CONTEXT AWARE TRAFFIC SCHEDULING ALGORITHM FOR COMMUNICATION SYSTEM IN POWER DISTRIBUTION NETWORK

NURSHAZLINA BINTI SUHAIMY

A Thesis Submitted to the College of Graduate Studies, Universiti Tenaga Nasional in Fulfillment of the Requirements for the Degree of

Master of Electrical Engineering

November 2019

DECLARATION

I hereby declare that the thesis is my original work except for quotations and citations which has been duly acknowledged. I also declare that it has not been previously, and is not concurrently submitted for any other degree at Universiti Tenaga Nasional or at any other institutions. This thesis may be made available within the university library and may be photocopies and loaned to other libraries for the purpose of consultation.

(Signature)

NURSHAZLINA SUHAIMY

Date: 4 November 2019

ABSTRACT

Smart grid is an evolution from conventional power grid in which power communication system has migrated from one-way to bi-directional communication network. This enables smart grid to communicate with each of its devices and applications in monitoring and controlling the components remotely. Thus, an enormous amount of data will be collected from substations which consist of various types of traffic, and send to master station through utility network. Different kinds of traffic have their own pre-defined priority. However, the priority of data may change depending on the context or situation at a particular time. Hence, the priority of data needs to be redefined according to its context to ensure reliability and timeliness of critical data. Lost or delayed data result in incorrect control actions and risk the stability of smart grid. In order to support diversity of smart grid traffics, a contextual aware based Quality of Service (QoS) management system is needed to be introduced across the entire grid so that the system is aware of the changing context for each traffic. This thesis proposes a contextual aware traffic scheduling algorithm to manage the traffics according to their level of priorities by having the ability to adapt to the preferred requirements. The designated algorithm is tested with Multi-Protocol Label Switching (MPLS) protocol using Network Simulator 3 (NS-3) as a simulation platform. NS-3 is chosen as it is a discrete-event network simulator for research and development purposes. The simulation results from NS-3 is validated with the results obtained from MATLAB. It proves that the contextual aware ranking of each traffic are correctly matched to one another. In order to verify the advantages of the proposed algorithm, it is compared with simulation results without contextual aware algorithm. The quantitative results obtained demonstrated improvements in terms of delay and jitter with an average of 68.49% and 44.87% respectively. Based on these, it is proven that the algorithm with contextual aware performs better for delay and jitter while maintaining the throughput as compared to the algorithm without contextual aware.

ACKNOWLEDGEMENT

It is a genuine pleasure to express my deep sense of thanks and gratitude to my supervisor, Ir. Dr. Nurul Asyikin binti Mohamed Radzi, Prof. Dr. Md Zaini bin Jamaludin, Dr. Fairuz bin Abdullah, Dr. Wan Siti Halimatul Munirah binti Wan Ahmad and Mr. Aiman bin Ismail. Their dedication, expertise, understanding, generous guidance and support have made it possible for me to accomplish this thesis and to learn useful and precious experience from each of them.

Secondly, I wish to acknowledge the help and support provided by Mr. Mohammad Azmi Ridwan and Mr. Mohd. Hakeem, members of the research group. I would also like to extend my thanks to Mr. Mohd Nasim bin Zakaria, our collaborator from TNB ICT in providing the necessary resources in my study.

Finally, I would like to thank my parents and my family for giving me motivations, advices, and supports throughout the period of my Master program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

C	Set of priority classes
S_x^c	Data size in binary of *C* traffic class with *x* service parameters
R_y^c	Data rate in binary of *C* traffic class with *y* service parameters
D_z^c	Delay in binary of previous *C* traffic class with *z* service parameters
S_n^c	Data size in *Bytes* of *C* traffic class
R_n^c	Data rate in *bps* of *C* traffic class
D_n^c	Delay in *seconds* of previous *C* traffic class
$F_c(t)$	Number of flows belonging to each class
$F_c\left(S_x^c, R_y^c, D_z^c\right)$	Traffic flow of particular traffic class *C* for each service parameters
$T_{transmission}$	Transmission time
ε_c^i	Pre-defined priority
ε_c^j	Contextual aware priority
S_{min}^c	Minimum threshold values of data size for *C* traffic class
S_{max}^c	Maximum threshold values of data size for *C* traffic class
R_{min}^c	Minimum threshold values of data rate for *C* traffic class
R_{max}^c	Maximum threshold values of data rate for *C* traffic class
D_{min}^c	Minimum threshold values of delay for *C* traffic class
D_{max}^c	Maximum threshold values of delay for *C* traffic class
$S_{threshold}^c$	Threshold values of data size for *C* traffic class

$R^{c}_{threshold}$	Threshold values of data rate for *C* traffic class
$D^{c}_{threshold}$	Threshold values of delay for *C* traffic class

LIST OF ABBREVIATIONS

AF	Assured Forwarding
AMI	Advanced Metering Infrastructure
API	Application Programming Interfaces
BE	Best Effort
CATSchA	Contextual Aware Traffic Scheduling
CIGRE	International Council on Large Electric Systems
CWDM	Coarse Wavelength Division Multiplexing
DiffServ	Differentiated Services
DSCP	DiffServ Code Point
DWDM	Dense Wavelength Division Multiplexing
E1/T1	Primary Multiplex System
EF	Expedited Forwarding
EXP	Experimental
FEC	Forwarding Equivalence Class
GSM	Global System for Mobile
HAN	Home Area Network
ICS	Industrial Control Services
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Devices
IntServ	Integrated Services
IP	Internetworking Protocol
IP-MPLS	Internet Protocol-Multiprotocol Label Switching
IT	Information Technology
ITU-T	International Telecommunication Union – Telecommunication Section
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Label Switch Path
LSR	Label Switch Router
MATLAB	Matrix Laboratory
MOQoS	Matrix of QoS

MPLS	Multiprotocol Label Switching
NAN	Neighbourhood Area Network
NLDC	National Load Dispatch Center
NS-3	Network Simulator 3
OAM	Operation, Administration and Maintenance
OLT	Optical Line Termination
ONU	Optical Network Units
OT	Operational Technology
OTN	Optical Transport Network
P/E	Distribution Substation (Pencawang Elektrik)
PE	Provider Edge
PLC	Power Line Carriers
PMU	Transmission Main Intake (Pencawang Masuk Utama)
PON	Passive Optical Network
PPU	Main Distribution Substation
QoS	Quality of Service
RSVP	Resource Reservation Protocol
RSVP-TE	Resource Reservation Protocol – Traffic Engineering
RTU	Remote Terminal Units
S	Stack
SCADA	Supervisory Control and Data Acquisition
SDH	Synchronous Digital Hierarchy
SDN	Software Defined Networking
SLA	Service Level Agreement
SONET	Synchronous Optical Network
SSU	Main Switching Station (Stesen Suis Utama)
STM	Synchronous Transport Modules
STS	Synchronous Transport Signal
TDM	Time Division Multiplexing
TNB	Tenaga Nasional Berhad
TTL	Time to Live
WAN	Wide Area Network
WDM	Wavelength Division Multiplexing

LIST OF PUBLICATIONS

The following list shows the publication done by author up to the date of thesis.

1. N. Suhaimy, W. Ahmad, N. Radzi, F. Abdullah, M. Jamaludin, and M. Zakaria, "Analysis of MPLS-TP Network for Different Applications," *International Journal of Engineering & Technology,* vol. 7, pp. 892-898, 2018.

2. N. Radzi, N. Suhaimy, W. Ahmad, A. Ismail, F. Abdullah, M. Jamaludin, *et al.*, "Context Aware Traffic Scheduling Algorithm for Power Distribution Smart Grid Network," *IEEE Access,* vol. 7, pp. 104072-104084, 2019.

CHAPTER 1

INTRODUCTION

1.1 Background

Power grid consists of four main divisions namely generation, transmission, distribution and last mile consumer. Generation division comprises of power plants which are responsible to produce power and then transmit them via step-up transformer over transmission lines. Transmission division is in charge of transporting stepped up generated power via high voltage grid to be passed onto distribution. Distribution division is responsible to distribute stepped down generated power via numerous substations to last mile consumers. Last mile consumer can be divided into two different categories which are commercial and industrial, and residential. In last mile division, applications of smart grid are scattered which involves areas that are not yet been fiberized due to geographical areas thus, data traffic is collected through Global System for Mobile (GSM) communication.

Each of these divisions has their own roles in delivering electrical power to users. Current power grid is in the process of evolving from conventional power grid to smart grid. Conventional power grid involves only one-way communication system [1]. Whereby smart grid involves bi-directional communication system that requires robust and scalable network infrastructure [2].

Tremendous changes are happening in electrical power delivery system from aspects of economic, environmental, organizational and technological factors. These changes are spread across the whole divisions in power grid. Electrical power utilities are becoming more dependent upon fast, secure and reliable communication services which interconnect Intelligent Electronic Devices (IEDs) from technical sites. This situation is becoming critical as smart devices are transmitting data over the utility networks simultaneously; with different applications, constraints and priorities.

This thesis addresses some of the challenges in managing the increasing complexity of delivering electricity in smart grid while ensuring reliability and timeliness of critical data. The priority of data needs to be dynamic according to its context in order to ensure the communication services are provisioned, managed and maintained accordingly. A contextual aware based Quality of Service (QoS) traffic scheduling algorithm is needed to be introduced in power distribution division as to fully utilize the capability of smart grid networks by reducing its delay and jitter in guaranteeing a flawless traffic transmission.

1.1.1 Smart Grid

Smart grid is introduced where it allows two-way communication between power utility companies and their customers to improve reliability, energy and asset efficiency, while increasing operational efficiency and anticipating customer participation. Bi-directional communication allows each IEDs to communicate with one another in monitoring and controlling the components remotely. Hence, allowing smart grid in facilitating an advanced load management as well as generating and distributing electricity efficiently [3]. Smart grid is also able to utilize a self-healing protection mechanism for overall power system in becoming more reliable and energy efficient.

Smart grid telecommunication network requires secure, scalable, reliable, future proof and cost effective service as to support real time critical applications. Communication infrastructure in smart grid comprises of three different areas which are Home Area Network (HAN), Neighbourhood Area Network (NAN) and Wide Area Network (WAN) [4]. HAN or also known as access network, is focusing on small scale data communication located in last mile division. Access network involves large number of sites with diverse communication options. Whereby NAN or also known as metro or aggregation network, situated in distribution division, acts as backbone in transmitting data from multiple HANs. Metro or aggregation network comprises of simple but critical traffic flows with large number of sites. WAN or also known as core network, is placed in transmission division as a connecter to a grid control center and acts as a data concentrator of NAN [5, 6]. Core network consists of high density data traffic with limited number of sites. Thus, metro or aggregation network is chosen for implementation of developed algorithm due to its critical network environment over large number of sites.

Standards are essential in ensuring interoperability between devices and seamless data integration in smart grid. International Telecommunication Union (ITU-T) has established a focus group specifically for smart grid [3]. This focus group is to develop recommendations and to evaluate implementation of smart grid standards. Other agency involved in standardization effort is International Council on Large Electric Systems (CIGRE) [6]. CIGRE has established a specific study committee in identifying issues of concern, developing new international standards and producing guidelines on smart grid applications.

For substation communication and environment, IEC 61850 standard is used as communication network and system for power utility automation which enable interoperability between IEDs via different communication services [7]. This standard is designed by International Electrotechnical Commission (IEC) for power industry communication [8]. IEC 61850 standard covers substation-to-substation, substation-to-control center, asset condition monitoring, synchrophasor applications and other power system domain. IEC 61850 standard is extended to IEC 61850-90-1 for communication between substations, IEC 61850-90-2 for communication between control center and substations, IEC 61850-90-3 for condition monitoring, IEC 61850-90-4 for network engineering guidelines, IEC 61850-90-5 for synchrophasor transmission information and IEC 61850-90-6 for distribution feeder automation system [6].

There are five major applications in smart grid; namely Advanced Metering Infrastructure (AMI), demand response, substation automation, distributed energy resources and wide area measurement [9]. Substation automation communication can be classified into two types which are intra-substation and inter-substation communications. Intra-substation is communication with external network of other substations or utility control center via gateway meanwhile, inter-substation is communication between machine to machine in the same substation.

Applications in smart grid can be characterized into two major classes depending on its urgency and criticality. Operational Technology is responsible in handling critical grid operations while Information Technology is in charge of dealing with non-critical grid operations and corporate system. Previously, both critical and non-critical operations are managed via separate network techniques which are Time Division Multiplexing (TDM) and packet technologies [10]. However, TDM technology is almost obsolete [11]. Utility company has decided to migrate to fully integrated packet technology and thus, both critical and non-critical operations will be handled through the same techniques [6].

1.1.2 Quality of Service Support in Smart Grid

QoS is the ability to guarantee performance of user data flow and to meet requirements of network user. In Internetworking Protocol (IP) context, QoS is defined as service differentiation and performance assurance or priority assignment mechanism. In assuring QoS, the simplest and most secure approach is by having dedicated network resources for every critical services [12]. Priority assignment, Resource Reservation (RSVP) and traffic engineering can be employed as QoS control mechanisms.

Priority assignment is to distinguish different classes of service by using priority indication field. Integrated Services (IntServ) is a principal approached in QoS which is used to allocate individualized QoS assurances to individual sessions [13, 14]. IntServ is based on per flow basis which involves packet stream with common source and destination addresses as well as port numbers.

Differentiated Services (DiffServ) is another principal approached in QoS according to type of service in qualifying data flow into different classes [13, 15]. It classifies traffics into different categories with different service level for each classes. The highest service level is Expedited Forwarding (EF) followed by Assured Forwarding (AF) and the lowest service level is Best Effort (BE). EF supports time-sensitive traffic with minimal latency and jitter whereas, AF supports medium priority traffic with guaranteed bandwidth. BE supports non-time-sensitive traffic with high latency and jitter.

QoS parameters consist of delay, jitter and throughput. Delay is defined as time taken for a packet to travel from source to destination. Jitter is defined as delay variations which cause packet to arrive at destination in inconsistent rate. Throughput is defined as the amount of data transferred between source and destination. Each of these parameters have stringent requirements that must be fulfilled by time-critical applications as to ensure reliability and timeliness of critical data, especially in utility backbone.

QoS support in smart grid is related to network technical performance as well as mechanisms that are used to differentiate traffic flows in multi-service environment and entrusting them with priorities. Specification and provisioning of communication services for electrical power utility operational applications are decisive criteria in accomplishing specific communication services in smart grid. Network and services in smart grid can be monitored, operated and maintained through an assurance of high QoS which is necessary in power system operation.

In smart grid network, QoS has the capability to serve given traffic flow promptly based on their pre-defined priority in avoiding queuing delays or loss. Traffic prioritization is done by classifying various applications in smart grid. Real time application has high priority whereas, non-real time application has low priority. At each point of potential congestion or also known as router, all incoming packets are checked and the order of packets is rearranged according to their relative priorities before transmission process takes place. The highest priority traffic is transmitted ahead of lower priority traffic to ensure QoS is met.

1.2 Problem Background

There are numerous traffics to be transmitted in utility network which can be categorized into two major groups namely Operational Technology (OT) and Information Technology (IT). OT network handles critical and sensitive grid operations traffics, hence no alteration or modifications are allowed in its traffic transmission. OT traffics always have guaranteed and undisrupted transmission in utility network. Whereas, for IT network, the traffics are in charge of dealing with non-critical and less sensitive grid operations where modifications involving traffic transmissions are allowed. Currently, IT traffics are transmitted simultaneously in utility network with different data size, data rate and delay requirement. There is no traffic classification and no priority assigned to each traffic which cause the traffics to be competing in delivering data and affecting the traffic outcome upon their transmission. When this occur, the traffic that needs higher priority compared to the rest of the traffics may be delayed, causing instability of the grid.

One of the example is in Pencawang Masuk Utama (PMU) Ayer Tawar, Perak which involves controlling of the surveillance cameras remotely for monitoring purposes. The cameras are defined to have low priority by default. However, when the camera needs to be panned, tilted or zoomed, then the priority of the camera needs to change to a higher priority level. Otherwise, the video outcome is lagging due to its default packet arrangement. This situation can also occur for other traffics including Smart Meter and Internet.

1.3 Problem Statements

Vast number of network elements from substations comprise different types of traffic along with their priority. However, there are times or situations where packets that have least priority need to be served or transmitted first based on its context. Thus, the needs to redefine priority of traffics is vital as the context of data may change over the time. QoS support in smart grid network needs to be aware of traffic with unusual data according to its situation as to ensure reliability and timeliness of critical data by evaluating its current data size, data rate and previous transmission delay. Different set of transmission time which include start and stop time for each traffic are applied in ensuring end-to-end QoS. Lost or delayed data results in incorrect control actions and risk the stability of smart grid.

1.4 Research Questions

Different traffics have different characteristics and requirements. Traffics such as SCADA must be transmitted first all the time in all possible situations and there is also traffics that require to be transmitted urgently only for certain situation. A contextual aware based QoS algorithm is necessary to accomplish these characteristics.

The research questions of this thesis are as follows:

1. What are the characteristics and priorities of smart grid network traffic? How are they categorized according to QoS?
2. How contextual aware traffic scheduling algorithm is developed for smart grid network? What kind of environment is involved for the algorithm development?
3. How is the performance of contextual aware traffic scheduling algorithm evaluated? What types of evaluations are involved in assessing the algorithm performance?

1.5 Research Aim

A contextual aware based QoS algorithm is required to be introduced in packet network environment for communication system in power distribution network. This can be achieved through characterizing and prioritizing of smart grid network traffic according to their characteristics in utility network in which the traffic scheduling can be modified according to their situations. Then, the performance of contextual aware traffic scheduling algorithm can be evaluated and assessed via simulation platform.

1.6 Research Objectives

The main objective of this project is to develop a contextual aware based QoS algorithm in packet network environment for power distribution network.

The specific objectives of this thesis are as follows:

1. Characterization and prioritization of smart grid network traffic according to QoS
2. Development of a contextual aware traffic scheduling algorithm for smart grid network in MPLS environment
3. Performance evaluation of contextual aware traffic scheduling algorithm by simulation work

1.7 Scope of Work

The overall scope of work of this study is presented in K-chart as shown in Figure 1.1. The system being studied in this thesis is smart grid, particularly in distribution division. This is because distribution substations need an advanced technique in dealing with enormous amount of data flows which will be sent to control center or master station via utility network. Other divisions including generation and transmission are not as important to have advanced technique since the amount of data flows are lesser compared to stream of data in distribution [16].

In this thesis, a contextual aware traffic scheduling algorithm is proposed to manage traffics of IT network. IT or enterprise network is selected as it consists more streams of data to be transmitted from multi sites to control center with various QoS requirements that needs proper handling. Recently, power utility operational network is migrating from current circuit based to packet based network as it follows the trend of telecom markets [17]. Vendors involved in providing the products are investing in packet based equipment by disregarding any further development of TDM networking equipment, which is the reason why packet switching network is chosen to be studied in this thesis.

The selected protocol for packet switching network is Multiprotocol Label Switching (MPLS) due to its simple priority assignment in order to support communication requirements of existing applications as well as data transmission capabilities in smart grid network. This technology is appropriate for large number of sites with enterprise or IT network which is responsible in managing utility non-operational or unrelated in directly handling with critical and time-sensitive operational traffic flows.

IT network comprises of utility non-operational and critical traffic flows with large number of sites thus, requires contextual aware to be implemented in aspects of traffic scheduling. The designated algorithm is studied using simulation method by validating the results obtained via two different simulator platforms. Simulation is elected as it is the first step to be taken to ensure the algorithm is working before implementing it in testbed. The performance of proposed algorithm is determined through comparison between contextual aware and without contextual aware results.

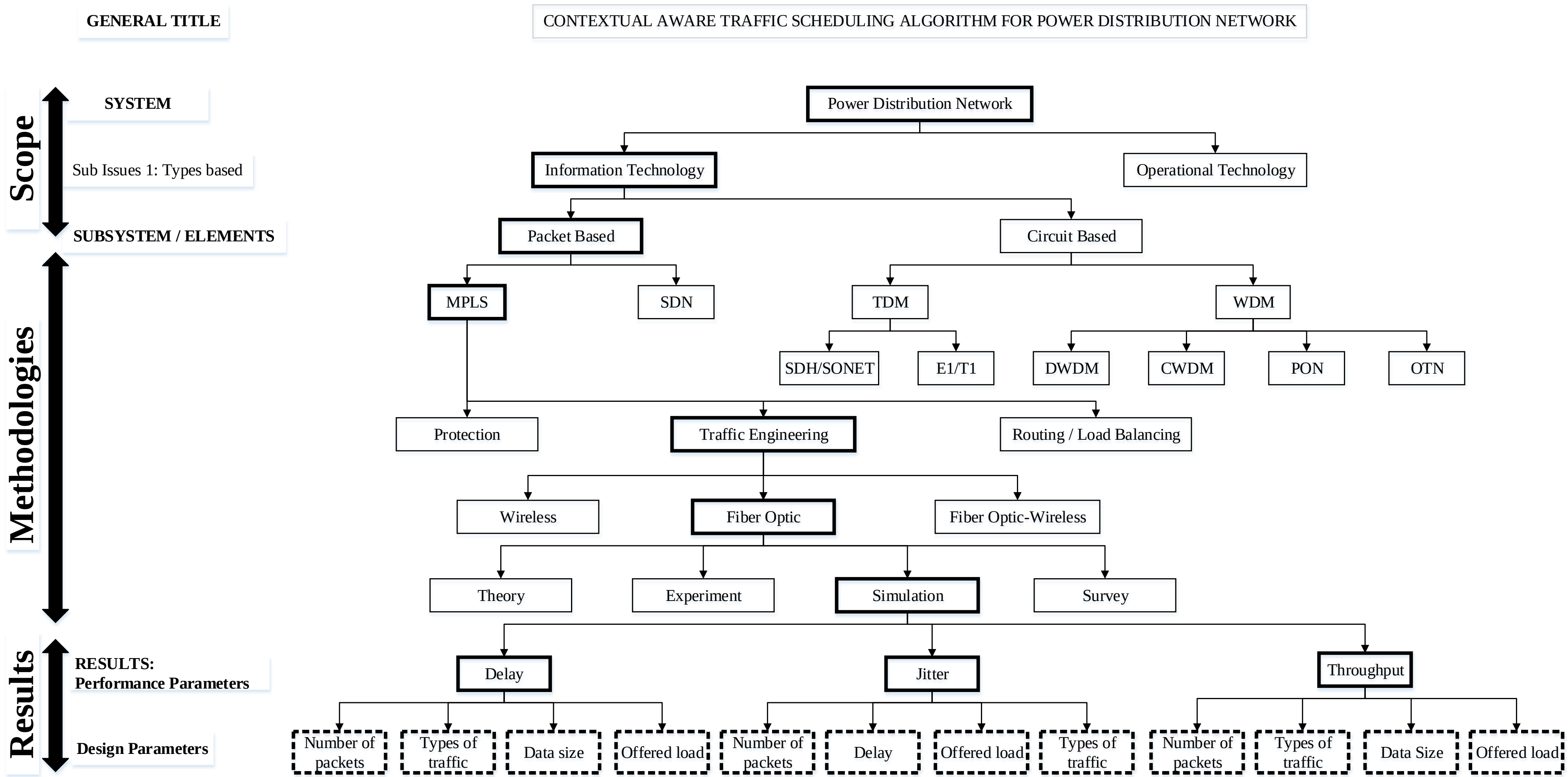

Figure 1.1 Scope of study

The performance parameters studied are delay, jitter and throughput. These three parameters are selected to show and to prove the significance in having contextual aware traffic scheduling algorithm by controlling IT traffic flows in utility network. Whereby, design parameters studied are number of packets, types of traffic, data size, delay and offered load. These design parameters are needed as control variables of the mentioned performance parameters.

1.8 Thesis Outline

This thesis consists of five chapters. Chapter 1 contains a background explanation on smart grid and QoS. Problem statements, research objectives, scope of work and thesis outline are also presented in this chapter.

Chapter 2 explains on the fundamental of networking technologies which comprises of operational and information technologies. Networking methods are then described involving circuit and packet switching networks together with their respective protocols. A review of related works in traffic scheduling algorithm is also discussed.

Chapter 3 introduces methodology used for this project. Traffic characterization is discussed together with the proposed algorithm of contextual aware traffic scheduling, the tool used for simulation, network environment and topology. System parameters which comprise of design and performance parameters are also highlighted.

Chapter 4 presents the simulation results and performance evaluation using simulation method. The simulation results of proposed algorithm from Network Simulator 3 (NS-3) is validated with the results obtained from Matrix Laboratory (MATLAB) software. Performance of contextual aware traffic scheduling algorithm is studied in terms of delay and throughput. It is then compared against algorithm without contextual aware in terms of delay, jitter and throughput.

Finally, Chapter 5 completes this thesis with a concluding section as well as recommendation for future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Numerous applications in smart grid network exist with different characteristics, especially at distribution division. Some of them are used to provide data concerning system parameters to be used in operating utility system; for example voltage, current and power values. Substation is one of the important elements in utility system as it controls power flow in delivering and transforming voltage from one substation to another. It is also used to isolate faulty elements by allowing affected device to be disconnected from the rest of the utility system for maintenance or repair work. Several types of substations in Malaysia handled by Tenaga Nasional Berhad (TNB) are Transmission Main Intake (PMU), Main Distribution Substation (PPU), Main Switching Station (SSU) and Distribution Substation (P/E). However, PMU is managed by transmission division while the rest are managed by distribution division. Each of these substations have their own responsibilities in ensuring functioning grid network system of the whole nation.

This chapter begins with a brief description on power distribution network in smart grid in Section 2.2 and network technologies in distribution division which comprises of Operational Technology (OT) and Information Technology (IT) in Section 2.3. In Section 2.4, two methods in networking system are explained followed by relevant protocols for both methods are described. Section 2.5 dwells on Multiprotocol Label Switching (MPLS) of packet switching network while Section 2.6 presents a critical review of traffic engineering algorithm in MPLS network. Summary is done on Section 2.7.

2.2 Power Distribution Network in Smart Grid

Power distribution network is an essential part in smart grid to guarantee a quality power transmission [18]. The typical setup of power distribution network in smart grid is based on centralized monitoring and controlled through SCADA system as the control center.

The function of this control center is to communicate with remote terminal units (RTUs) through several communication channels. RTU exists in a remote substation as a device that is connected to multiple sensors for collecting various measured quantities. The measured quantities such as voltage, current and status of circuit breakers are then reported to the control center for further actions [19].

RTU is also responsible in carrying out commands from the control center to remotely open or close the circuit breakers in substation in case of power failure. As for now, SCADA system is used to monitor high and medium voltage distribution networks such as 132/33 kV and 33/11 kV respectively [19].

2.3 Network Technologies

Generally, there are two main technologies involved in network services of smart grid; namely Operational Technology (OT) and Information Technology (IT) [6, 20]. These two technologies are introduced as they are responsible in handling numerous smart grid applications according to their criticality and requirements. Suitable protocols are needed in ensuring utility system to operate with high reliability and secured network. Such network is crucial in electrical substations to control and monitor each equipment or devices inside the substations. This access requirements must be supported and guaranteed end-to-end from control center to utility devices located in substations to support remote configuration, maintenance and parameter setting.

Figure 2.1 shows the current network infrastructure of both OT and IT technologies together with their situations and locations in large number of service utilities. It also shows an extension from Substation Operational Telecom Network to the Enterprise Network and IT infrastructure with an addition of packet switched communication for future development which is portrayed as dotted box as shown in Figure 2.1.

2.3.1 Operational Technology (OT)

OT technology is responsible in managing critical and time-sensitive applications [21]. This technology corresponds to a closed network interconnecting intelligent field assets over local and wide area networks via dedicated communication infrastructure, or also known as field network. Basically, OT is a collection of devices designated to work together as an integrated and homogeneous system. The whole utility system could be in disastrous domino effect if only one of the system fails to operate. OT technology is divided into two different categories which are Industrial Control Services (ICS) and non-ICS [22]. ICS involves controlling traffics, teleprotection and Supervisory Control and Data Acquisition (SCADA) [23] whereby, non-ICS comprises of online monitoring system [24].

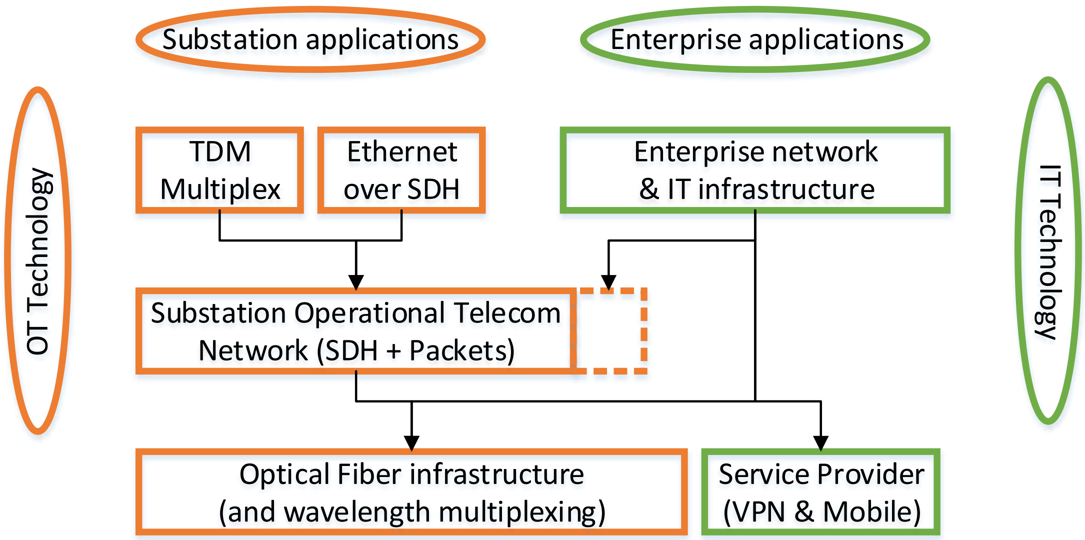

Figure 2.1 Present network overlay of OT and IT technologies in utility

Requirements for OT network includes an extensive legacy services and devices in which both of them must integrate with new IP services to communicate [6, 25]. Structure and traffic of OT technology involve a massive number of substations with small volume of data exchange since this technology only covers for small number of applications. Quality of OT focuses more on delay, continuity of services and fault tolerance.

Communication peers in OT technology are mostly pre-determined peer [6]. This means that it is mostly fixed and pre-established communication between substations and control center that is established via unique and pre-determined service access point. This is to allow security filtering and authentication.

Besides, OT network requires a stable service in order to evolve with power applications [6, 25]. It is relatively rare for creation and modification of connections to happen in operational network [6]. When network is changed it implies on application re-commissioning in OT technology.

Management process in OT technology needs fast fault recovery as well as continuous monitoring [6, 25]. These are essential in pre-determined communication peers. Operation, Administration and Maintenance (OAM) process requires simple and fast reaction in operating network of substations.

Security aspect of OT communication network is concentrating on service availability [6, 25]. The operational network is pre-determined in which it is closed network for time-sensitive device-to-device communication.

2.3.2 Information Technology (IT)

IT technology is accountable in handling less critical and less time-sensitive applications compared to OT technology [25]. This technology covers corporate enterprise, contractual and commercial communications in utility offices and information platforms with an extensive connection beyond its perimeter such as, other utilities, system operators and contractors. IT or also known as corporate enterprise network corresponds to IT infrastructure and activities across power utility [21]. Generally, IT is defined as entire spectrum of technologies for information processing that includes software, hardware, communication technologies and other related services. Among application services which belong to this group are Wi-Fi, CCTV, Internet, smart meter and SCADA.

IT network services are exclusively for IP with no legacy integration [6]. They are practically inexistent in enterprise network where all required services are basically IP and favours networking with lot of features.

Although minority of sites belong to IT network, large volume of data exchanged are involved [6]. This means that enterprise network composed of small number of utility offices with substantial volume of on-going increasing irregular traffic which results in deployment of complex integrated approach in substations.

Quality of IT communication is converging on throughput, flexibility, cost consideration as well as efficiency [6, 26]. Enterprise network prioritize bandwidth efficiency and flexible network services due to vast of data exchanged compared to OT network services. Thus, maintenance and installation cost involving enterprise network must be kept as minimum as possible while maintaining high throughput.

There is no restriction for communication peers in IT network services [6, 27]. Access between substations and control center are not fixed and does not require pre-established communication peers via unique service access point. Thus, security filtering and authentication are excluded for enterprise network.

On the other hand, frequent network updates are needed as this technology is growing with IT evolution [6]. Enterprise network cannot handle a fixed network service since that this technology involved huge amount of data exchanged with their respective constraints and continuously evolving.

Service provisioning and configuration are necessary to be regularly changed [6, 28]. Generally, enterprise network is a formal management associated with IT system. Priority is given to frequent service provisioning and change for mostly any-to-any communication.

External service provisioning and outsourced of OAM capabilities are common for IT technology [6]. In most cases, enterprise communication service is provisioned externally or operated and maintained via external entity. For example, through a telecom service provider.

Security of IT network concentrates on service integrity and confidentiality [6, 26]. Security assurance mechanism in enterprise network comprises of various traffics from substations to control center. Communication between devices in enterprise network are mainly transactional without time-sensitive constraint.

2.4 Networking Methods

Power distribution system is about developing and maintaining an efficient and safe distribution system in delivering electrical energy to consumers. Besides, ensuring availability of electrical energy supply and power quality are done at appropriate levels for all consumers. There are two different approaches included in networking system namely as circuit switching and packet switching networks. These methods can be implemented in maintaining network communication within utility system to run smoothly.

2.4.1 Circuit Switching Network

Circuit switched technology establishes connection between devices through a fixed path over the utility network. For simultaneous transmission of services, the same medium are shared by them in which all signals are combined through TDM. Each of these signals have their own allocated time slot with short gap time. On the other hand, Wavelength Division Multiplexing (WDM) provides separate frequency band allocated for each service which is employed over optical fiber. Resource allocation through TDM and also WDM can be simultaneously used in dissimilar hybrid configurations.

Both E1 and T1 are telecommunication standards of digital carrier which are referred to as primary multiplex system (ITU-T G.703) [6, 29]. E1 is a transmission link used in European countries which comprises of 32 channels with data rate of 2048 *kbps* and a fixed elementary digital stream at 64 *kbps*. T1 is used in North American countries that consists of 24 channels with data rate of 1544 *kbps* and a fixed elementary digital stream at 56 *kbps* [6].

Synchronous Digital Hierarchy (SDH) is a synchronous data exchange as applied in European countries. Synchronous Optical Network (SONET) supports multiple digital data streams simultaneously over long distance communication in United States region. Both of them are communicating over optical fiber in transporting circuit based communication. Synchronous Transport Modules (STM) and Synchronous Transport Signal (STS) are used by SDH and SONET respectively as standardized rates for interconnection between different regional network systems and services [6]. SDH/SONET is capable in managing network through performance monitoring, identifying type of traffic, recognizing connectivity as well as detecting and reporting any failures occurred with an extensive management information.

Coarse Wavelength Division Multiplexing (CWDM) is a technology with fewer than eight active wavelengths per fiber. Multiple signals on laser beams are combined in CWDM at various wavelengths for transmission along optical fiber media [30]. This technology is applicable for short range communication in which a wide range of frequencies are employed with its wavelengths located far apart from one another [31]. CWDM is a compact and cost-effective option as well as consume less power due to low precision of lasers [32]. Energy from lasers in CWDM is scattered over vast range of wavelengths as compared in DWDM technology [33].

Dense Wavelength Division Multiplexing (DWDM) is used to increase bandwidth over existing fiber networks. DWDM works as combining and transmitting multiple signals simultaneously at different wavelengths on same fiber [34, 35]. DWDM is specifically for system containing with more than eight active wavelengths per fiber [36]. This technology is designed for long haul transmission with its wavelengths are packed tightly together [37].

Passive Optical Network (PON) is a telecommunication technology which implements point-to-multipoint architecture that brings optical fiber cabling and signals to end users as to offer higher transmission rates without amplification [38]. Fiber optic splitter is used as to enable a single optical fiber to serve multiple end points without having to provision individual fibers between hub and users. PON consists of Optical Line Termination (OLT) and Optical Network Units (ONU) which are located at service provider of control center and near to end users respectively [39].

Optical Transport Network (OTN) technology can be referred to ITU-T G.872 for its general terms and ITU-T G.709 for its network interface [6, 40]. OTN is able to provide high capacity technology with high accuracy and transparent transport of user signals [41]. Other than that, powerful forward error correction function, simplified network design, low cost and ease of installation are additional features of this technology.

However, circuit switching network is not as widely used in power utility grid as TDM technology is almost obsolete and it is going to be replaced with packet switching network in stages [42]. Besides, longer time is needed to establish the communication between two ends in circuit switching network since it is implemented at physical layer compared to packet switching network where it is implemented between Layer 2 and Layer 3 [43].

2.4.2 Packet Switching Network

Packet switched technology is an enhancement structure of network resources by circuit switched technology. Transmitted data are split into various size of packets with source and destination address labelled at each of it. Thus, packets can be sent towards destination via multiple routes. The performance of packet network is depending on network load and packet routing across the network. The most substantial parameters for monitoring purposes are packet delay, jitter and also number of packet loss [44, 45].

Software Defined Networking (SDN) is an emerging architecture which separates between network control plane and data forwarding plane. Control plane is in charge of carrying traffic signaling for routing purposes, system configuration and management [46]. Data plane is accountable for carrying user traffic and manage communication remotely [47]. The separation between control plane and data plane allows in providing user applications with centralized view of distributed network states [48]. A set of Application Programming Interfaces (API) called north-bound API are supported as communication between application and control plane layers in order to enable network services. Data plane layer in SDN technology employs programmable OpenFlow switches which are able to interact with its SDN controller via south-bound API [49].

In this thesis, SDN is not considered for smart grid implementation because there is no guarantee for end-to-end information protection. The technology is vulnerable to intrusion and cyber-attacks as it involves remotely connected devices in networking system with no direct physical control [50]. Other than that, limited QoS support is provided by the current OpenFlow standard for SDN. Besides, fast failure recovery which is necessary for high reliability is difficult to be achieved due to its reliance on centralized controller in heterogeneous smart grid environment.

This thesis discusses about another packet switching network protocol called MPLS. MPLS is discussed in details in Section 2.5.

2.5 Multiprotocol Label Switching (MPLS) of Packet Switching Network

MPLS is a packet transport technology in which its transmission is based on reading the label in Shim header where it sits between Layer 2 and Layer 3 or also known as Layer 2.5 as can be seen in Figure 2.2 [24]. There are four parts in labelling the packets in MPLS which are ‘Label Value’, ‘Experimental (EXP)’, ‘Bottom of Stack (S)’ and ‘Time to Live (TTL)’. Label consists of 20 bits, EXP possesses of 3 bits, S involves 1 bit and TTL contains 8 bits which makes the overall size of MPLS label is 32 bits or 4 bytes.

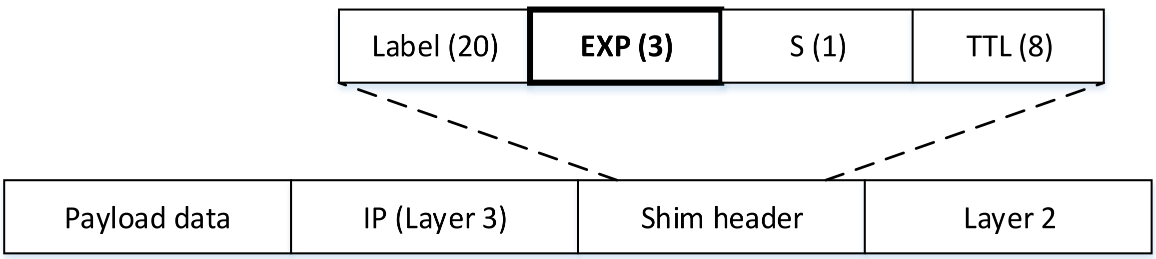

Figure 2.2 MPLS label in Shim header

As shown in Figure 2.3, there are several components involved in making MPLS works i.e. Ingress Label Edge Router (LER), Label Switch Path (LSP), Label Switch Router (LSR) and Egress LER [51].

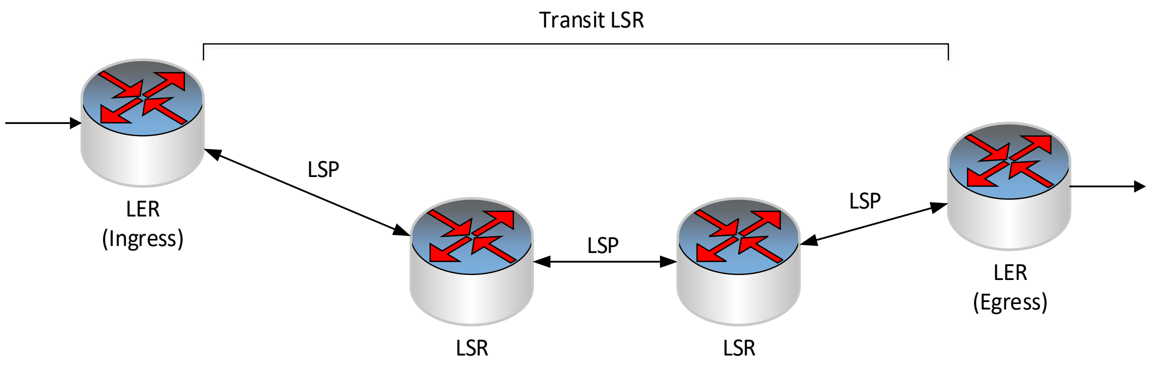

Figure 2.3 Components in MPLS network

Ingress LER operates at the edge of MPLS network by determining the final destination router and applying the right labels to packets through IP lookup. It is the first router which is responsible to encapsulate a packet to make the initial path selection.

Then, the packets will be forwarded to LSR via its label to route the traffic to its desired destination without needing to perform additional IP lookups. This is because LSR is functioning to switch the labels and route the packets to the next available LSR or egress LER. LSR is a router which only does the label switching in the middle of their journey by removing the old label and replaced it with a new one as it is located in the middle of MPLS network.

LSP has a pre-determined path which is a route through the MPLS network that is established from source to destination by referring to the applied labels. It is a unidirectional tunnel between a pair of routers across the MPLS network in which a clear perspective of LSP between routers is needed in order to forward the packets to their next destination efficiently.

The label of the packet is then removed at Egress LER as it is the final router which is located at the end of LSP tunnel. After the label is removed, the packet will be forwarded using the IP protocol.

There are two main protocols for MPLS routing namely Label Distribution Protocol (LDP) and Resource Reservation Protocol with Traffic Engineering (RSVP-TE) [52, 53]. LDP is a simple non-constrained protocol which does not support traffic engineering mechanisms whereby, RSVP-TE is a complex protocol with more overhead but supports traffic engineering events via network resource reservation.

MPLS allows traffic engineering implementation with the ability to control the chosen path for traffic to be routed in network, to prioritize different traffics and to prevent congestion at each router. Besides, MPLS permits multi-service network implementation which is the capability to deliver data transport and IP routing services across the same packet switching network infrastructure. In addition, MPLS is able to improve network resiliency with MPLS fast reroute.

2.5.1 Routing or Load Balancing in Power Grid

Smart grid network should have a QoS management system including requirements on the functional behaviour, robustness, reliability and timeliness since data transmission is a critical issue especially when it is involving real-time monitoring and control system [3, 54]. Thus, routing or load balancing is an important criteria in distribution division as to make sure that each information are transmitted within their requirements and consistent from substations to the control center and vice versa. Routing is a process of packets transmission across the network through a chosen path from one host to another which is performed by dedicated devices called as routers. Load balancing is a feature that is responsible to distribute network traffic among multiple servers to avoid overloading on any of the host for network performance improvement.

The QoS differentiation for smart grid in NAN through multiple Routing Protocol for Low Power and Lossy Networks (RPL) instances was proposed by Rajalingham et al. [55]. The proposed QoS mechanism involves MAC and Network layers in which prioritization level of MAC is to focus on channel access instead of queue scheduling. Two traffic classes were introduced with low and high priorities. Low priority is dedicated for periodic meter reading that requires medium reliability as both delay and jitter are tolerable. High priority is for alert messages with stringent delay requirements for notification purposes. Three variants of RPL are studied i.e. single instance RPL, multi-instance RPL (RPL-M) and multi-instance RPL with prioritized channel back-offs (RPL-M+). RPL-M is able to meet all traffic class requirements and has outperform both RPL and RPL-M+ performances. It is because RPL alone is unable to support the various QoS requirements whereas, the premature channel access in RPL-M+ leads to extensive back-off which produce high amount of delay.

Sahin et al. has proposed QoS-aware routing algorithm over single-path and multi-path routing for smart grid applications based on wireless sensor network [56]. Both single-path and multi-path routing are compared to evaluate their capabilities for service differentiation in terms of reliability and timeliness. The simulations have been carried out with four different scenarios in which different sets of reliability and timeliness requirements are involved. Several metrics are observed for performance evaluation which include average delay, on-time reliability, end-to-end delay and packets control.

Another method is to use Multi-class Multipath Routing Protocol for Low power and Lossy Networks (M^2RPL) and Heuristic Optimal Load Distributor (HOLD) [57]. This method is to develop packet scheduler and allow equalization in distributing load among traffics so that maximum lifetime can be achieved. Adaptive weighted round robin which is one of the packet forwarder technique available in Internet Protocol-Multiprotocol Label Switching (IP-MPLS) is chosen to be used in this method. DiffServ principal approached in QoS is referred to qualify data flow into different classes and thus, three different classes of service are available comprise of Expedited Forwarding (EF) as the highest priority, Assured Forwarding (AF) as medium priority and Best Effort (BE) as the lowest priority. The efficiency from the proposed framework leads to improvement in lifetime and average delay by 30% and 10% respectively whereby, reliability is observed to remain the same without any improvements.

A Gossip-based Local Recovery Policy (GLRP) is proposed as congestion control mechanism by Rodríguez-Pérez et al. [58]. Congestion control is used to set up static LSPs that are congruent by having pre-defined backup paths in case of link failure. Hence, reliability and performance of prioitized QoS-aware service can be improved as the number of packet loss is reduced. Besides, GLRP is cooperated with RSVP-TE protocol in obtaining local retransmission of lost traffic via gossip nodes in case of LSP failure occurs which then, the lost packet will be forwarded towards its destination. It was found that efficiency of the proposed policy is not affected by the large size of data buffer since it is depending on percentage of packets loss. Instead, an optimum size of data buffer is required for gossip nodes to reduce congestion level in the network. When speed of link increases, ratio of data packets could also increase which lead to lesser delivery time for gossip request. Packets of data can be stored temporarily at node and it will be replaced once there is new incoming data packet.

Pijanka et al. has proposed a conceptual mobile Multiprotocol Label Switching-Transport Profile (MPLS-TP) with the use of OAM channels for aggregation and access networks in which both of them are located in distribution division to support the mobility of terminal devices or users [59]. The concept is to optimize the handoff or handover management procedure without need to modify MPLS standard significantly to produce mobile MPLS-TP since all signalling messages will be transmitted by the standardized OAM channels. Transmission break time of mobile MPLS-TP within four different priority levels in its hierarchy was found to be shorter than mobile MPLS.

