes it to make split-second decisions. If this data exchange is not accurate or consistent across all vehicles, the consequences could range from minor traffic disruptions to severe accidents. Herein lies the invaluable contribution of distributed consensus in such ecosystems. It ensures that irrespective of the volume of data or the number of nodes (vehicles) involved, there is a consistent and reliable understanding of the shared data. In essence, distributed consensus can act as a foundation that upholds the integrity, safety, and reliability of operations conducted by these interconnected vehicles.

Furthermore, as the horizons of wireless technologies expand, it is anticipated that distributed consensus will find its applications in various other sectors like smart cities, industrial IoT, and even remote healthcare. The potential of distributed consensus, when tethered to real-world applications, is not only vast but also transformative. In these scenarios, more adaptive protocols of distributed consensus will be implemented. The performance analysis of the adaptive protocols and corresponding optimization to the routing scheme should be investigate to reach the requirement of specific applications.

5.2.2 Promising Future Direction

Based on the extension of current research discussed in this section, the future direction of wireless distributed consensus can be summarized as follows:

- The integration of Artificial Intelligence (AI) represents a growing research trend for distributed consensus in wireless networks. AI is efficient in bolstering the performance of distributed consensus by optimizing communication resource allocation, and it's capable of identifying suspicious patterns of behavior in the network, such as potential Sybil attacks or intrusion attempts. AI can be applied for predictive modeling in distributed networks, forecasting the future states of nodes based on their historical data, thereby making the consensus process more efficient through the preemptive identification of potential conflicts or bottlenecks. Additionally, AI can be utilized to develop adaptive consensus protocols, which can dynamically adjust their parameters according to the network state. For instance, generative artificial intelligence (GAI) applications such as Chatgpt and PaLM have shown their tremendous potential in multiple-tasks processing. This ability can enable one node to process different types of distributed consensus protocols in wireless network. Moreover, GAI has proved its capability in wireless channel estimation [115], which can provide essential parameters for communication resource allocation and improve the performance of wireless distributed consensus.

- The assimilation of metaverse concepts and functionalities emerges as a cutting-edge direction for distributed consensus within wireless networks. In the sprawling digital expanse of the metaverse, where virtual environments intersect with real-world data, distributed consensus becomes vital to maintain a seamless and consistent user experience. Imagine virtual worlds where assets, ranging from digital real estate to avatars, need validation and synchronization across a vast network. Distributed consensus ensures that these assets maintain their integrity and value across various virtual environments, irrespective of the user's entry point into the metaverse. Furthermore, users are the most essential object to frequently make critical decisions in the virtual world. Their choices and interactions play a pivotal role in shaping the dynamics of the digital realm, influencing everything from the economic landscape to the social fabric of the metaverse. As they navigate through this virtual space, their decisions not only affect their individual experiences but also have far-reaching implications on the collective virtual ecosystem. This underscores the importance of robust and reliable distributed consensus mechanisms, ensuring that the metaverse operates smoothly and fairly, reflecting the collective will and actions of its diverse user base. Therefore, integrating distributed consensus with the metaverse could potentially revolutionize digital ownership and critical decision making management.
- Smart cities, with their interconnected systems and data-driven infrastructure, also stand to benefit immensely from advances in wireless distributed consensus. The urban environments of the future will hinge on numerous IoT devices, sensors, and systems communicating continuously. For example, traffic management systems, public transport scheduling, and even waste management will require real-time data exchanges. Distributed consensus guarantees that this data remains consistent across all nodes, ensuring that city services run smoothly and efficiently. This not only enhances the residents' quality of life but also ensures sustainable and optimal utilization of resources. Furthermore, in the realm of Industrial IoT (IIoT), distributed consensus is poised to redefine how manufacturing and production processes

are monitored and optimized. The factories of tomorrow will rely on a multitude of sensors, devices, and machines communicating simultaneously. Distributed consensus can ensure that production schedules, machine health data, or inventory levels are coherent and up-to-date across the entire facility. This leads to improved production efficiencies, reduced downtimes, and better resource allocation. By intertwining wireless distributed consensus with IIoT, industries can achieve unparalleled levels of automation and precision in their operations.

