Adaptive Quality of Service for IoT-based Wireless Sensor Networks

Syarifah Ezdiani Binti Syed Nor Azlan

A thesis submitted to

Auckland University of Technology

in fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

School of Engineering, Computer and Math Sciences

Auckland, New Zealand

2018

Supervisors

Prof. Adnan Al-Anbuky & Assoc. Prof. Nurul Sarkar

Abstract

The future of the Internet of Things (IoT) is envisaged to consist of a high amount of wireless resource-constrained devices connected to the Internet. Moreover, a lot of novel real-world services offered by IoT devices are realised by wireless sensor networks (WSNs). Integrating WSNs to the Internet has therefore brought forward the requirements of an end-to-end quality of service (QoS) guarantees.

In this thesis, a QoS framework for integrating WSNs with heterogeneous data traffic is proposed. The concept of Adaptive Service Differentiation for Heterogeneous Data in WSN (ADHERE) is proposed based on the varying QoS factors and requirements analysis of mixed traffic within an IoT-based WSN. The objective of the QoS framework is to meet the requirements of heterogeneous data traffic in the WSN - in the domain of timeliness and reliability. Another objective is to implement an adaptive QoS scheme that can react to dynamic network changes.

This thesis provides the literature analysis and background study for integrating a WSN which contains heterogeneous data traffic with the Internet. In the discussion of network modelling and implementation tools for the testing, this thesis provides an insight into the different tools that are available and their ability to investigate the concept of service differentiation among heterogeneous traffic within the IoT-based WSN network. Furthermore, the major components of ADHERE are presented in the Concept chapter. The major components are: a heterogeneous traffic class queuing model that encompasses a service differentiation policy, a congestion control unit and a rate adjustment unit that supports the adaptive mechanism.

Network modelling and the simulation of an ADHERE QoS framework which is carried out primarily using the network simulation tool, Riverbed Modeler, are also presented. Additionally, a proposed co-simulation between Riverbed Modeler and MATLAB is introduced, which aims to provide a seamless QoS monitoring using the ADHERE concept. The simulation results suggest that real-time traffic achieves low bound delay while delay-tolerant traffic experiences a lower packet drop. This indicates that the needs for real-time and delay-tolerant traffic can be better met by treating both packet types differently using ADHERE. Furthermore, a verification and added-value to the ADHERE QoS model using a neural network is also presented. The learning capabilities in ADHERE optimise the QoS framework's performance by accommodating the QoS requirements of the network through the unpredictable traffic dynamics and when complex network behaviour takes place. Before concluding the thesis, the implementation of ADHERE QoS as a use-case on a physical test environment is also discussed. The test environment offers a flexible system that is capable of reacting to the dynamic changes of process demands. Physical network performance can be predicted by analysing the historical data in the background on a network simulator or virtual network. Finally, this thesis offers a conclusion with an indication of our future research work.

List of Publications

[1] S. Ezdiani and A. Al-Anbuky, "Modelling the integrated QoS for wireless sensor networks with heterogeneous data traffic," *Open Journal of Internet of Things (OJIOT)*, vol. 1, 2015, pp. 1-15.

[2] S. E. S. N. Azlan and A. Al-Anbuky, "Quality of Service Modelling for Federated Wireless Sensor Network Testbed Gateways," in *Proc. of 5th Int. Conf. on Commun., Theory, Reliability, and Quality of Service (CTRQ 2012)*, Chamonix/ Mont Blanc, France, 2012, pp. 14-18.

[3] S. Ezdiani, I. S. Acharyya, S. Sivakumar, and A. Al-Anbuky, "An IoT environment for WSN adaptive QoS," in *Proc. of IEEE Int. Conf. on Data Science and Data Intensive Systems*, 2015, pp. 586-593.

[4] S. E. Syed Nor Azlan, I. S. Acharyya, S. Sivakumar, and A. Al-Anbuky, "An architectural concept for sensor cloud QoSaaS testbed," in *Proc. of the 6th ACM Workshop on Real World Wireless Sensor Networks (RealWSN 2015)*, pp. 15-18.

[5] S. Ezdiani and A. Al-Anbuky "Integrating WSN with the Internet: QoS Analysis and Modeling for Heterogeneous Data Traffic", presented at the Wireless Telecommunication Symposium (WTS 2014), Washington DC, 2014.

[6] S. Ezdiani, A. Indrajit S, S. Sivakumar, and A. Al-Anbuky, "Wireless Sensor Network Softwarization: Towards WSN Adaptive QoS," *IEEE Internet of Things Journal,* vol. 4, pp. 1517 - 1527, 2017

Contents

List of Figures

List of Tables

List of Abbreviations

ADHERE	Adaptive Service Differentiation for Heterogeneous Data in WSN
AF	Assured Forwarding
AQoS	Adaptive Quality of Service
AR	Arrival Rate
BS	Base Station
CBR	Constant Bit Rate
CSMA	Carrier Sense Multiple Access
DES	Discrete Event-based Simulation
DT	Delay-Tolerant
DTN	Delay-Tolerant Network
EF	Expedited Forwarding
FIFO	First-in-First-out
FQ	Fair Queuing
FTP	File Transfer Protocol
GlomoSim	Global Mobile Information System Simulator
HRT	Hard Real-Time
IoT	Internet of Things
MSE	Mean Square Error
non-RT	non-Real-Time
PCCP	Priority-based Congestion Protocol
PDR	Packet Delivery Ratio
PQ	Priority Queuing
PSC	Physical Sensor Cloud
PSN	Physical Sensor Network
QoS	Quality of Service
RFID	Radio-Frequency Identification
RSSI	Received Signal Strength Indicator
RT	Real-Time
RT-Rel	Real-Time and Reliable
RTT	Round-Trip Time
SeNSe	Sensor Network and Smart Environment Research Centre
SR	Service Rate
SRT	Soft Real-Time
Tcl	Tool command language

TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
ToS	Type of Service
UDP	User Datagram Protocol
VSN	Virtual Sensor Network
WFQ	Weighted Fair Queuing
WRR	Weighted Round Robin
WMSN	Wireless Multimedia Sensor Networks
WSAN	Wireless Sensor and Actuator Networks
WSN	Wireless Sensor Networks

Attestation of Authorship

“I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been accepted for the award of any other degree or diploma of a university or other institution of higher learning.”

Syarifah Ezdiani Binti Syed Nor Azlan: ___ [illegible] ____________

Date: __30 August 2018___

Acknowledgements

In the name of Allah, the most Gracious, the most Merciful.

First of all, I would like to thank my supervisor Professor Adnan Al-Anbuky for being such an inspiring person to work with. This thesis would not have been possible without Professor Adnan's endless support, understanding and encouragement. His enthusiasm and love for knowledge and positivity in problem-solving have been such an inspiration to help me persevere in this PhD journey. Many thanks also go to my co-supervisor, Associate Professor Dr. Nurul Sarkar for his reviews and thoughtful suggestions throughout my study.

I dedicate this thesis to my late parents, Normah and Syed Nor Azlan. I am endlessly grateful to my mother who, despite no longer being with us, still manages to inspire me to be a positive and strong, yet kind, individual. Without the blessing and encouragement from my father, the decision to start my PhD in New Zealand would not have been made in the first place. My gratitude also goes to my late father-in-law, Zulkifli, who had passed away a few weeks before my viva voce exam, and my mother-in-law, Azizah, for their continuous support and prayers. My deepest thanks also to my sisters and brother for taking a general interest in my work, and most importantly, for taking great care of our father until he left to meet his Creator in the middle of my studies.

Thanks also to all members of the Sensor Network and Smart Environment Research Centre (SeNSe) lab for contributing to an inspiring work environment. Special thanks to Siva, Indrajit, Craig and Duaa for many interesting and fruitful discussions. My gratitude also goes to my *usrah* sisters in Auckland for being part of my support system throughout my PhD journey.

Finally, my deepest gratitude to my dearest husband, Sazli Zulkifli, for your love, patience and encouragement. Thank you for being such a great father to our children Faidhi, Fadlan, Farhad and our newest addition, Fatimah.

Chapter 1 Introduction

Rapid technological advances in wireless communication systems, small-scale energy supplies, microprocessors, low power digital electronics and low power radio technologies today enable low-power multi-functional sensor devices to detect and react to changes in their surrounding environments. These sensors devices are equipped with a tiny microprocessor, a small battery and a set of transducers that are used to acquire information that reflects the changes in the surrounding environment of the sensor devices. Consequently, the emergence of low power and minute wireless sensor devices has led to the development of wireless sensor networks (WSNs).

WSNs have been deployed in diverse applications such as health monitoring, environmental observation, structural monitoring, habitat monitoring and disaster management. With the emergence of various important WSN applications, the integration of WSNs with the Internet has become inevitable. The integration provides seamless access to the unattended devices, hence offering high-resolution knowledge about the sensed phenomena. Indeed, the integration of WSN to the cloud and through the Internet has become one of the prime technologies that bring the Internet of Things (IoT) to reality.

In numerous WSN applications, the co-existence of two types of data traffic can be observed - those that must be sent promptly and those that must be sent reliably. Many applications also involve unpredictable traffic flows, typically from generations of bursts of data traffic during the occurrence of important events. This highlights the need for an application-specific mechanism that manages the heterogeneous data which also takes into account the traffic dynamics and the varying QoS requirements of different traffic types.

This chapter starts with an overview of WSN and its role in the IoT. Then, the typical architecture of IoT-based WSN is presented. In Section 1.2, the research background is given, focussing on WSNs applications with heterogeneous data traffic, the traffic dynamics of mixed traffic and the pertaining traffic QoS domains classifications. Then, the research motivation, problem statement, research objectives and research contribution are identified in Section 1.3 to 1.5. Finally, the organisation of the thesis is outlined in Section 1.6.

1.1 WSN and the Internet of Things

The future of the Internet of Things (IoT) is envisaged to consist of a high amount of wireless resource-constrained devices connected to the Internet. The pervasive presence of a variety of things or objects such as Radio-Frequency Identification (RFID) tags, sensors, actuators and mobile phones supports interactions and cooperation between these objects or things to reach common goals. This enables the IoT [7, 8] to rapidly gain ground in the scenario of modern wireless telecommunications. WSNs integrated to the Internet allow for an autonomous and intelligent link between the virtual world and the physical world. Consequently, numerous novel real-world services offered by IoT devices are realised by WSNs. This Section discusses the role of WSNs in the IoT and gives an overview of the IoT architecture.

1.1.1 WSNs Role in the Internet of Things

The efforts to integrate WSN with the Internet have been around for more than a decade, especially with the emergence of many important WSN applications. The integration of WSN to the cloud [9] and through the Internet has also become one of the prime technologies that bring the IoT to reality. The multi-faceted potential of the IoT makes the development of a wide range of applications possible. These applications can be categorised into transportation and logistics, healthcare, smart environments, personal and social domains [7].

By integrating WSN to the Internet, the notion of managing sensor nodes as well as accessing data streams produced by WSN from any geographical location can become a reality. The integration has become a solution for seamless knowledge sharing and large-scale testing. Realising the tremendous benefits of the integration, the area of networking WSNs with the Internet has gained considerable attention from the WSN research community over the past few years.

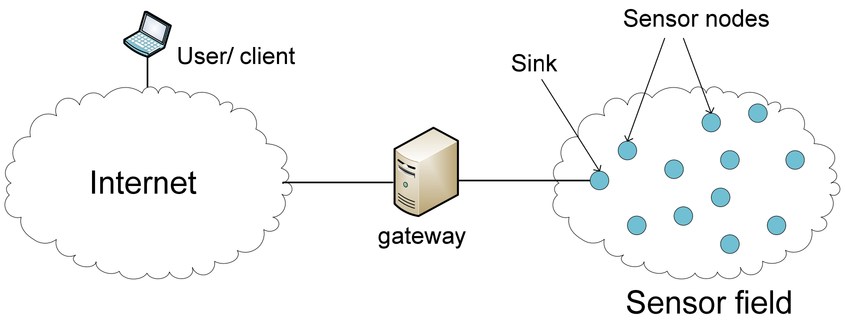


Figure 1-1 A typical WSN and the Internet integration topology

Figure 1-1 shows a typical WSN and the Internet integration topology. WSNs, along with the Internet, enable a user or client to monitor a phenomenon of interest remotely. The sensor nodes are typically resource-constrained devices in terms of energy supply, storage capacity and computational capabilities. Each sensor node collects event data monitored around their vicinity and periodically sends its reading to a more powerful node which acts as a data warehouse, commonly referred to as the sink. In addition, multiple sinks may be required in a large WSN. For the Internet, hosts called the users (or clients) send queries for data collection and process the received query responses. In this typical architecture, an interface usually referred to as the gateway provides an interface between the sensor network and the Internet. Hence, it serves as the last mile of connectivity by bridging the WSN sink to the Internet.

The most common integration approach is to employ a gateway-based solution [10, 11]. In this strategy, the sink, or the base-station of the WSN serves directly as an interface between the sensor network and the Internet. The sink operates as a gateway, i.e. a proxy, that performs the translation of lower layer protocols from the WSN to the Internet, and vice versa. There are variations to this solution, specifically by having different gateway capabilities, namely the application-level gateway solution and the delay-tolerant network (DTN) gateway [12] solution. Another approach is through direct integration of the IP stack on the smart sensor level, which makes it possible to connect WSNs and the Internet without requiring proxies or gateways. In this approach, the sink or the base-station acts as a router, mainly to forward the packets from and to sensor nodes. An overview of IP-based integration over recent years is given in [13].

In many circumstances nowadays, disparate WSNs need to be integrated into one virtual sensor network over wired/wireless networks to provide comprehensive services to users [14]. For example, in some WSN applications, the actual condition of a phenomenon may be determined through a combination of sensory data from nodes that may be constituents of different WSNs. Figure 1-2 shows a simple diagram of WSN islands connected through the Internet.

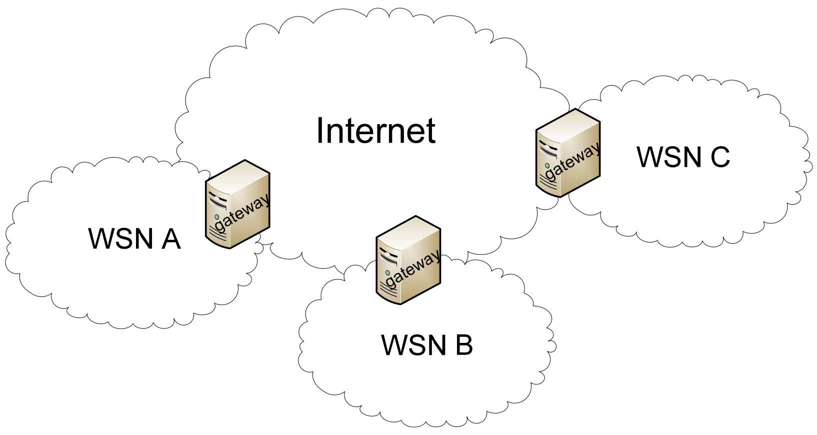


Figure 1-2 Integration of isolated WSNs

1.1.2 IoT Architecture

Figure 1-3 shows a typical IoT-WSN architecture. The network is established with 3 tiers comprised of physical sensor devices, Internet interface and remote server or cloud support architecture.

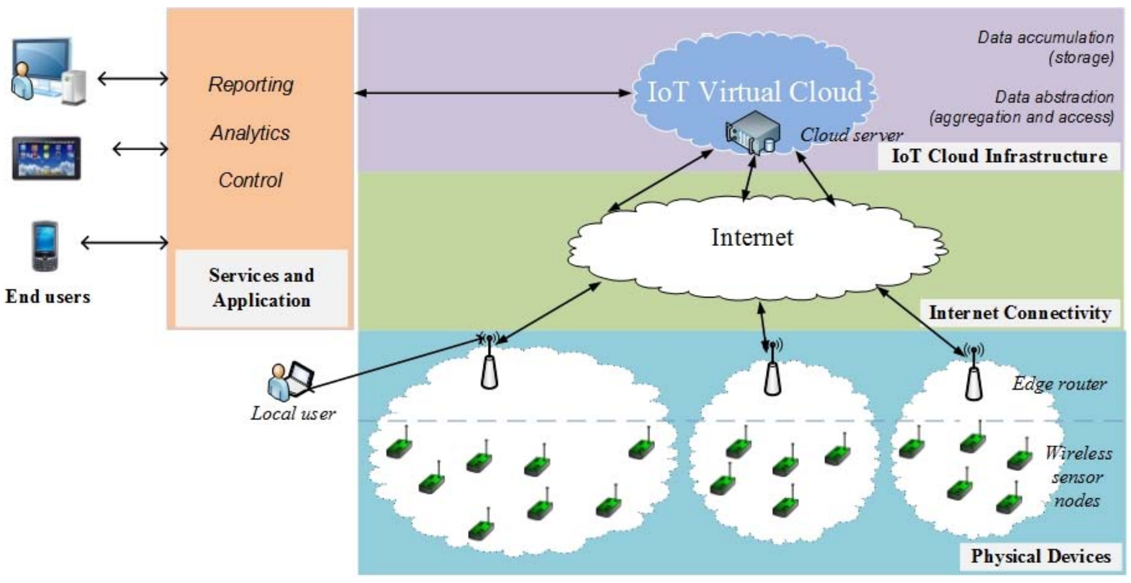


Figure 1-3 IoT architecture

The lowest tier consists of physical devices such as the sensor nodes and edge routers. Sensor nodes send captured data to the edge router, which acts as a sink to the WSN. The routers are typical IP addressable sensor nodes representing an IoT access point that gains connectivity to the cloud server through the Internet routers. Data passes through the multi-hop Internet routers, which eventually reside in the cloud where data accumulation and data abstraction take place. Next, the intermediate level consists of the Internet as the interconnection backbone. This architecture is one of the common approaches when integrating the WSN to the Internet as discussed in the previous section and categorised as the gateway solution [15]. Finally, the upper level represents the IoT virtual cloud infrastructure and user terminals. Data from the physical sensor nodes are pushed to the IoT cloud server through the edge router.

The common services and applications offered to the IoT include reporting, analytics and control of the physical devices. To provide IoT services and applications to end users, the system may be equipped with a database for storing and reporting related historical data and a software environment for hosting analytics and control activity. This is demonstrated in [16, 17] which categorised the IoT applications and their associated service model when identifying the QoS requirements related to the application domain. Indeed, the QoS requirement is closely dependant on the end-to-end interaction in a cloud centric IoT framework [8]. Hence, the research background involves the domains of application-specific WSN QoS and its relevance to the application traffic's QoS requirements. This is discussed in the next section.

1.2 Research Background

It is evident that the concept and emerging applications of the IoT-based WSN have promoted a diverse research effort among the industry and academic community. Work in this area mainly involves IoT architecture and its future direction, surveys of its applications and enabling technologies [7] and security [18, 19]. Nevertheless, there have been efforts investigating the QoS issues in the IoT [17], particularly for specific IoT applications [20]. In addition, the QoS support in WSNs remains an open research field from various perspectives [21]. The background to this research is formulated within the context of the WSN application with heterogeneous traffic types and the relevant QoS requirements. Therefore, WSN applications with heterogeneous data traffic are discussed in the following section. This is followed by a consideration of the traffic dynamics of the WSN application and then the domain of QoS requirements in WSN.

1.2.1 Heterogeneous Data in WSN Applications

WSN can be deployed in various domains and applications such as agriculture and environmental sensing, health care, wildlife or habitat monitoring, military surveillance, home automation and security. There has also been a growing deployment of wireless multimedia sensor networks (WMSN) [22] and an extended application of WSNs. A WMSN is equipped with a multimedia device such as a miniaturised microphone, battery or video transceiver. A WMSN sensor node may have different sensors that gather different data at a different sampling rate. WMSNs can also transmit multimedia data such as still images, video, and audio streams along with the ability to monitor data. It is evident that the co-existence of at least two types of data traffic can be observed in many WSN applications – real-time data and delay-tolerant data. The traffic data within an application typically differ regarding the time required for information to be sent to the destination.

An inspired example of a WSN application consisting of different data types is the MIT CarTel project [23], which collects multiple real-time and delay-tolerant data within a vehicular network. In this network, a mobile sensor computing system was designed and implemented to collect, process, deliver and visualise data from sensors located on mobile units such as automobiles. A node in the WSN application is a mobile embedded computer, coupled to a set of sensors. Each node gathers and processes sensor readings locally before delivering them to a central portal, where the data is stored in a database for further analysis and visualisation. Data on cars is delivered to a portal, where users can browse and query it via a visualisation interface and local snapshot queries.

The application provides a simple query-oriented programming interface and handles large amounts of heterogeneous data from sensors. Such data may include GPS data about road traffic speed and delays, the quality and prevalence of Wi-Fi access points on drive routes, images from an attached camera and onboard automotive diagnostic data. In addition, the nodes rely primarily on opportunistic wireless connectivity to the Internet, or to "data mules" such as other mobile nodes to communicate with the portal. The system’s applications run on the portal, using a delay-tolerant continuous query processor to specify how the mobile nodes should summarise, filter and dynamically prioritise data. The collected and processed data is accessible to users through the portal of a website.

1.2.2 QoS Domains Classifications

QoS refers to the kind of quality perceived by the user or application. In networking communities, QoS is interpreted as a measure of service quality that the network offers to the end user or application. QoS has also been defined as a set of service requirements that are fulfilled when transmitting a stream of packets from source to destination.

The advent of multiple traffic types in WSNs highlights the need for a QoS mechanism to handle this heterogeneous traffic data. Real-time and delay-tolerant data have different QoS requirements. Typically, real-time data need to be sent in promptly, and delay-tolerant data is sent to the destination reliably. Therefore, the QoS requirements can be classified into two domains: *timeliness* and *reliability* [21, 24]. Within the timeliness domain, different types of data may have different deadlines – some may be shorter, and others may be longer. Similarly, the sensory data may also have diverse reliability requirements – some data can tolerate a certain percentage of loss during transmission, whereas others may need to be delivered to the destination without any loss.

For example, the real-time and the delay-tolerant data in the vehicular network project [23] introduced in Section 1.2.1 are represented by GPS data and road-surface anomalies data, respectively. The real-time data is defined as the GPS data from the vehicles – they need to be collected quickly as they are used to model traffic delay; however, they do not necessarily need to be sent reliably. On the contrary, the data that detects road-surface anomalies such as potholes is categorised as the delay-tolerant data – they require high reliability to avoid false alarm, but do not need to be sent quickly.

An animal farming application [25] which consists of animal farms tagged with sensor nodes may generate two types of packets – one for the environment data of the surrounding environments and the other for the health condition of the herds. The former is considered to be delay-tolerant data. The latter must be sent quickly, especially during emergencies, hence is assumed to be real-time data.

1.2.3 Traffic Dynamics

In various WSN applications, the occurrence of an important event can suddenly generate a burst of data traffic. Data in a WSN are normally generated and sent periodically to the sink. However, a burst of data traffic can also be suddenly generated when an important event is triggered or detected. This is demonstrated in WSN applications such as fire hazard monitoring applications [26] and intruder detection systems [27].

In an application of fire hazard monitoring [26], the burst of data in a bushfire application when a fire occurs is considered to be very critical. Hence, different data packets generated within the network might have differing importance. Typically, the network should make more effort to deliver higher priority packets.

Traffic dynamics can also be seen in an intruder detection systems [27] [28]. Typically, data in an intruder detection application is periodically sent to the sink node. However, when an important event occurs in the system, the sensor node that detected the event should send an alarm message to the sink. This alarm message could be in the form of multiple packets containing information such as the time and place of the intrusion. Usually, this kind of high priority occurs in bursts. In other words, high priority traffic is generated within a short period, while low priority traffic exists in the network and periodically generates thousands of packets.

1.3 Research Motivation

The motivation behind this study is twofold:

Firstly, traffic in WSNs represents two kinds of co-existing data packets: those with real-time constraints and those with reliability-constraints. These packets have different QoS requirements. Thus, by treating these packets differently, the needs of both packet types can be better met. Furthermore, service differentiation is a common approach to manage heterogeneous traffic in WSN applications and provide the required QoS imposed by different kinds of network traffic. Due to significant differences between WSN applications and the Internet, the QoS requirements generated by both networks vary greatly [29]. The traditional approach for QoS traffic is simply unacceptable in WSNs. Hence, the interoperability between WSNs and the Internet that employs different QoS mechanism may also influence the network performance. A mechanism for an end-to-end service differentiation can preserve the QoS implemented between different network layers.

Secondly, there is a need for an adaptive WSN network performance matrix to follow the demands of traffic dynamics and the physical process. We are motivated to devise an adaptive QoS mechanism that can react to dynamic changes in the network to ensure seamless QoS interactions between WSNs and the Internet. Specifically, the service differentiation scheme can work with adaptive capabilities as it reacts to traffic dynamics.

1.4 Research Objectives

In this thesis, a QoS framework for integrating WSN with heterogeneous data traffic to the Internet is proposed. The concept of Adaptive Service Differentiation for Heterogeneous Data in WSN (ADHERE) is proposed based on the varying QoS factors and requirement analysis of mixed traffic within an IoT-based WSN. The objective of the QoS framework is to meet the requirements of heterogeneous data traffic in the WSN both in the domain of timeliness and reliability. Another objective is to implement an adaptive QoS scheme that can react to traffic dynamics. This research also aims to carry out the verification and validation of the simulated QoS model on the physical setting of a sensor network.

1.5 Research Scope and Contribution

Most of the work in previous studies only supports two major types of traffic classes: real-time traffic and non-real-time traffic. Hence, they only consider the QoS associated with timeliness requirements. This thesis is based on the belief that both the timeliness and reliability QoS requirements need to be addressed in a WSN. In addition, in contrast to a common belief that real-time data is much more important than the delay-tolerant data, it has been shown that delay-tolerant data has reliability requirements that must be carefully managed. Furthermore, the traffic dynamics among heterogeneous data traffic needs to be addressed to maintain the QoS of all traffic types. In this context, this research proposes an adaptive QoS model that also considers the traffic dynamics within a WSN application. In addition, the network QoS conditions are also monitored and maintained by introducing adjustments to network parameters.

This thesis has three unique contributions. Firstly, it involves the design and implementation of an ADHERE QoS model, and the testing is performed on Riverbed Modeler, where different network scenarios and test cases are simulated. The second contribution is the implementation of the QoS model on an IoT-based WSN physical test environment. Thirdly, this thesis proposed the use of a neural network to introduce an added-value to the proposed ADHERE QoS model.

1.6 Thesis Organisation

This structure of the thesis is organised as follows:

Chapter 1 gives an overview of the current status of WSNs and the IoT. The background outlined in this chapter includes WSN applications with heterogeneous data traffic and QoS pertaining to the IoT-based WSN and the QoS domain classifications. The research motivation, problem statement, research objectives and research contribution are also discussed.

Chapter 2 provides an insight into the work related to the aspect of QoS for IoT-based WSNs and heterogeneous data traffic within WSNs. A critical analysis of service differentiation for real-time and delay-tolerant data is presented, and the requirements for both QoS domains are highlighted.

Chapter 3 deals with the modelling and implementation tools for the testing. It provides an insight into the different available tools and their ability to investigate the concept of service differentiation among heterogeneous traffic within an IoT-based WSN network.

Chapter 4 introduces the ADHERE QoS framework. This chapter presents the ideology of the adaptive QoS concept and provides an overview of the major components of ADHERE.

Chapter 5 presents the network modelling and simulation of the ADHERE QoS framework. A proposed co-simulation between Riverbed Modeler and MATLAB is also presented in this chapter, which aims to provide seamless QoS monitoring using the ADHERE concept.

Chapter 6 presents verification of the simulated ADHERE QoS framework. An added-value to the ADHERE QoS model is presented using a neural network. It gives an overview of the design of the learning algorithm, which complements the developed adaptive QoS model.

Chapter 7 presents the implementation of ADHERE QoS as a use-case on a physical test environment, which offers a flexible system capable of reacting to the dynamic changes of process demands.

Chapter 8 concludes this thesis with suggestions for future research work.

Chapter 2 Literature Review

2.1 Introduction

The task of connecting a WSN to the Internet brings several challenges, including the quality of service (QoS) provisioning for the integration. QoS must be taken into account to provide reliable network performance for the integration. One of the major challenges integrating WSNs to the Internet is to provide a reliable and efficient connection between the two networks. WSNs should interwork with the Internet to build an end-to-end application system for their users. However, due to the significant differences between WSNs and the Internet, the QoS requirements generated by both networks may differ greatly. Internet QoS is typically defined with certain parameters such as packet loss, delay, jitter, and bandwidth, whereas the QoS requirements in WSNs are mainly application-specific and defined as data accuracy, aggregation delay, coverage, exposure, fault tolerance, and network lifetime. This is due to the difference in WSN application domains and network properties.

In an environment of WSN integrated to the Internet [4], the heterogeneous traffic travelling on the WSN side is local traffic arriving from sensor nodes and targeted to the gateway to be relayed to the end-points on the other end of the Internet side. The service differentiation algorithm representing the QoS mechanism within the WSN typically governs this traffic.

In this chapter, firstly, a QoS requirement analysis in integrating WSNs with heterogeneous data traffic with the Internet is presented in Section 2.2. Next, Section 2.3 discusses the service differentiation QoS of previous literature, including the priority-based scheduling and congestion control mechanisms. The focus areas in the literature analysis are the WSN-IoT network components and the traffic dynamics of WSN applications. We highlight the need for an adaptive QoS mechanism to ensure the network could react to the dynamics of network traffic, which will be an integral part of our discussion about the envisioned QoS framework in Section 2.4. Finally, Section 2.5 concludes the Chapter.

2.2 Quality of Service for the WSN-Internet Integration

This section presents the QoS requirements for various levels of the IoT-based WSN networks components. The factors for enabling QoS based on various WSN-IoT service models are discussed first. Then, the QoS requirements and mechanisms in WSNs are distinguished from the QoS for the Internet. Sections 2.2.3 and 2.2.4 provide insights into the end-to-end communication and the aspects of QoS over heterogeneous networks, respectively.

2.2.1 Enabling QoS based on the IoT-based WSN Service Model

Figure 2-1 illustrates the architecture of WSNs integrated to the Internet. The network architecture comprises a three-level network. The bottom level represents multiple isolated WSNs, whereas the intermediate and upper levels consist of the Internet and user terminals, respectively.

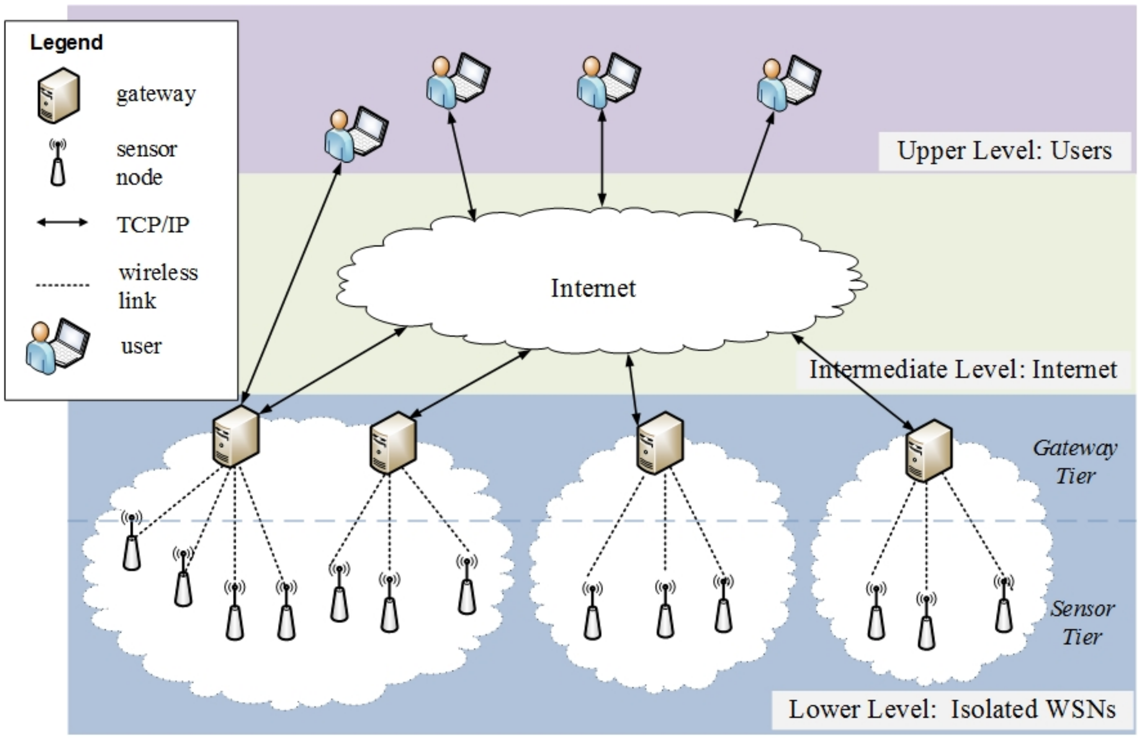


Figure 2-1 IoT-based WSN reference architecture

In a typical gateway-based integration solution, the gateway of the WSN serves directly as an interface between the sensor network and the Internet. It also operates as a proxy that performs the translation of lower layer protocols from the WSN to the Internet, and vice versa. In the IoT-based WSN environment, the gateway is also referred to as the *IoT access point* between the sensor network and the Internet.

M. Nef et al. [16] offers a description of the service model provided by WSNs in IoT. WSN-IoT service models are categorised by three factors: interactivity, delay, and the criticality of the WSN applications. The delay factors are categorised by the nature of the application traffic, namely non-real-time (non-RT), soft real-time (SRT) and hard real-time. Examples of WSN-IoT application fields are summarised in Table 2-1 [16, 17].

Table 2-1 Characteristics of the WSN-IoT Service Model

	Open Services Model	**Supple Services Model**	**Complete Services Model**
Interactivity	Yes	Subscription-specific	No
Delay	Non-RT	SRT	SRT/HRT
Criticality	No	Yes	Yes
Example of application field	-web or mobile application which gives information about nodal points (hospitals, drugstore, banks, museums), road incidents, meteorological data)	-management centre of traffic surveillance - collaborates with local institutions and users (radio stations, local authorities) and informs about traffic characteristics	-can be offered by service providers for professional use -monitors a patient in an intensive treatment unit over 24-hours -local authorities control the motorways and the streets with real-time management systems, locating dangerous drivers and deterring accidents
WSN-IoT applications: ***Transportations and Logistics***	• Augmented maps	• Logistics • Mobile ticketing • Monitoring environmental parameters	• Assisted Driving
Healthcare		• Identification and authentication • Data collection	• Tracking • Sensing
Smart Environment	• Smart museum and gym	• Comfortable homes and offices • Industrial plants	
Personal and Social	• Social networking • Historical queries		• Losses and thefts
Futuristic		• City information model	• Robot taxi • Enhanced game room

The first model is the *Open Services Model*. It is interactive as it is based on the user's queries, non-RT and non-mission-critical. The second model is the *Supple Services Model*. This model is sometimes interactive, sometimes not, depending on the user's subscription, it is SRT and mission-critical. The third model is the *Complete Services Model*. It is not interactive as there is a continuous flow of data; it is SRT or HRT, depending on the application, and is mission-critical.

The vehicular network application [23] briefly discussed in Chapter 1 consists of different data types that collect multiple real-time and delay-tolerant data within the network. The application provides a simple query-oriented programming interface and handles large amounts of heterogeneous data from sensors. These may include GPS data about road traffic speed and delays, the quality and prevalence of Wi-Fi access points on drive routes, images from an attached camera, and on-board automotive diagnostic data. In addition, the nodes rely primarily on opportunistic wireless connectivity to the Internet, or "data mules" such as other mobile nodes to communicate with the portal. The system’s applications run on the portal, using a delay-tolerant continuous query processor to specify how the mobile nodes should summarise, filter, and dynamically prioritise data. All of the collected and processed data is accessible to users through a website portal.

In a real-time system or through delay intolerant WSN applications, QoS guarantees can be categorised into two classes: hard real-time (HRT) and soft real-time (SRT). As stated in [30], a deterministic end-to-end delay bound should be supported in an HRT system. Hence, the arrival of a message after its deadline is considered a failure of the system. On the other hand, a probabilistic guarantee is required in an SRT system. Therefore, some lateness is tolerable.

Traditional QoS, such as those employed in the Internet, mainly result from the rising popularity of end-to-end bandwidth–hungry multimedia applications. On the contrary, as can be clearly seen from Table 2-1, the QoS solutions such as IntServ and DiffServ [31, 32] developed for traditional networks cannot be easily adopted in WSN. This is due to severe resource constraints, and the random deployment of sensor nodes, combined with application-specific and data-centric communication protocols in WSNs. Indeed, many existing studies have concluded that the end-to-end QoS parameters employed in traditional data networks such as the Internet are not sufficient to describe the QoS in WSNs [21, 29]. Consequently, in recent years, considerable efforts has been directed towards defining WSN QoS, which include QoS strategies through MAC protocols, routing protocols, data processing strategies, middleware and cross-layer designs. The difference between WSN QoS and Internet QoS must be distinguished to provide end-to-end QoS between WSN and the Internet, as presented in the following sub-section.

2.2.2 The difference between WSN QoS and Internet QoS

Since WSNs are envisioned to be employed in diverse applications, many researchers suggest that different WSN applications impose different QoS requirements. The two perspectives of QoS in WSNs are described in [33] to focus on the way the underlying network can provide the QoS to different applications:

Application-specific QoS. Regarding the application-specific QoS, the QoS parameters are chosen based on the way an application imposes specific requirements on sensor deployments, on the number of active sensors, or on the measurement precision of sensors. These attributes are all related to the quality of applications. The following QoS parameters may be considered to achieve the quality of applications: coverage, exposure, measurement errors, and the number of active sensors.