Another possible method is to use Hybrid Wireless Mesh Protocol (HWMP) based on NAN of QoS-aware routing scheme namely HWMP-NQ [1]. HWMP-NQ is an integrated routing metric which decides the chosen route together with effective link condition and queue optimization via NS-3. Different applications including control information data, demand response, requested AMI, power quality data, meter reading data and video surveillance are distinguished into four different traffic classes which is categorized from high to low priority respectively. Data size and transmission rate were considered during simulation of HWMP-NQ which shows an improvement in terms of average packet delivery ratio, delay, routing control information overhead and throughput.

A routing metric known as Interference Aware Expected Transmission Time (IAEET) is functioning to capture effect resulted from intra- and inter-flow interference whereby an interference aware QoS routing protocol namely Interference-Aware Expected Transmission Time Routing (IAEETR) is to identify route with minimum interference and delay for data transmission [60]. Combination of both proposed routing techniques have improved delay and packet delivery ratio with multiple connections simultaneously. Packets are competing in accessing a channel according to their designated priorities where high priority traffic will have small delay compared to low priority traffic. Four traffic classes are used in prioritizing different traffics varying from high to low priority respectively.

A combination of NAN QoS-Aware and Load-Balance Routing Scheme (NQA-LB) with Adaptive Priority Adjustment Scheme (AP-EDCA) is able to modify priority of packets under low load condition to increase their throughput [61]. Besides, collision can be mitigated under heavy load condition in order to improve reliability. Low load and heavy load are simulated to differentiate various NAN applications through DiffServ principal approached. The simulation was done using NS-3 and showed improvements in terms of packet delivery ratio, delay and throughput.

Communication in smart grid involves heterogeneous networks (HetGrid) which is the combination of Internet and private networks as there are numerous smart grid applications with different requirements. Demir et al. [62] proposed three different routing concepts namely Source Routing-based QoS Routing (SRQR), Altruistic Resource Allocation (AFA) and Compensative Multi-Routing (CMR). SRQR is functioning to find the best path, AFA is to reserve the best path for highly critical applications and CMR is to employ suitable paths for multipath routing. Distribution Automation (DA) applications are considered for simulation of the proposed concepts and they have been categorized into six different traffic classes. The six traffic classes are Ash for high sensing applications, Asm for medium sensing applications, Asl for low sensing applications, Ach for high controlling applications, Acm for medium controlling applications and Acl for low controlling applications. The proposed concepts have improved reliability, delay and bandwidth of DA traffics in HetGrid environment.

Table 2.1 shows summary of routing or load balancing analysis for QoS management system in smart grid network.

Table 2.1 Summary of routing or load balancing analysis

Methods	Analysis	Parameters Study
Proposed QoS mechanisms for RPL consisting of MAC layer and Network layer of QoS [55]	• Distribution division • Two traffic classes (low and high priority) • Study three variants of RPL (RPL, RPL-M and RPL-M+) • Using OMNET++ and MiXim	• Packet delivery ratio • Aggregate packet transmission rate • Delay
Proposed QoS-aware routing algorithms [56]	• Distribution division • Use single-path and multi-path • Evaluate reliability and timeliness • Performance evaluation metrics (average delay, on-time reliability, end-to-end delay, and control packets)	• Number of nodes • Number of traffic flows • Packet length • Traffic type • Average delay • On-time reliability • Number of control packets • Data packets
Proposed M^2RPL and HOLD methods [57]	• Distribution division • Develop packet scheduler • Load distribution equalization • Adaptive weighted round robin packet forwarder (IP-MPLS) • Three service classes (EF, AF and BE) • DiffServ principal • Improve lifetime and delay • Remain reliability	• Topology (area size, population, number of sinks and distribution) • Links (packet loss ratio) • Nodes (initial energy, transmission (TX) energy, received (RX) energy, physical type, MAC type and TX/RX range) • Traffic (generation function, EF, AF, BE and packet size) • End-to-end delay • Network lifetime • Percentage of packet drop • Number of node (node population) • Reliability constraint (packet delivery requirement) • Packet delivery ratio • Collision rate

Methods	Analysis	Parameters Study
Proposed GLRP and cooperates it with RSVP-TE [58]	• Distribution division • GLRP as congestion control mechanism • Improve reliability and performance of prioritized QoS-aware service • Self-management capable routes • Obtain local retransmission of lost traffic • Gossip nodes for tracking lost packet • Minimize packet loss	• Buffer-hit-ratio • Incoming packet ratio • End-to-end delay • Packet loss • Delivery ratio of locally recovered packets • Number of successful buffer reads • Number of buffer accesses • Total number of delivered packets • Delay for every packet delivered at egress node
Proposed conceptual mobile MPLS-TP with the use of OAM channels for aggregation and access networks [59]	• Distribution division • Involve aggregation and access networks • Support mobility of users • Optimize handover management procedure • Not required to modify current MPLS standard significantly	• Number of LERs • Number of subordinate LSRs • Number of handoffs • Movement type (progressive) • Type of traffic class • Number of neighbourhood base stations • Average transmission break time • Number of signalling message transmission • Number of network levels • Link delay • Switching time • Routing tables update time

Methods	Analysis	Parameters Study
Proposed HWMP based in NAN for QoS-aware routing scheme namely HWMP-NQ [1]	• Distribution division • Integrated routing metric • Decide chosen route with effective link condition probing and queue optimization • Using NS-3 • Four traffic classes (AC_VO, AC_VI, AC_BE, and AC_BK) • Improve average packet delivery ratio, delay, routing control information overhead and throughput	• Control information data (AC_VO with priority 0) • Demand response, requested AMI, and power quality data (AC_VI with priority 1) • Meter reading data (AC_BE with priority 2) • Video surveillance (AC_BK with priority 3) • Application size • Transmission interval • Requirement latency • Average throughput • Number of nodes • Average packet delivery ratio • Average end-to-end delay • Average overhead • Failure rate of nodes
Proposed routing metric of IAEET and Interference aware QoS routing protocol of IAEETR [60]	• Distribution division • Improve delay and packet delivery ratio with multiple connections simultaneously • IAEET captures effect resulted from intra- and inter-flow interference • IAEETR identify better route with minimum interference and delay for transmission • Using NS-3 • Prioritize different traffics • Four traffic classes (AC_VO, AC_VI, AC_BE, and AC_BK)	• Simulation time and area • Communication range • Sensing range • Propagation delay model • Propagation loss model • Bandwidth • AMI management (AC_VO with priority 0) • Requested power quality (AC_VI with priority 1) • Periodic power quality (AC_BE with priority 2) • Video surveillance (AC_BK with priority 3) • QoS identification • Transmission interval • Application size • Average end-to-end delay • Average packet delivery ratio

Methods	Analysis	Parameters Study
Proposed NQA-LB and AP-EDCA [61]	• Distribution division • Using DiffServ principal • Involved various NAN applications • Able to adjust priority of packet • To increase throughput under low load condition • To mitigate collision under heavy load condition • Improve packet delivery ratio, delay and throughput • Using NS-3	• Simulation time • Physical standards • Rate adaptive • Transmission interval • Number of packets • Application size • Requirement latency • Average packet delivery ratio • Number of nodes • Average end-to-end delay • Average throughput
Proposed HetGrid for Internet and private networks [62]	• Distribution division • SRQR is to find the best path • AFA is to reserve the best path for high critical traffic • CMR employs adequate paths for multipath routing • Consider DA applications • Improved reliability, delay and bandwidth	• Message size • Types of applications models (Ash, Asm, Asl, Ach, Acm and Acl) • Priority • Delivery rate of sensing apps • Delivery mode • Latency, bandwidth and loss rate of adjacent links • Current reliability of link • Current latency of link

In this thesis, both routing and load balancing in power grid are not considered as there are already plenty of research work done in the area for MPLS previously. Besides, prioritization of traffic in terms of routing and load balancing have limit the QoS performance by pre-setting the priority of certain traffics without allowing any modifications to be made. Routing and load balancing are used in choosing the right path for packet transmission and distributing network traffic between several servers. The usage of QoS in these field is more general instead of controlling the traffics individually as each of them have different characteristics and requirements that have to be complied with [63, 64].

2.5.2 Protection

Communication service in smart grid requires protection of information that are transmitted between substations and control center. It is highly critical to support electrical power system operations for the whole nation. It addresses the unintentional situations due to failures of equipment, system and network, human error, natural disaster and unexpected phenomena in grid infrastructure. Protection is important for system reliability to operate effectively in case of failure in smart grid network system. Prediction of the failure is able to prevent unexpected events from occurring. If failure occurred, the identification of it must be done in the shortest possible time to avoid any damage. Diagnosis of the failure can be done in several ways for failure evaluation to tackle failure recovery.

An Optical Carrier Ethernet Switch (OCES) was then proposed by Kim et al. [65] by using the existing Protection State Coordination (PSC) as protection mechanism for MPLS-TP packet transport network. Besides, MPLS-TP equipment from Electronics and Telecommunications Research Institute (ETRI) has also been introduced. It was shown that the performance of OCES is less than 50 *ms* for protection switching and it possessed characteristics as service provider equipment that includes availability, deterministic and scalability which makes it to have the ability to protect the LSP tunnels. During the restoration process, there is no packet loss due to merging selector which functioning to reroute the traffic from faulty path to working path.

Ryoo et al. has introduced Automatic Protection Coordination (APC) as a unified solution for linear protection switching mechanism [66]. The proposed APC is able to improve the pre-existing solutions comprise of PSC and Automatic Protection Switching (APS) solutions. Another important feature of APC is that it is backward compatible with the installed equipment without the needs to roll out the existing equipment for the whole network infrastructure. APC solution is also applicable to several type of topologies including ring and shared mesh protection switching.

Another possible method is an Integrated Proxy Mobile MPLS-TP (IPM-TP) as proposed by Cortés-Polo et al. [67]. IPM-TP has the ability to reduce overhead using MPLS-TP tunneling mechanism while improving QoS in wireless heterogeneous network with high rates of mobility. The main function of IPM-TP is to serve as path-protection mechanism and able to support dynamic topology changes and network optimization. High value of ratio of session arrival rate-to-mobility rate represents low mobility which incurred low cost of signalling update. Besides, creation of new LSP tunnels can be reduced in the event of handoff procedure as creation of new tunnels can increase the probability of serivce blocking in local mobility anchor.

Collective Signal Fail (C-SF) mechanism is proposed as a solution for per-leaf protection scheme [68] which fast traffic recovery can be obtained while reusing the existing technologies of protection switching with minimal modification. It is related to Inter-Processor Communication (IPC) which is the time that could affect the restoration time for signal fail notification in case of network failure. Another protection scheme that has been chosen is tree protection scheme due to its ability to increase the agility of connection which makes it to have the best performance compared to other protection schemes in terms of restoration time. Thus, hybrid protection scheme that combine tree and per-leaf protection schemes are proposed as it can improved network availability by utilizing bandwidth resources.

Choi et al. proposed Path Computation Element (PCE) management framework architecture and integrates with MPLS-TP switches and Generalized Topology Discovery, Operational Monitoring and Provisioning (G-TOP) controller to collect the information on network topology for unified control and monitoring [69]. The proposed solution is done by setting up LSP tunnel so that both states from and to the control center can be monitored in unified way. From the simulation, it shows that both LSP provisioning time and path computation time increased proportionally to the number of nodes except for LSP updating time. These results indicate that any deficiency in its function can be identified and correct protocol design can be ensured.

Table 2.2 shows summary of protection analysis for QoS management system in smart grid network.

Table 2.2 Summary of protection analysis

Methods	Analysis	Parameters Study
Proposed OCES using the existing PSC format and introduced ETRI's MPLS-TP equipment [65]	• OCES performance for protection switching is less than 50 *ms* • OCES has requirements as service providers equipment that is able to provide scalability, availability and deterministic performance • Able to protect LSP tunnels • No packet loss during restoration process due to merging selector • Traffic can be rerouted from faulty path to working path	• Average out-of-service time • Packet loss • Service interruption • Traffic restoration time • Confirmation time • Transfer time
Introduced APC solution [66]	• Unified solution for linear protection switching mechanism • Improved both PSC and APS solutions • Backward compatibility with installed equipment • Applicable to other topologies (ring and shared mesh protection switching)	• Signal fail • Signal degrade • Operator command (lockout of protection, forced switch, manual switch) • Timing of protection switches • Wait-to-restore
Proposed IPM-TP [67]	• Able to reduce overhead using MPLS-TP tunnelling mechanism • Provides QoS in wireless heterogeneous network • Provides high rates of mobility • Serves as path-protection mechanism • Supports dynamic topology changes and network optimization • Reduces signalling and creation of new LSP tunnels in the event of handoff procedure	• Signalling cost • Velocity of mobile node • Packet delivery cost • Session arrival rate to a mobile node • Total cost • Session arrival rate to mobility rate • Service blocking probability

Methods	Analysis	Parameters Study
Proposed C-SF mechanism and hybrid protection schemes [68]	• Fast traffic recovery • C-SF as a solution for per-leaf protection scheme • Related to IPC • Tree protection scheme is to increase agility of connection and has the best performance • Hybrid protection scheme improves network availability	• Restoration time • Number of LPPs (linear protection processes) • Confirmation time • Transfer time • Number of affected leaf nodes • Data rate • Average received data rate • Hold-off time interval • IPC processing time • End-to-end delay
Proposed PCE and integrates with MPLS-TP switches and G-TOP controller [69]	• Based on unified control and management framework architecture • To collect information on network topology • Setting up LSP tunnel • Monitoring states in unified way • Identify deficiency in its function and ensuring correct protocol design	• Protection switching time • Throughput • Delay • Traffic load • Number of nodes • Packet size

A lot of research work involving protection in MPLS have been done previously in assuring security of the network system in power grid. Most works are focusing on protection switching time in case of service interruption. Protection of power grid network covers QoS aspect as it functions to protect the LSP tunnel and reroute the traffic in case of faulty path without considering the traffic individually as each of them have different characteristics and requirements that have to be complied with. All these reasons explain on why protection is not considered in this thesis.

2.6 Traffic Engineering Algorithm in Packet Switched Network

Generally, traffic engineering is dealing with design of traffic controlling associated with any selected protocol for a smooth traffic flow in network system. In this thesis, traffic engineering is responsible for optimizing the performance of communication network in smart grid environment by dynamically analysing and regulating the behaviour of transmitted data over heterogeneous network. This technique is applied in controlling and changing the traffic ranking based on priority level upon their data size, data rate and previous transmission delay. Traffic engineering covers three different areas including wireless, fiber-wireless and fiber optic.

Bandwidth aware scheduling with SDN (BASS) was proposed by Qin et al. [70] in an open source implementation namely Hadoop. The proposed scheme is to join Hadoop system with SDN technology for tasks assignment to be done in global view or when there is available bandwidth as the foundation of scheduling activity. SDN technology is specifically utilized for job scheduling of big data processing in Hadoop cluster as it is able to guarantee data locality in global view and efficiently allocate tasks in optimized way. Besides, unusual network bandwidth from OpenFlow controller is considered to be important for task scheduling. Function of BASS algorithm is to handle and assign bandwidth in time slot manner through SDN utilization. In addition, BASS is responsible in allocating task locally or remotely depending on the completion time. However, there is no traffic classification, prioritization and contextual aware in BASS as this algorithm is focusing on bandwidth aware.

Another proposed algorithm is the collaboration of multi path Transmission Control Protocol (MPTCP) with segment routing in multi rooted data center network topologies by Pang et al. [71]. The aims are to provide better solution in traffic management especially during peak hours of data center network and to resolve issue on resource consumption in SDN-based data center network. Architecture of data center network is considered based on SDN technology which comprises of three layers namely control layer, data layer and host cluster. Control layer is responsible for path allocation of sub-flows by SDN controller whereby, data layer consists of SDN switches that utilize MPLS technology for segment routing implementation that is accountable in defining chosen path. Host cluster is supported by the proposed protocol namely MPTCP that is in charge of traffic flow transmission and to maintain communication with control layer by allowing flow splitting. However, there is no traffic classification, prioritization and contextual aware involved.

A detection device namely Radio Frequency Partial Discharge (RFPD) is proposed for condition monitoring and integrating sensor signals into smart grid monitoring system by Baki et al. [24]. The RFPD detection system applied conversion technique from partial discharge pulse into milli-ampere (*mA*) signal. The communication channel for remote online monitoring from the control center is based on MPLS protocol. This enables online diagnostic from the remote online monitoring and facilitate preventive maintenance by predicting the failure risk which can minimize the number of outage rate. However, there is no involvement of traffic classification, prioritization and contextual aware in this proposed solution.

Optimized Multi-Class of Routing Protocol for Low Power and Lossy network (OMC-RPL) is proposed by Alishahi et al. [72] as the virtual version of RPL protocol. It is a primary solution for last mile communication of NAN network in smart grid environment. OMC-RPL runs centrally in a controller by utilizing distinctive metrics related to QoS performance to support data classification. There are four traffic classes that is categorized according to weighting parameters including required delay and required packet loss. OMC-RPL is implemented in network architecture where SDN technology is supported consisting of three different parameters namely Network Link Discovery, Topology Manager and Virtual Routing. Network link discovery is accountable for discovering and maintaining the status of all physical links in network, topology manager is responsible to build and maintain the global network topology and virtual routing is to provide interoperability between SDN controller and other SDN nodes in network. Although traffic classification and prioritization have been done, there is no contextual aware available in managing the smart grid traffics with the implementation of MPLS technology.

Alhowaidi et al. [73] proposed Greedy-Heuristic algorithm that comprises of cost-based and load-based algorithms. It is proposed due to problem of resource scheduling and bandwidth assignment to cater dynamic demands in cloud network as demand is increasing while resources are limited. Greedy-Heuristic algorithm acts as a dynamic resource allocation in multi-layer of optical backbone network. Resource allocation involved routing and wavelength assignment whereby multi-layer involved IP-MPLS and WDM layers. Cost-based algorithm is to serve demands assignment to the lowest cost data center whereas, load-based algorithm is functioning to allocate demands to data center that has the lowest number of running demands. Generally, these proposed algorithms only consider dynamic provisioning of cloud resources which is bandwidth scheduling according to demands of user with traffic classification and prioritization while no contextual aware of traffic scheduling is taken into account.

Al-Anbagi et al. has proposed Adaptive QoS scheme (AQoS) and Adaptive Guaranteed Time Slot (AGTS) allocation scheme for delay critical smart grid applications in distribution division [54]. The proposed algorithms are done specifically for IEEE 802.15.4 based on Wireless Sensor Network (WSN) which involved cluster-tree and mesh topologies. AQoS is implemented for cluster-tree topology to modify the guaranteed time slot adaptively based on request made from the end devices. AGTS is implemented for mesh topology to tune the time slots dynamically depends on various traffics and network conditions. Besides, multiple number of time slots to sensor nodes for high priority data can be granted adaptively until minimum amount of delay is achieved. Hence, end-to-end delay for critical data can be reduced while maintaining the reliability and energy efficiency values in acceptable manner. However, this solution is not preferred since it is done for wireless communication and contextual aware is not taken into account.

Another possible solution is multi-QoS data traffic scheduling algorithm (Modas) as proposed by Liu et al [74]. It is applicable in sustaining cyber physical system in smart grid environment which includes multiple traffics from secondary equipment, distributing generation and dispatching center to be able in exchanging electrical information effectively via resources allocation fairly. Hence, congestion can be avoided during high traffic load. Network architecture of QoS for this solution is integrated with DiffServ principal and MPLS technology in order to support complex traffic control in cyber physical system. However, no contextual aware of traffic scheduling is included in this proposed solution.

Kamoun et al. [75] has proposed hard pipe approach in IP-MPLS network as an advanced networking architecture in providing low delay, high reliability and guaranteed end-to-end bandwidth. It is specifically to support ubiquitous critical applications with stringent end-to-end bandwidth requirement, ultra-low delay and restoration time in case of failure. Hard pipe stratum is able to guarantee time slot for each critical traffic flows so that no contention can happen between different flows. All traffics are encapsulated with MPLS label and dedicated tunnel is exclusively reserved with respect to specific traffic that possess the same labelling. Therefore, routers are able to differentiate hard pipe traffic from the soft pipe traffic that is also known as regular IP-MPLS traffic with less critical applications. Hard pipe services are scheduled for high priority traffic whereby, soft pipe services are scheduled for normal IP-MPLS traffic with low priority. However, transmission between high and low priority traffics are not flexible as each of them have fixed priority with their own dedicated tunnel regardless of content in each of their packet.

QoS guarantee in smart grid infrastructure is done by RIO algorithm which is based on queuing discipline communication via traffic classification [51]. Various types of applications are categorized into four traffic classes by referring to DiffServ principles which are Class 1 is for low delay and low packet loss traffics, Class 2 is for high bandwidth, low delay and low packet loss traffics, Class 3 is for moderate bandwidth traffics and Class 4 is for delay tolerance traffics. These traffics are prioritized based on service delivery time and required amount of bandwidth which then DiffServ Code Point (DSCP) are assigned to each packet before transmission takes place. The simulation has been carried out with IP-MPLS protocol using OPNET as simulator platform and shown an improvement has been made in terms of delay. However, each traffic has fixed priority without having the ability to change the priority depending on certain situation. Therefore, a contextual aware traffic scheduling algorithm is required in order to fulfil this desire.

Another solution is proposed by Rezaee et al. [76] which suggested Wide Area Measurement System (WAMS) communication infrastructure with utilization of SDN technology that is functioning to measure, collect and analyze data in power system. Phasor Measurement Units (PMU) and Phasor Data Concentrator are essential components of WAMS communication. Utilization of SDN infrastructure in WAMS enables the best path for each PMU to be obtained, allows switches to be configured dynamically in providing required bandwidth of each PMU and selecting backup path for each PMU to increase network reliability. There are two traffic classes that have been considered and context-aware Active Queue Management is responsible to improve delay performance of PMU flows by dropping less important packets that contain older data. Hence, available bandwidths are assigned for high priority packets that contain newer data. Performance of delay, jitter, bandwidth and fairness have improved due to the proposed model which has been implemented in Mininet environment using Ryu controller. However, there are various traffics with different requirements in smart grid environment whereas Rezaee et al. consider only two traffic classes. Besides, context aware is defined to drop packets containing older data rather than taking them into account and transmitting them as defined in this proposed solution which is utilizing SDN technology.

Thus, in this thesis, a new contextual aware traffic scheduling algorithm is proposed where it supports QoS through four traffic classifications with different priorities of each class in order to meet various smart grid traffic requirements. Context aware will be done using MPLS technology as SDN is unable to guarantee end-to-end information protection and it is vulnerable to intrusion and cyber-attacks as it involves remotely connected devices in networking system with no direct physical control. It is also not done in SDH/SONET environment to support grid of the future. SDH/SONET is in the process of migrating to MPLS due to product aging, lack of expertise and can no longer cope to new IP based applications.

Table 2.3 shows the summary of traffic engineering algorithms analysis for QoS management system in smart grid network.

2.7 Context Aware Traffic Scheduling Algorithm

Context aware is defined as a system that is able to collect information and analyze the data with regards to its environment in order for the system to adapt automatically at any given time, according to its situation. The system is using either software or hardware platform in realizing the context aware system.

Context includes any kind of data or information which are related to a given device or an application.

Table 2.3 Summary of traffic engineering algorithms in packet switched network

QoS Algorithm	Analysis	Technology	Advantages	Research Gap
Proposed BASS [70]	<u>BASS:</u> • Utilizes SDN to manage bandwidth and allocates it in a time slot manner • Decides whether to assign a task locally or remotely depending on the completion time	SDN	• BASS flexibly assign tasks in an optimized way	No traffic classification, prioritization and contextual aware
Proposed MPTCP [71]	<u>MPTCP:</u> • Aims to provide better traffic management solution especially during peak hours of data center network • To resolve resource consumption problem in SDN-based data center network	SDN	• Able to provide better traffic management solution especially during peak hours of data center network	No traffic classification, prioritization and contextual aware and also utilizing SDN technology
Proposed RFPD [24]	<u>RFPD:</u> • Apply conversion technique from partial discharge pulse into *mA* signal • Allow remote online monitoring from control center via MPLS as data communication channel • Enable online diagnostic • Preventive maintenance • Predict failure risk • Minimize number of outage	MPLS	• Support online monitoring for remote control of smart grid devices	No traffic classification, prioritization and contextual aware
Proposed OMC-RPL [72]	<u>OMC-RPL:</u> • A virtual version of RPL protocol • Primary solution for last mile communication network in smart grid (NAN) • Run centrally in the controller • Utilizes distinctive metrics related to QoS	SDN	• Reduce complexity and controls interactions to distribute network states and other related information in network	No contextual aware for various traffic and utilizing SDN technology

QoS Algorithm	Analysis	Technology	Advantages	Research Gap
Proposed Greedy-Heuristic algorithm [73]	<u>Cost-based algorithm:</u> • Assignment of demands to the lowest cost data center <u>Load-based algorithm:</u> • Assigns the demands to the data center which has the lowest number of running demands	MPLS	• Evaluate if a specific demand can be accepted by the scheduler with the available data center resources and bandwidth	Only considers dynamic provisioning of cloud resources which is bandwidth scheduling for user demands without contextual aware of traffic scheduling
Proposed AQoS and AGTS [54]	<u>AQoS</u>: • For cluster-tree topology • Modifies guaranteed time slot adaptively based on requests made from end devices <u>AGTS:</u> • For mesh topology • Flexibly tune time slots according to various traffics and network conditions • Grants multiple number of time slots to sensor nodes of high priority data adaptively until minimum delay is achieved	MPLS	• Flexible in tuning the guaranteed time slot for delay critical smart grid applications • Support traffic scheduling for smart grid applications	No contextual aware of traffic scheduling
Proposed Modas [74]	<u>Modas:</u> • Applies in sustainable cyber physical systems • Multiple traffic from secondary equipment, distribute generation and dispatching centre can effectively exchange electrical information • Avoids congestion in high traffic load • To support the complex traffic control in cyber physical system	MPLS	• The algorithm can count fast forward path with multi-agents • Can guarantee transmitting quality	No contextual aware of traffic scheduling

QoS Algorithm	Analysis	Technology	Advantages	Research Gap
Proposed hard pipe approach in IP-MPLS networks [75]	<u>Hard Pipe:</u> • An advanced networking architecture • To provide end-to-end guaranteed bandwidth, low delay and high reliability • To support pervasive mission-critical applications • To support stringent end-to-end guaranteed bandwidth requirements, ultra-low delay and failure restoration times • Guarantees a time-slot for each traffic flow • All traffic are MPLS-encapsulated with dedicated tunnel labels that are exclusively reserved • Scheduled with the highest priority	MPLS	• To enhance bandwidth usage, provide support for delivering end-to-end QoS and maximize revenues	Transmission of traffics are not flexible as each of them have fixed priority with dedicated tunnel
Proposed RIO [51]	<u>RIO:</u> • Queuing discipline • Traffic prioritization based on service delivery time and required amount of bandwidth • Assigned DSCP • DiffServ principle • IP-MPLS protocol • Improved delay	MPLS	• Assign appropriate code point for traffic classification	No contextual aware of traffic scheduling for various traffics

QoS Algorithm	Analysis	Technology	Advantages	Research Gap
Proposed WAMS and context-aware Active Queue Management [76]	WAMS: • WAMS is to measure, collect and analyze data in power system Context-aware Active Queue Management: • To improve delay performance of PMU flows in WAMS networks • Dropping less important packets which contain older data • Available bandwidths are allocated to high priority packets which contain newer data	SDN	• Backup path is provided for each PMU to increase network reliability • Improve delay, jitter, bandwidth and fairness performance	Only consider two traffic classes in smart grid environment and context-aware is defined as packet containing older data to be dropped

A context aware system may accumulate the relevant information based on the defined context via sensors equipped at each device or equipment available in substation to establish a newly-defined ranking for smart grid traffics. Context aware can conduct and direct services in power distribution network and enhance the collected data delivery in reducing delay and jitter of traffics by adapting to its context.

With the usage of contextual aware in scheduling various traffics in smart grid, priority of each traffic can be changed dynamically according to its context so that communication services are provisioned, managed and maintained accordingly. The traffic priority ranking can be rearranged according to their criticality and sensitivity by firstly releasing the highest priority ranking traffic followed by the lower rank traffics accordingly.

Context aware traffic scheduling algorithm is still new for MPLS, but it has been done in other network before. Yunfei et al. [77] has proposed a data processing model with context awareness ability in power communication network namely Context Aware Data Processing Model (CDPM). This model is designed with three different layers which consists of context aware data gathering layer, context aware data storing layer and context aware data computing layer. In context aware data gathering layer, two kinds of filter have been designed comprising of attribute and rule filters in order to exclude invalid data which is inaccurate and inefficient in real world. For context aware data storing layer, the collected data will be categorized into several types which are static and stream by classifier. In case of failure in the system, the lost data stream can be recovered with recovery mechanism. There is real-time feature extraction algorithm in context aware data computing layer which functions to detect and support equivalent between the new incoming and previous collected context aware data.

Context Aware Opportunistic Resource-Based Routing Protocol (CORB) has been proposed by Elias et al. [78] for static wireless sensor networks. This approach focuses on limited resources of nodes and its context as root cause to unstable multi-hop connection. The limited resources comprise of battery level, rank of node, strength of connection and buffer space. Whereas, context of node includes processing capability, variety of data and flexibility that can either link to or separate from the network. CORB functions to raise the awareness of available resources between a node and its nearby neighbours in getting consistent communication through context aware routing. The context aware element in CORB allows a certain node in static wireless sensor network to select the best path for its routing.

Bhajantri et al. [79] has proposed a protocol with an implementation of context aware in securing the routing process of distributed sensor network. The main concern of this proposed protocol is to avoid attacks due to vulnerability of sensor networks towards the environment. Therefore, Context Aware Secured Routing Protocol (CSRP) has been introduced in selecting the cluster head efficiently, discovering the context, interpreting the context and securing the routing mechanisms. Context aware in CSRP is introduced in order to differentiate priority of data into three categories which are low, medium and high level for routing purposes.

Obaidat et al. [80] has proposed an architecture for middleware with context awareness capability namely Context Aware Middleware Smart Grid Communication (CAMSC) into the fundamental architecture of smart grid communication. CAMSC functions in adapting with context environment and handling the information according to the varying context. This proposed architecture is specifically designed for smart home application based on wireless network communication with contexts including delay, bandwidth, stability and connectivity. There are five distinct elements involve in CAMSC architecture which are context sensing, modelling, repository, reasoning and discovery. Context sensing is to collect data from various sensors, context modelling is to interpret the collected data, context repository is to store the interpreted data, context reasoning is to deduce high level context from low level data and context discovery is to deliver the processed information to its applications.

Another framework has been proposed by Choi et al. [81] by suggesting context aware in managing power equipment operation in smart cities. The communication between different applications can be synchronized in providing an excellence service and obtaining an optimal hardware devices performance. This proposed framework requires sensing technologies in collecting contextual information which is gathered via numerous sensors. Some of the vital desires in achieving intelligent power equipment with context aware technology comprising of gathered information with several contexts via sensors and low level of context need to be converted into high level of context based on the collected information.

Table 2.4 shows the summary of context aware functions for communication system in smart grid network.

Table 2.4 Summary of context aware functions in smart grid network

Methods	Analysis	Function of Context Aware
Proposed CDPM [77]	• Context aware data processing model • Safety of power generation equipment • Security of data transmission • Security of transmission and distribution lines	• To filter collected context aware data • To classify the context aware data • To detect and maintain similar data stream between new and previous data
Proposed CORB [78]	• Context aware opportunistic resource-based routing protocol • Limited resources • Context of node • Strength of connection between nodes • Rank of node with minimum number of transmission • Available buffer space	• To select the best path in routing
Proposed CSRP [79]	• Context aware secured routing protocol • Efficient cluster head selection • Context discovery • Context interpretation • Secure routing mechanism	• To secure routing in distributed sensor network • To prioritize data into three categories; low, medium and high transmission
Proposed CAMSC [80]	• Context aware middleware architecture • Context acquisition • Context aggregation • Intelligent context environment • Service adaptation • Manage information	• To collect data • To interpret data • To store data • To presume high level context • To deliver processed data

Methods	Analysis	Function of Context Aware
Proposed Context Aware Framework for Power Equipment in Smart Cities [81]	• Context aware framework • Context ontology • Specification of inference rules for context ontology • Context aware inference service	• To sense and collect various data • To convert low level context into high level

2.8 Summary

In this chapter, the fundamental of power distribution network in smart grid, network technologies, methods and protocols are presented where several of the proposed algorithms are discussed in detail in order to design and customize a new desired algorithm in the next chapter where it supports contextual aware traffic scheduling for power distribution network. From the literature, network technologies can be divided into two categories namely Operational Technology and Information Technology. Operational technology is to support substation applications which is critical and time-sensitive over TDM and Ethernet whereby, information technology is to support enterprise applications which is less critical and less time-sensitive over IT network and infrastructure.

There are two categories in networking methods comprise of circuit switching and packet switching networks. Circuit switching network is responsible to establish connection between devices over a fixed path whereas, packet switching network is responsible to transmit data that is split into various size of packets over multiple routes. Besides, packet switching network can be classified into three classes consisting of routing or load balancing in power grid, protection and traffic engineering in which all of them are supporting MPLS protocol.

This chapter also reviews traffic engineering algorithms that can be divided into three categories; traffic engineering algorithms that support MPLS protocol in wireless communication, fiber-wireless communication and optical fiber communication. An explanation on theory of context aware is also included in this chapter. With this background in mind, the next chapter will describe the proposed contextual aware traffic scheduling algorithm for power distribution network.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter proposes a new traffic scheduling algorithm called Contextual Aware Traffic Scheduling (CATSchA) algorithm for power distribution network. Four traffics are chosen to be tested in this algorithm namely SCADA, Smart Meter, CCTV and Internet. Firstly, characterization of traffics are explained in Section 3.2 to fulfil the first objective of this work. This section comprises of pre-defined traffic priority ranking, ranges of different parameters used for each traffic and their threshold values to justify each traffic classification. Next, CATSchA algorithm is presented in Section 3.3 to describe the development of proposed algorithm in details aided with flow charts and equations. This section is done to realize the second objective of this thesis. Since this algorithm is tested via simulation method, Section 3.4 consists of simulation tool, network environment and its topology. Simulation tool involves a selected simulator platform for the development, validation and testing of CATSchA algorithm. Network environment in utility system is portrayed together with the network topology of the proposed algorithm. Section 3.5 highlights the required system parameters in designing and developing the CATSchA algorithm. System parameters is categorized into two different categories namely design parameters and performance parameters. Parameters involved in designing CATSchA algorithm are types of traffic, data size, data rate, amount of offered load, number of packets transmitted and delay of previous transmission. Meanwhile, for performance parameters; delay, jitter and throughput are responsible in stipulating the performance of CATSchA algorithm. Chapter 3 is summarized in Section 3.6.

Figure 3.1 shows the overall research framework for this project which consists of three stages to meet each objective as in Section 1.6. Phase 1 is where literature review and study on traffic engineering algorithm have been done in order for smart grid traffics to be characterized and prioritized according to their QoS. Phase 2 is on developing and validating of CATSchA algorithm for smart grid network in MPLS environment.

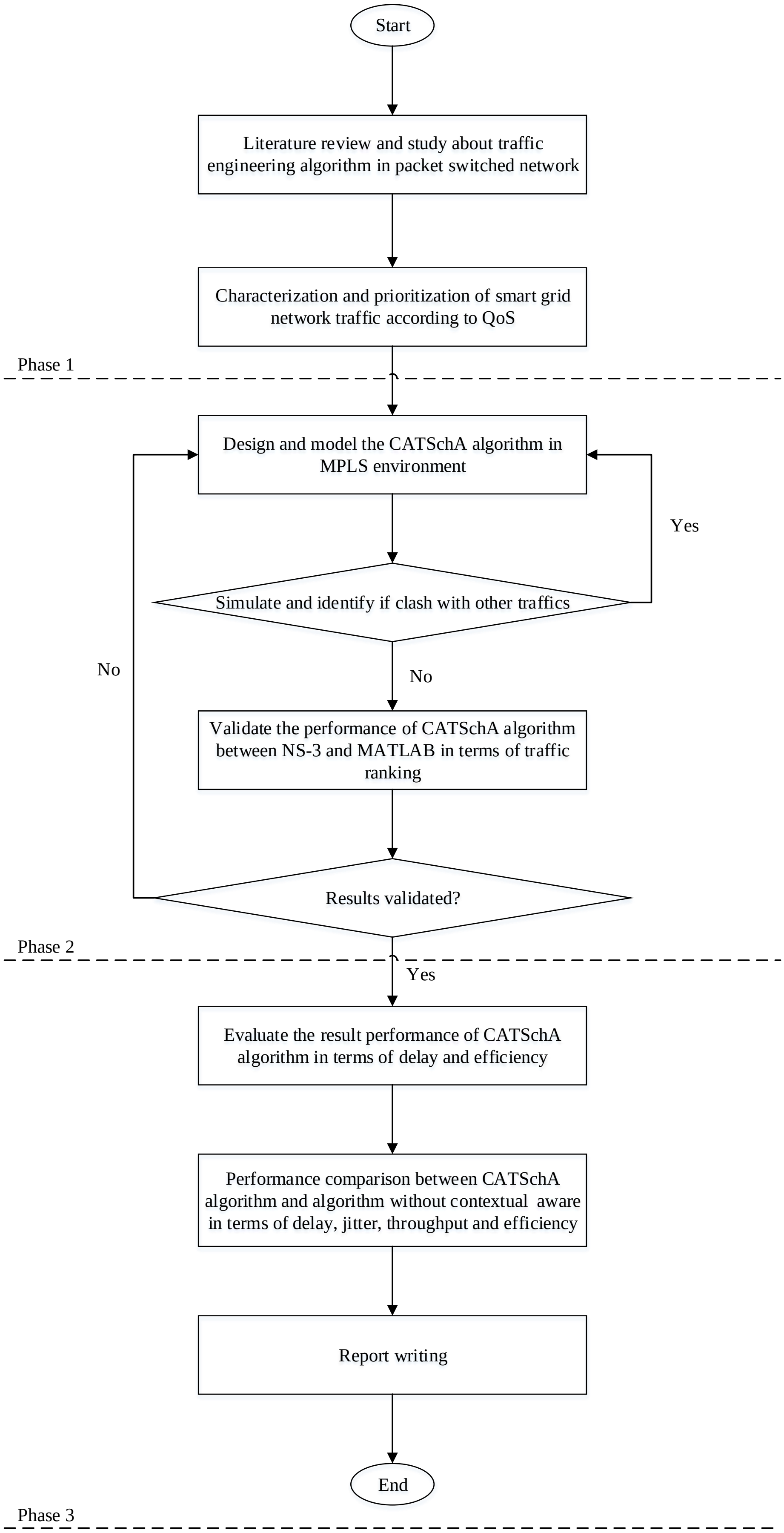

Figure 3.1 Research framework of overall project

The validation of CATSchA algorithm is done between NS-3 and MATLAB in terms of traffic ranking in Phase 2. The performance of CATSchA algorithm is then evaluated in terms of delay and efficiency for proof of the proposed concept. The performance between CATSchA algorithm is compared to algorithm without contextual aware in terms of delay, jitter, throughput and efficiency with real parameters as used in utility network. These are done in Phase 3. Finally, the report writing documentation is done as the last step in overall project.

3.2 Traffic Characterization

In this section, traffics in smart grid are characterized to different parameters in terms of data size, data rate and delay. Each traffic is categorized according to their specific requirements in power distribution network which produced a set of pre-defined priorities.

In the proposed CATSchA algorithm, characterization of each traffic is the first step that needs to be done to ensure that the algorithm supports QoS and Service Level Agreement (SLA). In order to support variable service quality necessities of heterogeneous traffic demand of smart grid applications along with their varying context, the traffic requirements are modelled in distinct priority classes.

Typically, teleprotection, SCADA, online monitoring system, Smart Meter, Internet, CCTV, Wi-Fi, smart building automation system, indicator control and fault recorders are some distinctive traffics available in smart grid distribution network. The traffics are mapped to the respective traffic classes as in Table 3.1 and Table 3.2 by taking into consideration their QoS specifications. Four different traffics are selected to be tested in CATSchA algorithm namely SCADA, Smart Meter, CCTV and Internet. These four traffics are preferred since they are used in IT technology which has less risk compared to OT technology for modification purposes as it is responsible in managing utility non-operational or unrelated in directly handling with critical and time-sensitive operational traffic flows [6, 82]. They are also the most significant traffics in IT technology that require proper handling as their streams of data are frequently transmitted from multi sites to the control center.

Table 3.1 Traffic classes for smart grid applications with service parameters

Classes	Data Size	Data Rate	Delay
Limited size/Critical rate/Real time ($S_1R_1D_1$)	Small	Low	Small
Limited size/Critical rate/Non-real time ($S_1R_1D_0$)	Small	Low	High
Limited size/Non-critical rate/Real time ($S_1R_0D_1$)	Small	High	Small
Limited size/Non-critical rate/Non-real time ($S_1R_0D_0$)	Small	High	High
Unlimited size/Critical rate/Real time ($S_0R_1D_1$)	Large	Low	Small
Unlimited size/Critical rate/Non-real time ($S_0R_1D_0$)	Large	Low	High
Unlimited size/Non-critical rate/Real time ($S_0R_0D_1$)	Large	High	Small
Unlimited size/Non-critical rate/Non-real time ($S_0R_0D_0$)	Large	High	High

Table 3.2 Smart grid applications mapping to traffic classes and their context identities

Smart Grid Applications	Traffic Classes	Context Identities
Teleprotection	$S_1R_1D_1$	High speed protection information
SCADA	$S_1R_1D_1$	Data poll response
Smart Meter	$S_1R_1D_0$	Advanced Metering Infrastructure (AMI)-periodic measurements
Smart building automation system	$S_1R_0D_1$	Energy conservation, comforting lifestyle
Fault isolation and service restoration	$S_1R_0D_0$	Reporting distribution applications function
Online monitoring system	$S_0R_1D_1$	Monitoring and alarm system
Fault recorders	$S_0R_1D_0$	Power quality monitor/recorder
CCTV	$S_0R_0D_1$	Theft, trespassing reporting, accident reporting
Wi-Fi	$S_0R_0D_0$	Social network integration
Internet	$S_0R_0D_0$	Enterprise data

Let C be the set of priority classes each of which is attributed by quality parameters namely data size in binary, S_x^c, data rate in binary, R_y^c, and delay in binary, D_z^c, of x, y and z service parameters passing through any router. Here, data size is defined in terms of various packet sizes for different traffics transmitted in power distribution network given in *Bytes*, data rate is given in *bps* defined by the speed of packets transmitted from source to destination for each traffic, and delay is defined in terms of end-to-end packet delay of previous packet transmission over selected route given in *s*. Service parameters x, y and z are defined in terms of 0s and 1s for each traffic depending on their inputs as will be explained in details in this section. Number of flows belonging to each class are described in its corresponding flow set, $F_c(t)$. Thus, the traffic flow, $F_c\left(S_x^c, R_y^c, D_z^c\right)$ belonging to a particular traffic class C will use the respective time slots. These three parameters are chosen to monitor QoS performance of the selected four traffics.

Figure 3.2 to Figure 3.4 show each traffic being categorized according to their respective characteristics in terms of data size, data rate and delay. S_1 denotes small data size where SCADA and Smart Meter fall in this category meanwhile, S_0 indicates large data size where CCTV and Internet are classified in this category. R_1 symbolizes low data rate required for transmitting SCADA and Smart Meter traffic whereas, R_0 indicate high data rate for packet transmission of CCTV and Internet. D_1 represents small end-to-end delay where SCADA and CCTV fall in this category while D_0 denotes high end-to-end delay where Smart Meter and Internet fall in this category.

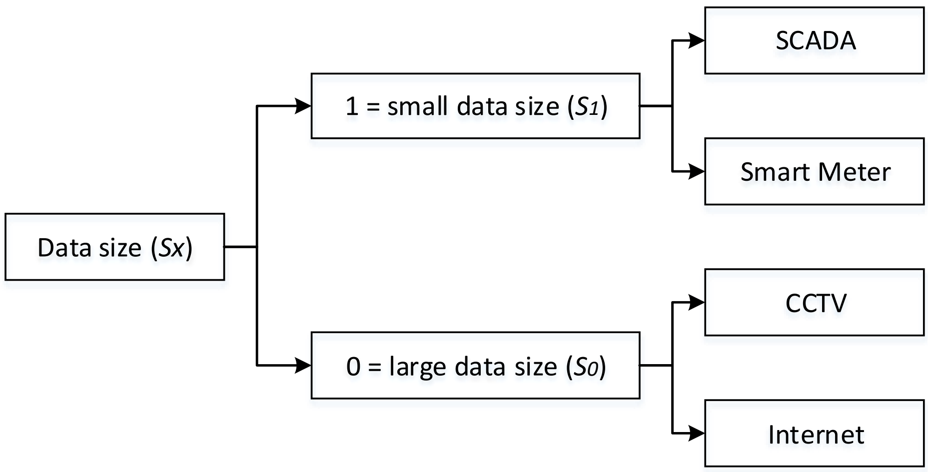

Figure 3.2 Traffic characterization based on data size, $\boldsymbol{S_x}$

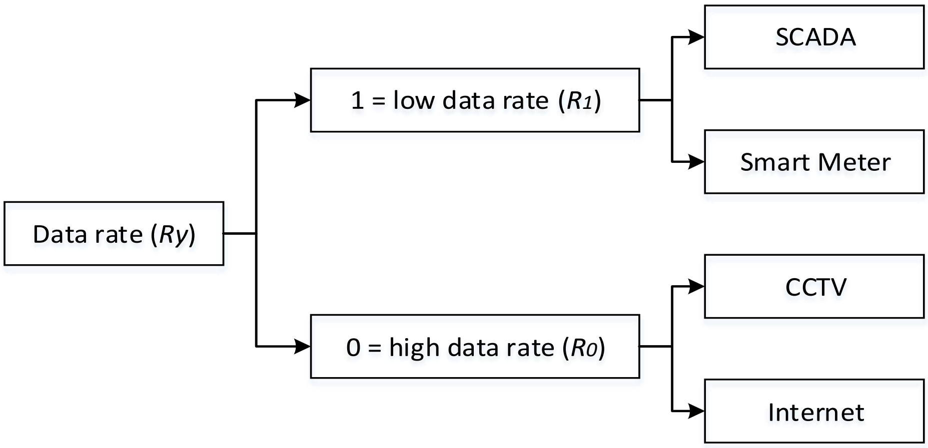

Figure 3.3 Traffic characterization based on data rate, $\boldsymbol{R_y}$

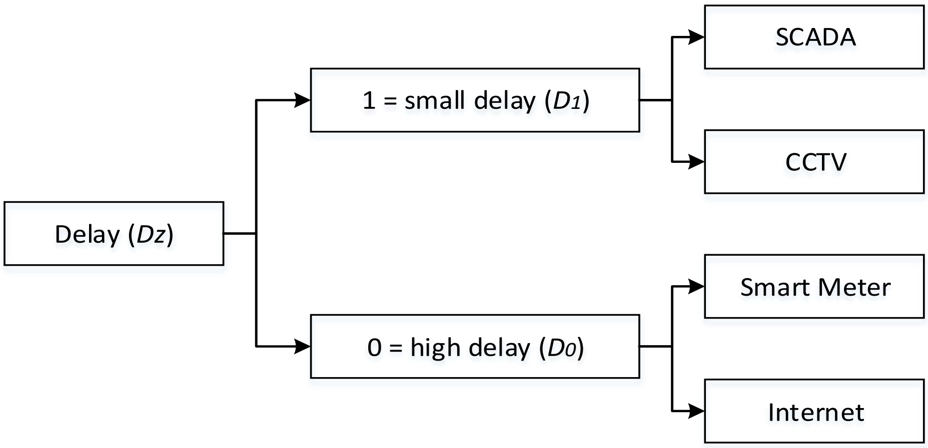

Figure 3.4 Traffic characterization based on end-to-end delay, $\boldsymbol{D_z}$

SCADA is characterized as traffic with small data size as its packets carrying only small size of collected information from the field. The packets are continuously being transmitted to control center that consist of real time data received from IEDs, relays, bay controllers, Remote Terminal Units (RTUs) or Power Line Carriers (PLCs) connected in the power utility system. Therefore, low data rate is required for transmitting the small amount of data size for this traffic. SCADA is also characterized as traffic with small delay since it cannot tolerate delay as it provides an active operator interface for supervisory control and remote configuration of IEDs and other devices. Larger delay causes huge production losses from flashover due to late commands or switching actions performed on devices in substation.