Appendices

A Proof of Equation (2.2)

In the summations of the binomial distributions of the consensus success rate P_C, the largest term dominates the summation if the link success rate P_l increases to reasonably large. Thus, the inner summation of the binomial distributions can be replaced by the largest term of it for simplification

$$\sum_{j=\frac{N-1}{2}}^{i} \binom{i}{j} {P_l}^j (1-P_l)^{i-j} \approx \binom{i}{i} {P_l}^i (1-P_l)^{i-i} = {P_l}^i. \tag{1}$$

The result of equation (1) can be substituted into the (2.1). According to cumulative distribution function of binomial distribution, the consensus failure rate 1-P_C is

$$1-P_C = \sum_{i=0}^{\frac{N-3}{2}} \binom{N-1}{i} {P_l}^i (1-P_l)^{N-1-i} \approx \binom{N-1}{\frac{N-3}{2}} (1-P_l)^{\frac{N+1}{2}}. \tag{2}$$

The largest term in the summation of the consensus failure rate in equation (2) is also dominating. Since P_l is reasonably large, the summation in equation (2) can be simplified in the same way of equation (1). And when the consensus failure rate $1-P_C$ is converted to logarithm form, it will correspond to the linear relation in equation (2.2)

$$\log(1-P_C) = (\frac{N+1}{2}) \log(1-P_l) + \log(\binom{N-1}{\frac{N-3}{2}}) + \Delta h, \tag{3}$$

where Δh is the corrected value of the intercept in equation (2.2) to get the minimum error between equation (2.1) and equation (2.2).

B Proof of Proposition 1

The dominant term of consensus reliability P_C from (3.10) is a discrete function, which means P_C cannot determine its tendency through derivation. If the Raft consensus with N followers can reach the maximum consensus reliability $P_C(N)$, $P_C(N)$ should be less than the consensus reliability of the network that contains $N-2$ and $N+2$ followers.

$$\begin{cases} P_C(N) > P_C(N+2) \\ \\ P_C(N) > P_C(N-2). \end{cases} \tag{4}$$

In the problem of communication resource allocation, if the network with N followers can reach the minimum consensus failure rate, the overall communication resource can be regarded as adequate for this network, which means the dominant term of (3.10) can replace the whole consensus reliability P_C. Therefore, the difference among the average link reliability P_l in the network of N, $N-2$, and $N+2$ followers can be negligible. The dominant term of the consensus failure rate is substituted into (4) to solve the N_{max}

$$\begin{cases} \frac{\binom{N}{f+1}(1-P_l{}^2)^{f+1}(P_l{}^2)^{N-f-1}}{\binom{N-2}{f}(1-P_l{}^2)^{f}(P_l{}^2)^{N-f-2}} < 1 \\ \\ \frac{\binom{N}{f+1}(1-P_l{}^2)^{f+1}(P_l{}^2)^{N-f-1}}{\binom{N+2}{f+2}(1-P_l{}^2)^{f+2}(P_l{}^2)^{N-f}} < 1. \end{cases} \tag{5}$$

Eventually, the conclusion in Proposition. 1 can be derived by replacing the number of fault tolerant nodes $f = \frac{N}{2}$ in (5) when the distributed consensus protocol is Raft.

Bibliography

[1] Leslie Lamport. 'Time, clocks, and the ordering of events in a distributed system'. In: *Concurrency: the Works of Leslie Lamport*. 2019, pp. 179–196.

[2] J.H. Wensley et al. 'SIFT: Design and analysis of a fault-tolerant computer for aircraft control'. In: *Proceedings of the IEEE* 66.10 (1978), pp. 1240–1255.

[3] Leslie Lamport, Robert Shostak and Marshall Pease. 'The Byzantine generals problem'. In: *Concurrency: the works of leslie lamport*. 2019, pp. 203–226.

[4] Leslie Lamport et al. 'Paxos made simple'. In: *ACM Sigact News* 32.4 (2001), pp. 18–25.

