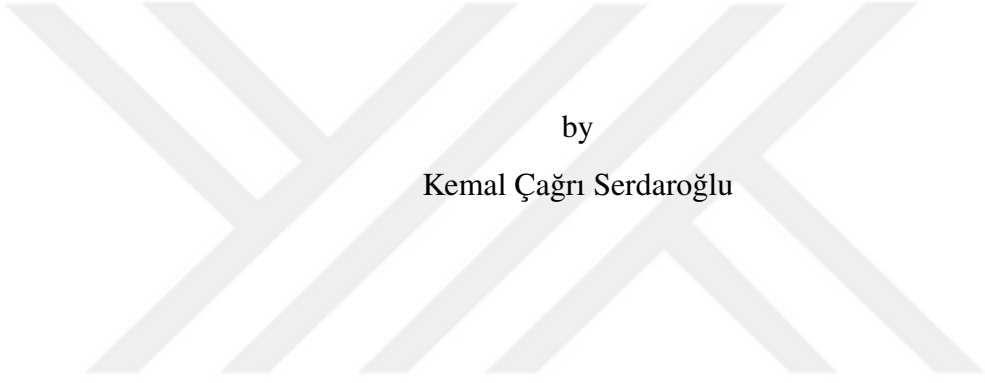


DATA TRAFFIC MANAGEMENT IN IOT APPLICATIONS USING COMPLEX EVENT
PROCESSING



by
Kemal Çağrı Serdaroglu

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Computer Engineering

Yeditepe University

2022

DATA TRAFFIC MANAGEMENT IN IOT APPLICATIONS USING COMPLEX EVENT
PROCESSING

APPROVED BY:

Prof. Dr. Şebnem Baydere

(Thesis Supervisor)

(Yeditepe University)

Prof. Dr. Gürhan Küçük

(Yeditepe University)

Prof. Dr. Suat Özdemir

(Hacettepe University)

Assist. Prof. Dr. İsmail Arı

(Özyeğin University)

Assist. Prof. Dr. Tacha Serif

(Yeditepe University)

DATE OF APPROVAL: /.... /20...

I hereby declare that this thesis is my own work and that all information in this thesis has been obtained and presented in accordance with academic rules and ethical conduct. I have fully cited and referenced all material and results as required by these rules and conduct, and this thesis study does not contain any plagiarism. If any material used in the thesis requires copyright, the necessary permissions have been obtained. No material from this thesis has been used for the award of another degree.

I accept all kinds of legal liability that may arise in case contrary to these situations.

Name, Last name

Signature

ACKNOWLEDGEMENTS

This thesis was supported by TUBITAK under the grant number 119N049.

It is with immense gratitude that I acknowledge the support and help of my supervisor Prof. Sebnem Baydere. Her support and infinite tolerance were a big motivation for me.

I thank members of Things Network Laboratory, especially Baris Inanc and Kerem Irgan for their support, and also Soner Eren, Ahmet Ayyildiz, Baris Inanc, Cem Onel for helping in development of this thesis.

Finally, I would like to thank my family for their endless love and support, which makes everything more beautiful.

ABSTRACT

DATA TRAFFIC MANAGEMENT IN IOT APPLICATIONS USING COMPLEX EVENT PROCESSING

With the proliferation of heterogeneous services in IoT, the cloud of things has begun to face latency-based network challenges. As a solution, the fog of things (FoT) paradigm has emerged. FoT promises numerous advantages for real-time IoT services by offloading computation from the cloud to resource-constrained edge networks. Therefore, to utilize these advantages, an effective data traffic management and resource provisioning approach is needed. This thesis addresses the following three problems associated with IoT data traffic management in a scalable FoT system from the perspective of latency-sensitive applications. Problem-1: an efficient data traffic scheduling mechanism is needed to support multiple services having different QoS characteristics where the decision outputs of these services are transmitted with multi-priority traffic patterns. Problem-2: transmission delay faced on reliable data transfer between fog components should be reduced to experience better QoS. Problem-3: a sensor data freshness model is needed to understand and achieve stable integration of FoT services. To this end, the contributions of the thesis are threefold; First, a CEP-based scheduling policy is proposed for the mitigation of the starvation observed in multi-level queue design. In the proposed approach, arrival-service model of multi-priority FoT data traffic is given using finite-size multi-level waiting lines and then arrival-service waiting parameters are derived. The proposed scheduling policy is compared with FIFO and MPDQ policies through a comprehensive analysis of the waiting times and wait-time gap characteristics. Second, a message queue-based data transfer method is studied to reduce round trip delays between the components of the fog system. A comprehensive analysis of the queue-based messaging model and the conventional subscribe-update-notify-based data flow approach is given in terms of timing constraints and delay characteristics. In the analysis, the factors that result in data transmission latency are examined by formulating end-to-end latency faced in sensor and decision data for both methods. Third, a stateful freshness model is devised to quantify the sensor data freshness rate required by the

application. An analysis for the behavior of the overall system in the context of data freshness is given and the proposed method is compared with the state-of-the-art stateless data freshness time calculation methods. Overall, comprehensive tests are conducted in simulations and on real a testbed environment as a proof of concept for the proposed solutions. The results reveal that the proposed solutions provide significant contributions to the FoT design in the given setup. It is also shown that, the proposed scheduling policy reduces the waiting time gaps and latency for a fog system scaling up to 800 clients and 256 sensor devices.



ÖZET

KOMPLEKS OLAY İŞLEME KULLANARAK IOT UYGULAMALARINDA VERİ TRAFİĞİ YÖNETİMİ

IoT’de heterojen hizmetlerin yaygınlaşmasıyla birlikte nesnelerin bulutu, gecikmeye dayalı ağ sorunlarıyla yüzleşmeye başlamıştır. Çözüm olarak nesnelerin sisi (FoT) paradigması ortaya çıkmıştır. FoT, hesaplamayı buluttan kaynak kısıtlı uç ağlara devrederek gerçek zamanlı IoT hizmetleri için sayısız avantaj vaat etmektedir. Bu nedenle, bu avantajlardan yararlanmak için etkin bir veri trafiği yönetimi ve kaynak sağlama yaklaşımına ihtiyaç vardır. Bu tez, gecikmeye duyarlı uygulamalar açısından ölçeklenebilir bir FoT sisteminde IoT veri trafiği yönetimi ile ilgili aşağıdaki üç sorunu ele almaktadır. Problem-1: Bu servislerin karar çıktılarının çok-öncelikli trafik kalıpları ile iletildiği farklı QoS özelliklerine sahip çoklu servisleri desteklemek için, verimli bir veri trafiği çizelgeleme mekanizmasına ihtiyaç vardır. Problem-2: Sis bileşenleri arasında güvenilir veri transferinde karşılaşılan iletim gecikmesi, daha iyi QoS deneyimi için azaltılmalıdır. Problem-3: FoT servislerinin kararlı entegrasyonunu anlamak ve elde etmek için bir sensör veri tazeliği modeline ihtiyaç vardır. Bu amaçla, tezin katkıları üç yönlüdür; İlk olarak, çok seviyeli kuyruk tasarımında gözlemlenen açlığın azaltılması için CEP tabanlı bir zamanlama politikası önerilmiştir. Önerilen yaklaşımda, sonlu boyutlu çok seviyeli bekleme hatları kullanılarak çok öncelikli FoT veri trafiğinin varış-hizmet modeli verilmiş ve ardından varış-servis bekleme parametreleri türetilmiştir. Önerilen çizelgeleme politikası, bekleme süreleri ve bekleme süresi boşluğu özelliklerinin kapsamlı bir analizi yoluyla FIFO ve MPDQ politikalarıyla karşılaştırılmıştır. İkinci olarak, sis sisteminin bileşenleri arasındaki gidiş-dönüş gecikmelerini azaltmak için mesaj kuyruğuna dayalı bir veri aktarım yöntemi incelenmiştir. Kuyruğa dayalı mesajlaşma modelinin ve geleneksel abone ol-güncelle-bildir-temelli veri akışı yaklaşımının kapsamlı bir analizi, zamanlama kısıtlamaları ve gecikme özellikleri açısından verilmiştir. Analizde, her iki yöntem için sensör ve karar verilerinde karşılaşılan uçtan uca gecikme formüle edilerek veri iletim gecikmesine neden olan faktörler incelenmiştir. Üçüncüsü, uygulamanın gerektirdiği sensör veri tazelik oranını

ölçmek için durum bilgisi olan bir tazelik modeli tasarlanmıştır. Veri tazeliği bağlamında tüm sistemin davranışı için bir analiz verilmiş ve önerilen yöntem, mevcut durumsuz veri tazeliği zaman hesaplama yöntemleri ile karşılaştırılmıştır. Genel olarak, kapsamlı testler, önerilen çözümler için bir kavram kanıtı olarak simülasyonlarda ve gerçek bir test ortamı ortamında gerçekleştirilmiştir. Sonuçlar, önerilen çözümlerin, verilen test ortamında FoT tasarımına önemli katkılar sağladığını ortaya koymaktadır. Ayrıca önerilen çizelgeleme politikasının 800 istemciye ve 256 sensör cihazına kadar ölçeklenebilen bir sis sistemi için bekleme süresi boşluklarını ve gecikmeyi azalttığı da gösterilmiştir.



TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
ABSTRACT	v
ÖZET	vii
LIST OF FIGURES	xii
LIST OF TABLES	xvi
LIST OF SYMBOLS/ABBREVIATIONS	xvii
1. INTRODUCTION	1
1.1. CLOUD BASED INTERNET OF THINGS SOLUTION	2
1.2. FOG COMPUTING BASED INTERNET OF THINGS.....	3
1.3. PROBLEMS IN FoT WITH THESIS PERSPECTIVE.....	4
1.3.1. Contributions of Thesis	5
1.4. ORGANIZATION OF THESIS	7
2. BACKGROUND INFORMATION	8
2.1. INTERNET OF THINGS	8
2.1.1. General Characteristics of IoT Systems.....	8
2.1.2. General Characteristics of IoT Data.....	9
2.2. COMPLEX EVENT PROCESSING.....	10
2.2.1. Definition	10
2.2.2. Terms used for CEP.....	10
2.2.3. Layers in CEP Operation.....	11
2.2.4. Key Characteristics of CEP Engines	12
2.2.5. Key Characteristics of CEP Systems Under Uncertain Data.....	12
3. RELATED WORK.....	13
3.1. APPLICATION PROTOCOLS USED FOR INTERNET OF THINGS	13
3.2. STATE OF ART FOR CEP APPROACHES	14
3.2.1. Categories of Event Processing Applications	14
3.3. FoT DATA TRAFFIC SCALABILITY AND QUEUING.....	16
3.4. FoT DATA TRAFFIC FLOW ORCHESTRATION	18
3.5. SENSOR DATA FRESHNESS	20
4. PUBLISH/SUBSCRIBE BASED FoT ARCHITECTURE	21

4.1. ARCHITECTURAL SCHEME	21
4.2. COMPONENTS OF THE ARCHITECTURE	23
4.2.1. Context Broker	23
4.2.2. Fog Connector	23
4.2.3. Publish/Subscribe Machine	24
4.3. INTERACTION SCHEME OF ARCHITECTURAL COMPONENTS	25
4.4. OPERATING P/S SCHEME	26
4.4.1. Functional Parts in P/S Machine	27
4.4.2. Interaction Scheme for P/S Model	28
4.4.3. Client Rule Structure	29
4.4.4. Client Rule Examples	31
4.4.5. Derivation of Decisions	32
4.5. QoS FOR FoT ARCHITECTURE DETAILS	34
4.5.1. Data Traffic Classification	35
5. DATA TRAFFIC ANALYSIS	36
5.1. DATA TRAFFIC EVENT MODEL AND DATA TYPES	36
5.1.1. Primitive and Complex Event Model	36
5.1.2. IoT Data Types	37
5.2. SENSOR DATA TRAFFIC ANALYSIS	38
5.2.1. Sensor Data Freshness	38
5.3. DATA TRAFFIC FLOW IN FoT ARCHITECTURE	43
5.3.1. Processing and Data Transmission Delays	44
5.3.2. Latency Analysis	46
5.4. DECISION DATA TRAFFIC SCHEDULING	48
5.4.1. FIFO Scheduling	48
5.4.2. Non-preemptive MPDQ Scheduling	48
5.4.3. CEP based FIFO Scheduling	49
6. QUEUING ANALYSIS	51
6.1. MODEL DEFINITION	51
6.2. OPTIMAL QUEUE SIZE ANALYSIS	57
7. TESTS AND RESULTS	59

7.1. TESTBED SETUP	59
7.1.1. Real Testbed Setup	59
7.1.2. Emulation Based Setup	59
7.2. INTERARRIVAL TIME ANALYSIS	61
7.3. SENSOR DATA FRESHNESS TESTS	67
7.3.1. Standard Deviation Based Freshness Time Analysis	71
7.3.2. Probabilistic Data Freshness Time Analysis	72
7.3.3. Evaluation of Freshness Time Analysis	72
7.3.4. Effect of Client Period	72
7.4. FOG DATA TRAFFIC TESTS	75
7.4.1. RTT Tests	75
7.4.2. Data Traffic Latency Tests	75
7.5. QUEUE MODEL TESTS	85
7.5.1. Service Time Analysis	85
7.5.2. Performance of FIFO Queuing System	87
7.5.3. Performance of MPDQ Queuing System	89
7.5.4. Performance of CEP Approach	91
8. CONCLUSIONS	94
REFERENCES	96

LIST OF FIGURES

Figure 4.1.	Architectural scheme	22
Figure 4.2.	Interaction between components.....	25
Figure 4.3.	General message type	29
Figure 4.4.	Structure of observation start message	29
Figure 4.5.	Structure of client subscription message	30
Figure 4.6.	Structure of publish message in PSM	30
Figure 4.7.	Structure of PSM initialization message.....	30
Figure 4.8.	Client Rule Structure	32
Figure 4.9.	Example client identifier	33
Figure 4.10.	Example decision making alert.....	33
Figure 4.11.	Example decision criteria	33
Figure 4.12.	Example decision criteria 2.....	33
Figure 4.13.	Example physical area definition	34
Figure 5.1.	Detailed freshness parameters illustration.....	41
Figure 5.2.	Static freshness times with client periods.	42
Figure 5.3.	Timing diagram for sensor data transmission with scenario 1.....	43
Figure 5.4.	Timing diagram for decision data transmission with scenario 1.	44
Figure 5.5.	Timing diagram for sensor data transmission with scenario 2.....	44
Figure 5.6.	Timing diagram for decision data transmission with scenario 2.	45
Figure 6.1.	Change of $r_{wait_{k,l}}$ respect to ρ when $K=2$ or INF , $\rho \leq 0.9$, for 2 levels	55

Figure 6.2.	Change of $r_{wait_{k,l}}$ respect to ρ when $K=2$ or INF , $\rho \leq 0.9$, for 3 levels	56
Figure 6.3.	Change of $r_{wait_{k,l}}$ respect to ρ when $K=2$ or INF , $\rho \leq 0.9$, for 4 levels	56
Figure 6.4.	Change of $\omega(k, \rho)$ with respect to k , when $\rho \leq 0.9$	58
Figure 6.5.	Change of $\omega(k, \rho)$ with respect to k , when $\rho \geq 0.9$	58
Figure 7.1.	IAT samples when $P=2s$ and $N=1$	62
Figure 7.2.	IAT samples when $P=2s$ and $N=2$	62
Figure 7.3.	IAT samples when $P=2s$ and $N=4$	63
Figure 7.4.	IAT samples when $P=2s$ and $N=8$	63
Figure 7.5.	IAT occurrence histogram for $P=2s$ and $N=1$	63
Figure 7.6.	IAT occurrence histogram for $P=2s$ and $N=2$	64
Figure 7.7.	IAT occurrence histogram for $P=2s$ and $N=4$	64
Figure 7.8.	IAT occurrence histogram for $P=2s$ and $N=8$	64
Figure 7.9.	IAT samples when $P=5s$ and $N=1$	65
Figure 7.10.	IAT samples when $P=5s$ and $N=2$	65
Figure 7.11.	IAT samples when $P=5s$ and $N=4$	65
Figure 7.12.	IAT samples when $P=5s$ and $N=8$	66
Figure 7.13.	IAT occurrence histogram for $P=5s$ and $N=1$	66
Figure 7.14.	IAT occurrence histogram for $P=5s$ and $N=2$	66
Figure 7.15.	IAT occurrence histogram for $P=2s$ and $N=4$	67
Figure 7.16.	IAT occurrence histogram for $P=5s$ and $N=8$	67
Figure 7.17.	Freshness rates of static freshness time method for $P=2s$ and Zigbee.....	68
Figure 7.18.	Freshness rates of static freshness time method for $P=2s$, $N=8$ and Zigbee	69

Figure 7.19. Freshness rates of static freshness time method for $P=5s$ and Zigbee.....	69
Figure 7.20. Freshness rates of static freshness time method for $P=5s$, $N=8$ and Zigbee.	70
Figure 7.21. Freshness rates of static freshness time method for $P=2s$ and WiFi.	70
Figure 7.22. Freshness rates of static freshness time method for $P=5s$ and WiFi.	71
Figure 7.23. Freshness rates of several freshness time methods for $P=2s$	73
Figure 7.24. Freshness rates of several freshness time method for $P=5s$	73
Figure 7.25. Freshness rates of RTO-like freshness time calculation method for $P=2s$.	74
Figure 7.26. Freshness rates of RTO-like freshness time calculation method for $P=5s$.	74
Figure 7.27. Histogram of $D_i^{s,B}$ values when $N=1$, for HTTP-based scenario.....	77
Figure 7.28. Histogram of $D_i^{s,B}$ values when $N=1$, for MQ-based scenario.....	77
Figure 7.29. Histogram of $D_i^{s,B}$ values when $N=4$, for HTTP-based scenario.....	77
Figure 7.30. Histogram of $D_i^{s,B}$ values when $N=4$, for MQ-based scenario.....	77
Figure 7.31. Histogram of $D_i^{s,B}$ values when $N=16$, for HTTP-based scenario	78
Figure 7.32. Histogram of $D_i^{s,B}$ values when $N=16$, for MQ-based scenario	78
Figure 7.33. Histogram of $D_i^{s,B}$ values when $N=64$, for HTTP-based scenario	78
Figure 7.34. Histogram of $D_i^{s,B}$ values when $N=64$, for MQ-based scenario	78
Figure 7.35. Histogram of $D_i^{s,B}$ values when $N=256$, for HTTP-based scenario	79
Figure 7.36. Histogram of $D_i^{s,B}$ values when $N=256$, for MQ-based scenario	79
Figure 7.37. Normal distribution curve when $N=1$	79
Figure 7.38. Normal distribution curve when $N=16$	80
Figure 7.39. Normal distribution curve when $N=256$	80
Figure 7.40. Change in average $D^{s,B}$ and $D^{s,M}$ values with increasing N and M	82

Figure 7.41. Change in average $\Delta^{s,B}$, $\Delta^{s,M}$ values with increasing N and M	82
Figure 7.42. Latency Difference with increasing N	83
Figure 7.43. Change in average $D^{s,B}$ and $D^{s,M}$ values with increasing N and S	83
Figure 7.44. Change in average $\Delta^{s,B}$, $\Delta^{s,M}$ values with increasing N and S	84
Figure 7.45. Latency Difference with increasing S	84
Figure 7.46. Service time distributions when $M=100$	86
Figure 7.47. Service time distributions when $M=400$	87
Figure 7.48. Service time distributions when $M=800$	87
Figure 7.49. r_{wait} results when $k=2, 3$ and 5 , for FIFO	88
Figure 7.50. Distribution of r_{wait} when $M = 800$, $k=2$ and 5 , for FIFO	88
Figure 7.51. Distribution of r_{wait} when $M = 800$, $k=2, 3$ and 5 , for FIFO	89
Figure 7.52. r_{wait} results when $k=2$ or INF , for MPDQ.	90
Figure 7.53. Comparison of r_{wait} of MPDQ with FIFO when $k=2$	90
Figure 7.54. Gap of wait times for MPDQ	91
Figure 7.55. Comparison of r_{wait} of MPDQ with CEP when $k=2$	92
Figure 7.56. Gap of waiting times for different priority based schedulers when $k=2$	92
Figure 7.57. Comparison of r_{wait} of MPDQ with CEP when $k=INF$	93
Figure 7.58. Gap of waiting times for different priority based schedulers.	93

LIST OF TABLES

Table 3.1. Some current CEP approaches	15
Table 4.1. Data traffic classes	34
Table 4.2. QoS guarantees in the layers	35
Table 5.1. Data traffic scheduling approaches	50
Table 6.1. Key System Parameters.....	52
Table 6.2. Optimal Queue Capacities	57
Table 7.1. Test Parameters	61
Table 7.2. Average RTT values obtained in the first phase of tests.	76
Table 7.3. Average CPU and Memory Utilization of HTTP method	81
Table 7.4. The traffic rates according to number of clients	86

LIST OF SYMBOLS/ABBREVIATIONS

IoT	Internet of things
FoT	Fog of things
CoT	Cloud of things
QoS	Quaility of service
FIFO	Frst-in-first-out
MPDQ	Multi priority discipline queue
CEP	Complex event processing
EPL	Event processing language
EPA	Event processing agent
EPN	Event processing network
CoAP	Constrained application protocol
REST	Representational state transfer
HTTP	Hyper text transfer protocol
WSN	Wireless sensor network
MQTT	Message queue telemetry transfer
TCP/IP	Transmission control protocol / internet protocol
UDP	User datagram protocol
JSON	Java script object notation
XML	Extensible markup language
M2M	Machine to machine
D2D	Device to device
NJN	Nearest job next
AQM	Active queue management
CPU	Central processing unit
RAM	Random access memory
CCN	Content centric network
DIS	Data integration service
P/S	Publish/subscribe

SC	Sensor connector
CB	Context broker
MQ	Message queue
MQP	Message queue protocol
URI	Universal resource indicator
LAN	Local area network
PSM	Publish/subscribe machine
FC	Fog connector
CCS	Client connection server
SDA	Sensor data adapter
SA	Sensor agent
ED	Event detector
UA	User agent
SLA	Service level agreements
IAT	Inter arrival time
N	Number of sensor nodes in edge
P	Sensor data acquisition period
M	Number of clients subscribed to the system
CP	Client defined decision sending period
SP	Sensor data sampling period
FT	Data freshness time span
FR	Data freshness rate
RTT	Round trip time
RTO	Retransmission timeout

1. INTRODUCTION

IoT paradigm is used for building the technical backbone required to bring the Internet to anything, anytime and anywhere, and refers to data exchange through this interconnection network [1, 2]. Internet of the real world requires the information autonomously obtained by the "things" which are the sources of actions and events of the physical world [3, 4]. Therefore, this cutting-edge concept utilizes technologies with a tight connection to the physical world. With this concept, the growing number of physical objects connect to the internet for sharing information and coordinate each other [3, 5, 6]. Hence, IoT creates a smart environment for everyday physical objects and it provides the ways of inter-networking for those objects with less human intervention [1, 3, 7].

Researchers consider IoT as an ecosystem in which physical objects produce valuable information and communicate with each other and even with human-beings using the internet [8–10]. These physical objects have sensors, actuators, the data transmission circuitry, and the software which combines these hardware units for collection of data, make and react on several decisions [3, 11]. With the application of this paradigm, new digital services, applications for the control of the daily objects have emerged in the virtual world for the improvement of the quality of our life [6, 12].

An IoT network consists of generic multipurpose internet devices that perform physical data based heterogeneous services and applications [3, 13, 14]. In order to provide a complete observation system, these devices interoperate with remote observation clients in WANs, via interconnection devices which are fundamental inter-operation parts in IoT. In addition, these devices should apply services with different Quality of Service (QoS) requirements and utilize sensors that have different states of observation in the same time period [15]. This heterogeneity involves the different data characteristics in terms of volume and variety, as well as various processing capabilities [3]. Hence, over time, it becomes very hard to manage things with a single platform having limited computing resources. So, IoT begins to need sufficient networking and computing infrastructure for meeting low latency and fast response expectations from itself [16].

In the design of IoT infrastructure, for facilitating the required operations, researchers follow various architectural models in which similar functionalities are grouped into the same layer [5]. The most common architectural model used for an IoT system is composed of three layers [17]. This model has the edge, cloud and fog layer. The edge layer is composed of sensor and actuator devices. These devices are typically resource-constrained for data processing and they are useful in sensing or actuation of a certain physical phenomenon. Cloud layer is the centralized remote network which has relatively more powerful devices for data processing and storage [5]. The devices in the cloud layer are capable to process complex data analytic which could be problematic for the edge layer [18]. So, cloud layer serves complex services needed for IoT applications. Located on closed proximity of the edge, fog layer is the local distributed network of the nodes (i.e., fog nodes) having the adequate capacity for data storage and complex data analytic processing [19]. A fog node is relatively limited than a cloud node in the terms of computational power, whereas a fog node is more powerful than a node located on the edge. So, the delay-sensitive applications and services are performed in the fog layer in order to get the data analytic result in a relatively short time.

1.1. CLOUD BASED INTERNET OF THINGS SOLUTION

For solving the performance-related problems of edge layer of IoT without violation of resource management and processing power related QoS requirements, the IoT vendors adapt the cloud to support the edge for IoT data analytics which require big data processing [3, 18, 20]. This concept is called CoT and it is useful in generating reactions for different IoT based events, with using the decision outputs of the complex data analytic [21]. Although CoT has the power of processing and brings advantages for IoT [1] and it is considered as a successful IoT services provider which gives reasonable cost/benefit ratio [18, 22, 23], it is geographically centralized [3] and cloud-based IoT systems face several network-related problems [24, 25] caused by the high load of sensor data packet traffic over interconnection network, unreliable and high latency, high response time, with an increasing variety and number of things connected to itself [26–29]. So, CoT could not potentially guarantee latency and network connection centric QoS requirements even

over a central resource-rich computational platform as the number of IoT devices increase and proliferation of heterogeneous real-time reactive services which needs low latency requirements [1, 30, 31]. It is necessary to tackle these network-related problems with a geographically decentralized and location-aware approach in which we can apply the processing and computing capabilities of cloud computing near the edge of IoT devices [29]. So, it is possible to serve real-time and latency-sensitive IoT applications [27] and obtain more complex vital, and smarter decisions [32] from a high load of sensor data nearly in real-time without high and unreliable delays faced.

1.2. FOG COMPUTING BASED INTERNET OF THINGS

To overcome high latency problems faced in the cloud computing based IoT and for providing the guarantee of real-time processing and latency based QoS requirements, as well as for addressing the networking bottlenecks [33–35], a geographically decentralized, more closed-proximity and latency-aware computing paradigm, named as fog computing has emerged [5, 36, 37]. This concept brings the power of cloud computing to the closer of the physical phenomenon, with the name of Fog of Things (FoT) [19], where the edge of IoT devices and local device pool for storage and data processing exists, in order to guarantee low latency for the processing of huge amounts of sensor data [38–40]. Therefore, rather than transmission of vast amounts of IoT data to the cloud, fog located services effectively make local decisions about the situations by using the load of sensor and historic data which are located on the locals of the edge [41]. If possible, these services react to those situations with local and delay-sensitive actions without need of further data traffic onto the cloud [42, 43], else, they deliver the filtered or aggregated sensor data or decision data to the cloud for further data analysis, reactions or data integration which could be done with a centralized services on cloud and or are beyond the locals of the fog [33]. Hence, fog computing is a crucial tier for IoT data processing near to the edge of IoT devices, and it guarantees the aforementioned QoS challenges whose solutions are potentially problematic with the cloud computing for a scalable IoT system [44]. Besides, current fog computing solutions adapt several standardized data traffic management mechanisms for dynamic orchestration of data availability and processing of geo-distributed microservices [28]. Moreover, these solutions

provide the ways of effective autonomous IoT device maintenance in the fog-level without the intervention of third parties in the cloud [45].

1.3. PROBLEMS IN FOT WITH THESIS PERSPECTIVE

The following gives the several issues for the FoT which we deal in this thesis.

1. Problem-1 : Although FoT is advantageous for the latency-sensitive IoT services design, finding an effective approach for latency based QoS aware IoT data traffic scheduling is still a crucial challenge for FoT. This is more critical when the data traffic scheduling mechanism favors higher priority packets and perpetually denies lower priority packets to utilize the available network bandwidth. Thus, such a mechanism may cause lower-priority packets to suffer from higher wait times in the fog system, and consequently starve for the network resources. Mitigation of the resource starvation problem needs to be addressed even in a small fog system as it may support diversified IoT services with its limited resources. Therefore, in a scalable fog system of heterogeneous IoT applications, modelling the data traffic generated from IoT devices [3, 46] becomes more critical and we need to devise an effective scheduling approach for the management of the outgoing multi-priority IoT data packet traffic.
2. Problem-2 : For the data traffic management domain of FoT, there is the lack of the effective method for the management of the reliable data transfer in the inner fog [17]. Such a technique is a requirement on the communication backbone for reducing intra-fog reliable data transfer latency to experience better QoS both from time and data-reliability sensitive IoT application scenarios [45]. Although it is very critical to save even a short time for time-sensitive IoT applications, conventional data traffic orchestration approaches potentially result in high latencies in a scalable fog architecture, since they prefer to use transient and connectionless messaging models for reliable data transfer [28]. Thus, for achieving higher QoS, it is critical to optimize the data transfer performance by considering the fundamental steps of reliable data transfer performed in a scalable fog. Besides, such a method should be aware of

latency based QoS requirements of horizontally distributed fog services.

3. Problem-3 : The emergency observation process may generate huge sensor data traffic on the IoT infrastructure, especially on the interconnection devices. Increase in the number of sensing devices deployed for an observation system is an important drawback for FoT based real-time emergency observation. Thus, in order to meet multiple clients' SLA, real-time requirements of the system and achieve the required service quality in a scalable fog, a cache based time stable and periodic emergency decision support mechanism could be preferred on the fog architecture [47]. Such systems generally favor a service-centric asynchronous push / pull data integration scheme for scheduling, resulting an emergency decision to be marked as old [47]. Thus, data freshness rate becomes a crucial traffic modeling parameter to be observed in these systems and there is a need to examine freshness time calculation methods in terms of achievable freshness rates.

1.3.1. Contributions of Thesis

Considering the problems defined above, the following describes the appropriate solutions and the contributions made in the thesis.

1. Solution-1 : We address Problem-1 by modelling the arrival-service process of the multi-priority IoT data packet traffic generated from a scalable fog architecture. This model has its basis on finite-size multi-priority-discipline based queuing theory [48]. By studying this model, we examine the FIFO and MPDQ based scheduling policies and reveal the pros and cons of them, considering the potential starvation problem. Using the outcomes obtained from this model study, we propose an efficient packet scheduling approach. This approach tackles the starvation problem that may occur in low priority queues. In its scheduling mechanism, our approach uses batch windowing based CEP. Since CEP has been proven to provide solutions for timely and low overhead event pattern recognition and making reactions [49], the required efficiency can be achieved. For the proof of the concept, we compare the proposed policy with First-In-First-Out and Multi-Priority-Discipline Queue policies through a

comprehensive analysis of waiting times and wait-time gap characteristics. We have also conducted extensive simulation tests to explore the performance of these policies in our testbed for up to 800 clients communicating with the fog system. The results reveal that the empirical values obtained from the tests verify the theoretical model and the proposed approach can successfully relieve the waiting time gaps observed in priority levels. The proposed theoretical model, the evaluation and the outcomes of this study is accepted as a journal paper publication [50].

2. Solution-2 : We tackle Problem-2 by examining a persistent and connection-oriented message queue based data transfer method in the scalable fog architecture. This method has several benefits in terms of fast and reliable data transfer and, it improves latency based QoS for data transmission in a scalable fog design. We compare the proposed messaging model with a conventional subscribe-update-notify based data flow approach for the orchestration of reliable data transfer. We provide a comprehensive analysis of timing constraints and delay characteristics of both methods. We then present the performance results of extensive simulation scenarios for up to 100 clients communicating with 256 sensors simultaneously over the proposed platform. The results reveal that the message queue-based approach successfully relieves the latency cost of reliable data transmission in the inner fog. The outcomes of this study is prepared as a journal paper.
3. Solution-3 : In this thesis, we tackle Problem-3, firstly by providing a Publish-Subscribe and, time stable and periodic decision making scheme suitable for a scalable FoT architecture. Secondly, we analyze the sensor data traffic and decision data freshness characteristics of a such FoT based emergency observation system. With using such scheme, we devise an analysis for understanding the behavior of the overall system in the context of data freshness. The results obtained from our experimental setup illustrate the appropriateness of freshness time calculation methods for obtaining the required service quality. Two conference and one chapter papers which describe the method and its evaluation are accepted as publication [11,51,52].

In overall, in this thesis, we aim to build a scalable FoT architecture supported with an end-to-end interconnection based multi-user and heterogeneous sensor devices supported

scalable FoT architecture, its data integration scheme and IoT services platform in it. In this architecture, we integrate the given solutions and provide efficient FoT based services platform.

1.4. ORGANIZATION OF THESIS

Chapter 2 covers the background information needed for the thesis with focusing on an overview of IoT data and the system characteristics and key concepts of the CEP. Chapter 3 reviews the state of art for the thesis. Chapter 4 introduces the internals of the Publish/Subscribe based FoT architecture in which we realize our proposed concepts for the thesis. This chapter elaborates on the design scheme, the functions, interactions of the components in the architecture, and how P/S pattern is operated to make decisions. Additionally, this chapter gives the details of the IoT data traffic classes for this architecture. Chapter 5 presents the details of the data traffic analysis in the given FoT architecture developed for our thesis. Chapter 6 explores the details of the analysis for the limited capacity and multi level queues used in decision data traffic scheduling. In Chapter 7, we present the experiments for proof-of-concept and the evaluate the outcomes for the tests. Lastly, we conclude the thesis with giving the summary of the study and its outcomes.

2. BACKGROUND INFORMATION

This chapter presents the background information and gives the details about Internet of things and Complex Event Processing considering data traffic management domain. This chapter elaborates on the Internet of Things with investigating general characteristics of it and the IoT data. Then it examines CEP, how it operates and its key characteristics under uncertain data.

2.1. INTERNET OF THINGS

This section elaborates on Internet of Things with giving the details of its general characteristics and the details of the features of IoT data.

2.1.1. General Characteristics of IoT Systems

- **Resource Constrained Environment:** The things in IoT generally suffer from a severe lack of resources. Therefore, the computational and storage capabilities is restricted [53]. So, transmission of data generated from things to a more powerful devices located on a central network (i.e., CoT) can be preferred in the services design and data analytic [54].
- **Volume of data:** A typical IoT system generates data at enormous rates and easily overwhelms a central system that collects the data for processing [27, 54]. This situation can easily result that centralized data integration systems may not handle those generated data in real-time manner [53, 55]. For the solution of this problem, an intermittent connectivity technique, such as FoT, can be used to augment the IoT services performed by a centralized cloud system [53].
- **Heterogeneity in Sources:** Different devices attached to any one of the IoT system put down IoT data of variety in type, size and formation [56].
- **Self-configuration:** Devices in an IoT System have self-configuring capability in association with infrastructure, in a manner with minimal manual or user intervention [56].

- **Interconnection with Low Power Personal Area Networks (LoWPANs):** Devices in an IoT system support interconnection protocols and communicate with devices which monitors physical conditions and also with infrastructure [56]. In order to support an heterogeneous edge network containing devices supporting different communication protocols for a fog network, the IoT system should support an adaptation mechanism for data transformation.
- **Integrated into information network:** Devices in a typical IoT network interact with the information network which allows them to intercommunicate with other devices and systems [53]. Several protocols could be employed for the operations that other devices and user applications can dynamically discover those devices in the network, let those devices to describe their characteristics, send data with each other or information network and for management purposes [56].