Smart Meter is considered as traffic with small data size since its packets carrying only meter readings of energy usage from customer premises and demand response which requires for transmission typically once a month for billing purposes. Thus, low data rate is necessary for packet transmission of this traffic to the control center. Smart Meter is also considered as high delay traffic since it consists of non-real time data which can handle high amount of delay for transmission to be done.

On the other hand, CCTV is reflected as traffic with large data size due to its packets carrying video information from each substations across the country and thus, requires high data rate for transmission to the control center. CCTV is also reflected as small delay traffic since it cannot tolerate delay as video disruption may interfere the monitoring of the substation. By default, CCTV is always on standby mode as it is only being triggered when the traffic is on demand.

Internet is categorized as traffic with large data size due to its packets carrying data information to serve internal communication inside control center and as external communication between other regional master stations. Hence, this traffic requires high data rate for packet transmission to ensure successful communication. Internet is also categorized as high delay traffic as it can tolerate high amount of delay in power grid as opposed to its importance in telecommunication network.

After the traffics are characterized, a pre-defined priority ranking for each traffic is tabulated in Table 3.3. The highest ranking belongs to SCADA and it is fixed for the entire algorithm as SCADA is responsible in delivering real time information which is compulsory for it to have the best characteristics in power distribution network. SCADA forms the virtual brain of power automation system which functions to collect information from substations and transfers the data over physical medium to National Load Dispatch Center (NLDC). The collected information will be processed and data will be displayed to be used for centralized control over the communication network to initiate external commands to remote devices.

Second ranking goes to Smart Meter, followed by CCTV and Internet is placed at the last ranking. The application-to-class mapping for traffics other than SCADA will vary at the runtime according to the variable context information which are data size, data rate and previous delay. Hence, an application is mapped to a particular class in accordance to its criticality as per contextual aware priority, ε_c^j.

Table 3.3 Pre-defined traffic priority, $\boldsymbol{\varepsilon_c^i}$ ranking

Traffics	**Characteristics**	**Symbols**	**Data Size (S_x)**	**Data Rate (R_y)**	**Delay (D_z)**	**Rank**
SCADA	• Small data size • Low data rate • Small delay	$S_1R_1D_1$	1	1	1	1
Smart Meter	• Small data size • Low data rate • High delay	$S_1R_1D_0$	1	1	0	2
CCTV	• Large data size • High data rate • Small delay	$S_0R_0D_1$	0	0	1	7
Internet	• Large data size • High data rate • High delay	$S_0R_0D_0$	0	0	0	8

Priority of class *C* in data delivery is defined in terms of its pre-defined priority which is ε_c^i as well as in terms of its contextual aware priority which is ε_c^j. The pre-defined priority factor, ε_c^i defines the initial priority belonging to class *C* for allocating resources. While parameter ε_c^j represents the contextual aware priority of class *C*, whose pre-defined priority is ε_c^i.

Each class is bounded by minimum and maximum threshold values of data size (S_{min}^c, S_{max}^c), data rate (R_{min}^c, R_{max}^c) and delay (D_{min}^c, D_{max}^c). Table 3.4 shows parameters of each traffic ranging between allowed minimum and maximum values. Ranges of data size for SCADA is obtained from Kuzlu et al. [4] whereas, delay ranges is acquired from IEEE Standard Communication [83]. Both ranges of data size and delay of Smart Meter and CCTV are attained from Kuzlu et al. respectively [4]. Internet comprises of standardized values based on Uribe et al. for both of its data size as well as delay ranges [84].

Table 3.4 Ranges of different parameters for each traffic

Traffics	Data Size (*Bytes*)	Data Rate (*bps*)	Delay (*s*)
SCADA	S_{min}^{SCADA}: 25	R_{min}^{SCADA}: 12	D_{min}^{SCADA}: 4
	S_{max}^{SCADA}: 1000	R_{max}^{SCADA}: 480	D_{max}^{SCADA}: 5
Smart Meter	$S_{min}^{Smart\ Meter}$: 0	$R_{min}^{Smart\ Meter}$: 0	$D_{min}^{Smart\ Meter}$: 0
	$S_{max}^{Smart\ Meter}$: 100	$R_{max}^{Smart\ Meter}$: 80	$D_{max}^{Smart\ Meter}$: 15
CCTV	S_{min}^{CCTV}: 100	R_{min}^{CCTV}: 73.6	D_{min}^{CCTV}: 0
	S_{max}^{CCTV}: 1000	R_{max}^{CCTV}: 736	D_{max}^{CCTV}: 5
Internet	$S_{min}^{Internet}$: 125	$R_{min}^{Internet}$: 70	$D_{min}^{Internet}$: 2
	$S_{max}^{Internet}$: 525	$R_{max}^{Internet}$: 294	$D_{max}^{Internet}$: 15

These four elected utility traffics obtained their data size and delay from the mentioned references. On the other hand, range of data rate is calculated using Equation 3.1.

$$R_y = \frac{S_x}{T_{transmission}} \qquad \text{(Equation 3.1)}$$

where data rate with y service parameter, R_y, is equal to data size with x service parameter, S_x, divided by transmission time, $T_{transmission}$. Service parameters of x and y are defined in terms of 0s and 1s for each traffic depending on their inputs.

The overall transmission time, $T_{transmission}$ is 0.165 seconds. Each traffic will be transmitted to destination according to their priority ranking by referring to Table 4.2 which comprises of $T_{transmission}$ for each traffic.

From these ranges, a set of threshold values can be formed to be used in this work as presented in Table 3.5. These threshold values are required to determine the parameters inequality comparison as in Table 3.8 in setting traffic priority ranking. If the input parameter is higher than the threshold, it is denoted with '>', and if the input parameter is lower than the threshold, it is denoted with '<'; in which both of them are indicated by bit 0 and 1 respectively.

Table 3.5 Threshold values of each traffic

Traffics	**Data Size, $S^{c}_{threshold}$ (*Bytes*)**	**Data Rate, $R^{c}_{threshold}$ (*bps*)**	**Delay, $D^{c}_{threshold}$ (*sec*)**
SCADA	41.67	20	4.5
Smart Meter	50	40	7.5
CCTV	203.80	150	2.5
Internet	446.43	250	8.5

These threshold values are selected differently for each parameters as shown in Table 3.6.

Table 3.6 Methods in selecting threshold values of each traffic

Traffics	**Data Size, $S^{c}_{threshold}$ (*Bytes*)**	**Data Rate, $R^{c}_{threshold}$ (*bps*)**	**Delay, $D^{c}_{threshold}$ (*sec*)**
SCADA	$R_y = \frac{S_x}{T_{transmission}}$	20	$D^c_z = \frac{D^c_{max} - D^c_{min}}{2}$
Smart Meter	$R_y = \frac{S_x}{T_{transmission}}$	40	$D^c_z = \frac{D^c_{max} - D^c_{min}}{2}$
CCTV	$R_y = \frac{S_x}{T_{transmission}}$	150	$D^c_z = \frac{D^c_{max} - D^c_{min}}{2}$
Internet	$R_y = \frac{S_x}{T_{transmission}}$	250	$D^c_z = \frac{D^c_{max} - D^c_{min}}{2}$

Threshold values of delay are chosen based on calculation in finding its midrange values. It is obtained from the mean of highest and lowest values as in Equation 3.2.

$$D^c_z = \frac{D^c_{max} - D^c_{min}}{2} \qquad \text{(Equation 3.2)}$$

where delay in binary of previous *C* traffic class with *z* service parameter, D^c_z, is equal to an average of difference between maximum threshold values of delay for *C* traffic class, D^c_{max} and minimum threshold values of delay for *C* traffic class, D^c_{min}.

Whereby, threshold values of data rate are elected in increasing trend from SCADA to Internet comprising of 20, 40, 150 and 250 *bps* respectively. These chosen values must obey their minimum and maximum requirement as presented in Table 3.4.

Table 3.7 shows description of each variables used in modelling the contextual aware traffic scheduling algorithm.

Table 3.7 Definition of variables used in the model

Symbols	Description
C	Set of priority classes
S_x^c	Data size in binary of *C* traffic class with *x* service parameter
R_y^c	Data rate in binary of *C* traffic class with *y* service parameter
D_z^c	Delay in binary of previous *C* traffic class with *z* service parameter
S_n^c	Data size in *Bytes* of *C* traffic class
R_n^c	Data rate in *bps* of *C* traffic class
D_n^c	Delay in *seconds* of previous *C* traffic class
$F_c(t)$	Number of flows belonging to each class
$F_c(S_x^c, R_y^c, D_z^c)$	Traffic flow of particular traffic class *C* for each service parameters
$T_{transmission}$	Transmission time
ε_c^i	Pre-defined priority
ε_c^j	Contextual aware priority
S_{min}^c	Minimum threshold values of data size for *C* traffic class
S_{max}^c	Maximum threshold values of data size for *C* traffic class
R_{min}^c	Minimum threshold values of data rate for *C* traffic class
R_{max}^c	Maximum threshold values of data rate for *C* traffic class
D_{min}^c	Minimum threshold values of delay for *C* traffic class
D_{max}^c	Maximum threshold values of delay for *C* traffic class
$S_{threshold}^c$	Threshold values of data size for *C* traffic class
$R_{threshold}^c$	Threshold values of data rate for *C* traffic class
$D_{threshold}^c$	Threshold values of delay for *C* traffic class

3.3 Contextual Aware Traffic Scheduling (CATSchA) Algorithm

In this section, an overview of the proposed CATSchA algorithm is presented. As the name suggests, the objective of CATSchA algorithm is to develop a contextual aware traffic scheduling algorithm for power distribution network in MPLS environment to enhance overall performance of smart grid system.

As illustrated in Figure 3.5, CATSchA algorithm is to be implemented in an MPLS router at ingress LER since queuing of traffics need to be modified according to their priorities each time before transmission takes place. The placement of the algorithm at ingress LER has also been done by Moghadam et al. and Demir et al. [57, 62].

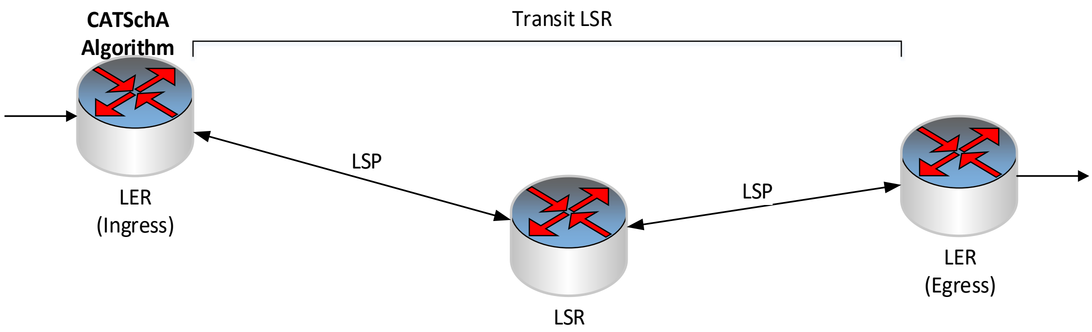

Figure 3.5 Architecture of MPLS network

LER operates at the edge of MPLS network by determining and applying the right labels to packets which then will be forwarded to LSR. Both ingress and egress LER routers are known as Provider Edge (PE) to MPLS domain. LSR is used to switch the labels of the packets and route it through the network. Transit LSR functions to switch and route the packets to the next LSR. LSP is a route through the MPLS network that has been set up according to the standards in Forwarding Equivalence Class (FEC). Across the transit LSR is bidirectional LSP throughout the network.

Characterization of traffics as explained in Section 3.2 is dedicated for pre-defined priorities. When there is new incoming packet, each of them will undergo the proposed algorithm as to decide the new priority and the new rank of each traffics. The flow of CATSchA algorithm is described in Figure 3.6.

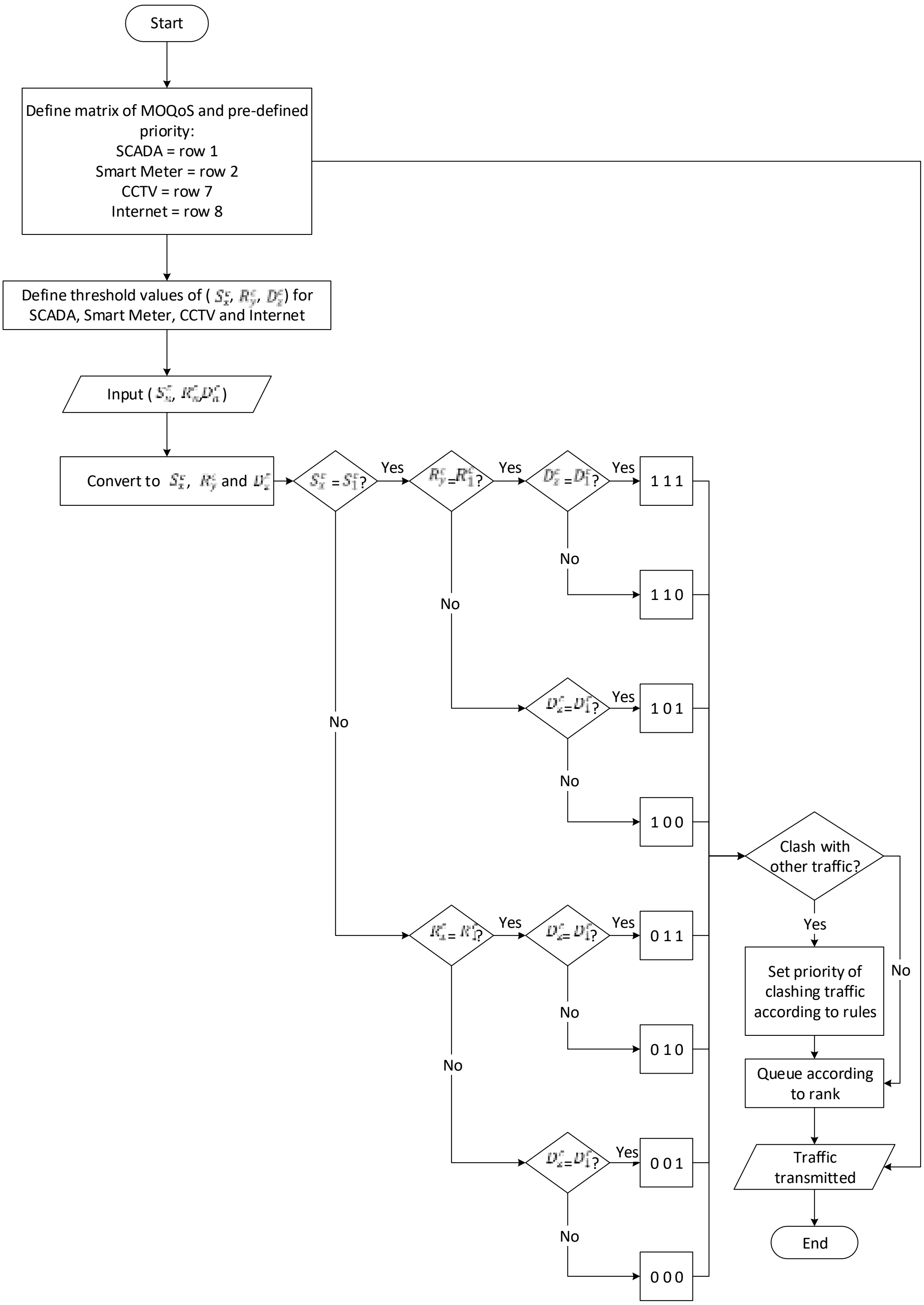

Figure 3.6 Detail flow chart of contextual aware traffic scheduling algorithm

MPLS transmission is based on reading the label in Shim header at each LSRs in network. There is a three bits area called traffic class field in the Shim header where it can carry DiffServ Code Point (DSCP) for handling various types of packet. Traffic class field or also known as EXP bits field compose of three bits and allow mapping up to 2^3 = eight possible combinations. All four traffics which are SCADA, Smart Meter, CCTV and Internet are evaluated based on their data size, data rate and delay.

These three parameters represent the three experimental (EXP) bits in MPLS Shim header as can be seen in Figure 3.7.

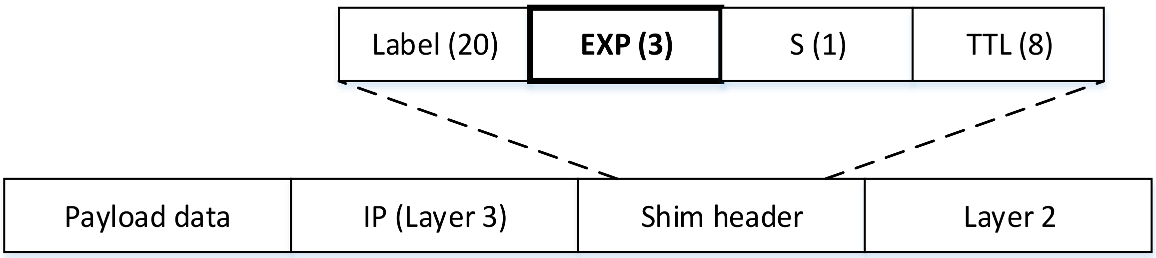

Figure 3.7 MPLS Shim header

Besides EXP bits, there are also Label with 20 bits, Stack (S) with 1 bit and Time to Live (TTL) with 8 bits which makes the total size of MPLS Shim header to be 32 bits or 4 bytes. MPLS Shim header is located between Layer 2 and Layer 3 or also known as Layer 2.5 for transmission to be carried out based on label reading.

The first step in developing CATSchA algorithm is to define a matrix named as Matrix of QoS (MOQoS) which is expressed with eight rows and five columns (8 x 5) as can be seen in Figure 3.8. This matrix comprises of three EXP bits, one ‘types of traffic’ bit and one ‘rank’ bit in each of its column. The three EXP bits which comprise of ‘data size’, ‘data rate’ and ‘delay’ are located in Columns 1, 2 and 3 respectively; ‘types of traffic’ is situated in Column 4; and ‘rank’ is placed in Column 5. This 8 x 5 matrix is considered for MOQoS in CATSchA algorithm in NS-3.

EXP bit1, Data Size | EXP bit 2, Data Rate | EXP bit 3, Delay | Types of Traffic | Rank

$$Matrix\ of\ QoS = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 2 \\ 1 & 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 0 & 4 \\ 0 & 1 & 1 & 0 & 5 \\ 0 & 1 & 0 & 0 & 6 \\ 0 & 0 & 1 & 3 & 7 \\ 0 & 0 & 0 & 4 & 8 \end{bmatrix}$$

Figure 3.8 Matrix of QoS (MOQoS)

The three EXP bits consist of binary bits of 0s and 1s. The 'types of traffic' column contains 0, 1, 2, 3 and 4 that represent SCADA as 1, Smart Meter as 2, CCTV as 3, Internet as 4 and other traffics as 0. The 'rank' column involves 1, 2, 3, 4, 5, 6, 7 and 8 which denote the highest priority ranking as 1 to the lowest priority ranking as 8.

Meanwhile, eight rows exist to display traffic position between rank 1 to rank 8. Subsequently, pre-defined priorities of each traffic, '$S_1R_1D_1$', '$S_1R_1D_0$', '$S_0R_0D_1$' and '$S_0R_0D_0$' can be introduced which consist of SCADA, Smart Meter, CCTV and Internet at Row 1, 2, 7 and 8 respectively.

After the MOQoS has been defined, threshold values of $S^C_{threshold}$, $R^C_{threshold}$ and $D^C_{threshold}$ which represent data size, data rate and delay respectively are defined for each traffic as has been explained in Section 3.2 These threshold values are defined as they are required for inequality comparison to produce output of the three EXP bits later.

Afterwards, the incoming traffic, S^C_n, R^C_n and D^C_n are analysed in terms of their current data size, data rate and previous delay. These input values are converted into binary form for inequality comparison purposes. Both threshold values and input parameters need to undergo inequality comparison to determine each traffic priority ranking.

The three input parameters are then compared with their threshold requirements in order to set priority ranking for Smart Meter, CCTV and Internet based on inequality comparison as in Table 3.8. SCADA is exceptional since its priority remain as the highest ranking traffic where its input parameters are not going through comparison process, although its threshold parameters have been defined. This is because, SCADA forms the virtual brain of power automation system as to receive data and information in real time from each substations for supervisory control and remote configuration. Thus, priority of SCADA is strictly immune to any possible situations. Whereby, priorities of other three traffics are ranked based on inequality comparison between their input parameters and threshold values.

The first decision making is done in terms of data size, S_x, followed by data rate, R_y, and lastly delay, D_z. Both data size and data rate consist of values from current cycle, i whereas, delay value is obtained from previous cycle, i-1. Previous delay is used to set priority of the traffic as current delay is not yet available due to transmission that has not yet happened. Therefore, delay of previous transmission is taken into account as to determine the priority ranking of the next packet transmission cycle. Data size is chosen as the first parameter for comparison purposes as it is the most influential factor in determining traffic transmission which is also affecting the delay.

Default priority and default ranking for each traffics have been summarized in Column 2 and Column 3 respectively of Table 3.8. Rank is denoted by three bits binary which rank from the highest priority (111, rank 1) to the lowest priority (000, rank 8). These three bits of packet header represent the input parameters i.e. data size, data rate and delay respectively, as has been bold in Table 3.8.

SCADA will always have the highest priority and ranked as 1 due to its crucial application and thus, its rank remain unchanged throughout the CATSchA algorithm. If the input is higher than the threshold, it is denoted with ‘>’, and if the input is lower than the threshold, it is denoted with ‘<’; in which both of them are indicated by bit 0 and 1 respectively.

Table 3.8 Summary of parameters inequality comparison

Traffic	**Default priority**	**Default rank**	**Parameters Inequality Comparison**								
			Data Size, *Sx*			**Data Rate, *Ry***			**Delay, *Dz***		
			Inequality	**Priority**	**Rank**	**Inequality**	**Priority**	**Rank**	**Inequality**	**Priority**	**Rank**
SCADA	111	1	Any	111	1	Any	111	1	Any	111	1
Smart Meter	110	2	>	**0**10	6	>	0**0**0	8	>=	00**0**	8
									<	00**1**	7
						<=	0**1**0	6	>=	01**0**	6
									<	01**1**	5
			<=	**1**10	2	>	1**0**0	4	>=	10**0**	4
									<	10**1**	3
						<=	1**1**0	2	>=	11**0**	2
									<	11**1**	1
CCTV	001	7	>=	**0**01	7	>=	0**0**1	7	>	00**0**	8
									<=	00**1**	7
						<	0**1**1	5	>	01**0**	6
									<=	01**1**	5
			<	**1**01	3	>=	1**0**1	3	>	10**0**	4
									<=	10**1**	3
						<	1**1**1	1	>	11**0**	2
									<=	11**1**	1
Internet	000	8	>=	**0**00	8	>=	0**0**0	8	>=	00**0**	8
									<	00**1**	7
						<	0**1**0	6	>=	01**0**	6
									<	01**1**	5
			<	**1**00	4	>=	1**0**0	4	>=	10**0**	4
									<	10**1**	3
						<	1**1**0	2	>=	11**0**	2
									<	11**1**	1

For example, CCTV by default is ranked at 7 and if the input of data size is greater than or equals to the threshold, then it is denoted as 0. If the input of data rate is less than the threshold, then it is denoted as 1. If the input of delay is less than or equals to the threshold, then it is denoted as 1, in which the final rank of CCTV is at 5. Therefore, the performance of CCTV can be improved in terms of delay and jitter by possessing newly defined ranking which is at 5 instead of its default ranking which is at 7.

Equation 3.3 refers to data size comparison of Smart Meter.

$$S_x = \begin{cases} 0 & ; S_n > S_{threshold} \\ 1 & ; S_n \leq S_{threshold} \end{cases} \qquad \text{(Equation 3.3)}$$

where data size in *Bytes*, S_n, is input data size. Data size in binary with *x* service parameter, S_x, is equal to 0 when input data size is greater than threshold data size and data size in binary with *x* service parameter, S_x, is equal to 1 when input data size is less than or equal to threshold data size.

On the other hand, for CCTV and Internet, Equation 3.4 is used.

$$S_x = \begin{cases} 0 & ; S_n \geq S_{threshold} \\ 1 & ; S_n < S_{threshold} \end{cases} \qquad \text{(Equation 3.4)}$$

where data size in binary with *x* service parameter, $S_x = 0$ when input data size is greater than or equal to threshold data size and data size in binary with *x* service parameter, $S_x = 1$, when input data size is less than threshold data size. CCTV and Internet differ from the calculation of Smart Meter because both of them have large data size whereas, Smart Meter has small data size.

Equation 3.5 refers to data rate comparison of Smart Meter.

$$R_y = \begin{cases} 0 & ; R_n > R_{threshold} \\ 1 & ; R_n \leq R_{threshold} \end{cases} \qquad \text{(Equation 3.5)}$$

where data rate in *bps*, R_n, is input data rate. Data rate in binary with *y* service parameter, R_y is equal to 0 when input data rate is greater than threshold data rate and data rate in binary with *y* service parameter, R_y, is equal to 1 when input data rate is less than or equal to threshold data rate.

On the other hand, for CCTV and Internet, Equation 3.6 is used.

$$R_y = \begin{cases} 0 & ; R_n \geq R_{threshold} \\ 1 & ; R_n < R_{threshold} \end{cases} \qquad \text{(Equation 3.6)}$$

where data rate in binary with *y* service parameter, $R_y = 0$ when input data rate is greater than or equal to threshold data rate and data rate in binary with *y* service parameter, $R_y = 1$ when input data rate is less than threshold data rate. CCTV and Internet differ from the calculation of Smart Meter because both of them have high data rate whereas, Smart Meter has low data rate.

Equation 3.7 refers to delay comparison of CCTV.

$$D_z = \begin{cases} 0 & ; D_n > D_{threshold} \\ 1 & ; D_n \leq D_{threshold} \end{cases} \qquad \text{(Equation 3.7)}$$

where delay in *seconds*, D_n, is input delay. Delay in binary with *z* service parameter, D_z, is equal to 0 when input delay is greater than threshold delay and delay in binary with *z* service parameter, D_z, is equal to 1 when input delay is less than or equal to threshold delay.

On the other hand, for Smart Meter and Internet, Equation 3.8 is used.

$$D_z = \begin{cases} 0 & ; D_n \geq D_{threshold} \\ 1 & ; D_n < D_{threshold} \end{cases} \qquad \text{(Equation 3.8)}$$

where delay in binary with *z* service parameter, $D_z = 0$ when input delay is greater than or equal to threshold delay and delay in binary with *z* service parameter, $D_z = 1$ when input delay is less than threshold delay. Smart Meter and Internet differ from the calculation of CCTV because both of them are able to tolerate high end-to-end delay whereas, CCTV requires small end-to-end delay.

After the traffics are ranked, CATSchA algorithm will check for clashing between them. There are certain conditions where SCADA, Smart Meter, CCTV or Internet are having the same priority with each other after inequality comparison process. In case of such events, the following rules are applied:

1. If priority of newly-defined traffics, $P_{i,j}^{new}$ other than priority of SCADA, $P_{1,1}^{SCADA}$ is ranked as 1 (111), $P_{i,j}^{new}$ is demoted to rank 2 (110) as rank 1 is reserved for SCADA only.

$$P_{i,j}^{new} = \begin{cases} P_{i,2}^{new} & ; P_{i,j}^{new} = P_{1,1}^{SCADA} \\ P_{i,j}^{new} & ; P_{i,j}^{new} \neq P_{1,1}^{SCADA} \end{cases} \qquad \text{(Equation 3.9)}$$

   where *i* represents types of traffic which is $i = 1, 2, 3, 4$; *j* denotes rank number which is $j = 1, 2, 3, 4, 5, 6, 7, 8$; and *P* indicates priority of a particular traffic. If newly-defined ranking of $P_{i,j}^{new}$ is not the same as $P_{1,1}^{SCADA}$ therefore, $P_{i,j}^{new}$ can maintain in its new rank directly.

   For example in the case where CCTV, $P_{3,1}^{new}$ is desiring for rank 1 which permanently belongs to SCADA, $P_{1,1}^{SCADA}$. According to Equation 3.9, CCTV is demoted to rank 2, $P_{3,2}^{new}$ as its newly-defined ranking. On the other hand, if CCTV desires for rank 8 which is other than rank 1 then, it can directly acquire $P_{3,8}^{new}$ as its newly-defined ranking.

2. However, if rank 2 (110) is also occupied with other traffic, the existing traffic, $P_{i,j}^{exist}$ is ranked according to $P_{i,j}^{new}$ pre-defined priority. Thus, $P_{i,j}^{new}$ can take possession of rank 2 as its newly-defined priority.

$$P_{i,j}^{new} = \begin{cases} P_{i,2}^{new} & ; P_{i,j}^{new} = P_{i,2}^{exist} \\ P_{i,j}^{new} & ; P_{i,j}^{new} \neq P_{i,2}^{exist} \end{cases} \qquad \text{(Equation 3.10)}$$

where $P_{i,2}^{exist}$ represents existing traffic available in rank 2 whereas, $P_{i,j}^{new}$ denotes demoted traffic from rank 1 which desires to occupy rank 2. $P_{i,2}^{new}$ indicates the newly-defined ranking of demoted traffic that is positioned in rank 2. If there is no existing traffic in rank 2 thus, $P_{i,j}^{new}$ can take over rank 2 directly. Otherwise, if $P_{i,j}^{new}$ is having ranking other than rank 2 then, it can stay in that position.

$$P_{i,j}^{exist} = \begin{cases} P_{i,pre-defined}^{new} & ; P_{i,2}^{new} = P_{i,2}^{exist} \\ P_{i,2}^{exist} & ; P_{i,2}^{new} \neq P_{i,2}^{exist} \end{cases} \quad \text{(Equation 3.11)}$$

whereby $P_{i,pre-defined}^{new}$ signifies the existing traffic in rank 2 that is taking over pre-defined ranking of $P_{i,2}^{new}$ as its new rank in the event of $P_{i,2}^{new}$ is conquering rank 2. Otherwise, $P_{i,2}^{exist}$ remain at its rank when there is no other traffic wishing to have rank 2 as their new priority rank.

For example when the demoted CCTV, $P_{3,2}^{new}$ is clashing with Smart Meter, $P_{2,2}^{exist}$ which is the existing traffic on rank 2. According to Equation 3.10, CCTV has the right to stay in rank 2, $P_{3,2}^{new}$ as its newly-defined ranking meanwhile, Smart Meter is taking over pre-defined ranking of CCTV, $P_{2,7}^{new}$ as its newly-defined ranking according to Equation 3.11. On the other hand, if Smart Meter is positioned in rank 8, $P_{2,8}^{exist}$ then, CCTV can take over rank 2 directly as $P_{3,2}^{new}$.

3. In general, if two traffics encountered the same rank between rank 2 until rank 8 without desiring for rank 1 (SCADA), the higher rank of pre-defined traffic, $P_{i,j}^{new\,(1)}$ is going to take possession of the newly-defined ranking. Whereas, the lower rank of pre-defined traffic, $P_{i,j}^{new\,(2)}$ is to give up its desired new rank by taking over the pre-defined rank of $P_{i,j}^{new\,(1)}$ as its newly-defined ranking.

$$P_{i,j}^{new} = \begin{cases} P_{i,j}^{new(1)} & ; P_{i,j}^{new} = P_{i,j}^{new\,(1)} \\ P_{i,pre-defined}^{new\,(1)} & ; P_{i,j}^{new} = P_{i,j}^{new\,(2)} \end{cases} \quad \text{(Equation 3.12)}$$

where $P_{i,j}^{new}$ represents the two input traffics that are clashing, $P_{i,j}^{new\,(1)}$ denotes the higher rank traffic and $P_{i,j}^{new\,(2)}$ signifies the lower rank traffic. $P_{i,pre-defined}^{new\,(1)}$ indicates the pre-defined ranking of higher rank traffic, $P_{i,j}^{new\,(1)}$ when the input traffic belongs to lower rank traffic.

For example when both Smart Meter, $P_{2,7}^{new\,(1)}$ and Internet, $P_{4,7}^{new\,(2)}$ are desiring for rank 7. According to Equation 3.12, Smart Meter has the right to stay in rank 7, $P_{2,7}^{new\,(1)}$ as its newly-defined ranking meanwhile, Internet is taking over pre-defined ranking of Smart Meter, $P_{4,2}^{new\,(2)}$ as its newly-defined ranking.

4. If three traffics are having the same rank between rank 2 until rank 8 in which one of the traffic is the existing traffic in the rank, the higher rank of pre-defined traffic, $P_{i,j}^{new\,(1)}$ is going to take possession of the newly-defined ranking. The lower rank of pre-defined traffic, $P_{i,j}^{new\,(2)}$ is to give up its desired new rank by taking over the pre-defined rank of $P_{i,j}^{new\,(1)}$ as its newly-defined ranking. Whereas, the existing traffic, $P_{i,j}^{exist}$ is going to take over the pre-defined rank of $P_{i,j}^{new\,(2)}$ as its newly-defined ranking.

$$P_{i,j}^{new} = \begin{cases} P_{i,j}^{new(1)} & ; P_{i,j}^{new} = P_{i,j}^{new\,(1)} \\ P_{i,pre-defined}^{new\,(1)} & ; P_{i,j}^{new} = P_{i,j}^{new\,(2)} \end{cases} \quad \text{(Equation 3.13)}$$

where $P_{i,j}^{new}$ represents the two input traffics that are clashing other than existing traffic, $P_{i,j}^{new\,(1)}$ denotes the higher rank traffic and $P_{i,j}^{new\,(2)}$ signifies the lower rank traffic. $P_{i,pre-defined}^{new\,(1)}$ indicates the pre-defined ranking of higher rank traffic, $P_{i,j}^{new\,(1)}$ when the input traffic belongs to lower rank traffic.

$$P_{i,j}^{exist} = \begin{cases} P_{i,pre-defined}^{new\,(1)} & ; P_{i,j}^{exist} = P_{i,j}^{new\,(1)} \\ P_{i,pre-defined}^{new\,(2)} & ; P_{i,j}^{exist} = P_{i,j}^{new\,(2)} \end{cases} \quad \text{(Equation 3.14)}$$

where $P_{i,j}^{exist}$ represents the existing traffic that occupied a certain rank which desired by the other two traffics. $P_{i,j}^{new\,(1)}$ denotes the higher rank traffic and $P_{i,j}^{new\,(2)}$ signifies the lower rank traffic. $P_{i,pre-defined}^{new\,(1)}$ specifies the pre-defined ranking of higher rank traffic, $P_{i,j}^{new\,(1)}$ as newly-defined ranking for the existing traffic. Meanwhile, $P_{i,pre-defined}^{new\,(2)}$ indicates the pre-defined ranking of lower rank traffic, $P_{i,j}^{new\,(2)}$ as newly-defined ranking for the existing traffic.

For example when Smart Meter, $P_{2,7}^{new\,(1)}$, Internet, $P_{4,7}^{new\,(2)}$ and CCTV $P_{3,7}^{exist}$ are desiring for rank 7. According to Equation 3.13, Smart Meter has the right to stay in rank 7, $P_{2,7}^{new\,(1)}$ as its newly-defined ranking meanwhile Internet is taking over pre-defined ranking of Smart Meter, $P_{4,2}^{new\,(2)}$, rank 2 as its newly-defined ranking. According to Equation 3.14, CCTV is taking over pre-defined ranking of Internet, $P_{3,8}^{exist}$, rank 8 as its newly-defined ranking.

After clashes between the traffics have been resolved, a new priority is set for each traffics that will decide their rank in queuing of packets before the transmission process begin. If there is no changes in characteristics of each traffics then, both of their priorities and ranks will be the same as pre-defined. Meanwhile, if there are changes in characteristics of each traffics then, both of their priorities and ranks will be set to a new one.

Then, these traffics will go through the next step which is to queue the packets according to their newly-defined priorities and rank. These newly-defined priorities and ranks are reflected in the matrix of MOQoS as the matrix need to update its content for each traffic before the traffics are transmitted accordingly.

Lastly, traffics are transmitted according to their priorities and ranks. Basically, input values are the main factors in determining rank of traffic correspond to their priority which consequently in charge of controlling transmission time of each traffic.

Figure 3.9 illustrates general flow chart in setting the newly-defined priority ranking of each traffic in CATSchA algorithm. Each layers in decision making act as determining factors in generating the three EXP bits and consequently setting new priority ranking that comply with the stated rules after the inequality comparisons of parameters for each traffic have been done.

The first level of priority setting is to determine and establish traffic ranking after input parameters are compared with their threshold requirements respectively. After the traffics are ranked, CATSchA algorithm will check for clashing between the traffics. There are certain conditions where SCADA, Smart Meter, CCTV or Internet have the same priority with each other. In case of such events, Equation 3.9 until Equation 3.14 are applied. This condition is represented as second level of priority setting in deciding and recognizing the final traffic ranking.

Figure 3.10 shows the general pseudocode on context aware priority ranking of CATSchA algorithm.

3.4 Simulation

The architecture of power distribution network is complex as it deploys an advanced communication technology for two-way communication networks to be established. The analysis of distribution network behavior mostly are based on simulation techniques [85-89] since it is not recommended to directly interfere with utility system as it can cause a national electricity catastrophe.

3.4.1 Simulation Tool

The simulation of CATSchA algorithm for power distribution network is done by using Network Simulator 3 (NS-3) version 3.18 for its performance study. The significance of using NS-3 as a simulation platform is that it is a discrete-event network simulator which is suitable for research and development purposes [90]. NS-3 is the most appropriate software to be used in this work as it has been widely used in developing algorithm with MPLS protocol for smart grid network [1, 60, 61]. Intel Core i5-2500 processor and 8 *Gigabyte* (*GB*) RAM are used to run the simulation.

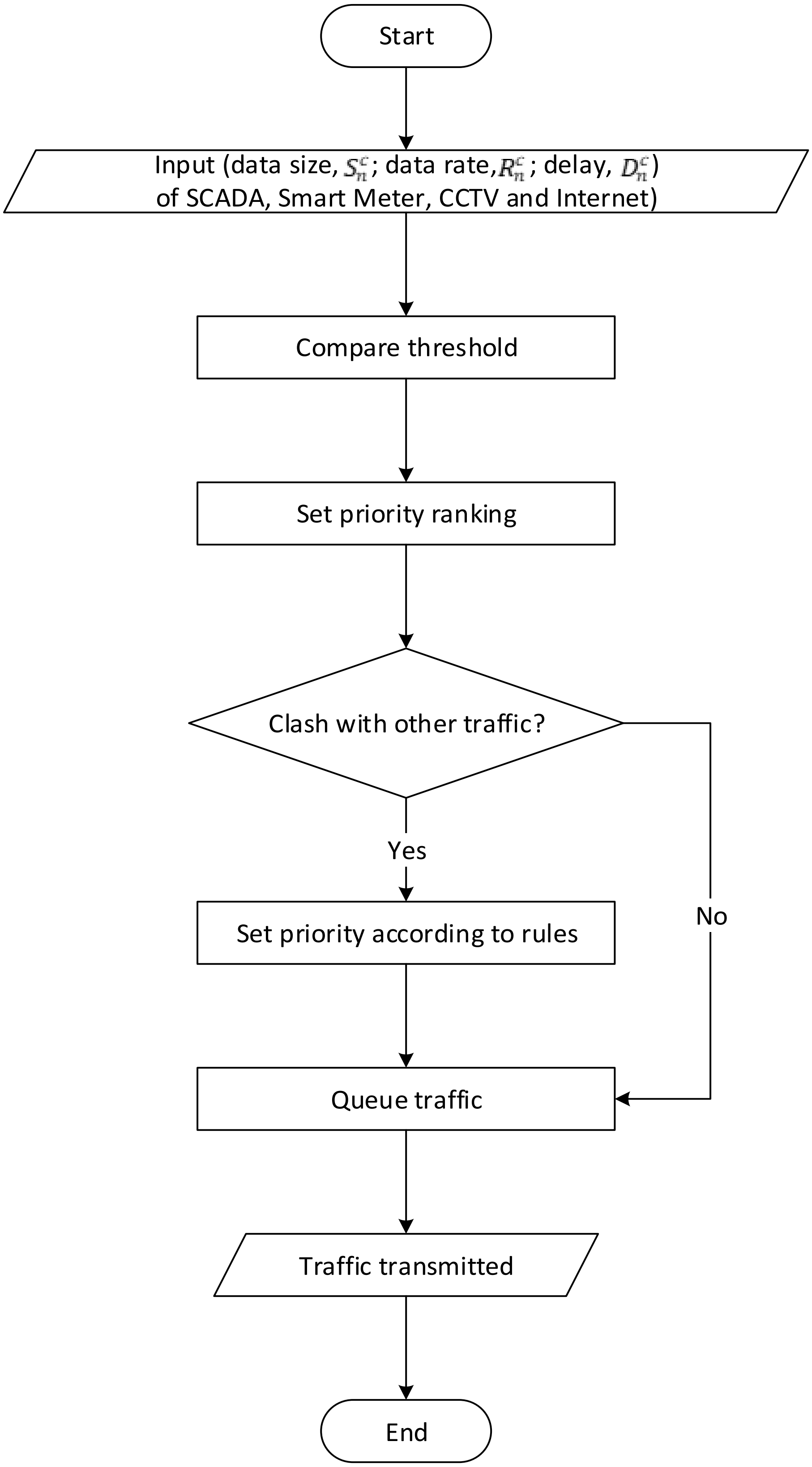

Figure 3.9 General flow chart of contextual aware traffic scheduling algorithm

Pseudocode 1: Context aware priority ranking

Initialize: MOQoSA[8][5]; // for 8 rows, 5 columns (1: data size in binary; 2: data rate in binary; 3: delay in binary; 4: types of traffic, 5: traffic rank)

Define: threshold values for all traffics and parameters (data size, data rate, delay)

Assign pre-defined rank:

ε_1^1=MOQoSA[0][4]; // SCADA

ε_2^2=MOQoSA[1][4]; // Smart Meter

ε_3^7=MOQoSA[6][4]; // CCTV

ε_4^8=MOQoSA[7][4]; // Internet

Initial input:

HeaderTraffic, *C*; // 1 for SCADA, 2 for Smart Meter, 3 for CCTV, 4 for Internet

Data Size for *C*, S_x^c;

Data Rate for *C*, R_y^c;

Delay for *C*, D_z^c;

if (*C*==2)

    check inequality for S_n^2;

        change ranking (S_x^2) based on threshold value;

    check inequality for R_n^2;

        change ranking (R_x^2) based on threshold value;

    check inequality for D_n^2;

        change ranking (D_x^2) based on threshold value;

end if

if (*C*==3)

    check inequality for S_n^3;

        change ranking (S_x^3) based on threshold value;

    check inequality for R_n^3;

        change ranking (R_x^3) based on threshold value;

    check inequality for D_n^3;

        change ranking (D_x^3) based on threshold value;

end if

if (*C*==4)

check inequality for S_n^4;
change ranking (S_x^4) based on threshold value;
check inequality for R_n^4;
change ranking (R_x^4) based on threshold value;
check inequality for D_n^4;
change ranking (D_x^4) based on threshold value;
end if
if ε_c^j == any ε_c^i (ε_1^1 or ε_2^2 or ε_3^7 or ε_4^8)
swap ε_c^j with ε_c^i
end if
Initialize start time for all traffic: $T_{transmission}$
Compare rank ε_c^j for all traffic
if ε_c^j increase (priority increase)
decrease $T_{transmission}$;
else if ε_c^j decrease (priority decrease)
increase $T_{transmission}$;
end if

Figure 3.10 Pseudocode on context aware priority ranking

Further reasons in choosing NS-3 in simulating the proposed algorithm are due to easily written and modification of the source code. This simulator utilizes C++ language which makes the source code simpler although it is only compatible with Linux and run on Ubuntu 14.04 operating system. Besides, performance parameters of traffic scheduling that have been programmed in NS-3 such as delay, jitter and throughput can be captured from the output results easily. NS-3 also supports real-time scheduler for interaction with real network devices.

C++ language is one of the most powerful, efficient and versatile language as it allows user to control resources available in computer so that designated creation will have the right speed when running on computer. This language is also often used by other researchers in developing their algorithm as it is scalable, fast and controllable language to code functions [91-93].

3.4.2 Network Environment

In this work, the proposed CATSchA algorithm for power distribution network is evaluated via two different scenarios. The first scenario is on evaluating performance of the CATSchA algorithm via proof of concept. Whereas, the second scenario is to compare CATSchA algorithm with the algorithm without contextual aware as to prove its advantages. Both scenarios are carried out using six different cases chosen as shown in Table 3.9. These six distinctive cases are selected as each of them comprise of different output traffics arrangements that represent different priority ranking of traffics.

3.4.3 Network Topology

Network topology is simulated using NS-3 with IP-MPLS environment as well as IPv4 for implementation of CATSchA algorithm.

There are eight number of nodes created as shown in Figure 3.11, in which three of them are configured as routers namely R1, R2 and R3; four nodes serve as hosts for source consisting of PC1, PC2, PC3 and PC4; while the remaining node functions as host for destination namely PC5. Such topology is created to mimic the real topology used in utility network but with simple and less complexity of network design for proof of concept.