[5] Diego Ongaro and John Ousterhout. 'In search of an understandable consensus algorithm'. In: *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*. 2014, pp. 305–319.

[6] Miguel Castro, Barbara Liskov et al. 'Practical byzantine fault tolerance'. In: *OSDI*. Vol. 99. 1999. 1999, pp. 173–186.

[7] Bessem Zaabar et al. 'HealthBlock: A secure blockchain-based healthcare data management system'. In: *Computer Networks* 200 (2021), p. 108500.

[8] Christian Cachin et al. 'Architecture of the hyperledger blockchain fabric'. In: *Workshop on distributed cryptocurrencies and consensus ledgers*. Vol. 310. 4. Chicago, IL. 2016, pp. 1–4.

[9] Deepak Tosh et al. 'CloudPoS: A proof-of-stake consensus design for blockchain integrated cloud'. In: *2018 IEEE 11Th international conference on cloud computing (CLOUD)*. IEEE. 2018, pp. 302–309.

[10] Edoardo Gaetani et al. 'Blockchain-based database to ensure data integrity in cloud computing environments'. In: (2017).

[11] Ailidani Ailijiang, Aleksey Charapko and Murat Demirbas. 'Consensus in the cloud: Paxos systems demystified'. In: *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2016, pp. 1–10.

[12] Marshall Pease, Robert Shostak and Leslie Lamport. 'Reaching agreement in the presence of faults'. In: *Journal of the ACM (JACM)* 27.2 (1980), pp. 228–234.

[13] Leslie Lamport and P Michael Melliar-Smith. 'Synchronizing clocks in the presence of faults'. In: *Journal of the ACM (JACM)* 32.1 (1985), pp. 52–78.

[14] Leslie Lamport. 'Generalized consensus and Paxos'. In: (2005).

[15] Robbert Van Renesse and Deniz Altinbuken. 'Paxos made moderately complex'. In: *ACM Computing Surveys (CSUR)* 47.3 (2015), pp. 1–36.

[16] Leslie Lamport. 'Fast paxos'. In: *Distributed Computing* 19 (2006), pp. 79–103.

[17] Leslie Lamport and Mike Massa. 'Cheap paxos'. In: *International Conference on Dependable Systems and Networks, 2004*. IEEE. 2004, pp. 307–314.

[18] Mike Burrows. 'The Chubby lock service for loosely-coupled distributed systems'. In: *Proceedings of the 7th symposium on Operating systems design and implementation*. 2006, pp. 335–350.

[19] Patrick Hunt et al. 'ZooKeeper: wait-free coordination for internet-scale systems.' In: *USENIX annual technical conference*. Vol. 8. 9. 2010.

[20] Kevin Driscoll et al. 'Byzantine fault tolerance, from theory to reality'. In: *Computer Safety, Reliability, and Security: 22nd International Conference, SAFECOMP 2003, Edinburgh, UK, September 23-26, 2003. Proceedings 22*. Springer. 2003, pp. 235–248.

[21] Muhammad Saad et al. 'Exploring the Attack Surface of Blockchain: A Comprehensive Survey'. In: *IEEE Communications Surveys and Tutorials* 22.3 (2020), pp. 1977–2008.

[22] Maofan Yin et al. 'Hotstuff: Bft consensus with linearity and responsiveness'. In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. 2019, pp. 347–356.

[23] Mathieu Baudet et al. 'State machine replication in the libra blockchain'. In: *The Libra Assn., Tech. Rep* (2019).

[24] Harish Sukhwani et al. 'Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric)'. In: *2017 IEEE 36th symposium on reliable distributed systems (SRDS)*. IEEE. 2017, pp. 253–255.

[25] Jae Kwon and Ethan Buchman. 'Cosmos whitepaper'. In: *A Netw. Distrib. Ledgers* (2019), p. 27.

[26] Marta Lokhava et al. 'Fast and secure global payments with stellar'. In: *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 2019, pp. 80–96.

[27] Richard Zurawski. *Industrial communication technology handbook*. CRC Press, 2014.