2.1.2. General Characteristics of IoT Data

- **Uncertain, errorneous and noisy:** Considering the characteristics of IoT Systems presented in Section 2.1.1 and existence of interconnection gateways between LoWPANs and Internet, generated sensor data in IoT are considered to be inherently uncertain and errorneous [54]. This is the reason of the limited environment of IoT and signal disturbance in IoT networks [57].
- **Periodicity:** Dataset related to many IoT data integration systems may present an inherent periodic pattern where the same value occur at specific intervals [54]. Sampling is used primarily to achieve energy-efficiency because the monitored phenomenon does not often change in sudden. A duty cycling process could be applied. Considering this feature, application of Publish/Subscribe and Update/Notify based architectural pattern is more suitable for IoT [58].
- **Continuity:** Data produced from a physical phenomenon is continuous even a sampling strategy is adapted [54].

2.2. COMPLEX EVENT PROCESSING

In this section, we provide the brief definition of CEP, the terms used commonly for CEP, layers of CEP Operation, key characteristics of CEP Engines, key characteristics of CEP Systems Under Uncertain Data and categories of CEP Applications.

2.2.1. Definition

CEP can be defined briefly as the process of recognition of valuable information or a meaningful situation from stream of huge amount of events and reaction to those detected situations in real time manner [57]. CEP recognizes occurrence of any phenomenon based on event detection rules [59].

2.2.2. Terms used for CEP

A collection of terms to define CEP operation is given below.

- **Primitive Event:** Primitive event is atomic occurrence or change of interest within a system or domain [59]. For the operation of a CEP system, those programming entity should be defined clearly.
- **Complex Event:** Complex Event is combination of primitive events or other complex events [60].
- **Event Processing Rule:** Event processing rules transform low level primitive events into more complex events in order to gain more insight into the current state of the environment [60]. Such a rule is consisting of two parts [61]: a condition part with event patterns and an action part.
- **Event Processing Language (EPL):** EPL is language to express the event processing rules and logic of a CEP Operation [60,61].
- **Event Processing Agent (EPA):** EPA is a software module that applies a set of logic to generate complex events as output for a set of input of primitive events [62]. In process of EPA, event correlation and event aggregation is detected according to specific rules predefined by users. These rules are combination of event operators [60].

Users can define his/her rules using some EPL designed for CEP.

- **Event Processing Network (EPN):** EPN is network of collection of EPAs [57].
- **CEP Engine:** A CEP engine is the combination of EPAs and analyzes the stream of continuously incoming events and executes the matching rules [62].
- **Event Source:** Event source is the creator of the primitive event. Event source provides event for the complex event processing cycle [62].
- **Event Handlers:** An event handler is the operational part in a CEP engine which reacts to a determined complex event.

2.2.3. Layers in CEP Operation

The operations of a typical CEP engine can be divided into some layers defined below.

- **Event Source Layer:** This layer is responsible for capturing the primitive events in internal and external environments. The access to the different event sources is implemented in this layer [61]. This layer creates event stream for event processing layer.
- **Event Processing Layer:** The event processing layer is responsible for the core processing of continuously arriving event streams using CEP. EPAs are the main processing units of this layer. The incoming technical and low level events in event stream are filtered, aggregated, correlated, and synthesized to a higher level of abstraction with the assistance of rules defined by CEP user [61]. By accessing local databases or external services events can be enriched with domain data. If a significant situation is detected in the event stream that requires an suitable reaction, the event handling layer is invoked.
- **Event Handling Layer:** The appropriate reactions on matched event patterns detected by CEP are performed in this layer [61]. In most cases the occurrence of an event pattern triggers the invocation of a business process. Another possible reaction is the notification of a human expert on a graphical user interface or by visualization of the system state.

2.2.4. Key Characteristics of CEP Engines

The key characteristics of the CEP engines are summarized as follows [63]:

- **Event Type:** A CEP Engine takes a continuous and infinite stream of primitive events as input from the external sources. CEP system should be able to do a real-time processing and event detection of a vast number of events with low latency.
- **Event Source Type:** A CEP Engine is capable of filtering, correlating, and aggregating data from multiple external sources to infer the high-level events and semantic information between the events.
- **Correlation of Events:** A CEP Engine mainly correlates the strong temporal relationships between the incoming events and their patterns.
- **In-Memory Pattern Matching:** A CEP engine is designed to handle event stream directly in memory.
- **Processing of Big Data:** A event processing application should process a large volume of data due to pervasive use of IoT networks, smart phones and social networks.

2.2.5. Key Characteristics of CEP Systems Under Uncertain Data

Uncertainty in data is defined as a lack of precision in data with incomplete knowledge of the information that leads to a complex situation to describe the possible outcome [63]. The uncertainty is the result of insufficient, imprecise and vague information in data.

Data generated in an IoT system is a typical uncertain data as expressed in Section 2.1.1 and 2.1.2.

There are several ways for handling the uncertain data. Firstly, the uncertainty is usually treated with probabilities in CEP Engines [57]. Secondly, a data cleansing process could be applied before application of CEP process.

3. RELATED WORK

This chapter presents the state of the art for the thesis with mainly examining the studies of the application protocols needed for IoT, studies of CEP approaches and the studies related with the proposed approaches in our thesis. In this chapter, we compare the proposed approaches with the conventional approaches in the literature.

3.1. APPLICATION PROTOCOLS USED FOR INTERNET OF THINGS

This subsection presents the commonly used protocols and standards for the communication, messaging in Internet of Things.

CoAP (Constrained Application Protocol) [64–66] is a request/response protocol and a REST standard for resource constraint devices. With the help of CoAP, these resource constraint devices can communicate with network nodes which use HTTP messaging [66]. CoAP uses a proxy between constrained environments (e.g. a WSN) and internet. With the help of the CoAP proxy, HTTP server in the internet cloud can communicate with a resource constraint node [67]. A CoAP proxy operates a dual stack for interconnection. CoAP uses UDP as transmission protocol between the gateway and the resource constrained node. Since the data transmission is unreliable, CoAP provides a timeout mechanism for retransmission of lost packets. In addition it supports unicast and multicast request [66].

MQTT protocol [65] provides device data collection service for telemetry applications. It was designed to support publish-subscribe model [66]. Its goal is to collect data from many devices called as publisher and transport that data to the IT infrastructure via brokers [68]. It targets large networks of small devices that need to be monitored or controlled from the cloud. MQTT can run over TCP/IP. In addition, it has a notification mechanism when abnormal situation occurs. However, the MQTT topic must be placed in each message sent to the server which generates an additional overhead in the overall system [51]. To get rid of this drawback, in our P/S based architectural model, sensors layer is unaware of the topics [51]. Our model only requires an identifier of the sensor node in the sensor data

packet to determine the right sensor agent. Therefore, for publish operation, a sensor node in the sensors layer performs only periodic sensing and sending without the need of managing topics in itself.

Websocket protocol supports client/server architecture and streaming capability. This feature provides a basic connection scheme, such as UDP / TCP socket connection, as well as the ability to manage all client connections by a single port. Websocket allows to use a single port for many-to-many connections with low protocol overhead. Websocket is suitable for publish/subscribe architectural design and supports periodicity and continuity characteristics of IoT data. Additionally, when using a protocol that does not support IP; such as zigbee, z-wave, or other no-IP WSNs, the websocket connection structure provides full duplex communication and has convenience in architectural design, since multiple connection information can be transmitted through a single interface stream.

REST is an architecture that provides web services which allow communication and data exchange between different devices using HTTP in IoT environment [65]. In this architecture, server provides resources and access permission to clients. So clients use HTTP messages to access to sources. Different representations such as JSON, XML and text can be used to provide data. It supports request/response messaging model, by using HTTP commands. RESTful HTTP provides easy implementation. So it is used broadly in M2M communication.

3.2. STATE OF ART FOR CEP APPROACHES

Table 3.1 gives some CEP methods with giving data structures used in operation and their structure.

3.2.1. Categories of Event Processing Applications

Some categories of operation of CEP applications can be summarized as follows: [78]

Table 3.1. Some current CEP approaches

CEP Method	Data Structure for CEP Operation	CEP Structure
Yuanping et al. [69]	Tree-based Event Group	Fixed, centralized, do not support uncertainty
Xingyi et al. [70]	Time based Petri Net	Fixed, centralized, do not support uncertainty
ESPER [71]	directed graph	Fixed, centralized, do not support uncertainty
SASE [72]	NFA, AIS (Active Instance Stack)	Dynamic, centralized, do not support uncertainty
Cayuga [73]	NFA with instance lists	Dynamic, centralized, do not support uncertainty
Agrawal et al. [74]	NFA,AIS	Dynamic, centralized, do not support uncertainty
ProbLog [75]	Binary Decision Diagrams	Not a direct CEP Method, fixed, centralized, supports uncertainty
Sadilek et al. [76]	Hybrid MLNS (Markov Logic Networks)	Dynamic, centralized, supports uncertainty
Morariu et al. [77]	MLNS	Dynamic, centralized, supports uncertainty

- **Observation:** CEP is used to monitor a system or process by looking for exceptional behavior and generating alerts when such behavior occurs. In those system, the reaction is left to the consumer of the alert. The patient monitoring system presented in [79] and children surveillance system explained in [80] are examples for CEP-based observation systems.
- **Information dissemination:** CEP is used to deliver the right information to the right consumer at the right granularity at the right time. Personalized banking system and emergency systems are example of information systems.

- **Dynamic operational behavior:** CEP is often used to drive the actions performed by a system dynamically so as to react to incoming events. The online trading system, road tolling system are examples with dynamic operational behavior.
- **Active diagnostics:** A CEP application can diagnose a problem, based on observed symptoms. The mechanical failure detection and a coal mine safety monitoring system [81] are examples of this category.
- **Predictive processing:** Another category for a CEP application is to identify events before they have happened, so that they can be eliminated or their effects mitigated. The fraud detection system is an example to this category.

3.3. FOT DATA TRAFFIC SCALABILITY AND QUEUING

Several works in the literature study on queueing theory based scalability modeling for IoT data traffic management domain. Strielkina et al. [82] introduce a queueing theory based Healthcare IoT data traffic model. This study applies the model to a specific health-care system use-case and models the operation using a queueing system with a restricted number of states in the arrival-service process. Maheswari et al. [83] analyze the performance and scalability of an edge-cloud system for latency-sensitive augmented reality and IoT based applications. This study investigates latency and response time issues in a moderately scalable system and considers single priority application data traffic. Abdelradi et al. [84] strive for a queueing model concerning traffic offloading problems for D2D communication and aims to maximize throughput while maintaining the queue system stable. Their work analyzes arrival and service rates for the corresponding traffic. Oluwadere et al. [85] study a queue model with combining a regression technique for determination the variability in the data traffic flow. However, these studies does not cover the waiting dynamics of the queueing system in details and, the models deal with a single priority data traffic aggregated on an infinite size wait line. Rathod et al. [46] conduct a more comprehensive study on the scalability and queueing model for IoT. This study presents a FIFO based queue model for a fog-cloud based scalable IoT middleware. They consider the request/response interaction pattern and specifically evaluate the system with an increasing workload of requests arrived in a second. Our study has a more specific difference from this study that, we observe

outgoing IoT traffic from the fog by scaling the fog services with an increasing number of subscribed users.

Several data packet scheduling policies for multi-priority IoT data traffic have been investigated and studied for restricted IoT networks in the literature. One of the example studies for WSN is in [86]. This study proposes a multi-priority based scheduling scheme for data collection in WSNs with an alternative approach of the NJN algorithm. Although the authors consider multi-priority data traffic scheduling as critical, the analysis for multi-priority packet scheduling in their study remains restricted and mainly focuses on single-priority packet traffic. Another example study in [87] proposes a traffic scheduling approach designed for the energy usage optimization in Wireless nodes and is applicable for multi-hop IoT infrastructures with multi-class traffic. Although their work relies on a multi-level queue analysis, queue wait time dynamics and starvation problem are not analyzed. Nasser et al. [88] proposes a multi-priority packet scheduling approach designed for WSNs. This study provides a comprehensive analysis of the proposed scheduling approach and compares it with FIFO and Multi-Level Preemptive Queuing policies. Nevertheless, the study does not rely on queuing analysis, and, as the authors state, this study needs further evaluation of the starvation prevention problem that occurs in the lowest level queue.

Data traffic management with multiple waiting lines has been studied, considering the practical aspects of the queue management in the literature. Kua et al. [89] evaluate the fair capacity sharing performance of several proprietary AQM mechanisms for an home broadband network in which heterogeneous IoT traffic flow is induced. Hamdi et al. [90] tackle the congestion problem in large network architectures and evaluate the performance of the several well-known AQM algorithms with fundamental network metrics. Although, these studies provide a comprehensive analysis for the comparison of AQM algorithms, they remain restricted with a static network and they do not focus on the theoretical aspects of the queue management.

Our proposed queuing model and scheduling policy differ from the models in the given state-of-the-art studies in several aspects. **(1)** Rather than using request/response based data traffic delivery that is preferred in most of these model studies, we consider the publish/subscribe

based IoT data traffic, which is more appealing. (2) Since IoT data traffic is heterogeneous and potentially has different priority classes, we use multi-level waiting lines in our queuing model. (3) Our queuing model can analyze the arrival-service-waiting process for an unlimited number of priority levels in the queue. (4) With our queuing model, we examine the queue processing resource starvation characteristics of different data packet scheduling policies. (5) The proposed data traffic scheduling model strives to reduce wait time gaps that occur in multi-priority data scheduling, as it aims to prevent the starvation problem in the lower priority wait lines of the queue.

3.4. FOT DATA TRAFFIC FLOW ORCHESTRATION

For minimizing data transfer latency of the IoT services for a scalable design, fog offloading strategy is offered in several studies. This strategy is based on making a decision that a service request is performed in the fog or the cloud. Several algorithmic models, which are used for this decision, are devised in the literature. Study in [17] proposes an analytical fog offloading model which uses an estimated wait time of the services in the system. The model compares this estimated value with service execution time threshold given by the service user for the offloading decision. Study in [91] uses another fog offloading model that take into account of the workload on the fog nodes for the offloading decision. Balevi et al. [92] offers an offloading approach which finds the optimal number of fog nodes in the network when the network channel condition and total number of the nodes are given. Kattepur et al. [93] deal with another fog offloading approach that is based on optimization of energy and communication latency costs, for resource constrained devices located on the fog. In [94], Maiti et al. study another strategy for fog offloading which considers the distance between fog nodes and interconnection gateways and uses fog nodes clustering method. The proposed models reduce the data transfer latency with reasonable amounts.

Cheng et al. [28] introduced a distributed fog development framework based on Openfog standards and developed a smart city application scenario with this framework. In this study, they chose the conventional transient and connectionless RESTful NGSI/HTTP [95] messaging model with a broker based context management approach for the orchestration of reliable data transfer inside the fog. We designed our proposed data transfer model on

a similar fog framework and presented a comparative analysis of the messaging models, demonstrating the limited scalability of the conventional approach.

Skarlat et al. [96–98] introduced an optimization model for minimizing data transfer latency of the fog by attacking the resource allocation mechanism. In their model, they apply colony based fog architecture in which a coordinator node orchestrates the data flow and optimizes the data traffic usage of the IoT system to relieve delays faced in the reliable data transfer inside the fog. Their model considers the current CPU and RAM usage of a fog node for the optimization. Whereas, in our study, we change the data transmission scheme of the inner fog rather than giving the latency control to a node. In our proposed approach, a fog connector controls all outgoing and incoming data traffic to/from the outer fog, so it does not affect the optimization of data transmission delays faced in inner fog.

Tran et al. [99] introduced a message queue based resource management model for elastic scaling of data processing in cloud environments. In their study, they aim to distribute the load in the cloud by using an optimized queue exchange model to the cloud resources. They use the producer-listener approach in the design of this Message Queue Model. In our study, we apply a similar producer-listener based message queue approach in our proposed messaging model for reducing delays of data transfer in the fog domain.

Besides these models, some open-source messaging systems are available for developers. Dobbelaere et al. [100] presented a comparative study of Apache Kafka and Rabbit MQ, which provide the most widely used message delivery techniques for fast, reliable bulk data transfer. The authors offer a critical analysis of the essential features of both methods and compare them with numerous performance metrics. For the scope of our study, the most crucial parameter is scalability. Using the analysis given in this comparative study, we can state that RabbitMQ provides limited scalability as it can only publish an incoming message to a single consumer, and increasing the number of producer channels can moderately improve its performance. Whereas, Kafka provides incoming messages to different consumers simultaneously and stores them for a determined period. Also, increasing the number of message partitions improves its throughput significantly. So, it is suitable for the development of a scalable fog solution. Hence, we choose the message

queue mechanism of Kafka for building the proposed messaging model in our test-bed environment.

3.5. SENSOR DATA FRESHNESS

Data freshness is studied in several CCN and Opportunistic Network approaches with distributed manner. Jaber et al. [101] proposed a distributed and time synchronization based data freshness model for sensor nodes in CCNs. Lee et al. [102] developed a real time data freshness model for distributed networks considering global time synchronization. Systems adapting these models should consider network time re-synchronization and temporal refreshing which requires extra packet transmission, meaning high traffic load in the sensor network.

Several centralized data freshness models for low and high data sampling rates are also studied for DISs [103–106]. The general drawback of these studies is that they fail to meet the required scalability of IoT systems in real time applications.

We propose a fog-based solution where a P/S server caches on-the-fly sensor information in the fog layer rather than streaming data to the cloud. Our solution does not require network time synchronization in the sensor nodes, thus, no extra control traffic is imposed. We adapt the caching based data integration model to utilize more transient sensor data for real-time freshness calculations. As data freshness is critical in real-time emergency detection, we provide ways of integration of freshness concept with our P/S scheme.

4. PUBLISH/SUBSCRIBE BASED FoT ARCHITECTURE

This chapter gives the details of the fog computing architecture, the interaction scheme of the components of the system, in which we realize our FoT based data packet scheduling and data traffic flow orchestration approach. Additionally, this chapter elaborates on how P/S Scheme is operated for the decision-making process with client-defined rules in the given architecture.

4.1. ARCHITECTURAL SCHEME

As illustrated in Figure 4.1, the fog-based IoT applications and services architecture, in which we develop our data packet scheduling and data traffic orchestration approach, is composed of three main layers and three sub-layers between sensor nodes and the client interconnection cloud. This architecture uses a decision-making process based on Publish/Subscribe pattern conventionally used in IoT applications [107–109]. We prefer this architectural pattern since it solves the scalability and dynamicity issues of the network architecture.

The first layer, named as Edge and protocol abstraction layer, operates between sensor devices and the fog. This layer interacts with the sensor devices located on the edge and manages them according to the communication protocols they use. For supporting heterogeneous interconnection of sensor networks in the edge, different data messaging protocols designed for the sensor devices in the edge are adapted in this layer for the fog computing. This layer converts the sensor data acquired from the sensor devices into messages to be understandable in the fog. So, from the FoT data traffic management perspective, this layer is the sensor data traffic generator part of the system. This layer possesses the SCs and an SC continuously captures heterogeneous sensor data from the sensor edge, which can consist of different types of sensor devices. Whenever it captures sensor data from these devices, an SC transforms the acquired sensor data into messages having a standard message format, to be understandable in the fog. After the conversion process, SCs forward data to the network layer of the fog architecture.

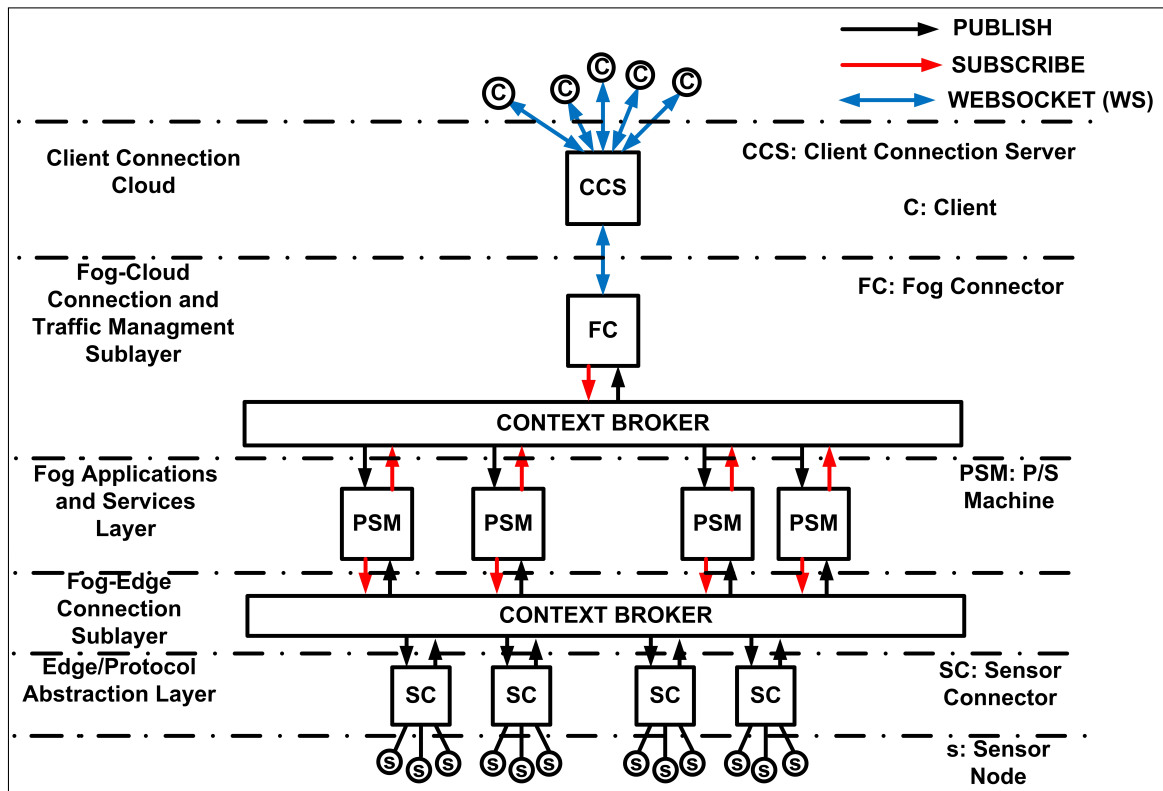


Figure 4.1. Architectural scheme

The second layer is the network layer and provides communication services used in the data transmission in the fog. This layer interconnects components located inside or outside of the fog and orchestrates sensor, control, decision data transmission. This layer consists of two sub-layers. The former is the Fog-Edge Connection sub-layer and is between fog and edge of sensor devices. This sub-layer collects sensor data from SCs and transfers them to IoT services performed in the fog. The latter is the fog-cloud connection sub-layer, which interconnects the cloud to the fog services and has a centralized fog connector for forwarding data packets to the cloud. Decision data traffic incoming from independent Fog Services is aggregated for scheduling in the fog connector.

The third layer, called as Fog Application and Services layer, applies services of IoT data analytics and periodically processes sensor data to make IoT-based decisions with Publish/Subscribe pattern. This layer consists of the P/S machines. The output of this layer is the data packets containing decisions. In a scalable fog architecture, multiple P/S machines may perform simultaneous operations and generate low-rate periodic IoT decision

traffic patterns independently [3, 89].

4.2. COMPONENTS OF THE ARCHITECTURE

This section gives the details of the components of the proposed FoT architectural design, and elaborates on how these components performs data transmission or processing for FoT based services.

4.2.1. Context Broker

CB is the data-driven processing component in the network layer, and it uses a subscribe/update-notify-based data integration scheme for the orchestration of the data flow between the fog layer components [28]. This component forwards the decision data traffic to FC, as well as it forwards sensor data traffic to PSMs. A fog layer component subscribes to any one of the data contexts in the CB to obtain data updated by a component in the fog. In an update operation, CB forwards updated data in the related data context to the context subscribers.

For the design of this component, in this thesis, we elaborate on two different data transmission mechanism for orchestration of the reliable data transfer in the fog. The details are given in Section 5.3.

4.2.2. Fog Connector

FC manages the data traffic resources in the fog, coordinates the fog with control messages, accepts incoming clients, and un/subscribes them from/to the decision-making operation performed in the system. For data integration, FC forwards data published by PSMs to the related client through the CCS in the client connection cloud. When a client wants to subscribe to the system, FC updates the related client subscription context in the CB. FC subscribes to the contexts related to the decision data publish. So, whenever CB obtains publish data, it forwards them to FC. After that, FC forwards published data to the related clients.

Data packets with different priorities arrive at the centralized FC and wait for the data forwarding service provided by this component. So, FC needs to maintain a wait queue for the incoming data packets. In addition, it manages data traffic according to the traffic classes and schedules data packets, considering their priority levels.

4.2.3. Publish/Subscribe Machine

PSM is the P/S-based data processing unit that performs IoT service-related decision-making operations using Publish/Subscribe Pattern [51].

A PSM obtains various types of contextual data from the CB for its decision-making services. The first one is the raw sensor data which can be directly used by decision-making operations performed by the PSM. This type of contextual data exists as scalar sensor data and represents one current state of the physical phenomenon in the time domain. The second one is the intermediate result of an external data analysis process which requires a complex data analytic (e.g., a Machine-Learning based pattern recognition process) for making a decision. The resulting label generated by this process is periodically acquired and cached by the PSM. The information in the label cached by the PSM is used in the decision-making process.

A PSM operates like a sensor data proxy and uses a two-phase asynchronous push/push data integration scheme for its decision-making process. In the first phase, a PSM acquires the contextual data and stores them in its intermediate cache. For the sensor data acquisition of this phase, it subscribes to the related contexts in CB. In the second phase, it processes the IoT Service, and it periodically makes decisions for multiple clients based on any combination of instant offline data in its cache, according to the client rules defined for its operation [52]. Then, it pushes this decision data to the CB to deliver it to the corresponding client subscribed to the system.

To get client rules, a PSM subscribes to a related client rule context of CB. When the FC subscribes a client to the system, it updates the related client rule context on CB, and then CB sends the decision-making rule to the related PSM. In this step, when a PSM obtains

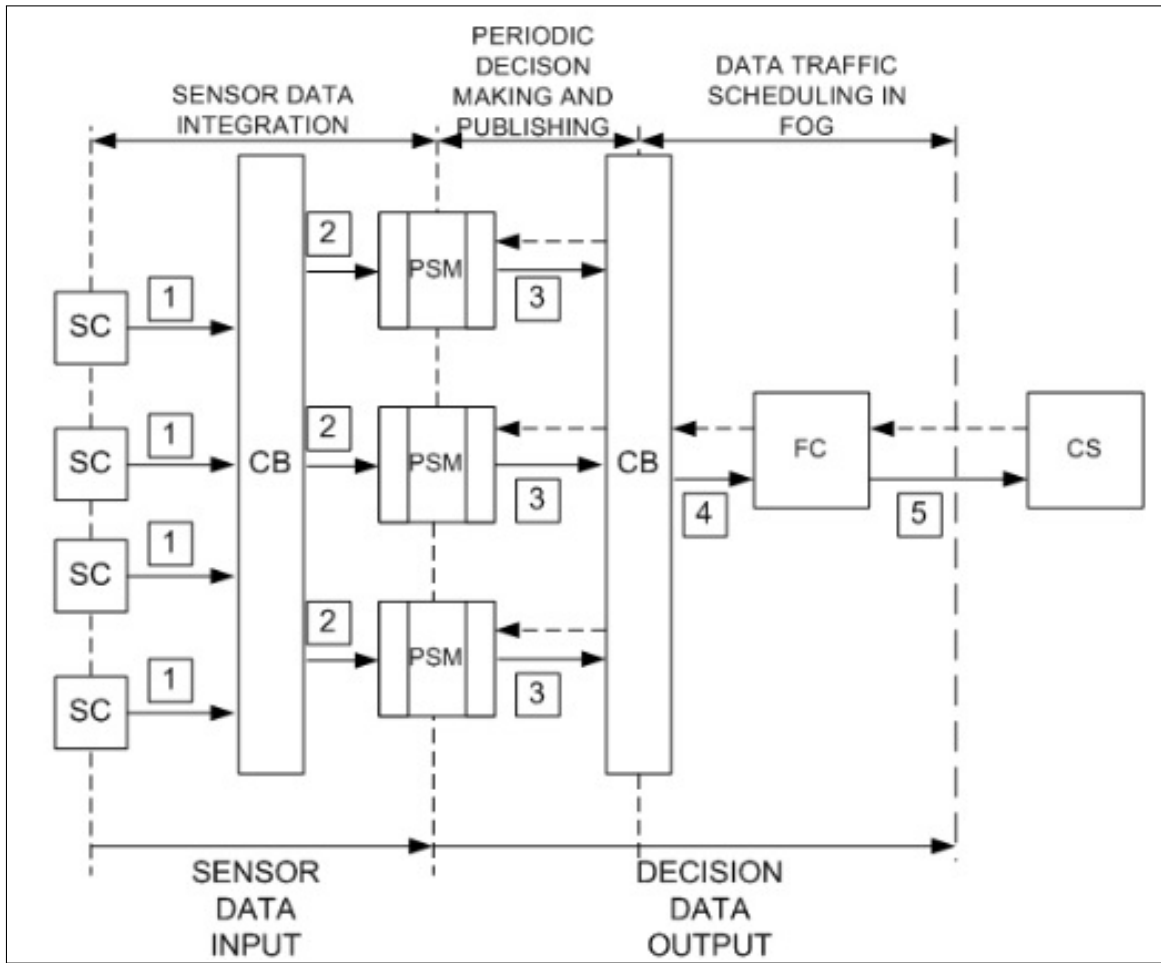


Figure 4.2. Interaction between components.

the decision-making rule, it reads the decision-making rule and determines the sensor or intermediate data contexts for the operation. After that, it subscribes to the related data contexts to obtain those data from the CB for the decision-making process.

4.3. INTERACTION SCHEME OF ARCHITECTURAL COMPONENTS

Fig. 4.2 illustrates the interaction scheme of the components involved in the fog architecture. The communication between the hosts in the fog is provided over CB with NGSI based JSON messages. For the orchestration data flow by CB, the required steps are defined below.

- For accepting incoming messages of a notification operation, a fog component

determines an HTTP URI and a unique data context string.

- This fog component, then subscribes to CB with this URI and this data context. (Subscribe)
- CB opens a data context and associates this URI to this data context.
- When any one of the hosts update a data context with a NGSI message, CB updates the content of the data context, then notifies the subscriber of the data context with using the HTTP URI associated to the data context. (Update-Notify)

Different steps of the interaction scheme illustrated in Fig. 4.2 of raw sensor data integration in the fog are explained below.

1. SC transmits sensor data that is acquired periodically from the physical environment to the CB.
2. CB notifies PSM for sensor data update with incoming sensor message. PSM updates related data in the cache.
3. A PSM generates decision outputs periodically according to the client rules by using any combination of offline sensor data in its cache. Then, it transmits the decisions to the CB and updates the related publish data context.
4. CB notifies the FC with an incoming decision message. FC saves this message in its wait queue.
5. FC schedules the decision packets and forwards them to the cloud for client connection.

4.4. OPERATING P/S SCHEME

This section provides the details about how P/S scheme is operated in the given architecture. This section elaborates on the functional parts designed for P/S Machine, the interaction scheme necessary for the Publish/Subscribe pattern performed in the proposed architecture and the client rule structure.

4.4.1. Functional Parts in P/S Machine

The functional parts and in the P/S Machine and their functions are the following:

- **Sensor Data Adapter (SDA):** Sensor Data Adapter transforms sensor data to be processed by the P/S, or it converts the actuation data generated by PSM to be useful in the sensor network. An SDA subscribes to the sensor data contexts in the CB for sensor data acquisition. On a sensor data notification coming from CB, it accepts incoming HTTP connections created by CBs. Then, it extracts the related contextual data and stores them in the sensor data cache of the PSM.
- **Sensor Agent (SA):** Sensor data cache of the PSM consists SAs. An SA is the virtual representation of a sensor node or the intermediate decision data. [51].
An SA holds the cached data that can be processed directly from an ED. It holds the timestamp of the moment when PSM caches the data and the time between each arrival of the sensor data into the cache and the previous (i.e., the sensor data inter-arrival time). Each state of this agent represents a primitive event for the decision-making process performed in the ED. Any one of the emergency decisions is the derived event from these primitive events.
- **User Agent (UA):** A UA represents a client in the PSM. It subscribes the related client for the decision making process. It obtains a client rule incoming from CB and prepares it for decision making. It interacts tightly with an ED and sends the decision made by the ED to the interacted client.
- **Emergency Detector (ED):** Emergency detector operates between UA and SA. It periodically detects an emergency according to a client-defined emergency detection rule as time flows. For detection of an emergency, it uses state of the SA.s associated with itself. This module operates in two steps. In the former, which is the subscription step, it reads the client rule passed to itself by the UA and caches it. In the latter, it periodically performs an event detection operation and makes a decision. It determines the freshness of this decision. As the second phase of the publish operation, it pushes this decision to its User-Agent.

4.4.2. Interaction Scheme for P/S Model

The interaction of the P/S model is provided over application messaging protocols and TCP/IP protocol stack. Over these, JSON-like messages are used for the interaction between the client (i.e, the subscriber) and PSM (i.e., the publisher). The general structure of a message needed for the interaction is illustrated in Fig. 4.3. The possible values of the type field and the corresponding message body structures are the following:

- **101 - ClientToServerObservationMessage:** A Client sends this message to the client connection server located in the client interconnection cloud when s/he wants an emergency observation service from the system. As illustrated in Fig. 4.4, the body of this message contains the identifier of the client. When the server obtains this message, it gets the corresponding client rule according to the client id from the rule database. After that, it prepares a type 102 message consisting of this client rule and the decision-making conditions which are related to the client rule. Then it sends this message to the related PSM(s) in the fog.
- **102 - ServerToPSMachineSubscribeMessage:** The server sends this message to the PSM for the subscription operation. As illustrated in Fig. 4.5, the jsonMessage contains the client rule and the subscription objects needed in the client rule. P/S Machine exploits this message in subscription operation. It creates UA and ED for the client using the information in clientIdentifier structure. After it exploits the subscriptionObjects and caches them for the periodic emergency decision-making of ED.
- **103 - ServerToClientPublishMessage:** The server sends this message to the client in the publish operation. This message consists of the parts of the decision result which is useful by the client.
- **104 - PSMachineToServerPublishMessage:** As described in 4.4.1, an ED periodically performs an emergency decision-making process according to a client rule associated with itself. ED writes the decision result into this message. In addition, this message consists of the emergency decision type, the freshness of the decision. Fig. 4.6 illustrates the structure of this message.

