ADMISSION CONTROL IN A HETEROGENEOUS SOFTWARE-DEFINED NETWORK

Subaharan Kailayanathan
skn09011@student.mdh.se

Jimmy Norling
jng15001@student.mdh.se

Examiner: Moris Behnam
Mälardalen University, Västerås, Sweden

Supervisors: Mohammad Ashjaei
Mälardalen University, Västerås, Sweden

Svetlana Girs
Mälardalen University, Västerås, Sweden

February 21, 2019
Abstract

Software Defined Networking (SDN) provides centralized control by separating the control plane from the data plane on network devices. Subareas of networking such as Quality of Service (QoS) can greatly benefit from this separation as QoS policies can be provided globally for the network. One way of providing QoS is to reserve and monitor network resources to guarantee a specific data rate for a requested transmission end-to-end. The presented thesis looks into possible ways of controlling the wireless medium using SDN to provide QoS. A method for providing QoS in a multihop SDN network supporting wired and wireless communication was implemented. The method was evaluated using network performance metrics such as throughput and packet jitter. The results of the experiments showed that the implemented method could limit bandwidth utilization and prioritize bandwidth usage for higher priority nodes. The performance of the network was concluded to have severe issues with dropped packets and irregular packet jitter spikes.
## Contents

1 Introduction 1
   1.1 Motivation and problem formulation 1
   1.2 Overview 2

2 Background 3
   2.1 SDN basics 3
   2.2 OpenFlow 4
   2.3 SDN Controller 5
   2.4 Network Virtualization 5
   2.5 Quality of Service 6
      2.5.1 Queue management and scheduling 6
      2.5.2 Resource Reservation 7
   2.6 Quality of Service in Wireless Software Defined Networks 7
      2.6.1 Collision Avoidance in Wireless Networks 7
      2.6.2 EDCA & DEDCA 8

3 Related Work 9

4 Method 11

5 System Design 13
   5.1 Backoff timer 13
   5.2 Traffic Queuing in Access Point 13
   5.3 Final design 14

6 Implementation 16
   6.1 Admission Control Mechanism 16
   6.2 Zodiac FX Switch 17
   6.3 Zodiac WX Access Point 18
   6.4 Floodlight SDN Controller 19
   6.5 FlowVisor 20

7 Experiments 21
   7.1 Proposed cases for Proof of Concept 21
   7.2 Experimental Setup 21
      7.2.1 Heterogeneous SDN Network 22
      7.2.2 ACM for Heterogeneous SDN Network 23
   7.3 Measurements 23
   7.4 Parameters 24
   7.5 Limitations 24

8 Results 25
   8.1 Heterogeneous SDN Network - Measured Throughput 25
   8.2 Heterogeneous SDN Network - Jitter 26
   8.3 Admission Control for Heterogeneous SDN Network - Case 1 27
   8.4 Admission Control for Heterogeneous SDN Network - Case 2 28

9 Discussion 30

10 Conclusions 31

11 Future Work 32

References 35
List of Figures

1. A simple comparison between traditional and SDN switches .......................... 3
2. An overview of the SDN architecture. ......................................................... 4
3. List of header fields available for matching. .............................................. 4
4. List of possible counters. ............................................................................ 5
6. Flowchart of the ACM. ................................................................................. 15
7. Northbound Networks Zodiac FX switch .................................................. 17
8. Northbound Networks OpenFlow Compliant AP. ....................................... 18
9. A diagram of the Floodlight architecture. .................................................. 19
10. Topology 1 - Heterogeneous SDN Network. ............................................... 22
11. Topology 2 - Heterogeneous SDN Network with ACM. .............................. 23
12. Measured throughput for Heterogeneous SDN Network. ............................. 25
13. TCP Retransmission for measured throughput. ........................................... 25
14. Jitter for the SDN network. ....................................................................... 26
15. Throughput for ACM Case 1. ..................................................................... 27
16. TCP Retransmission for ACM Case 1. ....................................................... 28
17. Throughput for ACM Case 1. ..................................................................... 28
18. TCP Retransmission for ACM Case 1. ....................................................... 29

List of Tables

1. Case 1 – Low priority first ............................................................................. 21
2. Case 2 – High priority first ........................................................................... 21
1 Introduction

Computer networks today is an essential part of our modern life. The demand for network infrastructure that supports high communication speed, reliability and scalability increases as networking continues to interact with our daily lives [1]. With this increasing growth, it is apparent that the underlying systems that enable us to interconnect become more complex to manage.