[28] Satoshi Nakamoto. 'Bitcoin: A peer-to-peer electronic cash system'. In: *Decentralized Business Review* (2008), p. 21260.

[29] Richard Gendal Brown et al. 'Corda: an introduction'. In: *R3 CEV, August* 1.15 (2016), p. 14.

[30] David Schwartz, Noah Youngs, Arthur Britto et al. 'The ripple protocol consensus algorithm'. In: *Ripple Labs Inc White Paper* 5.8 (2014), p. 151.

[31] Cong Lin et al. 'Performance analysis for a wireless sensor network of star topology with random nodes deployment'. In: *Wireless Personal Communications* 97.3 (2017), pp. 3993–4013.

[32] Elli Androulaki et al. 'Hyperledger fabric: a distributed operating system for permissioned blockchains'. In: *Proceedings of the thirteenth EuroSys conference*. 2018, pp. 1–15.

[33] Leslie Lamport. 'The part-time parliament'. In: *Concurrency: the Works of Leslie Lamport*. 2019, pp. 277–317.

[34] Wenbing Zhao. 'Fast paxos made easy: Theory and implementation'. In: *International Journal of Distributed Systems and Technologies (IJDST)* 6.1 (2015), pp. 15–33.

[35] Arati Baliga et al. 'Performance evaluation of the quorum blockchain platform'. In: *arXiv preprint arXiv:1809.03421* (2018).

[36] Sebastian Pedersen, Hein Meling and Leander Jehl. 'An Analysis of Quorum-based Abstractions: A Case Study using Gorums to Implement Raft'. In: *Proceedings of the 2018 Workshop on Advanced Tools, Programming Languages, and PLatforms for Implementing and Evaluating Algorithms for Distributed systems*. 2018, pp. 29–35.

[37] Julien Polge, Jérémy Robert and Yves Le Traon. 'Permissioned blockchain frameworks in the industry: A comparison'. In: *Ict Express* 7.2 (2021), pp. 229–233.

[38] Dachao Yu and Lei Zhang. 'Centralized and Distributed Consensus in Wireless Network: An Analytical Comparison'. In: *2022 IEEE 20th International Conference on Embedded and Ubiquitous Computing (EUC)*. IEEE. 2022, pp. 81–89.

[39] Maofan Yin et al. 'HotStuff: BFT consensus in the lens of blockchain'. In: *arXiv preprint arXiv:1803.05069* (2018).

[40] Guy Golan Gueta et al. 'Sbft: a scalable and decentralized trust infrastructure'. In: *2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE. 2019, pp. 568–580.

[41] Arvind Giridhar and PR Kumar. 'Toward a theory of in-network computation in wireless sensor networks'. In: *IEEE Communications magazine* 44.4 (2006), pp. 98–107.

[42] Luca Schenato and Giovanni Gamba. 'A distributed consensus protocol for clock synchronization in wireless sensor network'. In: *2007 46th ieee conference on decision and control*. IEEE. 2007, pp. 2289–2294.

[43] Jianping He et al. 'Multiperiod scheduling for wireless sensor networks: A distributed consensus approach'. In: *IEEE Transactions on Signal Processing* 63.7 (2015), pp. 1651–1663.

[44] Bin Cao et al. 'Performance analysis and comparison of PoW, PoS and DAG based blockchains'. In: *Digital Communications and Networks* 6.4 (2020), pp. 480–485.

[45] Hao Xu et al. 'Blockchain-enabled resource management and sharing for 6G communications'. In: *Digital Communications and Networks* 6.3 (2020), pp. 261–269.

[46] Chenglin Feng et al. 'Wireless Distributed Consensus in Vehicle to Vehicle Networks for Autonomous Driving'. In: *IEEE Transactions on Vehicular Technology* (2022).

[47] Zongyao Li et al. 'Design and Implementation of a Raft based Wireless Consensus System for Autonomous Driving'. In: *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE. 2022, pp. 3736–3741.

[48] Bin Cao et al. 'When Internet of Things meets blockchain: Challenges in distributed consensus'. In: *IEEE Network* 33.6 (2019), pp. 133–139.