Network QoS. From the perspective of network QoS, the QoS parameters are chosen based on how data is delivered to the sink and corresponding requirements. The main objective is to ensure that the communication network can deliver the QoS-constrained sensor data while efficiently utilising network resources. The QoS parameters from this perspective include latency, delay, and packet loss, which are similar to traditional end-to-end QoS metrics. However, since WSNs are envisioned to be employed in diverse applications, a number of studies suggest that every different application imposes different QoS requirements.

Regarding the Internet QoS, the RFC 2368 [34] definition of QoS-based routing in the Internet characterises QoS as a set of service requirements to be met when transporting a packet stream from the source to its destination. Furthermore, Internet QoS refers to an assurance from the Internet to provide a set of measurable service attributes to the end-to-end users regarding the delay, jitter, and available bandwidth and packet loss. Therefore, QoS efforts have been pursued to end-to-end support using a large number of mechanisms and algorithms in different protocol layers, while maximising bandwidth utilisation.

QoS support from the Internet can be obtained using an over-provisioning of resources and traffic engineering. While traffic bursts in the network could cause congestion, the default approach of over-provisioning that treats users as the same service class may not always provide an acceptable solution. As a QoS-enabled network should be able to handle different traffic streams in different ways, this necessitates a traffic engineering approach that classifies users into classes with different priorities.

The IntServ model and the DiffServ model are typical QoS models employed in the Internet. They employ reservation-based and reservation-less approaches, respectively. While network resources are assigned according to an application's QoS request and subject to bandwidth management policy in IntServ, QoS in DiffServ is achieved via some strategies such as admission control, traffic classes, policy managers, and queuing mechanisms.

The QoS solutions such as IntServ and DiffServ are developed for traditional networks like the Internet. They cannot be easily ported in WSNs because of: severe resource constraints in sensor nodes, large-scale and random deployment of sensor nodes, and application specific and data-centric communication protocols in WSNs. Therefore, to employ network QoS for WSNs, D. Chen and P. K. Varshney [29] classifies WSN applications based on the basic data delivery models [35] to map the QoS requirements. The data delivery models of the upstream traffic, i.e., from sensor nodes to the gateway can be classified into four categories: event-based, continuous, query based, and hybrid. In event-based delivery, a sensor node does event reporting if and only if target events occur. In continuous delivery, in some cases, sensor nodes need to report to the gateway and generate continuous data transmission periodically. In query-based delivery, sensory data is stored inside a network and is queried by and then transmitted to the gateway on demand. Practical applications might trigger hybrid data delivery including event-based, continuous, and query-based. For example, an application would not only be interested in all temperature changes (continuous delivery) but also interested in some specific temperature change (e.g.: below zero degrees; event-based delivery) as well as querying temperature at a specific time (query-based delivery).

The differences between the WSNs and the Internet network QoS discussed earlier are illustrated in Table 2-2 below.

Table 2-2 Comparisons between the WSN and the Internet network QoS

	WSN Network-QoS		Internet QoS
	End-to-end QoS	**Reliability Assurance**	**End-to-end QoS**
QoS Target	- Network lifetime - Cost and energy efficiency - Timely response - Speed of packet delivery	Data Reliability: - Loss sensitive - Successful transmission of all packets Event Reliability: - Successful event detection (successful transmission of all packets not required)	- Provides services to bandwidth-hungry multimedia applications
QoS Parameter	- Delay - Throughput - Bandwidth	- Reliability - Throughput - Packet Loss - Delay	- Latency - Bandwidth - Delay - Jitter - Packet loss
Approach/ Technology	- Routing protocol - Service differentiation - Traffic priority	- Transport protocol - Service differentiation - Data aggregation - Congestion control - Data criticality	- Over-provisioning of resources - Traffic engineering (admission control, policy managers, traffic classes, queuing mechanism)
QoS Efforts	- SAR - SPEED - MMSPEED - Other PHY and MAC protocols for energy efficiency	- ESRT - ReInForm - Reliable information forwarding - Information assurance	- IntServ - DiffServ - MPLS

It is evident that the QoS mechanisms proposed for WSNs are different from traditional end-to-end QoS employed in the Internet. Nevertheless, in an environment of interconnected WSNs whereby the Internet plays an integral role as the backbone of the interconnections, network QoS shall be considered to achieve the QoS. This can be done by giving close attention to the related QoS factors related to the applications of interest, and the network requirements imposed by both the Internet and WSN. Hence, Section 2.3 presents an overview of service differentiation, which is a predominant approach in managing heterogeneous data traffic in WSNs.

2.2.3 Performance Measure in End-to-End Communication

The end-to-end data flows from a WSN to its users impose various transmission times in different communication layers. As mentioned earlier in Section 2.2.1, a WSN-IoT service model such as the Supple Service Model [16], which provides periodically collected sensorial or geographical information, can be either interactive if it is query-based, or non-interactive if the user subscription defines a semi-continuous flow of data at regular intervals. Therefore, the transmission time includes the communication between a sensor node to the gateway, gateway to the Internet router, and the Internet router to another WSN gateway. Apart from these transmission times, processing delays and queuing delays within gateway devices further augment the communication time.

Table 2-3 illustrates end-to-end communication in the network. The table depicts the steps involved in different layers of the network, i.e. ranging from requests initiated by a user, to communication between nodes which are constituents of different WSNs, until a response is received by the user.

Table 2-3 End-to-end communication in the WSN-IoT network

	Source and Destination			**Communication Layers**	**Description**
1	💻	→	🗄	Local user and Gateway	Local users initiate requests and gain responses to/from sensor nodes through WSN gateway
2	🗄	→	📡	Gateway to Sensor node	The gateway device acts as the sink that has a direct connection with the sensor nodes
3	📡	→	🗄	Sensor node to Gateway	Sensor node sends captured data to the gateway device
4	🗄	→	🔀	Gateway to Internet router	Gateway device passes data to an Internet router
5	🔀	→	🔀	Internet router to Internet router	Internet propagation based on no. of hops
6	🔀	→	💻	Internet router and Remote user	Remote users' communication via Internet routers

Several studies have demonstrated the network performance testing and QoS measurements of WSN-Internet integration. One of the main challenges concerning WSN-Internet interconnection is in providing access to each sensor node through the TCP/IP based network where the end-to-end communication time is commonly investigated.

In [36], Su and Almaharmeh proposed an integration module that can ensure QoS for different network applications. The QoS is provided by an integration controller that runs software modules and can reconfigure the QoS parameters on the network edge router. In this application-level gateway approach, the performance is evaluated in terms of inter-arrival time (the time between adjacent packets), packet delay, round-trip time (RTT) and cumulative distribution function of the RTT.

Similarly, the RTT measured in [37] is defined from the time the user issues a request (e.g. GET/mode_id/light) until the actual light value appears on the user's screen. The overhead included in this period is analysed as follows, starting from the point at which the request arrives at the gateway:

$$\mathrm{RTT} = 2 \text{ x } (t_{java} + t_{serial} + t_{air}) + t_{processing} \tag{2.1}$$

where t_{java} is the Java gateway overhead, t_{serial} is the gateway to the base station communication time via the serial interface, t_{air} refers to the base station to sensor/actuator over-the-air communication at 256Kbps, and $t_{processing}$ is the request service time by sensor/actuator. In other words, the latency or RTT is measured from the time the user issues a request to the sensor node until the actual response is received if the lengths of messages travelling to and from the sensors/actuators are equal.

Furthermore, by distinguishing sensor nodes as TCP/IP addressable units, the work in [38] aimed to reduce the node access time and increase data transfer efficiency by employing a translation table within the gateway architecture that can provide multiple node addresses for a single node. In this work, 'successful data transfer' of one message between two nodes (message transfer and response) depends on the shortest end-to-end delay. This kind of communication delay includes transmission delay, propagation delay, processing delay, and queuing delay.

The total communication time is defined as:

$$\mathrm{T}_{total} = 2(m+1)h\mathrm{T}_c + 2(m+h-1)\mathrm{T}_t + 2h\mathrm{T}_p + 2m(h-1)\mathrm{T}_q \tag{2.2}$$

where m is a number of transferred messages, and h corresponds to a number of hops.

T_t (transmission delay) – time for the transmission of one message. (It depends on the channel bandwidth, bit rate, message length, and coding techniques),

T_p (propagation delay) – signal propagation time between two sensor nodes,

T_c (processing delay) – the time needed for processing one message, and

T_q (queueing delay) – average time a message waits in queue for transmission.

It is evident that one of the main contributing factors of the end-to-end communication is the queuing delay within the gateway. The next sub-section explains the role of the gateway that acts as the IoT access point of the WSN-Internet integration.

2.2.4 QoS Over Heterogeneous Networks

In a gateway-based integration network, the QoS implementation is commonly provided on the gateway side of the WSN. Indeed, being in the unique position of having the full knowledge and control over both the WSN and the Internet, the gateway that acts as the IoT access point plays a vital role guaranteeing QoS for the integration.

As the QoS employed in WSN differs greatly from that of the Internet, interconnectivity issues between the two domains are inevitable. Hence, the QoS provisioning becomes increasingly important as the network is made up of heterogeneous components. The challenge for generic heterogeneous networks is to offer an end-to-end QoS guarantee transparently.

The overall problem of QoS interworking may be structured into two different actions; vertical QoS mapping and horizontal QoS mapping [39]. The concept of vertical QoS mapping [40] is based on the idea that a telecommunication network is composed of functional layers and that each single layer must have a role in end-to-end QoS provisions. The overall result depends on the QoS achieved at each layer of the network and is based on the functions performed at the layer interfaces. On the other hand, the concept of horizontal QoS mapping refers to the need to transfer QoS requirements among network portions that implement their technologies and protocols.

A cross-domain QoS that provides some integrated QoS mapping mechanism between both varying WSN QoS and the Internet QoS may be employed to address the end-to-end QoS. The WSN gateway links the QoS models employed in the WSN with the QoS employed by the Internet. A framework to address the cross-domain QoS problem is proposed in [41, 42]. The proposed framework is designed to facilitate a seamless QoS interaction between an ad-hoc network and an access network, i.e., the Internet. While the QoS solutions for the ad-hoc network are defined to solve specific problems such as mobility and the fading of wireless channels, the common QoS solutions such as DiffServ, on the access network are designed to address the issues of fixed structure networks.

Figure 2-2 depicts the cross-domain QoS interaction that can be addressed in the WSN-IoT application. As shown in the figure, the WSN implements a service differentiation QoS solution, whereas the Internet that acts as the access network implements a typical model like DiffServ or IntServ.

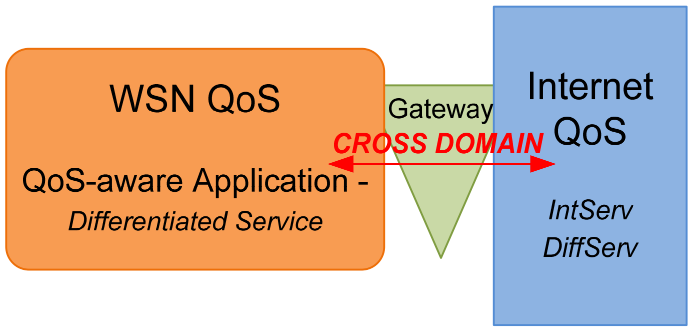


Figure 2-2 Cross-domain QoS mapping between different QoS mechanisms

Thus, a framework that runs on a QoS driven gateway may be employed to solve the interconnectivity issues between two different domains. The model may rely on the QoS models implemented on each side of the gateway to provide detailed services in the relevant domain while focusing on QoS concatenation issues. From the end-to-end viewpoint service quality can be measured in terms of four comprehensive parameters, namely end-to-end bandwidth, end-to-end delay, end-to-end jitter, and end-to-end loss rate.

2.3 Service Differentiation QoS in WSN

This section presents the related work on WSN QoS, which addresses the QoS requirements of the *timeliness* and *reliability* domains. It also discusses the approach of service differentiation in WSN through the priority-based scheduling approach established in previous studies.

2.3.1 Related Work on Timeliness and Reliability QoS

QoS solutions through service differentiation [43, 44] algorithms for WSN have been proposed in previous studies. Service differentiation has become a common approach to achieve the QoS for real-time WSN applications. Starting with one of the earliest works in differentiated service-based QoS in [45], subsequent efforts in this area of research have demonstrated this approach to QoS provisioning, specially designed to suit resource constraint WSN [24, 43, 46-51]. While the proposed mechanisms involve different aspects of service differentiation, namely, QoS-aware routing, priority-based scheduling, probabilistic QoS guarantee and MAC protocols, the works are based on the common nature of WSNs – the network is comprised of different data types, hence the demand for different levels of QoS from the network. However, like many other real-time QoS solutions in WSNs [30], the differentiated service strategy gives the primary attention to delay-sensitive [44] packets – the aim is mainly to cater for real-time packets that need to arrive at the sink in a required time frame, ensuring low latency and low delay. Nevertheless, there has been limited work [25, 43, 52] that addresses the varying QoS requirements of different traffic classes.

In contrast to real-time systems, a delay-tolerant WSN [53] is characterised by long-delay and intermittent connectivity. The main feature of the QoS provisions in delay-tolerant applications, for example, in sparse mobile sensor networks such as vehicular networks [23] and wildlife tracking networks [54], is reliable message delivery. In addition, the delay-tolerant network (DTN) concept [12], which makes use of store-and-forward techniques within the network, is employed to compensate for unstable connectivity. Research activities in this area are mainly focussed on routing protocols [25, 55-57] geared towards minimising delivery delay.

Studies of the domains of *timeliness* and *reliability* were demonstrated in other previous studies on WSN QoS. The work in [25, 52] is geared to address both *timeliness* and *reliability* QoS requirements. In [25], to route packets through a WSN with mixed priorities traffic, the real-time packets are allocated more bandwidth, whereas the delay tolerant data with reliability constraints are allocated more storage in the buffer within sensor nodes. The work is designed to represent a farm with tagged animal with sensor nodes. In this work, two types of packets are generated – one for the environmental data of the surrounding environments and the other for the health condition of the herds. The former is considered to be delay-tolerant data while the latter must be sent quickly, especially during emergencies; hence, it is assumed to be real-time data.

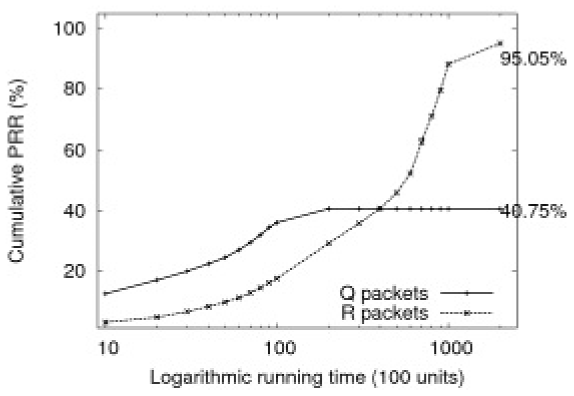


Figure 2-3 Comparison of delivery times between real-time and delay-tolerant packets [25]

Figure 2-3 shows the different ways that the real-time (Q packets) and delay-tolerant (R packets) are delivered to the base station in the service differentiation protocol in [25]. Points in the graph indicate the percentage of packets that are delivered to the station in a time bound scenario, for example, the first two points mean that around 12 percent of Q packets and three percent of R packets reach the base station in 1,000-time units. These results show that both Q and R packets achieve their QoS requirements. The Q packets arrive relatively quickly, typically before 4,738-time units. However, they are often lost before delivery, and the overall packet delivery ratio (PDR) is as low as 40.75%. R packets travel much more slowly to the base station, many of them not arriving until 10,000 or even 50,000-time units after being generated. However, these packets are delivered with much higher reliability, and the final PDR is as high as 95.05%. The total PDR is less than 100% because of extreme buffer capacity constraints, and so some R packets are evicted due to high storage pressure. With larger packet buffers, the reliability of both the R and the Q packets would increase.

The work in [1] investigates the performance of a network with the co-existence of heterogeneous data traffic. The aim of the study is to point out the pitfalls of integrating the WSN to the Internet without considering the QoS requirements of packet timeliness and reliability. Network traffic was simply categorised as high and low priority. For its resource allocation scheme, the network gateway implements some queuing discipline that governs how packets are buffered while waiting to be transmitted. Two typical scheduling schemes, i.e., priority queuing (PQ) and weighted fair queuing (WFQ) were employed to treat packets with high and low priority differently. In the PQ policy, all high priority packets are sent before any low priority packets. In the WFQ policy, one queue is maintained for each priority class, and weights are associated with the traffic classes based on their importance.

Figure 2-4 shows the amount of dropped traffic, due to buffer overflow, when typical PQ and WFQ scheduling is used within the gateway. The traffic dropped for both schemes occurs at almost a similar rate. This is due to the small weight difference between packet types. Since packets are categorised merely as high and low priority, the high priority queue shows a lower drop rate than the low priority queue. However, this should not be the case for reliability-constrained packets, mainly because these types of packets cannot tolerate or can only tolerate a small percentage of loss. Hence, they should have a high packet delivery percentage.

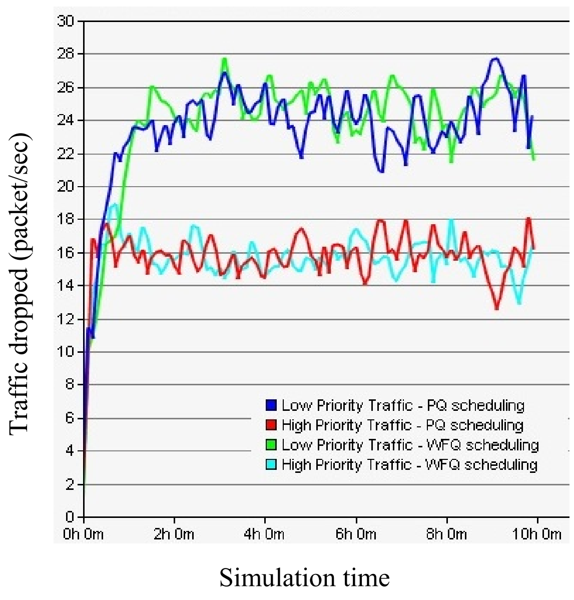


Figure 2-4 Traffic dropped for PQ and WFQ scheduling

2.3.2 Packet Scheduling in WSN Gateway

As mentioned in the previous Section, the network gateway implements a queuing discipline to control the resource allocation. Different priority traffic or traffic with different QoS requirements is placed in different queues in the network gateway. An ideal queuing approach could contribute to the success of the QoS solution. An overview of queuing disciplines and packet scheduling mechanisms in a WSN gateway is presented herein.

Classifying network traffic is the foundation for enabling the service differentiation in a network. This is particularly vital for wireless multimedia sensor networks (WMSN)

[22, 58] where different types of data traffic are organised into traffic classes based on their QoS requirements. Grouping network traffic based on user-defined criteria is a means of classifying the network traffic. Hence, the resulting groups of network traffic can be subjected to specific QoS treatments. The treatments include faster forwarding by network nodes for high priority real-time traffic or reducing traffic drop due to a lack of buffering resources for traffic that does not tolerate packet loss.

An illustration of heterogeneous traffic flows arriving at a gateway is shown in Figure 2-5. Different types of traffic sources, i.e., traffic source 1, 2, 3…. *n* are generated by the sensor nodes. Packets arrive at the gateway, which typically has a QoS function such as packet classification and access control to support the service differentiation. Traffic classification will enable placing the traffic into specific queues. Consequently, packets can be scheduled more efficiently.

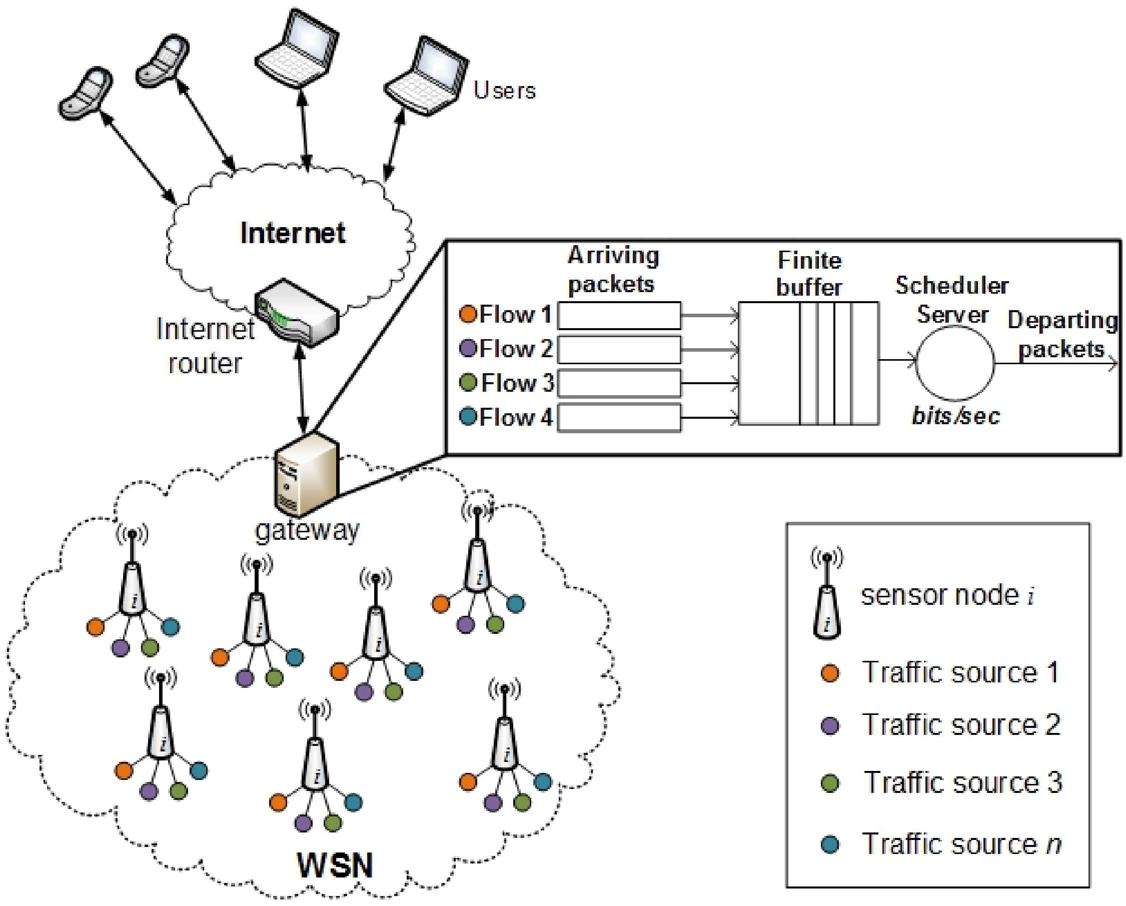


Figure 2-5 Packet scheduling in WSN gateway

As illustrated in the figure, the queuing model implemented in the gateway governs the way packets of different traffic classes are buffered while waiting to be transmitted. Once data arrives at the gateway, it will be allocated to buffer queues, depending on the scheduling algorithm implemented on the gateway. Various queuing disciplines can be used to control which packets are transmitted (using the bandwidth allocation scheme) and which packets are dropped (depending on buffer space allocation). The queuing discipline also affects the latency experienced by a packet by determining how long a packet waits to be transmitted. Examples of the common queuing disciplines are priority queuing (PQ), and weighted-fair-queuing (WFQ).

PQ is a simple variation of the basic First-in-First-out (FIFO) queuing. The idea of FIFO queuing is that the first packet that arrives at the gateway is the first packet transmitted. In FIFO, given that the amount of buffer space at each gateway is finite, if a packet arrives and the queue (buffer space) is full, then the router discards that packet. This occurs without regard to which flow the packet belongs to or how important the packet is. On the other hand, in PQ policy, all high priority packets are sent before any low priority packets. The low priority transmission will be pre-empted if any new, high priority packets arrive. Each packet is treated according to its marked priority, potentially using the IP Type of Service (ToS) field. The gateway then implements multiple FIFO queues, one for each priority class. Within each priority, packets are still managed in a FIFO manner.

The idea of the fair queuing (FQ) discipline is to maintain a separate queue for each flow currently handled by the gateway. The gateway then services these queues in a round-robin manner. On the other hand, in WFQ [59], weights are associated with the classes based on their importance. The WFQ scheduling discipline is an important method for providing bounded delay, bounded throughput and fairness among traffic flows [51]. The weights are assigned to each flow (queue). Hence, one queue is maintained for each priority class. This weight effectively controls the percentage of the link's bandwidth each flow will get. ToS bits can be used in the IP header to identify that weight. Queues are then serviced (i.e., packets are taken from the queues and sent on the outgoing line) at rates based on their weights. For instance, if the high priority queue was assigned a weight of '2', and the low priority queue was assigned a weight of '1', then two packets will be sent from the high priority queue for every one sent from the low priority queue.

WFQ is commonly referred as "bit-by-bit round robin," because it implements a queuing and scheduling mechanism in which the queue servicing is based on bits instead of packets. Weighted Round Robin (WRR) is a scheduling discipline that addresses the shortcomings of priority queuing and fair queuing. The basic concept of

WRR is that it handles the scheduling for classes that require different bandwidth. WRR accomplishes this by allowing several packets to be removed from a queue each time that queue receives a scheduling turn. WRR also addresses the issue with PQ in which one queue can starve queues that are not high-priority queues. WRR does this by allowing at least one packet to be removed from each queue containing packets in each scheduling turn.

The main difference between WFQ and WRR is that WFQ services bits at each scheduling turn, whereas WRR handles packets in each scheduling turn. The number of packets to be serviced in each scheduling turn is decided by the weight of the queue. The weight is usually a percentage of the interface bandwidth, thereby reflecting the service differences between the queues and the traffic classes assigned to those queues.

WRR has no knowledge of the true sizes of the packets in the buffers that are to be scheduled. The queues and scheduling are generally optimized for an average packet size. However, the sizes are all just estimates and have no true meaning with regard to the actual traffic mix in each queue. This operation of WRR is both an advantage and an issue. Because WRR has no complex resources that demand state computation as with WFQ, which must transform bits to bandwidth scheduling, it is fairly simple to implement WRR. The result is a solution well-suited for handling a large number of flows and sessions, making WRR into something of a core QoS solution that can deal with large volumes of traffic and with congestion. The drawback of WRR is that it is unaware of bandwidth because it does not handle variable-sized packet.

The main benefit of WFQ is that its implementations provide service differentiation between classes and their aggregated traffic, rather than merely differentiating between individual flows. A weight allocated to each class divides the scheduling and bandwidth ratio for each class. In addition, because WFQ is bits aware, it can handle packets of variable lengths. However the limitation of WFQ is that the original WFQ design is more of a queuing theory. The existing implementations do not follow the original concept in which each flow is allocated a weight. Instead, flows are aggregated by being classified into different service classes, and these classes are then assigned to queues.

2.3.3 Congestion Control

Network congestion occurs when offered traffic load exceeds the available capacity at any point in a network [60]. Congestion in WSN [61] causes overall channel quality to degrade and loss rate to rise, leading to buffer drops and increased delay. Provisioning a WSN so that congestion is a rare event is extremely difficult. Sensor networks deliver myriad types of traffic, from simple periodic reports to unpredictable bursts of messages triggered by the external events that are sensed. Even under a known, periodic traffic pattern, and a simple network topology, congestion occurs in the WSN. This is because radio channels often vary in time and concurrent data transmissions over different radio links interact with each other, causing channel quality to depend not just on noise but also on traffic densities. Moreover, the addition or removal of sensors or a change in the report rate can cause previously uncongested parts of the network to become under-provisioned and congested. Furthermore, when sensed events cause bursts of messages, congestion becomes even more likely.

Two types of congestion could occur in WSNs [62]. The first type is link-level congestion. For WSNs where wireless channels are shared by several nodes using Carrier Sense Multiple Access (CSMA) protocols, collisions could occur when multiple active sensor nodes try to seize the channel at the same time. Link-level congestion increases packet service time, and decreases both link utilisation and overall throughput, and wastes energy at the sensor nodes. The second type is node-level congestion, which is common in conventional networks. A node-level congestion is caused by buffer overflow in the node, which can result in packet loss and increased queuing delay. Packet loss will degrade reliability and application QoS, and waste the limited node energy and degrade link utilisation. When packet arrival rate exceeds the packet service rate, buffer overflow may occur. This is more likely to occur at the sensor nodes close to the gateway, or the gateway itself, which carries the combined upstream traffic. Upstream traffic could have high bit rate with the introduction and development of wireless multimedia sensor networks (WMSNs) [22]. Such high-speed upstream traffic is prone to cause congestion, which will impair QoS of multimedia applications in WMSNs.

Both node-level and link-level congestions have direct impact on energy efficiency and QoS. Therefore, congestion must be efficiently controlled. Congestion control protocol efficiency depends on how much it can achieve the following potential objectives [63]: The first objective pertains to energy efficiency for extending a system lifetime.

Congestion control protocols need to avoid or reduce packet loss due to buffer overflow and remain lower control overheads that consume less energy. Secondly, some fairness needs to be guaranteed so that each node can achieve fair throughput. Most of the existing work [60, 62] guarantees simple fairness in that every sensor node obtains the same throughput to the sink. In fact, sensor nodes might be either outfitted with different sensors or geographically deployed in different places, and therefore they may have different importance or priority and need to gain different throughput. In this perspective, weighted fairness is required. Thirdly, it is also necessary to support traditional QoS metrics such as packet loss ratio, packet delay, and throughput. For example, multimedia applications in WMSNs require not only packet loss guarantee (timeliness requirement) but also delay guarantee (reliability requirement). Congestion control methods involving heterogeneous data traffic have been studied in previous literature [28, 64, 65]. [28] proposed a weighted priority-based rate control scheme to control congestion by adjusting transmission rates relative to various data types.

The two general approaches to control congestion are through network resource management and traffic control [63]. The first approach tries to increase network resources to mitigate congestion when it occurs. In the wireless network, power control and multiple radio interfaces can be used to increase bandwidth and weaken congestion. With this approach, it is necessary to guarantee precise and exact network resource adjustment to avoid over-provided or under-provided resources. However, this is a hard task in wireless environments. Unlike the approaches based on network resource management, traffic control implies controlling congestion through adjusting traffic rates at source or intermediate nodes. This approach is helpful for saving network resources and more feasible and efficient when the exact adjustment of network resources becomes difficult. Most existing congestion control protocols belong to this type. According to the control behaviour, there are two general methods for traffic control in WSNs: end-to-end and hop-by-hop. The end-to-end control can impose exact rate adjustments at each source node and simplify the design at intermediate nodes; however, it results in slow response and relies highly on the round-trip time (RTT). In contrast, the hop-by-hop congestion control has a faster response. However, it is usually difficult to adjust the packet-forwarding rate at intermediate nodes mainly because the packet-forwarding rate is dependent on MAC protocols and could be variable.

A congestion control solution may consist of three important components: congestion detection, congestion notification, and rate adjustments [28]. Congestion detection in

WSNs often makes use of a congestion indicator, which has been proposed in terms of buffer occupancy, queue length [60, 66], packet service time [62], or the ratio of packet service time over packet inter-arrival time at the intermediary nodes [63]. Typically, when detecting congestion in the network, the transport protocol needs to propagate congestion notifications from the congested node to the upstream sensor nodes or the source nodes that contribute to the congestion. This can be done explicitly by sending a special control message to the other sensors or implicitly using the piggy-back technique in data packets. Hence, when a node receives the congestion notification message, it should adjust its transmission rate using a rate control technique.

2.3.4 Adaptive Service Differentiation

In a service differentiation approach, adaptive QoS may be used to maintain QoS relative to the dynamics of the sensor network. As mentioned in Section 2.3.1, sensor network dynamics may include the change of network parameters through a course of time, which may include the intensity of traffic flows due to bursts of traffic, the number of active sensor nodes and gateway devices, and bandwidth availability.

Related work in adaptive service differentiation is demonstrated in [28], by adopting congestion control [61] and rate adjustment solutions. The work shows that using a weighted priority-based rate control scheme; congestion can be controlled by adjusting transmission rates relative to various data types. Furthermore, an adaptive system for service differentiation through the fuzzy logic controller was proposed in [67]. The work introduced an extension from the service differentiation in [28] as it includes a fuzzy logic controller for traffic load parameters with priority-based rates in the network. It is reported in [67] that the adaptive QoS system supports prolongs the system lifetime as adjustments to the network can be performed to enhance system performance.

2.4 Envisioned QoS Framework

The QoS requirement and literature analyses facilitate the formulation of our envisioned QoS concept. This section discusses the components of the QoS framework.

2.4.1 Management of Timeliness and Reliability Traffic Requirements

The main drawback in the service differentiation QoS according to previous studies is that primary attention is predominantly given only to the real-time packet with low bound delay [28, 67], while the desired QoS of the delay-tolerant packet is not taken into consideration. In fact, it is a common notion that the timeliness constraints of real-time traffic are of greater concern than the reliability constraints of the delay-tolerant packet [58]. However, we argue that both QoS domains are equally vital [1]. Thus, both timeliness and reliability requirements need to be carefully considered in the adaptive QoS mechanism.

Most of the work in previous studies only supports two major types of traffic classes, i.e., real-time traffic and non-real-time traffic, hence only considers the QoS associated with timeliness requirements [28]. To improve the QoS for co-existing real-time and delay-tolerant traffic, the proposed model should provide service differentiation for traffic classes possessing *timeliness* and *reliability* constraints. In addition, the model should potentially take into account the timeliness and reliability requirements with multiple levels of tolerance and priorities. We aim to devise a service differentiation model that explicitly deals with different QoS requirements for different types of data by applying a prioritisation scheme among WSN traffic.

2.4.2 End-to-end Service Differentiation

The service differentiation mechanism discussed in the literature review was adopted in the WSN sensors level. However, further attention is required to enable the QoS for WSNs that is part of the IoT domain [17]. In this perspective, it is an interesting challenge to define a QoS mechanism that involves components beyond the scope of sensors - potentially the management of heterogeneous traffic from the gateway level which acts as the IoT access point.

As discussed in Section 2.3.3, due to the convergent nature of upstream traffic, congestions are more likely to appear in the upstream direction [63]. The upstream traffic from sensor nodes to the gateway is a many-to-one communication. Particularly, at the IoT access point side, congestion occurs due to the lower capacity of the access point's outgoing link when compared to incoming traffic. Furthermore, being in a unique position with full knowledge of and control over both the WSN and the Internet, the IoT access point plays a vital role guaranteeing QoS for the integration. The QoS requirements of different traffic types need to be carefully considered in the traffic management running within the network especially within the IoT access point. This will facilitate a seamless QoS interaction between both the WSN and the Internet. Furthermore, the network QoS shall be measured and analysed by giving close attention to the various QoS requirements imposed by both the Internet and WSN. Therefore, the application-specific QoS requirements of WSNs and the end-to-end QoS requirements of the Internet must be well distinguished.

Since the QoS mechanism employed in WSNs and the Internet differs greatly, the interoperability between both networks may also influence the network performance. A mechanism for an end-to-end service differentiation will be able to preserve the QoS implemented between different network layers. Due to distinguishable characteristics of WSN QoS and Internet QoS, a mechanism to communicate the varying QoS should be made available. A QoS mapping framework will facilitate a seamless QoS interaction between both networks built over heterogeneous components. The motivation lies in having seamless QoS interaction between these two networks. The IoT access point needs to preserve the WSN QoS employed. If the ordinary access point is employed, an Internet QoS such as DiffServ will potentially be used. The proposed QoS concept is to make sure that service differentiation is preserved when integrating WSNs to the Internet.

2.4.3 Adapting to Traffic Dynamics

The buffers required for QoS traffic may suffer the same issue of scarcity as other WSN network resources. Not having adequate buffer sizes would complicate traffic classification, introduce delays, and reduce the possibility of granting QoS guarantees. Therefore, due to limited resources and to ensure optimum resource utilisation (in terms of buffer usage and bandwidth allocation), a congestion control algorithm and efficient resource allocation will also be major components of the QoS framework. The congestion control algorithm will adapt to the event of burst traffic when the accumulation of packets in the buffer becomes more rapid. This imposes the requirements of adaptivity to the QoS framework.

The QoS framework should comprise of an adaptive QoS mechanism that is capable of reacting to dynamic changes in the network to ensure seamless QoS interactions between the WSN and the Internet. Specifically, the service differentiation scheme may be featured with adaptive capabilities as it reacts to traffic dynamics. Furthermore, it is also important to gain real-time and continuous assessment of the current QoS conditions to ensure the required application-specific QoS is always met. With knowledge of QoS performance such as queuing delay and traffic drop, informed adaptations can be made to the network. This can potentially be done through reconfigurations of node attributes such as buffer size, service rates, and bandwidth allocations.

2.5 Summary

In this chapter, a QoS requirement analysis and a literature review pertaining to WSN-IoT integration are presented. A literature analysis of WSN with heterogeneous data traffic and service differentiation of QoS to accommodate traffic dynamics has been presented. It is evident that there is a glaring lack of research in the area of end-to-end QoS support, particularly as a means of ensuring the preservation of WSN QoS, especially concerning the *timeliness* and *reliability* of QoS imposed by heterogeneous data traffic.

The literature analysis also focused on WSNs' ability to adapt and react to the dynamic changes of the network, potentially via a resource control mechanism. In this perspective, the capacity of the network including the resources within the network gateway needs to be considered carefully. Since upstream packets are queued at the gateway, which also acts as the IoT access point, it has complete knowledge about them. As a consequence of the unique position of full knowledge and control over both the WSN and the Internet, the IoT access point plays a vital role guaranteeing QoS for the integration.

