UNIVERSITÉ DE MONTRÉAL

# QOS-AWARE AND STATUS-AWARE ADAPTIVE RESOURCE ALLOCATION FRAMEWORK IN SDN-BASED IOT MIDDLEWARE

ATEFEH MESHINCHI
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
MAI 2018

QOS-AWARE AND STATUS-AWARE ADAPTIVE RESOURCE ALLOCATION FRAMEWORK IN SDN-BASED IOT MIDDLEWARE

# DEDICATION

*I dedicate my thesis to my husband, Hossein, who has been a constant source of support and encouragement during the challenges of graduate school. I am thankful for having you in my life.*

## AKNOWLEDGEMENTS

I would like to offer my sincere gratitude to my research supervisor, Professor Alejandro Quintero, and my co-advisor, Professor Yann-Gaël Guéhéneuc, for their support and guidance to successfully complete this project, and the opportunity to study under their supervision. I would like to thank Professor Samuel Pierre and Professor Martine Bellaïche for taking part in my thesis committee. Also, I would like to express my thanks to my husband, my parents, and my colleagues at LARIM(Mobile computing and networking research laboratory) who have supported me.

## RÉSUMÉ

«L’Internet des objets (IdO) est une infrastructure mondiale pour la société de l’information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l’information et de la communication interopérables existantes ou en évolution. »[1]

La vision de l’Internet des Objets est d’étendre l’Internet dans nos vies quotidiennes afin d’améliorer la qualité de vie des personnes, de sorte que le nombre d’appareils connectés et d’applications innovantes augmente très rapidement pour amener l’intelligence dans différents secteurs comme la ville, le transport ou la santé. En 2020, les études affirment que les appareils connectés à Internet devraient compter entre 26 milliards et 50 milliards d’unités. [2, 3]

La qualité de service d’application IoT dépend non seulement du réseau Internet et de l’infrastructure de communication, mais aussi du fonctionnement et des performances des appareils IoT. Par conséquent, les nouveaux paramètres de QoS tels que la précision des données et la disponibilité des appareils deviennent importants pour les applications IoT par rapport aux applications Internet.

Le grand nombre de dispositifs et d’applications IoT connectés à Internet, et le flux de trafic spontané entre eux rendent la gestion de la qualité de service complexe à travers l’infrastructure Internet. D’un autre côté, les dispositifs non-IP et leurs capacités limitées en termes d’énergie et de transmission créent l’environnement dynamique et contraint. De plus, l’interconnexion de bout en bout entre les dispositifs et les applications n’est pas possible. Aussi, les applications sont intéressées par les données collectées, pas à la source spécifique qui les produit.

Le Software Defined Networking (SDN) est un nouveau paradigme pour les réseaux informatiques apparu récemment pour cacher la complexité de l’architecture de réseau traditionnelle (par exemple de l’Internet) et briser la fermeture des systèmes de réseau dans les fonctions de contrôle et de données. Il permet aux propriétaires et aux administrateurs de réseau de contrôler et de gérer le comportement du réseau par programme, en découplant le plan de contrôle du plan de données. SDN a le potentiel de révolutionner les réseaux informatiques classiques existants, en offrant plusieurs avantages tels que la gestion centralisée, la programmabilité du réseau, l’efficacité des coûts d’exploitation, et les innovations.

Dans cette thèse, nous étudions la gestion de ressources sur l’infrastructure IoT, y compris les réseaux de transport/Internet et de détection. Nous profitons de la technologie SDN comme le futur d'Internet pour offrir un système de support QoS flexible et adaptatif pour les services IoT. Nous présentons un intergiciel basé sur SDN pour définir un cadre de gestion de QoS pour gérer les besoins spécifiques de chaque application à travers l'infrastructure IoT. De plus, nous proposons un nouveau modèle QoS qui prend en compte les préférences de QoS des applications et l'état des éléments de réseau pour allouer efficacement les ressources sur le réseau transport/Internet basé sur SDN tout en maximisant les performances du réseau.

## ABSTRACT

The Internet of Things (IoT) is an integration of various kinds of technologies, wherein heterogeneous objects with capabilities of sensing, actuation, communication, computation, networking, and storage are rapidly developed to collect the data for the users and applications. The IoT vision is to extend the Internet into our everyday lives, so the number of connected devices and innovative applications are growing very fast to bring intelligence into as many domains as possible.

The QoS for IoT application not only depends on the Internet network and communication infrastructure, it is also impacted by the operation and performance of IoT sensing infrastructure. Therefore, the new QoS parameters such as data accuracy, sampling rate, and device availability become important for the IoT applications compared to the Internet applications. The huge number of the Internet-connected IoT devices and application, and the spontaneous traffic flow among them make the management of the quality of service complex across the Internet infrastructure. On the other hand, the non-IP devices and their limited capabilities in terms of energy and transmission create the dynamic environment and hinder the direct interaction between devices and applications.

The quality of service is becoming one of the critical non-functional IoT element which needs research and studies. A flexible and scalable QoS management mechanism must be implemented in IoT system to keep up with the growth rate of the Internet-connected IoT devices and applications as well as their heterogeneity and diversity. The solution should address the IoT application requirements and user satisfaction while considering the system dynamism, limitations, and characteristics.

Software-Defined Networking (SDN) is an emerging paradigm in computer networking which separates the control plane and the data plane of the network elements. It makes the network elements programmable via the centralized control plane. This approach enables more agile management and control over the network behavior.

In this thesis, we take advantage of SDN technology as the future of the Internet to offer a flexible and adaptive QoS support scheme for the IoT services. We present an SDN-based middleware to define a QoS management framework to manage the application specific QoS needs across the IoT infrastructure including transport and sensing network. Also, we propose a new QoS model that takes into account the application QoS preferences and the network elements status to allocate effectively the resources for the applications across SDN network while maximizing network performance.

# TABLE OF THE CONTENTS

# LIST OF THE TABLES

# LIST OF THE FIGURES

## TABLE OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| IoT | Internet of Things |
| SDN | Software Defined Networking |
| QoS | Quality of Service |
| WSN | Wireless Sensor Network |
| SLA | Service Level Agreement |
| 6LoWPAN | Internet Protocol(IPv6) and Low-power Wireless Personal Area Networks |
| M2M | Machine to Machine |
| CAPEX | Capital Expenditure |
| OPEX | Operating Expenses |
| RFID | Radio-Frequency IDentification |
| BW | Bandwidth |

# LIST OF APPENDICES

# CHAPTER 1 INTRODUCTION

The Internet of Things (IoT) represents the future of the Internet. Referring to ITU-T and IERC [1, 16], the IoT, first introduced in 1999, creates an Internet-based interconnected platform wherein smart objects have interfaces and identities. These smart objects can communicate with one another through standard and interoperable communication protocols without human interference. Smart objects with sensing, interaction, communication, computing, and decision-making capabilities, using locally- or globally-gathered data, are going to make IoT happen.

IoT employs all technologies in the field of communication, data, application, and device to improve the efficiency and lower the cost of business processes. By 2020, IoT studies argue that Internet-connected devices are expected to number between 26 billion and 50 billion [2, 3]. Therefore, IoT is going to cover all the objects in our environment and it is expected to have a significant impact on all aspects of daily life to provide various benefits by connecting objects together in support of intelligent decision making. Figure 1.1 shows the vision of the IoT which is the creation of the smart environments such as Smart City, Smart Home, Smart Health, and Smart Transport.

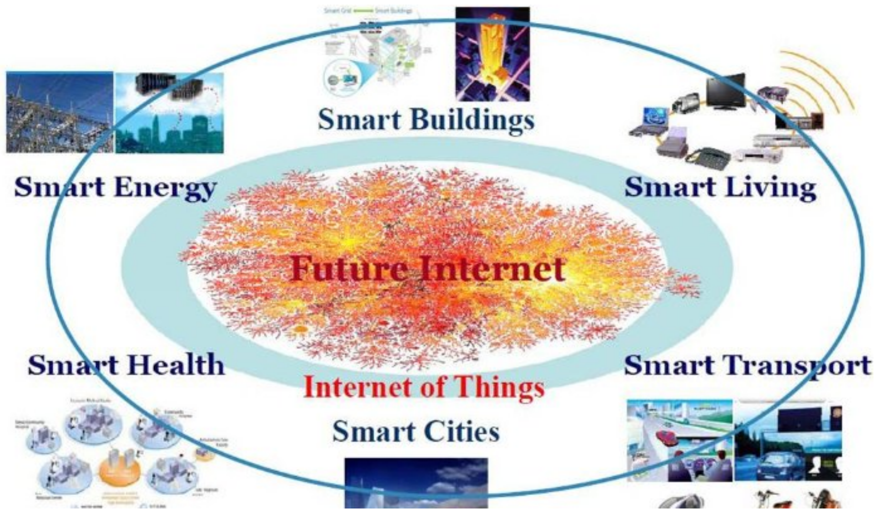


Figure 1.1 IoT Strategic Research Roadmap [5]

For example, in the context of the Smart City, information and communication technologies provide the critical infrastructure and services for city administration, transportation, education, health-care, public safety, and utilities. The connected infrastructure and efficient services empower the creation of more intelligent and innovative living and working environment.[16]

In IoT, The smart sensors produce a large volume of data that must be processed and analyzed to be transformed into useful output. Therefore, IoT offers a new class of applications, also called data-centric application, which harvest data from the environment and transform them into desired information for users. For example, a smart traffic management application monitors the vehicles and pedestrians across a city to optimize driving and walking routes, and video surveillance applications monitor the environment to identify suspicious activities.

To work properly and effectively, IoT applications present different QoS requirements comparing with Internet applications. Their concerns are not only the speed of communication and error rate, but they have other concerns such as data accuracy, coverage, and the limitation of the sensing resources, which could be the major problem hindering their effectiveness. [17]

QoS management in IoT systems is a very complex task because of the extremely large variety of devices and services, as well as the technologies and techniques involved in the systems architecture. Therefore, multiple systems, such as devices, data, and communication are involved in the fulfillment of the service and accordingly, the Quality of services. QoS must not only be embedded in the design of all individual component, it must be considered in the system architecture design and the cross-layer interactions among components and technologies as well.

Wireless Sensor Networks (WSNs), widely used in IoT infrastructure, are comprised of hundreds or thousands of sensor nodes, and these sensors have the ability to communicate either among each other or directly to an external base station(BS). These sensors are generally low-end with strong constraints in energy, processing, storage, and transmission capabilities. Besides, implementation of IP stack in such low-end devices is too complex, so most of the sensors does not support IP addressing scheme. Therefore, the direct interaction between the applications and the sensors could not be established. In this case, IoT-gateway as a powerful device acts as a bridge between IP-based systems and non-IP addressable sensors. Although, with the introduction and development of the 6LoWPAN (Internet Protocol (IPv6) and Low-power Wireless Personal Area Networks) technologies [18], a low-power sensor network could communicate with other IP-based systems directly(e.g. an application server, storage server), sensors still suffer from resource and capability limitation. Hence, Wireless Sensor Networks (WSNs) differ significantly from traditional wireless networks with respect to the network and device architecture, traffic characteristics, scale, and design goals.

The Internet as a large-scale networking system has had great success in the interconnection of computer networks and with the creation of IPv6, it extends the TCP/IP address identifiers and offers plenty of advanced capability in terms of security, connectivity, and scalability.

Therefore, Internet world-wide availability plus IP stack capability added to smart objects make the Internet the best available choice as transport and communication infrastructure in IoT system [19]. However, the legacy computer network and the Internet still face some limitations. On one hand, the control intelligence, which is implemented by various routing and management protocols, is embedded in every network elements (e.g., router, switch) and is difficult to change. Vendor-dependent platforms and interfaces make the Internet evolution complex and slow. On the other hand, the Internet provides best-effort services and it is not capable to meet the more specific requirements of applications in terms of service quality. In the IoT, heterogeneous networks and devices create opportunities for a wide range of applications with varying QoS requirements, which cannot be guaranteed by the best-effort Internet mechanisms. Moreover, the number of connected devices and the amount of generated data will become an increasing stress on the Internet network.

As a conclusion, the organization and management of a large number of devices and applications require novel ideas in the design of the IoT to enhance the system performance and the quality of provided services. Although the limitations imposed by the technologies integrated into IoT system( e.g., WSNs and Internet) must be considered in the design of the QoS management mechanisms, they need to be evolved and adapted to the IoT characteristics and IoT application-specific needs to make system services useful and effective.

Software Defined Networking (SDN) is a new paradigm for computer networking that appeared recently to hide the complexity of traditional networking architecture (e.g., the Internet) and break down the closeness of network systems in both control and data functions. It allows the network owners and administrators to programmatically initialize, control, and manage network behavior ; by decoupling the control plane from the data plane. SDN has the potential to revolutionize existing classical computer networks, by offering several benefits such as the centralized management, network programmability, operation and capital cost efficiency, and inherent innovations. [10].

Therefore, SDN is a potential solution to the problems faced by traditional computer networks such as the core transport network and the Internet. Network device producers are working together on standardization process around protocols and features in the Open Networking Foundation(ONF). SDN technical characteristics and the success stories extend the SDN use cases to the networking segments of the data-centers and cloud environment. For example, Google has deployed a software-defined networking to interconnect its data-centers across the globe and provide workload optimization in its distributed systems [20].

In recent years, software-defined system and the idea of having a centralized control layer gained more interest in the research community to solve the challenges related to the closeness of the different type of the access and communication networks, such as 5G networks [21], VANET [22], and WSN [23]. It is also desirable to integrate SDN into the IoT to utilize its capabilities to simplify the control and management process in terms of data storage, data communication, and data security [13].

To ensure an acceptable level of QoS for applications specifically mission-critical application, several QoS approaches have been proposed in any IoT layer or subsystem. In this thesis, we focus on the impact of IoT network elements, which are the Internet ( also called Core Transport network) and sensing networks, on the quality of services expected by IoT application. We aim to provide a centralized and unified QoS management service by integrating SDN technology into IoT architecture. This approach could be classified into the middleware-based approaches since the SDN control layer is going to provide the abstraction layer between the IoT infrastructure and the application layer. Resource allocation decision is a fundamental requirement of the QoS and performance management framework. The main question addressed in this work is how resource allocation can be matched to the IoT application QoS needs while considering the resource capabilities and limitations. Therefore, the goal is to build an adaptive and flexible QoS framework which could keep pace with dynamic business and application requirements.

## 1.1 Definitions and basic concepts

This section presents background information to provide a thorough understanding of the basic concepts of the Internet of Things (IoT), Software Defined Networking (SDN), and Quality of Service (QoS).

### 1.1.1 Internet of Things architecture

For a common understanding of the IoT, we present definitions of the main building blocks of the IoT. The common IoT architecture illustrated in Figure 1.2 comprises multiple layers: device layer, network and communication layer, service support layer, and application layer. Also, security and system management are vital parts of the architecture and must be considered in the design of all these layers to provide reliable, secure, extensible, and flexible systems. [2, 16]

— The device layer, also called sensing layer, includes all IoT devices, sensors, and actuators. Sensors are sensing and gathering data from real-world objects, machines, and people. Actuators perform actions based on the sensed data or the requests from users. Wireless Sensor Networks (WSNs)and RFID are the main competences of IoT sensing

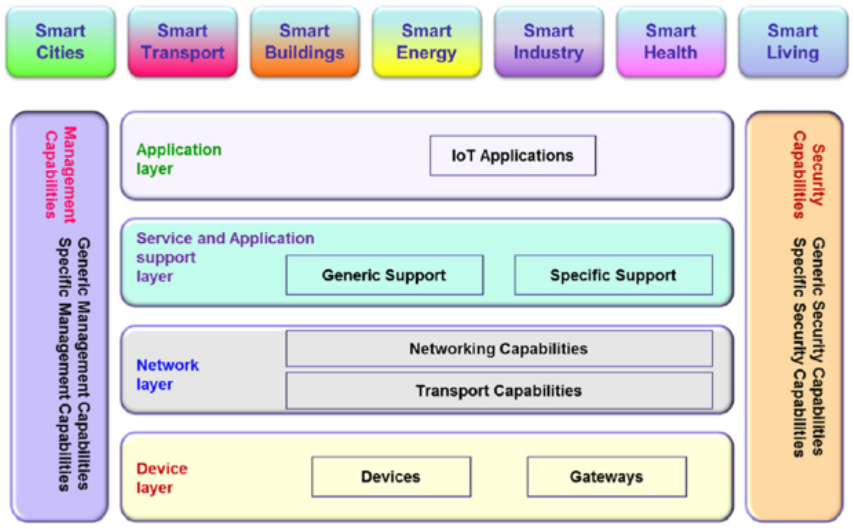


Figure 1.2 IoT Architectural Reference Model [6]

layer.

- The network and communication layer, also called access and transmission network, securely transfers the data from sensor devices to information processing systems and applications, and the opposite direction from the application to the sensing layer. Internet could provide the transmission infrastructure and other technologies such as 5G, LTE, 3G, Wi-Fi, Bluetooth, and Zig-Bee, depending upon the IoT devices and their capabilities could be used as the communication network and access network to the Internet.
- The service support layer, or middle-ware layer, includes all functional services to provide an inter-operable and context-aware communication among heterogeneous IoT devices, and between devices and applications. This layer also performs service management, data storage, and information processing between applications and sensor devices.
- The application layer provides global management of IoT applications that are the consumers of the data with the purpose of in-depth data analysis, data computation, and action creation depending on business goals. The applications form the user-interface of the IoT. They are essential for a proper utilization of the collected data.

Managing all devices, and their communication and interaction with the application layer of IoT architecture and keeping track of the failures, configurations, and performance are definitely challenging in such complex system. Therefore, management capabilities in terms of fault, configuration, accounting, performance, and security are another aspects of IoT, which can be found in almost all layers, are critical to being addressed in IoT system design.

All technologies, in terms of device, communication, connectivity, mobility, database, and application are involved in designing the IoT. Academic and industrial researchers are working on solving IoT-specific issues and adapting the existing technologies to be appropriate for the IoT. For example, to improve the scalability and performance of the IoT, some technologies, such as Cloud Computing and Fog Computing, are converging with the IoT. Cloud Computing [16, 24] offers unlimited storage, computation, and networking capabilities. It makes cross-domain application opportunities possible and offers fine-grained IoT resources in a pay-as-you-use manner. Fog Computing [25], as an extension of cloud computing, improves the response time to the end users and provides near real-time experience by moving service provisioning and data processing to the edge of the network instead of entirely in the cloud. Additionally, it also reduces data flow volume through the Internet.

### 1.1.2 Software Defined Networking(SDN) architecture

Software Defined Networking has been introduced by University of California at Berkeley and Stanford University in 2008. The Open Networking Foundation (ONF) [10] is a non-profit and user organization dedicated to the promotion and adoption of SDN through the creation of software-defined networking standards and protocols.

In current networking devices (see Figure 1.3), the control plane, which is responsible for traffic handling, and the forwarding plane, also called data plane, which forwards the packets based on control plane's decision, are installed and tied together. In such vertically networking devices, the closeness of the planes reduces flexibility and hinders the innovation and evolution of the networking infrastructure, because deploying new services, resource optimization mechanisms, and traffic differentiation methods are bundled with hardware and vendor-specific interfaces.

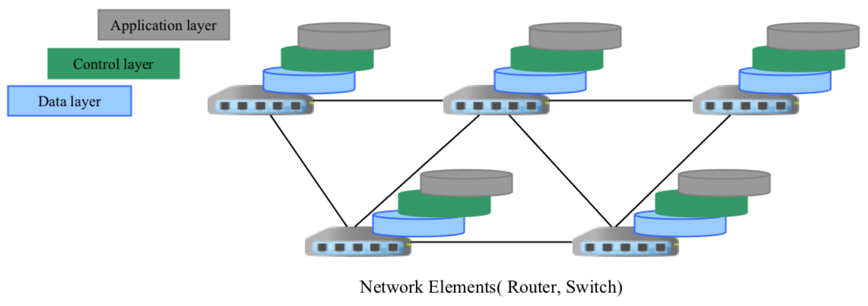


Figure 1.3 Traditional network architecture

SDN decouples the control plane from the data plane to enhance the network evolution, interoperability, and scalability. SDN architecture depicted in Figure 1.4 is made of three logical layers [7]:

— The data plane layer, which includes the network infrastructure, and is composed of physical forwarding devices, such as switches and routers.

— The control plane, which includes the centralized networking controller, supervises all network traffic and makes decisions about where the traffic must be forwarded using a global view of the network.

— The application layer, which represents the services that interact with the controller to specify the networking needs of the applications in terms of security, configuration, and management. Network operators and administrators can directly write and deploy customized services and make them operational through the controller without depending on the vendor-dependent releases or updates.

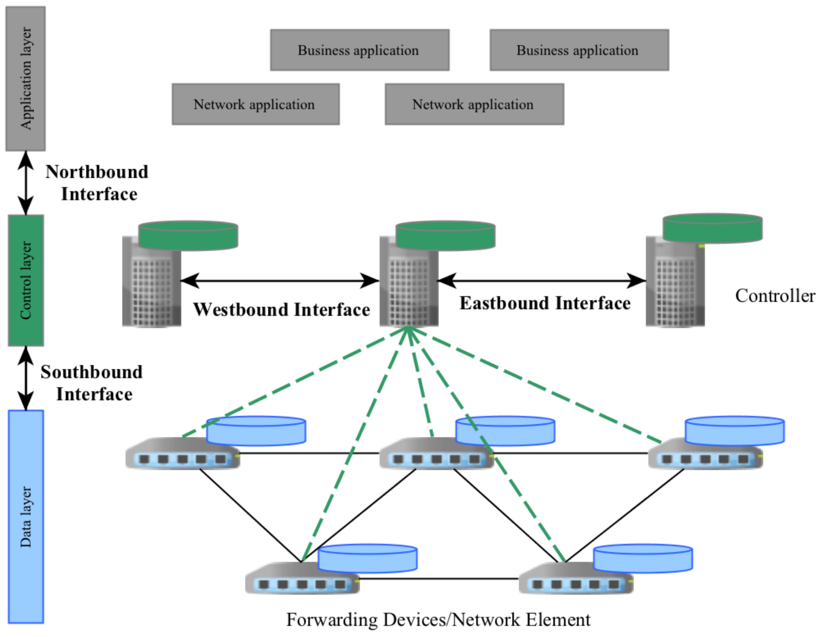


Figure 1.4 SDN architecture [7]

The communication between the forwarding devices and controller is done through newly designed interfaces known as *Southbound interfaces*. A controller exercises direct control over the states of the data-plane devices via well-defined Application Programming Interfaces (APIs). OpenFlow [26] is the first standard southbound interface.

The control layer communicates with the upper layer, application layer, through a *Northbound interface*. The SDN controller provides services to the application layer through APIs that allow implementing customized applications in terms of routing, security, traffic engineering, and policy management.

Supporting APIs in SDN hide the complexity and heterogeneity of the physical infrastructure and allow sharing physical infrastructure which significantly reduces the difficulty of application development and shortens the time to market of new applications. *East/westbound interfaces* are a special case of interfaces required by distributed controllers. They are used to interconnect the SDN architecture with external SDN-based network architectures or the legacy networks. [7]

### 1.1.3 Quality of Service (QoS)

Quality is defined by ISO 9001 as "The totality of features and characteristics of a product or service bear on its ability to satisfy stated or implied needs" [27]. QoS in the communication network refers to "the capability to provide assurance that the service requirements of applications can be satisfied." [28]

It is a group of indicators that reflect the properties of services. Therefore, the service provider must consider the applications and users preferences in terms of QoS to provide useful and adaptable services and resources. In general, QoS indicators are included in Service Level Agreement (SLA) [29]. SLA is a formal negotiated agreement between service providers and customers and it can cover many aspects of their relationship such as the performance of services, customer care, billing, and provisioning. For performance, the service providers must monitor SLA-based QoS indicators to verify whether the offered services meet the agreed level of service quality and, if the SLA is broken, they will be penalized.

SLAs are widely used in the IP-based networks and the Internet services. The common functional QoS indicators are throughput, loss rate, and response time, while non-functional indicators contain non-measurable parameters, such as availability, reliability, delivery, and sustainability [30].

SLAs become increasingly important with the IoT and it is essential to maintain and manage SLAs of IoT networks to ensure the service quality received by the consumer devices. According to the IoT architecture (see Figure 1.2), the application and service layers directly interact with user requirements while the network layer helps to transmit the data to the upper layer and the device layer, which is responsible for sensing and collecting data. Therefore, SLAs for IoT include user and application expectations relative to network, devices, and data. It is necessary to clarify the QoS indicators in each of IoT layers and determine the interrelations among them.

**Application and service layer.** The QoS of applications and services, which are perceived directly by customers, may have many possible indicators. The main QoS properties from the application point of view, generally agreed in SLA, are as follows: [31]

— Service time, defined as the time interval between accepting a service request and finishing of it.

— Service cost, defined as the price to be paid due to the service usage.

— Availability, denoting the probability that the service is accessible at some point in time.

— Reliability, also called service accuracy, denoting the probability that a request is correctly served within the expected time, providing the appropriate service information.

**Network layer.** User expectations of the access and core transport network service providers are not much different from that of the Internet-based services [32]:

— Bandwidth (Bit rate), defined as the rate of successful carrying the traffic by the network.

— Latency, defined as the time needed for one packet to reach its destination.

— Jitter, defined as the variation in the delays of received packets related to the same traffic flow.

— Packet loss ratio, determined as the percentage of packets discarded by the network devices or lost with respect to packets sent.

**Device and sensing layer.** The definition of the QoS and metrics to evaluate the performance of wireless sensor network (WSNs) are different than that for traditional networks. QoS parameters depend on the sensing applications. The common QoS indicators are: [31, 33]:

— Quality of Information(QoI)/Quality of Data (QoD), defined as the accuracy and sensitivity of the measurements provided by physical sensors, which depends on the quality of the hardware and sensor capabilities.

— Bandwidth, measured as the available data transmitting-receiving rate for sensors/devices, which depends on the communication technologies supported by the sensors.

— Sampling rate, defined as the frequency of sampling, which affects bandwidth and storage utilization ratio. Efficient data collection process boosts performance and scalability.

— Network lifetime, which is impacted by sensor energy consumption while sensing, processing, and transmitting.

— Coverage, defined as the area range where the sensing layer is working effectively.

— Delay which refers to delay in data collection from sensors.

## 1.2 Aspects of the problem

Applications are fundamental to the IoT. They provide the presentation layer for the data captured from the billions of devices around the world. IoT application could be classified from different perspectives such as the type of information they manage, the type of recipient (person or system oriented), and their criticality. In general, IoT applications are classified into three different types: (1) control applications, also called mission-critical applications, such as city-traffic management and emergency management, which need very fast response with as less error as possible, (2) monitoring applications, such as intelligent security surveillance tasks, which are fed by cameras and need more throughput, and (3) analysis and inquiry applications, such as the inquiry into the transported item state in the intelligent logistics, which are throughput and delay tolerant.

Although IoT can be seen as an Internet connecting things, it is not only the Internet. IoT boosts its own attributes and applications. The level of services not only depends on the communication network (e.g., Internet), it also depends on the performance of the sensing network as the sources of the collected data. Thus, some new QoS parameters such as data accuracy and sampling rate are desired from the application point of view to be measured in the service delivery. Therefore, multiple elements such as the Internet and WSNs are contributing to the quality of services.

QoS management on the core transport network and Internet seeks to provide optimal performance for computer networks while ensuring data flow at an acceptable level of quality. Performance optimization and Traffic Engineering (TE) within computer networks have been studied for many years. Many effective solutions in terms of traffic classification, traffic scheduling, prioritization, and resource reservation have been proposed and widely implemented in the computer networking [34]. Traffic Engineering (TE) methods intend to manage the network traffic and design optimized resource allocation and routing mechanisms to improve resource utilization while better meeting the requirement of the network QoS. Two main standardized QoS implementation models in classical IP networks are Integrated Services (IntServ) [35], with the idea of per-flow resource reservation, and Differentiated Services (DiffServ) [36] with the idea of traffic classification and prioritization. While both models are offering advanced features in TE, but the scalability and robustness of IntServ approach and lack of end-to-end connection guarantee and per-flow QoS setup in DiffServ are the drawbacks of the two approaches. These limitations are not suitable for the IoT system since the Internet of Things (IoT) is placing new demands on network infrastructure due to the diverse application domain (e.g., smart health, smart city). Under different contexts and domains, IoT applications could adopt different classification methods and QoS policies compared with the current web applications.

On the other hand, the characteristics of WSNs are very different from other conventional networks. For example, sensor nodes may read data at different rates or they may generate redundant data. Besides, WSNs are highly constrained in terms of battery, bandwidth, storage and communication capabilities. Sensors may fail or be blocked due to lack of energy, physical damage, or interference. Therefore, the QoS management must address the WSN specific challenges as well as energy conservation to schedule the tasks and allocate resources for the different type of IoT applications. Although several efficient protocols, middleware-based QoS models, and QoS routing mechanisms have been proposed in the literature [37], there is still ongoing research to provide more optimized QoS control and management framework within WSN.

The heterogeneous nature of the IoT systems and a wide variety of application domains, as well as the fast growth of IoT devices and applications, make QoS management a complex challenge. Multiple service providers, such as network service providers, sensing service providers, and storage service providers, are involved in maintaining the QoS and different SLAs could be established between the IoT customers and service providers. Moreover, multiple WSN providers or network transport service providers might contribute together to extend IoT coverage and have overlapping area providing same services but with different level of quality and service cost depending on the capabilities of the system and components. Besides, there are a large number of QoS factors which can be taken into account in the IoT environment. Therefore, the IoT makes the network control and resource allocation more complicated than for the Internet.

In the literature, several QoS models and approaches have been proposed for IoT system, each attacks different architectural layer and subsystem and covers one or multiple quality factors. Despite all research and studies in QoS management within IoT network elements(sensing and communication network), still, they need enhanced solutions to be adapted to IoT specific characteristics and QoS requirements. Besides, we remarked that IoT system lacks the standardized and unified QoS support service framework which considers the inherent stochastic and dynamic nature of IoT. The successful framework must provide the QoS across multiple IoT systems based on the user and application expectation and must be flexible, extensible, and scalable to pace the system evolution.

Application SLA enforcement in the service delivery and QoS-aware resource allocation in the IoT framework is the main motivation for this thesis. Our approach is to provide a middleware-based QoS support service for the IoT application. We propose a flexible and dynamic resource allocation scheme that directly applies the SLA-related application QoS requirements and the infrastructure resource status in the resource allocation process. Having the resource status and application preferences create an adaptive best-fit resource allocation over time. To achieve this, we leverage the capabilities introduced by SDN technology and integrate it into the IoT architecture. So, the main contribution of our work is the proposition of the scalable and adaptive QoS management framework which support the diverse and dynamic QoS requirements of the application from the sensing and transport network.

Also, our framework provides the centralized and unified way to apply the SLA across the IoT network infrastructure. Therefore, it abstracts the complexity of the SLA enforcement in the infrastructure resources. It rapidly adapts to the dynamic business goals and SLA changes without any complex reconfiguration and policy definition of the infrastructure resources.

## 1.3 Research objectives

Our main goal is to design a QoS management framework for the IoT application. We aim to manage the IoT infrastructure resources composed of Internet/transport network and sensing network devices to provide QoS support services across the IoT layered architecture. To accomplish this goal, we integrate the SDN technology into IoT architecture to use its characteristics and features to implement a QoS-aware resource allocation for various IoT application with different QoS needs. Thus, this thesis has the following goals:

- Propose an appropriate QoS support framework to enforce the diverse application SLA and improve network resource allocation mechanism for the IoT.
- Model QoS in SDN-based core transport network/Internet to guarantee application QoS in terms of packet loss, delay, and bandwidth, taking into account application preferences and network constraints.
- Analyze and evaluate the performance of the proposed QoS model through experiments.

## 1.4 Outline

This thesis describes a new architecture that allows enhanced SLA-aware QoS support in the IoT by taking advantage of SDN integration into the network and communication layer of IoT architecture. The architecture is comprised of several modules deployed in different layers of the IoT to make the required information accessible and to provide dynamic network resource management. The remainder of the thesis is structured as follows:

— Chapter 2 presents the several approaches and methods introduced by previous works which investigate different aspects of QoS in IoT systems such as architecture, application, and network. This chapter also provides a general view of the Software Defined Networking(SDN) as the future of the networking paradigm. It is followed by the ideas and prototypes employing SDN techniques in IoT environments.

— Chapter 3 provides the proposed QoS support IoT architecture, taking advantages of SDN based transport networking, and describes all relevant components.

— Chapter 4 details the mathematical model provided for QoS support routing across SDN-based transport network, based on a multi-criteria approach.

— Chapter 5 presents our experimental design and the numerical results obtained from the performance assessment of the offered model.

— Chapter 6 concludes the thesis and suggests some limitations and directions which could be attacked as the future work.

# CHAPTER 2 RELATED WORK AND OVERVIEW

IoT has attracted significant attention from research community both in academia and industry. The basic idea of IoT is to have sensors and smart devices (commonly called “Connected devices”), are designed in such a way that they capture the data and utilize them to complete tasks on several aspects of everyday life. IoT world is growing at a surprising speed, from 2 billion objects in 2006 to a projected 20 billion by 2020 [38].

Research communities investigate different aspects of IoT system and they present surveys of IoT issues and challenges to be solved such as security and privacy, architectural issue, big data as well as energy efficiency and QoS (Quality of Service)[16]. The Internet of Things (IoT) continues to evolve and expand in terms of domains, interconnected-devices, data, and application, so the challenges are becoming more complex. From QoS point of view, IoT presents several issues such as availability, reliability, mobility, performance, scalability, and interoperability. [2, 16, 39, 40]

The performance of IoT services is a big challenge because IoT architecture is made of multiple layers, so the service performance depends on the performance of many components that need continuous development and improvement to meet the requirements of the system and applications. In the literature, there are various conceptual and technological ideas to improve the IoT system performance and the quality of the services. This chapter presents the state-of-the-art of approaches and solutions proposed to enhance the IoT service quality, either attacking one particular IoT component or the IoT architectural design.

Software Defined Networking(SDN) has been regarded as a future networking paradigm to improve the flexibility and scalability of networks by decoupling the control plane from the data plane. Later, we briefly introduce the SDN standards and its characteristics, more specifically we highlight the QoS management within SDN architecture and the most-valuable academic papers in this regard.

