Toward Improving the Internet of Things: Quality of Service and Fault Tolerance Perspectives

Dissertation by

Maha Saleh Alaslani

In Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

March, 2021

EXAMINATION COMMITTEE PAGE

The dissertation of Maha Saleh Alaslani is approved by the examination committee

Committee Chairperson: Prof. Basem Shihada, KAUST
Committee Members: Prof. Mohamed-Slim Alouini, KAUST, Prof. Xiangliang Zhang, KAUST, Prof. Alysson Bessani, University of Lisbon

ABSTRACT

Toward Improving the Internet of Things: Quality of Service and Fault Tolerance Perspectives

Maha Saleh Alaslani

The Internet of Things (IoT) is a technology aimed at developing a global network of machines and devices that can interact and communicate with each other. Supporting IoT, therefore, requires revisiting the Internet's best effort service model and reviewing its complex communication patterns.

In this dissertation, we explore the unique characteristics of IoT traffic and examine IoT systems. Our work is motivated by the new capabilities offered by modern Software Defined Networks (SDN) and blockchain technology. We evaluate IoT Quality of Service (QoS) in traditional networking. We obtain mathematical expressions to calculate end-to-end delay, and dropping. Our results provide insight into the advantages of an intelligent edge serving as a detection mechanism. Subsequently, we propose SADIQ, SDN-based Application-aware Dynamic Internet of things QoS. SADIQ provides context-driven QoS for IoT applications by allowing applications to express their requirements using a high-level SQL-like policy language. Our results show that SADIQ improves the percentage of regions with an error in their reported temperature for the Weather Signal application up to 45 times; and it improves the percentage of incorrect parking statuses for regions with high occupancy for the Smart Parking application up to 30 times under the same network conditions and drop rates.

Despite centralization and the control of data, IoT systems are not safe from cyber-crime, privacy issues, and security breaches. Therefore, we explore blockchain technology. In the context of IoT, Byzantine fault tolerance-based consensus protocols

are used. However, the blockchain consensus layer contributes to the most remarkable performance overhead especially for IoT applications subject to maximum delay constraints. In order to capture the unique requirements of the IoT, consensus mechanisms and block formation need to be redesigned. To this end, we propose Synopsis, a novel hierarchical blockchain system. Synopsis introduces a wireless-optimized Byzantine chain replication protocol and a new probabilistic data structure. The results show that Synopsis successfully reduces the memory footprint from Megabytes to a few Kilobytes with an improvement of 1000 times. Synopsis also enables reductions in message complexity and commitment delay of 85% and 99.4%, respectively.

ACKNOWLEDGEMENTS

This work is dedicated to the greatest man and women.

To my beloved parents...

All the praises and thanks to Allah, Lord of the worlds. The Entirely Merciful, the Especially Merciful, for all his blessings to my family and me. For the strength he gives me each day and for all the people around me who make life more meaningful.

I would like to express my sincere gratitude to my research supervisor, Prof. Basem Shihada. I would like to thank him for his friendship, empathy, and a great sense of humble. I would also like to thank Prof. Mohamed-Slim Alouini, Prof. Xiangliang Zhang, and Prof. Alysson Bessani for being part my committee, and for all their time. I would also like to especially thank Enas Ahmad, Prof. Ahmad Showail, Prof. Faisal Nawab, and my friends for all the time and support during my research journey.

I would also like to express my extreme gratitude to my parents. They believed in me, protected me, strengthened me, and loved me unconditionally. There are not enough words to describe what they mean to me and what an influence they have on my life.

I would also like to express my deepest gratitude to my best friend, my loving, and supportive husband. Thank you for being the person that completes my life. Finally, my heartfelt thanks to my sisters and brothers for their support and for loving me more than I ever loved myself.

TABLE OF CONTENTS

Examination Committee Page **2**

Copyright **3**

Abstract **4**

Acknowledgements **6**

Table of Contents **7**

List of Figures **10**

List of Tables **13**

1 Introduction **14**

1.1 Problem Statement and Motivation 14

1.2 Thesis Objectives 22

1.3 Thesis Contributions 23

1.4 Thesis Organization 25

2 Background and State of the Art **27**

2.1 Internet of Things 27

2.2 Internet of Things Challenges 28

2.3 Internet of Things Communication Protocols 30

2.4 Internet of Things Computing Paradigm 32

2.5 Internet of Things Network Paradigm 33

2.5.1 Traditional Internetworking 33

2.5.2 Software Defined Networking (SDN) 34

2.5.3 Blockchain Technology 38

3 General IoT System Modeling and Analysis 51
3.1 Queuing Modeling and Kendall Notation 51
3.2 IoT-based System Model 54
3.3 End-to-End Delay and Dropping Analysis 55
3.4 Model Analysis 58
3.5 Performance Evaluation 63

4 Traffic Volume and Load Modeling and Analysis 69
4.1 Instantaneous Detection of IoT Volumetric Traffic Model 69
4.2 Model Analysis 71
4.3 Performance Evaluation 74

5 Context-Driven QoS-Enabled Networking 81
5.1 QoS Requirements 81
5.2 SADIQ: SDN-based Application Aware Dynamic IoT QoS 82
5.2.1 IoTFlow Abstraction 84
5.2.2 Policies 85
5.2.3 QoS Controller 89
5.2.4 Complete illustrative example 92
5.3 SADIQ versus SDN-based QoS-enforcement Proposals 93
5.4 Implementation 94
5.4.1 QoS Controller 94
5.4.2 IoTFlow Abstraction 95
5.5 Performance Evaluation 95

6 Blockchain Modeling and Analysis 104
6.1 Blockchain-IoT Integration Requirements 105
6.2 Byzantine-Based Blockchain System Model 107
6.2.1 Delay Threshold Analysis 108
6.2.2 Network Hops Analysis 109
6.2.3 Consensus Replicas Analysis 112
6.3 Performance Evaluation 114

7 Fault-Tolerant Scalable Networking 120
7.1 Synopsis: a Scalable Byzantine Distributed Ledgers 121
7.2 Synopsis Data Format 122
7.3 Synopsis Consensus 127

7.3.1 Problem Assumptions 127
7.3.2 Problem Definition 128
7.3.3 Synopsis Chain Replication Protocol 129
7.3.4 Synopsis Chain Replication Analysis and Discussion 131
7.3.5 Efficiency and Chain Replication Optimization 133
7.3.6 Global Blockchain Commitment 136
7.3.7 Global Agreement Analysis and Discussion 138
7.3.8 Recovery from failures 139
7.3.9 Correctness 140
7.4 Synopsis Consensus versus BFT Proposals 142
7.5 Performance Evaluation 145
7.5.1 Large Scales Simulation 146
7.5.2 Real Testbed Performance 155

8 Concluding Remarks **159**
8.1 Summary 159
8.2 Open Directions and Future Works 162

References **165**

Publications **178**

LIST OF FIGURES

1.1 Smart parking actions on New Year eve for Melbourne. Some areas see significantly more activity compared to other regions. 19
1.2 CDF in log scale of the number of samples per city. Some cities contribute many more samples than other cities. 21

2.1 The structure of a generic blockchain [1] 39
2.2 An Overview of normal PBFT case operation [2]. 44
2.3 An Overview of normal Paxos case operation [3]. 45

3.1 General overview of the IoT-based System: we have multiple sensors injecting information to the network switches, which forward packets across the network to the application server. 54
3.2 An overview of the Internet of Multimedia Things system. 56
3.3 End-to-end delay for one hop. 64
3.4 End-to-end delay for different number of hops (K). 65
3.5 End-to-end delay distribution. 66
3.6 Packet dropping probability v.s number of hops. 67

4.1 An overview of IoT-MQTT system. 70
4.2 Packet arrival rate distribution. 75
4.3 The MQTT broker utilization. 76
4.4 System response time. 77
4.5 Total number of packets. 77
4.6 Queuing delay v.s sampling interval. 78
4.7 Queuing delay v.s number of IoT publishers. 79

5.1 Current sensors-to-server flow abstraction. 82
5.2 SADIQ architectural overview. 83
5.3 Military Grid Reference System (MGRS). 84
5.4 SADIQ policy language syntax. 86

5.5 Example of flow entries translated from high-level QoS polices. 91
5.6 IoT packet header structure. 95
5.7 IPv4 source address vs IoTFlow address 98
5.8 Improvement in the percentage of regions with reported temperature error compared with the baseline. 99
5.9 Total percentage of incorrect parking statuses and improvement in the percentage of incorrect parking statuses across regions when the parking occupancy >= 60%. 101
5.10 Effect of location aggregation on the percentage of incorrect parking statuses across regions when occupancy >= 60% over different switch flow entry capacities. 102

6.1 General architecture of Byzantine-based IoT blockchain. 105
6.2 An overview of Byzantine-based blockchain system. 107
6.3 End-to-end delay CDF for average size factories. 115
6.4 Percent deviation from the average delay for different hop-counts (K) and four replica machines (N=4). 116
6.5 Percent deviation from the average delay for different number of replica machines (N) and one-hop (K=1). 118
6.6 End-to-end delay CDF for average size cities. 119

7.1 Data items over the lifetime of the system create a single Synopsis. The collective of tiny blocks at time interval τ creates the main block. 125
7.2 Chain replication in normal case operation. 129
7.3 Worst case scenario in which the replicas are outside wireless domain of other replicas. 133
7.4 Optimal scenario in which the replicas are organized within one communication zone. 134
7.5 Sub-optimal scenario in which the replicas are organized into two communication zones. 135
7.6 Block generation time intervals. 146
7.7 Storage complexity for different consensus protocols. 147
7.8 Message complexity vs. local Tinychain commitment delay. 149
7.9 Committee size scalability. 151
7.10 Wireless chain replication optimization. 152
7.11 Synopsis accuracy and probabilistic errors (Data Redundancy). . . . 153
7.12 Synopsis accuracy and probabilistic errors (Sketch Width). 154

7.13 Synopsis accuracy and probabilistic errors (Sketch Depth). 155
7.14 Local commitment delay against varied batch size. 156
7.15 Local throughput against varied batch size. 157
7.16 Global commitment delay against varied batch size. 158

LIST OF TABLES

1.1 Latency requirements of Intelligent Transport System (ITS) use cases [4]. 16
1.2 Latency requirements for different smart factory use cases [5]. 17
1.3 Traffic characteristics of massive IoT in smart city scenario[6] 18

5.1 Packet loss effect on elephants vs. mice regions. 99

7.1 Synopsis Consensus versus BFT Proposals. 142

Chapter 1

Introduction

IoT connects different types of online devices ranging from simple sensors to smart devices such as coffee machines, smart-phones, webcams, health monitoring devices. The emerging IoT can transform the Internet from traditional human-centric communication (e.g., web browsing) to billions of smart objects communicating in a Machine-to-Machine fashion. Statistics from International Data Corporation (IDC), Inc. [7] indicate that more than 50 Billion IoT devices will generate more than 73 Zettabytes of data by 2025. However, modern IoT systems come with stringent requirements. As the current infrastructure barely meets requirements of IoT and most of the strict needs cannot be fulfilled in today's configurations, new technologies such as SDN and blockchain provide promising solutions for the infrastructure challenges presented by rapidly emerging IoT applications.

1.1 Problem Statement and Motivation

The huge amount of data generated by IoT devices need to be accurately controlled to satisfy specified QoS constraints. Traditional traffic management techniques with the legacy models are unsuitable for ensuring high-performance delivery of IoT applications and services where dummy IoT devices do not react appropriately to high network traffic as human-centric devices. In many cases, prioritizing or limiting redundant data plays a vital role in mitigating high network load. Therefore, the main challenge is to distinguish the data packets and/or the devices from each other. Fun-

damentally, the network should dynamically regulate the bandwidth and QoS for the large-scale, cross-domain IoT real-time applications by designing intelligent, transparent, and light-weight mechanisms for high IoT traffic. This can be accomplished through suitable network policies and implement specific functions (e.g., traffic detectors and prioritizers) through light-weight functions that are implemented at the network edges using SDN. Through delegation, IoT devices can leverage those mechanisms to support different QoS functions (e.g., traffic prioritization) while providing transparency to other end points. The predicted deluge of IoT traffic presents a significant concern for the stability of network services; volumetric traffic must be intelligently alleviated. IoT devices generate a packet stream that often make them indistinct. Selecting a reaction mechanism, therefore, becomes a trade-off between effectively preventing the flood of traffic as well as the damage to useful information.

In addition to traffic regulation, data security presents a specific challenge. IoT applications generate and exchange vast amounts of critical and sensitive data that are subject to attack and data manipulation. IoT systems are attractive environments for malicious activities. Attackers are able to take destructive actions leading to networks shutting down and suffering economic losses. These attacks threaten physical infrastructure and jeopardize national security. In 2017, the malicious software called Triton, was able to disable industrial safety systems, causing catastrophic damages [8]. Triton targeted the industrial infrastructures in the Middle East, and now it has expanded to North America and other countries. Such attacks can take place due to the lack of suitable protection mechanisms. With respect to security issues, blockchain arises as a promising foremost solution. Blockchain is a distributed, immutable ledger technology made up of valid and encrypted data to prevent malicious nodes from modifying the data. The information is transmitted/broadcasted throughout the network to be received by specialized machines that verify, accumulate, and add the data into the blockchain. Nevertheless, a bottleneck in blockchain

Use case	min Latency	max Latency
Road safety (urban)	10 ms	100 ms
Road safety (highway)	10 ms	100 ms
Urban intersection	-	<100 ms
Traffic efficiency	-	<100 ms

Table 1.1: Latency requirements of Intelligent Transport System (ITS) use cases [4].

performance remains one of the most critical challenges, where the communication overhead, low throughput, and long delay strictly limit the adoption of blockchain in IoT.

To motivate our problem, we provide an overview of different IoT application needs and requirements. For the sake of this work objectives, we categorized IoT applications into two main categories: critical (delay-sensitive) and massive applications. Critical IoT applications have high levels of delay and reliability requirements. Common examples of such applications include traffic safety, automatic machinery and vehicles, artificial intelligence-based machines, and remote surgery in the healthcare sector. Massive applications, on the other hand, require high accuracy, reliability, numerous connections, and small data volumes. Weather monitoring, transport logistics, smart buildings, and smart metering are examples of massive applications. Detailed uses cases are given as follows:

- Internet of Multimedia Things (IoMT): IoT has been deployed in real-time multimedia applications at health, security, transportation, and disaster detection and management systems. The smart heterogeneous multimedia devices interact and cooperate with one another and with other devices through the Internet create a novel paradigm called the Internet of Multimedia Things (IoMT) [9]. For the real-time IoMT application, the end-to-end delay requirement of an Intelligent Transport System (ITS) is given as shown in table 1.1. Transportation Services such as road safety, intersection and traffic efficiency at both Urban or

Use Case	**Priority**	**Latency (ms)**	**Average Payload Size**	**Device Density (** $100\ m^2$**)**
Motion Control	Very high	0.5 - 2.0	63 bytes	185
Mobile Control	High	4.0 - 12.0	135 bytes	22
Process Monitoring	Medium	<50.0	60 bytes	1000
Video Remote Control	Low	10.0 -100.0	80 Kbytes	10

Table 1.2: Latency requirements for different smart factory use cases [5].

highways use smart cameras that are distributed all over the targeted area to send traffic warnings about collisions or dangerous situations. Therefore, the communication system must operate with communication latency of less than 100 milliseconds while ensuring high reliability [4].

- Smart Industry (industry 4.0): Industry 4.0 [10] is the 4th stage of the industrial revolution. It is the next era of industrial production, which is aimed to improve the flexibility and usability of future smart industries. Generally, industry 4.0 is about the integration of IoT and its services extending to industrial manufacturing technologies. The industrial domain has diverse requirements as it comprises a large number of heterogeneous use cases and applications. Among the critical characteristics of different use cases needing to be considered are QoS, cost-effectiveness, security, safety, reliability, and availability. Domain-specific personnel are responsible for considering these aspects with respect to their importance. Table 1.2 summarizes the latency requirements for different use cases in the industrial sector. Motion control [5] is one of the most challenging and demanding use cases among all listed. The motion control system essentially deals with moving and rotating parts of a machine in a well-defined manner. A use case with such functions is expected to have strict requirements

Use Case	Message Interval	Average Payload Size	Device Density (Km^2)
Vending Machine	24 hours	150 bytes	150
Bike Management	30 minutes	150 bytes	200
Pay-as-You Drive	10 minutes	150 bytes	2250
Electricity Meters	24 hours	100 bytes	10000
Water Meters	12 hours	100 bytes	10000
Gas Meters	30 minutes	100 bytes	10000

Table 1.3: Traffic characteristics of massive IoT in smart city scenario[6]

of determinism, ultra-low latency, and reliability.

- Smart Cities: A smart city, like a smart industry, works to improve the quality and effectiveness of government services for the sake of public welfare. Smart cities also use communication technologies to enhance operational efficiency and share information with the common public. A network of connected devices is distributed in a particular area for better results in such utilities as electricity, water, gas meters, vending machines, parking monitoring devices, and accelerometers in cars to keep an eye on driver behavior. A dense area with 10,000 devices per each kilometer is deployed as a base for smart city services scenario, as in the central areas of New York, London, and Beijing [6]. A most significant feature of these devices is the creation of non-deterministic behavior. This behavior is created through the wireless communication interface and the sharing of the same communication channels that introduce the contention on radio module and interference between different data traffic sources. With this contention, even minimum performance requirements cannot be guaranteed. Table 1.3 shows the traffic characteristics of IoT technology in a smart city scenario.

More precisely, a special group of the IoT applications can be defined as location-based applications in which the data's importance is highly dependent on the loca-

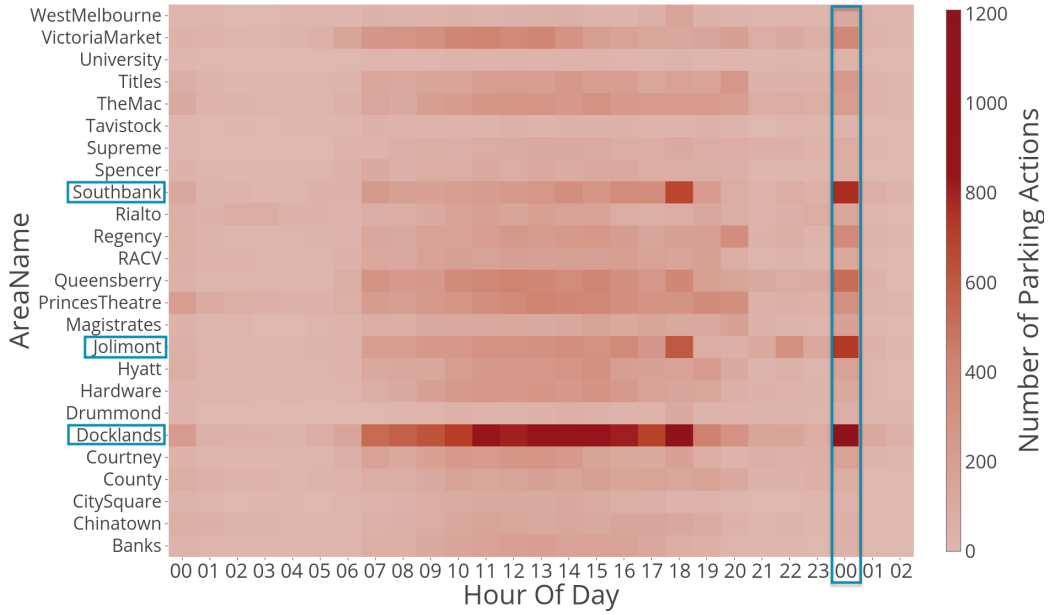


Figure 1.1: Smart parking actions on New Year eve for Melbourne. Some areas see significantly more activity compared to other regions.

tion, context, and application needs. We provide examples to illustrate how these location-based applications can benefit from richer QoS support at the network level. Our examples build on a trace-driven analysis of two popular categories of applications, event-based applications, and participatory sensing applications. The underlying theme in these examples is that for such applications, not all data is equal.

- Event-Based Applications: Some IoT applications are impacted by "events". For example, data from video surveillance cameras in a certain area may become critical if an incident occurred in that area. Similarly, data of a smart parking application that finds vacant parking spaces becomes vital during a sporting event or festival, for instance. The same applies to disaster prediction applications, where some sensor readings enabled prediction of an upcoming hurricane or flood in a specific region [11]. This type of IoT applications would benefit from a network that gives preferential treatment to packets originating in areas of events over packets coming from areas and situations considered *"normal"* at that time.

  - Smart Parking Application: Melbourne Smart Parking [12] is an example of an event-driven application. It is based on a system of nearly 7000 in-ground sensors monitoring on-street parking in the city of Melbourne, Australia. Each in-ground sensor detects a parking action, i.e., a car parks or leaves a parking space and uploads this action to an online database. The city of Melbourne published the parking data for the year 2014 covering 27 different locales in Melbourne. Fig.1.1 displays a heat map of the number of parking actions across the 27 areas on the New Year eve. We can observe that at midnight, the number of parking actions increases in certain areas (highlighted in blue boxes) such as Docklands (1183), Southbank (798), and Jolimont (729). We also observe high activity during other important events like fireworks displays. Online updates during such events require special handling at the network level to increase accuracy for drivers seeking parking spaces at a busy time.

- Participatory Sensing Applications: Many IoT applications are based on participatory sensing, with people contributing environmental sensory information using their cellphones or special kits. Weather Signal [13] and Air Quality Egg [14] are some well-known examples. Such sensory data are spatially and temporally correlated. Sensors from the same geographical area over the same period of time try to report similar information. The more data samples we have, the higher the accuracy of the results. However, the distribution of the participated readings often varies significantly over both space and time because the system lacks control over the participants' locations. The quality of the information reported by such applications would be highly impacted if the network drops packets from regions with a low number of samples, compared to dropping packets from highly represented regions.

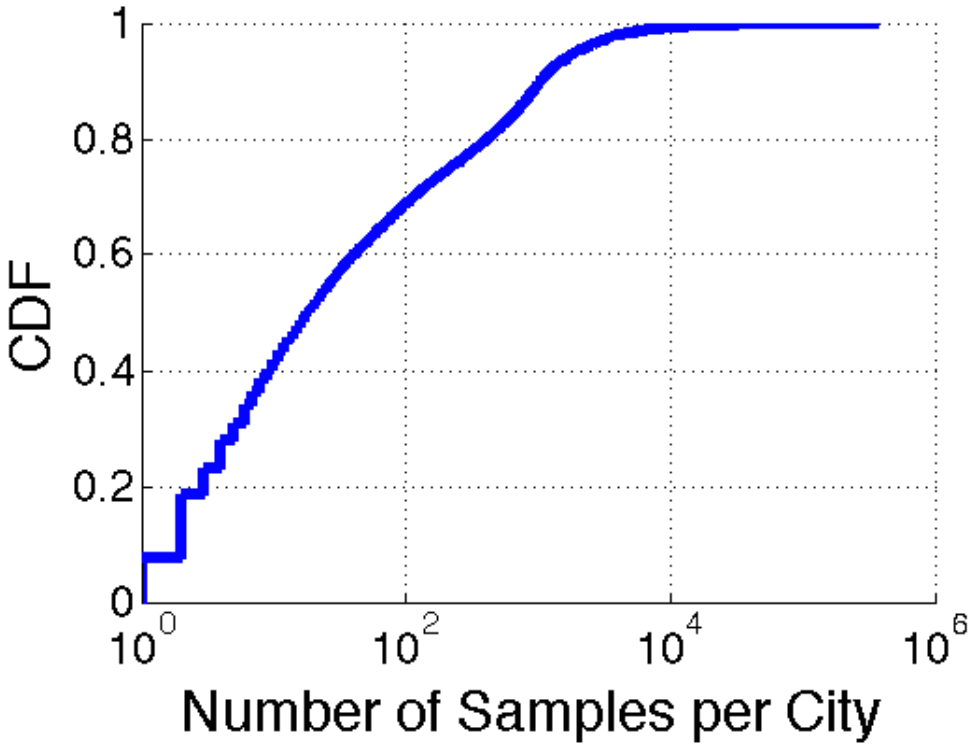


Figure 1.2: CDF in log scale of the number of samples per city. Some cities contribute many more samples than other cities.

– Weather Signal Application: We select Weather Signal as an example of a participatory sensing application [13]. Weather Signal is a system that collects temperature data from users' cellphones to create a comprehensive live weather map. We collect six months of Weather Signal sensor data (October 2014-March 2015), and filtered out the US samples (8,550,768 samples). Subsequently, we use Google Reverse Geocoding API to translate the GPS coordinates of the US samples into city names, which provided us 15,680 different US cities. Fig.1.2 displays a CDF of the number of samples per city in these six months (note the log scale). The figure shows the non-uniform nature of the number of samples contributed from each city. Specifically, 98% of the cities cumulatively contribute to only 55% of the data, while 2% of the cities contribute to the remaining 45%. We refer to these two types of cities as *mice* and *elephants* cities. This

shows how highly skewed and long-tailed the sensor sample distribution can be depending on the number of contributors in each area.

1.2 Thesis Objectives

This work's ultimate objective is to achieve a minimum performance level, QoS, and fault tolerance guarantees in IoT networks. IoT application-level demands and the network configurations are the main driving factors in achieving the end-users' (machine or human) expectations. In this thesis, IoT system is modeled and analyzed. This analysis realizes that QoS mechanisms for IoT applications need to be tightly integrated with the applications. The applications need to express and precisely control their QoS objectives, and the network needs to have suitable mechanisms to support these requirements.

Therefore, we examine the Message Queuing Telemetry Transport (MQTT) communication protocol where IoT devices communicate in a Publish/Subscribe architecture to overcome the limitations of the legacy communication model. The Publish/Subscribe model can help distinguish IoT devices from each other and prevent unnecessary IoT traffic from accessing the production network. We propose a detection method that builds based on a mathematical model. The detection method aims to evaluate the network level infrastructure and compute the end-to-end delay and packet dropping probability that meets predefined performance metrics. We then propose, design, and implement SADIQ (SDN-based Application-aware Dynamic Internet of things QoS) equipped with functions essential for alleviating traffic at the edges. In this platform, we introduce a new location-based abstraction for IoT packets. The IoT applications are given the ability to specify high-level QoS policies based on their specific requirements and real-time context. And, a QoS controller implements these policies in SDN switches. SADIQ's ultimate objective is to provide an intelligent layout for the IoT applications to meet the required performance metrics.

These metrics, with the help from SADIQ, can be evaluated and updated in real-time.

Furthermore, IoT applications need to meet the necessitates of reliability, availability, and security constraints. Distributed peer-to-peer systems infrastructure, however, can be considered a solution to the wide-scale adoption of such applications. Blockchain is one way to spread the data across the network in the form of immutable ledgers. For this reason, we conduct a performance study to evaluate the potentiality of blockchain technology to meet the strict delay and availability requirements. Byzantine-based fault tolerance is analyzed as the primary consensus candidate for the IoT blockchain environment. This study concludes that the current blockchain implementations need to be redesigned to overcome the scalability and performance limitations. Thus, an innovative blockchain system, Synopsis, is built base on a hierarchical structure. Data block format and the consensus method are both redesigned. The IoT data records are encapsulated in a compact probabilistic data structure to be fully distributed on resource-constrained IoT devices. And the consensus method is characterized by two main features, locality and hierarchy, and are optimized for the wireless nature of IoT devices. The main objectives of Synopsis are to reimplement blockchain in a lightweight framework entirely distributed on the resource-constrained IoT devices, to meet the minimum performance requirements, and to mitigate a range of system failures.

1.3 Thesis Contributions

The main contributions of this thesis are summarized as follows:

- Provide background and literature review of IoT, SDN, and blockchain. We illustrate IoT main challenges. IoT is a complex system that spans from simple sensors to more complicated machines connected through a wide variety of technologies. From the interconnection perspective, networking solutions use either conventional communications networking or modern SDN mechanisms; both

mechanisms are discussed. We conclude that the distributed nature of IoT systems requires distributed solutions, such as those available through blockchain technology.

- Conduct performance evaluation and analysis of the network structure required by a variety of IoT applications. IoT multimedia services are given as a case study for the performance analysis. We provide a mathematical model to compute the maximum allowable network hops without exceeding some predefined QoS constraints. We also compute the packet dropping probability and the network delay for a given network hop-count to provide a guaranteed performance level.

- Study IoT high traffic load, and propose an instantaneous detection method for IoT (IDIoT). We examine the IoT communication protocol where IoT devices communicate in a Publish/Subscribe architecture. We employ MQTT communication protocol as our Publish/Subscribe model. Furthermore, we use the queuing model to characterize the MQTT central point and to propose a detection technique to prevent flooding the network near the edges.

- Explore and utilize SDN to propose SADIQ, SDN-based Application-aware Dynamic Internet of things QoS. We use traces from two real IoT applications to make a case for location-aware, context-driven QoS for IoT applications. We propose a new location-based abstraction for IoT traffic, and we define a high-level QoS policy language for IoT applications. We design an SDN-based QoS controller that translates application policies into OpenFlow rules. SADIQ is implemented using an existing SDN controller and commodity SDN switches. The performance is evaluated using real application traces.

- Evaluate Byzantine-based blockchain systems in IoT context. We obtain a mathematical expression to calculate the end-to-end delay with different net-

work configurations, number of network hops, and consensus machines. The network hop-counts and the number of consensus machines are considered the key configuration parameters that directly affect the IoT applications' performance. We use two IoT use cases with different application requirements to validate our proposed mathematical model. Results are generated from various network setups and applications.

- Propose a novel blockchain system, Synopsis. We design and implement Synopsis as a hierarchical blockchain framework to address unique IoT characteristics. We propose a novel Byzantine chain replication framework for implementing a Byzantine fault tolerance algorithm over a wireless channel. We also propose a novel data structure that constructs the first hierarchical level of data chain based on the summarization sketches. The sketch is a probabilistic data structure providing an efficient compressed data structure for IoT constrained devices. At the second hierarchical level, conventional tiny blocks are used to overcome the sketch's probabilistic limitations. We evaluate our system over large-scale simulations and real-testbed experiments.

1.4 Thesis Organization

The main objective of this thesis is to provide and develop QoS-enabled and fault tolerant framework for IoT systems. We provide an overview of IoT and related networking models in Chapter 2 along with state-of-the-art solutions. In Chapter 3, the traditional internetworking model is analyzed using as a case study the Internet of Multimedia Things (IoMT). Then, IDIoT, an instantaneous detection method for high IoT traffic load is modeled and evaluated in Chapter 4. Chapter 5 presents SDN-based Application-aware Dynamic Internet of things QoS (SADIQ), SADIQ is designed, implemented, and evaluated in this chapter. Chapter 6 presents a model of blockchain with Byzantine-based consensus algorithm. The model is extensively examined with

different IoT application requirements. However, the results in Chapter 6 motivate us to propose a new blockchain framework, Synopsis, for IoT networks. Synopsis is designed, built, and evaluated in Chapter 7. Chapter 8 concludes this thesis.

Chapter 2

Background and State of the Art

2.1 Internet of Things

The IoT ecosphere is dependent on sensors located in the remotest corners of the earth that translate the analog signals of physical effects into digital signals, which comprise the language of the Internet. After this translation, data covers a complicated journey, passing through several wireless signals, protocols, natural atmosphere, and electromagnetic mechanisms. It eventually reaches the ether of the Internet, where it is packetized. This packetized data then arrives at remote servers, e.g., a data center. Important to mention here is that IoT is not dependent on a few sensors; instead, it is dependent on the aggregate of millions of sensors, and corresponding devices [15]. In light of previous research studies, the major characteristics of IoT are listed below [16].

- Interconnectivity: Any device in IoT can be interconnected with the global communication infrastructure.

- Heterogeneity: The devices are heterogeneous based on different hardware and software platforms. This feature enables these devices to interact and communicate with each other using different network infrastructures.

- Dynamic changes: The state of devices in IoT network changes dynamically (e.g., connected, disconnected, sleeping, and waking). Additionally, the number of devices and their context also changes dynamically with time and location

variations.

- Enormous Scale: The number of devices connected with IoT is at least an order of magnitude larger than the number of devices connected and managed by the current Internet.

2.2 Internet of Things Challenges

IoT network faces several challenging obstacles [17, 18]. To ensure successful adoption and diffusion of IoT, overcoming these difficulties is essential. Some key challenges are:

- Stringent latency requirements: Numerous modern control frameworks, manufacturing systems, smart grid, oil and gas frameworks, require an end-to-end delay between the sensor and the control nodes within a few milliseconds [5]. Numerous other IoT applications, vehicle-to-vehicle, and vehicle-to-roadside communications, drone flights with control applications, augmented reality applications, gaming applications, and ongoing money-related exchanging applications, may require latencies in less than several milliseconds. These necessities fall far outside the achievements of current IoT systems.

- Network bandwidth constraints: The immense and rapidly growing number of connected devices creates data at an exponential rate. Sending all of it to remote servers requires restrictively high network bandwidth. Thus, the endpoint information should be prepared and stored locally because of guidelines and information security concerns.

- Data challenges: IoT devices generate a huge amount of data, and this massive amount of data is defined by the term, "big data". Big data has been characterized by three dimensions: volume, variety, and velocity. Volume describes

the amount of data; variety describes the different types of data; and velocity describes the speed of data generation [15]. From the standpoint of data accuracy, wireless communication may lead to erroneous or altered data, potentially compromising the validity of the data coming over the network. Unless it can be secured through a generation phase, the system cannot ensure the data's accuracy. This is a challenge as security embedded in resource-constrained devices is often difficult to implement and maintain.

- Reliability and availability constraints: Uninterrupted, continuous, and safe operations are the top need in IoT cyber-physical frameworks such as industrial control systems, smart cities, and connected vehicles and transportation. Under any circumstances, taking a system disconnected can cause significant misfortune to business or extreme inconvenience to clients.

- Privacy and security challenges: Enterprise networks, data centers, and end-user devices are protected using traditional cyber security solutions. Precisely, such frameworks are placed behind firewalls, and system operators use intrusion detection and prevention mechanisms to prevent security threats. Currently, security services continue to provide traditional protection functions, such as redirecting email and web traffic to centralized servers for threat analysis and detection and redirecting authentication and authorization processing. If a threat occurs, the human operators take the system offline, clean up, or replace compromised files and devices, and then put the system back online. This existing security worldview will never again be satisfactory for the numerous new security challenges in the emerging IoT.

2.3 Internet of Things Communication Protocols

In an IoT network, communication protocols facilitate the transportation of messages to achieve certain tasks. Two communication patterns are very common in IoT contexts. The first of these is the request/response model; the second is publish/subscribe model [19].

Request/Response Model

This basic IoT communication pattern allows the client to request the information in real-time. In this pattern, the words "client" and "server" designate their roles, not a hierarchy in a network. The client typically utilizes a request-response design to communicate through the traditional web. The client requests while the server responds accordingly.

Publish/Subscribe Model

The publish/subscribe pattern makes possible an efficient mass distribution of information to the involved devices. It allows the publisher of information to send its data only once to a publish/subscribe server, which then retransmits it to subscribers. Therefore, it reduces the traffic by up to half. Examples of publish/subscribe protocols are as follow:

- Message Queue Telemetry Transport (MQTT): MQTT [20] is a common data messaging protocol related to IoT. Minimal power consumption and low bandwidth are the two primary features of MQTT [15]. In this protocol, the new data comes into an intermediary node, named broker, as a message and is then delivered to end-users. In this process, the end sever is known as a "subscriber"; while IoT device is recognized as a "publisher". Each message that is published to the broker is associated with a specific topic, and every MQTT client must

have a unique client ID. The broker uses the topic as a routing information where each client who wants to receive messages subscribes to a certain topic. The broker is responsible for distributing all messages that belong to the matching topic. MQTT is a topic-centric communication protocol in which clients communicate over the topics without dependencies between the data publishers and the subscribers. However, the message on broker always exists. It is not dependent on the status of publisher and subscriber, giving the MQTT robust protection against intentional and unintentional connection losses. The MQTT is considered a light protocol in which a device can act freely. It can periodically switch, connect to the broker, send its data, and then go to sleep regardless of the subscribers' status. However, a persistent connection is maintained in most implementations, and in practice the communication tends to be near real-time [15].

- MQTT for Sensor Networks (MQTT-SN) Protocol: MQTT-SN [21] is the derivative of MQTT and is used for sensor networks. The operational functions within this protocol are kept to the same MQTT functions. By design, it is specified for sensor environments. The key objectives of MQTT-SN include support for low bandwidth links, link failures, short message length, and resource constrained hardware.

Other Messaging Protocols

Several other messaging protocols can be used for IoT and machine-to-machine (M2M) deployments [21]. Most commonly used protocols include HTTP and CoAP. HTTP is used by various web applications for the exchange of data. HTTP protocol follows the request/response pattern and it operates at the application layer. There are several cases in which HTTP is not considered ideal for IoT applications. Latency is not predictable in these cases and polling is needed for the detection of state changes.

CoAP also follows the request/response model and is developed for low-power devices and it has low overhead requirements where the data packets are much smaller than HTTP. To support the given protocol, an architect is supposed to consider all necessary features, power, bandwidth, and other overhead resources, for developing a vision for the future and ensuring the scalability of provided solutions.

2.4 Internet of Things Computing Paradigm

On the Internet, three popular computing paradigms are used to manage and deliver services to the end-users and applications. Cloud computing [22] is the first paradigm that provides reliable services through next-generation data centers. Cloud computing is aimed at simplifying web-based systems for the end-users. This platform acts as a receiver of data, which is interpreted and analyzed after receiving it from ubiquitous IoT sensors. Ubiquitous sensors are not apparent to the users, and they always work in the background. Cloud computing provides the necessary computation time, scalable storage, and other required tools to build and construct new businesses. However, providing the computing resources at the edge and in proximity to IoT devices at the time of data generation is a significant feature for IoT solutions. The ability to compute locally is mandatory when IoT solutions are constructed in remote and different locations, as in several cases, the device is distant from the administrator or programmer [22]. Conventional IoT systems send captured data to remote servers for analysis, management, or distribution purposes. However, the long-distance transmission of a large volume of data chunks may cause congestion and delay and, as well, affect the required performance level. Edge computing has been introduced to bring the remote servers benefits closer to where data is created, reducing transmission delay [21, 23]. In this Edge computing fashion, fog computing emerged to distribute resources horizontally and vertically along the cloud to the edge [24]. A capable fog node allows IoT devices to meet the application demands by reducing the networking

bandwidth's latency and consumption. But the fog nodes usually work under multiple third-party entities; hence, it is difficult to ensure absolute trust and security. The edge model provides a faster response for IoT applications in which fog nodes have some form of distribution and is more resourceful rather than end devices. The cloud servers can provide high computation power and large storage. Therefore, the IoT takes benefits from cloud, fog and edge services.

2.5 Internet of Things Network Paradigm

2.5.1 Traditional Internetworking

A typical internetworking architecture is supposed to use a collection of well-managed hardware and software components with a single operational purpose. Common functions include routing, firewalls, load balancers, managed switches, deep packet inspection, intrusion detection, and data analyzers. Skilled and trained IT staff are generally required to operate, maintain, and administrate these dedicated appliances. However, because these appliances are sourced from a wide range of vendors, significantly different methods are required to manage each unit.

In such a configuration, both the data plane and control plane are unified. Many dedicated systems need to be updated through a virtual local area network (LAN) setting in order to set up a new data path and add or eliminate an additional node. Apart from the virtual LAN setting, additional factors need to be considered, such as QoS parameters, static routes, firewall pass-through, and lists of access controls. This setting can be managed with several thousand endpoints. However, these settings become unsustainable in remote moving, connecting, and disconnecting millions of nodes.

2.5.2 Software Defined Networking (SDN)

SDN [25] is a rapidly growing networking paradigm intended to alter the limits of existing network infrastructure. This type of networking first separates the control plane from the data plane to break vertical integration. The control plane is the network's control logic, while the data plane is composed of underlying routers and switches that move the traffic. Secondly, the control logic is executed in a logically centralized controller or in a network operating system, while network switches are shaped to be a simple traffic forwarding device. This separation results in the simplification of policy enforcement, network configuration, and evolution of solutions. It must be considered, however, that this separation can impact network performance, scalability, and reliability. Moreover, the above-mentioned solution should be designed appropriately to maintain an adequate level of performance.