In traditional computer networks, each network device has a control plane that is used for managing the control of data packets, most commonly in the form of routing processes. Through routing protocols, network devices can negotiate and exchange routing paths to build a routing and forwarding table on each device. These tables can be used by the data plane to decide on what action the network device will take on received packets by performing lookups on information stored in these tables. The combination of the control- and data plane on each device leads to a static and decentralized network architecture. Considering the static nature of such an architecture, the issues of maintaining a reasonable level of management and configuration is likely to be more difficult.

A new network paradigm, called Software Defined Networking (SDN), has surfaced in response to these issues. SDN aims to address the static nature of traditional networks with centralized control of the network through the separation of the control- and data planes. An example of a subarea of networking that can greatly benefit from this in many ways is Quality of Service (QoS). Traditional network architectures commonly use the IntServ or DiffServ models to treat certain traffic flows with higher priority. While the DiffServ model is strong when it comes to a simple configuration, there is an administrative burden as each and every device requires static DiffServ configurations to carry a flow with the specified QoS requirements. The IntServ model also has its drawbacks with scalability and complexity issues as network states have to be stored in each network device. Using the SDN paradigm, global visibility of the network can be maintained and network operators can manage and dynamically optimize network resources used throughout the network.

1.1 Motivation and problem formulation

When computer networks grow larger, network complexity and management will inevitably increase. To mitigate this problem, there is high value in studying well-established network concepts that are seen in current traditional networks and try to adapt or improve them using the SDN architecture. Concepts such as QoS is of high importance in networks where resources are limited and critical applications must function with high reliability. Utilizing QoS with SDN can provide a more reliable and scalable approach for larger, growing networks. Most implementations of SDN today exist in wired provider networks and data centers [2] where dealing with large amounts of data in an efficient way is important. However, studies on wireless networks are currently being explored to find good use cases of SDN. It is for example known that networks such as Wireless Sensor Networks (WSNs) that include embedded sensor nodes often lack the resources to support tasks that require high computational power for efficient communication. WSNs also often operate in star-like topologies with a centralized point for data collection. Networks that operate in this way could potentially leverage the SDN architecture to provide high control of the network while also being resource efficient. Using QoS to prioritize certain nodes in the network can be a way to optimize wireless communication. Other challenges include being able to have multiple logical networks coexist in the same physical network.

This thesis aims to contribute to the research of QoS in SDN networks by studying various approaches for QoS and implementing a system providing QoS. The objective is to provide QoS in the form of an Admission Control Mechanism (ACM) to manage and prioritize the usage of network resources by wireless nodes in a multihop heterogenous SDN network. Experiments will be made focused on packet jitter and throughput in the network. The results will be presented in the end of this paper in the pursuit of making future developments of similar implementations more stable and reliable. As such, the research questions are the following:
• How can the wireless medium be controlled to achieve QoS using SDN?
• How can admission control using SDN in a network slice be applied to wireless communication?
• How will the heterogeneous SDN network perform in a multihop network with respect to throughput and packet jitter?

1.2 Overview

The rest of the thesis is structured as follows. Section 2 describes relevant technical concepts needed to understand this paper. Section 3 will look into other papers related to this thesis. Section 4 will explain the research methodology used throughout the thesis. Section 5 begins with introducing relevant system approaches. Section 6 goes into the used hardware and software that will be used for experiments. Section 7 describes the proposed setup that is used in the experiments. Section 8 describes the results of the experiments. Section 9, 10, 11 discusses the results, conclusions and possible future work.
2 Background

This section of the report covers technical and theoretical aspects that are relevant to the thesis. These include SDN basics, OpenFlow, SDN Controller, Network Virtualization, Quality of Service and Wireless Communication mechanisms such as Collision Avoidance and Enhanced Distributed Channel Access. All topics described in this underlying section has an important role for understanding this thesis.