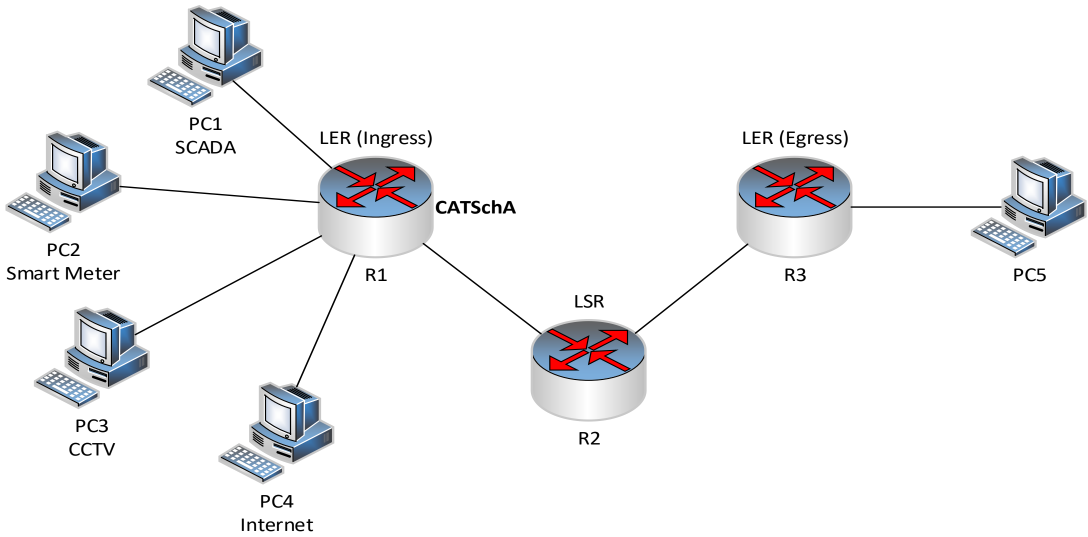

Figure 3.11 Network topology of CATSchA algorithm

Table 3.9 Six different cases with its inequality comparison

Cases	Parameters	Input Traffic				Output Traffic			
		1	2	3	4	1	2	3	4
		SCADA	**Smart Meter**	**CCTV**	**Internet**				
Case 1	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 1	0 0 0	SCADA	Smart Meter	CCTV	Internet
Case 2	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 0	0 0 1	SCADA	Smart Meter	Internet	CCTV
Case 3	Data Size Data Rate Delay	Any Any Any	0 0 1	1 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet
Case 4	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 0	0 0 1	SCADA	CCTV	Internet	Smart Meter
Case 5	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 0	1 1 0	SCADA	Internet	Smart Meter	CCTV
Case 6	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 1	1 1 0	SCADA	Internet	CCTV	Smart Meter

The default parameter values used in CATSchA algorithm is as shown in Table 3.10. The link capacity of the network is 100 *Mbps* to emulate the real amount of data transmission in utility environment. Whereas, the cycle time is 2 *ms* to ensure size of each packet is fixed to 1024 *Bytes* throughout the simulation to accurately evaluate the performance parameters. The maximum number of packets is kept to 50 in queue for transmission to cater high number of queued packets in each router.

Table 3.10 Default parameters used in CATSchA algorithm

Default Parameters	Values
Link capacity	100 *Mbps*
Cycle time	2 *ms*
Packet size	1024 *Bytes*
Maximum packets in transmission queue	50

There are four incoming traffics i.e. SCADA, Smart Meter, CCTV and Internet represented by PC1, PC2, PC3 and PC4 respectively, pumped into one router, R1. Packets of each traffic are transmitted from R1 into destination host, PC5 over intermediary nodes which is R2 and R3.

3.5 System Parameters

System parameters consist of design and performance parameters. Design parameters are variables required in designing the proposed algorithm so that performance parameters of the algorithm can be achieved.

3.5.1 Design Parameters

Parameters involved in designing CATSchA algorithm are types of traffic, data size, data rate, amount of offered load, number of packets transmitted and delay of previous transmission. Four types of traffic involved namely SCADA, Smart Meter, CCTV and Internet, with different data size, data rate and previous delay. Amount of offered load for each traffic varies from 0 to 100 *Mbps* with 10 *Mbps* step to analyse the performance of CATSchA algorithm with different values. Data size and number of packets transmitted are diverse depending on data rate as well as transmission time of each traffic. Delay of previous transmission for each traffic are taken into account as to determine the priority ranking of next packet transmission of certain traffic. These selected parameters are chosen as they have high impact on the performance of CATSchA algorithm.

Data size is selected as each traffics carry different amount of collected information from the field according to their priorities. Data rate is chosen as each traffics require different amount of data rate for transmitting different data size to the control center. Delay is elected as each traffics especially in smart grid environment are relying on time taken for packet arrival at control center.

3.5.2 Performance Parameters

There are three different parameters responsible in stipulating the performance of proposed algorithm i.e. delay, jitter and throughput. Delay is selected as performance of traffics in smart grid defined as time taken for a packet to traverse from source to destination. Performance of delay is identified by design parameters consisting of number of packets, types of traffic, size of data and also amount of the offered load.

Jitter is a timing variations which can cause packet arrival at an inconsistent rate. Performance of jitter is described by types of traffic, previous transmission delay, number of packets transmitted and also offered load by designated network.

Throughput is defined as the amount of data transferred between source and destination. Throughput performance is affected by size of data, amount of offered load, number of packets transmitted as well as types of traffic involved. These three chosen parameters answer the third research question in evaluating the performance of CATSchA algorithm as in Section 1.4.

3.6 Summary

In this chapter, a contextual aware based traffic scheduling algorithm named as CATSchA algorithm has been proposed. The invented CATSchA algorithm is designed to handle smart grid traffics with various characteristics and priorities. These traffics are characterized into several classes based on their data size, data rate and previous delay. Four traffics in power distribution network have been chosen and each of them possess different characteristics which created a pre-defined traffic priority ranking. Threshold values of data size, data rate and previous delay are presented to characterize the incoming traffic for contextual aware priority ranking.

Development of CATSchA algorithm for power distribution network is described in more details aided with an architecture of MPLS network, flowcharts, equations and table of summary for parameters inequality comparison. Table of summary for parameters inequality comparison is used to decide the priority of each traffic depending on their threshold requirements since they are diverse in terms of data size, data rate and delay. Several rules are applied in the case when traffics are having the same priority with each other after the inequality comparison process. From the priority decision that has been made, transmission time is controlled by CATSchA algorithm by firstly releasing the highest priority ranking traffic and followed by least rank traffics accordingly. Thus, desired performance parameters are achieved with the employed design parameters.

Simulation method is selected in testing the CATSchA algorithm. The algorithm is simulated with MPLS protocol using NS-3 as a simulation platform which utilizes C++ language for the development, validation and testing purposes. Six different cases are carried out with two different scenarios and will be explained in Chapter 4.

System parameters are required in designing and developing the CATSchA algorithm and it is categorized into design and performance parameters. Design parameters involved are types of traffic, data size, data rate, offered load, number of packets transmitted and delay of previous transmission. Performance parameters involved are delay, jitter and throughput.

CHAPTER 4

SIMULATION RESULT AND PERFORMANCE EVALUATION

4.1 Introduction

This chapter discusses on simulation results and performance evaluation of Contextual Aware Traffic Scheduling (CATSchA) algorithm. Four traffics will be considered for the simulation; namely SCADA, Smart Meter, CCTV and Internet. Firstly, the performance of CATSchA algorithm is validated in Section 4.2 between simulation results obtained from NS-3 with simulation results attained from MATLAB. This is to validate the correctness of the program and the contextual aware ranking. Next, the performance of CATSchA algorithm is studied for six different cases and presented in Section 4.3 in terms of delay and efficiency for proof of concept. Section 4.4 is dedicated to show the advantages of CATSchA algorithm whereby traffic scheduling with contextual aware is compared with traffic scheduling without contextual aware. The performance of both algorithms are evaluated in terms of delay, jitter, throughput and efficiency for six distinct cases using actual traffic parameters. Chapter 4 is summarized in Section 4.5.

4.2 Algorithm Validation between NS-3 and MATLAB

This chapter discusses on validation of CATSchA algorithm to assure that the coding is designed and functioning correctly before its performance is studied and evaluated. CATSchA algorithm simulated in NS-3 is compared with simulation via MATLAB in terms of traffic ranking as shown in Table 4.1.

Each traffic is experiencing eight different sets of data size, data rate and delay i.e. 000, 001, 010, 011, 100, 101, 110 and 111. As there are three different varying traffics (Smart Meter, Internet and CCTV), $8^3 = 512$, 512 possible combinations exist for the algorithm. Each combination comprises of different sets of data size, data rate and delay as factors in deciding the output ranking of each traffics. However, for proof of concept, only 302 combinations of input traffics are tested.

Table 4.1 Validation results between NS-3 and MATLAB

		Input Traffic				**Output Traffic**							
						NS-3				**MATLAB**			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
1	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	0 0 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
2	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 0	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
3	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 1	0 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
4	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 1	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
5	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
6	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	0 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

This is because, out of 512 combinations, only 302 combinations are realistic for power distribution network since each traffics are bounded by their respective minimum and maximum requirements in terms of data size, data rate and delay. The remaining combinations are having either extremely small or extremely big values which is impractical in real utility network. The validation for the rest of the combinations are shown in Appendix A.

The first decision making is done in terms of data size, followed by data rate and lastly delay in evaluating input traffics. Both data size and data rate comprise of values from current cycle whereby, delay value is acquired from previous cycle as transmission is not happening yet in obtaining current delay value. Data size is the most significant parameter in determining traffic transmission which explained why it is chosen as the first decision maker in CATSchA algorithm.

There are four levels in placement of the output traffic in this thesis as there are four traffics involved. Ranking of output traffic is explained as follows; 1 represents the highest rank traffic, 2 denotes the second highest rank traffic, 3 indicates the second lowest rank traffic and 4 signifies the lowest rank traffic. Placement of output traffics will be varied depending on the input data size, data rate and delay. Input SCADA is not varied as it remains as the highest rank traffic throughout the validation process. Pre-defined priority ranking of traffics are given as follows; SCADA is at rank 1 (111), Smart Meter is at rank 2 (110), CCTV is at rank 7 (001) and Internet is at rank 8 (000).

The overall transmission time chosen to run in simulation platform is 0.165 seconds. SCADA is transmitted first from 0 to 0.06 seconds, followed by second traffic transmission which starts from 0.065 seconds to 0.165 seconds. Third traffic transmission starts from 0.073 seconds to 0.165 seconds and the last traffic is transmitted from 0.095 seconds to 0.165 seconds as shown in Table 4.2. For second, third and fourth traffics are going to be either Smart Meter, CCTV or Internet based on their contextual aware ranking.

Table 4.2 Transmision time for traffics with CATSchA algorithm

Traffics	Start transmission time (seconds)	End transmission time (seconds)
SCADA	0	0.06
2nd traffic	0.065	0.165
3rd traffic	0.073	0.165
4th traffic	0.095	0.165

The transmission time used in NS-3 are deemed to be sufficient as they are able to handle the traffic transmission as in actual utility network.

As can be seen from Table 4.1, it shows the ranking of output traffics between NS-3 and MATLAB are similar. This proves that the program for CATSchA algorithm is correct and has been validated. The rest of Table 4.1 can be found in Appendix A.

4.3 Performance Study of Contextual Aware Traffic Scheduling (CATSchA) Algorithm

The proposed CATSchA algorithm is studied for six different cases in terms of delay and efficiency for proof of concept. The study is done using six different cases namely Case 1 (priority from high to low: SCADA → Smart Meter → CCTV → Internet), Case 2 (priority from high to low: SCADA → Smart Meter → Internet → CCTV), Case 3 (priority from high to low: SCADA → CCTV → Smart Meter → Internet), Case 4 (priority from high to low: SCADA → CCTV → Internet → Smart Meter), Case 5 (priority from high to low: SCADA → Internet → Smart Meter → CCTV) and Case 6 (priority from high to low: SCADA → Internet → CCTV → Smart Meter).

4.3.1 Delay

Delay is defined as time taken for a packet to traverse from source to destination. It is an important parameter in evaluating the performance of CATSchA algorithm as SCADA, Smart Meter, CCTV and Internet require minimum delay. Therefore, the lower the delay, the better the performance of CATSchA algorithm. The calculation of delay can be referred to Appendix B which is the main source code for CATSchA algorithm as it shows the command line in getting the delay by using NS-3 [90].

Offered load is varied from 0 to 100 *Mbps* with 10 *Mbps* step to observe their delay variations. Full offered load of 100 *Mbps* is used because to emulate the real amount of data transmission in utility environment [6]. For proof of concept, similar data size and data rate is used for all traffics. The delay results for Case 1 is shown in Figure 4.1, where four different traffics are taken into consideration during the simulation process.

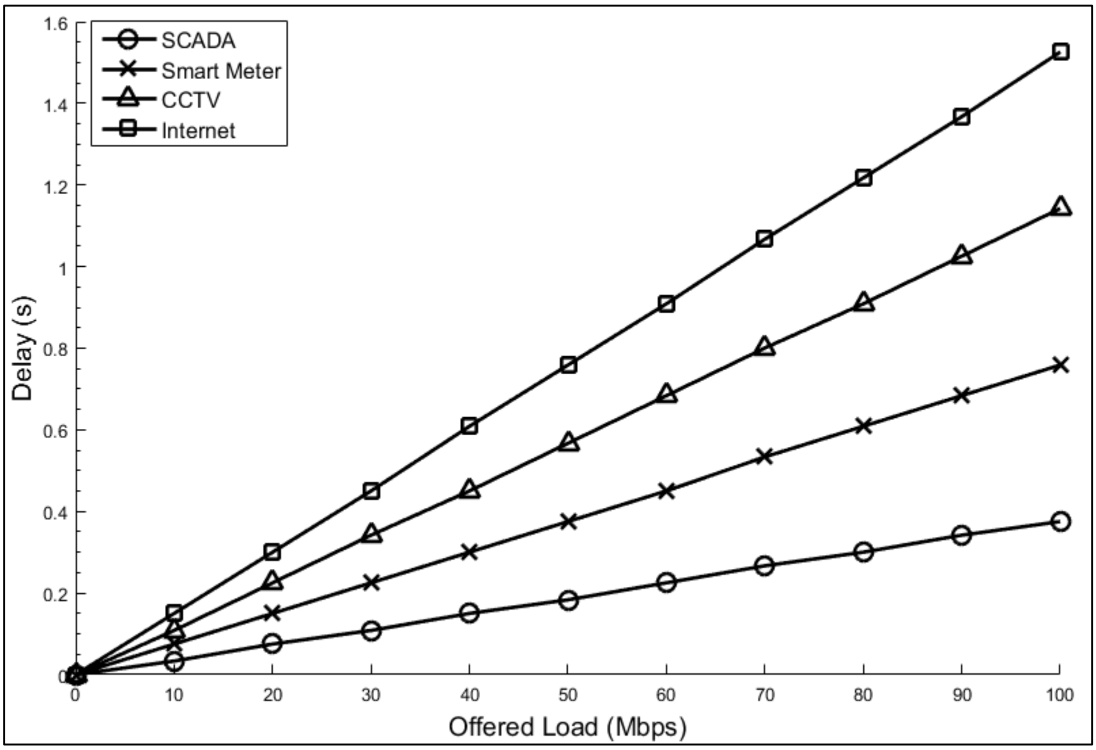

Figure 4.1 Delay versus offered load for CATSchA in Case 1

It can be seen that the trend of four different traffics observed is such that SCADA has the shortest delay, followed by Smart Meter, CCTV and finally Internet as has been expected. All traffics increase linearly until all of them reach a delay of 0.38 *s*, 0.76 *s*, 1.14 *s* and 1.53 *s* at 100 *Mbps* offered load respectively.

SCADA has the shortest delay because in CATSchA algorithm, SCADA will always have the highest rank and will be transmitted first. Since Case 1 is the pre-defined priority, Smart Meter is rank as 2, CCTV is at rank 7 and Internet is at rank 8 as expected. Hence, it can be proven as in the graph that the delay of each traffic are correctly matched with their respective ranking.

In this case, SCADA has the shortest delay as it always have the highest rank without taking into account of its inequality parameters which involved data size, data rate and previous transmission delay. Smart Meter is ranked as 2 as both of its data size and data rate are smaller than threshold (bit 1) whereby its previous transmission delay is larger than threshold (bit 0) and thus, its header is formed as (110). CCTV is ranked as 7 as both of its data size and data rate are larger than threshold (bit 0) whereby its previous transmission delay is smaller than threshold (bit 1) and thus, its header is formed as (001). Internet is ranked as 8 as all three of its data size, data rate and previous transmission delay are larger than threshold (bit 0) and thus, its header is formed as (000).

Figure 4.2 shows the delay result of Case 2 with CATSchA algorithm. These scenarios illustrate different trend in traffic priority ranking where the shortest delay belongs to SCADA, followed by Smart Meter, Internet and CCTV. Delays for all traffics increase linearly as the offered load increases from 0 to 100 *Mbps* where each of them reach a delay of 0.34 *s*, 0.68 *s*, 1.03 *s* and 1.37 *s* at 90 *Mbps* offered load respectively.

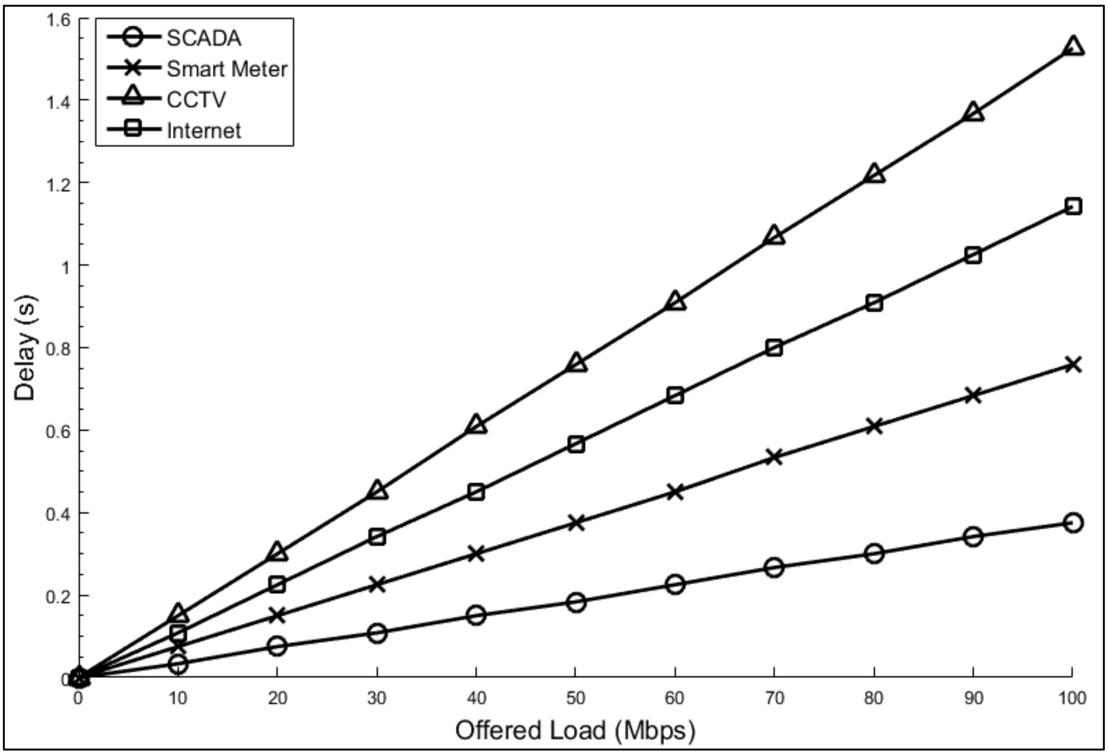

Figure 4.2 Delay versus offered load for CATSchA in Case 2

In this case, SCADA has the shortest delay because in CATSchA algorithm, SCADA will always have the highest rank and will be transmitted first without taking into account of its inequality parameters which involve data size, data rate and previous transmission delay. Smart Meter is ranked as 2 as both of its data size and data rate are smaller than threshold (bit 1) whereby its previous transmission delay is larger than threshold (bit 0) and thus, its header is formed as (110). Internet is ranked as 7 as both of its data size and data rate are larger than threshold (bit 0) whereby its previous transmission delay is smaller than threshold (bit 1) and thus, its header is formed as (001). CCTV is ranked as 8 as all three of its data size, data rate and previous transmission delay are larger than threshold (bit 0) and thus, its header is formed as (000).

For Case 3 of delay result in Figure 4.3, it shows that SCADA, CCTV, Smart Meter and Internet are arranged from the shortest to the longest delay with CATSchA algorithm. The delays of all traffics with CATSchA algorithm increase steadily where each of them reach a delay of 0.3 *s*, 0.61 *s*, 0.91 *s* and 1.22 *s* at 80 *Mbps* offered load respectively.

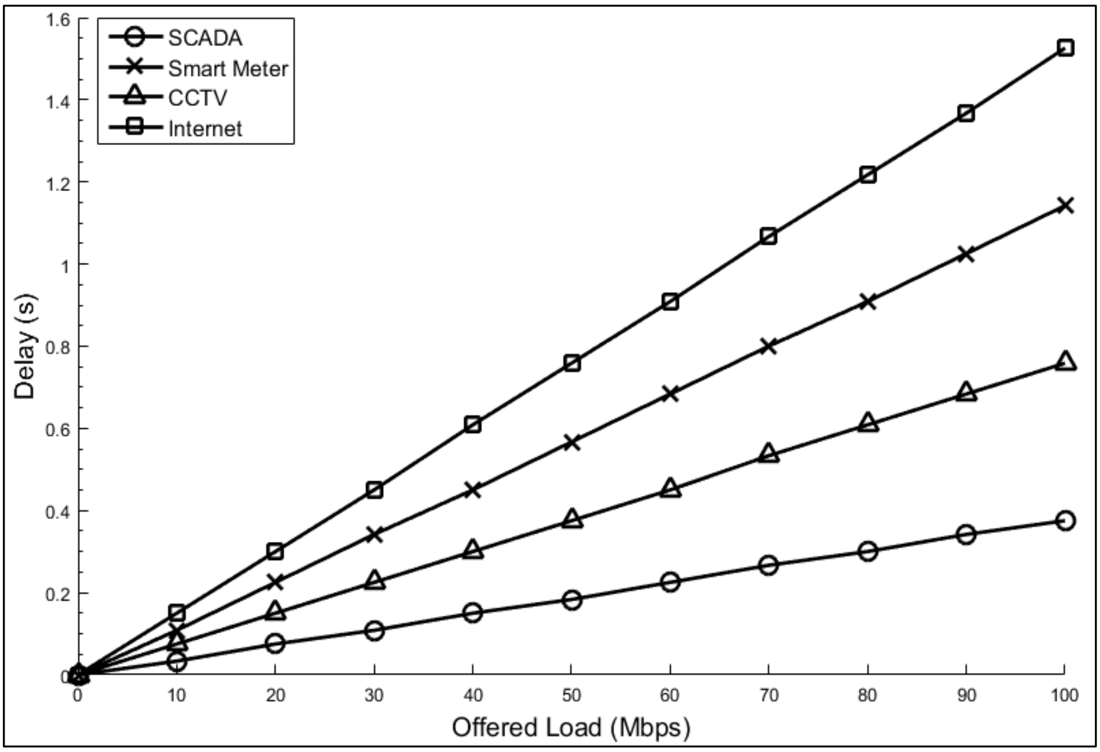

Figure 4.3 Delay versus offered load for CATSchA in Case 3

In this case, SCADA has the shortest delay because in CATSchA algorithm, SCADA will always have the highest rank and will be transmitted first without taking into account of its inequality parameters which involved data size, data rate and previous transmission delay. CCTV is ranked as 2 as both of its data size and data rate are smaller than threshold (bit 1) whereby its previous transmission delay is larger than threshold (bit 0) and thus, its header is formed as (110). Smart Meter is ranked as 7 as both of its data size and data rate are larger than threshold (bit 0) whereby its previous transmission delay is smaller than threshold (bit 1) and thus, its header is formed as (001). Internet is ranked as 8 as all three of its data size, data rate and previous transmission delay are larger than threshold (bit 0) and thus, its header is formed as (000).

Case 3 traffic priority ranking from high to low consists of SCADA at rank 1, CCTV at rank 2, Smart Meter at rank 7 and Internet at rank 8. Therefore, it can be proved as in the graph that the delay of each traffic are tally to their respective ranking.

The result in terms of delay for Case 4 is shown in Figure 4.4, where the trend of four traffics observed below show that SCADA, CCTV, Internet and Smart Meter are arranged from the shortest to the longest delay with CATSchA algorithm. Delays of all traffics with CATSchA algorithm increase linearly until they reach a delay of 0.38 *s*, 0.76 *s*, 1.14 *s* and 1.53 *s* at 100 *Mbps* offered load respectively.

In this case, SCADA has the shortest delay as it always have the highest rank without taking into account of its inequality parameters which involved data size, data rate and previous transmission delay. CCTV is ranked as 2 as both of its data size and data rate are smaller than threshold (bit 1) whereby its previous transmission delay is larger than threshold (bit 0) and thus, its header is formed as (110). Internet is ranked as 7 as both of its data size and data rate are larger than threshold (bit 0) whereby its previous transmission delay is smaller than threshold (bit 1) and thus, its header is formed as (001). Smart Meter is ranked as 8 as all three of its data size, data rate and previous transmission delay are larger than threshold (bit 0) and thus, its header is formed as (000).

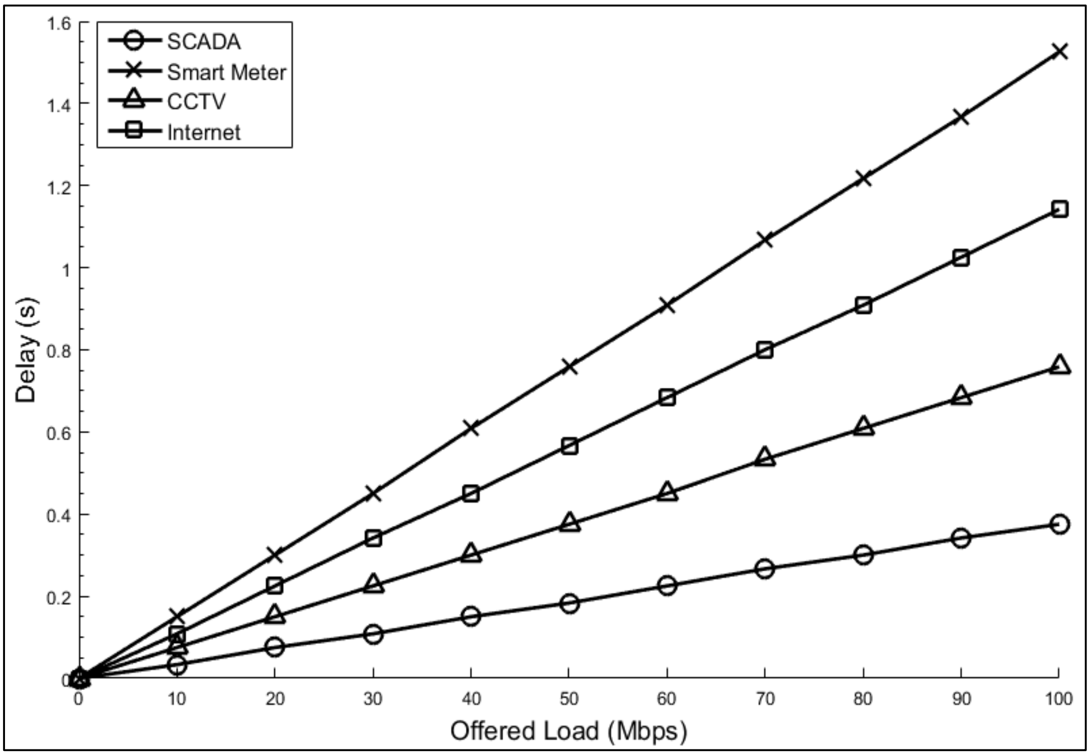

Figure 4.4 Delay versus offered load for CATSchA in Case 4

Priority ranking of each traffic in Case 4 from high to low involves SCADA at rank 1, CCTV at rank 2, Internet at rank 7 and Smart Meter at rank 8 where it can be proven as in Figure 4.4 that the delay of each traffic resembles its respective ranking.

The trend for delay result of Case 5 in Figure 4.5 is such a way that SCADA has the shortest delay. Right after SCADA is Internet with the second shortest delay, followed by Smart Meter and lastly CCTV. It can be seen that delays of all traffics increase linearly where each of them reach a delay of 0.38 *s*, 0.76 *s*, 1.14 *s* and 1.53 *s* at full offered load respectively.

In this case, SCADA has the shortest delay as it always have the highest rank without taking into account of its inequality parameters which involved data size, data rate and previous transmission delay. Internet is ranked as 2 as both of its data size and data rate are smaller than threshold (bit 1) whereby its previous transmission delay is larger than threshold (bit 0) and thus, its header is formed as (110). Smart Meter is ranked as 7 as both of its data size and data rate are larger than threshold (bit 0) whereby its previous transmission delay is smaller than threshold (bit 1) and thus, its header is formed as (001). CCTV is ranked as 8 as all three of its data size, data rate and previous transmission delay are larger than threshold (bit 0) and thus, its header is formed as (000).

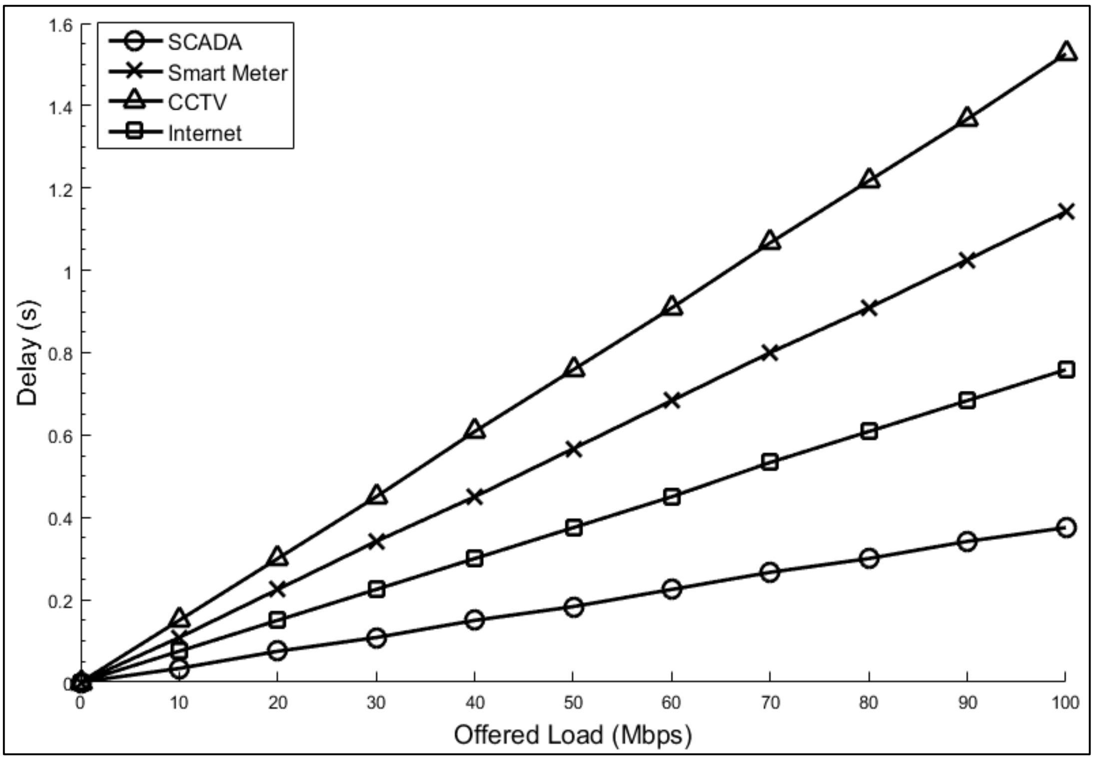

Figure 4.5 Delay versus offered load for CATSchA in Case 5

Priority ranking in Case 5 from high to low comprises of SCADA at rank 1, Internet at rank 2, Smart Meter at rank 7 and CCTV at rank 8 and thus, it can be proven as in the graph that the delay of each traffic are similar with their respective ranking.

As shown in Figure 4.6, for Case 6 the shortest delay is SCADA, followed by Internet, CCTV and lastly Smart Meter where each of them have delay of 0.27 *s*, 0.53 *s*, 0.8 *s* and 1.07 *s* at 70 *Mbps* offered load respectively.

In this case, SCADA has the shortest delay because in CATSchA algorithm, SCADA will always have the highest rank and will be transmitted first without taking into account of its inequality parameters which involved data size, data rate and previous transmission delay. Internet is ranked as 2 as both of its data size and data rate are smaller than threshold (bit 1) whereby its previous transmission delay is larger than threshold (bit 0) and thus, its header is formed as (110). CCTV is ranked as 7 as both of its data size and data rate are larger than threshold (bit 0) whereby its previous transmission delay is smaller than threshold (bit 1) and thus, its header is formed as (001). Smart Meter is ranked as 8 as all three of its data size, data rate and previous transmission delay are larger than threshold (bit 0) and thus, its header is formed as (000).

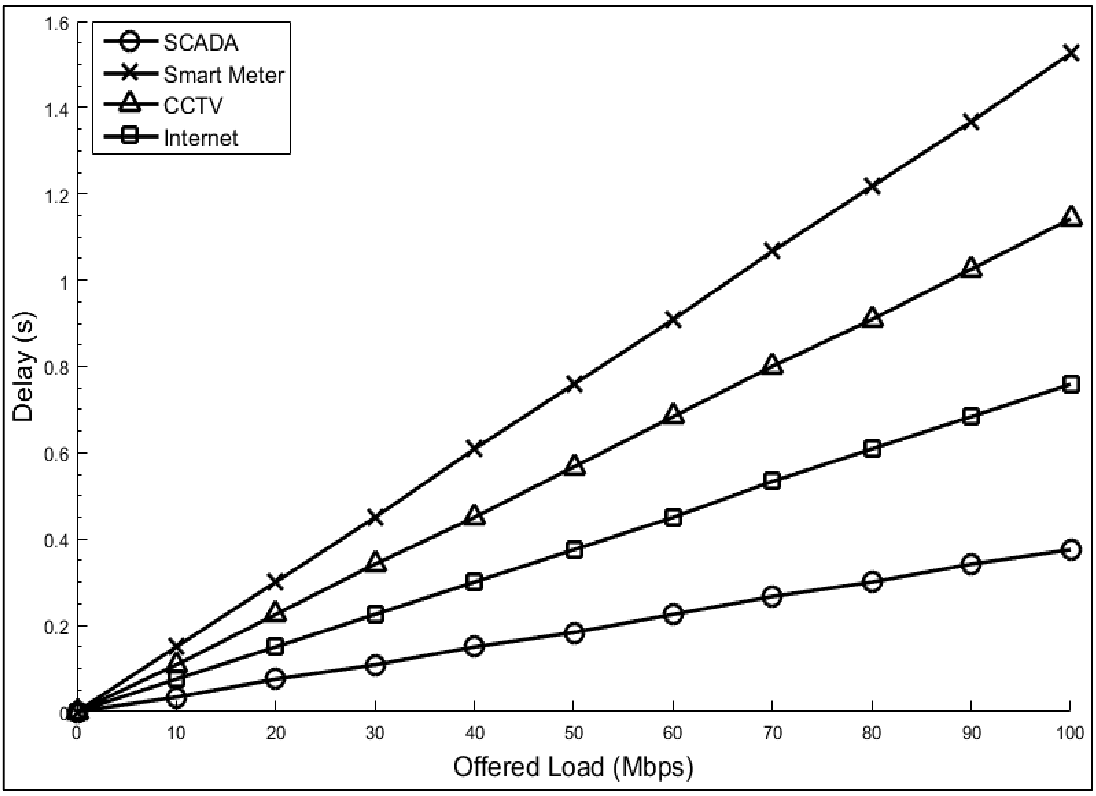

Figure 4.6 Delay versus offered load for CATSchA in Case 6

Case 6 traffic priority ranking from high to low consists of SCADA at rank 1, Internet at rank 2, CCTV at rank 7 and Smart Meter at rank 8. Hence, it can be proven as in the graph that the delay of each traffic corresponds to their respective ranking.

4.3.2 Efficiency

Efficiency is defined as the quality of throughput or data transferred between source and destination as it reflects to the performance of CATSchA algorithm. The higher the throughput, the better the efficiency of CATSchA algorithm. The formula of calculating efficiency, *Eff* is shown in Equation 4.1 [94].

$$Eff = \frac{Thr_{total}}{Link\ capacity} \times 100 \qquad \text{(Equation 4.1)}$$

where Thr_{total} denotes the total throughput of SCADA, Smart Meter, CCTV and Internet at full load divided by the link capacity of the network which is 100 *Mbps* times 100 to get the efficiency in terms of percentage.

The efficiency from Case 1 until Case 6 are shown in Figure 4.7, where four different traffics are taken into consideration during the simulation process at 100 *Mbps* offered load with CATSchA algorithm. It can be seen that the trend of six different cases observed shows that all traffics with CATSchA algorithm has the same amount of efficiency which is 81.65%.

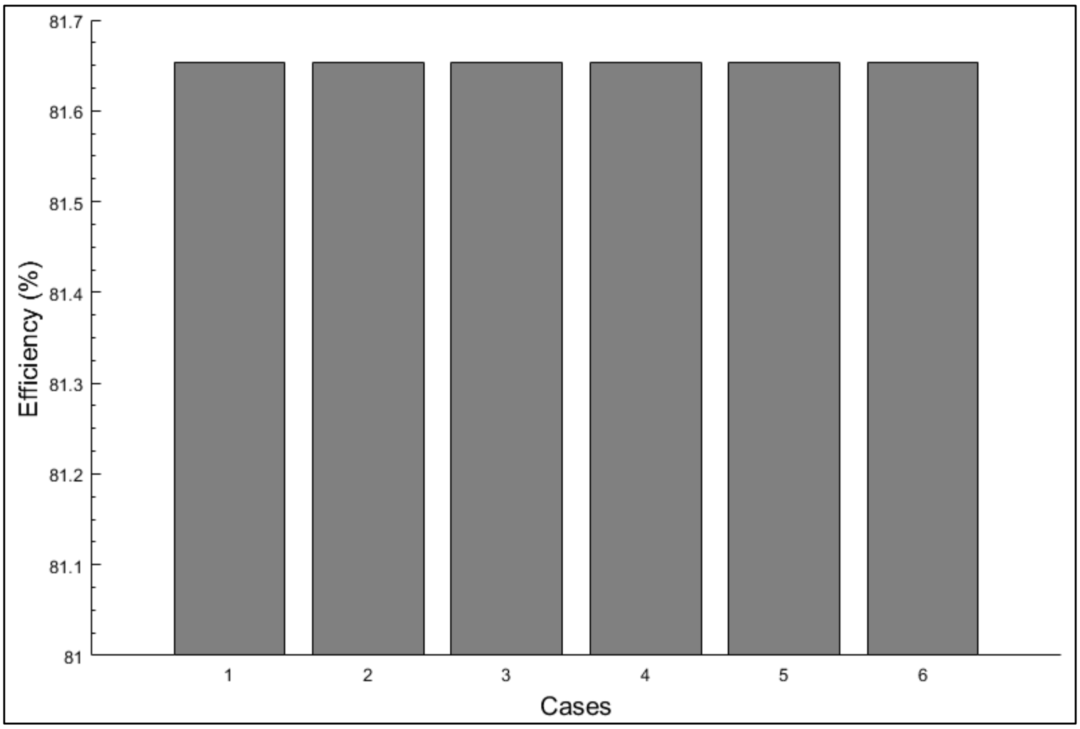

Figure 4.7 Efficiency for all cases

It is expected in this performance study that the CATSchA algorithm has similar efficiency for all cases. This is because the input design parameters which are data size and data rate are the same for SCADA, Smart Meter, CCTV and Internet by depending on the total amount of offered load. Therefore, the concept of CATSchA algorithm is proven in this performance study.

4.4 Performance Evaluation of Traffic Scheduling (CATSchA) Algorithm: With Contextual Aware versus Without Contextual Aware

The proposed CATSchA algorithm is compared with without contextual aware to show its advantages and for the purpose of comparison. The same design parameters are used for both algorithms for comparison purposes. The comparison is done in terms of delay, jitter, throughput and efficiency by using six different cases namely Case 1 (priority from high to low: SCADA → Smart Meter → CCTV → Internet), Case 2 (priority from high to low: SCADA → Smart Meter → Internet → CCTV), Case 3 (priority from high to low: SCADA → CCTV → Smart Meter → Internet), Case 4 (priority from high to low: SCADA → CCTV → Internet → Smart Meter), Case 5 (priority from high to low: SCADA → Internet → Smart Meter → CCTV) and Case 6 (priority from high to low: SCADA → Internet → CCTV → Smart Meter). In this section, actual data size and data rate are used, varied for each traffic to replicate traffics condition in the grid. These traffics are bounded by their minimum and maximum threshold.

4.4.1 Delay

The delay result for Case 1 is shown in Figure 4.8, where it can be seen that the trend of four different traffics observed show that all traffics with CATSchA algorithm has lower delay compared to traffics without contextual aware.

SCADA delays with CATSchA algorithm and without contextual aware increase almost at a linear rate until both of them reach a delay of 0.3 *s* and 1.08 *s* at 100 *Mbps* offered load respectively. Smart Meter delays with CATSchA algorithm and without contextual aware increase steadily as the offered load increases from 0 to 100 *Mbps* where at full offered load, the delay with CATSchA algorithm is 0.71 *s* whereas for without contextual aware is 1.45 *s*. CCTV delays with CATSchA algorithm increases linearly and without contextual aware increases almost at a linear rate for 0 to 100 *Mbps* offered load. The delay of CCTV with CATSchA algorithm is 3.1 *s* whereas for without contextual aware is 7.2 *s* at full load. However, at 90 to 100 *Mbps* offered load, the delay without contextual aware had sudden increase from 5.06 *s* to 7.2 *s*. Internet delays with CATSchA algorithm increases linearly whereby without contextual aware increases almost at a linear rate as the offered load increases from 0 to 100 *Mbps* where both of them reach a delay of 3.94 *s* and 12.1 *s* at full offered load respectively. However, at 90 to 100 *Mbps* offered load, the delay without contextual aware has sudden increase from 8.27 *s* to 12.1 *s*.

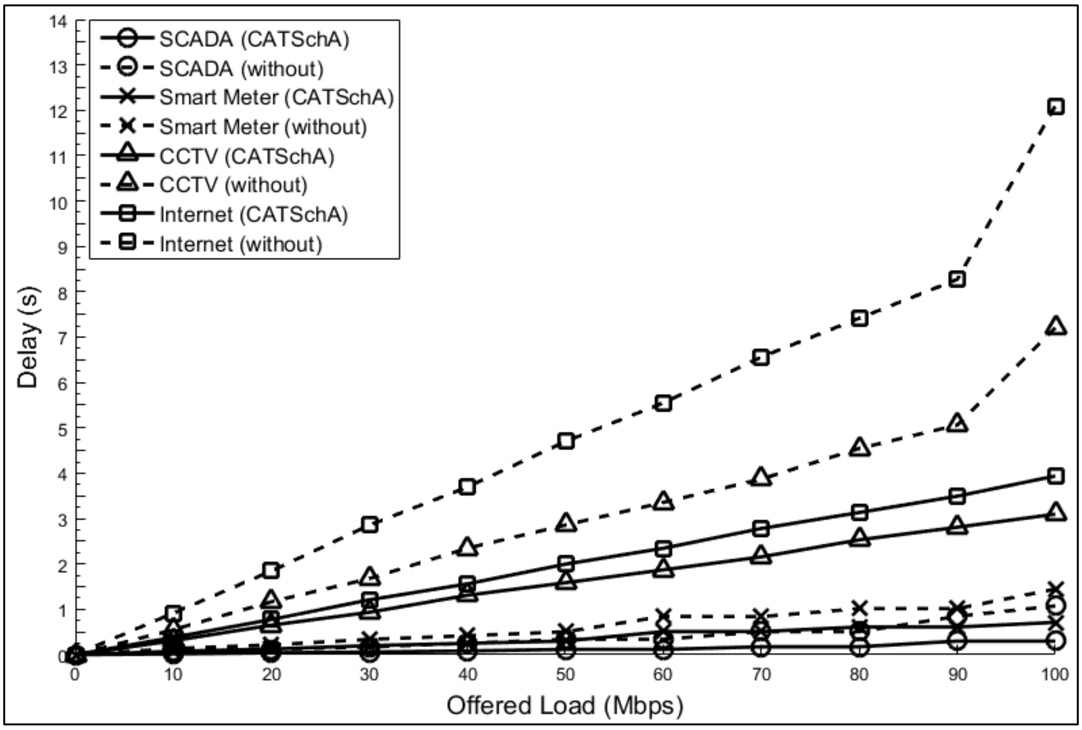

Figure 4.8 Delay versus offered load for CATSchA and without contextual aware in Case 1

Both CCTV and Internet without contextual aware are having sudden increase from 90 *Mbps* to 100 *Mbps* offered load due to large number of packets to be transmitted from source to destination. Whereby SCADA and Smart Meter without contextual aware have small number of packets which their transmission is completed much faster. Therefore, the remaining number of packets for CCTV and Internet are to be completed in longer time which explained the sudden rise as shown in the graph.

All traffics have improved their delays with CATSchA algorithm upon without contextual aware in which all of them have shorter delays compared to without contextual aware. This is because traffics in CATSchA algorithm have been queued according to their contextual aware priority whereby SCADA packets are transmitted first before considering Smart Meter, CCTV and Internet packets. Each traffic has their own specific allocated transmission time as has been set in CATSchA algorithm in which SCADA packets are to be completed first. Then, followed by Smart Meter, CCTV and Internet. This differs from without contextual aware that grant the packets for transmission according to pre-defined priority and number of packets. Each traffic transmits their packets by having to share the same transmission time. Thus, every packet needs to queue longer before it is being transmitted to the destination.

Figure 4.9 shows the delay of Case 2 for both CATSchA algorithm and without contextual aware. These scenarios illustrate different trend in traffic priority ranking where the shortest delay belongs to SCADA, followed by Smart Meter, Internet and CCTV for CATSchA algorithm. Meanwhile for without contextual aware, SCADA has the shortest delay, followed by Smart Meter, CCTV and Internet. All traffics with CATSchA algorithm has lower delay compared to traffics without the contextual aware.