SDN Architecture

SDN architecture is composed of different layers, each performing a particular function [25]. We follow the bottom-up approach and explain southbound API, northbound API, network controllers, and network applications layers.

1. Infrastructure: There are two major elements in SDN architecture that use OpenFlow [26] communications protocol. The first is the controller; the second is the forwarding device. By definition, a data plane is a hardware or software device, whose function is to forward the packets. The controller, on the other hand, is a software stack operating on a hardware platform. An OpenFlow-enabled forwarding device is composed of a pipeline containing a series of flow tables. An entry in the flow table is further composed of three key parts. The first part is known as a matching rule. The second part is comprised of the actions executed on the matching packets. The third part is a counter that counts the number of matching packets and keeps their statistics. Currently,

this model of OpenFlow is widely used because of its simplification and high level of reliability.

Inside the OpenFlow device, there is a sequence of flow tables that define the handling process of the packets. A lookup process starts in the first table on the arrival of the first packet. This process ends with finding a match for that packet or a miss if the match is not available. A flow rule can be developed, however, by matching the different fields and combining them. A packet is discarded in the absence of a default rule. There exists the possibility of installing a new default rule directing the switch to send the packet to the controller. The rules priorities follow a natural sequence of numbers and row order in the flow table.

It is required that each match filed be supported by at least one flow table. Moreover, that flow table should meet all match filed prerequisites to enable the matching in the pipeline. Moreover, OpenFlow Extensible Match (OXM) [26] is introduced in OpenFlow version 1.2, increasing the matching capability of the network as the experimenter can add his matching filed. OXM is based on type-length-value (TLV) structures. The first four bytes of OXM TLV's body are based on an experimenter identifier. Experimenter-specific flow match fields may be defined using the $oxm\_class = OFPXMC\_EXPERIMENTER$.

OpenFlow devices are available as both open source and commercial products. Among other appliances, several OpenFlow switches are off-the-shelf and ready to deploy. A growing number of software switches offer a promising and effective solution for data centers and virtualized networks. OpenVswitch [27] is a prominent example of an OpenFlow switch.

2. Southbound interface: Southbound APIs allow the SDN controller to make the dynamic changes according to the needs and demands of real-time scenarios to effectively control the entire network. OpenFlow [26]], developed by Open

Networking Foundation (ONF), regarded the foremost southbound interface, has become the industry standard that directs the SDN controller to interact with the forwarding plane and make adjustments to the network according to business requirement demands.

3. Network controllers: Also known as Network Operating System (NOS), a network controller is considered a key part of SDN design as it is an essential supporter of control logic producing the network configuration. This network configuration is developed according to the policies set by network operators. The control platforms materialize the network policies by abstracting lower-level details of connecting and interacting with forwarding devices. A diverse set of controllers is available with different designs and architectural choices and can be used for different purposes.

4. Northbound interface: The northbound application program interfaces (APIs) are used to communicate between the SDN controller, administrations, and applications running over the system. The northbound APIs can be utilized to encourage innovation and empower the proficient organization and computerization of the network to line up with the requirements of various applications. Existing controllers, such as Floodlight controller [28], propose and characterize their very own northbound APIs, and every one of them has its own definitions.

5. Network applications: Network application can be referred to as network brains because they are used to implement the control logic translating the commands installed in the data plane. It is also used in dictating the behavior of forwarding devices. Existing network applications perform customary functions, for example, routing, load balancing, and security policy enforcement. It also performs the function of investigating novel approaches, such as reducing power consumption, fail-over and reliability functionalities to the data plane, end-to-end

QoS requirements, network virtualization, and management in mobile wireless networks.

SDN and Mass Internet of Things Deployments

SDN networking can be considered for mass IoT deployments. It is important in guaranteeing the level of service when dealing with congestion or variable network loads. For instance, in an IoT use case, when live video streams and public Wi-Fi are mixed, the video feeds may require a higher priority and a guaranteed level of quality, particularly in a circumstance of safety, security or surveillance.

- QoS in the Internet: A large body of prior work on providing QoS on the Internet. Proposals like IntServ [29] support QoS through a combination of admission control, policing, and intelligent scheduling techniques but suffer from scalability concerns. Other more scalable proposals include DiffServ [30], and MPLS-TE [31] which mark packets at the edges depending on the desired QoS treatment. The routers use this marking to decide how packets should be scheduled or dropped [32, 33, 34, 35].

- QoS in modern settings: Recent proposals have explored the potential of SDN in automating QoS control [36, 37, 38]. They use SDN-enabled mechanisms, such as Intelligent congestion control and flow aggregation, to provide applications with adaptive QoS. However, their QoS policies are only adaptive to changes in the network workloads and are unaware of changes in the application context. Other proposals provide more application-awareness in utilizing SDN to provide QoS in domains other than IoT [39, 40], such as video transmission [41, 42, 43, 44] and interactive online gaming [45, 46]. These studies incorporate application-specific characteristics such as video encoding methods in their QoS policies to achieve better user experience. Finally, some studies

[47, 48, 49] consider QoS requirements of different IoT applications while relying on existing QoS mechanisms.

- Location awareness: Recent work has called for location awareness for IoT and sensor applications. The work in [50] suggests matching on sensor attributes, including the address to better utilize the network resources. The model in [51] proposes routing/forwarding packets based on the geographical location.

- Richer abstractions: Many researchers have noted limitations in the current flow-based abstractions for data center applications and have proposed richer abstractions that capture the needs of specific applications. This include proposals like Baraat[52] and CoFlow [53] which target distributed applications such as MapReduce [54].

2.5.3 Blockchain Technology

Blockchain [1] is defined as a distributed peer-to-peer ledger that appends cryptographically secure and immutable data. Blockchain can be shown as a layer of a distributed peer-to-peer network, such as Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP), running on top of Internet Protocol (TCP/IP). A block is essentially a selection of data records packaged together to be sorted logically. Depending on the implementation, the data records and the block size can be defined. Each block has only one parent block. The hash of the previous block is always contained in the block header. Genesis block is the first block of blockchain, and it has no parent block. Items fundamental to the functionality of a block include the block header, pointers to previous blocks, the time stamp, a special number called nonce, records counter, and other attributes. The structure of a nonexclusive blockchain can be envisioned with the assistance of Fig.2.1.

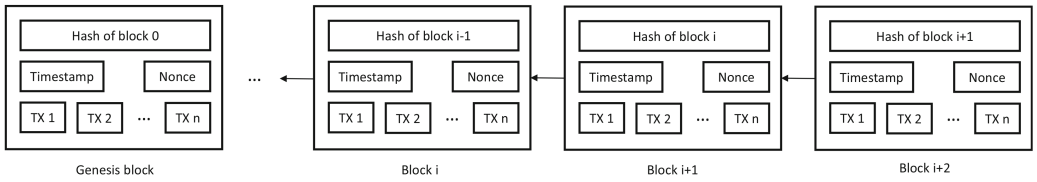


Figure 2.1: The structure of a generic blockchain [1] .

Blockchain Models

Current blockchain systems can be categorized into two models: permissionless and permissioned blockchain [1].

1. Permissionless (public) blockchain: As the name indicates, permissionless blockchain networks are open to the public and anyone can join them. The devices in these networks perform the computing functions required to verify data records. A copy of ledger is hosted on participating devices where a distributed consensus mechanism finalizes these ledgers. The majority of blockchain data are recorded under pseudonyms. It is possible to track these data records, however, as well as the identity of the person carrying these data records. This is due to the fact that a permissionless a blockchain network cannot guarantee data privacy or anonymity. Moreover, a permissionless blockchain is always at the risk of "51% attack" [1]. Such attacks happen when malicious users control half of the compute power connected with the blockchain network. Even with a short period of time, this attack can result in complete destruction of data recorded on the blockchain network. Although 51% attacks happen rarely, they are considered a real threat to the integrity of data recoded on these networks. Bitcoin [55], and Ethereum [56] are two common examples of permissionless blockchain network.

2. Permissioned (private) blockchain: A primary element that differentiates the permissioned blockchain from permissionless blockchain is an access control layer built into permissioned blockchain nodes. In permissioned blockchain,

no one can join the network without permission allowing participants complete control over the network. Only approved parties are given full autonomy to participate in the consensus mechanism and validate the blocks. This step serves to mitigate the security challenges associated with permissionless blockchain. The reason for this is, in terms of data privacy, that a permissioned blockchain network is more effective than permissionless. Moreover, a permissioned blockchain using Byzantine Fault Tolerant protocols can tolerate sybil attacks because of the membership aspect as long as the number of faulty nodes does not exceed one-third of the total consensus machines. Sybil attack is a type of attack seen when a node attempts to gain inappropriate control over network peers by creating fake identities [57]. In addition, maximum data privacy is assured as outsiders are not allowed to access the data records. Due to the above advantages, permissioned networks are used by several large-scale companies such as IBM [58].

Blockchain Challenges and limitations

1. Space complexity: The typical blockchain has its problems in regard to data storage. These problems require new protocols to be integrated on top of existing blockchains. Bitcoin [59], for instance, reaches approximately 300 Gigabytes in size a level of growth that is incomparable with the large volumes of data generated from the different applications. Storage mechanisms are one of the most significant challenges to be addressed in the domains of both research and industry.

2. Data finality time: The end-to-end delay of the current blockchain consensus algorithms, occurs when the data is generated until the final enclosure in the blockchain, needs further improvements. Finding a mechanism that maintains a balance between the delay performance measurements and the decentralization

advantages of the blockchain is, therefore, an essential step to be taken.

3. Privacy threat: Blockchain is expected to be extremely safe as clients make data records using the produced addresses, not their real identity. Conversely, permissionless blockchain cannot ensure data privacy since the data are publicly available and shared between the participated blockchain's users.

State of the Art Consensus Protocols

In a blockchain network, the concept of consensus is used to introduce a single version of the truth to all peers. However, the achievement of agreement can be challenging for blockchain technology. They assume the network is unreliable and are never sure about the arrival of communicated data. The consensus protocol attempts to bring the agreement across a distributed network of devices. However, attackers can use faulty nodes to undermine the network. In the existence of such nodes, assuring safe and reliable communication is an arduous task. We briefly introduce the commonly used consensus mechanisms in the blockchain environment.

- PoW (Proof of Work) [55] is the consensus strategy utilized by the Bitcoin network. Although random selection is the best approach to record the data in a decentralized network, it is also vulnerable to attacks. For this reason, much work is needed to assure that the data block published by a node is not vulnerable to attack. In PoW, this means great effort into computer calculations with each node calculating the hash value of the header of the block. The block header contains a special value, a nonce, that helps the block verifiers (miners) to accrue different hash values by frequently changing the nonce. The consensus requires that the calculated hash value be equal or smaller than the given value. After achieving the target value, a node broadcasts this value to other nodes for a mutual confirmation. If the block is confirmed and validated, it

is appended by other miners to their blockchain networks. In this entire process, the nodes that calculate the hash values are called miners, and this procedure is known as mining, a complex process, requiring a computer equipped with highly specialized hardware. This process consumes large amounts of power, the significant cost of which is considered a major drawback.

- PoS (Proof of Stake) [60], regarded a viable alternative to PoW, is a less energy-consuming process. In the PoS network, a miner is supposed to prove ownership of an amount of currency. While true that PoS is a robust and cost-saving system, it has a downside, known as the "nothing at stake problem". This problem occurs when block-generators have nothing to lose by supporting the historical records of various blockchains while preventing the conflict from resolving. To overcome this problem, the companies have developed modified protocols such as Delegated Proof of Stake.

- DPoS (Delegated Proof of Stake) [61] the main factor that differentiated the DPoS from PoS is an election of delegations by stakeholders. In DPoS, the stakeholders are allowed to elect their delegates to generate and validate blocks. The confirmation process is very quick as only a few nodes are needed for verification. Due to this, the speed is faster in DPoS than in PoW or PoS, a key advantage of DPoS. Similar to other protocols, DPoS also comes with limitations, foremost being incomplete decentralization. Because DPoS is not truly decentralized, power and currency remain in the hands of a small group of nodes.

- Ripple [62] is a consensus algorithm making use of collectively-trusted subnetworks within a larger network. In a Ripple network, the nodes are divided into two categories: server and client. Server nodes participate in the consensus process while client nodes transfer the funds. Each server has a unique node

list (UNL). A server searches the nodes in its UNL. If the received agreement is more than 80%, the data record is placed into the ledger. The ledger remains correct until and unless the percentage of faulty nodes is 20%. Ripple can handle 1500 records per second, confirming data in few seconds. This function gives Ripple a marked edge over other cryptocurrencies although Ripple carries the same associated risks. The strength of its model is not a guarantee of its reliability or performance.

- Traditional Byzantine Fault Tolerance (BFT) consensus: are a family of protocols that tolerate malicious failures, known as Byzantine faults. The term Byzantine arbitrarily encompasses any action that is unexpected or malicious. Several BFT protocols have been designed, such as PBFT (Practical Byzantine Fault Tolerance) [2], BFT-SMaRt (Byzantine Fault Tolerant-State Machine Replication) [63], and HotStuff [64]. For example, PBFT is a form of state machine replication (SMR) in which a service is replicated across different nodes in a distributed system. The service state and operations are maintained by each replica in the system. The protocol has at least $3f_b + 1$ replica machines, where f_b is the maximum number of replicas that may be faulty. The PBFT protocol is a well-known leader-based Byzantine consensus protocol utilizing one primary node (leader) and multiple backup nodes (replica nodes). Consequently, intra-node communication is present among all nodes within the system. The intra-node communication aims to reach an agreement on the state of the system through quorum voting. Nodes interact extensively to prove that a message derives from a specific peer node and to verify that the message was not altered during the network transmission. As shown in Fig 2.2, PBFT in normal-cases has four phases; preprepare, prepare, commit, and reply. A client calls a specific service by sending a <Request> to the leader node. A sequence number is attached to the <Request> by the leader node, which multicasts a <PrePrepare>

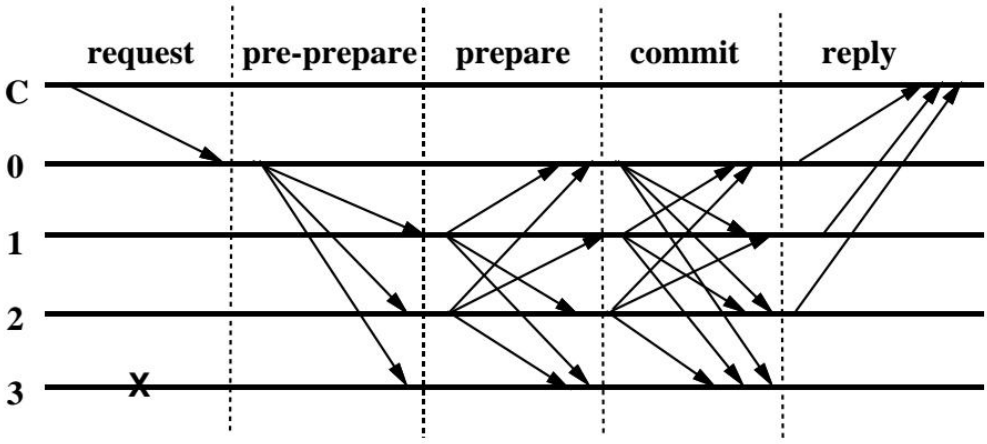


Figure 2.2: An Overview of normal PBFT case operation [2].

message to the replicas. A replica must verify the received message and drop the message if invalid. A replica then enters the prepare phase by multicasting a <Prepare> message to all the nodes inclusive of the leader. A <Prepare> message indicates a replica's willingness to accept a given request. Once $2f_b$ <Prepare> and a <PrePrepare> messages are received, nodes must ensure that enough nodes have all been prepared before applying the changes. Therefore, <Commit> messages are sent to other participating nodes. A node must receive $2f_b + 1$ <Commit> messages before applying a change. Finally, at least $f_b + 1$ nodes must send <Reply> messages to the client to commit the acknowledgment. One major advantage of PBFT, compared to permissionless protocols, is the ability to provide an immediate transaction finality. This immediate finality is due to the fact that there is an agreement by all the honest nodes on the state of the system through the intra-communications with each other. The PBFT consensus mechanism has some limitations despite its advantages, such as the high communication overhead $O(n^2)$ between the replicas.

- Non-Byzantine consensus [3] Fail-stop, or crash failures, are another type of fault that occurs when a failed node halts, becomes unresponsive, or crashes. Paxos is one of the consensus protocols that tolerate crash failures f_c, and it follows a two-

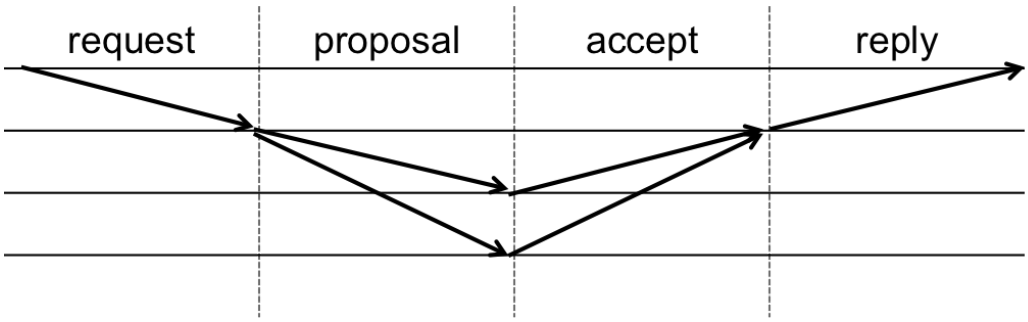


Figure 2.3: An Overview of normal Paxos case operation [3].

phase process, as shown in Fig.2.3, Prepare/Promise and Propose/Accepted. If the leader is relatively stable, Multi-Paxos [65] can be used to reduce the failure-free communication complexity where the Prepare/Promise phase cab be skipped.

The Paxos is started by sending a <PaxosPrepare> message to a group of acceptors $2f_c + 1$. The prepare message specifies a numeric sequence number that must be larger than any proposal sent. Upon receiving the proposal, the other participating nodes check if the sequence number from the prepare message is greater than any other proposal. If yes, it replies *via* a message called <PaxosPromise> to inform the leader that it will never accept any proposed value with a sequence number less than one proposed. When the sender receives promises from $f_c + 1$ nodes, it begins the second phase to commit the data value. The leader then sends a message <PaxosPropose> to the acceptors. This message can be seen as a command asking the acceptors to accept the proposal. Whenever an acceptor receives an <PaxosPropose> message and has not seen another message with a higher sequence number, it accepts the proposal and sends an <PaxosAccepted> message to all participated nodes. Upon receiving <PaxosAccepted> message from $f_c + 1$ nodes, the new value is committed. Paxos has fewer communication rounds, two rounds compared to three for PBFT. Thus, when Byzantine faults are not a concern, Paxos can be used

to improve communication overhead and network latency.

Blockchain in Internet of Things

In the last few years, most of the blockchain-IoT integration efforts are about overcoming the limitations mentioned above by filling the gap between several architectural solutions.

- Cloud-Centric blockchain model: One way to provide the blockchain for IoT is by merging IoT and cloud computing. This integration has been used to overcome the IoT limitations of processing, storage, and access. Therefore, blockchain as a Service (BaaS) has emerged as a model allowing consumers to use cloud-based services to develop, use, and host their blockchain systems [66]. The BaaS providers have to manage the back-end services, specifically the complicated services for clients and businesses. It is their responsibility to look after the crucial blockchain technology artifacts and keep them running without any downtime. Blockchain services also include hosting requirements and security protocols such as distribution of resources, anti-hack layering, bandwidth management, and much more. With a BaaS model, customers will be able to focus more on the core business functionalities and strategies while depending on the BaaS partner to manage the blockchain infrastructure and performance. In addition to the hosting challenges, trust and privacy are two major drawbacks that prevent integrating BaaS in the IoT networks. Moreover, cloud computing usually provides a centralized architecture, which, in contrast to blockchain, complicates reliable data sharing with the participated devices.

- Fog-Centric blockchain model: Fog computing is used as an option that takes the cloud computing paradigm to the network edges [67]. IoT systems usually leverage fog computing to guarantee a smooth movement from traditional IoT cloud models toward a blockchain-based network. The fog nodes can work as

blockchain gateways and enable interaction with core blockchain and IoT devices. The nodes of a fog layer have multiple segments of blockchain, which reduce the processing and memory requirements. However, they generate a large amount of traffic to maintain synchronization with the global blockchain. A multitude of aggregation and synchronization protocols are initiated to aggregate the blockchain segments of IoT fog to the global blockchain without overburdening the blockchain network [68]. The essential target is not keeping the devices synchronized but to synchronize them according to the application's needs. The information is gathered and forwarded when the end devices need them. These protocols need to consider the varied communication requirements, such as the total time required by the blockchain synchronization process, along with the minimum requirement of networking bandwidth for synchronization to occur. Nevertheless, aggregation and synchronization timing overheads are not compatible with a large number of delay-sensitive IoT applications. The fog model, furthermore, cannot guarantee secure and seamless data propagation between the IoT devices and the fog nodes. To overcome these limitations, researchers need to reconsider designing new alternative blockchain solutions that can fulfill the strict challenges and requirements of IoT.

- Other solutions: Distinct research efforts focus on amplifying the scale of the blockchain and boosting overall performance. Data management solutions that use deep reinforcement learning is proposed [69] to maximize the amount of data collection, geographic fairness, and to minimize energy consumption. Another databased solution is proposed [70] in which the blockchain platform uses a lossless data compression method to store data in the main chain. However, the estimated blockchain size is not feasible for a large IoT network. One way to address the challenge previously mentioned is by using local chain techniques to provide the needed scalability and practicability [67]. In this context, the

data processing is done according to the complexity level, wherein the off-chain deals with the complex problems that the on-chain network cannot solve. The proposed solution does not store the exact data in the off-chain and only stores the data's hash-values. It cannot, therefore, assure information availability and reliability. On the other hand, the authors propose a blockchain-based non-repudiation service provisioning scheme [71] for Industrial IoT network. The requested services are split into non-executable parts and then implemented *via* on-chain and off-chain channels in different phases to limit the burden on the blockchain network. This method targets the non-repudiation services and needs to be extended to include other security factors, including trust and privacy. Other effective distributed ledger technologies are established with diverse architectures. Among these techniques, the Directed Acyclic Graph (DAG) is the most popular blockchain alternative based on a non-linear ledger framework [72]. The major insight of DAG is that the performance is not restricted by the limitations of the consensus approach but rather by ledger formulation. DAG introduces the blockless chain concept in which the data records are not encapsulated inside a block. Each data record is verified and run directly into the DAG networks. Blockless DAG evades the overheads linked to the chain of blocks and effectively improves performance. IOTA Tangle [73], and Nano [74] are two examples of a chain using Blockless DAG. In IoT networks, blockless DAG systems pose some challenges due to storage and data processing limitations. For instance, IOTA Tangle has an approximate size of 80 Gigabytes [75], and nano uses 29 Gigabytes [76] of memory. Thus, the resource-constrained IoT devices are incapable of storing the DAG. Nonetheless, the research efforts cannot provide satisfactory results until the fundamental inherited problems from the underlying networking infrastructure are removed and improved.

Consensus Protocols in Internet of Things

An instant consensus agreement is needed to satisfy the speedy transaction aspect of IoT systems. State of the art consensus protocols can take from a few seconds up to several minutes to finalize the data on the blockchain. PoW consensus mechanism with different implementations such as Bitcoin [55], Litecoin [77], Ethereum [56] has a finality time between 2.5 to 60 minutes [78]. PoS [60] in Ethereum Casper [79] has a better finality ranging from 2 to 15 minutes [78]. On the other hand, consensus algorithms that follow the Byzantine Fault Tolerance (BFT) principles, e.g., PBFT [2], DBFT [80], and DPoS with PBFT [81] can provide a block finality in the order of seconds [82]. For that, the BFT-based consensus protocols, with some improvements, can be considered the most suitable of candidate protocols addressing this issue. As a proof of suitability, a list of established IoT blockchain projects that implement BFT and its variants is introduced as follows:

- IBM Watson IoT Platform [58] is specially designed for IoT devices providing a managed, cloud-hosted blockchain and analytic services. The data can be captured and explored to help the organizations achieve their objectives. The consensus is provided by IBM Hyperledger Fabric, which uses a BFT algorithm.

- IoT Chain [83] is a small operating system that built on top of the blockchain concept and provides a proof-of-hardware security option. The large number of IoT nodes within the network help to provide decentralized data protection. IoT Chain uses PBFT to achieve main chain consensus.

- IoTeX [81] consists of two blockchains, the root-chain and the sub-chains, which are connected to form a single architecture for heterogeneous computing. To accelerate the transaction speed, IoTeX uses Delegated Proof of Stake (DPoS) and PBFT as the consensus mechanisms.

- NEO [80] the smart economy is the main objective of NEO that can be activated by using blockchain technology and digital identity. Smart economy is, therefore, the result of smart contracts, digital identity, and digital assets. Nodes on NEO use a Delegated Byzantine Fault Tolerance (DBFT) algorithm.

BFT is partially decentralized, presenting the limitation that trust is placed in known validator nodes. Moreover, these algorithms are believed to have high communication complexity. BFT-based consensus algorithm is widely studied in the literature. However, a limited number of works have focused on performance evaluation of BFT and its variants in the context of IoT networks and blockchains. Ripple, Hyperledger Fabric v0.6 with PBFT consensus, and Hyperledger Fabric, are based on v1.0, with BFT-SMaRt [63] consensus and are evaluated [84]. The experimental results show that Byzantine consensus algorithms offer a reasonable throughput. That being stated, their performance does not scale to a large number of devices and drops dramatically as the number of participated devices increases. Moreover, the work in [85] provides an overview of the blockchain platforms used in industrial IoT. The authors conclude that the current systems need to be modified to meet the special needs of the IoT network.

Chapter 3

General IoT System Modeling and Analysis

The IoT system is composed of different heterogeneous devices interconnected to each other to collect different kinds of information. It is an incredibly diverse space, encompassing a large variety of applications and services with different requirements and needs. In this chapter, we provide the required analysis of the underlying network to serve IoT applications. Such analysis shows if the underlying network under various parameters can successfully serve the target IoT application requirements, if not, which parameters or knobs need to be tuned.

3.1 Queuing Modeling and Kendall Notation

In a queuing system [86], the arriving packets are serviced by one or more servers. Packets who arrive to find all servers busy generally join one or more queues (lines) in front of the servers, hence the name queuing systems. Describing the queuing system using mathematical expressions is done by defining a system model that makes probabilistic assumptions about the arrival rates, service times, the number and type of servers, and the provided queues' discipline. Queuing model provides several performance measures formulas to design a new service system or improve an existing one.

- Arrival Process: In specifying a queueing model, we must make assumptions about the arrival processes' probabilistic nature. A Poisson process is one of the most known assumptions about arrival rates. Poisson arrivals are named

according to the Poisson distribution, where the number of arrivals in any given period can be random. In time duration t, there is $N(t)$ number of arrivals, and $N(t)$ has a Poisson distribution that can be expressed as,

$$Probability\,\{\mathrm{N(t)} = \mathrm{n}\} = \frac{e^{-\lambda\, t}(\lambda\, t)^n}{n!}$$

where e is the natural number, and λ, called the *rate*, is the expected number of arrivals per unit time. An exponential distribution, which is the time between consecutive arrivals, is another way to characterize the Poisson process, called the *interarrival time*. So if IA is the interarrival time of a Poisson process with rate λ,

$$Probability\,\{\mathrm{IA} \leq \mathrm{t}\} = 1 - e^{-\lambda\, t}$$

Memoryless is an important property of the exponential distribution that means the independent time of arrivals between two consecutive events. Therefore if the packets' arrival process is Poisson process then the packets arrive independently from one another and that can be expressed analytically. For this reason, the Poisson process is considered the most *random* arrival process.

- Service Mechanism: A queuing system's service mechanism is specified by the number of servers, each server is having its queue or a common queue, and the probability distribution of packets service time. let S be the service time then, the mean service time μ is, $\mu = 1/S$

- Queue Discipline: If all servers are busy upon a packet's arrival, they must join a queue. Queues are physical lines of things or packets. The queue discipline is the rule that determines the order for serving the queued packets. The most common discipline is the familiar first-in, first-out (FIFO) discipline. However,

priority queue discipline is used to increase efficiency or reduce the latency for more delay-sensitive packets. Priority queues can be further categorized into preemptive or non-preemptive queues, depending upon whether service in progress can be interrupted when a packet with a higher priority arrives.

D. G. Kendall [87] introduces a set of notations which have become standard in the literature of queuing models. A general queuing system is denoted by,

$$A/B/m/K/n/D, where$$

- **A:** distribution function of the interarrival times
- **B:** distribution function of the service times
- **m:** number of servers
- **K:** capacity of the system, the maximum number of packets in the system including the one being serviced
- **n:** population size, number of sources of packets
- **D:** service discipline

M is expressed as exponentially distributed random variables, which is the abbreviation for $Markovain$ or $memoryless$. In a Kendall notation, we can omit the population size and the capacity if they are infinite and the service discipline is considered as FIFO. E.g., $M/M/1$ means a queuing model with Poisson arrivals, exponentially distributed service times, and a single server. $M/G/m$ denotes $m - server$ system with Poisson arrivals and generally distributed service times. $D/M/r/K/n$ stands for a system where the packet arrived from a finite-source with $n - elements$ where they have a constant or deterministic D arrival, the service times are exponentially

distributed, the service is carried out according to the request's arrival by $r - severs$, and the system capacity is K.

3.2 IoT-based System Model

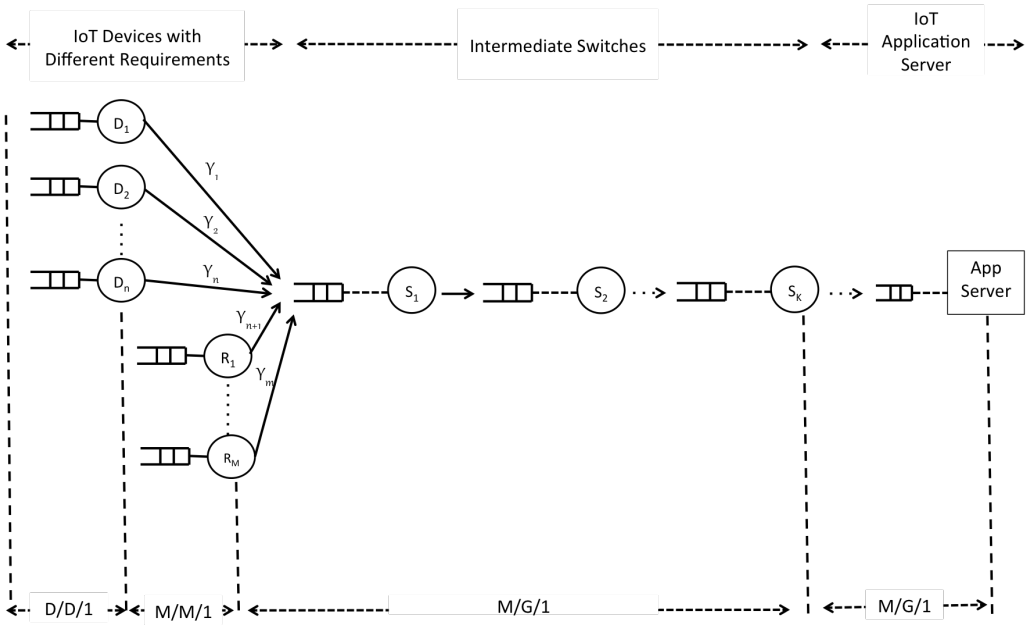


Figure 3.1: General overview of the IoT-based System: we have multiple sensors injecting information to the network switches, which forward packets across the network to the application server.

Throughout this chapter, we consider a general IoT system model. As shown in Fig.3.1, There are M IoT devices collect certain information, and then, transmit them over multiple switches to an IoT application server. IoT information i.e., data packet arrival can be model with deterministic and/or random packets inter-arrival times with rates $\gamma_m, m = 1, 2, ..., M$. The intermediate switches K and the application server are modeled as single server facilities with M/G/1 queue, with exponentially distributed packet arrival rate and arbitrary distribution for service rate of mean $\overline{X}_i$ for node $i \in \{1, 2, \ldots, K\}$. They have multiple priority queues that are responsible for forwarding the data to the application server [86]. These switches are considered

to be always running. Deterministic IoT device has an inter-arrival rate of

$$\lambda_D = \sum_{i=1}^{n} \gamma_i, \tag{3.1}$$

while random packet arrivals are assumed to be Poisson distributed with rate

$$\lambda_R = \sum_{j=n+1}^{M} \gamma_j \tag{3.2}$$

3.3 End-to-End Delay and Dropping Analysis

With the advancement of today's converged next generation networks (NGN), the conceptual framework of providing various IoT real-time and mission-critical services over the Internet is becoming a reality. Some of the major IoT real-time services that are expected to be widely available over the Internet in the near future are the multimedia (voice and video) services. These services are delay-sensitive in nature and providing QoS assurance over the Internet Service Provider (ISP) networks has been a major research challenge. In order to provide guaranteed QoS for delay-sensitive applications, the ISP network resources need to be reserved and allocated efficiently and optimally such that these services are assured. There is no doubt that such resource reservation for guaranteed services requires the service subscribers to pay higher than other best-effort service classes. It is a challenge to identify how these real-time services can be guaranteed over ISP infrastructure with minimal cost, and something that is scalable in nature. Therefore, analysis studies for achieving the strict guarantees of such kinds of IoT applications need to be studied closely.

IoMT applications are considered where the QoS requirements of IoMT applications vary greatly, and the underlying network must be aware of these dynamic variations. An application such as weather or environment monitoring which aggregates samples over an hour or even a day is much less delay sensitive than a surveillance

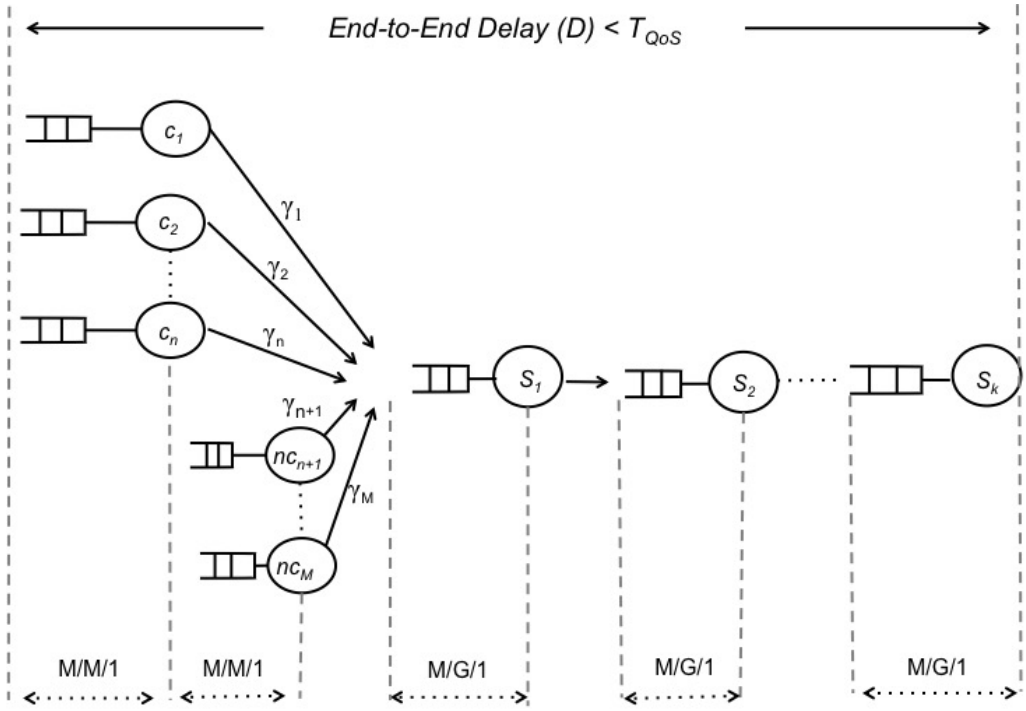


Figure 3.2: An overview of the Internet of Multimedia Things system.

monitoring system for example, which requires an in-time response. Moreover, due to the dynamic characteristics of IoMT applications, it happens that some packets of the same application might gain higher importance due to the current context. Packets that hold alerts or are coming from an area where a security breach occurred will suddenly become more important than other sensing packets of that same application. We categorize these IoMT packets into critical and non-critical packets.

On the other hand, QoS focuses on the objective measurement of network parameters such as jitter, throughput, loss, and delay only, without giving full attention to the service delivery quality (QoE). QoS is considered an influencing factor for the QoE perceived by the end-users (machines or humans). For the multimedia traffic, the QoE adding a new dimension to provide a complete understanding of the system characteristics with respect to the network measures of the QoS. However, with the new advances in SDN, network policies and applications deployment can be implemented on the spot and the dynamic changes on the network QoS policies and users

QoE requirements can be reflected. Each application can define the QoS thresholds (e.g., real-time road safety in highway service requires the round-trip time (RTT) to be between 10 and 100 milliseconds [4]), provide the needed QoE (Video bitrate, frame rate, the resolution of the mobile device, etc. [88]), and the network should make the correct decisions such as, which priority queue to use, which routing path to select, where to deploy the server middleboxes and, etc. that would enable these packets to meet their requirements. In this chapter, we provide the required analysis of an underlying network to serve real-time IoMT applications.

We consider a typical IoMT application scenario, shown in Fig.3.2, we assume that that the network is aware of the critical IoMT packets (through packet classification or flow matching), and thus, schedule them to the higher priority queue denoted as Q_k^H, where $k = 1, 2, ..., K$ is the switch index. The non-critical packets are scheduled to the lower priority queue $Q_k^L, k = 1, 2, ..., K$. In some special cases, some non-critical packets could suddenly become critical according to the current context.

Recall, IoMT data that is collected from an IoMT application. Therefore, such packets are scheduled in the high-priority queue with probability $1 - p_t$, where p_t is the probability of scheduling a sampling data packet to Q^L. In general, both critical (c) and non-critical (nc) packet arrivals are assumed to be Poisson distributed. Poisson traffic is used to obtain closed-form results in M/G/1 queues, with rates λ_c and $\lambda_{nc} = \sum_{m=1}^{M} \gamma_m$, respectively. Since there are two different types of traffic and consequently priority queues, the total rate of high-priority traffic λ_H and low-priority traffic λ_L is given by:

$$\lambda_H = \lambda_c + (1 - p_t) \cdot \lambda_{nc} \tag{3.3}$$

$$\lambda_L = p_t \cdot \lambda_{nc} \tag{3.4}$$

In order to maintain a guaranteed behavior for the real-time IoMT traffic, the

expected delay experienced by each data packet since its transmission till it reaches the data sink node must not exceed a predefined threshold T_{QoS}. Therefore, a data packet with a cumulative delay exceeding T_{QoS} upon arrival to the application server is flagged, and then an appropriate decision is employed. Thus, if we let D be a random variable that stands for end-to-end delay, then the probability of dropping a data packet at the application server is given by:

$$P_{\text{blocking}} = \Pr\{D > T_{\text{QoS}}\} \tag{3.5}$$

In this case, our objective is to compute the maximum allowable hop-count K^* such that $\mathbb{E}[D] < T_{QoS}$, and estimate the packet dropping probability for a given number of hops in the network. Looking into the above objectives, we aim to analyze how the above metrics vary with the network's application policies. It presents a complex model that captures many of the inherent tradeoffs in IoMT traffic engineering and network design.