2.1 SDN basics

Software Defined Networking is an approach to computer networks that aims to centralize the control of the network by separating the control plane from the data plane on network devices and offloading it to dedicated controllers. This enables the switches to become forwarding devices that are controlled by a centralized SDN controller [3]. A comparison between traditional and SDN switches can be seen in Figure 1.

![Figure 1: A simple comparison between traditional and SDN switches](image)

With SDN making networks programmable, policies, algorithms, packet shaping and other functions can be defined as software modules through an Application Programming Interface (API). The SDN architecture generally has two interfaces referred to as the: South- and Northbound interfaces.

The Southbound interface, also known as the Southbound API, focuses on the communication between the SDN controller and forwarding devices. The Northbound interface, also known as Northbound API, is the application layer where all modules are defined. It works as a top layer of the SDN controller. A logical view of the SDN architecture can be visualized in Figure 2.
2.2 OpenFlow

OpenFlow [5] is a commonly used Southbound API and is based around the concept of programmable packet forwarding tables (flow tables) which are placed in an OpenFlow switch. A flow table is populated by flow entries which consist of header fields, actions and counters [6]. The header fields include a variety of fields such as ingress port, ethernet source/destination and more. Incoming packets are matched against the defined header field(s) and each flow entry can have an associated action and counter to it. See Figure 3 for a table of header fields available for matching. Actions are separated into required and optional actions. Required actions are associated with data plane forwarding to physical and virtual ports and must be implemented for an OpenFlow-compliant switch. Optional actions include using the traditional L2 pipeline for forwarding, flooding, queues and flow modification. Flow modification allows an administrator to change packet header fields such as VLAN, Ethernet destination addresses and more.

Counters keep track of statistics on a per table, flow, port and queue basis. Statistics can be polled and retrieved by a controller from a switch. Examples include flow duration, crc errors, drops, received and sent packets. See Figure 4 for the list of possible counters.

![Figure 2: An overview of the SDN architecture.](image-url)

![Figure 3: List of header fields available for matching.](image-url)

![Figure 4: List of possible counters.](image-url)
2.3 SDN Controller

The SDN Controller has a major role in a SDN network, the platform handles all network flows between the devices and the application modules through a Northbound API. As core elements of the network can be managed and configured by the SDN controller, tasks like monitoring and troubleshooting with a complete view of the network can be achieved using the controller. The purpose of an SDN controller is to achieve centralized control of the network. Based on the requirements of the network, the controller can push configurations to manage each device. For providing the functionality of the Southbound- and Northbound API a variety of SDN controllers exist. Examples of existing controller platforms used with OpenFlow include NOX/POX, OpenDaylight and Floodlight [8].

2.4 Network Virtualization

Network virtualization or network slicing, is a paradigm in which network operators can provide multiple logical networks and isolate network resources among them within the same physical network. As all networks support different services and require different requirements, a general “one-size-fits-all” network is sometimes lacking when more specific services and requirements are needed in the same physical network [9]. Network slicing attempts to break this static model by allowing multiple logical networks to co-exist in the same physical network. This form of network virtualization can be beneficial when merged with the SDN concept. As such, the concept has seen multiple use cases in different types of networks. In the more common production SDN network using OpenFlow, slicing has been used to allow production networks and experimental networks to co-exist [10]. Having multiple concurrent logical networks with the SDN architecture means that one controller can be used for each slice. Each controller can then provide a variety of functions for the network while maintaining isolation.
2.5 Quality of Service