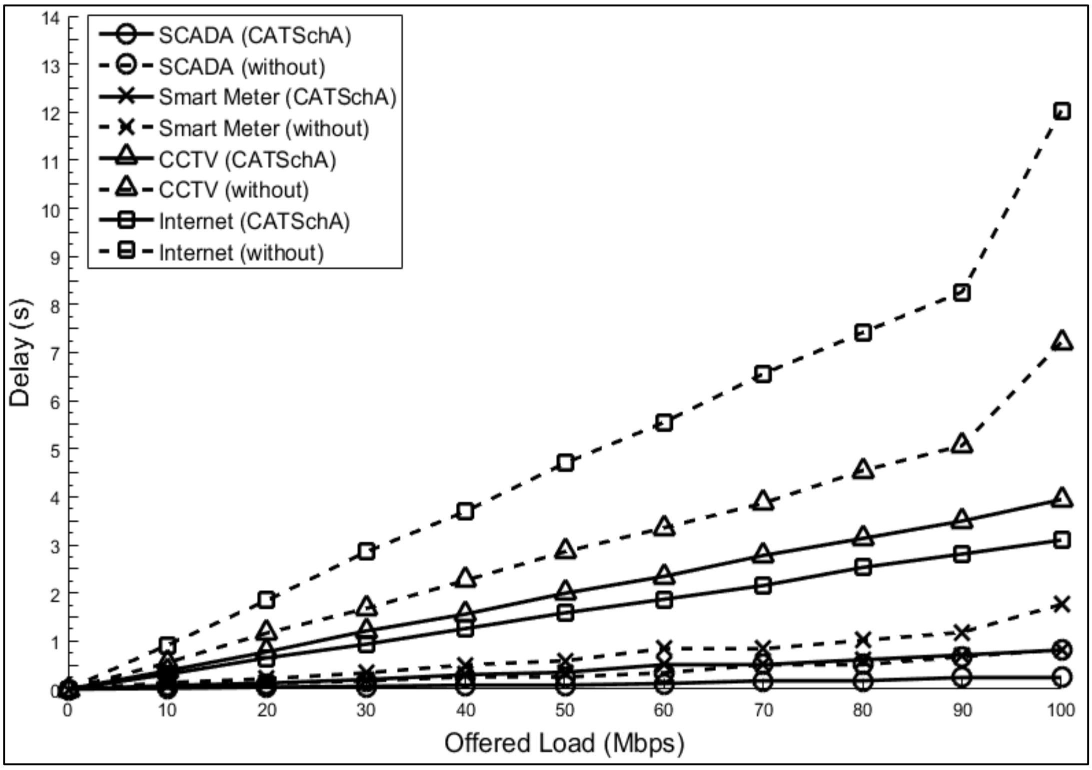

Figure 4.9 Delay versus offered load for CATSchA and without contextual aware in Case 2

SCADA delays for both algorithms increase linearly as the offered load increases from 0 to 100 *Mbps* where CATSchA algorithm and without contextual aware reach a delay of 0.24 *s* and 0.68 *s* at 90 *Mbps* offered load respectively. On the other hand, Smart Meter delays increase steadily from 0 to 100 *Mbps* offered load with CATSchA algorithm and without contextual aware until they reach 0.81 *s* and 1.76 *s* at full offered load respectively. Internet delay with CATSchA algorithm increases linearly as the offered load increases from 0 to 100 *Mbps* where at 90 *Mbps* offered load, the delay is 2.81 *s*. However for without contextual aware, the delay increases almost at a linear rate and had a sudden increase at 90 *Mbps* offered load which is from 8.26 *s* to 12.04 *s*. From 0 to 100 *Mbps* offered load, CCTV delay with CATSchA algorithm increases linearly whereas for without contextual aware the delay increases almost at a linear rate due to the sudden increase at 90 *Mbps* offered load. At 90 *Mbps* offered load, the delay has risen from 5.06 *s* until 7.22 *s* at full offered load.

Both CCTV and Internet without contextual aware are having sudden increase from 90 *Mbps* to 100 *Mbps* offered load due to large number of packets to be transmitted from source to destination. Whereby SCADA and Smart Meter without contextual aware have small number of packets which their transmission is completed much faster. Therefore, the remaining number of packets for CCTV and Internet are to be completed in longer time which explained the sudden rise as shown in the graph.

Delays for all traffics display an improvement in CATSchA algorithm compared to without contextual aware. This is due to the queuing arrangement of each traffic in CATSchA algorithm which has been done according to their contextual aware priority. Therefore, SCADA packets are transmitted first, followed by Smart Meter, Internet and lastly CCTV. Meanwhile, different queuing arrangement is applied for each traffic of without contextual aware since there is no priority ranking to be implemented. Thus, each traffic is following the pre-defined ranking in completing their packets transmission.

For Case 3 of delay result in Figure 4.10, it shows that SCADA, CCTV, Smart Meter and Internet are arranged from the shortest to the longest delay with CATSchA algorithm. On the other hand, the sequence for without contextual aware rank from SCADA, Smart Meter, CCTV and Internet.

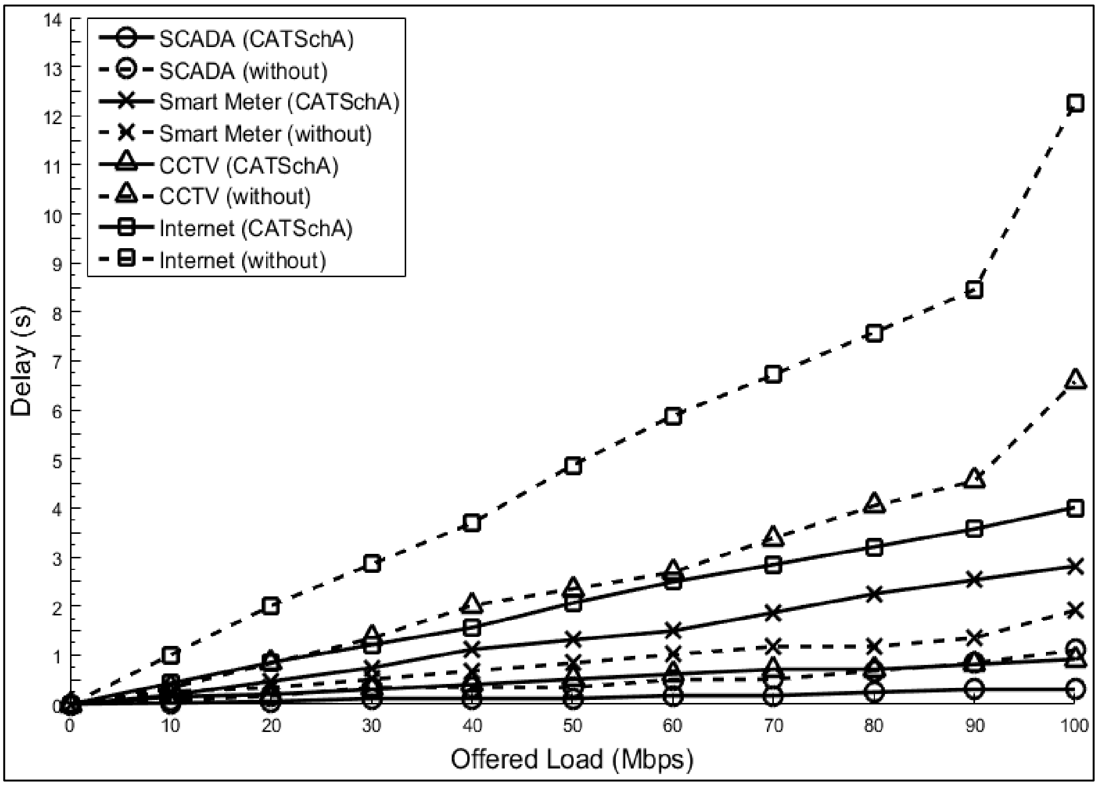

Figure 4.10 Delay versus offered load for CATSchA and without contextual aware in Case 3

The delays of SCADA with CATSchA algorithm and without contextual aware increase steadily where both of them reach a delay of 0.24 *s* and 0.68 *s* at 80 *Mbps* offered load respectively. CCTV delay with CATSchA algorithm increases linearly with 0.92 *s* at full offered load whereas the delay of CCTV without contextual aware increases almost at a linear rate with 6.58 *s* at 100 *Mbps* offered load. At 90 *Mbps* offered load, there is sudden rise of the delay in without contextual aware from 4.55 *s* to 6.58 *s* at full load. The delays of Smart Meter increase almost at a linear rate for both with CATSchA algorithm and without contextual aware from 0 to 100 *Mbps* offered load. The delay with CATSchA algorithm is 2.82 *s* whereas for without contextual aware is 1.91 *s* at full offered load respectively. Internet delay with CATSchA algorithm increases linearly as the offered load increases from 0 to 100 *Mbps*. In the meantime, the delay of Internet for without contextual aware increases steadily where there is abrupt delay increase from 8.45 *s* up to 12.26 *s* at 90 *Mbps* offered load and beyond.

Both CCTV and Internet without contextual aware are having sudden increase from 90 *Mbps* to 100 *Mbps* offered load due to large number of packets to be transmitted from source to destination. Whereby SCADA and Smart Meter without contextual aware have small number of packets which their transmission is completed much faster. Therefore, the remaining number of packets for CCTV and Internet are to be completed in longer time which explained the sudden rise as shown in the graph.

SCADA, CCTV and Internet have improved their delays with CATSchA algorithm except for Smart Meter in which all three of them have shorter delays compared to without contextual aware. It is expected for Smart Meter with CATSchA algorithm to have longer delay compared to without contextual aware due to its priority ranking is demoted from rank 2 to rank 7. Thus, Smart Meter packets in CATSchA algorithm have been queued according to its newly-defined ranking where SCADA packets are transmitted first, followed by CCTV, Smart Meter and lastly Internet packets.

The trend for delay result of Case 4 in Figure 4.11 is such a way that SCADA has the shortest delay with CATSchA algorithm. Right after SCADA is Internet with the second shortest delay, followed by Smart Meter and lastly CCTV. The trend differs from without contextual aware where the shortest delay belongs to SCADA, followed by Smart Meter, CCTV and lastly Internet with the longest delay.

It can be seen that delays of SCADA increase almost at a linear rate upon with CATSchA algorithm and without contextual aware where both of them reach a delay of 0.36 *s* and 1.31 *s* at full offered load respectively. Internet delay with CATSchA algorithm increases steadily as the offered load increases from 0 to 100 *Mbps*. On the other hand, the delay of Internet without contextual aware increases linearly for 0 to 90 *Mbps* offered load. Beyond 90 *Mbps* offered load, the delay abruptly increase until it reaches 12.07 *s* at 100 *Mbps* offered load. Smart Meter delays with CATSchA algorithm and without contextual aware increase steadily as the offered load increases from 0 to 100 *Mbps* where at 80 *Mbps* offered load, the delay with CATSchA algorithm is 1.72 *s* whereas for without contextual aware is 1.18 *s*. CCTV with CATSchA algorithm has a higher delay compared to the CCTV delay without contextual aware for 0 to 94 *Mbps* offered load approximately. Above 94 *Mbps* offered load onwards, CCTV with CATSchA algorithm starts to have a lower delay than the CCTV delay without contextual aware as high as 14.46% difference.

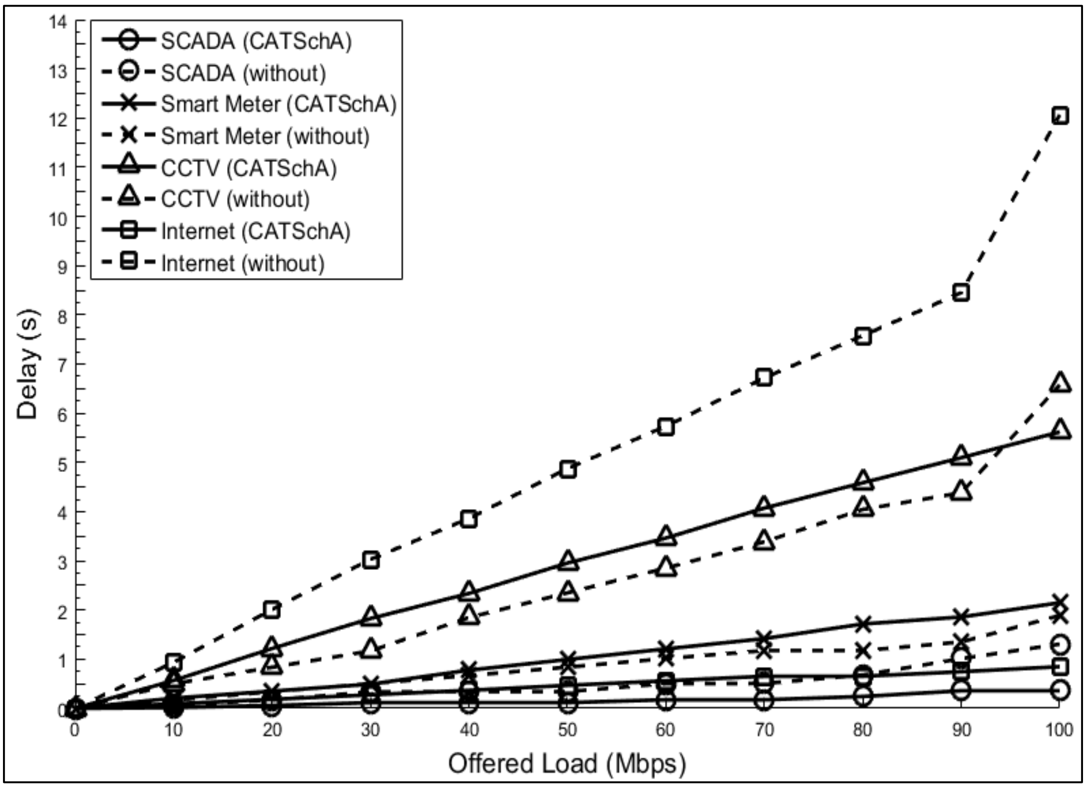

Figure 4.11 Delay versus offered load for CATSchA and without contextual aware in Case 4

Both CCTV and Internet without contextual aware are having sudden increase from 90 *Mbps* to 100 *Mbps* offered load due to large number of packets to be transmitted from source to destination. Whereby SCADA and Smart Meter without contextual aware have small number of packets which their transmission is completed much faster. Therefore, the remaining number of packets for CCTV and Internet are to be completed in longer time which explained the sudden rise as shown in the graph.

SCADA and Internet have improved their delays with CATSchA algorithm in which both of them have maintained shorter delays compared to without contextual aware as the offered load increases from 0 to 100 *Mbps*. CCTV has improved its delay beyond 94 *Mbps* offered load due to its large amount of data size although its newly-defined ranking (rank 2) is higher than its pre-defined ranking (rank 7). This is because for CCTV with CATSchA algorithm, the number of packets to be transmitted are smaller with specific allocated transmission time compared to CCTV without contextual aware which have greater number of packets to be transmitted without specific allocated transmission time. Smart Meter delay has not improved with CATSchA algorithm since its newly defined ranking (rank 8) is lower than its pre-defined ranking (rank 2). It is expected for Smart Meter with CATSchA algorithm to have longer delay compared to without contextual aware due to its priority ranking is demoted from rank 2 to rank 8. Supposedly in Case 4 the ranking of each traffic with CATSchA algorithm starts with SCADA at rank 1, CCTV at rank 2, Internet at rank 7 and Smart Meter at rank 8. However as can be seen in Figure 4.11, the traffics are not following their respective order due to constrained of input amount of data size, data rate and previous transmission delay that has been limited by their threshold requirements as in the standard.

There are few criteria in transmitting packets that limit the output in following the correct rank which includes that each traffic is bounded by minimum and maximum threshold values of data size, data rate and previous transmission delay with very small and large amount of offered load. Thus, when large data size traffic is required to upgrade its priority ranking with very small or large amount of offered load then, this requirement is unable to be fulfilled. Same goes for small data size traffic when it is required to downgrade its priority ranking with very small or large amount of offered load then, this requirement is also unable to be fulfilled. Hence, this limitation has reflected in the output traffic ranking.

The result in terms of delay for Case 5 is shown in Figure 4.12, where the trend of four traffics observed below show that SCADA, CCTV, Internet and Smart Meter are arranged from the shortest to the longest delay with CATSchA algorithm. In the meantime, the trend of the traffics without contextual aware shows a different arrangement of SCADA, Smart Meter, CCTV and Internet from shortest to longest delay rank.

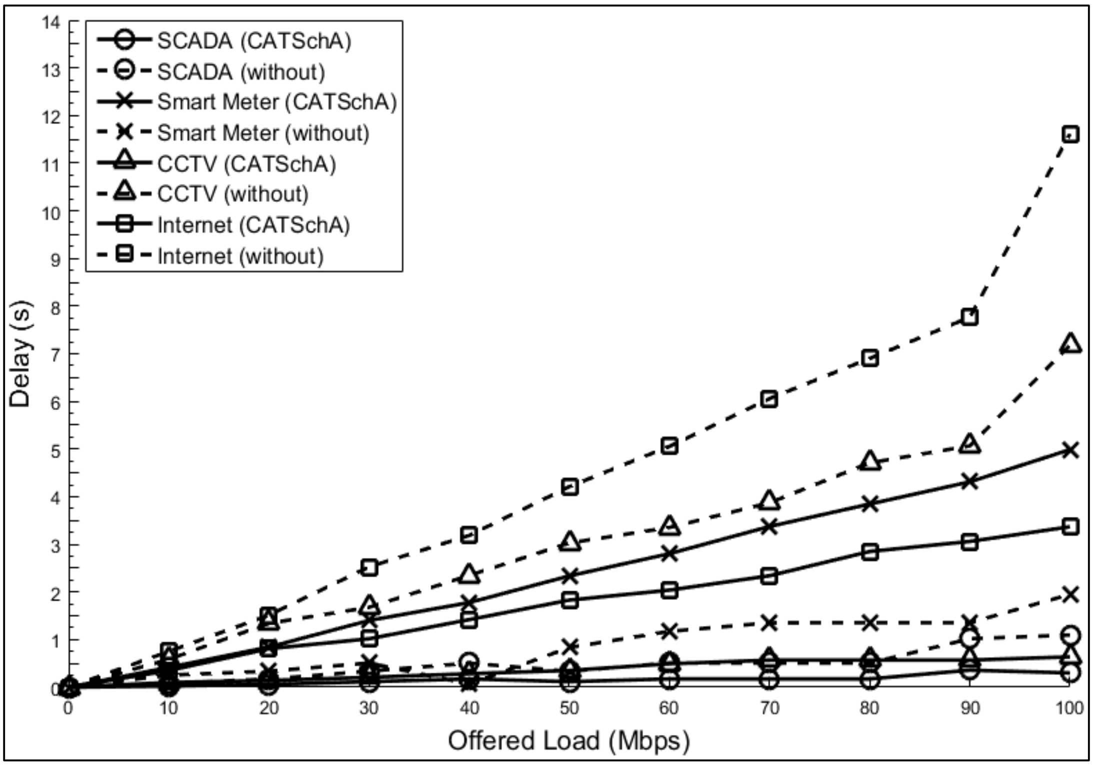

Figure 4.12 Delay versus offered load for CATSchA and without contextual aware in Case 5

Both delays of SCADA with CATSchA algorithm and without contextual aware increase steadily until they reach a delay of 0.3 *s* and 1.1 *s* at 100 *Mbps* offered load respectively. The delay of CCTV increases almost at a linear rate with CATSchA algorithm where the delay is 0.57 *s* at 80 *Mbps* offered load. For the time being, the delay of CCTV increases steadily without contextual aware where the delay is 4.71 *s* at 80 *Mbps* offered load. Beyond 90 *Mbps* offered load, CCTV delay has a sudden rise without contextual aware from 5.07 *s* up to 7.18 *s* at 100 *Mbps* offered load. Internet delay with CATSchA algorithm increases almost at a linear rate for 0 to 100 *Mbps* whereas without contextual aware the delay increases linearly from 0 to 90 *Mbps*. At 90 *Mbps* offered load, Internet delay has a sudden increase from 7.77 *s* to 11.61 *s* at full load. Smart Meter with CATSchA algorithm has a higher delay compared to without contextual aware as the offered load increases from 0 to 100 *Mbps* where at full offered load, the delay with CATSchA algorithm is 4.99 *s* whereas for without contextual aware is 1.95 *s*.

Both CCTV and Internet without contextual aware are having sudden increase from 90 *Mbps* to 100 *Mbps* offered load due to large number of packets to be transmitted from source to destination. Whereby SCADA and Smart Meter without contextual aware have small number of packets which their transmission is completed much faster. Therefore, the remaining number of packets for CCTV and Internet are to be completed in longer time which explained the sudden rise as shown in the graph.

Traffics other than Smart Meter have improved their delays with CATSchA algorithm upon without contextual aware in which all of them have shorter delays compared to without contextual aware. However, Smart Meter has not improved its delay with CATSchA algorithm due to its newly-defined ranking (rank 7) is lower than its pre-defined ranking (rank 2). It is expected for Smart Meter with CATSchA algorithm to have longer delay compared to without contextual aware due to its priority ranking is demoted from rank 2 to rank 7. Thus, its packets are queued as the third traffic to be transmitted. Supposedly for Case 5, the queue of traffic is arranged in a manner of SCADA, Internet, Smart Meter and CCTV for CATSchA algorithm. However, the trend differs from Figure 4.12 because the threshold requirements of data size, data rate and previous transmission delay has restricted the input design parameters.

There are few criteria in transmitting packets that limit the output in following the correct rank which includes that each traffic is bounded by minimum and maximum threshold values of data size, data rate and previous transmission delay with very small and large amount of offered load. Thus, when large data size traffic is required to upgrade its priority ranking with very small or large amount of offered load then, this requirement is unable to be fulfilled. Same goes for small data size traffic when it is required to downgrade its priority ranking with very small or large amount of offered load then, this requirement is also unable to be fulfilled. Hence, this limitation has reflected in the output traffic ranking.

As shown in Figure 4.13, traffic delays of Case 6 are categorized into CATSchA algorithm and without contextual aware. From the shortest delay which is SCADA, followed by Internet, CCTV and lastly Smart Meter with the longest delay are the trend observed for CATSchA algorithm. Whereas for without contextual aware, SCADA has the shortest delay, followed by Smart Meter, CCTV and Internet has the longest delay.

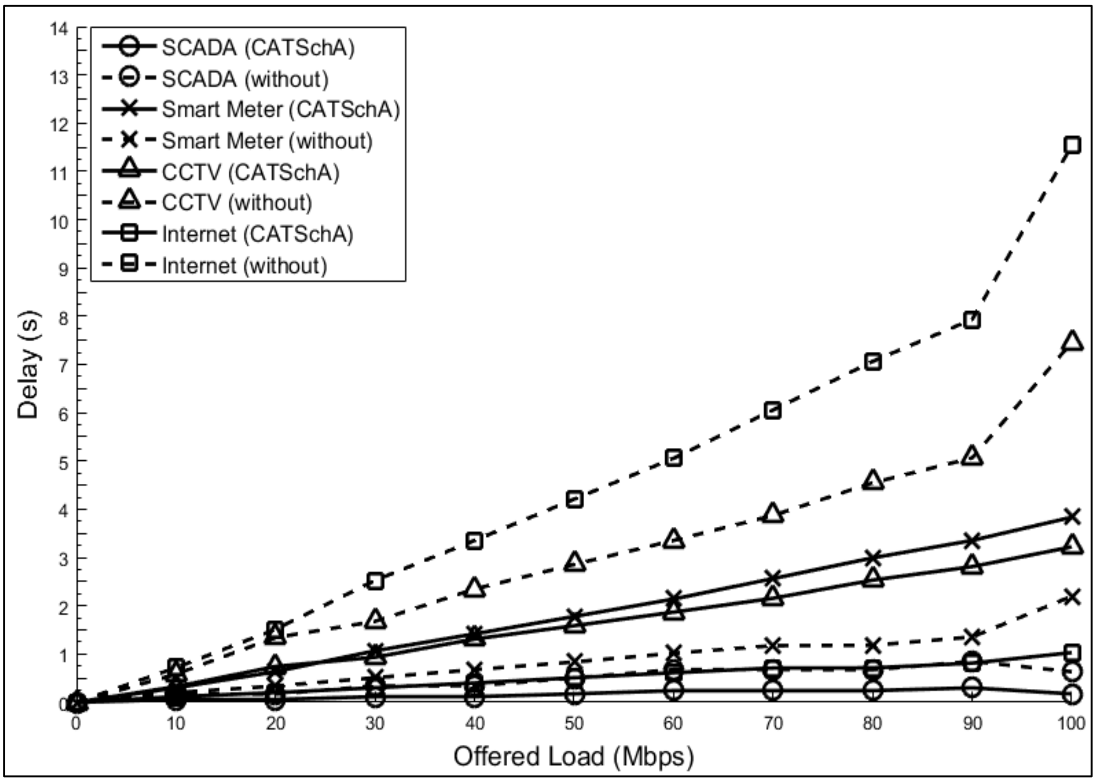

Figure 4.13 Delay versus offered load for CATSchA and without contextual aware in Case 6

SCADA, Internet and CCTV with CATSchA algorithm have lower delay compared to without contextual aware for 0 to 100 *Mbps* offered load. However, Smart Meter with CATSchA algorithm has a higher delay compared to without contextual aware as the offered load increases from 0 to 100 *Mbps* where the delay is 3.83 *s* and 2.2 *s* at full offered load respectively.

Both CCTV and Internet without contextual aware are having sudden increase from 90 *Mbps* to 100 *Mbps* offered load due to large number of packets to be transmitted from source to destination. Whereby SCADA and Smart Meter without contextual aware have small number of packets which their transmission is completed much faster. Therefore, the remaining number of packets for CCTV and Internet are to be completed in longer time which explained the sudden rise as shown in the graph.

SCADA, Internet and CCTV have improved their delays with CATSchA algorithm except for Smart Meter where its delay has not been improved as its newly-defined ranking (rank 8) is lower than its pre-defined ranking (rank 2). It is expected for Smart Meter with CATSchA algorithm to have longer delay compared to without contextual aware due to its priority ranking is demoted from rank 2 to rank 8. Therefore, packets of Smart Meter are queued as the last traffic to be transmitted.

4.4.2 Jitter

Jitter is a timing variations which can cause packets to arrive at an inconsistent rate. It is caused by congestion in the network. Jitter is an important parameter in evaluating the performance of CATSchA algorithm to show its advantages upon without contextual aware. SCADA, Smart Meter, CCTV and Internet require low amount of jitter. Therefore, the lower the jitter, the better the performance of CATSchA algorithm. The calculation of jitter can be referred to Appendix B which is the main source code for CATSchA algorithm as it shows the command line in getting the jitter by using NS-3 [90].

The jitter results for Case 1 is shown in Figure 4.14, where four different traffics are taken into consideration during the simulation process. It can be seen that the trend of most traffics observed below is in such a way that at full load, traffics with CATSchA algorithm has lower jitter compared to traffics without contextual aware. SCADA with CATSchA algorithm has a constant zero jitter due to its exclusiveness in transmission time for all of its packets to be transmitted in prior without competing with other traffics for their packets to arrive at a consistent rate.

SCADA jitter with CATSchA algorithm is constantly zero as the offered load increases from 0 to 100 *Mbps*. On the other hand, SCADA jitter without contextual aware consists of three peaks where at 60 *Mbps* offered load, the jitter has a maximum peak at 0.17 *ms*. Smart Meter jitter with CATSchA algorithm comprises of three peaks where at 80 *Mbps* offered load, the jitter is 0.058 *ms* at the maximum peak. Whereby, Smart Meter jitter without contextual aware includes four peaks which is at 50 *Mbps* offered load, the jitter is 0.17 *ms* at the maximum peak. CCTV jitter with CATSchA algorithm contains four peaks where at 90 *Mbps* offered load, the jitter has the maximum peak of 0.033 *ms*. Whereas, CCTV jitter without contextual aware encompasses three peaks which is at 10 *Mbps* offered load, the jitter reaches 0.056 *ms* at maximum peak. Internet jitter with CATSchA algorithm has two peaks where at 50 *Mbps* offered load, the jitter has the maximum peak at 0.021 *ms*. Meanwhile, Internet jitter without contextual aware consists of three peaks which is at 90 *Mbps* offered load, the jitter is 0.05 *ms* at maximum peak.

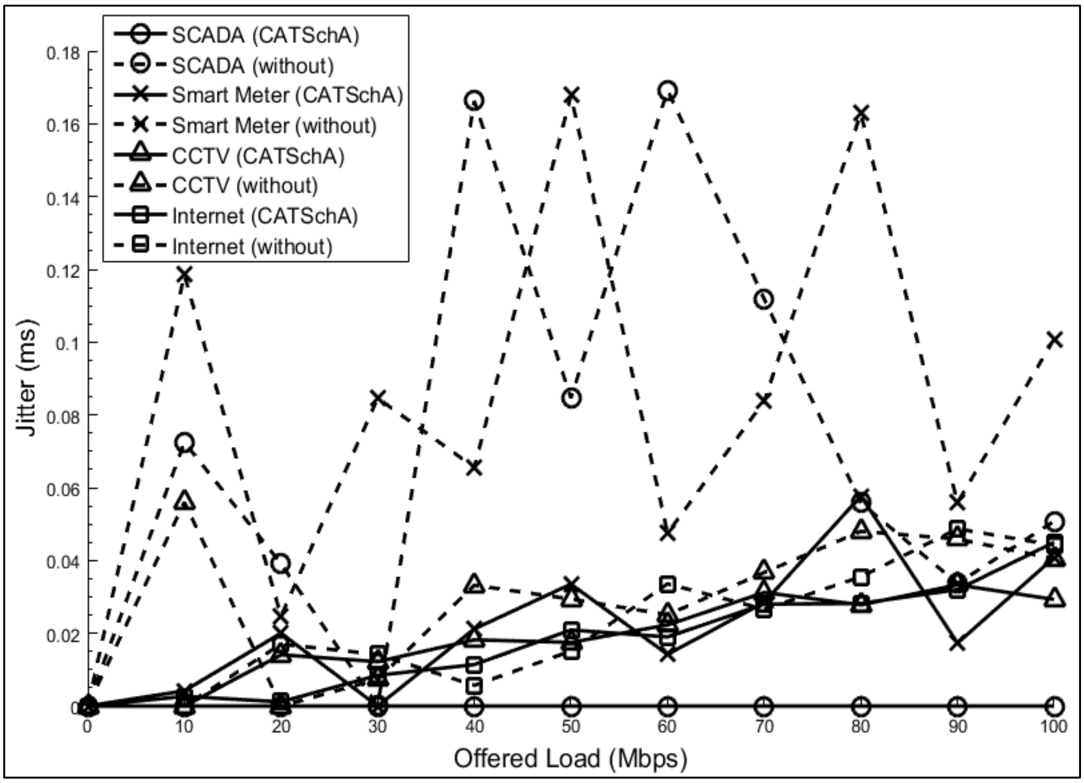

Figure 4.14 Jitter versus offered load for CATSchA and without contextual aware in Case 1

All traffics have improved their jitter with CATSchA algorithm in which all of them have less jitter compared to without contextual aware. This is due to the less timing variations in CATSchA algorithm which can cause packet to arrive at minimal inconsistent rate compared to more timing variations in without contextual aware which cause packets to arrive at high inconsistent rate. Traffics transmitted via CATSchA algorithm have their own allocated transmission time which is able to reduce the timing variations for packets arrival. This differs from traffics without contextual aware that are taking their turns in sending their packets which cause the timing variations to be highly inconsistent since all four traffics are competing with each other. Thus, the arrival of all their packets vary from time to time.

Figure 4.15 shows the jitter results of Case 2 for both CATSchA algorithm and without contextual aware. These scenarios illustrate different trend where the traffics with CATSchA algorithm has lower jitter compared to traffics without contextual aware at full offered load. On the other hand, SCADA with CATSchA algorithm has a constant zero jitter for all of its packets to be transmitted in prior to be arrived at a consistent rate.

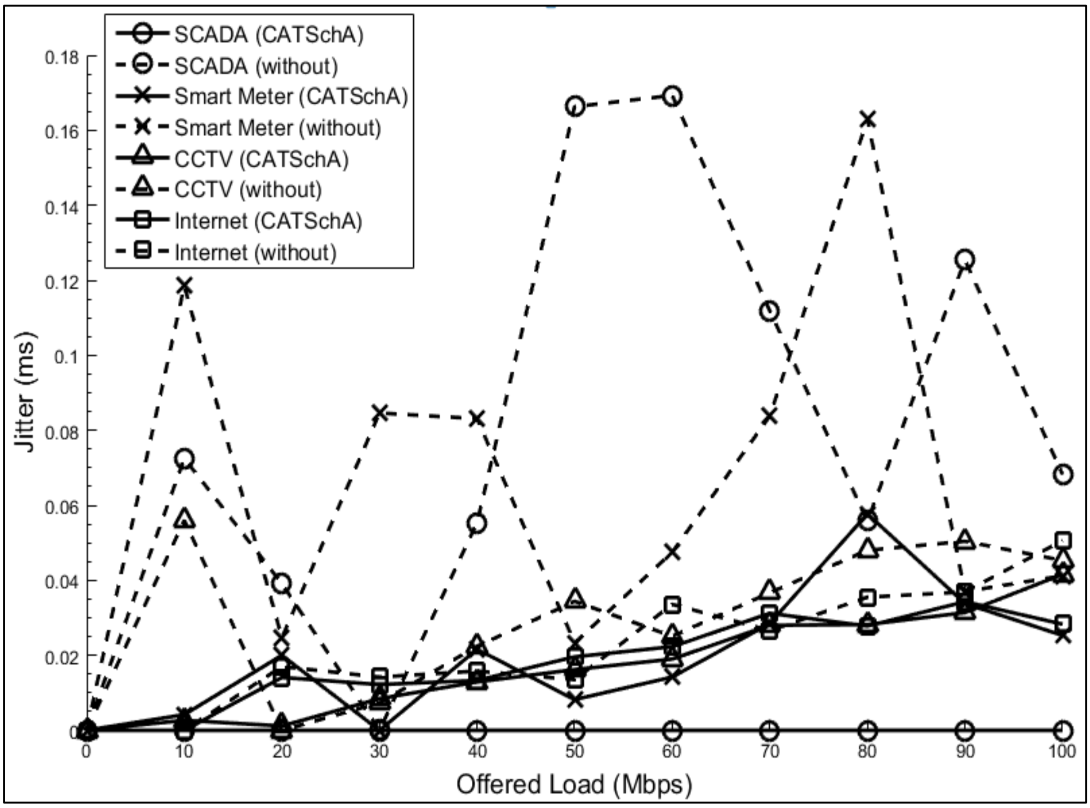

Figure 4.15 Jitter versus offered load for CATSchA and without contextual aware in Case 2

SCADA jitter for CATSchA algorithm has remained as zero from 0 to 100 *Mbps* offered load. Whereas SCADA jitter without contextual aware consists of four peaks where at 60 *Mbps* offered load, the jitter has a maximum peak at 0.17 *ms*. On the other hand, Smart Meter jitter comprises of three peaks with CATSchA algorithm where at 80 *Mbps* offered load, the jitter is 0.058 *ms* at the maximum peak. Without contextual aware includes four peaks which is at 80 *Mbps* offered load, the jitter is 0.16 *ms* at the maximum peak for Smart Meter. Internet jitter with CATSchA algorithm contains three peaks where at 90 *Mbps* offered load, the jitter has the maximum peak of 0.034

ms. Meanwhile, for without contextual aware, the jitter encompasses three peaks which is at 60 *Mbps* offered load, the jitter reaches 0.034 *ms* at maximum peak. From 0 to 100 *Mbps* offered load, CCTV jitter with CATSchA algorithm has one peak where at 10 *Mbps* offered load, the jitter is at 0.003 *ms.* Whereas for without contextual aware, the jitter consists of three peaks which is at 90 *Mbps* offered load, the jitter is 0.05 *ms* at maximum peak.

Jitter for all traffics displayed an improvement via CATSchA algorithm upon without contextual aware as each one of them have less jitter than without contextual aware. This is due to the queuing arrangement of each traffic in CATSchA algorithm which has been done according to their contextual aware priority along with their own allocated transmission time. Therefore, timing variations for packets arrival can be reduced at minimal inconsistent rate.

For Case 3 of jitter result in Figure 4.16, it shows that SCADA, Smart Meter and Internet have lower jitter with CATSchA algorithm at full offered load whereby CCTV has higher jitter at 100 *Mbps* offered load. In the meantime, SCADA jitter with CATSchA algorithm has remained zero as the offered load increases from 0 to 100 *Mbps* since all of its packets are expected to be arrived at a consistent rate.

The jitter of SCADA without contextual aware consists of three peaks where at 50 *Mbps* offered load, the jitter has a maximum peak of 0.17 *ms*. CCTV jitter with CATSchA algorithm contains four peaks where at 50 *Mbps* offered load, the jitter has maximum peak of 0.055 *ms.* Whereas the jitter of CCTV without contextual aware encompasses three peaks which is at 20 *Mbps* offered load, the jitter reaches 0.033 *ms* at maximum peak. The jitter of Smart Meter comprises of two peaks where at 90 *Mbps* offered load, the jitter is 0.027 *ms* at the maximum peak for CATSchA algorithm. Without contextual aware includes four peaks which is at 30 *Mbps* offered load, the jitter is 0.17 *ms* at the maximum peak. Internet jitter with CATSchA algorithm has three peaks where at 60 *Mbps* offered load, the jitter has the maximum peak at 0.024 *ms*. In the meantime, the jitter of Internet for without contextual aware consists of three peaks which is at 10 *Mbps* offered load, the jitter is 0.056 *ms* at maximum peak.

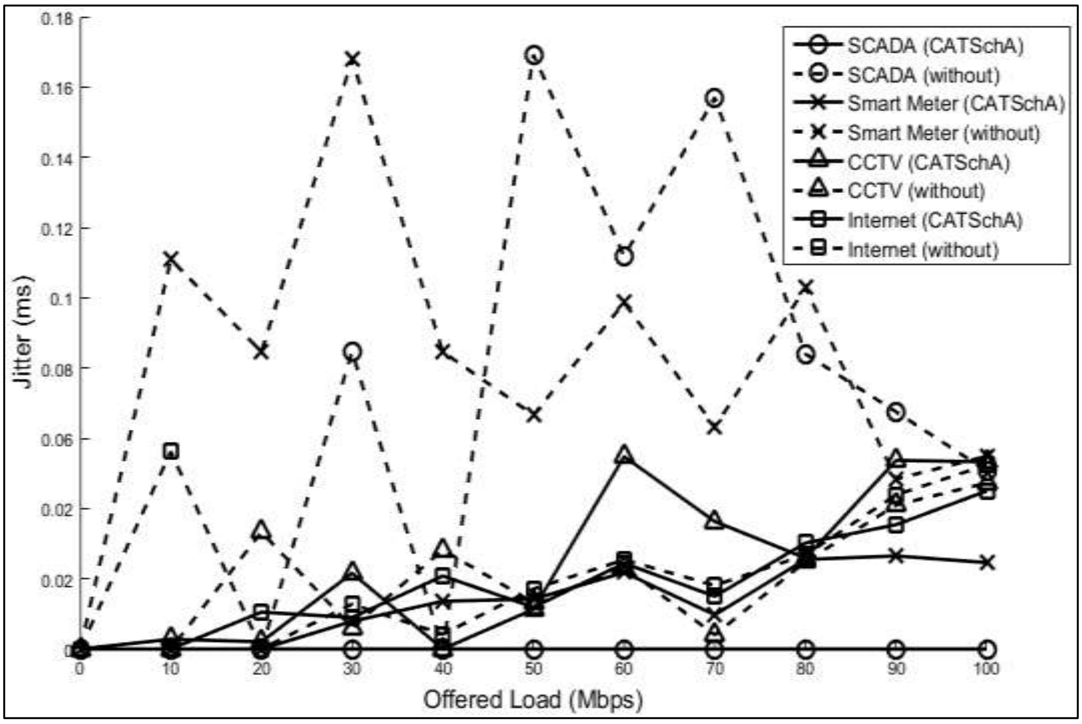

Figure 4.16 Jitter versus offered load for CATSchA and without contextual aware in Case 3

SCADA, Smart Meter and Internet have improved their jitter with CATSchA algorithm in which all of them have less jitter compared to without contextual aware. Less amount of jitter means less timing variations for a packet to arrive at minimal inconsistent rate in CATSchA algorithm. However, CCTV has not improved its jitter with CATSchA algorithm due to its newly-defined ranking (rank 2) is higher than its pre-defined ranking (rank 7) which means that all of its packets are required to be arrived at a faster rate compared to its previous transmission rate. This situation causes CCTV packets to be arrived at slightly higher inconsistent rate.

The trend for jitter result of Case 4 in Figure 4.17 is such a way that SCADA, CCTV, Internet and Smart Meter have lower jitter with CATSchA algorithm at 100 *Mbps* offered load. For SCADA with CATSchA algorithm, jitter is constantly zero from 0 to 100 *Mbps* offered load. Meanwhile, SCADA, Smart Meter, CCTV and Internet without contextual aware have higher jitter at full offered load.

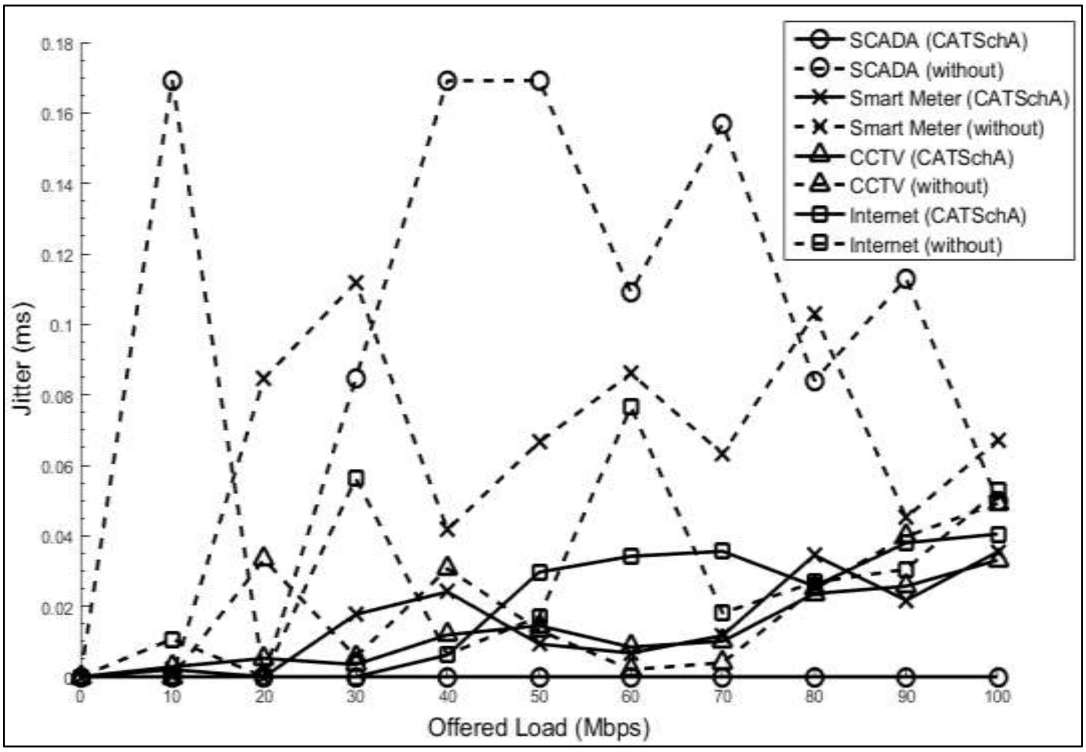

Figure 4.17 Jitter versus offered load for CATSchA and without contextual aware in Case 4

It can be seen that jitter of SCADA consists of five peaks where at 10 *Mbps* offered load, the jitter has a maximum peak at 0.17 *ms* for without contextual aware. CCTV with CATSchA algorithm contains two peaks where at 50 *Mbps* offered load, the jitter has the maximum peak of 0.015 *ms* compared to the CCTV jitter without contextual aware encompasses two peaks which is at 20 *Mbps* offered load, the jitter reaches 0.033 *ms* at maximum peak. Internet jitter with CATSchA algorithm has one peak where at 70 *Mbps* offered load, the jitter has the maximum peak at 0.036 *ms*. On the other hand, the jitter of Internet without contextual aware consists of three peaks which is at 60 *Mbps* offered load, the jitter is 0.077 *ms* at maximum peak. Smart Meter jitter with CATSchA algorithm and without contextual aware comprises of three peaks where at 80 *Mbps* offered load, the jitter is 0.035 *ms* at the maximum peak whereby at 30 *Mbps* offered load, the jitter is 0.11 *ms* respectively.

SCADA, CCTV, Internet and Smart Meter have improved their jitter with CATSchA algorithm in which all of them have less jitter compared to without contextual aware as the offered load increases from 0 to 100 *Mbps*. Less timing variations in CATSchA algorithm have improved jitter for all traffics via minimizing packet arrival at inconsistent rate.

The result in terms of jitter for Case 5 is shown in Figure 4.18, where the trend of four traffics observed below show that SCADA, Internet, Smart Meter and CCTV are having lower amount of jitter with CATSchA algorithm at 100 *Mbps* offered load. In the meantime, the trend of the traffics without contextual aware shows that SCADA, Smart Meter, CCTV and Internet are having higher amount of jitter at full offered load. The jitter of SCADA with CATSchA algorithm has stayed zero for all of its packets to be transmitted and arrived at a consistent rate.

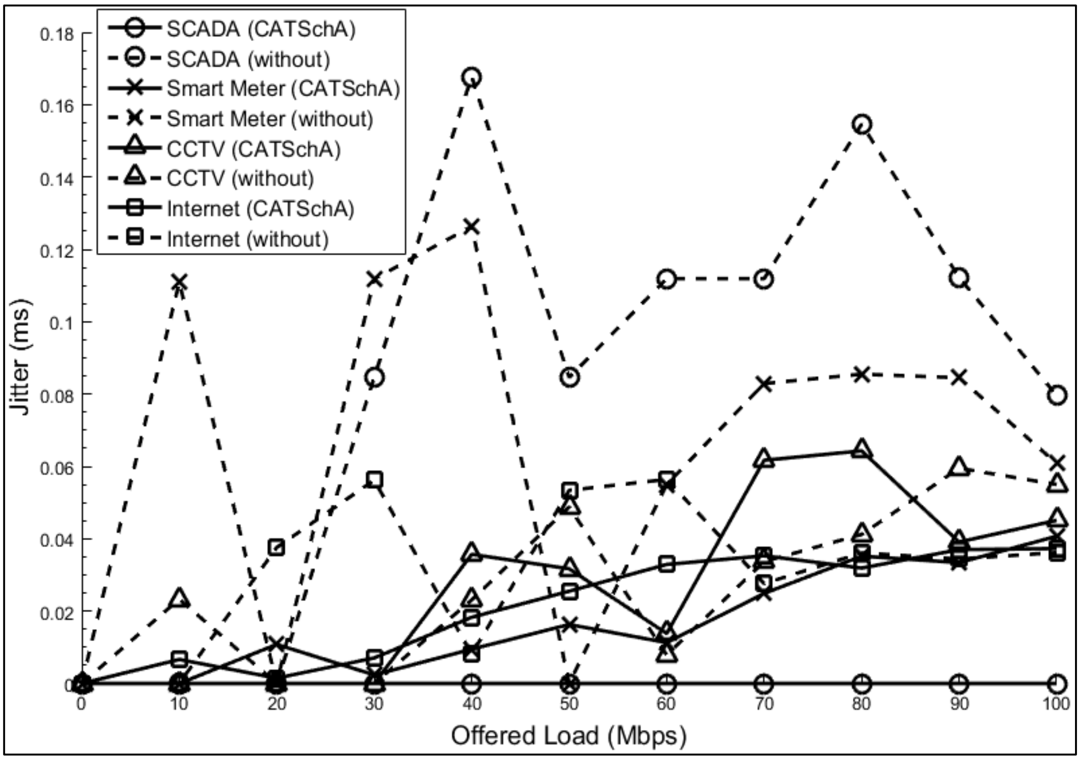

Figure 4.18 Jitter versus offered load for CATSchA and without contextual aware in Case 5

Jitter of SCADA without contextual aware consists of two peaks where at 40 *Mbps* offered load, the jitter has a maximum peak at 0.17 *ms*. Internet jitter with CATSchA algorithm has two peaks where at 70 *Mbps* offered load, the jitter has the maximum peak at 0.035 *ms* whereas without contextual aware the jitter consists of two peaks which is at 60 *Mbps* offered load, the jitter is 0.056 *ms* at maximum peak. Smart Meter with CATSchA algorithm comprises jitter with three peaks where at 80 *Mbps* offered load, the jitter is 0.035 *ms* at the maximum peak compared to without contextual aware the jitter includes three peaks which is at 40 *Mbps* offered load, the jitter is 0.13 *ms* at the maximum peak. The jitter of CCTV contains two peaks where at 80 *Mbps* offered load, the jitter has the maximum peak of 0.064 *ms* with CATSchA algorithm. For the time being, the jitter of CCTV encompasses three peaks which is at 90 *Mbps* offered load, the jitter reaches 0.06 *ms* at maximum peak without contextual aware.