3.4 Model Analysis

As mentioned earlier, our network consists of K switches with two priority queues and, possibly, different service rate distributions. Initially, we are interested in obtaining the average waiting time $\overline{W_p}$ and second moment $\overline{W_p^2}$ for priority $p \in \{L, H\}$ experienced by sampling data packets at each switch. Since the output of each queue is approximately Poisson distributed process, we look into each switch in isolation and temporarily drop the switch index i. Thus, the waiting time of a high priority packet, denoted as W_H, is

$$W_H = \sum_{j=1}^{N_H} X_j + R, \tag{3.6}$$

where N_p Number of packets scheduled in the queue with priority p, $p \in \{L, H\}$,

X_j is the service time of packet j and R is the residual time. From (3.6), using Little's formula, we obtain the average waiting time of the high priority queue $\overline{W}_H$ as,

$$\overline{W}_H = \frac{\overline{R}}{(1-\rho_H)}, \tag{3.7}$$

where the first moment of this residual time is

$$\overline{R} = \frac{1}{2}\left((\rho_H + \rho_L) \cdot \frac{\overline{X^2}}{\overline{X}}\right) \tag{3.8}$$

To reiterate, $\rho_H = \lambda_H \cdot \bar{X}$ is the fraction of time a switch is serving high priority traffic. Thus, the second moment of waiting time for high-priority traffic:

$$\begin{aligned} \overline{W_H^2} &= \overline{N}_H \cdot \mathrm{Var(X)} + \mathrm{Var(R)} + \mathrm{Var(N_H)} \cdot \overline{\mathrm{X}}^2 \\ &\approx \overline{N}_H \cdot \mathrm{Var(X)} + \mathrm{Var(R)}, \end{aligned} \tag{3.9}$$

where

$$\begin{aligned} \overline{N}_H &= \lambda_H \, \overline{W}_H \\ \rho_H &= \lambda_H \cdot \overline{X} \\ \mathrm{Var}(X) &= \overline{X^2} - \overline{X}^2 \\ \mathrm{Var}(R) &= \overline{R^2} - \overline{R}^2 \\ \mathrm{Var}(N_H) &= \overline{{N_H}^2} - \overline{N_H}^2 \end{aligned}$$

From the above, we understand that if high priority traffic is limited, then high priority queues contain at most one packet almost all the time, and waiting time for high priority packets is chiefly due to residual time only.

For lower-priority traffic, the waiting time of a low priority packet can be expressed

as

$$W_L = \sum_{j=1}^{N_H} X_j + \sum_{j=1}^{N_L} X_j + \sum_{j=1}^{W_L \cdot \lambda_H} X_j + R \tag{3.10}$$

Equation (3.10) essentially implies that the waiting time of a low-priority packet is the summation of four components: residual time due to service, the time to serve existing high priority packets, the time to serve existing low-priority packets that are ahead in the queue, and the time to serve new high priority packets that arrive while the low-priority packet is waiting. Thus the average waiting time of the low priority queue denoted by $\overline{W}_L$ is,

$$\overline{W}_L = \frac{\overline{R}}{(1-\rho_H)\,(1-\rho_H-\rho_L)}, \tag{3.11}$$

and, the second moment is

$$\begin{aligned} \overline{W_L^2} &= \overline{R^2} + s \cdot \mathrm{Var}(X) \\ &+ s^2 \cdot \overline{X}^2 + 2\,s\,\overline{X} \cdot \overline{R} + q \cdot \overline{X}^2 \\ &\approx \overline{R^2} + s \cdot \mathrm{Var}(X) + s^2 \cdot \overline{X}^2 + 2\,s\,\overline{X} \cdot \overline{R}, \end{aligned} \tag{3.12}$$

where the coefficient s is defined as

$$s = \overline{N}_L + \overline{N}_H + \lambda_H \cdot \overline{W}_L$$

Here, q is variance of the number of packets in the three cases mentioned earlier and is assumed to be negligible in steady state, hence the last approximation follows.

One immediate check is to note that (3.12) reduces to (3.9) whenever $\lambda_{nc} \to 0$, which agrees with expectation because in the latter case lower-priority traffic essentially becomes higher-priority traffic.

Maximum Hop Count

The maximum allowable hop-count that respects the QoS delay constraint is given by

$$K^* = \arg\max_K \left\{ \sum_{i=1}^{K} \overline{T}_i \leq T_{QoS} \right\}, \tag{3.13}$$

where

$$\overline{T}_i = (1 - p_t) \cdot \overline{W}_{H,i} + p_t \cdot \overline{W}_{L,i} + \overline{\chi}_i + \tau_i, \tag{3.14}$$

where $\overline{W}_{H,i}$ and $\overline{W}_{L,i}$ are respectively given by (3.7) and (3.11) for each $i = 1, 2, \cdots K$, χ_i is the service time random variable at switch i, and τ_i is the propagation delay between switch $i-1$ and switch i.

In the special case where all switches are identical, we obtain the simpler expression:

$$K^* \approx \frac{T_{QoS}}{\overline{T}}, \tag{3.15}$$

with $\overline{T} = \overline{T}_i$, $i = 1, 2, \cdots K$. To draw qualitative insights, we note in the latter case that if traffic is limited, i.e. $\rho \approx 0$, then K^* is approximately given by the first order Taylor expansion:

$$K^* \approx \frac{T_{QoS}}{\overline{\chi} + \bar{\tau}} \cdot \left(1 - \rho \frac{\overline{R}}{\overline{\chi} + \bar{\tau}}\right) \tag{3.16}$$

Here, $\bar{\chi} + \bar{\tau}$ is the minimum average delay at a node regardless of traffic utilization. Hence, the impact of increasing traffic utilization in IoMT application is largely influenced by the ratio of residual service time $\overline{R} = \overline{X^2}/(2\overline{X})$ to such minimum average delay.

Blocking Probability

In order to obtain the average number of packets that exceed a predefined threshold T_{QoS}, we approximate the sum of all service times and waiting times experienced by a packet from source to sink by a normal distribution [89]. We also know the first

and second moments of the random variables as derived earlier. We are interested to obtain the end-to-end delay $D = \sum_{i=1}^{K} W_i + \sum_{i=1}^{K} X_i$ for both high-priority traffic D_H and low-priority traffic D_L. Thus, we have

$$D_p \sim \mathcal{N}(\mu_p, \sigma_p), \tag{3.17}$$

where

$$\mu_p = \sum_{i=1}^{K} \overline{\chi}_i + \sum_{i=1}^{K} \overline{\omega}_{p,i}, \tag{3.18}$$

$$\sigma_p^2 = \sum_{i=1}^{K} \mathrm{Var}(\chi_i) + \sum_{i=1}^{K} \mathrm{Var}(\omega_{p,i}), \tag{3.19}$$

where χ_i denotes service time at node i and $\omega_{p,i}$ denotes waiting time at node i for traffic with priority $p \in \{L, H\}$.

Therefore, the final desired probability of arriving later than T_{QoS} is a weighted sum according to whether a data packet is scheduled in the high-priority queue or the low-priority queue:

$$\begin{aligned} \mathrm{P}[\mathrm{D} > T_{\mathrm{QoS}}] \;&\approx\; \frac{1}{2}(1-p_t)\left[1 - \mathrm{erf}\left(\frac{1}{\sqrt{2}} \cdot \frac{T_{\mathrm{QoS}} - \mu_{\mathrm{H}}}{\sigma_{\mathrm{H}}}\right)\right] \\ &+\; \frac{1}{2}\,p_t\left[1 - \mathrm{erf}\left(\frac{1}{\sqrt{2}} \cdot \frac{T_{\mathrm{QoS}} - \mu_{\mathrm{L}}}{\sigma_{\mathrm{L}}}\right)\right] \end{aligned} \tag{3.20}$$

where $\mathrm{erf}(\cdot)$ is the error function [90, Eq.(8.250.1)].

Service Placement

For the multimedia IoMT traffic, providing the QoS constraints are the first step for the network operators to deploy the application servers. The objective QoS metrics are considered an influencing factor for the multimedia QoE parameters where delayed packets create representation quality problems and packet losses cause gaps in the received traffic for real-time IoMT applications. QoE can be defined as a function of

the QoS provided by the network as follow:

$$QoE = f(QoS) \tag{3.21}$$

This can be done by considering quality models able to map objective measures (QoS) with subjective measurements such as The Mean Opinion Score (MOS) [91]. MOS is calculated as an arithmetic mean by:

$$MOS = \frac{\sum_{i=1}^{N} R_i}{N}, \tag{3.22}$$

Where R_i the individual ratings for a given stimulus by N subjects. Such subjective test results can be collected by the application operators and injected to the SDN controllers that use both the QoS and QoE measures to set the polices in order to place the service near to the end users.

3.5 Performance Evaluation

In this section, the performance of the above framework is evaluated using MATLAB/Simulink. We evaluate the QoS measurements and assume the subjective QoE tests are collected and injected by the human operators. For this work's analysis purpose, we ignore the underlying communication technologies and use the abstract model as shown in Fig.3.2.

Two types of IoMT applications are considered, critical and non-critical applications. For the real-time critical IoMT application, the end-to-end delay requirement of an Intelligent Transport System (ITS) is evaluated, as shown in table 1.1. The ITS services have to operate with end-to-end delay of less than 100 milliseconds [4]. By taking the given requirements into consideration, we define our end-to-end delay threshold T_{QoS} to be 100 milliseconds.

We simulate critical application that generates data at a Poisson rate λ_c of one

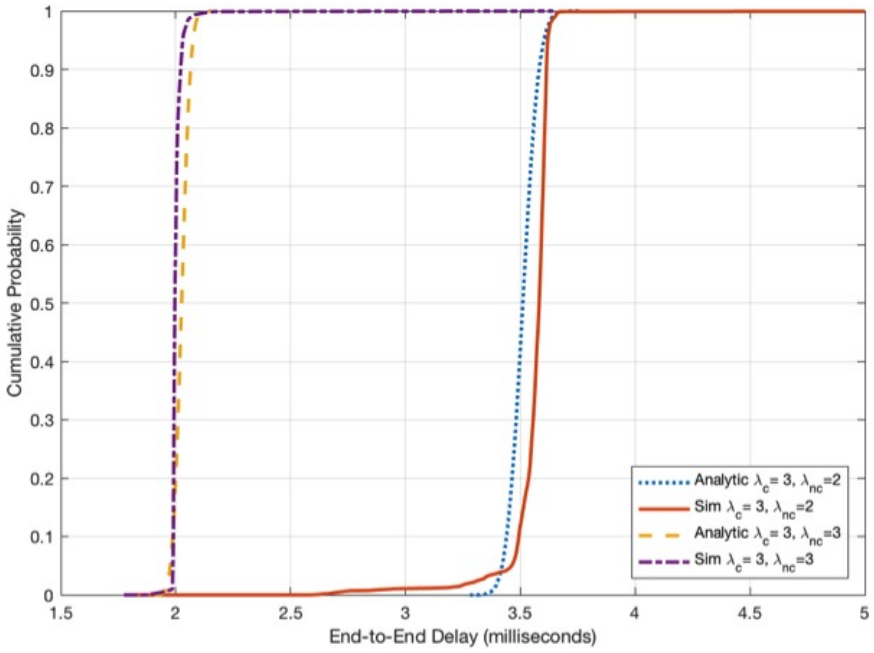


Figure 3.3: End-to-end delay for one hop.

packet every 3 milliseconds. We also simulate non-critical application that produces Poisson traffic with two different arrival rates λ_{nc} of one packet every 2 and 3 milliseconds. As described above, two measures play a major role in providing acceptable QoS requirements for critical applications, end-to-end delay, and maximum allowable network hop-count.

With regard to end-to-end delay, test results for one hop are shown in Fig.3.3. The average end-to-end delay for the critical application is around 2 milliseconds when both types of applications have an arrival rate of one packet every 3 milliseconds $\lambda_c = \lambda_{nc} = 3$. As the number of low priority packets increases, $\lambda_{nc} = 2$, the contention with the high priority data packets increases, making the average delay to be 3.5 milliseconds for the critical application.

In order to verify the effects of the number of hops on the end-to-end delay, a simple experiment is done while increasing the hop-count K from 10 to 70, Fig.3.4 demonstrates the average end-to-end delay for the critical application. Both analytic

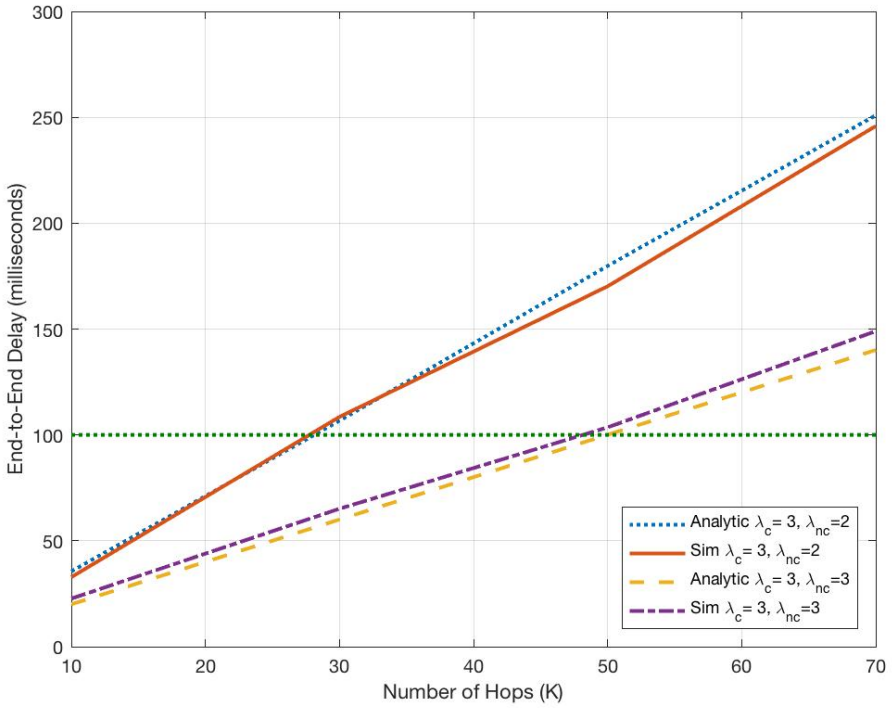
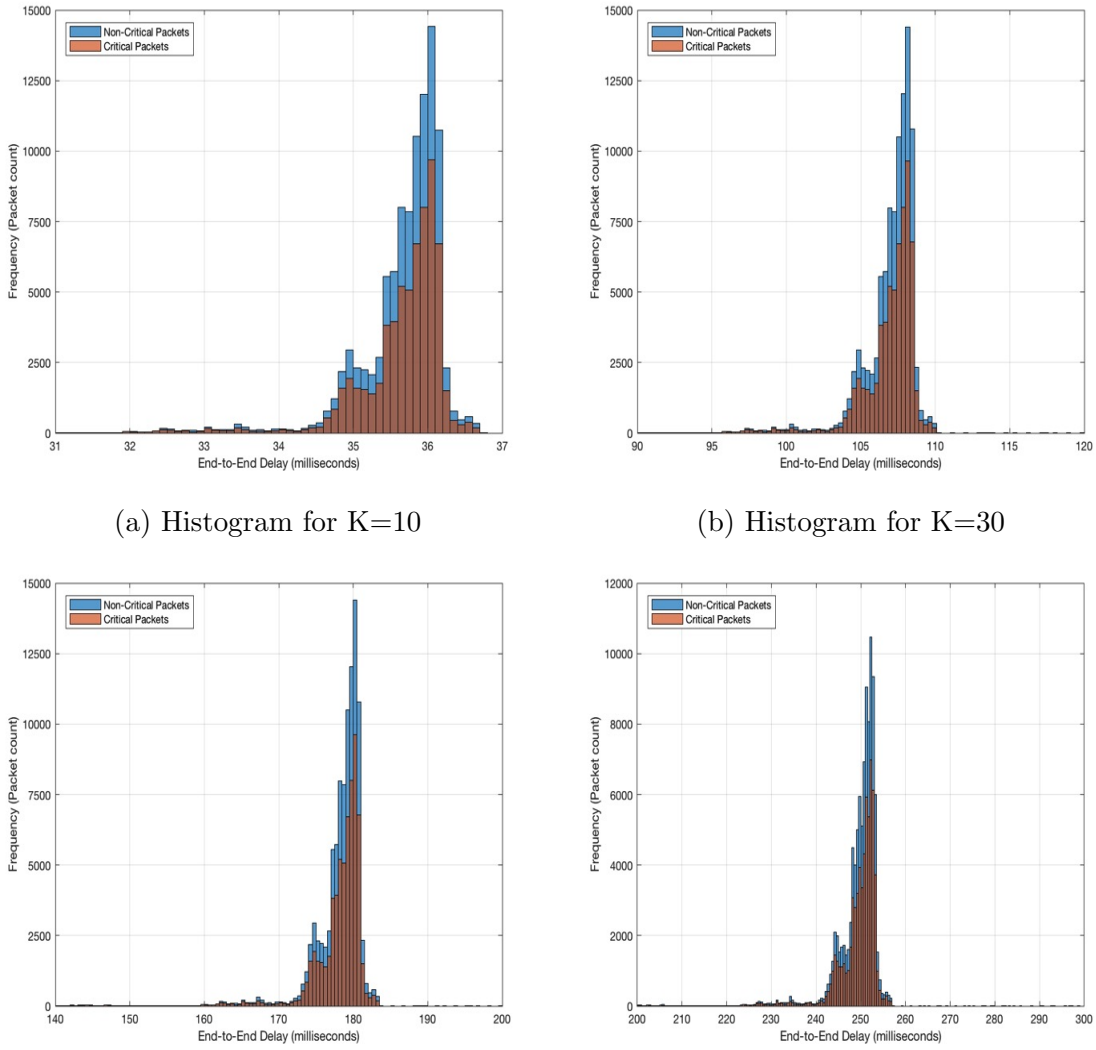


Figure 3.4: End-to-end delay for different number of hops (K).

and simulation results are obtained. For the case of $\lambda_c = \lambda_{nc} = 3$, the average delay ranges between 20 and 60 milliseconds for 10 and 30 hops, respectively, that is still within the acceptable 100 milliseconds threshold T_{QoS}. The delay reaches the 100 milliseconds limit at the 50^{th} network hop. With non-critical packets arrival rate $\lambda_{nc} = 2$, the end-to-end delay easily hit the T_{QoS} at the 30^{th} hops with an average of 106 milliseconds. This gives us a good reason to define the limit on the number of hops a packet is allowed to traverse at the design stage of IoMT network infrastructure. Propagating such performance information toward the network edges can help to find the best route for the critical packets to the destination and meet the required end-to-end delay constraints.

The end-to-end delay distributions for various hop counts, and both critical $\lambda_c = 3$ and non-critical $\lambda_{nc} = 2$ applications are considered, as shown in Fig.3.5. The histogram in Fig.3.5a illustrates the delay spread from about 32 milliseconds to 37



(a) Histogram for K=10

(b) Histogram for K=30

(c) Histogram for K=50

(d) Histogram for K=70

Figure 3.5: End-to-end delay distribution.

milliseconds at $K = 10$. This spread is getting wider as the number of hops increased. For $K = 30$ in Fig.3.5b, the delay ranges between 95 and 120 milliseconds, and between 140 and 200 for $K = 50$ in Fig.3.5c. The delay data expand to a range of 100 milliseconds (between 200 and 300 milliseconds) for $K = 70$ in Fig.3.5d. For all cases and as expected, the end-to-end delay maintains its distribution characteristics.

Finally, the packet dropping probability as a function of K is plotted in Fig.3.6. Both simulation and analytic results are displayed for the two different arrival rates

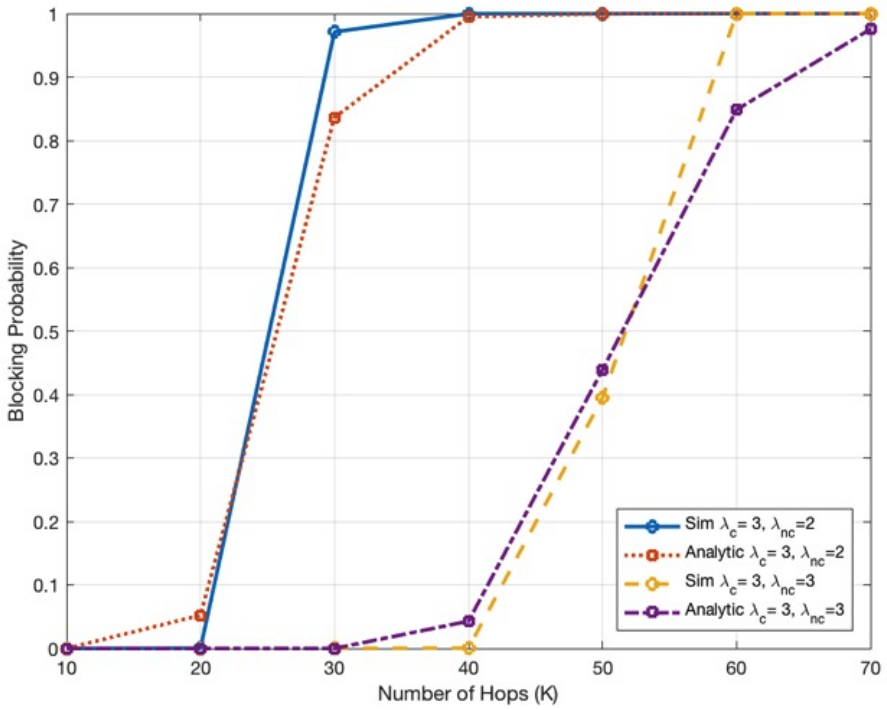


Figure 3.6: Packet dropping probability v.s number of hops.

of non-critical applications $\lambda_{nc} = 2$ and $\lambda_{nc} = 3$. When $\lambda_{nc} = 2$, the dropping probability is less than 1% for $K = 10$ and $K = 20$ and escalated up to more than 80% for $K = 30$. Thus, 100% of the packets have an end-to-end delay above the adequate limit of T_{QoS} when K is greater than 30 hops. Again with the case of less contention between critical and non-critical packets, $\lambda_{nc} = 3$, the packet dropping probability is below 1% for $K = 10$, $K = 20$, and $K = 30$. This probability rises up to 5% for $K = 40$, and then to 43% for $K = 50$ and continues to grow to more than 80% at 60^{th} and 70^{th} network hops. As shown from the Figures, our simulation closely matches the analytical results for all performance metrics.

Considering these results, it is critical for systems designers to analyze and quantify the inherent tradeoffs between QoS and service placement. End-to-end delay and packet dropping are key metrics for the applications performance that need to meet strict threshold values. In the next chapter, we analyze the IoT high traffic load and

propose a method to deploy the required specifications dynamically toward the users at real-time to fulfill the IoT critical requirements.

Chapter 4

Traffic Volume and Load Modeling and Analysis

In Chapter 3, we modeled and analyzed the IoT-based systems and concluded that the legacy networking configuration can not satisfy the application-level requirements. Moreover, faulty IoT devices can produce a high volume of data traffic and drain the system resources. Controlling and distinguishing data traffic under faulty controlled devices have been quite challenging. To address the mentioned issue, we propose an instantaneous detection method for IoT (IDIoT) and examine IoT communication protocol where IoT devices communicate in a Publish/Subscribe architecture. We employ Message Queuing Telemetry Transport (MQTT) communication protocol as our Publish/Subscribe model. Furthermore, we use the M/G/1 queuing model to characterize the MQTT central point and propose a detection technique that helps prevent flooding the network near the edges.

4.1 Instantaneous Detection of IoT Volumetric Traffic Model

Several IoT protocols have been designed to fit low-cost, low-power, and resource constrained devices. Message Queuing Telemetry Transport (MQTT) [20] provides standard communication foundations for the IoT implemented as Publish/Subscribe architecture. In this section, we adopt two kinds of IoT applications, event-based and participatory applications, and their associated traffic patterns. It is worth noting that each traffic type requires different processing time.

Our system considers MQTT application scenario, shown in Fig.4.1, in which M

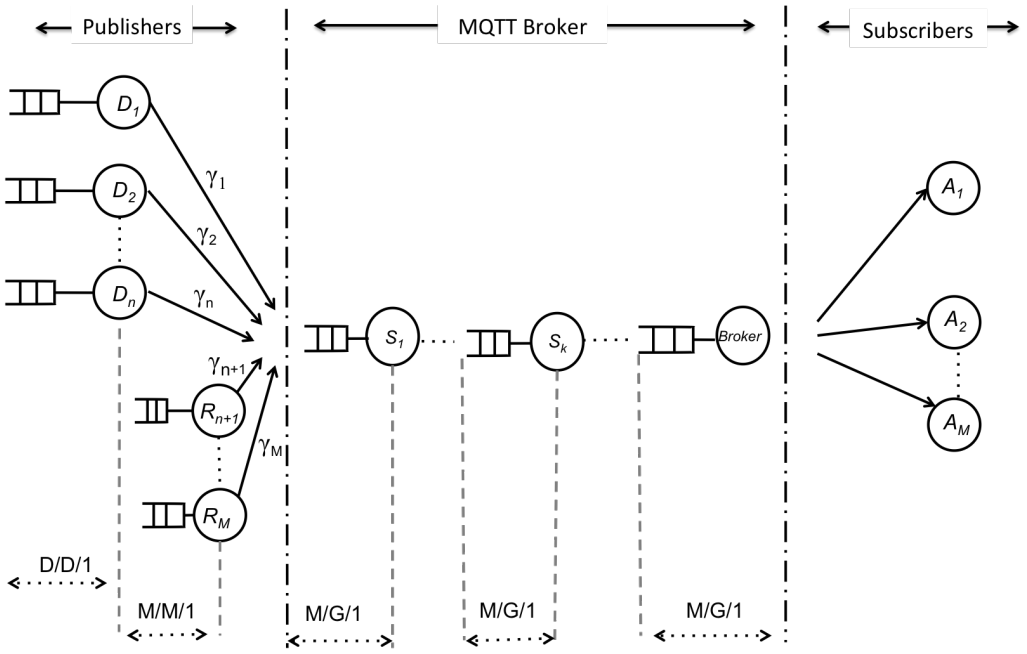


Figure 4.1: An overview of IoT-MQTT system.

is the number of MQTT publishers (IoT devices). The broker diffuses the interest data to the subscribe applications ($A_i, i = 1, 2, ..., M$) and the traffic management information to the switches on the path to block or allow MQTT clients to alleviate high network load. We employ two priority queues with exponentially distributed packet arrival rate and arbitrary distribution for service rate. Hence, the intermediate node and the relay switches are modeled as single server facilities with M/G/1 queue that responsible for forwarding the IoT data to the subscribed clients. Due to the distributed nature of the IoT devices, the broker is the bottleneck point in our network. We aim to limit the number of packets that arrive to the central broker to prevent overwhelming the network. The broker operates in non-preemptive M/G/1 priority queues with exponentially distributed inter-arrival time λ, and mean service time of $\overline{X}$. Random packets (i.e., event-based IoT traffic) are considered to have the higher priority.

Packets arrival rates (traffic velocity), and total number of publishers (traffic volume) are two important factors to reduce network load. Our model experimentally

maintains and analyzes a number of previous packet arrival rate at the IoT broker to identify the confidence in which an increased arrival rate (λ) is in the application's defined range or not. In this case, the rate of the packets must not be greater than the maximum arrival rate λ_{max}. On the other hand, the maximum allowable publishers M_{max} should be determined to meet the QoS requirements. Thus, the probability of blocking an IoT device to prevent high network load is given by,

$$P_{\text{blocking}} = \Pr\{\lambda > \lambda_{\text{max}} \parallel \text{M} > \text{M}_{\text{max}}\} \tag{4.1}$$

Our ultimate objective is to analyze the prevention method for high traffic load in IoT networks (IDIoT) while minimizing the negative effect on the system users. To address the above concern in IoT network, we analyze the above mentioned queuing model for an IoT-MQTT network, and evaluate a detection mechanism based on the collected traffic arrival rate and the number of users before a high network load takes place.

4.2 Model Analysis

In this section, we present our analytical expressions for the major QoS performance measurements that can be affected by a high network load in terms of the system response time, the queuing delay, and the total number of packets arrived to the central broker [86]. In our system, we are interested in analyzing two types of data packets, random (R) and deterministic (D) packets with different priority classes. For each priority class $p \in \{R, D\}$, the packets arrive according to a Poisson process with $\lambda = \lambda_D + \lambda_R$, service time is of mean $\overline{X}$, and the second moment is $\overline{X^2}$.

To achieve our goal, we first note that the average waiting time of high priority packets (i.e., random packets), denoted as $\overline{W}_R$, is given by,

$$\overline{W}_R = \frac{\overline{R}}{(1-\rho_R)}, \tag{4.2}$$

where $\rho_R = \lambda_R \cdot \bar{X}_R$ is the fraction of time a broker is serving high priority traffic, and $\overline{R}$ is the residual time. This residual time can then be written as,

$$\overline{R} = \frac{1}{2}\left((\rho_R + \rho_D) \cdot \frac{\overline{X^2}}{\overline{X}}\right) \tag{4.3}$$

The average waiting time expressions, denoted as $\overline{W}_D$, of the lower class priority packets (i.e., deterministic packets) can be expressed as,

$$\overline{W}_D = \frac{\overline{R}}{(1-\rho_R)\,(1-\rho_R-\rho_D)}, \tag{4.4}$$

where the load ρ_D is defined as $\rho_D = \lambda_D \cdot \bar{X}_D$. The total time packets from class $p \in \{R, D\}$ spend in the system is,

$$\overline{T} = \overline{W} + \overline{X} \tag{4.5}$$

The total number of packets for each class $p \in \{R, D\}$ is given by,

$$\overline{N} = \lambda \cdot \overline{W} \tag{4.6}$$

By collecting and experimentally analyzing sufficiently large number of historical arrival rate λ_{max} for each type of traffic at the MQTT broker, we achieve a clean separation between IoT devices. The proposed model enhances the responsiveness of the IoT broker upon any packets' flooding. When any superfluous requests from single or multiple MQTT publishers are identified, the broker reacts accordingly to address such volumetric traffic and stops the IoT devices with abnormal behavior. From(4.6) and by applying Little's law, the arrival rates for each class $p \in \{R, D\}$

can be calculated as,

$$\lambda = \frac{\overline{N}}{\overline{W}} \tag{4.7}$$

In large packet counts caused by a legitimate burst, a maximum value of consecutive λ in short time interval is expected to be close to λ_{max} since the high utilization of network resources remains only for a short period of time. A larger value of λ can be considered that the event is more likely due to non-legitimate traffic load.

To maintain a predefined QoS, the total delay experienced by the detection method must not exceed a predefined threshold T_{QoS}. Therefore, the cumulative delay can be directly linked with the sampling interval, denoted by I, to collect and calculate the current maximum λ. Thus, if we let $\overline{Q}$ be a random variable that stands for total queuing delay, then the maximum allowable sampling interval that respects the QoS delay constraint is given by,

$$I = \arg\max_{I} \left\{ \overline{Q} \leq T_{QoS} \right\}, \tag{4.8}$$

where,

$$\overline{Q} = \overline{W}_R + \overline{W}_D, \tag{4.9}$$

with $\overline{W}_R$ and $\overline{W}_D$ are given by (4.2) and (4.4) for event-based and participatory-based clients, respectively.

Moreover, the target system can be flooded with requests from large amount of distributed IoT publishers. Thus, there is a hard limit on the number of publishers that each system can serve. This limit can be defined by the total delay constraints T_{QoS} imposed by the application requirements. Then, the maximum allowable number of publishers the system can serve is,

$$M_{max} \approx \frac{T_{QoS}}{\overline{Q}}, \tag{4.10}$$

where M_{max} is the maximum number of IoT publishers for both types (participatory-based and event-based) applications. By delegating the above traffic management information (λ_{max} and M_{max}) to the network edges, we can avoid high network load and control IoT volumetric traffic.

4.3 Performance Evaluation

In order to verify our proposed model, a simulation is conducted using MATLAB/Simulink, where the network topology is presented in Fig.4.1. The MQTT core node is simulated as a single M/G/1 queue with a generally distributed service time of 2000 [92] packets/second. The publisher nodes are attached to the MQTT broker, and as mentioned above, two categories of published applications are considered with constant and Poisson data traffic. We simulate weather signal application of three cities where each city has 1000 IoT devices that generate data at a constant rate of one packet every 15 milliseconds [12]. We also simulated four car parking lots with 1000 IoT sensors produce Poisson traffic at a mean arrival rate of one packet every 64 milliseconds [12]. As described above, two factors played a major role in avoiding high network load, publisher's maximum arrival rate λ_{max}, and total number of publisher nodes M_{max}.

The simulation results related to the maximum arrival rate λ_{max} are obtained. For this reason, our simulation runs for a relatively long time to maintain a number of historical arrival rates. From Fig.4.2, we observe that smart parking traffic rates range between 0 and 9 packets/millisecond, while the weather signal application has a maximum of 8 packets/millisecond. MQTT broker maintains a list of the maximum allowed arrival rate for each client and communicates the traffic policies (maximum

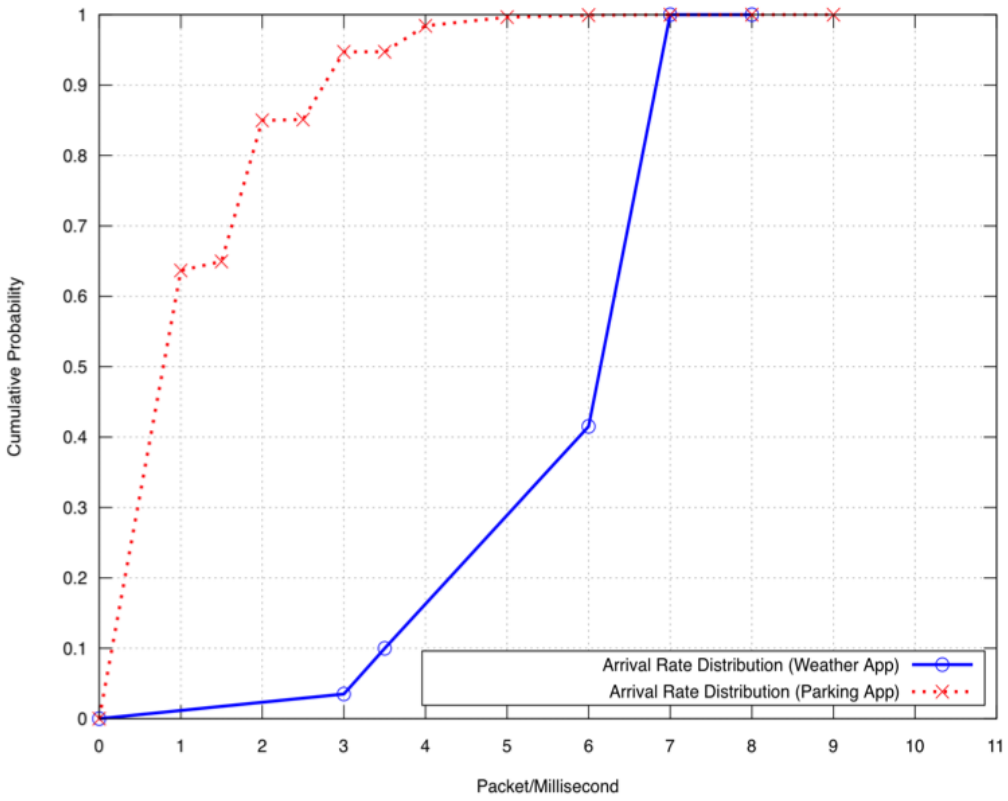


Figure 4.2: Packet arrival rate distribution.

arrival rate) to the programmable entry-point switches. The switches continuously collect the statistics and update their decisions based on network conditions and the broker's needs. At this point, if the arrival rate of a specific MQTT client (IoT device) within a sampling interval of 100 milliseconds is higher than the maximum arrival rate with 98% confidence, then that client is blocked to eliminate network load.

The reaction of the system with and without our IDIoT method has been simulated. We run the simulation for 100 seconds. At the 50th second, one city and two parking lots start sending at higher rates of one packet every 1.5 milliseconds and one packet every 6.4 milliseconds, respectfully. These kinds of changes in the data rates can be done intentionally or not. Our system identifies the overrated devices and only reacts to them while maintaining the system behavior stable to other normal users. As per Fig.4.3, MQTT server's average utilization is 0.13 in normal condition,

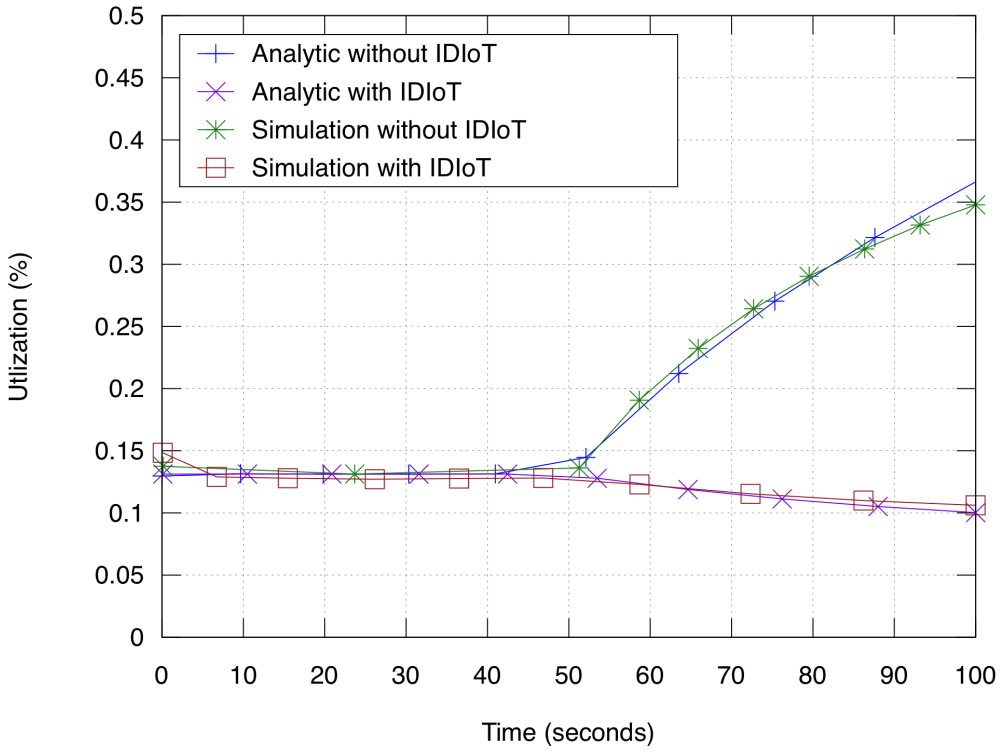
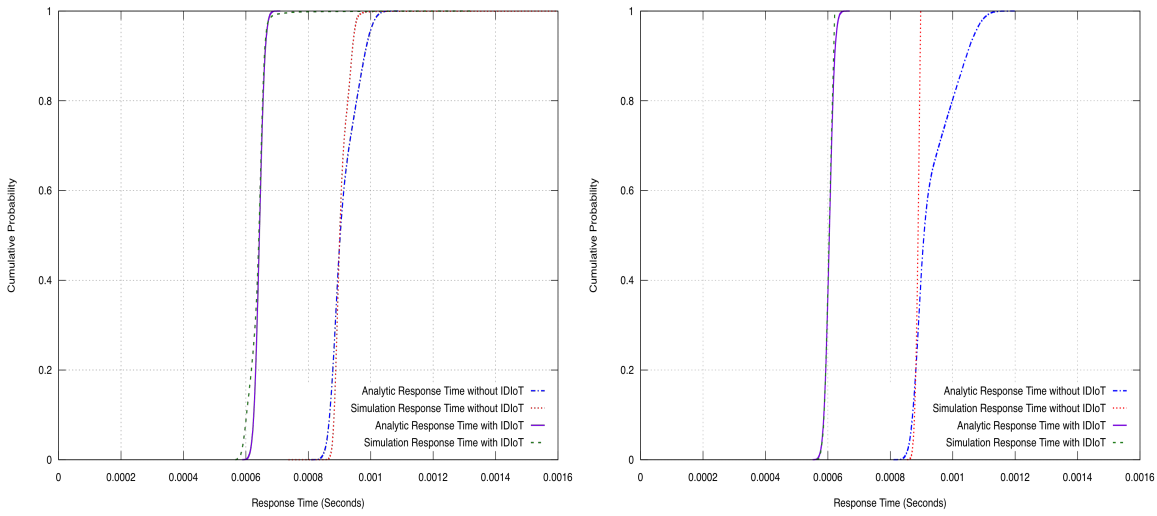


Figure 4.3: The MQTT broker utilization.