Although lots of prior research has exposed the potential benefits of applying SDN in computer networks to facilitate the evolution of network technologies, there has been a few studies about how to apply the SDN to the management of the other networking systems such as telecommunication, wireless networks, and IoT. In the end, we discuss some of the relevant research which aims to bring the openness in the wireless networks and IoT, addressing the system specific complexities which hinder their evolution. Considering the benefits and weaknesses of all current approaches and solutions, the last section draws a conclusion.

## 2.1 Quality of Service approaches in IoT system

In IoT environment, smart things are able to have interaction and communication capabilities with each other and with the environment. They sense, collect, exchange data, and react to the real-world events in an autonomous way without human intervention. A huge number of applications are going to be deployed to make use of sensing data and information collected by thousands of IoT-devices.

The authors in [16, 41] present surveys on IoT vision, technologies, applications and the research issues in IoT. However, with the rapid development of technologies and consequently, with the growth of the sensing devices and connected things all challenges and issues are getting more bigger and complex [40]. Academia research communities and enterprises work on IoT-specific challenging issues and try to solve all technological barriers.

The IoT vision is to provide intelligent services in various domains such as Transport, Health, Environment, Building, and City. The basic IoT application model is demonstrated in Figure 2.1. Therefore, it deals with different kind of applications and traffic, each accepts a specific quality level of performances to operate appropriately. To provide QoS in the IoT, it is necessary to ensure suitable mechanisms at each layer of the IoT since various applications could have the dependency on the different QoS attributes which must be provided by a specific IoT layer.

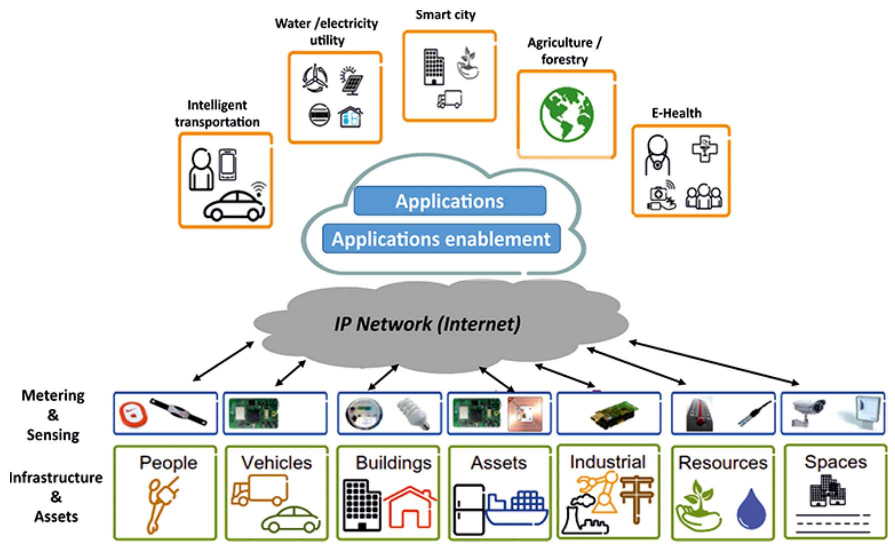


Figure 2.1 Simplified IoT application model [8]

QoS management as one of the critical subsystem needs research and investigation. The solution must consider volume and the variety of the devices, the diversity of the application domains and their QoS needs, and multi-layer IoT architecture n the design of the IoT system to provide a stable, robust and scalable QoS framework.

QoS requirements can be investigated from the application and user perspective, or from the networking perspective in terms of reliability, timeliness, robustness, trustworthiness, and adaptability. The applications are just concerned how the services provided by the network impact the quality of the application. Various applications and users drive the specific needs with a specific level of performance. The feasibility and practicality of application could be derived from one or multiple of these qualitative attributes. From the network point of view, the goal is to provide the agreed services for the applications while maximizing network resource utilization.

In the previous chapter, we defined the main QoS attributes in different layers of IoT architecture. QoS parameters are important in the performance evaluation of the service, they should be considered in function of the quality perceived by the end-user and as a fulfillment of Service Level Agreements.

The specific architecture would depend on the IoT application domains and the enabling technologies used in specific implementations. Unlike Internet, Internet of Things is not a single technology and it is compounded with many functions including sensing, processing, transmission, analysis, and deciding. So, many different technologies in terms of hardware, software, data, and communication are involved in the different layers of IoT architecture to implement the smart environments. Therefore, Quality of Service not only must be embedded in the production and design of all hardware and software components in IoT infrastructure, their integration into IoT system might raise needs for adaption and adjustment with the dynamic nature of IoT and application requirements. In other words, the overall IoT service quality and end-user satisfaction depend on various service providers like sensing service, network service, and cloud and it is carried out by all involved technologies and components of IoT service.

Some of the IoT enabling technologies like the Internet and the cellular access networks (e.g. 3G, LTE) have the evolved QoS functions within their closed systems, although their integration into IoT bring issues and limitations. With the realization of the 5G technology, the barriers of the access networks are rectified, since 5G provides less-delay high-bandwidth access network for the IoT system. The Internet imposes some limitations in terms of the supported QoS mechanisms. Compared with the web application, IoT applications have dynamic and data-centric nature and they could have diverse QoS needs. Therefore, current QoS differentiation mechanism could not meet the QoS needs of the diverse and progressive of the IoT applications.

Another subsystem such as WSN needs more research and investigation. WSNs have very limited resources in terms of energy, processing, and memory. Additionally, they are made of heterogeneous nodes and often operate in the dynamic and unpredictable environment. Thus, meeting QoS requirements in WSNs would be difficult. Improving energy efficiency, bandwidth, storage optimization, coverage in IoT and accuracy as some of the critical QoS parameters. Several papers focus on the improvement of the WSN QoS attributes through embedding QoS-aware mechanisms in the aspects of protocols, routing, topology control and data acquisition to optimize the energy consumption and increase the network lifetime. Accordingly, network availability is increased for the associated applications [37]. In the context of the QoS-aware resource allocation which is the focused subject of our work, Sequential Assignment Routing (SAR) [42] as one of the first protocols for wireless sensor networks provides the notion of QoS routing criteria. In this routing algorithm, each sensor considers several parameters such as the packet priority, the energy resources, and the QoS metrics (delay and energy cost) in the path selection process to achieve energy efficiency and fault tolerance. SPEED (Stateless Protocol for End-to-End Delay) [43] as the next QoS-based routing protocol uses the geographic information to estimate the delay of the transmitted packets and selects the sensor node which meets the application speed requirement. The proposed protocol in [44] provides an energy efficient and a least cost path to meet the end-to-end delay requirement within intra-network connection for the real-time traffic generated by video or imaging sensors. Some efforts have argued the sensor selection based on a trade-off between application-perceived benefit and energy consumption of the selected sensor set. Utility-based sensor selection [45] enables sensor network applications to express utilities associated with retrieving data from sets of sensors, a price-based resource management scheme [46] provides the self-organized framework aiming to balance between the sensors profit and power consumption. That far, we note that these papers cover the QoS management aspects within individual IoT sensing layer. Authors in [47] present the computational QoS model for the QoS routing within WSNs which considers the response time, reliability, and availability using the directed graph theory. The proposed mathematical model has been implemented for different service composition modes: serial, parallel, branch, and circulation.

Various research communities and academic organizations have attempted to define the QoS schemes and QoS architectures based on the careful study and understanding of service components, enabling technologies, data classification, application domain areas, and interactions between each of these modules. Jin et al [48] analyze the design approach, connectivity model, in-network processing, and QoS complexity within several types of the network architecture including autonomous network, ubiquitous network and service-oriented network architecture in a smart city, but they do not provide in-depth analysis of the QoS issues. Irfan et al. [49] do the investigation into the different kinds of traffic with various QoS requirements and different level of sensitivity. They provide an analytical model to do priority-based traffic scheduling in a finite-capacity queuing system and evaluate the performance of the model for the delay-sensitive traffic comparing with low-priority traffic. Li et al. [31] propose a QoS-aware service scheduling model using three-layer service-oriented IoT architecture as the basis. They model the QoS optimization problem as a Markov decision process (MDP) in the application layer. They model the cost minimization problem at network layer using decision making algorithm/programming method. Similarly, they use the sensing ability and QoS requirement of the application to model an optimal sensing-cost decision at sensing layer.

Other papers focus on the use of the QoS monitoring and evaluation techniques. [33] defines an IoT architecture composed of three layers: sensing, networking, and application ; in each layer, QoS monitoring function is implemented to measure the QoS parameters. This structure introduces the broker-based interaction between the QoS functions across the layer. Agents resided in each layer take the QoS requirements from the upper layer and provide it for the lower layer, and vice versa to take the feedback from the lower layer components and pass them to the upper layer. This is shown in Figure 2.2.

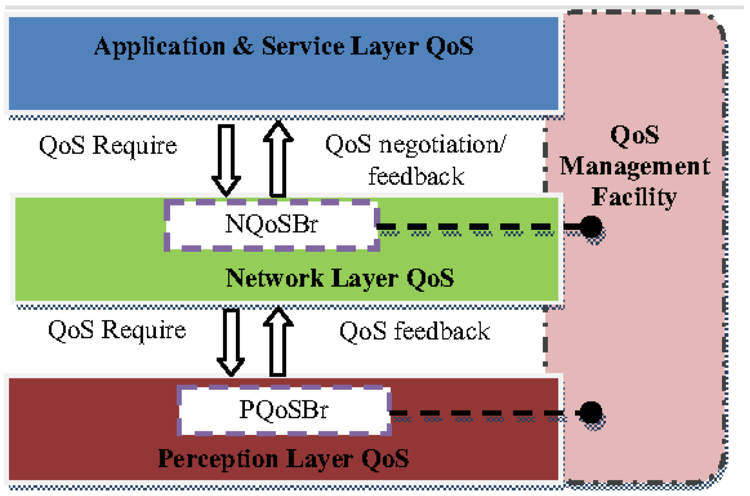


Figure 2.2 QoS architecture of IoT

Recently, one of the promising approaches in the IoT QoS management is the middleware-based mechanism. Middleware could provide access to the heterogeneous resources and support interoperability within diverse applications. In [9], Heinzelman proposes that the application sends the QoS requirement to the middleware and middleware configures the networks and the devices to meet the application expectations. This middleware approach enables the adaptation of the code allocation on the basis of the current application requirements. If the QoS requirements from the application are not feasible to fulfill in the network, the middleware may negotiate a new QoS guarantee with both the application and the network. (see Figure 2.3)

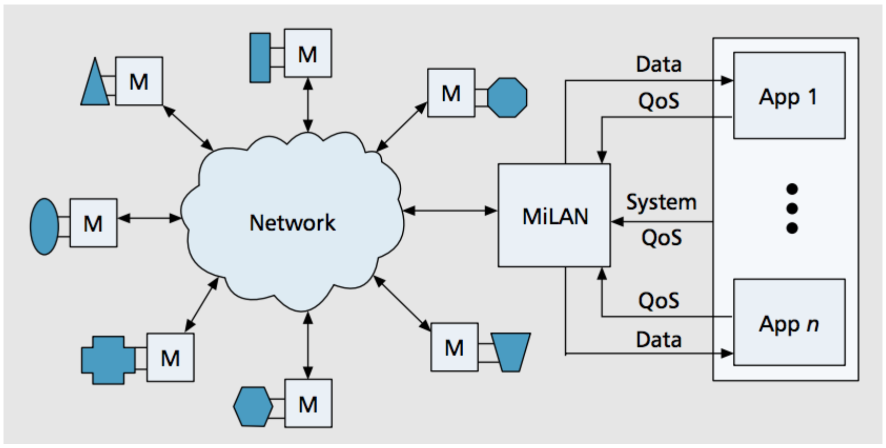


Figure 2.3 A system that employs MiLAN [9]

Therefore, we conclude that most of the QoS schemes and approaches proposed in the literature concentrate on one or multiple QoS factor within a particular subsystem such as WSN. They seek the optimized hardware design, protocol and decision-making algorithms for a particular QoS issue. Very limited works have tried to figure out the implementation of the QoS framework considering the dynamic and diverse nature of the IoT in terms of the application and device as well as their limitations and constraints.

Current research trends suggest the power of middleware to ease the application development task in complex environments. Inspired by the relevant research, we propose the middleware-based IoT QoS architecture based on the SDN technology. In the following section, we describe more detail information about the SDN operation mechanism and its features. Moreover, we insist on the advantages of the SDN architecture to design the QoS support techniques and algorithms within the Internet and core computing networks.

## 2.2 Software Defined Networking (SDN)

Software-Defined Networking (SDN) is an emerging network paradigm that gives hope to change the restriction of current network infrastructures by separation of control plane and data plane of network elements. SDN technology enables the possibility of network control by a software application and makes the network management more efficient, quick, and flexible.

SDN as an open technology improves the interoperability between multi-vendor products and removes vendor lock-in state which it is common drawbacks in the traditional network. So, it decreases the overhead of network element configuration and troubleshooting, leading to the optimized CAPEX and OPEX for IT enterprises.

In the following subsections, first, we give an overview of SDN standard protocol and the way of operation. Further, the major papers and efforts in the field of QoS support within SDN are highlighted.

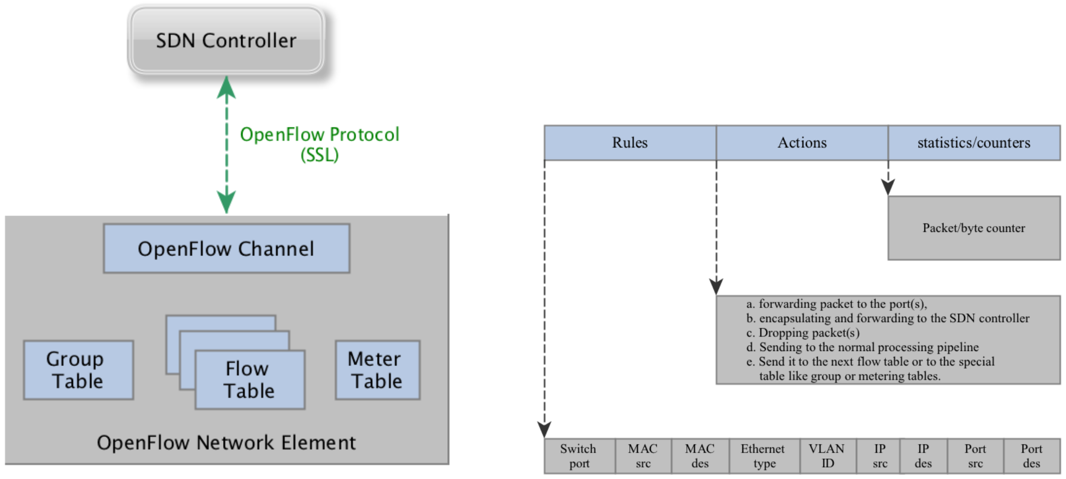
### 2.2.1 OpenFlow Protocol

OpenFlow is the first standard communication interface defined between the controller and forwarding device. It provides the direct access to the forwarding plane of network devices and makes their manipulation and configuration possible. Accordingly, the controller as the control plane element must support the OpenFlow protocol to be able to understand the contents of the OpenFlow messages and to convey the instructions to the data plane on how to forward data.[10]

The ONF (Open Networking Foundation) manages the OpenFlow standard. Version 1.1.0 of the OpenFlow protocol was released on February 28, 2011. More recent OpenFlow version is 1.5.1 released in April 2015. At the beginning, the OpenFlow protocol was developed at the Stanford University around 2008 for enabling researchers to run experimental protocols in the campus networks [26], but recently OpenFlow is added as a feature to the commercial network devices (called “OpenFlow-enabled devices”) as the interface to access the Ethernet switches, routers, and wireless access points. The OpenFlow protocol defines the requirements of OpenFlow-enabled devices (based on OpenFlow protocol specification 1.4.0).

The main components of the OpenFlow-enabled devices are an OpenFlow channel, a Group Table, a Meter Table and one or more Flow Tables (see Figure 2.4a). The OpenFlow channel is the interface that connects each network element to an external controller providing secure SSL channel or direct over TCP. The external controller uses the OpenFlow protocol to manage one or more OpenFlow elements. The Flow Tables and the Group Table are responsible for performing packet lookups and packet forwarding. Noting that multiple Flow Tables could be defined in one network element, and each one could have multiple numbers of flow entries. Multiple flow entries could be grouped in Group Table to point to as one group. Meter-table enables OpenFlow to implement simple QoS features like traffic shaping and rate-limiting. [10]

In OpenFlow protocol, a set of defined messages are exchanged between the controller and



(a) (b)

Figure 2.4 (a)OpenFlow network element components (b)Flow Table entry format (Reproduced from [10])

the forwarding devices over a secure channel. OpenFlow channel supports three types of messages: Controller-to-Switch, Asynchronous, Symmetric. We provide the definition of the messages and some of the important sub-messages because we take advantage of them in designing our proposed architecture. [4]

— Controller-to-Switch is initiated by the controller to manage or query the state of the switch. The message may or may not require a response from the switch.

  — *Packet-out*: Sent by the controller in response to packet-In message to inform switch to forward a packet on a specific interface

  — *Modify-state*: Used to add/delete/modify entries in the Flow Table.

  — *Read-state*: Used to collect various information from switch such as configuration and statistics.

  — *Features request/reply*: Used to request switch capability.

  — *Configuration request/reply*: Used to request switch configuration.

— Asynchronous message re-initiated by the switch to denote packet arrival, switch state change or error.

  — *Packet-In*: Sent by a switch when a packet needs to be processed by the controller.

  — *Flow-removed*: Sent by a switch to inform the controller that flow entry is deleted because of timer expiry.

    - *Port Status*: Sent by a switch when a port configuration or state changes.
    - *Flow-Monitor*: Sent to inform the controller of the change in the Flow Table.
- Symmetric messages can be sent by either the controller or the switch, without solicitation.
    - *Hello*: Used to exchange information between a switch and associated controller when a switch comes up, controller learns about the switch from Hello packet.
    - *Echo Request/Reply*: Sent from either the switch or the controller to verify the liveness of the OpenFlow-connection and must return an echo reply.
    - *Error*: Used by switch or controller to notify problems to the other side of the connection.

### 2.2.2 SDN operation mechanism

In SDN, the OpenFlow devices have no idea how packets are to be routed. The control logic inside the controller is responsible to express the packet behavior and then the detailed configuration information is placed in the forwarding Flow Tables in every device along the path. As described in the previous subsection, OpenFlow-enabled devices could consist of multiple Flow Table. These tables are the important part of the OpenFlow devices. They are used to determine what action should be taken based on receiving packets.

Each Flow Table has multiple flow entry. Each entry is associated with actions to be applied to a certain flow. The flow entry consists of three fields as shown in Figure 2.4b:

- *Header field or rule field* which is used to define the match condition to an exact flow.
- *Actions* which define the action or network element behavior to be applied to a specific flow.
- *Counters* which collect statistics for the particular flow, such as the number of received packets, number of bytes, and duration of the flow.

When a packet arrives at an OpenFlow-enabled network element, the network element looks at the existing entries in the Flow Tables, so the action is executed if the header field is matched and the value of the counters are updated. If there is no matching rule in the Flow Table, it will be sent to the controller over the secure channel. The controller then decides on whether to forward the packet or drop it by creating new flow rules and inserting them in the Flow Table of the relevant network elements, considering that the Flow Table has a priority associated with every entry ; a higher number means a higher priority and the matched flow entry with the higher priority is used.

Depicted in Figure 2.5, the Flow Tables of an OpenFlow-enabled element are sequentially numbered, starting at 0. Pipeline processing mechanism specifies how the packets have to interact with each Flow Table and always starts at the first Flow Table (Flow Table 0). A flow entry can only direct a packet to a Flow Table number which is greater than its own Flow Table number. In other words, pipeline processing can only go forward and not backward. If a packet does not match a flow entry in a Flow Table, this is called *Table Miss* which the device sends the packet to the controller and asks for the appropriate actions. [4]

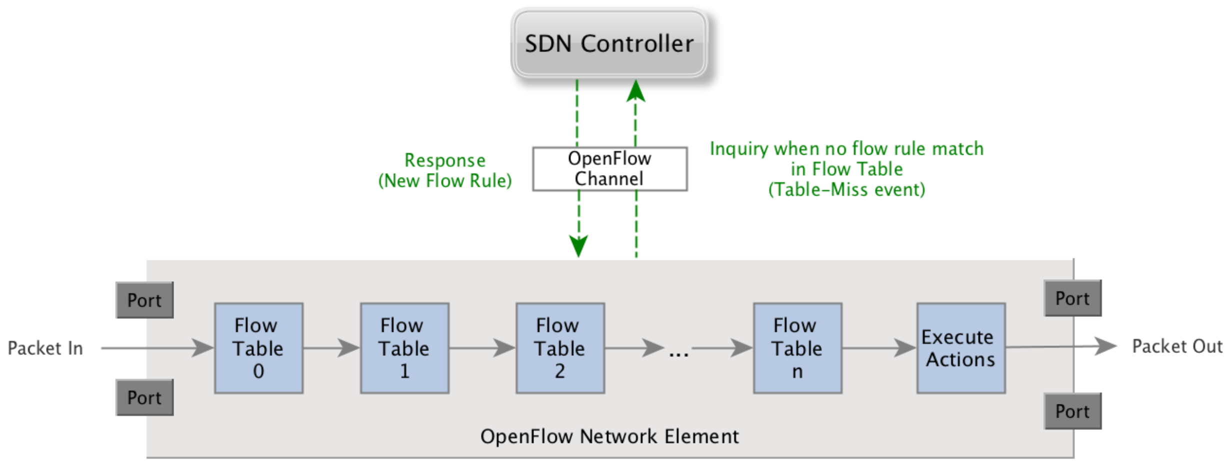


Figure 2.5 How an OpenFlow-enabled element handles incoming packets (Reproduced from [10])

OpenFlow controller can modify the content of flow-tables in two ways: (1) reactively, where the controller reacts when receiving the packet-in message from OpenFlow-enabled network elements because of non-matched rule for arrival packet. (2) pro-actively, where the controller takes the initiative based on the programs and configurations given to the controller and adds rules before packet arrival into the network based on details like network topology change, or increasing the utilization in specific parts of the network, indicating potential forthcoming performance bottlenecks. The major advantages of proactive mode are that all packets are forwarded based on one centralized decision maker dynamically and no delay is added. The hybrid model as the third way exploits the advantages of both reactive and proactive mode so that it follows the flexibility of reactive mode for a set of traffic and low-latency forwarding for the rest of the traffic. [4]

### 2.2.3 QoS management in SDN

As described in section 2.2.1 one of the main components of OpenFlow-enabled network elements is Meter Table which consists of the meter entries. Each meter entry is composed of three fields: Meter ID, Meter Band, and Counters ( see Figure 2.4a). The functionality of Meter Table is to measure and control the rate of the packets directed to the Meter Table based on a defined rule in Flow Table and assigned Meter-ID. Meter Band specifies the rate of the band and the way to process the packet, and Counters keep track of the number of the packets processed by a given meter. Therefore, the simple QoS operations like rate-limiting per-flow basis can be implemented using Meter Table. Comparing to the queuing which accepts packets for output and processes them at a min/max specified rate, meters can be installed, modified, and removed at runtime using OpenFlow messages. Meter Table could be defined in OpenFlow 1.3 and upper versions as an optional feature since the OpenFlow protocol specification does not contain any required meter band types.

However, ONF is actively enhancing the QoS support mechanisms in OpenFlow, SDN allows for a centralized control with a global network view and a feedback control with information exchanged between different layers (application and forwarding layers) in the network architecture. As such, many challenging performance optimization problems would become manageable and automated with properly designed centralized algorithms because of the flexibility and programmability of the controller. Therefore, more complex QoS management functions like Traffic Engineering (TE), Load Balancing (LB), and even the user-customized QoS mechanisms need to be implemented as a program or SDN application and then be enforced in the network infrastructure devices through the controller. For these reasons, SDN technology becomes a promising solution for IT enterprises since it makes QoS more agile which the future Internet demands it.

There have been many studies in the QoS management and Traffic Engineering(TE) over the software-defined networking. They aim to implement the generic traffic optimization techniques and some propose the innovative QoS strategies which address the specific aspects of traffic control in SDN. All of them deal with the techniques of the flow management, topology update, and traffic characterization. [50, 51]

Authors in [52] classify the QoS frameworks using SDN in four main categories: resource reservation, per-flow routing, queuing management, and policy enforcement. They review existing proposals and solutions in each class. For instance, Seddiki et al. [53] propose an SDN-based QoS framework, also called FlowQoS, managing the QoS in the home broadband networks. This framework comprises two components: a flow classifier mapping application traffic to different parts of flow spaces and a rate shaper. Policycop [54] is an automated and extensible QoS policy management framework. It validates and enforces QoS policies which are used in the route decision process in the control layer. QoSFlow [55] enables the packets reordering in the queues of OpenFlow-enabled elements by using Hierarchical Token Bucket, priority-based flow scheduling based on Stochastic Fairness Queuing(SFQ), and Congestion avoidance by using Randomly Early Detection to improve the QoS control capability in SDN. Egilmez et al. in [56] propose dynamic QoS routing mechanism which guarantees the service delivery on an optimal path for multimedia flows, such as VoIP or video streaming. Similarly, Yan et al. [57] propose a QoS-guarantee solution in the SDN, called HiQoS. The HiQoS identifies multiple paths between the source and destination nodes by queuing mechanisms to guarantee QoS for different types of traffic. Experimental results show that the HiQoS scheme provides better performance for the delay-centric application.

In sum, all of the proposed techniques rely on new network applications to implement the control logic which will be translated into commands and installed in the data plane, dictating the behavior of the forwarding devices.

Since the SDN controller has a centralized view of the network topology and the OpenFlow channel supports the statistical information on the basis of the flow and port by the counter filed activated in the Flow Table, Group Table, and Meter Table, the novel routing algorithms intend to employ the link level information to provide the more optimized paths between the network elements. Counters collect the particular information about the flow, port, table, and queue based on the packets or bytes received/transmitted. Several counters must be implemented in all OpenFlow elements, also called “required counters”. “Optional counters” are implemented depending on their user-cases on the specific network application and provide more depth statistical information. Main counters are listed in Table 2.1.

Table 2.1 “Required” counters in OpenFlow [4]

| **Per Port Counters** | **Per Flow Counters** |
|---|---|
| Received packets | Received packets |
| Transmitted packets | Transmitted packets |
| Received bytes | Duration (s, ms) |
| Transmitted bytes | **Per Queue Counters** |
| Received drops | Transmitted packets |
| Transmitted drops | Transmitted bytes |
| Received errors | **Per Table Counters** |
| Transmitted errors | Active entries |
|  | Packet lookups |
|  | Packet matches |

However, the existing SDN challenges in terms of scalability, security, interoperability, and performance have been rectified in a certain degree and it has been evolving very fast with new features, still, research in both academia and industry is going on to extend the OpenFlow protocol and simplify the implementation of novel creative programs. [58]

OpenFlow interface has recently been embraced by the network element manufactures to be integrated into their products and many SDN controllers have been developed by the product vendors such as Big Switch Networks, HP, IBM, VMWare, and Juniper. The most widespread controller in the industrial enterprises is Open Daylight [59] written in Java, and in the research section are Floodlight [60] and Ryu [61] written in Java and Python, respectively.

Recently, the flexibility and simplicity of the SDN architecture have attracted a lot of attention and approaches inspired by Software Defined Networking (SDN) represent a very promising research direction in all kind of the networking paradigm. In the following section, we are going to point to the works which intend to enter SDN concept into the wireless networks and IoT system.

## 2.3 Software-Defined IoT system

Although much prior research has exposed the potential benefits of applying SDN in wired networks, because of increasing its success and reputation, the interest in exploring and adapting SDN design in the wireless networks and IoT system is growing in the literature and industry. Therefore, several papers [62, 63, 64] are trying to bring the concept of softwarization in IoT ecosystem, and others are arguing on the SDN application in the other systems such as mobile wireless networks and wireless sensor networks.

The appearance of IoT and a need for a flexible architecture to manage a large amount of traffic in the wireless network and to promote rapid innovation in the network motivate the formation of Software Defined Wireless Network (SDWN)[65]. Orienting towards the characteristics of the mobile and wireless network, SDWN aims to study the network architecture and a series of relatively crucial technologies for the future mobile and wireless network. The SDWN architecture depicted in Figure 2.6 is supposed to benefit from all the network entities, network operators, service providers, and end users through efficient resource management, mobility management, QoE (Quality of Experience) improvement. The main vision is to provide a unified control plane to manage dynamic and heterogeneous wireless technologies such as WiFi, WiMAX, 3G, LTE, and 5G.

For instances, OpenRoad [66] as the first work in this field proposes an OpenFlow-based open and backward compatible wireless network infrastructure. OpenRoad uses the SDN

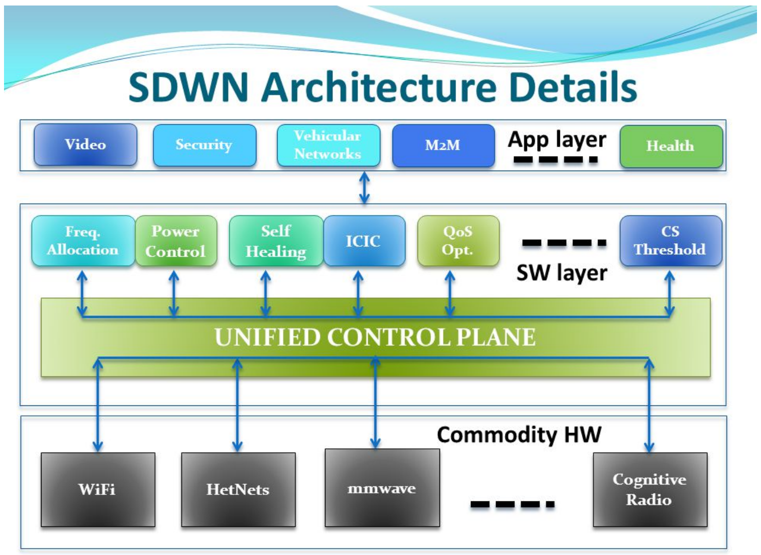
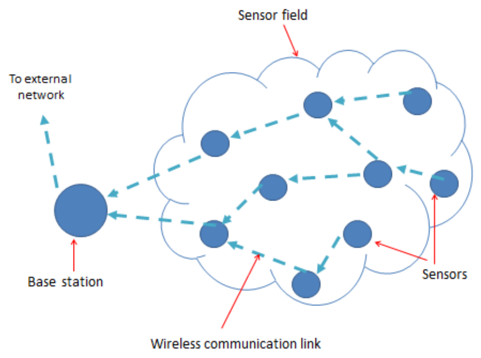


Figure 2.6 Architecture of Software Defined Wireless Networks (SDWN) (source: Stanford wireless systems lab)

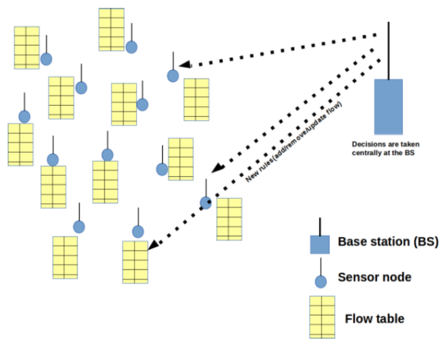
characteristics to support multiple co-located technologies like LTE, WiFi, or WiMAX to support a seamless user mobility. OpenRadio [67] proposes a programmable wireless data plane which makes it flexible and adaptable to the new application. The main idea behind it is to support systematically different protocols in the wireless network while optimizing operation across all of them due to the network hardware in-dependency for any future evolution. SoftRAN [68] uses a software-defined centralized control plane to abstract a set of base-stations as a single virtual base-station that in principle could provide a near-optimal radio resource allocation, however communication delay between data and control plane is an issue in this design. Moreover, [69, 70] are the more recent papers with the focus on mobility management leveraging SDN architecture and its application within heterogeneous wireless environments in data flow basis.

Wireless Sensor Networks (WSNs) differ substantially from wireless networks with respect to network and node architecture, resource capability, traffic characteristics, scale, and design goals. Recent significant research on wireless sensor networks (WSNs) has led to the widespread adoption of software-defined WSNs (SDWSNs) [23, 71], which seeks to solve the inherent issues present in WSNs by separating the control and data layer. The objective is to provide the WSNs reconfiguration capability even after deployment. Although data-centric nature of WSNs, no support of TCP/IP stack and out of band signaling mechanism make SDN concept integration too challenging, a few exploratory studies have been done in this space.

WSNs architecture relies on one or more centralized base station/sink to schedule the sensor-based tasks and gather the data (Shown in Figure 2.7a). In the software-defined sensor network framework shown in Figure 2.7b, the centralized control logic is supposed to be implemented in the base station. The centralized controller manages tasks such as routing, QoS, mobility management, and localization in more efficient and flexible way. The sensor energy also could be optimized better through the centralized management point. [12]



(a)


New rules(add/remove/update flow)
Decisions are taken centrally at the BS
Base station (BS)
Sensor node
Flow table

(b)

Figure 2.7 (a)WSN architecture [11], (b)Software-Defined Wireless Sensor Network architecture [12]

SDN-WISE [72, 73] as an extension of SDWSN, provides a flexible vendor-independent policy implementation in WSN. The other papers have argued about multi-tasking WSNs scheduling (QoSen) [74], energy-optimized Quality-of-Sensing resource allocation [75], and the efficient sensor clustering [76] through softwarization of WSNs. The performance studies in these solutions show an SDN-based architecture could improve network scalability and stability as well as flexibility to define more fine-grained policies, operational tasks, and security control methods. Though the integration of the SDN-based wireless networks could profit the IoT system, these studies are still in the primary steps. And more efforts and consideration need in the way toward the productive implementation phases.

Recently, research domain subjected as SD-IoT aims to integrate SDN into IoT framework to improve the system control and management. The SD-IoT framework model in [13] offers as centralized control system over security services(SDSec), storage services (SDS) and infrastructure resources (see Figure 2.8). The authors argue that SDSec can provide visibility of all the traffic flows in the network, which gives the possibility to detect suspicious traffic using automated policies and fine-grained analyzing process. The data gathered by the sensors is sent to the SDSec controller for security checking, authentication, and authorization. Controlling rules and policies defined by the SDN controller on the basis of collected data are stored in SDStore module. Motivated by this proposal, few studies [77, 78] try to improve the resource utilization in terms of data acquisition, transmission, and processing within IoT framework. UbiFlow by Di. WU et al [79] proposes an efficient flow control and mobility management in urban multi-networks using SDN distributed controllers. In UbiFlow, IoT network is partitioned into the small network. Each network is controlled by a physically distributed SDN controller. The IoT devices in each network could connect to the different access point depending on their positions.