All traffics have improved their jitter with CATSchA algorithm upon without contextual aware in which all of them have less jitter compared to without contextual aware. Each traffic have less timing variations which directly cause their packets to be arrived at destination with low inconsistent rate.

As shown in Figure 4.19, each traffic jitter of Case 6 are been categorized into CATSchA algorithm and without contextual aware. All traffics with CATSchA algorithm have less amount of jitter whereas for without contextual aware, the traffics have higher amount of jitter at full offered load respectively. The jitter of SCADA is continuous to be zero as the offered load increases from 0 to 100 *Mbps* with CATSchA algorithm due to its exclusivity in transmission time for all of its packets to be transmitted and arrived at a consistent rate.

SCADA jitter without contextual aware consists of four peaks where at 80 *Mbps* offered load, the jitter has a maximum peak at 0.13 *ms*. Internet jitter with CATSchA algorithm has two peaks where at 60 *Mbps* offered load, the jitter has the maximum peak at 0.036 *ms*. Meanwhile, Internet jitter without contextual aware consists of two peaks which is at 60 *Mbps* offered load, the jitter is 0.056 *ms* at maximum peak. CCTV jitter with CATSchA algorithm contains three peaks where at 70 *Mbps* offered load, the jitter has the maximum peak of 0.031 *ms*. Whereas, CCTV jitter without contextual aware encompasses three peaks which is at 90 *Mbps* offered load, the jitter reaches 0.054 *ms* at maximum peak. Despite that, Smart Meter with CATSchA algorithm comprises of jitter with three peaks where at 60 *Mbps* offered load, the jitter is 0.044 *ms* at the maximum peak compared to without contextual aware the jitter includes three peaks which is at 60 *Mbps* offered load, the jitter is 0.11 *ms* at the maximum peak.

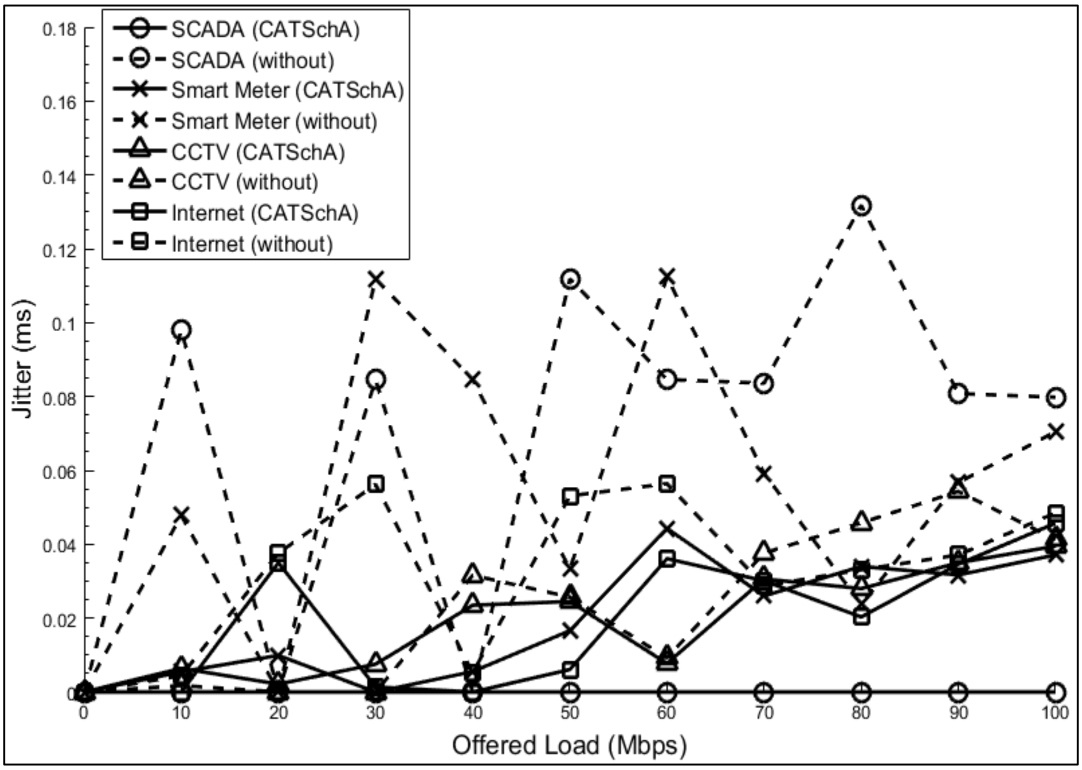

Figure 4.19 Jitter versus offered load for CATSchA and without contextual aware in Case 6

SCADA, Internet, CCTV and Smart Meter have improved their jitter with CATSchA algorithm where their packets arrival are having less timing variations due to their own allocated transmission time. Therefore, packets of all traffics be able to arrive at minimal inconsistent rate compared to packets without the contextual aware.

4.4.3 Throughput

Throughput is defined as the amount of data transferred between source and destination as to show the performance between CATSchA algorithm and without contextual aware. SCADA, Smart Meter, CCTV and Internet require high amount of throughput or at least maintained the same throughput as without contextual aware. The throughput, *Thr* is calculated by using Equation 4.2 [95].

$$Thr\ (Mbps) = \frac{S}{T} \qquad \text{(Equation 4.2)}$$

where S denotes data size of SCADA, Smart Meter, CCTV and Internet combined at full load divided by T which represents cycle time of the network in order to get the amount of throughput in terms of *Mbps*.

The total throughput result for Case 1 is shown in Figure 4.20, where four different traffics are taken into consideration during the simulation process. It can be seen that the trend of traffics observed are similar for both CATSchA algorithm and without contextual aware from 0 until it reaches full offered load.

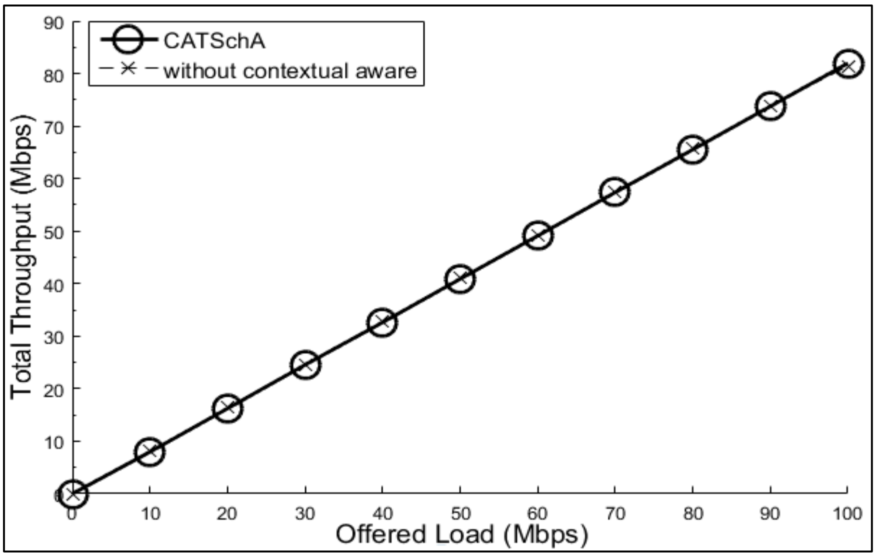

Figure 4.20 Throughput versus offered load for CATSchA and without contextual aware in Case 1

Figure 4.21 shows the total throughput result of Case 2 for both CATSchA algorithm and without contextual aware. These scenarios illustrate similar trend in traffic priority ranking between CATSchA algorithm and without contextual aware from 0 to 100 *Mbps* offered load.

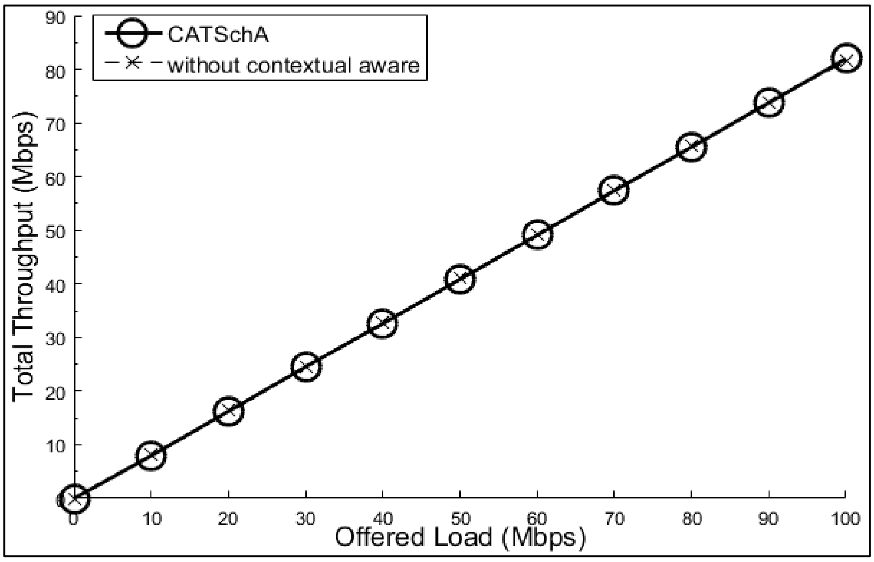

Figure 4.21 Throughput versus offered load for CATSchA and without contextual aware in Case 2

For Case 3 of total throughput result in Figure 4.22, it shows that SCADA, Smart Meter, CCTV and Internet with CATSchA algorithm have the same trend as compared to without contextual aware as the offered load increases from 0 to 100 *Mbps*.

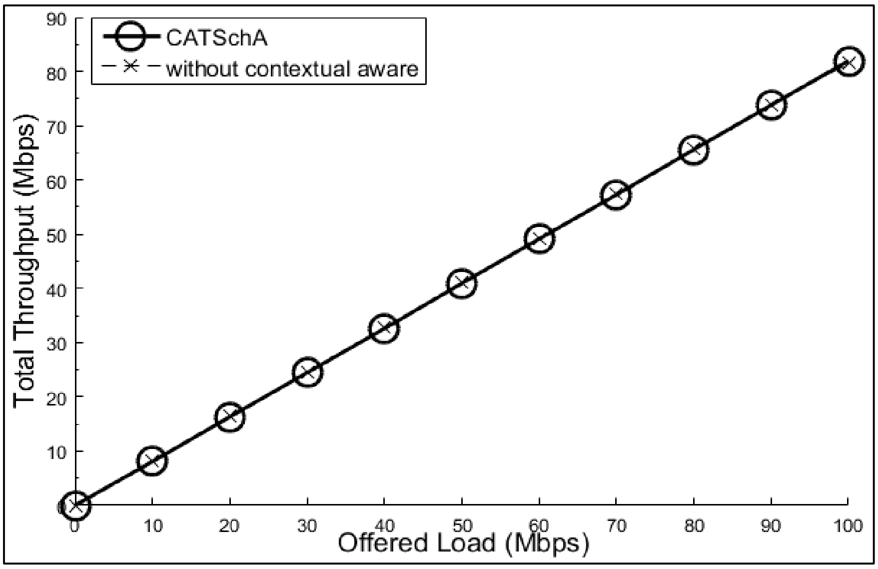

Figure 4.22 Throughput versus offered load for CATSchA and without contextual aware in Case 3

The trend for total throughput result of Case 4 in Figure 4.23 is such a way that SCADA, Smart Meter, CCTV and Internet with CATSchA algorithm is the same with the trend of traffics without the contextual aware until they reach full offered load.

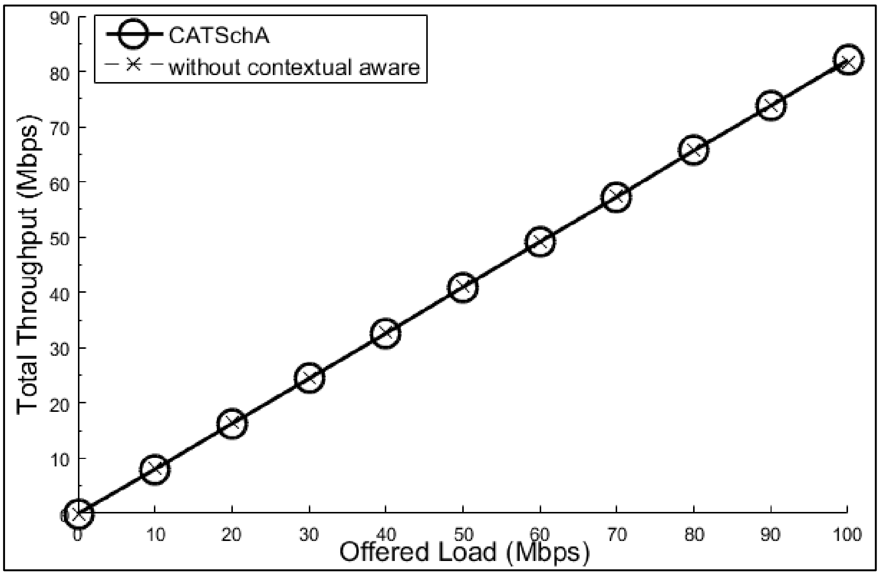

Figure 4.23 Throughput versus offered load for CATSchA and without contextual aware in Case 4

The result in terms of total throughput for Case 5 is shown in Figure 4.24, where the trend of four traffics observed below show that SCADA, Smart Meter, CCTV and Internet with CATSchA algorithm is similar with the traffics of without contextual aware from 0 to 100 *Mbps* offered load.

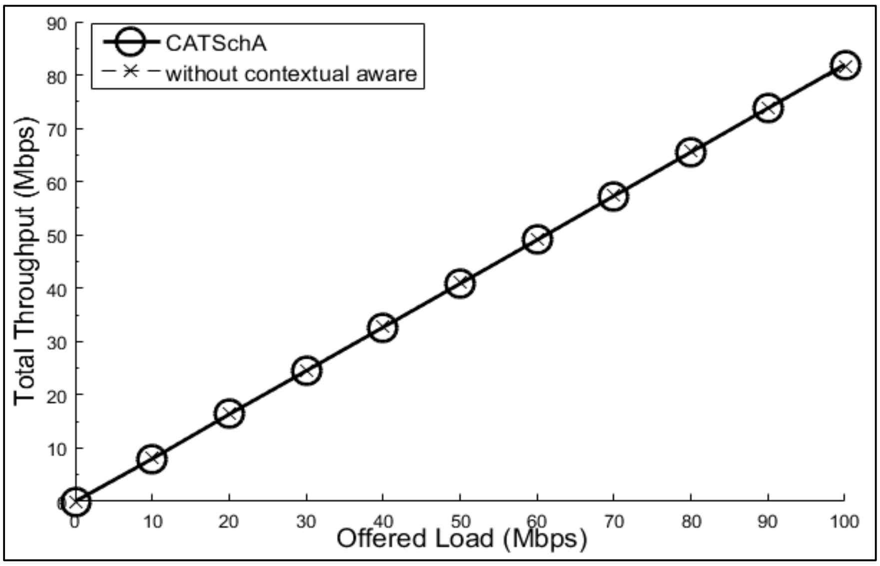

Figure 4.24 Throughput versus offered load for CATSchA and without contextual aware in Case 5

As shown in Figure 4.25, the total throughput of each traffic in case 6 are been categorized into CATSchA algorithm and without contextual aware. As can be observed, the trends between these two conditions are the same as the offered load increases from 0 to 100 *Mbps*.

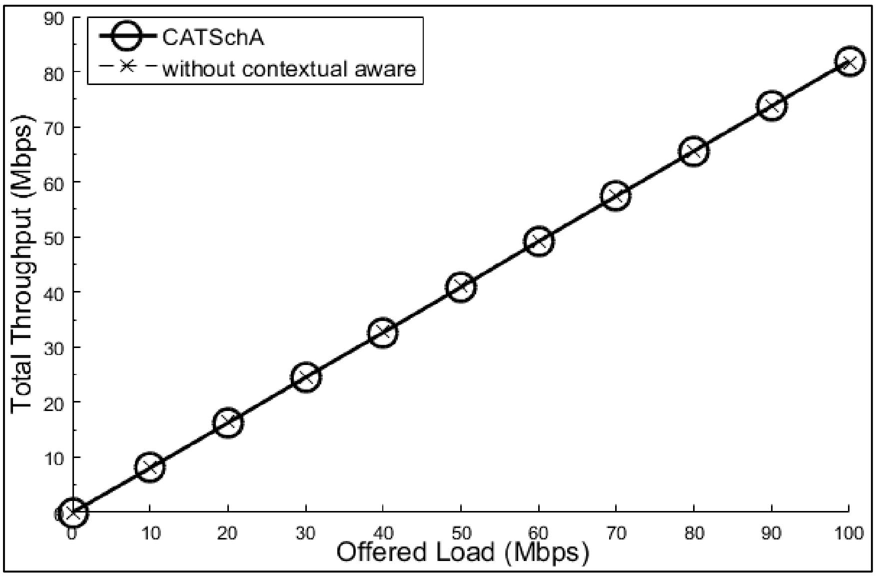

Figure 4.25 Throughput versus offered load for CATSchA and without contextual aware in Case 6

Throughput with CATSchA algorithm is the same with without contextual aware due to the amount of data transferred for each traffic between source and destination is similar. Furthermore, both of them are using the same design parameters that includes link capacity of 100 *Mbps*, cycle time of 2 *ms*, packet size of 1024 *Bytes* and 50 number of maximum packets in transmission queue. This observation is expected, hence conclude that CATSchA algorithm improves the delay and jitter upon without contextual aware but maintains the throughput.

4.4.4 Efficiency

The efficiency from Case 1 until Case 6 are shown in Figure 4.26, where four different traffics are taken into consideration during the simulation process at 100 *Mbps* offered load with CATSchA algorithm. It can be seen that the trend of six different cases observed below show that all traffics with CATSchA algorithm has more than 81.92% efficiency. This efficiency is sufficient as has been mentioned in [62, 94, 96, 97].

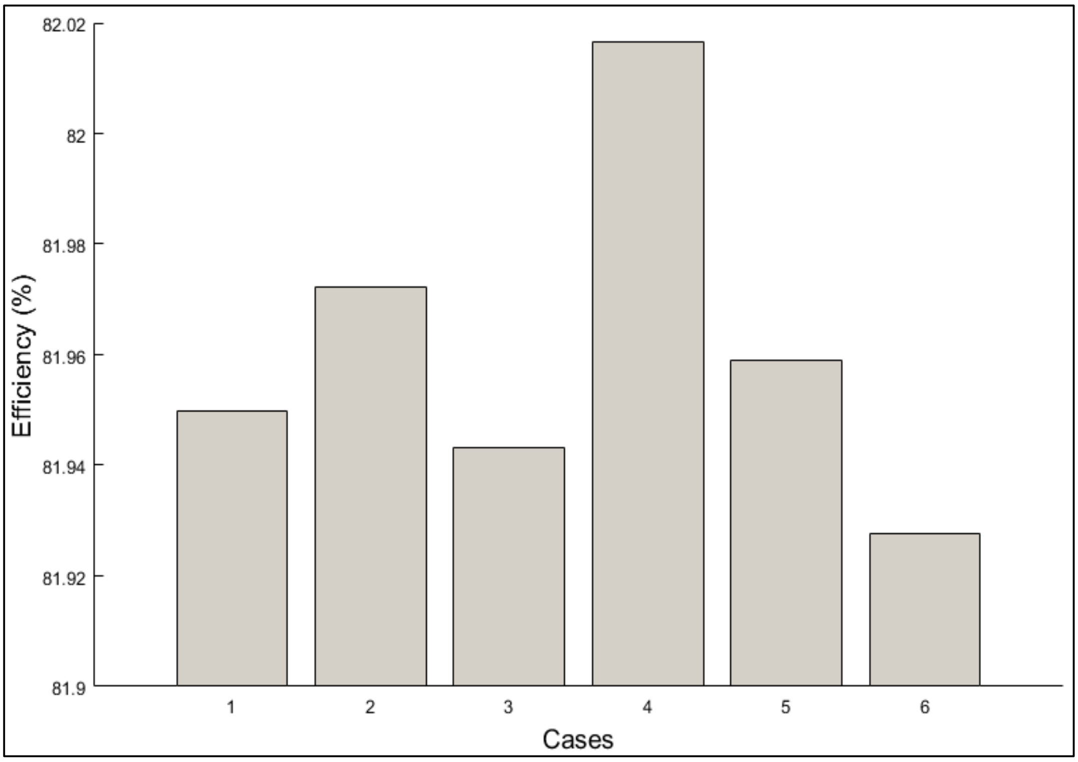

Figure 4.26 Efficiency for all cases

Case 4 has the highest efficiency of 82.02%, followed by Case 2 with 81.97% efficiency. Right next after Case 2 is Case 5 with 81.96% efficiency, Case 1 with 81.95% efficiency, Case 3 with 81.94% efficiency and lastly Case 6 with 81.93% efficiency. Efficiency is referred as the total amount of throughput at 100 *Mbps* offered load. Hence, the larger the amount of throughput for each traffic at full offered load for certain case, the higher the efficiency of particular case.

Overall, all traffics have improved their delay and jitter as well as maintained their throughput with CATSchA algorithm upon without contextual aware in which all of them have proved their performance in each case by having more than 75% [62, 97] efficiency. This is because each traffic in CATSchA algorithm have been queued according to their contextual aware priority whereby each one of them have their own allocated transmission time to reduce their delay and timing variations for packets arrival at minimal inconsistent rate.

4.5 Summary

In this chapter, the performance of CATSchA algorithm has been validated to prove that the simulation is following the computation correctly. The simulation results of CATSchA algorithm between NS-3 and MATLAB have been compared in order to validate the correctness of the program and the contextual aware ranking of SCADA, Smart Meter, CCTV and Internet.

The simulation results of CATSchA algorithm in terms of delay and efficiency are presented for six different cases using the same data rate and data size for all four traffics for proof of concept. SCADA has remained as the shortest delay for all cases since it will always have the highest rank and is transmitted first. The performance of SCADA in CATSchA algorithm is different from without contextual aware as SCADA with CATSchA algorithm has its own dedicated transmission slot. SCADA with CATSchA algorithm is guaranteed to be arrived with minimal delay at consistent rate without having to compete with other traffics in the network. Whereas, SCADA without contextual aware may arrive with higher delay as it has to compete with other traffics in the network. Hence, although SCADA is set to have the highest priority for both algorithms, the overall performance for SCADA with or without CATSchA algorithm still vary.

Other traffics are ranked differently for each cases according to their inequality parameters i.e. data size, data rate and previous transmission delay. The delay of each traffic is tally to their respective contextual aware ranking for all six cases. In the meantime, the efficiency for all cases are the same as predicted since the input design parameters which are data size and data rate remained the same for all traffics throughout the simulation process at 100 *Mbps* offered load. Hence, the concept of CATSchA algorithm is proven in this performance study.

CATSchA algorithm is then compared with traffic scheduling without contextual aware to show the advantages of the proposed algorithm. The performance of both scenarios are evaluated in terms of delay, jitter and throughput for six distinct cases by using actual traffic parameters. Overall, the delay results of CATSchA algorithm for all cases are less than delay results without contextual aware except for traffic with lower newly-defined ranking than its pre-defined ranking. Nonetheless, this is acceptable because average delay of CATSchA algorithm is still lower than without contextual aware by 68.49%.

Jitter for all traffics displayed an improvement via CATSchA algorithm by 44.87% upon without contextual aware because, each traffic have their own allocated transmission time in which all packets are able to arrive at minimal inconsistent rate compared to packets without contextual aware. SCADA with CATSchA algorithm always remain with zero jitter for all six cases due to its exclusivity in transmission time. As can be seen from Table 4.2, SCADA has its own allocated transmission time which does not involve other traffics and thus, all of its packets are guaranteed to be arrived at a consistent rate with no congestion in the network.

On the other hand, throughput for all cases with CATSchA algorithm are the same with throughput without contextual aware as the amount of data transferred for each traffic are the same for both algorithms.

CHAPTER 5

CONTRIBUTIONS, ISSUES AND FUTURE WORKS

5.1 Introduction

This chapter presents several achievements or contributions in Section 5.2 that has been gained from developing CATSchA algorithm for communication system in power distribution network based on specified objectives in Section 1.6. Some of the challenges or issues faced during the research phase are explained in Section 5.3. Section 5.4 highlights future works recommended to be done.

5.2 Achievements or Contributions

MPLS is seen as one of the best protocols to provide contextual aware based QoS traffic scheduling algorithm in power distribution division to fully utilize the capability of smart grid networks by reducing its delay and jitter. The priority of data needs to be dynamic according to its context in order to ensure the communication services are provisioned, managed and maintained accordingly due to increasing complexity in delivering electricity in smart grid network while ensuring reliability and timeliness of critical data.

Four traffics are considered for the simulation of CATSchA algorithm; namely SCADA, Smart Meter, CCTV and Internet. The traffics are characterized to different parameters in terms of data size, data rate and delay according to their specific requirements in power distribution network which produced a set of pre-defined priorities. Pre-defined priority ranking of traffics are given as follows; SCADA is at rank 1 (111), Smart Meter is at rank 2 (110), CCTV is at rank 7 (001) and Internet is at rank 8 (000). This reflects the first specific objective which is on characterizing and prioritizing the smart grid traffics according to their respective QoS.

The second objective of this thesis is to develop a contextual aware traffic scheduling algorithm for power distribution network in MPLS environment. This is achieved by rearranging the traffic priority ranking according to their criticality and sensitivity by firstly releasing the highest priority ranking traffic followed by the lower rank traffics accordingly. There are four levels in placement of the output traffic as follows; 1 represents the highest rank traffic, 2 denotes the second highest rank traffic, 3 indicates the second lowest rank traffic and 4 signifies the lowest rank traffic. Placement of output traffics will be varied depending on the input data size, data rate and delay. Input SCADA is not varied as it remains as the highest rank traffic throughout the validation process.

The proposed CATSchA algorithm is studied for six different cases namely Case 1 (priority from high to low: SCADA → Smart Meter → CCTV → Internet), Case 2 (priority from high to low: SCADA → Smart Meter → Internet → CCTV), Case 3 (priority from high to low: SCADA → CCTV → Smart Meter → Internet), Case 4 (priority from high to low: SCADA → CCTV → Internet → Smart Meter), Case 5 (priority from high to low: SCADA → Internet → Smart Meter → CCTV) and Case 6 (priority from high to low: SCADA → Internet → CCTV → Smart Meter).

The performance of CATSchA algorithm has been validated to prove that the simulation is following the computation correctly. The simulation results of CATSchA algorithm between NS-3 and MATLAB have been compared in order to validate the correctness of the program and the contextual aware ranking for all traffics. The simulation results of CATSchA algorithm in terms of delay and efficiency are presented for six different cases using the same data rate and data size for all four traffics for proof of concept. SCADA has remained as the shortest delay for all cases since it will always have the highest rank and is transmitted first. Other traffics are ranked differently for each cases according to their inequality parameters i.e. data size, data rate and previous transmission delay. Hence, the concept of CATSchA algorithm is proven in this thesis. CATSchA algorithm is then compared with traffic scheduling without contextual aware to show the advantages of the proposed algorithm. The performance of both scenarios are evaluated in terms of delay, jitter and throughput for six distinct cases by using actual traffic parameters. This answers the third specific objective which is evaluating the performance of CATSchA algorithm via simulation work.

Overall, the delay results of CATSchA algorithm for all cases are less than delay results without contextual aware except for traffic with lower newly-defined ranking than its pre-defined ranking which is acceptable due to average delay of CATSchA algorithm is still lower than without contextual aware by 68.49%. Jitter for all traffics displayed an improvement via CATSchA algorithm by 44.87% upon without contextual aware as each traffic have their own allocated transmission time in which all packets are able to arrive at minimal inconsistent rate compared to packets without contextual aware. Meanwhile, throughput for all cases with CATSchA algorithm are the same with throughput without contextual aware as the amount of data transferred for each traffic are the same for both algorithms. The trend of six different cases show that all traffics with CATSchA algorithm has more than 81.92% efficiency.

5.3 Challenges or Issues

There are different types of network traffic in smart grid and each of them have different characteristics along with priorities. Each traffics must be grouped or categorized according to their QoS in prior to develop the CATSchA algorithm.

An appropriate environment is necessary in developing the CATSchA algorithm for smart grid network. It is important in realizing the main objective which is to develop a contextual aware based QoS algorithm in packet network environment for power distribution network.

A suitable simulation platform is essential in evaluating the performance of CATSchA algorithm. This is to validate the correctness of the program especially on the contextual aware ranking and to prove the advantages of the algorithm.

5.4 Future Works and Recommendations

The development of CATSchA algorithm has been deliberated in this thesis. Some of the future works recommended is to design a testbed for the CATSchA algorithm to be able to run on it. Currently, CATSchA algorithm has only been validated via simulation platforms and thus, it is preferred to run the algorithm using hardware platform for further verification. In the future, real traffics are planned to be injected in the testbed in order to make the performance analysis to be as close to the real network as possible. Therefore, real traffics must be allowed to be used in the testbed associated with the CATSchA algorithm.

TNB is embarking into the MPLS-TP technology in managing their utility backbone network between substations and control center. The CATSchA algorithm can be fine-tuned to make it suitable to be used with MPLS-TP protocol. The basic concept of the algorithm is the same, only the parameters and the protocol need to be fine-tuned. This is useful in order to ensure that the QoS is supported in new technology while fully utilizing it.

Nowadays, wireless is gaining more popularity due to smart devices evolution. Hence, by combining the capacity of optical fiber network with the ubiquity and mobility of wireless network, it can be seen as a powerful platform for the support and creation of emerging as well as future unforeseen applications and services. Testbed enhancement to become a fiber-wireless network is seen in which base station needs to be added and the protocol of the algorithm needs to be changed accordingly.

REFERENCES

[1] X. Deng, L. He, X. Li, Q. Liu, L. Cai, and Z. Chen, "A reliable QoS-aware routing scheme for neighbor area network in smart grid," *Peer-to-Peer Networking and Applications*, vol. 9, pp. 616-627, 2016.

[2] M. Yigit, V. C. Gungor, E. Fadel, L. Nassef, N. Akkari, and I. F. Akyildiz, "Channel-aware routing and priority-aware multi-channel scheduling for WSN-based smart grid applications," *Journal of Network and Computer Applications*, vol. 71, pp. 50-58, 2016.

[3] N. S. Nafi, K. Ahmed, M. A. Gregory, and M. Datta, "A survey of smart grid architectures, applications, benefits and standardization," *Journal of Network and Computer Applications*, vol. 76, pp. 23-36, 2016.

[4] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Communication network requirements for major smart grid applications in HAN, NAN and WAN," *Computer Networks*, vol. 67, pp. 74-88, 2014.

[5] G. Wang, Y. Zhao, Y. Ying, J. Huang, and R. M. Winter, "Data Aggregation Point Placement Problem in Neighborhood Area Networks of Smart Grid," *Mobile Networks and Applications*, vol. 23, pp. 696-708, 2018.

[6] C. Samitier, *Utility Communication Networks and Services: Specification, Deployment and Operation*. Switzerland: Springer, 2016.

[7] B. B. Gupta and T. Akhtar, "A survey on smart power grid: frameworks, tools, security issues, and solutions," *Annals of Telecommunications*, vol. 72, pp. 517-549, 2017.

[8] M. Niedermeier and H. de Meer, "Constructing Dependable Smart Grid Networks using Network Functions Virtualization," *Journal of Network and Systems Management*, vol. 24, pp. 449-469, 2016.

[9] S. Ma, H. Zhang, and X. Xing, "Scalability for Smart Infrastructure System in Smart Grid: A Survey," *Wireless Personal Communications*, vol. 99, pp. 161-184, 2018.

[10] W. Li, M. Ferdowsi, M. Stevic, A. Monti, and F. Ponci, "Cosimulation for Smart Grid Communications," *IEEE Transactions on Industrial Informatics*, vol. 10, pp. 2374-2384, 2014.

[11] H. L. Vo, "An implementation of packet-switched communication for pilot protection at Tennessee Valley Authority," *Honors Theses,* Dept. Elect. Eng., Tennessee Univ., Chattanooga, Tennessee, 2018.

[12] K. V. Katsaros, W. K. Chai, N. Wang, G. Pavlou, H. Bontius, and M. Paolone, "Information-centric networking for machine-to-machine data delivery: a case study in smart grid applications," *IEEE Network,* vol. 28, pp. 58-64, 2014.

[13] S. Nixon and D. Dana, "Modeling Network Optimization by Optimize the Current Network by physical and logical architectures to improve the Quality of Services (QoS)," *International Journal of Engineering Science,* vol. 7, no. 8, pp. 14715-14717, 2017.

[14] A. Dumka, H. L. Mandoria, V. Fore, and K. Dumka, "Implementation of QoS algorithm in integrated services (IntServ) MPLS network," in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2015, pp. 1048-1050.

[15] A. Mushtaq and M. S. Patterh, "QOS parameter comparison of DiffServ-aware MPLS network using IPv4 and IPv6," in *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*, 2017, pp. 113-118.

[16] L. Gelazanskas and K. A. A. Gamage, "Demand side management in smart grid: A review and proposals for future direction," *Sustainable Cities and Society*, vol. 11, pp. 22-30, 2014.

[17] A. Usman and S. H. Shami, "Evolution of Communication Technologies for Smart Grid applications," *Renewable and Sustainable Energy Reviews,* vol. 19, pp. 191-199, 2013.

[18] X. Meng and Z. Pian, "Chapter 1 - Introduction," in *Intelligent Coordinated Control of Complex Uncertain Systems for Power Distribution Network Reliability*, X. Meng and Z. Pian, Eds., ed Oxford: Elsevier, 2016, pp. 1-13.

[19] A. Abdrabou, "A Wireless Communication Architecture for Smart Grid Distribution Networks," *IEEE Systems Journal,* vol. 10, pp. 251-261, 2016.

[20] I. Ivanković, A. Kekelj, R. Rubeša, and I. Kuzle, "SCADA Maintenance and Refurbishment with Security Issue in Modern IT and OT Environment," in *11th Mediterranean Conference on Power Generation, Transmission, Distribution and Energy Conversion*, Cavtat, Hrvatska, Nov. 12-15, 2018.

[21] N. Agarwal and A. Brem, "Strategic business transformation through technology convergence: implications from General Electric's industrial internet initiative," *International Journal of Technology Management,* vol. 67, pp. 196-214, 2015.

[22] A. Hahn, "Operational technology and information technology in industrial control systems," in *Cyber-security of SCADA and Other Industrial Control Systems*, ed: Springer, 2016, pp. 51-68.

[23] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A. Sadeghi, M. Maniatakos, *et al.*, "The Cybersecurity Landscape in Industrial Control Systems," *Proceedings of the IEEE,* vol. 104, pp. 1039-1057, 2016.

[24] A. K. M. Baki, "Continuous Monitoring of Smart Grid Devices Through Multi Protocol Label Switching," *IEEE Transactions on Smart Grid,* vol. 5, pp. 1210-1215, 2014.

[25] M. Madrigal, R. Uluski, and K. Mensan Gaba, *Practical Guidance for Defining a Smart Grid Modernization Strategy: The Case of Distribution (Revised Edition)*. Washington, USA: The World Bank, 2017.

[26] A. Mahmood, N. Javaid, and S. Razzaq, "A review of wireless communications for smart grid," *Renewable and sustainable energy reviews,* vol. 41, pp. 248-260, 2015.

[27] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A Survey on Smart Grid Communication Infrastructures: Motivations, Requirements and Challenges," *IEEE Communications Surveys & Tutorials,* vol. 15, pp. 5-20, 2013.

[28] K. Mets, J. A. Ojea, and C. Develder, "Combining Power and Communication Network Simulation for Cost-Effective Smart Grid Analysis," *IEEE Communications Surveys & Tutorials,* vol. 16, pp. 1771-1796, 2014.

[29] I. T. U. (ITU-T), "G.703: Physical/electrical characteristics of hierarchical digital interfaces," 2016.

[30] H. D. Kim, I. Kim, H. S. Lee, J. B. Yoon, H. K. Shin, and S. W. Ryu, "Low-Cost CWDM Transceiver Module by Spatial Power Combination and Division for Multimode Fiber Links," *IEEE Photonics Technology Letters,* vol. 29, pp. 39-42, 2017.

[31] P. Pan, J. An, Y. Wang, J. Zhang, L. Wang, Y. Qi, *et al.*, "Compact 4-channel AWGs for CWDM and LAN WDM in data center monolithic applications," *Optics & Laser Technology,* vol. 75, pp. 177-181, 2015.

[32] M. Nooruzzaman, H. Elbiaze, R. Zahilah, O. Koyama, and M. Yamada, "Design and performance evaluation of amplifier modules in stackable ROADMs for low-cost CWDM access networks," *Photonic Network Communications,* vol. 28, pp. 237-250, 2014.

[33] H. Xu and Y. Shi, "Flat-Top CWDM (De)Multiplexer Based on MZI With Bent Directional Couplers," *IEEE Photonics Technology Letters,* vol. 30, pp. 169-172, 2018.

[34] M. Ekici, Q. Zhou, A. Sonmezoglu, S. P. Moshokoa, M. Z. Ullah, A. Biswas, *et al.*, "Optical solitons with DWDM technology and four-wave mixing," *Superlattices and Microstructures,* vol. 107, pp. 254-266, 2017.

[35] A. E. P. Rondelli and M. Galimberti, "Improving MV grid control, remote operations and reliability through advanced TLC network and SCADA architecture," *CIRED - Open Access Proceedings Journal,* vol. 2017, pp. 1219-1222, 2017.

[36] A. Biswas, M. Ekici, A. Sonmezoglu, Q. Zhou, A. S. Alshomrani, S. P. Moshokoa, *et al.*, "Optical network topology with DWDM technology for log law medium," *Optik,* vol. 160, pp. 353-360, 2018.

[37] A. Goel and G. Pandey, "Design of broadband dispersion flattened fiber for DWDM system: Performance analysis using various modulation formats," *Optical Fiber Technology,* vol. 42, pp. 109-118, 2018.

[38] I. Aldaya, C. Del-Valle-Soto, G. Campuzano, E. Giacoumidis, R. González, and G. Castañón, "Photonic millimeter-wave bridge for multi-Gbps passive optical networks," *Physical Communication,* vol. 28, pp. 138-146, 2018.

[39] M. De Andrade, M. Maier, M. P. McGarry, and M. Reisslein, "Passive optical network (PON) supported networking," *Optical Switching and Networking,* vol. 14, pp. 1-10, 2014.

[40] Y. Ye, T. Jiménez, and V. López, "Spectral, cost and energy efficiencies analysis in WDM MLR networks with OTN switching," *Photonic Network Communications,* vol. 34, pp. 422-431, 2017.

[41] S. S. Gorshe, "OTN Interface Standards for Rates Beyond 100 Gb/s," *Journal of Lightwave Technology,* vol. 36, pp. 19-26, 2018.

[42] C. S. K. Vadrevu, M. Tornatore, C. P. Guok, I. Monga, and B. Mukherjee, "Vertical and horizontal circuit/packet integration techniques for the future optical internet," *IEEE Network,* vol. 27, pp. 52-58, 2013.

[43] G. M. Saridis, S. Peng, Y. Yan, A. Aguado, B. Guo, M. Arslan, *et al.*, "Lightness: A function-virtualizable software defined data center network with all-optical circuit/packet switching," *Journal of Lightwave Technology,* vol. 34, pp. 1618-1627, 2016.

[44] A. Hajjawi and M. Ismail, "A scheduling algorithm based self-learning technique for smart grid communications over 4G networks," *Journal of Communications,* vol. 10, pp. 876-881, 2015.

[45] C. Qu, W. Chen, J. B. Song, and H. Li, "Distributed Data Traffic Scheduling With Awareness of Dynamics State in Cyber Physical Systems With Application in Smart Grid," *IEEE Transactions on Smart Grid,* vol. 6, pp. 2895-2905, 2015.

[46] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid, "A distributed and robust SDN control plane for transactional network updates," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 190-198.

[47] C. Cascone, L. Pollini, D. Sanvito, and A. Capone, "Traffic Management Applications for Stateful SDN Data Plane," in *2015 Fourth European Workshop on Software Defined Networks*, 2015, pp. 85-90.

[48] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Network*, vol. 30, pp. 52-58, 2016.

[49] A. Khorsandi Koohanestani, A. Ghalami Osgouei, H. Saidi, and A. Fanian, "An analytical model for delay bound of OpenFlow based SDN using network calculus," *Journal of Network and Computer Applications,* vol. 96, pp. 31-38, 2017.

[50] Z. Jianchao, S. Boon-Chong, L. Tek-Tjing, and F. Chuan Heng, "Opportunities for Software-Defined Networking in Smart Grid," in *2013 9th International Conference on Information, Communications & Signal Processing*, 2013, pp. 1-5.

[51] M. H. Yaghmaee, Z. Yousefi, M. Zabihi, and S. Alishahi, "Quality of service guarantee in smart grid infrastructure communication using traffic classification," in *22nd International Conference and Exhibition on Electricity Distribution (CIRED)*, 2013, vol. 2013, pp. 803-806.

[52] N. Dorsch, F. Kurtz, and C. Wietfeld, "Communications in distributed smart grid control: Software-defined vs. legacy networks," in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, 2017, pp. 1-6.

[53] M. Popovic, "Redundancy in Communication Networks for Smart Grids," EPFL2016.

[54] I. Al-Anbagi, M. Erol-Kantarci, and H. T. Mouftah, "Delay critical smart grid applications and adaptive QoS provisioning," *IEEE Access*, vol. 3, pp. 1367-1378, 2015.

[55] G. Rajalingham, Y. Gao, Q.-D. Ho, and T. Le-Ngoc, "Quality of service differentiation for smart grid neighbor area networks through multiple RPL instances," in *Proceedings of the 10th ACM symposium on QoS and security for wireless and mobile networks*, 2014, pp. 17-24.

[56] D. Sahin, V. C. Gungor, T. Kocak, and G. Tuna, "Quality-of-service differentiation in single-path and multi-path routing for wireless sensor network-based smart grid applications," *Ad Hoc Networks*, vol. 22, pp. 43-60, 2014.

[57] M. N. Moghadam, H. Taheri, and M. Karrari, "Multi-class Multipath Routing Protocol for Low Power Wireless Networks with Heuristic Optimal Load Distribution," *Wireless Personal Communications*, vol. 82, pp. 861-881, 2015.

[58] F. J. Rodríguez-Pérez, J. L. González-Sánchez, J. Carmona-Murillo, and D. Cortés-Polo, "An OAM function to improve the packet loss in MPLS-TP domains for prioritized QoS-aware services," *International Journal of Communication Systems*, vol. 28, pp. 1037-1052, 2015.

[59] M. M. Pijanka and G. W. Różański, "Mobile MPLS-TP–Support the mobility of terminal devices using OAM channel," *International Journal of Electronics and Telecommunications*, vol. 62, pp. 215-223, 2016.

[60] X. Deng, Q. Peng, L. He, and T. He, "Interference-aware QoS routing for neighbourhood area network in smart grid," *IET Communications*, vol. 11, pp. 756-764, 2016.

[61] X. Deng, L. He, C. Zhu, M. Dong, K. Ota, and L. Cai, "QoS-aware and load-balance routing for IEEE 802.11 s based neighborhood area network in smart grid," *Wireless Personal Communications*, vol. 89, pp. 1065-1088, 2016.

[62] K. Demir, D. Germanus, and N. Suri, "Robust QoS-aware communication in the smart distribution grid," *Peer-to-Peer Networking and Applications,* vol. 10, pp. 193-207, 2017.

[63] H.-S. Kim, H. Kim, J. Paek, and S. Bahk, "Load balancing under heavy traffic in RPL routing protocol for low power and lossy networks," *IEEE Transactions on Mobile Computing,* vol. 16, pp. 964-979, 2017.

[64] I. S. Bayram, G. Michailidis, and M. Devetsikiotis, "Unsplittable load balancing in a network of charging stations under QoS guarantees," *IEEE Transactions on Smart Grid,* vol. 6, pp. 1292-1302, 2015.

[65] D.-U. Kim, Y.-C. Ryoo, B. Kim, and J. Lee, "OAM and protection mechanisms for MPLS-TP packet transport networks," in *Optical Internet 2014 (COIN), 2014 12th International Conference on,* 2014, pp. 1-4.

[66] J.-d. Ryoo, T. Cheung, D. King, A. Farrel, and H. van Helvoort, "MPLS-TP linear protection for ITU-T and IETF," *IEEE Communications Magazine,* vol. 52, pp. 16-21, 2014.

[67] D. Cortés-Polo, J.-L. González-Sánchez, J. Carmona-Murillo, and F. J. Rodríguez-Pérez, "Proposal and analysis of integrated PTN architecture in the mobile backhaul to improve the QoS of HetNets," *EURASIP Journal on Wireless Communications and Networking,* vol. 2015, no. 1, pp. 116-128, 2015.

[68] D. U. Kim, J. d. Ryoo, J. H. Lee, B. C. Kim, and J. Y. Lee, "Protection Switching Methods for Point-to-Multipoint Connections in Packet Transport Networks," *Etri Journal,* vol. 38, pp. 18-29, 2016.

[69] J. S. Choi, "Design and implementation of a stateful PCE-based unified control and management framework for carrier-grade MPLS-TP networks," *Journal of Lightwave Technology,* vol. 34, pp. 836-844, 2016.

[70] P. Qin, B. Dai, B. Huang, and G. Xu, "Bandwidth-Aware Scheduling With SDN in Hadoop: A New Trend for Big Data," *IEEE Systems Journal,* vol. 11, pp. 2337-2344, 2017.

[71] J. Pang, G. Xu, and X. Fu, "SDN-Based Data Center Networking With Collaboration of Multipath TCP and Segment Routing," *IEEE Access,* vol. 5, pp. 9764-9773, 2017.

[72] M. Alishahi, M. H. Yaghmaee Moghaddam, and H. R. Pourreza, "Multi-class routing protocol using virtualization and SDN-enabled architecture for smart grid," *Peer-to-Peer Networking and Applications,* vol. 11, pp. 380-396, 2018.

[73] M. Alhowaidi, P. Yi, and B. Ramamurthy, "Dynamic provisioning in virtualized Cloud infrastructure in IP/MPLS-over-WDM networks," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 83-87.

[74] S. Liu, Q. Wang, H. Wu, Y. Fan, H. Peng, G. Zhang*, et al.*, "Traffic scheduling with sustainable Cyber Physical Systems applying in smart grid," in *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*, 2016, pp. 1-6.

[75] F. Kamoun and F. Outay, "IP/MPLS networks with hardened pipes: service concepts, traffic engineering and design considerations," *Journal of Ambient Intelligence and Humanized Computing,* vol. 10, no. 7, pp. 2577-2584, 2018.

[76] M. Rezaee and M. H. Y. Moghaddam, "SDN-based Quality of Service Networking for Wide Area Measurement System," *IEEE Transactions on Industrial Informatics,* pp. 1-11, 2019.

[77] G. Yunfei, W. Kun, L. Shidong, G. Jinghong, and L. Heng, "A Context-aware Data Processing Model in power communication networks," in *2015 IEEE/CIC International Conference on Communications in China - Workshops (CIC/ICCC)*, 2015, pp. 180-183.