- **105 - PSMachineToServerInitializeMessage:** In its initialization step, PSM creates this message to introduce its physical networks to the server. Fig. 4.9 gives the structure of this message.

```
{  "type" : <int>
   "jsonMessage" : <message body as string> }
```

Figure 4.3. General message type

```
{
  "type":101,
  "jsonMessage":"{"clientID":<client id>}"
}
```

Figure 4.4. Structure of observation start message

4.4.3. Client Rule Structure

Client rules are essential parts in the operation of P/S based decision-making scheme of the PSM of the FoT architecture. As illustrated in the ontology in Figure 4.8, a client rule (i.e, client identifier) refers the collection of emergency detection alerts. As depicted in Fig. 4.10, an emergency alert consists of the sequence of several criteria combined with event definition algebra which is introduced in Section 4.4.5. A criteria denotes the conditional decision-making parts, and is used to detect any emergency occurred in the physical area monitored by the system. An ED decides any one of the emergencies exists or not by using the sequences of the criteria in the client rules whose structure is illustrated in Fig. 4.11. The fields of the criteria are the following:

- name : The identifier of the criteria.
- physicalArea : The array which contains the physical area definitions for detection of the criteria. An example for the physical area definition is given in Fig. 4.13.

```
{
  "clientIdentifier":{
    <client identifier>
  },
  "subscriptionObjects":{
    "criterias":[
      <emergency detection criterias>
    ],
    "emergencyAlerts": [
      <client defined alerts>
    ]}
}
```

Figure 4.5. Structure of client subscription message

```
{ "clientId":<client id as string>,
  "message":{
    "messageParts":[<parts of decision result>]],
    "trafficClass":<int>,
    "fresh":<boolean> }
```

Figure 4.6. Structure of publish message in PSM

```
{
  "physicalAreasOfPSMachine":
    [<Physical Area Definitions>]
}
```

Figure 4.7. Structure of PSM initialization message

- **observationType:** If this field is 1, the client only wants the periodic observation result from the environment. If this field is 2, the client wants the observation result and the emergency decision result. If this field is 3, the client only wants the emergency decision result.
- **sensor:** This field refers to the sensor type. If this field is 1, an ED looks the accelerometer data in S.A. If it is 2, an ED controls the temperature sensor data in S.A. If it is 3, an ED controls humidity sensor data in S.A. If it is 4, an ED controls light intensity sensor data in S.A, etc.
- **comparisonOperator:** This field determines the conditional operator of a criteria. "GT" refers to "greater than", "LT" refers to "less than", "GE" refers to "greater or equal" etc.
- **value:** The threshold value for the emergency alert.
- **algorithm:** This field refers various algorithms to be used in the emergency detection.

4.4.4. Client Rule Examples

The below lists several emergency scenarios whose decision making criterion can be written using the client rule structure given in Section 4.4.3. The other topic combinations could be generated using the client rule structure.

- **Example1 : One-dimension Thresholding: Object Body Temperature Observation:** Observe a bulk storage fuel-oil tank in every second with a sensor node and fire an emergency alert when the temperature is detected above 80. Note that one-dimension thresholding can be applicable in a significant set of emergency alert cases observed by scalar sensors such as in-door toxic gas detectors, oxygen deficiency observers, gas detectors, smoke detectors and human body temperature observers etc...
- **Example2 : Two-dimension or Multi-dimension Thresholding:** Observe leakage in a bulk storage fuel-oil tank in every second with a sensor node and observe temperature in same time. If the leakage exceeds 50 liters per second, generate immediately an alarm, else look at temperature, if it exceeds 80 then generate an alert. Note that any combination of the One-Dimension Thresholding can apply to this case.

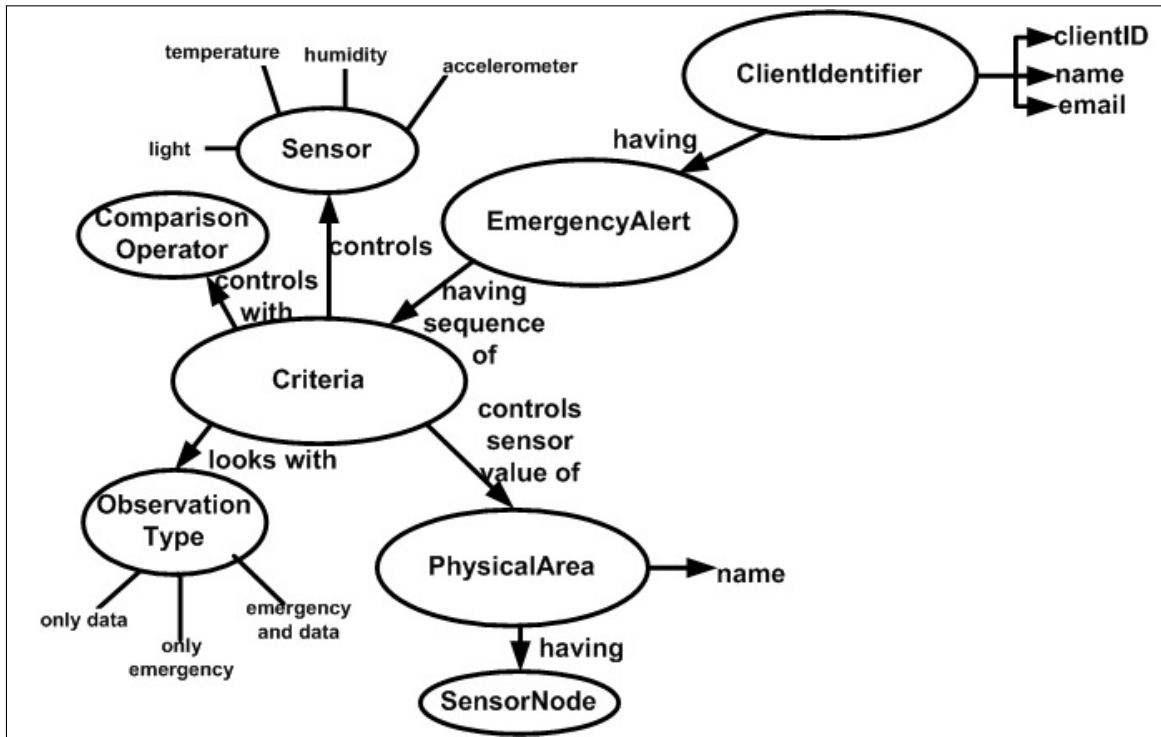


Figure 4.8. Client Rule Structure

4.4.5. Derivation of Decisions

In the creation of the emergency based data traffic event with primitive events in a P/S based observation system performed by PSMs, a set of decision making criteria are used with an event derivation algebra. The set of rules of the event algebra which uses emergency decision making criteria are the following:

- **AND** : $E = C1 \& C2 \equiv E = C1 \wedge C2$. In an iteration, control the related primitive event if it meets the criteria C1 and C2. If it meets all of C1 and C2, then generate emergency alert E.
- **OR** : $E = C1 || C2 \equiv E = C1 \vee C2$. In an iteration, control the related primitive event if it meets the criteria C1 or C2. If it meets the any one of C1 or C2, then generate emergency alert E.
- **SEQ** : $E = C1 - > C2 \equiv E = SEQ(C1, C2)$. If C1 is true, in the next iteration control if C2 is true. If so, generate emergency alert E.

```
{
  "clientPeriod": 1000,
  "userId": "...",
  "predefinedEmergencyAlerts": ["A1"],
  "email": "...",
  "userName": "...",
}
```

Figure 4.9. Example client identifier

```
{"name": "A1", "criterias": "C1&C2", "message": ".."}
```

Figure 4.10. Example decision making alert

```
{
  "name": "C1", "physicalArea": ["Room1"],
  "observationType": 2,
  "sensor": 2, // temperature
  "comparisonOperator": "GT", "value": 80
}
```

Figure 4.11. Example decision criteria

```
{"name": "C2", "physicalArea": ["Room1"],
  "observationType": 2,
  "sensor": 6, // water tank leakage
  "comparisonOperator": "GT", "value": 50}
```

Figure 4.12. Example decision criteria 2

```
{ "nodes" : ["Room2#Node1"], "nameAlias" : "Room2" }
```

Figure 4.13. Example physical area definition

4.5. QOS FOR FOT ARCHITECTURE DETAILS

Table 4.1 gives the IoT data traffic types with potential utilization area and their characteristics.

Table 4.1. Data traffic classes

Data Traffic Type	IoT Application Area	Characteristics
Mission Critical	Emergency Detection, Anomaly Detection, Natural Disasters Detection	infrequent, delay sensitive, high demand on QoS.
Real-Time Interactive Services	Actuation Data, Industrial Control, Automated Lighting and Heat Control, Control of Smart Appliances	infrequent, high demand on QoS.
Result of CPU Bound Services	Cognitive Services Process	infrequent, high demand on QoS.
Best Effort	Actuation Data, Industrial Indoor Climate Control, Personal Condition Monitoring, Air Pollution Monitoring, WildLife Tracking, Energy and Water usage Monitoring, Building Condition Monitoring	frequent, tolerable delays

4.5.1. Data Traffic Classification

Table 4.2 gives potential QoS guarantees expected by the sub/layers of the given fog architecture.

Table 4.2. QoS guarantees in the layers

Layer	Expected QoS Guarantees from Layer
Application Layer	Real time delays in data processing [110–118], higher accuracy and validity in decision, acceptable service load in the condition of mass sensor data transmission, support for SLAs between applications and clients.
Network Layer (Fog-Edge Connection Sublayer)	Sensor packet transmission delay [22, 119], support of vast amount of sensor data stream transmission both in volume and variety (i.e., heterogeneity) dimension, high support of scalability of the interconnected components
Network Layer (Fog-Cloud Connection Traffic Management Sublayer)	Accurate and fast application specific data traffic scheduling in the existence of vast amount of decision and control data packets, both in volume and variety dimension, support for scalability, support for availability of fog components
Sensing and Perception Layer	Guarantee for service lifetime

5. DATA TRAFFIC ANALYSIS

This chapter gives the details of the data traffic analysis with presenting four concepts which focus on data traffic analysis of our thesis. In the first, we explain the primitive and complex event model needed for the data traffic management concept studied in the thesis. Additionally, in this section, we present the data types which are the main parts of the FoT data traffic. In second, we give the details of the sensor data traffic modeling with sensor data freshness concept contributed by our thesis. In the third, we analyze the data traffic flow orchestration of the inner fog with elaborating on the internals of the two alternative communication models. In the last, we focus on the decision data traffic scheduling and provide the details of the CEP based data traffic scheduling concept of our thesis.

5.1. DATA TRAFFIC EVENT MODEL AND DATA TYPES

This section defines the primitive event model and complex event model used in the thesis. Section 5.1.1 examines the primitive event for the thesis and the definition of the primitive event. Section 4.4.5 gives the details of the mechanism to obtain traffic events using primitive events. This section describes an event algebra which is useful to generate complex events from primitive events.

5.1.1. Primitive and Complex Event Model

As described in Section 4.3, data that arrives at PSM is cached in SAs with an IAT information. Any change in an SA represents an occurrence of a primitive event for the client rule-based decision-making process of PSM. In PSMs, these events are combined to make emergency decisions according to a client rule and an event composition algebra described in Section 4.4.5.

In its decision data traffic scheduling process, an FC captures the incoming decision messages and enqueues them. The occurrence of this event is considered as a primitive event in FC. In the data traffic scheduling process, these events are combined in a time-based

sliding window. The details for this complex event and the decision data traffic scheduling mechanism are given in Section 5.4.

5.1.2. IoT Data Types

In the IoT ecosystem, the major subset of applications benefit periodic observation services [10]. These applications acquire one or a group of sensor measurements from the IoT Data Integration System. Data contains the periodic monitoring result of the physical phenomena represents a crucial class of data traffic to be used in data traffic management.

The interaction scheme defined for the architectural model is compatible with IoT sensor data measurement observation services by allowing a client to define physical phenomenon observation criteria in a client rule. As described in Section 4.4.3, a client can define an observation criteria with setting `observationType` to 1.

IoT-based real-time emergency observation systems necessitate an operational environment for remote environmental control and emergency decision-making. In addition to this, those systems need several emergency decision-making criteria. The output of the overall emergency decision-making process of these systems is a decision result that contains an emergency alert, and these systems prefer to publish those results to their clients. Therefore, the traffic of the messages which contains decision output is an important IoT data traffic class.

The client rule semantics described in 4.4.3 supports the IoT-based real-time emergency decision making. It provides the ways of a client-friendly emergency and the emergency criteria definition. So the proposed model is compatible with those services in the IoT ecosystem.

Interaction between the IoT network components necessitates the control messages to be transmitted by each other. Those messages handled in the intermediate hosts in the IoT backbone for the service composition perspective [10] form a crucial class of the IoT data traffic.

Our proposed model presents several alternatives of the control data which are useful in the observation service. Section 4.4.2 gives the details of such alternatives for control data.

5.2. SENSOR DATA TRAFFIC ANALYSIS

This section gives the details of the sensor data traffic analysis considering the sensor data freshness.

5.2.1. Sensor Data Freshness

Data freshness is to determine the data in a system is recent or old. We need to study data freshness in this thesis for several reasons:

- Data freshness rate is a critical QoS parameter of a periodic decision-making system. This parameter gives the age certainty of any decision result generated from the system. For achieving high QoS from this kind of system, a higher data freshness rate is a crucial requirement.
- In most IoT-based real-time systems, we should deal with on-fly data within the strict time constraints, which leads deep sensor data cleansing and washing to be unfeasible within the time domain [57, 63]. Old sensor data is a crucial subset of uncleaned and inaccurate data and we should take into account those old marked data in decision-making processes due to the time constraints. This leads an uncertainty in terms of freshness. [120, 121]. Modeling maximal data freshness is critical for on-fly and real-time primitive event cleansing. The outcomes of the data freshness modeling are crucial in the definition for the success of the operation of a Complex Event Processing approach designed for decision forwarding in the fog.
- In Chapter 4, we define a timely stable data processing style which is based on client-defined periods in client SLAs [15], since a PSM provides an offline and an asynchronous decision making mechanism for the periodic decision making process. Since we provide this process, we try to minimize the effect of IAT fluctuations of the sensor network on the fog domain. However, this affects the data freshness rate and any subset of the decisions may be marked as old. So, we should investigate data

freshness and so, a real-time system requires a maximal data freshness calculation model to get high data age certainty results for the real-time system.

- We can consider emergency decision freshness rate as a part of our IoT data traffic modeling. The data traffic both created by the service client and sensor nodes in the fog affects the data freshness. So modeling this parameter is a crucial requirement for data traffic management in the fog domain. Thus, overall, we need to model data freshness rates in the fog.

Data freshness time can be calculated statically or with a probabilistic manner. The data freshness time calculation methods that we have studied is given as a list below.

- As a static and stateless method, the system can utilize a fixed data freshness time.
- As an alternative, rather than using a fixed freshness time, we can calculate a static and stateless freshness time using offline IAT information obtained from sensor data transfer. This method considers the traffic characteristics of the sensor network connected to the fog. For this method, a standard deviation based time calculation model can be used.
- We can calculate a probabilistic and stateful data freshness time with considering IATs in the S.A.s dynamically.

Parameters which are related with data freshness are:

- N : Number of the sensor nodes connected to the fog.
- P : Sensor data acquisition period.
- Me : IAT measurement read from one of the SA.
- A_i : Smoothed IAT value in the i th data freshness time calculation. Initially, it is set to 0. The calculation of this value is given in Equation 5.3.
- Err : Difference between the measured IAT value and A .
- h : The gain value for the deviation. In Jacobson RTO calculation, this value is set to 0.25.
- g : The gain value for the new IAT value. In Jacobson RTO calculation, this value is

set to 0.125.

- D_i : Smoothed mean deviation of IAT values obtained in i th iteration of the data freshness time calculation. Initially, it is set to D^*h .
- $\sigma_{IAT_{N,P}}$: The standard deviation of IAT values for any one of the N and P values.
- FT : Data freshness time calculated with one of the models.

For sake of the clarity, we define some terms about data freshness:

- Data Freshness Time (FT) : As illustrated in the time chart in Figure 5.1, data freshness time is the time span in which a primitive event in SA can be marked as recent.
- Fresh Emergency Decision : The decision result which is marked as recent by an ED.
- Old Emergency Decision : The decision result which is marked as old by an ED. This is the uncertain data for the time domain of the time stable decision making system.
- Data Freshness Rate (FR) : The rate of the fresh decisions in an emergency observation.
- Pessimistically Fresh Emergency Decision : If any one of the emergency decision is marked as pessimistically fresh, we mark all primitive event which is related to it as fresh. If we mark one of them as old, the related emergency decision will not be pessimistically fresh.
- Optimistically Fresh Emergency Decision : If any one of the emergency decision is marked as pessimistically fresh, we mark one of primitive event which is related to it as fresh.
- If any one of the data is marked as pessimistically fresh, we mark the primitive event which is related to it as fresh with a percentage of n . For example, if $n=25$ and $N=8$, marking two of the primitive event in the related SAs as fresh is enough to determine the emergency decision result as fresh.

Figure 5.1 illustrates the client period (CP) and Freshness Time (FT) in detail. The first line in the figure shows the time flow and the lines which are perpendicular to this line shows the expiration of the CP and the detection time of the emergency. The second line

shows the CP and FT in detail. $CP - FT$ gives the time difference between client period and freshness time. As $CP - FT$ increases, we expect more sensor data in the SAs are marked as old and the freshness rate decreases. Detailed evaluation of the effect of CP to the freshness rate is given in Chapter 7.

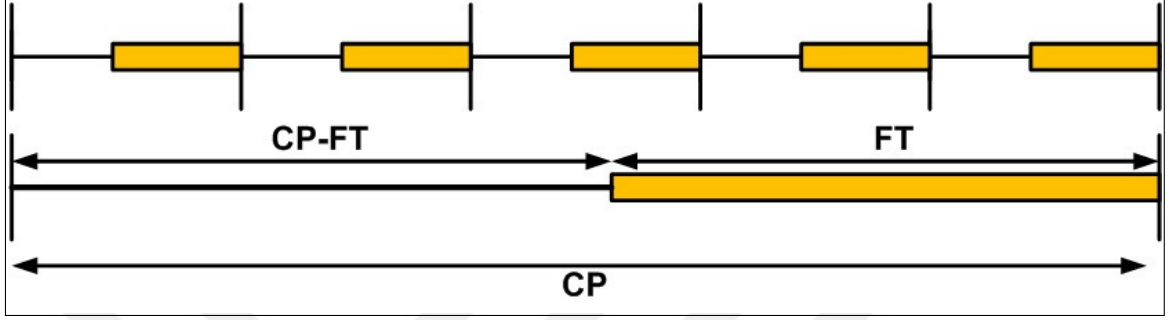


Figure 5.1. Detailed freshness parameters illustration

As a fixed data freshness calculation method, an ED in the PSM in the fog marks the decision as old or fresh according to the FT given in one of the SLA between client. Several time chart examples are given in Figure 5.2. The evaluation of this model is given in Chapter 7.

We can calculate FT to obtain a maximal data freshness rate with considering the traffic characteristics of the sensor network of the fog. So in this method, we consider the IAT distributions of the sensor data coming from the sensor network to the PSM. As discussed in Chapter 7, especially for the zigbee medium, more than 90 percent of the IATs obtained in the PSM of the fog domain are smaller than the time value given in Equation 5.1. Thus, this method can give us a maximal data freshness rate. So, in this method, we use the standard deviation and the mean of the sensor data IATs obtained from the PSM.

This model requires significant IAT samples pre-collection to calculate the mean and standard deviation values before its application. So, in the next section, we present a data freshness calculation model which do not require any IAT sample pre-collection.

$$FT = \overline{\overline{IAT}}_{N,P} + \overline{\sigma}_{IAT_{N,P}} \quad (5.1)$$

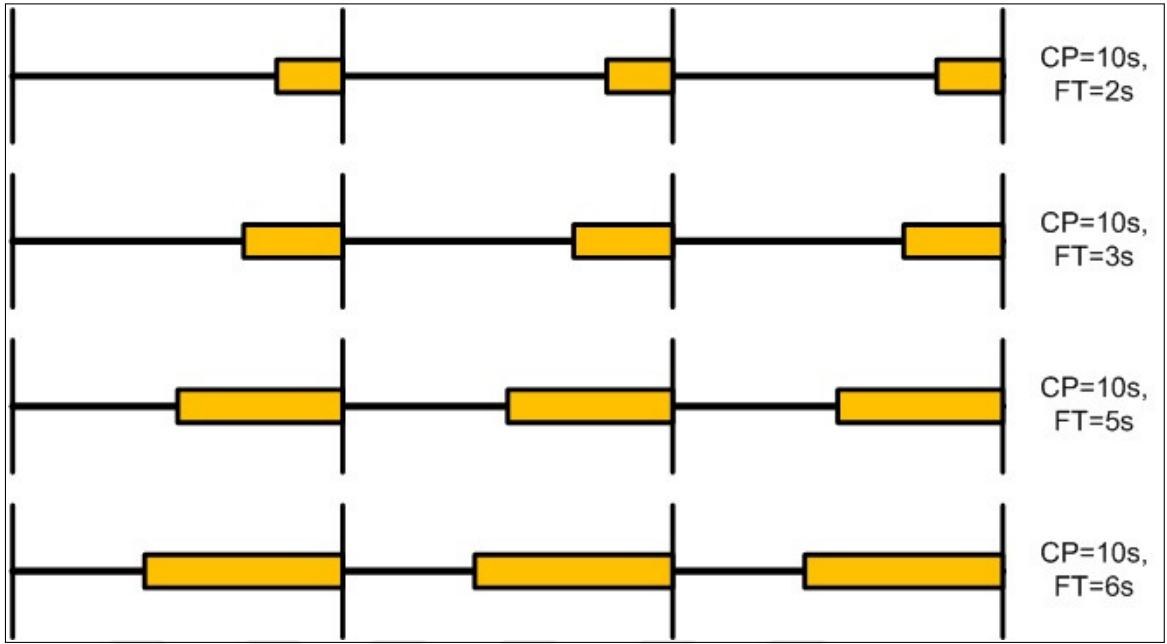


Figure 5.2. Static freshness times with client periods.

$$Err = Me - A_{i-1} \quad (5.2)$$

$$A_i = A_{i-1} + g * Err \quad (5.3)$$

$$D_i = D_{i-1} + h * (|Err| - D) \quad (5.4)$$

$$FT = A_i + (1/h) * D \quad (5.5)$$

Data freshness time can be calculated with an exponential averaging method using an on-fly IAT value. For this, we adapt the Jacobson's TCP's RTO formulation for the data freshness calculation. The evaluation of this method is given in Chapter 7. The steps for the data freshness time calculation method in every CP expiration are:

- (Step 0) : Set FT initially to CP . Set A initially to 0. Set D initially to $CP * h$.
- (Step 1) : For each N nodes, calculate the last IAT of the sensor data arrival. Any one of the IAT value can be calculated using time stamp value in the corresponding SA.
- (Step 2) : For each N nodes, calculate a candidate A using Equation 5.3, D using Equation 5.4 and FT using Equation 5.5. Use IAT value in SA. as Me
- (Step 3) : Find the maximum freshness time value from the candidate freshness times

obtained in Step 2, set FT to this value and update A and D according to this new FT value.

5.3. DATA TRAFFIC FLOW IN FOT ARCHITECTURE

In this section, we analyze the transfer of sensor data and publish data flow using two alternative communication models. In the first model, the orchestration of CB is designed without MQ. This scenario assumes the conventional method for Publish/Subscribe based reliable sensor data transfer inside the fog [28]. Fig. 5.3 and 5.4 show the corresponding timing diagrams for one sensor and decision data transmission for this case. In this scheme, CB uses a conventional RESTful messaging style over HTTP non-persistent connection. Hence, the reliable data context update and notify operations are based on the reliability achieved by the underlying TCP connection. In such a scheme, a CB accepts incoming connections from the data context updater as an HTTP server and as the initial step, illustrated in Fig. 5.3 and 5.4, the connection pairs should perform a typical three-way handshake process before reliable data transfer for any of the update data to the CB. Additionally, for the data orchestration, a CB performs data update notification operation and sends data to the context subscribers with establishing a typical non-persistent HTTP connection to the URI pre-defined by the subscriber. As in the data context update, a three-way handshake process should be performed before any of the data notification operations too. Although, initial three-way handshake process is a requirement for establishing a reliable data transfer to be done for CBs, performing this process before every data update and notification operation costs one extra RTT delay in the data transmission.

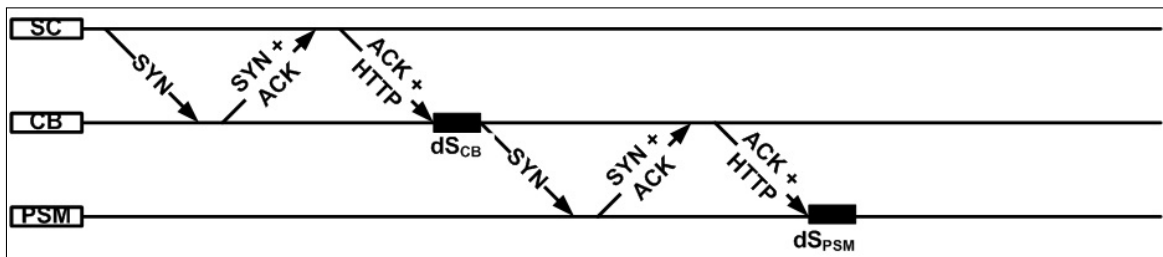


Figure 5.3. Timing diagram for sensor data transmission with scenario 1.

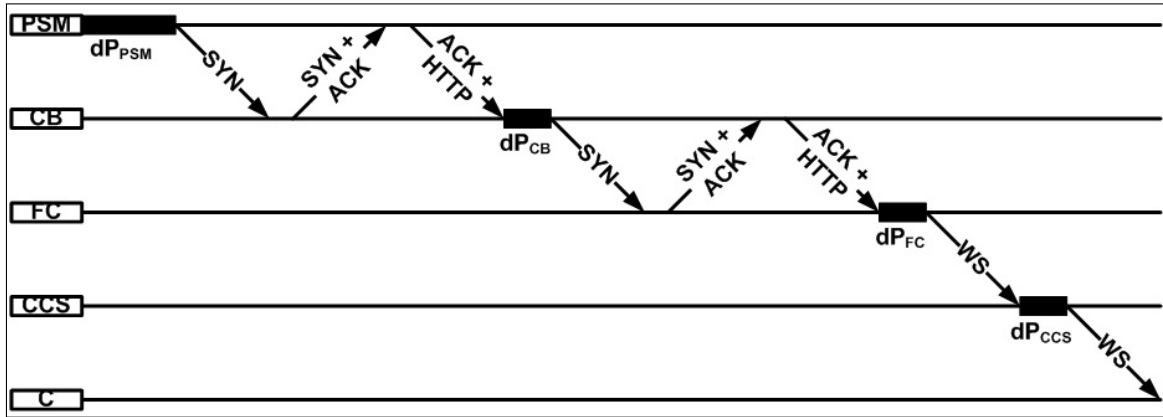


Figure 5.4. Timing diagram for decision data transmission with scenario 1.

We can partially eliminate latency costs related to the three-way handshake process needed in the reliable data transfer by providing a connection-oriented MQ scheme over CB as an alternative data flow model in the fog. Fig. 5.5 and 5.6 give the corresponding timing diagrams for one sensor and decision data transmission for MQ augmented CB scenario. In this scheme, we assume a basic producer-listener based message queue such as MQP of Kafka [100] for transmission of data over a persistent connection that does not need a three-way handshake process in reliable data transfer from CB to other system components.

5.3.1. Processing and Data Transmission Delays

With considering the data flow scenarios for the proposed architecture and timing diagrams, following parameters can be defined for the data transfer latency in the fog.

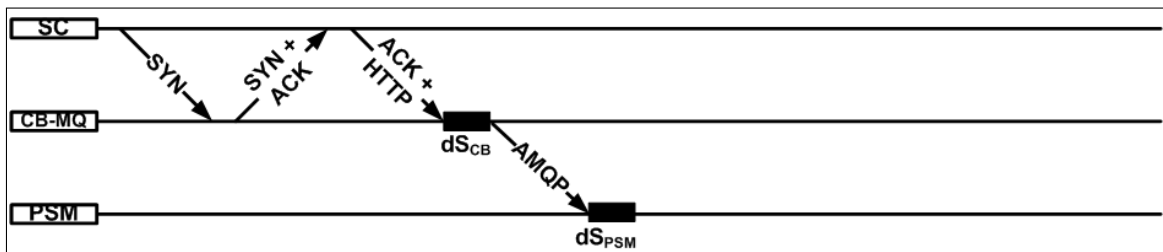


Figure 5.5. Timing diagram for sensor data transmission with scenario 2.

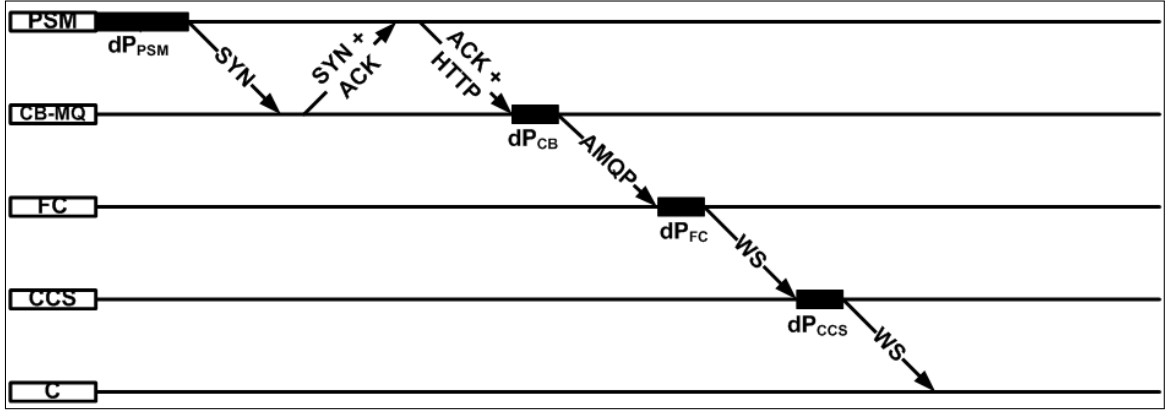


Figure 5.6. Timing diagram for decision data transmission with scenario 2.

- dS_i^B : The delay caused by update and notification of i^{th} sensor data message in CB and additional delay of transmission of bytes of i^{th} sensor data.
- dS_i^{BQ} : The delay caused by update and notification of i^{th} sensor data message faced in CB-MQ system and additional delay of transmission of bytes of i^{th} sensor data.
- dS_i^P : Caching delay of i^{th} sensor data message in the PSM.
- dP_j^P : Processing delay of j^{th} decision making in the PSM and additional delay of transmission of bytes of j^{th} decision data.
- dP_j^B : The delay caused by update and notification of j^{th} decision data message in CB and additional delay of transmission of bytes of j^{th} decision data.
- dP_j^{BQ} : The delay caused by update and notification of j^{th} decision data message in CB-MQ system and additional delay of transmission of bytes of j^{th} decision data.
- dP_j^F : The delay of j^{th} decision data transfer faced in the Fog Coordinator and additional delay of transmission of bytes of j^{th} decision data.
- dP_j^{CS} : The delay of j^{th} decision data publish faced in the Client Connection Server in the cloud and additional delay of transmission of bytes of j^{th} decision data.
- $RTT_i^{a,b}$: The round trip time for i^{th} data message transmission between components of a and b . It is the baseline latency faced between hosts in the network. For this term, a and b can be one of S which denotes SC, B which denotes CB, BQ which denotes CB-MQ, P which denotes PSM, F which denotes FC, CS which denotes CCS.

5.3.2. Latency Analysis

Let,

- $\Delta_i^{S,B}$ be the total processing and additional delays faced in transmission of i^{th} sensor data in CB and PSM, and computed with:

$$\Delta_i^{S,B} = dS_i^B + dS_i^P \quad (5.6)$$

- $D_i^{s,B}$ be end-to-end Latency faced in i^{th} sensor data transmission over the path of SC, CB and PSM for the first scenario and represented with:

$$D_i^{s,B} = 1.5RTT_i^{S,B} + 1.5RTT_i^{B,P} + \Delta_i^{S,B} \quad (5.7)$$

- $\Delta_i^{S,BQ}$ be the total processing and additional delays faced in transmission of i^{th} sensor data in SC, CB-MQ and PSM, and computed with:

$$\Delta_i^{s,BQ} = dS_i^{BQ} + dS_i^P \quad (5.8)$$

- $D_i^{s,BQ}$ be end-to-end latency faced in i^{th} sensor data transfer over the path SC, CB-MQ and PSM for the second scenario and expressed as:

$$D_i^{s,BQ} = 1.5RTT_i^{S,BQ} + 0.5RTT_j^{BQ,P} + \Delta_i^{S,BQ} \quad (5.9)$$

- $\Delta_j^{p,B}$ be the total processing and additional delays faced in transmission of j^{th} decision data in CB, FC, CCS, C, and computed with:

$$\Delta_j^{p,B} = dP_j^P + dP_j^B + dP_j^F + dP_j^{CS} \quad (5.10)$$

- $D_j^{p,B}$ be end-to-end latency faced in j^{th} decision data transfer over the path PSM, CB,

FC, CCS and C for the first scenario and defined with:

$$D_j^{p,B} = 1.5RTT_j^{P,B} + 1.5RTT_j^{B,F} + RTT_j^{F,CS} + \Delta_j^{P,B} \quad (5.11)$$

- $\Delta_i^{S,BQ}$ be the total processing and additional delays faced in transmission of j^{th} decision data in SC, CB and PSM, and computed with:

$$\Delta_j^{p,BQ} = dP_j^P + dP_j^{BQ} + dP_j^F + dP_j^{CS} \quad (5.12)$$

- $D_j^{p,BQ}$ be end-to-end latency faced in j^{th} decision data transfer over the path PSM, CB-MQ, FC, CCS and C for the second scenario and represented with the following:

$$D_j^{p,BQ} = 1.5RTT_j^{P,BQ} + 0.5RTT_j^{BQ,F} + RTT_j^{F,CS} + \Delta_j^{P,BQ} \quad (5.13)$$

Superposition property for latency - In a scalable fog system, components in the network layer schedule numerous simultaneous message update-notification events that affect the timing of each message transmission. Additionally, multiple sensors or decision data arrive simultaneously for processing in the application layer or traffic management sub-layer components in the fog. Hence, $\Delta_i^{s,B}$, $\Delta_i^{s,BQ}$, $\Delta_j^{P,B}$, $\Delta_j^{P,BQ}$ depend on the sensor data update-notification event occurrence which is related with number of sensor nodes (N), data sampling period (SP) and the number of active clients exist in the system (M). In Section 7, we give a detailed empirical analysis of this property with different N , SP and M values and investigate the dependence of these values with $\Delta_i^{s,BQ}$, $\Delta_j^{P,B}$, $\Delta_j^{P,BQ}$ in a scalable fog.

If we locate SCs, PSMs, FC in the same local area network (LAN) and assume that RTT values in data transmission between these components are same and compare $D_i^{s,BQ}$ with $D_i^{s,BQ}$, $D_j^{p,B}$ with $D_j^{p,BQ}$, the MQ based scenario could save at least 1 RTT for each sensor and decision data transmission latency in the fog.

5.4. DECISION DATA TRAFFIC SCHEDULING

In the following sections, we examine the details of three alternative data traffic scheduling policies that can be used in the FC; First-In-First-Out (FIFO) scheduling, non-preemptive Multi Priority Discipline Queue (MPDQ) scheduling and the proposed Complex Event Processing (CEP) based FIFO scheduling policies. Table 5.1 compares several aspects of these policies.

5.4.1. FIFO Scheduling

In the FIFO Scheduling policy, the system maintains a single queue. It enqueues incoming packets of the traffic flow into this single queue while the FC performs a data forwarding service [122]. After the service is completed, the system dequeues the packet in front of the queue.

FIFO scheduling treats all packets equally [123] and does not consider any prioritization of them during enqueue or dequeue process. So, on average, high priority packets may have longer wait times than low priority packets with FIFO scheduling.