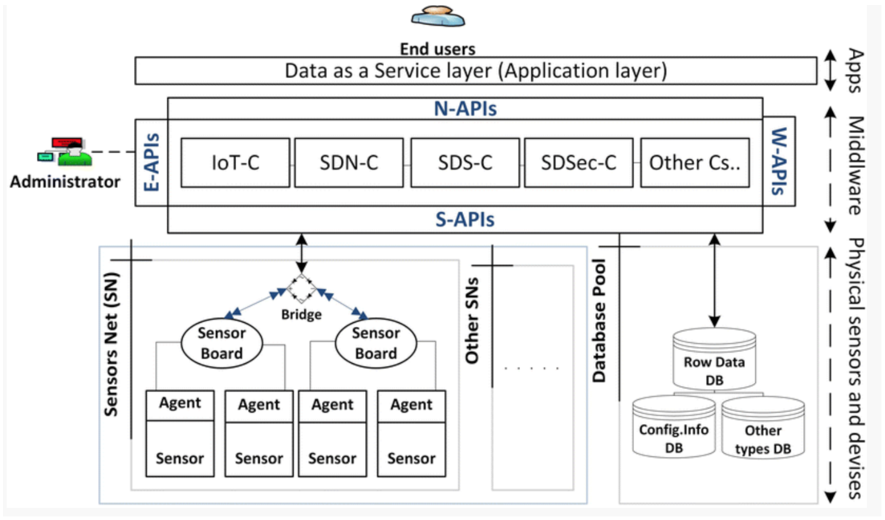


Figure 2.8 The proposed SDIoT architecture design [13]

However all of these efforts argue that SDN can facilitate the system and resource management in the WSN, the existing architecture and frameworks are more like conceptual and analytical models and they are not established so far [80]. The design of controllers and the management programs would be a very complex task considering the IoT network scale and the application complexities. As a conclusion, the whole concept of SD-IoT is in its infancy and standardization efforts in terms of framework, protocols, SDN applications, and assessment tools are still underway. Additionally, we found out the lack of study in the field of performance and quality of service management within the unified SDN and IoT system. The aim of this work is to provide the QoS management architecture for IoT applications. The architecture leverages the SDN technology to build a flexible and adaptive QoS support framework which evaluates the performance of the infrastructure resources across multiple layers of IoT architecture and allocates the resources based on user/application QoS needs.

# CHAPTER 3 SYSTEM ARCHITECTURE

The context of the Internet of Things (IoT) is the radical evolution of the current Internet into a network of the interconnected objects. IoT system is comprised of a large number of heterogeneous devices which lead to having diverse services and a multitude of applications.

IoT is entered to cover a wide area of our daily life in the concept of the Smart Transport, Smart City, Smart Home, Smart Health, and so on. The vision of the Smart City is to utilize the most advanced technologies to build the sustainable infrastructures and develop the better-quality and enhanced services for the citizens. Therefore, there are plenty of smart city projects up and running around the world, each one is working on specific applications such as Traffic Congestion, Structural Health, Smart Parking, and Environmental Monitoring. All those projects utilize sensor technologies to collect application-specific data, and the Internet ( generally, core transport network) to do data exchange between applications and sensors. Therefore, sensors and the Internet are shared between a wide range of applications. Also for the cost efficiency, sensing networks are mostly multi-services and they provide more than one application and services.

Various applications demand the various QoS requirements from the system framework to achieve the desired and appropriate objectives. As mentioned earlier, IoT presents different QoS requirements from conventional computer networks. In IoT, besides all well-known QoS attributes (e.g., delay, throughput and packet loss), new QoS attributes such as data accuracy, sensing coverage, sampling rate, and energy consumption are concerned too. Noting that this list is not exhaustive and other attributes could be added to the list. For instances, the QoS requirements of city traffic congestion and monitoring applications[1] are relatively stringent in terms of throughput and delay as well as data accuracy collected by sensors. The environmental monitoring applications[2] tolerate delay and packet loss, but their primary concerns are bandwidth and IoT coverage. The structural health monitoring applications[3] are sensitive to the quality of information and delay. Besides, the data collection must be processed in an energy efficient way to increase the sensing network lifetime [16]. Thus, to ensure that the system can provide the guaranteed service delivery, the QoS requirements of the application must be addressed at all involved subsystems and layers of the IoT architecture.

IoT systems are growing fast in terms of devices, services, and applications. The lack of a

1. To monitor vehicles and pedestrian levels to optimize driving and walking routes and to avoid the congestion.
2. To measure pollution, noise, temperature, humidity and smoke levels.
3. To monitor vibrations and material conditions in buildings, bridges, and historical monuments.

standardized end-to-end protocol for establishing QoS, the dynamic nature of the network, the unpredicted growth rate of the system, and the diverse QoS requirements of applications make the QoS guarantee in IoT system more complicated. The organizations need to design a flexible and scalable QoS framework to keep up with system growth, diverse application types and complexity of the system.

Allocating sufficient resources to different applications in order to satisfy various requirements is a fundamental function of QoS management framework. So, the IoT framework must provide effective resource allocation and scheduling methods to meet the different QoS requirements of the applications and users in every layer of IoT.

In this chapter, we propose an SDN-based middleware to provide generic QoS support framework for IoT system in the context of Smart City. We focus on the IoT application QoS needs from the sensing layer and the core transport network( or the Internet), which both have the significant impact on the service performance in IoT architecture. The framework aims to provide QoS-aware resource allocation within sensing networks (WSNs) and end-to-end QoS guaranteed route across the core transport network in a coherent way. To achieve this, our framework leverages Software-Defined core transport network and its capabilities. The control layer within SDN architecture could play the role of the middleware to provide the general and customized services for the IoT applications through Northbound APIs. Besides, The centralized controller could control the network elements through Southbound APIs.

Our structure is made of several databases and functionalities across different layers of IoT architecture. The following sub-chapters include the proposed QoS architecture, assumptions, principles, and the main modules. Later, the rules which govern the architecture, and the details about the modules are described. To help clarify the framework operation and objective, we describe the workflow in the sequential style. Finally, we analyze the values of the proposed architecture from different perspectives.

## 3.1 Proposed architecture

The proposed architecture has been illustrated in Figure 3.1. To provide a flexible and scalable QoS support service for IoT system, we offer to have SDN-based Internet/core transport network. The control layer of SDN architecture could provide the abstraction layer between the application and IoT infrastructure.

It also enables to have the central control over the transmission and sensing resources. Our architecture focus lies in the QoS requirements imposed by the IoT applications. Thus, applica-

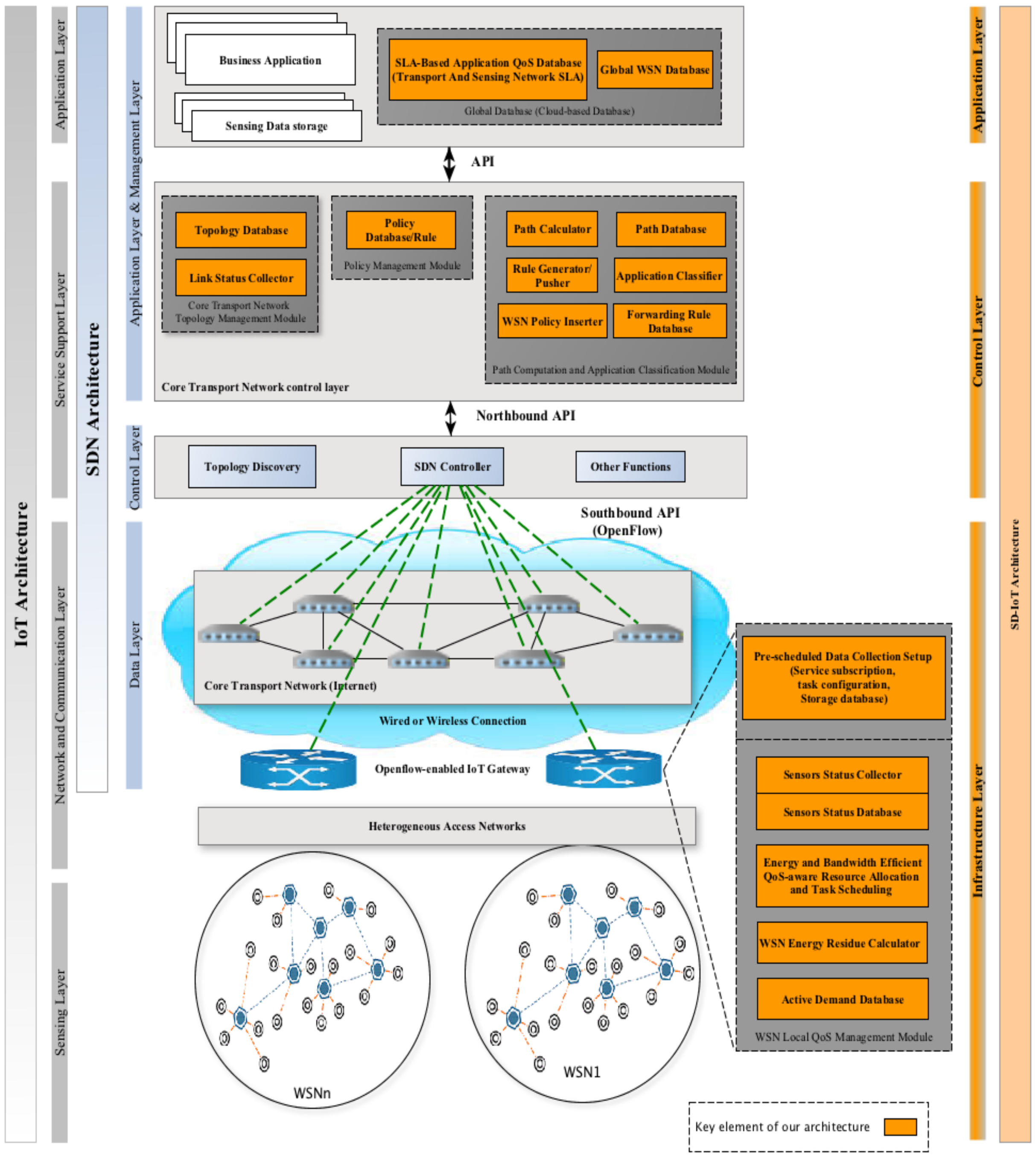

IoT Architecture
Application Layer
Service Support Layer
Network and Communication Layer
Sensing Layer
SDN Architecture
Application Layer & Management Layer
Control Layer
Data Layer
Business Application
Sensing Data storage
SLA-Based Application QoS Database (Transport And Sensing Network SLA)
Global WSN Database
Global Database (Cloud-based Database)
API
Topology Database
Link Status Collector
Core Transport Network Topology Management Module
Policy Database/Rule
Policy Management Module
Path Calculator
Path Database
Rule Generator/ Pusher
Application Classifier
WSN Policy Inserter
Forwarding Rule Database
Path Computation and Application Classification Module
Core Transport Network control layer
Northbound API
Topology Discovery
SDN Controller
Other Functions
Southbound API (OpenFlow)
Core Transport Network (Internet)
Wired or Wireless Connection
Openflow-enabled IoT Gateway
Heterogeneous Access Networks
WSNn
WSN1
Pre-scheduled Data Collection Setup (Service subscription, task configuration, Storage database)
Sensors Status Collector
Sensors Status Database
Energy and Bandwidth Efficient QoS-aware Resource Allocation and Task Scheduling
WSN Energy Residue Calculator
Active Demand Database
WSN Local QoS Management Module
Key element of our architecture
Application Layer
Control Layer
Infrastructure Layer
SD-IoT Architecture

Figure 3.1 QoS-aware and status-aware QoS support IoT framework through SDN-based middleware solution

tions can customize their own specific QoS needs at any time. These changes are dynamically enforced in the resource allocation process running on the middleware. Also, on one hand, the framework continuously collects the QoS status of the networks, in both sensing and core transport network. On the other hand, it accesses to the application QoS needs and the specific network constraints. This solution is adaptive to the ever-changing conditions of the network status and the application needs. So, the awareness of the current status of network resources, the features of performance required by each application and the network policy direct not only to have intelligent resource allocation, it improves the application satisfaction and network throughput.

SLA as the quality of service agreement between customer and service provider establishes the customer expectations on a service performance and quality. SLA includes several QoS parameters and various level of services [81]. In IoT, SLA comprises user and application expectations relative to the network, data, and devices. The success and failure of a requested service are determined by the service availability in both network and sensing layers. The sensing layer possesses the essential functions and resources to extract required data for any given services. Transport network (core and access) provides the access and routing functions between the application and the sensing devices. Therefore, the overall IoT service performance depends on the performance of both layers in providing services [82].

The application QoS attributes are mapped into the QoS parameters of transport and sensing networks based on their impact on any of these attributes. Earlier, we introduced the main QoS attributes over three layers, which are briefly shown in Table 3.1. Noting that the quantitative QoS parameters are just demonstrated and the list is non-exhaustive.

Table 3.1 QoS parameters mapped in IoT architecture layers

| **IoT layer** | **Application layer** | **Network layer** | **Sensing layer** |
|---|---|---|---|
| QoS parameters | Service time<br>Service availability<br>Service cost<br>Service reliability | Bandwidth<br>Packet loss<br>Jitter<br>Delay | Data accuracy<br>Data collection delay<br>Sampling rate<br>WSN lifetime<br>WSN coverage |

Various applications want to obtain different resources to fulfill their diverse requirements in order to operate appropriately. Thus, the QoS framework should be able to allocate resources to the different service demands while maximizing the overall system throughput. If there are multiple networks or devices available to serve the request, the decision process needs to allocate the most appropriate one considering all constraints imposed by both application and system. The framework must have the capabilities to handle the situation which within any SLA-based QoS premises are violated based on the current network situation. Besides, in case of resource shortage to serve all the requests received together, the system must differentiate the priority of the demands and schedule them based on their criticality. Furthermore, QoS support management in the IoT needs a flexible and scalable solution to keep up with the diversity and evolvement of IoT applications, and system complexity in terms of heterogeneity and dynamism.

To achieve such QoS support framework for IoT, we propose an SDN-based core transport network and Internet. Because of the control plane and data plane separation, the operational instructions are provided by the software programs running in the SDN controller instead of the multiple vendor-specific devices and protocols. The controller could manage the forwarding devices with various characteristics and functions through the southbound interfaces(e.g., OpenFlow protocol).

The northbound interfaces enable the programmable network functions that tell the controller how to manage the network. It yields the cost-effective network operation and management in the globally extended Internet. The value of the northbound interfaces is tied to the innovative and adaptive network services it can potentially support and enable. The new services could be aligned exactly with the business and users needs. The customized network services are completely a software and could be implemented as plug-ins to the centralized controller or any other standalone servers which could interact with the controller through APIs. In this work, we take full advantages of the SDN characteristics and the benefits of the northbound interfaces to design a centralized QoS management service to efficiently control the core transport network resources. Since we integrate SDN into IoT architecture, we could customize the QoS management services based on IoT application needs.

In our architecture, the management layer and control layer in SDN architecture act as the middleware layer (or software layer) between IoT application layer and IoT infrastructure layer. It abstracts the complexity and heterogeneity of physical devices from the application point of view. Figure 3.1 shows how we map our design and its components into the IoT and SDN architecture layers individually (at the left side of figure), and finally into the SD-IoT architecture layers (at the right side of figure). In SD-IoT architecture, the sensor and core transport network elements are considered as the IoT infrastructure (data plane), the control layer and management layer have the middleware functionality, and finally, application layer includes the IoT applications.

In general, sensing networks including WSNs and RFID are made of low-end devices with the limitation in addressability and IP-based connectivity. The most common approach to connect WSNs with an IP network and the Internet is through a gateway node. Along with the connectivity, the gateway-based approach enables us to implement more efficient protocols and algorithms in terms of clustering, routing, data gathering and fusion processes inside the WSNs in the centralized powerful gateway [16]. So, the IoT-gateway within our architecture would be the entry point to access the low-end sensors under its control. It controls and manages the sensor resources and the tasks associated with the WSNs through implemented algorithms and mechanisms.

In our design, the well-defined southbound APIs provide a centralized view of the core transport network topology, and the status of the network links and network elements' ports in terms of the QoS metrics such as the available bandwidth of link, packet loss ratio, and delay. Also, we assume that the SLA-related QoS requirements of the IoT applications from either the core transport network or sensing network are quantified and represented by real numbers in a Database. The WSNs setup configurations in terms of the supported services and the IoT-gateway addresses are provided in a specific database. Also, the sensors QoS attributes such as energy level, quality of information, and sensor distances from the gateway, are gathered by the IoT-gateway continuously and stored in the local storage. The sensing network energy level and the availability of the sensing resources are updated in the global (cloud-based) database which is accessible through the standard database APIs.

Depending on the requested services and the availability of the WSNs, our designed framework could select the most optimized network for the services. Accordingly, the associated IoT-gateway is considered as the destination of the routing path across the core transport network. The dynamically gathered network status information (sensing and transport network) and the application QoS preferences are utilized by the designed QoS function to provide best possible routing resources across the core transport network between the application as the source of demand and IoT-gateways as the destination of the demand. The IoT-gateway is supposed to be OpenFlow-enabled so that it can be remotely programmed by the SDN controller through southbound interfaces. Hence, the sensing requirements of the application, which are stored in the SLA-related database, are enforced dynamically in the IoT-gateway by the controller. The IoT-gateway receives the request and the QoS requirements, then it schedules the task and sends the collected data back to the source through the arranged routing path.

Therefore, the QoS support service implemented on top of the controller provides the following functions:

— Allocate the QoS-aware sensing network based on application request.

— Classify the application traffic based on the defined policy.

— Calculate the dynamic QoS support routing path across the core transport network.

— Generate and insert the flow table rules in the OpenFlow-enabled elements.

— Provide the sensing-relevant application QoS requirement and application priority for the IoT-gateway.

and, IoT-gateway provides the following functions:

— Connect low-end sensing devices to the Internet and IP network.

— Collect the sensors dynamic status data in terms of QoS-related attributes to calculate the network energy level, availability, and quality of information level for the storage database.

— Manage the QoS-aware task scheduling and resource allocation within the WSN under its coverage.

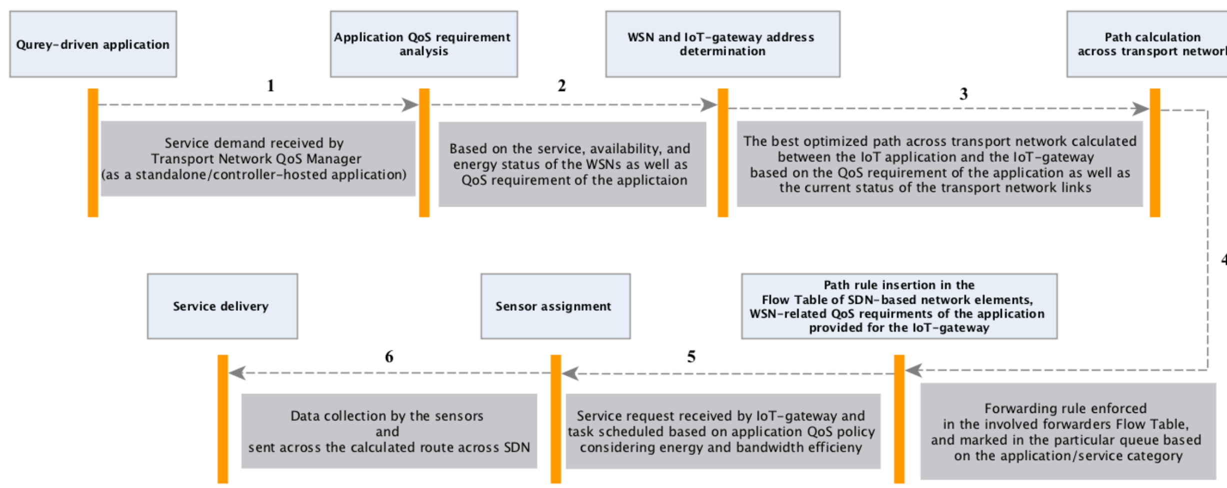
As a summary, we consider how the underlying IoT infrastructure can deliver the sensor data while efficiently utilizing resources and respecting the required quality of service. Since it is not sufficient to analyze all possible application in sensing networks, we classify them based on the data delivery model. In general, there are two main data delivery model in WSNs: [83]

— Query-driven in which data is generated on demand.

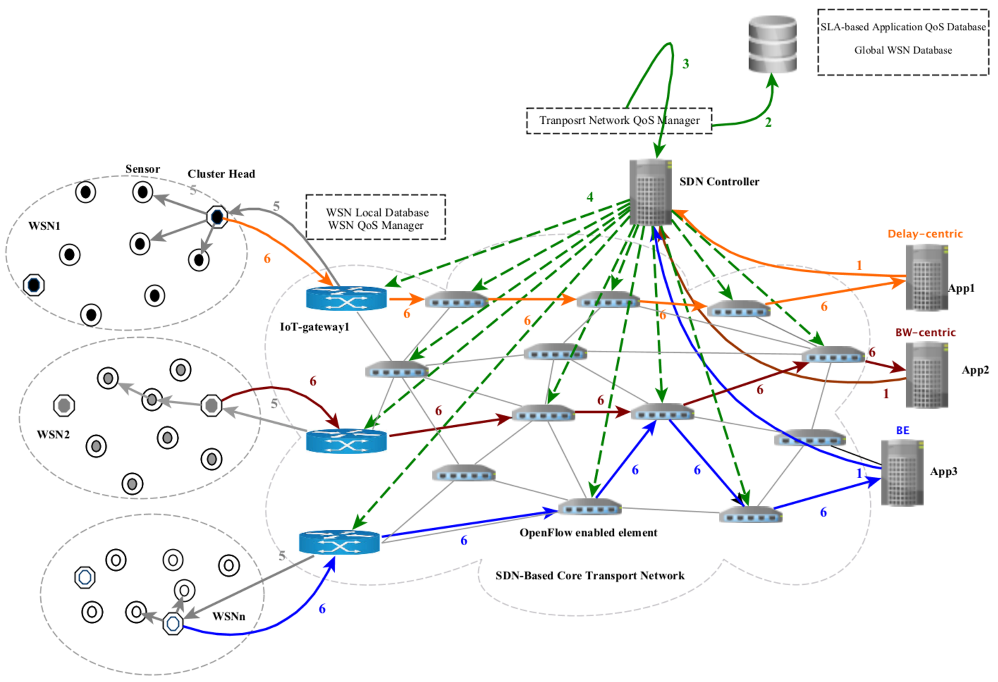
— Event-driven in which data generated in response to an event.

In query-driven model, the service demand is started from the application side toward sensing infrastructure. Therefore, the service request is received by the QoS management module on top of the SDN controller. If we assume that the application servers are connected to the Internet, then the request will pass through the core transport network elements and reach the SDN controller and QoS management module. First, it verifies the user subscription for the requested service. If the request is admitted, it queries the WSNs databases to determine energy-efficient WSN to serve the request. Then, it computes the optimized routing path across the core network toward the IoT-gateway of determined WSNs. To calculate the path, the designed QoS support routing function takes into account the source and the destination of the data, the QoS requirements of the application, and the network link status. Then, the forwarding rules for this path are inserted into the Flow Table of the associated core network elements by the controller. Besides, application QoS expectation from the WSNs are deployed within the programmable IoT-gateway by the remote controller, so IoT-gateway could assign the resources and schedule the task by applying the received QoS policy in the sensor allocation and routing algorithms running on it. Diagram 3.2a summarizes the flow of steps to handle QoS-aware routing path and resource assignment for the query-driven application. To make the workflow more understandable, we map the steps in the components of IoT architecture in Figure 3.2b.

In the event-driven data model, the sensors are programmed to report the data only when an event of interest occurs. Therefore, data is flowed from sensing layer toward application


Qurey-driven application
Application QoS requirement analysis
WSN and IoT-gateway address determination
Path calculation across transport network
1
Service demand received by Transport Network QoS Manager (as a standalone/controller-hosted application)
2
Based on the service, availability, and energy status of the WSNs as well as QoS requirement of the applictaion
3
The best optimized path across transport network calculated between the IoT application and the IoT-gateway based on the QoS requirement of the application as well as the current status of the transport network links
4
Service delivery
Sensor assignment
Path rule insertion in the Flow Table of SDN-based network elements, WSN-related QoS requirments of the application provided for the IoT-gateway
6
Data collection by the sensors and sent across the calculated route across SDN
5
Service request received by IoT-gateway and task scheduled based on application QoS policy considering energy and bandwidth efficieny
Forwarding rule enforced in the involved forwarders Flow Table, and marked in the particular queue based on the application/service category

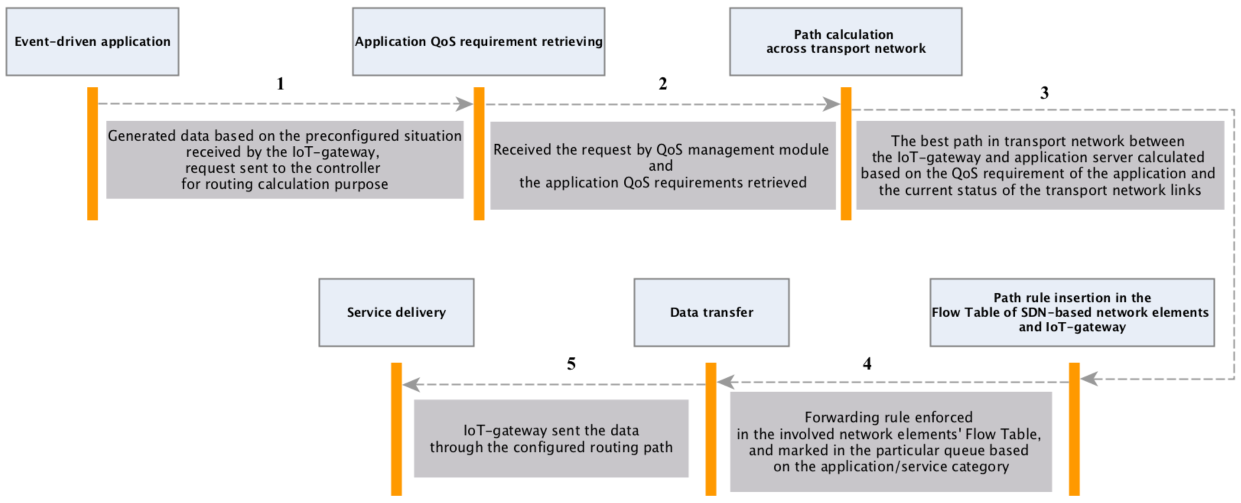
(a)



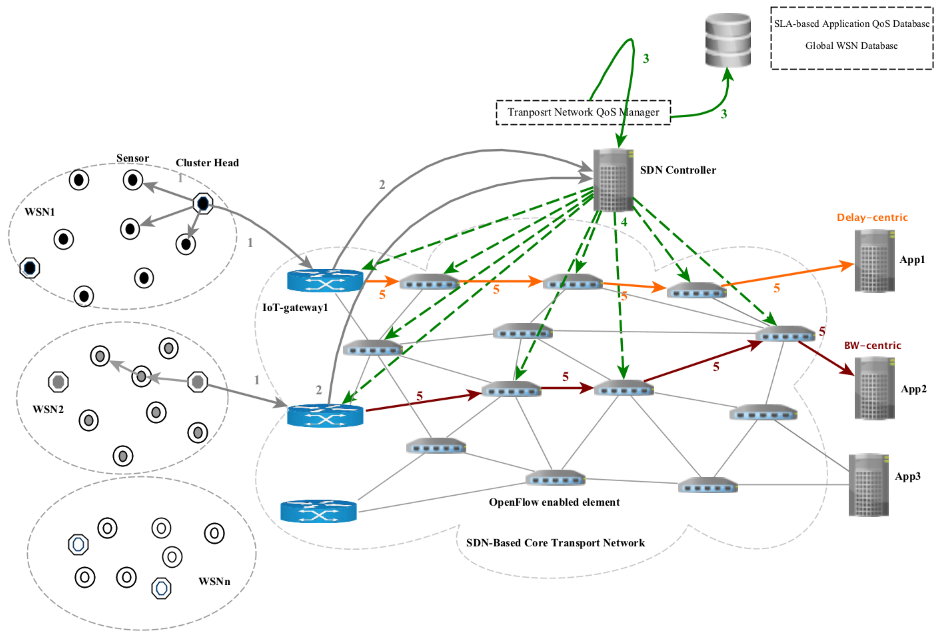
(b)

Figure 3.2 (a)Sequence diagram of designed QoS support resource allocation for query-driven IoT application mapped into the architecture components illustrated in (b)

layer. When IoT-gateway receives the collected data, it sends the data transfer request to the controller. Again, the QoS management module on top of controller verifies the request QoS needs and calculates the routing path across the core transport network respecting its preferences and the network status. Diagram 3.3 shows the sequences followed in our framework for this type of application. The application with pre-defined WSN setup for continuously or periodically data collection could follow the same steps to transfer the data from IoT-gateway to the appropriate application server or database.


Event-driven application
Application QoS requirement retrieving
Path calculation across transport network
1
2
3
Generated data based on the preconfigured situation received by the IoT-gateway, request sent to the controller for routing calculation purpose
Received the request by QoS management module and the application QoS requirements retrieved
The best path in transport network between the IoT-gateway and application server calculated based on the QoS requirement of the application and the current status of the transport network links
Service delivery
Data transfer
Path rule insertion in the Flow Table of SDN-based network elements and IoT-gateway
5
4
IoT-gateway sent the data through the configured routing path
Forwarding rule enforced in the involved network elements' Flow Table, and marked in the particular queue based on the application/service category

(a)



(b)

Figure 3.3 (a)Sequence diagram of designed QoS support resource allocation for event-driven IoT application/pre-setup IoT services mapped into the architecture components illustrated in (b)

## 3.2 Assumption

Our idea is to provide the QoS support service delivery within IoT system. Generally, the flows are classified into QoS flow and best-effort flow. The QoS flows are the flows which need the specific level of QoS so that the QoS parameters such as bandwidth, packet loss, jitter, and delay must be guaranteed [84]. Our design considers the QoS support framework for the QoS flow which has an SLA agreement regarding the service performance expectation. The best-effort flow could be responded by the existing best-effort mechanisms.

Our architecture needs to have the updated information about the QoS attributes of the wireless sensor networks. Thus, we assume that there are algorithms and mechanisms implemented in the IoT-gateway to collect the status information, calculate the QoS metrics value and store them in the database. There have been plenty of approaches which have been explained in Chapter 2 and later, we will mention more relevant solutions.

Additionally, our assumption is that the core transport network elements plus IoT-gateways are OpenFlow-enabled and they support OpenFlow version 1.3 and upper versions which have the extended QoS capabilities.

Also, we suppose that all the databases are structured in standard storage styles and data format. So, the data could be queried and retrieved by the external applications. NoSQL database is increasingly used in big data and real-time web applications because of its simplicity, scalability, and flexibility for rapidly changing data. In the offered framework, the databases are going to be updated very frequently. NoSQL database and standard data interchange format like RDF/XML could be used to simplify the database query.

## 3.3 Component description

This section aims to focus on the logical components of the architecture and the interaction between them. The proposed architecture is made of five modules (Indicated with a gray dashed-line box in Figure 3.1), and each one has several components (Orange-colored box) to perform specific functions within our structure.

(a) Global Database, which could be implemented in cloud system to be globally accessible, is made up of two main databases:

**Global WSN Database.** This database (see Figure 3.4) includes the general and static information about the WSNs such as the supported services, the supported bandwidth, the address of the IoT-gateways which connect WSNs to the external IP-based networks (e.g., the Internet).

Considering IoT sensing layer deployed by multiple providers, several WSNs could provide the same services but with different level of data quality and level of cost. Being aware of the WSNs status helps us to decide the most appropriate one for a given application request. Therefore, this database comprises the overall status of the WSNs in terms of the energy residue level, availability, the quality level of the collected data (QoI), and the cost.

The initial information of WSNs could be provided by Sensing Network Provider(SNP) in the network deployment phases. The dynamic characteristics of the sensing networks could be estimated over the time by the IoT-gateways based on the implemented algorithms. (**WSN energy residue calculator** and **Sensors status collector** functions in Figure 3.1).

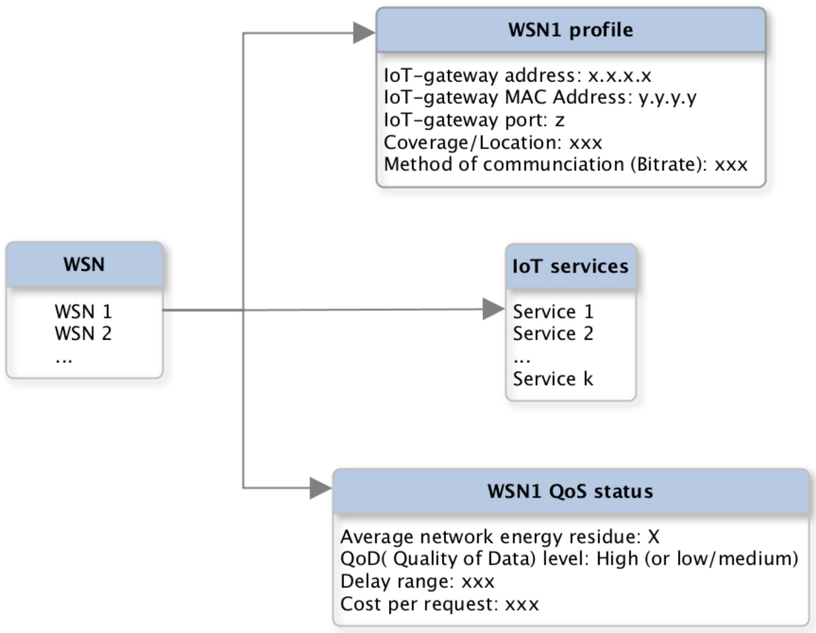

WSN1 profile
IoT–gateway address: x.x.x.x
IoT–gateway MAC Address: y.y.y.y
IoT–gateway port: z
Coverage/Location: xxx
Method of communciation (Bitrate): xxx
WSN
WSN 1
WSN 2
...
IoT services
Service 1
Service 2
...
Service k
WSN1 QoS status
Average network energy residue: X
QoD( Quality of Data) level: High (or low/medium)
Delay range: xxx
Cost per request: xxx

Figure 3.4 Logical Format of Global WSN Profile Database. (It shows the particular WSN1 profile and its association with IoT services)

**SLA-based Application QoS Database.** This database stores the QoS requirements of IoT application from transport network and sensing network point of view. Either application owner or service provider can provide the QoS-specific information in this database based on the agreed SLA. It includes the subscription of the applications to the IoT services as well as the accepted level of QoS indicators from IoT infrastructure. Assuming this database has the logical format illustrated in Figure 3.5. This figure explains that *application1* has subscribed to the IoT service $k$, expecting a specific degree of QoS parameters: $D^k_{max}$ refers to the maximum acceptable delay across transport network, $B^k_{min}$ is the minimum required bandwidth, and $PL^k_{max}$ refers to the maximum tolerable packet loss ratio. Further, required the quality level of the sensing data, the accepted sampling rate and cost of the sensing services are associated with the service $k$. Then, this database includes the unified format for SLA-related QoS constraints of services for the user or the application.