[78] M. Elias, A. Khattab, and K. M. F. Elsayed, "CORB: Context-aware opportunistic resource-based routing for stationary wireless sensor networks," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, 2015, pp. 166-170.

[79] L. B. Bhajantri and S. S. Hiremath, "Cluster based context aware secure routing in distributed sensor networks," in *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, 2017, pp. 378-384.

[80] M. S. Obaidat, N. V. R. Reddy, K. P. Venkata, and V. Saritha, "Context-Aware Middleware Architectural Framework for Intelligent Smart Grid Data Management," in *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2018, pp. 1-5.

[81] C. Choi, C. Esposito, H. Wang, Z. Liu, and J. Choi, "Intelligent Power Equipment Management Based on Distributed Context-Aware Inference in Smart Cities," *IEEE Communications Magazine,* vol. 56, pp. 212-217, 2018.

[82] N. Enose, "Implementing an integrated security management framework to ensure a secure smart grid," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2014, pp. 778-784.

[83] "IEEE Standard Communication Delivery Time Performance Requirements for Electric Power Substation Automation," *IEEE Std 1646-2004,* pp. 1-24, 2005.

[84] N. Uribe-Pérez, I. Angulo, D. de la Vega, T. Arzuaga, I. Fernández, and A. Arrinda, "Smart Grid Applications for a Practical Implementation of IP over Narrowband Power Line Communications," *Energies,* vol. 10, no. 11, pp. 1782-1797, 2017.

[85] F. Ye, Y. Qian, and R. Q. Hu, "Energy Efficient Self-Sustaining Wireless Neighborhood Area Network Design for Smart Grid," *IEEE Transactions on Smart Grid,* vol. 6, pp. 220-229, 2015.

[86] C. Lo and N. Ansari, "Decentralized Controls and Communications for Autonomous Distribution Networks in Smart Grid," *IEEE Transactions on Smart Grid,* vol. 4, pp. 66-77, 2013.

[87] N. Nikmehr and S. N. Ravadanegh, "Optimal Power Dispatch of Multi-Microgrids at Future Smart Distribution Grids," *IEEE Transactions on Smart Grid,* vol. 6, pp. 1648-1657, 2015.

[88] G. Tuna, V. C. Gungor, and K. Gulez, "Wireless sensor networks for smart grid applications: a case study on link reliability and node lifetime evaluations in power distribution systems," *International Journal of Distributed Sensor Networks,* vol. 9, pp. 796248-796258, 2013.

[89] D. Pudjianto, P. Djapic, M. Aunedi, C. K. Gan, G. Strbac, S. Huang, *et al.*, "Smart control for minimizing distribution network reinforcement cost due to electrification," *Energy Policy,* vol. 52, pp. 76-84, 2013.

[90] Nsnam. (2013). *Network Simulator (NS-3) release 3.18*. Available: https://www.nsnam.org/news/ns-3-18-release/

[91] S. Yoon, S. Jang, Y. Kim, and S. Bahk, "Opportunistic Routing for Smart Grid With Power Line Communication Access Networks," *IEEE Transactions on Smart Grid,* vol. 5, pp. 303-311, 2014.

[92] C. Shum, W. Lau, T. Mao, H. S. Chung, K. Tsang, N. C. Tse, *et al.*, "Co-Simulation of Distributed Smart Grid Software Using Direct-Execution Simulation," *IEEE Access*, vol. 6, pp. 20531-20544, 2018.

[93] D. Bhor, K. Angappan, and K. M. Sivalingam, "Network and power-grid co-simulation framework for Smart Grid wide-area monitoring networks," *Journal of Network and Computer Applications*, vol. 59, pp. 274-284, 2016.

[94] B. Afzal, S. A. Alvi, G. A. Shah, and W. Mahmood, "Energy efficient context aware traffic scheduling for IoT applications," *Ad Hoc Networks*, vol. 62, pp. 101-115, 2017.

[95] G. A. Shah, V. C. Gungor, and O. B. Akan, "A Cross-Layer QoS-Aware Communication Framework in Cognitive Radio Sensor Networks for Smart Grid Applications," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 1477-1485, 2013.

[96] B. P. Rimal, D. P. Van, and M. Maier, "Mobile Edge Computing Empowered Fiber-Wireless Access Networks in the 5G Era," *IEEE Communications Magazine*, vol. 55, pp. 192-200, 2017.

[97] H. Daki, A. El Hannani, A. Aqqal, A. Haidine, and A. Dahbi, "Big Data management in smart grid: concepts, requirements and implementation," *Journal of Big Data*, vol. 4, pp. 13-31, 2017.

APPENDIX A

Appendix A : Validation results between NS-3 and MATLAB

		Input Traffic				**Output Traffic**							
						NS-3				**MATLAB**			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
1	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	0 0 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
2	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 0	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
3	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 1	0 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
4	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 1	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
5	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
6	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	0 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
7	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
8	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 0	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
9	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 1	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
10	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 1	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
11	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
12	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
13	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV

		Input Traffic				**Output Traffic**							
						NS-3				**MATLAB**			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
14	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
15	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 0	0 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
16	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 0	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
17	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 1	1 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
18	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 1	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
19	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 1	1 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
20	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 1	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
21	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 1	0 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
22	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 1	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
23	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
24	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 1	0 0 0	SCADA	Smart Meter	CCTV	Internet	SCADA	Smart Meter	CCTV	Internet
25	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
26	Data Size Data Rate Delay	Any Any Any	0 0 1	1 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
27	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
28	Data Size Data Rate Delay	Any Any Any	0 0 1	1 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
29	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
30	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 1	0 0 0	SCADA	Smart Meter	CCTV	Internet	SCADA	Smart Meter	CCTV	Internet
31	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
32	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 0	1 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
33	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 0	1 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
34	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 0	1 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
35	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 0	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
36	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 0	0 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
37	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 0	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
38	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 0	1 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
39	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 0	1 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
40	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 0	1 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
41	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 1	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				**Output Traffic**							
						NS-3				**MATLAB**			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
42	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 1	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
43	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 0	0 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
44	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 1	0 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
45	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 0	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
46	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 1	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
47	Data Size Data Rate Delay	Any Any Any	0 0 1	1 1 0	0 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
48	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 0	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
49	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 1	1 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
50	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 1	1 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
51	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 1	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
52	Data Size Data Rate Delay	Any Any Any	0 0 1	1 1 1	0 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
53	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 1	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
54	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 1	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
55	Data Size Data Rate Delay	Any Any Any	0 0 1	1 1 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

		Input Traffic				**Output Traffic**							
						NS-3				**MATLAB**			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
56	Data Size Data Rate Delay	Any Any Any	0 0 1	1 1 1	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
57	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 0	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
58	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 1	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
59	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 0	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
60	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 1	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
61	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 0	1 0 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
62	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 1	1 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				**Output Traffic**							
						NS-3				**MATLAB**			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
63	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
64	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
65	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	1 0 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
66	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 0	1 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
67	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 0	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
68	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 1	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
69	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 0	0 0 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
70	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 1	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
71	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
72	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
73	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 0	0 0 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
74	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
75	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 1	0 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
76	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 1	0 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
77	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 1	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
78	Data Size Data Rate Delay	Any Any Any	0 0 0	0 0 1	1 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
79	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 0	0 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
80	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 0	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
81	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 0	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
82	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 0	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
83	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 0	1 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
84	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
85	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 0	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
86	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 1	0 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
87	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 1	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
88	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 1	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
89	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 1	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
90	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 1	1 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
91	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 1	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
92	Data Size Data Rate Delay	Any Any Any	0 0 0	0 1 1	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
93	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 0	0 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
94	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 0	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
95	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 0	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
96	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 0	1 0 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
97	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 0	1 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
98	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
99	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 0	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
100	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 1	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
101	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 1	0 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
102	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 1	0 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
103	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 1	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
104	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 1	1 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
105	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 1	1 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
106	Data Size Data Rate Delay	Any Any Any	0 0 0	1 0 1	1 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
107	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 0	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
108	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 0	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
109	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 0	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
110	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 0	1 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
111	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 1	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
112	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 1	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
113	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 1	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
114	Data Size Data Rate Delay	Any Any Any	0 0 0	1 1 1	1 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
115	Data Size Data Rate Delay	Any Any Any	0 0 1	0 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
116	Data Size Data Rate Delay	Any Any Any	0 0 1	0 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
117	Data Size Data Rate Delay	Any Any Any	0 0 1	1 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
118	Data Size Data Rate Delay	Any Any Any	0 0 1	1 0 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
119	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 1	0 0 0	SCADA	Smart Meter	CCTV	Internet	SCADA	Smart Meter	CCTV	Internet
120	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
121	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
122	Data Size Data Rate Delay	Any Any Any	0 1 0	1 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
123	Data Size Data Rate Delay	Any Any Any	0 1 0	1 0 1	0 0 0	SCADA	Smart Meter	CCTV	Internet	SCADA	Smart Meter	CCTV	Internet
124	Data Size Data Rate Delay	Any Any Any	0 1 0	1 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
125	Data Size Data Rate Delay	Any Any Any	0 1 0	1 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
126	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 1	0 0 0	SCADA	Smart Meter	CCTV	Internet	SCADA	Smart Meter	CCTV	Internet
127	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
128	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
129	Data Size Data Rate Delay	Any Any Any	0 1 1	1 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
130	Data Size Data Rate Delay	Any Any Any	0 1 1	1 0 1	0 0 0	SCADA	Smart Meter	CCTV	Internet	SCADA	Smart Meter	CCTV	Internet
131	Data Size Data Rate Delay	Any Any Any	0 1 1	1 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
132	Data Size Data Rate Delay	Any Any Any	0 1 1	1 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
133	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 1	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
134	Data Size Data Rate Delay	Any Any Any	1 0 0	0 1 0	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
135	Data Size Data Rate Delay	Any Any Any	1 0 0	0 1 1	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
136	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 0	0 0 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
137	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 1	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
138	Data Size Data Rate Delay	Any Any Any	1 0 0	1 1 0	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
139	Data Size Data Rate Delay	Any Any Any	1 0 0	1 1 1	0 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
140	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
141	Data Size Data Rate Delay	Any Any Any	1 0 1	0 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
142	Data Size Data Rate Delay	Any Any Any	1 0 1	0 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
143	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
144	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
145	Data Size Data Rate Delay	Any Any Any	1 0 1	1 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
146	Data Size Data Rate Delay	Any Any Any	1 0 1	1 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
147	Data Size Data Rate Delay	Any Any Any	1 1 0	0 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
148	Data Size Data Rate Delay	Any Any Any	1 1 0	0 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
149	Data Size Data Rate Delay	Any Any Any	1 1 0	1 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
150	Data Size Data Rate Delay	Any Any Any	1 1 0	1 0 1	0 0 0	SCADA	Smart Meter	CCTV	Internet	SCADA	Smart Meter	CCTV	Internet
151	Data Size Data Rate Delay	Any Any Any	1 1 1	0 1 0	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
152	Data Size Data Rate Delay	Any Any Any	1 1 1	0 1 1	0 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
153	Data Size Data Rate Delay	Any Any Any	1 1 1	1 0 0	0 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
154	Data Size Data Rate Delay	Any Any Any	1 1 1	1 0 1	0 0 0	SCADA	Smart Meter	CCTV	Internet	SCADA	Smart Meter	CCTV	Internet
155	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 0	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
156	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 0	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
157	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 0	0 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
158	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 0	0 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
159	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 0	0 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
160	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 0	0 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
161	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 0	0 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
162	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 0	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
163	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 0	0 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
164	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 0	0 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
165	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 0	0 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
166	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 0	0 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
167	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 0	0 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
168	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 0	0 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
169	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 0	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
170	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 0	0 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
171	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 0	0 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
172	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 0	0 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
173	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
174	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
175	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
176	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 0	1 0 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
177	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
178	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
179	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
180	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 0	1 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
181	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 0	1 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
182	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 0	1 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
183	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 0	1 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
184	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 0	1 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
185	Data Size Data Rate Delay	Any Any Any	1 1 0	0 0 0	1 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
186	Data Size Data Rate Delay	Any Any Any	1 1 1	0 0 0	1 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
187	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 0	1 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
188	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 0	1 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
189	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
190	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 0	1 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
191	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 0	1 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
192	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 0	1 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
193	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 0	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
194	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 0	1 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
195	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 0	0 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
196	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 1	0 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
197	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 0	0 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
198	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 1	0 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
199	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 1	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
200	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 1	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
201	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 0	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
202	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 1	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
203	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 0	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
204	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 1	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
205	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 1	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
206	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 0	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
207	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 0	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
208	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 1	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
209	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 0	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
210	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 1	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
211	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 1	1 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
212	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 0	1 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
213	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 1	1 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
214	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 1	1 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
215	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 0	1 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
216	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 1	1 0 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				**Output Traffic**							
						NS-3				**MATLAB**			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
217	Data Size Data Rate Delay	Any Any Any	0 0 1	0 0 1	1 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
218	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 0	1 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
219	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 1	1 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
220	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 0	1 0 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
221	Data Size Data Rate Delay	Any Any Any	1 1 0	1 1 0	1 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
222	Data Size Data Rate Delay	Any Any Any	1 1 1	1 1 1	1 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
223	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 0	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1 SCADA	2 Smart Meter	3 CCTV	4 Internet	1	2	3	4	1	2	3	4
224	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 1	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
225	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
226	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 1	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
227	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 0	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
228	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 1	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
229	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 0	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
230	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 1	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
231	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 1	0 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
232	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 1	0 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
233	Data Size Data Rate Delay	Any Any Any	1 0 0	0 0 1	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
234	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 1	1 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
235	Data Size Data Rate Delay	Any Any Any	0 0 1	0 1 0	0 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
236	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 0	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
237	Data Size Data Rate Delay	Any Any Any	1 0 0	0 1 0	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
238	Data Size Data Rate Delay	Any Any Any	1 0 1	0 1 0	1 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
239	Data Size Data Rate Delay	Any Any Any	1 1 0	0 1 0	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
240	Data Size Data Rate Delay	Any Any Any	1 1 1	0 1 0	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
241	Data Size Data Rate Delay	Any Any Any	0 0 1	0 1 1	0 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
242	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 1	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
243	Data Size Data Rate Delay	Any Any Any	1 0 0	0 1 1	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
244	Data Size Data Rate Delay	Any Any Any	1 0 1	0 1 1	1 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
245	Data Size Data Rate Delay	Any Any Any	1 1 0	0 1 1	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
246	Data Size Data Rate Delay	Any Any Any	1 1 1	0 1 1	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
247	Data Size Data Rate Delay	Any Any Any	0 0 1	1 0 0	0 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
248	Data Size Data Rate Delay	Any Any Any	0 1 0	1 0 0	0 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
249	Data Size Data Rate Delay	Any Any Any	0 1 1	1 0 0	0 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
250	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 0	1 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
251	Data Size Data Rate Delay	Any Any Any	1 1 0	1 0 0	1 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
252	Data Size Data Rate Delay	Any Any Any	1 1 1	1 0 0	1 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
253	Data Size Data Rate Delay	Any Any Any	0 0 1	1 0 1	0 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
254	Data Size Data Rate Delay	Any Any Any	0 1 0	1 0 1	0 1 0	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
255	Data Size Data Rate Delay	Any Any Any	0 1 1	1 0 1	0 1 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
256	Data Size Data Rate Delay	Any Any Any	1 0 0	1 0 1	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
257	Data Size Data Rate Delay	Any Any Any	1 1 0	1 0 1	1 1 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
258	Data Size Data Rate Delay	Any Any Any	1 1 1	1 0 1	1 1 1	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
259	Data Size Data Rate Delay	Any Any Any	0 1 0	1 1 0	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
260	Data Size Data Rate Delay	Any Any Any	0 1 1	1 1 0	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
261	Data Size Data Rate Delay	Any Any Any	1 0 0	1 1 0	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter
262	Data Size Data Rate Delay	Any Any Any	1 0 1	1 1 0	1 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
263	Data Size Data Rate Delay	Any Any Any	0 1 0	1 1 1	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
264	Data Size Data Rate Delay	Any Any Any	0 1 1	1 1 1	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
265	Data Size Data Rate Delay	Any Any Any	1 0 0	1 1 1	1 0 0	SCADA	Internet	CCTV	Smart Meter	SCADA	Internet	CCTV	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
266	Data Size Data Rate Delay	Any Any Any	1 0 1	1 1 1	1 0 1	SCADA	Internet	Smart Meter	CCTV	SCADA	Internet	Smart Meter	CCTV
267	Data Size Data Rate Delay	Any Any Any	0 0 1	0 1 0	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
268	Data Size Data Rate Delay	Any Any Any	0 0 1	0 1 1	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
269	Data Size Data Rate Delay	Any Any Any	0 0 1	1 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
270	Data Size Data Rate Delay	Any Any Any	0 0 1	1 0 1	1 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
271	Data Size Data Rate Delay	Any Any Any	0 1 0	0 0 1	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
272	Data Size Data Rate Delay	Any Any Any	0 1 0	0 1 1	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
273	Data Size Data Rate Delay	Any Any Any	0 1 0	1 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
274	Data Size Data Rate Delay	Any Any Any	0 1 0	1 0 1	1 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
275	Data Size Data Rate Delay	Any Any Any	0 1 0	1 1 0	1 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
276	Data Size Data Rate Delay	Any Any Any	0 1 0	1 1 1	1 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
277	Data Size Data Rate Delay	Any Any Any	0 1 1	0 0 1	0 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
278	Data Size Data Rate Delay	Any Any Any	0 1 1	0 1 0	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
279	Data Size Data Rate Delay	Any Any Any	0 1 1	1 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
280	Data Size	Any	0	1	1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
	Data Rate	Any	1	0	0								
	Delay	Any	1	1	1								
281	Data Size	Any	0	1	1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
	Data Rate	Any	1	1	1								
	Delay	Any	1	0	0								
282	Data Size	Any	0	1	1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
	Data Rate	Any	1	1	1								
	Delay	Any	1	1	1								
283	Data Size	Any	1	0	0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
	Data Rate	Any	0	0	0								
	Delay	Any	0	1	1								
284	Data Size	Any	1	0	0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
	Data Rate	Any	0	1	1								
	Delay	Any	0	0	0								
285	Data Size	Any	1	0	0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
	Data Rate	Any	0	1	1								
	Delay	Any	0	1	1								
286	Data Size	Any	1	1	1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
	Data Rate	Any	0	0	0								
	Delay	Any	0	1	1								

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
287	Data Size Data Rate Delay	Any Any Any	1 0 0	1 1 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
288	Data Size Data Rate Delay	Any Any Any	1 0 0	1 1 1	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
289	Data Size Data Rate Delay	Any Any Any	1 0 1	0 0 1	0 0 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
290	Data Size Data Rate Delay	Any Any Any	1 0 1	0 1 0	0 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
291	Data Size Data Rate Delay	Any Any Any	1 0 1	0 1 1	0 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
292	Data Size Data Rate Delay	Any Any Any	1 0 1	1 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
293	Data Size Data Rate Delay	Any Any Any	1 0 1	1 1 0	1 1 0	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	2	3	4	1	2	3	4	1	2	3	4
		SCADA	Smart Meter	CCTV	Internet								
294	Data Size Data Rate Delay	Any Any Any	1 0 1	1 1 1	1 1 1	SCADA	CCTV	Internet	Smart Meter	SCADA	CCTV	Internet	Smart Meter
295	Data Size Data Rate Delay	Any Any Any	1 1 0	0 1 0	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
296	Data Size Data Rate Delay	Any Any Any	1 1 0	0 1 1	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
297	Data Size Data Rate Delay	Any Any Any	1 1 0	1 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
298	Data Size Data Rate Delay	Any Any Any	1 1 0	1 0 1	1 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
299	Data Size Data Rate Delay	Any Any Any	1 1 1	0 1 0	0 1 0	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet
300	Data Size Data Rate Delay	Any Any Any	1 1 1	0 1 1	0 1 1	SCADA	CCTV	Smart Meter	Internet	SCADA	CCTV	Smart Meter	Internet

		Input Traffic				Output Traffic							
						NS-3				MATLAB			
		1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
		SCADA	**Smart Meter**	**CCTV**	**Internet**								
301	Data Size Data Rate Delay	Any Any Any	1 1 1	1 0 0	1 0 0	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV
302	Data Size Data Rate Delay	Any Any Any	1 1 1	1 0 1	1 0 1	SCADA	Smart Meter	Internet	CCTV	SCADA	Smart Meter	Internet	CCTV

APPENDIX B

Appendix B : Source code for algorithm validation between NS-3 and MATLAB and performance study of CATSchA algorithm

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/mpls-module.h"
#include "ns3/log.h"
#include "ns3/flow-monitor-module.h"

#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace ns3;
using namespace mpls;
using namespace std;

NS_LOG_COMPONENT_DEFINE ("mpls-ppp-example");

int
main (int argc, char *argv[])
{
  CommandLine cmd;
  cmd.Parse (argc, argv);

  Time::SetResolution (Time::NS);

  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("mpls::MplsProtocol", LOG_LEVEL_DEBUG);
  LogComponentEnable ("mpls::Ipv4Routing", LOG_LEVEL_DEBUG);
  LogComponentEnable ("MplsNetworkDiscoverer", LOG_LEVEL_DEBUG);

  NodeContainer hosts;
  NodeContainer routers;
  NodeContainer routers2;

  PointToPointHelper pointToPoint;
  Ipv4AddressHelper address;
  NetDeviceContainer devices;
  InternetStackHelper internet;
  MplsNetworkConfigurator network;
```

```
  hosts.Create (5);
  internet.Install (hosts);
  routers = network.CreateAndInstall (3);

  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("100Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

//------Start of Context-aware algorithm-----------------

int MOQoSA[8][5];

MOQoSA[0][0]=1; MOQoSA[0][1]=1; MOQoSA[0][2]=1; MOQoSA[0][3]=1;
MOQoSA[0][4]=1;
MOQoSA[1][0]=1; MOQoSA[1][1]=1; MOQoSA[1][2]=0; MOQoSA[1][3]=0;
MOQoSA[1][4]=2;
MOQoSA[2][0]=1; MOQoSA[2][1]=0; MOQoSA[2][2]=1; MOQoSA[2][3]=0;
MOQoSA[2][4]=3;
MOQoSA[3][0]=1; MOQoSA[3][1]=0; MOQoSA[3][2]=0; MOQoSA[3][3]=2;
MOQoSA[3][4]=4;
MOQoSA[4][0]=0; MOQoSA[4][1]=1; MOQoSA[4][2]=1; MOQoSA[4][3]=3;
MOQoSA[4][4]=5;
MOQoSA[5][0]=0; MOQoSA[5][1]=1; MOQoSA[5][2]=0; MOQoSA[5][3]=0;
MOQoSA[5][4]=6;
MOQoSA[6][0]=0; MOQoSA[6][1]=0; MOQoSA[6][2]=1; MOQoSA[6][3]=0;
MOQoSA[6][4]=7;
MOQoSA[7][0]=0; MOQoSA[7][1]=0; MOQoSA[7][2]=0; MOQoSA[7][3]=4;
MOQoSA[7][4]=8;

cout << MOQoSA[0][0] << MOQoSA[0][1] << MOQoSA[0][2] << MOQoSA[0][3]
<< MOQoSA[0][4] << "\n";
cout << MOQoSA[1][0] << MOQoSA[1][1] << MOQoSA[1][2] << MOQoSA[1][3]
<< MOQoSA[1][4] << "\n";
cout << MOQoSA[2][0] << MOQoSA[2][1] << MOQoSA[2][2] << MOQoSA[2][3]
<< MOQoSA[2][4] << "\n";
cout << MOQoSA[3][0] << MOQoSA[3][1] << MOQoSA[3][2] << MOQoSA[3][3]
<< MOQoSA[3][4] << "\n";

cout << MOQoSA[4][0] << MOQoSA[4][1] << MOQoSA[4][2] << MOQoSA[4][3]
<< MOQoSA[4][4] << "\n";
cout << MOQoSA[5][0] << MOQoSA[5][1] << MOQoSA[5][2] << MOQoSA[5][3]
<< MOQoSA[5][4] << "\n";
cout << MOQoSA[6][0] << MOQoSA[6][1] << MOQoSA[6][2] << MOQoSA[6][3]
<< MOQoSA[6][4] << "\n";
cout << MOQoSA[7][0] << MOQoSA[7][1] << MOQoSA[7][2] << MOQoSA[7][3]
<< MOQoSA[7][4] << "\n";

//pre-defined priority
int SmartMeter=MOQoSA[1][4];
int CCTV=MOQoSA[6][4];
int Internet=MOQoSA[7][4];
```

```
//threshold values
double start2 = 0.02; //Smart Meter
double start3 = 0.055; //CCTV
double start4 = 0.105; //Internet

double stop_time_SCADA = 0.015;
double stop_time2 = 0.05;
double stop_time3 = 0.1;
double stop_time4 = 0.165;

double gap_time2 = stop_time2-start2;
double gap_time3 = stop_time3-start3;
double gap_time4 = stop_time4-start4;

double SmartMeterTrafficLatency_th=7.5;
double SmartMeterTrafficDataRate_th=30;//Mbps
double
SmartMeterTrafficDataSize_th=SmartMeterTrafficDataRate_th/gap_time2;//Mb

double CCTVTrafficLatency_th=2.5;
double CCTVTrafficDataRate_th=20;//Mbps
double CCTVTrafficDataSize_th=CCTVTrafficDataRate_th/gap_time3;//Mb

double InternetTrafficLatency_th=8.5;
double InternetTrafficDataRate_th=20;//Mbps
double InternetTrafficDataSize_th=InternetTrafficDataRate_th/gap_time4;//Mb

//initial input
double SCADATrafficDataRate=25;

double SmartMeterTrafficLatency=6;
double SmartMeterTrafficDataRate=25;
double SmartMeterTrafficDataSize=SmartMeterTrafficDataRate/gap_time2;

double CCTVTrafficLatency=2;
double CCTVTrafficDataRate=25;
double CCTVTrafficDataSize=CCTVTrafficDataRate/gap_time3;

double InternetTrafficLatency=7;
double InternetTrafficDataRate=25;
double InternetTrafficDataSize=InternetTrafficDataRate/gap_time4;

std::cout << "Before Context-aware:" << "\n";
std::cout << "SmartMeter is = " << SmartMeter << "\n";
std::cout << "CCTV is = " << CCTV << "\n";
std::cout << "Internet is = " << Internet << "\n";
std::cout << "Header Traffic is Smart Meter" << "\n";
```

```
if (SmartMeterTrafficDataSize > SmartMeterTrafficDataSize_th)
{
 SmartMeter=MOQoSA[5][4];//[0 1 0];
 if (SmartMeterTrafficDataRate > SmartMeterTrafficDataRate_th)
 {
  SmartMeter=MOQoSA[7][4];//[0 0 0];
   if (SmartMeterTrafficLatency >= SmartMeterTrafficLatency_th)
   {
     SmartMeter=MOQoSA[7][4];//[0 0 0]
   }
   else if (SmartMeterTrafficLatency < SmartMeterTrafficLatency_th)
   {
     SmartMeter=MOQoSA[6][4];//[0 0 1]
   }
 }
 else if (SmartMeterTrafficDataRate <= SmartMeterTrafficDataRate_th)
 {
  SmartMeter=MOQoSA[5][4];//[0 1 0];
   if (SmartMeterTrafficLatency >= SmartMeterTrafficLatency_th)
   {
     SmartMeter=MOQoSA[5][4];//[0 1 0]
   }
   else if (SmartMeterTrafficLatency < SmartMeterTrafficLatency_th)
   {
     SmartMeter=MOQoSA[4][4];//[0 1 1]
   }
 }
}

else if (SmartMeterTrafficDataSize <= SmartMeterTrafficDataSize_th)
{
 SmartMeter=MOQoSA[1][4];//[1 1 0]
 if (SmartMeterTrafficDataRate > SmartMeterTrafficDataRate_th)
 {
  SmartMeter=MOQoSA[3][4];//[1 0 0];
   if (SmartMeterTrafficLatency >= SmartMeterTrafficLatency_th)
   {
     SmartMeter=MOQoSA[3][4];//[1 0 0]
   }
   else if (SmartMeterTrafficLatency < SmartMeterTrafficLatency_th)
   {
     SmartMeter=MOQoSA[2][4];//[1 0 1]
   }
 }
 else if (SmartMeterTrafficDataRate <= SmartMeterTrafficDataRate_th)
 {
  SmartMeter=MOQoSA[1][4];//[1 1 0]
   if (SmartMeterTrafficLatency >= SmartMeterTrafficLatency_th)
   {
     SmartMeter=MOQoSA[1][4];//[1 1 0]
```

```
        }
        else if (SmartMeterTrafficLatency < SmartMeterTrafficLatency_th)
        {
           SmartMeter=MOQoSA[1][4];//[1 1 0]%MOQoSA(1,:);%[1 1 1]
        }
     }
    }

std::cout << "Header Traffic is CCTV" << "\n";

    if (InternetTrafficDataSize >= InternetTrafficDataSize_th)
    {
     Internet=MOQoSA[6][4];//[0 0 1];
     if (InternetTrafficDataRate >= InternetTrafficDataRate_th)
     {
       Internet=MOQoSA[6][4];//[0 0 1];
        if (InternetTrafficLatency > InternetTrafficLatency_th)
        {
           Internet=MOQoSA[7][4];//[0 0 0]
        }
        else if (InternetTrafficLatency <= InternetTrafficLatency_th)
        {
           Internet=MOQoSA[6][4];//[0 0 1]
        }
     }
     else if (InternetTrafficDataRate < InternetTrafficDataRate_th)
     {
       Internet=MOQoSA[4][4];//[0 1 1];
        if (InternetTrafficLatency > InternetTrafficLatency_th)
        {
           Internet=MOQoSA[5][4];//[0 1 0]
        }
        else if (InternetTrafficLatency <= InternetTrafficLatency_th)
        {
           Internet=MOQoSA[4][4];//[0 1 1]
        }
     }
    }

    else if (InternetTrafficDataSize < InternetTrafficDataSize_th)
    {
     Internet=MOQoSA[2][4];//[1 0 1];
     if (InternetTrafficDataRate >= InternetTrafficDataRate_th)
     {
       Internet=MOQoSA[2][4];//[1 0 1];
        if (InternetTrafficLatency > InternetTrafficLatency_th)
        {
           Internet=MOQoSA[3][4];//[1 0 0]
        }
        else if (InternetTrafficLatency <= InternetTrafficLatency_th)
```

```
        {
          Internet=MOQoSA[2][4];//[1 0 1]
        }
     }
     else if (InternetTrafficDataRate < InternetTrafficDataRate_th)
     {
      Internet=MOQoSA[1][4];//[1 1 0];%MOQoSA(1,:);%[1 1 1]
        if (InternetTrafficLatency > InternetTrafficLatency_th)
        {
          Internet=MOQoSA[1][4];//[1 1 0]
        }
        else if (InternetTrafficLatency <= InternetTrafficLatency_th)
        {
          Internet=MOQoSA[1][4];//[1 1 0]%MOQoSA(1,:);%[1 1 1]
        }
     }
    }

std::cout << "Header Traffic is Internet" << "\n";

    if (CCTVTrafficDataSize >= CCTVTrafficDataSize_th)
    {
     CCTV=MOQoSA[7][4];//[0 0 0]
     if (CCTVTrafficDataRate >= CCTVTrafficDataRate_th)
     {
      CCTV=MOQoSA[7][4];//[0 0 0]
        if (CCTVTrafficLatency >= CCTVTrafficLatency_th)
        {
          CCTV=MOQoSA[7][4];//[0 0 0]
        }
        else if (CCTVTrafficLatency < CCTVTrafficLatency_th)
        {
          CCTV=MOQoSA[6][4];//[0 0 1]
        }
     }
     else if (CCTVTrafficDataRate < CCTVTrafficDataRate_th)
     {
         CCTV=MOQoSA[5][4];//[0 1 0]
        if (CCTVTrafficLatency >= CCTVTrafficLatency_th)
        {
          CCTV=MOQoSA[5][4];//[0 1 0]
        }
        else if (CCTVTrafficLatency < CCTVTrafficLatency_th)
        {
          CCTV=MOQoSA[4][4];//[0 1 1]
        }
     }
    }
```

```
     else if (CCTVTrafficDataSize < CCTVTrafficDataSize_th)
     {
          CCTV=MOQoSA[3][4];//[1 0 0]
       if (CCTVTrafficDataRate >= CCTVTrafficDataRate_th)
       {
        CCTV=MOQoSA[3][4];//[1 0 0]
          if (CCTVTrafficLatency >= CCTVTrafficLatency_th)
          {
             CCTV=MOQoSA[3][4];//[1 0 0]
          }
          else if (CCTVTrafficLatency < CCTVTrafficLatency_th)
          {
             CCTV=MOQoSA[2][4];//[1 0 1]
          }
       }
       else if (CCTVTrafficDataRate < CCTVTrafficDataRate_th)
       {
        CCTV=MOQoSA[1][4];//[1 1 0]
          if (CCTVTrafficLatency >= CCTVTrafficLatency_th)
          {
             CCTV=MOQoSA[1][4];//[1 1 0]
          }
          else if (CCTVTrafficLatency < CCTVTrafficLatency_th)
          {
           CCTV=MOQoSA[1][4];//[1 1 0]//%MOQoSA(1,:);%[1 1 1]//
          }
       }
     }

std::cout << "After Context-aware, before swapping if clash:" << "\n";
std::cout << "SmartMeter is = " << SmartMeter << "\n";
std::cout << "CCTV is = " << CCTV << "\n";
std::cout << "Internet is = " << Internet << "\n";

//swapping if clash
int orilocSmartMeter=MOQoSA[1][4];
int orilocCCTV=MOQoSA[6][4];
int orilocInternet=MOQoSA[7][4];

if ((SmartMeter==CCTV) && (Internet==CCTV))
{
  SmartMeter=SmartMeter;
  CCTV=orilocSmartMeter;
  Internet=orilocCCTV;
  if (SmartMeter==orilocSmartMeter)
  {
    CCTV=orilocSmartMeter;
    SmartMeter=orilocCCTV;
  }
  if (SmartMeter==Internet)
```

```
  {
   SmartMeter=SmartMeter;
   Internet=orilocInternet;
  }
}

if (SmartMeter==CCTV)
{
 SmartMeter=SmartMeter;
 CCTV=orilocSmartMeter;

 if (SmartMeter==orilocSmartMeter)
 {
  CCTV=orilocSmartMeter;
  SmartMeter=orilocCCTV;
 }

 if (SmartMeter==Internet)
 {
  SmartMeter=SmartMeter;
  Internet=orilocInternet;

  if (SmartMeter==orilocInternet)
  {
   Internet=orilocCCTV;
   SmartMeter=SmartMeter;
  }
 }

 if (Internet==CCTV)
 {
  Internet=orilocInternet;
  CCTV=CCTV;

  if (SmartMeter==orilocInternet)
  {
   Internet=orilocCCTV;
   SmartMeter=SmartMeter;
  }
 }
}

if (SmartMeter==Internet)
{
 SmartMeter=SmartMeter;
 Internet=orilocSmartMeter;

 if (SmartMeter==orilocSmartMeter)
 {
  Internet=orilocSmartMeter;
```

```
    SmartMeter=orilocInternet;
  }

  if (SmartMeter==CCTV)
  {
    SmartMeter=SmartMeter;
    CCTV=orilocCCTV;

    if (SmartMeter==orilocCCTV)
    {
      CCTV=orilocInternet;
      SmartMeter=SmartMeter;
    }
  }

  if (Internet==CCTV)
  {
    Internet=Internet;
    CCTV=orilocCCTV;

    if (SmartMeter==orilocCCTV)
    {
      CCTV=orilocInternet;
      SmartMeter=SmartMeter;
    }
  }
}

if (CCTV==Internet)
{
  CCTV=CCTV;
  Internet=orilocCCTV;

  if (CCTV==orilocCCTV)
  {
    Internet=orilocCCTV;
    CCTV=orilocInternet;
  }

  if (SmartMeter==Internet)
  {
    SmartMeter=orilocInternet;
    Internet=Internet;

    if (Internet==orilocInternet)
    {
      CCTV=CCTV;
      SmartMeter=orilocSmartMeter;
    }
  }
```

```
  if (SmartMeter==CCTV)
  {
    SmartMeter=orilocInternet;
    CCTV=CCTV;

    if (SmartMeter==orilocInternet)
    {
      CCTV=CCTV;
      SmartMeter=orilocSmartMeter;
    }
  }
}

//swap time
double start_SmartMeter = start2;
double start_CCTV = start3;
double start_Internet = start4;

double stop_SmartMeter = stop_time2;
double stop_CCTV = stop_time3;
double stop_Internet = stop_time4;

double datarate1 = SCADATrafficDataRate;
double datarate2 = SmartMeterTrafficDataRate;
double datarate3 = CCTVTrafficDataRate;
double datarate4 = InternetTrafficDataRate;

cout << "datarate2= " << datarate2 << "\n";
cout << "datarate3= " << datarate3 << "\n";
cout << "datarate4= " << datarate4 << "\n";

if ((SmartMeter<Internet) && (Internet<CCTV))
{
  std::cout << "Case 2" << "\n";
  start_SmartMeter=start2;
  start_Internet=start3;
  start_CCTV=start4;

  stop_SmartMeter=stop_time2;
  stop_Internet=stop_time3;
  stop_CCTV=stop_time4;

  datarate2 = SmartMeterTrafficDataRate;
  datarate3 = InternetTrafficDataRate;
  datarate4 = CCTVTrafficDataRate;
}
```

```
if ((CCTV<SmartMeter) && (SmartMeter<Internet))
{
  std::cout << "Case 3" << "\n";
  start_CCTV=start2;
  start_SmartMeter=start3;
  start_Internet=start4;

  stop_CCTV=stop_time2;
  stop_SmartMeter=stop_time3;
  stop_Internet=stop_time4;

  datarate2 = CCTVTrafficDataRate;
  datarate3 = SmartMeterTrafficDataRate;
  datarate4 = InternetTrafficDataRate;
}

if ((CCTV<Internet) && (Internet<SmartMeter))
{
  std::cout << "Case 4" << "\n";
  start_CCTV=start2;
  start_Internet=start3;
  start_SmartMeter=start4;

  stop_CCTV=stop_time2;
  stop_Internet=stop_time3;
  stop_SmartMeter=stop_time4;

  datarate2 = CCTVTrafficDataRate;
  datarate3 = InternetTrafficDataRate;
  datarate4 = SmartMeterTrafficDataRate;
}

if ((Internet<SmartMeter) && (SmartMeter<CCTV))
{
  std::cout << "Case 5" << "\n";
  start_Internet=start2;
  start_SmartMeter=start3;
  start_CCTV=start4;

  stop_Internet=stop_time2;
  stop_SmartMeter=stop_time3;
  stop_CCTV=stop_time4;

  datarate2 = InternetTrafficDataRate;
  datarate3 = SmartMeterTrafficDataRate;
  datarate4 = CCTVTrafficDataRate;
}
```

```
if ((Internet<CCTV) && (CCTV<SmartMeter))
{
  std::cout << "Case 6" << "\n";
  start_Internet=start2;
  start_CCTV=start3;
  start_SmartMeter=start4;

  stop_Internet=stop_time2;
  stop_CCTV=stop_time3;
  stop_SmartMeter=stop_time4;

  datarate2 = InternetTrafficDataRate;
  datarate3 = CCTVTrafficDataRate;
  datarate4 = SmartMeterTrafficDataRate;
}

std::cout << "After swapping if clash:" << "\n";
std::cout << "SmartMeter is = " << SmartMeter << "; start time is = " <<
start_SmartMeter << "\n";
std::cout << "CCTV is = " << CCTV << "; start time is = " << start_CCTV << "\n";
std::cout << "Internet is = " << Internet << "; start time is = " << start_Internet <<
"\n";

//--------End of Context-aware algorithm-----------------

uint16_t port = 9;
cout << "datarate1= " << datarate1 << "\n";
std::ostringstream strs1;
strs1 << datarate1;
std::string str_datarate1 = strs1.str();
cout << "str_datarate1 =" << str_datarate1+"Mbps" << "\n";

  OnOffHelper onoff1 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff1.SetConstantRate (DataRate (str_datarate1+"Mbps"));
  onoff1.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps1 = onoff1.Install (hosts.Get (0));
  apps1.Start (Seconds (0.0));
  apps1.Stop (Seconds (stop_time_SCADA));

cout << "datarate2= " << datarate2 << "\n";
std::ostringstream strs2;
strs2 << datarate2;
std::string str_datarate2 = strs2.str();
cout << "str_datarate2 =" << str_datarate2+"Mbps" << "\n";

  OnOffHelper onoff2 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff2.SetConstantRate (DataRate (str_datarate2+"Mbps"));
  onoff2.SetAttribute ("PacketSize", UintegerValue (1024));
```

```
  ApplicationContainer apps2 = onoff2.Install (hosts.Get (1));
  apps2.Start (Seconds (start_SmartMeter));
  apps2.Stop (Seconds (stop_SmartMeter));

cout << "datarate3= " << datarate3 << "\n";
std::ostringstream strs3;
strs3 << datarate3;
std::string str_datarate3 = strs3.str();
cout << "str_datarate3 =" << str_datarate3+"Mbps" << "\n";

  OnOffHelper onoff3 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff3.SetConstantRate (DataRate (str_datarate3+"Mbps"));
  onoff3.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps3 = onoff3.Install (hosts.Get (2));
  apps3.Start (Seconds (start_CCTV));
  apps3.Stop (Seconds (stop_CCTV));

cout << "datarate4= " << datarate4 << "\n";
std::ostringstream strs4;
strs4 << datarate4;
std::string str_datarate4 = strs4.str();
cout << "str_datarate4 =" << str_datarate4+"Mbps" << "\n";

  OnOffHelper onoff4 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff4.SetConstantRate (DataRate (str_datarate4+"Mbps"));
  onoff4.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps4 = onoff4.Install (hosts.Get (3));
  apps4.Start (Seconds (start_Internet));
  apps4.Stop (Seconds (stop_Internet));

  // Create an optional packet sink to receive these packets
  PacketSinkHelper sink ("ns3::UdpSocketFactory",
  Address (InetSocketAddress (Ipv4Address::GetAny (), port)));
  apps4 = sink.Install (hosts.Get (4));
  apps4.Start (Seconds (0.0));

  // Hosts configuration
  devices = pointToPoint.Install (hosts.Get(0), routers.Get(0));
  address.SetBase ("192.168.1.0", "255.255.255.0");
  address.Assign (devices);

  devices = pointToPoint.Install (hosts.Get(1), routers.Get(0));
  address.SetBase ("192.168.2.0", "255.255.255.0");
  address.Assign (devices);

  devices = pointToPoint.Install (hosts.Get(2), routers.Get(0));
  address.SetBase ("192.168.3.0", "255.255.255.0");
  address.Assign (devices);
```

```
devices = pointToPoint.Install (hosts.Get(3), routers.Get(0));
address.SetBase ("192.168.4.0", "255.255.255.0");
address.Assign (devices);

devices = pointToPoint.Install (routers.Get(2), hosts.Get(4));
address.SetBase ("192.168.5.0", "255.255.255.0");
address.Assign(devices);

// Routers configuration
devices = pointToPoint.Install (routers.Get(0), routers.Get(1));
address.SetBase ("10.1.1.0", "255.255.255.0");
address.Assign (devices);

devices = pointToPoint.Install (routers.Get(1), routers.Get(2));
address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (devices);

// Address is not specified. Mpls interfaces will be disabled.
devices = pointToPoint.Install (routers.Get(2), routers.Get(0));

NhlfeSelectionPolicyHelper policy;
policy.SetAttribute ("MaxPacketsInTxQueue", IntegerValue (50));

MplsSwitch sw1 (routers.Get (0));
MplsSwitch sw2 (routers.Get (1));
MplsSwitch sw3 (routers.Get (2));

sw1.SetSelectionPolicy (policy);

sw1.AddFtn (Ipv4Source ("192.168.1.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (100), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.1.1"),
   Nhlfe (Swap (110), Ipv4Address ("10.1.3.1"))
);

sw1.AddFtn (Ipv4Source ("192.168.2.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (200), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.2.1"),
   Nhlfe (Swap (210), Ipv4Address ("10.1.3.1"))
);

sw1.AddFtn (Ipv4Source ("192.168.3.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (300), Ipv4Address ("10.1.1.2"))
);
```

```
sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.3.1"),
   Nhlfe (Swap (310), Ipv4Address ("10.1.3.1"))
);

sw1.AddFtn (Ipv4Source ("192.168.4.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (400), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.4.1"),
   Nhlfe (Swap (410), Ipv4Address ("10.1.3.1"))
);

sw2.AddIlm (100,
   Nhlfe(Swap (110), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (110,
   Nhlfe(Swap (100), Ipv4Address ("10.1.1.1"))
);

sw2.AddIlm (200,
   Nhlfe(Swap (210), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (210,
   Nhlfe(Swap (200), Ipv4Address ("10.1.1.1"))
);

sw2.AddIlm (300,
   Nhlfe(Swap (310), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (310,
   Nhlfe(Swap (300), Ipv4Address ("10.1.1.1"))
);

sw2.AddIlm (400,
   Nhlfe(Swap (410), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (410,
   Nhlfe(Swap (400), Ipv4Address ("10.1.1.1"))
);

sw3.AddIlm (110,
   Nhlfe(Pop ())
);

sw1.AddIlm (100,
   Nhlfe(Pop ())
);

sw3.AddIlm (210,
   Nhlfe(Pop ())
```

```
  );

  sw1.AddIlm (200,
     Nhlfe(Pop ())
  );

  sw3.AddIlm (310,
     Nhlfe(Pop ())
  );

  sw1.AddIlm (300,
     Nhlfe(Pop ())
  );

  sw3.AddIlm (410,
     Nhlfe(Pop ())
  );

  sw1.AddIlm (400,
     Nhlfe(Pop ())
  );

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  network.DiscoverNetwork ();

  pointToPoint.EnablePcapAll ("mpls-ppp-example");

 //Install FlowMonitor on all nodes
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll ();

 Simulator::Run ();

  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin ();
i != stats.end (); ++i)
    {
      // first 2 FlowIds are for ECHO apps, we don't want to display them
      if (i->first > 0)
        {
          Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
          std::cout << "Flow " << i->first - 0 << " (" << t.sourceAddress << " -> " <<
t.destinationAddress << ")\n";
          std::cout << "  Tx Bytes:   " << i->second.txBytes << "\n";
          std::cout << "  Rx Bytes:   " << i->second.rxBytes << "\n";
          std::cout << "  Tx Packets:   " << i->second.txPackets << "\n";
```

```
        std::cout << "  Rx Packets:   " << i->second.rxPackets << "\n";
        std::cout << "  Lost Packets:   " << i->second.txPackets - i->second.rxPackets
<< "\n";
        std::cout << "  Times Forwarded:   " << float(i->second.timesForwarded) <<
"\n";
        std::cout << "  Mean Hop Count:   " << float(i->second.timesForwarded) /
i->second.rxPackets + 1 << "\n";
        std::cout << "  Mean Delay:   " << i->second.delaySum.GetSeconds() /
i->second.rxPackets << "\n";
        std::cout << "  Sum Delay:   " << i->second.delaySum.GetSeconds() << "\n";
        std::cout << "  Mean Jitter:   " << i->second.jitterSum.GetSeconds() /
(i->second.rxPackets-1)<< "\n";
        std::cout << "  Last Delay:   " << i->second.lastDelay.GetSeconds() << "\n";
        std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / 10.0 / 1000 / 1000
<< " Mbytes\n";
        std::cout << "  Delay Histogram:   " << i->second.delayHistogram.GetNBins ()
<< "\n";
        std::cout << "  Jitter Histogram:   " << i->second.jitterHistogram.GetNBins ()
<< "\n";
        std::cout << "  Packet Size Histogram:   " <<
i->second.packetSizeHistogram.GetNBins ()  << "\n";
        std::cout << "  Flow Interruptions Histogram:   " <<
i->second.flowInterruptionsHistogram.GetNBins ()  << "\n";
       }
    }