5.4.2. Non-preemptive MPDQ Scheduling

MPDQ scheduling policy maintains multiple queues for several data packet classes. In the MPDQ policy, the system enqueues an incoming packet to one of its queues, considering the priority class associated with the data packet. The system chooses a data packet for its service process in descending order of priority, starting from the highest priority queue to the lowest priority queue. [123]. So, in the MPDQ policy, highest priority data packets get the service first, and thus the wait times of these packets are relatively lower.

A Non-Preemptive MPDQ scheduler does not suspend the service in progress for low priority packets when a higher priority packet arrives at the system [122]. Despite this feature, low priority packets could potentially starve, since the system can enqueue new higher priority packets and choose these packets for the service rather than the old ones with

lower priority [123].

5.4.3. CEP based FIFO Scheduling

FIFO scheduler has an advantage over MPDQ policy in that it prevents starvation for low priority packets by considering only the order of arrival of the packets for service, not their priority. However, the main drawback of the FIFO scheduling policy is that it may cause higher priority packets to have longer wait times when compared to the MPDQ scheduling policy. Thus, it is necessary to construct a packet scheduling policy that utilizes the advantage of the FIFO scheduling policy while negating its drawback. This can be achieved with an efficient and low-overhead priority-aware packet sequence adjustment method added to the FIFO scheduling policy. We develop such policy over dynamic batch windowing based CEP as it provides low overhead, real-time and timely processing in event aggregation, fast pattern identification, and event handling [124].

The proposed policy does not affect the enqueue mechanism of the FIFO scheduling. Whereas, it changes the dequeue mechanism of the FIFO scheduling. The steps of the CEP based FIFO scheduling in FC are the following.

1. Maintain single FIFO queue and an EPA.
2. Enqueue incoming packets to the queue.
3. In dequeue process, control the current number of packets in the queue.
4. If it is 1, then dequeue and service packet without sending it to EPA.
5. If it is greater than 1, set batch window size of EPA as the current number of elements in the queue, then dequeue all packets and send them to EPA.
6. EPA sorts all packet in the batch window and after that, immediately services all the packets.
7. As the service in step 6 continues, new incoming packets are enqueued into queue. After all packets in EPA are serviced, goto step 3.

Table 5.1. Data traffic scheduling approaches

	FIFO	MPDQ	CEP based FIFO
Description	First in first out scheduling	Multi Level Queues for prioritization	Batch Window based sorting according to priorities
Data Structure	Single Queue	Queues for every priority level	Single Queue and EPA for primitive event sorting
Primitive Event	-	-	Decision data packet incoming
Fairness	Fair for every data packet	Gap in wait times of different priority levels, Starvation problem	Approximates FIFO, decreases wait time gaps.

6. QUEUING ANALYSIS

This chapter presents the queuing model for a limited capacity queue system that employs FIFO or MPDQ scheduling policies. This model considers the waiting line(s) of different priority packets generated from periodic and asynchronous IoT decision-making processes in the scalable fog. Besides, this section gives details on how the proposed queuing model is applicable to the proposed CEP based scheduling policy.

6.1. MODEL DEFINITION

In the proposed scalable fog architecture, decision packet traffic patterns exhibited by individual and independent PSMs produce aggregated random arrivals on a centralized queuing system. We can determine the mechanism of these mutually independent random arrivals by the superposition process, such as the Poisson process of Palm-Khintchine theorem [3] and Markovian Models which provide an adequate approximation of a realistic queuing model [38]. Thus, for the limited capacity queuing system with random arrival, multi-priority levels, and one servicing facility, we develop our model over the non-preemptive $M/M/1/K/PQ$ queuing model. This model is the result of the Poisson Process of finite size queuing system with multi priorities. Table 6.1 lists the key system parameters and below lists the model parameters and their formulations used in the development of this queuing model.

Average arrival rate of class j data packet traffic into the queuing system is represented by λ_j which is represented by:

$$\lambda_j = \sum_{i=1}^m \left(\frac{M_i}{CP_i + p_{i,j} * d_j + L_j^{DT}} \right) * p_{i,j}. \quad (6.1)$$

Average arrival rate of the queue is λ which is found with:

$$\lambda = \sum_{j=1}^c \lambda_j. \quad (6.2)$$

Table 6.1. Key System Parameters

Parameter	Description
CP_i	Decision making period of PSM.
m	Number of client groups which utilize IoT application. A client group i represents the clients with same CP_i value.
M_i	Number of subscribed clients in client group i . M is the sum of all M_i values and represents the number of clients subscribed to the system.
$p_{i,j}$	Probability of generating class j data packet for client group i . $p_{i,j}$ implies the average workload sending rate of the IoT service in the fog.
c	Number of traffic classes served by the queuing system. FIFO scheduling policy considers c as 1.
d_j	Average duration for decision making and generation of j^{th} type decision packet in the system.
L_j^{DT}	Average data packet transmission latency from PSM to FC for a class j data packet.
μ	Average service rate of FC for any one of the decision packets in the queue. $1/\mu$ gives the average service time needed to forward packets.
k	Capacity of the finite size queuing system.

Utilization or traffic rate of the class j data packet traffic in the queuing system is represented by ρ_j . This parameter can be calculated using the following equation:

$$\rho_j = \frac{\lambda_j}{\mu}. \quad (6.3)$$

Total utilization or traffic rate of the queuing system is represented by ρ which can be calculated as follows:

$$\rho = \sum_j \rho_j. \quad (6.4)$$

The drop rate of the finite size queue is represented by r_{drop} . The following formulation gives the drop rate in which π_k represents the steady-state probability of the $M/M/1/K$ queue to have k packets. We obtain the formulation of π_k with the birth-death process of $M/M/1/K$ queue [48].

$$r_{drop} = \pi_k(k, \rho) = \frac{\rho^k - \rho^{k+1}}{1 - \rho^{k+1}}. \quad (6.5)$$

Waiting ratio (or wait ratio) of the packets in an FIFO scheduling policy based $M/M/1/K$ queue is represented by r_{wait_K} . We define the waiting ratio as the ratio between the average waiting time of packets spent in the queuing system (i.e, response time of the queuing system for a packet, the sum of the time spent in the queue and the service of a packet), and the average time needed for the service. We calculate r_{wait_K} value using the L_K value which represents the average packet number in the queuing system and the π_k value for any given k value, by using the following formulation:

$$L_K(k, \rho) = \frac{\rho}{1 - \rho} - \frac{(k + 1)\rho^{k+1}}{1 - \rho^{k+1}}. \quad (6.6)$$

$$r_{wait_K}(k, \rho) = \frac{L_K(k, \rho)}{\rho * (1 - \pi_k)}. \quad (6.7)$$

Average waiting ratio for $M/M/1$ queue is represented by r_{wait} . The following equation gives this value according to the ρ .

$$r_{wait} = \frac{1}{1 - \rho}. \quad (6.8)$$

Relative average waiting ratio of $M/M/1/K$ queue to $M/M/1$ queue is represented with δ_K . It can be found with the following equation.

$$\delta_K(k, \rho) = \frac{r_{wait_K}}{r_{wait}}. \quad (6.9)$$

Average waiting ratio of class l data packets in the $M/M/1/PQ$ queuing system is

represented with r_{wait_l} . We can obtain r_{wait_l} for l^{th} priority level queue with the following:

$$r_{wait_l} = 1 + \frac{\rho}{R_{l-1}R_l}. \quad (6.10)$$

where, $R_0 = 1$ and for any $l > 0$, $R_l = 1 - \sum_{b=1}^l \rho_b$

$r_{wait_{K,l}}$: Average waiting ratio of class l data packets in non-preemptive $M/M/1/K/PQ$ queuing system.

$$r_{wait_{K,l}} = r_{wait_l} * \delta_K(k, \rho_l). \quad (6.11)$$

Using the model defined with the equations and the theory given above, we can argue that:

(1) r_{drop} decreases drastically as k increases. When $k \geq 80$, r_{drop} approximates 0 very closely for any ρ . From this point on, the system begins not to drop packets and a finite-size queuing system approaches an infinite size $M/M/1$ queuing system. **(2)** δ_K values rise as k increases, and they approach 1. This means that if we choose a higher packet capacity for the queuing system, the average waiting time in the queue will increase. Moreover, with a higher packet capacity, average waiting times in $M/M/1/K$ queuing system approximates to those in $M/M/1$ queue. **(3)** A queuing system with finite and low capacity has a better average waiting time performance compared to an infinite queue. However, lowering queue capacity leads to higher average drop rates in the system. **(4)** r_{drop} and δ_K vary inversely, as k and ρ change. If we consider only r_{drop} values, it is obvious to choose a high k value for a lower drop rate. However, in this case, high k values cause higher wait times. So, choosing a small k for the queuing system results lower wait times but leads to higher drop rates. **(5)** Figures 6.1-6.3 depict the change of $r_{wait_{K,l}}$ over the traffic rate with different number of priority levels in a $M/M/1/K/PQ$ queue. For lowest priority packets, the wait time increases exponentially with traffic rate (ρ), which means that the lowest level of the queue usually starves in high traffic rates and, gaps between average waiting times of lowest and highest priority packets become apparent. This reveals that using the FIFO policy with a limited size queue instead of the MPDQ policy in the scheduling leads to reduced waiting times for packets in the lowest level queue. On the other hand, the FIFO policy is detrimental

for higher level queues as it may increase the waiting times. As mentioned in Section 5.4.3, we need a scheduling policy to eliminate these shortcomings. Since our proposed scheduler is mainly on the FIFO policy, the average wait times of lower level priority queues are kept low, very close to the ones obtained from a conventional FIFO scheduler. However, unlike FIFO, our scheduler preserves the waiting times of high priority packets by leveraging low overhead packet sequence adjustment. Since it can act both as an FIFO or an MPDQ scheduler based on priority, the proposed scheduler can be modeled in two parts. For high priority packets, the scheduler can be modeled as an MPDQ scheduler, while for low priority packets, it theoretically performs as an FIFO scheduler.

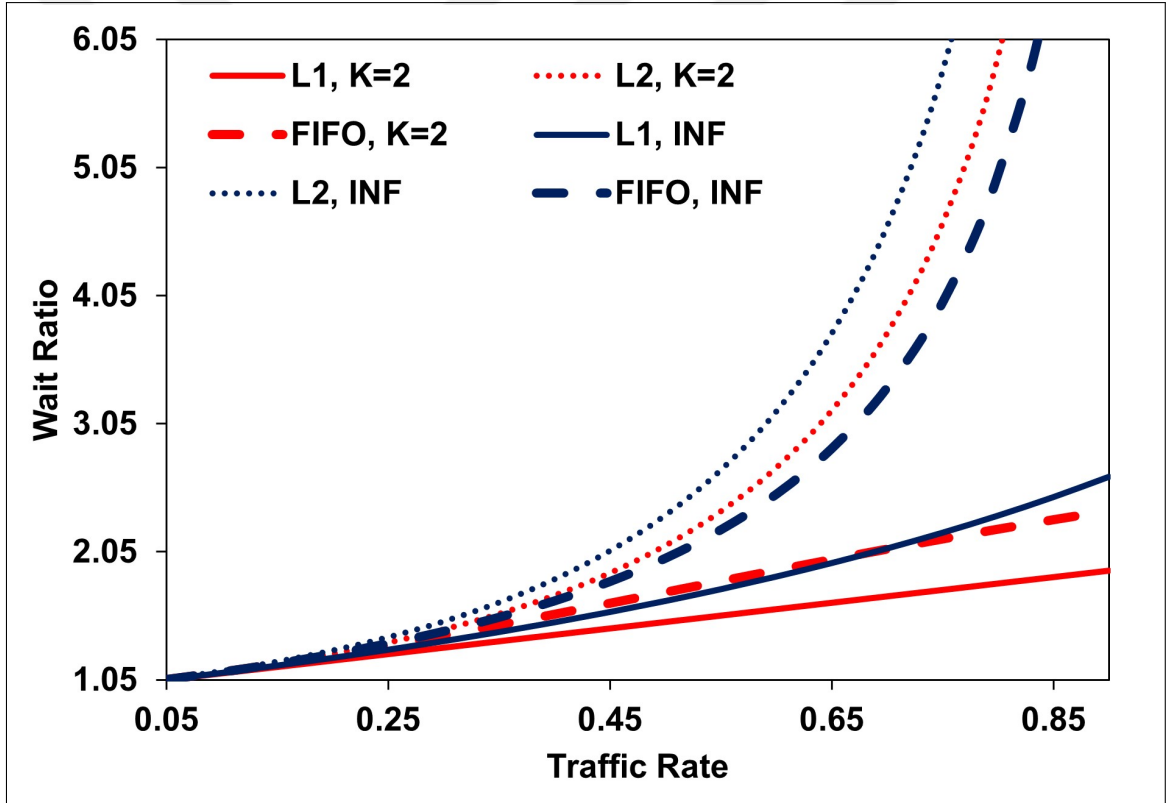


Figure 6.1. Change of $r_{wait_{k,l}}$ respect to ρ when $K=2$ or INF , $\rho \leq 0.9$, for 2 levels

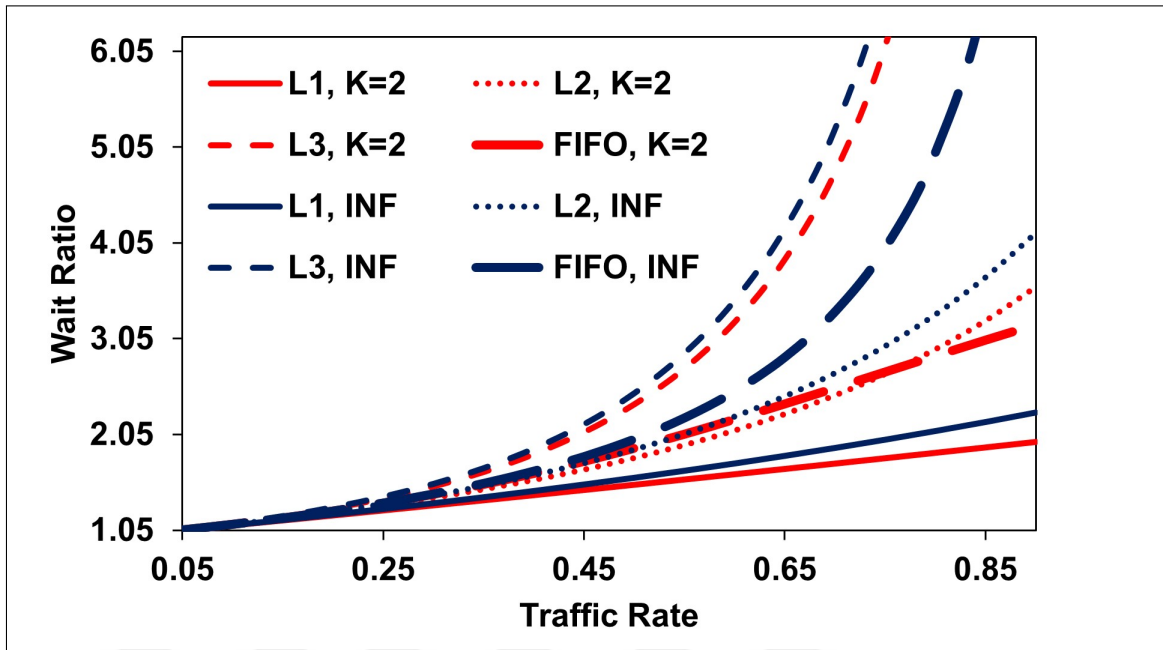


Figure 6.2. Change of $r_{wait_{k,l}}$ respect to ρ when $K=2$ or INF , $\rho \leq 0.9$, for 3 levels

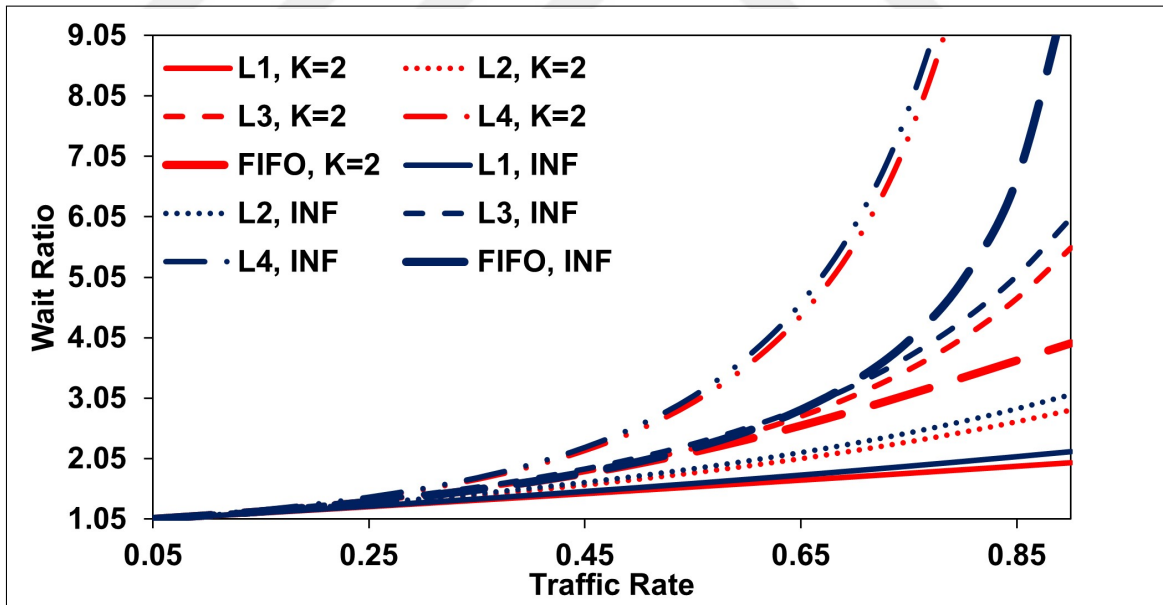


Figure 6.3. Change of $r_{wait_{k,l}}$ respect to ρ when $K=2$ or INF , $\rho \leq 0.9$, for 4 levels

6.2. OPTIMAL QUEUE SIZE ANALYSIS

It is crucial to determine a k value that results in both optimal packet drop rate and wait time for the queuing system. Equation 6.12 gives the formula of a scoring mechanism for this purpose. Using the given ρ and k values, a $\omega(k, \rho)$ value also called as the score is calculated, that represents the quality of optimization for the candidate capacity of the queuing system. Figures 6.4 and 6.5 illustrate the change in $\omega(k, \rho)$ with respect to ρ values corresponding to low and high traffic rates, for different k values. The k_{opt} value, is the optimal k value that yields the highest $\omega(k, \rho)$ score with optimal drop rates and wait times for any given ρ . Table 6.2 gives the corresponding k_{opt} values for changing ρ . From this analysis that takes into account the theoretical basis of the finite size queues, we can conclude that queue capacity that does not exceed 5 is optimal and sufficient for smaller ρ values. This limit is 35 for high traffic rates.

$$\omega(k, \rho) = \frac{1}{\rho}(k - 1)(1 - \delta_{MM1K}(k, \rho) - r_{drop}) \quad (6.12)$$

Table 6.2. Optimal Queue Capacities

ρ	k_{opt}	r_{drop}	δ_{MM1K}	top $\omega(k, \rho)$
0.1	2	0.009	0.9818	0.0917
0.2	2	0.032	0.9333	0.1720
0.3	3	0.019	0.9417	0.3249
0.4	3	0.039	0.8769	0.5168
0.5	4	0.032	0.8666	0.7354
0.6	4	0.056	0.7617	1.0974
0.7	6	0.038	0.7599	1.6593
0.8	8	0.038	0.6774	2.7738
0.9	16	0.022	0.6360	6.0405
0.95	34	0.01	0.6398	12.500

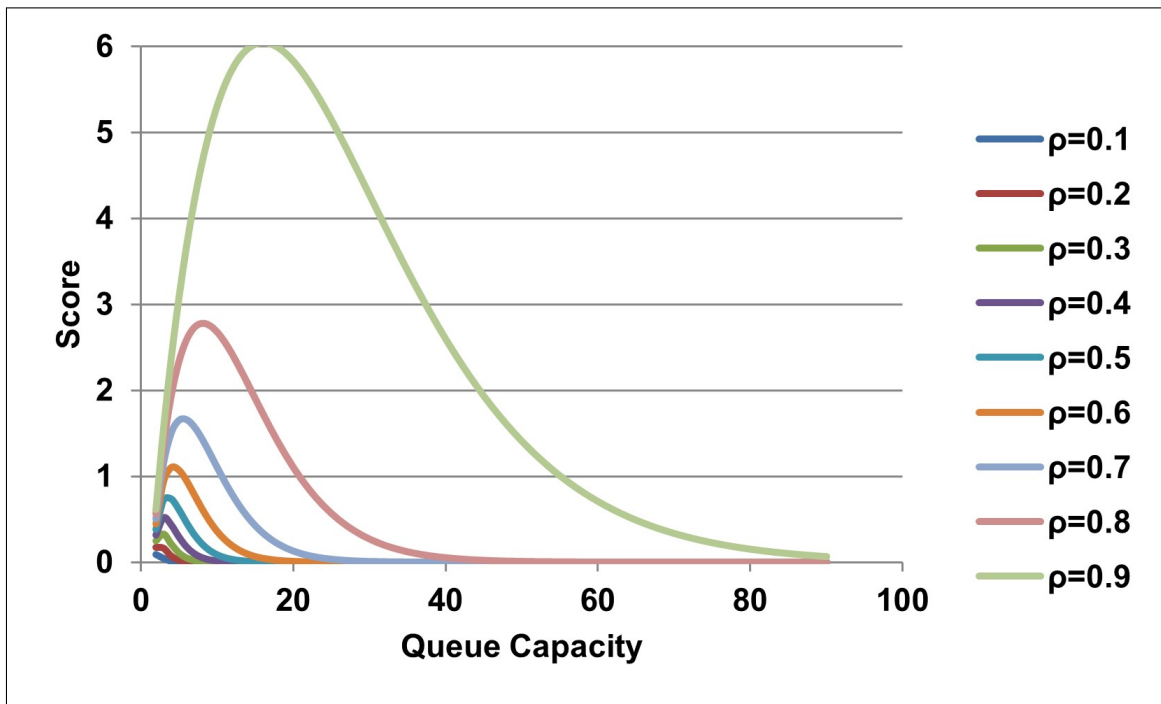


Figure 6.4. Change of $\omega(k, \rho)$ with respect to k , when $\rho \leq 0.9$.

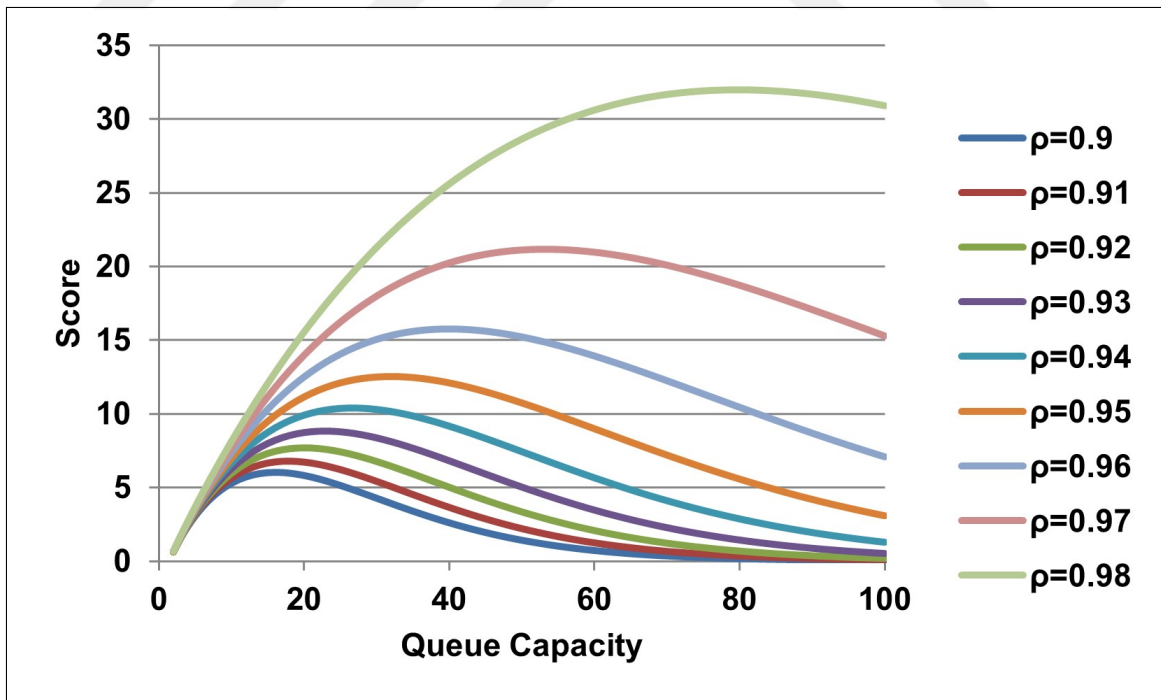


Figure 6.5. Change of $\omega(k, \rho)$ with respect to k , when $\rho \geq 0.9$.

7. TESTS AND RESULTS

This chapter presents the tests and analysis of the results, firstly, with giving the details of the test bed setup, secondly, with examining the characteristics of IATs of sensor data transmission then thirdly, with giving the detailed analysis of the tests devised for the proposed approaches of the thesis.

7.1. TESTBED SETUP

In this section, we present the real and emulation based test beds with giving the details of the setup and how we realize the proposed FoT architecture.

7.1.1. Real Testbed Setup

For evaluation of the freshness scheme described in Section 5.2.1 and collection of the sensor data traces from the environment, we devise tests in a real testbed and obtain sensor data traffic traces coming from edge network towards the PSM. The real testbed consists of a sensor network (i.e., the edge) connected with a local network of computers having 8 GB RAM and Intel 8th Generation i5 CPU (i.e., the fog). We connect sensor devices located the edge network to the fog domain with adaptation to NGSI messages, which is performed by SC.

For the edge network, we use two alternatives of the sensor networks. In the former, we use low power sensor nodes in a LoWPAN. In the latter, we use sensor nodes communicating with WiFi. The system parameters used in these test bed deployments are presented in Table 7.1.

7.1.2. Emulation Based Setup

For the evaluation of the data traffic scheduling and orchestration approaches, we devised tests in a emulation based testbed. We prefer such an test environment with mainly

considering the scalability of the components in the architecture and ease-of-setup of the software components in the network designed for FoT.

The test-bed built for the emulation based experiments consists the same setup of the real test bed, except that we simulate sensor nodes and clients in virtual processing environment using an emulation procedure. Additionally, in order to obtain precise time measurements, we run the software designed for SC, PSM, FC, CCS in a single computer. For the processing of CB, we use a different machine located in the same LAN of the fog. These two computers are connected via a 100-Mb/s network link.

As a use-case for the emulation based tests, we simulate a typical client-rule based periodic sensor data monitoring IoT application that generates decisions for different priority classes. We prefer such a use case to create aggregated short-lived and low-rate IoT traffic of different priority packets for the centralized FC with many scales. In this hypothetical use-case, we mainly focus on our proof-of-concept study for the proposed data traffic management approach and queuing model based data traffic scheduling policy.

In the emulation based test-bed, we perform an emulation based sensor data transmission using a single thread of SC. This thread uses a periodic timer that expires in every SP/N milliseconds. After each expiration of this timer, the thread generates and sends sensor data of the related sensor node to the fog. We emulate clients with a multi-threaded process and, each client thread subscribes to the PSM with a sensor data monitoring rule. The PSM generates decisions according to client rules, marks each decision packet by assigning different priority levels and transfers them to the FC through the CB. For the implementation of CB, we use Fogflow framework [28], as it is suitable for NGSI-HTTP [95] based subscribe/update-notification based data flow in the fog architecture.

For the MQ based data transfer orchestration, we use the Kafka framework [100] and its Producer-Consumer message queue based message delivery pattern over the conventional context broker of the Fogflow.

Table 7.1. Test Parameters

Parameter Name	Value
CEP engine for scheduling	Esper
MAC for LoWPAN	IEEE 802.15.4
MAC for WiFi	IEEE 802.11a
Communication Topology	Star
Radio Network Communication Stack	Rime [125]
Sensor Data Sampling Period (SP)	1s,2s,5s
Sensor Data Size (S)	30 - 3000
Number of Sensor Nodes (N)	1 - 256
Client Defined Decision Sending Period (CP)	1s, 2s, 5s, 10s
Physical Data Acquisition overhead (PA)	250ms
Serial Interface Rate (SR)	115200 bps (~ 10 ms)
Number of Clients M	1 - 800
r_1 (See Chapter 6)	1
m (See Chapter 6)	1
c (See Chapter 6)	2
$p_{1,1}$ (See Chapter 6)	0.5
$p_{1,2}$ (See Chapter 6)	0.5
k (See Chapter 6)	2 to 10
Sensors	Temperature, Humidity, Light
Number of repeats in test cases	20
Number of samples obtained in tests	200

7.2. INTERARRIVAL TIME ANALYSIS

As we described in Section 7.1.1, in the real testbed, we obtain IAT sequences of the sensor data transmission from the edge network towards the PSM. We analyze the distribution

of these IAT sequences by generation of histograms which show the PMF distribution percentages according to the mean and standard deviation of IATs. Figures 7.2 - 7.16 show the observed IAT samples of $P=2s$ and IAT occurrence frequency histogram of $P=2s$ for IEEE 802.15.4 medium. Figures 7.9 - 7.16 show the results for $P=5s$.

As illustrated in Figures 7.2 - 7.4 and 7.9 - 7.12, with increasing N and in a lower P , race for the sensor data sending and the medium complexity of the sensor network connected to fog increase. Thus, IAT samples starts to fluctuate more. So, the occurrence frequency for the average IAT starts to decrease with increasing N . However, as illustrated in the histograms in Figures 7.5 - 7.8 and 7.13 - 7.16, the occurrence frequency for IATs which situates near to $P + PA$ is more than 80 percent of all IAT samples. So, in the worse case for the tests, less than 20 percent of IAT fluctuates.