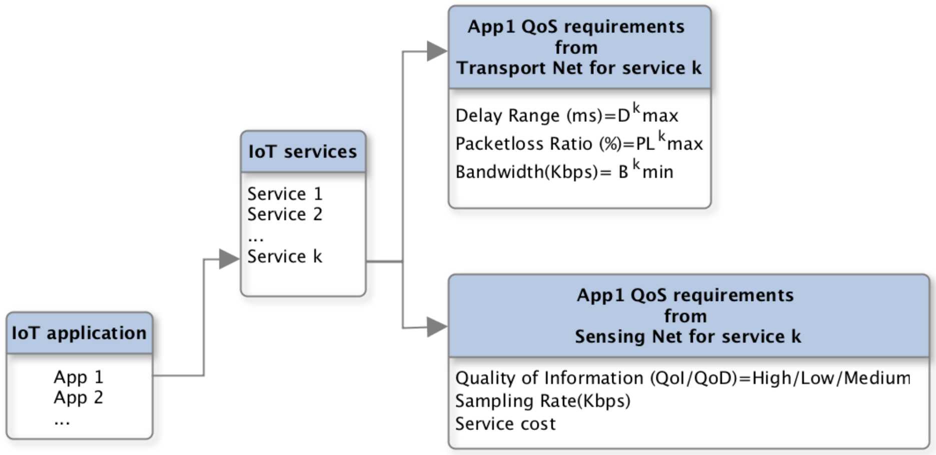

App1 QoS requirements from Transport Net for service k
Delay Range (ms)=D k max
Packetloss Ratio (%)=PL k max
Bandwidth(Kbps)= B k min
IoT services
Service 1
Service 2
...
Service k
App1 QoS requirements from Sensing Net for service k
Quality of Information (QoI/QoD)=High/Low/Medium
Sampling Rate(Kbps)
Service cost
IoT application
App 1
App 2
...

Figure 3.5 Logical format for SLA-based QoS Database

(b) WSN Local QoS Management Module. This module is implemented on the IoT-gateway to take care of the QoS-aware resource and task management inside the local WSNs. This module is made of several components and functions which are illustrated in Figure 3.6.

The IoT-gateway as the centralized and powerful device gets the requests from the upper layer, allocates the sensors for data collection and makes them available for the source of the requests. The powerful IoT devices with the Internet access capability could directly interact with the application. Although IoT-gateway could have sensing capabilities, they could provide the data collection services as well as the computation, storage, and Internet access functions for the low-end WSNs. (see Figure 3.7)

In this work, we focus on the low-end sensing devices which communicate with the external network through IoT-gateway, since they are widely used in the Smart City to sense the environment and provide the IoT services ; although the idea could be extended to the standalone powerful devices as well. QoS mechanisms used to support QoS in wired data networks cannot be directly applied to the wireless sensing networks

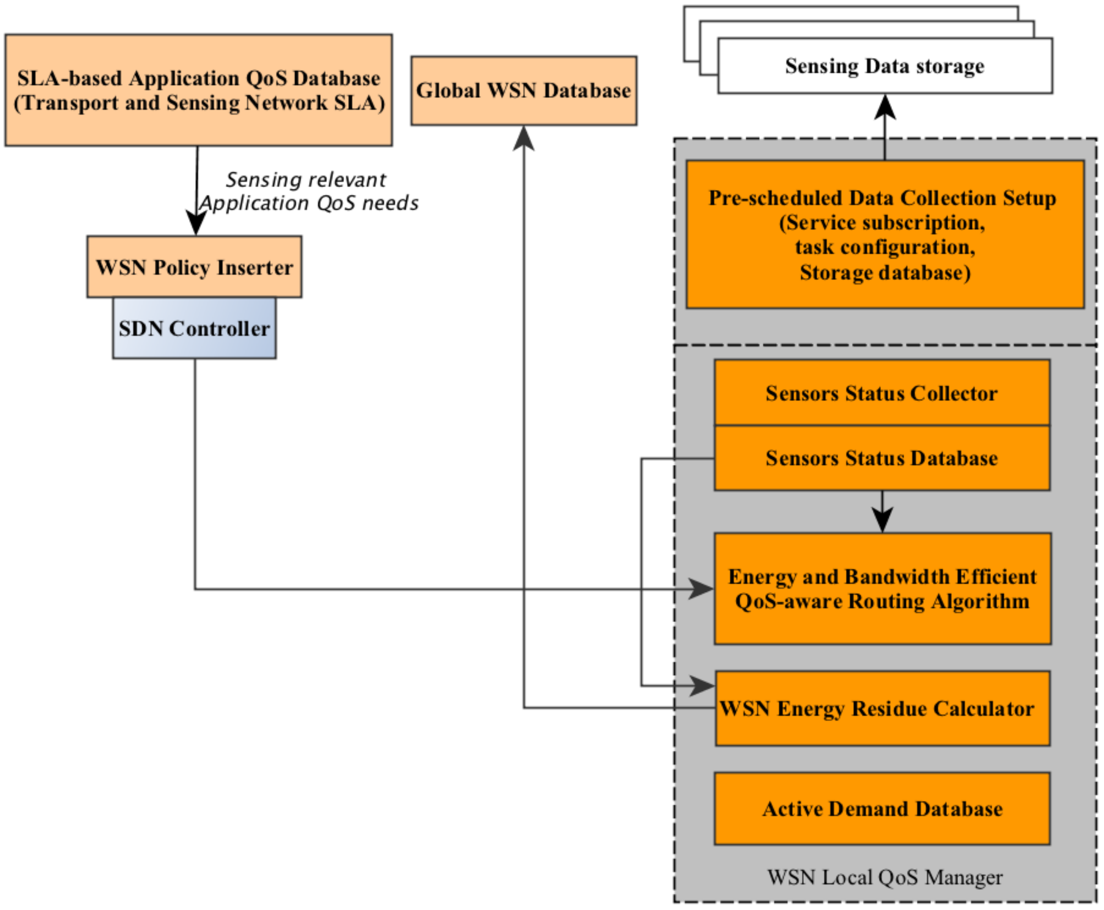

SLA-based Application QoS Database (Transport and Sensing Network SLA)
Global WSN Database
Sensing Data storage
Sensing relevant Application QoS needs
WSN Policy Inserter
SDN Controller
Pre-scheduled Data Collection Setup (Service subscription, task configuration, Storage database)
Sensors Status Collector
Sensors Status Database
Energy and Bandwidth Efficient QoS-aware Routing Algorithm
WSN Energy Residue Calculator
Active Demand Database
WSN Local QoS Manager

Figure 3.6 Building blocks of WSN Local QoS Manager module

because of the resource constraint, dynamic network topology, and different QoS attributes. From the network point of view, bandwidth and energy optimization would be the most important metrics. From the application perspective, the Quality of information (QoI), data collection rate, delay, and cost are the main QoS parameters. In the literature, there have been plenty of works to implement the complex energy-efficient QoS functionalities in sensor-based environments [37]. However, we are interested in the energy-aware and bandwidth-efficient network resource allocation methods in which the application QoS preferences could also be considered in the service delivery. The cost of service could have an opposite relation with QoI, so the WSN QoS management solution should balance between QoI and service cost while minimizing energy and bandwidth. These functions and algorithms are implemented in **Energy and bandwidth efficient QoS-aware resource allocation** component in our proposed framework.

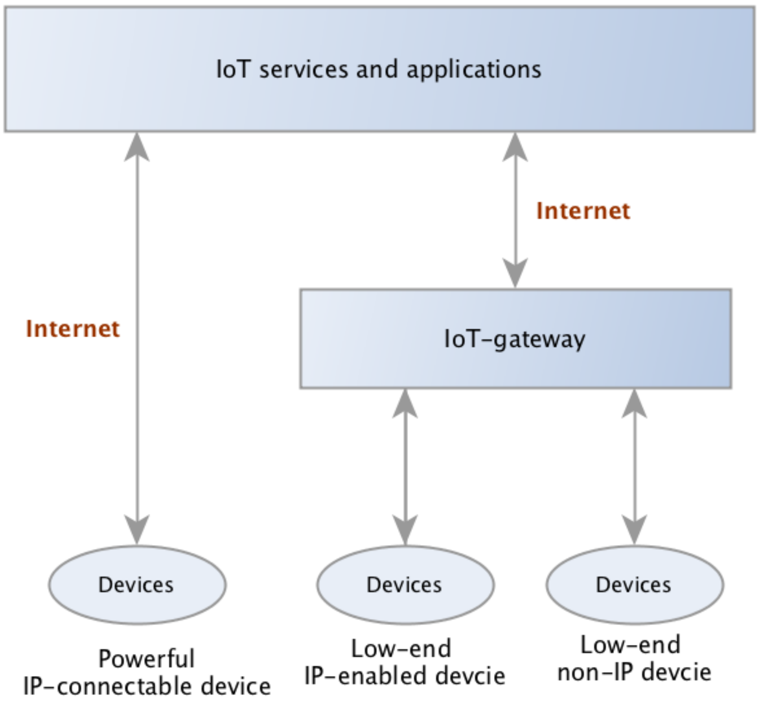

IoT services and applications
Internet
Internet
IoT–gateway
Devices
Devices
Devices
Powerful IP–connectable device
Low–end IP–enabled devcie
Low–end non–IP devcie

Figure 3.7 IoT device connection paradigm (reproduced from [14])

The application request might need the data acquisition by a single sensor or multiple sensors, continuously or periodically [37]. Though, two main approaches used for data collection in WSNs are:

(a) Push approach [85], which sensors pro-actively collect the data and store in the pre-defined storage. The applications which are subscribed to the data could query the storage and fetch the required data. This method could be more useful when multiple applications are subscribed to the same data.

(b) Pull approach [86], which keeps all the sensors silent until a request for their data arrives from associated application. IoT-gateway receives the application requests and provides the QoS specific data for them.

In the first approach, the QoS requirement for any services could be embedded in the sensors programs. Also, IoT-gateway could do more complex functions like data aggregation and filtering process to make the data more appropriate for the application. In our architecture, we put the application subscription information, pro-actively-planned task configuration and data collection setup in the component **Pre-scheduled Data Collection Setup**, although the collected data could be either in local storage or in the cloud (**Sensing data storage**). So, the applications subscribed to these services could use the storage address to query the data. In the second approach, IoT-gateway enforces the QoS requirements while assigning sensors and scheduling tasks. Thus, QoS support mechanism for the sensor allocation and task scheduling takes into account the desired QoS policy for sensor assignment, while optimizing the energy and network bandwidth.

IoT-gateway needs to have the sensor QoS status in terms of energy level, mode, location, and data quality to assign more appropriate and optimized resources for different application requests.

**Sensors Status Collector** collects periodically the sensor status data and stores in a local Database **Sensors Status Database** at the IoT-gateway. The QoS routing mechanism implemented on the IoT-gateway provides the QoS-aware local resource allocation while optimizing energy and bandwidth (**Energy and Bandwidth Efficient QoS-aware Routing Algorithm**) using the sensor status.

The component called **WSN Energy Residue Calculator** estimates the overall WSN energy residue based on the sensors energy level and provides it for the **Global WSN Database**. In the literature, the information regarding the amount of the residual energy distributed in the WSN is called an energy map. Escan [87] proposes one approach to provide the information about the WSN energy level.

To avoid redundancy in data collection tasks, IoT-gateway keeps the currently active service demand and the relevant configuration setup in the database called **Active Demand Database**. So, whenever a new request is received, IoT-gateway verifies this database to see whether this demand is replicated or any current active demand could satisfy this new request. This process eliminates the repetition in data acquisition, transmission and pre-processing. Hence, it enhances the resource efficiency and network lifetime.

Having the design or operational goals in terms of WSN QoS attributes is generally difficult since it is still very challenging to capture, analyze and utilize all kinds of QoS information in a consistent manner from heterogeneous and sensor devices in WSNs. The researchers in the literature have pursued WSN QoS support using a large number of mechanisms and algorithms in different protocol layers while maximizing sensor lifetime and bandwidth utilization. We classify the approaches into two big categories:

— SDN-based WSNs in which IoT-gateway operates as the controller for the OpenFlow-enabled sensors to adaptively manage them. For instances, authors in ([76, 88]) propose a prototype for Software Defined Clustered Sensor Networks (SDCSN), which facilitate the status collection of the sensors. More importantly, routing and resource management could be more efficient through having centralized IoT controller, argued in [12, 89]. In [90], the lightweight and efficient northbound and southbound API style are designed for the SDIoT.

— Classical WSNs in which the algorithms are installed on the IoT-gateway in terms of clustering, routing, energy and bandwidth management.

One of the most relevant approaches is implemented in MiLAN project [9]. Authors develop a middleware which allows applications to specify a policy for managing the sensor networks. MiLAN receives a description of application requirements, monitors sensing network conditions about available sensors and their capabilities and level of energy. Then, it configures the sensors in the way that applications are satisfied and network lifetime is maximized. This technique is deployed in the centralized control device (IoT-gateway) as an advanced feature of WSN-middleware.

Other studies are trying to measure one or multiple WSN QoS attributes in order to benefit from the QoS-aware data collection. QoI as the degree of the data accuracy in WSN considered as one of the centers of attraction for the application. Authors in [91] aims to provide a common quality of information model including principles and policies for exchanging the quality metadata about the information. It could be used as a base model from which application-specific models can be developed. [92] provides an estimation of the quality of information(QoI) perceived by the end-user through the impact of some attributes including latency, reliability, accuracy, relevance, and robustness. M. Mathew et al [93] present an optimization model towards co-improving quality and energy in sensor networks. They introduce a quality and energy-aware adaptive scheduling techniques to balance between energy consumption and application QoI requirement satisfaction. Another approach considers user feedback to evaluate the sensor quality of information [94].

Inspiring from these works, we could develop a QoS control model for the sensor-based networks, either SDN-based or classical sensing network, to adaptively manage the resources.

(c) Core Transport Network Topology Management Module. A network topology is the arrangement of a network including all network devices, their connection structure, link capacity and port status (up or down /active or inactive). In our architecture, we need to have the updated information about network topology and also the QoS state of the SDN network links.

According to the OpenFlow specification v1.0, *Topology Discovery* function as the de facto standard function is implemented in all controllers. This function enables the controller to discover a network topology of the entire SDN infrastructure. The controller discovers the network elements by exchanging *Hello* messages, and it detects their connection structures using OFDP(OpenFlow Discovery Protocol) mechanism illustrated in Figure 3.8. The controller encapsulates an LLDP (Link Layer Discovery Protocol) packet as a *Packet-out* message and sends it to the connected OpenFlow-enabled elements. Then after, the network element(NE) sends the received LLDP packet to all its neighbors which are connected directly to its active ports. When a network element receives an LLDP packet from another element, it sends this LLDP packet to the controller as a *Packet-in* message since there is no matching entry in its Flow Table. The controller learns which network elements are connected directly to each other through received *Packet-in* messages and builds the global network physical topology. In other words, the controller confirms a direct link between two network element according to two identical LLDP packets received from both network element. [15, 95].

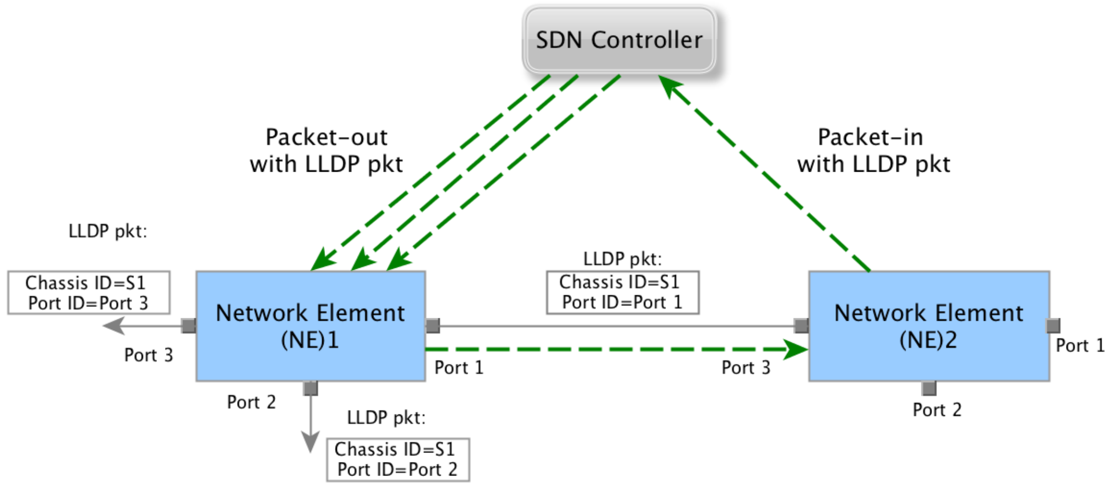


Figure 3.8 Topology discovery mechanism in SDN (Reproduced from [15])

The components of this module and the interaction between them are illustrated in Figure 3.9. **Topology Database** includes the up-to-date information about the network topology and the link status, which are provided by *Topology Discovery* and **Link Status Collector**, respectively.

**Link Status Collector** collects the network link status. Here, network link status refers to the network performance parameters including the packet loss ratio, available bandwidth, and delay of all active links within the network topology.

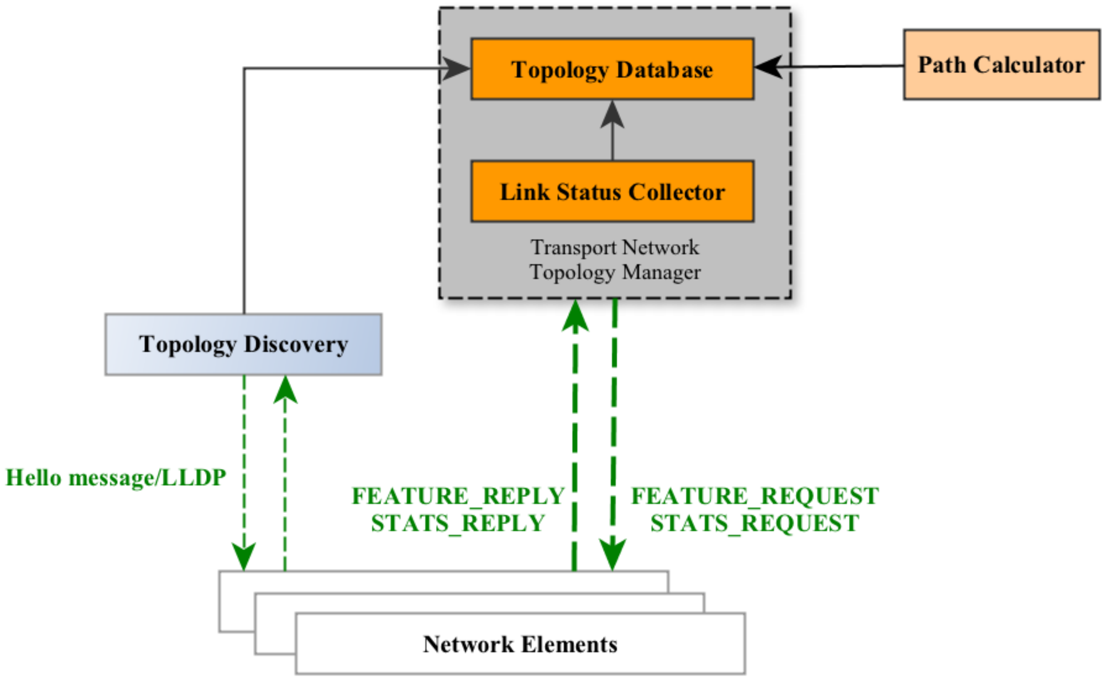


Figure 3.9 Building blocks of Core Transport Network Topology Management module

To obtain this information from the network, we take advantage of the implemented counters in the OpenFlow-enabled network element. As described in Section 2.2.3, there have been several counters implemented in Flow Table, Group Table, and Meter Table which store the packet processing record in each OpenFlow-enabled device (Table 2.1). Flow level counters provide information about a particular flow, e.g., how many bytes were matched against this flow, how many packets were forwarded, how many packets were dropped, how many errors occurred, the duration for which this flow entry was active etc. Table level counters aggregate statistical information regarding an entire flow table. Port-level counters provide more specific information about a particular port. Queue level counters provide information about how many bytes and packets were enqueued at a particular queue attached to a particular output port, how many packets were dropped, a duration for which this queue was active. In OpenFlow specification 1.3.0, support for querying meter level statistics was also added. Meter level statistics contain similar information e.g., how many bytes and packets were forwarded, a duration of this meter. The messaging mechanisms implemented in OpenFlow protocol facilitates the communication between the **Link Status Collector** function and the counters inside the network elements. *FEATURE_REQUEST/STATS_REQUEST* and *FEATURE_REPLY/STATS_REPLY* messages are used, respectively, to request and return the statistical information from the counters. [4]

This statistical information is playing a key role in our proposed model. They not only describe the estimate parameters of network conditions, they are used in the resource allocation and routing decision process across SDN. In the traditional network, several tools have been developed to monitor and measure the network conditions, passively [4] or actively [5]. In literature, new methods or tools such as OpenNetMon [96] and Flow-Sense [97] have been developed to measure the SDN network performance. They use different techniques to capture and analyze control messages between the network elements and controller. Their experimental results show the accepted level of accuracy in the small SDN network, although each one has been implemented for a specific controller. The ideas to obtain the packet loss ratio, delay and available bandwidth between pair network elements through OpenFlow counters could be implemented with some modification and adaptation in **Link Status Collector**.

**Topology Database** must be updated by both **Topology Discovery** and **Link Status Collector** periodically or in case of the changes in the network and link status. The interval time for status collection could be determined by the administrator based on the overall IoT traffic behaviors including traffic arrival rate and service time or the network dynamism history.

In summary, the goal of this module is to present a global transport network view. It could logically look like a graph (nodes and links) that has network link information associated with it: delay, available capacity, and loss rate. The control logic running on the SDN controller uses the APIs to interact with this module.

(d) Policy Management Module. **Policy Database** holds the network policy information provided by the network provider. This database includes the instructions which must be followed when SLA-violation or congestion happens in the network. Also the event-based temporary or permanent bandwidth reservation, admission control and load balancing procedure could be included in this module. The controller and the SDN applications could query this module to get the rules associated with the given network situation. (Figure 3.10)

(e) Path Computation and Application Classification Module. This module is supposed to handle QoS support routing management within the core transport net-

4. Passive measurement methods measure network traffic by observation, without injecting additional traffic in the form of probe packets

5. Active measurements inject additional packets into the network, monitoring their behavior. The Most popular application is ping which uses ICMP packets

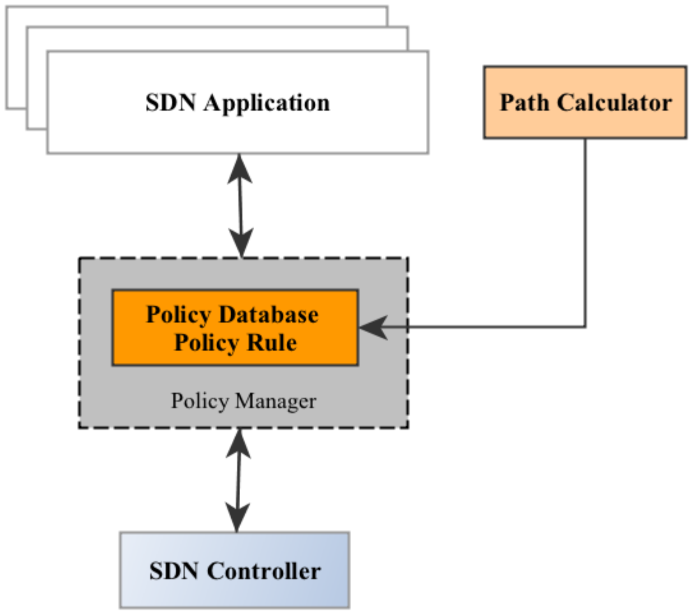


Figure 3.10 Policy management module

work. Since the OpenFlow(OF) proposal allows a straightforward QoS support mechanism, we propose an SDN application to provide the QoS support resource allocation in SDN-based core transport network for any QoS flow. This module interacts with the database in application layer in order to get application QoS requirements and then calculates the best feasible QoS-aware routing path based on the current network condition. The components which compose this module are called **Path Calculator**, **Application Classifier**, **Rule Generator/Pusher**, **Forwarding Rule Database**, **WSN policy pusher**, and **Path and Demand Database**. The components and their relationships are illustrated in Figure 3.11 and the functionality of each component is explained in the following paragraphs.

**Application classifier/scheduler.** In the implementation of the QoS enforcement in the networking environment, one main task is to differentiate the type of the application to do effective traffic scheduling and classification. Queuing mechanisms are the fundamental techniques to prioritize traffic in the network elements. The use of the queuing mechanisms also guarantees the network bandwidth to the different application traffic and to control the congestion. In the classical and SDN-based network, the packets, by default, cross the network with the well-known *First-In First-Out (FIFO)* scheduling method. This method is not suitable for the IoT system wherein the efficiency of the mission-critical applications is bounded to the delay. Besides, the QoS requirements of

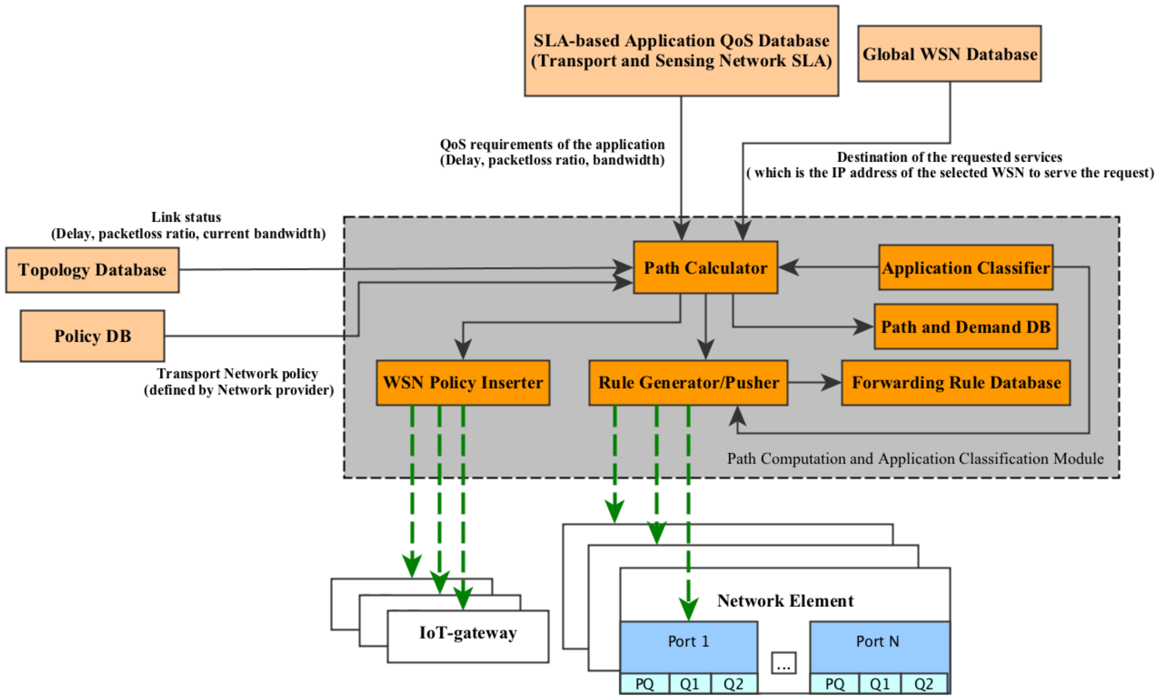

SLA-based Application QoS Database (Transport and Sensing Network SLA)
Global WSN Database
QoS requirements of the application (Delay, packetloss ratio, bandwidth)
Destination of the requested services ( which is the IP address of the selected WSN to serve the request)
Link status (Delay, packetloss ratio, current bandwidth)
Topology Database
Policy DB
Transport Network policy (defined by Network provider)
Path Calculator
Application Classifier
Path and Demand DB
WSN Policy Inserter
Rule Generator/Pusher
Forwarding Rule Database
Path Computation and Application Classification Module
IoT-gateway
Network Element
Port 1
Port N
PQ
Q1
Q2
...

Figure 3.11 Building blocks of Path computation and Application classification module

different applications might not meet in this method.

The *Application classifier* does the IoT application differentiation and queuing assignment for the IoT application. Network elements could have several limited queues in each port. Arrival packets are classified into a separate queue based on parameters like source/destination IP address, source/destination TCP port (application type), and Type of Service(ToS) field in IP header. We classify IoT applications based on delay sensitivity as described in Table 3.2.

Table 3.2 Application classification and queuing policy in OpenFlow Network Element

| **Application Class** | **QoS attributes** | **Priority** | **Type of queue** | **Traffic Class mapped onto the Cisco classification** | **Description** |
|---|---|---|---|---|---|
| Delay-Centric (Mission Critical) | $D^k_{max} \leq D_{Threshold}$ | 1 | PQ (Priority Queue) | EF (Expedited Forwarding) | Packet should experience no queuing delays, packets should have very low loss. |
| Bandwidth-Centric (Multimedia application) | $D^k_{max} \geq D_{Threshold}$ $BW^k_{min} \geq BW_{Threshold}$ | 2 | Q1 | AF (Assured Forwarding) | Packet should experience no mis-ordering , packets expect very limited loss rate. |
| General (Non-Real time analytic application) | N/A | 3 | Q2 | BE (Best Effort) | No strict QoS needs. |

We assume to have three different queues in each port of the OpenFlow-enabled network elements: (1) Priority Queue(PQ) as the most prioritized queue includes the traffics related to the mission-critical applications with intensive delay sensitivity. If the delay requirement of application is less than a pre-defined threshold, it is marked as a high-prioritized demand and it is inserted in PQ of egress ports of network elements. (2) Q1 is about the data-centric application with bandwidth sensitivity, and less sensitivity to the delay compared with the pre-defined threshold. (3) Q2 contains the applications with no strict QoS requirements.

Since the network elements have finite buffer capacity, *Complete Buffer Sharing* scheme between queues could be implemented to minimize the network delay for the mission-critical application. In this scheme, the highest priority traffic packets push-out the lower priority packets. All packets in higher priority queue are served before a lower priority queue. In general, in network devices such as routers, if a higher priority packet arrives while a lower priority packet is being served, it waits until the lower priority packet completes, unless *Preemptive Priority Scheduling* is used. In *Preemptive Priority Scheduling*, if a new process having a higher priority than the currently running process arrives, it gets selected immediately and the new process has not to wait until the currently running process finishes or yields. The port queues can be configured with standard protocols such as *CLI*, and *NetConf* directly in the network elements. Also, *OF-Config* protocol [98] could facilitate and automate the implementation of queuing mechanism and policy inside the OpenFlow-enabled devices too [99].

**Path Calculator/ Path Decision Maker.** Once the network topology and status information are provided by **Topology Manager**, the routing algorithm implemented in **Path Calculator** makes a decision about the routing paths for the application flow. This function first determines the best-fit WSN for the requested service based on the availability and the energy level of the WSNs. And then it considers the application QoS preferences and the network link status to provide SLA-respected routing path from the source of the demand to the IoT-gateway of the determined WSN.

Although the consideration of the dynamic status of the network link leads to avoid the congestion, we set the limit for the link utilization rate to keep the link load balanced across the network. Therefore, the links with the utilization rate more than a pre-defined threshold are excluded from the logical topology at a given time when calculating the routes. The link utilization threshold could have dynamic value depending on the traffic arrival rate and burst hours in the network. The link utilization policy could be defined in **policy Database** by the network administrator or the learning-based methods could be developed to determine this limit based on the network traffic history.

**Rule Generator/Pusher.** This function translates the routing decisions made by **Path Calculator** into the actual configuration commands for each of the network elements. So, it generates the flow rules of the route information and configures the Flow Table of all elements along the paths. Due to the design concept of OpenFlow, forwarding rules (flow entries) for all network-elements will be kept in a data structure of the controller. In this architecture, **Forwarding Rule Database** keeps the configured flow rules. To have the optimized messaging between the network elements and the control layer, when **Rule Generator/Pusher** receives the request of path insertion, first it checks the **Forwarding Rule Database** to verify if the associated flow rule exists in the Flow Tables, if not, the new flow entry is pushed in the Flow Tables. Then we need to always have the updated flow rules in this database.

The OpenFlow protocol supports two methods of removing entries from a flow table. In the first method, the controller sends an explicit *FLOW_DEL* message to the network element to specify which entry or entries should be removed. In the second one, the controller assigns a timeout to each flow entry. When the timeout expires, the network element erases the rule from the flow table and, optionally, notifies the controller the flow was removed [4]. To keep **Forwarding Rule Database** up-to-date, all the modification of the Flow Table entries must be reflected in this database. Therefore, the controller either sends a delete command for entry removal or receives the expiration notification from the network element, it modifies the database. Therefore, **Forwarding Rule Database** lists all active flows entries on the entire SDN network controlled by the controller.

In OpenFlow channel, *FLOW_MOD* message is used by the controller to modify the Flow Table entries in the network elements. It could ADD, DELETE or MODIFY flow entry. *SET_QUEUE action/EN_QUEUE action* specifies the queue ID in network elements' port which the flow entry should be entered.