The Internet Engineering Task Force (IETF) understands QoS as "a set of service requirements to be met by the network while transporting a flow" [11]. The service requirements usually pertain to a set of parameters that are meaningful when judging the quality of a network. Example of two common parameters

- Data transfer rate - the number of bits that pass a given point in a network in a given amount of time.
- Packet jitter - the variation in delay between packets

In computer networks running IP, using models such as differentiated services (DiffServ) and integrated services (IntServ) are common approaches to guarantee a level of QoS in traditional networks [12]. The DiffServ model relies on packet classification by marking packets as belonging to a certain traffic class. To maintain QoS throughout the network, per-hop behaviors (PHBs) are used to define how marked packets are handled as they traverse hops. Different PHBs can be used for different types of traffic such as best-effort and low-loss, low-latency traffic and more. Which PHB is used is determined by a 6-bit value called the differentiated service code point (DSCP) and can be found in the DS field of an IP header. It is important to note that other mechanisms for marking exist such as the Class of Service (CoS) field which is found in a 802.1q VLAN tag that can be used for marking at Layer 2 [13].

In the IntServ model, QoS is provided in the form of admission control based on requests called "Flow Specs" that routers receive. The Resource Reservation Protocol (RSVP) is the protocol used to handle Flow Specs and other control messages in the network. Flow Specs are divided into traffic specifications (TSpecs) and request specifications (RSpecs). TSpecs describes the traffic that a sender generates. A TSpec contains parameters such as the average bitrate and peak rate [14]. RSpecs specify what type of QoS, either “Guaranteed” or “Controlled Load” [15]. Guaranteed tries to provide QoS by strictly ensuring that traffic is shaped according to the parameters defined in a TSpec. Controlled Load tries to approximate end-to-end behavior of a network by best-effort in unloaded conditions. Its intended use is for applications that require better performance than a traditional best-effort service.

There are two major differences between how QoS is provided in networks using the IntServ model versus the DiffServ model. IntServ can be considered a fine-grained QoS system while DiffServ, a coarse-grained system. In DiffServ, QoS is provided by simply marking packets and employing traffic policies using PHBs for each hop, while in IntServ all routers along a traffic path must maintain the state of a requested traffic flow. Because of this difference, IntServ is not considered a scalable solution for larger networks because of the number of reservations that need to be maintained across hops in the network [12]. A disadvantage with DiffServ is that it is a burden to configure. Each and every router has to have static end-to-end DiffServ policies configured to maintain a QoS along a path.

Despite DiffServ and IntServ models being heavily used in traditional networks, packet classification is also used in SDN networks. As OpenFlow flow entries make use of matching logic on header fields of incoming packets, marking packets using DSCP or frames using CoS markings can be used to build a framework for developing QoS mechanisms using modules in the controller. Examples of where this has been utilized in OpenFlow SDN networks is for queue management using various packet scheduling disciplines and resource reservation [16]. Section 3 describes works related to this.

2.5.1 Queue management and scheduling

In OpenFlow 1.0 the enqueue action [7] can be used to forward an incoming packet on a port into a configured queue on a switch. In a standard implementation of an OpenFlow 1.0 network, queues have to be predefined in the switch. In a Linux-based system, traffic control through queues can be achieved using a variety of algorithms that control how packet scheduling is performed [17]. An
example of a simple traffic shaping algorithm is the Token Bucket Filter (TBF). In TBF a virtual bucket is created which is stocked with tokens at a specific rate. Tokens correspond to bytes. TBF is based on the expenditure of tokens. Each arriving token collects an incoming data packet from the data queue and expends it depending on the token rate and data rate. Its functionality can be summarized to three possible scenarios [18]:

- Data arriving at a rate equal to the rate of incoming tokens will have a matching token and passes the queue without delay.
- Data arriving at a rate smaller than the token rate will only expend a part of the available tokens in the bucket before sent to the queue. This causes tokens to accumulate until the bucket size. Unused tokens can be used to send data exceeding the standard token rate if data bursts occur.
- Data arriving at a rate bigger than the token rate will expend tokens until the bucket is empty. This causes throttling and may drop packets if the data rate is sustained for longer periods of time.