[49] Bin Cao et al. 'A many-objective optimization model of industrial internet of things based on private blockchain'. In: *IEEE Network* 34.5 (2020), pp. 78–83.

[50] Lei Zhang et al. 'How Much Communication Resource is Needed to Run a Wireless Blockchain Network?' In: *IEEE Network* (2021), pp. 1–8.

[51] Yao Sun et al. 'Blockchain-Enabled Wireless Internet of Things: Performance Analysis and Optimal Communication Node Deployment'. In: *IEEE Internet of Things Journal* 6.3 (2019), pp. 5791–5802.

[52] Hao Xu et al. 'Raft based Wireless Blockchain Networks in the Presence of Malicious Jamming'. In: *IEEE Wireless Communications Letters* (2020).

[53] Yuetai Li et al. 'RAFT Consensus Reliability in Wireless Networks: Probabilistic Analysis'. In: *IEEE Internet of Things Journal* 10.14 (2023), pp. 12839–12853.

[54] Hao Xu et al. 'Wireless Distributed Consensus for Connected Autonomous Systems'. In: *IEEE Internet of Things Journal* 10.9 (2023), pp. 7786–7799.

[55] Zhangchen Xu et al. 'Exact Fault-Tolerant Consensus with Voting Validity'. In: *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2023, pp. 842–852.

[56] Y Rao, J Jing et al. *New Services & Applications with 5G Ultra-reliable Low Latency Communication*. Tech. rep. 5G Americas, 2018.

[57] Beth Simone Noveck. 'The single point of failure'. In: *Innovating Government: Normative, policy and technological dimensions of modern government* (2011), pp. 77–99.

[58] M. Bennis, M. Debbah and H. V. Poor. 'Ultrareliable and Low-Latency Wireless Communication: Tail, Risk, and Scale'. In: *Proceedings of the IEEE* 106.10 (2018), pp. 1834–1853.

[59] Chengjian Sun et al. 'Optimizing resource allocation in the short blocklength regime for ultra-reliable and low-latency communications'. In: *IEEE Transactions on Wireless Communications* 18.1 (2018), pp. 402–415.

[60] B. Chang et al. 'Optimizing Resource Allocation in URLLC for Real-Time Wireless Control Systems'. In: *IEEE Transactions on Vehicular Technology* 68.9 (2019), pp. 8916–8927.

[61] U Leonhardt and H Paul. 'Phase measurement and Q function'. In: *Physical Review A* 47.4 (1993), R2460.

[62] Joseph Siryani, Bereket Tanju and Timothy J. Eveleigh. 'A Machine Learning Decision-Support System Improves the Internet of Things' Smart Meter Operations'. In: *IEEE Internet of Things Journal* 4.4 (2017), pp. 1056–1066.

[63] Francesco Piccialli et al. 'Decision Making in IoT Environment through Unsupervised Learning'. In: *IEEE Intelligent Systems* 35.1 (2020), pp. 27–35.

[64] Hexa-X. 'Expanded 6G Vision, use cases and key societal values'. In: (2020).

[65] Dachao Yu et al. 'Low Reliable and Low Latency Communications for Mission Critical Distributed Industrial Internet of Things'. In: *IEEE Communications Letters* (2020).

[66] Hao Xu et al. 'RAFT based wireless blockchain networks in the presence of malicious jamming'. In: *IEEE wireless communications letters* 9.6 (2020), pp. 817–821.

[67] Maarten Van Steen and Andrew S Tanenbaum. *Distributed systems*. Maarten van Steen Leiden, The Netherlands, 2017.

[68] Hyowoon Seo et al. 'Communication and Consensus Co-Design for Distributed, Low-Latency, and Reliable Wireless Systems'. In: *IEEE Internet of Things Journal* 8.1 (2021), pp. 129–143.

[69] Zhuo Sun et al. 'Two-Tier Communication for UAV-Enabled Massive IoT Systems: Performance Analysis and Joint Design of Trajectory and Resource Allocation'. In: *IEEE Journal on Selected Areas in Communications* 39.4 (2021), pp. 1132–1146.