**Path and Demand Database.** The path calculated for application requests are stored in this database. Whenever network topology is changed because of the fault or new design, **Path Calculator** verifies this database. If any currently installed path across the network is affected by this changes, it recalculates the new path and reinstalls them. This database is also updated based on new paths. This approach boosts the performance and resiliency of the system.

**WSN policy pusher.** To dynamically provide the application sensing-related QoS needs for the IoT-gateway, there could be two approaches. Whether IoT-gateway queries the SLA-based database to fetch the required information or *WSN policy pusher* is going to enforce the required data in the IoT-gateway. The later could be automated by employing *OF-Config* protocol [98] in the OpenFlow-enabled IoT-gateway.

## 3.4 Architecture workflow

In this section, to better understand the proposed architecture, we would like to explain the functions and process workflow in the diagram.

Flowchart 3.12 demonstrates the tasks associated with the SDN Controller and IoT-gateway to collect the status information of the OpenFlow-enabled network elements and the sensors, respectively. The SDN controller collects the network information about network structure by using LLDP and *Hello* messages. It measures the status of each network link such as the available bandwidth, delay, and loss rate by developed southbound APIs. It is set to update the topology database periodically and in case of topology change notification received by the controller. In WSNs, the IoT-gateway is responsible for monitoring the sensor's status and the energy level of them. Local routing mechanism inside WSN is managed by IoT-gateway through implemented energy-efficient QoS-aware routing algorithms [100, 101, 102, 103].

Flowchart 3.13 represents the steps followed when query-driven applications send the request to collect data. For the event-driven application, if the event is very critical, it might have reserved resources. If not, the workflow starts from the step that SDN controller receives the data transfer request from the IoT-gateway. Then, the same steps including SLA-DB query, path calculation, and flow rule enforcement are done consequently.

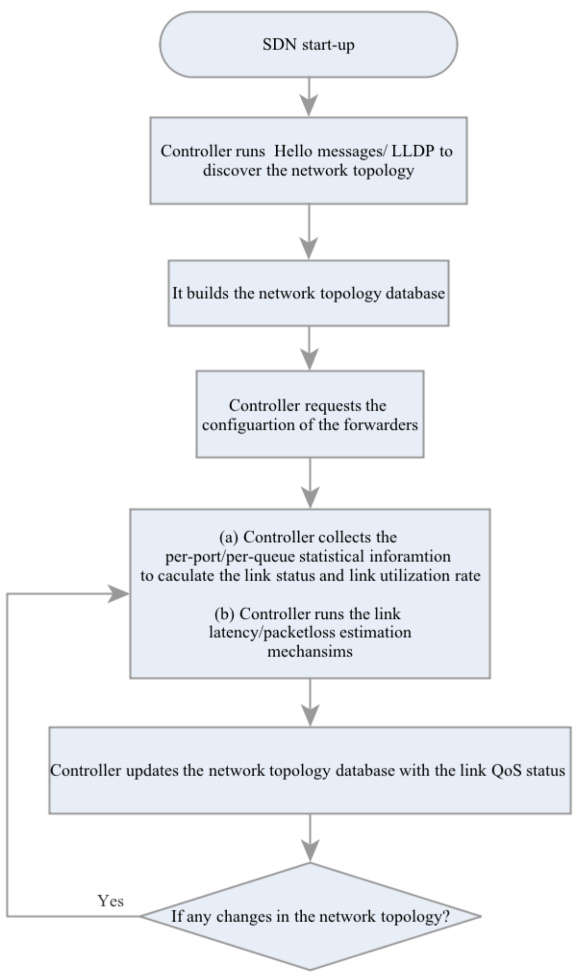
Since SDN controller supports multiple algorithms, the existing Best-Effort algorithms could be used for the application without the specific needs, in order not to do the complex calculation and energy/CPU resource consumption.

Block which is dark-blue colored in Figure 3.13 performs the routing path calculation for the application with specific QoS needs. The designed routing algorithm takes the variety of information in terms of network status, application needs, and network policy to calculate the best possible path considering the business and resource constraints. We are going to provide a mathematical model for this algorithm in Chapter 4.

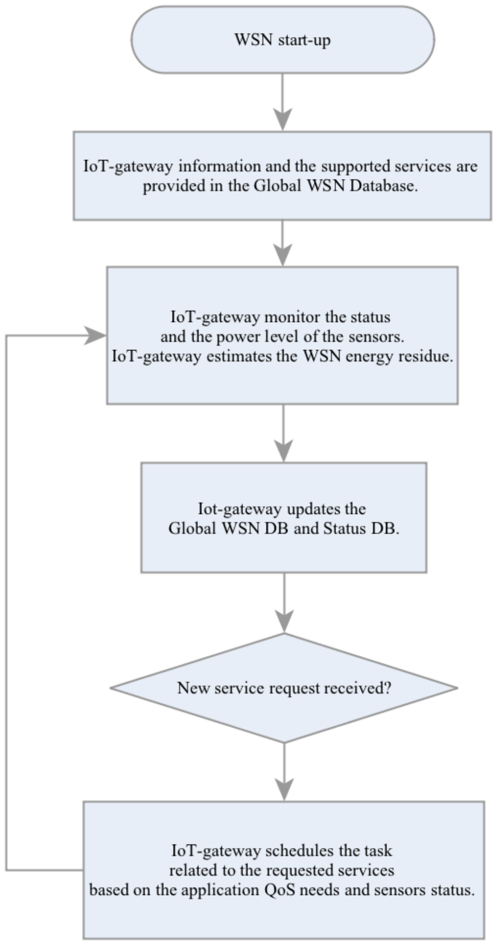
## 3.5 Architecture advantage

The dissertation studies the IoT application requirements from QoS point of view and it proposes the framework to manage their requirement through transport and sensing infrastructure. We describe the characteristics and advantages of our proposed framework from three perspectives:

**Architectural perspective.** The proposed framework has been mapped onto both the four-layer IoT architecture and SD-IoT architecture. The SDN control layer/management


SDN start-up
Controller runs Hello messages/ LLDP to discover the network topology
It builds the network topology database
Controller requests the configuartion of the forwarders
(a) Controller collects the per-port/per-queue statistical inforamtion to caculate the link status and link utilization rate
(b) Controller runs the link latency/packetloss estimation mechansims
Controller updates the network topology database with the link QoS status
If any changes in the network topology?
Yes

(a)


WSN start-up
IoT-gateway information and the supported services are provided in the Global WSN Database.
IoT-gateway monitor the status and the power level of the sensors. IoT-gateway estimates the WSN energy residue.
Iot-gateway updates the Global WSN DB and Status DB.
New service request received?
IoT-gateway schedules the task related to the requested services based on the application QoS needs and sensors status.

(b)

Figure 3.12 (a)Core Transport and (b)Sensing Network Status Collection diagram

layer provides a software layer between the IoT infrastructure and applications. It works as the middleware and it enables the implementation of unified support services for the IoT system. In our design, the QoS module implemented over the controller provides the service support for QoS management in IoT framework: the end-to-end QoS routing across the SDN, and QoS-aware sensing network allocation and sensor assignment. The QoS needs of IoT application aims to be respected in each relevant layer, either core transport or sensing layer. And, the resources and the routing path are allocated dynamically per-demand, depending on the specific service requirements and network resources status. Consequently, this design is adaptive to any changes in network and application QoS needs.

The SDN northbound interfaces enable the enforcement of the quantified SLA-related QoS attributes directly in the resource allocation functions, which in the closed network is not

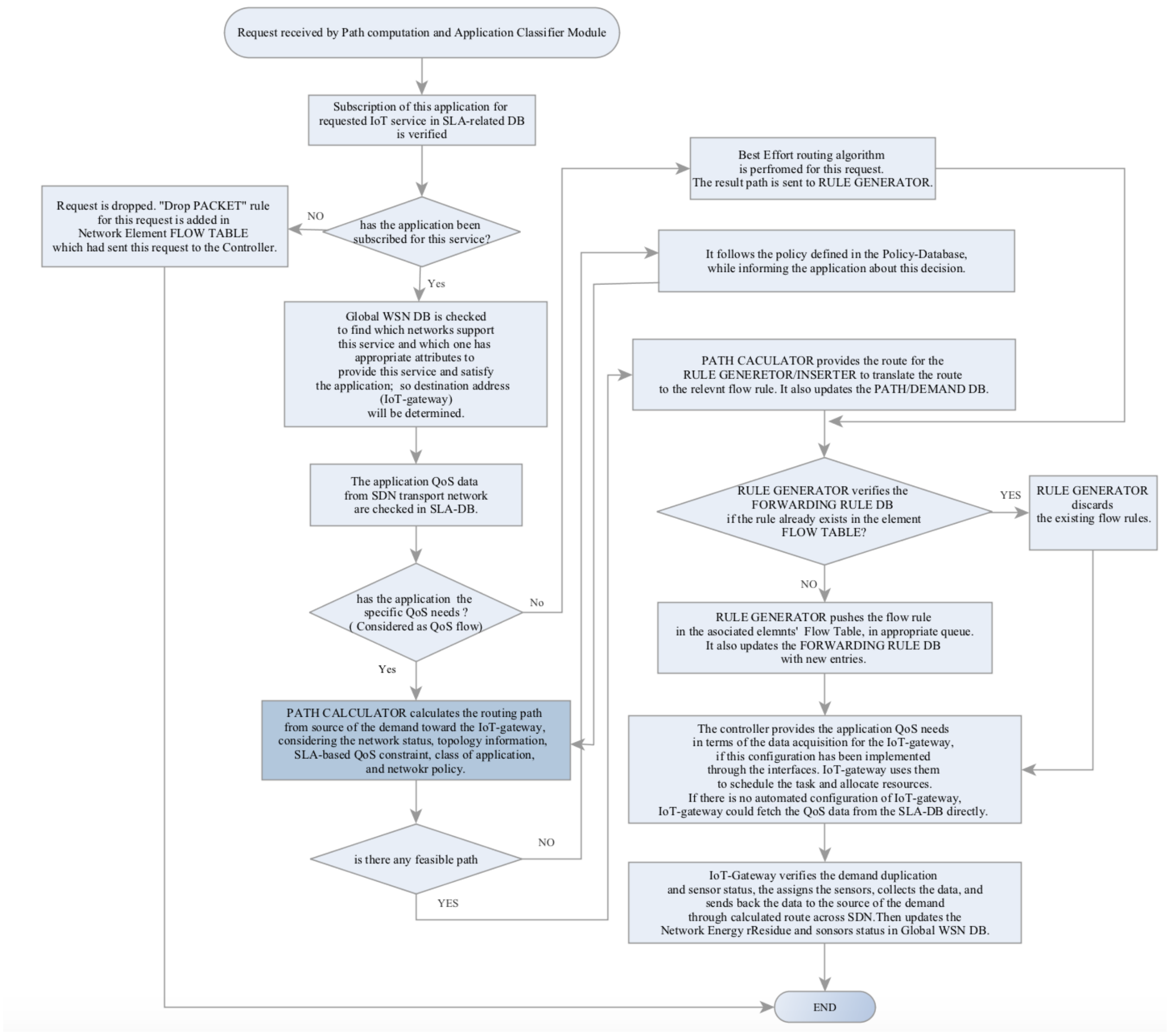

Request received by Path computation and Application Classifier Module
Subscription of this application for requested IoT service in SLA-related DB is verified
has the application been subscribed for this service?
NO
Request is dropped. "Drop PACKET" rule for this request is added in Network Element FLOW TABLE which had sent this request to the Controller.
Yes
Global WSN DB is checked to find which networks support this service and which one has appropriate attributes to provide this service and satisfy the application; so destination address (IoT-gateway) will be determined.
The application QoS data from SDN transport network are checked in SLA-DB.
has the application the specific QoS needs ? ( Considered as QoS flow)
No
Yes
PATH CALCULATOR calculates the routing path from source of the demand toward the IoT-gateway, considering the network status, topology information, SLA-based QoS constraint, class of application, and netwokr policy.
is there any feasible path
NO
YES
Best Effort routing algorithm is perfromed for this request. The result path is sent to RULE GENERATOR.
It follows the policy defined in the Policy-Database, while informing the application about this decision.
PATH CACULATOR provides the route for the RULE GENERETOR/INSERTER to translate the route to the relevnt flow rule. It also updates the PATH/DEMAND DB.
RULE GENERATOR verifies the FORWARDING RULE DB if the rule already exists in the element FLOW TABLE?
YES
RULE GENERATOR discards the existing flow rules.
NO
RULE GENERATOR pushes the flow rule in the asociated elemnts' Flow Table, in appropriate queue. It also updates the FORWARDING RULE DB with new entries.
The controller provides the application QoS needs in terms of the data acquisition for the IoT-gateway, if this configuration has been implemented through the interfaces. IoT-gateway uses them to schedule the task and allocate resources. If there is no automated configuration of IoT-gateway, IoT-gateway could fetch the QoS data from the SLA-DB directly.
IoT-Gateway verifies the demand duplication and sensor status, the assigns the sensors, collects the data, and sends back the data to the source of the demand through calculated route across SDN.Then updates the Network Energy rResidue and sonsors status in Global WSN DB.
END

Figure 3.13 QoS support workflow for query-driven application

possible due to the lack of such standard interfaces and programmable capabilities.

Since multiple programs could be implemented in the SDN controller, we are able to develop multiple algorithms and apply them in different conditions. we could use the best-effort routing path algorithm for the application without strict QoS needs. For the QoS-based applications, newly designed routing algorithm which considers the application constraint is applied. Since the framework uses the up-to-date information about the network status and application needs, it could provide the routing path dynamically over time.

Compared with the traditional QoS approaches such as IntServ and DiffServ, SDN-based core transport network resolves the limitation of the traffic differentiation and application classification according to their needs. As the IoT system grows, it might bring the new class of applications with different QoS needs. Our architecture is flexible and fast-adapted to adopt the new QoS paradigms due to the business changes.

Based on the SDN network status information and the different application request history, it would be easier to learn the traffic pattern and predict the future traffic trend, to extend the network and enhance its efficiency.

Implementing the IoT data preprocessing and aggregation process at the edge of the network within the local IoT-gateway not only enhances the transmission resource optimization and SDN network throughput, it speeds up the task scheduling and data acquisition for the user. The OpenFlow-enabled IoT-gateway also enables the sensing network programmability. The centralized algorithms and mechanisms in the IoT-gateway could be reprogrammed based on business and application needs, or when new optimized methods and solution are invented to manage the sensor resources in terms of clustering, routing, and task scheduling.

**Network perspective.** Network resource management in the globally distributed network such as transport network and the Internet is extremely complicated. One of the main benefits of the SDN-based core transport network is that it simplifies the network operation and management, compared with the traditional IP-based network which is defined in isolation and each vendor-dependent protocol only addresses a specific problem. We could leverage from SDN and its capabilities to develop customized network control and management services for core transport network. The controller as a centralized brain of SDN provides the single global map of the network and it abstracts the core transport network topology from the application layer. It enables the intelligent, and agile decisions making regarding flow direction, control, and speedy network reconciliation when a link fails. The SDN controller can run multiple algorithms simultaneously in the field of network operation and maintenance. Therefore, network developer could deploy a wide range of the customized network application in terms of security, monitoring, performance management and fault-diagnosis. Also, scalability can be improved by centralizing the controller, where there is more global and less detailed view of the network elements.

Our framework collects dynamically the transport network topology and links status information, it increases the awareness of the network resource status at any given time. The design routing path excludes the higher utilized and congested links from the logical network topology when computing the route across SDN. It aims to make the link utilization balanced and it prevents the congestion probabilities in the transport network. It increases the transport network availability, accordingly the IoT service availability. Suppose that the IoT sensing resources are available but the transport network is congested, it leads to failing the data transfer.

Furthermore, the centralized QoS management function does not deal with low-level configuration of data plane network elements. All the information such as SLA and network policy are specified in abstracted level. So, reconfiguration of low-level settings in the network elements is not needed. All lead to saving a lot of resources, workforces and time.

**Application perspective.** The Internet of Things (IoT) is placing new demands on network infrastructure due to the diverse application domain. With this architecture, IoT applications can customize their own QoS requirements in terms of data acquisition and transmission. The designed path computation and QoS management functions compute a path respecting to the application QoS constraints which increases the user satisfaction, as well as, enabling of innovative application deployment.

Furthermore, using the queue policy to classify different applications based on their QoS needs and the operation criticality guarantees the quality of service for high priority demands when multiple demands fight the available shared resources.

Also, the collection of the current state of the network elements and being notified in case of changes or failure, these approaches help the SDN controller to be aware of the current network status and the occurred events such as link up/down or the node join/leave. Network-state awareness and its involvement in the design of the routing computation algorithm decrease the possibility of link congestion and increase the network availability and throughput. Also, status-aware QoS-support resource allocation algorithm in sensing network fits the task and sensor data into application characteristic and needs. Additionally, the softwarization and centralization of the network services make the system to be converged with the changes. All of these approaches impact on the application satisfaction index (such as Quality of Experiences).

# CHAPTER 4 MATHEMATICAL MODELING

Quality of Service (QoS) in IoT as one of critical factor needs more research in terms of QoS implementation, management and optimization so that the applications could be served by the acceptable performance level while system resources utilization is maximized. In the previous chapter, we explained the proposed QoS support framework within IoT architecture and the QoS-aware resource allocation workflow within the framework. One of the key components of the architecture is **Path Calculator** which includes the routing algorithm to determine the best possible path across the core transport network for any demand. In this chapter, we focus on the route optimization problem within capacitated [1] SDN network and present a new mathematical modeling for the routing algorithm.

Our modeling approach is considered as multi-criteria approach since the model makes the decision about the routing path for the application traffic taking into consideration multiple constraints imposed by network situation and policy, and application. The result which it produces are the per-application routing paths from a set of origin locations to a set of destinations while maximizing network throughput. It is worth noting that the core transport network topology and link status are regularly gathered and updated by SDN controller in our proposed framework and the values are used in the path calculation process.

## 4.1 Multi-Commodity Constraint-based Routing Path Flow problem (MCCRPF)

The mathematical modeling is based on the well-known Network Design Problems (NDPs): Multi-Commodity Flow Problem (MCFP) and Constrained-Based Routing(CBR) which are being addressed by researchers and enterprises for years.

The term Multi-Commodity (opposed to a simpler single-commodity) is related to the fact that multiple demands could simultaneously arrive in the system and ask for routing resources in the network, which is very common in the communication and computer networks as well as IoT system. In the multi-commodity environment, each commodity has a unique set of characteristics and the commodities are not interchangeable ; meaning that system cannot satisfy a demand for one commodity with another commodity. The objective of the MCFP problem is to flow the different traffic flows (demands) from various sources to the distinct destinations through the network at minimum cost without exceeding the network link capacities.

1. If the network capacity is given, it is called a capacitated network.

Constraint-based routing (CBR) denotes a class of routing algorithms that base path selection decisions on a set of requirements or constraints, in addition to the destination. These constraints may be imposed by administrative policies, or by Quality of Service (QoS) requirements. Constraints imposed by QoS requirements, such as bandwidth, delay, or loss, are known as QoS constraints, and the associated routing is known as QoS routing. If there are no bandwidth constraints in a network, each pair of nodes could communicate over the shortest path between them to have the minimized delay and cost. Hence, in real capacitated networks, all network links have a bandwidth constraint. Besides, we consider the application delay and packet loss constraints in our formulation. This is known as constraint shortest path(CSP) problem as an extension of the shortest path problem. The objective is a minimum-cost feasible solution for the Constraint based routing (CBR) problem to find the cheapest possible way of sending a certain amount of flow through the network.

Therefore, we suppose to have a network of interconnected nodes where each link has a dedicated capacity. Assuming that all network nodes are OpenFlow-enabled and connected to one centralized SDN-controller. To start with, we give some definitions and notation used for the remainder of the dissertation.

The SDN-based core transport network (same as traditional computing network) can be described by a strongly connected graph $G = (V, E)$ where $V, V = \{1, 2, ..., v\}$ denotes the set of nodes (OpenFlow-enabled network elements) and $E, E = \{(i, j) : i, j \in V, i \neq j\}$ denotes the set of edges which refer to the bi-directional links between OpenFlow network elements. Each link $(i, j)$ has the associated maximum bandwidth $B_{ij}$, available bandwidth $b_{ij}$, delay $d_{ij}$ [2], and packet loss ratio $pl_{ij}$.

$K, K = \{1, 2, ..., k\}$ and $|K| = k$, represents the set of different commodities, also called demands, to be routed on the graph. For each demand $k \in K$, three parameters are given: $S^k$ as the source of the demand, $T^k$ as the destination of the demand and $F^k$ as the positive demand volume. Demand volume represents either the traffic volume or the required bandwidth between a pair of nodes. As an important point, the unit of the demand volume needs to be consistent with the unit of link capacities which could be Megabit per second (Mbps) or packets per second (pps). Let $D^k_{SLA}$, $PL^k_{SLA}$, and $B^k_{SLA}$ be the acceptable values of delay, packet loss ratio, and average required bandwidth, respectively, which are agreed in SLA for service $k$. The trio ($S^k$, $T^k$, $F^k$) plus $D^k_{SLA}$, $PL^k_{SLA}$, and $B^k_{SLA}$ are considered as the input for routing path algorithm. The algorithm takes this information for each demand as well as the network link information ($b_{ij}$, $d_{ij}$, and $pl_{ij}$) and calculates the best possible path $p^k$ for each new-arrival demand $k$ across the SDN network with minimum flow cost. This is also a good

2. Delay means the total link delay consisting of processing, propagation, transmission, and queuing delay.

time to point out that the term path is used as the finite sequence of network links which connect a sequence of distinct nodes, from one node to another.

The system could provide multiple paths for any demand $k$ aiming not to violate the application QoS limits. All determined paths for service $k$ are from source $S^k$ to the destination $T^k$ so that each one routes a portion of the whole demand volume $F^k$. We assume that there exists no pair of flows with the same origin and destination.

In the following paragraphs, we mathematically model the objective and constraints of our problem.

**Objective function:** The optimization objective is to route all the flows in the network with the minimum cost. Equation 4.1 as the objective function represents the flow cost minimization which depends on the cost of the links determined for all $K$ demands traffic. In the literature, this formulation is called node-link formulation. $C_{ij}$ is the unit cost of link $(i, j)$ and $X_{ij}^k$ as the variable of our model is the amount of volume corresponding to the demand $k$ routed on the link $(i, j)$.

$$Minimize \sum_{(i,j)\in E} \sum_{k} C_{ij} X_{ij}^k \tag{4.1}$$

**Constraint function:** We introduce the several types of conditions which are imposed by the network and application so that the feasible solution must satisfy these constraints.

— Path delay constraint for each demand is defined in Equation 4.2, where $d_p^k$ is the end-to-end delay for the routing path $p^k$ determined for demand $k$, and $D_{SLA}^k$ is the maximum acceptable delay for demand $k$ agreed in SLA.

$$d_p^k \leq D_{SLA}^k \tag{4.2}$$

— Path packet loss constraint for each demand is defined in Equation 4.3, where $pl_p^k$ is the total packet loss ratio for the routing path $p^k$ determined for demand $k$ and $PL_{SLA}^k$ is the maximum acceptable packet loss for demand $k$ agreed in SLA.

$$pl_p^k \leq PL_{SLA}^k \tag{4.3}$$

— Path capacity constraint which must be satisfied by the path $p^k$ for demand $k$, is formulated in Equation 4.4:

$$b_p^k \geq B_{SLA}^k \tag{4.4}$$

, where $b_p^k$ is the bandwidth for the path $p^k$ for demand $k$ and $B_{SLA}^k$ is the minimum required bandwidth for demand $k$ agreed in SLA.

— Link capacity constraint is formulated in Equation 4.5. In the multi-commodity environment, each link could be part of multiple routing paths used by different commodities. Then, the summation of the volume of the different commodity in any link $(i, j)$ must be less than the available link bandwidth $b_{ij}$. This constraint could fulfill the context of the congestion management in the network. Since in the SDN network we could estimate the available link capacity through accessing the network elements' counters, the available link capacity could be considered instead of maximum link capacity.

$$\sum_{k \in K} X_{ij}^k \leq b_{ij} \quad , \forall (i,j) \in E \tag{4.5}$$

— Link utilization/load balancing constraint. To reduce the congestion probability and balance the traffic volume on the links, we aim to consider the link utilization in the path allocation process too. In general, utilization rate on the link is measured by dividing the current link load with maximum link capacity, in a unit of Percentage. With the network link status information gathered by SDN controller, the utilization of link $(i, j)$ could be calculated by Equation 4.6.

$$Utilization\ rate\ in\ link(i,j) = \frac{B_{ij} - b_{ij}}{B_{ij}} \quad , \forall (i,j) \in E \tag{4.6}$$

To keep balanced the traffic distribution across the network links and to avoid the congestion, we consider the limit for the link utilization rate. The routing algorithm excludes the links with higher link utilization than the limit from the path calculation scenario to prevent the congestion and have the balanced load. In general, the link with utilization rate more than about $75\% \sim 80\%$ considered as the congested links, so the utilization threshold could be set in this range [104]. In our case, the default value could be pre-defined in the **policy database** and alternatively, SDN controller could dynamically adjust the link utilization limit based on the network situation such as traffic volume and demand arrival rate. Also, if in the worst case due to the traffic burst, no path does not satisfy the capacity condition by considering the pre-defined link utilization limit, it could be increased carefully.

If $U_{Threshold}$ represents the link utilization limit in our SDN network, we could have Equation 4.7 as the utilization limit formula:

$$(\sum_{k \in K} X_{ij}^k + B_{ij} - b_{ij})/B_{ij} \leq U_{Threshold} \quad , \forall (i,j) \in E \tag{4.7}$$

and consequently, load balancing principle in our model will be:

$$\sum_{k \in K} X_{ij}^{k} \leq b_{ij} + (U_{Threshold} - 1) \times B_{ij} \quad , \forall (i,j) \in E \tag{4.8}$$

— Flow Conservation Law. This law states that total incoming flow into each node in the network is equal to the total outgoing flow from that node, except for the source and destination nodes of flow. If the considered node is the source of the demand, the total outgoing flow minus the total incoming flow must be equal to the demand volume. If the considered node is the destination of the demand, the total incoming flow minus the total outgoing flow must be equal to the demand volume [105]. We formulate Flow Conservation Law in our network as Equation 4.9 which guarantees the same bandwidth in all links across the determined paths:

$$\sum_{(i,j) \in E} X_{ij}^{k} - \sum_{(j,i) \in E} X_{ji}^{k} = \begin{cases} F^{k}, & \text{for } i = S^{k} \quad (4.9a) \\ -F^{k}, & \text{for } i = T^{k} \quad (4.9b) \\ 0, & \text{for } i \neq S^{k} \text{ and } i \neq T^{k} \quad (4.9c) \end{cases}$$

## 4.2 Link cost design based on multiple metrics

In this section, we describe the link cost metrics used in our model. The objective is to weight the links based on the associated metrics to determine the cost of the various paths and accordingly to realize whether one path should be chosen over another. In traditional network scheme, most of the designed routing protocols consider just one QoS-related parameter (like packet loss probability, bandwidth, jitter, and delay) or another parameter such as the number of hops and link length in path decision process. For instances, OSPF (Open Shortest Path First) as the most known routing protocol in traditional IP networking, uses the link bandwidth as the link cost metric in the shortest path calculation algorithm. (see Appendix B)

The link cost metrics considered in our model are bandwidth, packet loss ratio, and delay. It is represented as a weighted sum of these metrics in Equation 4.10, where $C_{ij}$ represents the unit cost of the link $(i, j)$, the metrics $b_{ij}$, $pl_{ij}$, and $d_{ij}$ refer to the available link bandwidth, packet loss ratio, and delay in link $(i, j)$, respectively.

$$C_{ij} = \alpha \times b_{ij} + \beta \times pl_{ij} + \gamma \times d_{ij} \tag{4.10}$$

The coefficient $\alpha$, $\beta$, and $\gamma$ are the scaling factors of each metrics with the relation expressed in Equation 4.11. So, each metric could have the different weight to give a priority to a particular one.

$$\alpha + \beta + \gamma = 1 \qquad , 0 \leq \alpha, \beta, \gamma \leq 1 \tag{4.11}$$

This computation allows weighting the network links based on the importance of the delay, packet loss and bandwidth for a particular application regarding the application classification approach and application sensitivity. For example, Traffic Congestion Control and Monitoring application, served by camera or sensors, needs the strict end-to-end delay requirements and very low packet loss. Camera surveillance application needs high bandwidth. On the contrary, an analytical data application doesn't have any strict requirements. So, the priority given to the scale factors $\alpha$, $\beta$, and $\gamma$ could be different in each case. Therefore, in our model, we consider the dynamic metric for the link cost depending on the type of application and its requirements mentioned in **SLA-related QoS Database**.

Among these quality-related metrics, some are positive such as bandwidth: meaning the higher the value, the higher the service quality. Some are negative such as delay and packet loss ratio: meaning that higher the value, the lower the service quality. Additionally, each metric has a different unit. Daley unit is second, bandwidth unit is bps and finally, packet loss ratio is a digit in the percentage format. To express a weighted sum of independent metrics, the values of these metrics need to be adjusted to a notionally common scale. We use *Feature scaling* method [106] to normalize the range of these independent metrics. This method rescales the range of all values and brings all into the range $[0, 1]$. The general formula for *Feature scaling* method is given as:

$$\acute{x} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.12}$$

, where $x$ is the original value and $\acute{x}$ is the normalized value, assuming $x$ has the limited range $[x_{min}, x_{max}]$. Equations 4.13 $\sim$ 4.15 presents the normalization formulas used in our model:

$$\acute{b_{ij}} = \frac{b_{max} - b_{ij}}{b_{max} - b_{min}} \qquad , b_{min} \leq b_{ij} \leq b_{max} \qquad , b_{max} \neq b_{min} \tag{4.13}$$

$$\acute{pl_{ij}} = \frac{pl_{ij} - pl_{min}}{pl_{max} - pl_{min}} \qquad , pl_{min} \leq pl_{ij} \leq pl_{max} \qquad , pl_{max} \neq pl_{min} \tag{4.14}$$

$$\acute{d_{ij}} = \frac{d_{ij} - d_{min}}{d_{max} - d_{min}} \qquad , d_{min} \leq d_{ij} \leq d_{max} \qquad , d_{max} \neq d_{min} \tag{4.15}$$

Considering the fact that higher available bandwidth leads to the lower link cost, consequently to the lower path cost, 4.13 calculates the normalized value of available link bandwidth $b_{ij}$, whereas $b_{max}$ and $b_{min}$ represent the maximum and minimum range for link bandwidth in the core transport network, respectively. They could have a constant value or dynamically adjusted value based on the network topology information at any given time. There is a direct relationship between packet loss ratio and cost of the link so that lower link packet loss ratio, lower cost of the link. Equation 4.14 computes the normalized value of the packet loss ratio $pl_{ij}$ on the link $(i, j)$. The parameters $pl_{min}$ and $pl_{max}$ refer to the minimum and maximum range for link packet loss ratio in the network, $pl_{min}$ could be considered as zero and $pl_{max}$ could have the constant value or dynamically adjusted value based on the packet loss ratio information in the given network. Similarity, link delay has the direct relationship with the cost of the link. Equation 4.15 represents the normalized value of the link delay. The parameters $d_{min}$ and $d_{max}$ are lower and upper bound of link delay, respectively. The lower bound could be zero and upper bound could be any constant value regarding the link delay across the given network. Noting that if upper limit in any of these functions is equal to the lower limit which is rarely possible in the network, the value of the function will be equal to 1.

## 4.3 QoS parameters definitions and calculations formula

To calculate the cost of the link, we employ the link characteristics in terms of QoS indicators including packet loss, delay, and bandwidth. In this section, we explain how the link QoS parameters are calculated in the network. The path QoS parameters can be derived from the combination of the same parameters related to the links which make the path. In general, there is three basic composition rule as the *Additive metric*[3], *Multiplicative metric*[4] and *Concave metric*[5] [107, 108]. Further, we describe how to calculate the path QoS parameters in detail.

3. Additive metric is calculated by adding the metric of each link, this principle is applied for the delay, hop count, and cost.

4. Multiplicative metric is calculated by multiplying metrics of each link, it is applied for reliability and loss.

5. Concave metric is determined by selecting minimum or maximum value of each link, bandwidth follows this rule.

### 4.3.1 Delay

Delay in the packet-switch network is defined as the average time for a block of data to go from one end-node to another end-node. Four types of delays contribute to the overall delay on the link $(i, j)$ as demonstrated in Figure 4.1 and Equation 4.16. [109]

$$d_{ij} = d_{Transmission} + d_{Propagation} + d_{Processing} + d_{Queuing} \tag{4.16}$$

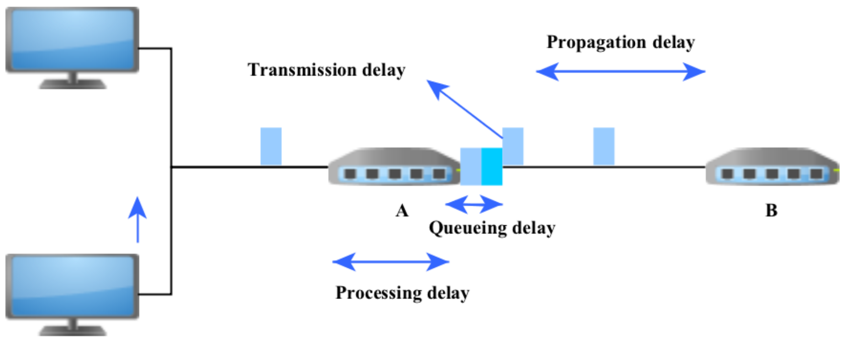


Figure 4.1 Delay in packet switch network

**Transmission delay** is the time it takes to transmit data on a link. It is determined by Equation 4.17:

$$d_{Transmission}[sec] = \frac{packet\ size\ [bit]}{link\ bandwidth\ [bps]} \tag{4.17}$$

*(on the order of* $1 \times 10^{-6}$ *seconds to* $1 \times 10^{-3}$ *seconds)*

**Propagation delay** is the time takes for a transmitted bit to travel from one end of a link to the other end. It depends on the signal speed of the transmission medium and the length of the link (meter). This type of delay is most considerable delay component in WAN (Wide Area Network).

$$d_{Propagation}[sec] = \frac{link\ length\ [m]}{link\ propagation\ speed\ [mps]} \tag{4.18}$$

*(on the order of* $1 \times 10^{-6}$ *seconds)*

**Processing delay** is the time to check packets for bit errors and look up the routing table for output determination before passing them to the output queue. Its range is $1 \times 10^{-6}$ seconds or less which is often negligible in the powerful network element.