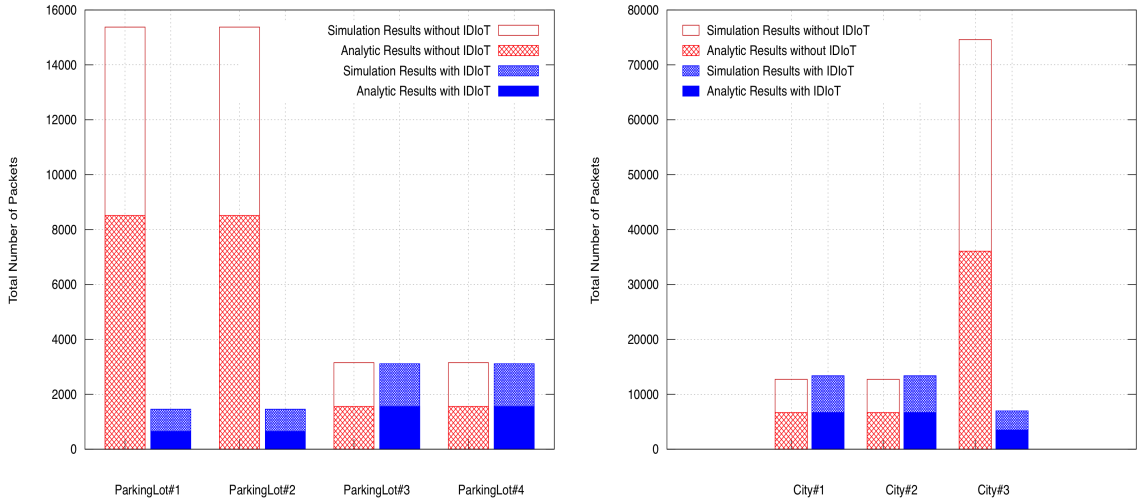
but it escalates to 0.39 under the network load. What can be inferred from this is that, under a high network load situation, the server is extremely burdened with high volume requests hence the increase in utilization. At some point, the server eventually reaches a very high utilization and cannot accept any requests from the system clients. As opposed, when our technique is considered, the MQTT broker can identify clients with abnormal behavior precisely by their client ID and maintain the system at a stable state in order to respond to other clients. That clarifies the benefits of incorporating the power of the MQTT-IoT protocol and the IDIoT method.

The network load merely delays or prevents data packets from being processed for legitimate connections; additional delays degrade the QoS of traffic ongoing connections. IDIoT should be robust and efficient to improve QoS performance for traffic when subjected to such a threat. To validate this, we further examine the correctness of the analytic expressions for $\overline{T}$ (i.e., response time in single M/G/1 priority queues).



(a) Parking lot application (b) Weather signal application

Figure 4.4: System response time.



(a) Parking lot application (b) Weather signal application

Figure 4.5: Total number of packets.

As shown in Fig.4.4, the proposed technique reduces the system response time and prevents QoS degradation performance by blocking the overrated nodes. Parking lots application in Fig.4.4a has an average response time of 0.91 milliseconds under high network load without employing our technique and an average of 0.64 milliseconds with the IDIoT method. Weather signal application in Fig.4.4b has almost the same reaction with an average of 0.65 milliseconds and 0.93 milliseconds with and without

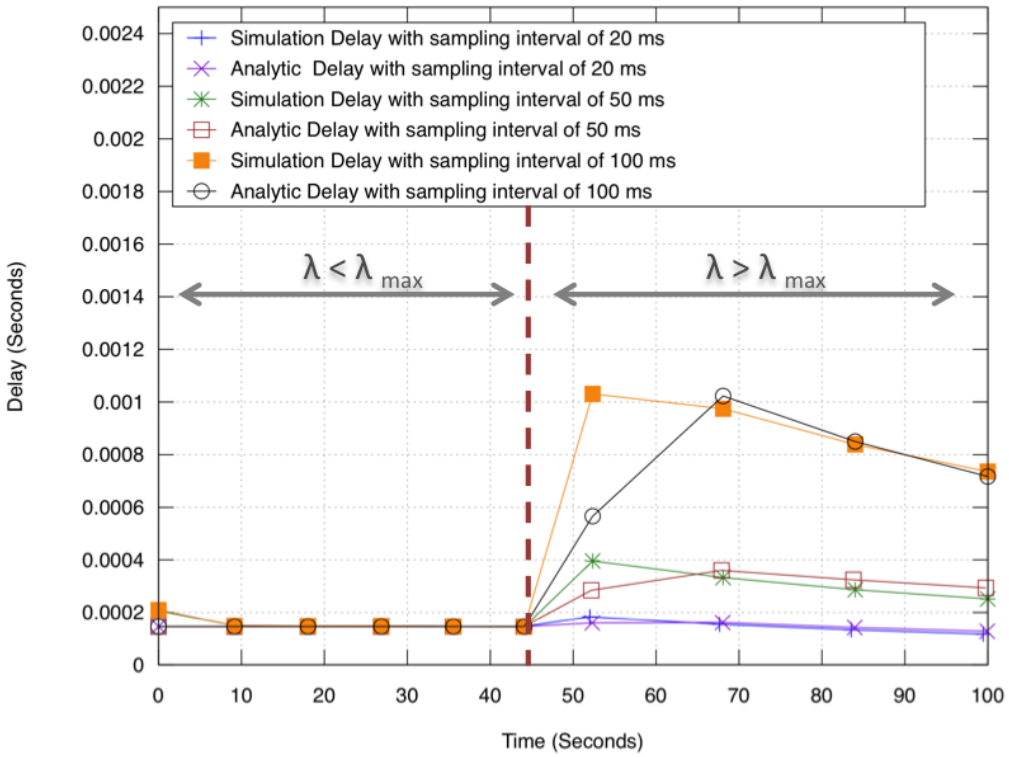


Figure 4.6: Queuing delay v.s sampling interval.

IDIoT, respectively.

The number of blocked packets is another important measure and a definite decision should be made at the right time to prevent flooding the system with huge amount of useless data packets. By considering the total number of packets $\overline{N}$ for each client, we can obviously show the advantage of our method. Under a normal situation in Fig.4.5a, we can see that each parking lot participates by around 1556 packets and when the data rate increases the misbehaved parking lots send out around 8513 packets. In our proposed method and before the load takes place, the first and second parking lot send out only 648 packets to the broker, then the broker detects an increase in the arrival rates and prevents them from accessing the system. Fig.4.5b shows the same situation is happened with the weather signal application. IDIoT prevents around 35000 unuseful packets from the third city which can be responsible for breakdown the overall system quality.

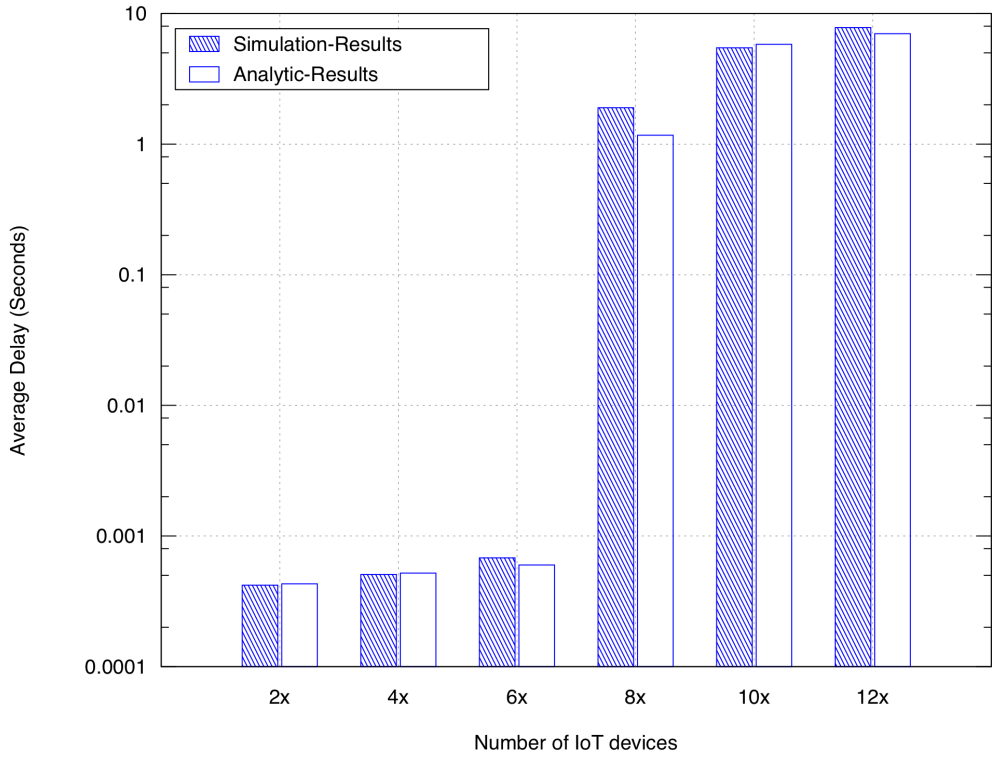


Figure 4.7: Queuing delay v.s number of IoT publishers.

The sampling interval is playing an important role to maintain the predefined QoS constraints. When the traffic volume increases, our system is still resilient against extreme cases (new arrival rate $\lambda_{new} = 5\lambda$) and identifying the overrated devices precisely by their client ID with some extra queuing delay cost. This queuing delay can be minimized by optimizing the sampling interval used to calculate the current arrival rate, as shown in Fig.4.6. With sampling interval of 100 milliseconds, the queuing delay is around 1 millisecond which can be further reduced to 0.2 millisecond by reducing the sampling interval to 20 millisecond. Optimizing the sampling interval should add another dimension to circumstantially reduce the performance overhead. We leave the sampling interval optimization as a future work.

The total number of IoT publishers must be controlled to maintain system stability and prevent IoT network load at the early stages. Any system can be able to serve a specified number of users. When the number of IoT devices is increased above

the adequate limit, a degradation in service quality appears. Thus, we conduct a simple experiment to measure the queuing delay as the number of IoT devices (for both applications) grows by a factor of (2x, 4x, 6x, 8x, 10x and 12x). As shown in Fig.4.7, the queuing delay is about 0.42, 0.5, and 0.68 millisecond for 2x, 4x, and 6x, respectfully. The delay continues to increase by an order of seconds till it reaches 7 seconds for (12x) which is totally unacceptable for most applications. Consequently, periodically monitoring the system at the network edges helps to achieve network QoS requirements. As shown from the above, our simulation results closely match the analytical results for all performance metrics.

Intelligent edge mechanisms can be supported by SDN. Therefore in the next chapter, we focus on providing QoS support for emerging IoT applications which is motivated by the new SDN capabilities and the unique QoS needs of IoT applications to support advance QoS mechanisms.

Chapter 5

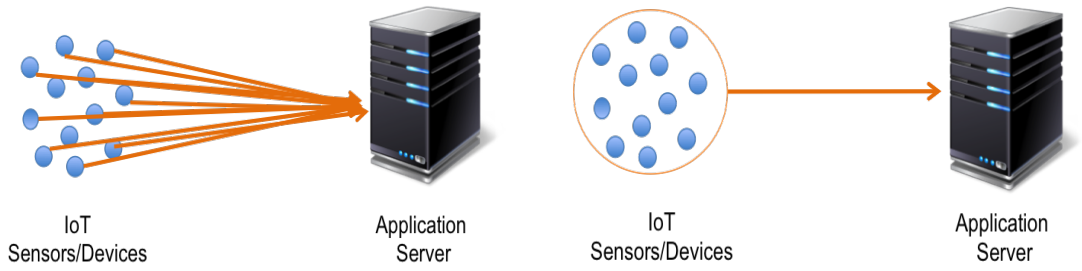
Context-Driven QoS-Enabled Networking

In Chapter 4, we educed that high IoT load needs to evade at the network edges intelligently. For this reason, we propose an SDN-based framework, SADIQ (**S**DN-based **A**pplication-aware **D**ynamic **I**nternet of things **Q**uality of service), that provides IoT applications with rich, context-driven QoS. With the growing trend towards SDN in cloud data centers, ISPs, and more recently in Wireless Sensor Networks (WSNs) and fog domains [93, 94, 95], we envision SADIQ to be deployed in any SDN-enabled network along the path of the IoT communication.

5.1 QoS Requirements

To support the QoS requirements of the location-based applications, we need to revisit both the abstraction that is used to express these requirements as well as the QoS mechanisms that are used to enforce these requirements.

- Static, across application QoS is insufficient: Traditional QoS techniques prioritize one application over the other (e.g., giving VoIP traffic higher priority over web browsing) but we need "Intra-application" differentiated (or preferential) services – so for the same application's traffic, some packets are more important than others, based on the context, and should be treated accordingly by the network. Note that the context depends on the application needs, and may extend beyond the state carried in the packet (e.g., prioritize video surveillance traffic if security alarm is triggered).



(a) A sensor-to-server flow. (b) All sensors-to-server flow.

Figure 5.1: Current sensors-to-server flow abstraction.

- Current flow abstraction no longer works: In today's Internet, a transport flow is a logical connection between two machines (or processes to be more specific). Using the current "flow" based abstraction for IoT applications, each (Sensor-to-Server) connection represents a single flow. This mapping is too fine-grained as seen in Fig.5.1a; each sensor connection on its own is insignificant and it will neither be practical nor scalable to define rules for each individual sensor connection. On the other extreme as shown in Fig.5.1b, considering (All Sensors-to-Server) connections as a single flow are too coarse-grained and do not allow any Intra-application QoS policies to be carried on. Therefore, a new high-level abstraction is needed to express suitable QoS polices for IoT applications. The abstraction should be simple, yet flexible enough to capture a wide range of applications' requirements.

5.2 SADIQ: SDN-based Application Aware Dynamic IoT QoS

In SADIQ, Fig.5.2 shows SADIQ high-level architectural view, we introduce a flexible, location-based *flow* abstraction for IoT called IoTFlow. A location-based abstraction is a natural fit for IoT applications since these applications typically involve sensing which is inherently tied to the location of the sensed data (e.g., where the photo was taken or where the temperature reading was done). IoTFlow also offers flexibility, as it leverages a grid-based location referencing system which allows dynamic control

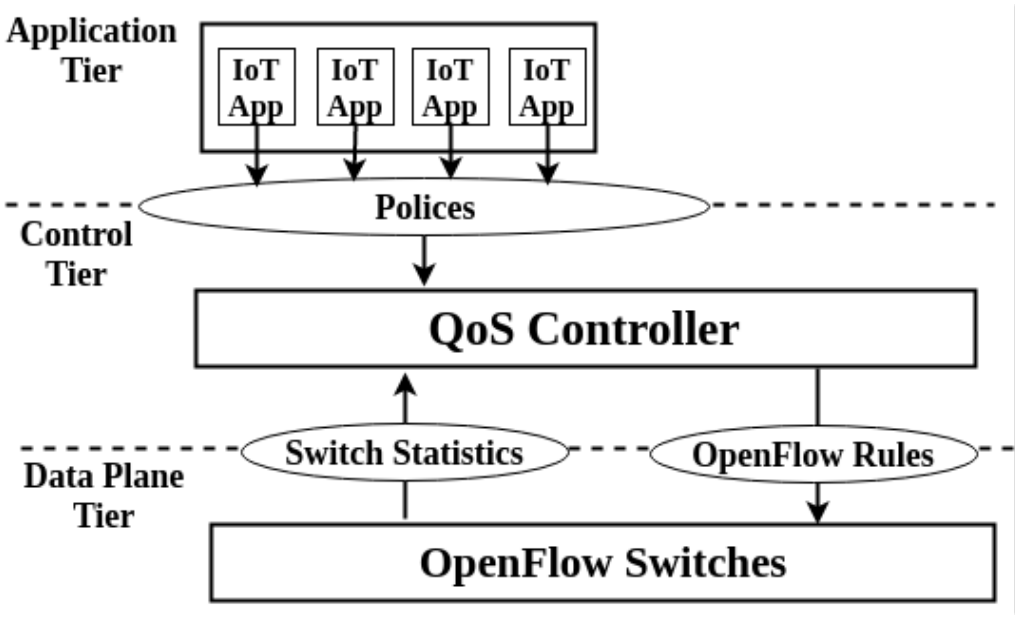


Figure 5.2: SADIQ architectural overview.

over the location granularity (going from fine-grained mapping to coarse-grained and vice-versa) at different levels of the system (i.e., from the application to the SDN controller and the individual switches).

Using this abstraction, each IoT application can express QoS policies in a high-level, SQL-like language. The policy language enables the application to specify typical QoS actions (e.g., prioritization, bandwidth guarantees) at the granularity of IoTFlows. For example, an IoT application can mark traffic coming from a particular location as more important in response to some event (e.g., an emergency) or because of lack of prior data from that location. The location can be specified in an exact fashion or in an approximate way, leveraging the flexibility of the grid-based referencing system.

IoT applications communicate their policies to SADIQ's QoS controller which translates these polices into data plane Openflow rules, which are applied on the IoTFlow abstraction. SADIQ leverages the QoS support in OpenFlow (e.g., marking, metering, priority queues, etc) to realize the QoS rules in a flexible and dynamic fashion. The controller continuously monitors the effectiveness and feasibility of the

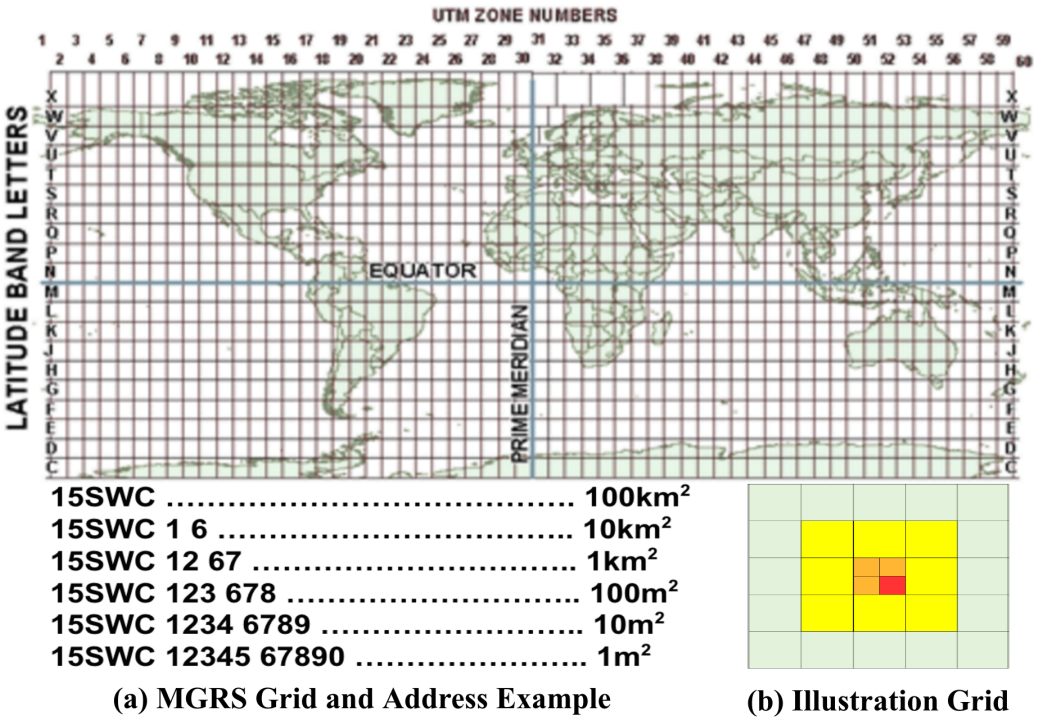


Figure 5.3: Military Grid Reference System (MGRS).

QoS policies by collecting statistics from switches – it updates its decisions based on network resource availability (e.g., level of congestion or number of flow entries available on a switch) as well as application needs. For example, if a switch has limited space for fine-grained policies, the controller can aggregate multiple locations into a single location, trading-off accuracy for switch rule space.

5.2.1 IoTFlow Abstraction

We propose IoTFlow, a high-level abstraction for IoT applications that allows rich QoS policies to be expressed. IoTFlow maps a group of IoT sensors from the same geographical location and working towards the same goal into a single flow. IoTFlow is enabled by adding a location to every IoT packet that is sent on the network. We use the MGRS [96] (Military Grid Reference System) as the location address. MGRS is a grid based referencing system (derived from the Universal Transverse Mercator (UTM) coordinate system) that defines geographical areas. Each MGRS location can

be represented in 16 bytes and has precision levels going from a 100 Km^2 to 1 m^2, which can cover an area as big as a city or as small as a bike parking space. The format of MGRS is 100 Km^2 zone designator followed by 0, 2, 4, 8, or 10 digits depending on the precision level. Fig.5.3(a) shows an example of an MGRS address with different precision levels. MGRS provides us three key benefits:

1. MGRS is considered easily interpretable since it is directly translated to distance being measured in meters, which is simpler compared to the degree increment of longitude and latitude.

2. MGRS allows defining geographical areas of different sizes, so we can choose a suitable granularity based on the requirements.

3. MGRS has a hierarchical grid nature that allows us to easily aggregate/disaggregate IoTFlows and apply QoS policies at different granularities based on the constraints of the network switches. For example, to go from a lower precision level to a higher one (from a small inner grid cell to the bigger outer grid cell), the address is simply truncated. This feature makes it attractive to create rules on commodity OpenFlow switches since the format is amenable to wildcards.

5.2.2 Policies

We design a simple, SQL-like, high-level policy language to specify QoS actions on IoTFlows. The policy language balances two aspects: it is fairly high-level, yet it is cognizant of the capabilities offered by the underlying SDN switches. The policy language uses new QoS specific keywords as well as other statistics and handles already available in SDN. Fig.5.4 presents the policy language syntax.

Each policy consists of a QoS action, the IoTFlows on which the actions needs to be performed, and optional clauses which put additional constraints on the policy:

- Actions: QoS actions depend on the underlying features of the SDN switches.

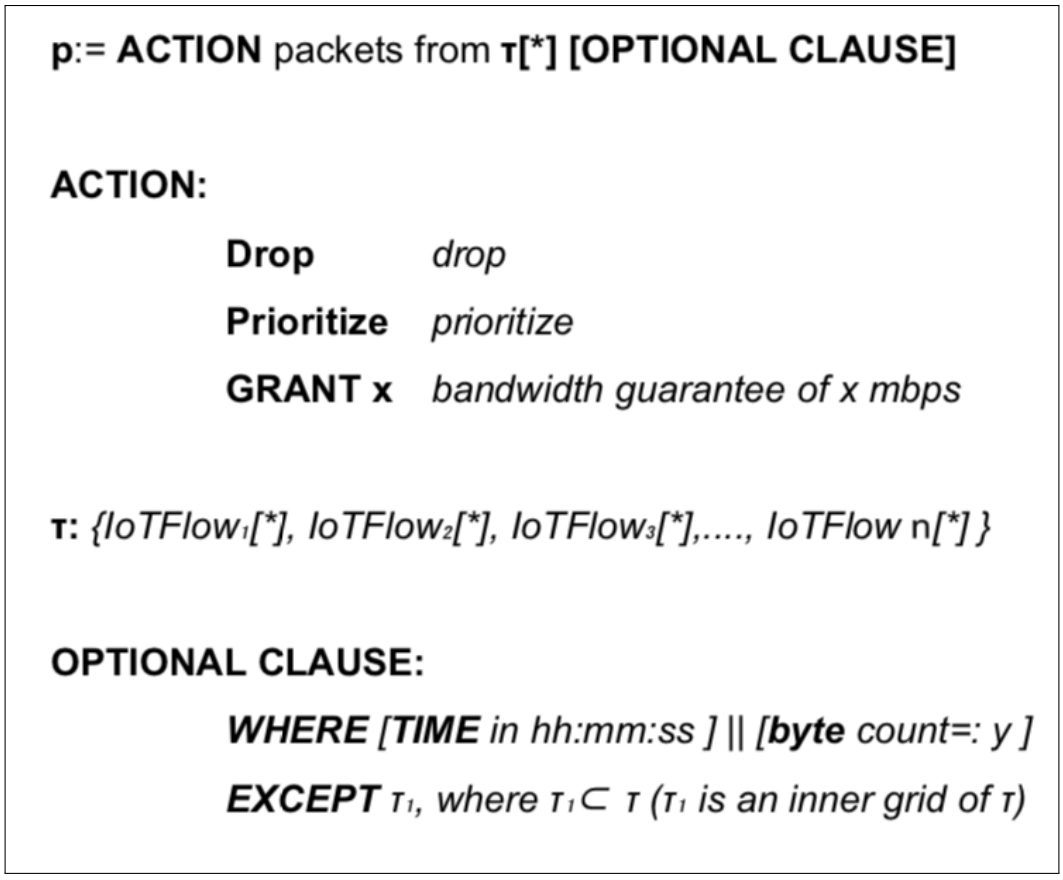


Figure 5.4: SADIQ policy language syntax.

SADIQ currently supports three actions: prioritizing, dropping, and bandwidth guarantees, but more actions can be added depending on the availability of new QoS features. Because of its richness, we focus on prioritization while describing the policy language.

- IoTFlows: For each action, we need to specify the IoTFlow(s) on which the action is performed. For the prioritization action, we can define a single IoTFlow or a list of IoTFlows (ordered or unordered). We can also define an approximate IoTFlow which can be interpreted differently by each switch based on its context. We describe these options by given the following examples:

    1. Exact addressing: This type of policy provides the least flexibility to the controller as it needs to use the exact same address to insert suitable rules

in the switches.

<PRIORITIZE packets from 15SWC1267 >

2. Unordered list addressing: In this case, the application is asking the network to prioritize those packets that belong to IoTFlows which are in the given list e.g., MICE list. The application can specify the MICE list itself or it can just specify the criterion for putting an address in the MICE list (e.g., if bytes sent are less than a threshold). The latter option allows the application to delegate the responsibility to the network (i.e., controller) to keep track of list membership. We can similarly also specify an ordered list, in which the IoTFlows are ordered based on their priority (from higher priority to lower priority).

<PRIORITIZE packets from LIST-MICE >

3. Approximate addressing: The example below shows the use of an approximate address:

<PRIORITIZE packets from 15SWC1267$*$ >

The $*$ indicates that this is an approximate address, so the network can consider other regions too, giving more preference to areas closer to this address compared to areas that are far off. The illustration grid in Fig.5.3(b) shows an example where the application specifies the smallest square in the middle to prioritize as an approximate address. Some switches (e.g., those experiencing high congestion) may just prioritize packets of this region but other switches may prioritize neighboring regions too, with lightly loaded switches prioritizing the whole big square shown in the figure. This example shows how SADIQ allows flexible policies that lets applications focus on what is important for them while the network can optimize based on the prevailing traffic conditions.

4. Multiple addressing: The above examples highlight the multiple uses of IoTFlow(s) in SADIQ with the PRIORITIZE action. As noted earlier, we support other actions too which can be applied in a similar manner. Also, we can combine two types of addresses with some operation between them:

<PRIORITIZE packets from LIST-MICE over LIST-ELEPHANT >

In the above example, the MICE list will only get preference over the ELEPHANT list and not against any other region.

- Optional clauses: Similar to SQL, SADIQ also supports two optional clauses – WHERE and EXCEPT – which can constrain the policy specification, add more information, or exclude specific IoTFlows. These clauses take predicates which support standard operations such as comparisons. The keywords that can be specified as part of these clauses include variables exposed by the SDN switches or controller (e.g., byte-count) or other predefined keywords like TIME. Some examples of using these clauses include:

<PRIORITIZE packets from 15SWC1267 WHERE TIME in [8am-10am] >

The above policy will prioritize packets of the specified IoTFlow only during the specified time. Similarly, the EXCEPT clause can be used in the following way:

<PRIORITIZE packets from 15SWC16 EXCEPT 15SWC1267 >

This will prioritize all packets from the 10 Km^2 except for packets coming from 1 Km^2 area inside it.

5.2.3 QoS Controller

The QoS controller translates the policies into OpenFlow rules that can be inserted in individual switches. The QoS controller extends a standard SDN controller – it has a global view of the topology as well as traffic conditions, but also has specific support for QoS, leveraging the QoS features available in SDN/OpenFlow switches. The key challenge for the controller is to translate the high-level policies into low-level switch rules, while keeping into account the constraints and characteristics of the switches and the traffic they are forwarding. We now describe the QoS building blocks that the controller uses, how it generates the OpenFlow rules, and how the rules are updated.

QoS Building Blocks

OpenFlow presents a flow table abstraction: each flow entry consists of a set of match header fields that define a flow, and an action field that defines how to process packets [97]. However, OpenFlow has many features that allow these simple flow entries to be enhanced to support richer rules. Our QoS controller uses the following building blocks to construct QoS specific rules.

- Set queue: Which forwards matched packets directly into a certain priority queue stating its ID.
- Set DSCP (Differentiated Services Code Point): Which changes the DSCP field in the packet's header that can be reflected on the queue selection once the packet is sent to the output port.
- Metering: Which directs matched packets into an OpenFlow meter table. Each OpenFlow meter table has a defined rate and an action, when a packet exceeds this rate the corresponding action is executed. Meter actions can be either to drop the packet or remark its DSCP. Metering is useful to create a simple rate limiter or to provide bandwidth guarantees.

- Wildcards: A flow entry can specify all match header fields explicitly, or allow aggregation of flows by using wildcards. Wildcards can be applied on an entire field, or just parts of the field using bitmasks.

- Rule priority: Each flow entry has a priority value that defines the flow entry's matching precedence. An incoming packet can match more than one entry in the flow table, however, only the entry with the highest priority is applied.

OpenFlow Switch Rules

The high-level QoS policies typically result in one or more switch entries that need to be inserted. The specific QoS actions – prioritization, bandwidth guarantees, etc – have corresponding building blocks in OpenFlow that are used. The controller keeps into account the state of the network in generating the entries, which provides context. For example, what applications are using the network, which hosts are involved, what are the ports and addresses used by these applications. This ensures that even though the policy is at a high-level, yet it is translated into suitable actions at the switch level. The controller also attempts to translate the high-level policies into as few flow entries as possible, thereby reducing the flow table size. For this purpose, the controller uses information about the network conditions based on the statistics reported by the switches. For example, if only few regions have events going on (i.e., they are of our interest) then the controller creates specific flow entries for every such region low-level routing their packets to a high priority queue, and a general flow entry that routes all other packets to the low priority queue. In contrast, if another application has a small number of elephant regions (and many mice regions) then the controller creates a flow entry per elephant region routing its packets to the low priority queue, and a single flow entry to route all other packets to a high priority queue. This is another example of how such low-level optimization can take place, with the help of the switches and the controller, without any explicit involvement of the application.

Eth_type	Src_IP	Dst_IP	IP_ Proto	UDP Dst_Port	IoTFlowAddress	Prioirty	Action
0x0800	*	173.255.200.24	17	9999	10SEG 98*** 32***	100	SET_QUEUE=1, Outport=4
0x0800	*	173.255.200.24	17	9999	10SEG 9**** 3****	50	SET_QUEUE=2, Outport=4
0x0800	*	173.255.200.24	17	9999	*	10	SET_QUEUE=3, Outport=4

Figure 5.5: Example of flow entries translated from high-level QoS polices.

We now walk through an example to show how the QoS controller uses these building blocks in translating application policies into switch rules. We consider a Smart City application that is monitoring a sports event taking place in a certain town in the city. The application wants to give high priority to packets coming from that town, but even higher priority to packets coming from the exact area of the sports arena. The QoS controller receives the policy from the application (to prioritize this location) – the policy contains the MGRS address of the 1 Km^2 grid cell surrounding the sports arena (e.g., 10SEG 98 32).

Fig.5.5 shows the flow entries required to support this policy. In this example, we assume that IoT packets are sent on top of UDP using a registered UDP destination socket port for IoT. Thus, IoT packets of this Smart City application are identified by matching the destination IP address, and the destination UDP port. However, matching against the new header field (IoTFlow address) enables differentiating between packets of the same application. By applying bitmask wildcards on the MGRS address, we are able to aggregate packets using different geographical granularity. All three entries map flows to the same output port but different queues with different priorities (1, 2, and 3). Packets coming from the sports arena area match all three entries, yet only the first entry is applied since it has the highest priority. Therefore, packets from the exact area of the event occupy the highest priority queue. The controller creates the second entry to aggregate packets coming from the same town but areas other than the sports arena by using the 10 Km^2 granularity of the address

(10SEG 9 3) and a lower priority. And lastly, all other packets coming from other towns that have no events only match the last entry leading them to the queue with the lowest priority.

5.2.4 Complete illustrative example

Before presenting the results of our experiments and their discussion, we first go through a full illustrative example of the workflow of our framework using one of the proposed applications, Melbourne Smart Parking [12]:

1. a Smart Parking application is based on a system of in-ground sensors that monitors parking spaces. Each in-ground sensor detects a parking action, i.e., when a car parks or leaves a parking space, and uploads this action to a cloud server. When receiving a parking action from a specific region, the application server recalculates the occupancy percentage of that region according to the occupancy equation mentioned earlier. The Smart Parking aims to provide higher parking accuracy to areas that are witnessing events, and thus, have high occupancy (since more people will be looking for parking spaces in that region).

2. The parking application sends the top x regions (or regions with occupancy higher that x%) to the controller to grant them higher priority, example:

   PRIORITIZE packets from 15SWC1267, 15SWC1289, 15SWC1437

   The controller receives these policies and updates the flow entries accordingly. If the controller detects high load in some of the switches it alters the flow entries to priorities only specific regions of events and not nearby regions to ensure the accuracy of parking reporting of those regions.

3. Steps 2 & 3 are periodically repeated with any change in the parking occupancy or the network load.

5.3 SADIQ versus SDN-based QoS-enforcement Proposals

SADIQ is inspired by SDN and the move towards separation of concerns between the data and control planes. In comparison with recent SDN Proposals such as [36, 38, 47, 48], SADIQ provides fine-grained control by focusing on the unique requirements of IoT applications, such as location-based QoS abstraction and policies, and leverages several SDN features to provide QoS. Unlike SADIQ, such studies rely on existing QoS mechanisms, only adaptive to changes in the network workloads, and are unaware of changes in the application context. Other proposals called for location awareness for IoT and sensor applications [50, 51], whereas SADIQ focuses on the QoS benefits from combining the location-awareness with the real-time context-awareness of IoT applications, and presents a complete system, including a new abstraction, policy language,and QoS controller.

Our work complements existing work on SDN controllers and SDN policy languages as it provides support which is specifically tailored for IoT applications. Similar to other controllers (e.g., Nox [98], Floodlight [28], Onix [99], etc), SADIQ's QoS controller bridges the gap between high-level application policies and low-level data plane support. Similarly, SDN languages, such as Frenetic [100] and Maple [101], also provide high-level policy abstractions to simplify controlling SDN networks. Recent proposals like QtKat [102] are most relevant as they define an SDN language with high-level abstractions for QoS primitives, such as queuing and rate limiters to simplify QoS control in SDN. QtKat reasons about QoS at the granularity of flows, which they generically define as set of packets that share some traffic characteristics. Moreover, their language also incorporates the concept of aggregation of multiple flows to be modeled as a single flow in terms of their QoS requirements. However, SADIQ introduces the location-based IoTFlow abstraction and flow aggregation based on location granularity.

5.4 Implementation

We now describe the implementation of SADIQ focusing on three key elements: the realization of IoTFlow, implementation of the QoS controller, and its interaction with OpenFlow switches.

5.4.1 QoS Controller

We extend the Floodlight [28] controller to implement SADIQ's QoS controller which consists of three modules:

- Policy manager: The policy manager defines a REST (Representational State Transfer) [103] API for the communication with applications. The applications send REST calls in a JSON (JavaScript Object Notation) [104] format that holds the application policy (IoTFlow, action, {optional clauses}) explained earlier.

- Network monitor: The network monitor periodically (every 5 seconds) collects statistics from the network switches, which include: ports, queues, and flow statistics. A notification is sent to the translator module if a certain switch statistics cross some defined thresholds. The notification includes the switch's datapath ID and its statistics list.

- Translator: The translator turns the application policies into FLOW_MOD messages to add or delete flow entries. The FLOW_MOD match specifies the IoTFlow address field with a bitmask corresponding to the granularity of the MGRS address selected by the application or the controller. The action sets the DSCP value according to the selected flow priority and forwards it to the next output port. The DSCP value is reflected on the queue selection once the packet reaches the port.

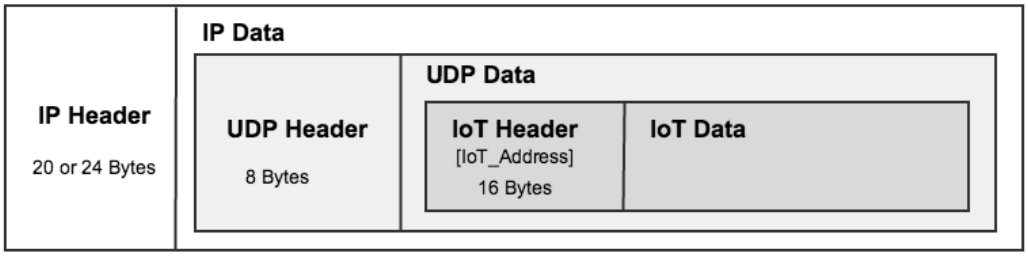


Figure 5.6: IoT packet header structure.

5.4.2 IoTFlow Abstraction

OpenFlow v1.3 (and later versions) supports the experimenter match fields: OFPXMC_EXPERIMENTER. SADIQ controller defines a new match field by adding its custom header to the OFPMP_TABLE_FEATURES request message to the switches. Many commodity switches, including the HP Aruba 2930F switch [105] that we use in our testbed, support defining custom match fields in Ternary Content-Addressable Memory (TCAM) tables with wildcard and mask capabilities. Using this feature we define IoTFlow address as an application layer (L7) header field in IoT packets which holds the MGRS address of the source sensor. The full MGRS address is 15 bytes, thus, IoTFlow address is defined to be 128 bits (15 bytes + a zero padding byte). We send IoT packets on top of UDP over a registered UDP port and use this port to identify IoT traffic. The switch parses the first 128 bits after the UDP header of IoT packets as the IoTFlow address. Fig.5.6 shows the IoT packet header structure on top of UDP.

5.5 Performance Evaluation

Our experiments comprise of micro-benchmarks, where we evaluate throughput of IoTFlow forwarding, as well as macro-benchmarks, where we evaluate SADIQ performance for Weather Signal [13] and Melbourne Smart Parking [12] applications against existing schemes. Our testbed consists of three 1Gbps HP Aruba 2930F switches [105] connected in a tree topology, a single controller that runs on a Dell workstation (8

core, 2.67GHz Intel Xeon processor), an application server running on a Lenovo workstation (24 core, 2.5GHz Intel Xeon processor), and 20 traffic generators running on 4 Shuttle XPCs (dual core, 2.93GHz Intel Core^{TM}).

We select four months of data from both applications, and convert the GPS coordinates of each sample in the dataset into the full MGRS address. The samples are then regenerated as UDP packets using our traffic generator. For both applications, we scale up the data to emulate a larger scale scenario which causes network congestion. For the Melbourne Smart Parking application, we also scale-up the data spatially by 120x by assuming that each single day of the 4 months data was coming from a different city. We split the data across the 20 generators which send the packets following a Poisson process. We vary the average rate of the traffic between 900Mbps-980Mbps. Moreover, to test SADIQ's response to increase in congestion, we increase the load by an extra 5% for 60 seconds in each experiment. The rise in congestion is detected by the increase in the packet drop rate, which is reported by the switches to the controller. We evaluate SADIQ against two other schemes:

- Baseline (No QoS): All packets are treated the same, and go into a single Drop Tail queue.
- Static QoS: The application sets its QoS policy, but it is static and not updated according to changes in the application context, nor tuned according to changes in the network state.