Therefore, an integrated QoS framework is envisioned, encompassing an IoT access point that runs a QoS mechanism that links the network-level QoS mechanism from both WSN and the Internet. While most of the differentiated services in previous literature has operated at the sensor node level, the envisioned QoS framework will focus on enabling QoS in the domain of WSN-IoT. The QoS framework will have the following components:

i) A QoS model that explicitly deals with different QoS requirements for different types of data by applying a prioritisation scheme among WSN traffic

ii) An adaptive QoS mechanism that is capable of reacting to dynamic changes of the network to ensure seamless QoS interactions between the WSN and the Internet

Firstly, the QoS framework should be able to address the varying QoS requirements imposed by heterogeneous data traffic and their association to *timeliness* and *reliability* domains. While it is typical that timeliness is of greater concern than reliability, we argue that both QoS domains are equally vital. It is imperative to consider both domains in an application, especially with the emergence of more complex sensor network applications that may need some support for multiple traffic types. Secondly, the service differentiation scheme features adaptive capabilities to react to application-specific traffic dynamics within the network.

Chapter 3 Modelling and Implementation Tools

3.1 Introduction

This approach to studying the QoS of WSNs was selected by considering factors related to the research aim of assessing the integrated influences of WSNs and the Internet on the QoS. In this research, it is of vital importance that the selection of tools and methods are based on investigating the envisioned QoS framework. As discussed at the end of Chapter 2, this research is inspired by two major goals. Firstly, to provide a solution for QoS provisioning that satisfies the QoS requirements for the mixed traffic nature of WSNs connected to the Internet. Secondly, to devise a scheme for validating and verifying network performance under the modelled QoS.

Generally, the approach for investigating networking protocols and evaluating network performance may fall into 1) Analysis and mathematical modelling, 2) Simulation – typically time-based simulation or discrete event-based simulation (DES), 3) Hybrid simulation, i.e., simulation using both mathematical modelling and simulation, and 4) experimentation using a locally established testbed. Modelling and simulation are means of verifying the working and measuring the effectiveness of different techniques proposed for WSNs. As a representation, analytical modelling provides quick insights about the ideal techniques developed for WSNs. Simulations provide a good approximation to verify the different schemes and applications developed for WSNs at low cost and in less time. They cannot offer real results because of the imprecise representations of WSN-specific constraints such as limited energy and the sheer number of sensor nodes. On the other hand, real-world implementation and testbeds offer more accurate data to verify the concepts, but they are restricted by size, costs, effort and time factors. Repeating environmental conditions are also challenging for a physical WSN testbed.

One of the main objectives of this research is to design a QoS model based on service differentiation and congestion control. The literature investigates congestion control through traffic control and resource control protocols' performance using simulations, experimentation and by modelling their behaviour [68]. The congestion control protocol is evaluated to identify its efficiency in the presence of overload traffic. The pre-dominant metrics used by congestion control protocols are packet drop, packet delivery ratio, end-to-end delay, throughput and queue length. The choice of models and the simulation environment is important for credible results through simulation. The selection of the simulation tool will also be based on the capabilities of the available simulation tools.

This chapter begins with an overview of the currently available simulation tools, along with a review of related work in WSN QoS studies. An overview of related work involving interactions between network modelling tools and computation tools will be discussed next, drawing attention to the complexity of the investigation. Drawing from the analysis, selected tools and approaches to designing and evaluating the integrated QoS model are presented. Important features of the selected tools are highlighted to reflect the research activities. Finally, some state-of-the-art WSN testbeds are presented, followed by an overview of the testbed architecture used for QoS model validation.

3.2 Related work on QoS in WSNs and the Internet

The study of WSNs and QoS have mostly been performed using simulators, particularly in the domain of service differentiation and priority-based schemes. Simulators are normally chosen as they do not require hardware for testing purposes, which means that thousands of nodes can be simulated. Since the simulation depends on machine processing capabilities, complex simulations are made possible with high-performance computers. Tools that can be used to study WSNs include J-Sim, NS-2, OMNeT++, GlomoSim, Riverbed Modeler (formerly known as OPNET Modeler), SENSE, Ptolemy II and VisualSense.

Typically, these simulation tools support the network simulation steps which have been discussed by S.A. Madani [69]. The spiral cycle approach with an incremental development process for WSN modelling and simulation includes: 1) Conceptual model, 2) Collection and analysis of input/output data, 3) Modelling, 4) Simulation, 5) Verification and validation, 6) Experimentation and 7) Output analysis. Transitions to the opposite direction can appear, and some steps can be skipped, depending on complexity. One important criterion is the tool’s capabilities to facilitate the measurement of QoS metrics. In addition, the domain of node and network modelling, heterogeneous traffic modelling, queue management and QoS attributes will be highlighted.

J-Sim has an autonomous component architecture-based simulation environment written in Java. The simulator has components as basic entities that are assembled to design nodes and scenarios. It offers support to languages such as Perl, Tcl (Tool command language) or Phyton. J-Sim [70] was used by Martinez et al. [26] to study the QoS related to an unbalanced mixture of traffic in forest surveillance application scenarios. By considering two traffic priorities, namely reliability and timeliness, the authors used J-Sim to compare three QoS routing protocols for WSN as the candidates for their forest surveillance network model. The study of MAC and network layer protocols defined to provide QoS in WSNs has been presented to various QoS routing protocols. Apart from predominant parameters such as the number of sensor nodes, bandwidth and radio range, the simulation environment setting included terrain size and morphology. In this work, J-Sim is primarily chosen due to its ability to model the node's deployment around a mountain distributed across four sectors - north, south, west and east. Another reason is due to the simulator's component-based feature, which enables users to modify or improve it.

NS-2 [71] is a discrete event simulator specifically designed for network research. It uses C++ for protocol designing and Tcl for scripting the interconnections in a scenario that includes detailed scripting. Its focus is the IP network. In the domain of QoS via congestion control protocol [68], the researchers in [72-74] employed NS-2 for their work on resource control protocol. The performance parameters obtained included dropped packets, power consumption [72], packet loss rate [73], throughput and packet delivery ratio [74]. To simulate WSNs with more or less 100 nodes, NS-2 can be a good choice because of its large community, but it is not scalable for 100+ nodes [75]. One of the disadvantages of ns-2 is its object-oriented design, which imposes unnecessary interdependence between modules. Such interdependence makes the addition of new protocol models extremely difficult as they can only be mastered by those who have intimate familiarity with the simulator. Another drawback is that it does not have a native graphical editor for scenario deployment, which is very important for WSN study.

A good graphical editor helps in the visualisation of deployment and facilitates researchers to maintain focus on the core idea and its performance analysis, rather than the coding and implementation of network deployment scenarios. Several WSN simulators has been identified with a good graphical editor. Some of the simulators are Global Mobile Information System Simulator (GlomoSim), OMNET++ and Riverbed Modeler.

GlomoSim [76] is a library based general purpose parallel simulator which can simulate up to 10,000 nodes [77] and can be very useful in studying large-scale WSNs. GlomoSim is superseded by QualNet, a commercial network simulator. sQualNet [78], an evaluation framework for sensor networks based on QualNet was released later. The QoS model SPEED [49] and interference-minimised multipath routing (I2MR) protocol [79] with congestion control in WSNs were simulated using GlomoSim. The evaluation parameter includes control packet overhead, throughput and energy consumption.

OMNeT ++ [80] is a discrete event, component-based, general purpose, public source modular simulation framework written in C++. It provides a strong GUI support for animation and debugging. The lack of a WSN-specific module library [81] may be a problem, but many research groups have been working to add WSN specific additional modules. SenSim [82] is an OMNeT-based simulation framework for WSNs. It provides for the basic implementation of different hardware (e.g., basic radio and CPU) and software (simple routing schemes) modules for WSNs. It provides a template with basic implementation or empty body, which can help anyone to jumpstart simulating WSNs. A QoS study for a priority-based scheme for delay-sensitive data transmission over WSN by Safaei et al. [44] was conducted in OMNeT. A priority-aware congestion control mechanism and a queue model to support bursty data were created using the software tools. The QoS performance was demonstrated regarding packet delay, packet drop and throughput.

Riverbed Modeler (formerly known as OPNET Modeler) is a network simulator tool. C.Wang et al. [65] used the OPNET Modeler to simulate a QoS model called the priority-based congestion control protocol (PCCP). In Riverbed Modeler, the packet inter-arrival time and packet service time was manipulated to produce a measure of congestion. The main performance parameter, namely queue length and node/system throughput were retrieved using Riverbed Modeler graphing tools. Another advantage of the Riverbed Modeler is the ability to implement queue management +[83]. Riverbed Modeler is used to conceive, develop and test new schemes, models and algorithms for improving the performance of queue management. Active queue management schemes and algorithms can be developed and deployed within the tool.

3.3 Selected Tools

Among the discussed network simulators, NS2, OMNET++ and Riverbed Modeler are the preferred simulators. From the above analysis, the simulation tools chosen for this work are Riverbed Modeler and MATLAB. Network scenarios and QoS attributes are implemented in Riverbed Modeler, while further data analysis will be done using MATLAB.

Riverbed Modeler is selected as it comes with the Internet components that are needed for the architecture studied. Another reason is its queue management capability. Riverbed Modeler allows the creation of a custom queue model, which is one of the primary contributions of the research. Riverbed Modeler can be used to conceive, develop, and test new schemes, models and algorithms for improving the performance of queue management required in this study.

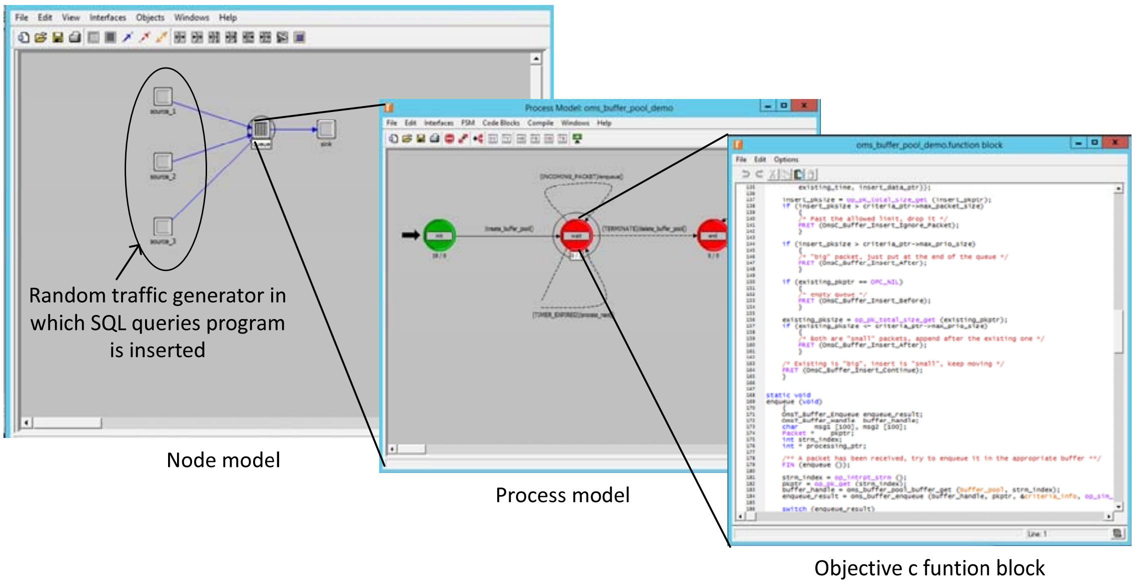
In service differentiation, adaptive QoS can be achieved through various approaches. Tools are needed to help to monitor the QoS performance continuously. Therefore, the simulation data from Riverbed Modeler needs to be analysed in an analytical tool such as MATLAB to support the adaptive QoS. The features of the selected tools used to implement the QoS concept are presented in the next section.

3.3.1 Features of Riverbed Modeler

Riverbed Modeler models the network in different layers. '*Network model*', '*Node model*' and '*Process model*' are used to specify the network and nodes and to define the QoS concept respectively. The compartmentalisation of different models allows models to be easily reused and duplicated.

The QoS model, which employs a queuing discipline on the IoT access point, is designed using Riverbed Modeler. The service differentiation QoS model governs the way that heterogeneous data packets are buffered while waiting for transmission. Once data arrives at the IoT access point, it will be allocated to buffer queues depending on the scheduling algorithm implemented on the coordinator. This model can be designed and simulated in Riverbed Modeler. Furthermore, the QoS performance of the heterogeneous data traffic of a sensor network is computed in the network simulation tool.

Figure 3-1 shows the Riverbed Modeler environment in which the queue model is developed.


File Edit View Interfaces Objects Windows Help
source_1
source_2
source_3
queue
sink
Random traffic generator in which SQL queries program is inserted
Node model
Process Model: oms_buffer_pool_demo
File Edit Interfaces FSM Code Blocks Compile Windows Help
Process model
oms_buffer_pool_demo.function block
File Edit Options
Line: 1
Objective c funtion block

```
            existing_time, insert_data_ptr));

        insert_pksize = op_pk_total_size_get (insert_pkptr);
        if (insert_pksize > criteria_ptr->max_packet_size)
            {
            /* Past the allowed limit, drop it */
            FRET (OmsC_Buffer_Insert_Ignore_Packet);
            }

        if (insert_pksize > criteria_ptr->max_prio_size)
            {
            /* "big" packet, just put at the end of the queue */
            FRET (OmsC_Buffer_Insert_After);
            }

        if (existing_pkptr == OPC_NIL)
            {
            /* empty queue */
            FRET (OmsC_Buffer_Insert_Before);
            }

        existing_pksize = op_pk_total_size_get (existing_pkptr);
        if (existing_pksize <= criteria_ptr->max_prio_size)
            {
            /* Both are "small" packets, append after the existing one */
            FRET (OmsC_Buffer_Insert_After);
            }

        /* Existing is "big", insert is "small", keep moving */
        FRET (OmsC_Buffer_Insert_Continue);
        }


    static void
    enqueue (void)
        {
        OmsT_Buffer_Enqueue enqueue_result;
        OmsT_Buffer_Handle  buffer_handle;
        char     msg1 [100], msg2 [100];
        Packet *     pkptr;
        int strm_index;
        int * processing_ptr;

        /** A packet has been received, try to enqueue it in the appropriate buffer **/
        FIN (enqueue ());

        strm_index = op_intrpt_strm ();
        pkptr = op_pk_get (strm_index);
        buffer_handle = oms_buffer_pool_buffer_get (buffer_pool, strm_index);
        enqueue_result = oms_buffer_enqueue (buffer_handle, pkptr, &criteria_info, op_sim_

        switch (enqueue_result)
```

Figure 3-1 Queue model design in Riverbed Modeler environment

The figure shows the hierarchical structure of Riverbed Modeler node models. At the lowest level, the behaviour of an algorithm or a protocol is encoded by a state/transition diagram, called state machine. This includes embedded code based on C-type language constructs. At the middle level, discrete functions such as buffering, processing, transmitting and receiving data packets are performed by separate objects. Some of these objects rely on underlying process models. In Riverbed Modeler these objects are called modules and they are created, modified, and edited in the Node Editor. Modules are connected to form a higher-level node model. At the highest level, node objects are deployed and connected by links to form a network model. The network model defines the purpose of the simulation. The design and development of protocols require the creation of a node model that is simulated as scenarios in the network model. The lower-level objects for the queue are provided by Riverbed Modeler, but they need to be combined to form a node model.

Networks for different applications can be set up using Riverbed Modeler. ‘*Application Config’* and ‘*Profile Config’* [84] can be configured in Riverbed Modeler to represent the application associated with the network. In the simulation, a traffic generator is simulated to represent steady traffic flows in one-hop transmitting data directly to the IoT access point. Therefore, multiple classes of traffic can be generated to simulate the co-existence of real-time and delay-tolerant traffic required to conduct this research.

Riverbed Modeler has a large database of nodes for different hardware and protocols that can be selected from the ‘Object Palette Tree.’ The WLAN nodes available within

Riverbed Modeler allow complete access to all the models, i.e., the node model, process model and objective C-code. Riverbed Modeler also comes with Internet and cloud components making it ideal for implementing the proposed QoS model and then running it on the IoT-WSN scenario.

3.3.2 Node and Queue Model Implementation in Riverbed Modeler

The Node Editor within Riverbed Modeler allows users to create and edit modules for the node model. The modules include a processor module, queue module, transceiver module, antenna module and an external system module. These modules can be connected by packet streams and statistic streams. The queue module is used to model a buffer. The node is configured to switch data packets at a predefined rate. Incoming data packets will first be pushed into the buffer. Data packets stored in the buffer will be sent out or serviced at another predefined rate. In the research, we need to investigate the way the buffer queues grow to ensure optimum resource consumption. If the incoming packet rate is greater than the service rate, then the transitional size of data stored in the buffer will grow until the incoming packet rate is reduced.

3.3.3 Co-Simulation between Riverbed Modeler and MATLAB to Support the Adaptive QoS

The network simulation tools discussed in the previous topic are ideal for developing node and network model solutions. A simulator such as Riverbed Modeler models the WSN network satisfactorily in most cases. However, it is inherently an event-driven simulator, which leads to compromises in simulating an environment with periodic and continuous monitoring of the QoS condition. Moreover, network attributes such as buffer size and transmission rates in the simulator editor are limited to a one-off setting; and the attribute cannot by default be changed from its initial value during a simulation run. This is worth improving, especially considering that research in WSN QoS is shifting more and more weight on the adaptive QoS model.

One of the features of the QoS model is to make it adaptive to the dynamic changes of network traffic. For this purpose, continuous and real-time analysis of the QoS condition is needed. In addition, the associated QoS performance computed by Riverbed Modeler needs to be analysed, and the results need to be used to perform reconfigurations of the traffic attributes to ensure QoS is maintained. While Riverbed Modeler is an ideal option to simulate the network, and collect performance statistics, analytical tools such as MATLAB can be used for the QoS analysis.

A mathematical analysis model within a tool such as MATLAB can use the performance data for analysing the QoS condition and identifying the corresponding adaptive QoS parameters that need to be altered. For this purpose, a co-simulation between a network modeler and a mathematical tool can be set up using external interfaces or API references. MATLAB will use the performance data to analyse the QoS condition and identify the corresponding adaptive QoS parameters. An adaptive QoS parameter represents the necessary adjustment made to network configurations to meet the QoS requirements of the application. For this purpose, co-simulation between Riverbed Modeler and MATLAB can be established using MATLAB MX functions and Riverbed Modeler APIs [85, 86].

The co-simulation between a network simulation tool such as Riverbed Modeler and MATLAB is greatly beneficial. While users may benefit from Riverbed Modeler as a tool that strives for closer representation to real network devices, the control mechanisms and decisions to support adaptive QoS are happening in MATLAB. MATLAB also supports high-level analysis, which enables future network complexity to become more manageable.

3.3.4 MATLAB Neural Network Tools for QoS Model Validation

The performance of the service differentiation can be further improved with a more powerful self-adaptation mechanism. This can be done potentially by adopting continuous learning and prediction of network parameters via the neural network [87]. The continuous learning achieved from neural network serves as an added-value to the proposed QoS model. For this purpose, the neural network tool in MATLAB is selected. Continuous learning may also constantly create awareness of network conditions. Hence, adaptation will be based on the QoS parameters resulting from a traffic dimensions such as traffic distributions and network load.

The system would be designed so that as new scenarios are experienced by the network the event neural network would be updated accordingly and the information could be used to predict future QoS parameters as the system evolves. This would lead to the creation of an unsupervised learning tool for maintaining the network QoS. It is envisioned that the approach will overcome the repetition of adjustment calculations for the adaptive QoS, contributing improvement to the system’s latency.

3.4 Physical WSN-IoT Testbed

One of the objectives of this research is to validate the QoS model under a WSN-IoT environment. As explained in the chapter introduction, physical testbeds offer the most accurate method of verifying WSN concepts [88]. Therefore, a real-setting of physical infrastructure, i.e., the WSN-IoT test environment has been set up. The test environment serves as an avenue for the validation and verification of the modelled QoS.

As most WSN deployments involve the placement of hundreds or thousands of nodes, it may be impractical to test and evaluate the sensor network on real WSN deployments due to complexity, cost and time consumption. Nevertheless, it is apparent that research activities in coordination and communication among sensors within WSN are of great importance. Thus, one of the main attentions of the present WSN research community is the development of WSN testbeds – established to provide avenues for experimentations. WSN testbeds have been developed to support the experimental research of WSN, equipped with the real world setting of sensor nodes in a controlled environment. These testbeds are shielded from hazards or external, uncontrollable factors, thus allowing optimum focus on the observation of a WSN’s behaviour. Typically, the WSN testbeds are integrated to the Internet to allow remote access to its users. Furthermore, these testbeds allow job submissions through web-interface, sensor reading visualisation, the remote programming of nodes and command line tools to control testbed nodes.

Physical WSN-IoT testbeds [89] offer great benefits to researchers when evaluating WSN design as they gather real-life data from physical settings. However, when testing new algorithms and protocols, the process of experimenting with different scenarios and performing comparisons amongst these scenarios can be quite challenging. In this case, simulators offer capabilities and features that make them favourable for the design and testing of new protocols. It is envisaged that a better analysis of WSN applications can be facilitated by exploiting the advantages of both virtualisation and real-life testbeds. Therefore, we propose an IoT-based WSN test environment that offers interactions between the behaviour of the physical environment as it interacts with the phenomenon and necessary analysis in a virtual remote environment.

An overview of a test environment architecture is presented in [4]. The test environment has been established by the Sensor Network and Smart Environment Research Centre (SeNSe) [90] laboratory research team. The effort of the SeNSe team aims to enable the implementation of several use-cases on the same testbed. The primary idea is to allow for a reconfiguration of the physical sensor nodes flexibly to adapt to the network’s QoS condition. The reconfiguration is based on the QoS evaluation by the software that resides on the server and analyses recent historical data. The architecture allows user-driven QoS-related experimentation and works on a case-by-case basis. As part of our continued investigation, we are looking at the applicability of the models by testing with application-specific WSN QoS parameters. Therefore, in this paper, the implementation of a proposed adaptive QoS model that serves as a use case on the test environment is presented.

The implementation of a virtual sensor network (VSN) as a software replicated image of the corresponding physical sensor network (PSN) is demonstrated by Barbato et al. [91]. The VSN contains all the metadata of the PSN. Data processing takes place within the virtualised network to cater for the different requirements of the users/clients. This covers the need for any data storage, processing or computation on PSN hardware. The large-scale IoT testbed implementation of SmartSantander [92] reaps the benefits of virtualisation hosted within a cloud infrastructure and provides experimentation and testing facilities to end users. Therefore, an extended function of specialised network modelling tools, such as Riverbed Modeler, within the cloud may facilitate an environment for hosting the QoS controller models on the virtualised network. It serves as a testing environment, i.e. to observe the impact of modifying virtual sensor node parameters such as data rate, service rate, buffer size and other functionalities of the network protocol on network performance before implementation on PSN hardware.

The virtualisation level within the Lysis platform [93] comprises of abstractions of the functionalities of the real world objects/devices such as smartphones, as well as their social capability. The virtualisation layer present within the iCore [94] architecture consists of virtual objects and facilitates the exposure of a virtual interface from the real object (either sensor or actuator). The first approach considerably reduces latency, whereas the second approach results in the decoupling of real objects, thus allowing for reuse of the virtual interface hardware implementation undergoing alteration.

A continuous QoS monitoring calls for a more involved IoT-based sensor network infrastructure that enables the virtualisation of the real physical world. With a test environment infrastructure, the adaptive QoS algorithm may be extended to allow informed adjustments of the IoT access point, which is highly desirable to ensure seamless interconnection with the Internet and to provide better balance to adaptive QoS strategies. With a software-based network modelling that provides virtualisation of the real physical world, important performance parameters related to the application of QoS requirements can be considered without having to make major changes to the physical setting.

3.5 Summary

The envisioned QoS model involves management of heterogeneous data traffic through queue management, which impacts on the performance of traffic types. Therefore, this thesis necessitates using a network simulator tool that supports the allocation of priorities to different traffic types, queue management and simulation of the Internet network. This chapter has given an overview of commonly-used and available simulation tools including OMNET, NS-2 and Riverbed Modeler. Riverbed Modeler has a complete graphical user interface and a hierarchical design methodology which are convenient when designing and debugging the network model. Riverbed Modeler also comes with an analysis tool that facilitates the statistics collection and investigations of pre-dominant QoS metrics such as packet delay, packet drop and throughput. One of the features of the QoS model is to make it adaptive to the dynamic changes of network traffic. For this purpose, continuous and real-time analysis of the QoS condition is needed. Co-simulation between Riverbed Modeler and MATLAB is chosen to perform simulations of the network and to collect the performance statistics, as well as continuous QoS analysis. The neural network tool in MATLAB is also chosen to introduce a value-added validation of the QoS model through continuous learning and the prediction of network adjustment parameters to maintain network QoS conditions.

Chapter 4 Integrating WSN to the Internet: ADHERE QoS Concept for Heterogeneous Data Traffic

4.1 Introduction

This chapter presents the concept of **A**daptive Service **D**ifferentiation for **He**te**r**og**e**neous Data in WSN (ADHERE) QoS framework. ADHERE is proposed based on the identified QoS factors and requirement analysis presented in Chapter 2. The aim of the QoS framework is to achieve an adaptive service differentiation in integrating WSN with mixed traffic requirements with the Internet. ADHERE QoS preserves the QoS mechanism of both WSN and Internet through seamless service differentiation on the network's IoT access point.

The objective of the QoS framework is to meet the requirements of heterogeneous data traffic in the WSN both in the domain of *timeliness* and *reliability*. Another objective is to implement a scheme of an adaptive QoS that is capable of reacting to dynamic network changes.

This chapter is organised as follows: The next section provides an overview of the QoS framework, along with the presentation of the network model. The sub-components of the ADHERE QoS model, namely the queuing model, congestion control unit, and rate adjustment scheme are presented in section 4.3. Then, an overview of neural network learning tools for the proposed ADHERE QoS concept is presented in section 4.4. Finally, section 4.5 concludes the chapter.

4.2 Overview of ADHERE QoS Framework

The proposed ADHERE QoS framework combines a service differentiation model with an adaptive scheme. The primary QoS target is to ensure that real-time traffic with *timeliness* requirements arrives to the end users with low delay, while delay-tolerant traffic with *reliability* requirements achieves a high throughput and high packet delivery ratio.

4.2.1 Network Model

It is assumed that the application running over the heterogeneous traffic data represents the Supple Service Model [17], as discussed in Chapter 2. In the Supple Service Model, sensor nodes generate a continuous flow of data at regular intervals, forming a many-to-one line of convergent traffic in the upstream direction. Figure 4-1 shows the network model with heterogeneous traffic classes received at the network IoT access point.

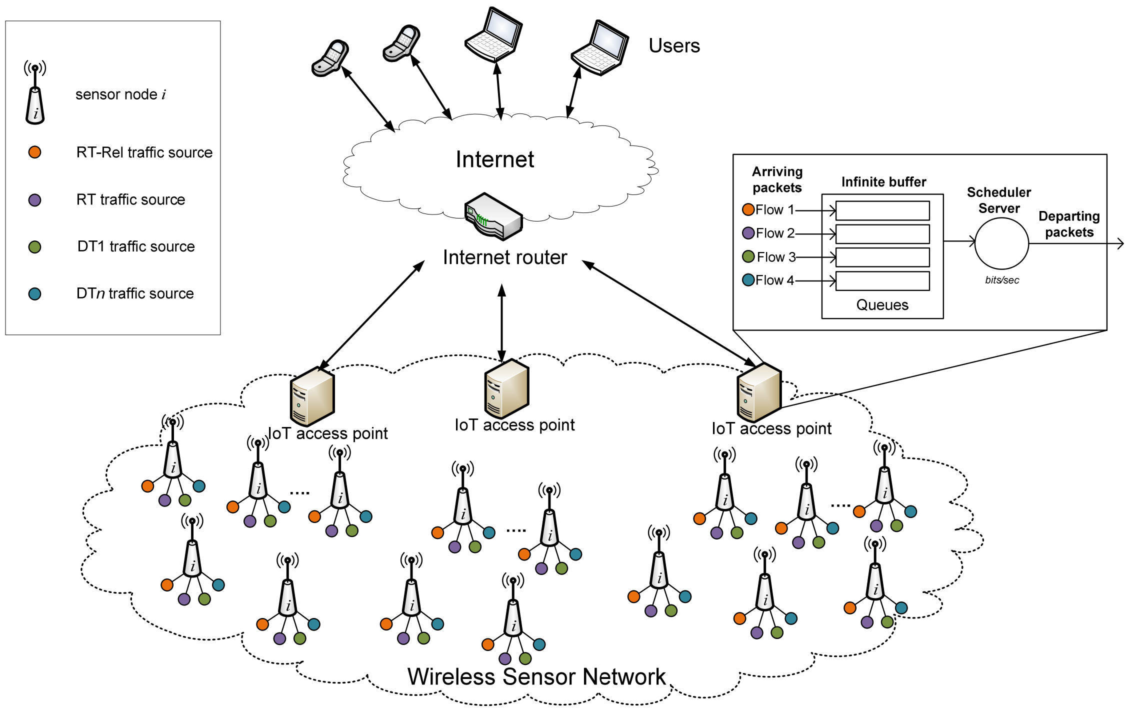


Figure 4-1 Network Model with heterogeneous traffic classes

We assume that the sensor nodes have a two-way communication with the IoT access point. However, no sensor-to-sensor communication is assumed. Each node generates packets of different types, depicted by different colours in the figure. The IoT access point, which acts as the IoT access point to the sensor network, relays the received packet to users through the Internet. The IoT access point may also act as a cluster head for a particular physical space, hence the assumption of a multiple IoT access point network. The figure shows the incoming traffic flows. Once they arrive at the IoT access point, they are allocated to buffer queues. Within the IoT access point, the weighted fair queuing (WFQ) model is employed. The WFQ scheduling discipline is an ideal method for providing bounded delay, bounded throughput and fairness among traffic flows [51, 95].

As shown in Figure 4-1, we consider the WSN network from the point of view of distributed clusters network. Each of the distributed clusters would be IoT-driven. This means, each sensor network has a cluster head that become the IoT connectivity to the Internet. Taking this nature of the sensor network into consideration, the architecture of the IoT-based WSNs adapted in this research is a single hopping network.

4.2.2 Queuing System

Figure 4-1 shown previously also illustrates a queuing model for regulating the buffer for data packets queued for transmission at the IoT access point. Upon arrival at the IoT access point, the data will be allocated to different buffer queues based on service differentiation, which runs from the IoT access point. As shown in Figure 4-1, different traffic types are buffered in separate queues in the IoT coordinator.

To discriminate traffic classes from each other, we assume that each sensor node adds a traffic class identifier to its local sensor packets and puts them in proper queues. This identifier represents the traffic class of each packet. A buffer is allocated for each traffic class; hence, increasing the number of traffic classes imposes a greater number of required buffers as well as hardware requirements [28].

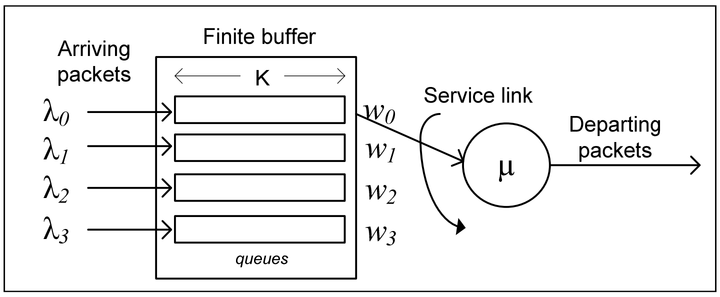


Figure 4-2 WFQ queuing system

Figure 4-2 illustrates a WFQ queuing system. Each traffic class is served on a fixed weight assigned to the related queue. The weight is determined according to the traffic's QoS requirement, such as its delay deadline. For instance, let's consider a four classes job system with a finite buffer with size K [59]. This means, the maximum number of packets that can be in the system at any time is K, and any additional packet are refused entry to the system and will depart immediately without service. Packets of class 1 to class 4 arrive with rate λ_i (where $i = 0,1,2,3$) and require exponential service times with a mean $1/\mu_i$. The queue i is served at rate w_i for some $w_i > 0$. The coefficient w_i is such that $w_0 + w_1 + w_2 + w_3 = 1$ [51, 59].

Each queue of finite size serves packets, and the inter-service arrivals are exponentially distributed with an average service rate, which is synonymous with the capacity of the outgoing link to the IoT access point. This assumption allows us to apply M/M/1 queuing theory to analyse the WFQ scheduling [59]. An M/M/1 queue consists of a First-in-First-out (FIFO) buffer queue with each packet arriving according to the Poisson arrival process, and a processor that retrieves packets from the buffer at a specified service rate. The three main parameters that affect the performance of an M/M/1 queue are packet arrival rate, packet size, and the service capacity.

The M/M/1 model is used as an approximation of delay, which is the key application-specific QoS parameter for delay-tolerant data. Despite being an infinite buffer model, M/M/1 is deemed to be appropriate model as its adoption for the purpose of delay bound approximation only takes place during the initialization phase. Furthermore, it is appropriate due to the nature of lightly-loaded queue coming from the delay-tolerant traffic.

4.2.3 Traffic Classes and QoS Requirements

It is assumed that the network contains different classes of traffic, namely real-time traffic classes with and without *reliability* constraints, and multiple priority delay-tolerant traffic classes. The system can support several different types of traffic classes, for example:

1. Real-time and reliable traffic class (RT-Rel class)
2. Real-time traffic class (RT class)
3. High priority, delay-tolerant traffic class (DT1 class)
4. Lower priority, delay-tolerant traffic classes (DT2… DT*n* class)

The RT-Rel class is the highest priority traffic class. It imposes both *timeliness* and *reliability* QoS requirements. From the application point of view, the RT-Rel traffic class can be categorised as alarm data or real-time data that is part of an integrated data stream that does not tolerate delay, such as audio/video data. This critical data needs to reach the users with a low bound delay. It cannot tolerate packet drop and needs to stay longer in the buffer for future retrieval. On the other hand, the RT class represents real-time (RT) traffic, which has *timeliness* QoS requirements but has no *reliability* constraint. The RT class is the second highest priority traffic and needs to have high throughput and low delay bound. Furthermore, there is also delay-tolerant traffic, which is divided into several classes according to its priority. For this kind of traffic, having a low delay is not too important, but information needs to be reliably sent to the users. Typically, delay-tolerant data types with *reliability* constraint are allocated more storage in the buffer within the IoT access point. Note that in a typical WSN, each node may not be having the same sensors. This means, one node may be having all possible traffic classes (RT-Rel, RT, DT1, DTn), while another node may contain only two types of traffic classes.

The QoS requirement for a specific traffic class must be explicitly defined to ensure achievement of the QoS target. For example, to accommodate the *timeliness* QoS requirements for real-time traffic, a key parameter such as ‘delay bound’ must be observed. The performance of the queuing system is analysed with the aim of minimising the average queuing delay through the buffer.

The delay of a single queue can be expressed using the M/M/1 system delay formula [96]:

$$T_D = \frac{\frac{1}{c}}{1 - \frac{\lambda}{c}} \tag{4.1}$$

where λ is the arrival rate of packets into the service queue and c is the service rate, which is equal to the capacity of the outgoing link. The delay bound is vital information that offers accurate values in the initialisation phase. This will be explained in the next section.

4.2.4 ADHERE QoS Architecture

Table 4.1 shows the ADHERE QoS architecture. In the initialisation phase, the model identifies the data rate of the traffic sources and their QoS requirements, namely the RT packet’s tolerable delay and the DT packet’s tolerable packet loss. The algorithm also estimates the required buffer size [25] of different traffic types based on this information. The required buffer size is estimated to ensure that reliability-constraint DT packets can take up more space in the IoT coordinator’s buffer. Therefore, in the QoS model, more bandwidth is allocated to RT traffic to meet low delays. In addition, buffer storage is allocated to DT traffic to avoid buffer overflow, which may result in higher traffic drop.

Table 4-1 ADHERE QoS Architecture

Function/Task	**Related Parameter**
QoS requirement identification	delay bound, loss tolerance, buffer usage requirements
Initialization	source data rate, calculate required buffer size
Queue System	*arrival and departure of packets using WFQ scheduling
QoS Monitoring (Congestion Control and Rate Adjustment Unit	
Calculate Performance Indicator	congestion degree, traffic dropped, queue size, buffer usage, ratio between packet inter-arrival time over packet service time
Adjustment Decision	e.g., if *queueThreshold*==TRUE *data_rate* = *new_rate* e.g., if *bufferusage* < 80% *buffer_size* = *new_buffer_size* e.g., if *traffic_dropped* > 30% *data_rate*= *new_rate*
Implementation of New Adjustment Parameter	Adjustment parameters include new data rate and new buffer size

To ensure optimum resource utilisation (regarding buffer usage and bandwidth allocation), a congestion control algorithm is proposed as one of the major components of the QoS framework. Combined with a proposed rate adjustment unit, the ADHERE QoS model adapts to the network dynamics typically in the event of burst and higher intensity traffic, where the accumulation of packets in the buffer becomes more rapid. These QoS components are proposed to react to these changes and maintain the QoS requirements of all traffic classes in the network.

Network performance is monitored continuously, and the QoS condition is determined using the congestion control unit. Congestion detection in WSN has been proposed in the literature using indicators such as buffer occupancy, queue length, packet service time, and the ratio of packet service time over packet inter-arrival time at the intermediary nodes. In addition, parameters such as traffic drop or packet delivery rate [25] can also be used to decide the adjustment as these parameters are closely related to the buffer's congestion degree. The rate adjustment unit calculates the new data rate of the traffic source, using the results of the congestion index [63] and the predefined source traffic priority [28]. Then, the adjustment parameter, i.e., data rate and buffer size, is calculated. This approach will ensure that the QoS requirements for different traffic classes are met while maintaining steady queues in the IoT coordinator buffer. This also ensures that allocated buffer resources are utilised efficiently.

The major components of the ADHERE QoS model, namely the queuing system and the congestion control and rate adjustment unit are presented in detail in the following section.

4.3 ADHERE QoS Components

This section describes the major components of the ADHERE QoS framework. A heterogeneous traffic class queuing model that encompasses the service differentiation policy of ADHERE is discussed first. Then the congestion control and rate adjustment scheme that supports the adaptive mechanism within ADHERE is presented.