**Queuing delay** is the time packet spend waiting in output buffers and it depends on intensity and nature of traffic arriving at queues. The Queuing delay can vary significantly from packet to packet because the number of the earlier-arriving packets in the queue which are waiting for transmitting across the link will affect its waiting time on queue.

Queuing delay is the most complicated component of total delay. Since packet arrivals rate and packet lengths are random, the prediction of the network behavior and queuing delay become a complex task. A huge number of papers and books have discussed the queuing model through employing *Queuing Theory*, *Poisson*, and *Markov* processes to make the queue lengths and waiting time predictable in the traditional IP network. The same techniques are being verified in the SDN environment through research and studies. For instances, authors in [110] argue that M/M/1 queue model for SDN controller and M/G/1 queue model for network elements could improve the packet sojourn time and the performance of the system. Depending on the queuing model, the queuing delay would follow different rules and principles. Equation 4.19 gives the high-level intuitive formula of average queuing delay. The average queue length depends on the load factor, which is the ratio of the attempted link transmission rate to the link maximum transmission rate.

$$d_{Queuing}[sec] = \frac{packet\ size\ [bits] \times queue\ length}{link\ bit\ rate[bps]} \tag{4.19}$$

*(on the order of* $1 \times 10^{-6}$ *seconds to* $1 \times 10^{-3}$ *seconds)*

Delay metric is an additive metric. So, total delay in the path from one source to the destination is calculated by the summing of the delay on each link of the path. Equation 4.20 formulates the delay in the path $p$ where $d_p$ refers to the path delay and $d_{ij}$ refers to the delay of each link $(i, j)$ which path $p$ is composed of.

$$d_p = \sum_{(i,j)\in p} d_{ij} \tag{4.20}$$

### 4.3.2 Packet loss ratio

Packet loss is also associated with quality of service consideration. It determines the failure of one or more transmitted packets from the sender to the destination during a specific time interval. The average packet loss can depend on many points along a TCP connection such as physical transmission error and link capacity limitation. Buffer size at the destination or even at the intermediate nodes could affect the packet loss probability too. If it has not properly been designed by considering the near-real traffic arrival rate, the node starts to drop the packets because of congestion and lack of space in the queuing buffer. Therefore, buffer size could be one of the bottlenecks in reducing TCP throughput. If the link has enough bandwidth, the available buffer size in ingress node causes the limitation too.

Packet loss ratio between a pair of nodes is calculated by Equation 4.21:

$$Packet\ loss\ ratio = \frac{number\ of\ sent\ packet\ from\ source - number\ of\ received\ packet\ in\ destination}{number\ of\ sent\ packet\ from\ source} \times 100\% \tag{4.21}$$

Referring to the proposed queuing configuration in Table 3.2 in Chapter 3, three different queues are configured in the OpenFlow network elements based on application classification approach. However, the queuing mechanism has a direct impact on the packet loss of the different type of classes, Equation 4.22 could be used to estimate packet loss ratio on the link $(i, j)$:

$$pl_{ij} = \begin{cases} (T_{PQ} + T_{Q1} + T_{Q2} - b_{ij})/(T_{PQ} + T_{Q1} + T_{Q2}), & \text{for } (T_{PQ} + T_{Q1} + T_{Q2}) > b_{ij} \quad (4.22a) \\ 0, & \text{for } (T_{PQ} + T_{Q1} + T_{Q2}) \leq b_{ij} \quad (4.22b) \end{cases}$$

, where $T_{PQ}$ refers to the number of packets waiting in the queue PQ, $T_{Q1}$ as the number of the packets waiting in the queue Q1, $T_{Q2}$ as the number of packets in the queue Q2, and $b_{ij}$ as the available bandwidth on the link $(i, j)$.

Packet loss is a multiplicative metric. Packet loss ratio along the path is determined in Equation 4.23, where $pl_p$ refers to the packet loss ratio for path $p$ and $pl_{ij}$ refers to the packet loss ratio for a particular link $(i, j)$ of the path $p$: Packet loss is a multiplicative metrics. Packet loss ratio along the path is determined in Equation 4.23, where $pl_p$ refers to the packet loss ratio for path $p$ and $pl_{ij}$ refers to the packet loss ratio for a particular link $(i, j)$ of the path $p$:

$$pl_p = 1 - \prod_{(i,j)\in p} (1 - pl_{ij}) \tag{4.23}$$

If the packet loss ratio in the network links is very small and close to zero, packet loss measure could be considered as an additive measure and can be approximately simplified by Equation 4.24: [109]

$$pl_p = \sum_{(i,j)\in p} pl_{ij} \tag{4.24}$$

### 4.3.3 Bandwidth

Bandwidth describes the rate at which data can be transferred on the link. It determines the efficiency and speed of data transmission activity. The link maximum bandwidth is defined based on the interface capacity. Bandwidth in a concave metric and the amount of bandwidth available in the path is affected by the slowest link found in the path. Equation 4.25 explains the bandwidth measurement in the path, where $b_p$ is the available bandwidth for the path $p$ and $b_{ij}$ is the available bandwidth for any link $(i, j)$ in the path.

$$b_p = min\{b_{ij} \; |(i, j) \in p\} \tag{4.25}$$

Available link bandwidth is calculated based on the link utilization rate $u_{ij}$ and maximum possible link capacity $B_{ij}$ through Equation 4.26

$$b_{ij} = B_{ij} \times (1 - u_{ij}) \tag{4.26}$$

Throughput is an important indicator of the performance and quality of a network connection, and it is different than bandwidth. Network throughput is defined as the total demand volume carried successfully by the network and typically measured in bits per second (bps). So, bandwidth is the capacity available for the data transfer but throughput is the actual data rate. Throughput is affected by the link utilization, loss rate, and link congestion, so its value depends on the network conditions.

## 4.4 Summary

In general, two main algorithms are used to compute the shortest paths from a single source to all the other destinations in a weighted graph: *Bellman-Ford's algorithm* [111] first proposed by Alfonso Shimbel in 1955 but published by Richard Bellman and Lester Ford Jr. in 1958 and 1956, respectively. *Dijkstra's algorithm* [112] which was conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959. Both are a graph search algorithm that solves the single source shortest path problem for a weighted graph. The only difference between two is that Dijkstra's algorithm cannot handle negative edge weights which Bellman-Ford handles. Bellman-Ford also tells us whether the graph contains the negative cycle. If a graph does not contain negative edges then Dijkstra's is always better.

The classical Dijkstra's algorithm [112] gets the network topology and the link weighs, then it determines the shortest distance (or the lowest cost) from the source node to every other node. This algorithm is the base of the current network routing protocols which the cost metric is based on the link status information.

In this chapter, we designed a new QoS support routing model, leveraging all advantages of the SDN-based core transport network. The Northbound interface makes SDN controller have access to the application layer and the databases to fetch directly the application preferences in terms of QoS attributes. The southbound interface (OpenFlow) enables it to collect the network link status from the network elements and provide the links data in terms of the packet loss, delay, available link bandwidth, and link load.

Our model would be an extension of the Dijkstra algorithm with more input data such as link status information and application QoS needs, as well as different link cost metrics. All these information could be available for our model because of the SDN controller capabilities. This model aims to propose least-cost status-aware and SLA-respected routing path across the SDN network. In the classical network, there is no way to fetch the status information directly from the network elements in a centralized and real-time way, and consequently the network topology and the QoS parameters. Therefore current routing path mechanisms do not consider the current network status e.g., packet loss, delay, or available bandwidth in the path calculation process. Besides, link cost metrics used in the routing mechanism are same for all the application type. But, our model assigns different path for the same data dynamically depending on the current network status and link cost metrics are diverse depending on the application type.

Decision-making framework and the procedure followed in our model are displayed in Figure 4.2 and Algorithm 1, respectively. The topology information, application QoS needs, and link weighting mechanism are taken into accounts to provide a new efficient algorithm to compute a minimum-cost path between pairs of network elements.

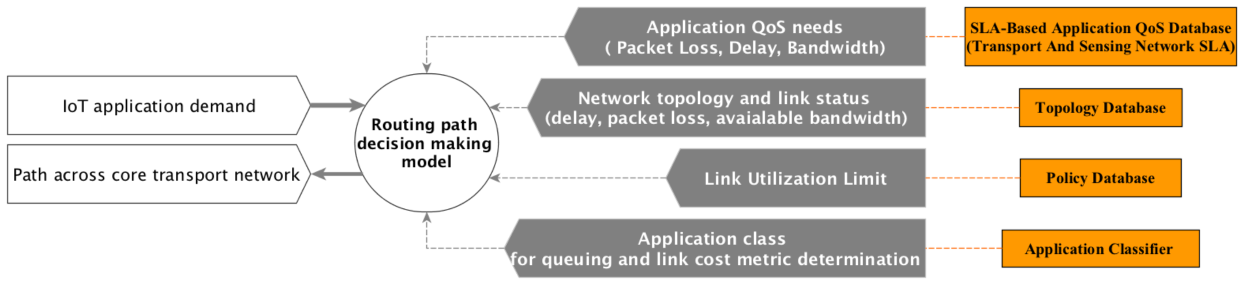

Application QoS needs
( Packet Loss, Delay, Bandwidth)
SLA-Based Application QoS Database (Transport And Sensing Network SLA)
IoT application demand
Routing path decision making model
Network topology and link status (delay, packet loss, avaialable bandwidth)
Topology Database
Path across core transport network
Link Utilization Limit
Policy Database
Application class for queuing and link cost metric determination
Application Classifier

Figure 4.2 Routing path decision-making framework

**Algorithm 1:** Routing path algorithm to find the least-cost possible path across the core transport network, taking into account application needs and network link situation

```
1  Procedure ;
   Input  : $G = (V, E)$ as the topology graph of the SDN network including nodes and
            bidirectional links: $V = \{1, 2, ..., v\}$ and $E = \{(i, j) : i, j \in V, i \neq j\}$
2  for $k$ in $K$ do
       Input  : Source $S^k$, Destination $T^k$ and Volume $F^k$
       Output : Paths from Source $S^k$ to Destination $T^k$
3  end
4  for $(i, j)$ in $E$ do
5      Read the link QoS parameters including $b_{ij}$, $pl_{ij}$, and $d_{ij}$;
6      Calculate the current link utilization rate;
7      Read the link utilization limit $u_{Threshold}$ ;
8      if Link utilization rate $>= u_{Threshold}$ then
9          It excludes this link from the logical network topology used to calculate the path ;
10     end
11 end
12 for $k$ in $K$ do
13     Read SLA-DB to have Max acceptable delay $D^k_{SLA}$, Max acceptable packet loss $PL^k_{SLA}$,
         and Min required bandwidth $B^k_{SLA}$;
14     Set the link cost metrics depending on the application class;
15 end
16 for $k$ in $K$ do
17     Make decision about the path based on the mathematical model;
18 end
```

# CHAPTER 5 EVALUATION AND RESULT

In Chapter 4, we have provided the mathematical model of the routing path computation in the SDN-based transport network. In this chapter, we aim to validate our proposed model to guarantee that it certainly enhances the function of QoS routing in the system. First, we explain the implementation of the model and the test scenarios. Then, we validate and analyze its performance through presenting the numerical results, compared to the currently-used routing method in the Internet and transport network.

## 5.1 Model implementation

To investigate the feasibility and the performance of the model, we implement the proposed MCCRPF problem in *AMPL*(A Mathematical Programming Language). *AMPL* [113] is an algebraic modeling language to prototype the mathematical models and describe the complex problems, e.g., optimization and scheduling problems. It also unifies the interface for setting both the problems and solvers for the linear and non-linear problems. Depending on the type of the problem and the mathematical formulation, the appropriate solver could be used to find the optimal or feasible solutions.

First, we put all together the expressions including objective, constraints, and variables as well as the parameters used to define the problem in *AMPL*. We represent the SDN-based core transport network by an oriented graph $G = (V, E)$ where $V$ is the list of network elements and $E$ is the set of links between each pair of network elements. We assume that $k$ is the number of new demands which enter the system simultaneously.

Sets:

— List of network elements: $1...v$, v $\in V$ ;

— List of bidirectional links between network elements: $(i, j) \in E \cup (j, i)$ ;

— Demands: $1...k$ ;

Parameters:

Network link status and policy parameters:

| | |
|---|---|
| $B_{ij} > 0$ | Maximum capacity on the link $(i, j), [Mbps]$ |
| $b_{ij} \geq 0$ | Available capacity on the link$(i, j), [Mbps]$ |
| $d_{ij} \geq 0$ | Delay on the link $(i, j), [Second]$ |
| $pl_{ij} \geq 0$ | Packet loss ratio on the link $(i, j), [Percentage]$ |
| $u_{Threshold} > 0$ | Link utilization limit on the link, $[Percentage]$ |

Service demand-relevant parameters:

| | |
|---|---|
| $S^k \in V$ | Source of demand $k$ |
| $T^k \in V$ | Destination of demand $k$ |
| $F^k \geq 0$ | Total demand volume $k$, $[Mbps]$ |
| $D^k_{SLA} \geq 0$ | Maximum acceptable delay for demand $k$, agreed in SLA |
| $PL^k_{SLA} \geq 0$ | Maximum acceptable packet loss ratio for demand $k$, agreed in SLA |
| $B^k_{SLA} \geq 0$ | Minimum bandwidth required for demand $k$, agreed in SLA |
| $P^k$ | Determined path across the network for demand $k$ |

Variable:

| | |
|---|---|
| $0 \leq X^k_{ij} \leq b_{ij}$ | Amount of demand volume $k$ on the link $(i, j), [Mbps]$ |

Objective function:

| Minimize $\sum_{(i,j)\in E} \sum_{k\in K} C_{ij} X^k_{ij}$ |
|---|
| $C_{ij} = \alpha \times b'_{ij} + \beta \times pl'_{ij} + \gamma \times d'_{ij} \quad , \forall (i, j) \in E$ <br> $\alpha + \beta + \gamma = 1 \qquad , 0 \leq \alpha, \beta, \gamma \leq 1$ <br> $b'_{ij}$, $pl'_{ij}$, and $d'_{ij}$ as the normalized value of $b_{ij}$, $pl_{ij}$, and $d_{ij}$, respectively. |

Subject to the constrains:

| |
|---|
| $\sum_{k\in K} X_{ij}^k \leq b_{ij},\ \forall (i,j)\in E$ |
| $\sum_{k\in K} X_{ij}^k \leq b_{ij} - (1-u_{Threshold}) \times B_{ij},\ \forall (i,j)\in E$ |
| $\sum\limits_{(i,j)\in E} X_{ij}^k - \sum\limits_{(j,i)\in E} X_{ji}^k = \begin{cases} F^k, & \text{for } i = S^k \\ -F^k, & \text{for } i = T^k, \quad \forall k\in K, i\in V \\ 0, & \text{for } i \neq S^k \text{ and } T^k \end{cases}$ |
| $\sum_{(i,j)\in E,P^k} d_{ij} \leq D_{SLA}^k,\ \forall k\in K$ |
| $\sum_{(i,j)\in E,P^k} pl_{ij} \leq PL_{SLA}^k,\ \forall k\in K$ |

In the offered mathematical formulation, the number of variables is $|E||K|$ and the number of constraints is $|E||K| + |K| + |E|$. Since the number of the QoS parameters used in our model is more than one, it is proven to be $N\rho$-complete [114] as the complex problem.

In optimization context, linear programming efficiently solves problems where the objective function and constraints are linear with respect to the decision variables. On the contrary, the problem is called a nonlinear programming problem if the objective function is nonlinear and/or the feasible solution is determined by nonlinear constraints. However in our model, the objective function is linear and the constraints are linear, except the delay and packet loss constraints. In each demand $K$, the delay and packet loss constraints employ the link delay and packet loss ratio independently to the exact volume of the demands in the link. Therefore, we convert $X_{ij}^k$ into binary representation 0 and 1 in these constraints. We structure the problem in both styles: nonlinear and linear. To solve the nonlinear style, we use *MINOS* [115] solver. *MINOS* is a software package for solving large-scale optimization problems (linear and nonlinear programs). *MINOS* is highly effective and numerically stable algorithms. It uses augmented Lagrangian methods to solve the nonlinear optimization problem. In case of the linear style, since the values of the variable $X$ is going to be discrete, it is classified as Mixed Integer Programming (MIP) problem. We pair *AMPL* with the *CPLEX* [116] to solve the problem. *CPLEX* is the optimization engine developed by IBM and it is used for mostly solving integer programming problems, very large linear programming problems and quadratic programming problems. *CPLEX* uses branch-and-bound algorithm to find the optimal solution for the Mixed Integer Linear Programming problem. Though the feasible solution found by both style were the same in most of the experiments we did, we analyze the results provided by the *CPLEX* in the next section.

## 5.2 Experiments and performance evaluation

In the concept of the packet-switched networking, a routing protocol specifies how network elements exchange information together, determines optimal network data transfer paths between network elements and facilitates the overall network topology understanding for each of network elements. In general, path determination includes two main steps: destination determination which is based on the destination IP address and, packet routing toward the destination which is based on the rules. In the traditional network, the rules could be defined either manually by the network administrator or automatically by the routing protocol and they are stored in the routing table. [117]

In the large networks, there is the possibility of having multiple paths between a particular pair of source and destination. Different routing protocols use different metrics to evaluate and differentiate between all available paths. Depending on the metrics used by the routing protocols, two different protocols might choose different paths to the same destination. Both Bellman-Ford and Dijkstra algorithms have enabled the development of different routing protocols. Bellman-Ford's algorithm has enabled the development of distance-vector routing protocols[1] while Dijkstra's algorithm has paved the way for the introduction of link-state routing[2]. Link-state routing protocols enable a router to build and track a full map of all network links while distance-vector protocols allow routers to work with less information about the network area. In both, the router learns about remote networks from the neighbor routers or the configuration to build the routing table. The characteristics of the widely-used routing protocols are provided in Appendix B.

The general idea in link-state routing protocol is that a cost (called also weight) is assigned to the network link based on the metrics and the less cost shortest path between a particular node pair is calculated. Open shortest-path first (OSPF) [118] is one of the well-established and widely-adopted link-state routing protocols in the traditional network. OSPF makes the routing decision based on the link bandwidth. Link cost calculation formula in OSPF is determined in Equation 5.2:

$$Interface\ cost = \frac{Reference\ bandwidth}{Interface\ bandwidth} \tag{5.2}$$

In Cisco product, the default reference bandwidth value in OSPF is $100Mbps$ (equal to $10^8 bps$). Hence, we have the following equation as the cost of link $(i, j)$: [119]

1. In Distance-vector routing protocols, routers compute the best path from information received from neighbors periodically. The information means the distance or metric to reach the remote network. It is recommended to be used in small networks

2. In Link-state routing protocols, routers inform all the nodes in a network of topology changes, and since the convergence speed is higher than distance vector routing protocol, it is recommended to be implemented in the large network.

$$cost_{ij} = \frac{100}{B_{ij}} \tag{5.3}$$

Referring to the link cost formulation in our model in Equation 4.10, multiple link metrics including delay, packet loss, and bandwidth are used. The QoS requirements of the IoT applications are not clearly defined because of the diverse, innovative, and data-centric nature of the application, so there is no standard SLA in the IoT. Referring to Table 3.2, for queuing purpose, we classify the application based on their sensitivity to the delay and bandwidth. To facilitate the implementation of our experiment environment, we map the IoT application classes onto the IoT data delivery model as Table 5.1:

Table 5.1 IoT application class

| **IoT Application** | **Application class** |
|---|---|
| Mission-critical, Event-related application | Delay-centric |
| Continious application (Query-driven, Real-time monitoring) | Bandwidth-centric |
| General application (Non-real time monitoring) | BE |

We design our model to assign dynamic cost metrics for different application classes. For the delay-sensitive application, packet loss and delay would be the metrics, for the bandwidth-sensitive application, the combination of the packet loss and bandwidth are considered in link cost estimation, and for the BE application, the combination of all three metrics would be used or the traditional less-complex best-effort routing protocol could be applied.

The approach used to implement the experiment is demonstrated in Figure 5.1. So, we characterize the application from different classes with different QoS requirements, then we evaluate the performance of the routing paths calculated by our model in terms of delay, packet loss and link utilization rate in several network topology, and compare with the characteristics of calculated paths by OSPF routing model.

The particular network topologies designed to run the test, ensure the path diversity between any pair of nodes. Therefore, we define the network topology and assign the maximum capacity, available bandwidth, delay, and packet loss ratio for network links. Also, we present the service demands specifying the source, destination, and volume as well as its QoS requirements in terms of delay, packet loss, and minimum bandwidth. The simulated demands are directed towards the bottlenecks to investigate latencies and throughput of the delay-centric and bandwidth-centric traffics, respectively. Both single-commodity and multi-commodity scenarios are investigated under the same network situation.

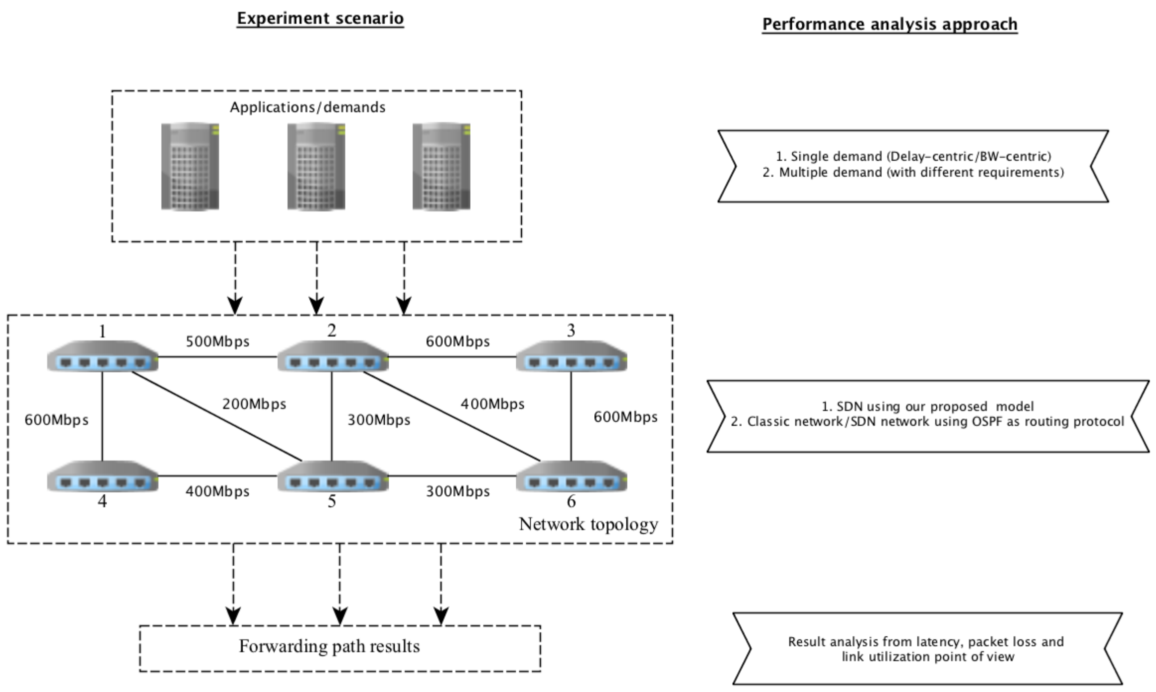

Experiment scenario
Performance analysis approach
Applications/demands
1. Single demand (Delay-centric/BW-centric)
2. Multiple demand (with different requirements)
1
2
3
4
5
6
500Mbps
600Mbps
600Mbps
200Mbps
300Mbps
400Mbps
600Mbps
400Mbps
300Mbps
Network topology
1. SDN using our proposed model
2. Classic network/SDN network using OSPF as routing protocol
Forwarding path results
Result analysis from latency, packet loss and link utilization point of view

Figure 5.1 Experiment and performance analysis scenario

The network topologies used in our experiments ( topology A, B, C, and D) and the associated link configurations are demonstrated in Appendix A. To do the normalization of the link cost metrics, we assign the lower-bound and upper-bound for each metrics as Table 5.2, based on the network link configurations of our topologies.

Table 5.2 QoS parameters limits for normalization

| | Minimum | Maximum |
|---|---|---|
| Link packet loss ratio range | 0% | 5% |
| Link delay range | 0 s | 0.0001 s |
| Link bandwidth range | 0 bps | 1000Mbps |

### 5.2.1 Single-demand scenario

The first experiment is in the single commodity environment. We characterize multiple delay-centric and BW-centric demands with the different source, destination and QoS levels in network topology A, B, and C. Each demand is defined individually and the behavior of our routing model and OSPF model are studied in terms of the network QoS metrics: delay, packet loss, and link utilization.

The path delay and packet loss ratio for all single demands characterized in Topology A, B, and C are demonstrated in the columns charts 5.2, 5.3, and 5.4, respectively. In the column charts, the horizontal axis contains the demands called *delay-centric* and *BW-centric* which point to the type of the demands.

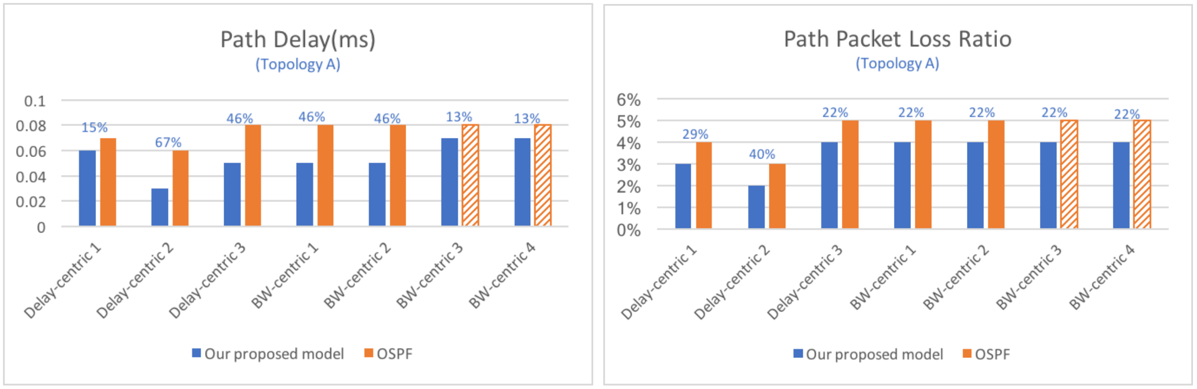


Figure 5.2 QoS attributes associated for the calculated path by the proposed model and OSPF - Topology A

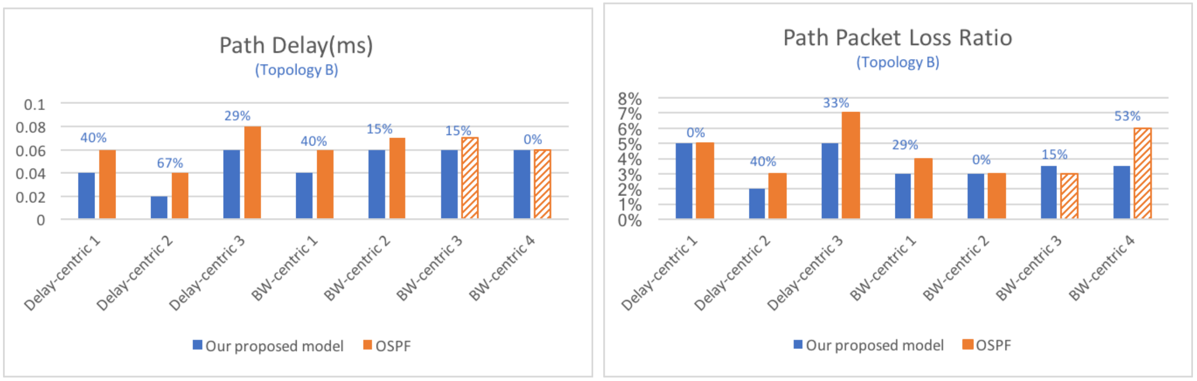
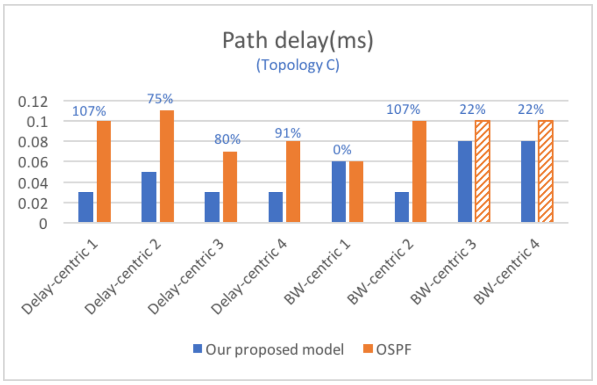
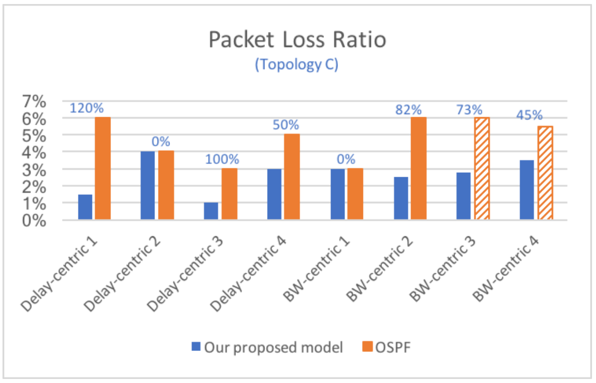


Figure 5.3 QoS attributes associated for the calculated path by the proposed model and OSPF - Topology B

The delay of the paths assigned for each demand are depicted in Figure 5.2a, 5.3a, and 5.4a. That is the estimation of the end-to-end delay that a data flow would suffer between the source and destination of the demand. The path loss ratio for each case are illustrated in Figure 5.2b, 5.3b, and 5.4b. As mentioned earlier, our model considers the delay and packet loss ratio as the metric of link cost for the delay-centric demands, and OSPF just considers



(a)



(b)

Figure 5.4 QoS attributes associated for the calculated path by the proposed model and OSPF - Topology C

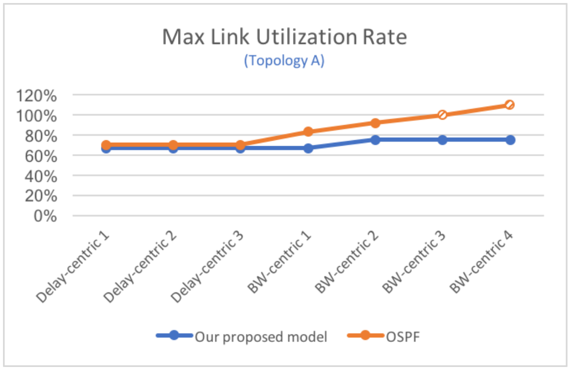
the link bandwidth as the link cost metric. Consequently, our model and OSPF could find different paths for the same demand in the same topology. Referring to the charts and the *delay-centric* columns, it can be seen that the delay and the loss rate for the demands are less in our proposed model compared to the OSPF. Thus, it is perceived that our model provides more optimized paths in terms of delay and packet loss ratio for this type of the applications. Therefore, the mission-critical application gets the requested service and data with minimized delay and error rate. To compare values determined by our model and OSPF, we calculated *PERCENT DIFFERENCE*[3] for each application and displayed in each columns in the diagrams. It shows the average improvement in our model compared to OSPF.

In general, the multimedia applications including the high-resolution images and videos need more bandwidths compared with the delay-centric and event-based application. So, we characterize multiple BW-centric applications with different throughput needs in each topology, as it can be seen in Figure 5.2, 5.3, and 5.4. Analyzing the results obtained for the *BW-centric* demands shows that our model assigns the less delay and less loss rate paths for these types of the demands too, although their most important concern is the bandwidth and throughput. Our model considers the link available bandwidth and packet loss ratio as the cost metric for the BW-centric demands. Less delay in the service delivery would boost the application satisfaction from the system performance.

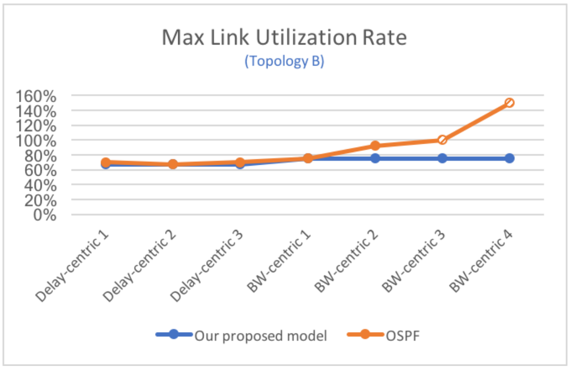
3. PERCENT DIFFERENCE as a percentage of their average value is calculated using the following Equation:

$$PERCENTDIFFERENCE = |value(OSPF) - value(our\ model)| / (((value(OSPF) + value(our\ model))/2) \qquad (5.4)$$

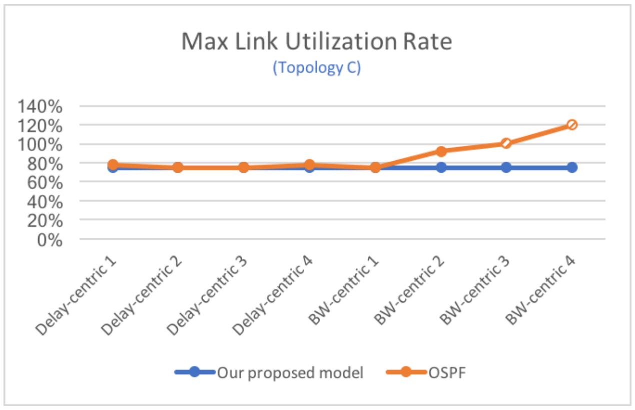
Besides, to examine the behavior of the proposed model about this type of application, we observe the link bandwidth utilization rate in each test cases. The test direction is that the demand volumes surpass the link available bandwidth and increase the rate of the congestion. The maximum link utilization across the network for each single-commodity demand in Topology A, B, and C are depicted in Figure 5.5a, 5.5b, and 5.5c, respectively.