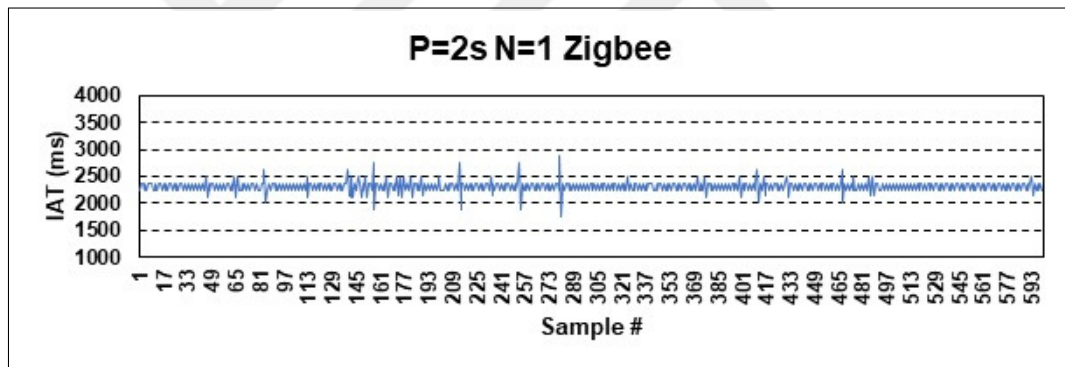


Figure 7.1. IAT samples when $P=2s$ and $N=1$

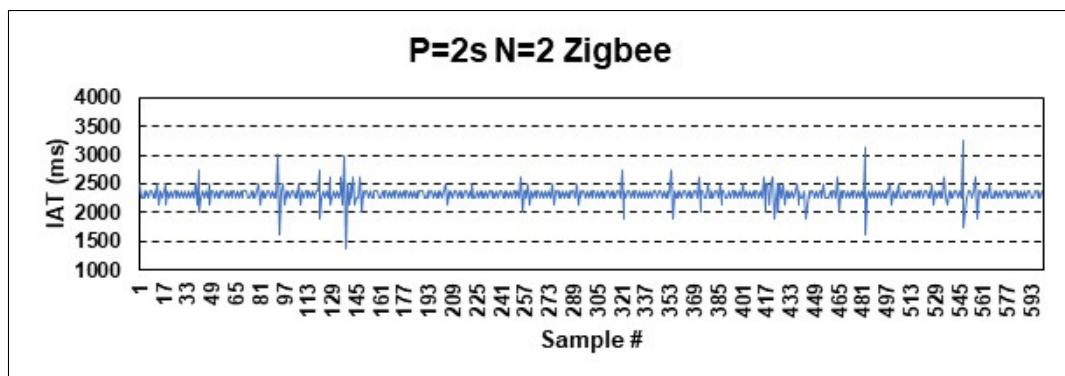


Figure 7.2. IAT samples when $P=2s$ and $N=2$

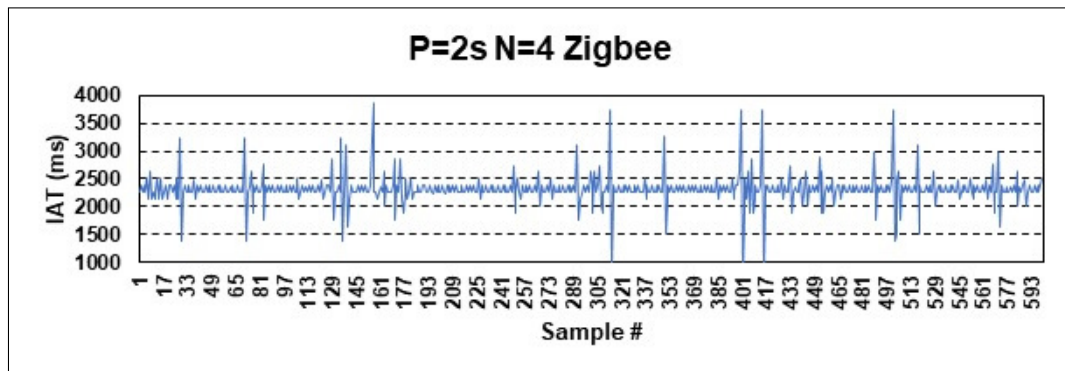


Figure 7.3. IAT samples when P=2s and N=4

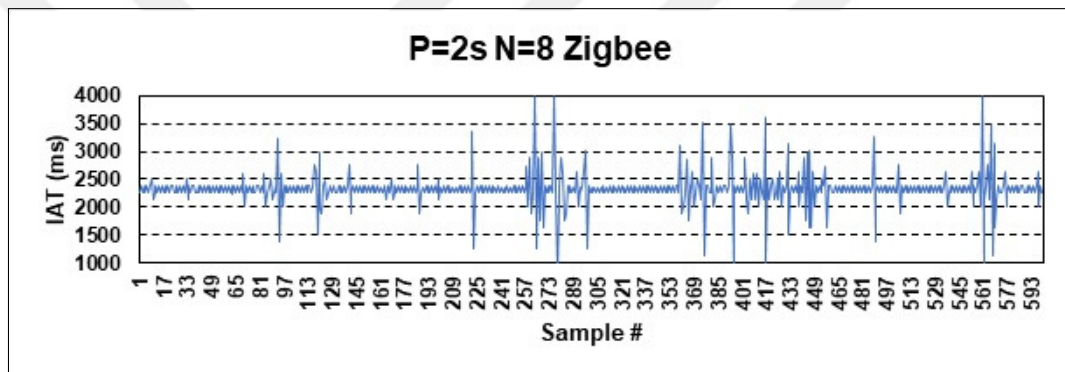


Figure 7.4. IAT samples when P=2s and N=8

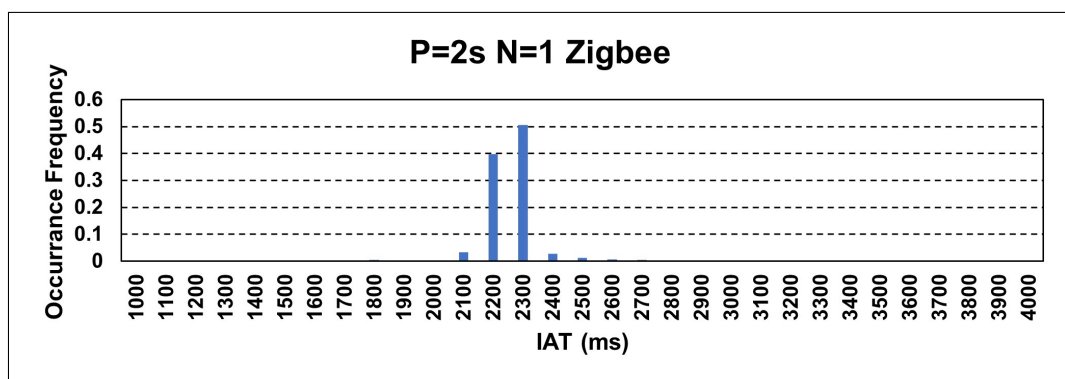


Figure 7.5. IAT occurrence histogram for P=2s and N=1

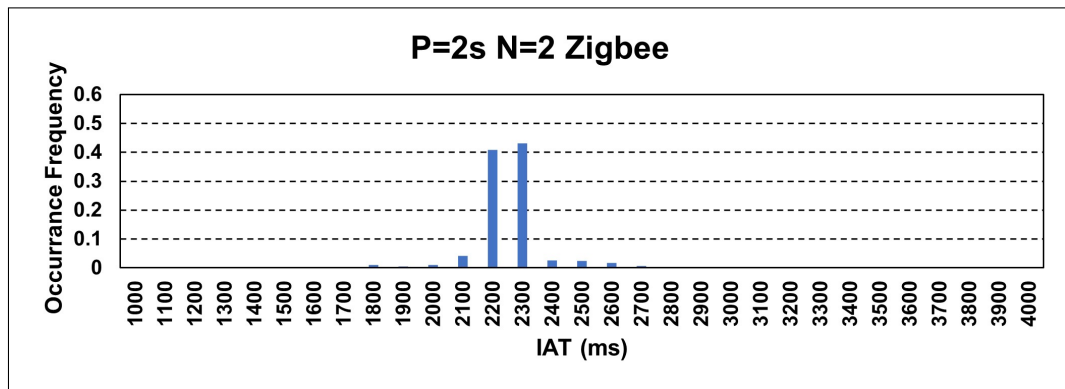


Figure 7.6. IAT occurrence histogram for P=2s and N=2

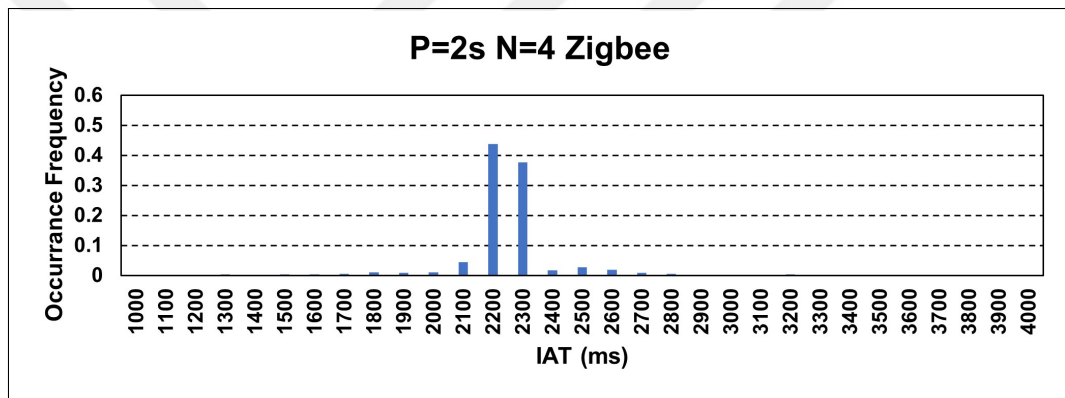


Figure 7.7. IAT occurrence histogram for P=2s and N=4

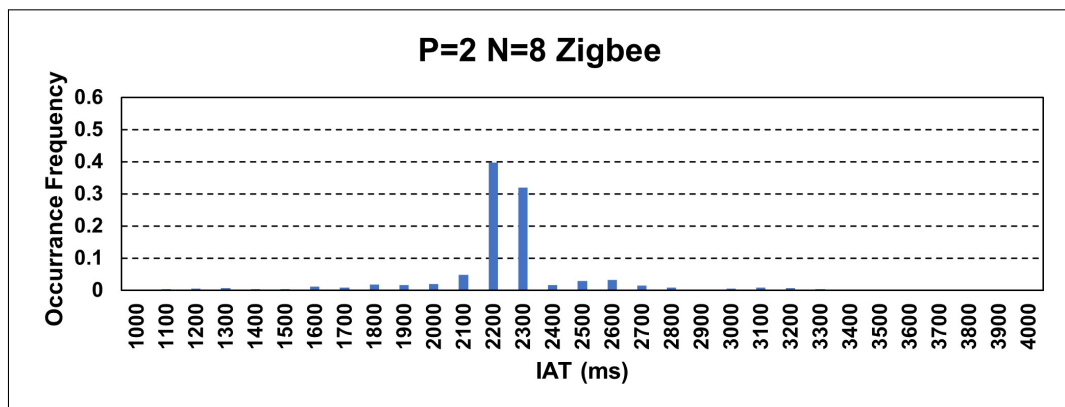


Figure 7.8. IAT occurrence histogram for P=2s and N=8

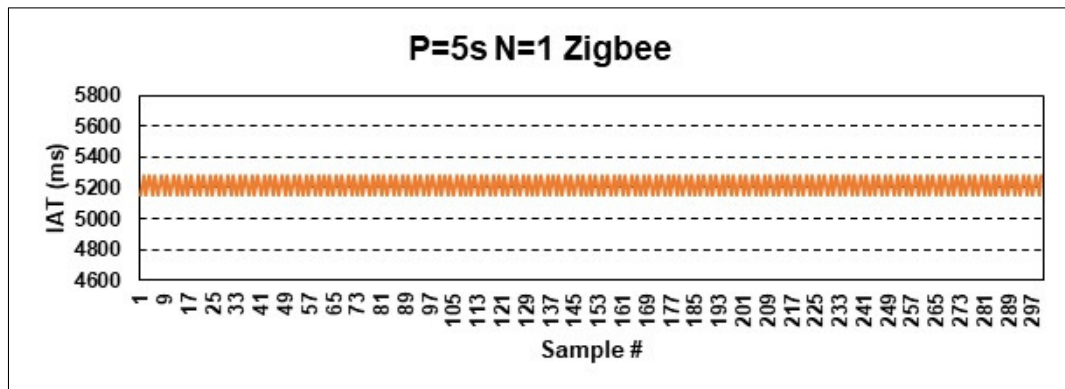


Figure 7.9. IAT samples when $P=5s$ and $N=1$

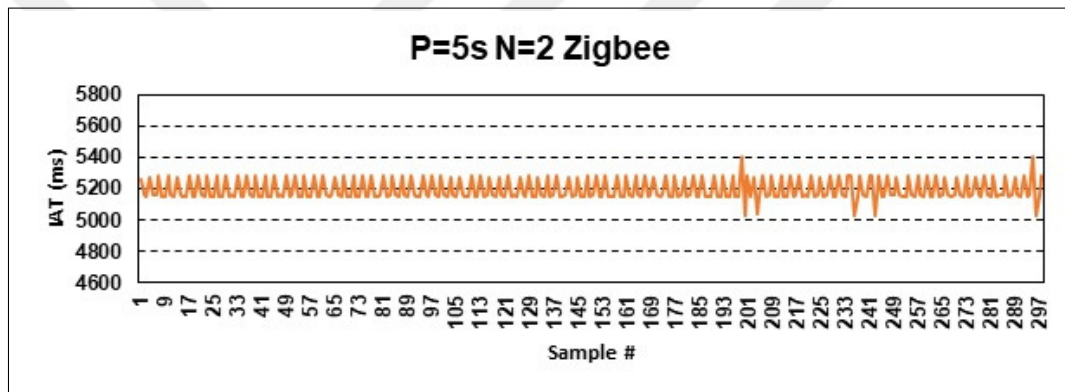


Figure 7.10. IAT samples when $P=5s$ and $N=2$

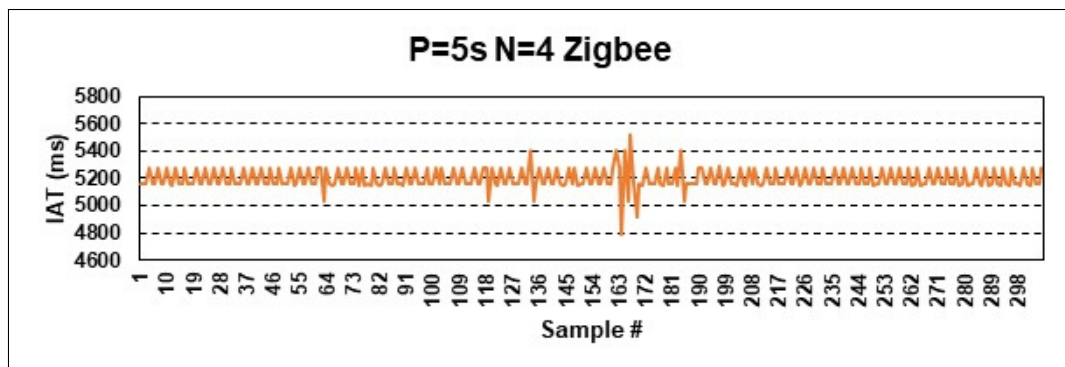


Figure 7.11. IAT samples when $P=5s$ and $N=4$

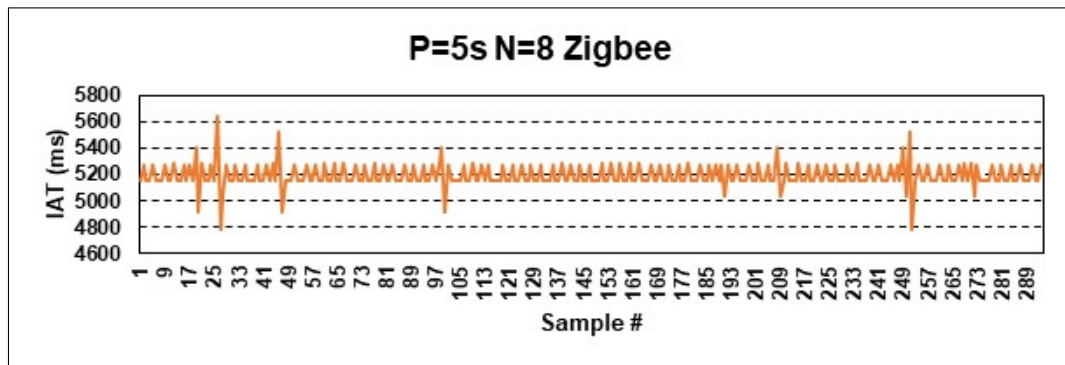


Figure 7.12. IAT samples when $P=5s$ and $N=8$

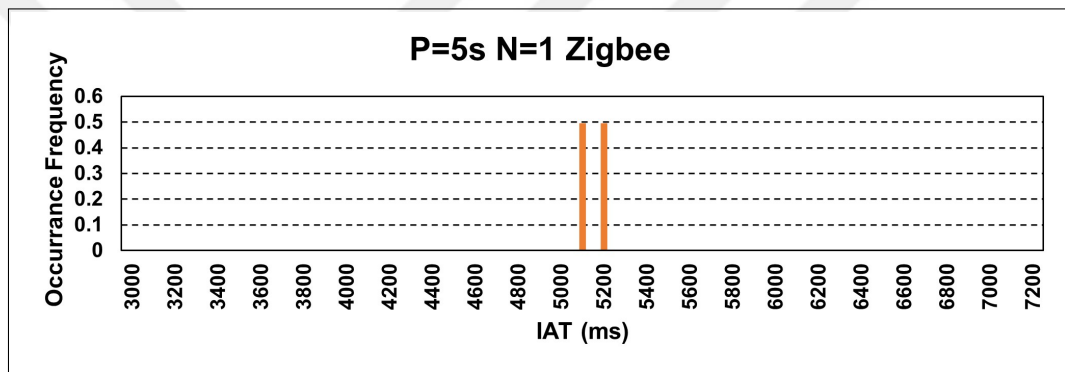


Figure 7.13. IAT occurrence histogram for $P=5s$ and $N=1$

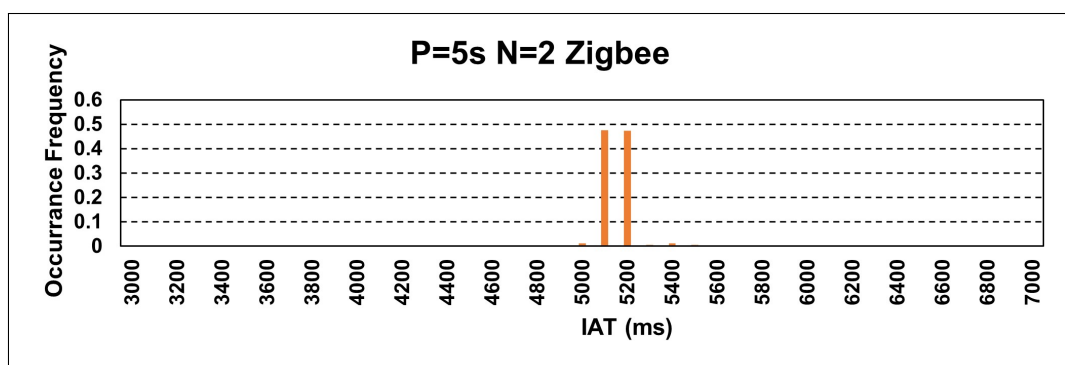


Figure 7.14. IAT occurrence histogram for $P=5s$ and $N=2$

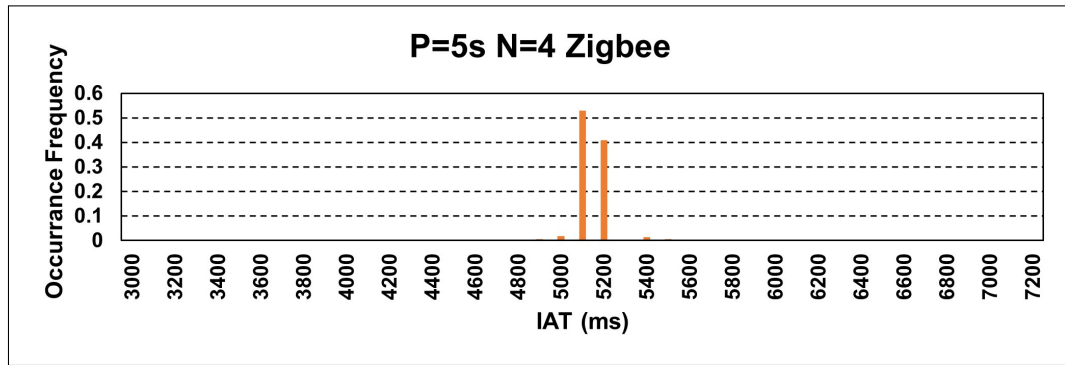


Figure 7.15. IAT occurrence histogram for P=2s and N=4

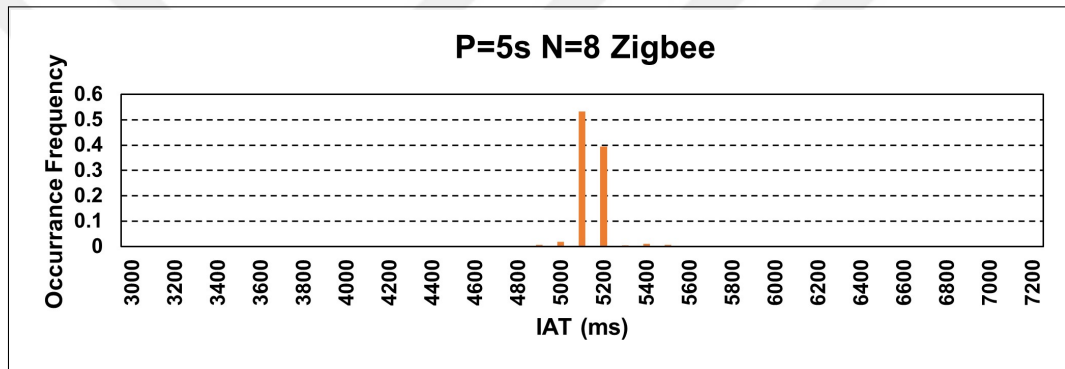


Figure 7.16. IAT occurrence histogram for P=5s and N=8

7.3. SENSOR DATA FRESHNESS TESTS

For the sensor data freshness tests, firstly we evaluate the fixed data freshness time calculation method. For the analysis considering the fixed data freshness time calculation method, we apply FT starting from 1s to 7.5s with emergency detection scheme with different pessimistic percentages into the time trace set.

Figure 7.17 illustrates the freshness rate results for different N and FT with a pessimistic percentage of 100 for $P=2s$. In addition, Figure 7.18 illustrates the results for different pessimistic percentages for $P=2s$ and $N=8$. Figures 7.19 and 7.20 illustrates the results when $P=5s$. Figures 7.21 and 7.22 shows the results for WiFi nodes with $P=2s$ and 5s. As seen in these results, especially for the Zigbee environment, we can obtain a maximal

freshness rate when we apply FT more than $P + PA$. If FT is much less than $P + PA$, we can obtain less data freshness ratios. In addition as decreasing FT , FR decreases dramatically. Although setting up a real test bed using WiFi nodes and limiting N up to 8 leads the same conditions in the context of freshness rate, the freshness rate could be low if we choose FT less than $P + PA$. In addition, with an increasing N , same FT leads lower freshness rates. Different CP values affects the freshness rates with smaller ratios so we choose only the graphics for $CP = 2s$.

The results obtained from this analysis reveal that maximal freshness rate is strongly related with P , PA and the condition of the medium of the nodes in the sensor network of the fog domain. Thus, the application of any freshness time without knowing the conditions of the sensor data acquisition medium could lead low freshness rates of the decisions of emergency detection.

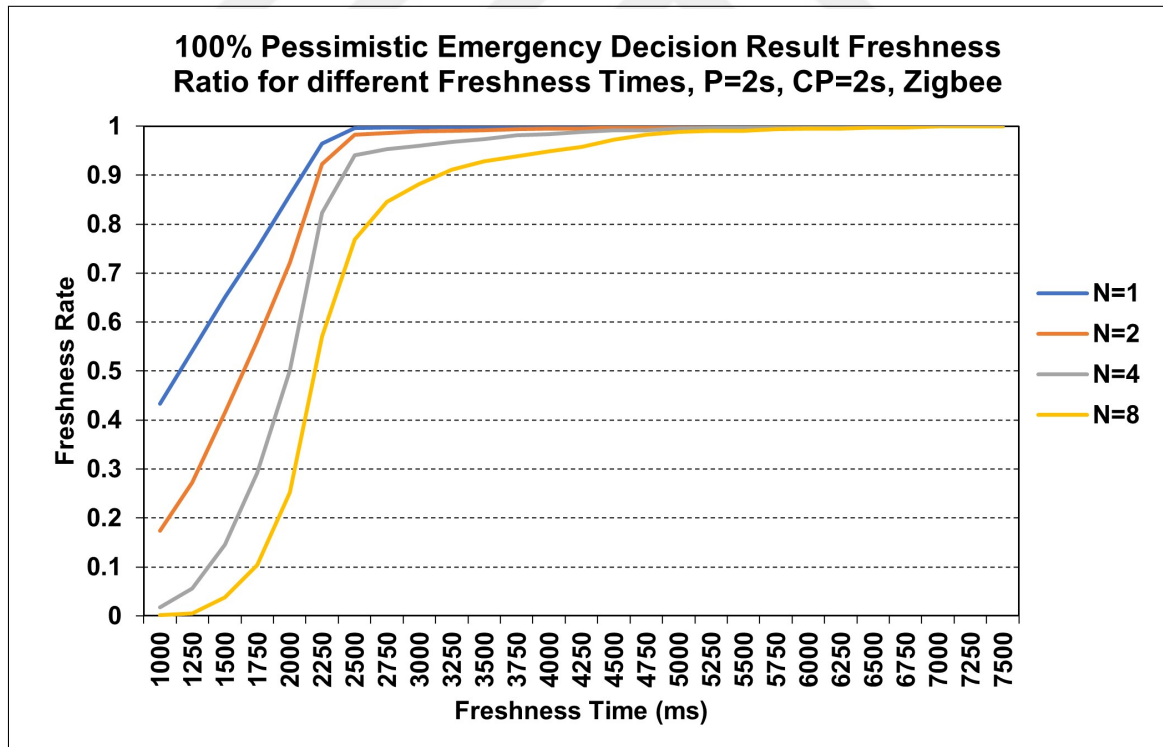


Figure 7.17. Freshness rates of static freshness time method for $P=2s$ and Zigbee

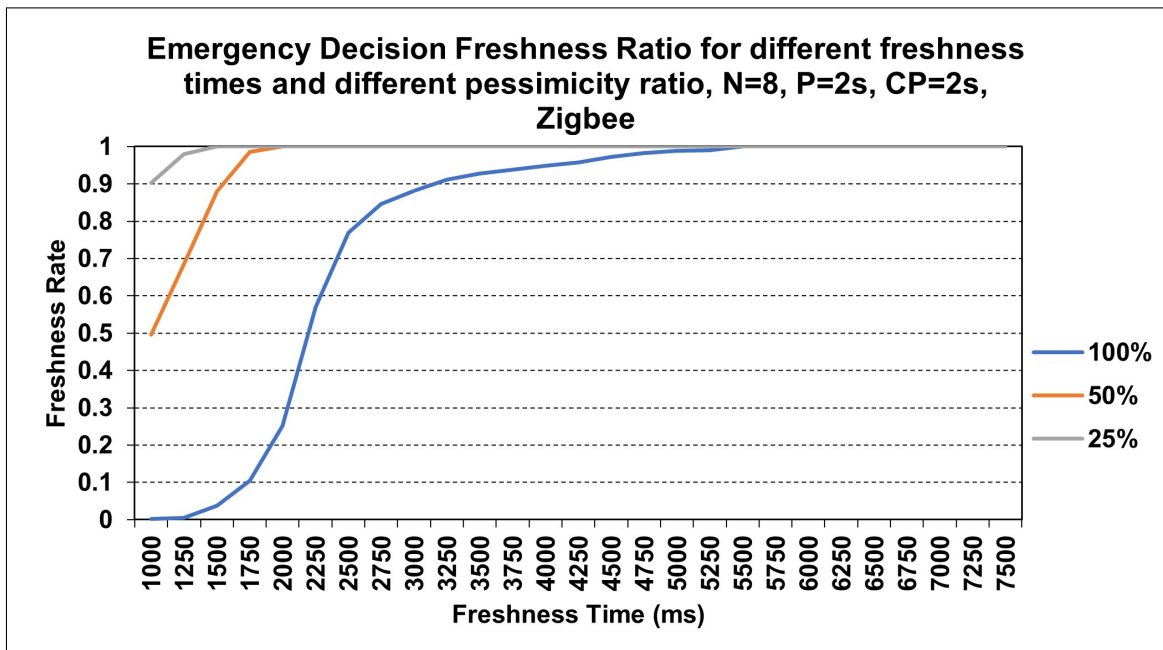


Figure 7.18. Freshness rates of static freshness time method for P=2s, N=8 and Zigbee

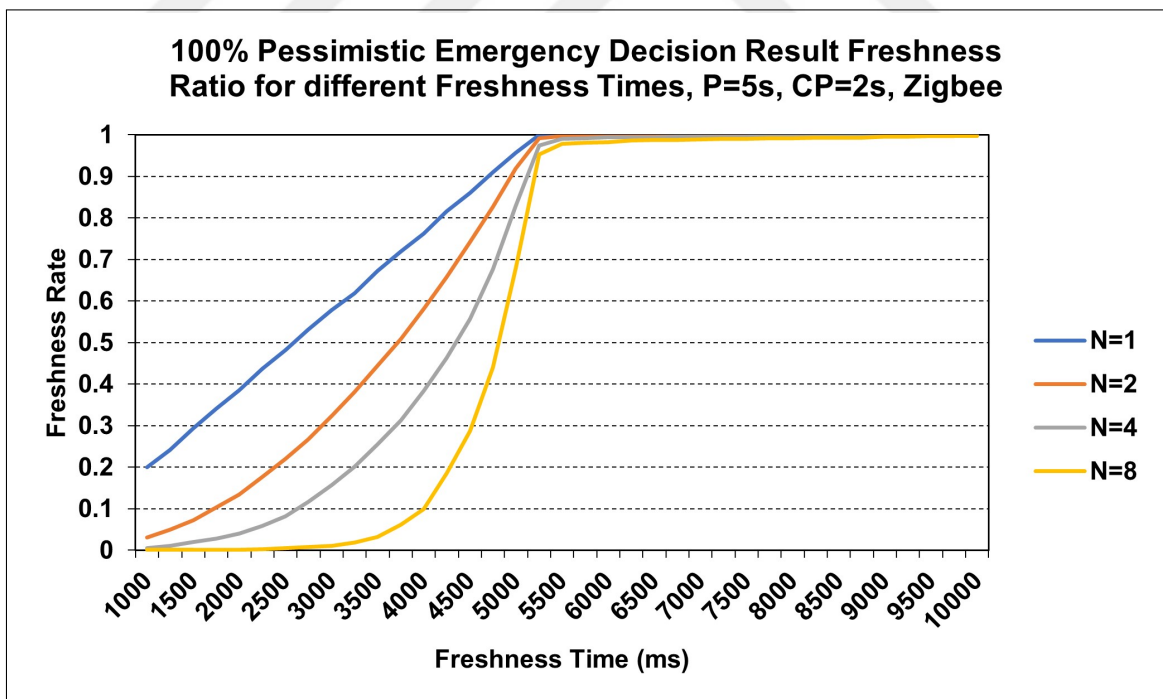


Figure 7.19. Freshness rates of static freshness time method for P=5s and Zigbee

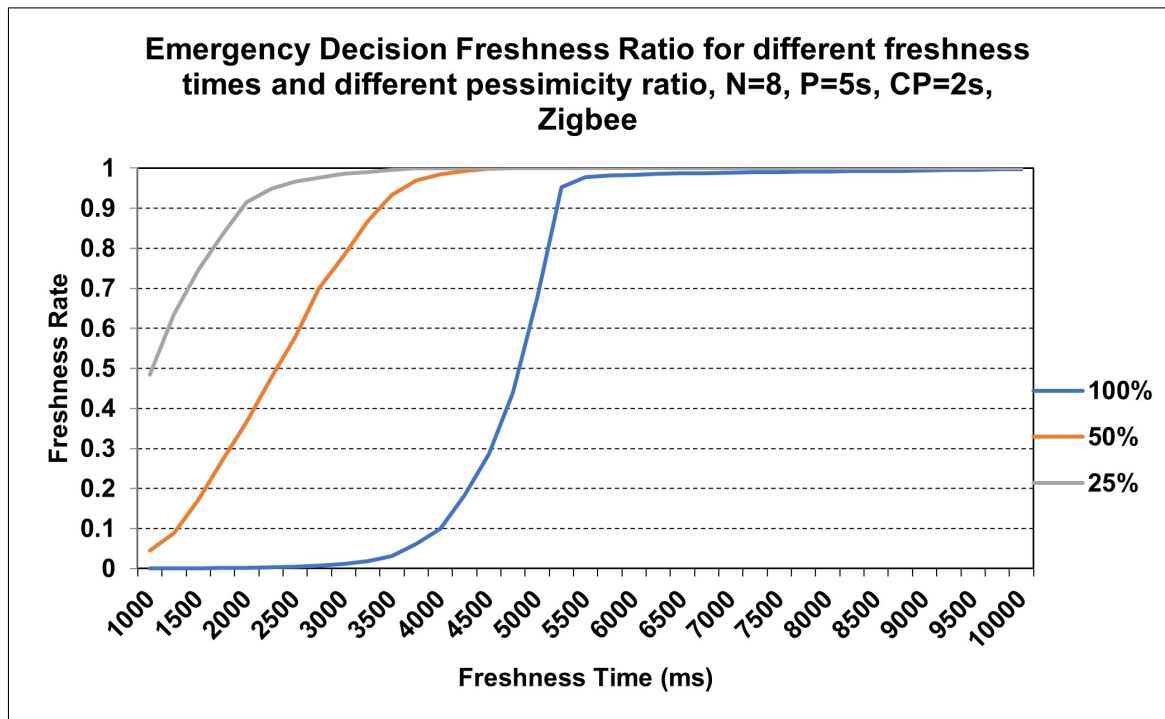


Figure 7.20. Freshness rates of static freshness time method for P=5s, N=8 and Zigbee.

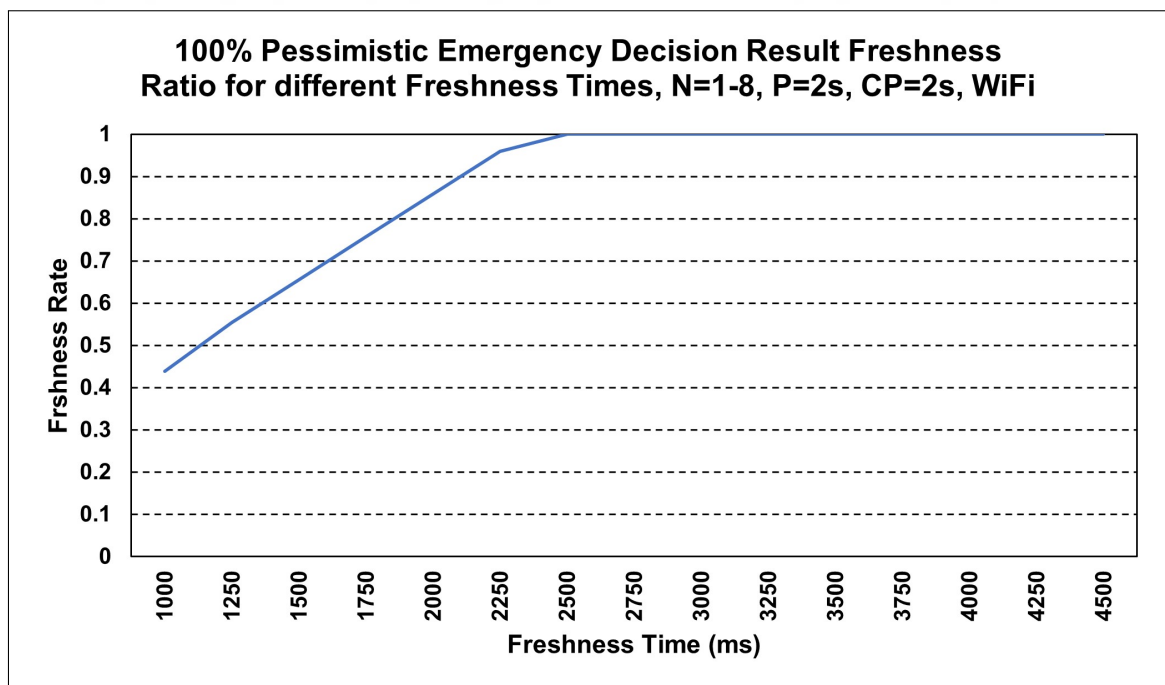


Figure 7.21. Freshness rates of static freshness time method for P=2s and WiFi.

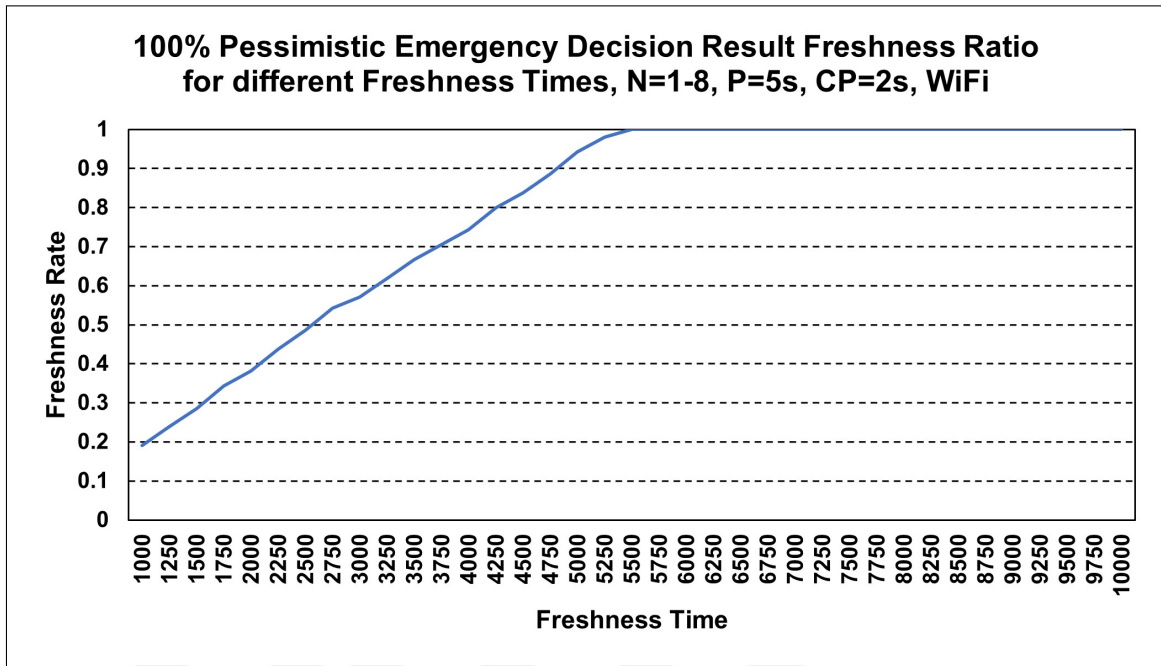


Figure 7.22. Freshness rates of static freshness time method for P=5s and WiFi.

7.3.1. Standard Deviation Based Freshness Time Analysis

We could consider the traffic characteristics of the sensor network of the fog to obtain a maximal freshness rate of the emergency decision results. We perform a two-step data freshness analysis to time traces and find emergency situation freshness rates. These steps are:

- (Step 1) : Calculate FT using the Equation 5.1 for time traces for same P and medium. Do this for different P and mediums.
- (Step 2) : Apply FT to each time traces and find FR .

The evaluation of this analysis is given in Section 7.3.3.

7.3.2. Probabilistic Data Freshness Time Analysis

In our analysis, firstly, we set g value to 0.125 and h value to 0.25 like in the Jacobson's RTO formula, secondly, we apply different g and h values. For each, we calculate freshness times with Equation 5.5 and find data freshness rates of the emergency situation detection with 100 percent pessimistic way.

The evaluation of this analysis is given in Section 7.3.3.

7.3.3. Evaluation of Freshness Time Analysis

Figure 7.23 illustrates the decision result freshness ratios with different freshness time calculation methods and sensor network mediums for $P=2$. Figure 7.24 depicts the results for $P=5$ s. The freshness rates reveals that application of gradually changed data freshness time calculation leads better data freshness ratios for same P and N . Because of that WiFi is more stable than Zigbee, the obtained data freshness rate is better.