4.3.1 A Heterogeneous Traffic Classes Queuing Model

The first component of the ADHERE QoS model is the service differentiation model which manages the heterogeneous data traffic with varying *timeliness* and *reliability* requirements. The IoT access point addresses upstream and downstream traffic flow of real-time and delay-tolerant packets between the sensor nodes and the IoT access point. The continuous delivery of packets creates multiple queues in the IoT access point, as shown in Figure 4-3.

There are two main types of network models which have been proposed in the literature for analysing network performance. They are packet-level and flow-level models. The proposed ADHERE QoS is a packet-level model, which considers a fixed number of flows, each with an infinite stream of packets. These packets are injected into the network according to a process that is governed by a congestion control mechanism. The packets are often approximated to explicitly model the relationships between round-trip times, packet drop probabilities and rate-allocation among the flows. Consequently, they capture the role of buffers as well as queing and packet delay. Packet-level models are also useful to estimate quantities such as queue sizes at buffers and throughput, hence is used study the stability of congestion control algorithm. Since only long-lived flows are considered, packet models do not capture flow-level dynamics such as flow transfer durations or the number of active flows resulting from flow bandwidth allocation procedure.

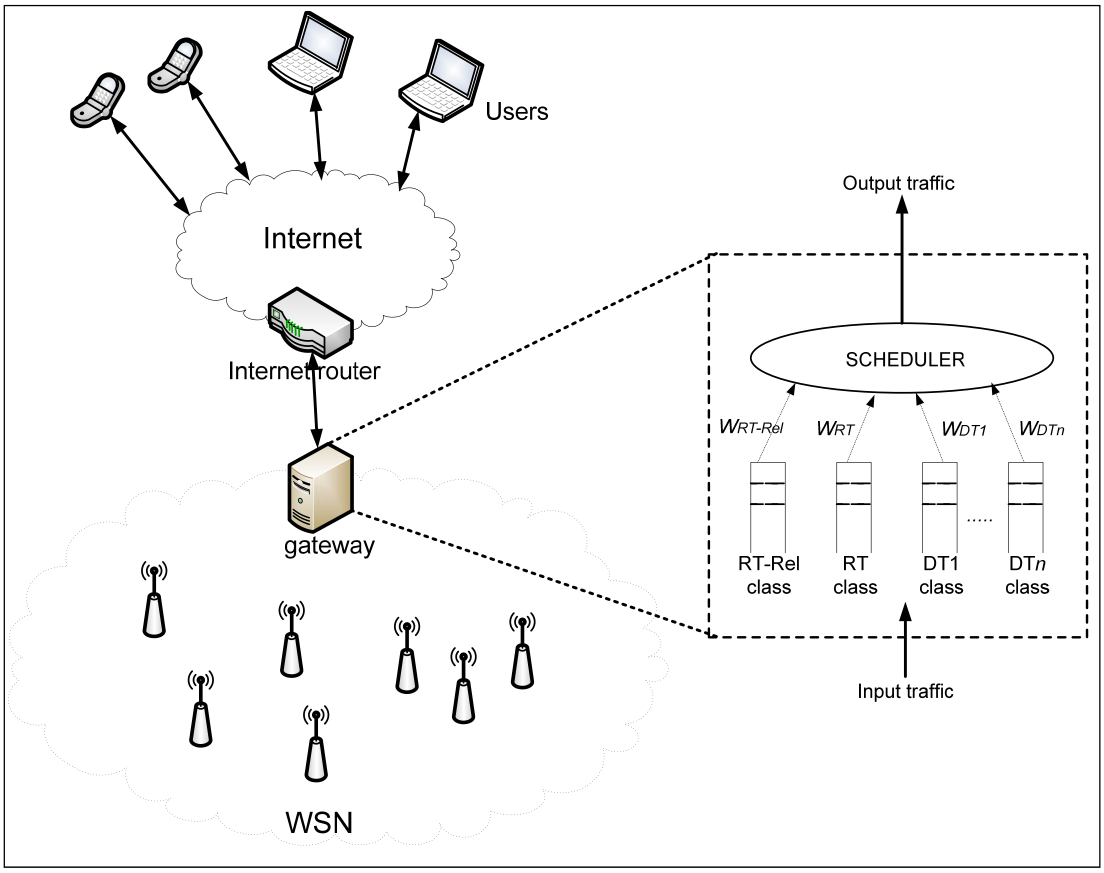


Figure 4-3 Queuing model for heterogeneous traffic classes

The generation of a WSN traffic load heavily depends on the application, which can be categorised as event-driven or periodic data generation. The event-driven scenario such as intruder detection and tracking generates bursty traffic. In the WSN literature, the constant bit rate (CBR) data traffic and the variable bit rate (e.g., Poisson distribution) are commonly employed [16, 97, 98]. Each traffic class is exponentially generated with an average packet generation, typically using packets per second.

Traffic flows are fed into the IoT access point buffer, which in turn serves the packets based on predefined service rates and using the WFQ algorithm. In WFQ policy, each traffic type is maintained in separate queues. Weights are associated with the traffic classes based on their importance and QoS requirements. Queues are then serviced using FIFO at rates based on their weights. The weights are associated with the traffic classes and their *priority index*. For instance, in a two-classes system with RT and DT traffic, the RT queue may be assigned a weight of two, and queue DT may be assigned a weight of one. Hence, two packets would be sent from queue RT for every one sent from queue DT. Figure 4-4 illustrates the working of the two separate and independent shift buffers for the incoming RT and DT data traffic.

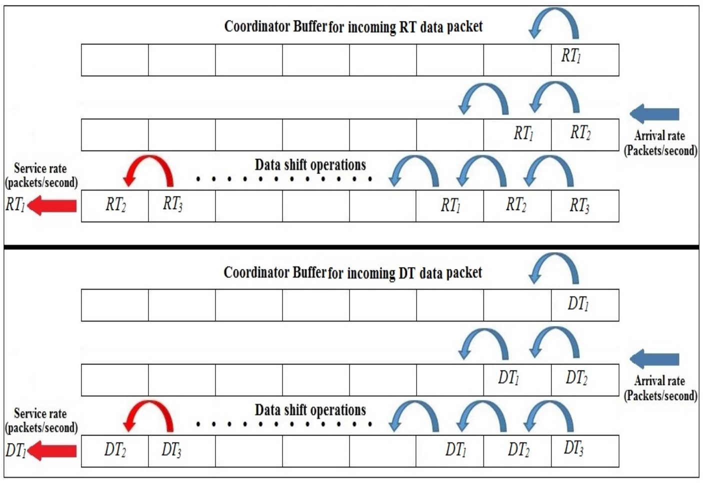

Coordinator Buffer for incoming RT data packet
Arrival rate (Packets/second)
Service rate (packets/second)
Data shift operations
RT1 RT2 RT3
Coordinator Buffer for incoming DT data packet
DT1 DT2 DT3

Figure 4-4 An example of two-classes shift buffers in an IoT access point

In the service differentiation algorithm, the source data rate is first initialised. The queue model is designed by taking into account the QoS requirements of the different traffic type. Typically, the QoS requirements of the RT packet is defined as *delay bound*, whereas the DT packet QoS requirement is defined using *packet loss tolerance*. This application-specific QoS requirement is used to calculate the data rate of the traffic sources, using the M/M/1 queuing formula:

$$\text{mean delay} = 1 / (\text{service rate} - \text{arrival rate}) \qquad (4.2)$$

Next, by considering the expression in [25], the expected buffer requirements for a system with multiple traffic types can be defined. The expected buffer storage requirements for real-time (RT) and delay-tolerant (DT) traffic types can be estimated using the following expressions:

$$\text{RT\_buffer} = (\text{datarate\_RT}/(2*\text{N}))*(\text{N-1})*\log((\text{N}/(1\text{-}0.99))\text{-}(\text{N-1})) \qquad (4.3)$$

$$\text{DT\_buffer} = \text{SERV\_RATE\_DT}*(\text{datarate\_DT}/(2*\text{N}))*(\text{N-1})*\log((\text{N}/(1\text{-}0.99))\text{-}(\text{N-1})) \qquad (4.4)$$

where N is the number of sensor nodes.

The required buffer size is estimated to ensure that DT data with a *reliability*-constraint is allocated more storage in the buffer. Therefore, in the QoS model, more bandwidth is allocated to RT traffic (by setting higher weights) to meet the low delay bound. In addition, more buffer storage is allocated to DT traffic to avoid buffer overflow which may result in a high traffic drop.

It is assumed that each node in the WSN, indicated as i, has different kinds of traffic sources. Let j represent the *traffic class*. Hence, j represents one of the traffic classes RT-Rel, RT, DT1, DT2 and so on. Then, let SP^i_j denote the traffic *source priority* in sensor node i. The value of SP^i_j could be set manually to achieve service differentiation. Ideally, the value of SP^i_j is set high enough for high priority traffic, so that it can be discriminated against the other low priority traffic.

Each sensor node, i, has a priority index, traffic class priority, P^i_{TC}, defined as the sum of the traffic source priority. This can be expressed as follows:

$$P^i_{TC} = \sum_j SP^i_j \tag{4.5}$$

where j is the traffic class, represented by RT-Rel, RT, DT1… DTn.

As each sensor has different traffic classes, each node's *global priority* associated to each traffic classes RT-Rel, RT, DT1 and DT2 can be expressed as follows:

$$GP^i_{RT-Rel} = SP^i_{RT-Rel} \tag{4.6}$$

$$GP^i_{RT} = SP^i_{RT} \tag{4.7}$$

$$GP^i_{DT1} = SP^i_{DT1} \tag{4.8}$$

$$GP^i_{DTn} = SP^i_{DTn} \tag{4.9}$$

Note that GP^i_j is calculated only for active traffic sources. If a traffic source is not active, then regardless of its type of traffic class, the value of SP^i_j is set to zero. This is applicable when some nodes do not have certain traffic classes. This setting ensures that the algorithm will share the existing network capacity only between active nodes.

4.3.2 Congestion Control and Rate Adjustment Scheme

A congestion control approach through flexible and distributed rate adjustment in the IoT access point is proposed to introduce the adaptive approach of the proposed scheme. The ADHERE QoS model avoids congestion in the network by determining the traffic's data rate, which is appropriate to the network condition. It acts as a scheduler between the network layer and the MAC layer, which maintains queues according to traffic types. Figure 4-5 shows the architecture of the congestion control and rate adjustment scheme. It consists of three major parts; namely, congestion detection unit, rate adjustment unit, and congestion notification unit.

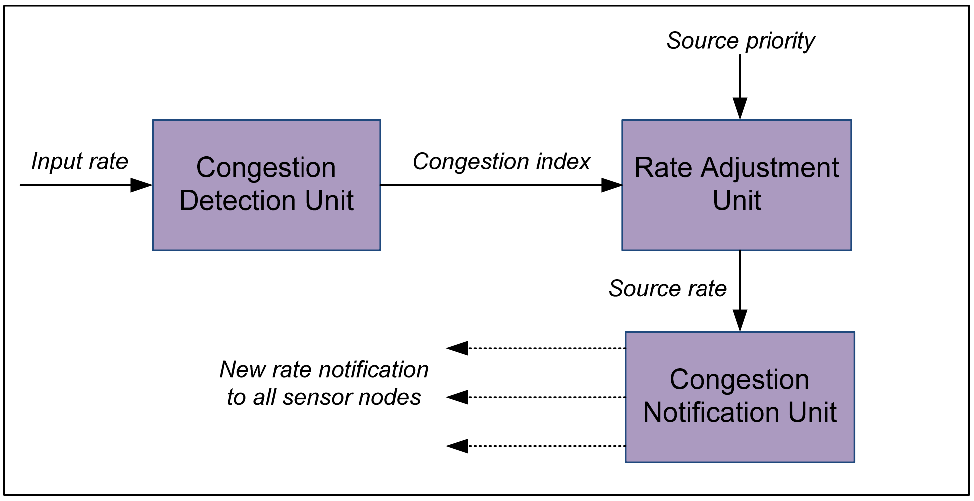

Source priority
Input rate
Congestion Detection Unit
Congestion index
Rate Adjustment Unit
Source rate
New rate notification to all sensor nodes
Congestion Notification Unit

Figure 4-5 Structure of congestion control and rate adjustment scheme

The congestion detection unit is responsible for detecting any congestion in advance. It measures the input rate determining the congestion intensity. Each sensor node measures its input traffic load and calculates the difference between its input rate and its maximum allowable transmission rate. The output of this unit is the difference between the input rate and the output rate. This can be of a positive or a negative value. In each predefined time interval, the IoT access point calculates the sending rates of all sensor nodes' traffic sources as well as its local traffic source.

The rate adjustment unit calculates the new rate for each sensor node's traffic source, as well as the local traffic source. The computation is based on the current congestion index and the source traffic priority. The new rate will be sent to the congestion notification unit.

The congestion notification unit is responsible for notifying all sensor nodes of the newly computed rate. The unit uses an implicit congestion notification by adding the new rate of each sensor node to the sending data of the IoT access point. When a sensor node detects any congestion, it will adjust its traffic source rate accordingly.

A congestion indicator strategy similar to the RED active queue management algorithm [30, 99] is employed in each network node. In each queue associated to a particular traffic class, two different fixed thresholds are defined. When the queue length is less than a minimum threshold, it implies that there is no congestion in the queue. Hence, the congestion index is set to 0. In this case, the sampling rate can be modified. On the other hand, when the queue length exceeds the maximum threshold, there is a significant congestion in the queue. Thus, the congestion index is set to 1. In this case, the sensor node should decrease its transmission rate to avoid any packet loss. Furthermore, whenever the queue length is between the two thresholds, the congestion index is set to a value between 0 and 1, depending on the queue length.

A. Initialisation phase:

In the initialisation phase, the output rate of the IoT access point is measured. Let T_s^{AP} denote the service time of the packet in the IoT access point. Using the exponentially weighted sum, the *average service time* $\bar{T}_s^{AP}$ is calculated as follows:

$$\bar{T}_s^{AP} = (1 - \alpha)\bar{T}_s^{AP} + \alpha . T_s^{AP} \qquad (4.10)$$

where α is a constant, $0 \leq \alpha \leq 1$.

The average service time is the time taken to transmit a data packet over the MAC layer successfully. It is measured starting from the time when the network layer first sends the packet to the MAC layer to the time the MAC layer notifies the network layer that the packet has been transmitted.

After computing the average service time, the *IoT access point output rate*, r^{AP}*,* can be obtained:

$$r^{AP} = \frac{1}{\bar{T}_s^{AP}} \qquad (4.11)$$

Then, the IoT access point calculates the maximum transmission rate for each sensor node, r_{max}^i. This is computed based on the sensor node's global priority (GP^i) and the IoT access point's global priority (GP^{AP}), as shown below:

$$r_{max}^i = r^{AP} . \frac{GP^i}{GP^{AP}} \qquad (4.12)$$

where GP^{AP} is the sum of the global priority of all of the sensor nodes sending their packets to the IoT access point. This step is repeated for each of the sensor nodes in the network to assign the nodes with an initial maximum transmission rate.

B. Calculate new transmission rate of the IoT access point at each periodic time interval $T_{measure}$

The total input rate at the IoT access point (r_{in}^{AP}) is computed as the sum of output rates from the sensor nodes. Let $C(AP)$ be the set of sensor nodes connected to IoT access point. Then r_{in}^{AP} is calculated as follows:

$$r_{in}^{AP} = \sum_{k \in C(AP)} r_{out}^{k} \tag{4.13}$$

where r_{out}^{k} is the output rate of the *k*th node from the IoT access point.

Then the transmission rate difference at the IoT access point is computer as follows:

$$\Delta r^{AP} = \beta . r^{AP} - r_{in}^{AP} \tag{4.14}$$

where β is a constant close to 1.

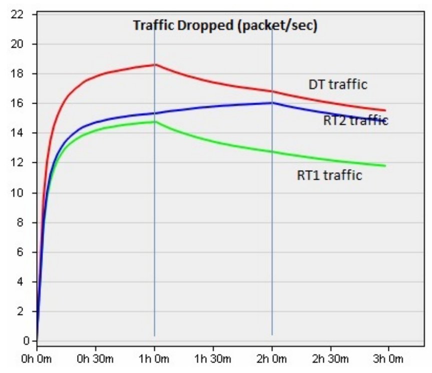
Using the value of Δr^{AP}, the IoT access point calculates and propagates the new maximum transmission rate for the sensor nodes as follows:

$$r_{out}^{i} \leftarrow r_{out}^{i} + \Delta r^{AP} . \frac{GP^{i}}{GP^{AP}} \tag{4.15}$$

The congestion control and rate adjustment unit provide continuous monitoring of a WSN-IoT network performance through assessment of the QoS on the IoT access point and makes necessary adjustments to the network configuration.

4.4 Adaptive QoS Proof of Concept

A test of an adaptive QoS concept was conducted as part of the proof of concept. A sample of ADHERE QoS outcome is illustrated in Figure 4-6. To demonstrate the continuous monitoring of the QoS condition, an adaptive QoS concept was implemented to react to the data flow dynamics close to WSN applications in the physical world. The figure shows the performance of three source variables with different priority levels and different QoS requirements.



Traffic Class	Weightage	Buffer size (packet)	Data rate (kbps)	
			Initial rate	Adjusted rate
RT1	0.5	20	32	16, 8
RT2	0.3	20	32	19.2, 12.8
DT	0.2	30	32	16, 12.8

Figure 4-6 An example of ADHERE QoS outcome within three adaptive cycles

In this example, RT1, RT2, and DT represent real-time traffic with high priority, real-time traffic with low priority and delay-tolerant traffic, respectively. The figure depicts the traffic performance during three QoS monitoring cycles. In each cycle, a performance indicator is given by average traffic dropped. Based on the status of traffic dropped, the system's QoS controller reacts by setting a new traffic source data rate. As shown in the figure, although the traffic drop of all traffic types increased in the first cycle, the rate adjustments managed to reduce the traffic drop of DT and RT traffic in the second and third cycle. The test shows that continuous network improvement and traffic reliability can be achieved through the adaptive approach. It also highlights the impact of the interaction of the WSN with the dynamics of the phenomenon monitored.

4.5 Adaptive QoS using the Neural Network

To provide a verification and added-value to the ADHERE QoS model, we propose the use of neural network for developing the learning concept that provides an extension to the proposed ADHERE QoS framework. The aim is to design a neural network-learning algorithm for the developed adaptive QoS provisioning model. Using neural network tools in MATLAB, a learning algorithm to complement and improve the developed adaptive QoS framework is developed. The learning capabilities in ADHERE should optimise the QoS framework's performance by accommodating the QoS requirements of the network through the dynamic changes of a particular application scenario.

Indeed, there is a need for building intelligence within the network such that it can adapt to the network dynamics. This can be achieved through learning and prediction of the network behaviour in relation to network dynamics. These techniques can follow and learn the known variation in a system and retrain when unknown information occurs. Soft computing techniques like Fuzzy Logic, Neural Network, Bayesian Networks and Evolutionary Algorithms enable finding solutions for complex problems with incomplete definitions and do not require a system definition. The use of a fuzzy logic controller in a service differentiation [67] indicates that the performance of the service differentiation algorithm can be further improved through the learning cycles. A fuzzy logic controller is adopted to determine the optimal traffic load parameter in a service differentiation scheme, which features a priority-based rate control system. The transmission performance in the wireless multimedia sensor network (WMSN) was improved in terms of a significant reduction of traffic delay and packet loss probability.

The assumption in the knowledge-based fuzzy logic is that the relationship between the input and output is known, whereas our assumption behind the use of the neural network model is that the relation between the input and output is vaguely known. This is because neural networks do not need a system model for prediction. Based on their training requirements, neural networks are classified into two categories; unsupervised and supervised artificial neural network. The supervised neural network needs to be trained before use. The neural network prediction capability is limited to trends, which are an extrapolation of the network's training. Unsupervised neural networks have the capability to learn and adapt to trends that they have not been trained within. This is ideal when dealing with WSN applications that have complex system behaviours and unpredictable traffic dynamics.

4.5.1 Learning and Prediction

The network QoS conditions are learnt to predict the required adjustments parameters to ensure the QoS is maintained. Figure 4-7 shows the organisation adopted for continuous learning and prediction of the neural network [85]. Continuous learning is required to constantly create awareness of the network dynamics due to the ever-changing transmission rates from different traffic types. The artificial neural network algorithm goes through three stages of the continuous learn, predict and adapt to the changing requirements of the network. The following explains the discreet processes of training, prediction and retraining:

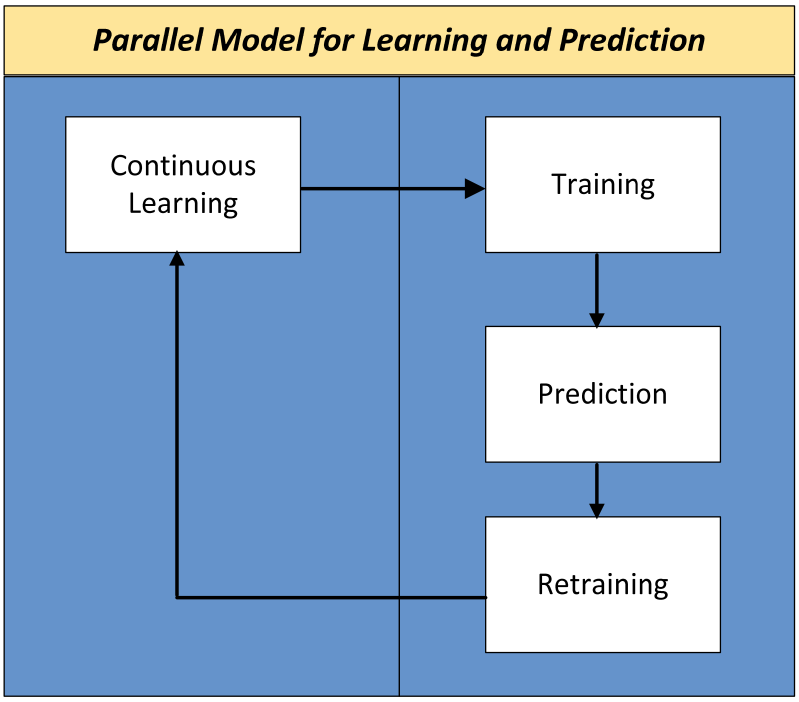


Figure 4-7 Parallel model for learning and prediction

Training: For training the neural network, the complete QoS condition is created as a map and fed to training. The neural network is trained once it discovers the adjustment parameters to the associated QoS condition.

Prediction: During the prediction, the input data comprises a subset of the QoS condition at the end of a monitoring cycle.

Retraining: A retraining is required when new and unknown QoS conditions occur, hence new adjustments are discovered. The retraining module is connected to the continuous learning module, which transfers collected data to the training module for adding new information to the neural network.

The sets of results obtained from modelling and simulation activities on the Riverbed Modeler form the basis of the neural network algorithm. In the formulated ADHERE algorithm, the network's QoS performance is observed periodically, i.e., for a predefined adjustment cycle. Based on the QoS condition, adjustments are made to the network parameters to maintain QoS requirements. The QoS condition parameters and the adjustment parameters form the neural network input and output categories, as shown in Table 4-2.

Table 4-2 Neural Network input and output parameters

	NN Input	**NN Output**
Definition	The performance indicator of the network	The adjustment parameter (determined based on the network's QoS condition)
Parameters	Traffic delay	Buffer size
	Traffic drop	Source data rate
	Delivery ratio	
	Buffer usage	

The input is defined as the behaviour of the network, i.e. the QoS performance indicator of the network. The parameters are traffic delay, traffic drop, delivery ratio and buffer usage. On the other hand, the output is defined as the adjustment parameter that has been determined using the QoS algorithm by taking into account the network's QoS condition. These control values are used to improve the network performance. The output parameters are buffer size and source data rate.

As discussed earlier in Figure 4-6, the traffic performance of heterogeneous data traffic are determined during each monitoring cycle. These traffic flows are generated under an initial data rate and serviced through queues on a predefined buffer size. The dynamics of different traffic classes, for example, in an event of an emergency which generates a burst of data, may cause the increase of traffic drop in the high priority queue as the allocated buffer size is not sufficient for the higher intensity traffic. Consequently, buffer usage will increase accordingly. Therefore, rate adjustments in the ADHERE algorithm should be able to reduce the traffic drop. While network improvement and traffic reliability can be achieved through these monitoring cycles, continuous monitoring and repetitive QoS evaluation are required. On the other hand, the use of a neural network will facilitate the system's learning of network behaviour which will be useful as the system evolves and more complexity is expected. The proper system learning, prediction and retraining in neural network will enable seamless adjustment of decision in future events, hence will accommodate the maintenance of traffic QoS requirements more efficiently without undergoing repetitive QoS evaluations.

4.6 Summary

In this chapter, the proposed ADHERE QoS framework for integrating WSN to the Internet is discussed. ADHERE is a service differentiation-based QoS framework handling various levels of real-time traffic and delay tolerant traffic within WSN. In addition, the service differentiation scheme is designed to function in the IoT access point that interconnects a WSN and the Internet. The model's main objective is to preserve the service differentiation employed by the sensor network by adapting to changes in the network traffic.

The ADHERE QoS framework is encompassed by two major components. The first component of the framework is the heterogeneous traffic queuing model that defines the way separate queues are used for each type of traffic class. The model is designed to function on different traffic classes with different QoS requirements, i.e., the timeliness and *reliability* QoS domain. The traffic classes are broadly categorised as real-time traffic classes with or without *reliability* constraints, and multiple priorities delay-tolerant traffic classes. The other major component is a service differentiation-based QoS mechanism to manage the heterogeneous data traffic. An adaptive QoS scheme is proposed by implementing a congestion control unit and a rate adjustment unit, which reacts to dynamic changes in the network.

This chapter also presents the use of neural network to offer a means of validation and optimisation for the proposed adaptive QoS mechanism. The concept of learning and prediction is presented along with a discussion of the input and output parameters of the neural network. It is targeted that the proper learning through ADHERE neural network can optimise the QoS framework's performance by providing a seamless QoS maintenance of the network with unpredictable traffic intensity.

Chapter 5: Modelling and Simulation of ADHERE QoS Framework

5.1 Introduction

This chapter presents the modelling of the **A**daptive Service **D**ifferentiation for **He**te**r**og**e**neous Data in WSN (ADHERE) QoS ideology proposed in Chapter 4. A detailed description of the modelling of key components of the ADHERE QoS concept is also presented herein. The model implementation and the testing of the individual components are presented, followed by the results from the modelling and simulation activities.

The network models and main components of the service differentiation algorithm were designed and analysed using Riverbed Modeler and MATLAB. As explained in Chapter 3, the Riverbed Modeler is a good discrete event-based network simulator, but it lacks a strong mathematical simulation framework. MATLAB, on the other hand, offers a better mathematical environment. Therefore, the characteristics of the service differentiation model were gained through MATLAB to anticipate the QoS framework performance. The service differentiation algorithm was initially implemented in MATLAB to understand and investigate its characteristics. MATLAB provides visual output for users to validate and debug the algorithm studied. The service differentiation's response to continuously streaming data with different QoS requirements (i.e. real-time or delay-tolerant) was then analysed.

5.2 Modelling and Simulation Phases

The modelling of ADHERE was separated into several phases. Figure 5-1 shows an overview diagram of the modules involved in the simulation planning phase.

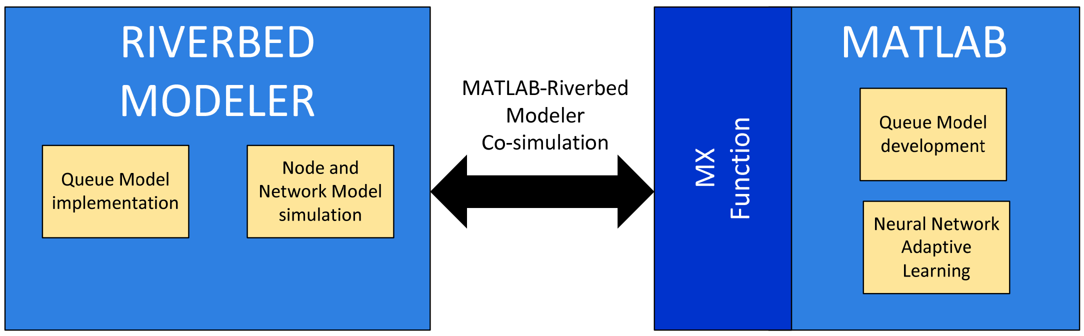

RIVERBED MODELER
Queue Model implementation
Node and Network Model simulation
MATLAB-Riverbed Modeler Co-simulation
MX Function
MATLAB
Queue Model development
Neural Network Adaptive Learning

Figure 5-1 ADHERE system organisation

The queue model was initially developed in MATLAB. The aim was to understand and investigate the characteristics of the model when service differentiation is used to treat heterogeneous data traffic with different QoS requirements. The queue model was modelled with a clear abstraction from reality, and its formal specifications of conceptualisation were based on queuing theory and associated, underlying assumptions. Based on the model produced, the queue components were implemented on the Riverbed Modeler using the process editor in the simulation tool. The queue model, along with the QoS profile created in the Riverbed Modeler, was then embedded in the node and access point model. Then, network simulation was based on the proposed network architecture, and the simulation results provided an understanding of the overall model performance. In addition, the MATLAB-Riverbed Modeler co-simulation was established to provide an avenue for a real-time QoS monitoring system that adapted to the dynamic changes of the simulated network. A mathematical analysis model within a tool such as MATLAB used the performance data derived from a network simulator, such as the Riverbed Modeler, for analysing the QoS condition and identifying the corresponding AQoS parameters for alteration. For this purpose, a co-simulation between a network modeller and a mathematical tool was set up using external interfaces or API references [85].

In addition, using the simulated data sets and the simulation results gained from the Riverbed Modeler, system training was conducted in MATLAB neural network tools. As discussed in Section 4.5, the neural network potentially offers a value-added and validation to the ADHERE model. In the neural network activity, the adaptive component of ADHERE has undergone a learning process, and the outcomes are envisaged to be a great potential for the seamless adjustment decisions in future events. The work on the neural network will be presented in Chapter 6.

5.3 Development of the Queue Model

The development of the queue model on which the service differentiation algorithm runs was initially implemented in MATLAB. MATLAB provides visual output for users to validate and debug the algorithm studied. The aim was to understand and investigate its characteristics. The service differentiation’s response to continuously streaming data with different QoS requirements (i.e. real-time and delay-tolerant) was analysed. In this phase, the main constraints of buffer usage were considered [1] before porting the algorithm as an embedded task suitable for the next simulation phase (i.e.

through the implementation of the queue model and custom QoS profile in the Riverbed Modeler).

In the simulation, two types of heterogeneous traffic were first defined - real-time traffic and delay-tolerant traffic denoted as RT and DT, respectively. The purpose of the simulation was to investigate the way heterogeneous traffics' arrival and service rate, as well as the IoT access point's buffer size influence the traffic performance in the network. The predominant performance parameters observed were buffer usage, delay, and the drop in traffic.

The parameters associated with a WSN application with RT and DT traffic were first defined. Based on the application's QoS requirements, the following were initialised:

(i) the RT and DT sources data rate
(ii) the required buffer size (in the coordinator) for both RT and DT data traffic

In the simulation, RT and DT traffic flows were fed into the IoT access point buffer, which in turn served the packets based on a predefined service rate and using a weighted fair queuing (WFQ) algorithm. In WFQ policy, each traffic type is maintained in separate queues. Weights are associated with the traffic classes based on their importance and QoS requirements. Queues are then serviced (i.e. packets are taken from the queues and sent to the outgoing line) using First-In-First-Out (FIFO) at rates based on their weights. For instance, if queue RT was assigned a weight of two, and queue DT was assigned the weight of one, then two packets would be sent from queue RT for every one sent from queue DT. Figure 5-2 illustrates the working of the two separate and independent shift buffers for the incoming RT and DT data traffic.

Riverbed Modeler supports several mechanisms for providing QoS guarantees, including Traffic Scheduler, which determines how the packets buffered in the logical queues are scheduled for departure, including WFQ and WRR. As discussed in Section 2.3.2, the main benefit of WFQ is that its implementations provide service differentiation between classes and their aggregated traffic, rather than merely differentiating between individual flows. In addition, because WFQ is bits aware, it can handle packets of variable lengths, which are more practical in WSN scenarios. Therefore WFQ is adopted in the simulation activity to offer more extensive future work. The configuration and deployment of the QoS support through the **QoS Attribute Config** within Riverbed Modeler is discussed in Section 5.4.3.

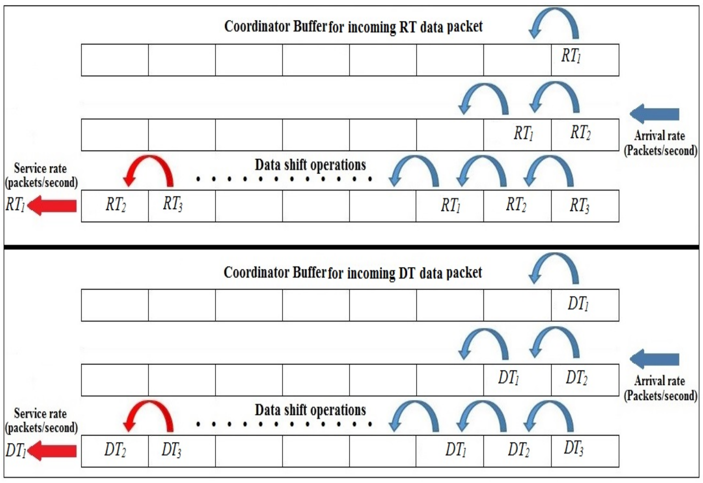

Coordinator Buffer for incoming RT data packet
RT_1
RT_1
RT_2
Arrival rate (Packets/second)
Service rate (packets/second)
Data shift operations
RT_1
RT_2
RT_3
RT_1
RT_2
RT_3
Coordinator Buffer for incoming DT data packet
DT_1
DT_1
DT_2
Arrival rate (Packets/second)
Service rate (packets/second)
Data shift operations
DT_1
DT_2
DT_3
DT_1
DT_2
DT_3

Figure 5-2 Shift buffers at the IoT access point

In the service differentiation algorithm, the source data rate is first initialised. The queue model is designed by taking into account the QoS requirement of the different traffic types. Typically, the QoS requirement of the RT packet is defined by the *delay bound*, whereas the DT packet QoS requirement is defined using *packet loss tolerance*. This application-specific QoS requirement is used to calculate the data rate of traffic sources using equation 4.1, i.e., the M/M/1 queuing formula.

The pseudo codes for deadline and service rate definition and data rate initialisation in MATLAB are shown below:

```
Initialise application-specific RT deadline

Initialise application-specific DT deadline

Set buffer service rate for RT queue

Set buffer service rate for DT queue

Initialise data rate based on M/M/1 queuing formula

datarate_RT = SERV_RATE_RT -(1/RT_deadline);                 (5.1)

datarate_DT = SERV_RATE_DT -(1/DT_deadline);                 (5.2)
```

Algorithm 5-1 Pseudo-code for deadline, service rate and data rate initialisation in MATLAB

After obtaining the data rate initialisation based from the user-specific delay approximation, next is to determine the required buffer size for both RT and DT traffic. For this purpose, the required buffer size was estimated by using the delay tolerant sensor networks expression by Liu et al. [25], i.e. equation 4.3 and 4.4. The required buffer size is estimated to ensure that DT data with a reliability-constraint is allocated more storage in the buffer. Therefore, in the QoS model, more bandwidth is allocated to RT traffic (by setting higher weights) to meet low delay bound. In addition, more buffer storage is allocated to DT traffic to avoid buffer overflow, which may result in high traffic drop.

5.4 Development of Node and Network Simulation Models

In this section, the simulation work conducted on the Riverbed Modeler is presented. In the simulation, the overall network components were modelled in such a way that the node and network models represent real infrastructures in real network settings. The similar queue model, which had been tested in MATLAB, was implemented using the Riverbed Modeler. The implementation of heterogeneous traffic sources and the simulation setup are discussed. The simulation cases and associated design parameters of the study are also presented.

5.4.1 WSN-IoT Network Simulation Setup

We first present the simulation from the preliminary simulation setup using standard network models in the Riverbed Modeler, as shown in Figure 5-3. The network model was generated based on the reference architecture presented in Chapter 4. A WSN with all sensor nodes communicating directly with the IoT access point were organised in a star topology.

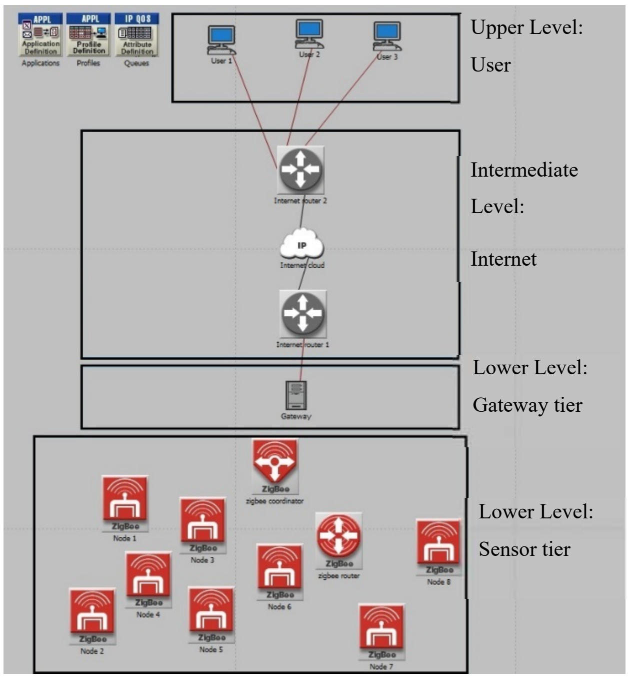


Figure 5-3 Riverbed Modeler network simulation environment

A network carrying different applications was setup. *Application Config* and *Profile Config* [84] were defined to represent the application associated with the network. The simulated application service was comparable to the Supple Service Model architecture discussed by M.Nef et al. [17] in enabling QoS in the IoT. The Supple Service Model in the IoT provides periodically collected sensory or geographical information to users. In this architecture, there are also query-based user interactions when real-time information is needed.