(a)



(b)



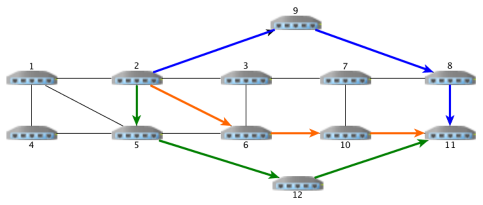
(c)

Figure 5.5 Maximum link utilization rate in the calculated paths by the proposed model and OSPF

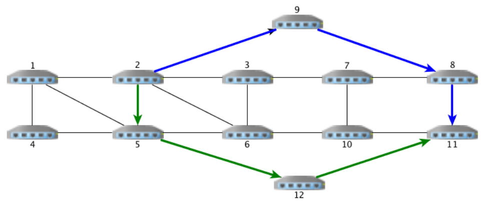
In our test, the demand volume for the delay-centric application is inconsiderable compared with the available link bandwidth. Conceivably, the link load is normal and stable for both models. In the case of the BW-sensitive application, we characterize the volume demand with the different bit-rate request in a way to go beyond what is available in the links. Considering that our model is aware of the current link bandwidth, and it calculates the current link utilization and considers the current situation when computing the paths for new incoming demands. Besides, we defined a limit for the link utilization rate within policy database. In our experiments, we set the link utilization rate to 75%. Therefore, our model uses multi-path approach to allocate the desired bandwidth for the demands while controlling the link utilization based on the pre-defined limit across the network. This mechanism could decrease the congestion rate and increase the network throughput since the congested links could augment the packet loss and delay for the demands passing through these links.

On the contrary, OSPF has not the knowledge about the available link bandwidth. OSPF just considers the link bandwidth information to determine the least-cost path. Between a pair of nodes, it forwards the flow through the same path which is the same way at all times. Therefore, the network experiences the link congestion. The lines in Figure 5.5a, 5.5a, and 5.5a show that link utilization rate goes beyond 100% in some of the test situations ( remarked with "pattern fills"). It means that OSPF assigns the path which has not enough available bandwidth to serve the demand request. Consequently, we displayed the relevant demands with "pattern fills" in Figure 5.2, 5.3, and 5.4 to declare that the real delay and packet loss would be more than the one determined in the charts. The congestion in one particular link affects the service delivery for all demands which are transferred through this link. Therefore, it leads to decreasing the network throughput and impacting significantly on the customer satisfaction.

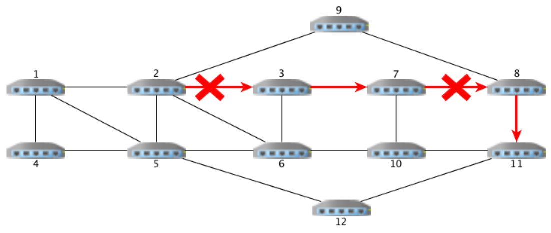
For instance, we detail the path allocated for the demand *BW-centric 4* in Topology C. The demand is defined from source 2 to destination 11 with high bit-rate volume. In the first place, our model follows the multi-path approach when one path could not satisfy the requested volume of the demand. Secondly, the defined link utilization rate is regarded by the model to keep the link utilization balanced across the network and prevent the congestion. Depending on the link utilization limit, different paths could be allocated for the same demands based on the current network situation. Figure 5.6a and 5.6b show the paths allocated for a particular demand by our proposed model under the same network situation but with different link utilization limit, 75% and 90%, respectively. When link utilization limit is set to 75%, three paths ( displayed by multiple-colored) are allocated, and in 90%, two paths ( displayed by multiple-colored) are determined by the demand. The utilization limit could be assigned dynamically based on the demand arrival rate and demand QoS needs ; meaning that the default value could be used as long as the demand SLA is not violated. In conclusion, our model distributes the demand volume in the multiple paths to first provide the required bandwidth, aiming to keep the delay and packet loss as less as possible which improve the demand satisfaction index. OSPF leads to having link congestion in high bit-rate demands since there is no knowledge about the current link bandwidth. Figure 5.6c explains that one path is allocated for the demand which causes the congestion in two links ( red-colored crossed line).



(a) Multi-path with $u_{Threshold} = 75\%$ in the proposed model



(b) Multi-path with $u_{Threshold} = 90\%$ in the proposed model



(c) Congested links in OSPF model

Figure 5.6 Paths allocated in case of high-BW demand in Topology C

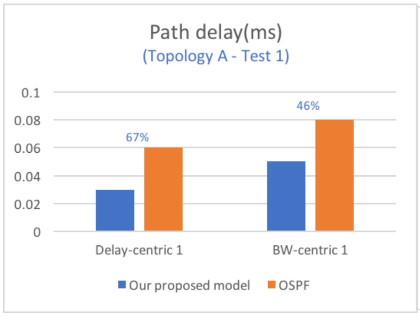
### 5.2.2 Multi-demand scenario

In this section, we investigate the behavior of our model in the multi-commodity environment. Similar to the experiments in the single-commodity scenario, we characterize different demands from both delay-centric and BW-centric classes, with different QoS requirements, but this time we study the QoS attributes of the paths when multiple demands are entered in the network simultaneously.

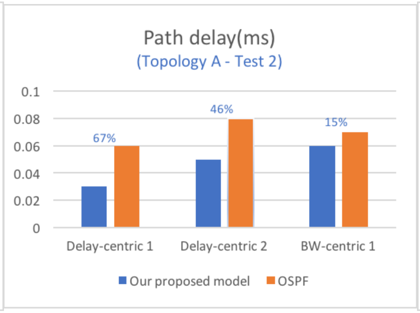
**Topology A.** We perform three different tests (*Test 1* to *Test 3*). The *Test 1* includes two different demands from different classes of the application (Daley-centric and BW-centric). In *Test 2* and *Test 3*, we add, respectively, one other delay-centric and BW-centric demand for the first test. The multiple demands come simultaneously into the system and the goal is to allocate the routing paths between the source and destination of each demand.

As illustrated in the Figure 5.7 and 5.8, it can be seen that the delay and packet loss of the paths associated with the delay-centric demands by our model are lower than the relevant attributes in the paths associated by OSPF model. It is observed that this statement can be extended to the BW-centric demands too. Then the optimized paths in terms of the delay and the loss rate are found by our model.

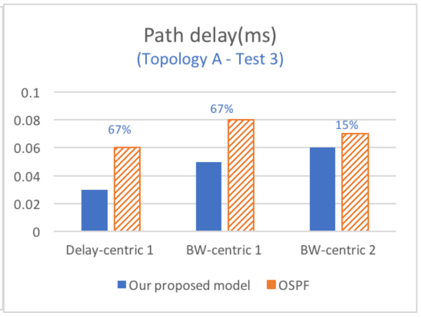
Figure 5.9 shows the maximum link utilization across the network in each test. It declares that our model keeps the link utilization rate stable in all three tests since the adjusted threshold are respected by the model. On the contrary, it expresses that the utilization rate



(a) Test1

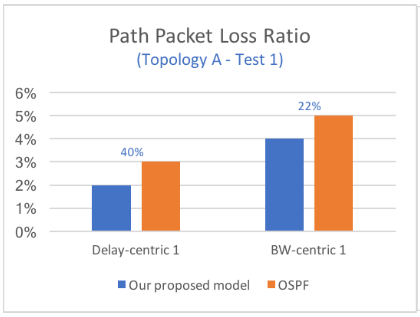


(b) Test2

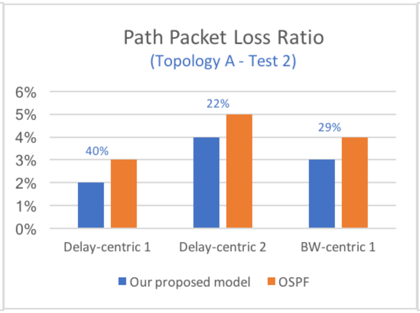


(c) Test3

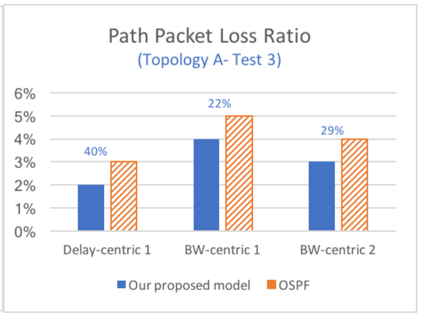
Figure 5.7 Path delay - Topology A



(a) Test1



(b) Test2



(c) Test3

Figure 5.8 Packet Loss Ratio of the paths - Topology A

in some of the links reaches 100% in *Test 3* when running the OSPF. When link utilization is more than 100%, the link is considered as the congested link and consequently, the demands passing through the congested link could not be served as desired and they may suffer from more delay and loss.

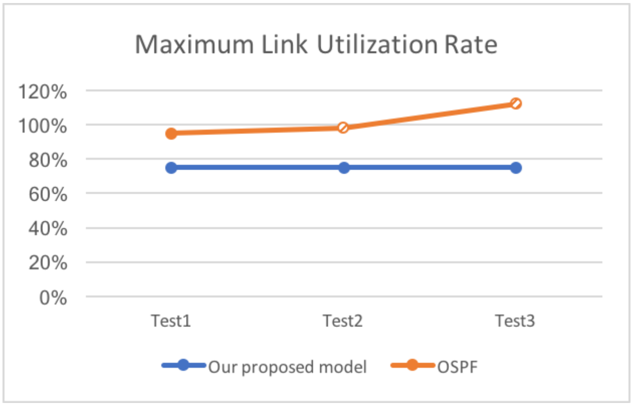


Figure 5.9 Maximum Link Utilization across the network in different tests - Topology A

To facilitate understanding the congestion scenario, we demonstrate details of the OSPF allocated paths and the congested links in *Test 3* in Table 5.3. The congestion happens when multiple demands are entered into the system and the higher bit-rate is demanded by them. In this test, *BW-centric 2* is the one which demands high volume and it leads to occurring the saturation in one link across the demand path. Since the other two demands (*BW-centric 1* and *BW-centric 1*) are also transferred through this link, their performances in terms of delay and loss could also be affected. Because of that, we displayed the columns of all three demands with "pattern fills" in Figure 5.7c and 5.8c ; insisting that they might suffer more delay and loss as computed. The effectiveness of the delay-sensitive demands might be degraded by undesired delay, or even the transferred data could be useless because of its late arrival. Though the queuing and priority scheduling policies implemented in the nodes could impact the behavior of the system in order to determine what demands to remove in priority in the congestion situation, the active demands and the network situation have the major impacts on the consequences.

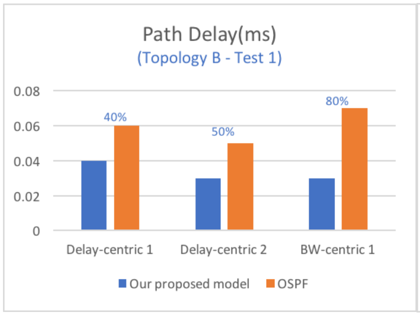
Table 5.3 Paths and congested links in *Test 3* when OSPF routing model is used - Topology A

| | **Demand class** | **Source /Destination** | **Computed path (OSPF)** | **Congested link** |
|---|---|---|---|---|
| Test 2 | Delay-centric 1 | 2 →4 | (2 5)(5 4) | (5 2)/(2 5) |
| | BW-centric 1 | 3 →4 | (3 2)(2 5)(5 4) | |
| | BW-centric 2 | 5 →3 | (5 2)(2 3) | |

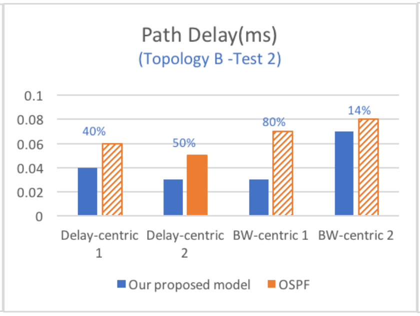
We conclude that our model provides the less-delay and less-loss paths for the demands, especially for delay-centric demands. The model performs somehow the load-balancing across the network links by keeping the link load under the defined threshold. Also, the model excludes the links with current load higher than the threshold from the logical topology when calculating the paths for newly entered demands. This approach decreases the congestion rate in the multi-commodity environment.

**Topology B.** we repeat the previous test on Topology B with more flow demands. Topology B has more nodes and links compared with Topology A. similarly, the characterized demands belong to the different QoS classes based on our application classification approach.

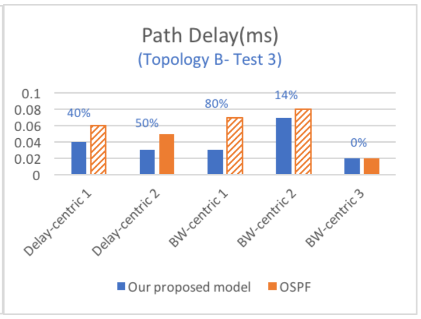
First, we execute the request with three demands and then we continue with four and five within the same network situation (*Test 1*, *Test 2*, and *Test 3*). The delay and packet loss of the calculated paths in each test are illustrated in Figure 5.10 and 5.11, respectively. Analysis of the diagrams shows that the same interpretation can be applied for this experiment too since the optimized paths in terms of the delay and loss rate are discovered by our model compared to the OSPF results, either in delay-centric demand or in BW-centric demand.



(a) Test1

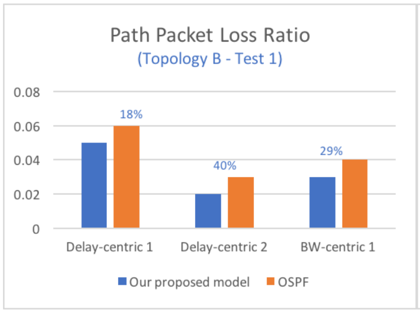


(b) Test2

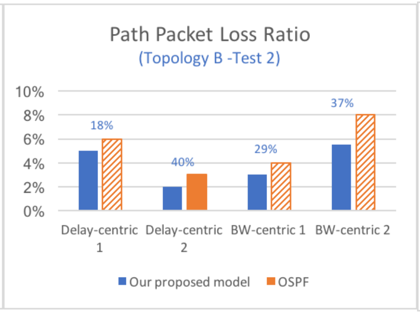


(c) Test3

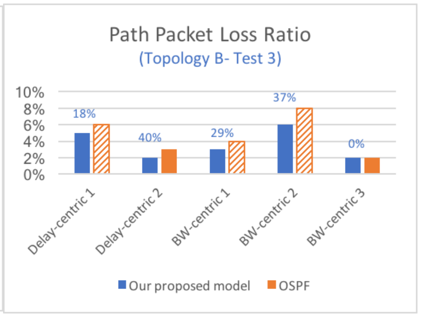
Figure 5.10 Path delay result - Topology B



(a) Test1



(b) Test2



(c) Test3

Figure 5.11 Packet Loss Ratio of the paths - Topology B

In terms of the link utilization rate, Figure 5.12 shows that the same scenario happens on Topology B ; the link utilization rate is normal in all the steps within our model and the network doesn't experience the link overload and congestion. In OSPF case, *TEST 2* and *TEST 3* have the links with more than 100% utilization rate, since there is no awareness of the available link bandwidth in OSPF.

Here, we express the situation of the network in *TEST 3*. In Table 5.4, the allocated paths calculated by OSPF for different demands are determined as well as the links with utilization rate 100% or more. The links with highest congestion rate are being used with four demands out of five. So, the QoS of all demands could be affected by the link failure and they may endure the delay and loss more than expected in the associated paths.

**Topology D.** The same scenario performed on Topology A and Topology B are redone on Topology D which have more nodes and links. So, we characterize more delay-centric and BW-centric demands in this case. All of them are entered in the network at the same time,

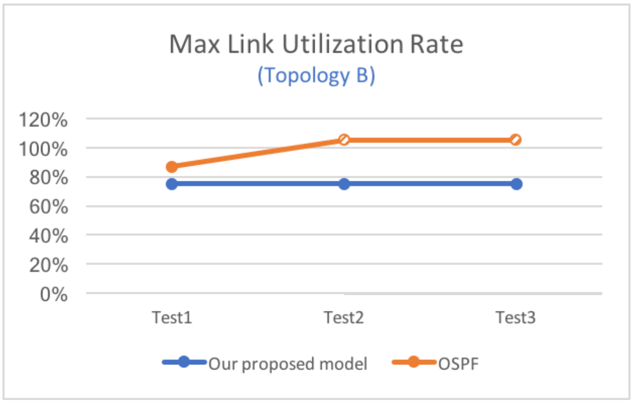


Figure 5.12 Maximum Link Utilization across the network in different tests - Topology B

Table 5.4 Paths and congested links in *Test 3* when OSPF routing model is used - Topology B

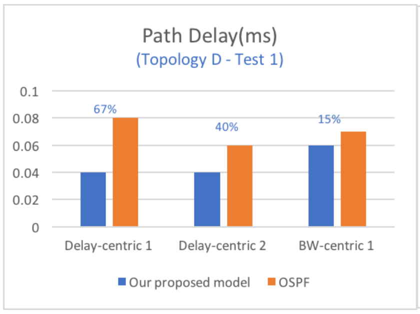
| | **Demand class** | **Source /Destination** | **Computed path (OSPF)** | **Congested link** |
|---|---|---|---|---|
| Test 3 | Delay-centric 1 | 4 →9 | (4 7)(7 8)(8 9) | (4 7)/(7 4) (7 5)(5 7) |
| | Delay-centric 2 | 3 →4 | (3 5)(5 7)(7 4) | |
| | BW-centric 1 | 6 →2 | (6 3)(3 2) | |
| | BW-centric 2 | 1 →9 | (1 4)(4 7)(7 8)(8 9) | |
| | BW-centric 3 | 7 →6 | (7 5)(5 6) | |

so the objective is to assign the paths between the source and destination for each of the demands.

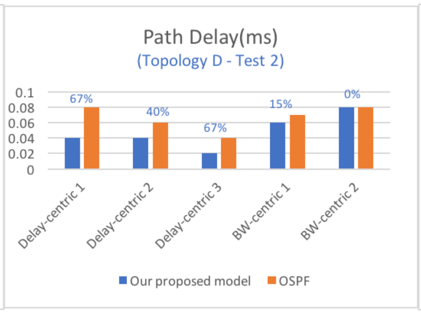
Figure 5.13a, 5.13b, and 5.13c display the delay associated for each demands in *Test 1*, *Test 2*, and *Test 3* scenarios, respectively. The columns in the charts declare that the delay associated with the delay-centric as well as the BW-centric demands are smaller in our model. According to Figure 5.14a, 5.14b, and 5.14c, the same statement can be declared for the packet loss of the paths. Similar to the conclusion made by previous experiments, we realize that our model finds the less-delay and less-loss paths among multiple choices for demands. Less delay is the major concern of the delay-centric demands, although the quick response time will increase the satisfaction of the other class of the application too.

Referring to Figure 5.15, the network experiences the link overload in *Test 3* when OSPF model is used for path determination and the link utilization rate exceeds the 100%. But, the maximum link utilization is kept around 75% in our model and the demands flow among the paths with high available bandwidth.

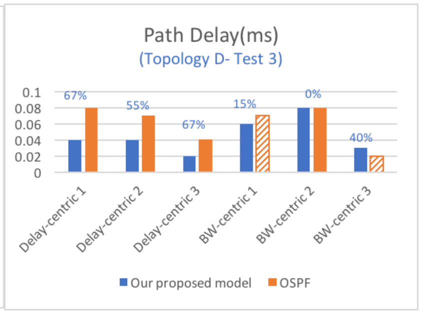
In Table 5.5, we detail the paths decided by OSPF model for the demands in *Test3*. The utilization of two links surpasses 100% and leads to having the congestion. Since two demands



(a) Test1

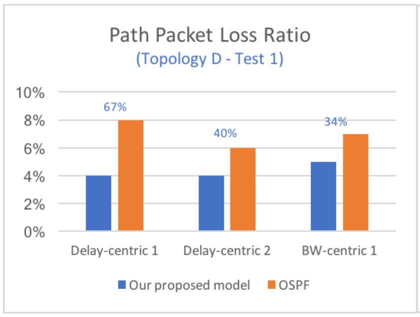


(b) Test2

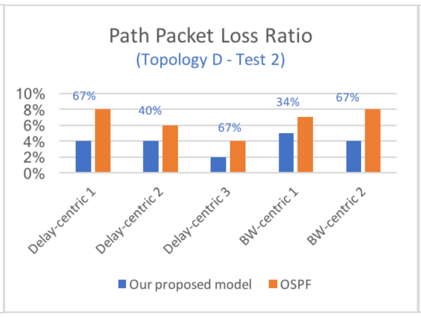


(c) Test3

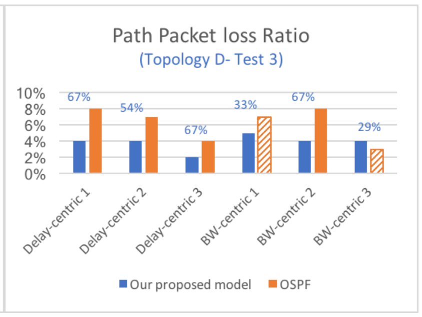
Figure 5.13 Path delay results - Topology D



(a) Test1



(b) Test2



(c) Test3

Figure 5.14 Packet Loss Ratio of the paths - Topology D

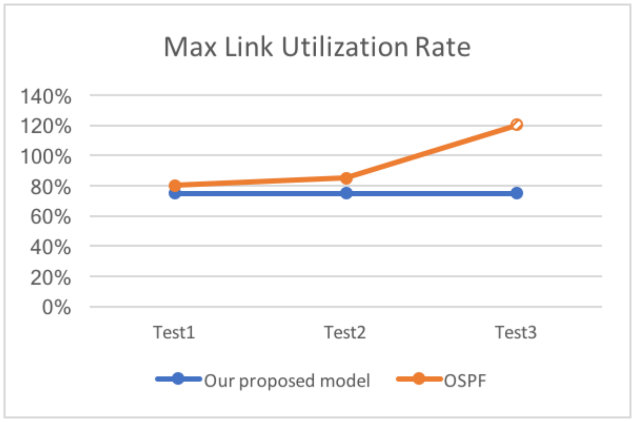


Figure 5.15 Maximum Link Utilization across the network in different tests - Topology D

*BW-centric 1* and *BW-centric 3* flow through these links, the delay and loss ratio could be increased and might exceed the acceptable level.

Table 5.5 Paths and congested links in *Test 3* when OSPF routing model is used - Topology D

| | **Demand class** | **Source /Destination** | **Computed path (OSPF)** | **Congested link** |
|---|---|---|---|---|
| Test 3 | Delay-centric 1 | 4 →12 | (4 5)(5 6)(6 9)(9 12) | (8 11)/(11 8) (11 14)(14 11) |
| | Delay-centric 2 | 7 →15 | (7 8)(8 9)(9 12)(12 15) | |
| | Delay-centric 2 | 2 →14 | (2 5)(5 6)(6 9)(9 11)(11 14) | |
| | BW-centric 1 | 10 →11 | (10 9)(9 6)(6 11) | |
| | BW-centric 2 | 1 →9 | (1 4)(4 5)(5 6)(6 9) | |
| | BW-centric 3 | 8 →14 | (8 11)(11 14) | |

## 5.3 Result analysis

Similar to the Internet, telecommunication, and cloud services, also IoT applications have to be evaluated in order to measure the quality provided to the end-users who use the application. The objective of the QoS model proposed by this thesis is to have control of the allocated resources and delivered services as expected by the IoT application from QoS point of view while maximizing the utilized resources in the network.

Our proposed model uses the characteristics of the SDN technology to be aware of the network elements status and the application QoS needs. SDN technology gives us these possibilities by southbound and northbound interfaces. The model employs the multi-path approach to find the optimal QoS-aware routing path for the applications which demand the data transfer service across the SDN network. The QoS policies are applied per individual application, although it can be applied per-customer through the modification application definition structure in SLA-database. In our model, different metrics are considered as the link cost metrics for the different application types: link delay and packet loss for the delay-sensitive applications, and bandwidth and loss rate for the bandwidth-sensitive applications, so the idea is to provide the least-cost resources using dynamic cost metrics and desired QoS levels per various applications.

In this chapter, we did multiple experiments in the single-commodity and multi-commodity environment. The scenario was to characterize different type demands ( delay-sensitive and BW-sensitive) with particular QoS needs in different network topology, and then measure the delay, loss rate, and throughput of network paths calculated by the proposed routing model and OSPF. Here, we summarize our analysis results from two perspectives: delay and throughput.

### Delay analysis

Minimizing the delay for packets traversing through the network is one of the main goals of network delivery. Most event-driven applications in IoT are mission-critical and delay intolerant, such as the emergency signals and safety-related applications in the Smart City. To be effective, the information should be transmitted in a limited time frame. Hence, it is important to verify the operability of applications in the service delivery subsystem such as core transport network in terms of the imposed delay.

Referring to the experimental results, we observed that our model finds the optimized routing paths in terms of the delay and packet loss for either delay-centric and bandwidth-centric applications. Seeing that our model uses the link QoS status in the path determination process, it analyzes available network paths and dynamically calculates delay and loss rate associated with the paths, and then it redirects the demand flow to the path with minimal delay and loss rate, respecting to the QoS constraints of the applications. Besides, the proposed framework monitors the current status of the network link and updates the topology databases. In case of the change in the path delay, the model calculates the alternative paths for the flows based on new network situation. Measuring the (near) real-time available link bandwidth and the utilization rate boosts the system performance and decreases the rate of the service level degradation since it avoids considering the high-load or congested links in the path determination process.

In the BW-centric demands, although the model looks for the links with the optimized combination of the high available bandwidth and lower packet loss rate, it still looks for the best-fit path with the acceptable level of delay and loss rate. The results achieved by the experiments affirm that our model provides the better performance paths compared with the OSPF in the BW-centric application scenario too.

In OSPF, the link cost metric is bandwidth, so the idea is to forward the demand through the links with the highest bandwidth. The high-bandwidth links could not be always considered as the less-delay links since link delay is affected by other facts such as queuing and congestion.

Therefore, we conclude that SDN-based architecture makes our model capable to be dynamically aware of the network status and SLA-related application QoS. Thus, we designed our model to directly apply them in the resource allocation process. Secondly, the criticality of the delay for the mission-critical application makes the model to seek for the delay-less and loss-less paths. The outcome is the least-cost (least-delay) and SLA-respected resource assignment for the applications.

#### **Throughput analysis**

Multimedia applications are going to be used widely in the Smart City, for instances, camera surveillance and traffic congestion management. Such applications have to exploit a considerable amount of data, which may be difficult to achieve especially in the dense urban environment and shared network capacity. These applications collect data either periodically or continuously from the IoT sensing devices. From the QoS point of view, this class of applications concerns about the bandwidth assigned by the system, and it may not be guaranteed in the best-effort Internet.

To assess the network throughput, we measure the maximum link utilization in the network after allocating the paths for the demands. When the link utilization reaches 100% and more, it signifies that congestion occurs in this link. The rate of the congestion is increased in the multi-commodity environment with more bandwidth-intensive demands[4]. The congestion may affect the performance of the demands passing through the congested links. The congestion costs a lot for the delay-sensitive applications and they might not fulfill their mission depending on to what extent they are impacted and delayed.

In our model, when multiple BW-intensive demands request for data transfer services, the multi-path approaches are applied if one path could not provide the requested bandwidth. The BW-intensive application is delay-tolerant compared with the delay-centric application. Since the model has currently available bandwidth of network links, it seeks to direct the flow toward the links with the higher available bandwidth which cost less. Secondly, we implemented the congestion prevention method in our model so that the link utilization rate is estimated at a given time and the links with higher link load are excluded from the path determination process. Besides, the model considers the limit for the link utilization rate to balance the load across the network links. So, the links with highest available bandwidth and less utilization rate are discovered by the model to prevent the congestion. Our model could apply different link utilization limit depending on the network status, demand arrival rate, or when it could not find the path fitted with the application QoS. The limit definition policy could be provided in the policy database in our framework.

As a conclusion, our model objective is not only to meet the application needs, it also maximizes the resource utilization and the network throughput. The centralized controller in the SDN-based network makes possible to control all aspects of the resource allocation process centrally and dynamically, instead of the complex and definitely static per-node configuration. It is worth noting that the link congestion could be happened because of the different issues in the network such as hardware and SW problem. In our thesis, when we point to the link congestion, we mean the congestion which happens because of the bad operation of the routing algorithm and link overload.

Differently, OSPF does not have access to the currently available bandwidth and the utiliza-

4. This situation could occur in the network when multiple high-resolution images or videos are transferred.

tion rate. It directs the demand flows toward the high-bandwidth links. OSPF could apply the multi-path approach but based on the knowledge of the maximum link capacity. The same paths could be assigned for a particular demand, independent of the current network status. Then it could cause the congestion and failure in the high-loaded links. Consequently, the demand performance flown through the failed links are impacted in terms of the delay and loss. If the impacted demand is the delay-sensitive, its effectiveness and usefulness might be significantly degraded. To avoid the congestion and have the load-balancing across the OSPF-based network, the QoS mechanism such as DiffServ, IntServ, queuing, and congestion control methods should be implemented in all the network elements. Implementation of the QoS mechanisms across the large network is the resource-intensive, time-intensive and complex task. It is error-prone because of the technical resource involvement. Moreover, the speed of convergence and the system adaptability to application needs and the network changes are low.

Concisely, our model takes advantages of the SDN controller to have the updated information regarding the network topology, network links, and SLA-relevant application QoS needs. Besides, the network policies regarding the event-based link reservations, link utilization rate, and load-balancing are provided for the routing calculator. Adaptive link cost metrics to the various application type are also applied in our model, aiming to minimize the cost as each type application desires. Compared to the best effort approach, the proposed model could guarantee the end-to-end application QoS across the core transport network. Thus, the centralized, status-aware, and SLA-aware resource allocation is performed both to satisfy the different types of demands and maximize the network throughput. This model is flexible and scalable since application SLA could be changed over time, the new policy may be applied, or application classification strategy could be changed on the business basis. The changes are deployed in the components of the framework, then they are applied in the routing calculation process as soon as they go and live in the system. This transformation is abstracted from the low-level configuration of the network elements, so this framework leads to optimizing the time and resources by skipping the complex vendor-dependent configuration tasks.

# CHAPTER 6 CONCLUSION

Our proposal aims to provide a customized and centralized QoS support service for the IoT application. The objective was to provide a QoS-aware resource allocation in the IoT system which enhances the evaluation metrics: network efficiency, user satisfaction, and application quality of service. Also, we intended to model a dynamic QoS-aware routing method within SDN-based core transport network to guarantee application QoS in terms of packet loss, delay, and requested bandwidth. Further, we evaluated the performance of the proposed QoS model.

In this section, we present a summary of our proposal and the main contributions of this work. Then, we discuss the limitations of the current work. Later on, some future works and possible extensions to enhance the proposed framework and model are described.

## 6.1 Summary of the work and contribution

The development and generation of the cheaper and smaller wireless devices such as RFID, sensors, and actuators have led to the formation of the IoT. The number of physical devices which are being connected to the Internet is growing at an increasing rate. A large number of devices enables a wide variety of services in many different application domains such as Smart City, Smart Transport, Smart Home, and Smart Health. Each application depending on their goals and criticality may expect various QoS requirements from IoT system in terms of data acquisition, transmission, and processing.

Despite QoS in traditional core networks which is more straightforward, QoS in IoT is very complex. Referring to the multi-layer IoT architecture (Figure 1.2), the quality parameters from users perspective would require the combination of all the schemes involved in the service delivery. Therefore, apart from time delay, packet loss rate, and capacity in the network layer, it also involves the quality of the service provided by sensing network. To ensure an acceptable level of QoS especially for safety-critical applications, there must be QoS approaches at every layer of the IoT architecture to measure the layer-relevant QoS parameters. Because of the data-centric and dynamic nature of the sensing network, new QoS factors such as data accuracy, coverage, data sampling rate become important for the IoT application. Besides, the growth of the innovative applications and their spontaneous deployment in the IoT system make the IoT SLA more dynamic and diverse. The IoT system still suffers from the lack of standardized mechanisms to represent the diverse application requirements. Therefore, the

Internet of Things (IoT) which is increasing on an unprecedented scale, faces challenges to properly support the applications demands on IoT infrastructure resources. So, the platform must be enabled to enforce the diverse and dynamism SLA. Apart from the flexibility and scalability of the IoT solutions, the simplicity of system control and management must be considered in the design of the solution so that IoT applications could be deployed easily and creatively.

In this thesis, we first described the main QoS parameters related to the IoT architectural layer. Application layer directly responds to the user requirements and it demands the desired data from the IoT devices. Network and communication layer made of core and access network provides the information transfer service. The Internet is going to be part of the core network to provide the globally connected things. The main QoS factors in the transport network are the delay, packet loss, jitter, and bandwidth. Sensing layer collects the desired data and the important QoS factors are data accuracy, sampling rate, coverage, delay, and lifetime. Further, we discussed that current QoS mechanisms in the communication network and Internet impose the limitations in terms of the diverse SLA support, scalability, and flexibility. Besides, we remarked that sensing network constraints make the QoS management very challenging due to the lack of end-to-end communication and unified interfaces between the application and devices.

Our investigation into the current IoT QoS approaches in the literature shows the lack of unified QoS support framework which could control the application QoS needs across different IoT subsystem including the data transfer and data acquisition services.

Middleware-based approaches have got the high attraction in the IoT system since the abstraction layer could hide the complexity and heterogeneity of the lower layer from the upper layer and increase the system flexibility and performance. There have been offered several middleware solutions for the IoT systems [120]. In this thesis, we proposed an SDN-based middleware to design the generic QoS support service for the IoT applications. The SDN technology is implemented in the core transport network/Internet and it provides the centralized layer to develop the support services. In the SDN-based network, applications can tell the controller how to program the network through the northbound interfaces, while the controller control and manage the network devices through the southbound interfaces (OpenFlow protocol). SDN principles not only abstract the complexity of the lower layers from the application layer, it makes the network programmable and fast-converged based on the business needs.

Although much prior research has exposed the potential benefits of applying SDN in computer networks to facilitate the network management, there have been a few studies about how to apply the SDN in the IoT system. As discussed earlier, most of the ideas are mostly preliminary proposal about the softwarization of the WSNs or with more focus on security and big data challenges of IoT. Here, we focused on the QoS management service.