7.3.4. Effect of Client Period

We analyze the effect of changing CP to the FR with applying different FT calculation methods. Figures 7.25 and 7.26 depicts the results for RTO-like FT calculation in terms of FR . As discussed in Section 5.2.1, increasing CP leads FR to decrease as expected. However, we do not observe dramatic decrease if we compare results for a low and a high CP values.

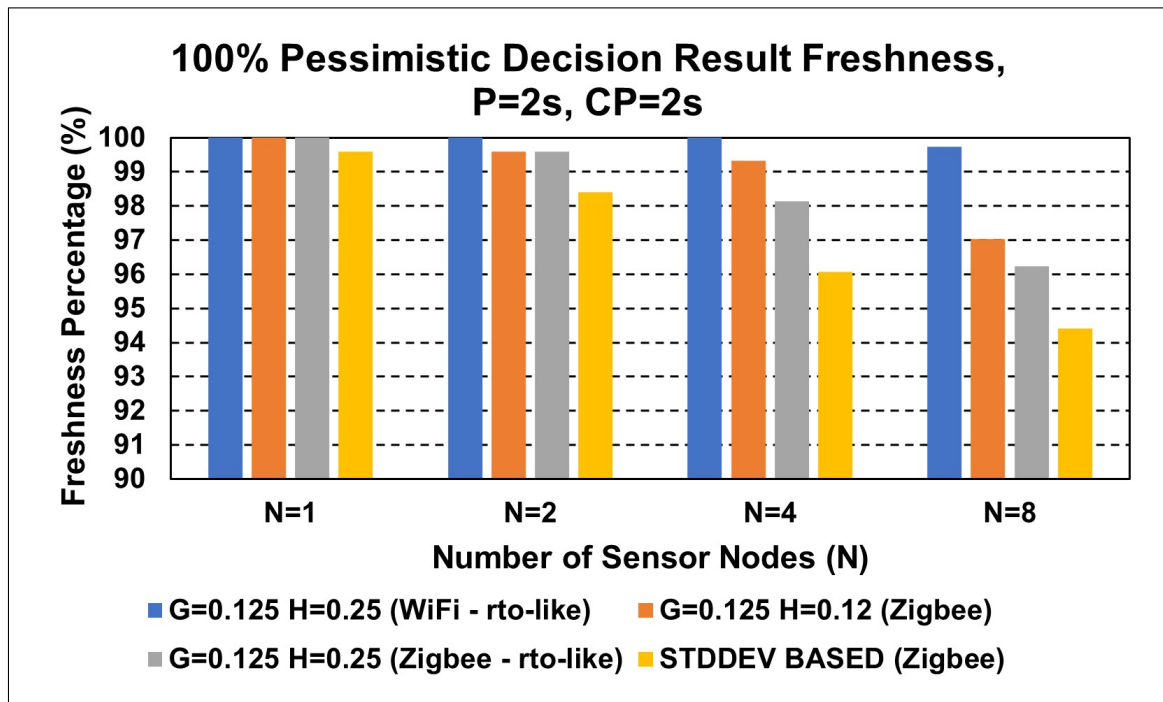


Figure 7.23. Freshness rates of several freshness time methods for P=2s.

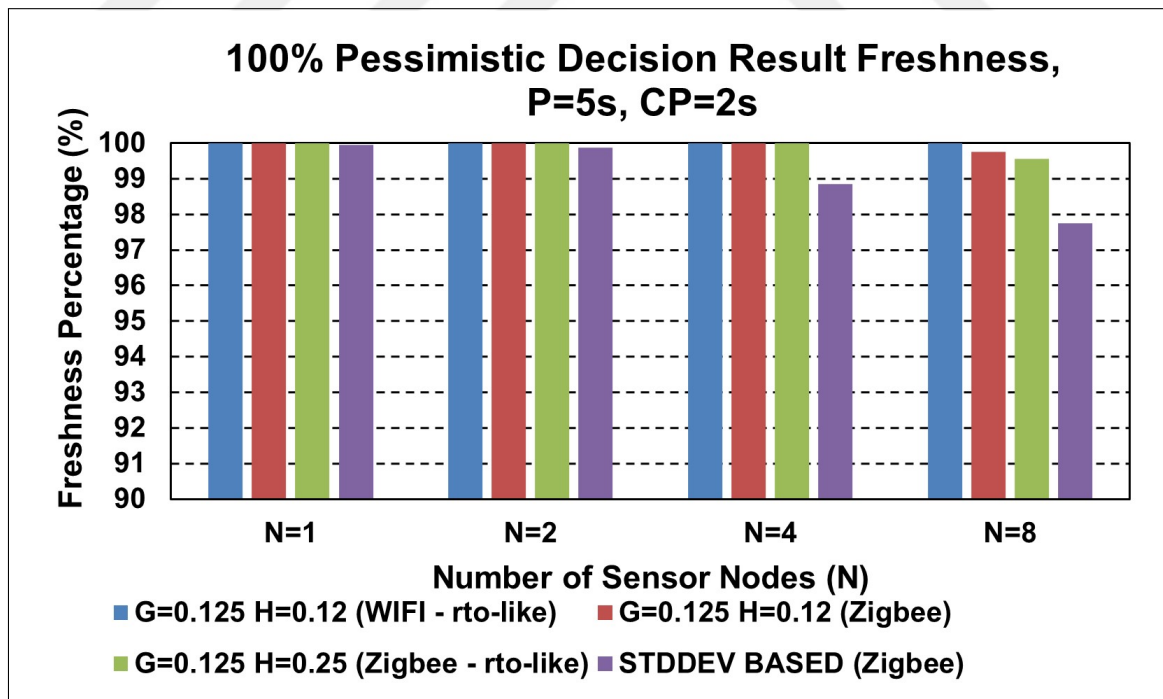


Figure 7.24. Freshness rates of several freshness time method for P=5s.

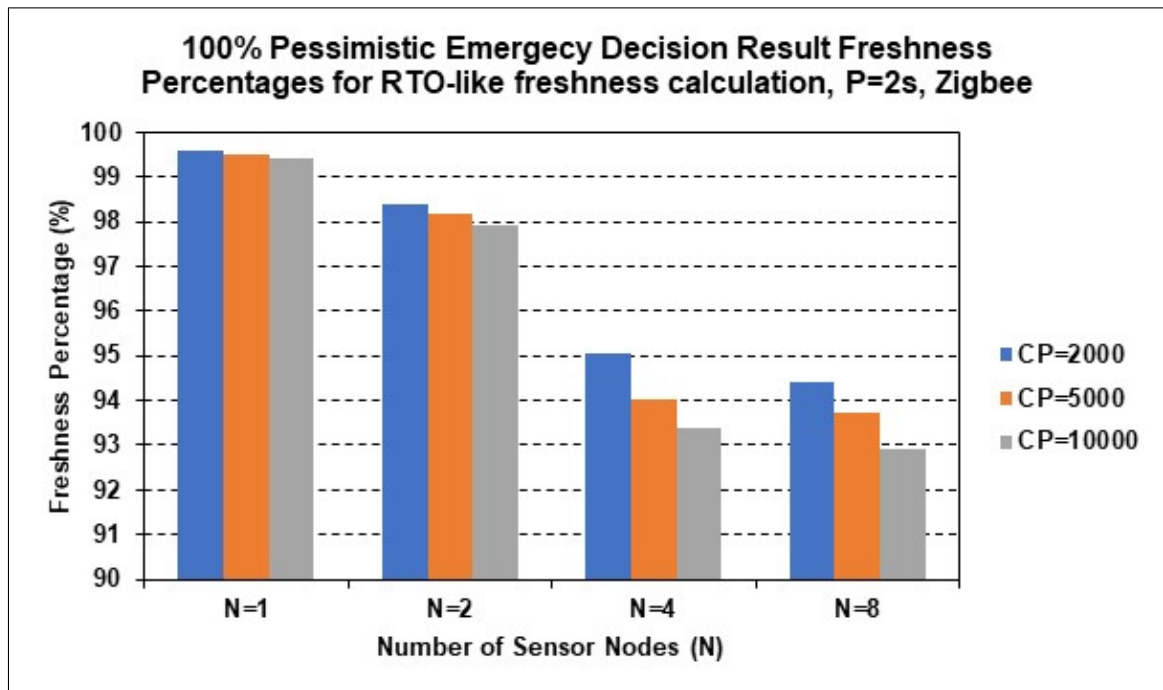


Figure 7.25. Freshness rates of RTO-like freshness time calculation method for P=2s.

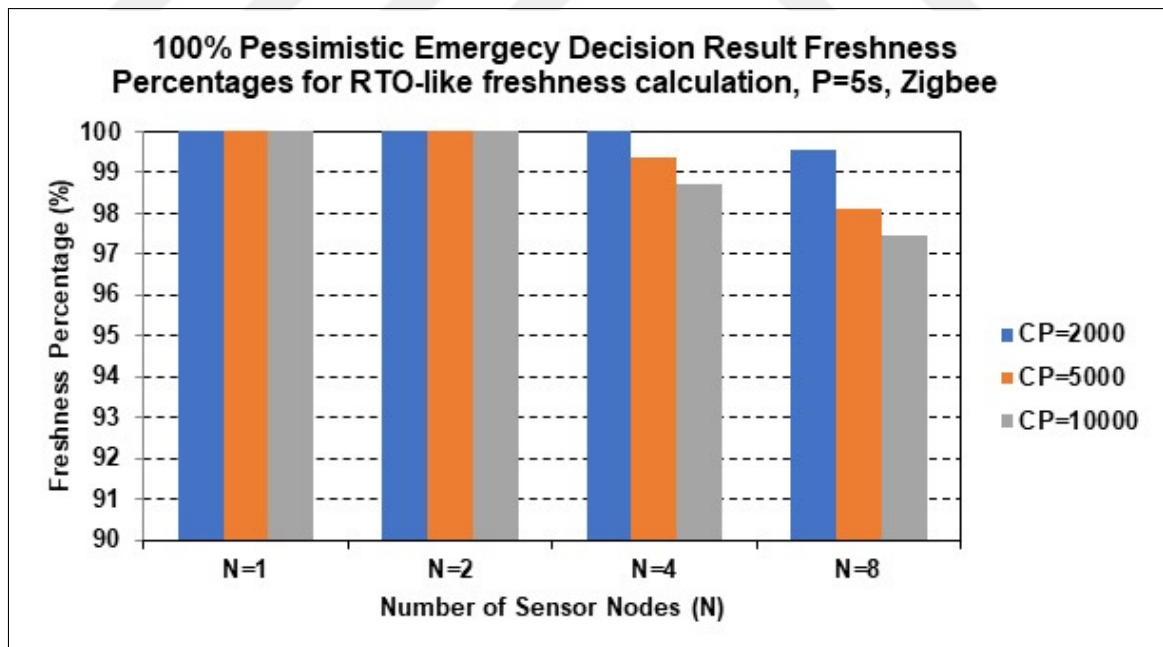


Figure 7.26. Freshness rates of RTO-like freshness time calculation method for P=5s.

7.4. FOG DATA TRAFFIC TESTS

This section provides the evaluation of the analysis for the inner fog latency when the data traffic flow methods described in Section 5.3 is operated.

7.4.1. RTT Tests

We take RTT tests and compare the latency of the small size HTTP message transmission between a host to a host in the fog and a host to a host in the Azure Cloud which is located in Western Europe (3400 km, 40ms). Table 7.2 gives the results. As seen in this table, even for a small packet, average latency faced in data transmission to cloud is much more than in the local network of the fog. This analysis reveals that, if we make decisions and make local reactions about the environmental factors occurring in our locals, using fog computing has an crucial benefit for reducing delays. Thus, it is beneficial to make decisions and make reactions as local as possible for delay sensitive applications.

7.4.2. Data Traffic Latency Tests

We devise experiments in two phases. In the first phase, in the given testbed setup, we perform periodic RTT tests and get an adequate number of RTT values of small size message packet transmission between machines in the testbed. We prefer the smaller size of packets for observing only the transmission latency without any extra delay in processing or waiting in the hosts. We calculate the period by dividing SP value given in Table 7.2 with a changing number of N . Table 7.2 gives average RTT values, with different N value, obtained in the first phase of the experiments.

In the second phase of the experiments, we devise comprehensive scalability tests to observe the latency of the inner fog with changing N and M , for each data flow scenarios. These experiments are based on simulation tests. In these tests, we simulate simultaneous sensor nodes in an SC, by generating a periodic CBR sensor data traffic to the fog. For a test case, we calculate this constant period by dividing SP value with the current N value of the test case. Besides, we simulate simultaneous clients with a multi-threaded program, in which

Table 7.2. Average RTT values obtained in the first phase of tests.

Lag after echo packet reception	Avg. RTT Value in Fog	Avg. RTT Value in Cloud
1000ms	2.753ms	41.718ms
500ms	2.328ms	40.981ms
250ms	2.061ms	40.552ms
125ms	1.971ms	40.257ms
62ms	1.8ms	40.126ms
30ms	1.714ms	40.987ms
15ms	1.573ms	40.713ms
7ms	1.412ms	40.651ms
4ms	1.282ms	40.251ms

each thread issues a client subscription rule including CP value to the fog, to periodically monitor the physical environment with sensors given in Table 7.1. Additionally, in this table, we give the values for parameters changed in the simulation tests and the number of repeats.

For observing the confidence intervals of $D_i^{s,B}$, $D_i^{s,BQ}$ values, we sketch the histogram of these values and determine the distribution of the occurrence frequency of the values, with different N values, for each scenario. Figure 7.27 - 7.36 present the histogram for these values. We can argue that occurrence frequencies of $D_i^{s,B}$, $D_i^{s,BQ}$ values follow a normal distribution and, these frequencies have a peak in the near of the mean values. Additionally, we do further analysis for the distribution of these values by changing M . As illustrated in Figures 7.37 - 7.39, we observe a minimum positive shift on the distribution curves. This analysis reveals that most of $D_i^{s,B}$, $D_i^{s,BQ}$ values situate around their mean values with changing N, M and, the confidence interval of these values places on near of the mean values.

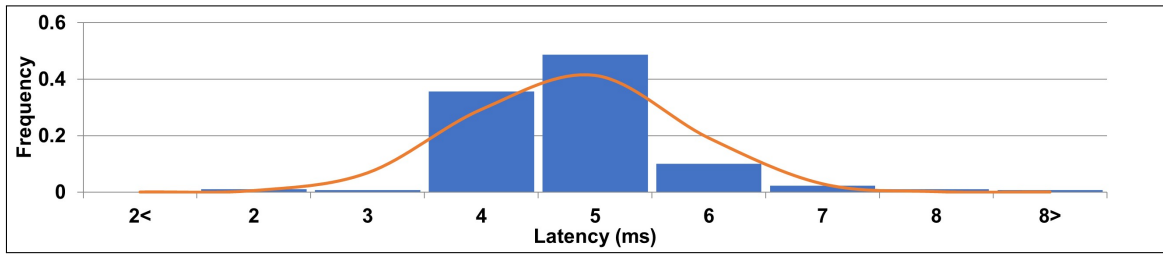


Figure 7.27. Histogram of $D_i^{s,B}$ values when N=1, for HTTP-based scenario

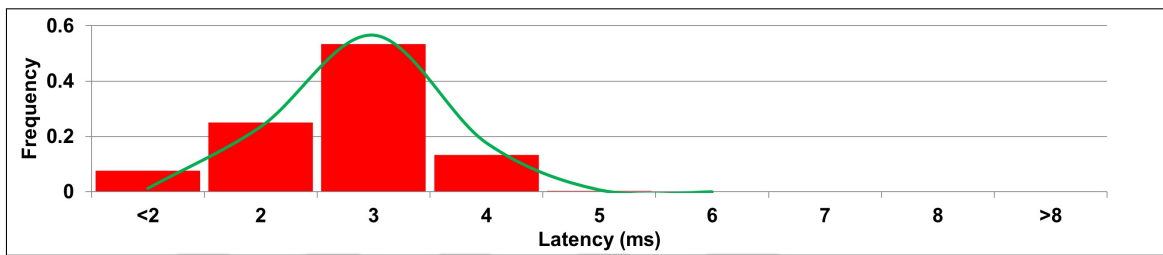


Figure 7.28. Histogram of $D_i^{s,B}$ values when N=1, for MQ-based scenario

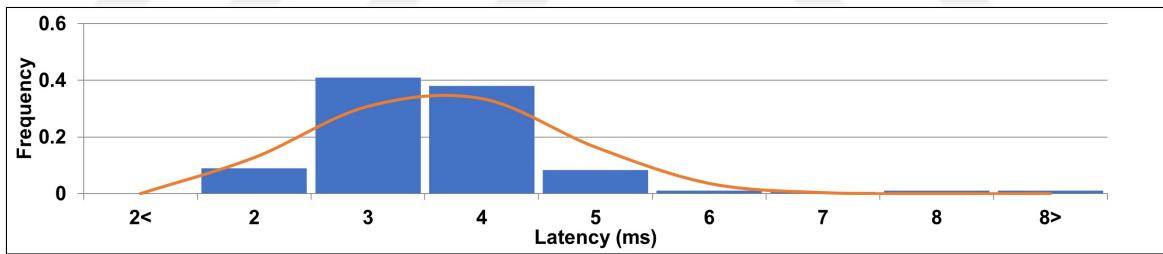


Figure 7.29. Histogram of $D_i^{s,B}$ values when N=4, for HTTP-based scenario

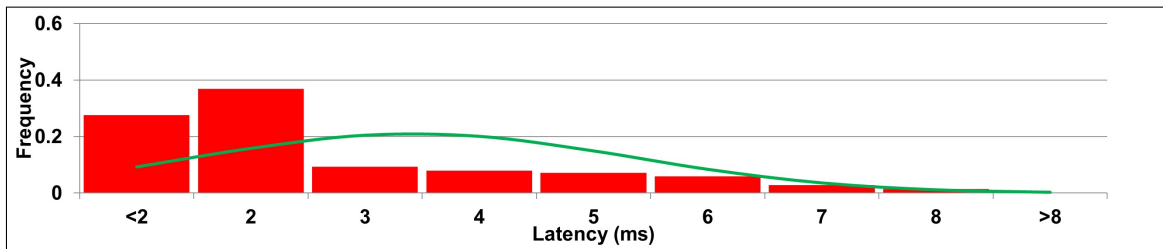


Figure 7.30. Histogram of $D_i^{s,B}$ values when N=4, for MQ-based scenario

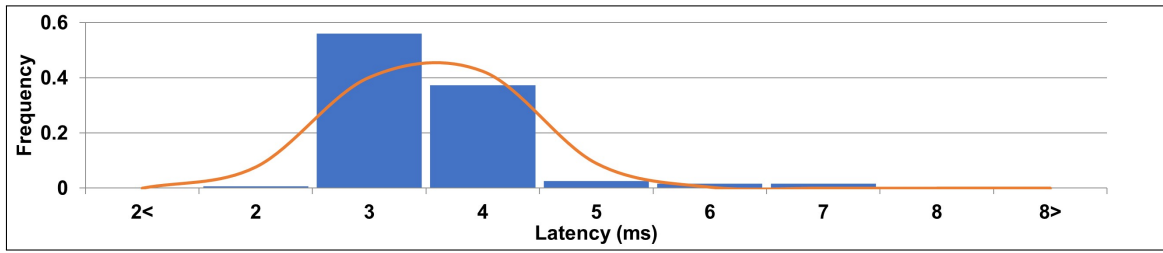


Figure 7.31. Histogram of $D_i^{s,B}$ values when N=16, for HTTP-based scenario

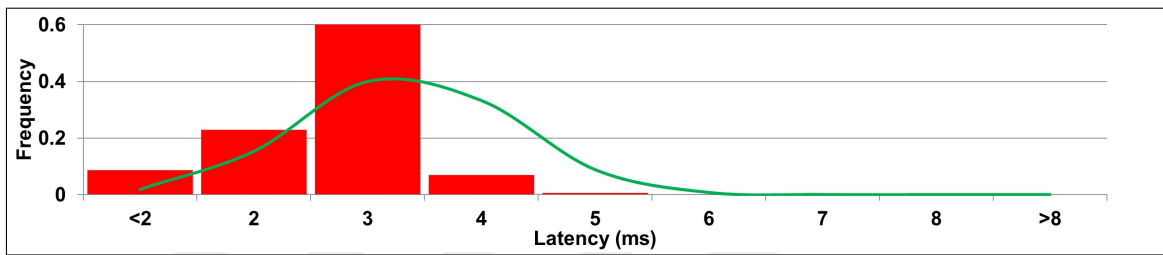


Figure 7.32. Histogram of $D_i^{s,B}$ values when N=16, for MQ-based scenario

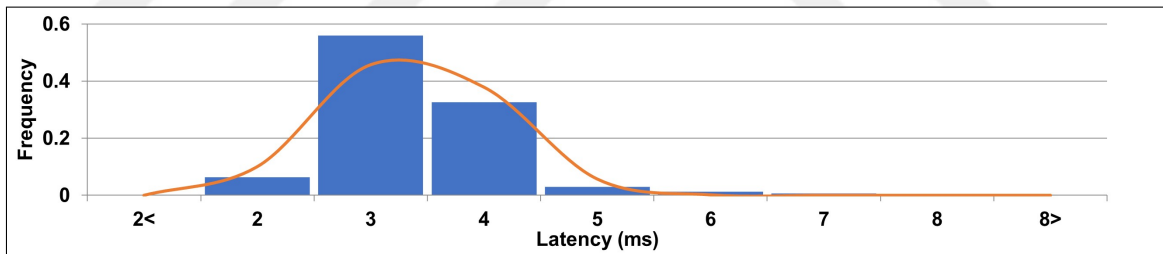


Figure 7.33. Histogram of $D_i^{s,B}$ values when N=64, for HTTP-based scenario

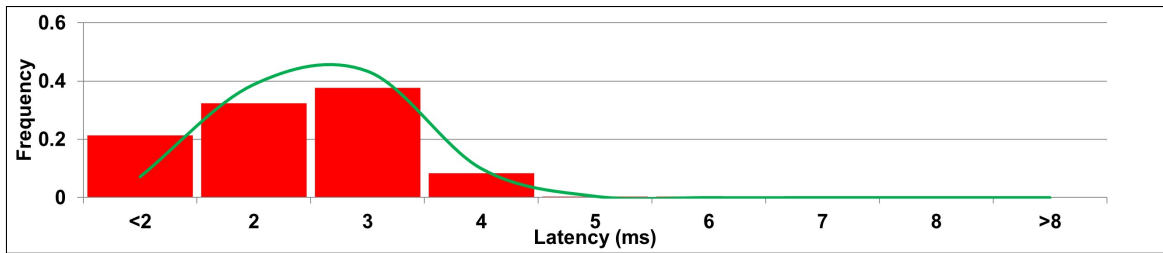


Figure 7.34. Histogram of $D_i^{s,B}$ values when N=64, for MQ-based scenario

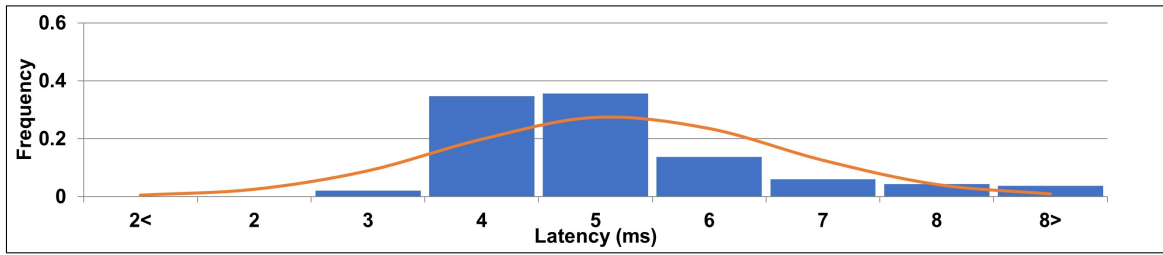


Figure 7.35. Histogram of $D_i^{s,B}$ values when $N=256$, for HTTP-based scenario

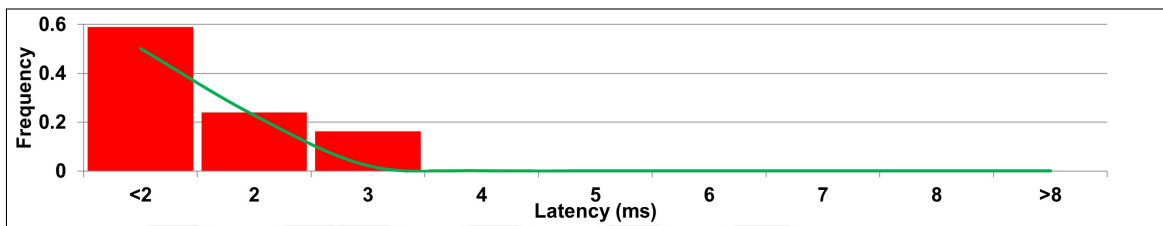


Figure 7.36. Histogram of $D_i^{s,B}$ values when $N=256$, for MQ-based scenario

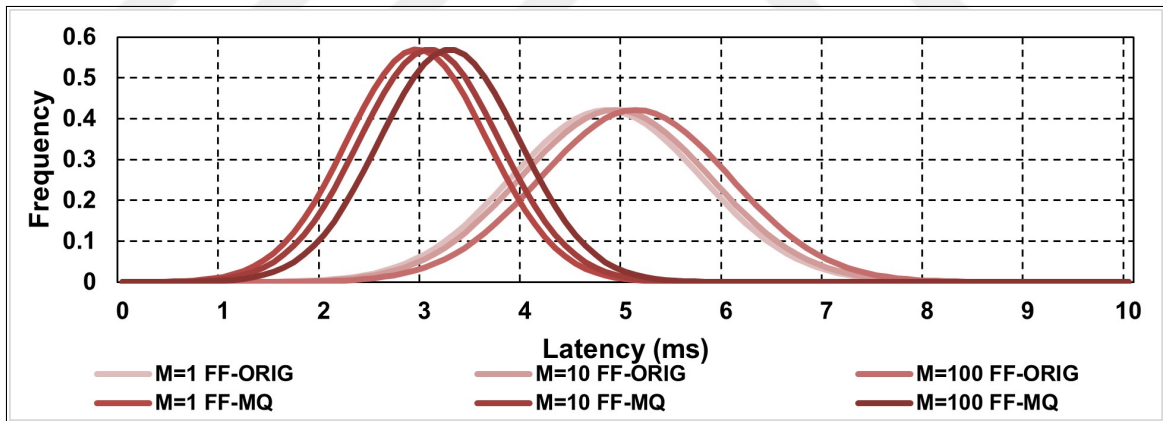
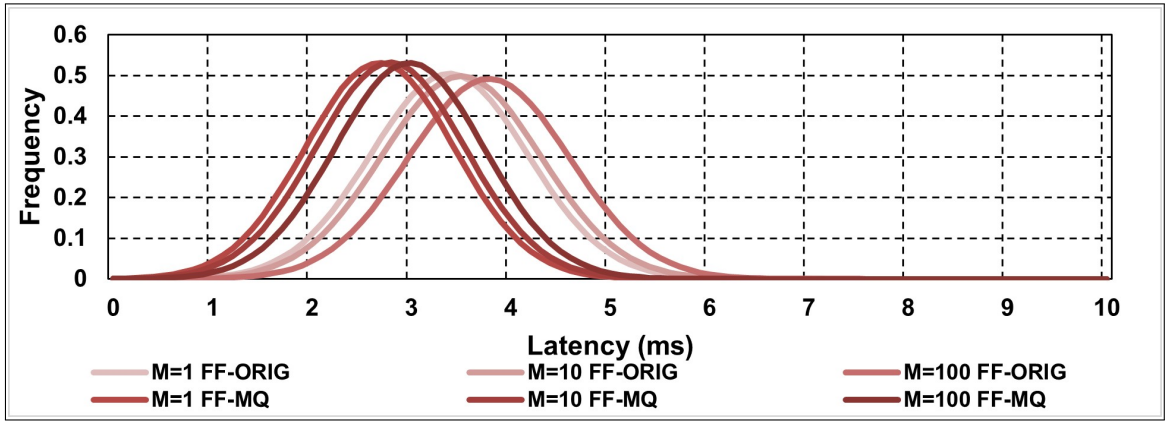
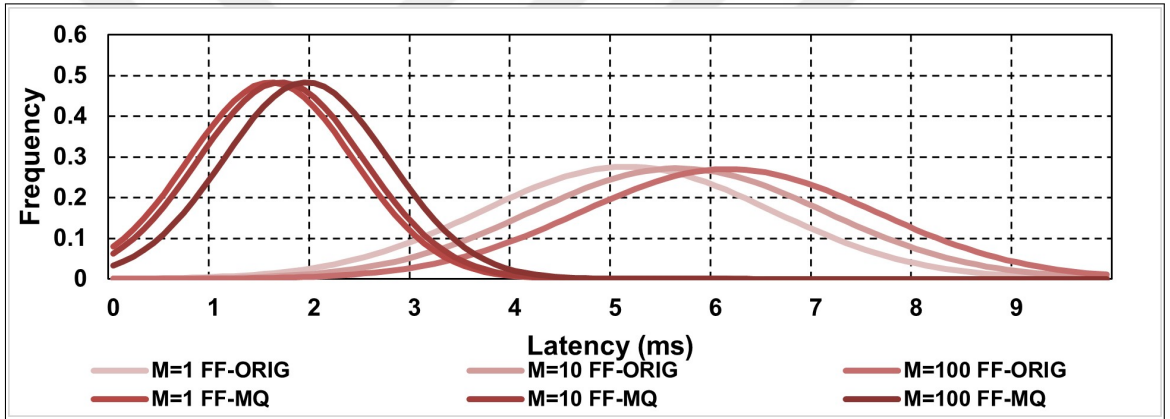


Figure 7.37. Normal distribution curve when $N=1$

Figure 7.38. Normal distribution curve when $N=16$ Figure 7.39. Normal distribution curve when $N=256$

After elaborating on the confidence intervals of the distribution of values of data transmission latency in inner fog, we analyze the variation of latency values when we scale N and M up. Figures 7.40, 7.41, 7.43 and 7.44 depict the change in $\overline{D_i^{s,B}}$, $\overline{D_i^{s,BQ}}$, $\overline{\Delta^{s,B}}$ and $\overline{\Delta^{s,BQ}}$ values. For the HTTP based scenario, when we increase N from 1 to 128, average end-to-end sensor data transmission latency in the inner-fog lowers gently from 5ms to 3ms and $\overline{\Delta^{s,B}}$ increases very slowly between 0.01ms to 0.7ms. Additionally, the effect of the alteration of M values is very small to the change in these metrics. As seen in Table 7.3, for the first scenario, utilization of computing resources does not raise as expected when we increase N from 128 to 256. In such a case, HTTP message scheduling does not work efficiently. This leads a dramatic increase in $\overline{\Delta^{s,B}}$, so in $\overline{D_i^{s,B}}$. However, we do not observe

any dramatic change in these values with MQ augmented scenario. Moreover, MQ method reduces the end-to-end sensor data transmission delays in inner fog with the degradation of nearly 1ms to 2ms compared to HTTP based scenario, and latency change becomes so slow even we increase N . Besides, as depicted in Figures 7.42,7.45, MQ scenario decreases the extra processing delays ($\Delta^{s,BQ}$ values) faced in the sensor data transmission in the inner fog. This analysis reveals that, as we argue in Section 5.3.2, MQ based scenario saves at least 1 RTT in data transmission in inner fog.