QoS has traditional well-known metrics: bandwidth, delay, jitter, and packet loss. In our evaluation, we focus on the packet loss aspect, however, from an application-level perspective. Instead of comparing the rate of packet loss of different schemes, we rather measure the effect of the same packet loss rate from the application point of view under the different schemes. Therefore, we introduce the following application-specific metrics for the comparison:

- Error in temperature: For Weather Signal, we calculate the absolute error in temperature per region compared with the ideal case of zero packet loss.

$$Error = |IdealTemperature - ReportedTemperature|$$

- Error in parking status accuracy: For Melbourne Smart Parking, any lost packet compared to the ideal case results in an incorrect reporting of a parking space status in that region, i.e., application reports space as vacant when it is occupied or as occupied when it is vacant. However, an incorrect status of a parking space when a region is highly occupied becomes much more critical than when a region has many vacant parking spaces. Thus, we also report the occupancy percentage of the region while reporting the error in parking status, with greater focus on high occupancy regions. We calculate the occupancy percentage of each region r at time t as:

$$Occupancy_{rt} = \frac{\#OccupiedParkingSpaces_{rt}}{\#ParkingSpaces_{r}} * 100\%.$$

We implement two server applications that emulate the processing of both the Weather Signal and the Smart Parking applications. Both application servers receive the IoT packets over UDP. The Weather Signal server calculates the running average temperature of 10 Km^2 regions and updates the QoS controller with a list-based policy that contains an ordered list of mice and elephant regions, and a request to prioritize the mice regions. As for the Smart Parking server, it calculates the real-time parking occupancy percentage of each 100 m^2 and 1 Km^2 regions and updates the QoS controller with a list of regions with highest parking occupancy to prioritize them.

We first evaluate the performance of IoTFlow forwarding in comparison with IP forwarding before presenting the QoS benefits of SADIQ on both the Weather Signal

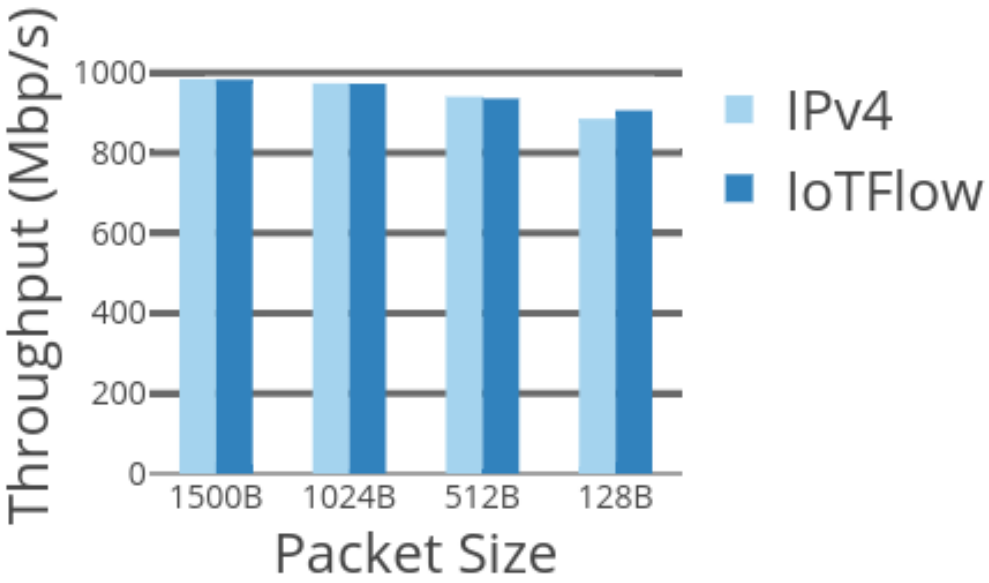


Figure 5.7: IPv4 source address vs IoTFlow address

and Melbourne Smart parking applications. Then, we show how the QoS controller tunes the IoTFlow granularity when faced with different switch resource levels.

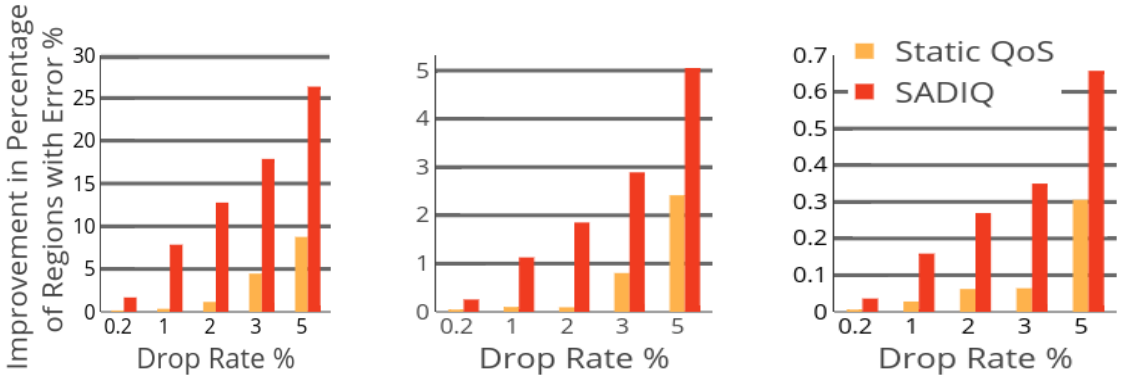
- IoTFlow forwarding performance: We evaluate the performance of forwarding based on the new IoTFlow address and compare it with forwarding based on the IP address. We make the comparison by assessing the throughput as a function of packet size. We fill up the switch with 1400 flow entries (max table capacity) that match on random IPv4 addresses and generate traffic from 4 hosts with IPv4 source addresses randomly selected from the same list. We then repeat the experiments with the same settings for the IoTFow address.

  Fig.5.7 shows the results. As expected, the performance decreases as the packet size decreases. However, the forwarding performance of the IoTFlow address is very similar to the IPv4 source address, which we anticipated since both the IPv4 and the IoTFlow fields are supported in TCAM.

- WeatherSignal QoS: To better reason about the Weather Signal results, we first

Region	Ideal Sample Count	Number of lost Samples	Error in Temperature
47PPR62	46262	1215	0.0048
18STD32	53	4	1.95
59GNP30	2	2	Null

Table 5.1: Packet loss effect on elephants vs. mice regions.



(a) Improvement in the percentage of regions with temperature error >= 0.1 C°.

(b) Improvement in the percentage of regions with temperature error >= 1 C°.

(c) Improvement in the percentage of completely lost regions.

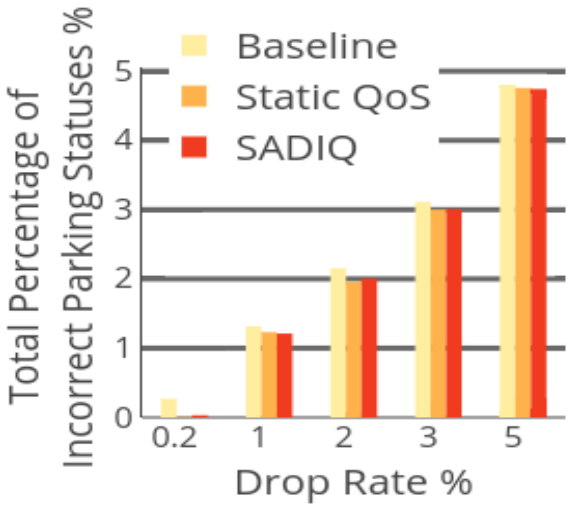
Figure 5.8: Improvement in the percentage of regions with reported temperature error compared with the baseline.

show the impact of packet loss on mice *versus* elephants regions as seen in Table 5.1. Region 47PPR62, a city named Krung Thep in Thailand, is one of the largest elephant regions in our dataset. Due to the high number of samples received from 47PPR62, a loss of 1215 packets resulted in only 0.0048 absolute error in temperature compared with the ideal case. On the other hand, region 18STD32, Pender County in North Carolina, USA is one example of a mice region. Only 4 samples are lost from 18STD32, however it translated into 1.95 absolute error in temperature. Finally, an extreme case is region 59GNP30, an area in Westland, New Zealand, which had only 2 samples and both samples are lost. Such regions have 100% loss in coverage, and no representation of that period in time. For some IoT applications, this type of loss can be very critical and may result in significant loss of functionality for the users.

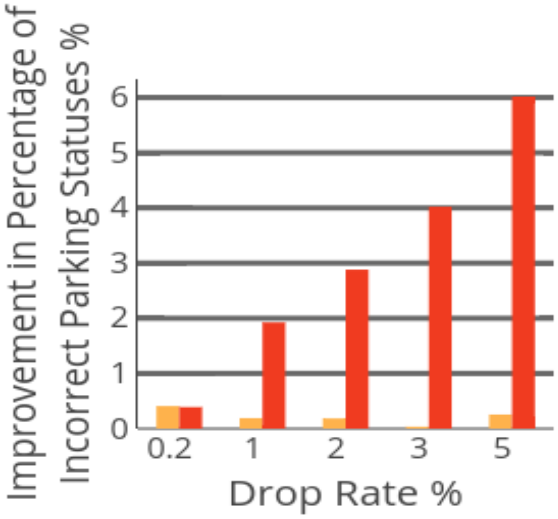
Fig.5.8a compares the improvement in the percentage of regions with temperature error >= 0.1 C° over the baseline and across different drop rates (0.2% to 5%). The Static QoS method slightly decreases the percentage of those regions at a maximum of 8.7% less than the baseline when the drop rate reached 5%. However, SADIQ shows more reduction in the percentage of regions with error which increases significantly as the drop rate increases, reaching 26.3% reduction at drop rate of 5%.

We further highlight that SADIQ not only reduces the number of regions with error but also the degree of error by showing the improvement in the percentage of regions with error >= 1 C° (Fig.5.8b). SADIQ reduces the percentage of regions with error >= 1 C° from 5.1% to only 0.06%, while the Static QoS method is only able to reduce it to 2.7%. Moreover, Fig.5.8c shows the improvement in the percentage of completely lost regions. SADIQ has zero lost regions until 3% drop rate and it reaches a maximum of only 0.006% at 5% drop rate with 0.6% fewer completely lost regions than the baseline.

- Melbourne Smart Parking QoS: We calculate the number of incorrect parking statuses of each region at each occupancy level. Fig.5.9a shows that the total percentage of incorrect parking statuses across all regions using the three methods is almost the same. However, Fig.5.9b shows the improvement of the percentage of incorrect parking statuses across regions when the occupancy of those regions is greater than 60%. Static QoS has 0.2% incorrect parking statuses less than the baseline, while SADIQ has 6.0% less, reducing it from 6.2% to only 0.2% which is around a 30x improvement. This shows that although SADIQ is under the same network and packet loss conditions, it gives priority to the more important packets from an application perspective. Moreover, the reason that the static QoS method does not show as much improvement over baseline as in the Weather Signal scenario is due to the more frequent changes



(a) Total percentage of incorrect parking statuses.



(b) Improvement when occupancy >= 60%.

Figure 5.9: Total percentage of incorrect parking statuses and improvement in the percentage of incorrect parking statuses across regions when the parking occupancy >= 60%.

in the parking occupancy which make the dynamic update of the QoS policies more crucial.

- Tuning IoTFlow granularity: The QoS controller selects the suitable IoTFlow granularity level based on the switch flow entry capacity to maintain the highest QoS possible for the application. To show that, we vary the flow entry capacity of the edge switch connected to the application server (which suffers from the highest congestion) and run SADIQ with and without the location aggregation feature under the same traffic load (980Mbps). Without location aggregation, flow entries of the high occupancy regions are sent with the fine-grained granularity (in this case 1 Km^2) and if the flow capacity of the switch is lower than the number of flow entries, then only the top ones are inserted and the rest are ignored. However, using the location aggregation feature in SADIQ, the QoS controller selects a coarser-grained granularity (10 Km^2) for some regions and groups them into a single flow entry. The controller tries to keep the majority of

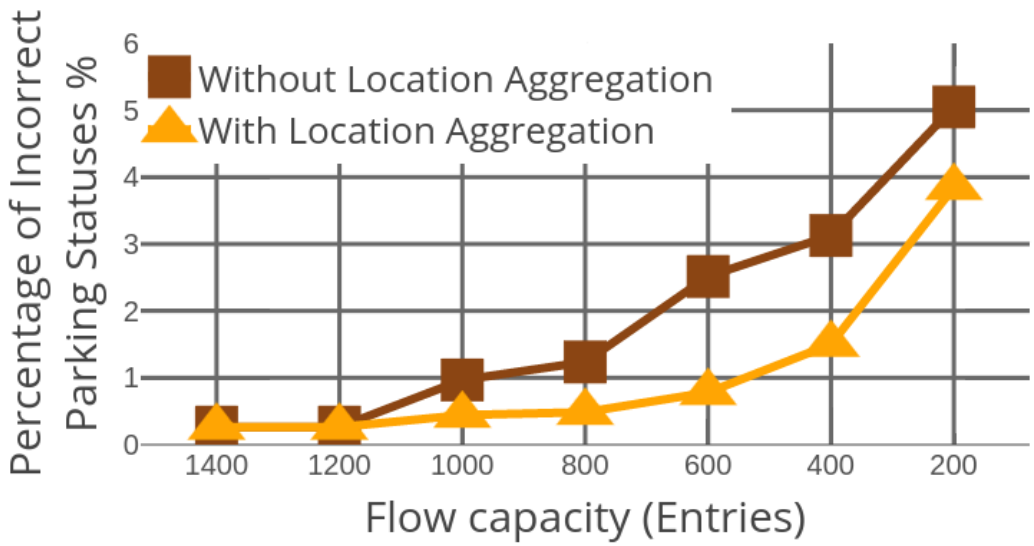


Figure 5.10: Effect of location aggregation on the percentage of incorrect parking statuses across regions when occupancy >= 60% over different switch flow entry capacities.

flow entries in the fine-grained granularity level and only performs aggregation into a coarser-grained level when it is absolutely required.

Fig.5.10 shows the percentage of incorrect parking statues when the occupancy >= 60% across different switch flow entry capacities and compares between running SADIQ with and without location aggregation (note the decreasing scale of the x-axis). At flow capacity of 1400 and 1200, all flow entries with the fine-grained granularity fit in the switch, therefore, there is no location aggregation and both methods perform the same. However, as the flow entry capacity decreases, the QoS controller starts to aggregate locations into a coarser-grained granularity to keep providing high priority to all high occupancy regions. The results show that aggregation yielded between 2-3x improvement over having no aggregation. However, since aggregation also causes some unintended regions to get high priority, it creates a higher competition for the high occupancy regions for the bandwidth. This explains the decrease of improvement to only 1.3x as the flow capacity gets smaller to 200 flow entries.

SADIQ provides a solution to regulate the IoT traffic and improve the system performance. Besides QoS and traffic management, prevent system failures with minimum performance overheads is another crucial objective to consider. Therefore, we next provide the required analysis for peer-to-peer blockchain systems.

Chapter 6

Blockchain Modeling and Analysis

In this chapter, our work is motivated by the strict delay and availability requirements as well as the advantages provided by applying the blockchain technology in the IoT context. To guarantee a minimum performance for the IoT applications, we analyze the network delay for IoT-blockchain system with different network configurations that serves different IoT applications. Thus, we obtain a mathematical expression to calculate the end-to-end delay with different network configurations such as the number of network hops and replica machines.

Our system is composed of multiple IoT devices with different delay requirements, as shown in Fig.6.1. The data gathered from these devices is sent to the consensus cloud to be verified and appended to the blockchain. In the blockchain setting, the consensus layer contributes the most delay overhead. The current performance of blockchain technology is incompatible with the majority of IoT applications. Latency limitation is a primary cause of this incompatibility and an improvement is needed to overcome this limitation. Thus, in IoT networks Byzantine Fault Tolerant (BFT) family of protocols is used as consensus mechanism. From the moment that an IoT device sends its data to the consensus cloud until the data is confirmed and appended to the blockchain, many network hops and consensus machines (replicas) are used. However, adding more machines to the path or the consensus can increase the delay.

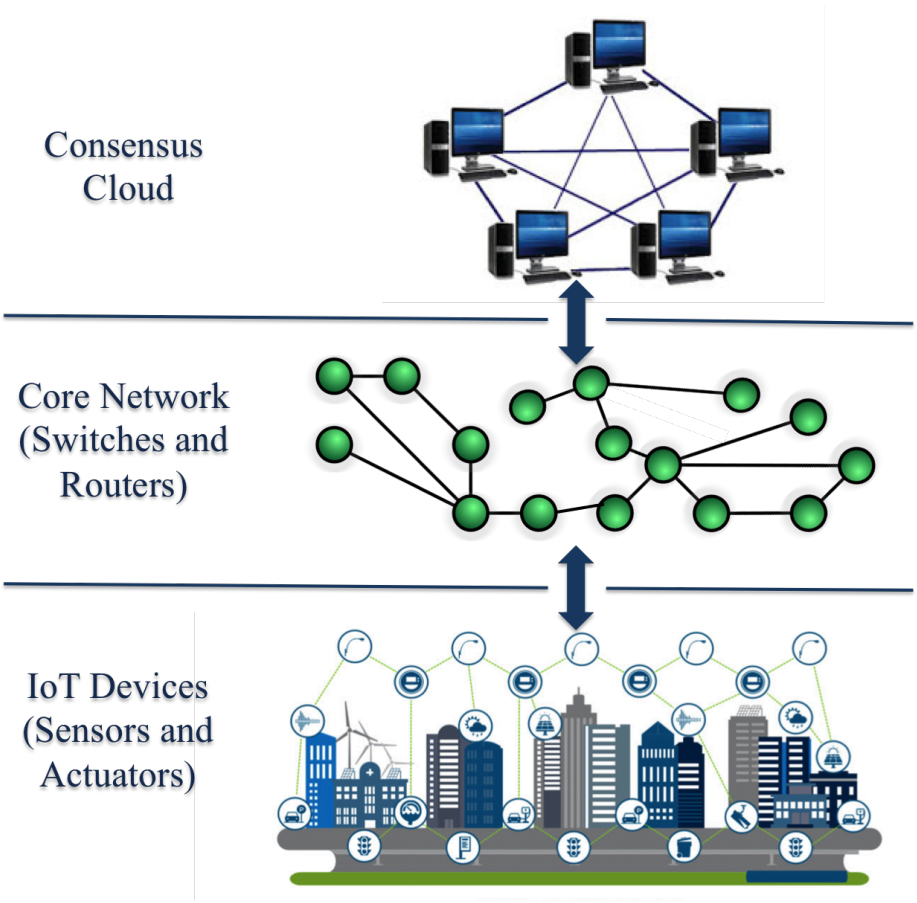


Figure 6.1: General architecture of Byzantine-based IoT blockchain.

6.1 Blockchain-IoT Integration Requirements

To support the IoT applications' strict requirements, we need to revisit the consensus layer for the current blockchain implementations.

- Peer-to-peer distributed ledger: Decentralization of network and verification of blockchain data can create many challenges and it is necessary to address these issues for the development of a robust, strong, useful, and accessible system. More precisely, the blockchain implementations currently process few transactions per second and the estimates show that number can go as high as 1500 transactions per second for Ripple blockchain system [62]. An individual record takes from a few minutes to many days to be confirmed [106], while IoT devices

can generate up to 5 quintillion bytes of data every day (2.5 followed by 18 zeros) [107]. For that, latency becomes an obvious bottleneck for the mainstream adoption of blockchain technology in IoT scenarios.

- Byzantine-based consensus: This family of protocols can provide near instantaneous finality on the inclusion of data into the blockchain. Yet, classic Byzantine-based protocols have high networking communication costs. For example, the legacy PBFT incurs at least $O(n^2)$ messages per block [108]. This huge communication overhead is investigated by several successors protocols such as Zyzzyva [109], HotStuff [64], ,and SBFT [110]. However, the IoT network infrastructure need to be further analyzed to provide the needed performance guarantees.

- Permissioned blockchain model: For IoT, permissioned blockchain based on Byzantine consensus is a trustworthy and promising technology that can track a massive number of connected devices. However, permissioned blockchain model is comparatively more secure than other blockchain models because the identity verification is asked when a device joins the network. After verification, the new device is trusted and allowed to contribute information to the collective. In case of performance, permissioned blockchain network is more performant than the permissionless blockchain framework. In permissioned blockchain network, each node is dealing with a single application and it is required to perform the computation for that application. Thus, permissioned blockchain network keeps a balance between security, scalability, and performance. For the balance reason, permissioned blockchain model can be considered as the future of IoT infrastructure where security and trust are two key requirements of an IoT device to report the critical metrics.

6.2 Byzantine-Based Blockchain System Model

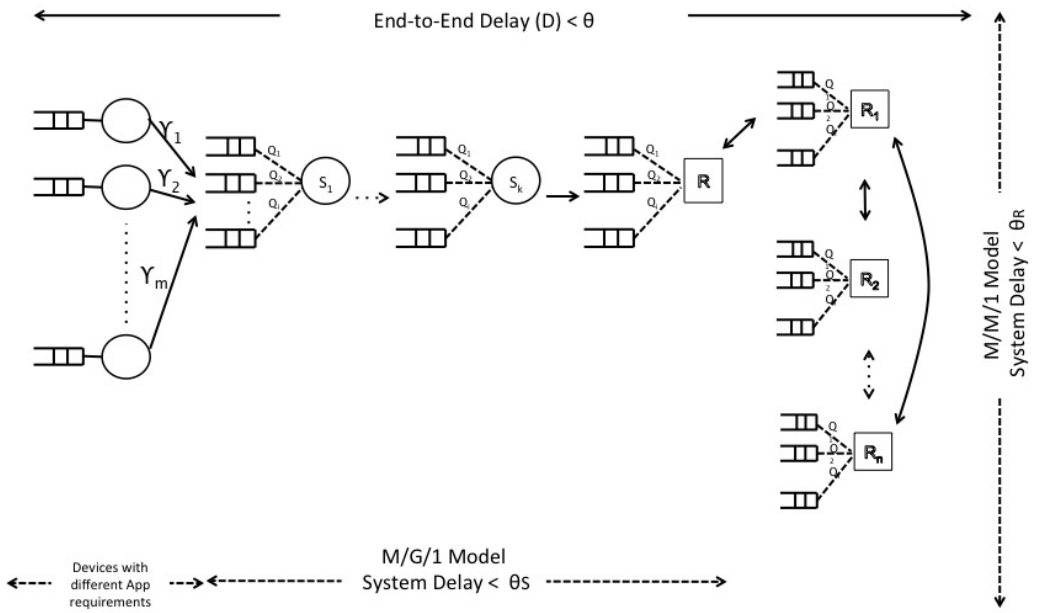


Figure 6.2: An overview of Byzantine-based blockchain system.

Fig.6.2 shows a high-level architectural view of our system. A blockchain system with a Byzantine fault tolerance consensus protocol (BFT) is modeled. The data is collected from M IoT devices which belong to different application's use cases with different requirements. Then, the data is broadcasted to the consensus sever (the leader) over K intermediate switches. Upon receivingend-user the data at the consensus cloud, the replicas communicate with each other to reach a consensus on the same data block (set of data records). The switches and the replicas are considered to be always running [86].

In the first part of the model, a single server facility with M/G/1 queue is considered where a maximum number of hops needs to be calculated to satisfy the application requirements:

- An exponential inter-arrival times are used to model the IoT data packet arrival, with rates $\gamma_m, m = 1, 2, ..., M$.

- The intermediate switches and the sink server are modeled as single server facilities with M/G/1 queue and multiple priority queues Q_p with a priority class $p \in \{1, 2, 3, ...\}$, where $p = 1$ is considered as the highest priority.

- The intermediate switches and the sink server are assumed to operate following a general distributed service time of mean $\overline{X}_{S_i}$ for node $i \in \{1, 2, \ldots, K\}$, where each type of IoT traffic requires different service times.

The second part of the model concerns about the number of consensus replicas R that needed to maintain the end-to-end requirements:

- The internal communications between the replicas are modeled as single server facilities with M/M/1 queue.

- The replicas are operated with an exponential service time distribution of mean $\overline{X}_{R_j}$ for a replica node $j \in \{1, 2, \ldots, N\}$.

6.2.1 Delay Threshold Analysis

In order to maintain a guaranteed behavior for the IoT traffic, the expected delay to reach a consensus confirmation stage must not exceed a predefined delay threshold Y [111]. The predefined threshold can be seen as a part of a service level agreement (SLA) contract between the service provider and the end-user [112]. Our predefined threshold Y has two main components a threshold value for a data packets to reach the sink node (consensus leader) Y_S, end-to-end and a threshold value to finish the consensus confirmation stage Y_R. Therefore, a data packet with a cumulative delay exceeding Y is treated by an appropriate decision. Thus, we define the total end-to-end delay as a random variable called D, then the probability for a data packet to violate delay requirement is given by,

$$P_{\text{violating}} = \Pr\{D > (\mathrm{Y_S} + \mathrm{Y_R})\} \tag{6.1}$$

In such circumstances, our main intent is to calculate the acceptable hop-counts K^* and the number of the consensus replicas N^* such that the end-to-end delay $\mathbb{E}[D] < (Y_S + Y_R)$. Analyzing the application requirements, especially the delay, help us to capture many of the inherent tradeoffs between the underlying network design and the reliability of the current blockchain system from one hand and the predefined requirements of the IoT applications at the other hand.

6.2.2 Network Hops Analysis

The network has a set of K switches and each switch has its own service rate distribution, and each switch is accommodated with multiple priority queues. The packet arrivals are assumed to be Poisson distributed, Poisson traffic is used to obtain closed-form results. Thus, we define the total arrival as,

$$\lambda_i = \sum_{m=1}^{M} \gamma_m \tag{6.2}$$

Each data packet belongs to a different priority class $p \in \{1, 2, 3, ...\}$ that is experienced by sampling the packet at each relay node. The utilization can be expressed as the fraction of time an intermediate node S is serving a class p data packet, $\rho_{S_p} = \lambda_p \cdot \bar{X}_{S_p}$. The waiting time of a high priority packet, denoted as W_{S_1}, is

$$W_{S_1} = \sum_{j=1}^{L_1} X_{S_j} + Z, \tag{6.3}$$

where L_1 is the number of packets belongs to the high priority class, X_{S_j} is the service time of packet j at node S, and Z is the residual time, the first moment of this residual time is

$$\overline{Z} = \frac{1}{2}\left(\sum_{i=1}^{p} \rho_{S_i} \cdot \frac{\overline{X_S^2}}{\overline{X_S}}\right) \tag{6.4}$$

By Little's formula, the high priority packet average waiting time, denoted as $\overline{W}_{S_1}$, is

$$\overline{W}_{S_1} = \frac{\overline{Z}}{(1-\rho_{S_1})}, \tag{6.5}$$

And, the high priority traffic has a second moment of waiting time as,

$$\begin{aligned}\overline{W_{S_1}^2} &= \overline{L}_1 \cdot \mathrm{Var(X_S)} + \mathrm{Var(Z)} + \mathrm{Var(L_1)} \cdot \overline{\mathrm{X_S}}^2 \\ &\approx \overline{L}_1 \cdot \mathrm{Var(X_S)} + \mathrm{Var(Z)}, \end{aligned} \tag{6.6}$$

where

$$\begin{aligned} \overline{L}_1 &= \lambda_1 \, \overline{W}_{S_1} \\ \rho_{S_1} &= \lambda_1 \cdot \overline{X_S} \\ \mathrm{Var}(L_1) &= \overline{L_1^2} - \overline{L_1}^2 \\ \mathrm{Var}(X_S) &= \overline{X_S^2} - \overline{X_S}^2 \\ \mathrm{Var}(Z) &= \overline{Z^2} - \overline{Z}^2 \end{aligned}$$

From this, we understand that the waiting time for high priority packets is primarily due to residual time only and high priority queues contain at most one packet almost all the time. However, the waiting time of the lower priority packets can be calculated as,

$$W_{S_2} = \sum_{j=1}^{L_1} X_{S_j} + \sum_{j=1}^{L_2} X_{S_j} + \sum_{j=1}^{W_{S_2} \cdot \lambda_1} X_{S_j} + Z \tag{6.7}$$

As per equation (6.7), a low priority packet has a waiting time that can be expressed as the summation the service residual time and the time that needed to serve existing high priority packets, new high priority packets, and existing low priority packets that are ahead in the queue. Thus, a lower priority packets have an average

waiting time, denoted by $\overline{W}_{S_2}$ as,

$$\overline{W}_{S_2} = \frac{\overline{Z}}{(1-\rho_{S_1})(1-\rho_{S_1}-\rho_{S_2})}, \tag{6.8}$$

And the second moment of the average low priority packet waiting time is,

$$\begin{aligned} \overline{W_{S_2}^2} &= \overline{Z^2} + s \cdot \mathrm{Var}(X_{S_j}) \\ &+ s^2 \cdot \overline{X}_{S_j}^2 + 2\, s\, \overline{X}_{S_j} \cdot \overline{Z} + q \cdot \overline{X}_{S_j}^2 \\ &\approx \overline{Z^2} + s \cdot \mathrm{Var}(X_{S_j}) \\ &+ s^2 \cdot \overline{X}_{S_j}^2 + 2\, s\, \overline{X}_{S_j} \cdot \overline{Z} \end{aligned} \tag{6.9}$$

where the coefficient s is defined as

$$s = \overline{N}_2 + \overline{N}_1 + \lambda_1 \cdot \overline{W}_{S_2}$$

Here, q is the variance of the number of packets, and the last approximation follows when it is assumed to be negligible in a steady state.

In general, for a class p packets, the mean waiting time can be calculated using the same preceding approach as,

$$\overline{W}_{S_p} = \frac{\overline{Z}}{(1-\rho_{S_1}-\rho_{S_2}-....\rho_{S_{p\_1}})(1-\rho_{S_1}-\rho_{S_2}-.....\rho_{S_p})} \tag{6.10}$$

Where the total time a packet of priority class p_i spends in the system is,

$$\overline{T_{S_p}} = \overline{W}_{S_p} + X_{S_p} \tag{6.11}$$

Finally, the total number of packets in the system is given as,

$$L_p = \lambda_p \overline{T_{S_p}} \tag{6.12}$$

To this point, our objective is to obtain the maximum hop-count that respects the delay constraint,

$$K^* = \arg\max_K \left\{ \sum_{i=1}^{K} \overline{T}_{S_i} \leq Y_S \right\}, \tag{6.13}$$

where

$$\overline{T}_{S_i} = \overline{W}_{S_{p,i}} + \overline{X}_{S_i} + \tau_i, \tag{6.14}$$

where $\overline{W}_{S_{p,i}}$ is given by (6.10) for each $i = 1, 2, \cdots K$, $\overline{X}_{S_i}$ is node i service time, and τ_i is transmission delay between two consecutive nodes, node $i-1$ and node i.

Some exceptions may apply when all the nodes are indistinguishable, K^* can be simply calculated as:

$$K^* \approx \frac{Y_S}{\overline{T}_S}, \tag{6.15}$$

6.2.3 Consensus Replicas Analysis

In our model, IoT devices across the network create and send the data packets to a leader machine at the consensus cloud that runs the PBFT consensus protocol [2]. More details about PBFT were given in Chapter 2.

We have N replicas which are modeled as M/M/1 queue with exponential inter-arrival times with mean $1/\lambda_j$, and exponential service times with mean $1/\mu_j$. The data is served in order of arrival and the traffic intensity is calculated as,

$$\rho_{R_j} = \frac{\lambda_j}{\mu_j} \tag{6.16}$$

Where j is the index that represents the replicas $j \in \{1, 2, \ldots, N\}$. The total time the traffic spends on a replica is given by,

$$\overline{T}_{R_j} = \frac{1}{\mu_j(1 - \rho_{R_j})} \tag{6.17}$$

And we can derive the expression for the mean number of packets in the system

by applying Little's law,

$$L_{R_j} = \lambda \overline{T}_{R_j} \tag{6.18}$$

The average waiting time can be calculated by subtracting the mean service time from (6.17) as,

$$\overline{W}_{R_j} = \overline{T}_{R_j} - 1/\mu \tag{6.19}$$

Or by applying Little's law, this yields the follows,

$$\overline{W}_{R_j} = \frac{\rho_{R_j}/\mu}{1 - \rho_{R_j}} \tag{6.20}$$

Our model considers a legacy PBFT protocol where all the nodes communicate with each other. For simplifying, each node is assumed to process all the received packets, including its own. Therefore, for each data request that enters the system, a replica needs to process $2N$ PBFT packets (PBFT packets are the number of prepare (including the leader's preprepare) and commit messages), where N is the number of the replica machines, and all PBFT packets have the same service time. In order to obtain the allowable number of the replica machines that meet our defined delay threshold,

$$N^* = \arg\max_{j \in N} \left\{ 2j\overline{T}_{R_j} \leq Y_R \right\}, \tag{6.21}$$

The number of replicas machines is given by,

$$N^* \approx \frac{Y_R}{2j\overline{T}_R}, \tag{6.22}$$

In another word, the total delay D experienced by the system should be,

$$D \leq \frac{Y}{K^* + N^*} \tag{6.23}$$

Or alternatively as,

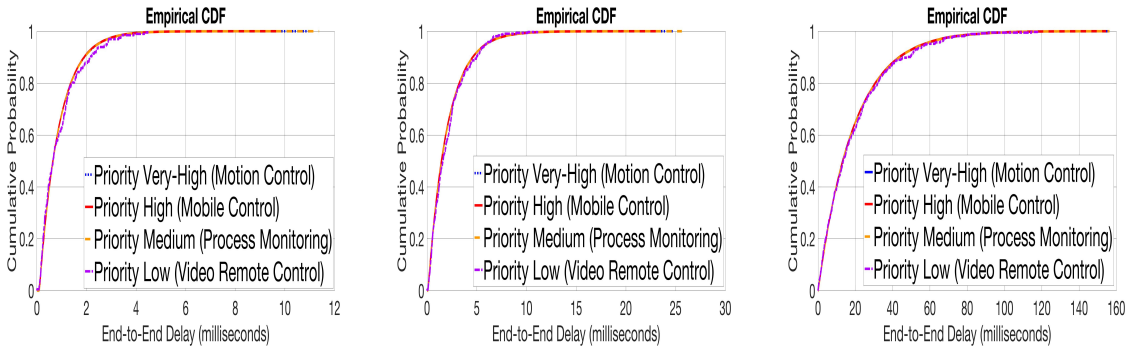
$$D \leq \sum_{i=1}^{K} \overline{T}_{S_i} + 2j\overline{T}_R \tag{6.24}$$

6.3 Performance Evaluation

In this section, the performance of the above framework is evaluated using MATLAB/Simulink package. We are evaluating an abstract model and the details of the underlying infrastructure and technologies are ignored. The general IoT network topology for conducting our experiments is shown in Fig.6.2.

An extensive evaluation of our model is done, using two IoT applications, smart factory and smart city. Smart factory or Industry 4.0 can be considered as delay-sensitive IoT application where strict latency requirements should be fulfilled. Four smart factory's use cases are considered in our experiments. As shown in Table 1.1 [5], the delay requirement varies between 0.5 milliseconds to 100 milliseconds. By considering these requirements, we define the end-to-end delay threshold Y to be 2, 12, 50, and 100 milliseconds for the motion control, mobile control, process monitoring, and video control use cases, respectively. Also, we consider average size factories [113], 40,000, 65,000, and 100,000 sq. ft where the density of the IoT devices is given in Table 1.1 [5]. However, the smart city's use cases are examples of massive IoT. In such application, we relax our delay threshold Y to be 100 seconds [6] and any packet arrives after this time is useless and action needs to be taken. Devices' density per Km^2 and detailed packet arrival rates are given in Table 1.3 [6]. In our experiments, three cities scenarios with 300, 600, and 900 Km^2 are evaluated; these values are obtained according to average sized cities in the USA [114]. In our evaluation we focus on the end-to-end delay, two metrics are used for this assessment:

- CDF expresses the probability that if a defined variable takes a value less than or equal to x [115]. Our results show CDF for the end-to-end delay D.



(a) 40,000 sq.ft (b) 65,000 sq.ft (c) 100,000 sq.ft

Figure 6.3: End-to-end delay CDF for average size factories.

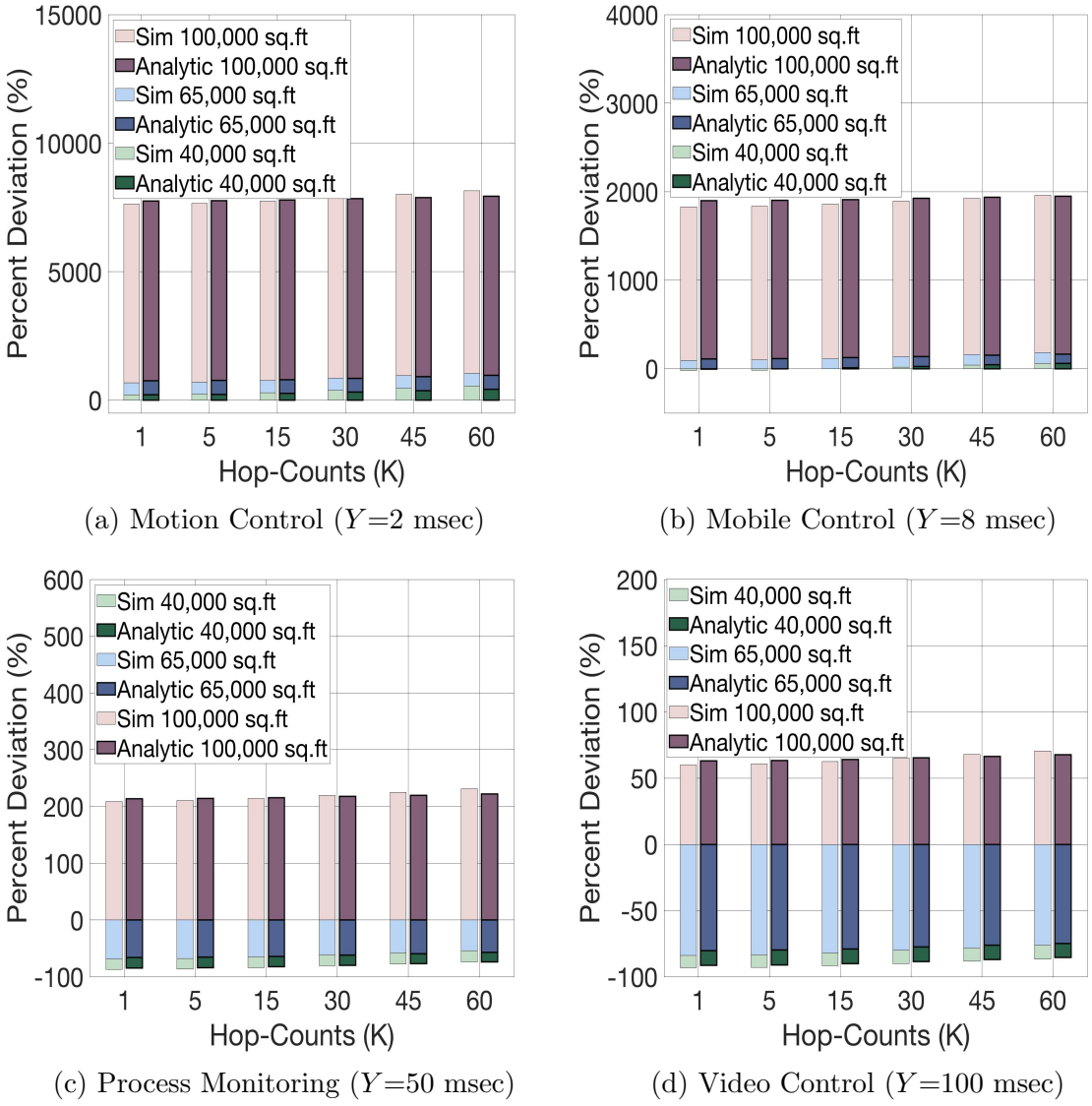
$$F(x) = Pr[D \leq x] = \alpha$$

- Percent Deviation is defined as the difference between the mean of a set of data from a theoretical or predefined value. This can be useful in our experiments to show the delay deviation from the applications' defined metrics. We study the percent deviation from the delay threshold Y:

$$PercentDeviation = \frac{D - Y}{Y} * 100\%$$

The negative value signifies that the given mean of the delay is lower than the threshold Y. If the percent deviation is positive, it signifies that the delay mean value is higher than expected. As described above, we want to study the direct effect of the network hop-counts and the number of replica machines involved in the consensus protocol on the delay sensitive and massive IoT applications.