We mapped the IoT architecture reference model onto the SDN architecture, so the IoT infrastructure is made of the sensing networks and transport network elements, the control layer/management layer is going to provide the middleware and support services for the IoT system, and finally, application layer includes the IoT application and services.

The contributions of this work are to propose a flexible and programmable software layer through integration core transport SDN into IoT architecture to provide customized and generic support services for the IoT applications. This framework overcomes the challenge of the dynamic and diverse definition of the SLA and the application QoS in the IoT. It is obtained through the programmable interfaces between the QoS management services and application layer provided by SDN technology.

The proposed scheme is made of several databases and functionalities implemented across different layers of SD-IoT architecture. The core functions are implemented in the SDN-based transport layer to provide the status-aware and QoS-aware application scheduling and routing mechanisms across the core transport network. Additionally, it provides unified interfaces towards the IoT-gateways to enforce dynamically the application QoS needs in the data acquisition process. IoT-gateway contains the resource optimization and task allocation mechanisms to manage the local sensing devices.

The framework provides the global knowledge about the transport network topology and the (near) real-time network status through developing the SDN application and OpenFlow interface characteristics. To enforce the application SLA in the QoS-aware resource allocation process, we assumed that all the quantitative QoS parameters and the accepted level are stored in the central database. This database is accessible by the SDN application through the standard API. This approach not only enables SLA enforcement in the resource management, it also resolves the dynamic definition of the SLA in the IoT. The IoT application could modify the QoS needs overtime in the database based on the business changes, so the QoS framework adopts them in the resources allocation to meet the new demands of the application. The path decision-making module provides the end-to-end QoS routing based on the current network status and the application preferences. Besides, the network policies are applied in the path determination process. This module includes the application classification principle which is based on the application sensitivity to the different QoS metrics. This principle is used for queuing mechanism in the network elements ports. We defined three classes of the application, each mapped into the different queues. However, the SDN characteristics enable the flexible and unrestricted number of the classes. The IoT-gateway continuously collects the sensing devices status and updates the global database in terms of the network lifetime. This information is used to assign QoS-ware sensing network and status-ware sensing device allocation for the IoT application.

Further, we proposed a model-based method to design status-aware and SLA-aware routing mechanism across the core transport SDN. This model uses the network topology and link status data collected by the SDN controller and the SLA-based application QoS constraints from the associated database to determine the best fit routing path for the data transfer. We mathematically formulated the QoS routing problem as Multi-Commodity Constraint Based Routing approach. The link cost metrics used in our model are packet loss, delay, and available link bandwidth. But different metrics are associated with the different application classes: delay and loss rate for the delay-centric, and bandwidth and loss rate for the BW-sensitive applications.

At last, we investigated the performance of the model in several network topology. The experimental results showed that our model provides the less-delay and less-loss rate paths for both delay-centric and BW-centric application, it performs link load-balancing and decreases the possibility of the congestion and service performance degradation. For the bandwidth-sensitive applications, it distributes the demand volumes across multiple paths to balance the network link utilization and to prevent the congestion. Multi-path and link load-balancing approach implemented in the model lead to enhancing the network throughput and increasing the system availability which especially is critical for the mission-critical application.

In sum, our proposal could be considered as a step into the definition of a centralized QoS architecture and adaptive SLA-aware resource allocation framework in the IoT system since the blocks, components, and their interactions have been described in detail. The unified SDN transport and IoT simplify the network management and configuration. It also enhances the definition of the customized and adaptive support services, not only in terms of QoS but in terms of security and storage.

## 6.2 Limitation of the work

The work presents several limitations that should be taken into consideration when using the proposal and when defining a future research path.

The proposed architecture relies on SDN technology implemented in the core transport network. So, the performance of the offered architecture depends on the accuracy of the gathered data from the IoT infrastructure (meaning transport network and sensing network elements).

The SDN controller could query the transport network element state either when receiving the service demand or periodically based on the time set. The collection of the information for the large network would be inefficient. The framework should determine the optimized time frame to gather the network status information, based on the demand characteristics and behaviors. It might make a trade-off between the system throughput and the cost of the status collection.

A centralized controller may be enough to manage all network elements of the small-scale network. Naturally, it represents a single point of failure and may have scaling limitations. Since OpenFlow allows the connection of multiple controllers to a network element, redundant controllers could be implemented in *Hot-standby* method to increase the network reliability and availability. So, the backup controller could take over in the case of a failure.

In IoT, large-scale sensing network and a large number of IoT applications produce a huge amount of data flows which must be transferred through the transport network. Generally, the large core transport networks often extend over multiple domains. Since SDN controller has a limited capacity in the request processing, independent SDN controller could be distributed across the large-scale network, each of them managing a network domain. Different design models have been proposed to distribute the network information among SDN controllers such as hierarchal and flat model [121]. The general idea is to have a logically centralized, but physically distributed control plane which still provides a simplified central view of the network from the application perspective. The East/Westbound interfaces are the communication channel for distributed SDN controllers. ONOS [122], HP VAN controller [123], and Onix [124] are examples of distributed controllers which are being widely used in the research and studies. [51]

## 6.3 Future work

Based on the proof of concept developed and the presented limitations, our study opens the door to several extensions.

First, we plan to study the QoS management in the distributed SDN control plane and multi-domain network, and accordingly the performance of the proposed QoS model in the different architectural model.

To increase the performance of the QoS support framework and minimize the system response time to the network topology changes or link failures, we plan to integrate *k shortest path routing algorithm* into our QoS model. For any particular demand, specifically delay-centric applications, more than one feasible solution is found, in case of the failure and SLA-violation by already-configured path, the alternative feasible solution is deployed in the network elements. Removing the calculation time of the new path could boost the satisfaction level of the delay-centric applications. For the best effort application, the current best efforts algorithms can be implemented to optimize the controller resources.

As our main future work, we think of extending the QoS support scheme by building an intelligent decision-making process on the use of the learning-based algorithms. The objective will be to make the SDN controller learn the network demand behavior on a real-time basis and provide the optimal behavior policy at any given time. The idea is to predict the volume demand at any segment of the network and apply the self-learned policy to determine the optimized routing path, aiming to minimize the cost of the paths for the different class of application and maximize the network throughput.

# BIBLIOGRAPHY

[1] ITU-T Y.2060, "Overview of the internet of things," International Telecommunication Union, Tech. Rep., 2012. [Online]. Available: https://www.itu.int/rec/T-REC-Y.2060-201206-I

[2] P. F. Ovidiu Vermesan, *Internet of Things – From Research and Innovation to Market Deployment*. River Publishers, 2014.

[3] "The internet of things: How the next evolution of the internet is changing everything," CISCO, Tech. Rep., April 2011. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[4] ONF, "Openflow switch specification (version 1.5.1)," Open Networking Foundation, Tech. Rep., 2015. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[5] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer *et al.*, "Internet of things strategic research roadmap," *Internet of Things-Global Technological and Societal Trends*, vol. 1, no. 2011, pp. 9–52, 2011.

[6] T. Kurakova, "Overview of the internet of things," *Proceedings of the Internet of things and its enablers (INTHITEN)*, pp. 82–94, 2013.

[7] ONF, "Software-defined networking: The new norm for networks," Open Networking Foundation, Tech. Rep., April 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[8] A. Haidine, S. El Hassani, A. Aqqal, and A. El Hannani, "The role of communication technologies in building future smart cities," in *Smart Cities Technologies*. InTech, 2016.

[9] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo, "Middleware to support sensor network applications," *IEEE network*, vol. 18, pp. 6–14, 2004.

[10] "Open networking foundation." [Online]. Available: https://www.opennetworking.org

[11] H. Ouchitachen, A. Hair, and N. Idrissi, "Optimal placement of sensors in mission-specific mobile sensor networks," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, no. 1, pp. 191–198, 2015.

[12] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *Communications (QBSC), 2014 27th Biennial Symposium on*. IEEE, 2014, pp. 71–75.

[13] Y. Jararweh, M. Al-Ayyoub, E. Benkhelifa, M. Vouk, A. Rindos *et al.*, "Sdiot: a software defined based internet of things framework," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, no. 4, pp. 453–461, 2015.

[14] [Online]. Available: http://forklog.net/

[15] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in software defined networks," in *Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on.* IEEE, 2014, pp. 1–8.

[16] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[17] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[18] Z. Shelby and C. Bormann, *6LoWPAN: The wireless embedded Internet.* John Wiley & Sons, 2011, vol. 43.

[19] "Official internet protocol standards." [Online]. Available: https://www.rfc-editor.org/standards

[20] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[21] M. Palkovic, P. Raghavan, M. Li, A. Dejonghe, L. Van der Perre, and F. Catthoor, "Future software-defined radio platforms and mapping flows," *IEEE Signal Processing Magazine*, vol. 27, no. 2, pp. 22–33, 2010.

[22] I. Ku, Y. Lu, M. Gerla, F. Ongaro, R. L. Gomes, and E. Cerqueira, "Towards software-defined vanet: Architecture and services," in *Ad Hoc Networking Workshop (MED-HOC-NET), 2014 13th Annual Mediterranean.* IEEE, 2014, pp. 103–110.

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

[24] R. Caceres and A. Friday, "Ubicomp systems at 20: Progress, opportunities, and challenges," *IEEE Pervasive Computing*, vol. 11, no. 1, pp. 14–21, 2012.

[25] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing.* ACM, 2012.

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

[27] ANSI/ASQC ISO A8402, "Quality management and quality assurance— vocabulary," American society for quality control, Tech. Rep., 1994. [Online]. Available: https://www.iso.org/standard/20115.html

[28] F. Xia, "Qos challenges and opportunities in wireless sensor/actuator networks," *Sensors*, vol. 8, no. 2, pp. 1099–1110, 2008.

[29] T. Forum, *SLA Management Handbook.* The Open Group, 2014, vol. 4.

[30] ——, "Performance reporting concepts and definitions," Tech. Rep., 2001.

[31] L. Li, S. Li, and S. Zhao, "Qos-aware scheduling of services-oriented internet of things," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1497–1505, 2014.

[32] M. Peuhkuri, "Ip quality of service," *Helsinki University of Technology, Laboratory of Telecommunications Technology*, pp. 2–0, 1999.

[33] R. Duan, X. Chen, and T. Xing, "A qos architecture for iot," in *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing.* IEEE, 2011, pp. 717–720.

[34] Cisco, "Enterprise qos solution reference network design guide," Tech. Rep., 2014.

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

[36] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," Tech. Rep., 1998.

[37] D. Chen and P. K. Varshney, "Qos support in wireless sensor networks: A survey." in *International conference on wireless networks*, vol. 233, 2004, pp. 1–7.

[38] Egham, "Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016," Gartner, Tech. Rep., 2017. [Online]. Available: https://www.gartner.com/newsroom/id/3598917f

[39] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[40] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[41] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[42] J. N. Al-Karaki and A. E. Kamal, “Routing techniques in wireless sensor networks: a survey,” *IEEE wireless communications*, vol. 11, no. 6, pp. 6–28, 2004.

[43] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, “Speed: A stateless protocol for real-time communication in sensor networks,” in *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on.* IEEE, 2003, pp. 46–55.

[44] K. Akkaya and M. Younis, “An energy-aware qos routing protocol for wireless sensor networks,” in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on.* IEEE, 2003, pp. 710–715.

[45] F. Bian, D. Kempe, and R. Govindan, “Utility based sensor selection,” in *Proceedings of the 5th international conference on Information processing in sensor networks.* ACM, 2006, pp. 11–18.

[46] G. Mainland, D. C. Parkes, and M. Welsh, “Decentralized, adaptive resource allocation for sensor networks,” in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2.* USENIX Association, 2005, pp. 315–328.

[47] Z. Ming and M. Yan, “A modeling and computational method for qos in iot,” in *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on.* IEEE, 2012, pp. 275–279.

[48] J. Jin, J. Gubbi, T. Luo, and M. Palaniswami, “Network architecture and qos issues in the internet of things for a smart city,” in *Communications and Information Technologies (ISCIT), 2012 International Symposium on.* IEEE, 2012, pp. 956–961.

[49] I. Awan, M. Younas, and W. Naveed, “Modelling qos in iot applications,” in *Network-Based Information Systems (NBiS), 2014 17th International Conference on.* IEEE, 2014, pp. 99–105.

[50] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A roadmap for traffic engineering in sdn-openflow networks,” *Computer Networks*, vol. 71, pp. 1–30, 2014.

[51] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[52] A. Mirchev, “Survey of concepts for qos improvements via sdn,” *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, vol. 33, p. 1, 2015.

[53] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y.-Q. Song, “Flowqos: Qos for the rest of us,” in *Proceedings of the third workshop on Hot topics in software defined networking.* ACM, 2014, pp. 207–208.

[54] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "Policycop: An autonomic qos policy enforcement framework for software defined networks," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For.* IEEE, 2013, pp. 1–7.

[55] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelém, "Control of multiple packet schedulers for improving qos on openflow/sdn networking," in *Software Defined Networks (EWSDN), 2013 Second European Workshop on.* IEEE, 2013, pp. 81–86.

[56] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks," in *Signal & Information processing association annual summit and conference (APSIPA ASC), 2012 Asia-Pacific.* IEEE, 2012, pp. 1–8.

[57] Y. Jinyao, Z. Hailong, S. Qianjun, L. Bo, and G. Xiao, "Hiqos: An sdn-based multipath qos solution," *China Communications*, vol. 12, no. 5, pp. 123–133, 2015.

[58] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74–98, 2014.

[59] "Opendaylight." [Online]. Available: https://www.opendaylight.org

[60] "Project floodlight." [Online]. Available: http://www.projectfloodlight.org/floodlight

[61] "Ryu sdn framework." [Online]. Available: https://osrg.github.io/ryu/

[62] P. Martinez-Julia and A. F. Skarmeta, "Empowering the internet of things with software defined networking," *FP7 European research project on the future Internet of Things*, 2014.

[63] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *Network Operations and Management Symposium (NOMS), 2014 IEEE.* IEEE, 2014, pp. 1–9.

[64] M. Mendonca, K. Obraczka, and T. Turletti, "The case for software-defined networking in heterogeneous networked environments," in *Proceedings of the 2012 ACM conference on CoNEXT student workshop.* ACM, 2012, pp. 59–60.

[65] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: Unbridling sdns," in *Software Defined Networking (EWSDN), 2012 European Workshop on.* IEEE, 2012, pp. 1–6.

[66] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "Openroads: Empowering research in mobile networks," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 125–126, 2010.

[67] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "Openradio: a programmable wireless dataplane," in *Proceedings of the first workshop on Hot topics in software defined networks.* ACM, 2012, pp. 109–114.

[68] A. Gudipati, D. Perry, L. E. Li, and S. Katti, “Softran: Software defined radio access network,” in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 25–30.

[69] F. Meneses, D. Corujo, C. Guimaraes, and R. L. Aguiar, “Multiple flow in extended sdn wireless mobility,” in *Software Defined Networks (EWSDN), 2015 Fourth European Workshop on*. IEEE, 2015, pp. 1–6.

[70] H. Yang and Y. Kim, “Sdn-based distributed mobility management,” in *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 2016, pp. 337–342.

[71] T. Miyazaki, S. Yamaguchi, K. Kobayashi, J. Kitamichi, S. Guo, T. Tsukahara, and T. Hayashi, “A software defined wireless sensor network,” in *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE, 2014, pp. 847–852.

[72] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, “Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks,” in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 513–521.

[73] ——, “Reprogramming wireless sensor networks by using sdn-wise: A hands-on demo,” in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*. IEEE, 2015, pp. 19–20.

[74] D. Zeng, P. Li, S. Guo, T. Miyazaki, J. Hu, and Y. Xiang, “Energy minimization in multi-task software-defined sensor networks,” *IEEE transactions on computers*, vol. 64, no. 11, pp. 3128–3139, 2015.

[75] D. Zeng, P. Li, S. Guo, and T. Miyazaki, “Minimum-energy reprogramming with guaranteed quality-of-sensing in software-defined sensor networks,” in *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 288–293.

[76] F. Olivier, G. Carlos, and N. Florent, “Sdn based architecture for clustered wsn,” in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2015 9th International Conference on*. IEEE, 2015, pp. 342–347.

[77] J. Liu, Y. Li, M. Chen, W. Dong, and D. Jin, “Software-defined internet of things for smart urban sensing,” *IEEE communications magazine*, vol. 53, no. 9, pp. 55–63, 2015.

[78] N. Bizanis and F. A. Kuipers, “Sdn and virtualization solutions for the internet of things: A survey,” *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[79] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann, “Ubiflow: Mobility management in urban-scale software defined iot,” in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 208–216.

[80] S. K. Tayyaba, M. A. Shah, O. A. Khan, and A. W. Ahmed, "Software defined network (sdn) based internet of things (iot): A road ahead," in *Proceedings of the International Conference on Future Networks and Distributed Systems.* ACM, 2017, p. 10.

[81] C. Gudipalley, C. Monden, J. Abbott, S. Amid, and R. Banke, "Service level agreement management," Feb. 21 2008, uS Patent App. 11/784,301.

[82] H.-J. Lee, M.-S. Kim, J. W. Hong, and G.-H. Lee, "Qos parameters to network performance metrics mapping for sla monitoring," *KNOM Review*, vol. 5, no. 2, pp. 42–53, 2002.

[83] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless microsensor network models," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 2, pp. 28–36, 2002.

[84] C. Xu, B. Chen, and H. Qian, "Quality of service guaranteed resource management dynamically in software defined network," *Journal of Communications*, vol. 10, no. 11, pp. 843–850, 2015.

[85] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *Proceedings of the 8th annual international conference on Mobile computing and networking.* ACM, 2002, pp. 148–159.

[86] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *ACM Sigmod record*, vol. 31, pp. 9–18, 2002.

[87] Y. Zhao, R. Govindan, and D. Estrin, "Residual energy scans for monitoring wireless sensor networks," 2002.

[88] O. Flauzac, C. Gonzalez, and F. Nolot, "Developing a distributed software defined networking testbed for iot," *Procedia Computer Science*, vol. 83, pp. 680–684, 2016.

[89] W. Li, D. Liu, B. Zhu, X. Wei, W. Xiao, and L. Yang, "Sdn control model for intelligent task execution in wireless sensor and actor networks," in *Vehicular Technology Conference (VTC Spring), 2016 IEEE 83rd.* IEEE, 2016, pp. 1–5.

[90] Z. Wen, X. Liu, Y. Xu, and J. Zou, "A restful framework for internet of things based on software defined network in modern manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 84, pp. 361–369, 2016.

[91] C. Bisdikian, L. M. Kaplan, M. B. Srivastava, D. J. Thornley, D. Verma, and R. I. Young, "Building principles for a quality of information specification for sensor information," in *Information Fusion, 2009. FUSION'09. 12th International Conference on.* IEEE, 2009, pp. 1370–1377.

[92] C. Bisdikian, L. M. Kaplan, and M. B. Srivastava, “On the quality and value of information in sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 4, p. 48, 2013.

[93] M. Mathew and N. Weng, “Quality of information and energy efficiency optimization for sensor networks via adaptive sensing and transmitting,” *IEEE Sensors Journal*, vol. 14, no. 2, pp. 341–348, 2014.

[94] V. Sachidananda, A. Khelil, and N. Suri, “Quality of information in wireless sensor networks: A survey,” *ICIQ (to appear)*, 2010.

[95] W.-Y. Huang, T.-Y. Chou, J.-W. Hu, and T.-L. Liu, “Automatically end to end topology discovery and flow viewer on sdn,” in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on.* IEEE, 2014, pp. 910–915.

[96] N. L. Van Adrichem, C. Doerr, and F. A. Kuipers, “Opennetmon: Network monitoring in openflow software-defined networks,” in *Network Operations and Management Symposium (NOMS).* IEEE, 2014, pp. 1–8.

[97] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, “Flowsense: Monitoring network utilization with zero measurement cost,” in *International Conference on Passive and Active Network Measurement.* Springer, 2013, pp. 31–41.

[98] “Openflow management and configuration protocol of-config1.1.1,” ONF(Open Networking Foundation, 3 2013.

[99] W. Wendong, Q. Qinglei, G. Xiangyang, H. Yannan, and Q. Xirong, “Autonomic qos management mechanism in software defined network,” *China Communications*, vol. 11, no. 7, pp. 13–23, 2014.

[100] J. Zhou, H. Jiang, J. Wu, L. Wu, C. Zhu, and W. Li, “Sdn-based application framework for wireless sensor and actor networks,” *IEEE Access*, vol. 4, pp. 1583–1594, 2016.

[101] S. Subbiah and V. Perumal, “Energy-aware network resource allocation in sdn,” in *Wireless Communications, Signal Processing and Networking (WiSPNET), International Conference on.* IEEE, 2016, pp. 2071–2075.

[102] Y. Chen, N. Nasser, T. E. Salti, and H. Zhang, “A multipath qos routing protocol in wireless sensor networks,” *International Journal of Sensor Networks*, vol. 7, no. 4, pp. 207–216, 2010.

[103] S. A. Chaudhry and J. Zhang, “Network-state-aware quality of service provisioning for the internet of things,” *Network*, vol. 7, no. 6, 2016.

[104] S. Song, J. Lee, K. Son, H. Jung, and J. Lee, “A congestion avoidance algorithm in sdn environment,” in *Information Networking (ICOIN), 2016 International Conference on.* IEEE, 2016, pp. 420–423.

[105] D. M. Micha Pióro, *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Morgan Kaufmann, 2004.

[106] “Feature scaling,” 1999. [Online]. Available: https://en.wikipedia.org/wiki/Feature_scaling

[107] P. Karkazis, P. Trakadas, H. C. Leligou, L. Sarakis, I. Papaefstathiou, and T. Zahariadis, “Evaluating routing metric composition approaches for qos differentiation in low power and lossy networks,” *Wireless networks*, vol. 19, no. 6, pp. 1269–1284, 2013.

[108] M. Dastbaz, C. Pattinson, and B. Akhgar, *Green information technology: A sustainable approach.* Morgan Kaufmann, 2015.

[109] D. Medhi and K. Ramasamy, *Network routing: algorithms, protocols, and architectures.* Morgan Kaufmann, 2017.

[110] B. Xiong, K. Yang, J. Zhao, W. Li, and K. Li, “Performance evaluation of openflow-based software-defined networks based on queueing model,” *Computer Networks*, vol. 102, pp. 172–185, 2016.

[111] R. Bellman, “On a routing problem,” *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[112] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[113] “Ampl, a mathematical programming language.” [Online]. Available: http://www.ampl.com

[114] R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of computer computations.* Springer, 1972, pp. 85–103.

[115] “User guide for minos 5.5: Fortran package for large-scale optimization.” [Online]. Available: https://web.stanford.edu/group/SOL/minos.htm

[116] “Ibm cplex optimizer.” [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer

[117] L. Tan, *Resource Allocation and Performance Optimization in Communication Networks and the Internet.* CRC press, 2017.

[118] S. Halabi, “Ospf design guide,” *Cisco Systems Network Supported Accounts*, 1996.

[119] CISCO, “How to configure ospf cost,” Tech. Rep., June 2009. [Online]. Available: https://supportforums.cisco.com

[120] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: a survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, 2016.

[121] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, "Distributed sdn controller system: A survey on design choice," *Computer Networks*, vol. 121, pp. 100–111, 2017.

[122] U. Krishnaswamy, P. Berde, J. Hart, M. Kobayashi, P. Radoslavov, T. Lindberg, R. Sverdlov, S. Zhang, W. Snow, and G. Parulkar, "Onos: An open source distributed sdn os," 2013.

[123] HP, "Hp sdn controller architecture," Tech. Rep., September 2013. [Online]. Available: http://h17007.www1.hpe.com/docs/networking/solutions/sdn/devcenter/06_-_HP_SDN_Controller_Architecture_TSG_v1_3013-10-01.pdf

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

[125] THE SCHOOL OF CISCO NETWORKING (SCN), "Protocol comparison ospf with eigrp, bgp and rip," Tech. Rep., 2013. [Online]. Available: https://premji-schoolofcisconetworking.blogspot.com/2013/06/protocol-comparison-ospf-with-eigrp-bgp.html

# APPENDIX A NETWORK TOPOLOGY

The particular network topologies used to evaluate the performance of the proposed QoS model in Chapter 5 are illustrated here:

Topology A:

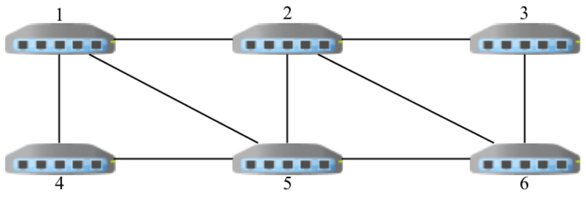


Figure A.1 Network topology-A

Table A.1 Topology-A link configuration

| **Link** | **Max BW(Mbps)** | **PacketLoss** | **Delay(ms)** | **Available BW(Mbps)** | **Link** | **Max BW(Mbps)** | **PacketLoss(%)** | **Delay(ms)** | **Available BW(Mbps)** |
|---|---|---|---|---|---|---|---|---|---|
| (1,2) | 300 | 1% | 0.01 | 300 | (2,6) | 400 | 2% | 0.01 | 200 |
| (1,4) | 400 | 1% | 0.02 | 400 | (3,6) | 600 | 2% | 0.05 | 400 |
| (1,5) | 200 | 2% | 0.01 | 200 | (4,5) | 400 | 1% | 0.01 | 300 |
| (2,3) | 600 | 2% | 0.02 | 300 | (5,6) | 300 | 1% | 0.01 | 300 |
| (2,5) | 600 | 2% | 0.05 | 200 | | | | | |

Topology B:

Table A.2 Topology-B link configuration

| **Link** | **Max BW(Mbps)** | **PacketLoss(%)** | **Delay(ms)** | **Available BW(Mbps)** | **Link** | **Max BW(Mbps)** | **PacketLoss(%)** | **Delay(ms)** | **Available BW(Mbps)** |
|---|---|---|---|---|---|---|---|---|---|
| (1,2) | 400 | 1 | 0.01 | 200 | (4,7) | 600 | 1 | 0.01 | 200 |
| (1,4) | 600 | 2 | 0.02 | 300 | (5,6) | 400 | 1 | 0.01 | 300 |
| (2,3) | 300 | 1 | 0.02 | 300 | (5,7) | 300 | 1 | 0.01 | 200 |
| (2,3) | 200 | 2 | 0.01 | 150 | (5,8) | 200 | 2 | 0.02 | 100 |
| (2,5) | 300 | 1 | 0.02 | 300 | (6,8) | 300 | 1 | 0.01 | 200 |
| (3,5) | 400 | 1 | 0.05 | 200 | (6,9) | 300 | 2 | 0.02 | 1500 |
| (3,6) | 600 | 2 | 0.03 | 400 | (7,8) | 400 | 2 | 0.03 | 400 |
| (4,5) | 200 | 2 | 0.01 | 100 | (8,9) | 600 | 2 | 0.02 | 400 |

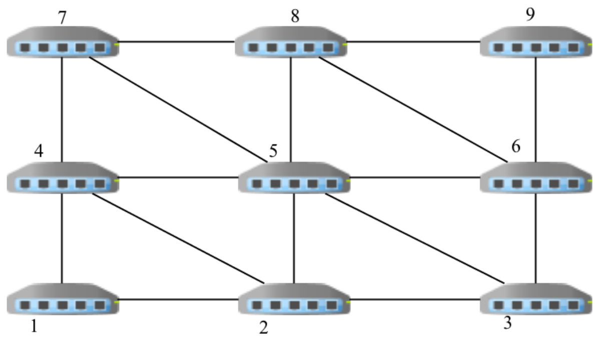


Figure A.2 Network topology-B

Topology C:

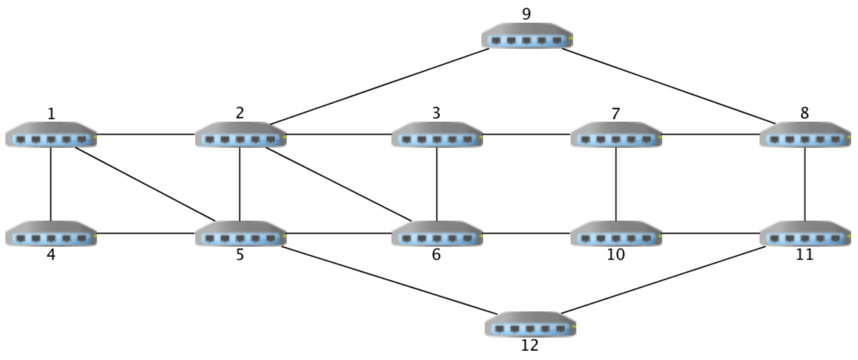


Figure A.3 Network topology-C

Table A.3 Topology-C link configuration

| **Link** | **Link BW (Mbps)** | **Packet loss(%)** | **Delay(ms)** | **Available BW(Mbps)** | **Link** | **Link BW(Mbps)** | **Packet loss(%)** | **Delay(ms)** | **Available BW(Mbps)** |
|---|---|---|---|---|---|---|---|---|---|
| (1 2) | 500 | 1 | 0.01 | 300 | (5 6) | 600 | 1.5 | 0.01 | 400 |
| (1 4) | 600 | 2 | 0.02 | 300 | (5 12) | 300 | 0 | 0.01 | 300 |
| (1 5) | 200 | 1.5 | 0.05 | 100 | (6 10) | 500 | 1.5 | 0.05 | 300 |
| (2 3) | 600 | 1.5 | 0.02 | 300 | (7 8) | 600 | 1.5 | 0.02 | 150 |
| (2 5) | 300 | 1.5 | 0.05 | 150 | (7 10) | 400 | 2 | 0.05 | 200 |
| (2 6) | 400 | 1.5 | 0.01 | 200 | (8 9) | 400 | 1.5 | 0.01 | 200 |
| (2 9) | 400 | 0 | 0.01 | 200 | (8 11) | 600 | 1.5 | 0.01 | 300 |
| (3 6) | 600 | 1.5 | 0.05 | 500 | (10 11) | 400 | 1.5 | 0.01 | 200 |
| (3 7) | 600 | 2 | 0.05 | 500 | (11 12) | 300 | 1 | 0.02 | 200 |
| (4 5) | 400 | 0 | 0.01 | 300 | | | | | |

Topology D:

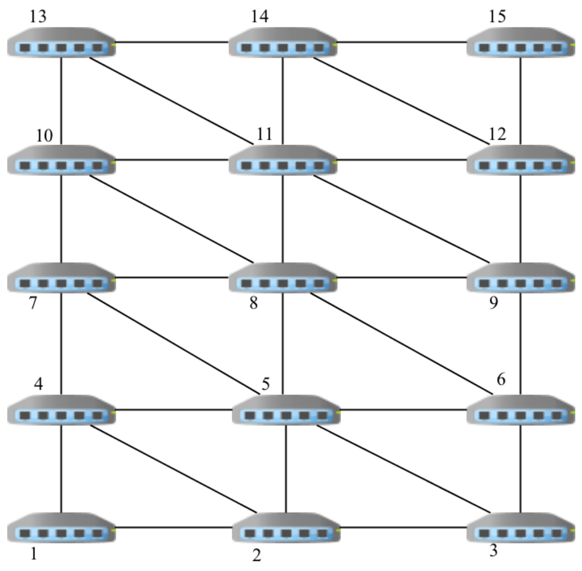


Figure A.4 Network topology-D

Table A.4 Topology-D link configuration

| **Link** | **Link BW (Mbps)** | **Packet loss(%)** | **Delay(ms)** | **Available BW(Mbps)** | **Link** | **Link BW(Mbps)** | **Packet loss(%)** | **Delay(ms)** | **Available BW(Mbps)** |
|---|---|---|---|---|---|---|---|---|---|
| (1 2) | 400 | 1 | 0.01 | 200 | (7 10) | 600 | 1 | 0.01 | 400 |
| (1 4) | 600 | 2 | 0.02 | 300 | (8 9) | 300 | 2 | 0.02 | 200 |
| (2 3) | 300 | 1 | 0.02 | 300 | (8 10) | 200 | 1 | 0.01 | 200 |
| (2 4) | 200 | 2 | 0.01 | 150 | (8 11) | 300 | 2 | 0.01 | 100 |
| (2 5) | 300 | 1 | 0.02 | 300 | (9 11) | 400 | 1 | 0.01 | 200 |
| (3 5) | 400 | 1 | 0.05 | 200 | (9 12) | 600 | 2 | 0.02 | 400 |
| (3 6) | 600 | 2 | 0.01 | 400 | (10 11) | 600 | 1 | 0.01 | 400 |
| (4 5) | 600 | 2 | 0.03 | 400 | (10 13) | 300 | 2 | 0.02 | 250 |
| (4 7) | 200 | 1 | 0.01 | 100 | (11 12) | 200 | 1 | 0.01 | 100 |
| (5 6) | 600 | 2 | 0.01 | 400 | (11 13) | 200 | 2 | 0.02 | 200 |
| (5 7) | 300 | 1 | 0.01 | 200 | (11 14) | 300 | 1 | 0.01 | 200 |
| ( 5 8) | 200 | 2 | 0.02 | 100 | (12 14) | 200 | 2 | 0.02 | 100 |
| (6 8) | 300 | 1 | 0.01 | 200 | (12 15) | 600 | 1 | 0.01 | 200 |
| (6 9) | 600 | 2 | 0.02 | 400 | (13 14) | 300 | 2 | 0.01 | 200 |
| (7 8) | 400 | 2 | 0.02 | 400 | (13 15) | 300 | 2 | 0.02 | 100 |

# APPENDIX B ROUTING PROTOCOL

In Table B.1, we demonstrate well-known routing protocols and their main characteristics. OSPF as a link-state routing protocol is used in the large network, but it consumes more computational resource. Its implementation also is complex task across the large network. [125]

Table B.1 Various routing protocols

| **Features** | **RIPv1** | **RIPv2** | **IGRP** | **OSPF** | **EIGRP** |
|---|---|---|---|---|---|
| Category | Distance vector | Distance vector | Distance vector | Link state | Hybrid |
| Metric | Hop | Hop | Composite (BW and delay) | BW | Composite (BW and delay) |
| Periodic advertisement (second) | 30s | 30s | 90s | N/A | 30s |
| Scalability – Size of network | Small | Small | Small | Large | Large |
| Resource usage | Low | Low | Low | High | Medium |
| Implementation and maintenance | Simple | Simple | Simple | Complex | Complex |