Table 7.3. Average CPU and Memory Utilization of HTTP method

N	Avg. CPU Utilization Percentage	Avg. Memory Utilization Percentage
16	3.4%	0.8%
32	6.3%	0.8%
64	15.7%	0.8%
128	32.3%	0.8%
256	48.1%	0.8%

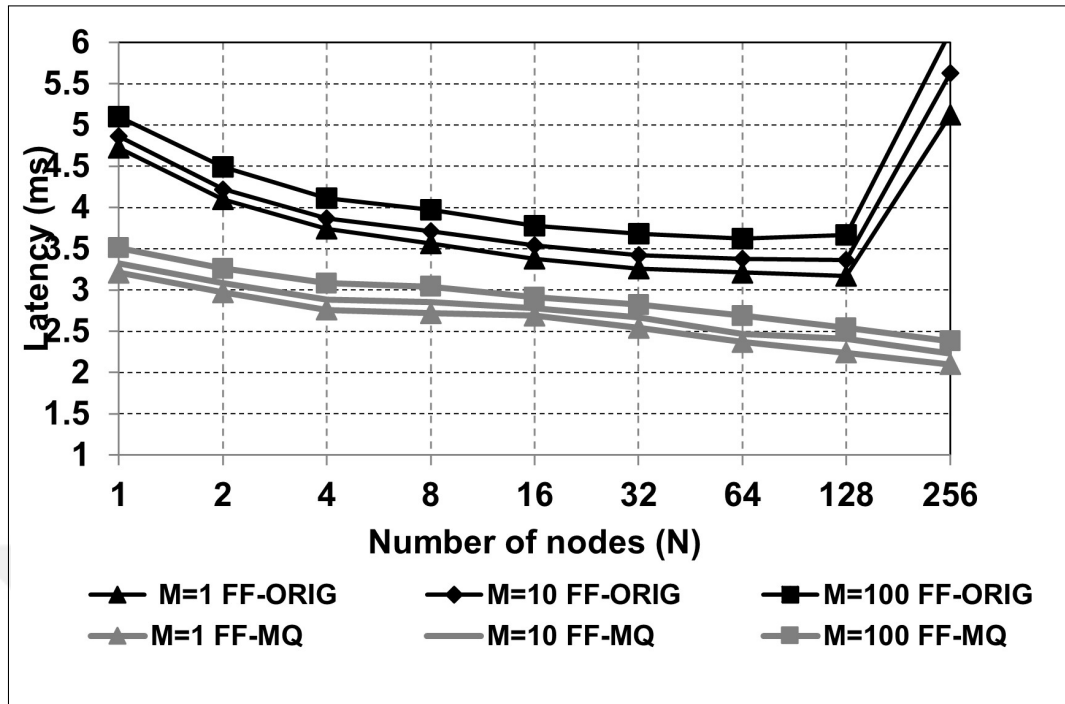


Figure 7.40. Change in average $D^{s,B}$ and $D^{s,M}$ values with increasing N and M

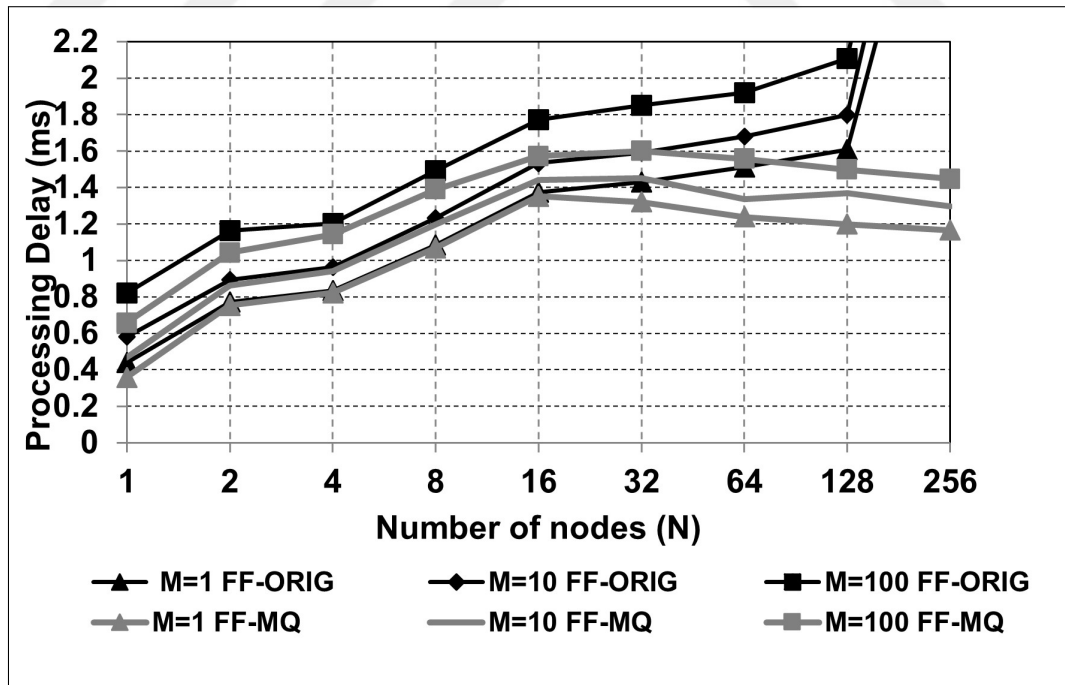


Figure 7.41. Change in average $\Delta^{s,B}$, $\Delta^{s,M}$ values with increasing N and M

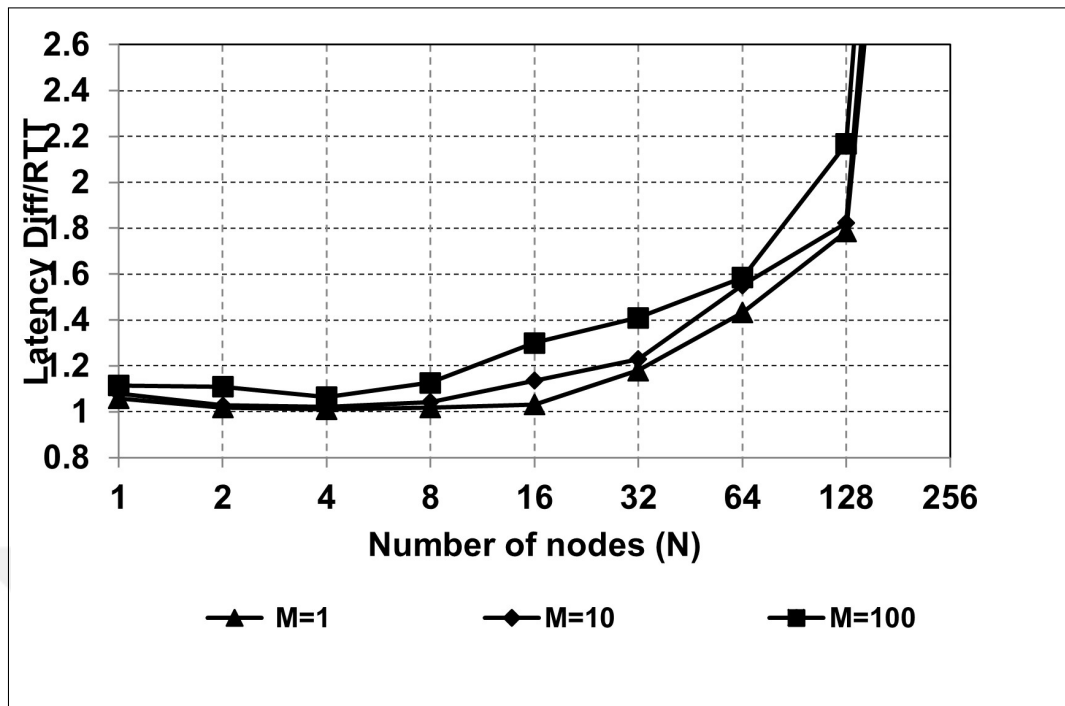


Figure 7.42. Latency Difference with increasing N

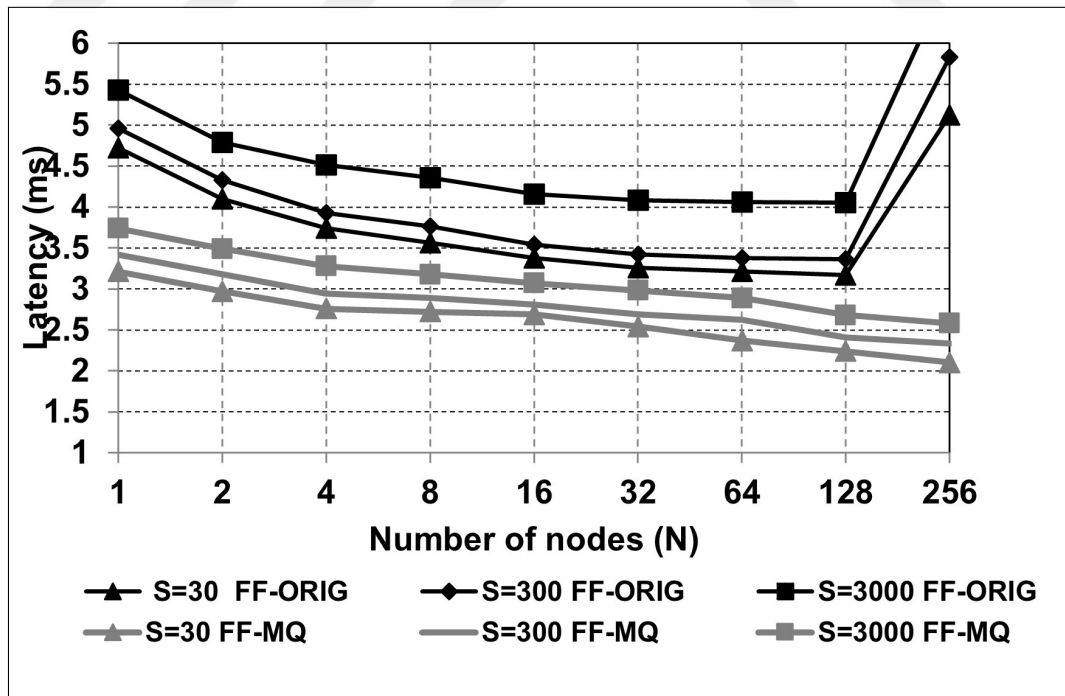


Figure 7.43. Change in average $D^{s,B}$ and $D^{s,M}$ values with increasing N and S

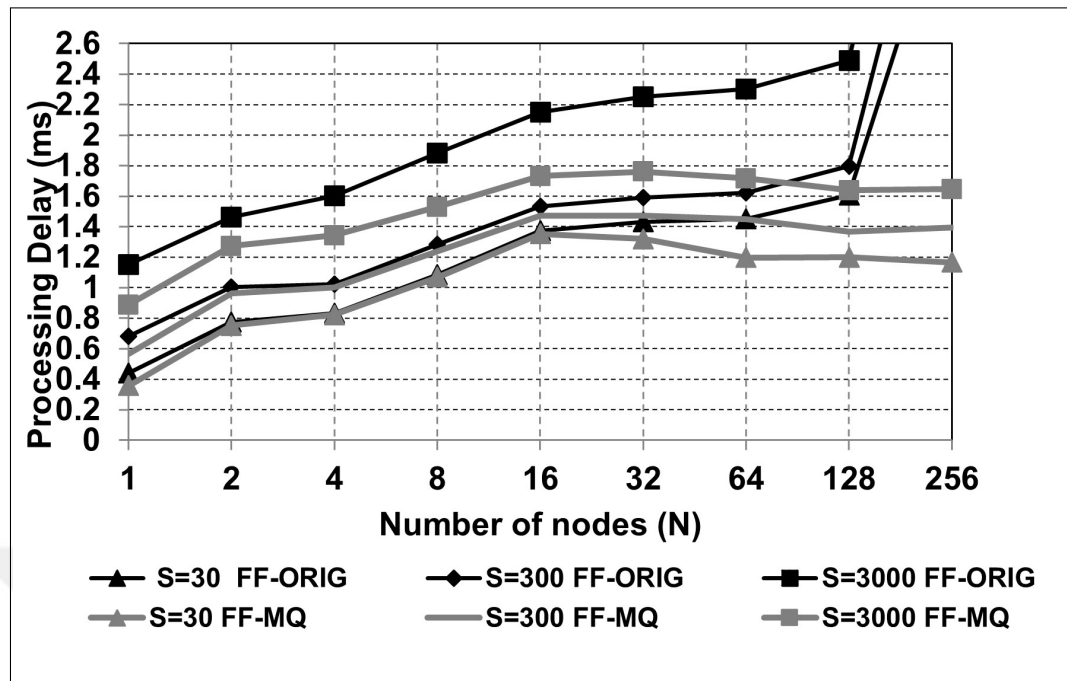


Figure 7.44. Change in average $\Delta^{s,B}$, $\Delta^{s,M}$ values with increasing N and S

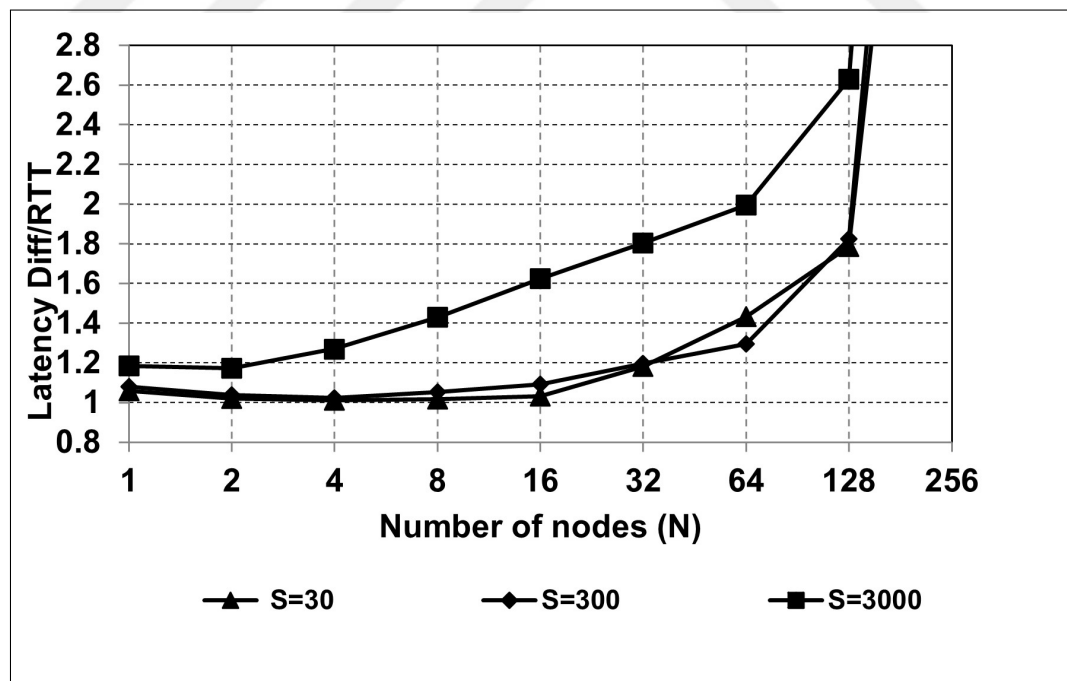


Figure 7.45. Latency Difference with increasing S

7.5. QUEUE MODEL TESTS

We carry out comprehensive simulation tests to observe the change in r_{wait} and waiting-time gap values of the decision packets that arrive at three different queuing systems, repeating the tests with different k values. We simulate simultaneous clients with varying numbers M using multi-threaded program, in which each thread issues a subscription rule, along with the CP value to the fog, to periodically monitor the physical environment with the sensors given in Table 7.1. This table also provides the parameters that were changed in the tests for evaluation of the proposed scheduling policy.

7.5.1. Service Time Analysis

We have computed the confidence intervals of the real service time values of the data packet forwarding in FC collected from the tests. For this, we have analyzed the occurrence frequencies of these values. As depicted in Figures 7.46-7.48, the distribution of service time values follow a normal distribution and, occurrence frequencies have a peak on the mean values. So these values are meaningful in the expression of the service rate and traffic rates in the queuing analysis. Besides, we observe the service rates remain constant even we change M in the experiments. Because the packet forwarding time depends on the packet size and it is independent of the amount of packet aggregation in the queue. Additionally, we calculate the traffic ratios using the mean of these constant service times and test parameters concerning M . Table 7.4 gives the experimental ρ values obtained and optimal queue size limits of corresponding M values in the tests.

Table 7.4. The traffic rates according to number of clients

M	ρ
50	0.0113
100	0.0267
200	0.0534
400	0.1068
800	0.2167

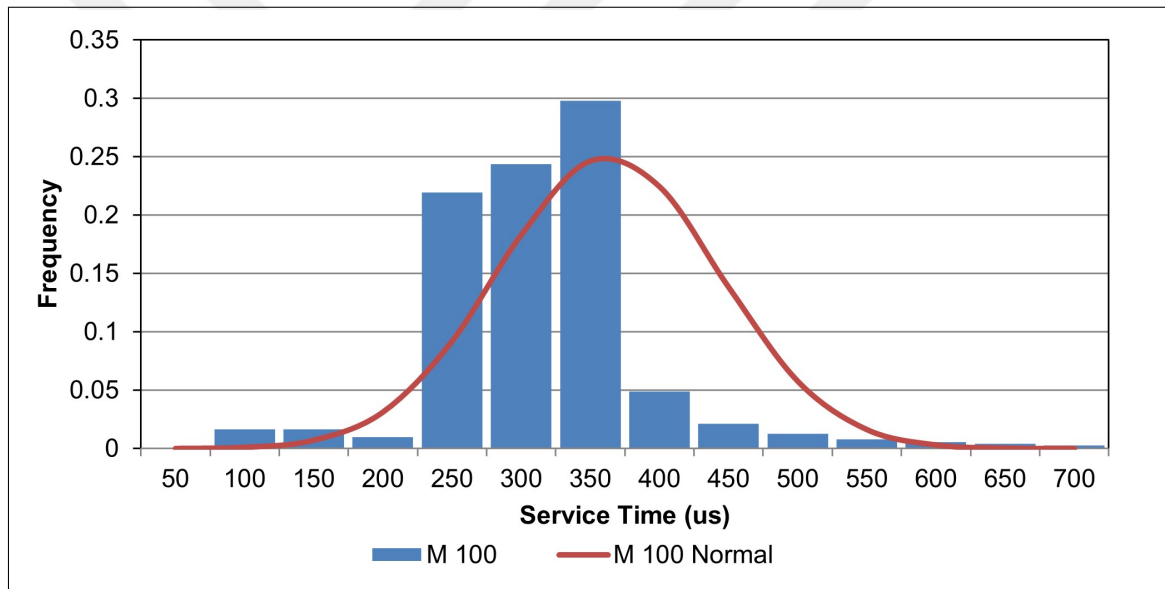


Figure 7.46. Service time distributions when M=100.

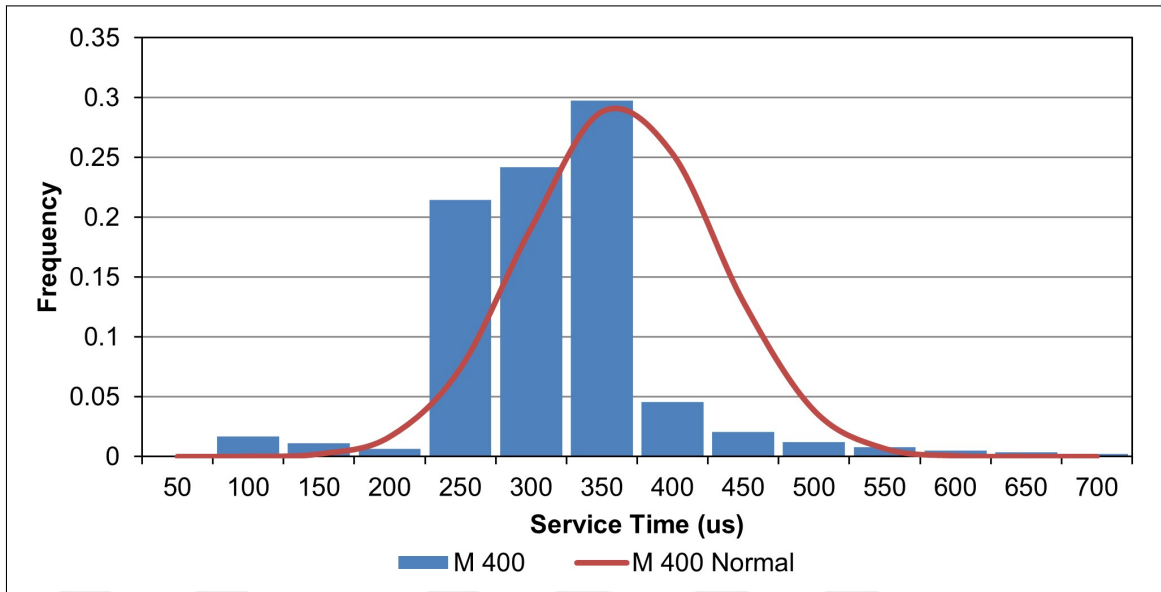


Figure 7.47. Service time distributions when M=400.

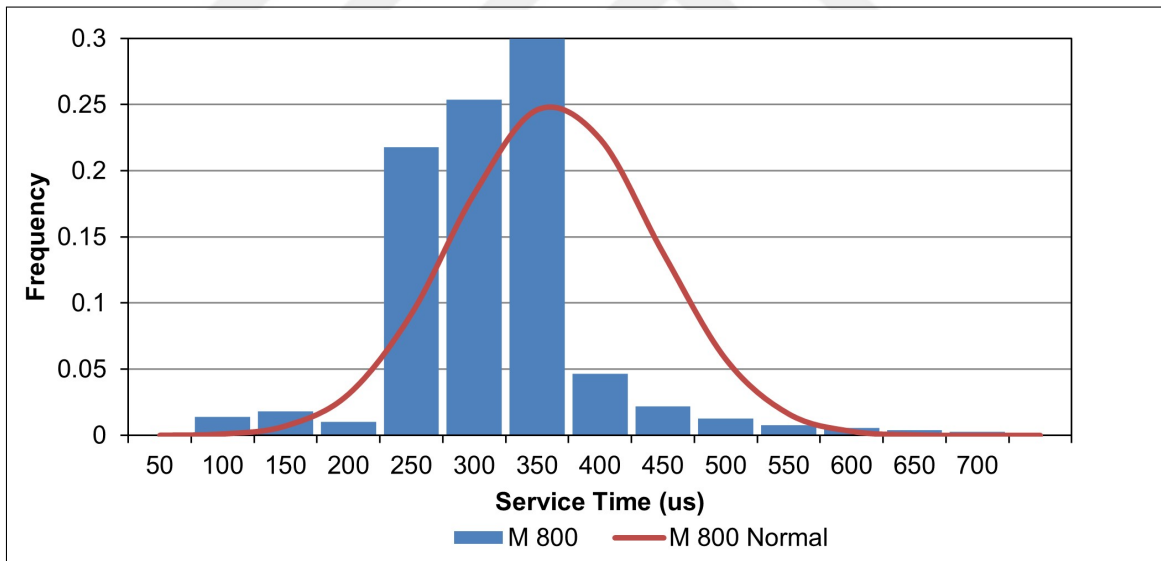


Figure 7.48. Service time distributions when M=800.

7.5.2. Performance of FIFO Queuing System

We have evaluated the performance of the system, firstly using the limited capacity FIFO queuing policy in FC. Figure 7.49 demonstrates the r_{wait_K} results for $k=2-5$. Besides,

Figures 7.50 and 7.51 depict the distribution of r_{wait_K} values with different k values when $M=800$. Experimental r_{wait_K} results follows theoretical values with small errors and, increase linearly, with increase of M from 50 to 800, and when $k=2-5$. Thus, r_{wait_K} values can be predictable with the theoretical queuing model given in Chapter 6.

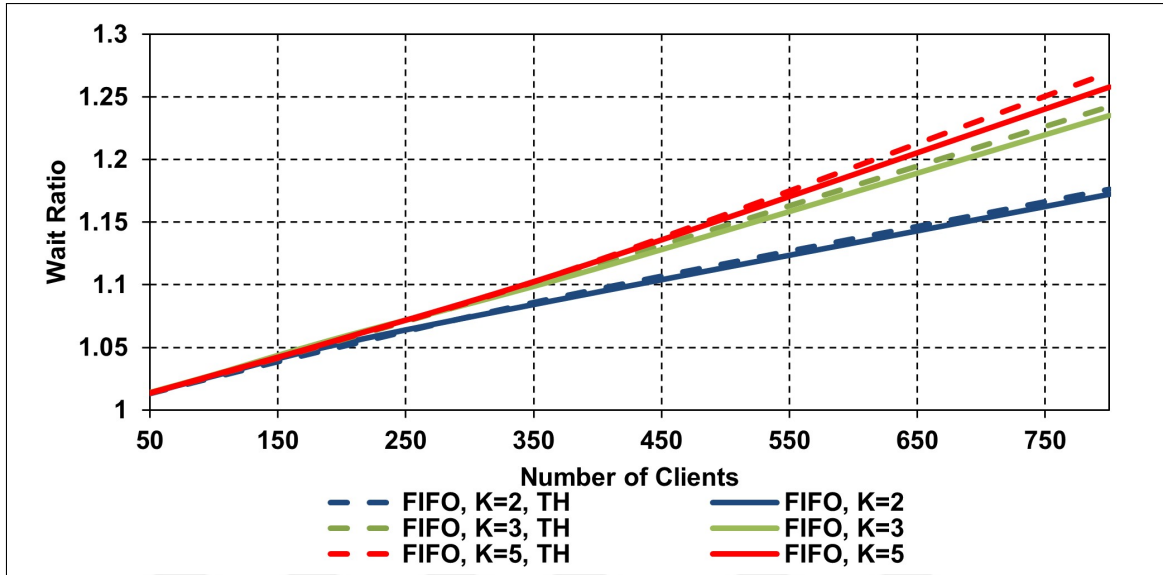


Figure 7.49. r_{wait} results when $k=2, 3$ and 5 , for FIFO

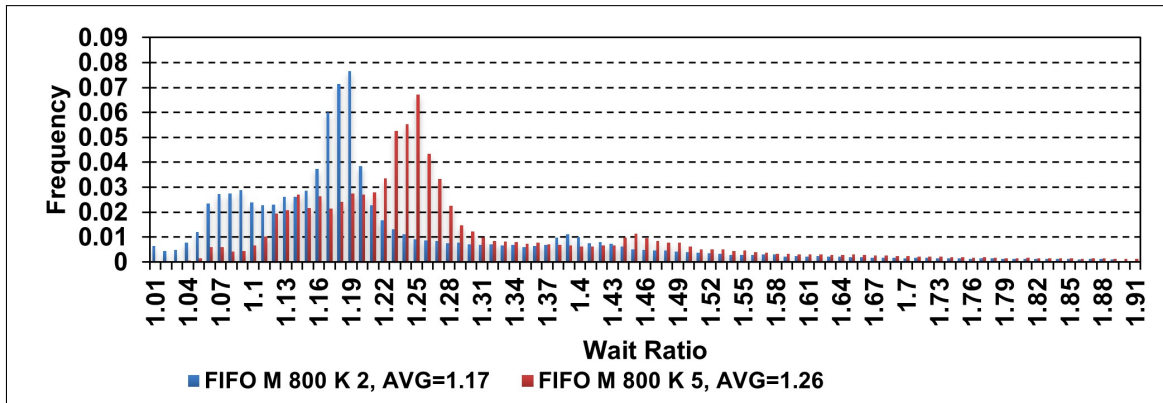


Figure 7.50. Distribution of r_{wait} when $M = 800$, $k=2$ and 5 , for FIFO

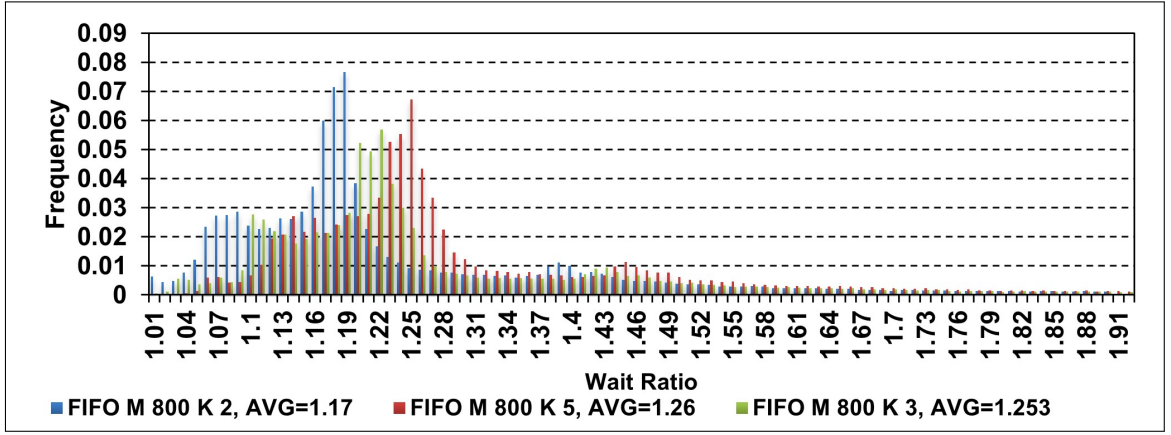


Figure 7.51. Distribution of r_{wait} when $M = 800$, $k=2, 3$ and 5 , for FIFO.

7.5.3. Performance of MPDQ Queuing System

We have changed the queuing policy of FC to limited capacity MPDQ, changed k starting from 2 to 5 and, got r_{wait} values for transmission of two packet classes. As seen in Figures 7.52 and 7.53, experimental r_{wait} values follow the values obtained from our theoretical analysis given in Chapter 6. Like in FIFO results, the effect of change in k value with the change of r_{wait} is more perceivable when M is 800. Packet scheduling with the MPDQ approach creates a gap in $r_{wait_{K,l}}$ values because of the prioritization of the packets. As shown in Figure 7.54, the wait time gap between increases with M , and it is more evident with a high M value. Thus, packets with low priority starve more as the increase of M with MPDQ policy.

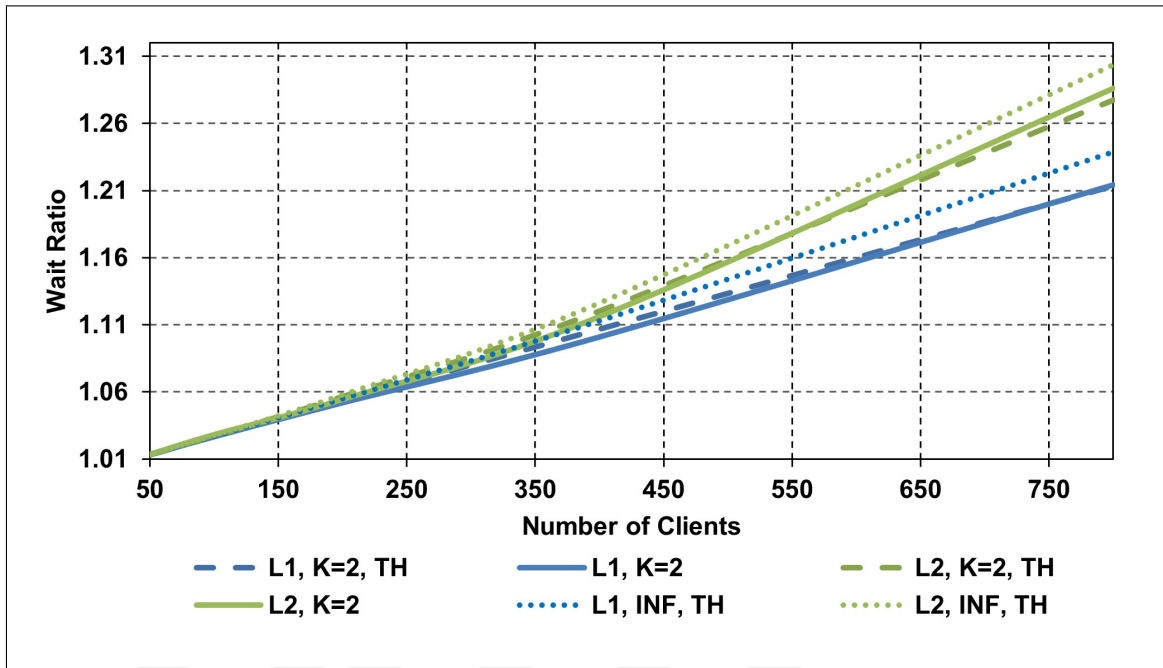


Figure 7.52. r_{wait} results when $k=2$ or INF, for MPDQ.

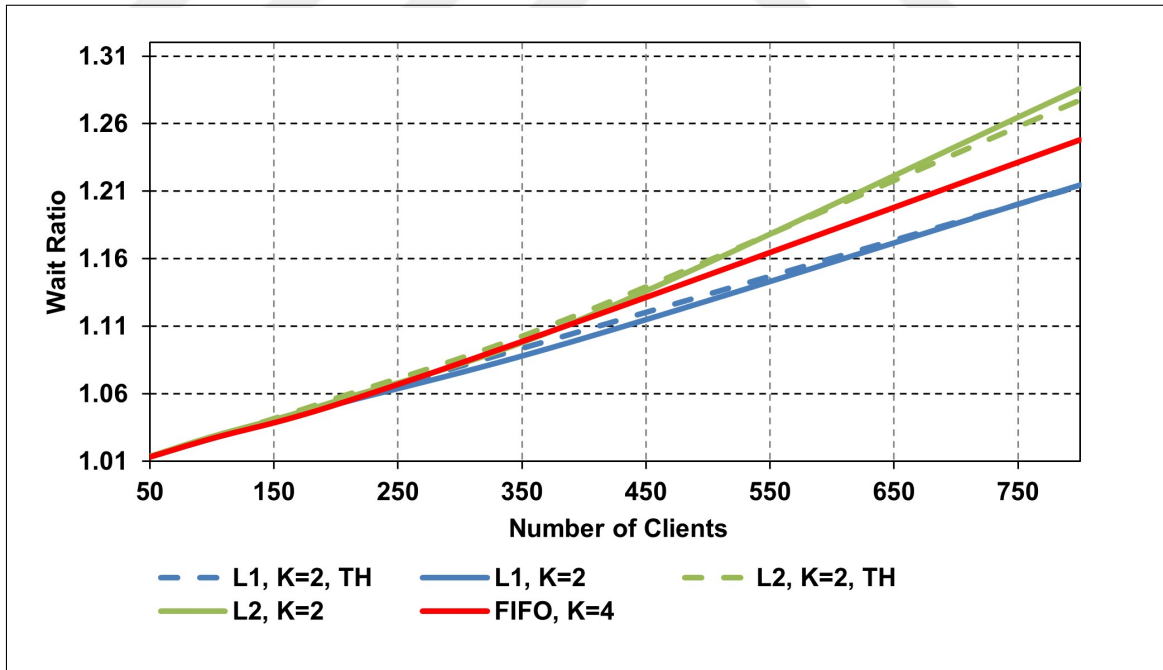


Figure 7.53. Comparison of r_{wait} of MPDQ with FIFO when $k=2$.

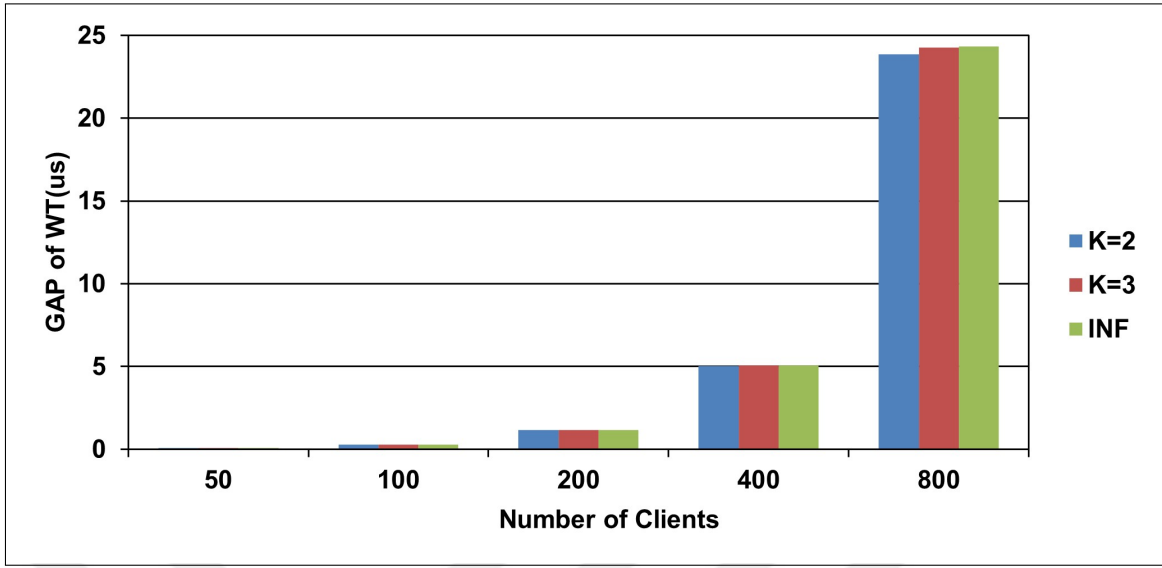


Figure 7.54. Gap of wait times for MPDQ

7.5.4. Performance of CEP Approach

In the same experimental setup, we have devised tests using the proposed CEP based FIFO approach with changing k and M values. In these tests, we observe the variation in r_{wait_K} values relative to those obtained with MPDQ policy. As presented in Figure 7.55 and 7.57, for a finite size queue having $k=2$ and for an infinite queue, our proposed CEP based FIFO packet scheduling approach decreases $r_{wait_{K,2}}$ values to ones which approximates theoretical FIFO results, while $r_{wait_{K,1}}$ values approximate to those obtained with MPDQ policy. Thus, our approach reduces the wait time gaps between wait time values of the high and low priority packets, as illustrated in Figures 7.56 and 7.58. As shown in Figure 7.58, any increase in k shortens this gap value. By this analysis, we can conclude that our proposed approach reduces the wait time gaps between wait time values of the high and low priority packets and prevents potential starvation.

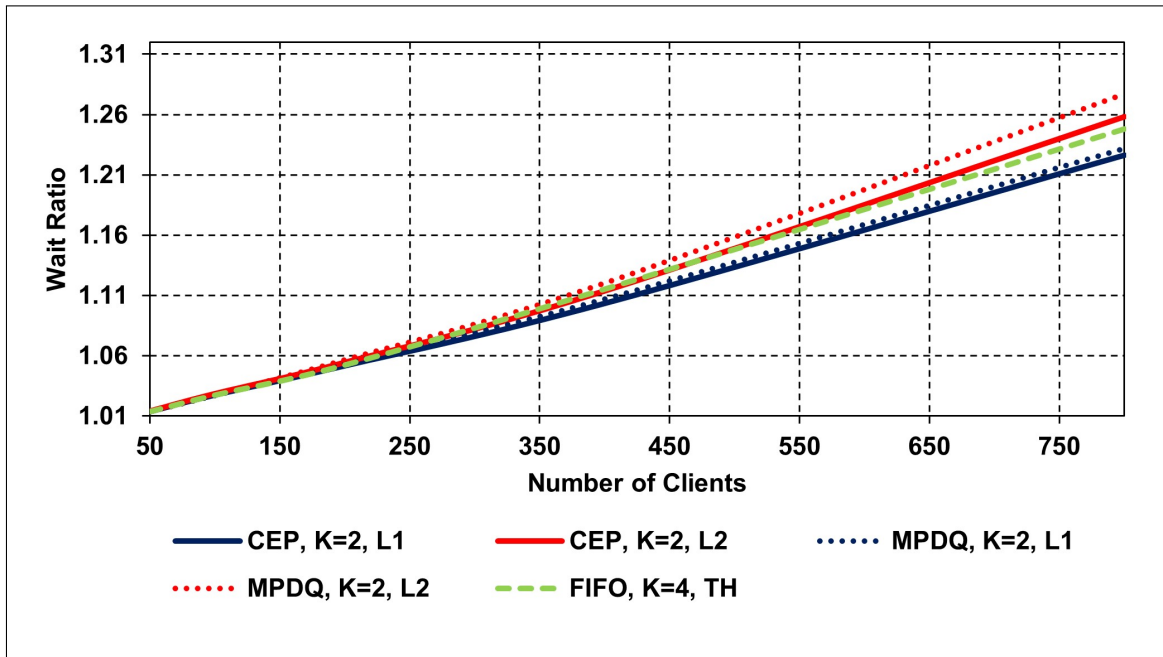


Figure 7.55. Comparison of r_{wait} of MPDQ with CEP when $k=2$.