As TBF can be used to achieve rate-controlled transmissions for senders in the network it is useful for when senders need to maintain a set data rate during transmissions.

2.5.2 Resource Reservation

Mechanisms using resource reservation typically make use of classifiers in combination with rate-shaping modules implemented in the controller [16]. Classification refers to matching on L2-L4 header fields in incoming packets and assigning it to a flow [19]. Based on a network policy that takes the network state into account can then attach flows to the rate limiter and queue the flow to achieve resource reservation. Other ways of implementing resource reservation is through the use of network monitoring mechanisms. Southbound APIs such as OpenFlow is able to poll statistics from OpenFlow switches which can be used as a framework for implementing QoS control [16].

2.6 Quality of Service in Wireless Software Defined Networks

Wireless networks use a shared medium which is prone to interference in communication due to problems such as, e.g. hidden and exposed node problem [20] [21], which can cause unpredictable changes in link quality and topology. This is compounded by the fact that the wireless medium is affected by outside interference caused by effects such as scattering, diffraction and absorption which limits available bandwidth in wireless networks [22]. As the wireless medium only supports half duplex communication, providing QoS in wireless networks often focuses on manipulating parameters that relate to mechanisms such as the DCF process used for collision avoidance [23]. DiffServ-like additions to the 802.11 protocol like the Enhanced Distributed Channel Access (EDCA) is an example of a mechanism that can manipulate this process with Layer 2 markings for traffic classification.

2.6.1 Collision Avoidance in Wireless Networks

In the 802.11 protocol, when multiple stations contend for the shared medium, collisions may occur. To deal with this problem it uses the Distributed Coordination Function (DCF) to ensure whether a station can get access to the medium or not based on whether the medium is idle at a given point in time. In DCF, stations get access in 3 steps [24]:

1. Perform a Clear Channel Assessment (CCA) for a period of time before attempting transmission. CCA indicates if the medium is busy for the current frame. If the channel is clear, proceed to the second step.
2. Check the Network Allocation Vector (NAV). Frames sent in the wireless network contain a duration field that specifies for how long the medium will be busy. Listening hosts read the duration field and set it as the NAV value. The NAV value specifies the amount of time the medium will be busy for frames following the current frame. Most commonly control frames.
3. If the channel is still busy after CCA or NAV checking, an exponential binary backoff timer is started and the steps are repeated after the timer expires. Otherwise, the station can start transmitting.

The time a station waits is based on the sum of an Interframe Space (IFS) and the backoff time which is the Contention Window (CW). The IFS most commonly used in DCF is called the DIFS. DIFS is defined as $\text{SIFS} + (2 \times \text{Slot time})$. The slot time and interframe space times are static parameters which differ depending on which 802.11 protocol is used. In some select cases, stations only wait for a short IFS (SIFS) period to access the medium. This is the case for control frames such as Layer 2 acknowledgements, and the request to send (RTS) frame which is used as a reply to clear to send (CTS) frames.

The procedure stations use to select a backoff time is based on a random value between zero and the Contention Window Minimum value (CWmin) [25]. The CWmin is an adjustable window which will double if a collision is detected up to a maximum value of CWmax. A collision is assumed if an ACK frame is not received after transmission. Unsuccessful transmissions cause retransmissions. If the CW increments up to CWmax, subsequent retransmissions will use CWmax as its back off timer until it is acknowledged or if the maximum retransmission limit is reached. A successful transmission resets CWmin to its default value. The backoff timer will always decrement for each slot time that passes if the channel is sensed as idle. If the counter reaches 0, the station transmits.

2.6.2 EDCA & DEDCA

QoS enhancements to 802.11 have been made in later amendments to the protocol such as 802.11e. With 802.11e, a medium access control (MAC) function called Enhanced Distributed Channel Access (EDCA) was added. In EDCA prioritization of channel access for stations can be accomplished by using another IFS called Arbitration Inter-Frame Spacing (AIFS) [26].