In the preliminary test, a traffic generator is simulated to represent steady traffic flows in one-hop transmitting data directly to the gateway. A service differentiation model that supports two major types of traffic classes was implemented [1] to simulate the co-existence of real-time and delay-tolerant traffic. The traffic classes were Expedited Forwarding (EF), which was assigned to real-time traffic, and Assured Forwarding (AF), which was assigned to delay-tolerant traffic. As shown in Table 5-1, EF traffic is generated using User Datagram Protocol (UDP) and Constant Bit Rate (CBR) traffic. AF traffic is provided using Transmission Control Protocol (TCP) and File Transfer Protocol (FTP) traffic. UDP is usually preferred over TCP in typical multimedia applications where timeliness is of greater concern than reliability [58].

Table 5-1 Simulation parameters to test a network with different traffic distributions

Parameters	**Value**	
Topology	Star	
Simulation time	1 hour	
Buffer Size	50 kBytes	
Traffic characteristic	**EF**	**AF**
Traffic types	CBR	FTP
Traffic distribution	20%	80%
Inter-arrival time	50 sec.	20 sec.
Traffic distribution	50%	50%
Inter-arrival time	20 sec.	20 sec.
Traffic distribution	80%	20%
Inter-arrival time	20 sec.	50 sec.
Packet size	40 bytes	

The effect of differentiated service was investigated by observing the network's ability to meet different QoS requirements. The performance was assessed by monitoring the packet queues under different traffic distribution (i.e. the different percentage of EF-AF traffic). In the simulation, EF-AF distributions of 50%-50%, 20%-80% and 80%-20% were generated. An inter-arrival data rate of 20 seconds was used for an equal EF-AF distribution, while 20 seconds and 50 seconds were set to simulate the 80%-20% traffic distribution. Simulation time was set to one hour, and a relatively small 50kBytes buffer size was configured. The results of the test are discussed in Section 5.6.1.

5.4.2 Node and Queue Model Implementation in the Riverbed Modeler

Node Editor within the Riverbed Modeler allows users to create and edit modules for the node model. As discussed in Section 3.2 of Chapter 3, the modules include the processor module, queue module, transceiver module, antenna module and the external system module. Packet streams and statistical streams can connect these modules. The queue module is used to model a buffer. The node is configured to switch data packets at a predefined rate. Incoming data packets will first be pushed into the buffer and data packets stored in the buffer will be sent out or serviced at another predefined rate. If the incoming packet rate is greater than the service rate, then the transitional size of data stored in the buffer will grow until the incoming packet rate is reduced.

5.4.3 Modelling Heterogeneous Data in the Network Model

Two traffic generators for the respective RT and DT traffic sources were first simulated. RT requires bounds on the delay that a packet will experience, whereas DT has a predefined loss tolerance value. To implement these bounds, RT and DT packets were treated differently within the IoT access point queues. To introduce heterogeneous traffic flow into the queue model, the following scenario was constructed in the Riverbed Modeler.

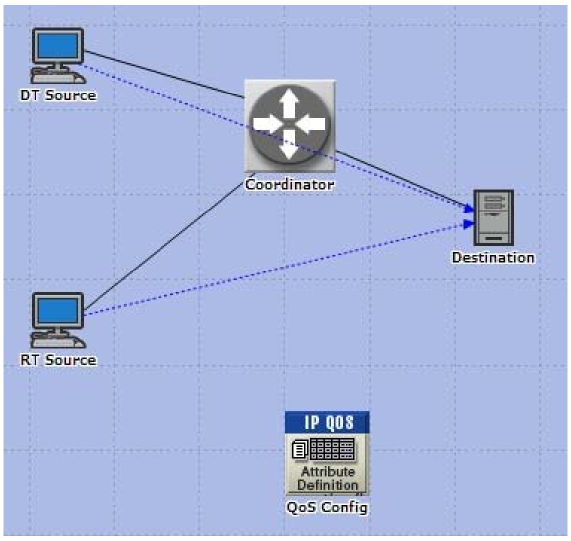
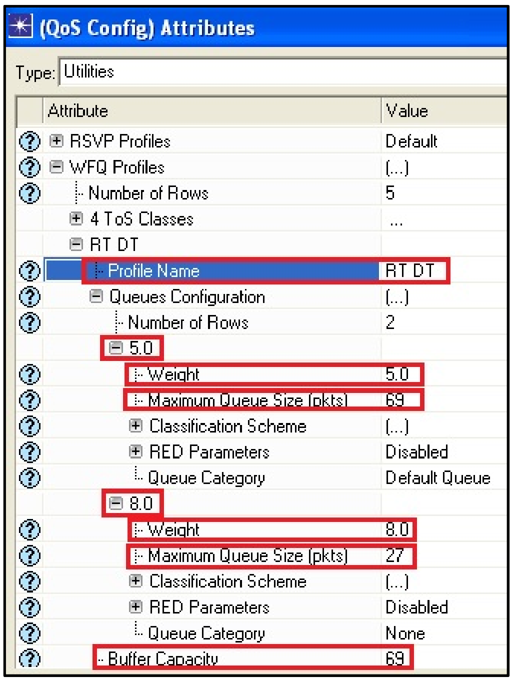


Figure 5-4 Testing the heterogeneous data in the queue model at the IoT access point

In this test, both the ‘RT Source’ and ‘DT Source’ nodes shown in Fig. 5.4 were standard node models in Riverbed Modeler (i.e. the *ppp_wkstn*). They are configured with FTP-based applications, which are identical except for the priorities associated with them. Service differentiation among RT and DT traffic was introduced by implementing the weighted fair queuing (WFQ) policy proposed in the ADHERE model. The access point handled separate queues for each of the RT and DT flows. Each traffic source can be assigned a weight that effectively controls the percentage of the link’s bandwidth each flow will get. A traffic source with a higher weight will receive more bandwidth than those with less weight [28].

The **Application Config** tool in Riverbed Modeler, discussed in Chapter 3 is used to create the applications associated with the RT and DT traffic. The **QoS Config** tool discussed in Chapter 3 is used to deploy the WFQ queuing in the IoT access point buffers. **QoS Config** is also the tool used to specify the priority level and service differentiation among RT and DT sources. Each WFQ profile configuration consists of attributes for specifying the profile name and configuration of the logical queues. In the simulation, a QoS profile, named “RT DT” was created and is deployed on the IoT access point’s interface attached to the link between the access point and Destination. Figures 5-5 and 5-6 show the RT DT Profile definition using **QoS Config** and the configuration of the access point node supporting the WFQ scheme, respectively.

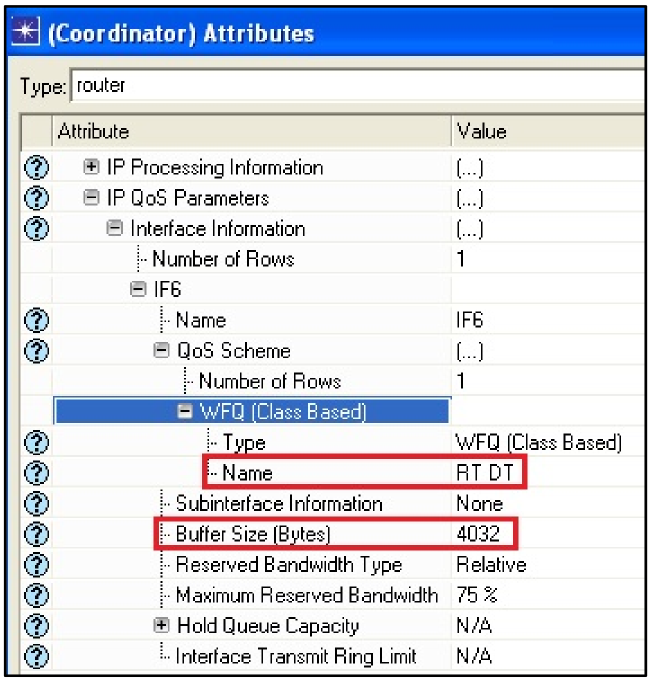


(QoS Config) Attributes

Type: Utilities

Attribute	Value
⊞ RSVP Profiles	Default
⊟ WFQ Profiles	(...)
Number of Rows	5
⊞ 4 ToS Classes	...
⊟ RT DT	
Profile Name	RT DT
⊟ Queues Configuration	(...)
Number of Rows	2
⊟ 5.0	
Weight	5.0
Maximum Queue Size (pkts)	69
⊞ Classification Scheme	(...)
⊞ RED Parameters	Disabled
Queue Category	Default Queue
⊟ 8.0	
Weight	8.0
Maximum Queue Size (pkts)	27
⊞ Classification Scheme	(...)
⊞ RED Parameters	Disabled
Queue Category	None
Buffer Capacity	69

Figure 5-5 *QoS Confiq* definition in the Riverbed Modeler



(Coordinator) Attributes

Type: router

Attribute	Value
⊞ IP Processing Information	(...)
⊟ IP QoS Parameters	(...)
⊟ Interface Information	(...)
Number of Rows	1
⊟ IF6	
Name	IF6
⊟ QoS Scheme	(...)
Number of Rows	1
⊟ WFQ (Class Based)	
Type	WFQ (Class Based)
Name	RT DT
Subinterface Information	None
Buffer Size (Bytes)	4032
Reserved Bandwidth Type	Relative
Maximum Reserved Bandwidth	75 %
⊞ Hold Queue Capacity	N/A
Interface Transmit Ring Limit	N/A

Figure 5-6 Configuration of ‘RT DT Profile’ in the access point node

As shown in the figures, the WFQ profile contains the following attributes:

- **Buffer Capacity -** specifies the buffer size in packets on the interface where the corresponding WFQ profile is deployed
- **Max Queue Size (pkts)** – determines the maximum number of packets that can be accumulated in the logical queue when the number of packets in the physical queue reaches the value of the attribute **Buffer Capacity**
- **Weight** – specifies the share of the allocated bandwidth for the corresponding queue

For example, using expressions 4.3 and 4.4, by letting N=15, the required buffer size of RT and DT yield to 27 packets and 69 packets, respectively. These values are set to the **Max Queue Size (pkts)** attribute, as shown in Figure 5-5. The buffer size represents the expected storage requirements for the different traffic types and their associated data rate.

The application supported by both 'RT Source' and 'DT Source' in Figure 5-4 were configured to transfer streams of data with predefined intervals between the transfer of subsequent files. The configuration was set on the attribute's dotted lines shown in Figure 5-4, which represented the traffic **Demand Objects**, used to specify traffic flows between two nodes. The traffic flow attribute of the Demand Objects, namely **Traffic (packets/sec)**, specifies the transmission rate of the traffic flow. In addition, each traffic demand object was also specified with the type of traffic carried by them, by setting the **Type of Service** attribute accordingly to the defined **Application Config**. The **Type of Service** associated with the applications were set to Excellent Effort (highest priority level) and Standard (lower priority level), for RT and DT traffic, respectively. Furthermore, as discussed through Algorithm 5-1, the service differentiation was also defined by setting different service rate parameters for both RT and DT using expressions (2) and (3). The configurations of these parameters will be discussed in the next section.

5.4.3.1 Test Cases to Study the Effects of Service Rates

The aim of the simulation was to identify the impact of service rate to network performance. Five experiment cases were set up – all of which were differentiated by the values of packet arrival rate and the access point's service rate. Arrival rate and service rate were defined as the number of packets that arrived at the buffer and are served by the buffer per second, respectively.

Table 5-2 shows the associated settings for the **Demand Object** and **Type of Service** attributes of the DT and RT sources in all experiment cases.

Table 5-2 Demand Object, Type of Service and transmission rate configuration

Demand Object	Attributes	Value
DT Source → Destination	Type of Service	Excellent Effort traffic
	Traffic (packets/sec)	5 packet/sec
RT Source → Destination	Type of Service	Standard traffic
	Traffic (packets/sec)	8 packets/sec

The application supported by both RT Source and DT Source was configured to transfer streams of 42-byte files with predefined intervals between the transfer of subsequent files. The arrival rate was set by defining the attribute **Traffic (packets/sec)** of the traffic sources, as shown in Table 5-2. The arrival rate of RT and DT was constant at eight and five packets per second, respectively. The packet size and data rates were inspired by a typical WSN intruder detection system [100], where RT data was represented by light sensory data whereas DT data was represented by temperature sensory data.

Table 5-3 shows the experiment case with different service rates. Five cases were set up – all of which had different service rates (SR), but the packet arrival rate (AR) and the buffer size was kept constant. The cases were:

- Case 1 when SR is equal to AR (SR=AR),
- Case 2 and Case 3 when SR is greater than AR (SR>AR),
- Case 4 and Case 5 when SR is less than AR (SR<AR)

Table 5-3 Test cases with different service rates

		Service Rate (SR), pkt/s	
Experiment Cases		RT Source	DT Source
Case 1	SR = AR	8	5
Case 2	SR > AR	16	10
Case 3	SR > AR	9	7
Case 4	SR < AR	4	2
Case 5	SR < AR	7	4

A relatively small buffer size of 20 packets was configured for each of the RT and DT buffers. The small buffer size was set to allow seeing the results of different arrival and service rates more easily. The experiment was conducted for 10 minutes.

As discussed through Algorithm 5-1, the service differentiation was also defined by setting different service rate parameters for both RT and DT using expression (2) and (3). The packet service rate is an important factor that determines the packet service time, which is the primary requirement of RT packets. Packets arriving at separate queues feeding into the IoT access point will be served at a predefined service rate. In a steady state, the service rate must be the same or higher than the packet arrival rate at the buffer. This will ensure that packets are not kept waiting to be served at the outgoing queue. On the other hand, when the packet arrival rate exceeds the packet service rate, a buffer overflow may occur. This is very likely to occur at the access point carrying the combined upstream traffic. However, the available service rate depends on the capacity of the access point nodes. Therefore, service rates for different queues need to be considered, especially when the traffic load increases due to the occurrence of an important event that generates bursts of RT data or when the total number of nodes increases.

If RT and DT have the same packet arrival rate, the trends of the buffer usage and serviced packets will show significant difference, especially if the access point's buffer size is set to a small value. RT packets are anticipated to have lower queuing delay, and DT traffic will be higher. The statistics serviced packets, buffer usage and packet drop are observed in this experiment.

5.4.4 Real-time QoS Monitoring using the MATLAB-Riverbed Modeller Co-Simulation

A mathematical analysis model within MATLAB can use the performance data for analysing the QoS condition and identifying corresponding ADHERE parameters that need to be adjusted. For this purpose, a co-simulation between a network modeller and a mathematical tool was set up using external interfaces or API references [85].

Figure 5-7 shows the ADHERE concept implementation on MATLAB-Riverbed Modeler co-simulation environment. It shows the service differentiation QoS model and a Congestion Control Unit run in Riverbed Modeler and MATLAB, respectively.

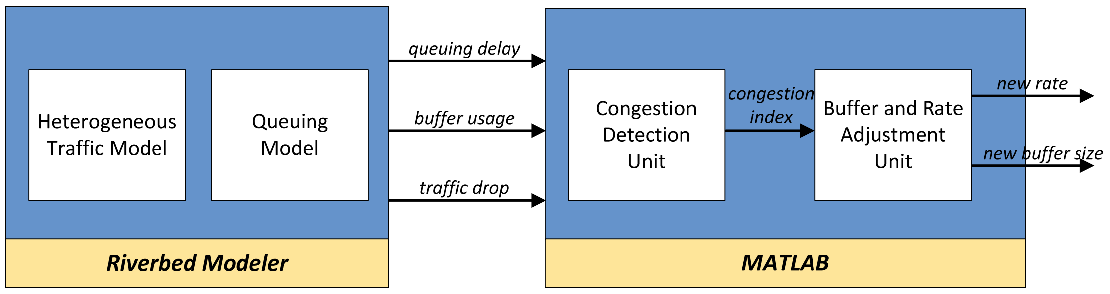


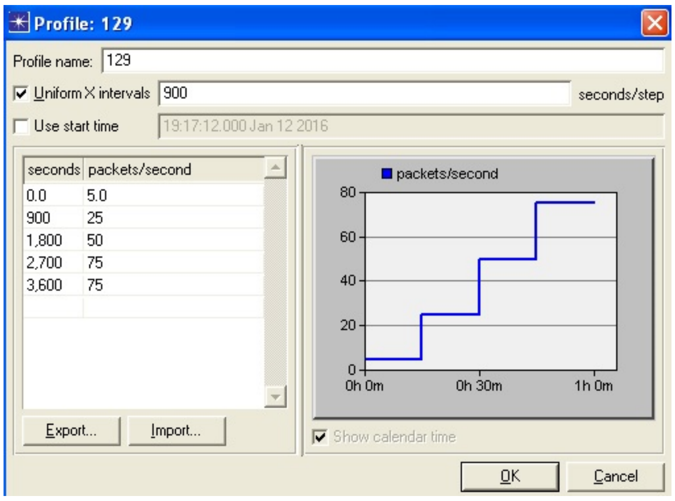
Figure 5-7 ADHERE implementation on the MATLAB-Riverbed Modeler co-simulation

The QoS algorithm shown in Table 4-1 in Chapter 4 was implemented. The Riverbed Modeler runs the queuing and heterogeneous traffic models, which were discussed in Sections 5.3 and 5.4. The network performance indicators derived from the model hosted by the Riverbed Modeler are queuing delay, buffer usage and traffic drop for all traffic types. These are defined as follows: queuing delay is the duration packets have to wait in the queue before being sent; buffer usage is defined as the number of packets waiting in the queue at any time during the simulation, and traffic drop is defined as the number of packets dropped due to buffer overflow. These parameters represented the QoS condition of the network passed to the Congestion Control Unit hosted by MATLAB. The Congestion Control Unit analysed the QoS condition and identified the corresponding network parameter for adjustment to maintain the required QoS. The Congestion Detection Unit first calculated the congestion index and the outcome was used by the Rate and Buffer Adjustment Unit to determine the new data rate for each traffic types and the buffer size for different queues at the access points. The aim was to ensure the QoS requirements for different traffic classes were met while maintaining steady queues at the access point buffer.

The simulation activity discussed in Section 5.4.1 was conducted to investigate the model performance discussed above and is based on predefined traffic distribution. This exercise investigated the way traffic dynamics affect the network. Sensor network dynamics are the effects of the change of certain network parameters through a course of time. These may include changes in intensity to traffic flows especially due to bursts of sensed data when an event is triggered, changes in the numbers of active sensor nodes and gateway devices, and bandwidth availability.

5.4.4.1 Test Cases to Study the Traffic Dynamics and Buffer Size

To investigate the network performance under different traffic load, simulation cases with varying data rates were conducted during this simulation phase. Varying traffic intensity is simulated to demonstrate traffic dynamics. To introduce greater traffic intensity, the value of **Traffic (packet/sec)** attribute in the **Demand Object** for both RT Source and DT Source are increased over the course of simulation time. This is to represent bursts of traffic for an event or network with greater number of sensor nodes. Figure 5-8 is an example of the way the traffic data rates change with time.



Profile: 129

Profile name: 129

☑ Uniform X intervals 900 seconds/step

☐ Use start time 19:17:12.000 Jan 12 2016

seconds	packets/second
0.0	5.0
900	25
1,800	50
2,700	75
3,600	75

☑ Show calendar time

Figure 5-8 Defining different rates of traffic intensity

Figure 5-8 shows the approach used to increase the arrival rate of DT traffic. The arrival rate is in packets/s. A simulation on Riverbed Modeler involving 15 nodes was conducted to investigate the behaviour of the network under varying data rates and buffer sizes. Over the period of one-hour simulation, the RT and DT packets' data rates were increased every 15 minutes. The following table shows the arrival rate values for RT and DT traffic over the simulation time.

Table 5-4 Arrival rate change over one-hour simulation

	Arrival Rate (AR), pkt/s	
Simulation Time (min)	RT Source	DT Source
0 - 15	8	5
16 - 30	40	25
31 - 45	80	50
46 - 60	120	75

A low buffer size of 30 packets was initially set. Then, during the simulation run-time, the ADHERE QoS model estimated the required buffer size for heterogeneous traffic. The estimated buffer size was then allocated to the associated RT and DT traffic.

ADHERE aims to allocate sufficient buffer storage to DT traffic to avoid buffer overflow, which may result in a high traffic drop, and to allocate more bandwidth to RT traffic to meet low bound delay. This will accommodate the timeliness and reliability QoS requirements of RT and DT traffic.

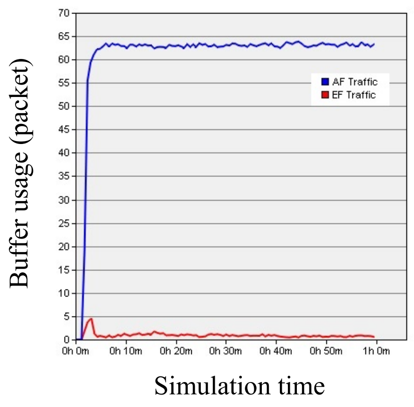
Buffer usage, queuing delay and traffic drop statistics were observed in this experiment. The required performance of the ADHERE QoS model to meet the timeliness and reliability requirements of both RT and DT packets was analysed.

5.5 Results and Evaluation

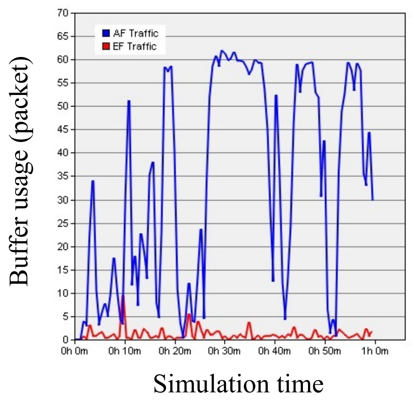
This section presents the performance measures and behavioural characteristics of the simulation cases.

5.5.1 Queue Model Performance under Different Traffic Distributions

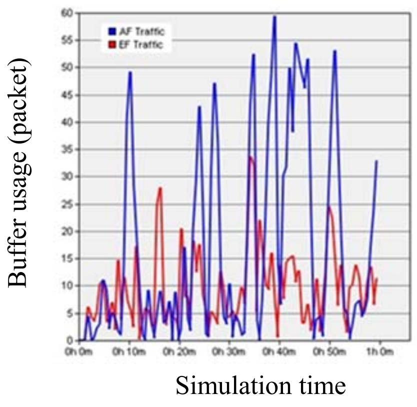
As discussed in Section 5.4.1, the network performance under different real-time and delay-tolerant traffic distributions was investigated. The network performance was assessed by monitoring the packet queues in the gateway's buffer under different kinds of traffic distributions. In the simulation, EF-AF distributions of 50%-50%, 20%-80% and 80%-20% were generated. The service differentiation's ability to meet both types of traffic QoS requirements was investigated through buffer usage, queuing delay, and the amount of traffic dropped.



(a) 50% - 50%



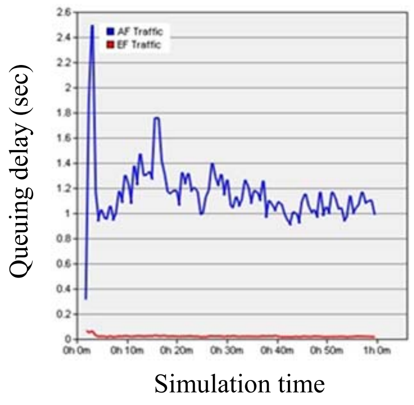
(b) 20% - 80%



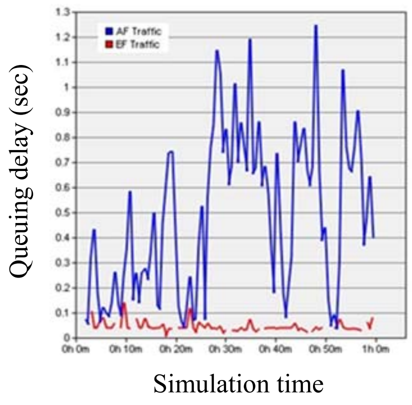
(c) 80% - 20%

Figure 5-9 Buffer usage (packet) vs simulation time for different modes of EF-AF traffic distributions

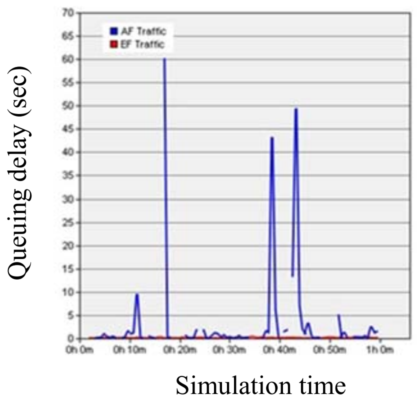
The first statistic was buffer usage, defined as the number of packets waiting in the queue at any time during the simulation. As shown in Figure 5-9, there were significantly greater AF packets waiting in the queue for the entire simulation, while EF packets were seldom kept waiting. However, the buffer usage of the EF traffic increased for the 80%-20% distribution due to a higher data rate that introduced a greater volume of data in the buffer. While EF packets were forwarded to the output traffic, the AF packets occupied larger buffer space. Hence, the results indicate that both EF and AF packets achieved their QoS requirements.



(a) 50% - 50%



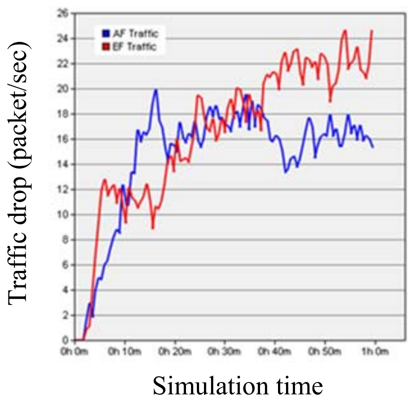
(b) 20% - 80%



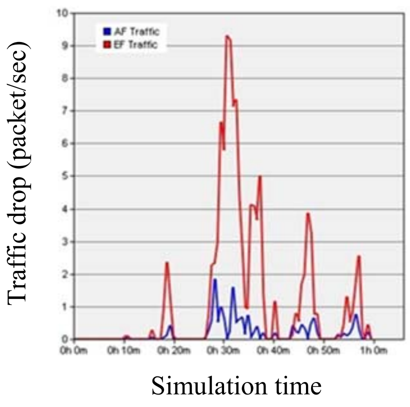
(c) 80% - 20%

Figure 5-10 Queuing delay (sec) vs. simulation time for different kinds of EF-AF traffic distributions

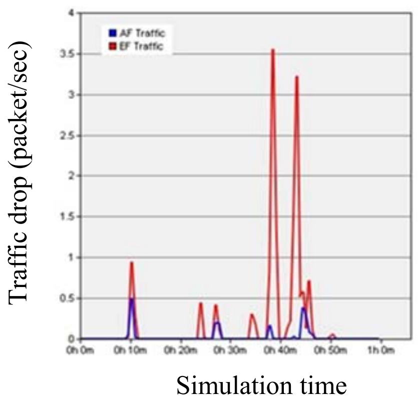
The second statistic is queuing delay (i.e. the duration that packets have to wait in the queue before sending). As shown in Figure 5-10, due to service differentiation, the AF traffic experienced a longer queuing delay than the EF traffic, especially in the 80%-20% distribution. The result also showed that the differentiated service provided a low delay bound for EF traffic and all traffic distribution. This indicated that the EF traffic with timeliness requirements was first to be forwarded to the external network, regardless of the order of arrival.



(a) 50% - 50%



(b) 20% - 80%



(c) 80% - 20%

Figure 5-11 Traffic drop (packet/sec) vs. simulation time for different EF- AF traffic distributions

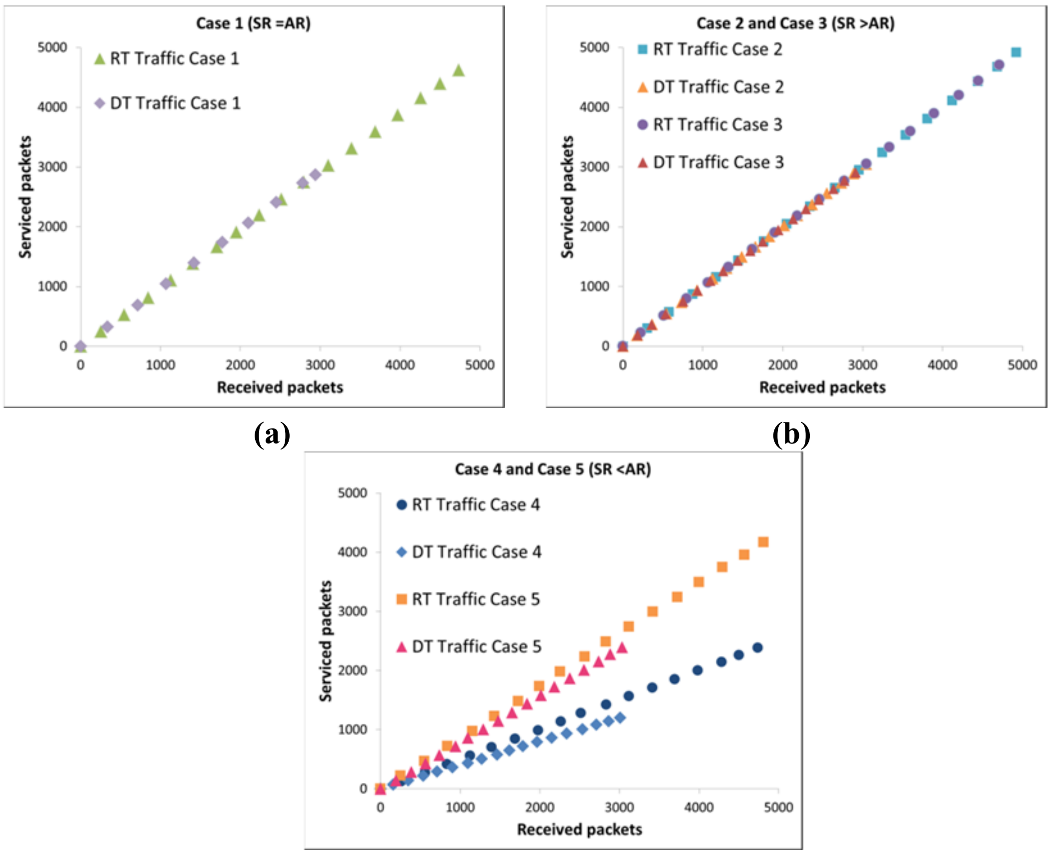
Last to be investigated is the traffic drop, defined as the number of packets dropped due to buffer overflow. As shown in Figure 5-11, the AF traffic has a lower drop rate than the EF queue. Although the EF traffic was serviced first, it was often lost before delivery. This was acceptable as the EF traffic had more tolerance to packet losses compared to the AF traffic. On the other hand, while the AF packets travel slower (due to higher queuing delays), they are delivered with much more reliability. In addition, due to constrained buffer capacity, a small percentage of EF packets were evicted due to high storage pressure. The reliability of both AF and EF packets can be improved with larger gateway buffers.

The results suggest that when the service differentiation and buffer eviction policy are used, both the timeliness and reliability of QoS requirements imposed by different packet types can be met. The scheme ensures low delay bound for EF packets while maintaining low packet loss for AF traffic. Hence, the framework is suitable for a network with mixed priorities and varying QoS requirements regarding timeliness and reliability.

5.5.2 Effect of Service Rate

Network performance in the simulations for Cases 1 to 5, as outlined in Table 5-3, is observed in this section.

The effect of the service rate was investigated through the amount of serviced packets by the IoT coordinator. It was observed that RT traffic and DT traffic are served according to assigned serviced rates for the different buffer queues.



(c)

Figure 5-12 Serviced packets vs. received packets for (a) Case 1 (SR = AR), (b) Case 2 and Case 3 (SR > AR), (c) Case 4 and Case 5 (SR < AR)

When SR and AR were equal (Case 1), more than 97% packets were served through the outgoing buffer queues for both RT and DT traffic. The small buffer size gradually built up the queues, but incoming packets were steadily served into the outgoing queues under SR=AR. Better performance was shown in Cases 2 and 3 when SR>AR. A 100% delivery rate is shown as all packets were successfully served. However, as shown in Figure 15-12 (c), it is observed that much less than 100% packets were served when SR <AR. The delivery rate for Case 4 was 50.1% and 39.8% for RT and DT traffic, respectively. Traffic was also dropped in Case 5, as only 86.5% and 78.9% was served for RT and DT traffic, respectively. The DT traffic reliability requirement was not

addressed due to insufficient buffer size. It is expected that even with a larger DT buffer size, traffic will eventually be dropped at a certain point with higher intensity incoming traffic. This suggests a need for continuous monitoring and network adjustments to accommodate the trade-offs between RT and DT QoS requirements.

As shown in Figure 15-12 (b), Case 2 and 3 had a 100% delivery rate, which indicates 0 buffer usage and 0 packet drop for the entire simulation time. This also indicates that when the packet rate for traffic source nodes is increased, the IoT access point is not fast enough to remove the packets in the buffer, which leads to continuous build ups of buffer occupancy. Consequently, once the buffer usage has reached its maximum capacity, packet starts to be dropped. Subsequently, more incoming packets will be dropped due to buffer overflow. A solution is to adjust the buffer size to match the requirements of the RT and DT traffic, which will be discussed in the next section.

5.5.3 Effect of Buffer Size and Arrival Rate

The results of incorporating the QoS model into our network modelling and simulation on the Riverbed Modeler are described in this section. A simulation on Riverbed Modeler involving 15 nodes was conducted to investigate the behaviour of the network under varying data rates and buffer sizes. Service differentiation among RT and DT traffic was introduced by implementing a WFQ policy, whereby each traffic source was assigned a weight. Traffic sources with higher weight receive more bandwidth than those with less weight. For this purpose, the normalized weight assigned to RT and DT traffic classes are set to to 0.7 and 0.3, respectively. In addition, a low buffer size of 30 packets and estimated buffer size were allocated for RT and DT traffic, respectively.

Over the period of a 1-hour simulation, the RT and DT packets data rate were increased every 15 minutes. Sufficient buffer storage was allocated to DT traffic to avoid buffer overflow, which could result in a high traffic drop. In addition, higher bandwidth was allocated to RT traffic to meet low bound delay. This would accommodate the timeliness and reliability of QoS requirements associated with RT and DT traffic. The queue performance at a range of traffic intensities was investigated and the way suitable estimated buffer size could accommodate the traffic's QoS requirements was observed.