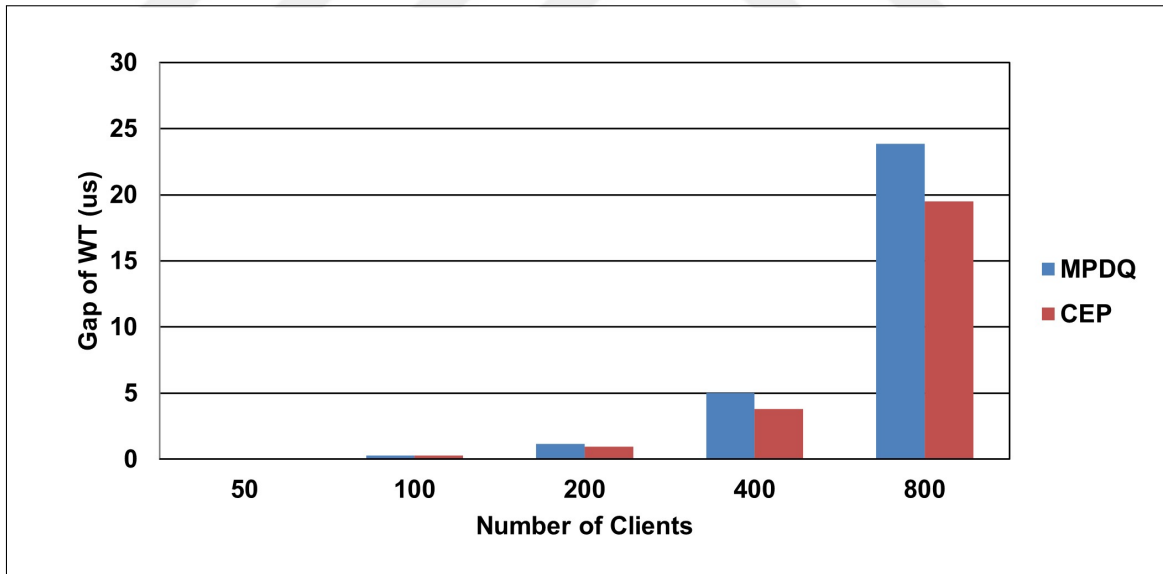


Figure 7.56. Gap of waiting times for different priority based schedulers when $k=2$.

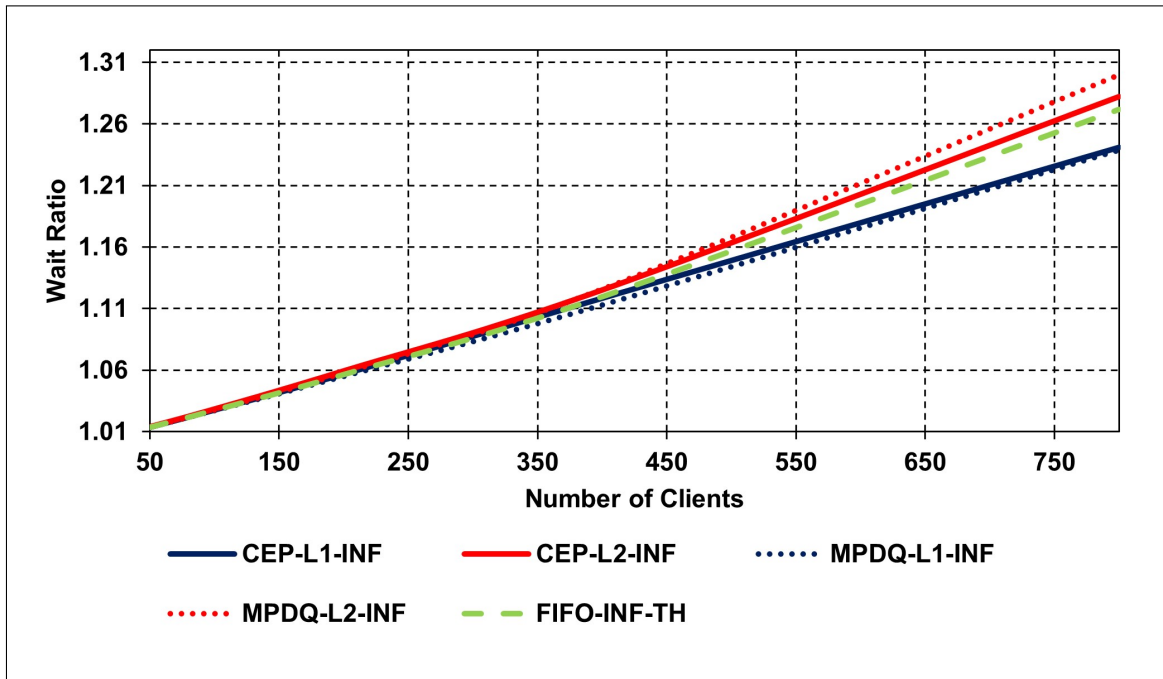


Figure 7.57. Comparison of r_{wait} of MPDQ with CEP when $k=INF$.

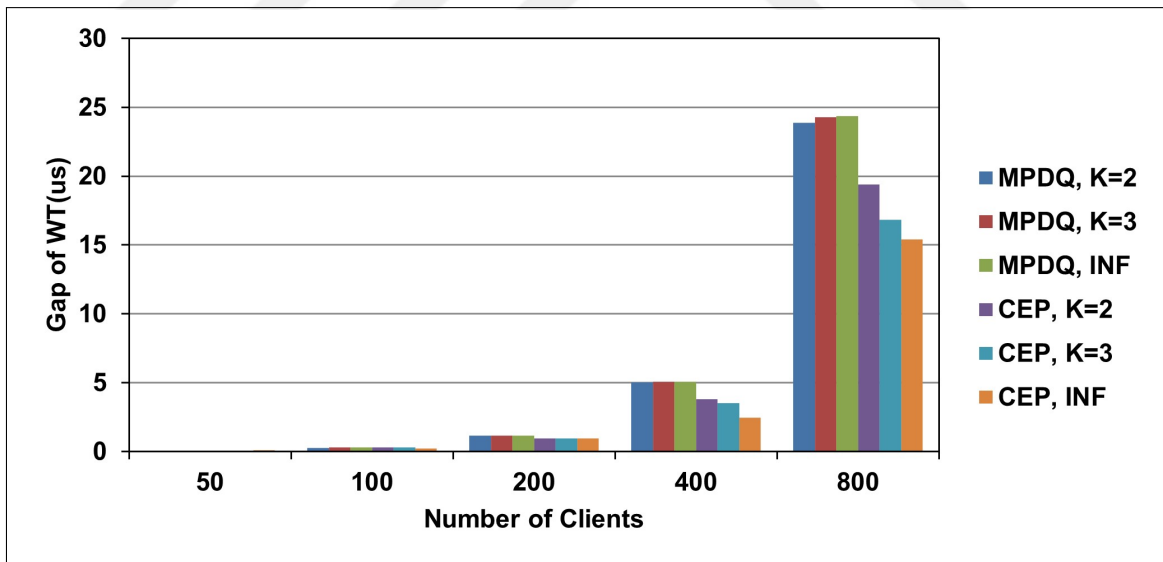


Figure 7.58. Gap of waiting times for different priority based schedulers.

8. CONCLUSIONS

In this thesis, we propose a fog computing model and a scalable fog architecture for IoT services and applications design. The proposed model solves several critical problems related to the FoT data traffic management domain and has several contributions. Firstly, for reducing wait time gaps observed in the multi-level priority queue design, we examine the starvation problem of the multi-priority IoT data traffic in the fog design and we present a scheduling policy which uses a Batch-Window CEP approach over FIFO scheduling. For the development of the model, we analyze the arrival, service, waiting process of finite size non-preemptive multi-level queues and formulate the parameters related to multi-priority discipline queues and our scheduling policy. We devise tests to compare our scheduling policy with non-preemptive MPDQ scheduling in terms of arrival, service, and waiting time. The results reveal that the proposed CEP based FIFO scheduling policy reduces average wait time gaps observed in MPDQ policy. We have shown that the empirical results obtained from the test-bed verify the theoretical model in the given setup in which we scale the number of client up to 800. Additionally, in the given setup, we have shown that the scheduling model reduces wait time gaps up to $15\mu s$. Secondly, for reducing latency faced in reliable data transfer in inner fog, we apply a message queue based connection-oriented data traffic flow mechanism. We compare this mechanism with a conventional non-persistent HTTP based data transfer method. We analyze the internals of the data flow of these mechanisms and timing constraints of the data transmission in the fog. With a latency analysis, we elaborate on the transmission and processing delays faced when we apply each scenario in a horizontally scalable fog system. This analysis reveals that the given MQ mechanism reduces the round trip delays encountered in the data messaging between fog components. For the proof of the concept, we implement each of the data flow scenarios. We get comprehensive tests and investigate end-to-end delays faced both in sensor data transmission and decision data transmission when we increase the sensor device and client number in the system. With these tests, we show that MQ based data flow scenario results in a shorter latency, and we can save approximately 1 RTT for data transmission latency even we increase the sensor number to 256. Thirdly, we examine the sensor data freshness and IoT freshness time concept. We compare several sensor data freshness time calculation

methods for the maximal data freshness rate in a real testbed. We devise experiments in a real test bed and show that both MAC protocol of the sensor network and the calculation method have significant effect on the achievable freshness rates. In addition, we show that the stateful methods used for the freshness time calculation give better freshness rates than those observed with stateless methods.

In this thesis, as the proof of concept for the proposed scheduling policy, we devised comprehensive simulation tests to observe the given policy with a 2-class IoT data traffic, and we apply the proposed data traffic scheduling mechanism with a two-level queue. For future work, it is possible to adapt the proposed scheduling model to the application cases which generate heterogeneous IoT data traffic and have different latency-based QoS requirements, since, the given scheme can model the arrival-service process for the forwarding of an N-class multi-priority IoT data traffic. One of the example use-cases which adapt the given scheduling model can be such an IoT application that generates the traffic of packets having both delay-sensitive video frames and periodically generated time-critical emergency detection results. The service types of the data traffic generated from such an application can be classified into several data traffic classes. The proposed model could be able to model the arrival-service process of this application with a queueing scheme having several waiting-line levels.

For the development of the data traffic latency model in the fog computing architecture, we focus on an analysis of the latency faced in inner fog when we use the given methods in a scalable fog architecture, and we consider a periodical monitoring IoT application case, which generates smaller overhead over data transmission latency in performing IoT services operations. Besides, we use the default MQ parameters defined by Kafka to realize the second data flow scenario, for comparison of the baseline MQ method with the conventionally used data flow orchestration method in fog designs. For a further study, the given scheme can be adapted for measuring the effect of MQ parameters such as queue size, queueing policy, and service priority types on the delay performance.

REFERENCES

1. Atlam H, Walters R, Wills G. Fog Computing and the Internet of Things: A Review. *Big Data and Cognitive Computing*. 2018;2:10–18.
2. Yang Y, Luo X, Chu X, Zhou M. *Fog-enabled intelligent IoT systems*. New York: Springer-Verlag Press; 2019.
3. Metzger F, Hobfeld T, Bauer A, Kounev S, Heegaard P. Modeling of Aggregated IoT Traffic and Its Application to an IoT Cloud. *Proceedings of the IEEE*. 2019;107:679–694.
4. Djama A, Djamaa B, Senouci MR. Information-Centric Networking solutions for the Internet of Things: A systematic mapping review. *Computer Communications*. 2020;159:37–59.
5. Kassab W, Darabkh K. AZ survey of Internet of Things: Architectures, protocols, applications, recent advances, future directions and recommendations. *Journal of Network and Computer Applications*. 2020;103:1–49.
6. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*. 2015;17:2347–2376.
7. DeMatos E, Tiburski RT, Moratelli CR, Filhoa SJF, Amaral LA, Ramachandran G, Krishnamachari B, Hesse F. Context information sharing for the Internet of Things: A survey. *Computer Networks*. 2020;166:106988:1–106988:19.
8. Asghari P, Rahmani A, Javadi H. Internet of Things applications: A systematic review. *Computer Networks*. 2019;148:241–261.
9. Bello O, Zeadally S. Toward efficient smartification of the Internet of Things (IoT)

- services. *Future Generation Computer Systems*. 2019;92:663–673.
10. Atzori L, Iera A, Morabito G. The Internet of Things: A survey. *Computer Networks*. 2010;54:2787–2805.
 11. Serdaroğlu KC, Baydere S. WiSEGATE: Wireless Sensor Network Gateway framework for internet of things. *Wireless Networks*. 2016;22:1475–1493.
 12. Moura J, Hutchison D. Fog computing systems: State of the art, research issues and future trends, with a focus on resilience. *Journal of Network and Computer Applications*. 2020;69:102784:1–102784:20.
 13. Singh M, Baranwal G. Quality of Service (QoS) in Internet of Things. In: *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages, IoT-SIU*. IEEE; 2018. p. 1–6.
 14. Hashem IAT, Yaqoob I, Anuar NB, Mokhtar S, Gani A, Ullah KS. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*. 2015;47:98–115.
 15. Gaillard G, Barthel D, Theoleyrey F, Valois F. Service Level Agreements for Wireless Sensor Networks: a WSN Operator's Point of View. In: *2014 IEEE International Conference on Network Operations and Management Symposium (NOMS)*. IEEE; 2014. p. 1–6.
 16. Puliafito C, Mingozzi E, Lonngo F, Puliafito A, Rana O. Fog Computing for the Internet of Things: A Survey. *ACM Transactions on Internet Technology*. 2019;19:18:1 – 18:36.
 17. Yousefpour A, Ishigaki G, Gour R, Jue J. On Reducing IoT Service Delay via Fog Offloading. *IEEE Internet of Things Journal*. 2018;5:998–1010.
 18. Vasconcelos DR, Andrade RMC, Severino V, De Souza JN. Cloud, Fog, or Mist in

- IoT? That is the question. *ACM Transactions on Internet Technology*. 2019;19:1–35.
19. C Prazeres, M Serrano. SOFT-IoT: Self-Organizing FOG of Things. *In: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*; 2016. p. 803–808.
 20. Caiza G, Saeteros M, Onate W, Garcia MV. Fog computing at industrial level, architecture, latency, energy, and security: A review. *Heliyon*. 2020;6:e03706:1–e03706:7.
 21. Samie F, Bauer L, Henkel J. From Cloud Down To Things: An overview of Machine Learning in Internet Of Things. *IEEE Internet of Things Journal*. 2019;4921–4934.
 22. Aazam M, Huh E. Fog Computing and Smart Gateway Based Communication for Cloud of Things. *In: 2014 International Conference on Future Internet of Things and Cloud*; 2014. p. 464–470.
 23. Markus A, Kertesz A. A survey and taxonomy of simulation environments modelling fog computing. *Simulation Modeling Practice and Theory*. 2019;101:102042:1 – 102042:18.
 24. Al-Rawahi AS, Lee K, Lotfi A. A Multiobjective QoS Model for Trading Cloud of Things Resources. *IEEE Internet of Things Journal*. 2019;9447 – 9463.
 25. Tahaei H, Afifi F, Asemi A, Zaki F, Anuar A. The rise of traffic classification in IoT networks: A survey. *Journal of Network and Computer Applications*. 2020;154:1 –20.
 26. Skarlat O, Schulte S, Borkowski M, Leitner P. Resource provisioning for IoT services in the fog. *In: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications: SOCA2016*. IEEE; 2016. p. 32–39.
 27. Bellavista P, Berrocal J, Corradi A, Das SK, Foschini L, Zanni A. A survey on fog computing for the Internet of Things. *Pervasive and Mobile Computing*. 2019;52:71–

99.

28. Cheng B, Solmaz G, Cirillo F, Kovacs E, Terasawa K, Kitazawa A. FogFlow: Easy Programming of IoT Services Over Cloud and Edges for Smart Cities. *IEEE Internet of Things Journal*. 2018;5:696–707.
29. Singh SP, Nayyar A, Kumar R, Sharma A. Fog computing: from architecture to edge computing and big data processing. *Journal of Super Computing*. 2019:2070 – 2105.
30. Mahmud R, Ramamohanarao K, Buyya R. Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions. *ACM Computing Surveys*. 2020;1:88:1 – 88:36.
31. Maiti P, Shukla J, Sahoo B, Turuk AK. QoS-aware fog nodes placement. In: *4th IEEE International Conference on Recent Advances in Information Technology: RAIT 2018*. IEEE; 2018. p. 1–6.
32. Buyya R, Dastjerdi VA. *Internet of Things: Principles and Paradigms*. New York: Elsevier; 2016.
33. Habibi P, Farhodi M, Kazemian S, Khorsandi, S, Leon-Garcia A. Fog Computing: A Comprehensive Architectural Survey. *IEEE Access*. 2020;8:102042:1 – 102042:18.
34. Nashaat H, Ahmed E, Rizk R. IoT Application Placement Algorithm Based on Multi-Dimensional QoE Prioritization Model in Fog Computing Environment. *IEEE Access*. 2020;8:111253–111264.
35. Gezer V, Wagner A. Real-time edge framework (RTEF) : task scheduling and Realization. *Journal of Intelligent Manufacturing*. 2021.
36. Hong C, Varghese B. Resource management in fog/edge computing: A survey. *ACM Computing Surveys*. 2019;52:97:1–97:37.
37. Dizdarevic J, Carpio F, Jukan A. A survey of communication protocols for internet of

- things and related challenges of fog and cloud computing integration. *ACM Computing Surveys*. 2019;51:116:1 – 116:29.
38. ElKafhali S, Salah K,. Efficient and dynamic scaling of fog nodes for IoT devices. *Journal of Supercomputing*. 2017;73:5261–5284.
 39. Ghobaei-Arani M, Souri A, Rahmanian A. Resource Management Approaches in Fog Computing: a Comprehensive Review. *Journal of Grid Computing*. 2020:1–48.
 40. Olaniyan R, Maheswaran M. Multipoint Synchronization for Fog-Controlled Internet of Things. *IEEE Internet of Things Journal*. 2019:9656 – 9667.
 41. Rahimi M, Songhorabadi M, Kashani MH. Fog-based smart homes: A systematic review. *Journal of Network and Computer Applications*. 2020;153:102531:1–102531:20.
 42. Haouari F, Faraj R, Alja'Am JM. Fog Computing Potentials, Applications, and Challenges. In: *2018 International Conference on Computer and Applications, ICCA 2018*; 2018. p. 399–406.
 43. Nezami Z, Zamanifar K, Djemame K, Pournaras E. Decentralized Edge-to-Cloud Load-balancing:Service Placement for the Internet of Things. *IEEE Access*. 2020;9:1–19.
 44. Alli AA, Alam MM. The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications. *Internet of Things*. 2020;9:100177:1–100177:20.
 45. Yi S, Yi C, Yi Q. A survey of Fog Computing. In: *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. ACM; 2015. p. 37–42.
 46. Rathod D, Chowdhary PG. Scalability of M/M/c Queue based Cloud-Fog Distributed Internet of Things Middleware of Things, middleware, queueing network, and

- scalability etc. *International Journal of Advanced Networking and Applications*. 2019;11:4162–4170.
47. Bouzeghoub M. A Framework for Analysis of Data Freshness. In: *2004 International Workshop on Information Quality in Information Systems*; 2004. p. 59–67.
 48. Hiller FS, Liberman GJ. *Introduction to Operations Research*. New York: Mc Graw Hill; 2000.
 49. Peer B, Rajbhoj R, Chathanur N. Complex Event Processing: Unburdening Big Data Complexities. *Infosys Lab Briefings*. 2013:53–95.
 50. Serdaroğlu KC, Kemal, Baydere S. An Efficient Multi-Priority Data Packet Traffic Scheduling Approach for Fog of Things. *IEEE Internet of Things Journal*. 2021 05:1–1.
 51. Serdaroğlu KC, Kadioglu T, Baydere S. Publish-Subscribe Based Monitoring Model For Wireless Sensor Networks. *New Advances In Internet of Things*. 2018:45–58.
 52. Serdaroğlu K, Baydere S. On the Data Freshness for IoT Traffic Modelling in Real-Time Emergency Observation Systems. In: *Proceedings of Fruct'23*. IEEE; 2018. p. 1–6.
 53. DeDonno M, Tange K, Dragoni N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. *IEEE Access*. 2019;7:150936 – 150948.
 54. Karkouch A, Mousannif H, Moatassime H, Noel T. Data quality in internet of things: A state-of-art survey. *Journal of Network and Computer Applications*. 2016;73:57–81.
 55. Weiner M, Jorgovanovic M, Sahai A, Nikolie B. Design of a low-latency, high-reliability wireless communication system for control applications. In: *IEEE Int. Conf. Commun. (ICC), ICC 2017*. IEEE; 2017. p. 3829 – 3835.
 56. Ray PP. A survey on Internet of Things Architectures. *Journal of King Saud*

University. 2016;23:1–6.

57. Wang YH, Cao K, Zhang XM. Complex event processing over distributed probabilistic event streams. *Computers and Mathematics with Applications*. 2013;66:1808 – 1821.
58. Elhadi S, Marzak A, Sael N, Merzouk S. Comparative Study of IoT Protocols. *In: Smart Application and Data Analysis for Smart Cities (SADASC'18)*. SSRN; 2018. p. 1–5.
59. Zhang Y, Sheng V. Fog-enabled Event Processing Based on IoT Resource Models. *IEEE Transactions on Knowledge and Data Engineering*. 2019;31:1707 – 1721.
60. Giatrakos N, Alevizos E, Artikis A, Deligiannakis A, Garofalakis M. Complex event recognition in the Big Data era: a survey. *The VLDB Journal*. 2020:313 – 352.
61. Stipkovic S, Bruns R, Dunkel J. Pervasive Computing by Mobile Complex Event Processing. *In: e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*; 2013. p. 318–323.
62. Etzion O, Liu L, Oszu MT. *Encyclopedia of Database Systems*. New York: Springer US; 2018.
63. Akila V, Govindasamy V, Sandosh S. Complex Event Processing over Uncertain Events: Techniques, Challenges, and Future Directions. *In: IEEE International Conference on Computation of Power, Energy Information and Communication (ICCPEIC'16)*. IEEE; 2016. p. 204 – 221.
64. Bormann C, Castellani AP, Shelby Z. CoAP: An application protocol for Billions of Tiny Internet Nodes. *IEEE Internet Computing*. 2012;16:62–67.
65. Yassein MB, Shatnawi MQ, Al-zoubi D. Application layer protocols for the Internet of Things: A survey. *In: Proceedings of Engineering & MIS (ICEMIS), International Conference on*; 2016. p. 1–6.

66. Bansal S, Kumar D. IoT Application Layer Protocols: Performance Analysis and Significance in Smart City. *In: 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*; 2019. p. 1–6.
67. Ghotbou A, Khansari M. Comparing application layer protocols for video transmission in IoT low power lossy networks: an analytic comparison. *Wireless Networks*. 2012;27:269–283.
68. Grgic K, Speh I, Hedi I. A web-based IoT solution for monitoring data using MQTT protocol. *In: 2016 International Conference on Smart Systems and Technologies (SST)*; 2016. p. 249–253.
69. Yuanping L, Wang J, Feng L, Xue W. Accelerating Sequence Event Detection through Condensed Composition. *In: 2010 Proceedings of the 5th International Conference on Ubiquitous Information Technologies and Applications*; 2010. p. 1–6.
70. Xingyi J, Xiaodong L, Kong N, Yan B. Efficient Complex Event Processing over RFID Data Stream. *In: Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*; 2008. p. 75–81.
71. Kuboi S, Baba K, Takano S, Murakami K. An Evaluation of a Complex Event Processing Engine. *In: 2014 IIAI 3rd International Conference on Advanced Applied Informatics*; 2014. p. 190–193.
72. Wu E, Diao Y, Rizvi S. High-performance complex event processing over streams. *In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*; 2006. p. 407–418.
73. Demers A, Gehrke J, Panda B, Riedewald M, Sharma V, White W. Cayuga: A General Purpose Event Monitoring System. *In: CIDR 2007: Third Biennial Conference on Innovative Data Systems Research*; 2007. p. 412–422.
74. Agrawal P, Benjelloun O. A system for data, uncertainty, and lineage. *In: Proceedings*

- of the 32nd International Conference on Very Large Data Bases*; 2006. p. 1151–1154.
75. Kimmig A, Demoen B, Raedt LD. On the implementation of the probabilistic logic programming language:ProbLog. *Theory and Practice of Logic Programming*. 2011;11(8):235262.
 76. Sadilek A, Kautz H. Location-based reasoning about complex multi-agent behavior. *Journal of Artificial Intelligence Research*. 2012;43:37–45.
 77. Morariu VI, Davis LS. Multi-agent event recognition in structured scenarios. *Computer Vision and Pattern Recognition (CVPR)*. 2011:32893296.
 78. Etzion O, Niblett P. *Event Processing in Action*. New York: Manning Publications; 2010.
 79. Baarah A, Mouttham A, Peyton L. Improving Cardiac Patient Flow Based On Complex Event Processing. In: *Applied Electrical Engineering and Computing Technologies (AEECT), 2011 IEEE Jordan Conference on*; 2011. p. 1–6.
 80. Lee RK, Yu CH, Liang MS, Feng MW. An Approach to Children Surveillance with Sensor-Based Signals Using Complex Event Processing. In: *2009 IEEE International Conference on e-Business Engineering*; 2009. p. 596–601.
 81. Peng Z, Bo C, Junliang C. A Complex Event Processing based Alarm System for Coal Mine Safety Monitoring. In: *2011 International Conference on Computer Science and Network Technology*; 2011. p. 943–947.
 82. Strielkina A, Uzun D, Kharchenko V. Modelling of Healthcare IoT Using the Queueing Theory. In: *The 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*. IEEE; 2017. p. 849–852.
 83. Maheshwari S, Raychaudhuri D, Seskar I, Bronzino F. Scalability and Performance

- Evaluation of Edge Cloud Systems for Latency Constrained Applications. *In: 2018 Third ACM/IEEE Symposium on Edge Computing*. IEEE; 2018. p. 286–299.
84. Abdelradi Y, El-Sherif A, Afify L. A Queueing Theory Approach to Small-Cell Assisted IoT Traffic Offloading. *In: 2019 IEEE Global Conference on Internet of Things (GCIoT)*. IEEE; 2019. p. 286–299.
 85. Oluwadere SA, Agbonifo OC, Babatunde AT. Network Traffic Analysis Using Queueing Model and Regression Technique. *Journal of Information*. 2019;16–26.
 86. Gomathi R, Mahendran N. An Efficient Data Packet Scheduling Schemes in Wireless Sensor Networks. *In: 2nd International Conference on Electronics and Communication System*. IEEE; 2015. p. 542–547.
 87. Afzal B, Alvi SA, Shah GA, Mahmood W. Energy efficient context aware traffic scheduling for IoT applications. *Ad Hoc Networks*. 2017;62:101–115.
 88. Nasser N, Karim L, Taleb T. Dynamic multilevel priority packet scheduling scheme for wireless sensor network. *IEEE Transactions on Wireless Communications*. 2013;12:1448–1459.
 89. Kua J, Nguyen S, Armitage G, Branch P. Using Active Queue Management to Assist IoT Application Flows in Home Broadband Networks. *IEEE Internet of Things Journal*. 2017;4:1399–1407.
 90. Hamdi MM, Rashid SA, Ismail M, Althahrawi A, Mansor MF, AbuFoul K. Performance Evaluation of Active Queue Management Algorithms in Large Network. *In: 2018 IEEE 4th International Symposium on Telecommunication Technologies (ISTT)*. IEEE; 2018. p. 1–6.
 91. Al-Khafajiy M, Baker T, Waraich A, Al-Jumeily D, Hussain A. IoT-Fog Optimal Workload via Fog Offloading. *In: IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*; 2018. p. 359–364.

92. Balevi E, Gitlin R. Optimizing the Number of Fog Nodes for Cloud-Fog-Thing Networks. *IEEE Access*. 2016:11173–11183.
93. Kattepur A, Dohare H, M Visali M, and Hemant KR, Anantha S. Resource Constrained Offloading in Fog Computing. *In: 1st Workshop on Middleware for Edge Clouds & Cloudlets*; 2016. p. 1–6.
94. Maiti P, Sahoo B, Turuk AK, Kumar A, Choi BJ. Internet of Things applications placement to minimize latency in multi-tier fog computing framework. *ICT Express*. 2021.
95. Martin B, Kovacs E, Schulke A, Ito N, Criminisi C, Goix LC, Valla M. The context API in the OMA next generation service interface. *In: 14th International Conference on Intelligence in Next Generation Networks (ICIN)*; 2010. p. 1–5.
96. Skarlat O, Nardelli M, Schulte S, Borkowski Leitner P. Optimized IoT service placement in the fog. *Service Oriented Computing and Applications*. 2018:427–443.
97. Skarlat O, Nardelli M, Schulte S, Borkowski Leitner P. Resource provisioning for IoT services in the fog. *In: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications, SOCA 2016*. IEEE; 2016. p. 32–39.
98. Skarlat O, Nardelli M, Schulte S, Dustar S. Towards QoS-Aware Fog Service Placement. *In: 2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC 2017*. IEEE; 2017. p. 89–96.
99. Tran N, Skhiri S, Zimanyi E. EQS: An elastic and scalable message queue for the cloud. *In: 2011 3rd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2011*; 2011. p. 391–398.
100. Dobbelaere P, Esmaili KS. Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry paper. *In: 11th ACM International Conference on Distributed and Event-based Systems*; 2017. p. 227–238.

101. Jaber G, Kacimi R, T Gayraud T. Data Freshness Aware Content-Centric Networking in WSNs. *In: 2017 Wireless Days*; 2017. p. 238–240.
102. Lee SH, Jin HW. A Model for Analyzing Data Freshness of Periodic Real-time Communication. *In: ACM/IEEE 4th International Conference on Cyber-Physical Systems*; 2013. p. 260–260.
103. Bright L, Raschid L. Using Latency-recency Profiles for Data Delivery on the Web. *In: 28th International Conference on Very Large Data Bases*; 2002. p. 550–561.
104. Cho J, Garcia-molina H. Synchronizing a database to Improve Freshness. *In: SIGMOD Record (ACM Special Interest Group on Management of Data)*; 2000. p. 550–561.
105. Labrinidis A, Roussopoulos N. Using Latency-recency Profiles for Data Delivery on the Web. *In: 29th International Conference on Very Large Data Bases*; 2003. p. 393–404.
106. Peralta V, Ruggia R, Bouzeghoub M. Analyzing and Evaluating Data Freshness in Data Integration Systems. *In: Ingénierie des Systèmes d'Information*; 2004. p. 145–162.
107. Happ D, Karowski N, Menzel T, Handziski V, Wolisz A. Meeting IoT Platform Requirements with Open Pub/Sub Solution. *Annals of Telecommunications*. 2016;72.
108. Johnsen F. Using Publish/Subscribe for Short-lived IoT Data. *In: 2018 Federated Conference on Computer Science and Information Systems*; 2018. p. 645–649.
109. Lindgren P, Kyusakov R, Eliasson J, Makitaavola H, Pietrzak P. A SOA approach to delay and jitter tolerant distributed real-time Complex Event Processing. *In: 2013 IEEE International Symposium on Industrial Electronics: ISIE13*; 2013. p. 1–6.
110. AlAli AR, Zualkernan IA, Rashid M, Gupta R, Alikarar M. A smart home energy management system using IoT and big data analytics approach. *IEEE Transactions on*

Consumer Electronics. 2017 11:426–434.

111. Shah J, Mishra B. IoT enabled environmental monitoring system for smart cities. *In: 2016 International Conference on Internet of Things and Applications (IOTA)*; 2016. p. 383–388.
112. Dandala TT, Krishnamurthy V, Alwan R. Internet of Vehicles (IoV) for traffic management. *In: 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*; 2017. p. 1–4.
113. Xiaojun C, Xianpeng L, Peng X. IOT-based air pollution monitoring and forecasting system. *In: 2015 International Conference on Computer and Computational Sciences (ICCCS)*; 2015. p. 257–260.
114. Shah T, Yavari A, Mitra K, Saguna S, Jayaraman PP, Rabhi F, Ranjan R. Remote health care cyber-physical system: quality of service (QoS) challenges and opportunities. *IET Cyber-Physical Systems: Theory Applications*. 2016:40–48.
115. Sharma R, Kumar N, Namratha BG, Srinivas T. Probabilistic Prediction based Scheduling for Delay Sensitive Traffic in Internet of Things. *Procedia Computer Science*. 2015:90 – 97. The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015).
116. Awan I, Younas M, Naveed W. Modelling QoS in IoT Applications. *In: 2014 17th International Conference on Network-Based Information Systems*; 2014. p. 99–105.
117. Hernandez R, Dixys L, Carames F, Tiago M, and Lamas F, Escudero P, Carlos J. A Plug and Play Human-Centered Virtual TEDS Architecture for the Web of Things. *Sensors*. 2018.
118. Farris I, Orsino, Militano L, Nitti M, Araniti G, Atzori, Iera A. Federations of connected things for delay-sensitive IoT services in 5G environments. *In: 2017 IEEE*

International Conference on Communications (ICC); 2017. p. 1–4.

119. Brogi A, Forti S. QoS-Aware Deployment of IoT Applications Through the Fog. *IEEE Internet of Things Journal*. 2017:1185–1192.
120. Cugola G, Alessandro M, Matteucci M, Tamburrelli G. Introducing uncertainty in complex event processing: model, implementation, and validation. *Computing*. 2015 2;97:103–144.
121. Bellur U. A Cost Model for Complex Event Operators. In: *8th ACM International Conference on Distributed Event-Based Systems: DEBS '14*. ACM; 2014. p. 314–317.
122. Rastogi S, Srivastava S. Comparative Analysis of Different Queuing Mechanisms in Heterogeneous Networks. *International Journal of Advanced Research in Computer and Communication Engineering*. 2013.
123. Momanyi E, Oduol V, Musyoki S. First In First Out And Priority Packet Scheduling Based On Type Of Service. *Journal of Information Engineering and Applications*. 2014.
124. Chen J, Chen C. Design of complex event-processing IDS in internet of things. In: *2014 6th International Conference on Measuring Technology and Mechatronics Automation: ICMTMA 2014*. IEEE; 2014. p. 226–229.
125. Dunkels A, Osterlind F, He Z. An Adaptive Communication Architecture for Wireless Sensor Networks. In: *SenSys'07*; 2007. p. 1–6.