  Simulator::Destroy ();

  return 0;
}
```

APPENDIX C

Appendix C : Source code for performance evaluation of CATSchA algorithm with contextual aware

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/mpls-module.h"
#include "ns3/log.h"
#include "ns3/flow-monitor-module.h"

#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace ns3;
using namespace mpls;
using namespace std;

NS_LOG_COMPONENT_DEFINE ("mpls-ppp-example");

int
main (int argc, char *argv[])
{
  CommandLine cmd;
  cmd.Parse (argc, argv);

  Time::SetResolution (Time::NS);

  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("mpls::MplsProtocol", LOG_LEVEL_DEBUG);
  LogComponentEnable ("mpls::Ipv4Routing", LOG_LEVEL_DEBUG);
  LogComponentEnable ("MplsNetworkDiscoverer", LOG_LEVEL_DEBUG);

  NodeContainer hosts;
  NodeContainer routers;
  NodeContainer routers2;

  PointToPointHelper pointToPoint;
  Ipv4AddressHelper address;
  NetDeviceContainer devices;
  InternetStackHelper internet;
  MplsNetworkConfigurator network;
```

```
  hosts.Create (5);
  internet.Install (hosts);
  routers = network.CreateAndInstall (3);

  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("100Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

//------Start of Context-aware algorithm-----------------

int MOQoSA[8][5];

MOQoSA[0][0]=1; MOQoSA[0][1]=1; MOQoSA[0][2]=1; MOQoSA[0][3]=1;
MOQoSA[0][4]=1;
MOQoSA[1][0]=1; MOQoSA[1][1]=1; MOQoSA[1][2]=0; MOQoSA[1][3]=0;
MOQoSA[1][4]=2;
MOQoSA[2][0]=1; MOQoSA[2][1]=0; MOQoSA[2][2]=1; MOQoSA[2][3]=0;
MOQoSA[2][4]=3;
MOQoSA[3][0]=1; MOQoSA[3][1]=0; MOQoSA[3][2]=0; MOQoSA[3][3]=2;
MOQoSA[3][4]=4;
MOQoSA[4][0]=0; MOQoSA[4][1]=1; MOQoSA[4][2]=1; MOQoSA[4][3]=3;
MOQoSA[4][4]=5;
MOQoSA[5][0]=0; MOQoSA[5][1]=1; MOQoSA[5][2]=0; MOQoSA[5][3]=0;
MOQoSA[5][4]=6;
MOQoSA[6][0]=0; MOQoSA[6][1]=0; MOQoSA[6][2]=1; MOQoSA[6][3]=0;
MOQoSA[6][4]=7;
MOQoSA[7][0]=0; MOQoSA[7][1]=0; MOQoSA[7][2]=0; MOQoSA[7][3]=4;
MOQoSA[7][4]=8;

cout << MOQoSA[0][0] << MOQoSA[0][1] << MOQoSA[0][2] << MOQoSA[0][3]
<< MOQoSA[0][4] << "\n";
cout << MOQoSA[1][0] << MOQoSA[1][1] << MOQoSA[1][2] << MOQoSA[1][3]
<< MOQoSA[1][4] << "\n";
cout << MOQoSA[2][0] << MOQoSA[2][1] << MOQoSA[2][2] << MOQoSA[2][3]
<< MOQoSA[2][4] << "\n";
cout << MOQoSA[3][0] << MOQoSA[3][1] << MOQoSA[3][2] << MOQoSA[3][3]
<< MOQoSA[3][4] << "\n";

cout << MOQoSA[4][0] << MOQoSA[4][1] << MOQoSA[4][2] << MOQoSA[4][3]
<< MOQoSA[4][4] << "\n";
cout << MOQoSA[5][0] << MOQoSA[5][1] << MOQoSA[5][2] << MOQoSA[5][3]
<< MOQoSA[5][4] << "\n";
cout << MOQoSA[6][0] << MOQoSA[6][1] << MOQoSA[6][2] << MOQoSA[6][3]
<< MOQoSA[6][4] << "\n";
cout << MOQoSA[7][0] << MOQoSA[7][1] << MOQoSA[7][2] << MOQoSA[7][3]
<< MOQoSA[7][4] << "\n";

//pre-defined priority
int SmartMeter=MOQoSA[1][4];
int CCTV=MOQoSA[6][4];
int Internet=MOQoSA[7][4];
```

```
//threshold values
double start2 = 0.065; //Smart Meter
double start3 = 0.073; //CCTV
double start4 = 0.095; //Internet

double stop_time_SCADA = 0.06;
double stop_time2 = 0.165;
double stop_time3 = 0.165;
double stop_time4 = 0.165;

double gap_time2 = stop_time2-start2;
double gap_time3 = stop_time3-start3;
double gap_time4 = stop_time4-start4;

double SmartMeterTrafficLatency_th=7.5;
double SmartMeterTrafficDataRate_th=9.57;//Mbps
double
SmartMeterTrafficDataSize_th=SmartMeterTrafficDataRate_th/gap_time2;//Mb

double CCTVTrafficLatency_th=2.5;
double CCTVTrafficDataRate_th=35.87;//Mbps
double CCTVTrafficDataSize_th=CCTVTrafficDataRate_th/gap_time3;//Mb

double InternetTrafficLatency_th=8.5;
double InternetTrafficDataRate_th=59.78;//Mbps
double InternetTrafficDataSize_th=InternetTrafficDataRate_th/gap_time4;//Mb

//initial input
double SCADATrafficDataRate=6;//

double SmartMeterTrafficLatency=6;//
double SmartMeterTrafficDataRate=10;//
double SmartMeterTrafficDataSize=SmartMeterTrafficDataRate/gap_time2;//

double CCTVTrafficLatency=3;//
double CCTVTrafficDataRate=36;//
double CCTVTrafficDataSize=CCTVTrafficDataRate/gap_time3;//

double InternetTrafficLatency=9;//
double InternetTrafficDataRate=58;//
double InternetTrafficDataSize=InternetTrafficDataRate/gap_time4;//

std::cout << "Before Context-aware:" << "\n";
std::cout << "SmartMeter is = " << SmartMeter << "\n";
std::cout << "CCTV is = " << CCTV << "\n";
std::cout << "Internet is = " << Internet << "\n";
```

```
std::cout << "Header Traffic is Smart Meter" << "\n";

    if (SmartMeterTrafficDataSize > SmartMeterTrafficDataSize_th)
    {
      SmartMeter=MOQoSA[5][4];//[0 1 0];
      if (SmartMeterTrafficDataRate > SmartMeterTrafficDataRate_th)
      {
        SmartMeter=MOQoSA[7][4];//[0 0 0];
          if (SmartMeterTrafficLatency >= SmartMeterTrafficLatency_th)
          {
              SmartMeter=MOQoSA[7][4];//[0 0 0]
          }
          else if (SmartMeterTrafficLatency < SmartMeterTrafficLatency_th)
          {
              SmartMeter=MOQoSA[6][4];//[0 0 1]
          }
      }
      else if (SmartMeterTrafficDataRate <= SmartMeterTrafficDataRate_th)
      {
        SmartMeter=MOQoSA[5][4];//[0 1 0];
          if (SmartMeterTrafficLatency >= SmartMeterTrafficLatency_th)
          {
              SmartMeter=MOQoSA[5][4];//[0 1 0]
          }
          else if (SmartMeterTrafficLatency < SmartMeterTrafficLatency_th)
          {
              SmartMeter=MOQoSA[4][4];//[0 1 1]
          }
      }
    }

    else if (SmartMeterTrafficDataSize <= SmartMeterTrafficDataSize_th)
    {
      SmartMeter=MOQoSA[1][4];//[1 1 0]
      if (SmartMeterTrafficDataRate > SmartMeterTrafficDataRate_th)
      {
        SmartMeter=MOQoSA[3][4];//[1 0 0];
          if (SmartMeterTrafficLatency >= SmartMeterTrafficLatency_th)
          {
              SmartMeter=MOQoSA[3][4];//[1 0 0]
          }
          else if (SmartMeterTrafficLatency < SmartMeterTrafficLatency_th)
          {
              SmartMeter=MOQoSA[2][4];//[1 0 1]
          }
      }
      else if (SmartMeterTrafficDataRate <= SmartMeterTrafficDataRate_th)
      {
        SmartMeter=MOQoSA[1][4];//[1 1 0]
          if (SmartMeterTrafficLatency >= SmartMeterTrafficLatency_th)
```

```
        {
           SmartMeter=MOQoSA[1][4];//[1 1 0]
        }
        else if (SmartMeterTrafficLatency < SmartMeterTrafficLatency_th)
        {
           SmartMeter=MOQoSA[1][4];//[1 1 0]%MOQoSA(1,:);%[1 1 1]
        }
     }
   }

std::cout << "Header Traffic is CCTV" << "\n";

   if (InternetTrafficDataSize >= InternetTrafficDataSize_th)
   {
     Internet=MOQoSA[6][4];//[0 0 1];
     if (InternetTrafficDataRate >= InternetTrafficDataRate_th)
     {
      Internet=MOQoSA[6][4];//[0 0 1];
        if (InternetTrafficLatency > InternetTrafficLatency_th)
        {
           Internet=MOQoSA[7][4];//[0 0 0]
        }
        else if (InternetTrafficLatency <= InternetTrafficLatency_th)
        {
           Internet=MOQoSA[6][4];//[0 0 1]
        }
     }
     else if (InternetTrafficDataRate < InternetTrafficDataRate_th)
     {
      Internet=MOQoSA[4][4];//[0 1 1];
        if (InternetTrafficLatency > InternetTrafficLatency_th)
        {
           Internet=MOQoSA[5][4];//[0 1 0]
        }
        else if (InternetTrafficLatency <= InternetTrafficLatency_th)
        {
           Internet=MOQoSA[4][4];//[0 1 1]
        }
     }
   }

   else if (InternetTrafficDataSize < InternetTrafficDataSize_th)
   {
     Internet=MOQoSA[2][4];//[1 0 1];
     if (InternetTrafficDataRate >= InternetTrafficDataRate_th)
     {
      Internet=MOQoSA[2][4];//[1 0 1];
        if (InternetTrafficLatency > InternetTrafficLatency_th)
        {
           Internet=MOQoSA[3][4];//[1 0 0]
```

```
        }
        else if (InternetTrafficLatency <= InternetTrafficLatency_th)
        {
           Internet=MOQoSA[2][4];//[1 0 1]
        }
     }
     else if (InternetTrafficDataRate < InternetTrafficDataRate_th)
     {
      Internet=MOQoSA[1][4];//[1 1 0];%MOQoSA(1,:);%[1 1 1]
        if (InternetTrafficLatency > InternetTrafficLatency_th)
        {
           Internet=MOQoSA[1][4];//[1 1 0]
        }
        else if (InternetTrafficLatency <= InternetTrafficLatency_th)
        {
           Internet=MOQoSA[1][4];//[1 1 0]%MOQoSA(1,:);%[1 1 1]
        }
     }
   }

std::cout << "Header Traffic is Internet" << "\n";

   if (CCTVTrafficDataSize >= CCTVTrafficDataSize_th)
   {
     CCTV=MOQoSA[7][4];//[0 0 0]
     if (CCTVTrafficDataRate >= CCTVTrafficDataRate_th)
     {
      CCTV=MOQoSA[7][4];//[0 0 0]
        if (CCTVTrafficLatency >= CCTVTrafficLatency_th)
        {
           CCTV=MOQoSA[7][4];//[0 0 0]
        }
        else if (CCTVTrafficLatency < CCTVTrafficLatency_th)
        {
           CCTV=MOQoSA[6][4];//[0 0 1]
        }
     }
     else if (CCTVTrafficDataRate < CCTVTrafficDataRate_th)
     {
         CCTV=MOQoSA[5][4];//[0 1 0]
        if (CCTVTrafficLatency >= CCTVTrafficLatency_th)
        {
           CCTV=MOQoSA[5][4];//[0 1 0]
        }
        else if (CCTVTrafficLatency < CCTVTrafficLatency_th)
        {
           CCTV=MOQoSA[4][4];//[0 1 1]
        }
     }
   }
```

```
    else if (CCTVTrafficDataSize < CCTVTrafficDataSize_th)
    {
        CCTV=MOQoSA[3][4];//[1 0 0]
     if (CCTVTrafficDataRate >= CCTVTrafficDataRate_th)
     {
      CCTV=MOQoSA[3][4];//[1 0 0]
       if (CCTVTrafficLatency >= CCTVTrafficLatency_th)
       {
         CCTV=MOQoSA[3][4];//[1 0 0]
       }
       else if (CCTVTrafficLatency < CCTVTrafficLatency_th)
       {
         CCTV=MOQoSA[2][4];//[1 0 1]
       }
     }
     else if (CCTVTrafficDataRate < CCTVTrafficDataRate_th)
     {
      CCTV=MOQoSA[1][4];//[1 1 0]
       if (CCTVTrafficLatency >= CCTVTrafficLatency_th)
       {
         CCTV=MOQoSA[1][4];//[1 1 0]
       }
       else if (CCTVTrafficLatency < CCTVTrafficLatency_th)
       {
        CCTV=MOQoSA[1][4];//[1 1 0]//%MOQoSA(1,:);%[1 1 1]//
       }
     }
    }

std::cout << "After Context-aware, before swapping if clash:" << "\n";
std::cout << "SmartMeter is = " << SmartMeter << "\n";
std::cout << "CCTV is = " << CCTV << "\n";
std::cout << "Internet is = " << Internet << "\n";

//swapping if clash
int orilocSmartMeter=MOQoSA[1][4];
int orilocCCTV=MOQoSA[6][4];
int orilocInternet=MOQoSA[7][4];

if ((SmartMeter==CCTV) && (Internet==CCTV))
{
  SmartMeter=SmartMeter;
  CCTV=orilocSmartMeter;
  Internet=orilocCCTV+1;
  if (SmartMeter==orilocSmartMeter)
  {
    CCTV=orilocSmartMeter;
    SmartMeter=orilocCCTV;
  }
}
```

```
if (SmartMeter==CCTV)
{
  SmartMeter=SmartMeter;
  CCTV=orilocSmartMeter;
  if (SmartMeter==orilocSmartMeter)
  {
    CCTV=orilocSmartMeter;
    SmartMeter=orilocCCTV;
  }
}

if (SmartMeter==Internet)
{
  SmartMeter=SmartMeter;
  Internet=orilocSmartMeter;
   if (SmartMeter==orilocSmartMeter)
    {
      Internet=orilocSmartMeter;
      SmartMeter=orilocInternet;
    }
   if (SmartMeter==CCTV)
    {
      SmartMeter=orilocCCTV+1;
      CCTV=CCTV;
    }
}

if (Internet==CCTV)
{
  CCTV=CCTV;
  Internet=orilocCCTV;
  if (CCTV==orilocCCTV)
  {
    Internet=orilocCCTV;
    CCTV=orilocInternet;
  }

  if (SmartMeter==Internet)
  {
    SmartMeter=orilocInternet-1;
    Internet=Internet;
  }
  if (SmartMeter==CCTV)
  {
    SmartMeter=orilocCCTV+1;
    CCTV=CCTV;
  }
}
```

```
//swap time
double start_SmartMeter = start2;
double start_CCTV = start3;
double start_Internet = start4;

double stop_SmartMeter = stop_time2;
double stop_CCTV = stop_time3;
double stop_Internet = stop_time4;

double datarate1 = SCADATrafficDataRate;
double datarate2 = SmartMeterTrafficDataRate;
double datarate3 = CCTVTrafficDataRate;
double datarate4 = InternetTrafficDataRate;

cout << "datarate2= " << datarate2 << "\n";
cout << "datarate3= " << datarate3 << "\n";
cout << "datarate4= " << datarate4 << "\n";

if ((SmartMeter<Internet) && (Internet<CCTV))
{
  std::cout << "Case 2" << "\n";
  start_SmartMeter=start2;
  start_Internet=start3;
  start_CCTV=start4;

  stop_SmartMeter=stop_time2;
  stop_Internet=stop_time3;
  stop_CCTV=stop_time4;

  datarate2 = SmartMeterTrafficDataRate;
  datarate3 = InternetTrafficDataRate;
  datarate4 = CCTVTrafficDataRate;
}

if ((CCTV<SmartMeter) && (SmartMeter<Internet))
{
  std::cout << "Case 3" << "\n";
  start_CCTV=start2;
  start_SmartMeter=start3;
  start_Internet=start4;

  stop_CCTV=stop_time2;
  stop_SmartMeter=stop_time3;
  stop_Internet=stop_time4;

  datarate2 = CCTVTrafficDataRate;
  datarate3 = SmartMeterTrafficDataRate;
  datarate4 = InternetTrafficDataRate;
}
```

```
if ((CCTV<Internet) && (Internet<SmartMeter))
{
  std::cout << "Case 4" << "\n";
  start_CCTV=start2;
  start_Internet=start3;
  start_SmartMeter=start4;

  stop_CCTV=stop_time2;
  stop_Internet=stop_time3;
  stop_SmartMeter=stop_time4;

  datarate2 = CCTVTrafficDataRate;
  datarate3 = InternetTrafficDataRate;
  datarate4 = SmartMeterTrafficDataRate;
}

if ((Internet<SmartMeter) && (SmartMeter<CCTV))
{
  std::cout << "Case 5" << "\n";
  start_Internet=start2;
  start_SmartMeter=start3;
  start_CCTV=start4;

  stop_Internet=stop_time2;
  stop_SmartMeter=stop_time3;
  stop_CCTV=stop_time4;

  datarate2 = InternetTrafficDataRate;
  datarate3 = SmartMeterTrafficDataRate;
  datarate4 = CCTVTrafficDataRate;
}

if ((Internet<CCTV) && (CCTV<SmartMeter))
{
  std::cout << "Case 6" << "\n";
  start_Internet=start2;
  start_CCTV=start3;
  start_SmartMeter=start4;

  stop_Internet=stop_time2;
  stop_CCTV=stop_time3;
  stop_SmartMeter=stop_time4;

  datarate2 = InternetTrafficDataRate;
  datarate3 = CCTVTrafficDataRate;
  datarate4 = SmartMeterTrafficDataRate;
}
```

```
std::cout << "After swapping if clash:" << "\n";
std::cout << "SmartMeter is = " << SmartMeter << "; start time is = " <<
start_SmartMeter << "\n";
std::cout << "CCTV is = " << CCTV << "; start time is = " << start_CCTV << "\n";
std::cout << "Internet is = " << Internet << "; start time is = " << start_Internet <<
"\n";

//--------End of Context-aware algorithm-----------------

uint16_t port = 9;
cout << "datarate1= " << datarate1 << "\n";
std::ostringstream strs1;
strs1 << datarate1;
std::string str_datarate1 = strs1.str();
cout << "str_datarate1 =" << str_datarate1+"Mbps" << "\n";

  OnOffHelper onoff1 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff1.SetConstantRate (DataRate (str_datarate1+"Mbps"));
  onoff1.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps1 = onoff1.Install (hosts.Get (0));
  apps1.Start (Seconds (0.0));
  apps1.Stop (Seconds (stop_time_SCADA));

cout << "datarate2= " << datarate2 << "\n";
std::ostringstream strs2;
strs2 << datarate2;
std::string str_datarate2 = strs2.str();
cout << "str_datarate2 =" << str_datarate2+"Mbps" << "\n";

  OnOffHelper onoff2 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff2.SetConstantRate (DataRate (str_datarate2+"Mbps"));
  onoff2.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps2 = onoff2.Install (hosts.Get (1));
  apps2.Start (Seconds (start_SmartMeter));
  apps2.Stop (Seconds (stop_SmartMeter));

cout << "datarate3= " << datarate3 << "\n";
std::ostringstream strs3;
strs3 << datarate3;
std::string str_datarate3 = strs3.str();
cout << "str_datarate3 =" << str_datarate3+"Mbps" << "\n";

  OnOffHelper onoff3 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff3.SetConstantRate (DataRate (str_datarate3+"Mbps"));
  onoff3.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps3 = onoff3.Install (hosts.Get (2));
  apps3.Start (Seconds (start_CCTV));
```

```
  apps3.Stop (Seconds (stop_CCTV));

cout << "datarate4= " << datarate4 << "\n";
std::ostringstream strs4;
strs4 << datarate4;
std::string str_datarate4 = strs4.str();
cout << "str_datarate4 =" << str_datarate4+"Mbps" << "\n";

  OnOffHelper onoff4 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff4.SetConstantRate (DataRate (str_datarate4+"Mbps"));
  onoff4.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps4 = onoff4.Install (hosts.Get (3));
  apps4.Start (Seconds (start_Internet));
  apps4.Stop (Seconds (stop_Internet));

  // Create an optional packet sink to receive these packets
  PacketSinkHelper sink ("ns3::UdpSocketFactory",
  Address (InetSocketAddress (Ipv4Address::GetAny (), port)));
  apps4 = sink.Install (hosts.Get (4));
  apps4.Start (Seconds (0.0));

  // Hosts configuration
  devices = pointToPoint.Install (hosts.Get(0), routers.Get(0));
  address.SetBase ("192.168.1.0", "255.255.255.0");
  address.Assign (devices);

  devices = pointToPoint.Install (hosts.Get(1), routers.Get(0));
  address.SetBase ("192.168.2.0", "255.255.255.0");
  address.Assign (devices);

  devices = pointToPoint.Install (hosts.Get(2), routers.Get(0));
  address.SetBase ("192.168.3.0", "255.255.255.0");
  address.Assign (devices);

  devices = pointToPoint.Install (hosts.Get(3), routers.Get(0));
  address.SetBase ("192.168.4.0", "255.255.255.0");
  address.Assign (devices);

  devices = pointToPoint.Install (routers.Get(2), hosts.Get(4));
  address.SetBase ("192.168.5.0", "255.255.255.0");
  address.Assign(devices);

  // Routers configuration
  devices = pointToPoint.Install (routers.Get(0), routers.Get(1));
  address.SetBase ("10.1.1.0", "255.255.255.0");
  address.Assign (devices);
```

```
devices = pointToPoint.Install (routers.Get(1), routers.Get(2));
address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (devices);

// Address is not specified. Mpls interfaces will be disabled.
devices = pointToPoint.Install (routers.Get(2), routers.Get(0));

NhlfeSelectionPolicyHelper policy;
policy.SetAttribute ("MaxPacketsInTxQueue", IntegerValue (50));

MplsSwitch sw1 (routers.Get (0));
MplsSwitch sw2 (routers.Get (1));
MplsSwitch sw3 (routers.Get (2));

sw1.SetSelectionPolicy (policy);

sw1.AddFtn (Ipv4Source ("192.168.1.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (100), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.1.1"),
   Nhlfe (Swap (110), Ipv4Address ("10.1.3.1"))
);

sw1.AddFtn (Ipv4Source ("192.168.2.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (200), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.2.1"),
   Nhlfe (Swap (210), Ipv4Address ("10.1.3.1"))
);

sw1.AddFtn (Ipv4Source ("192.168.3.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (300), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.3.1"),
   Nhlfe (Swap (310), Ipv4Address ("10.1.3.1"))
);

sw1.AddFtn (Ipv4Source ("192.168.4.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (400), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.4.1"),
   Nhlfe (Swap (410), Ipv4Address ("10.1.3.1"))
);
```

```
sw2.AddIlm (100,
   Nhlfe(Swap (110), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (110,
   Nhlfe(Swap (100), Ipv4Address ("10.1.1.1"))
);

sw2.AddIlm (200,
   Nhlfe(Swap (210), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (210,
   Nhlfe(Swap (200), Ipv4Address ("10.1.1.1"))
);

sw2.AddIlm (300,
   Nhlfe(Swap (310), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (310,
   Nhlfe(Swap (300), Ipv4Address ("10.1.1.1"))
);

sw2.AddIlm (400,
   Nhlfe(Swap (410), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (410,
   Nhlfe(Swap (400), Ipv4Address ("10.1.1.1"))
);

sw3.AddIlm (110,
   Nhlfe(Pop ())
);

sw1.AddIlm (100,
   Nhlfe(Pop ())
);

sw3.AddIlm (210,
   Nhlfe(Pop ())
);

sw1.AddIlm (200,
   Nhlfe(Pop ())
);

sw3.AddIlm (310,
   Nhlfe(Pop ())
);
```

```
  sw1.AddIlm (300,
     Nhlfe(Pop ())
  );

  sw3.AddIlm (410,
     Nhlfe(Pop ())
  );

  sw1.AddIlm (400,
     Nhlfe(Pop ())
  );

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  network.DiscoverNetwork ();

  pointToPoint.EnablePcapAll ("mpls-ppp-example");

//Install FlowMonitor on all nodes
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll ();

 Simulator::Run ();

  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin ();
i != stats.end (); ++i)
    {
      // first 2 FlowIds are for ECHO apps, we don't want to display them
      if (i->first > 0)
        {
          Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
          std::cout << "Flow " << i->first - 0 << " (" << t.sourceAddress << " -> " <<
t.destinationAddress << ")\n";
          std::cout << "  Tx Bytes:   " << i->second.txBytes << "\n";
          std::cout << "  Rx Bytes:   " << i->second.rxBytes << "\n";
          std::cout << "  Tx Packets:   " << i->second.txPackets << "\n";
          std::cout << "  Rx Packets:   " << i->second.rxPackets << "\n";
          std::cout << "  Lost Packets:   " << i->second.txPackets - i->second.rxPackets
<< "\n";
          std::cout << "  Times Forwarded:   " << float(i->second.timesForwarded) <<
"\n";
          std::cout << "  Mean Hop Count:   " << float(i->second.timesForwarded) /
i->second.rxPackets + 1 << "\n";
          std::cout << "  Mean Delay:   " << i->second.delaySum.GetSeconds() /
i->second.rxPackets << "\n";
          std::cout << "  Sum Delay:   " << i->second.delaySum.GetSeconds() << "\n";
```

```
        std::cout << "  Mean Jitter:   " << i->second.jitterSum.GetSeconds() /
(i->second.rxPackets-1)<< "\n";
        std::cout << "  Last Delay:   " << i->second.lastDelay.GetSeconds() << "\n";
        std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / 10.0 / 1000 / 1000
<< " Mbytes\n";
        std::cout << "  Delay Histogram:   " << i->second.delayHistogram.GetNBins ()
<< "\n";
        std::cout << "  Jitter Histogram:   " << i->second.jitterHistogram.GetNBins ()
<< "\n";
        std::cout << "  Packet Size Histogram:   " <<
i->second.packetSizeHistogram.GetNBins ()  << "\n";
        std::cout << "  Flow Interruptions Histogram:   " <<
i->second.flowInterruptionsHistogram.GetNBins ()  << "\n";
      }
    }

  Simulator::Destroy ();

  return 0;
}
```

APPENDIX D

Appendix D : Source code for performance evaluation of CATSchA algorithm without contextual aware

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/mpls-module.h"
#include "ns3/log.h"
#include "ns3/flow-monitor-module.h"

#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace ns3;
using namespace mpls;
using namespace std;

NS_LOG_COMPONENT_DEFINE ("mpls-ppp-example");

int
main (int argc, char *argv[])
{
  CommandLine cmd;
  cmd.Parse (argc, argv);

  Time::SetResolution (Time::NS);

  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("mpls::MplsProtocol", LOG_LEVEL_DEBUG);
  LogComponentEnable ("mpls::Ipv4Routing", LOG_LEVEL_DEBUG);
  LogComponentEnable ("MplsNetworkDiscoverer", LOG_LEVEL_DEBUG);

  NodeContainer hosts;
  NodeContainer routers;
  NodeContainer routers2;

  PointToPointHelper pointToPoint;
  Ipv4AddressHelper address;
  NetDeviceContainer devices;
  InternetStackHelper internet;
  MplsNetworkConfigurator network;
```

```
  hosts.Create (5);
  internet.Install (hosts);
  routers = network.CreateAndInstall (3);

  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("100Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

//------Start of Context-aware algorithm-----------------

int MOQoSA[8][5];

MOQoSA[0][0]=1; MOQoSA[0][1]=1; MOQoSA[0][2]=1; MOQoSA[0][3]=1;
MOQoSA[0][4]=1;
MOQoSA[1][0]=1; MOQoSA[1][1]=1; MOQoSA[1][2]=0; MOQoSA[1][3]=0;
MOQoSA[1][4]=2;
MOQoSA[2][0]=1; MOQoSA[2][1]=0; MOQoSA[2][2]=1; MOQoSA[2][3]=0;
MOQoSA[2][4]=3;
MOQoSA[3][0]=1; MOQoSA[3][1]=0; MOQoSA[3][2]=0; MOQoSA[3][3]=2;
MOQoSA[3][4]=4;
MOQoSA[4][0]=0; MOQoSA[4][1]=1; MOQoSA[4][2]=1; MOQoSA[4][3]=3;
MOQoSA[4][4]=5;
MOQoSA[5][0]=0; MOQoSA[5][1]=1; MOQoSA[5][2]=0; MOQoSA[5][3]=0;
MOQoSA[5][4]=6;
MOQoSA[6][0]=0; MOQoSA[6][1]=0; MOQoSA[6][2]=1; MOQoSA[6][3]=0;
MOQoSA[6][4]=7;
MOQoSA[7][0]=0; MOQoSA[7][1]=0; MOQoSA[7][2]=0; MOQoSA[7][3]=4;
MOQoSA[7][4]=8;

cout << MOQoSA[0][0] << MOQoSA[0][1] << MOQoSA[0][2] << MOQoSA[0][3]
<< MOQoSA[0][4] << "\n";
cout << MOQoSA[1][0] << MOQoSA[1][1] << MOQoSA[1][2] << MOQoSA[1][3]
<< MOQoSA[1][4] << "\n";
cout << MOQoSA[2][0] << MOQoSA[2][1] << MOQoSA[2][2] << MOQoSA[2][3]
<< MOQoSA[2][4] << "\n";
cout << MOQoSA[3][0] << MOQoSA[3][1] << MOQoSA[3][2] << MOQoSA[3][3]
<< MOQoSA[3][4] << "\n";

cout << MOQoSA[4][0] << MOQoSA[4][1] << MOQoSA[4][2] << MOQoSA[4][3]
<< MOQoSA[4][4] << "\n";
cout << MOQoSA[5][0] << MOQoSA[5][1] << MOQoSA[5][2] << MOQoSA[5][3]
<< MOQoSA[5][4] << "\n";
cout << MOQoSA[6][0] << MOQoSA[6][1] << MOQoSA[6][2] << MOQoSA[6][3]
<< MOQoSA[6][4] << "\n";
cout << MOQoSA[7][0] << MOQoSA[7][1] << MOQoSA[7][2] << MOQoSA[7][3]
<< MOQoSA[7][4] << "\n";

//pre-defined priority
int SmartMeter=MOQoSA[1][4];
int CCTV=MOQoSA[6][4];
int Internet=MOQoSA[7][4];
```

```
//threshold values
double start2 = 0; //Smart Meter
double start3 = 0; //CCTV
double start4 = 0; //Internet

double stop_time_SCADA = 0.165;
double stop_time2 = 0.165;
double stop_time3 = 0.165;
double stop_time4 = 0.165;

double gap_time2 = stop_time2-start2;
double gap_time3 = stop_time3-start3;
double gap_time4 = stop_time4-start4;

double SmartMeterTrafficLatency_th=7.5;
double SmartMeterTrafficDataRate_th=0.87;//Mbps
double
SmartMeterTrafficDataSize_th=SmartMeterTrafficDataRate_th/gap_time2;//Mb

double CCTVTrafficLatency_th=2.5;
double CCTVTrafficDataRate_th=3.26;//Mbps
double CCTVTrafficDataSize_th=CCTVTrafficDataRate_th/gap_time3;//Mb

double InternetTrafficLatency_th=8.5;
double InternetTrafficDataRate_th=5.43;//Mbps
double InternetTrafficDataSize_th=InternetTrafficDataRate_th/gap_time4;//Mb

double SCADATrafficDataRate=0.4;//

double SmartMeterTrafficLatency=8;//
double SmartMeterTrafficDataRate=0.8;//
double SmartMeterTrafficDataSize=SmartMeterTrafficDataRate/gap_time2;//

double CCTVTrafficLatency=3;//
double CCTVTrafficDataRate=3.3;//
double CCTVTrafficDataSize=CCTVTrafficDataRate/gap_time3;//

double InternetTrafficLatency=7;//
double InternetTrafficDataRate=5.5;//
double InternetTrafficDataSize=InternetTrafficDataRate/gap_time4;//

std::cout << "Before Context-aware:" << "\n";
std::cout << "SmartMeter is = " << SmartMeter << "\n";
std::cout << "CCTV is = " << CCTV << "\n";
std::cout << "Internet is = " << Internet << "\n";

std::cout << "After Context-aware, before swapping if clash:" << "\n";
std::cout << "SmartMeter is = " << SmartMeter << "\n";
std::cout << "CCTV is = " << CCTV << "\n";
std::cout << "Internet is = " << Internet << "\n";
```

```
//swapping if clash
int orilocSmartMeter=MOQoSA[1][4];
int orilocCCTV=MOQoSA[6][4];
int orilocInternet=MOQoSA[7][4];

if ((SmartMeter==CCTV) && (Internet==CCTV))
{
  SmartMeter=SmartMeter;
  CCTV=orilocSmartMeter;
  Internet=orilocCCTV+1;
  if (SmartMeter==orilocSmartMeter)
  {
    CCTV=orilocSmartMeter;
    SmartMeter=orilocCCTV;
  }
}

if (SmartMeter==CCTV)
{
  SmartMeter=SmartMeter;
  CCTV=orilocSmartMeter;
  if (SmartMeter==orilocSmartMeter)
  {
    CCTV=orilocSmartMeter;
    SmartMeter=orilocCCTV;
  }
}

if (SmartMeter==Internet)
{
  SmartMeter=SmartMeter;
  Internet=orilocSmartMeter;
   if (SmartMeter==orilocSmartMeter)
    {
      Internet=orilocSmartMeter;
      SmartMeter=orilocInternet;
    }
   if (SmartMeter==CCTV)
    {
      SmartMeter=orilocCCTV+1;
      CCTV=CCTV;
    }
}

if (Internet==CCTV)
{
  CCTV=CCTV;
  Internet=orilocCCTV;
  if (CCTV==orilocCCTV)
  {
```

```
    Internet=orilocCCTV;
    CCTV=orilocInternet;
  }

  if (SmartMeter==Internet)
  {
    SmartMeter=orilocInternet-1;
    Internet=Internet;
  }
  if (SmartMeter==CCTV)
  {
    SmartMeter=orilocCCTV+1;
    CCTV=CCTV;
  }
}

//swap time
double start_SmartMeter = start2;
double start_CCTV = start3;
double start_Internet = start4;

double stop_SmartMeter = stop_time2;
double stop_CCTV = stop_time3;
double stop_Internet = stop_time4;

double datarate1 = SCADATrafficDataRate;
double datarate2 = SmartMeterTrafficDataRate;
double datarate3 = CCTVTrafficDataRate;
double datarate4 = InternetTrafficDataRate;

cout << "datarate2= " << datarate2 << "\n";
cout << "datarate3= " << datarate3 << "\n";
cout << "datarate4= " << datarate4 << "\n";

if ((SmartMeter<Internet) && (Internet<CCTV))
{
  std::cout << "Case 2" << "\n";
  start_SmartMeter=start2;
  start_Internet=start3;
  start_CCTV=start4;

  stop_SmartMeter=stop_time2;
  stop_Internet=stop_time3;
  stop_CCTV=stop_time4;

  datarate2 = SmartMeterTrafficDataRate;
  datarate3 = InternetTrafficDataRate;
  datarate4 = CCTVTrafficDataRate;
}
```

```
if ((CCTV<SmartMeter) && (SmartMeter<Internet))
{
  std::cout << "Case 3" << "\n";
  start_CCTV=start2;
  start_SmartMeter=start3;
  start_Internet=start4;

  stop_CCTV=stop_time2;
  stop_SmartMeter=stop_time3;
  stop_Internet=stop_time4;

  datarate2 = CCTVTrafficDataRate;
  datarate3 = SmartMeterTrafficDataRate;
  datarate4 = InternetTrafficDataRate;
}

if ((CCTV<Internet) && (Internet<SmartMeter))
{
  std::cout << "Case 4" << "\n";
  start_CCTV=start2;
  start_Internet=start3;
  start_SmartMeter=start4;

  stop_CCTV=stop_time2;
  stop_Internet=stop_time3;
  stop_SmartMeter=stop_time4;

  datarate2 = CCTVTrafficDataRate;
  datarate3 = InternetTrafficDataRate;
  datarate4 = SmartMeterTrafficDataRate;
}

if ((Internet<SmartMeter) && (SmartMeter<CCTV))
{
  std::cout << "Case 5" << "\n";
  start_Internet=start2;
  start_SmartMeter=start3;
  start_CCTV=start4;

  stop_Internet=stop_time2;
  stop_SmartMeter=stop_time3;
  stop_CCTV=stop_time4;

  datarate2 = InternetTrafficDataRate;
  datarate3 = SmartMeterTrafficDataRate;
  datarate4 = CCTVTrafficDataRate;
}
```

```
if ((Internet<CCTV) && (CCTV<SmartMeter))
{
  std::cout << "Case 6" << "\n";
  start_Internet=start2;
  start_CCTV=start3;
  start_SmartMeter=start4;

  stop_Internet=stop_time2;
  stop_CCTV=stop_time3;
  stop_SmartMeter=stop_time4;

  datarate2 = InternetTrafficDataRate;
  datarate3 = CCTVTrafficDataRate;
  datarate4 = SmartMeterTrafficDataRate;
}

std::cout << "After swapping if clash:" << "\n";
std::cout << "SmartMeter is = " << SmartMeter << "; start time is = " <<
start_SmartMeter << "\n";
std::cout << "CCTV is = " << CCTV << "; start time is = " << start_CCTV << "\n";
std::cout << "Internet is = " << Internet << "; start time is = " << start_Internet <<
"\n";

//--------End of Context-aware algorithm-----------------

uint16_t port = 9;
cout << "datarate1= " << datarate1 << "\n";
std::ostringstream strs1;
strs1 << datarate1;
std::string str_datarate1 = strs1.str();
cout << "str_datarate1 =" << str_datarate1+"Mbps" << "\n";

  OnOffHelper onoff1 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff1.SetConstantRate (DataRate (str_datarate1+"Mbps"));
  onoff1.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps1 = onoff1.Install (hosts.Get (0));
  apps1.Start (Seconds (0.0));
  apps1.Stop (Seconds (stop_time_SCADA));

cout << "datarate2= " << datarate2 << "\n";
std::ostringstream strs2;
strs2 << datarate2;
std::string str_datarate2 = strs2.str();
cout << "str_datarate2 =" << str_datarate2+"Mbps" << "\n";

  OnOffHelper onoff2 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff2.SetConstantRate (DataRate (str_datarate2+"Mbps"));
  onoff2.SetAttribute ("PacketSize", UintegerValue (1024));
```

```
  ApplicationContainer apps2 = onoff2.Install (hosts.Get (1));
  apps2.Start (Seconds (start_SmartMeter));
  apps2.Stop (Seconds (stop_SmartMeter));

cout << "datarate3= " << datarate3 << "\n";
std::ostringstream strs3;
strs3 << datarate3;
std::string str_datarate3 = strs3.str();
cout << "str_datarate3 =" << str_datarate3+"Mbps" << "\n";

  OnOffHelper onoff3 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff3.SetConstantRate (DataRate (str_datarate3+"Mbps"));
  onoff3.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps3 = onoff3.Install (hosts.Get (2));
  apps3.Start (Seconds (start_CCTV));
  apps3.Stop (Seconds (stop_CCTV));

cout << "datarate4= " << datarate4 << "\n";
std::ostringstream strs4;
strs4 << datarate4;
std::string str_datarate4 = strs4.str();
cout << "str_datarate4 =" << str_datarate4+"Mbps" << "\n";

  OnOffHelper onoff4 ("ns3::UdpSocketFactory",
  InetSocketAddress (Ipv4Address ("192.168.5.2"), port));
  onoff4.SetConstantRate (DataRate (str_datarate4+"Mbps"));
  onoff4.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer apps4 = onoff4.Install (hosts.Get (3));
  apps4.Start (Seconds (start_Internet));
  apps4.Stop (Seconds (stop_Internet));

  // Create an optional packet sink to receive these packets
  PacketSinkHelper sink ("ns3::UdpSocketFactory",
  Address (InetSocketAddress (Ipv4Address::GetAny (), port)));
  apps4 = sink.Install (hosts.Get (4));
  apps4.Start (Seconds (0.0));

  // Hosts configuration
  devices = pointToPoint.Install (hosts.Get(0), routers.Get(0));
  address.SetBase ("192.168.1.0", "255.255.255.0");
  address.Assign (devices);

  devices = pointToPoint.Install (hosts.Get(1), routers.Get(0));
  address.SetBase ("192.168.2.0", "255.255.255.0");
  address.Assign (devices);

  devices = pointToPoint.Install (hosts.Get(2), routers.Get(0));
  address.SetBase ("192.168.3.0", "255.255.255.0");
  address.Assign (devices);
```

```
devices = pointToPoint.Install (hosts.Get(3), routers.Get(0));
address.SetBase ("192.168.4.0", "255.255.255.0");
address.Assign (devices);

devices = pointToPoint.Install (routers.Get(2), hosts.Get(4));
address.SetBase ("192.168.5.0", "255.255.255.0");
address.Assign(devices);

// Routers configuration
devices = pointToPoint.Install (routers.Get(0), routers.Get(1));
address.SetBase ("10.1.1.0", "255.255.255.0");
address.Assign (devices);

devices = pointToPoint.Install (routers.Get(1), routers.Get(2));
address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (devices);

// Address is not specified. Mpls interfaces will be disabled.
devices = pointToPoint.Install (routers.Get(2), routers.Get(0));

NhlfeSelectionPolicyHelper policy;
policy.SetAttribute ("MaxPacketsInTxQueue", IntegerValue (50));

MplsSwitch sw1 (routers.Get (0));
MplsSwitch sw2 (routers.Get (1));
MplsSwitch sw3 (routers.Get (2));

sw1.SetSelectionPolicy (policy);

sw1.AddFtn (Ipv4Source ("192.168.1.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (100), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.1.1"),
   Nhlfe (Swap (110), Ipv4Address ("10.1.3.1"))
);

sw1.AddFtn (Ipv4Source ("192.168.2.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (200), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.2.1"),
   Nhlfe (Swap (210), Ipv4Address ("10.1.3.1"))
);

sw1.AddFtn (Ipv4Source ("192.168.3.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (300), Ipv4Address ("10.1.1.2"))
);
```

```
sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.3.1"),
   Nhlfe (Swap (310), Ipv4Address ("10.1.3.1"))
);

sw1.AddFtn (Ipv4Source ("192.168.4.1") && Ipv4Destination ("192.168.5.2"),
   Nhlfe (Swap (400), Ipv4Address ("10.1.1.2"))
);

sw3.AddFtn (Ipv4Source ("192.168.5.2") && Ipv4Destination ("192.168.4.1"),
   Nhlfe (Swap (410), Ipv4Address ("10.1.3.1"))
);

sw2.AddIlm (100,
   Nhlfe(Swap (110), Ipv4Address ("10.1.3.2"))
);

sw2.AddIlm (110,
   Nhlfe(Swap (100), Ipv4Address ("10.1.1.1"))
);

sw2.AddIlm (200,
   Nhlfe(Swap (210), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (210,
   Nhlfe(Swap (200), Ipv4Address ("10.1.1.1"))
);

sw2.AddIlm (300,
   Nhlfe(Swap (310), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (310,
   Nhlfe(Swap (300), Ipv4Address ("10.1.1.1"))
);

sw2.AddIlm (400,
   Nhlfe(Swap (410), Ipv4Address ("10.1.3.2"))
);
sw2.AddIlm (410,
   Nhlfe(Swap (400), Ipv4Address ("10.1.1.1"))
);

sw3.AddIlm (110,
   Nhlfe(Pop ())
);

sw1.AddIlm (100,
   Nhlfe(Pop ())
);
```

```
  sw3.AddIlm (210,
     Nhlfe(Pop ())
  );

  sw1.AddIlm (200,
     Nhlfe(Pop ())
  );

  sw3.AddIlm (310,
     Nhlfe(Pop ())
  );

  sw1.AddIlm (300,
     Nhlfe(Pop ())
  );

  sw3.AddIlm (410,
     Nhlfe(Pop ())
  );

  sw1.AddIlm (400,
     Nhlfe(Pop ())
  );

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  network.DiscoverNetwork ();

  pointToPoint.EnablePcapAll ("mpls-ppp-example");

//Install FlowMonitor on all nodes
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll ();

 Simulator::Run ();

  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin ();
i != stats.end (); ++i)
    {
     // first 2 FlowIds are for ECHO apps, we don't want to display them
     if (i->first > 0)
       {
        Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
        std::cout << "Flow " << i->first - 0 << " (" << t.sourceAddress << " -> " <<
t.destinationAddress << ")\n";
        std::cout << "  Tx Bytes:   " << i->second.txBytes << "\n";
```

```
        std::cout << "  Rx Bytes:   " << i->second.rxBytes << "\n";
        std::cout << "  Tx Packets:   " << i->second.txPackets << "\n";
        std::cout << "  Rx Packets:   " << i->second.rxPackets << "\n";
        std::cout << "  Lost Packets:   " << i->second.txPackets - i->second.rxPackets
<< "\n";
        std::cout << "  Times Forwarded:   " << float(i->second.timesForwarded) <<
"\n";
        std::cout << "  Mean Hop Count:   " << float(i->second.timesForwarded) /
i->second.rxPackets + 1 << "\n";
        std::cout << "  Mean Delay:   " << i->second.delaySum.GetSeconds() /
i->second.rxPackets << "\n";
        std::cout << "  Sum Delay:   " << i->second.delaySum.GetSeconds() << "\n";
        std::cout << "  Mean Jitter:   " << i->second.jitterSum.GetSeconds() /
(i->second.rxPackets-1)<< "\n";
        std::cout << "  Last Delay:   " << i->second.lastDelay.GetSeconds() << "\n";
        std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / 10.0 / 1000 / 1000
<< " Mbytes\n";
        std::cout << "  Delay Histogram:   " << i->second.delayHistogram.GetNBins ()
<< "\n";
        std::cout << "  Jitter Histogram:   " << i->second.jitterHistogram.GetNBins ()
<< "\n";
        std::cout << "  Packet Size Histogram:   " <<
i->second.packetSizeHistogram.GetNBins ()  << "\n";
        std::cout << "  Flow Interruptions Histogram:   " <<
i->second.flowInterruptionsHistogram.GetNBins ()  << "\n";
      }
    }

  Simulator::Destroy ();

  return 0;
}
```