- Smart Industry (Industry 4.0): End-to-end delay CDF for different smart factories are shown in Fig.6.3. In 6.3a with 40,000 sq. ft, 90% of the packets



(a) Motion Control (Y=2 msec)

(b) Mobile Control (Y=8 msec)

(c) Process Monitoring (Y=50 msec)

(d) Video Control (Y=100 msec)

Figure 6.4: Percent deviation from the average delay for different hop-counts (K) and four replica machines (N=4).

encounter a delay of less than 2 milliseconds. On the other hand, 90% of the packets have a delay of less than 5 milliseconds for the 65,000 sq. ft, as shown in Fig.6.3b. While at 100,000 sq. ft, in Fig.6.3c, the four uses cases incur a delay of less than 40 milliseconds for 90% of the traffic.

The average end-to-end delay percent deviation results for multiple hops and four replica machines are shown in Fig.6.4. As shown in 6.4a, it is hard to meet the requirement of the motion control use case where the delay should be less

than Y of 2 milliseconds. With less devices' density in the case of 40,000 sq. ft, the average delay for different hop-counts is between $[+215\%, +549\%]$ percent deviation from the Y of 2 milliseconds. As presented, when the devices' density for both 65,000 and 100,000 sq. ft increase the percent deviation is escalated up to the range of $[+6.7e+2\%, +1.04e+3\%]$ and $[+7.62e+3\%, +8.14e+3\%]$.

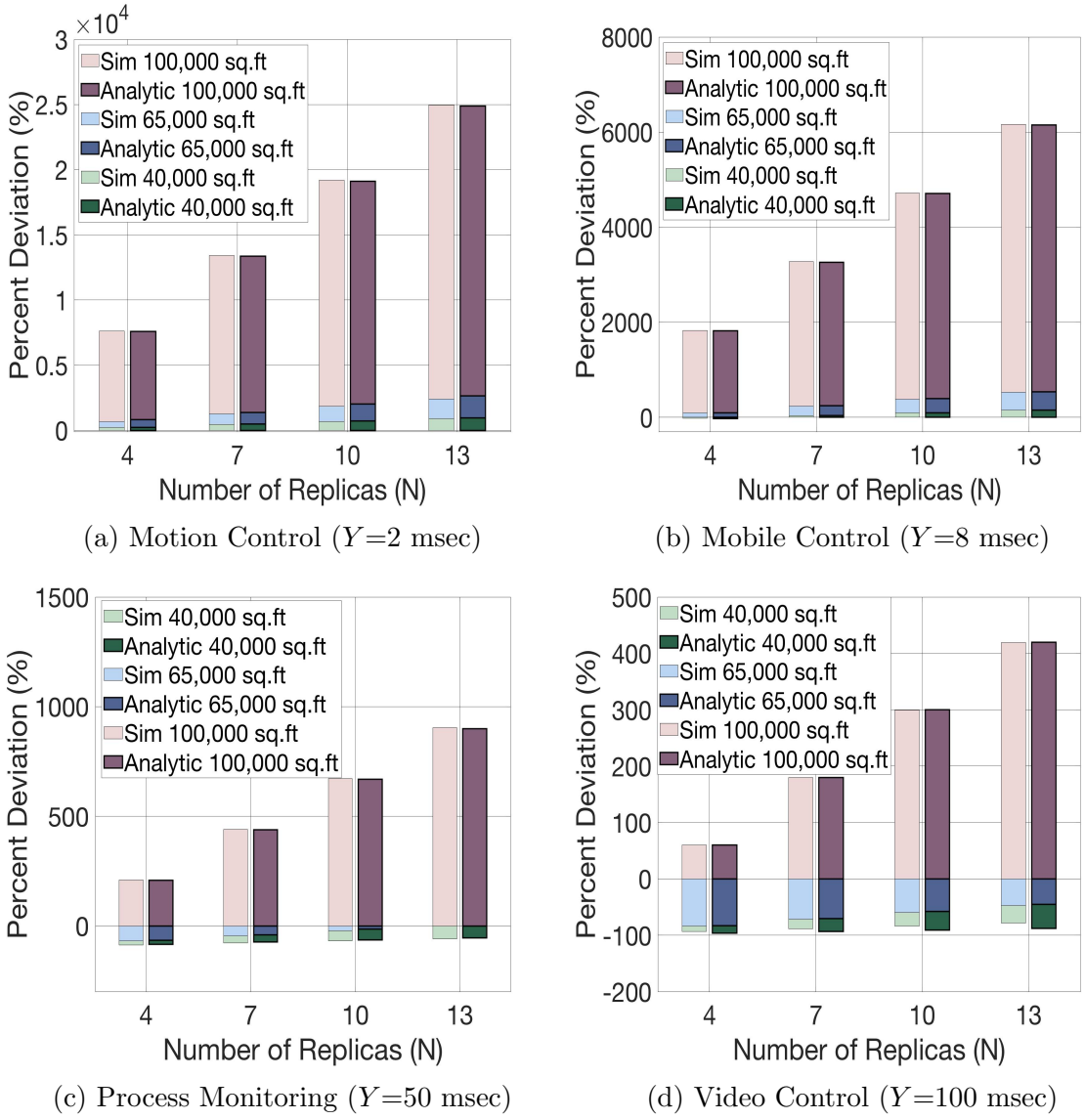
Mobile control yields better delay performance, Fig.6.4b, Y of 8 milliseconds is fulfilled for 40,000 sq. ft with maximum hop-counts of 15. It is impossible for the devices at 65,000 sq. ft and 100,000 sq. ft to meet their application requirements.

In the third use case in Fig.6.4c, process monitoring meets its predefined Y of 50 milliseconds for both 40,000 and 65,000 sq. ft with different hop-counts. For the 100,000 sq. ft, a percent deviation of $[+209\%, +230.8\%]$ is come across different hop-count configurations.

The video remote control with Y=100 milliseconds has the least restrictive delay requirement. This requirement is satisfied for both 40,000 and 65,000 sq. ft with different hop-counts, however, percent deviation of $[+59.8\%, +70.5\%]$ is obtained for the 100,000 sq. over a different number of network hops as illustrated in Fig.6.4d.

In the second experiment, we vary the number of replicas ($N = 3f_b$+1) from 4 ,7, 10, to 13 to tolerate 1, 2, 3, and 4 Byzantine faults f_b as shown in Fig.6.5 and we obtain the results for each use case. Fig.6.5a shows that it is impossible to meet the strict requirement when Y=2 even with the different network configurations. In Fig.6.5b, we can only meet the mobile control (Y=8) requirement for the area of 40,000 sq. ft with 4 replicas.

However, the different replica configurations, in Fig.6.5c and Fig.6.5d, can meet the process monitoring (Y=50) and the video control (Y=100)requirements for



(a) Motion Control (Y=2 msec)

(b) Mobile Control (Y=8 msec)

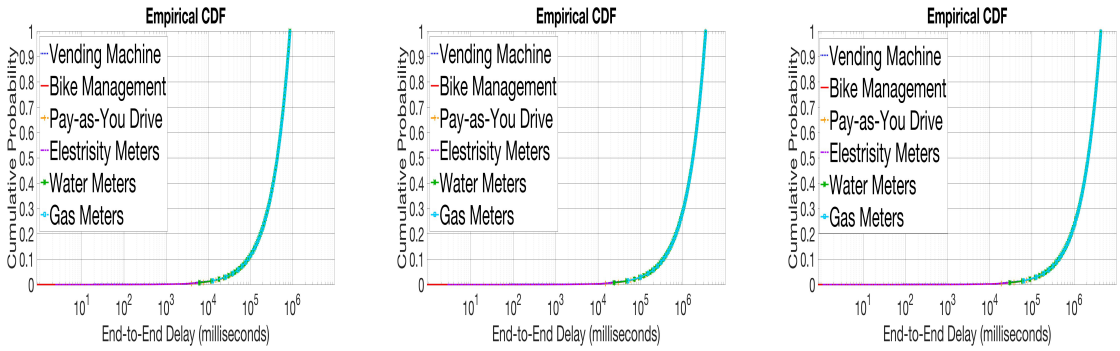
(c) Process Monitoring (Y=50 msec)

(d) Video Control (Y=100 msec)

Figure 6.5: Percent deviation from the average delay for different number of replica machines (N) and one-hop (K=1).

a surface area of 40,000 and 65,000 sq. ft but not for the 100,000 sq. ft.

- Smart cities: In a smart city, the huge amount of participated devices is one of the unique characteristics of IoT systems that negatively impact the usability of the currently deployed applications. For this, we extend our evaluation and test a smart city scenario with 6 different use cases. The end-to-end latency constraint Y is relaxed to be 100 seconds [6]. First of all, CDF results in Fig.6.6



(a) 300 Km^2 (b) 600 Km^2 (c) 900 Km^2

Figure 6.6: End-to-end delay CDF for average size cities.

are obtained. As shown, 90% of the packets for all the cases have a delay less than $10E6$ milliseconds at an area of 300 Km^2. For 600 Km^2, only 30% of the packets have the $10E6$ milliseconds delay. However in 900 Km^2, 20% of the packets have the $10E6$ milliseconds delay. As known the congestion occurs when the number of packets being transmitted through the network approaches the packet handling capacity of the network and the network delay increases with the congestion. In this scenario, the average delay is always higher than the predefined requirement (100 second).

Our main intent here is to conclude that the current BFT-base blockchain can not handle the huge amount of data generated by the different IoT applications. As a result we next propose a scalable distributed ledger that satisfies the required performance and fault tolerance requirements.

Chapter 7

Fault-Tolerant Scalable Networking

In Chapter 5, we analyzed the Byzantine-based blockchain in the IoT context and concluded that it is hard to meet the required performance guarantee with the current blockchain implementations. A well-established distributed Byzantine-based blockchain should be proposed. In this chapter, we propose Synopsis, a two-level hierarchical blockchain system that uses two types of consensus mechanisms and two types of data structures. In the first hierarchy level, Byzantine faults are mitigated through a chain replication protocol (SynopsisCR) which is a wireless optimized BFT protocol that addresses IoT performance issues. SynopsisCR takes advantage of the shared domain nature of wireless communications to accelerate the consensus process. We assume the network is composed of IoT nodes communicating *via* wireless radio with the goal of reaching an agreement in as few transmissions as possible. SynopsisCR consensus is responsible for constructing a local data chain using two kinds of data structures, i.e., sketch and tiny block. The sketch structure is a probabilistic data structure that provides a more efficient compressed data structure for IoT constrained devices. Conventional tiny blocks are used to overcome the probabilistic limitations of the sketch structure. In the second hierarchy level, a non-Byzantine consensus protocol is used to optimize communication latency by performing global blockchain commitment without overwhelming the local IoT network. The global blockchain comprises of the main blocks and is built by consolidating the tiny blocks using non-Byzantine consensus.

7.1 Synopsis: a Scalable Byzantine Distributed Ledgers

We propose a novel hierarchical peer-to-peer framework, Synopsis, that builds on blockchain technology. Synopsis is a lightweight framework providing a direct connection between devices where a small code footprint is required. It provides the required balance of performance, scalability, consistency, and partition-tolerance.

To construct Synopsis, firstly, we replaced the block-like structure at the first hierarchy level with a time adaptive Count Min Sketch (ada-CMS). The proposed data structure is a probabilistic data structure that can scale efficiently while maintaining the total error bounded. Consequently, each group of IoT devices forms a local ada-CMS chain; this is called Synopsis.

Secondly, our consensus scheme adopts locality and hierarchy in order to improve system performance and reduce the risk of failures. In the first level of the hierarchy, malicious failures are masked using a Byzantine SynopsisCR protocol. SynopsisCR replicas maintain a Tinychain of block-like structures. At the second hierarchy level, the entity data are recorded and the main block-like structures build the main blockchain. Committing to the main blockchain is done by nodes running a non-Byzantine consensus protocol; in this work, we use Paxos. For every action conducted by the replica, it requires the signed approval of nodes in its local group. Obtaining this approval is done *via* the SynopsisCR protocol.

- Synopsis data format: Synopsis uses a compact data structure, ada-CMS. The sketch has a complexity of $O(1)$ in both time and space, providing a truly distributed chain system. At each time interval τ, the ada-CMS is updated to take into account the newly generated IoT data records. A hash pointer is also generated to point to the $\tau-1$ version of the sketch forming the compacted chain, Synopsis. Thus, all constrained devices are able to maintain their Synopsis copy.

- Participants: Every consensus participant knows all other players' public key

and maintains a local log that describes its current state. A local log has 5-tuple entries = $(\tau, h_\tau, p_{h-1}, Synopsis, Tinychain_h)$. τ represents the creation time-stamp of both ada-CMS and the tiny block and h_τ provides the sequence number of the last committed ada-CMS and tiny block, whereas the pointer of the previous ada-CMS and the tiny block is held on p_{h-1}. $Synopsis$ is thus an updated ada-CMS. Moreover, replicas are required to hold the complete ($List_{data}$) in a block-like format, $Tinychain_h$, which spans multiple consecutive time-stamps with sequence number h. The $Tinychain_h$ is required to build the legacy blocks of the main blockchain. Upon committing the $Tinychain_h$ on the main blockchain, the $Tinychain_h$ is flushed and h is reset to reduce storage complexity. The second level participants hold a 4-tuple log = $(\omega, H_\omega, o_{H-1}, blockchain)$, where ω is a time-stamp, H_ω is the sequence number of the last committed main block, and the pointer to the previous block is held at o_{H-1}. The global chain of main blocks is preserved at the *blockchain*.

- Consensus rounds: The consensus mechanism is executed in two rounds. In the first round, a SynopsisCR protocol is conducted to form the local Synopsis and Tinychain. Replicas are topologically organized in a chain and the Byzantine agreement is reached with linear communication complexity. In the second round, a non-Byzantine agreement protocol consolidates the Tinychain into a conventional chain of blocks at the main blockchain.

7.2 Synopsis Data Format

Two kinds of data structures are required: probabilistic and deterministic. The probabilistic structure is used to provide a scalable scheme such that all devices can hold a data copy and support a fully distributed data system. Probabilistic Data Structures (PDSs) [116] are groups of data structures that are vital for processing large amounts of data without long delays in analytical processes. PDSs use hash

functions to efficiently denote a set of data values while providing approximations within error bounds and are consequently much faster and less memory consuming than traditional data structures. They also feature constant-factors in storage and low run-times. Therefore, such an approach is very effective for big data operations, such as fast processing, querying, predictions, storing data values, and reducing time or space trade-offs.

In an IoT environment, many devices are connected to the network and produce a huge volume of data daily. Some of these data items are redundant. As a result, there is a need to extract insights and value from IoT-produced data using solutions that approximate the data frequency. Among the most adaptive and robust techniques with lower computational cost and memory requirement is a sketch data structure called Count Min Sketch (CMS) [116]. The CMS algorithm is a popular method for assessing tallies of items over data streams. It is named after the two fundamental operations required to answer count queries, i.e., calculating counts and taking the minimum. CMS is a data structure consisting of a two-dimensional matrix M of width w and depth d. Each matrix cell is initialized by zero and, at every time interval τ, the frequency of item i is updated with count c_{τ_i} and added to the matrix by calculating the corresponding hash values as $M(j, h_j(i))$, where $j = 1, 2, \ldots, d$. Thus, the minimum of the related d values, i.e., $min_{j\in\{1,2,\ldots,d\}}M(j, h_j(i))$ is the count of item i.

In IoT applications that involve temporal data, recent events are usually the most significant for predictive purposes. However, it is also necessary to preserve older data, especially when frequent up-to-date occurrences of an item are not available. Some errors in item counts in the archive can be tolerated with limited space, although it is advantageous to obtain highly precise counts for freshly observed items. A modified version of CMS is presented in the ada-CMS [117].

Ada-CMS applies a pre-emphasis to artificially enlarge the counts of recent items

relative to older values, i.e., it makes newer values heavier. This pre-emphasis process is conducted by updating ada-CMS with $f(\tau) \times c_{\tau_i}$ instead of c_{τ_i}, where $f(\tau)$ is a function increasing monotonically with time τ. Therefore, the total stream size during $[\tau - 1, \tau]$ is given by $C^{\tau} = \sum_{i \in I} f(\tau) \times c_{\tau_i}$. Subsequently, ada-CMS preserves large values and maintains the accuracy of recent items relative to older items following artificial inflation. To answer an item i count query at time τ, de-emphasis processes are used by dividing results with $f(\tau)$. These processes decrease the errors of recent counts relative to those of older counts. The total estimated counts of an item i, donated by $\hat{c}_i$, are equal or larger than real counts c_i, meaning that false negatives are impossible. The newly estimated value is within a factor of ϵ such as $\hat{c}_i \leqslant c_i + \epsilon C$ and has an accuracy probability of $1 - \delta$, where ϵ and δ are user-specified parameters. Generally, ada-CMS has no additional overheads. If $f(\tau) = 1$, the vanilla version of the CMS algorithm is recovered.

Ada-CMS is used as the building primitive of Synopsis. Each updated ada-CMS at time τ points to a pre-version of ada-CMS at time $\tau - 1$. We also use a deterministic block-like structure, i.e., tiny and main blocks, to overcome the probabilistic factor inherent to the structure of Synopsis. The relationship between Synopsis, tiny blocks, and main blocks is illustrated in Fig.7.1.

- Synopsis: An ada-CMS is a probabilistic data structure that serves as a frequency table of IoT events and is produced by the SynopsisCR nodes. It is a two-dimensional structure with depth d and width w and is accessed *via* d pairwise independent hash functions $h_1, h_2, \ldots, h_d$:$\{1, 2, \ldots\} \rightarrow \{1, 2, \ldots, w\}$. Ada-CMS maps each produced count according to the given independent hash functions, yielding a value in the range of [1,w]. The main objective is to provide a fully distributed chain system in a constrained IoT environment with guaranteed accuracy in terms of the user-specified parameters ϵ and δ. These parameters indicate that the margin of error in answering an IoT query lies

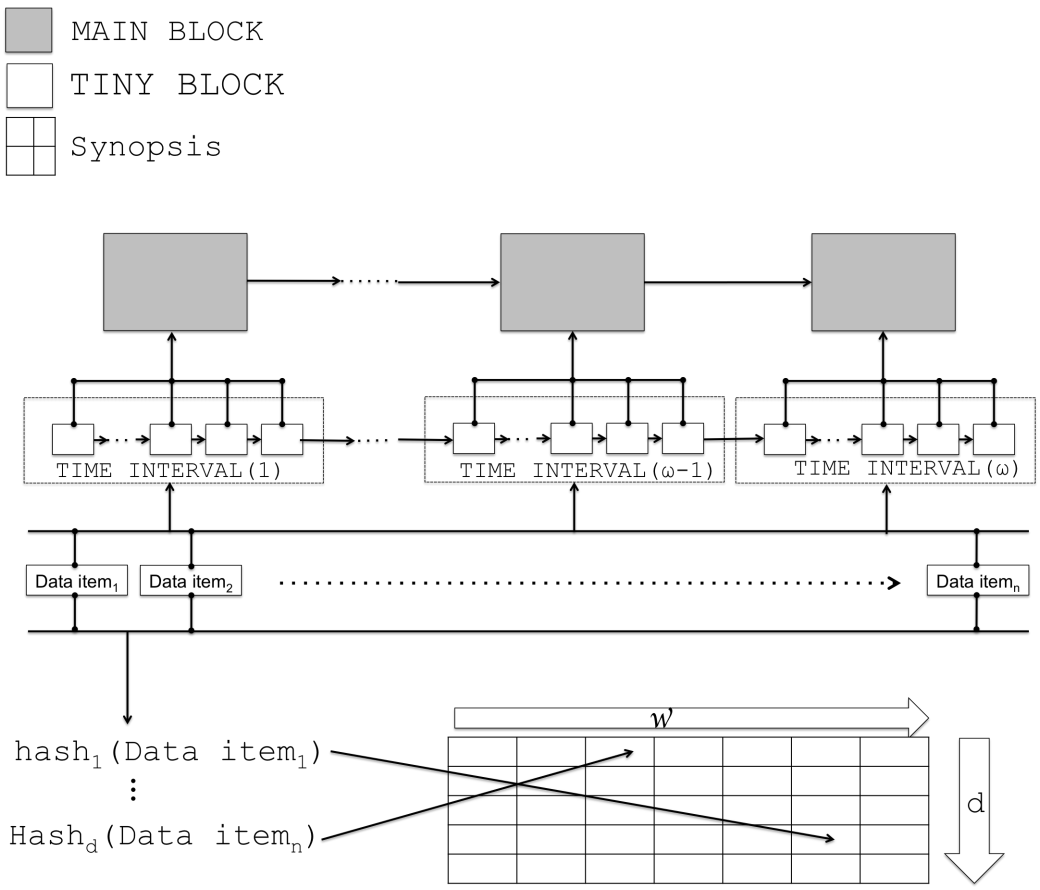


Figure 7.1: Data items over the lifetime of the system create a single Synopsis. The collective of tiny blocks at time interval τ creates the main block.

within a factor of ϵ and has δ probability. Thus, ϵ and δ are the driving factors of space and time in the sketch complexity. Given parameters (ϵ,δ), the SynopsisCR replica sets $w = \lceil \frac{e}{\epsilon} \rceil$ and $d = \lceil \ln \frac{1}{\delta} \rceil$, where e is the base of the natural logarithm function ln.

We assume a streaming IoT data model with the set of data items denoted by $I = \{1, 2, \ldots\}$. During time interval $[\tau - 1, \tau]$, the total count of an item $i \in I$ is denoted by c_{τ_i}. Upon arrival of a new IoT data value, the ada-CMS is modified and the local log is updated. Thus, Synopsis forms from this sequence of updated ada-CMS. The main goal of Synopsis is to make the most efficient use of the data such that errors are minimized while maintaining the flexibility to answer IoT application questions. Synopsis supports the following operations:

- INSERT: When a new data value arrives at time τ, the SynopsisCR replica updates the current Synopsis view. To perform the pre-emphasis and de-emphasis processes, we define a monotonically increasing function $f(\tau); \tau \geq 0$. Monotonicity implies that $f(\tau) \geq f(\grave{\tau}) \ \forall \ \tau > \grave{\tau}$. Therefore, for each pairwise-independent hash function h_j, where $j = 1, 2, \ldots, d$, cell M is updated with:

$$M(j, h_j(i, \tau)) \leftarrow M(j, h_j(i, \tau)) + f(\tau) \times c_{\tau_i}$$

- QUERY: The following applied when returning the total counts of data value i during time τ:

$$c_\tau \leftarrow min_{j \in \{1,2,\ldots,d\}} \frac{M(j, h_j(i))}{f(\tau)}$$

- Tiny block: A tiny block is a data block produced by SynopsisCR consensus replicas every few seconds. Each tiny block includes a hash pointer referring to the previous tiny block in order to ensure total ordering and data integrity. The CR protocol ensures the consistency and authenticity of the tiny block chain. Trust is assured among untrusted parties where at most f_b out of $3f_b+1$ members are malicious. Once multiple tiny blocks are committed, the SynopsisCR leader initiates the main block creation process. Once the main block is appended to the main blockchain, the SynopsisCR replicas choose either to flush the storage space or keep it. This choice depends on the storage capabilities of the replica. Tiny blocks follow the legacy blockchain block format, where blocks are organized into a linear sequence over time. Each block contains, among other things, the time, a record of the data values, and a reference to the immediately preceding block.

- Main block: The global blockchain reflects the main chain of blocks and consists of all committed tiny blocks. Consequently, it follows the same tiny block format. The main block is used to archive data for long-term usage. A non-Byzantine protocol is used to ensure finality among the committed main blocks.

7.3 Synopsis Consensus

The goal of our proposed framework is to maintain a hierarchical blockchain system that satisfies the strict IoT requirements, such as latency, memory, and computational complexity. Consequently, Synopsis tolerates both Byzantine and non-Byzantine failures with low-performance overheads. IoT devices or clients submit sensing data to the consensus nodes, termed SynopsisCR replicas. These replicas accumulate data in a specific order, pack it into tiny blocks, generate the corresponding Synopsis, and initiate the SynopsisCR phases required to eliminate Byzantine faults by committing both tiny blocks and Synopsis itself. When the SynopsisCR leader commits a predefined number of tiny blocks, it initiates the second-level chain, i.e., the main blockchain. The Paxos protocol creates this main blockchain as a means for tolerating non-Byzantine failures.

7.3.1 Problem Assumptions

Our assumptions can be summarized as follows:

- Finality: The Byzantine agreement guarantees finality. Thus, it ensures that clients need to wait only for the next block, rather than the next several blocks, to verify that data have been committed.

- Asynchronous network for safety: Synopsis considers an asynchronous network model to achieve safety. The asynchronous communication model assumes the messages can be dropped or delivered corrupted, delayed, or out of order. And

the safety is the guarantee that no two honest nodes would disagree about the content of an entry in the block.

- Synchronous network for liveness: Synopsis considers a synchronous model to ensure liveness. The synchronous model implies that, a message that is sent at time τ will be received at most at time τ' where $\Delta = \tau' - \tau$. And liveness is the guarantee that progress will be made and a new block will be committed.

- Probabilistic sketch safety: Ada-CMS with limited space guarantees probabilistic accuracy, tolerates bounded errors on item frequency, and obtains highly accurate results for new data items.

- Public Key Infrastructure (PKI): By considering a permissioned model in our proposed framework, we showed that there is a PKI to make authentication among consensus replicas. We also assume that the correctness of PKI primitives is guaranteed.

7.3.2 Problem Definition

We consider a system with N IoT devices and U consensus nodes that can tolerate number of crashes, known as fail-stop failures, denoted by f_c. The system can also tolerate f_b Byzantine failures. U are grouped into K clusters, each of which has a set of nodes P running a SynopsisCR consensus protocol that tolerates f_b failures on each local cluster, where $|P| \geq 3f_b + 1$ and $P = \{p_i$ is a node that runs SynopsisCR, $0 < i < |P|\}$. When a SynopsisCR leader is elected (e.g., due to view-change), it will also take the responsibility to run Paxos. The SynopsisCR leaders set-up a set R, where $|R| = K \geq 2f_c + 1$ and R is defined as $R = \{r_j$ is a SynopsisCR leader that also runs Paxos, $0 < j < |R|\}$. Therefore, we can define all consensus nodes in our system as U and $|U| = |P| \times |R|$. Note that the devices that run both SynopsisCR and Paxos need to be occupied with good storage and communication capabilities.

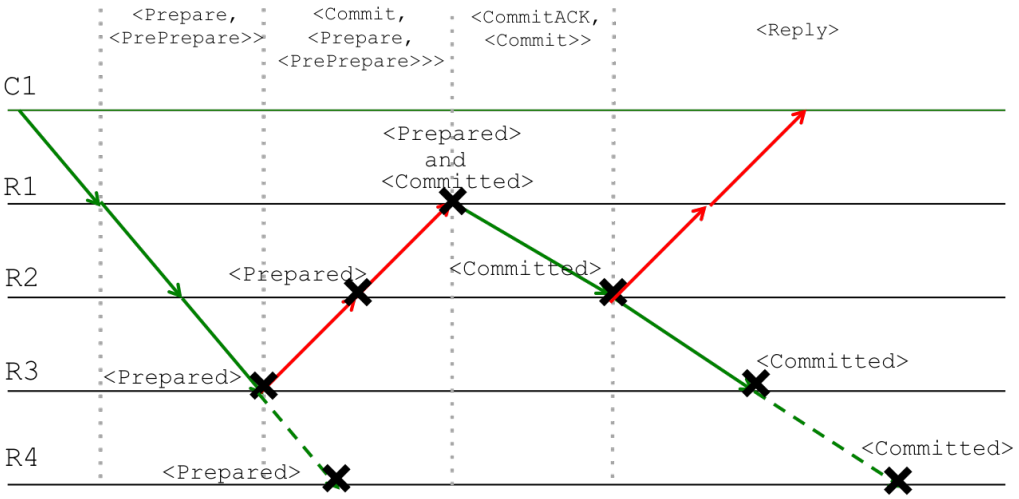


Figure 7.2: Chain replication in normal case operation.

Each node at a SynopsisCR consensus group outputs a series of ordered tiny blocks, $L[1], L[2], \ldots, L[h]$, where h is the height of the local Tinychain after attaching a new tiny block. The group also outputs the compacted ada-CMS, i.e., Synopsis. When the leader commits a predefined number of tiny blocks, L, it begins the second phase during which it commits the second-level of ordered main blocks, $G[1], G[2], \ldots, G[H]$, where H is the length of the main blockchain after attaching a block containing a group of local tiny blocks.

7.3.3 Synopsis Chain Replication Protocol

Chain replication is a protocol for scalable Byzantine agreements, enabling the leader to validate data of Synopsis and the tiny blocks in order to collectively sign it *via* a decentralized group of replicas. To verify each message, the leader then initiates a four-phase protocol with a linear message complexity of $O(|P|)$. Fig.1 and Algorithm 1 illustrate the normal case operation of the SynopsisCR protocol.

- PrePrepare: An IoT client c requests the execution of state machine operation o by sending a request, $m = <\mathsf{REQUEST}, \mathsf{o}, \tau>_c$, to the assigned replica, where τ is the time-stamp. At each time interval τ, the potential leader is required to

accumulate the received data into a data request M. M may includes a Synopsis and tiny block (L) or a Paxos action. The leader then initiates the SynopsisCR protocol by generating a <PrePrepare> message. The <PrePrepare,v,n,M>$_{\sigma_\chi}$ message is sent to the leader successor, where v is the view number, n is the message sequence number that took place during view v, and σ_χ is the leader signature.

- Prepare: A replica receives a valid <PrePrepare,v,n,M>$_{\sigma_\chi}$ message from the leader; this message contains a valid leader signature. The replica p_i validates the message, creates a <Prepare,<PrePrepare,v,n,M>>$_{\sigma_\beta}$ message that encapsulates the leader <PrePrepare> message, appends aggregated signature σ_β, and sends this to its successor. The <Prepare> message is sent continuously until is reaches the last replica in the path.

- Commit: When replica accepts $2f_b$ <Prepare> messages in addition to the <PrePrepare> sent by the leader, the node enters the prepared phase. It computes the signature, creates a <Commit,<Prepare,<PrePrepare,v,n>>$_{\sigma_\beta}$ message, and sends this to its predecessor.

- Commit acknowledge: Upon replica receiving a $2f_b + 1$ <Commit> messages, the replica enters the committed phase. Subsequently, each replica sends a <CommitACK,< Commit,v,n>>$_{\sigma_\beta}$ message to its successor to ensure that all $3f_b + 1$ replicas have committed the message. The <CommitACK> message is sent continuously until is reaches the last replica in the path.

- Reply: The replica that receives $f_b + 1$ <CommitACK> message, is required to send a <Reply> message to the leader by passing the <Reply> to its predecessor. The IoT client waits for a <Reply,Synopsis,τ >$_{\sigma_A}$ message which is collectively signed by group A, where $|A| = f_b + 1$ replicas, before it considers the data

committed. And the Paxos node waits for the signed approval from the local group.

Algorithm 1 Synopsis Chain Replication Protocol (SynopsisCR)

1: $M \leftarrow$ *(TinyBlock(L) and Synopsis)* $\vee$ *(PaxosAction)*

upon reception of request M **do**
2: $v \leftarrow$ *view*
3: $n \leftarrow$ *sequence number*
4: $\sigma_\chi \leftarrow$ *leader signature*
5: send to successor $< PrePrepare, v, n, M > \sigma_\chi$

upon reception of $< PrePrepare, v, n, M\sigma_\chi >$ **do**
6: **if** ($< PrePrepare, v, n, M > \sigma_\chi$ is correct) **then**
7: send to successor $< Prepare < PrePrepare, v, n, M >> \sigma_\beta$

upon reception of $< Prepare < PrePrepare, v, n, M > \sigma_\beta$ **do**
8: **if** ($< Prepare < PrePrepare, v, n, M >> \sigma_\beta$ is correct) **then**
9: **if** (Prepare messages $< 3f$) **then**
10: send to successor $< Prepare < PrePrepare, v, n, M >> \sigma_\beta$
11: **if** (Prepare messages $== 2f$) **then**
12: send to predecessor $< Commit < Prepare < PrePrepare, v, n >>> \sigma_\beta$

upon reception of $< Commit < Prepare\ PrePrepare, v, n >>> \sigma_\beta$ **do**
13: **if** ($< Commit < Prepare < PrePrepare, v, n >>> \sigma_\beta$ is correct) **then**
14: **if** (Commit messages $< 2f + 1$) **then**
15: send to predecessor $< Commit < Prepare < PrePrepare, v, n >>> \sigma_\beta$
16: **if** (Commit messages $== 2f + 1$) **then**
17: send to successor $< CommitACK < Commit, v, n >> \sigma_\beta$

upon reception of $< CommitACK < Commit, v, n >> \sigma_\beta$ **do**
18: **if** ($< CommitACK < Commit, v, n >> \sigma_\beta$ is correct) **then**
19: **if** (CommitACK messages $==$ $f + 1$) **then**
20: send to predecessor $< Reply, v, n > \sigma_\beta$
21: **if** (CommitACK messages $< 3f+1$) **then**
22: send to successor $< CommitACK < Commit, v, n >> \sigma_\beta$

upon reception of $2f + 1 < Commit, v, n > \sigma_\beta$ **do**
23: Execute requested operation
24: $LocalTinychain \leftarrow L$
25: $Synopsis \leftarrow Synopsis$

upon reception of $< Reply, v, n > \sigma_\beta$ **do**
26: **if** ($< Reply, v, n > \sigma_\beta$ is correct) **then**
27: **if** (Reply messages $==$ $f + 1$) $\wedge$ *replica_is_leader* **then**
28: send to client $< Reply, M, \tau > \sigma_A$
29: **if** (*replica_is_NOT_leader*) **then**
30: send to predecessor $< Reply, v, n > \sigma_\beta$

7.3.4 Synopsis Chain Replication Analysis and Discussion

The main objective of the SynopsisCR is to provide a scalable Byzantine consensus protocol for wireless-operated IoT devices. Given an IoT system that works on a

local area network and wireless settings, it is important to optimize the communication pattern to boost the network performance. Overall, most blockchain protocols are assumed to work on reliable wired communication and, therefore, less restrictive to wireless communication issues. Over wireless media, performance issues are observed when broadcast packet transmissions are used. The performance issues arise due to several reasons, such as high packet error rates, lack of acknowledgments, and low data rate [118]. Some enhancements are designed to overcome such limitations. However, these enhancements are usually implemented in limited deployments and not widespread on most wireless networks. As a result, Synopsis aims to eliminate the broadcast packet transmissions by employing unicast point-to-point communications where nodes are topologically organized in a chain, and each node transmits the packet to the next node in the chain. In SynopsisCR best-case scenario, communication between only the $|P| - f_b$ non-faulty replicas is needed to reach a consensus. This bounded value minimizes the number of intermediate nodes. The primary node initiates consensus by sending a *single* <PrePrepare> to the next node in the chain. Each replica forwards a *single* aggregated message to the next replica on the path. Upon receiving the predefined threshold of the <Prepare> aggregated message, an intermediate replica initiates the <Prepared> phase. In the best-case scenario, this replica is the $(|P| - f_b)^{th}$ replica and the communication is reversed toward the primary node. However, the remaining non-faulty f_b replicas stand by to receive all protocol aggregated messages and update their local log. Nonetheless, they do not participate in the communication unless needed, e.g., in the case of a faulty replica or timeout. The <Committed> phase is initiated by a replica when it receives the predefined threshold of the <Commit> aggregated message. This replica again forwards a *single* <CommitACK> aggregated message to the next neighbor, which is seen as an action to commit the data. Note that the primary does not send a <Reply> to the client until the participating replicas commit the data and send a <Reply> to the

primary. Finally, the primary sends a *single* <Reply>, collectively signed by $f_b + 1$ replicas, to the client.

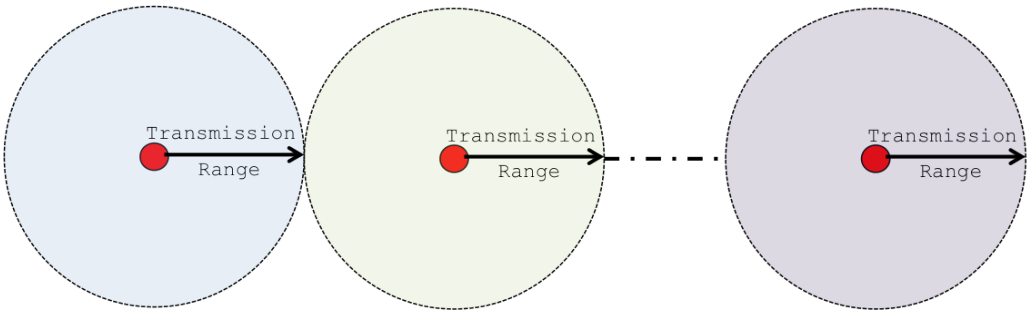


Figure 7.3: Worst case scenario in which the replicas are outside wireless domain of other replicas.

7.3.5 Efficiency and Chain Replication Optimization

In our next refinement, we tackle the scalability challenge resulting from typical SynopsisCR communication patterns that is needed to send the entity data block L with each <Prepare> message. In wireless networks, sniffing the data packet using trusted replicas can improve the performance by an order of magnitude. A node that intercepts a packet carrying the same information from its neighbors eliminates the sending of redundant data. By adopting digital signatures for authentication, sparser messages can be used, enabling the current node to collect and distribute verifiable evidence that specific SynopsisCR steps have succeeded. This process removes all necessity for trustees to communicate the entire bundle of information (blocks) at every phase along the chain. In an optimal scenario, the replicas share the same communication domain, each data block is sent once and all replicas receive it. In the worst case scenario, all communicating replicas are outside the domains of the others, as seen in Fig.7.3, and thus recover legacy SynopsisCR. However, devices mostly share their wireless domain in closed environments, e.g., industrial factories, smart buildings, and supply chain warehouses. Consequently, it is necessary to take advantage of such arrangements to eliminate unnecessary traffic. Moreover, we assume that Syn-

opsis devices can discover other nodes sharing the communication range using one of the neighbor discovery methods, e.g., Received Signal Strength (RSS) and Time of Flight (ToF) based methods [119, 120, 121].

Example: The following presents an example of using the optimized SynopsisCR protocol in a wireless environment. We set f_b as 1; therefore, we have 4 replicas, $|P| = 3 \times 1 + 1$.

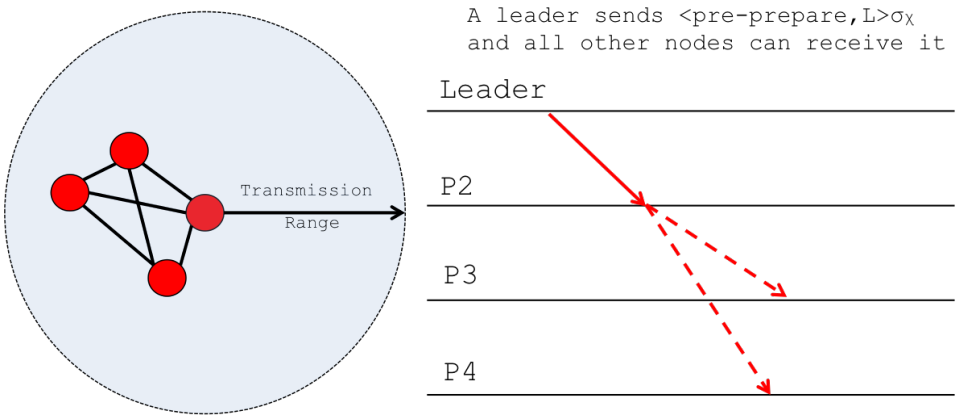


Figure 7.4: Optimal scenario in which the replicas are organized within one communication zone.

- Optimal case scenario: This is the case in which all the replicas can communicate with one another, as shown in Fig.7.4. The SynopsisCR leader generates and signs a <PrePrepare> message containing the data block; this is collected by all consensus parties. When a node enters the <Prepare> phase, it does not append the block again; instead, it preserves the communication bandwidth from needless overhead.