[70] Ermin Sakic and Wolfgang Kellerer. 'Response time and availability study of RAFT consensus in distributed SDN control plane'. In: *IEEE Transactions on Network and Service Management* 15.1 (2017), pp. 304–318.

[71] Andrea Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[72] Norman C. Beaulieu and Jeremiah Hu. 'A closed-form expression for the outage probability of decode-and-forward relaying in dissimilar Rayleigh fading channels'. In: *IEEE Communications Letters* 10.12 (2006), pp. 813–815.

[73] Paul T Boggs and Jon W Tolle. 'Sequential quadratic programming'. In: *Acta numerica* 4 (1995), pp. 1–51.

[74] Morris Kline. *Calculus: an intuitive and physical approach*. Courier Corporation, 1998.

[75] Riccardo Poli, James Kennedy and Tim Blackwell. 'Particle swarm optimization'. In: *Swarm intelligence* 1.1 (2007), pp. 33–57.

[76] Petar Popovski. 'Ultra-reliable communication in 5G wireless systems'. In: *1st International Conference on 5G for Ubiquitous Connectivity*. 2014, pp. 146–151.

[77] Shunqing Zhang et al. '5G: Towards energy-efficient, low-latency and high-reliable communications networks'. In: *2014 IEEE international conference on communication systems*. IEEE. 2014, pp. 197–201.

[78] Dachao Yu et al. 'Security Analysis of Sharding in the Blockchain System'. In: *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE. 2021, pp. 1030–1035.

[79] Wenyu Li et al. 'A Scalable Multi-Layer PBFT Consensus for Blockchain'. In: *IEEE Transactions on Parallel and Distributed Systems* 32.5 (2021), pp. 1146–1160.

[80] Dachao Yu et al. 'Communication Resource Allocation of Raft in Wireless Network'. In: *IEEE Sensors Journal* (2023).

[81] Bin Cao et al. 'How does CSMA/CA affect the performance and security in wireless blockchain networks'. In: *IEEE transactions on industrial informatics* 16.6 (2019), pp. 4270–4280.

[82] David Eckhardt and Peter Steenkiste. 'Measurement and analysis of the error characteristics of an in-building wireless network'. In: *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*. 1996, pp. 243–254.

[83] Chenglin Feng et al. 'A Low Communication Complexity Double-layer PBFT Consensus'. In: *Wireless Blockchain: Principles, Technologies and Applications* (2021), pp. 73–92.

[84] Parma Bains. *Blockchain consensus mechanisms: A primer for supervisors*. International Monetary Fund, 2022.

[85] William Su, Sung-Ju Lee and Mario Gerla. 'Mobility prediction and routing in ad hoc wireless networks'. In: *International journal of network management* 11.1 (2001), pp. 3–30.

[86] Fengxian Guo et al. 'Adaptive Resource Allocation in Future Wireless Networks With Blockchain and Mobile Edge Computing'. In: *IEEE Transactions on Wireless Communications* 19.3 (2020), pp. 1689–1703.

[87] Gao Liu et al. 'B4SDC: A Blockchain System for Security Data Collection in MANETs'. In: *IEEE Transactions on Big Data* 8.3 (2022), pp. 739–752.

[88] Jian Wang et al. 'Lightweight blockchain assisted secure routing of swarm UAS networking'. In: *Computer Communications* 165 (2021), pp. 131–140.

[89] Du Mingxiao et al. 'A review on consensus algorithm of blockchain'. In: *2017 IEEE international conference on systems, man, and cybernetics (SMC)*. IEEE. 2017, pp. 2567–2572.

[90] Jagannathan Sarangapani. *Wireless ad hoc and sensor networks: protocols, performance, and control*. 2017.

[91] Jun-Zhao Sun. 'Mobile ad hoc networking: an essential technology for pervasive computing'. In: *2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No. 01EX479)*. Vol. 3. IEEE. 2001, pp. 316–321.