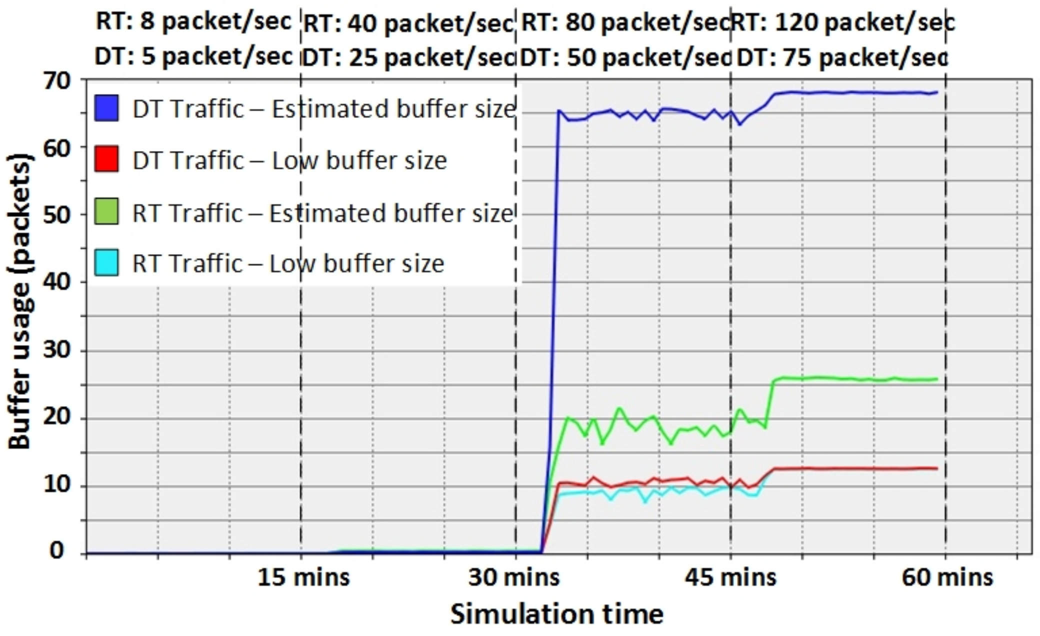


Figure 5-13 Buffer usage (packets) vs. simulation time

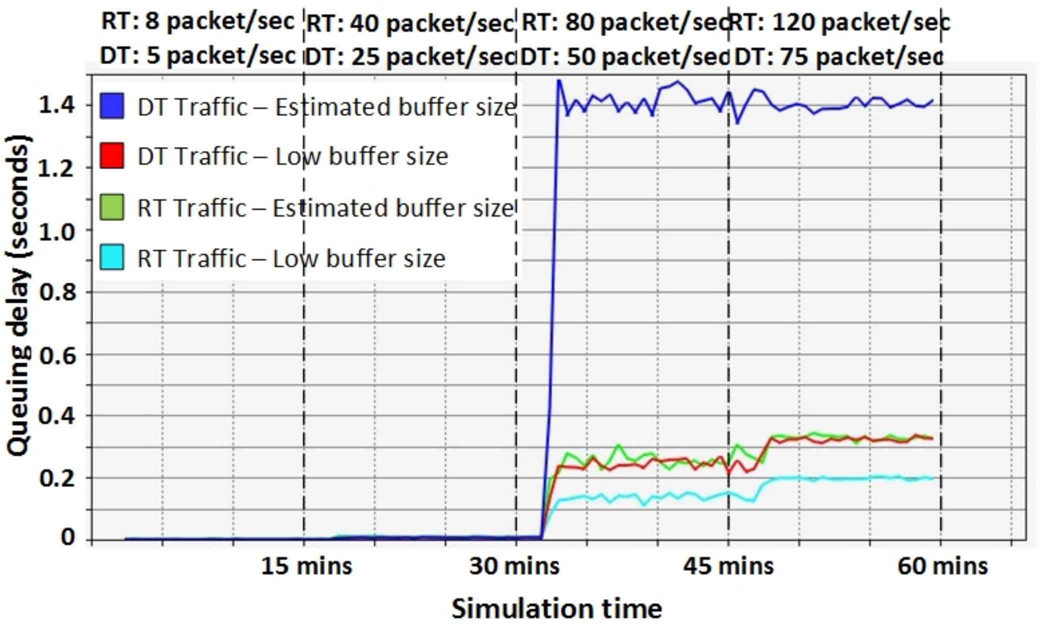


Figure 5-14 Queuing delay (seconds) vs. simulation time

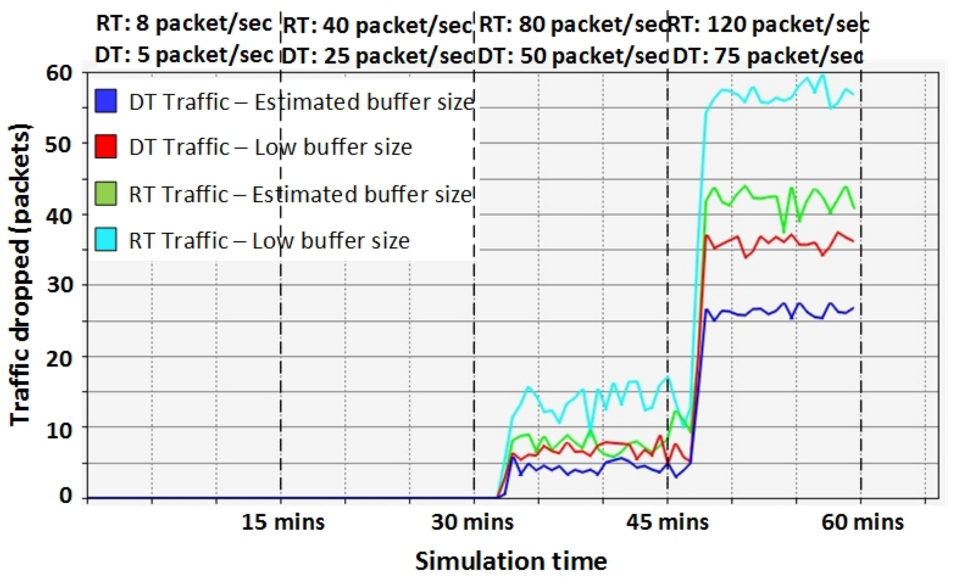


Figure 5-15 Traffic drop (packets) vs. simulation time

Figures 5-13 to 5-15 show the performance for varying RT and DT data rates in terms of buffer usage, queuing delay and traffic drop, respectively. It shows that when the arrival rate is increased, the estimated buffer can accommodate the traffic burst, even when the service rate for the IoT access point remains constant. The results confirm that by introducing service differentiation among traffic as well as estimating the required buffer size, both RT and DT traffic met their QoS requirements. As shown in Figure 5-14, by setting higher weights to the RT traffic, more bandwidth was allocated, and the low delay requirement of RT traffic was met. Furthermore, as shown in Figure 5-15, by estimating sufficient buffer size to DT traffic, traffic drop can be minimised. Hence, both timeliness and reliability requirements are accommodated.

5.6 Summary

The modelling and simulation of the ADHERE QoS model are presented in this chapter. A detailed description of simulation and experiment cases on Riverbed Modeler and MATLAB has been given. The network models and main components of the service differentiation algorithm are designed and analysed using Riverbed Modeler. The service differentiation's response to continuously streaming data with different QoS requirements (i.e. real-time or delay-tolerant) has been analysed.

Several design parameters have been discussed and implemented in the experiment case studies. The network model was tested under several design parameters; namely, traffic distribution, buffer size, traffic arrival rate, traffic service rate and WSN node density. The performance parameters analysed are queuing delay, packet drop, buffer usage and delivery rate. Results show that the varying timeliness and reliability QoS requirements of RT and DT traffic can be met with the ADHERE QoS concept. Our findings show that the service differentiation among traffic and estimation of sufficient buffer size can accommodate the RT traffic timeliness and DT traffic reliability QoS requirements. Particularly, the low delay requirement of RT traffic was met through sufficient bandwidth allocation. In addition, the higher delivery rate requirement of DT traffic was achieved by using the ADHERE estimation of sufficient buffer size.

The network modelling using the Riverbed Modeler simulation indicates the overall system performance and provide QoS-based design guidelines for an actual WSN-Internet integration system. Furthermore, the co-simulation concept of the Riverbed Modeler and MATLAB was viable and could facilitate analysis of the network when complexity takes place as the network evolves.

Chapter 6: ADHERE Validation and Optimization using a Neural Network

6.1 Introduction

This chapter presents the implementation of a neural network concept which offers an extension to the simulated ADHERE QoS framework. The objective is to provide a value-added feature of the ADHERE QoS model as well as to observe similar results of the initial observation from the modelling and simulation activities. The aim is to design a neural network learning algorithm for the developed adaptive QoS model. Using neural network tools in MATLAB, a learning algorithm to complement the proposed adaptive QoS framework is developed. The adaptive component of ADHERE undergoes a learning process, whose outcomes will be used for seamless adjustment of decisions in future events. The learning capabilities in ADHERE optimizes the QoS framework's performance by seamlessly accommodating the QoS requirements of the network which experience unpredictable dynamic changes in certain application scenarios.

6.2 Learning and Prediction Process

As introduced in Chapter 5 (refer Figure 5-1), using the simulated data sets and the simulation results gained from the Riverbed Modeler, a system training is conducted using MATLAB neural network tools. The network QoS conditions are learnt to predict the required adjustments parameters, to ensure that the QoS is maintained. Figure 6-1 shows the organisation adopted for continuous learning and prediction of the neural network. Continuous learning is required to constantly create awareness of the network dynamics due to changes of transmission rates from different traffic types. The neural network algorithm goes through three stages of the continuous learning, predicting and adapting to the changing requirements of the network. The following explains the discrete processes of training, prediction and retraining:

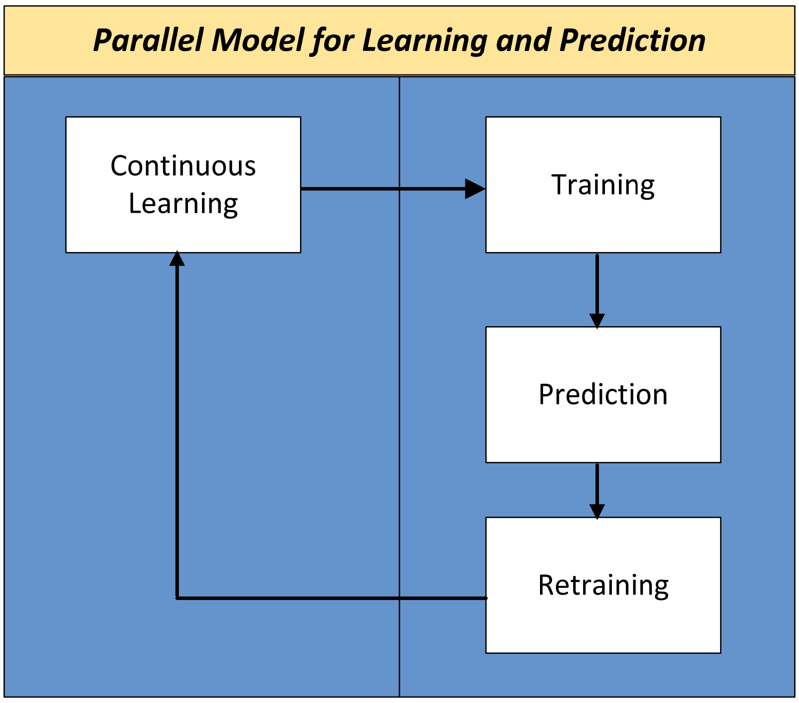


Figure 6-1 Parallel model for learning and prediction

Training: For training of the neural network, the complete QoS condition is created as a map and fed to training. The neural network is trained once it discovers the adjustment parameters to the associated QoS condition.

Prediction: During prediction, the input data comprises a subset of the QoS condition at the end of a monitoring cycle.

Retraining: Retraining is required when new, and demand for unknown changes in required QoS conditions occur. The retraining module is connected to the continuous learning module, which transfers collected data to the training module for adding new information to the neural network.

The method used for ADHERE Neural Networks concept is based on back-propagation neural network. In the formulated ADHERE algorithm, the WSN network performance is observed periodically, i.e., for a predefined adjustment cycle. Accordingly, adjustments are made to the network parameters to maintain the QoS requirements. The input parameters and output parameters of the neural networks in MATLAB are shown in Figure 6-2. The sets of results obtained from modelling and simulation activities on the Riverbed Modeler are collected, forming the basis on which the proposed ADHERE algorithm is implemented.

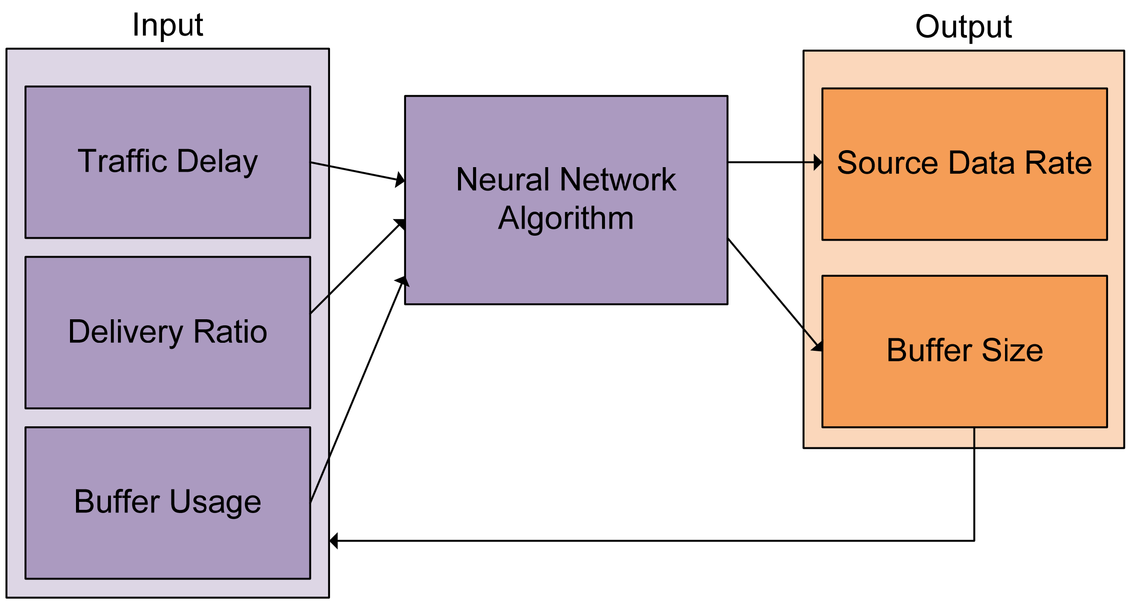


Figure 6-2 ADHERE neural network organisation

The neural network input is defined as the QoS instance of the network, which has been collected through the simulation activities. The input parameters are: traffic delay, delivery ratio and buffer usage, which are the QoS performance indicator of the network. On the other hand, the output is defined as the adjustment parameters which have been determined using the QoS algorithm by taking into account the network's QoS condition. These are the control values that are used to improve the network performance. The output parameters are buffer size and source data rate.

As discussed in Chapter 5, the queuing delay is defined as the duration packets have to wait in the queue before being sent. In addition, the buffer usage is defined as the number of packets waiting in the queue before being sent. A new QoS parameter observed in the neural network learning and prediction process is the delivery ratio - defined as the number of packets successfully serviced by the IoT access point over the packets that arrived at the queue.

In the simulation activities, the way different buffer buffer sizes accommodate different traffic QoS requirements has been observed. The initial observations indicate that when source data rate is increased, traffic drop is likely if the IoT access point's buffer size is small. As such, the delivery ratio and buffer usage are the main performance indicators for DT traffic. As discussed in the simulation results in Chapter 5, once the buffer usage has reached its maximum capacity, packets start to be dropped. Subsequently, more incoming packets will be dropped due to buffer overflow. When this happens, buffer usage is high and delivery ratio is very low. Hence by estimating the required buffer size, higher intensity traffic can be accommodated. In other words, the delivery ratio can be optimised with adequate buffer size. On the other hand, the indication from traffic delay is used to accommodate the timeliness requirements for RT data. Adjustment to the source data rate of the RT traffic is done to maintain the required timeliness QoS requirement. Therefore, by adjusting the RT source data rate and estimating the required buffer size for DT traffic, the timeliness and reliability QoS requirements can be met.

6.3 Data Collection

Data collection in MATLAB involves the following 5 steps, which are 1) Data selection, 2) Validation and test data, 3) Network architecture setup, 4) Train the network and 5) Evaluate the Network

Step 1: Data Selection

The selection of data is comprised of two major data – input data and target data. All data are collected from the statistics obtained from Riverbed Modeler of various network scenarios. The input data presented to the network are queuing delay, buffer usage and delivery ratio performance under a range of traffic intensity. On the other hand, target data defines the desired network output. The parameters are source data rate and buffer size.

The training samples have been selected to ensure that the network conditions would have adequate parameters variations, hence providing sufficient and proper training during the neural network learning. Network cases with different network attributes are simulated to represent the possible states of the network, from steady condition to extreme cases where congestion is likely due to high traffic intensity and insufficient buffer size. From the collected data, a number of network cases are categorised to represent different network conditions. The network cases are categorised as 1) settled cases, 2) high traffic intensity cases, 3) extreme cases and 4) very extreme cases. These categories will be discussed in detail in Section 6.4.

Step 2: Validation and Test Data

The samples for training, validation, and testing are formulated to represent different training cases. Each data set is comprised of 70% training samples, 15% validation samples and 15% testing samples. The training cases will be discussed in detail in Section 6.4.

Training samples are presented to the network during training, and the network is adjusted according to its error. Validation samples are used to measure network generalization, and to halt training when generalization stops improving. Testing samples have no effect on training and therefore provide an independent measure of network performance during and after training.

Step 3: Network Architecture

In this step the number of neurons in the fitting network's hidden layer is set. It is recommended to change the number of neurons if the network does not perform well after training.

Figure 6-3 shows a two-layer feed-forward network with sigmoid hidden neurons and linear output neurons which can fit multi –dimensional mapping problems. The network should be provided with consistent data and enough neurons in its hidden layer. As described in Section 6.2, the input parameters are packet delay, delivery ratio and buffer usage whereas the outputs are source data rate and buffer size.

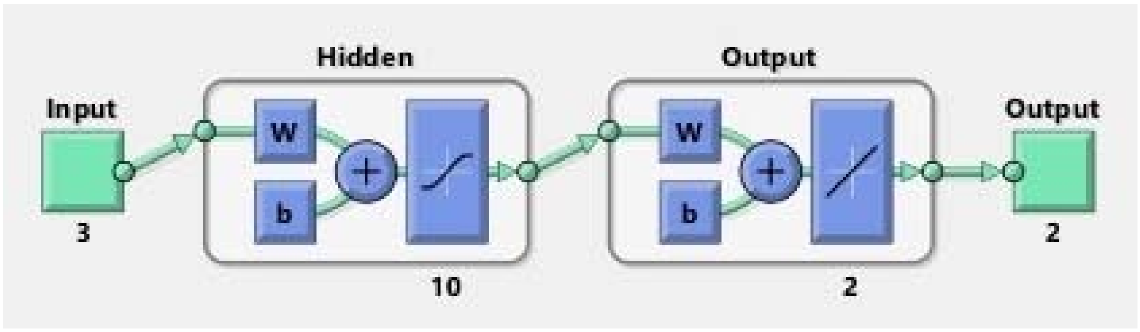


Figure 6-3 Neural network layers

Fitting networks are feedforward neural networks used to fit an input-output relationship. Feedforward networks consist of a series of layers. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer produces the network's output.

Feedforward networks can be used for any kind of input to output mapping. A feedforward network with one hidden layer and enough neurons in the hidden layers, can fit any finite input-output mapping problem. Specialized versions of the feedforward network include fitting and pattern recognition networks in MATLAB. A variation on the feedforward network is the cascade forward network in MATLAB, which has additional connections from the input to every layer and from each layer to all the following layers.

Step 4: Train the Network

As mentioned in Step 1 (Data selection) the scenario cases are selected based on several categories of network conditions. Data are selected to introduce a balance of variation for the training purpose. However, extreme cases are introduced later in the training. This means that on-going training will occur as a proper training to the system, as it would ensure smooth network parameter adjustment. The system should also be trained based on unknown conditions and should cope with extreme conditions that may occur in the future, hence it should be able to react to certain incoming conditions as expected. The aim of this training strategy is to ensure that retraining is included and to demonstrate that the system is working in an adaptive manner.

The network is trained with Levenberg-Marquardt backpropagation algorithm (*trainlm*), which is often the fast backpropagation algorithm in the Neural Network Toolbox in MATLAB. It is highly recommended as the first-choice supervised algorithm. This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error (MSE) of the validation samples. Conducting the training for multiple times will generate different results due to different initial conditions and sampling. Furthermore, retraining optimizes the network on inputs and targets.

Step 5: Evaluate the Network

After the neural network performance is observed to obtain an MSE value that indicates good training, the network can optionally be tested on more data. Iterating for improved performance includes doing the training all over again if the first attempt did not generate good results or if marginal improvement is required. In addition, the network size can also be adjusted by changing to higher neurons if the retraining did not help. A larger data set will need to be used if higher neurons did not offer any improvement.

6.4 Training and Testing Process

6.4.1 Data Sets

The behaviour of the model is evaluated using different traffic load by setting a range of packet arrival rates. Arrival rate ranging from 2 to 20 pkt/s at the IoT access point for a given buffer capacity has been collected using Riverbed Modeler. For the training data, the values of the inputs (traffic delay, delivery ratio and buffer usage) of the network have been obtained with 99% confidence level and 0.05 maximum relative error using the Riverbed Modeler simulator.

The training samples have been selected to ensure that the network conditions cover adequate parameters combinations, hence providing sufficient and proper training during the neural network learning. These cases differ in terms of the arrival rate of RT and DT traffic classes, queue conditions (SR=AR, SR>AR, SR<AR) and buffer service rates. Based on the arrival rates values of the RT and DT data, these cases are categorised into 4 categories, namely Settled Cases, High Traffic Intensity Cases, Extreme Cases and Very Extreme Cases. The following table shows the values range for the parameters used in the data collection.

Table 6-1 Network performance categories and related parameters for training data

WSN Network performance categories	**Arrival rate (AR) values range**	**Queue Condition (SR/AR)**	**Service rate (SR) values range**
Category 1: Settled Cases	RT = DT 2 to 13 pkt/s	SR=AR SR>AR SR<AR	(SR=AR, SR>AR, SR<AR) Range: 1 to 15 pkt/s
Category 2: High Traffic Intensity Cases	RT =DT 14 to 16pkt/sec	SR=AR SR>AR SR<AR	(SR=AR, SR>AR, SR<AR) Range: 7 to 18 pkt/s
Category 3: Extreme Cases	RT > DT RT: 13 to16 pkt/s DT: 9 to 10 pkt/s	SR<AR	(SR<AR) RT: 3 to 11 pkt/s DT: 1 to 6 pkt/s
Category 4: Very Extreme Cases	RT > DT RT: 6 to 12 pkt/s DT: 5 to 9 pkt/s	SR<AR	(SR<AR) RT: 4 to15 pkt/s DT:1 to 9 pkt/s

Category 1 until 4 defines the QoS condition of the network, starting from steady condition up to the condition when the network performs very poorly due to limited buffer space and high traffic rates. For example, the drastic reduction of the delivery ratio after the value of 16 pkt/s is due to the network congestion, as confirmed also by the increment of the drop rate for the corresponding values of arrival rate. Traffic drop, which increases as the arrival rate increases, is defined as the number of packets dropped due to buffer overflow. This increment is almost linear till the arrival rate 14 pkt/s and after the value of 16pkt/s it jumps up from 60% to 90%. This results in a huge reduction of delivery ratio for the corresponding arrival rate.

6.4.2 Training Samples

The system is trained using different training sets. The training sets are formulated to represent different data coverage. The aim is to compare the neural network performance using different training sets.

The training sets are distinguished by having different data coverage of the network variation. They represent the poor, moderate and good training data, which contain 30%, 60% and 100% data coverage, respectively. For example, the poor training data, only consists of part of the QoS condition category (Category 1 and Category 2 explained in Section 6.4.1). The initial neural network training includes this data set. On the other hand, the good training data set consists of all of the 4 categories. This data set is used in the later stages of the training.

Table 6-2 Definition of Training Sets

Training Set Coverage/ No. of samples	**No. of cases selected from WSN Network Performance Categories**			
	Category 1	**Category 2**	**Category 3**	**Category 4**
30% /23	20	3	0	0
60% / 47	30	5	12	0
100% /78	36	9	21	12

Within the MATLAB environment, the two variables that are loaded from the data set are *QoSConditionInputs* and *AdjustmentParameterTarget*. They represent the input data (3xm matrix) and target data (2xm matrix).

The *QoSConditionInputs* is a data matrix of 3 x *m* defining 3 QoS conditions under *m* different network cases. The QoS condition values are data from network simulation statistics taken every T interval at the end of a monitoring cycle. They represent the neural network input values as described in Section 6.2, which are:

1. packet delay
2. delivery ratio and
3. buffer usage

The *AdjustmentParameterTarget* is a matrix containing the associated adjustment values for the network cases. The data set is used to train the neural network to estimate the adjustment parameters from the QoS conditions. As explained in Section 6.2, the 2 x *m* matrix of the adjustment parameters to be estimated given the inputs are:

1. source data rate
2. buffer size

The three sets of training which have been conducted are summarized in Table 6-3. For instance, for the initial training which is Training 1, the input is a 3 x 23 matrix representing 23 samples of 3 elements. Furthermore, larger data sets are used for retraining, which are Training 2 and Training 3. Both of them consist of better coverage of QoS condition categories. In addition, validation and test samples are also set aside in the simulation training. As shown in the table, 70% of the Training 3 data (54 samples) was used for the retraining of the neural network, 15% (12 samples) was used for the validation and the remaining 15% (12 samples) was used for the testing of the network.

Table 6-3 Input and output data matrix

Training /Coverage	**Input data matrix**	**Target data matrix**	**70% Training samples**	**15% Validation samples**	**15% Testing samples**
Training 1 (30%)	3 x 23	2 x 23	17	3	3
Training 2 (60%)	3 x 47	2 x 47	33	7	7
Training 3 (100%)	3 x 78	2 x 78	54	12	12

The following codes are used to design a fitting neural network with 10 hidden neurons with this data at the command line.

```
LOAD QoS_dataset.MAT; //loads these two variables
[x,t] = QoS_dataset; //loads the inputs and targets into chosen
variables
net = fitnet(10); //Function fitting neural network
net = train(net,x,t);
view(net)
y = net(x);
```

The results from the learning process are observed through the error rate. The aim is to achieve a low error rate which is within the tolerable value. Function fitting is the process of training a neural network on a set of inputs in order to produce an associated set of target outputs. Once the neural network has fit the data, it forms a generalization of the input-output relationship and can be used to generate outputs for inputs it was not trained on. The dataset is used to train a neural network to estimate the relationship between the two sets of data.

6.5 Neural Network Training Results

This section presents the results concerning the training of the neural network. In the following, the neural network results after training is completed are discussed. The training performance are indicated using the mean square error (MSE) and regression (R). Regression values measure the correlation between outputs and targets. A regression value of 1 means a close relationship, whereas 0 indicates a random relationship.

6.5.1 Training and Validation Performance

Training automatically stops when generalization stops improving, as indicated by an increase in the MSE of the validation samples. The MSE is the average squared difference between outputs and targets. While lower values are better, zero MSE means no error. The following tables show the results in terms of MSE values for the three training sets.

Table 6-4 MSE values results

	Training 1 30%	**Training 2 60%**	**Training 3 100%**
		MSE	
Training	1.97656	1.50532	2.26030
Validation	4.46077	5.04052	5.38405
Testing	9.04224	19.85661	5.02986

The performance progress of all the trainings are shown in Figure 6-4 until 6-6. The property ‘best epoch’ indicates the iteration at which the validation performance reached a minimum. Figure 6-4 shows that validation stops at Epoch 14 when MSE has reached a minimum value. The training continued for 6 more iterations before it stopped.

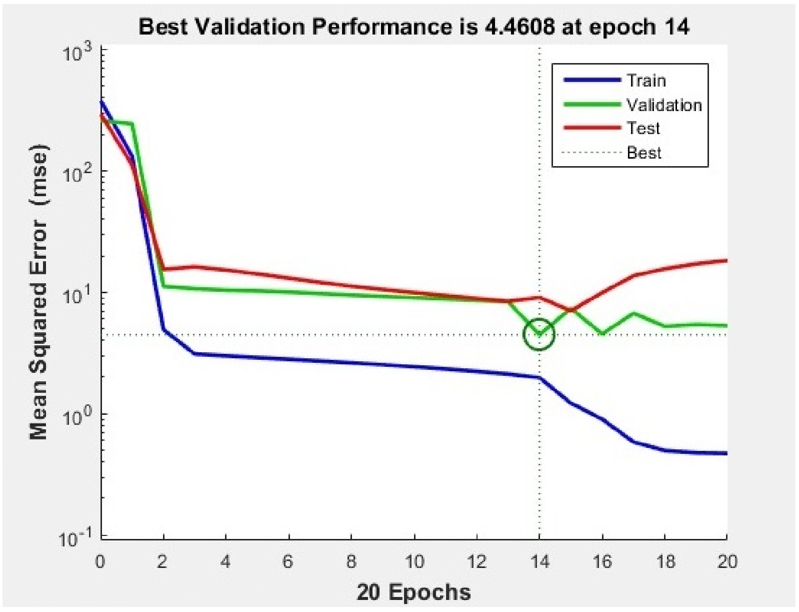


Figure 6-4 Training performance of Training 1 (Data Set of 30% coverage)

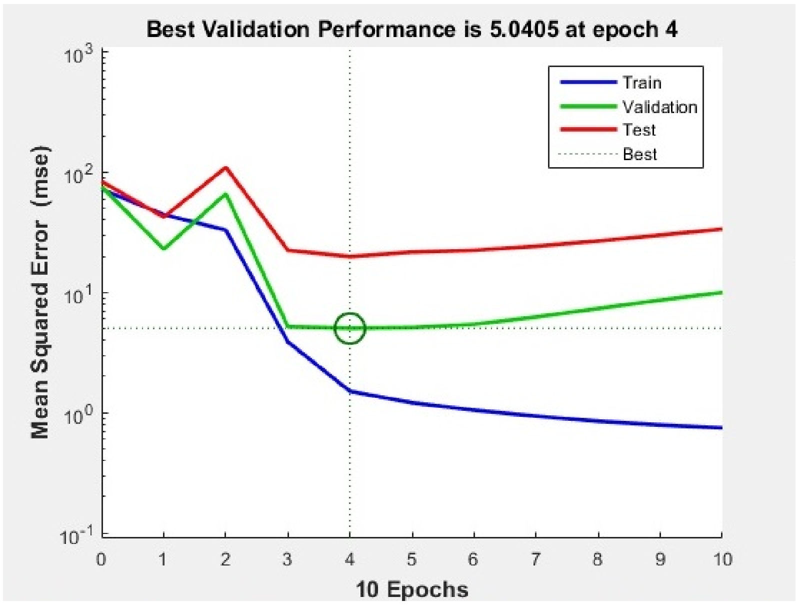


Figure 6-5 Training performance of Training 2 (Data Set of 60% coverage)

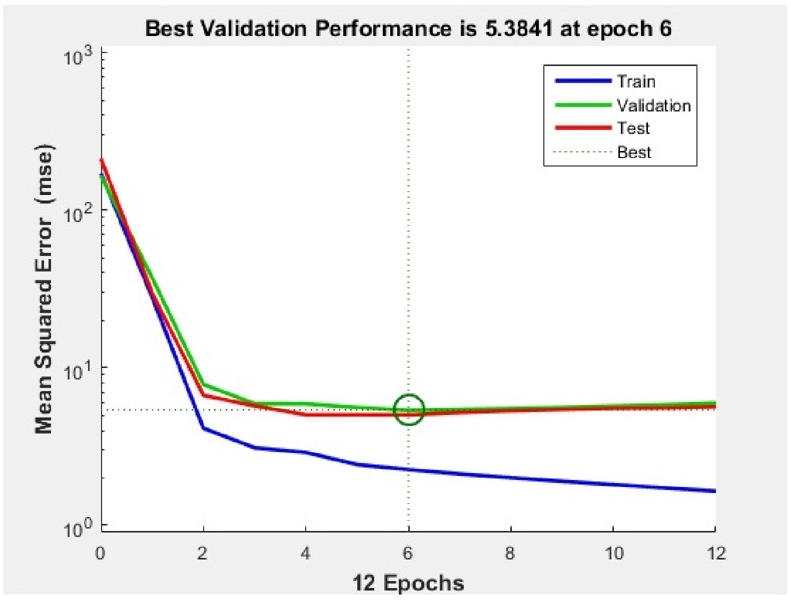


Figure 6-6 Training performance of Training 3 (Data Set of 100% coverage)

Figure 6-4 indicates that the training using 30% coverage of adaptive QoS cases generated poor results, as shown by the significant difference on training validation and test curves. Improvements are shown during retraining when larger data sets are used. Figure 6-5 and Figure 6-6 show improvement on precise data fittings with trainings using 30% and 60% coverage. The validation and test curves are very similar, particularly in Training 3 in which the data set has complete coverage of all QoS conditions' categories as defined in Section 6.4.2. The results show that by retraining using a larger set of training data, the network generalizes well to the new data and produces sufficiently accurate results.

6.5.2 Neural Network Training Regression

The next step is validating the network through the regression plot. Regression plot represents the relationship between the outputs of the network and the targets. If the training was good, the network outputs and the targets would be similar. A regression value of 1 means a close relationship, whereas a zero value indicates a random relationship.

The following tables show the results in terms of regression values for the network trainings.

Table 6-5 Regression values results

	Training 1 30%	Training 2 60%	Training 3 100%
		Regression	
Training	0.99490	0.99571	0.99353
Validation	0.93900	0.98909	0.98496
Testing	0.96940	0.90392	0.98518

The associated regression plots for training, testing and validation for Training 1, 2 and 3 are shown in Figure 6-7, Figure 6-8 and Figure 6-9, respectively.

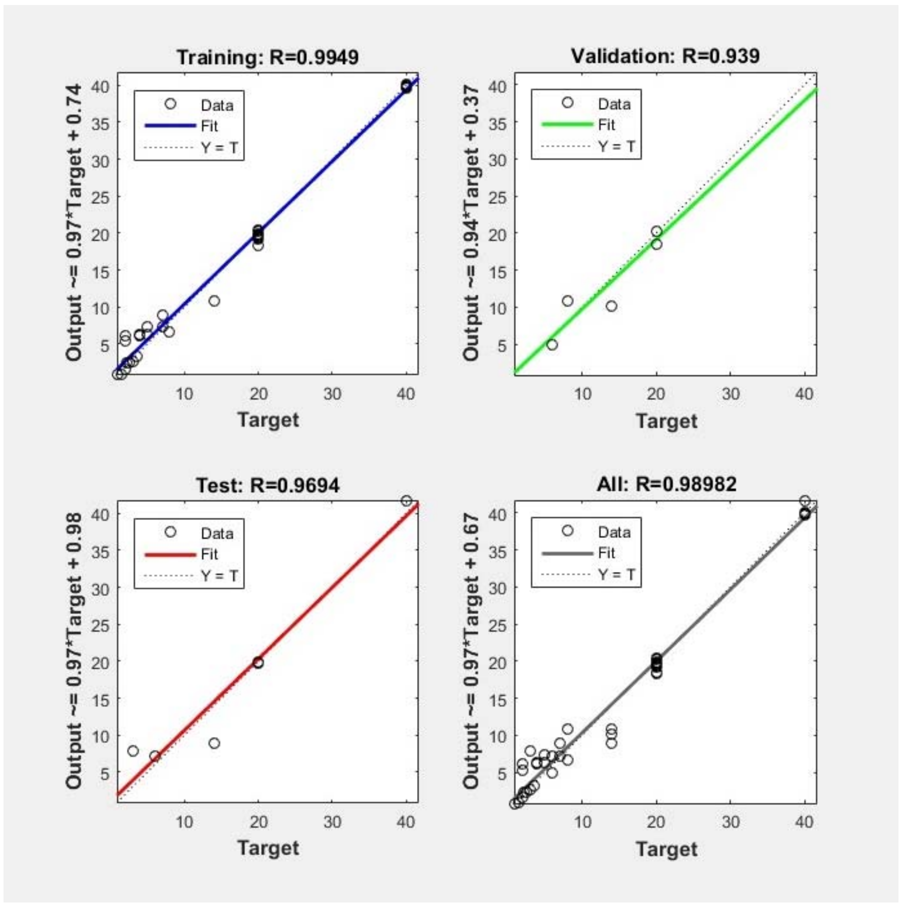


Figure 6-7 Regression values of Training 1 (Data Set of 30% coverage)

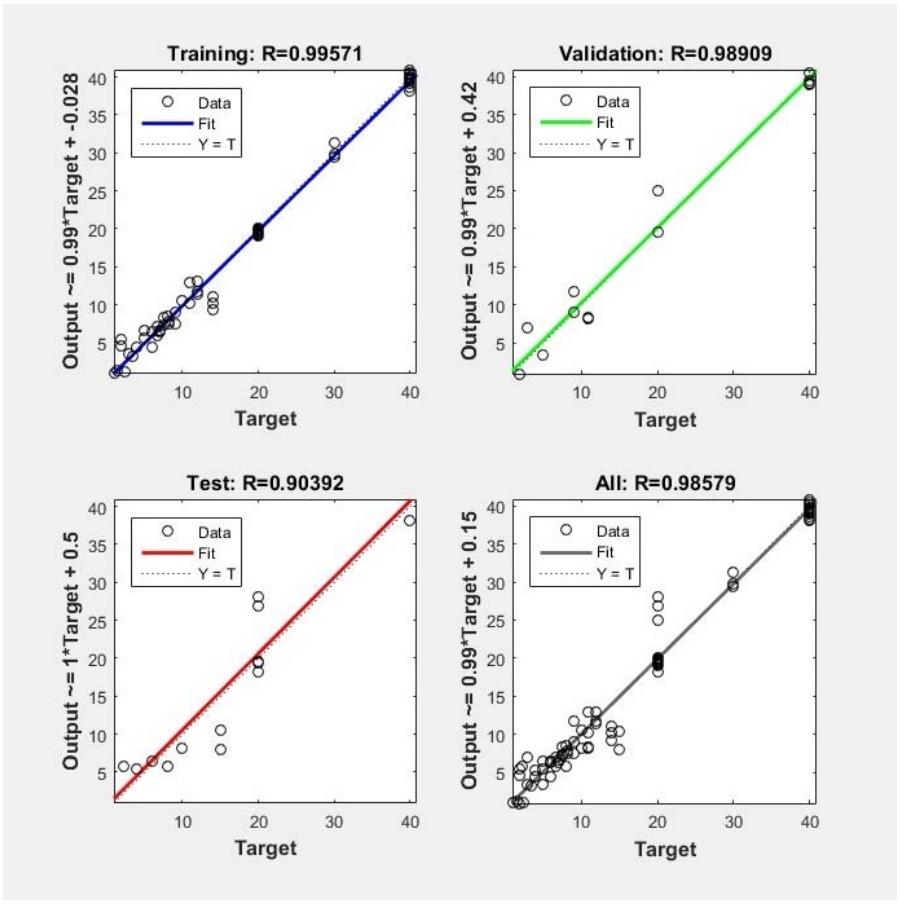


Figure 6-8 Regression values of Training 2 (Data Set of 60% coverage)

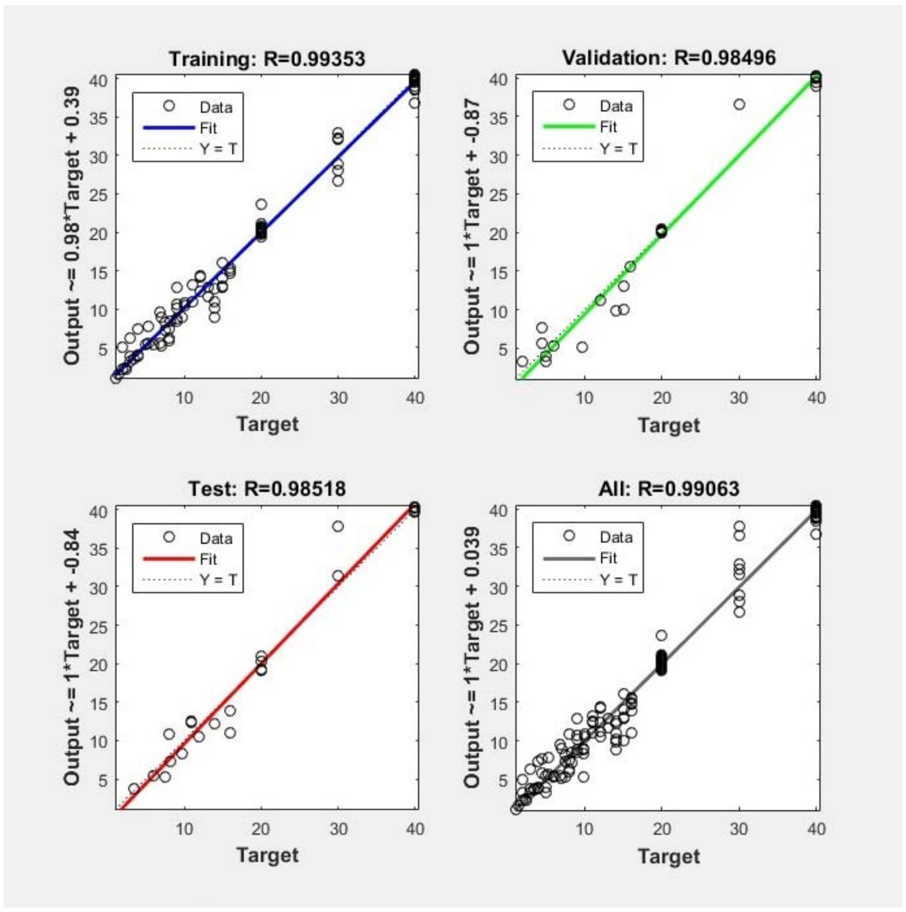


Figure 6-9 Regression values of Training 3 (Data Set of 100% coverage)

The dashed line in each plot represents the ideal result – outputs = targets. The solid line represents the best fit linear regression between the outputs and the targets. As the regression value is an indication of the relationship between the outputs and the targets, the value of R=1 shows that there is an exact linear relationship between the outputs and the targets. As shown in Figure 6-9, the training with complete QoS network categories results in a good fit. The overall R value also has the highest value as compared to Training 1 and Training 2 which are trained using smaller data sets and incomplete coverage of QoS condition categories.