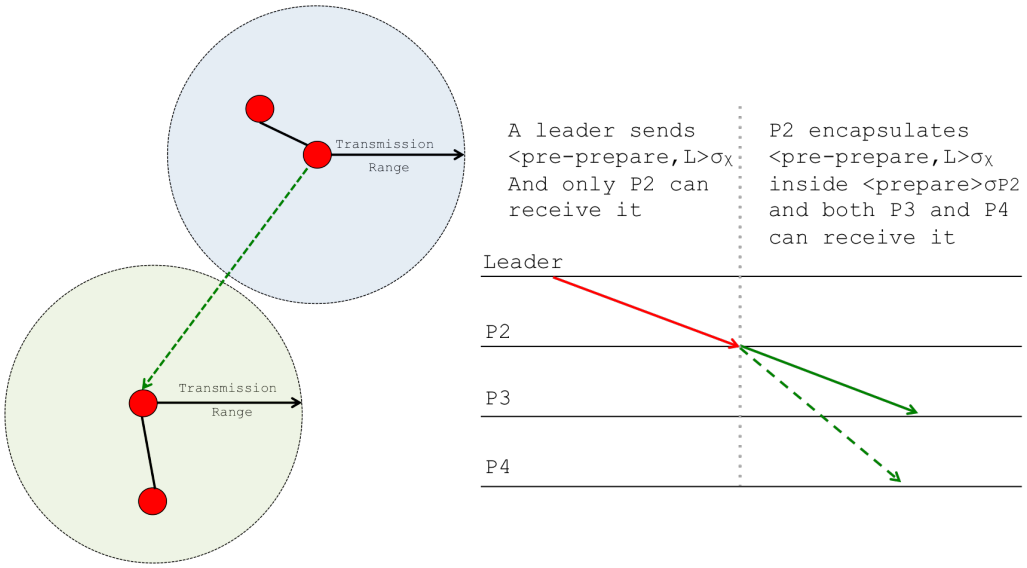


Figure 7.5: Sub-optimal scenario in which the replicas are organized into two communication zones.

- Sub-optimal scenario: In a case between the worst and optimal case scenarios, the replicas are organized into two communication zones with sub-optimal bandwidth overhead. In the given example, Fig.7.5, one replica receives the <PrePrepare> message, but the other two replicas are outside the communication domain. Consequently, the replica that receives the <PrePrepare> message is required to encapsulate the block at the generated <Prepare> message in order for it to be received at the second communication zone.

7.3.6 Global Blockchain Commitment

During the local Tinychain commitment round, each P local set tolerates f_b failure, maintaining both the Synopsis and local chain of tiny blocks. Additionally, it is required to group $|R|$ local Tinychains into one main blockchain. Provided that Byzantine failures are masked inside local clusters and in order to reduce the communication complexity, the main blockchain is committed using a non-Byzantine protocol, Paxos. Paxos is a consensus protocol that tolerates crash and fail-stop faults; $2f_c + 1$ nodes are able to tolerate f_c failures. The leader of each set P initiates the global blockchain construction process, obtains a certified proof from its local cluster on each Paxos action. For example, a Paxos node before sending a <PaxosPrepare> message to the other participated Paxos nodes, it first encapsulates the <PaxosPrepare> message into a SynopsisCR <PrePrepare> message. This message is treated exactly like any other SynopsisCR proposal. As a result, the Paxos node collects a $f_b + 1$ SynopsisCR reply messages on the <PaxosPrepare> message. That means the <PaxosPrepare> message is certified and no Byzantine action can take place on this message. The same is applied to all the participating Paxos nodes and every Paxos message due to the fact that each Paxos node is also a SynopsisCR leader. In consequence, every Paxos action is verified and Paxos agreements are collected on the main Blocks from the group of Paxos acceptors $|R| = K = 2f_c + 1$. Finally, the new main block, G, becomes a globally committed block and is added to the main blockchain. The vanilla version of Paxos is used, with the emphasis that every action is certified by the local clusters before being sent and verified by the receiver, as shown in Algorithm 2. The Paxos communication involves an inter-cluster communication.

- PaxosPrepare: a Paxos node initiates the Paxos process by generating a <PaxosPrepare,n>$_{\sigma_{A_i}}$ message, wher n is the sequence number, and σ_{A_i} in a signature from a group A_i, where $|A| = f_b + 1$ are SynopsisCR replicas and i is the cluster number. First, the <PaxosPrepare,n> is relayed to the local cluster to be certified by the local

SynopsisCR nodes. After verifying the <PaxosPrepare>$_{\sigma_{A_i}}$, it is broadcasted to the other Paxos nodes.

- PaxosPromise: On receiving a valid <PaxosPrepare,n>$_{\sigma_{A_i}}$ message, the Paxos replica generates and passes <PaxosPromise,n> to the local cluster, then obtains the local cluster signature from the SynopsisCR group A_i. After signing the <PaxosPromise> message with σ_{A_i} by the local cluster, the Paxos node sends this information to the sender of the <PaxosPrepare>$_{\sigma_{A_i}}$.

- PaxosPropose: When a Paxos node receives the <PaxosPromise>$_{\sigma_{A_i}}$ message from a majority of the acceptors, it broadcasts the <PaxosPropose,n,G>$_{\sigma_{A_i}}$ message, after certifying the <PaxosPropose> by the local group A_i, to the acceptors to ask them to accept the proposed block, G.

- PaxosAccepted: The Paxos replica receives an order to accept the <PaxosPropose,n,G>$_{\sigma_{A_i}}$ message and block G is finally attached to the main blockchain. However, this node is required to acknowledge the leader by sending the <PaxosAccepted,n>$_{\sigma_{A_i}}$ message. The <PaxosAccepted> is also verified by the local group A_i. The leader upon receiving a <PaxosAccepted,n>$_{\sigma_{A_i}}$ message from a majority of the acceptors, it finalizes the block commitment to the main blockchain and a consensus is reached.

Algorithm 2 Synopsis Global Consensus

1: $G \leftarrow$ *Main Block*

upon start **do**
2: **if** *node_is_valid_leader* **then**
3: Go to Line 14
4: **else**
5: $n \leftarrow$ *sequence number*
6: $verified \leftarrow SynopsisCR($
$< PaxosPrepare, n >)$
7: **if** (verified is True) **then**
8: Broadcast $< PaxosPrepare, n >_{\sigma_{A_i}}$

upon reception of
$< PaxosPrepare, n >_{\sigma_{A_i}}$ **do**
9: **if** ($< PaxosPrepare, n >_{\sigma_{A_i}}$ is correct) **then**
10: $verified \leftarrow SynopsisCR($
$< PaxosPromise, n >)$
11: **if** (verified is True) **then**
12: send to source $< PaxosPromise, n >_{\sigma_{A_i}}$

upon reception of
$< PaxosPromise, n >_{\sigma_{A_i}}$ **do**
13: **if** ($< PaxosPromise, n >_{\sigma_{A_i}}$ is correct) **then**
14: $verified \leftarrow SynopsisCR($
$< PaxosPropose, n, digest(G) >)$
15: **if** (verified is True) **then**
16: Broadcast $< PaxosPropose, n, G >_{\sigma_{A_i}}$

upon reception of
$< PaxosPropose, n, G >_{\sigma_{A_i}}$ **do**
17: **if** ($< PaxosPropose, n, G >_{\sigma_{A_i}}$ is correct) **then**
18: $MainBlockchain \leftarrow G$
19: $verified \leftarrow SynopsisCR($
$< PaxosAccepted, n >)$
20: **if** (verified is True) **then**
21: send to leader $< PaxosAccepted, n >_{\sigma_{A_i}}$

upon reception of
$< PaxosAccepted, n, >_{\sigma_{A_i}}$ **do**
22: **if** ($< PaxosAccepted, n >_{\sigma_{A_i}}$ is correct) **then**
23: $MainBlockchain \leftarrow G$

7.3.7 Global Agreement Analysis and Discussion

The main blockchain is constructed by concatenating several Tinychains, which may involve a wide-area communication, in order to transmit a considerable amount of locally committed data. Communication across such environments is less restrictive than wireless media. The replicas inside a wireless domain reach a consensus on the provided data and commit to the local Tinychain. Malicious attacks are eliminated up to f_b faulty replicas. Masking the Byzantine actions can be done by (1) certifying every outgoing and incoming Paxos message by the local level replicas (2) periodically monitoring the global commitment process to eliminate the leader from break down the global commitment process. Cross-communication is therefore relaxed to consider specified fail-stop failures. This relaxation further improves the complexity of communication between globally assigned nodes and improves their

networking performance. Paxos, for example, has a linear message complexity of $O(|R| = K)$. Moreover, committing data locally can instantaneously meet the stringent IoT requirements of intra-domain applications. When committing data globally, the non-Byzantine agreement maximizes performance for inter-domain applications.

7.3.8 Recovery from failures

All the replicas in Synopsis local levels can mitigate malicious or crash failures. The SynopsisCR protocol keeps a set of timers to ensure that a failure is detected. Each replica on the path requires a timer and an upper bound threshold. Initially, no faulty replicas exist and the timers are set based on average networking latency. The timer is refreshed each time a new message arrives. The timer and the upper bound limit help to identify any faulty or delayed activities.

- Replica p expects a <PrePrepare>, <Prepare>, or <CommitACK> message from its predecessor.
- Replica p expects a <Commit>, or <Reply> message from its successor.
- If the timer has expired, replica p is responsible for broadcasting a message indicating that suspicious behavior has occurred. This broadcasting message turn on the legacy PBFT protocol.

Moreover, SynopsisCR protocol inherits all the PBFT protocol features include view-change protocol. When a leader node is suspected to be failed, a PBFT view-change protocol is activated and a new leader is elected. Note that the leader also needs to inform the other replicas about the main blockchain states at every predefined time interval with a progress proof. The main blockchain status involves information about the global blockchain commitment process, such as the updated blockchain hight (H). Therefore, the view-change protocol provides liveness guarantees in a synchronous model for both local and global Synopsis levels. The new SynopsisCR

leader will take the responsibility to participate in the global commitment process using Paxos. A more detailed description for the view-change process can be found in [2].

7.3.9 Correctness

SynopsisCR is a wireless implementation of PBFT where the state transitions and messages are all the same as PBFT. The main two differences are that (1) SynopsisCR enforces a specific chain-like communication pattern that is most suitable for wireless communication, and (2) it makes nodes relay messages to enable the chain-like communication. Because the messages and state transitions are identical to PBFT, safety of SynopsisCR is inherited from PBFT. The chain-like communication pattern may change the liveness characteristics slightly. However, our design of making SynopsisCR fall back to the original broadcast communication pattern if progress is not being made. This makes SynopsisCR inherits both safety and liveness guarantees of PBFT, since in the worst-case, SynopsisCR would behave in an identical way to PBFT. Generally, Synopsis hierarchical idea is similar to Steward [122] which is a hierarchical algorithm that runs a BFT protocol within each local level, and a lightweight, crash fault-tolerant protocol across the second hierarchical level. Details proofs of correctness is given in [122]. However, we provide a proof sketch of SynopsisCR's correctness as follows:

Lemma 1 (SynopsisCR Safety)**.** *No two non-faulty nodes commit two different tiny blocks ($L_h \neq L'_h$) at the same height (h) inside a local cluster.*

Proof Sketch. By contradiction, suppose that two non-faulty nodes, p_1 and p_2, from the same local cluster agree on two distinct tiny blocks L_h and L_h at height h. That means L_h is committed by p_1 as a result of collecting a set of $2f_b+1$ commit messages. The same is applied to p_2 which commits L'_h by collecting a set of $2f_b + 1$ commit messages. Following the PBFT logic and messages, a non-faulty node collecting $2f_b+1$

commit messages guarantees that no other non-faulty node would receive a different block for the same height (h). As a result, two different non-faulty nodes agree about L_h, which is a contradiction and L_h is equal to L'_h. □

Lemma 2 (SynopsisCR Liveness)**.** *All non-faulty nodes make progress by committing a new tiny block within a limited time-bound* Δ.

Proof Sketch. Under the asynchronous assumption on time bound Δ, every proposed block will have two cases: $2f_b + 1$ commit messages will be received by non-faulty nodes or not. (1) When a collective of $2f_b + 1$ commit messages is received, liveness is achieved. (2) When a collective of $2f_b + 1$ commit messages is not received, the chain-like pattern is broken into a broadcast pattern. As a result of falling back to a broadcast pattern, the liveness is guaranteed as it is inherited from PBFT. □

Proposal	Wireless	hierarchical	Chain	Linear communi-cation	Speculative
SG-PBFT [123]	✓	✗	✗	✗	✗
Steward [122]	✗	✓	✗	✗	✗
Multi-layer PBFT [124]	✗	✓	✗	✗	✗
Zyzzyva [109]	✗	✗	✗	✓	✓
MinZyzzyva [125]	✗	✗	✗	✓	✓
HotStuff [126]	✗	✗	✗	✓	✗
Aliph-Chain [127]	✗	✗	✓	✓	✓
BChain [128]	✗	✗	✓	✓	✓
Synopsis (Our work)	✓	✓	✓	✓	✗

Table 7.1: Synopsis Consensus versus BFT Proposals.

7.4 Synopsis Consensus versus BFT Proposals

The Synopsis is a BFT blockchain framework optimized to work in a large-scale IoT deployment environment over LAN and wireless channels. IoT devices operating in wireless environments suffer from various factors, including electromagnetic interference, mechanical stress, critical temperature, and high humidity, which negatively influence the system performance. Therefore, when designing an IoT fault-tolerant network, one must revisit the participating devices' communication pattern. In summary, previous studies on BFT blockchain mainly focus on protocols that are initially designed for wired network applications. This means that such studies are unsuitable

for wireless scenarios, thus creating a need to investigate the consensus mechanisms for IoT communications integration. From a wireless communication perspective, an analytic modeling framework is provided to obtain a viable area for wireless PBFT-based blockchain networks [129]. The viable area aims to improve the energy savings and overall network performance by ensuring a minimum number of nodes is activated in each wireless PBFT network view. Other wireless-based consensus studies assume a high-level wireless network abstraction such as abstract MAC layer service (absMAC) [130, 131, 132]. Nevertheless, a more realistic assumption needs to be considered when designing a wireless-based blockchain system to maximize the necessary properties of efficiency and fault tolerance. In[133, 134], non-Byzantine protocols are proposed. However, the proposed algorithms can not mitigate Byzantine failures. In contrast, the proposed method in [123] uses a score grouping mechanism in an Internet of Vehicles (IoV) environment to tolerate Byzantine faults. On the other hand, the presented algorithm applies almost the same PBFT communication pattern and only propagates the commit responsibility to the leader.

On the downside, the classical BFT protocols suffer from the exhaustive cost of communication. Amongst the solutions available to reduce the communication overhead is to construct a hierarchical multilayer BFT system proposed in [122]. The proposed hierarchical scheme, steward, reduces the complexity of message on the wide-area transmissions, although it does not eliminate the all-to-all dissemination pattern inside the local level, which influences the performance of the wireless participated device negatively. A recent work [124] proposes a scalable multilayer PBFT-based consensus mechanism that can be achieved by hierarchically grouping nodes into different layers and limiting the communication within the group. However, the scalability is improved, although it results in a longer delay when the consensus process goes through multiple layers.

On the other hand, the speculative consensus methods are the other way of reduc-

ing the all-to-all communication patterns such as Zyzzyva [109], MinZyzzyva [125]. When replicas receive a request from the leader in such algorithms, they speculatively execute the request and immediately send the replies to the client. The speculative algorithms improve the communication overhead but are less resilient against node failures, requiring rollbacking the request executions. In this case, the rollbacking process is a computationally-intensive process that increases the total system delay. In a similar vein, FastBFT [135] also emphases on a linear form of PBFT and a single message client acknowledgement, but it relies on hardware security mechanisms. Another linear BFT protocol is HotStuff [126], and it changes the PBFT mesh topology into a star communication network, meaning that each communication will rely on the leader. The nodes no longer transmit the message to additional nodes, but they only send the message to the leader, which processes it before sending it to other nodes. HotStuff is intended for the wired transmission, and it does not address the wireless channel contention toward the leader.

Moreover, linear communication complexity can also be achievable through chain-based protocols such as Aliph-Chain [127] and BChain [128]. In this case, chain protocols have linear-form nodes, but a leader at the chain's head does not become a bottleneck as the clients generally communicate with the chain's head and tail only. This stabilizes the load between the nodes and allows the chain-replication methods to achieve the best-possible throughput at a higher normal-case latency. In comparison to broadcast-based protocols, the latency cost reduces when the number of concurrent client requests grows. However, existing chain-based protocols rely on speculative execution and have high reconfiguration, computational, and latency costs during replica failures.

SynopsisCR protocol implements a chain protocol to eliminate the speculation limitations while allowing the consensus replicas to communicate in a one-to-one channel, encapsulate the received message (including PBFT phases), and authenticate

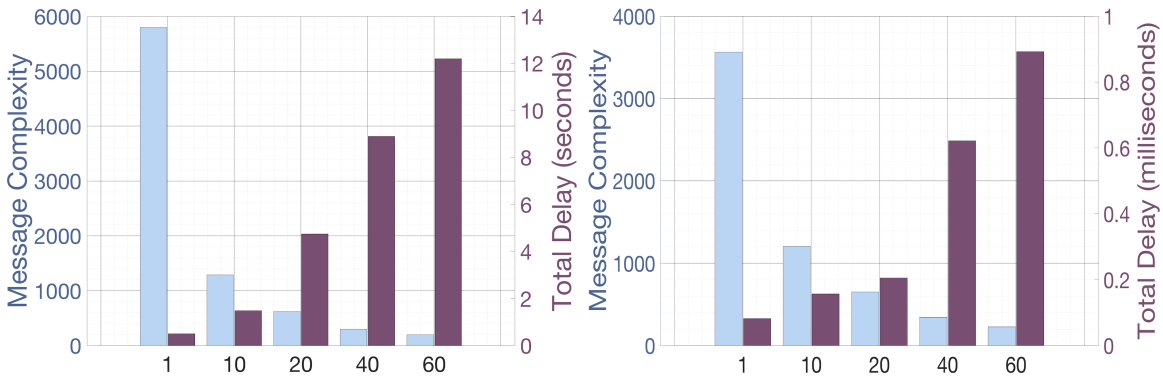
their signature on the path. Consequently, it reduces the huge messaging overhead, especially in an environment with hundreds of devices. It also ends the all-to-all communication pattern of the protocols mentioned above in wireless settings. Thus, SynopsisCR has the needed balance between designing a robust agreement protocol and enhancing the communication cost.

7.5 Performance Evaluation

We evaluate the performance of Synopsis using both large-scale simulations and real-testbed experiments. The main goal of these approaches is to assess the usability of Synopsis in constrained IoT environments without incurring large overheads. In particular, we focus on consensus messaging, storage complexity, and total latency for different parameter combinations.

- Large-scale simulations: We implement Synopsis, including SynopsisCR and ada-CMS in C++, as a new model in the NS-3 network simulator [136]. We depend on simulations to analyze larger-scale IoT behavior. Our proposal is tested using a large network with more than 500 IoT nodes and an area of 100 square meters (10 meters $\times$ 10 meters). SynopsisCR replicas are connected *via* wireless link using an IEEE 802.11n local area network (LAN) protocol operating on the 2.4 GHz frequency band. The second level messages are exchanged using a 10 Gbps Ethernet connection.

- Real-tested experiments: We test the performance metric for real-testbed implementation using Python. Experiments are conducted using Amazon (AWS) t2.micro EC2 instances [137], which have 1 Gigabyte memory and one virtual CPU. We use a physical processor from the Intel Xeon family, with storage based purely on Elastic Block Storage. Our experiments are conducted by spinning up 12 EC2 instances, four instances in three different AWS regions (tolerating



(a) $\lambda_{tinyblock}$. (b) $\lambda_{mainblock}$.

Figure 7.6: Block generation time intervals.

$f_c = 1$ and $f_b = 3$). The AWS regions are US EAST (North Virginia: us-east-1), US EAST (Ohio: us-east-2), and US WEST (Northern California: us-west-1). All connections within the AWS region are interconnected by high-bandwidth, low-latency networking over fully redundant dedicated metro fiber. The connections between different regions are achieved using a fully redundant 100 Gbps fiber network backbone. Note that the EC2 instances use wired connection setups. The main aim of this approach is to conduct an empirical study using real systems and thus measure performance in a regular (non-wireless) setting. These results support that Synopsis has communication-linearity advantages.

7.5.1 Large Scales Simulation

The performance of Synopsis depends on several parameters that are required to be tuned. The most important of these include $\lambda_{tinyblock}$ and $\lambda_{mainblock}$, which influence the achievable behavior. $\lambda_{tinyblock}$ is set at 1, 10, 20, 40, and 60 seconds, the results of which are presented in Fig.7.6a. Total message complexity decreases with increasing $\lambda_{tinyblock}$ from 5800, 1287, 622, 297, down to 193 messages, with respect to the times above. These improvements positively influence wireless node competition and total networking behavior. However, $\lambda_{tinyblock}$ is directly related to the local Tiny-

chain commitment delay, which increases from 0.5, 1.4, 4.7, 8.9, to 12.2 seconds for the times above. The same concept is applied to $\lambda_{mainblock}$, as seen in Fig.7.6b, in which communication complexity improves from 3562, 1204, 652, 346, to 227 messages and the main blockchain commitment delay grows from 0.08, 0.15, 0.2, 0.6, to 0.89 milliseconds for 1, 10, 20, 40, to 60, respectively. Trade-offs in the aforementioned parameters are a situational decision for application management.

Consensus Protocols Comparison

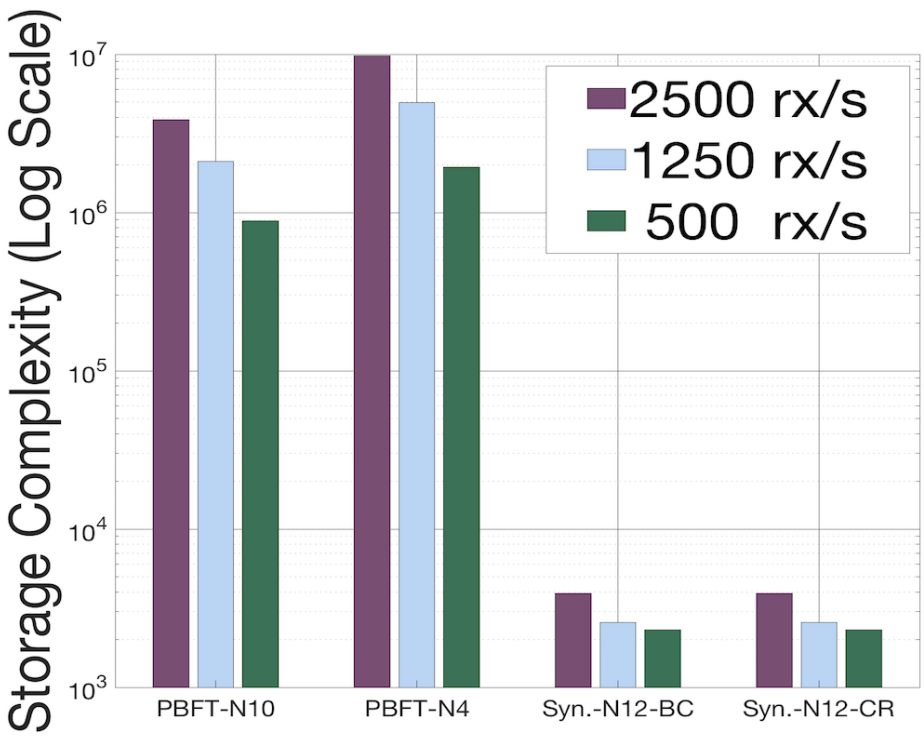


Figure 7.7: Storage complexity for different consensus protocols.

The performance of hierarchical SynopsisCR (Syn.-N12-CR) with a total number of nodes $N = 12$, 3 clusters each has 4 local SynopsisCR nodes, that tolerates $f_c = 1$, $f_b = 3$ is compared to:

1. Syn.-N12-BC: a hierarchical broadcast-PBFT Synopsis with a total number of nodes $N = 12$, 3 clusters each has 4 local SynopsisCR nodes, that tolerates

$f_c = 1$, $f_b = 3$.

2. PBFT-N4: a legacy non-hierarchical broadcast-PBFT with a total number of nodes $N = 4$ that tolerates $f_b = 1$.

3. PBFT-N10: a legacy non-hierarchical broadcast-PBFT with a total number of nodes $N = 10$ that tolerates $f_b = 3$.

- Storage complexity: We assess all protocols performance in dense IoT environments with 100, 250, and 500 nodes in an area of 10 meters $\times$ 10 meters and workloads of 500, 1250, and 2500 requests per second (rx/s). Synopsis is found to dramatically influence the storage capabilities of constrained IoT devices, as illustrated in Fig.7.7. To eliminate $f_b = 3$ failures, the total memory required for a legacy block-like structure is 0.4, 2.0, and 3.8 Megabytes (MB) for workloads of 500, 1250, and 2500 rx/s, respectively. With $f_b = 1$, the legacy block-like replicas store around 1.97, 4.9, and 9.8 MB for workloads of 500, 1250, and 2500 rx/s, respectively. On the contrary, Synopsis replicas store at most 3.7 Kilobytes (KB) of data at each time interval defined by $\lambda_{tinyblock}$. Synopsis thus reduces storage complexity from the order of Megabytes to Kilobytes with an improvement of 1000 times. This considerable reduction in storage overheads eliminates one of the most significant obstacles for massive blockchain employment in IoT networks.

- Message complexity: SynopsisCR aims to reduce the number of consensus messages sent and received by each replica in order to avoid network overloading. This is achieved by eliminating non-essential broadcast behavior in the PBFT protocol thereby boosting overall performance. As shown in Fig.7.8, the legacy PBFT with 10 replicas produces 18629 messages; the same protocol with 4 replicas generates 8237 messages. The Synopsis hierarchical structure with 12 broadcasting PBFT replicas improves the total number of exchanged messages

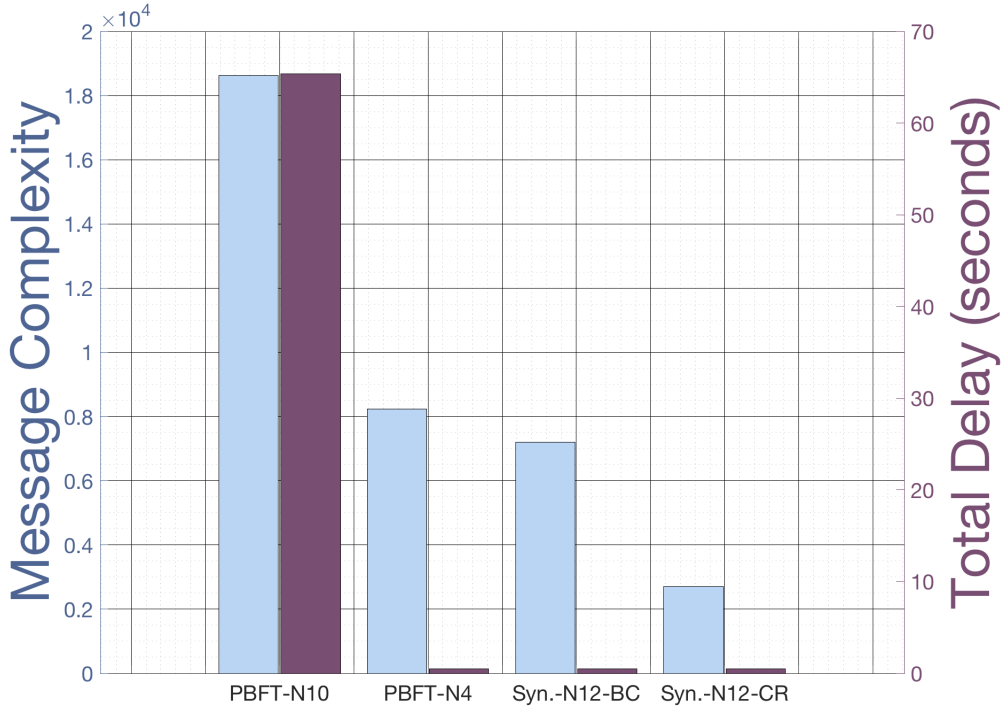


Figure 7.8: Message complexity vs. local Tinychain commitment delay.

to 7209. Compared with the legacy PBFT with 10 replicas, this reflects an enhancement of around 61%; however, the SynopsisCR enables greater messaging reduction. SynopsisCR has a message complexity of 2707 with 85% improvement compared to the legacy PBFT with 10 replicas.

- Local Tinychain commitment delay: The Synopsis structure has specific advantages, such as improved communication and storage efficiency. However, it is well-known that chain message ordering has latency drawbacks compared to the broadcasting nature of communication. Thus, in Synopsis, we introduce a hierarchy to minimize the number of communicated replicas and keep latency costs within an acceptable range, as shown in Fig.7.8. Three configurations, i.e., hierarchical SynopsisCR with 12 replicas, hierarchical broadcast-PBFT Synopsis with 12 replicas, and legacy broadcast-PBFT with 4 replicas, are constructed using 4 local replicas. They are characterized by similar latencies, averaging 400

milliseconds. Compared to the legacy broadcast-PBFT with 10 PBFT replicas, the Synopsis hierarchy reduces the latency by an order of magnitudes from 67 seconds to 400 milliseconds; this corresponds to a 99.4% improvement. Note that the simulation is conducted in a dense environment and the total delay can be further improved using a sparse network.

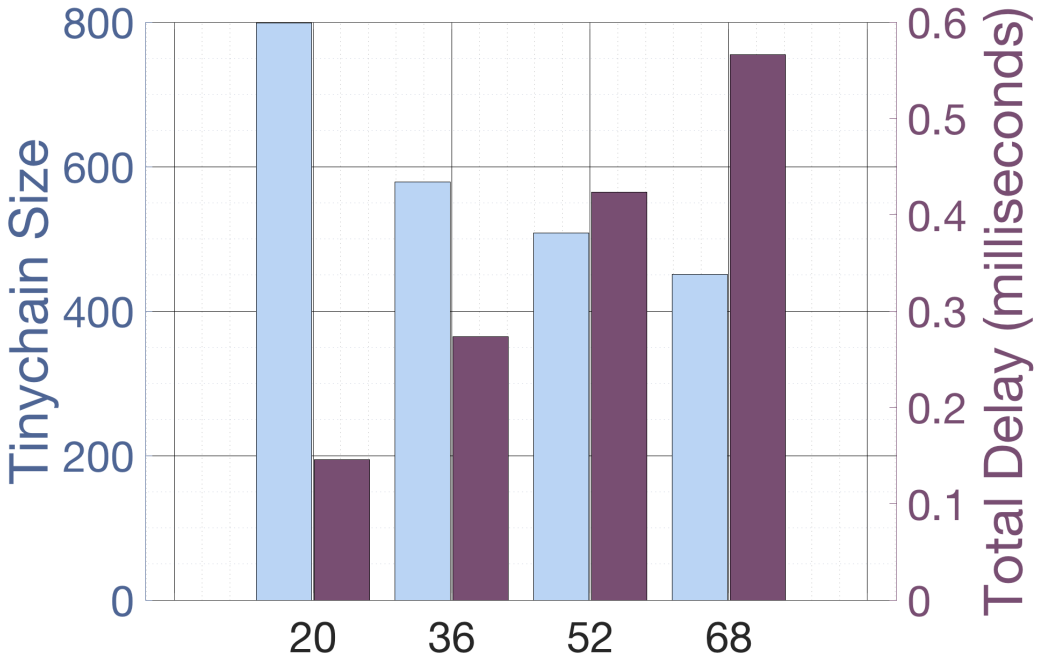


Figure 7.9: Committee size scalability.

Committee Size Scalability

We analyze Synopsis for large committee sizes. The first hierarchy level is fixed to 4 replicas in order to minimize scalability overheads and preserve the Tinychain commitment delay. The second hierarchy level, i.e., the Paxos level, is varied in order to obliterate 2, 4, 6, and 8 fail-stop failures (f_c). This is equivalent to 5, 9, 13, and 17 Paxos nodes and a total of 20, 36, 52, and 68 SynopsisCR replicas among which the data and workload is diffused; therefore, the local Tinychain size is reduced. Fig.7.9 shows that, at each time interval $\lambda_{tinyblock} = 1$, the Tinychain sizes are approximately 800, 580, 507, and 450 bytes for 20, 36, 52, and 68 SynopsisCR replicas, respectively. Storage capabilities thus improve by up to 43%. On the other hand, increasing the number of Paxos replicas increases the main blockchain commitment delay by 0.14, 0.27, 0.42, and 0.56 milliseconds for 5, 9, 13, and 17 Paxos nodes, respectively.

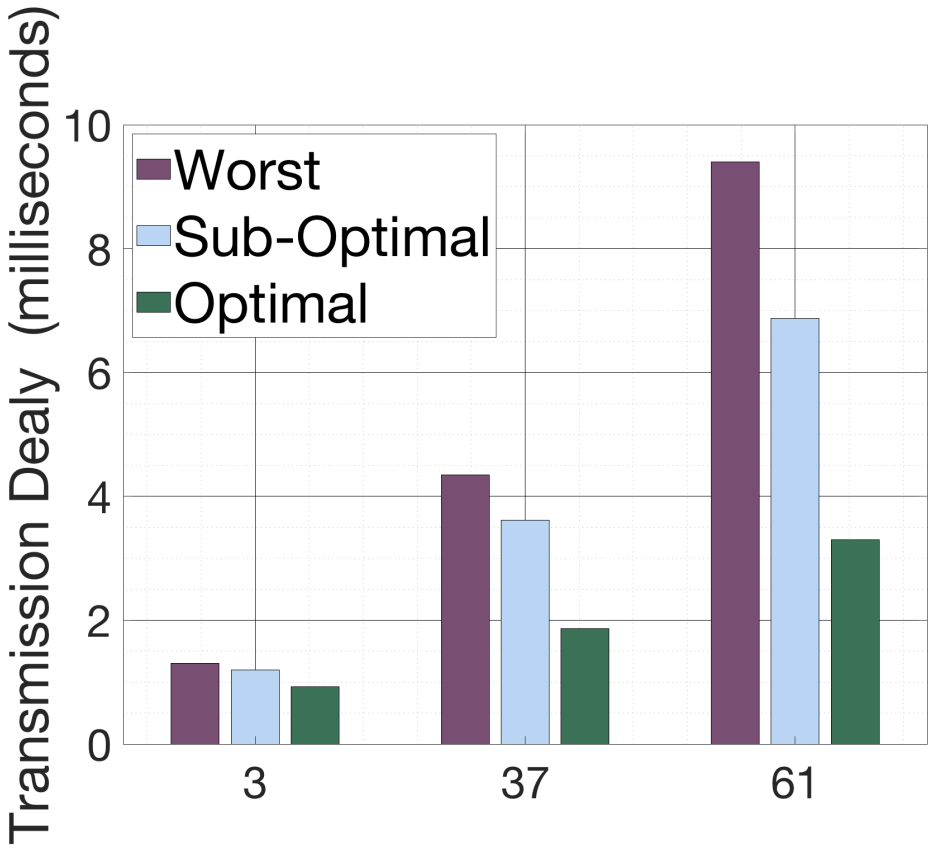


Figure 7.10: Wireless chain replication optimization.

Chain Replication Optimization Results

We evaluate the optimized SynopsisCR protocol in a wireless environment by setting $f_b = 1$ to test the example given in Section 7.3.5. Transmission delay *versus* block size is shown in Fig.7.10. This optimization reduces contention in the wireless domain and positively influences networking delay.

- Worst case scenario: The transmission delay increases from 1.3 milliseconds in the case of 3 KB blocks to 4.3 milliseconds for 37 KB blocks, and 9.4 milliseconds for 61 KB blocks.

- Sub-optimal case scenario: If the whole bundle of the data block is not sent to the fourth participating replica, the transmission delay is enhanced to 1.2 milliseconds for the case of 3 KB blocks, 3.6 milliseconds for the 37 KB blocks,

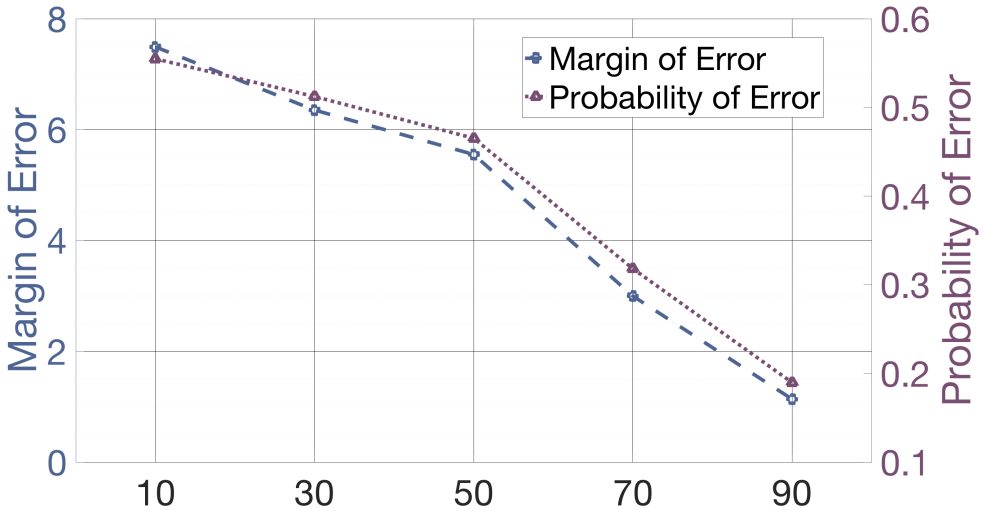


Figure 7.11: Synopsis accuracy and probabilistic errors (Data Redundancy).

and 6.8 milliseconds for 61 KB blocks with an average improvement of 17%.

- Optimal case scenario: The transmission delay improves to 0.9 milliseconds for the case of 3 KB blocks, 1.8 milliseconds for the case of 37 KB blocks, and 3.2 milliseconds for 61 KB blocks, i.e., a 64.9% reduction in transmission delay.

Synopsis Accuracy and Probabilistic Errors

We evaluate the sketch structure of Synopsis to obtain probabilistic guarantees for three parameters:

- Data redundancy percentage: A greater number of redundant data values provide less hashing collision and, therefore, better accuracy. Fig.7.11 shows that the margin of error drops from 7.4, 6.3, 5.5, 2.9, to 1.13 with redundancy percentage from 10%, 30%, 50%, 70%, to 90%, respectively. The probability of errors to occur improves from 55% to 18%.

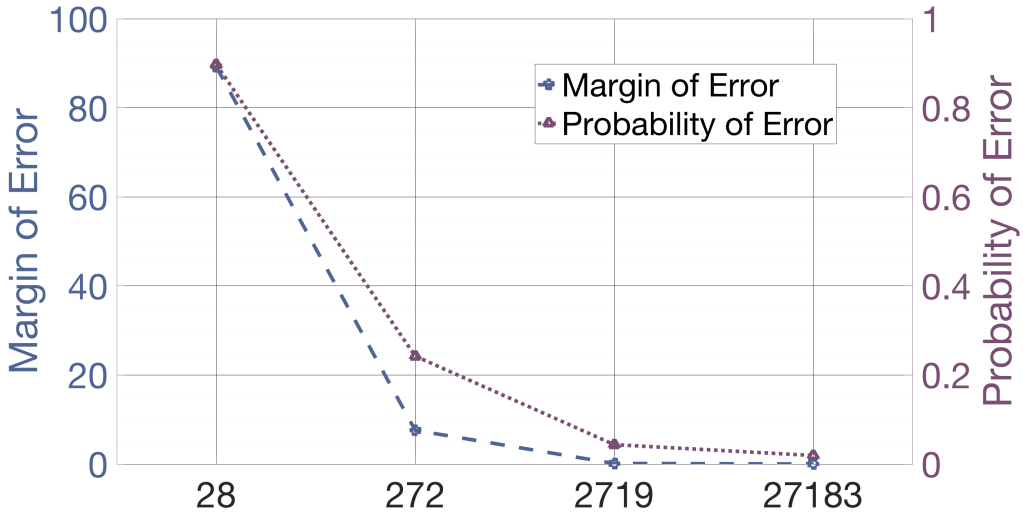


Figure 7.12: Synopsis accuracy and probabilistic errors (Sketch Width).

- Sketch width: The number of allowed hash-values defines the sketch width, such that a wide output range improves the error margin in answering the query. The ϵ parameter varies from 1.0E-1, 1.0E-2, 1.0E-3, to 1.0E-4, resulting in sketch widths of 28, 272, 2719, and 27183, as shown in Fig.7.12. The margin of errors decreases by up to 99% when the sketch width is increased from 27 to 27183. Sketch width also contributes to a 97% reduction in error probability.