[92] Charles Perkins, Elizabeth Belding-Royer and Samir Das. *Ad hoc on-demand distance vector (AODV) routing*. Tech. rep. 2003.

[93] Charles E Perkins and Pravin Bhagwat. 'Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers'. In: *ACM SIGCOMM computer communication review* 24.4 (1994), pp. 234–244.

[94] David B Johnson, David A Maltz, Josh Broch et al. 'DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks'. In: *Ad hoc networking* 5.1 (2001), pp. 139–172.

[95] Omid Abedi, Mahmood Fathy and Jamshid Taghiloo. 'Enhancing AODV routing protocol using mobility parameters in VANET'. In: *2008 IEEE/ACS International Conference on Computer Systems and Applications*. IEEE. 2008, pp. 229–235.

[96] Asma Tuteja, Rajneesh Gujral and Sunil Thalia. 'Comparative performance analysis of DSDV, AODV and DSR routing protocols in MANET using NS2'. In: *2010 International conference on advances in computer engineering*. IEEE. 2010, pp. 330–333.

[97] Arthur Gervais et al. 'On the security and performance of proof of work blockchains'. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 3–16.

[98] Melanie Swan. *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

[99] Mahdi Zamani, Mahnush Movahedi and Mariana Raykova. 'Rapidchain: Scaling blockchain via full sharding'. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 931–948.

[100] Bin Cao et al. 'Wireless Blockchain: Principles, Technologies and Applications'. In: (2020).

[101] Yuefei Gao and Hajime Nobuhara. 'A proof of stake sharding protocol for scalable blockchains'. In: *Proceedings of the Asia-Pacific Advanced Network* 44.1 (2017), pp. 13–16.

[102] Wenying Xu et al. 'Event/self-triggered control for leader-following consensus over unreliable network with DoS attacks'. In: *IEEE transactions on neural networks and learning systems* 30.10 (2019), pp. 3137–3149.

[103] Loi Luu et al. 'A secure sharding protocol for open blockchains'. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 17–30.

[104] Guangsheng Yu et al. 'Survey: Sharding in blockchains'. In: *IEEE Access* 8 (2020), pp. 14155–14181.

[105] Eleftherios Kokoris-Kogias et al. 'Omniledger: A secure, scale-out, decentralized ledger via sharding'. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 583–598.

[106] Ewa Syta et al. 'Scalable bias-resistant distributed randomness'. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 444–460.

[107] Dan Boneh, Manu Drijvers and Gregory Neven. 'Compact multi-signatures for smaller blockchains'. In: (2018), pp. 435–464.

[108] Ewa Syta et al. 'Keeping authorities" honest or bust" with decentralized witness cosigning'. In: (2016), pp. 526–545.

[109] Eleftherios Kokoris Kogias et al. 'Enhancing bitcoin security and performance with strong consistency via collective signing'. In: *25th usenix security symposium*. 2016, pp. 279–296.

[110] Vitalik Buterin and Virgil Griffith. 'Casper the friendly finality gadget'. In: *arXiv preprint arXiv:1710.09437* (2017).

[111] James C Corbett et al. 'Spanner: Google's globally distributed database'. In: *ACM Transactions on Computer Systems (TOCS)* 31.3 (2013), pp. 1–22.

[112] Yevgeniy Dodis and Aleksandr Yampolskiy. 'A verifiable random function with short proofs and keys'. In: *International Workshop on Public Key Cryptography*. Springer. 2005, pp. 416–431.

[113] Mohammad Hossein Manshaei et al. 'A game-theoretic analysis of shard-based permissionless blockchains'. In: *IEEE Access* 6 (2018), pp. 78100–78112.

[114] Jusik Yun, Yunyeong Goh and Jong-Moon Chung. 'Trust-Based Shard Distribution Scheme for Fault-Tolerant Shard Blockchain Networks'. In: *IEEE Access* 7 (2019), pp. 135164–135175.

[115] Eren Balevi and Jeffrey G Andrews. 'Wideband channel estimation with a generative adversarial network'. In: *IEEE Transactions on Wireless Communications* 20.5 (2021), pp. 3049–3060.