6.5.3 Error Distribution

Figure 6-10 up to Figure 6-12 show the distribution of errors for the training, validation and test sets for all neural network trainings.

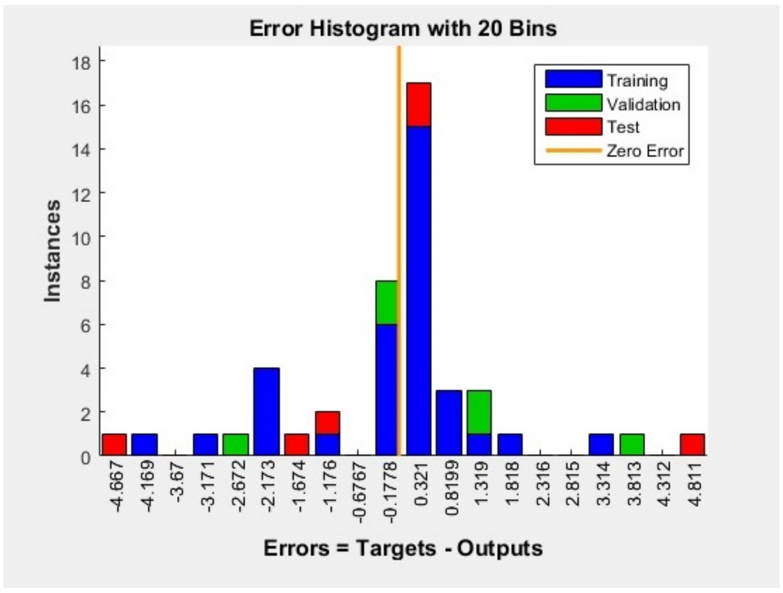


Figure 6-10 Error distribution of Training 1

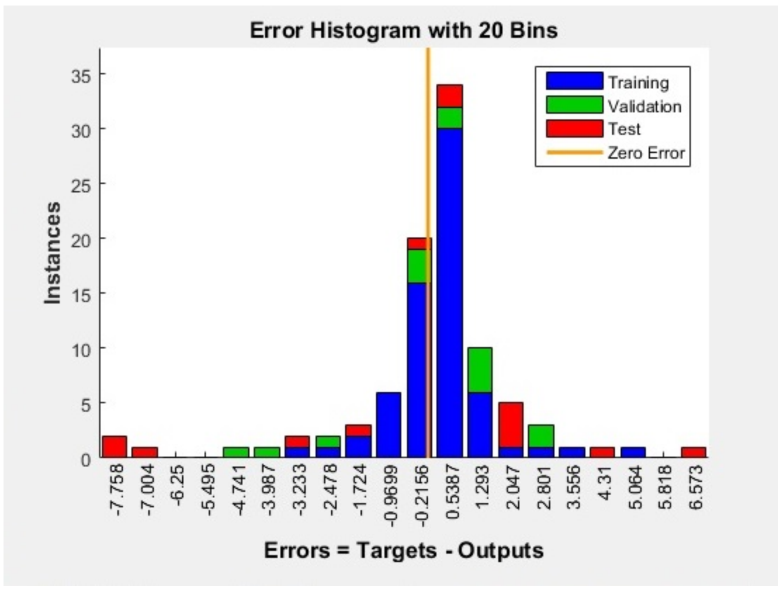


Figure 6-11 Error distribution of Training 2

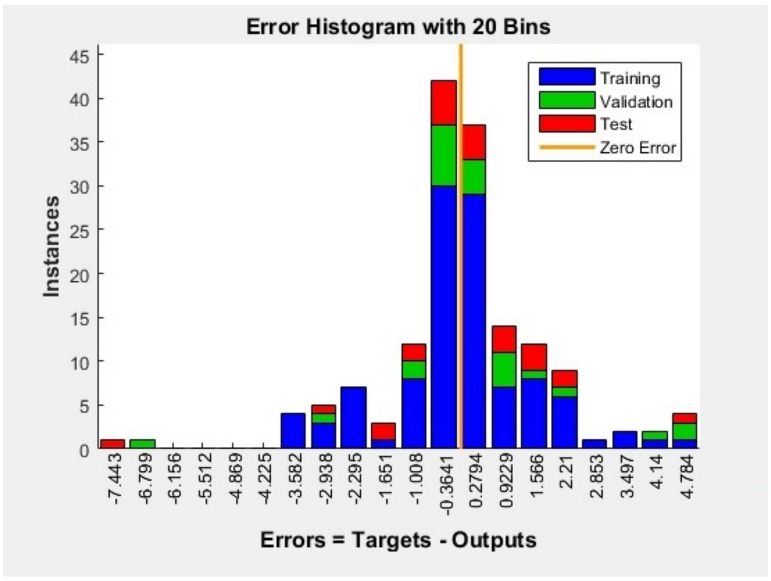


Figure 6-12 Error distribution of Training 3

The figures show that the data fitting errors for Training 2 and Training 3 with a larger data set have better error distribution. However, they are not distributed within a reasonably good range around zero. This can be further improved by using larger data sets or higher number of neurons. Nevertheless, the drop in error rate in the retraining indicates that the training has matured gradually.

6.6 Summary

The neural network learning activities in MATLAB provides a comprehensive validation to the proposed AQoS algorithm. Using the simulated data sets and the simulation results gained from the Riverbed Modeler, a system training is conducted in MATLAB neural network tools. In addition, to provide a value-added adaptive feature of the QoS framework, as well as to deliver similar results of initial observation from the modelling, system training is conducted using neural network on MATLAB.

The training samples have been selected to ensure that the network condition covers the optimum possibility of parameters combinations, hence providing sufficient and proper training during the neural network learning. Network conditions are categorised into 4 categories, namely Settled Cases, High Traffic Intensity Cases, Extreme Cases and Very Extreme Cases. The neural network is trained using different data sets, to show the impact of using larger data set with better coverage of QoS cases.

As shown by the results, when larger data set with complete coverage of QoS conditions are used to train the network, the values of MSE and R has improved. This means that the data fitting can be quite precise with larger data set. Furthermore, the drop in error rate in the retraining indicates that the training has mature gradually.

It is expected that a WSN with heterogeneous data which also involves dynamic traffic will evolve over time and therefore introduce complexity to the system as the network grows in size. Additional types of sensor data may also be needed to accommodate the WSN application’s requirements. Therefore, the learning capabilities in ADHERE can facilitate the optimisation of the QoS framework’s performance by accommodating the QoS requirements of the network when the traffic is dynamic, and also when complex network behaviour takes place.

Chapter 7: QoS Model Implementation and Physical Experiment

7.1 Introduction

This chapter presents the implementation of the ADHERE QoS model on a physical testbed. The physical experimentation serves as a means of verification and validation to the computer simulation environment. This environment, which enables the software adaptation of the IoT-based-WSN, was thoroughly discussed in an earlier publication in [6]. It was highlighted that there is a need for an environment which delivers constant monitoring of a WSN-IoT network performance through assessment of the QoS, and which ultimately makes required adjustments to a physical network's configuration. Based on historical sensor data captured from the physical network, the chosen approach involves modular organization that allows implementation and analysis of QoS for WSN.

The architecture of the test environment is first presented in Section 7.2. The test environment offers a system that is flexible enough to be capable of reacting to dynamic changes of process demands. By analysing the historical data in the background on a network simulator or virtual network, physical network performance can be predicted. This allows for estimation of the adjustments needed, which are necessary to improve the network performance without disturbing the physical system. Section 7.3 discusses the system implementation of the proposed architecture. Furthermore, Section 7.4 reports the case study and applicability of an adaptive QoS model on the system. The comparison between simulation and physical experiment is also discussed in Section 7.4.

7.2 Testbed Operational Architecture

Figure 7-1 depicts the proposed architecture of the test environment [4], which is established with a PSN, an IoT interface and a remote server or cloud support.

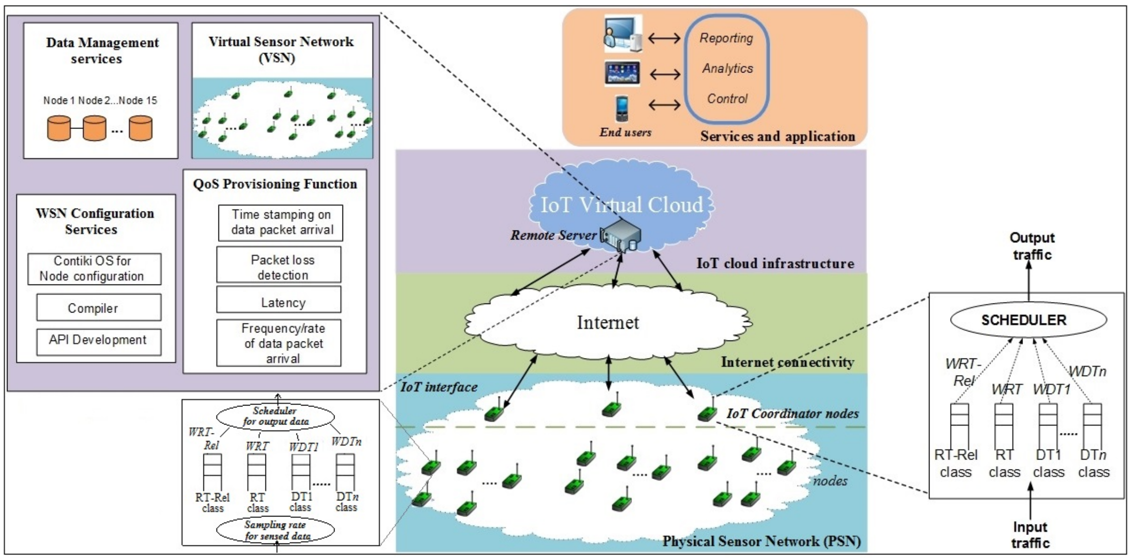


Figure 7-1 Testbed architecture

The PSN is organized in multiple physical sensor cloud (PSC) formations. There is an IoT coordinator for each PSC which serves as the access point between the Internet and the PSC. The remote server or the cloud hosts the data storage and necessary environment for network virtualization, intelligence and other processing that support decision-making for managing the PSN operation.

Upon receiving the sensor data, the IoT access point pushes them to the cloud server, which accommodates the database for storing network simulation tools for modelling the virtual sensor network (VSN), related historical data and a mathematical modelling tool for hosting QoS evaluation models. Moreover, a QoS controller program, which runs the QoS algorithm is part of the QoS Provisioning Function. It enables analytical activities, whilst suggesting alterations to be made on the sensor nodes' operational parameters to maintain the network's QoS condition. The alteration is done by remotely-configuring the physical leaf nodes.

Figure 7-1 also depicts a queuing model for regulating the buffer for data packets whilst in queue to get transmitted at the IoT access point. When data arrives at the IoT access point, it will be allocated to different buffer queues according to the service differentiation which runs within the IoT access point. As depicted by the figure, in the IoT access point, different traffic types are buffered in separate queues.

The VSN which resides within the virtualization environment offered by the network simulator is an exact replication of the nodes within the PSN. It mimics the network data flow which occurs in the actual physical network. Such cloud-level virtualization offers an environment for testing the network performance when subjected to different QoS parameters. These parameters are generated by the ADHERE QoS algorithms residing within the QoS provisioning function, in an iterative manner.

Additionally, to increase the level of accuracy of the obtained simulation results, historical real-life PSN data accumulated in the remote server can be reused on the VSN for simulation purposes. This helps the process of identifying and converging upon the required QoS parameters in an adaptive way by closing the loop between VSN and PSN.

PSN data within the database acts as the input for the QoS model residing in the QoS model simulators and the network. SQL queries are performed by the simulation tool to feed this data to the application layer which belongs to the simulation model. Furthermore, the historical data's associated time stamps can be used by the simulator to calculate the service time or other performance measures. Then, the network performance indicators that have been derived from the network simulator are passed to the analytical tool for further QoS analysis.

The decisions and control mechanism to support the ADHERE QoS are computed using MATLAB functions. The mathematical analysis feature also supports high-level analysis, which in turn makes future network complexity more manageable. MATLAB does offer various mathematical functions that make the computation easier. MATLAB MX permits use of MATLAB functions in C programs [85]. This results in more efficient executable functions in either the network edge or the Cloud.

The co-simulation approach may also ensure QoS for the virtualized environments, especially when the PSN application workload increases [101]. Therefore, this structure has great potential for solving complexity and for further analysis, while it acts as part of the WSN system operational management.

The performance of the network is continuously monitored, and the QoS condition is determined via the QoS provisioning algorithm. When an adjustment decision has been made, the adjustment parameters (e.g. buffer size and data rate) are written in a C header file, which will be used by the operating system to reconfigure the PSN. The test environment architecture is flexible in a way that other parameters related to other potential WSN applications can be adopted. As such, this closed loop conceptual framework supports the architecture in imposing the necessary dynamic changes on the PSN adaptively.

7.3 System Implementation

This section presents the system implementation of the proposed architecture. The selection of software and hardware tools to build the system is discussed in this section. The test environment has been established by the research team of the Sensor Network and Smart Environment Research Centre (SeNSe) [90] laboratory. The effort given by the SeNSe team aims at enabling the implementation of several use-cases on the same testbed.

7.3.1 Physical Sensor Network

As shown by Figure 7-2, in the PSN setup [3], CC2538 controllers are deployed on a functional basis and at different locations within the SeNSe laboratory. A total of 16 nodes were deployed, from which fifteen nodes act as end devices and one node acts as the IoT access point. The end devices sense multiple sensor data, namely, temperature, light and received signal strength indicator (RSSI) sensing. The sensed values are forwarded to the upper tier of the server database over the Internet.

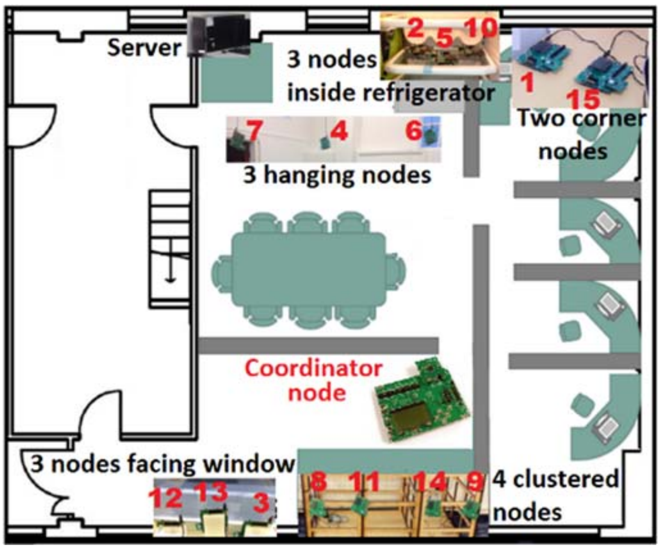


Figure 7-2 SeNSe laboratory plan showing the deployment of the sensor nodes

In the proposed architectural organization, each of the three sensor data captured by the end devices represents different traffic types with different priorities by means of packet remarking. Upon transmission of PSN data to the IoT access point, the data is then relayed to the cloud server over the Internet, wherein the incoming data is stored and organized by the Data Management Services Unit in its database.

The MAC protocol of the nodes is implemented with polling based on Time Division Multiple Access (TDMA). The end devices transmit their data to the IoT access point only when polled. In the current setup, new incoming data automatically updates the server database at intervals of 90 seconds. Table 7-1 describes the pseudo codes for both the IoT access point and the end devices.

Table 7-1 Algorithm pseudo codes for operation of IoT access point and end nodes

```
Pseudo code for IoT access point node

While TRUE {
Initialization:          Set and initialize data storage arrays of size n
                           light[n] =  {$0_1,0_2,….0_n$}
                           temp[n] = {$0_1,0_2,….0_n$}
                           rssi[n] =   {$0_1,0_2,….0_n$}
Polling request:         For each 'n' end devices: Transmit polling counter messages
                         (1, 2,….., n-1,n) to all the end devices.
INPUT:
Reception:               Keep storing received node data to arrays defined for the 3 sensor
                         variables
                         Repeat 'n' times{
                           light[n] =  {$l_1,l_2,…l_n$}
                           temp[n] = {$t_1,t_2,….t_n$}
                           rssi[n] =   {$r_1,r_2,…r_n$}
                         }
OUTPUT:
Serial data string:      Send sensor data received via serial output
                         For each 'n' end devices
                         Node ID: light[l], temperature[t], rssi[r];
Delay:                   Wait for 45 seconds;
}

Pseudo code for end node

While TRUE {
- Sampling:              Sense the 3 sensor values – light, temp and rssi
- Reception:             Receive polling counter messages i.e. 1,2,….,n-1,n
- Check condition:       Check if the condition (Counter = Node ID) is satisfied
        If Transmit flag is set {
                         Transmit Flag = TRUE;
- Transmission:          Transmit data array to IoT access point;
                           light[n] =  {$l_1,l_2,…l_n$}
                           temp[n] = {$t_1,t_2,….t_n$}
                           rssi[n] =   {$r_1,r_2,…r_n$}
                         Transmit Flag = FALSE;
                         }
}
```

Grouping of nodes on a functional basis renders a clear demarcation between the different sets of groups of nodes. This arrangement facilitates clear and distinguishable data plots on the webpage. During the course of our physical experimentation (through data logging), grouping of the nodes in separate geographical locations emerged as a viable option.

7.3.2 Test Environment Sensing Data

The sensor data plots shown in Figure 7-3 represent the samples of the signal trends of the captured data, i.e., ambient light, temperature and the RSSI values from a selection of nodes in the network.

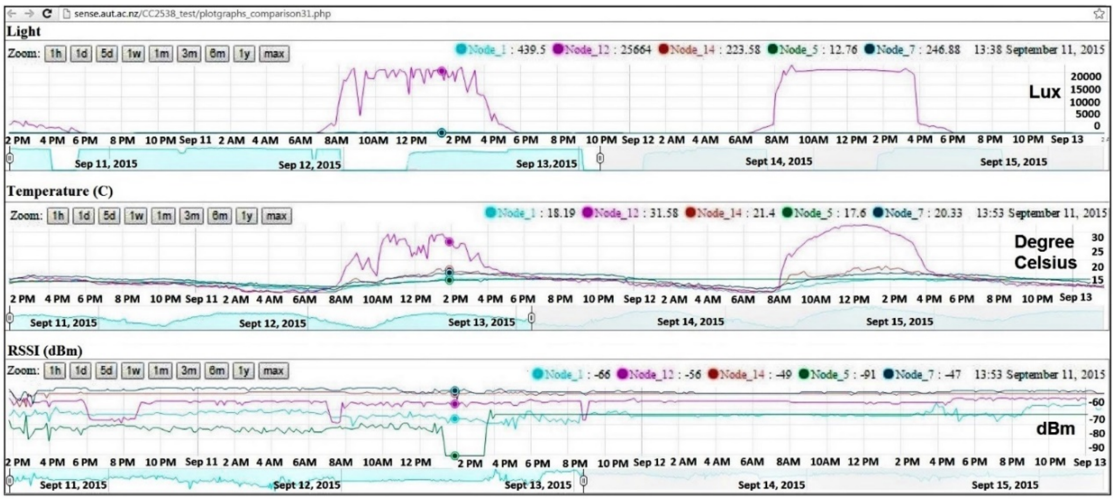


Figure 7-3 Sensor data representation from the database

The sensory data trends contain the information associated with the layout of the nodes deployment, as shown by Figure 7-2. The figure shows the readings for three consecutive days. The observation from the light sensor reading indicates that the light values are high during sunny parts of the day, particularly on the nodes facing the window. In contrast, a lower light reading is observed from the nodes facing the lab. It is also shown that the temperature increases during the day for those nodes facing the window, hence reflecting the internal heat inside the controller chips. Lastly, the RSSI data trend exhibits some activities during the day resulting from people movement within the lab. On the contrary, the RSSI values during the night are observed to be more stable.

7.3.3 Server Implementation

The server has been implemented to accommodate the major functions as discussed in Section 7.2. Figure 7-4 illustrates the inter-working of the three servers, namely, an application server, a database server and a web server.

The application server is installed with Contiki OS, mainly to carry out the physical and virtual node configuration. Contiki OS compiles the scheduler to perform network operations to the target node. For modelling and provisioning purposes, Riverbed Modeler also resides within the application server. Network simulators such as Riverbed Modeler allow replication of a PSN to form a VSN. This serves as abstraction levels for testing applications or protocols prior to execution on real hardware devices. The Riverbed Modeler also offers support for simulation of entire networks and access to model parameters, hence having a significant effect on the simulation accuracy.

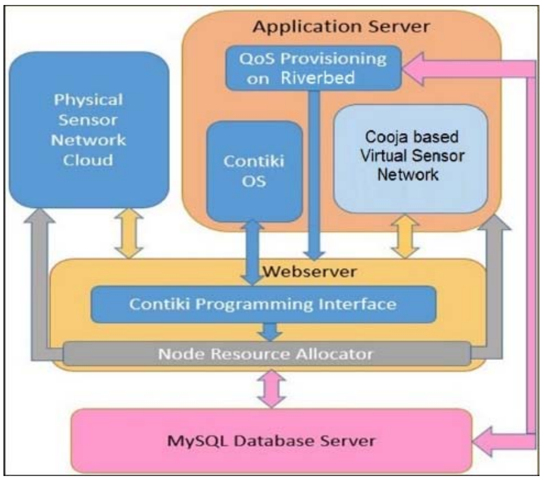

Application Server
QoS Provisioning on Riverbed
Physical Sensor Network Cloud
Contiki OS
Cooja based Virtual Sensor Network
Webserver
Contiki Programming Interface
Node Resource Allocator
MySQL Database Server

Figure 7-4 Remote server architecture - Inter-relationship between application server, web server and database server

In order to to run the QoS Provisioning module, MATLAB is also installed in the Application Server. Therefore, the co-simulation between Riverbed Modeler and MATLAB to support the QoS Provisioning takes place in the Application Server. While users may benefit from Riverbed Modeler as a tool that strives for closer representation to network devices, MATLAB serves as the control mechanism and decisions to support the ADHERE QoS. Future network complexity become more manageable as MATLAB also supports high level analysis. The co-simulation approach can also ensure QoS for the virtualized environments, especially when the PSN application workload increases [101]. Hence this structure offers the environment for analysis and solving complex computational requirements.

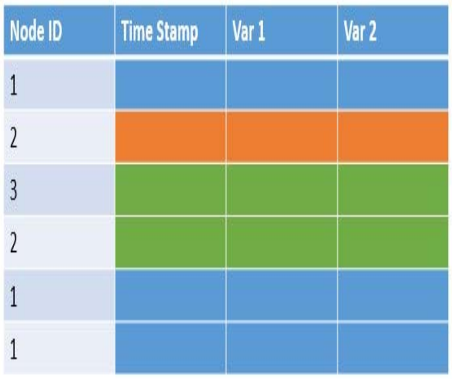
The application server also contains a Contiki-based Cooja simulator, which is a useful tool for the development of Contiki-based sensor network implementations. Besides serving as a testing environment for code testing [102], it observes the impact of modifying network parameters such as data rate and network protocol on network performance and power consumption prior to the implementation on PSN hardware.

The web server is the point of entry for the sensor data. Using a REST API , it receives data from the IoT access point. The REST API establishes communication with the users and exchanges information between the application server and the database. The user selects the nodes using GET and POST. The server uses PHP for scripting, while Python is used by the client side for collecting sensor data. These specific scripting languages are chosen due to a minimal required learning curve for the researchers. Data is forwarded by the client to the web server. The web server then forwards the data to the MySQL database for processing and determining sensor node statistics using REST APIs. The QoS provisioning module predicts QoS parameters that serve as inputs to the web server that use the Contiki programming interface to update the physical and virtual nodes. In the following, the implementations of the main functional blocks of the server are discussed.

For data management services, MySQL is implemented within the database server as the Relational Database Management System (RDBMS). The PSN data is retrieved by the database queries from the incoming data packet. Having organized the database such that each node has its own table, the SQL queries query the table to get information from the node.

The collected data packets are time stamped and stored in rows within the database. As the size of the database increases, the latency of data retrieval increases proportionally. This spatio-temporal increase has a performance implication. Our findings [3] show that data retrieval can be maintained at a minimal rate. This reflects the independence of retrieval rate on the number of nodes when individual tables for every sensor node are used.

Figure 7-6 shows a database table resulted in *O(n log (n))* complexity for data retrieval. Two operations are required: a search operation of the node and next a read operation to fetch the data. The temporal requirement for the search operation was improved from *O(log (n))* to *O(1)* by splitting the single table into multiple tables as illustrated in Figure 7-6. As shown in the figure, each table represents a specific physical or virtual node. As the complete node data is encapsulated in individual tables, the overall retrieval time is reduced from *O(n log (n))* to *O(n)*. This has also supported the one-to-one representation of the nodes within the PSN.



Node ID	Time Stamp	Var 1	Var 2
1	🟦	🟦	🟦
2	🟧	🟧	🟧
3	🟩	🟩	🟩
2	🟩	🟩	🟩
1	🟦	🟦	🟦
1	🟦	🟦	🟦

Figure 7-5 Single table database with NodeID and Variables

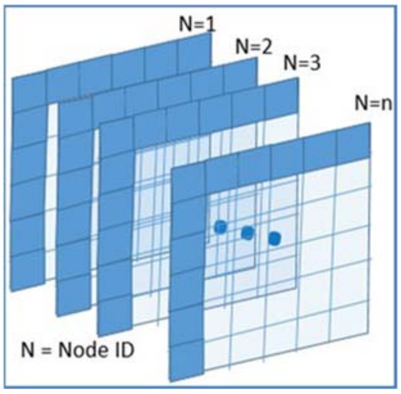


Figure 7-6 Node ID as database tables

7.3.4 ADHERE QoS Model Implementation

The implementation of the proposed ADHERE QoS concept is illustrated by Figure 7-7 [3]. PSN data within the database acts as the input for the QoS model residing within the Riverbed Modeler. The simulation tool performs SQL queries to feed the PSN historical data to the application layer belonging to the simulation model. In addition, the associated time stamps of the historical data can be used by the simulator to compute the service time or other performance measures.

.

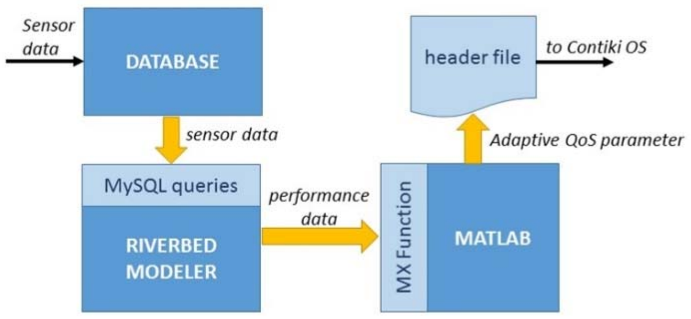


Figure 7-7 ADHERE concept implementation

Then, the network performance derived from the model hosted by the Riverbed Modeler is passed to the QoS analysis model hosted by MATLAB. MATLAB uses the performance data to analyze the QoS condition and identify the corresponding QoS parameters. In the system implementation, the co-simulation between Riverbed Modeler and MATLAB is set up using MATLAB MX functions and Riverbed Modeler APIs [85], as shown by Figure 7-7.

An example of ADHERE algorithm is shown in Table 7-2. The objective of the QoS model is to avoid congestion in the network by determining the traffic's data rate which is appropriate to the network condition. For instance, congestion detection in WSNs has been proposed using indicators such as buffer occupancy, queue length [60, 66], service time of packets [62], as well as the ratio of service time to inter-arrival time of the packets between the intermediary devices [63].

Table 7-2 Pseudo Code of Adaptive QoS Algorithm

```
Define QoS requirement:
                                RT packet delay bound, DT packet loss tolerance
Initialization:                 Identify data rate RT_datarate, DT_datarate,
                                Set required buffer size RT_buffer, DT_buffer
While TRUE {
Data transmission:              Transmit data array to IoT access point
Queue Algorithm…                Arrival and departure of packets in IoT access point buffer using
                                weighted priority queuing scheduling policy
QoS Condition Monitoring and Adjustment (Every T interval)
- Congestion Control Unit
                                Calculate congestion index
                                congestion_index = service_rate/arrival_rate
- Rate Adjustment Unit:
                                Adjust new RT data rate based on congestion index
                                if congestion_index>1
                                  RT_datarate = new_RT_datarate($_{priority, datarate}$)
                                else if congestion_index <1
                                  RT_datarate = new_RT_datarate($_{priority,datarate}$)
                                else
                                Adjust new DT data rate based on congestion index
                                if congestion_index>1
                                  DT_datarate = new_DT_datarate($_{priority, datarate}$)
                                else if congestion index <1
                                  DT_datarate = new_DT_datarate($_{priority,datarate}$)
                                Else
- Buffer Adjustment Unit:
                                Adjust buffer size based on new data rate
                                  RT_buffer ← min (size$_{RT}$)
                                  DT_buffer ← min(size$_{DT}$)
OUTPUT:
Write new value                 In c header file (to be read by OS for reconfiguration)
                                  RT_datarate, DT_datarate
                                  RT_buffer, DT_buffer
}
```

The implemented algorithm supports two types of traffic classes, i.e., real-time (RT) traffic and delay-tolerant (DT) traffic. In the initialization phase, the algorithm identifies the data rate of the traffic sources and their QoS requirements, namely the RT packet's tolerable delay and the DT packet's tolerable packet loss. The algorithm also estimates the required buffer size [25] of different traffic types based on this information. The required buffer size is estimated to ensure that the reliability-constraint DT packets can take up more space in the IoT access point's buffer. Therefore, in the QoS model, more bandwidth is allocated to RT traffic to meet low delay. In addition, more buffer storage is allocated to DT traffic to avoid buffer overflow which may result in higher traffic drop.

The rate adjustment unit calculates the new data rate of the traffic source, using the results of the congestion index [63] and the predefined source traffic priority [28]. Then, the adjustment parameter, i.e., data rate and buffer size, is passed to the WSN Configuration Services, to perform the physical nodes reconfiguration. This approach will ensure that the QoS requirements for different traffic classes are met while maintaining steady queues in the IoT access point buffer. Another aim is to ensure that allocated buffer resources are utilized efficiently.

7.3.5 WSN Reconfiguration Services using Contiki OS

In the system implementation, the low power wireless transceivers are subjected to many reconfigurations over the Internet. In addition, a self-configuring architecture [103] which requires no human intervention for the reconfiguration and deployment of IoT applications is a desirable approach. To deal with such stringent requirements of an IoT-based environment, Contiki OS offers advantages such as flexibility, multi-tasking and concurrency [104].

As shown in Figure 7-8, the QoS parameters predicted by the QoS provisioning module are inputs to the web server that uses the flexible and open source Contiki programming interface residing within the SeNSe server to update the physical and virtual nodes with the QoS parameters. Contiki OS is specifically compiled for the physical nodes and the code is in C. The reconfiguration header file generated by the QoS provisioning module is integrated into the directory that is compiled into the hex file for the nodes. With the current implementation, the reconfiguration file is generated as a C header file that is created in the directory where the Contiki OS code resides. Currently, the reprogramming of the nodes with the new reconfiguration settings is done manually. However, part of our ongoing work involves a fully automated system, whereby the VSN would be automatically reconfigured based on the information it received from the physical network.

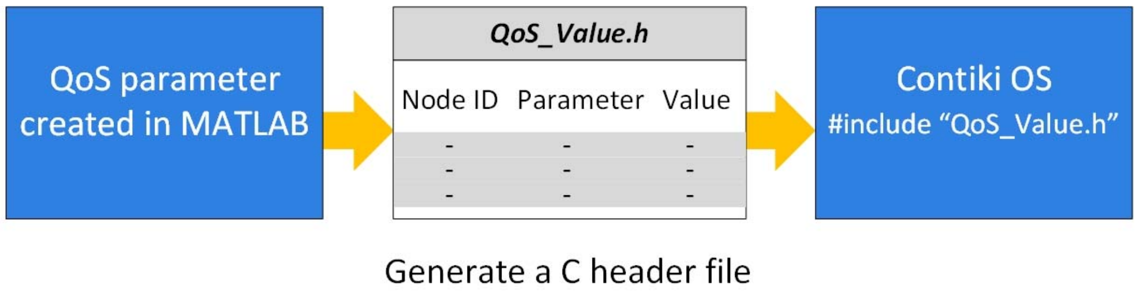


Figure 7-8 Passing QoS parameter to Contiki OS

7.4 Case Study and Results

Comparisons between PSN and simulation behaviour were conducted. The QoS performance of the network's heterogeneous data traffic was studied via the physical test environment experimentation and simulation of similar models. This activity also ensures a virtualization [6] with the closest replication of the PSN with the Riverbed Modeler simulation tool.

The communication between one IoT access point and the end devices in the PSN testbed has been established as a unicast on a single channel using Contiki OS. The polling-based network protocol was successfully implemented for the Contiki-based PSN setup. In the simulation, two types of traffic have been defined. RT and DT traffic flows are fed into the IoT access point buffer, which in turn serves the packets based on a predefined queuing policy. Furthermore, the queuing mechanism in the IoT access point is observed, and the QoS performance of the traffic is studied.

7.4.1 Comparison between Physical Sensor Network Experiment and Simulation - Effect of Service Rate

To ensure the comparability of both physical and simulated environments, an experiment with a single node communicating with a single IoT access point was done first. The aim is to identify the impact of service rate to network performance. Five cases have been set up in the PSN experiment – all of which are the simulated cases as indicated in Table 5-3 in Chapter 5. As discussed in Chapter 5, AR and SR are defined as the number of packets that arrived at the buffer and are served by the buffer per second, respectively. The AR in the PSN is set by defining the sampling rate of the traffic sources. This is also indicated in Table 7-1 within the pseudo code for the end nodes. Table 7-3 shows the exact parameters used in the simulation cases of SR=AR, SR>AR, and SR<AR as indicated in Table 5-3.

Table 7-3 Test cases to compare PSN experiment and simulation

Test Cases		**Service Rate (SR), pkt/s**	
		RT Traffic (Light)	**DT Traffic** (Temperature)
Case 1	SR = AR	8	5
Case 2	SR > AR	16	10
Case 3	SR > AR	9	7
Case 4	SR < AR	4	2
Case 5	SR < AR	7	4

To show association with a real WSN application, it is assumed that RT data is light sensory data [100], whereas DT data is represented by temperature sensory data. For all cases, the arrival rate of RT and DT are constant at 8 packets per second (pkt/s) and 5 pkt/s, respectively. Furthermore, a relatively small buffer size of 20 packets is also kept constant for each of both RT and DT buffers. As explained in Chapter 5, the small buffer size was set to allow seeing the results of different arrival and service rates more easily. As with the simulation, the PSN experiment was also conducted for a duration of 10 minutes.

The traffic performance in both PSN experiment and Riverbed Modeler has been observed. Figure 7-9 up to Figure 7-11 show the comparisons between both PSN experiment and computer simulation for all cases, i.e. Case 1 to Case 5. In both PSN experiments and simulation, it was observed that RT traffic and DT traffic are served according to the assigned serviced rates of the different queues in the buffer. As shown by Figure 7-9 to Figure 7-11, the trend of serviced packets against received packets at the buffer for all cases shows a good match. The difference between both environments was always smaller than 3% for Cases 1, 4 and 5. Furthermore, as shown in Figure 7-10, when SR>AR, there is very significant similarity in both PSN experiment and simulation.

In the PSN experiment, a small deviation occurs in terms of total packets received at the IoT access point. Meanwhile, the traffic generated in the simulation is always ideal based on theoretically-calculated values. For example, Figure 7-9 shows that the total RT packets received at the buffer was only 3680 packets in the PSN experiment, whereas the actual number of incoming packets shown by the simulation is 4800.