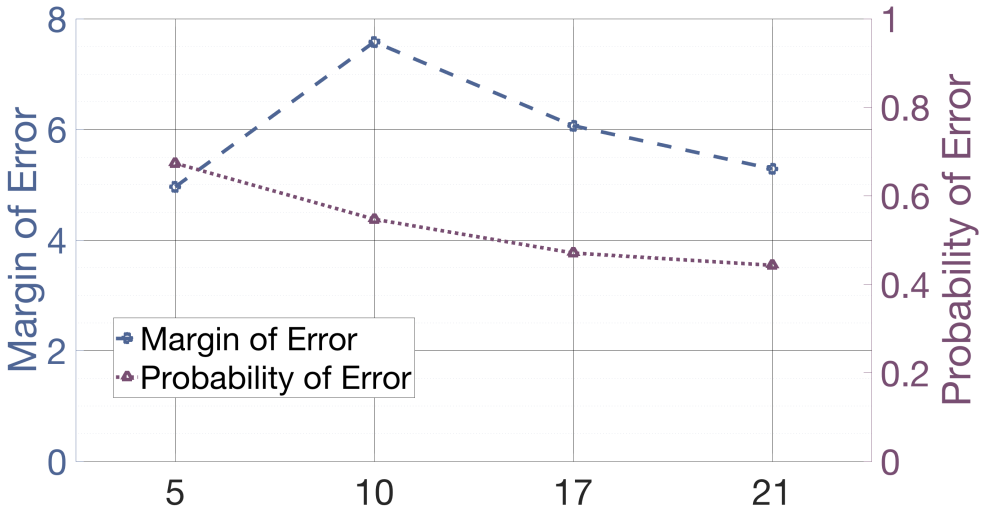


Figure 7.13: Synopsis accuracy and probabilistic errors (Sketch Depth).

- Sketch depth: More hash functions confer a larger sketch depth; however, while enlarging the sketch depth reduces the error probability, it does not the influence resulting accuracy. Varying the δ parameter as 1.0E-2, 1.0E-4, 1.0E-7, and 1.0E-9 resulted in corresponding sketch depths of 5, 10, 17, and 21, respectively. The margin of errors fluctuates between 5 and 8 for the tested hash functions, as shown in Fig.7.13. The probability of errors reduces from 67%, 54%, 47%, to 44% with 5, 10, 17, and 21 hash functions, respectively.

7.5.2 Real Testbed Performance

The performance of a hierarchical SynopsisCR with $f_c = 1$ and $f_b = 3$ is evaluated. 12 AWS EC2 instances are grouped into 3 clusters, each of which has a set of 4 instances running the SynopsisCR consensus protocol and tolerating 1 f_b failure. The input data are randomly generated and equal in size to the batch size, which is set dynamically at the beginning of the experiments.

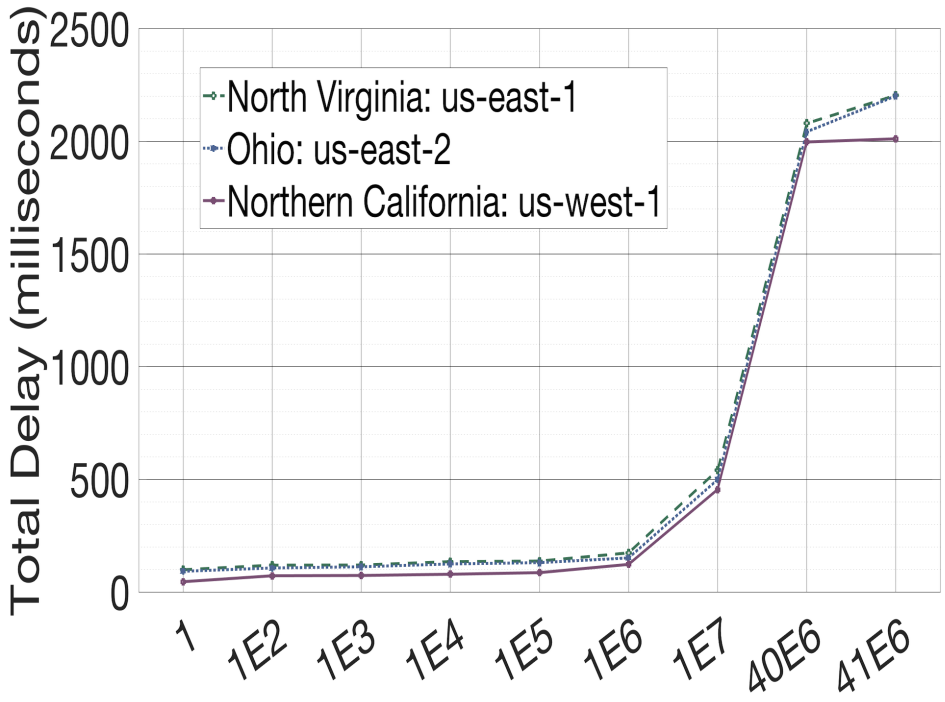


Figure 7.14: Local commitment delay against varied batch size.

- Local Tinychain commitment delay: The total commitment delay is measured for three studied regions, US EAST (North Virginia: us-east-1), US EAST (Ohio: us-east-2), and US WEST (Northern California: us-west-1) as shown in Fig.7.14. We vary the batch size from 1 Byte to 41 MB. Note that 41 MB is the maximum limit to flood AWS instances. The local commitment delay is found to be less than 100 milliseconds at batch sizes between 1 and 100 bytes, to average 126 milliseconds for batch sizes between 10 KB up to 1 MB, and to reach around 500 milliseconds at 10 MB. However, the latency escalates to more than 2 seconds at larger batch sizes of 40 and 41 MB.

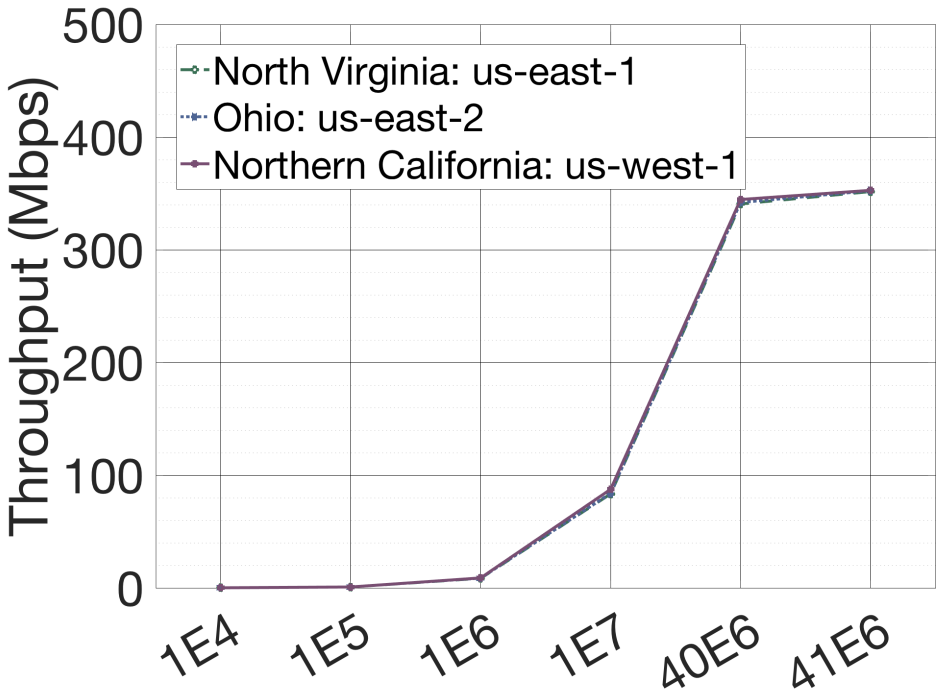


Figure 7.15: Local throughput against varied batch size.

- Local SynopsisCR replicas throughput: Calculated throughputs are presented in Fig.7.15. For a small batch size of 10 KB, the throughput is around 70 Kilobits Per Second (Kbps) and throughput is observed to increase with increasing the batch size, reaching up to 1 Megabit per second (Mbps), 8 Mbps, and 80 Mbps for batch sizes of 100 KB, 1 MB, and 10 MB, respectively. Throughput exhibits the same behavior as delay, increasing rapidly to more than 350 Mbps for a batch size of 41 MB.

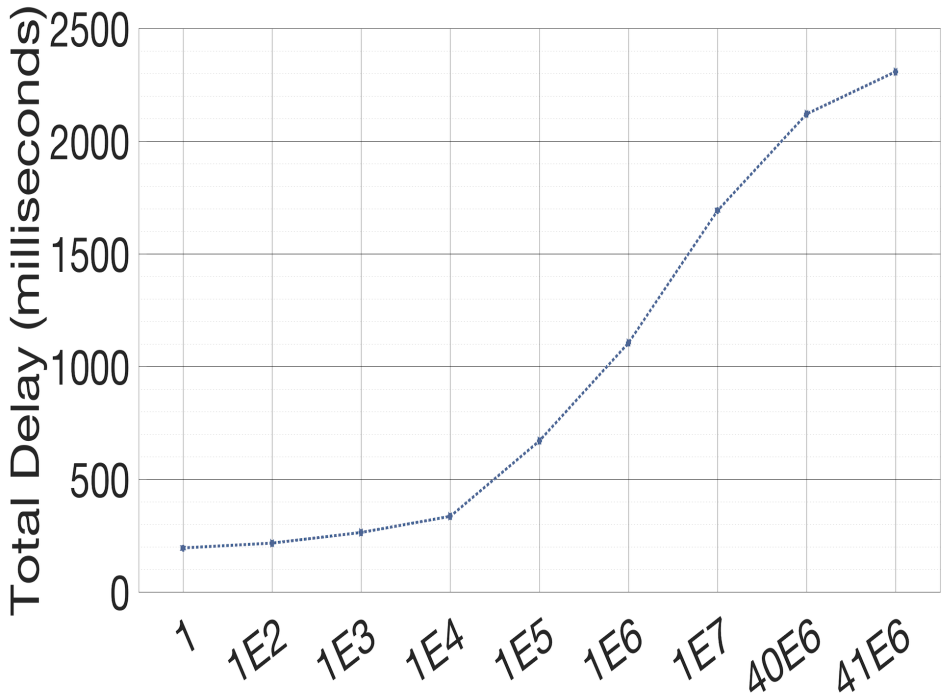


Figure 7.16: Global commitment delay against varied batch size.

- Main blockchain commitment delay: Inter-domain delay is evaluated using the Paxos algorithm shown in Fig.7.16 assuming that the Paxos leader node has the values committed during SynopsisCR within each region. The leader node initiates the Paxos algorithm together with the other two regions in order to mitigate non-Byzantine failures. The global commitment delay averages 250 milliseconds at batch sizes up to 10 KB, increasing to 670 milliseconds for a 100 KB batch. The delay is between 1 and 2 seconds at batch sizes greater than 1 MB.

In summary, the linearity of our proposed method reduces communication complexity and stabilizes system performance. By managing the overall workload and system resources, Synopsis reduces local commitment delays to less than 100 milliseconds and global delays to less than 300 milliseconds.

Chapter 8

Concluding Remarks

8.1 Summary

IoT is a complex cyber-physical system that integrates various devices equipped with sensing, identification, processing, communication, and networking capabilities. In particular, sensors and actuators are becoming increasingly powerful, less expensive, and smaller which make their use ubiquitous. The IoT still holds the promise of improving people's lives through both automation and augmentation. The capabilities offered by IoT can save people and organizations time and money, as well as help improve decision making and outcome in a wide range of application areas. The IoT builds on existing technologies along with standards and protocols to support machine-to-machine communication. The existing challenges in designing conventional IoT systems, including efficient architecture, data management, real-time constraints, and more, need to be addressed by considering new properties and requirements of IoT. Moreover, further research problems arise due to specific IoT requirements, including the connection of IoT devices, the quantity of devices, the coexistence of connected devices with wireless radios, i.e., interference problems, high traffic volume, etc.

We have presented an overview of IoT networks and categorized IoT applications and devices based on different criteria and parameters and highlighted the associated properties with each category. To provide the diverse space of applications with the expected performance commitments, we conducted analysis studies and provided the

mathematical models needed for such kinds of evaluations.

In this dissertation, we presented the strict requirements of IoT real-time traffic and used multimedia applications (IoMT) as a case study. The end-to-end delay was considered as an essential parameter for IoMT real-time applications. The whole system will be broken down, and every incoming packet will be useless if the delay exceeds some predefined threshold. With this potential in mind, we have presented our analytical expressions for the maximum hop-count as well as the packet dropping probability for an IoMT application. We employed M/G/1 queues and used an end-to-end delay threshold for determining delayed packets. We also studied the distributions of the system delay, and we found that the distribution maintains its main characteristics for various system settings. Our results may provide insight to network designers for determining the total number of hop-counts for their application. Using such information with SDN can serve to create and deploy distributed services in proximity to end-users and therefore meet QoS and QoE requirements for different types of real-time applications.

IoT traffic volume and load analysis were presented as a large spike in network traffic to a particular destination causes a widespread disruption of the Internet services for end-users, potentially resulting in billions in revenue losses for online businesses. To this end, we introduced IDIoT, an instantaneous network load detection and prevention technique in the IoT networks supported by the Publish/Subscribe model and MQTT protocol. The proposed scheme aims to prevent the massive traffic generated by IoT devices as close as possible to the network edges. The proposed scheme collects and analyzes the historical traffic arrival rates and allows the central broker node to react according to the collected arrival rates and the current number of publisher nodes at the occurrence of high demand events. M/G/1 queue analysis and simulations were used to verify the proposed model. We compared our mechanism with the typical case to evaluate the obtained results. Simulation results showed that

understanding the underlying traffic patterns can help to identify high network load at an early stage and without propagating the effects to the system users.

After that, we presented the design, implementation, and evaluation of SADIQ, an SDN framework, which provides IoT applications with dynamic and context-driven QoS. SADIQ enables IoT applications to specify high-level QoS policies based on their specific requirements and real-time contexts, using the location-based IoTFlow abstraction and a QoS controller that implements these policies in OpenFlow switches. Our proof-of-concept prototype implementation and testbed evaluation have demonstrated that SADIQ can be supported with today's commodity SDN switches and controllers and can benefit emerging IoT applications.

Furthermore, we studied the impact of the emerging IoT applications on the end-to-end delay incurred by Byzantine-based blockchain systems. An analytical model was driven, and the results were supported using simulations. The results showed the importance of incorporating the unique IoT characteristics in designing Byzantine-based blockchain for IoT networks. We analyzed network level configuration, hop-counts, the number of replicated machines, and its relation to the density and variability of devises. The high contention between packets had masked the delay introduced by the network hops. This prevented the data packets from meeting the application requirements even with the different hops configuration. Conversely, in an environment with reduced contention between its devices, hop-counts directly affected the delay performance. It is established that adding more replica machines will add more delay, but it will increase the system reliability. Thus, the design decision should be taken in respect of traffic characteristics and the application-level requirements.

According to IoT characteristics, we proposed Synopsis. Synopsis is a novel blockchain system for IoT networks. Synopsis follows a hierarchical system design wherein each group of participants form their blockchain network. The hierarchical structure reduces the number of messages and, therefore, positively influences the per-

formance. On the first hierarchical level, chain replication protocol was used where the replicas were organized in a chain and utilized the wireless domain to linearize the communication complexity. Data from this level were arranged in two structures, deterministic blocks, and probabilistic sketches. Upon finalizing the local chain, the second hierarchical level was constructed using a non-Byzantine consensus protocol. The experimental results demonstrated that Synopsis structure scale dramatically, and the device contention was reduced in dense environments. Workload overhead was evenly distributed across available devices and therefore boosted system performance.

8.2 Open Directions and Future Works

Blockchain has the potential to be significantly beneficial for IoT to develop secure networks. However, there is still an open direction to identify the issues and challenges experienced by using these two technologies together.

- Context-aware data-driven solution: IP host-centric model for communication presently is the most effective model on the Internet. Information Centric Networking (ICN) [138] is another noteworthy candidate for the future of Internet architecture. Here the content is the primary component irrespective of its location (host). This crucial component in ICN makes use of the data/content name rather than the address of the network. The name of the content must be unique, independent of location, and persistent. A consumer typically requests the content by name rather than the address of the provider. Therefore, in-network caching could be applicable at the time of communication, as it stores the data closer to the customer to improve data retrieval along with reducing network traffic. When the content is independent of location, ICN intrinsically handles nodes' mobility through reissuing the unsatisfactory requests.

  ICN can also provide convenient data retrieval dependent upon the request/re-

ply model. In ICN, content security is based on attaching all information related to security with the content itself, which is essentially useful in IoT networks. Most of IoT communications and their applications fundamentally follow the content orientated paradigm. For example, retrieving sensed values from IoT sensor and updating the corresponding application with content which has recently been published (weather data, traffic information, patient status, etc.).

Some of the main features of ICN include in-network caching, content naming, and delivery schemes. If these aspects are incorporated into the blockchain network, the content-based delivery model will minimize the communicated traffic and consequently boost the overall performance. Contents safety is another issue that needs to be ensured to guarantee an end-to-end protection for the IoT network, where the content security is carried on the transferable data rather than connection channels.

- Internet of Things future predictions: Cryptographic functionality holds the utmost importance to allow the IoT devices to participate with a blockchain network. The use of such computationally expensive functionality will make the IoT sovereign and self-dependent. Many companies open new opportunities and struggle to implement the blockchain solution into the IoT networks directly. Different devices were manufactured to enable full blockchain functionality on embedded devices and sensors, e.g., EthEmbedded [139].

  In the near future, the smart industry will include remote involvement of humans and machines in real time to support the Industrial Tactile Internet (ITI) [140]. As ITI is expected to use ultra low latency communication and offers exceptionally high availability, reliability, and security, implementing blockchain in such a system becomes a big challenge. Proper integration of blockchain in ITI can introduce a new revolution that enables efficient remote controlling,

tracking essential data, and providing safe environments to prevent accidents and hazards.

To fully support future predictions, we need to see scenarios where IoT devices will have better specifications like more data storage, processing power, and connection efficiency. Moreover, the price of hardware and the size of technology used in devices will be decreased. However, designing new IoT devices should incorporate the required security mechanisms in an early strategic planning phase to ensure a strong and secure smart system. Rather than building a blockchain system that disinvests most of the sophisticated functionalities to satisfy the constrained IoT devices.

REFERENCES

[1] I. Bashir, *Mastering Blockchain - Second Edition*. Packt-Publishing, March 2018.

[2] M. Castro, B. Liskov *et al.*, "Practical Byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, vol. 99. USENIX Association, 1999, pp. 173–186.

[3] L. Lamport *et al.*, "Paxos Made Simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, Nov 2001.

[4] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel *et al.*, "Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, 2017.

[5] G. Alliance, "5G for connected industries and automation," Electronic Manufacturers Association, 60528 Frankfurt am Main, Germany, Tech. Rep. 2, Feb 2019.

[6] A. L. Patrik Cerwall *et al.*, "Massive IoT in the city," Telefonaktiebolaget LM Ericsson, Sweden, Tech. Rep., Nov 2020.

[7] "International Data Corporation (IDC), Inc," https://www.idc.com/, Accessed: 3-Nov-2020.

[8] J. Van Randwyk, "CES-Annual Report," Lawrence Livermore National Lab (LLNL), Livermore, CA (United States), Tech. Rep. 21, Apr 2019.

[9] S. A. Alvi, B. Afzal, G. A. Shah, L. Atzori, and W. Mahmood, "Internet of Multimedia Things: Vision and challenges," *Ad Hoc Networks*, vol. 33, pp. 87–111, 2015.

[10] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, 2017.

[11] S. Martinis, A. Twele, and S. Voigt, "Towards operational near real-time flood detection using a split-based automatic thresholding procedure on high resolu-

tion terrasar-x data," *Natural Hazards and Earth System Sciences*, vol. 9, pp. 303–314, 2009.

[12] "Melbourne data. city of melbourne's open data platform," https://data.melbourne.vic.gov.au/, Accessed: 13-Feb-2019.

[13] "Weather Signal - the crowedsourced weather map," www.weathersignal.com/, Accessed: 21-May-2018.

[14] "Air quality egg- a community-led air quality sensing network," www.airqualityegg.com/, Accessed: 21-May-2018.

[15] A. Minteer, *Analytics for the Internet of Things (IoT)*. Packt Publishing Ltd, July 2017.

[16] K. K. Patel, S. M. Patel *et al.*, "Internet of things-IoT: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[17] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of IoT: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things journal*, vol. 1, no. 4, pp. 349–359, 2014.

[18] R. Van Kranenburg and A. Bassi, "IoT challenges," *Communications in Mobile Computing*, vol. 1, no. 1, p. 9, 2012.

[19] C. Rodríguez-Domínguez, K. Benghazi, M. Noguera, J. L. Garrido, M. L. Rodríguez, and T. Ruiz-López, "A communication model to integrate the request-response and the publish-subscribe paradigms into ubiquitous systems," *Sensors*, vol. 12, no. 6, pp. 7648–7668, 2012.

[20] "MQTT.org," http://mqtt.org, Accessed: 3-Nov-2020.

[21] P. Lea, *Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security*. Packt Publishing Ltd, 2018.

[22] S. Basu, A. Bardhan, K. Gupta, P. Saha, M. Pal, M. Bose, K. Basu, S. Chaudhury, and P. Sarkar, "Cloud computing security challenges & solutions-a survey," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2018, pp. 347–356.

[23] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing:

Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[24] H. Zahmatkesh and F. Al-Turjman, "Fog computing for sustainable smart cities in the IoT era: Caching techniques and enabling technologies - an overview," *Sustainable Cities and Society*, vol. 59, p. 102139, 2020.

[25] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2018.

[26] O. N. Foundation, "OpenFlow switch specification version 1.3.2 (wire protocol 0x04)," Open Networking Foundation, Tech. Rep., Mar 2015.

[27] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of Open vSwitch," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 117–130.

[28] "Project floodlight," http://www.projectfloodlight.org/, Accessed: 21-May-2018.

[29] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," Network Working Group, Tech. Rep., July 1994.

[30] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *Network Working Group RFC 2475*, Dec 1998.

[31] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over mpls," Network Working Group RFC 2702, Tech. Rep., Sept 1999.

[32] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe-a stateless active queue management scheme for approximating fair bandwidth allocation," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, vol. 2. IEEE, 2000, pp. 942–951.

[33] W.-c. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, "The blue active queue management algorithms," *IEEE/ACM transactions on networking*, vol. 10, no. 4, pp. 513–528, 2002.

[34] X. Xiao and L. M. Ni, "Internet qos: a big picture," *IEEE network*, vol. 13, no. 2, pp. 8–18, 1999.

[35] R. Guérin, S. Kamat, V. Peris, and R. Rajan, "Scalable QoS provision through buffer management," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 29–40, 1998.

[36] M. B. Hossain, "QoS-aware Intelligent Routing for Software Defined Networking," Ph.D. dissertation, University of Akron, 2020.

[37] F. S. Tegueu, S. Abdellatif, T. Villemur, P. Berthou, and T. Plesse, "Towards application driven networking," in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2016, pp. 1–6.

[38] A. T. Naman, Y. Wang, H. H. Gharakheili, V. Sivaraman, and D. Taubman, "Responsive high throughput congestion control for interactive applications over SDN-enabled networks," *Computer Networks*, vol. 134, pp. 152–166, 2018.

[39] A. Kucminski, A. Al-Jawad, P. Shah, and R. Trestian, "QoS-based routing over Software Defined Networks," in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2017, pp. 1–6.

[40] F. Li, J. Cao, X. Wang, and Y. Sun, "A QoS guaranteed technique for cloud applications based on Software Defined Networking," *IEEE access*, vol. 5, pp. 21 229–21 241, 2017.

[41] W. Wang, Y. Tian, X. Gong, Q. Qi, and Y. Hu, "Software defined autonomic QoS model for future internet," *Journal of Systems and Software*, vol. 110, pp. 122–135, 2015.

[42] H. E. Egilmez, S. Civanlar, and A. M. Tekalp, "A distributed QoS routing architecture for scalable video streaming over multi-domain openflow networks," in *2012 19th IEEE International Conference on Image Processing*. IEEE, 2012, pp. 2237–2240.

[43] ——, "An optimization framework for QoS-enabled adaptive video streaming over openflow networks," *IEEE Transactions on Multimedia*, vol. 15, no. 3, pp. 710–715, 2012.

[44] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using openflow-assisted adaptive video streaming," in *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*. ACM, 2013, pp. 15–20.

[45] S. Gorlatch and T. Humernbrum, "Enabling high-level qos metrics for interactive online applications using SDN," in *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015, pp. 707–711.

[46] S. Gorlatch, T. Humernbrum, and F. Glinka, "Improving QoS in real-time internet applications: from best-effort to Software-Defined Networks," in *2014 international conference on computing, networking and communications (ICNC)*. IEEE, 2014, pp. 189–193.

[47] J. Seeger, A. Brring, M. Pahl, and E. Sakic, "Rule-based translation of application-level QoS constraints into SDN configurations for the IoT," in *2019 European Conference on Networks and Communications (EuCNC)*, 2019, pp. 432–437.

[48] G. Deng and K. Wang, "An application-aware QoS routing algorithm for SDN-based IoT networking," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 00 186–00 191.

[49] N. Saha, S. Bera, and S. Misra, "Sway: Traffic-aware QoS routing in Software-Defined IoT," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2018.

[50] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor openflow: Enabling software-defined wireless sensor networks," *IEEE Communications letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[51] K. Ryu, Y. Koizumi, and T. Hasegawa, "Name-based geographical routing/forwarding support for location-based IoT services," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. IEEE, 2016, pp. 1–6.

[52] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized task-aware scheduling for data center networks," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM New York, NY, USA, 2014, pp. 431–442.

[53] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, 2012, pp. 31–36.

[54] J. Dean and S. Ghemawat, "Mapreduce: a flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[55] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Satoshi Nakamoto Institute*, Oct 2008.

[56] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[57] J. R. Douceur, "The sybil attack," in *International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

[58] "Watson Internet of Things-IBM watson IoT platform," https://www.ibm.com/internet-of-things/solutions/iot-platform/watson-iot-platform, Accessed: 3-Nov-2020.

[59] "Statista size of the bitcoin blockchain from 2010 to 2020," https://www.statista.com/, Accessed: 3-Nov-2020.

[60] S. King and S. Nadal, "PPcoin: Peer-to-peer crypto-currency with Proof-of-Stake," *self-published paper, August*, vol. 19, p. 1, 2012.

[61] D. Larimer, "Delegated Proof-of-Stake (DPoS)," BitShares Blockchain Foundation, Tech. Rep., Apr 2014.

[62] D. Schwartz, N. Youngs, A. Britto *et al.*, "The ripple protocol consensus algorithm," Ripple Labs Inc, Tech. Rep. 8, Sept 2014.

[63] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with BFT-SMART," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 355–362.

[64] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: BFT consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[65] T. D. Chandra, R. Griesemer, and J. Redstone, "Paxos made live: An engineering perspective," in *Association for Computing Machinery*, ser. PODC '07, New York, NY, USA, 2007.

[66] M. Samaniego and R. Deters, "Blockchain as a Service for IoT," in *IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE, Dec 2016, pp. 433–436.

[67] L. Bai, M. Hu, M. Liu, and J. Wang, "BPIIoT: A Light-Weighted Blockchain-Based Platform for Industrial IoT," *IEEE Access*, vol. 7, pp. 58 381–58 393, May 2019.

[68] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2017.

[69] C. H. Liu, Q. Lin, and S. Wen, "Blockchain-Enabled Data Collection and Sharing for Industrial IoT With Deep Reinforcement Learning," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3516–3526, June 2019.

[70] K. Toyoda, M. Shakeri, X. Chi, and A. N. Zhang, "Performance Evaluation of Ethereum-based On-chain Sensor Data Management Platform for Industrial IoT," in *IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 1–8.

[71] Y. Xu, J. Ren, G. Wang, C. Zhang, J. Yang, and Y. Zhang, "A Blockchain-based Non-Repudiation Network Computing Service Scheme for Industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3632–3641, June 2019.

[72] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[73] W. F. Silvano and R. Marcelino, "Iota Tangle: A cryptocurrency to communicate Internet of Things data," *Future Generation Computer Systems*, 2020.

[74] C. LeMahieu, "Nano: A feeless distributed cryptocurrency network," Nano, Tech. Rep., 2018.

[75] "IOTA NODES: statistics," https://iota-nodes.net/statistics, Accessed: 25-Nov-2020.

[76] "NANO CRAWLER: node status," https://nanocrawler.cc/status, Accessed: 25-Nov-2020.

[77] C. Lee, "Litecoin," https://litecoin.org, Accessed: 3-Nov-2020.

[78] Grayblock, "Blockchain finality in iot: The importance of finality in cross-chain communication," https://medium.com, Accessed: 3-Nov-2020.

[79] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint*, Oct 2017.

[80] "NEO-a distributed network for the smart economy - white paper," http://docs.neo.org/en-us/whitepaper.html, Accessed: 3-Nov-2020.

[81] "IoTeX:a decentralized network for Internet of Things (IoT)," https://iotex.io/white-paper, Accessed: 25-Nov-2020.

[82] R. Investment, "A blockchain platform for the smart economy," NEO, Tech. Rep., 2018.

[83] G. Greenspan, "IoT Chain: A high-security lite IoT OS white paper," multichain, Tech. Rep., 2015.

[84] R. Han, V. Gramoli, and X. Xu, "Evaluating blockchains for IoT," in *New Technologies, Mobility and Security (NTMS), 2018 9th IFIP International Conference on*. IEEE, 2018, pp. 1–5.

[85] N. Teslya and I. Ryabchikov, "Blockchain platforms overview for industrial IoT purposes," in *2018 22nd Conference of Open Innovations Association (FRUCT)*. IEEE, 2018, pp. 250–256.

[86] I. Adan and J. Resing, *Queueing Theory*. Eindhoven University of Technology, 2002.

[87] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain," *The Annals of Mathematical Statistics*, pp. 338–354, 1953.

[88] M. Oche, R. M. Noor, and C. Chembe, "Multivariate statistical approach for estimating QoE of real-time multimedia applications in vehicular ITS network," *Computer Communications*, vol. 104, pp. 88–107, 2017.

[89] S. Zahl, "Bounds for the central limit theorem error," *SIAM Journal on Applied Mathematics*, vol. 14, no. 6, pp. 1225–1245, 1966.

[90] I. S. Gradshteyn and I. Ryzhik, *Table of Integrals, Series, and Products*, 7th ed. Amsterdam: Elsevier/Academic Press, 2007.

[91] D.-H. Shin, "Conceptualizing and measuring quality of experience of the Internet of Things: Exploring how quality is perceived by users," *Information & Management*, vol. 54, no. 8, pp. 998–1011, 2017.

[92] "HIVEMQ," https://www.hivemq.com, Accessed: 3-Nov-2020.

[93] O. C. A. W. Group *et al.*, "Openfog reference architecture for fog computing," OpenFog Consortium, Tech. Rep., Feb 2017.

[94] K. Liang, L. Zhao, X. Chu, and H. H. Chen, "An integrated architecture for software defined and virtualized radio access networks with fog computing," *IEEE Network*, vol. 31, no. 1, pp. 80–87, Jan 2017.

[95] K. M. Modieginyane, B. B. Letswamotse, R. Malekian, and A. M. Abu-Mahfouz, "Software defined wireless sensor networks application opportunities for efficient network management: A survey," *Computers and Electrical Engineering*, vol. 66, pp. 274–287, Feb 2018.

[96] U. S. A. M. Service, *Military Grid Reference System: Plate II of AMS Technical Manual*. North Atlantic Treaty Organization, May 1953.

[97] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[98] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[99] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A distributed control platform for large-scale production networks." in *OSDI*, vol. 10, 2010, pp. 1–6.

[100] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing Software Defined Networks," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 1–13.

[101] A. Voellmy, J. Wang, Y. R. Yang, B. Ford, and P. Hudak, "Maple: Simplifying SDN programming using algorithmic policies," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM New York, NY, USA, 2013, pp. 87–98.

[102] C. Schlesinger, H. Ballani, T. Karagiannis, and D. Vytiniotis, "Quality of Service Abstractions for Software-Defined Networks," Microsoft Research, Tech. Rep., Microsoft Research, 2015.

[103] C. Pautasso, E. Wilde, and R. Alarcon, *REST: advanced research topics and practical applications*. Springer, 2014.

[104] "Json (javascript object notation)," http://www.json.org/, Accessed: 21-May-2020.

[105] "Hewlett packard enterprise. aruba 2930f switch series data sheet," http://www.arubanetworks.com/assets/ds/DS_2930FSwitchSeries.pdf, Accessed: 21-May-2020.

[106] "Blockchain speeds and the scalability debate," https://blocksplain.com, Accessed: 28-Feb-2018.

[107] T. Stack, "Internet of Things (IoT) data continues to explode exponentially. Who is using that data and how?" *cisco blogs*, Feb 2018.

[108] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in blockchains," *IEEE Access*, vol. 8, pp. 14 155–14 181, 2020.

[109] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative Byzantine fault tolerance," in *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, 2007, pp. 45–58.

[110] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. Reiter, D.-A. Seredinschi, O. Tamir, and A. Tomescu, "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.

[111] M. Alaslani and B. Shihada, "Analyzing latency and dropping in todays internet of multimedia things," in *IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 2019.

[112] S. Mubeen, S. A. Asadollah, A. V. Papadopoulos, M. Ashjaei, H. Pei-Breivold, and M. Behnam, "Management of service level agreements for cloud services in iot: A systematic mapping study," *IEEE Access*, vol. 6, pp. 30 184–30 207, 2018.

[113] T. C. C. E. D. C. (CCEDC), "Building guide for manufacturers," https://chippewa-wi.com, Accessed: 3-Nov-2020.

[114] Wikipedia, "List of United States cities by area — Wikipedia, the free encyclopedia," https://en.wikipedia.org/wiki/List_of_United_States_cities_by_area, 2018, Accessed: 3-Nov-2020.

[115] K. I. Park and Park, *Fundamentals of Probability and Stochastic Processes with Applications to Communications*. Springer, 2018.

[116] A. Singh, S. Garg, R. Kaur, S. Batra, N. Kumar, and A. Y. Zomaya, "Probabilistic data structures for big data analytics: A comprehensive review," *Knowledge-Based Systems*, vol. 188, p. 104987, 2020.

[117] A. Shrivastava, A. C. Konig, and M. Bilenko, "Time adaptive sketches (ada-sketches) for summarizing data streams," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 1417–1432.

[118] I. . W. Group *et al.*, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, amendment 6: Enhancements for high efficiency WLAN," *IEEE P802. 11ax D*, vol. 2, p. 2, 2018.

[119] L. Chen and K. Bian, "Neighbor discovery in mobile sensing applications: A comprehensive survey," *Ad Hoc Networks*, vol. 48, pp. 38–52, 2016.

[120] B. Sravankumar and N. Rao Moparthy, "A survey on continuous neighbor discovery for mobile low duty cycle wireless sensor network," *Materials Today: Proceedings*, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221478532100554X

[121] Y. Wang, T. Zhang, S. Mao, and T. T. S. Rappaport, "Directional neighbor discovery in mmwave wireless networks," *Digital Communications and Networks*, vol. 7, no. 1, pp. 1–15, 2021.

[122] Y. Amir, C. Danilov, D. Dolev, J. Kirsch, J. Lane, C. Nita-Rotaru, J. Olsen, and D. Zage, "Steward: Scaling Byzantine fault-tolerant replication to wide area networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 1, pp. 80–93, 2008.

[123] G. Xu, Y. Liu, J. Xing, T. Luo, Y. Gu, and S. Liu, "SG-PBFT: a secure and highly efficient blockchain pbft consensus algorithm for internet of vehicles," *arXiv preprint arXiv:2101.01306*, 2021.

[124] W. Li, C. Feng, L. Zhang, H. Xu, B. Cao, and M. A. Imran, "A scalable multi-layer PBFT consensus for blockchain," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1146–1160, 2021.

[125] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Verissimo, "Efficient Byzantine fault-tolerance," *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 16–30, 2011.

[126] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "HotStuff: BFT consensus with linearity and responsiveness," in *Association for Computing Machinery*, ser. PODC '19, New York, NY, USA, 2019.

[127] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 BFT protocols," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 4, pp. 1–45, 2015.

[128] S. Duan, H. Meling, S. Peisert, and H. Zhang, "BChain: Byzantine replication with high throughput and embedded reconfiguration," in *International Conference on Principles of Distributed Systems*. Springer, 2014, pp. 91–106.

[129] O. Onireti, L. Zhang, and M. A. Imran, "On the viable area of wireless practical Byzantine fault tolerance (PBFT) blockchain networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[130] H. Moniz, N. F. Neves, and M. Correia, "Byzantine fault-tolerant consensus in wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 12, pp. 2441–2454, 2012.

[131] C. Newport, "Consensus with an abstract MAC layer," in *Association for Computing Machinery*, ser. PODC '14, New York, NY, USA, 2014.

[132] C. Newport and P. Robinson, "Fault-tolerant consensus with an abstract MAC layer," in *32nd International Symposium on Distributed Computing, DISC 2018*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, 2018, p. 38.

[133] M. Xu, C. Liu, Y. Zou, F. Zhao, J. Yu, and X. Cheng, "wChain: A fast fault-tolerant blockchain protocol for multihop wireless networks," *arXiv preprint arXiv:2102.01333*, 2021.

[134] V. Poirot, B. Al Nahas, and O. Landsiedel, "Paxos made wireless: Consensus in the air," in *Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks*, 2019, pp. 1–12.

[135] J. Liu, W. Li, G. O. Karame, and N. Asokan, "Scalable Byzantine consensus via hardware-assisted secret sharing," *CoRR*, vol. abs/1612.04997, 2016.

[136] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.

[137] "AWS Elastic Computing (EC2) - Free Cloud Services with AWS," https://aws.amazon.com, Accessed: 25-Nov-2020.

[138] B. Nour, K. Sharif, F. Li, S. Biswas, H. Moungla, M. Guizani, and Y. Wang, "A Survey of Internet of Things Communication using ICN: A Use Case Perspective," *Computer Communications*, vol. 142-143, pp. 95–123, May, 2019.

[139] "Ethereum Computer Built on Embedded Devices," http://ethembedded.com/, Accessed 5 Jan 2021.

[140] A. Aijaz and M. Sooriyabandara, "The tactile Internet for industries: A review," *Proceedings of the IEEE*, vol. 107, no. 2, pp. 414–435, 2018.

Publications

• M. Alaslani, S. Suri, F. Nawab, and B. Shihada, "Synopsis: Scalable Byzantine Distributed Ledgers for IoT networks", to be submitted, IEEE International Conference on Network Protocols (ICNP), 2021.

• M. Alaslani, O. Amin, F. Nawab, and B. Shihada, "Rethinking Blockchain Integration with the Industrial Internet of Things," IEEE Internet of Things Magazine: Special issue on Blockchain-Enabled Industrial Internet of Things: Advances, Applications, and Challenges, Vol. 3, No. 4, pp. 70-75, 2020.

• E. Ahmad, M. Alaslani, F. Dogar, and B. Shihada, "Location-Aware, Context-Driven QoS for IoT Applications", IEEE Systems Journal, Vol. 14, No. 1, pp. 232-243, 2020.

• M. Alaslani, F. Nawab, and B. Shihada, "Blockchain in IoT Systems: End-to-End Delay Evaluation", IEEE Internet of Things Journal , Vol. 6, No. 5, pp. 8332-8344, 2019.

• M. Alaslani and B. Shihada, "Analyzing Latency and Dropping in Today's Internet of Multimedia Things", in Proc. IEEE Consumer Communications & Networking Conference (CCNC), pp. 1 - 4, 2019.

• M. Alaslani and B. Shihada, "Intelligent Edge: an Instantaneous Detection of IoT Traffic Load", in Proc. IEEE International Conference on Communications (ICC), pp. 1 - 6, 2018.