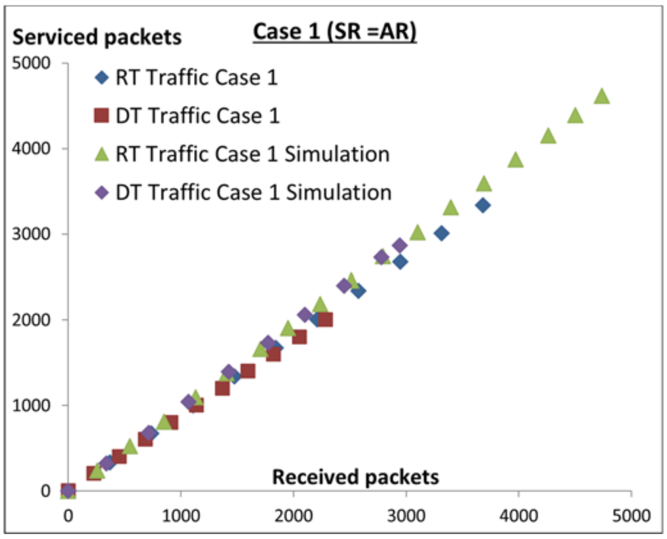


Figure 7-9 Comparison between PSN and simulation (serviced packets vs received packets) for SR=AR

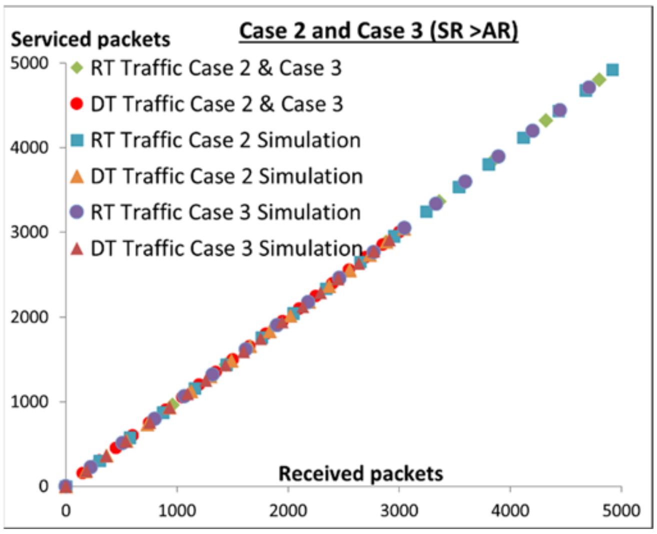


Figure 7-10 Comparison between PSN and simulation (serviced packets vs received packets) for SR>AR

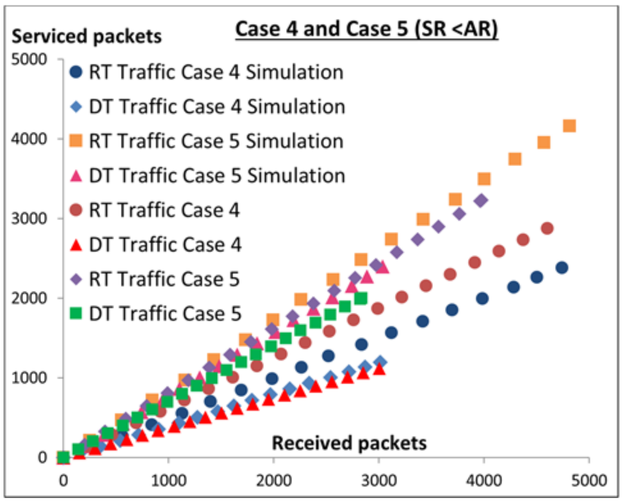


Figure 7-11 Comparison between PSN and simulation (serviced packets vs received packets) for SR<AR

The deviation of the packets received in the IoT access point may be due to the communication channel's stability between the sensor node and the IoT access point. However, the overall rate at which traffic is served in the RT and DT buffer queues through the course of both environments showed a good match. This validates the fact that the behaviour of the queue and node model in the simulation closely resembles the performance characteristics of an actual IoT access point.

7.4.2 Buffer Usage and Traffic Dropped

The buffer usage and traffic drop also showed a similar trend between both PSN experiments and simulation. Both Case 2 and Case 3, i.e. when SR > AR, have 100% delivery rate and zero buffer usage at all time. This indicates the traffic drop can be avoided when packets are served at a higher rate than the arrival rate. Figure 7-12 and Figure 7-13 depict the buffer usage and packet drop, respectively. Since Cases 2 and 3 have 100% delivery rate and zero buffer usage at all time, only the performance of the remaining cases, i.e. Cases 1, 4 and 5 are compared in the figures.

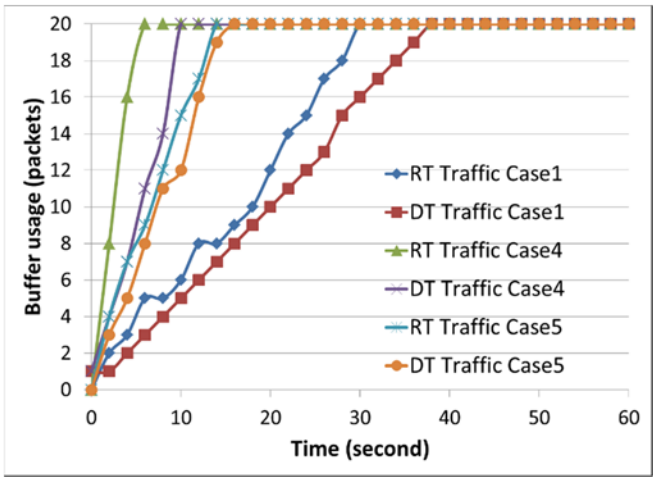


Figure 7-12 Buffer usage over 1-minute experiment

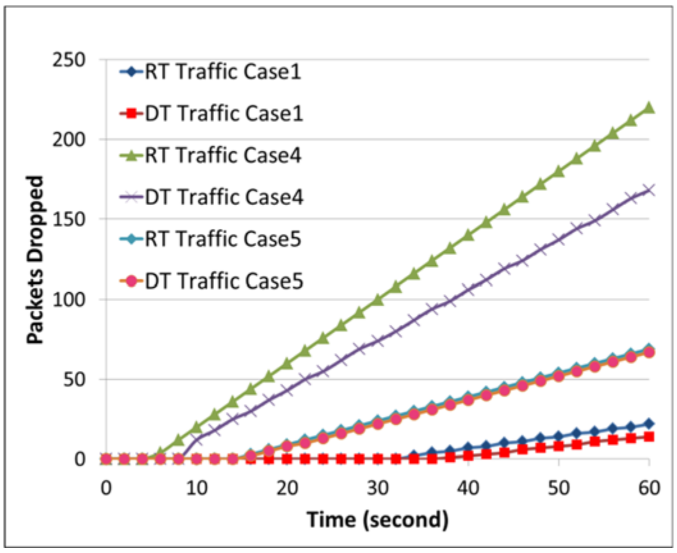


Figure 7-13 Packets dropped over 1-minute experiment

Figure 7-12 shows that when the packet rate for traffic source nodes is increased, the buffer usage of the IoT access point grows steadily. Figure 7-13 shows that once the buffer usage has reached its maximum capacity, packets start to be dropped. Consequently, more incoming packets will be dropped due to buffer overflow. This can be solved by adjusting the buffer size to match the requirements of the RT and DT traffic. Automated reconfigurations of the adjustment parameters such as the buffer size can be done within the PSN architecture presented in this chapter through continuous QoS monitoring.

7.5 Summary

In this chapter, the ADHERE QoS model is implemented on a physical testbed. The experimentation serves as a means of verification and validation to the computer simulation environment. The concept of virtualization has been presented and a conceptual organization targeting an adaptive QoS is presented. The architecture offers a system for interactions between the behaviour of a PSN and the necessary analysis in a virtual remote environment. The main contribution of the architecture is that it is capable of identifying the required adjustment for the PSN in order to enhance the QoS performance of the WSN applications.

The ADHERE QoS algorithm is implemented as a target application of the system. This represents a use case, which can provide the requested QoS for different traffic classes on the developed physical test environment. In the results section, comparisons of traffic QoS on the PSN and computer simulation are presented. The overall rate at which RT and DT traffic is served at the buffer queues through the course of both PSN experiments and computer simulations showed a good match. This validates the fact that the behaviour of the queue and node model in the simulation closely resembles the performance characteristics of an actual IoT access point. Therefore, the similarity of network performance within the simulation environment and the PSN experiment indicates the success of the virtualization concept.

The techniques for interoperability among the system components, namely PSN and a remote database, VSN and QoS provisioning unit, as well as Contiki OS and network reconfiguration are also discussed. With the encouraging preliminary outcomes between pairing components, the proposed architecture can be used as a test environment for experimentation involving other custom adaptive QoS parameters and real-time analysis of network performance. The co-simulation concept between network simulator and MATLAB is viable and this opens up more future research possibilities when more complexity takes place potentially at a business intelligence level.

Chapter 8: Conclusion and Future Work

8.1 Conclusion

The main motivation behind this thesis is the fact that traffic in WSNs represent two kinds of co-existing data packets: those with real-time constraints and those with reliability-constraints. By treating these packets that have varying QoS requirements differently, the needs of both packet types can be better met. Furthermore, nowadays the effort of connecting WSN to remote servers or clouds through the IoT inspires the formation of a highly complex system. Therefore, the contribution of the research is focused within the area of IoT-based WSNs with heterogeneous data traffic. A QoS requirement analysis pertaining to the integration conducted in the study reflects the QoS domains related to network and traffic heterogeneity.

Based on the review of the literature, it is evident that previous studies pertaining to the integration have concentrated on issues pertaining to the integration approach and its practical implication such as the security related issue. As such, there is a glaring lack of studies in the area of end-to-end QoS support for WSN-Internet integration, particularly to ensure preservation of QoS mechanism in both network domains. Furthermore, a significant research gap which has been highlighted in the literature review is the service differentiation mechanism to manage heterogeneous data traffic focused primarily on the real-time packets, whereas the QoS requirements of delay-tolerant data need to be carefully considered as well.

The QoS requirement analysis also indicates open research issues related to co-existence of real-time and delay-tolerant data packets, and the differences between WSN QoS and Internet QoS that imposes a mechanism for a seamless QoS interaction between both networks. The literature review has provide an insight of the integrated QoS components facilitating seamless interaction between both networks. Therefore, an integrated QoS framework is envisioned, that also operates on the IoT access point that supports the different QoS mechanisms used to manage the hetereogenous data within WSNs which are connected to the Internet.

A service differentiation-based QoS framework is proposed in handling various level of real-time traffic and delay tolerant traffic within the sensor network. The proposed ADHERE QoS framework is encompassed by two major components. The first component of the framework is a service differentiation-based QoS mechanism to manage heterogeneous data traffic. In the proposed model, separate queues are used for each type of traffic classes. Secondly, a prioritized buffer eviction policy is proposed to support different types of traffic classes, namely real-time traffic classes with and without reliability constraints, and multiple priorities delay-tolerant traffic classes.

The network modelling on Riverbed Modeler simulation indicates the ADHERE model system performance and provides a QoS-based design guideline for an actual WSN-Internet integration system. Several design parameters have been discussed and implemented in the experiment case studies. The network model was tested under several design parameters; namely, traffic distribution, buffer size, traffic arrival rate, traffic service rate and WSN node density. The performance parameters analysed are queuing delay, packet drop, buffer usage and delivery rate. Results show that the varying timeliness and reliability QoS requirements of RT and DT traffic can be met with the ADHERE QoS concept. Our findings show that the service differentiation among traffic and estimation of sufficient buffer size can accommodate the RT traffic timeliness and DT traffic reliability QoS requirements. Particularly, the low delay requirement of RT traffic was met through sufficient bandwidth allocation. In addition, the higher delivery rate requirement of DT traffic was achieved by using the ADHERE estimation of sufficient buffer size.

The learning activities through neural network provide a comprehensive validation to the proposed AQoS algorithm. The dropping in error rate indicates that the training has matured gradually. It is expected that a WSN with heterogeneous data which also involve traffic dynamic will evolve over time and introduce complexity to the system as the network grows in size. Additional types of sensor data may also be needed to accommodate the WSN application's requirements. Therefore, the learning capabilities in ADHERE can facilitate in optimising the QoS framework's performance by accommodating the QoS requirements of the network through the unpredictable traffic dynamics and when complex network behaviour takes place.

Finally, the ADHERE QoS model is implemented as a use case on a physical testbed. The experimentation serves as a means of verification and validation to the computer simulation environment. A conceptual organization targeting AQoS is also presented in this thesis. The architecture offers a system for interactions between the behaviour of a physical sensor network (PSN) and the necessary analysis in a virtual remote environment. The main contribution of the architecture is that it is capable of identifying the required adjustment for the PSN in order to enhance the QoS performance of the WSN applications. The concept of virtualization has also been presented in the implementation activities and the performance of both PSN and computer simulation under the QoS model has been distinguished. The virtualization has been demonstrated as the network performance shows similarity between the developed model in the simulation and the PSN experiment.

In the viewpoint of the implications of system deployments, we advocate that designing and evaluating ADHERE QoS model for an IoT-based WSN addresses the requirements for an adaptive WSN network performance matrix, which follows the demands of traffic dynamics in numerous IoT-WSN applications. The implementation of the QoS model on a physical testbed architecture indicates the potentials for future experimentation involving custom adaptive QoS parameters and real-time analysis of network performance. System planners would to be able to select relevant QoS parameters for certain application scenarios to evaluate the heterogeneous traffic performance as well as seamlessly accommodating the QoS requirements through the learning capability.

8.2 Future Work and Recommendation

Several future work has been identified from this research study. Firstly, further activities involving modelling and analysing a delay tolerant network (DTN) for WSNs integration can be carried out. The main activity may also include the validation and verification of QoS model under real-setting and open federated WSN testbeds [2, 105], in order to test the QoS model for interconnection of multiple WSNs. In this activity the influence of Internet propagation, under various number of router hops or intercontinental distance can be of interest.

In this research, a single-hopping network is considered. However, there are many WSN applications where multi-hop is involved. Therefore, another future work to be considered is to have more involved cases with a multi-hop network model. For this purpose, the queuing model components, namely the approximation of delay bound and buffer dimensioning will need to be revised to achieve a more suitable model which is suitable for multi-hop environment. In addition, the queuing model M/M/1/K, M/D/1 and M/G/1 can be considered as they correspond to finite buffers, and operate on fixed or generally distributed packet lengths.

Another aspect of future work recommendation is related to the techniques for interoperability among the system components, namely PSN and a remote database, virtual sensor network (VSN), QoS provisioning unit, and Contiki OS for network reconfiguration presented in Chapter 7. With the encouraging preliminary outcomes between pairing components, the testbed architecture can be used as a test environment for future experimentation involving custom AQoS parameters and real-time analysis of network performance. The co-simulation concept between network simulators and MATLAB is viable and this opens up more future research possibilities when more complexity takes place at the business intelligence level. Future work may include generating incremental outcomes from the modular organization to achieve a completely automated system. This model is well-suited to delay-tolerant applications such as this research's testbed- acquisition-system. In addition, the system may be tested under other several design parameters such as WSN node density, traffic distribution, network dimension, gateway capabilities and integration standards.

There are several future works identified from the neural network activities. For instance, through the data collection of the learning cycles and events, the information can be organised and deposited into a data warehousing architecture [106, 107]. This will provide a repository of the learning experiences, through organisations of network characteristic and related adjustments parameters adaptation to network dynamics. Furthermore, as the information could be used to predict the future QoS parameter as the system evolves, the unsupervised learning tool may also be used for controlling a physical sensor cloud in a virtualized environment as described in Chapter 5. This approach will also overcome the repetition of adjustment calculation for an adaptive QoS model like ADHERE, hence will contribute improvement to the latency of a virtualization system environment. Another potential work is to study the effect of using neural networks in an adaptive QoS from the application perspective. For example, an application that requires high resolution of sensor measurements will call for a sufficient sampling rate. In this perspective, the application's performance is influenced by the quality of obtained data; hence an adaptive feature which controls the traffic drop may give significant improvement to the network performance.

References

[1] S. Ezdiani and A. Al-Anbuky, "Modelling the integrated QoS for wireless sensor networks with heterogeneous data traffic," *Open Journal of Internet of Things (OJIOT),* vol. 1, 2015.

[2] S. E. S. N. Azlan and A. Al-Anbuky, "Quality of Service Modelling for Federated Wireless Sensor Network Testbed Gateways," in *Proc. of 5th Int. Conf. on Commun., Theory, Reliability, and Quality of Service (CTRQ 2012)*, Chamonix/ Mont Blanc, France, 2012, pp. 14-18.

[3] S. Ezdiani, I. S. Acharyya, S. Sivakumar, and A. Al-Anbuky, "An IoT environment for WSN adaptive QoS," in *Proc. of IEEE Int. Conf. on Data Science and Data Intensive Systems*, 2015, pp. 586-593.

[4] S. E. Syed Nor Azlan, I. S. Acharyya, S. Sivakumar, and A. Al-Anbuky, "An architectural concept for sensor cloud QoSaaS testbed," in *6th Workshop on Real World Wireless Sensor Networks (RealWSN 2015)*, Seoul, Republic of Korea, 2015.

[5] S. Ezdiani and A. Al-Anbuky, "Integrating WSN with the Internet: QoS Analysis and modeling for heterogeneous data traffic," presented at the Wireless Telecommunication Symposium (WTS 2014), Washington DC, 2014.

[6] S. Ezdiani, A. Indrajit S, S. Sivakumar, and A. Al-Anbuky, "Wireless Sensor Network Softwarization: Towards WSN Adaptive QoS," *IEEE Internet of Things Journal,* vol. 4, pp. 1517 - 1527, 2017.

[7] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks,* vol. 54, pp. 2787-2805, 2010.

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol. 29, pp. 1645-1660, 2013.

[9] N. Mitton, S. Papavassiliou, A. Puliafito, and K. S. Trivedi, "Combining cloud and sensors in a smart city environment," *EURASIP Journal on Wireless Commun. and Networking,* vol. 2012, pp. 1-10, 2012.

[10] L. Wu, J. Riihijarvi, and P. Mahonen, "A modular wireless sensor network gateway design," in *Proc. of Int. Conf. on Commun. and Networking in China (ChinaCom 2007)*, Shanghai, China, 2007.

[11] L. Shu, X. Wu, H. Xu, J. Yang, C. Jinsung, and L. Sungyoung, "Connecting heterogeneous sensor networks with IP based wire/wireless networks," in *Proc. of the 4th IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 2nd Int. Workshop on Collaborative Computing, Integration, and Assurance (SEUS 2006/WCCIA 2006)* p. 6 pp.

[12] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *Proc. of 2003 Conf. on Applications, Technologies, Architectures and Protocols for Comput. Commun. (SIGCOMM 2003)*, Karlsruhe, Germany, 2003, pp. 27-34.

[13] P. A. C. d. S. Neves and J. J. P. C. Rodrigues, "Internet Protocol over Wireless Sensor Networks, from Myth to Reality," *Journal of Communications,* vol. 5, pp. 189-196, March 2010.

[14] L. Shu, X. Hui, X. Wu, L. Zhang, C. Jinsung, and L. Sungyoung, "VIP Bridge: Integrating several sensor networks into one virtual sensor network," in *Proc. of Int. Conf. on Internet Surveillance and Protection (ICISP '06)*, Côte d'Azur, France, 2006, pp. 2-2.

[15] C. Alcaraz, P. Najera, J. Lopez, and R. Roman, "Wireless sensor networks and the internet of things: Do we need a complete integration?," in *1st Int. Workshop on the security of the Internet of Things (SecIoT'10)*, 2010.

[16] M.-A. Nef, S. Karagiorgou, G. I. Stamoulis, and P. K. Kikiras, "Supporting service differentiation in wireless sensor networks," in *Proc. of the 15th Panhellenic Conf. on Informatics (PCI 2011)*, 2011, pp. 127-133.

[17] M.-A. Nef, L. Perlepes, S. Karagiorgou, G. I. Stamoulis, and P. K. Kikiras, "Enabling QoS in the Internet of Things," in *Proc. 5th Int. Conf. Commun., Theory, Reliability, and Quality of Service (CTRQ 2012)*, Chamonix/ Mont Blanc, France, 2012, pp. 33-38.

[18] P. Porambage, C. Schmitt, P. Kumar, A. Gurtov, and M. Ylianttila, "PAuthKey: A pervasive authentication protocol and key establishment scheme for wireless sensor networks in distributed IoT applications," *International Journal of Distributed Sensor Networks,* vol. 10, p. 14, 2014.

[19] Z. Liang and C. Han-Chieh, "Multimedia traffic security architecture for the internet of things," *IEEE Network,* vol. 25, pp. 35-40, 2011.

[20] J. Jin, J. Gubbi, T. Luo, and M. Palaniswami, "Network architecture and QoS issues in the Internet of Things for a smart city," in *Proc. Int. Symp. Commun. and Inf. Technol. (ISCIT)*, 2012, pp. 956-961.

[21] B. Bhuyan, H. K. D. Sarma, N. Sarma, A. Kar, and R. Mall, "Quality of Service (QoS) provisions in wireless sensor networks and related challenges," *Wireless Sensor Network,* vol. 2, pp. 861-868, November 2010.

[22] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks,* vol. 51, pp. 921-960, 2007.

[23] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu*, et al.*, "CarTel: A distributed mobile sensor computing system," in *Proc. 4th Int. Conf. on Embedded Networked Sensor Systems (SenSys'06)*, Colorado, USA, 2006, pp. 125-138.

[24] E. Felemban, C.-G. Lee, E. Ekici, R. Boder, and S. Vural, "Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks," in *Proc. of 24th Annual Joint Conf. of the IEEE Computer and Commun. Societies (INFOCOM 2005)*, Miami, USA, 2005, pp. 2646-2657.

[25] H. Liu, A. Srinivasan, K. Whitehouse, and J. A. Stankovic, "Melange: Supporting heterogeneous QoS requirements in delay tolerant sensor networks," in *Proc. of 7th Int. Conf. on Networked Sensing Systems (INSS)*, Kassel, Germany, 2010, pp. 93-96.

[26] J.-F. Martinez, A.-B. Garcia, I. Corredor, L. Lopez, V. Hernandez, and A. Dasilva, "Modelling QoS for wireless sensor networks," in *IFIP International Federation for Information Processing*. vol. 248, Wireless Sensor and Actor Networks ed Boston: Springer, 2007, pp. 143-154.

[27] N. A. Alrajeh, S. Khan, and B. Shams, "Intrusion detection systems in wireless sensor networks: A review," *International Journal of Distributed Sensor Networks,* vol. 9, 2013.

[28] M. H. Yaghmaee and D. A. Adjeroh, "Priority-based rate control for service differentiation and congestion control in wireless multimedia sensor networks," *Computer Networks,* vol. 53, pp. 1798-1811, 2009.

[29] D. Chen and P. K. Varshney, "QoS support in wireless sensor networks: A survey " in *Proc. of Intl. Conf. on Wireless Networks (ICWN '04)*, Las Vegas, Nevada, USA, 2004.

[30] Y. Li, C. S. Chen, Y.-Q. Song, and Z. Wang, "Real-time QoS support in wireless sensor networks: A survey," in *Proc. of 7th IFAC Intl. Conf. on Fieldbuses & Networks in Industrial & Embedded Systems (FeT 2007)*, France, 2007.

[31] B. Braden, D. Clark, and S. Shenker, "Integrated service in the Internet architecture: An overview, RFC 1633," June 1994.

[32] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services, RFC 2475," December 1998.

[33] A. Ganz, Z. Ganz, and K. Wongthavarawat, *Multimedia Wireless Networks: Technologies, Standards and QoS*. Upper Saddle River, NJ: Prentice Hall PTR, 2004.

[34] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A Framework for QoS-based Routing in the Internet, RFC 2386," 1998.

[35] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxanomy of wireless micro-sensor network communication models " *ACM Mobile Computing and Communication Review (MC2R),* June 2002.

[36] W. Su and B. Almaharmeh, "QoS Integration of the Internet and wireless sensor network," *WSEAS Transaction on Computers,* vol. 7, pp. 253-258, April 2008.

[37] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, "TinyREST- a protocol for integrating sensor networks into the Internet," in *Proc. of REALWSN*, 2005.

[38] M. R. Kosanovic and M. K. Stojcev, "Implementation of TCP/IP protocols in wireless sensor networks," in *XIII Int. Scientific Conf. on Info., Comm. and Energy Systems and Technologies (ICESR 2007)*, Ohrid, Macedonia, 2007, pp. 143 - 146.

[39] M. Marchese, *QoS Over Heterogeneous Networks*: Wiley.com, 2007.

[40] M. Marchese and M. Mongelli, "Vertical QoS mapping over wireless interfaces," *IEEE Wireless Communications,* vol. 16, pp. 37-43, 2009.

[41] Y. L. Morgan and T. Kunz, "A proposal for an ad-hoc network QoS gateway," in *Proc. of the IEEE Int. Conf. on Wireless and Mobile Computing, Networking and Commun. (WiMod'05)*, Montreal, Canada, 2005, pp. 221-228.

[42] Y. L. Morgan and T. Kunz, "A design framework for wireless MANET QoS gateway," in *Proc. of 6th Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing and 1st ACIS Int. Wksp. on Self-Assembling Wireless Networks (SNPD/SAWN'05)* Towson, USA, 2005.

[43] B. Nefzi and Y.-Q. Song, "QoS for wireless sensor networks: Enabling service differentiation at the MAC sub-layer using CoSenS," *Ad Hoc Networks,* vol. 10, pp. 680-695, 2012.

[44] F. Safaei, H. Mahzoon, and M. S. Talebi, "A simple priority-based scheme for delay-sensitive data transmission over wireless sensor networks," *International Journal of Wireless & Mobile Networks,* vol. 4, 2012.

[45] S. Bhatnagar, B. Deb, and B. Nath, "Service differentiation in sensor networks," in *Proc. of 4th Intl. Symp. on Wireless Personal Multimedia Commun.*, 2001.

[46] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "RAP: A real-time communication architecture for large-scale wireless sensor networks," in *Proc. of 8th IEEE Real-Time and Embedded Technology and Applications Symp.*, 2002, pp. 55-66.

[47] N. Jain, D. K. Madathil, and D. P. Agrawal, "Exploiting multi-path routing to achieve service differentiation in sensor networks," in *Proc. of 11th IEEE Intl. Conf. on Networks (ICON 2003)*, Sydney, Australia, 2003.

[48] K. Akkaya and M. Younis, "An energy-aware QoS routing protocol for wireless sensor networks," in *Proc. of 23rd Intl. Conf. on Distributed Computing Systems Workshops*, 2003, pp. 710-715.

[49] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A stateless protocol for real-time communication in sensor networks," in *Proc. of 23rd Intl. Conf. on Distributed Computing Systems (ICDCS 2003)*, 2003, pp. 46-55.

[50] W. L. Tan, O. Yue, and W. C. Lau, "Performance vvaluation of differentiated services mechanisms over wireless sensor networks," in *Proc. of IEEE 64th Vehicular Technology Conf. (VTC-2006 Fall)* 2006, pp. 1-5.

[51] M. H. Yaghmaee and D. Adjeroh, "A model for differentiated service support in wireless multimedia sensor networks," in *Proc. of 17th Int. Conf. on Comput. Commun. and Networks (ICCCN 2008)* India, 2008, pp. 1-6.

[52] E. Felemban, C.-G. Lee, and E. Ekici, "MMSPEED: Multipath Multi-SPEED Protocol for QoS guarantee of reliability and timeliness in wireless sensor networks," *IEEE Transactions on Mobile Computing,* vol. 5, pp. 738-754, 2006.

[53] P. McDonald, D. Geraghty, I. Humphreys, S. Farrell, and V. Cahill, "Sensor network with delay tolerance (SeNDT)," in *Proc. of 16th Int. Conf. on Comput. Commun. and Networks (ICCCN 2007)*, Hawaii, USA, 2007, pp. 1333-1338.

[54] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," in *ACM Sigplan Notices*, 2002, pp. 96-107.

[55] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University,2000.

[56] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Computing and Communications Review,* vol. 7, pp. 19-20, 2003.

[57] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: An efficient routing scheme for intermittently connected mobile networks," in *Proc. of the 2005 ACM SIGCOMM Workshop on Delay Tolerant Networking (SIGCOMM 2005)*, Philadelphia, USA, 2005, pp. 252-259.

[58] T. Melodia and I. F. Akyildiz, "Research challenges for wireless multimedia sensor networks," in *Distributed Video Sensor Networks*, ed: Springer, 2011, pp. 233-246.

[59] A. Al-Sawaai, I. Awan, and R. Fretwell, "Analysis of the weighted fair queuing system with two classes of customers with finite buffer," in *Proc. of Int. Conf. on Advanced Information Networking and Applications Workshops (WAINA '09)*, 2009, pp. 218-223.

[60] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *Proc. of the 2nd Int. Conf. on Embedded Networked Sensor Systems*, 2004, pp. 134-147.

[61] M. A. Kafi, D. Djenouri, J. Ben-Othman, and N. Badache, "Congestion control protocols in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials,* vol. 16, pp. 1369-1390, 2014.

[62] C. T. Ee and R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks," in *Proc. of the 2nd Int. Conf. on Embedded Networked Sensor Systems*, 2004, pp. 148-161.

[63] C. Wang, B. Li, K. Sohraby, M. Daneshmand, and Y. Hu, "Upstream congestion control in wireless sensor networks through cross-layer optimization," *IEEE Journal on Selected Areas in Communications,* vol. 25, pp. 786-795, 2007.

[64] M. H. Yaghmaee and D. Adjeroh, "A new priority based congestion control protocol for wireless multimedia sensor networks," in *Proc. of 2008 Int. Symp. on World of Wireless, Mobile and Multimedia Networks (WoWMoM 2008)*, 2008, pp. 1-8.

[65] C. Wang, K. Sohraby, V. Lawrence, B. Li, and Y. Hu, "Priority-based congestion control in wireless sensor networks," in *Proc. of IEEE Int. Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2006, p. 8.

[66] Y. G. Iyer, S. Gandham, and S. Venkatesan, "STCP: a generic transport layer protocol for wireless sensor networks," in *Proc. of 14th Int. Conf. on Computer Comm. and Networks (ICCCN)* 2005, pp. 449-454.

[67] Y.-L. Chen and H.-P. Lai, "A fuzzy logical controller for traffic load parameter with priority-based rate in wireless multimedia sensor networks," *Applied Soft Computing,* vol. 14, Part C, pp. 594-602, 2014.

[68] M. A. Kafi, D. Djenouri, J. Ben-Othman, and N. Badache, "Congestion Control Protocols in Wireless Sensor Networks: A Survey," *Communications Surveys & Tutorials, IEEE,* vol. 16, pp. 1369-1390, 2014.

[69] S. A. Madani, J. Kazmi, and S. Mahlknecht, "Wireless sensor networks: modeling and simulation," ed: Vienna University of Technology, Vienna, Austria, 2010.

[70] A. Sobeih, J. C. Hou, L.-C. Kung, N. Li, H. Zhang, W.-P. Chen*, et al.*, "J-Sim: a simulation and emulation environment for wireless sensor networks," *IEEE Wireless Communications,* vol. 13, pp. 104-119, 2006.

[71] (July 2018). *The Network Simulator - ns-2*. Available: http://www.isi.edu/nsnam/ns/

[72] J.-M. Huang, C.-Y. Li, and K.-H. Chen, "TALONet: A power-efficient grid-based congestion avoidance scheme using multi-detouring technique in wireless sensor networks," in *Wireless Telecommunications Symposium (WTS 2009)*, 2009, pp. 1-6.

[73] D. Lee and K. Chung, "Adaptive duty-cycle based congestion control for home automation networks," *IEEE Transactions on Consumer Electronics,* vol. 56, 2010.

[74] M. A. Razzaque and C. S. Hong, "Congestion detection and control algorithms for multipath data forwarding in sensor networks," in *Proc. of 11th Int. Conf. on Advanced Communication Technology (ICACT 2009)*, 2009, pp. 651-653.

[75] V. Naoumov and T. Gross, "Simulation of large ad hoc networks," in *Proc. of the 6th ACM Int. Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, 2003, pp. 50-57.

[76] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," in *Proc. of 12th Workshop on Parallel and Distributed Simulation (PADS 98)*, 1998, pp. 154-161.

[77] E. Egea-Lopez, J. Vales-Alonso, A. S. Martinez-Sala, P. Pavon-Marino, and J. García-Haro, "Simulation tools for wireless sensor networks," in *Proc. of Summer Simulation Multiconference (SPECTS)*, 2005, pp. 2-9.

[78] M. Varshney, D. Xu, M. Srivastava, and R. Bagrodia, "sQualNet: A scalable simulation and emulation environment for sensor networks," in *Proc. of the Int. Conf. on Info. Processing in Sensor Networks*, New York, USA, 2007, p. 24.

[79] J. Y. Teo, Y. Ha, and C. K. Tham, "Interference-minimized multipath routing with congestion control in wireless sensor network for high-rate streaming," *IEEE Transactions on Mobile Computing,* vol. 7, pp. 1124-1137, 2008.

[80] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proc. of the 1st Int. Conf. on Simulation Tools and Techniques for Comm., Networks and Systems*, 2008, p. 60.

[81] E. Egea-Lopez, J. Vales-Alonso, A. Martinez-Sala, and P. Pavon-Marino, "Simulation scalability issues in wireless sensor networks," *IEEE Communications Magazine,* 2006.

[82] C. Mallanda, A. Suri, V. Kunchakarra, S. Iyengar, R. Kannan, A. Durresi*, et al.*, "Simulating wireless sensor networks with OMNET++," *submitted to IEEE Computer,* 2005.

[83] B. Zheng and M. Atiquzzaman, "Study of active queue management using OPNET."

[84] D. Akbaş and H. Gümüşkaya, "Real and OPNET modeling and analysis of an enterprise network and its security structures," *Procedia Computer Science,* vol. 3, pp. 1038-1042, 2011.

[85] S. Sivakumar, "Energy efficient opportunistic connectivity for wireless sensor network," Doctor of Philosophy, School of Engineering, Auckland University of Technology, 2013.

[86] G. Amoussou, B. Agba, Z. Dziong, M. Kadoch, and F. Gagnon, "Performances analysis of mobile ad hoc routing protocols under realistic mobility and power models," in *OPNETWORK*, 2006.

[87] R. Rajkamal and P. V. Ranjan, "Packet classification for network processors in WSN traffic using ANN," in *Proc. of 6th IEEE Int. Conf. on Industrial Informatics (INDIN 2008)*, 2008, pp. 707-710.

[88] K. Garg, A. Förster, D. Puccinelli, and S. Giordano, "Towards realistic and credible wireless sensor network evaluation," in *Ad Hoc Networks*. vol. 89, D. Simplot-Ryl, M. Dias de Amorim, S. Giordano, and A. Helmy, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 49-64.

[89] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo. (November 2011) A survey on facilities for experimental Internet of Things research. *IEEE Communications Magazine*. 58 - 66.

[90] (July 2018). *SeNSe Project - QoS Federated Sensor Networks over the Internet*. Available: http://sense.aut.ac.nz/CC2538_test/plotgraphs_comparison31.php

[91] A. Barbato, M. Barrano, A. Capone, and N. Figiani, "Resource oriented and energy efficient routing protocol for IPv6 wireless sensor networks," in *IEEE Online Conference on Green Communications (GreenCom)*, 2013, pp. 163-168.

[92] (July 2018). *Santander on Fire - Future Internet Research & Experimentation*. Available: www.smartsantander.eu

[93] R. Girau, S. Martis, and L. Atzori, "Lysis: a platform for IoT distributed applications over socially connected objects," *IEEE Internet of Things Journal,* vol. PP, pp. 1-1, 2016.

[94] (July 2018). *The iCore Project*. Available: http://www.iot-icore.eu

[95] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking,* vol. 1, pp. 344-357, 1993.

[96] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks* vol. 2: Prentice-Hall International New Jersey, 1992.

[97] J. Ben-Othman and B. Yahya, "Energy efficient and QoS based routing protocol for wireless sensor networks," *Journal of Parallel and Distributed Computing,* vol. 70, pp. 849-857, 2010.

[98] M. Chopde, K. Ramteke, and S. Kamble, "Probabilistic model for intrusion detection in wireless sensor network," *Int. J. Commun. Netw. Secur.(IJCNS),* vol. 1, pp. 19-23, 2011.

[99] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on,* vol. 1, pp. 397-413, 1993.

[100] Y. Li and L. E. Parker, "Intruder detection using a wireless sensor network with an intelligent mobile robot response," in *IEEE Southeastcon*, 2008, pp. 37-42.

[101] J. Liu, Y. Zhang, Y. Zhou, D. Zhang, and H. Liu, "Aggressive resource provisioning for ensuring QoS in virtualized environments," *IEEE Transactions on Cloud Computing,* vol. 3, pp. 119-131, 2015.

[102] I. Khan, R. Jafrin, F. Z. Errounda, R. Glitho, N. Crespi, M. Morrow*, et al.*, "A data annotation architecture for semantic applications in virtualized wireless sensor networks," *arXiv preprint arXiv:1501.07139,* 2015.

[103] S. Cirani, L. Davoli, G. Ferrari, L. R, x00E, one*, et al.*, "A scalable and self-configuring architecture for service discovery in the Internet of Things," *IEEE Internet of Things Journal,* vol. 1, pp. 508-521, 2014.

[104] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE Int. Conf. on Local Computer Networks*, 2004, pp. 455-462.

[105] D. Bimschas, O. Kleine, and D. Pfisterer, "Debugging the Internet of Things: A 6LoWPAN/CoAP testbed infrastructure," in *Ad-hoc, Mobile, and Wireless Networks*, ed: Springer, 2012, pp. 207-220.

[106] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: A survey," *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 553-576, 2016.

[107] M. Rifaie, K. Kianmehr, R. Alhajj, and M. J. Ridley, "Data warehouse architecture and design," in *Proc. of IEEE Int. Conf. on Info. Reuse and Integration (IRI 2008)*, 2008, pp. 58-63.