AIFS can prioritize sending wireless nodes by shortening or prolonging the period a node has to wait before it is allowed to send a frame. A shorter AIFS period of a sending node gives its packets a higher probability of being transmitted with a shorter delay, which is essential for delay-sensitive applications. EDCA utilizes AIFS access categories (AC) which map onto class of service (CoS) priority values [27]. Frames can be tagged before transmission to set a priority level adapted for traffic such as Background, Best Effort, Video and Voice. Different ACs have different CW boundaries which is how prioritization can be achieved. A new mechanism called Dynamic Enhanced Distributed Channel Access (DEDCA) has implemented as a solution for dynamic management of CW boundaries for wireless terminals in an SDN architecture [16]. In EDCA, the probability of a station winning the channel is based on static backoff parameters such as CWmin which varies depending on which 802.11 protocol is used. DEDCA proposes a mathematical formula that is used for increasing or decreasing the CWmin of wireless terminals which directly affects the probability of winning the channel. In the paper outlining DEDCA, an SDN controller was used and experiments using the mechanism were performed using an emulator called Estinet. Estinet was used to allow flexible modification and control over the implementation of the IEEE 802.11 protocol.
3 Related Work

A number of previous studies, both in the domain of network virtualization and different types of QoS in SDN have been made. As the research questions for this thesis deal with admission control in heterogeneous networks, the focus was primarily put on reviewing works around these subjects. Resource management for admission control is a hard QoS method. Its function is to manage resources and check if current resources are sufficient for a requested network flow. If they are sufficient, the resources are reserved for that particular network flow, thus providing a guarantee of usable resources.

The paper by S. Aglianó et al. [28] implemented network resource management and admission control in a network using virtualization through FlowVisor with Floodlight as its controller. In the topology, the authors used a physical SDN switch with one receiver and two senders separated in two network slices. The receiver link was shared between both slices. The implementation used polling to monitor current bandwidth utilization over the network links and depending on three defined parameters (bandwidth limit, threshold and priority) it could admit or reject incoming packets to the network. Bandwidth limit was defined as a static limit which could not be exceeded. The threshold value was defined to admit high prioritized packets but drop low prioritized packets. High prioritized packet flows were accepted into the network even if the bandwidth utilization was greater than a set threshold value but smaller than the bandwidth limit. Low prioritized packet flows were accepted until the utilization was less than the threshold. The packets in the experiments used fields present in an IEEE 802.1Q frame to set a priority value, VLAN ID and requested bitrate. FlowVisor used VLAN ID to map incoming packets to a slice.

In [28], experiments were done with real hardware in a single-hop topology using hosts supporting Ethernet. The presented experiments made use of 802.1Q tagging which is a networking standard officially only supported in the Ethernet protocol. While Ethernet packets can be sent using a wireless host using raw sockets, a similar approach with 802.1Q tagging was not implemented. Realistically, sending Ethernet packets from a WLAN host would not be a practical solution as the Ethernet frame differs greatly compared to a 802.11 data frame. The objective in this thesis, is to achieve resource reservation by implementing an admission control mechanism in a multi-hop SDN network in a single slice. A heterogeneous architecture is considered with wired and wireless nodes.

The paper by F. Köhler [29] focused on comparing one-way delay experienced in virtualized SDN networks with traditional networks. The implementation used two Zodiac FX SDN switches and one traditional Layer 2 switch. Floodlight was used as the SDN controller. The experiments primarily focused on measuring one-way delay in a wired SDN multihop environment in various network slicing scenarios. Measurement of the delay was done in a single slice, two slices, shared slice and a traditional network. The setup used in this thesis uses the same SDN switch but replaces one switch with a Zodiac WX AP making the multihop network.

Instead of one-way delay, measurements focused on packet jitter end-to-end from a WLAN host to an Ethernet host. As experiments had already been done and evaluated in a multi-slice environment, for the sake of simplicity, the packet jitter measurement is performed in a single slice.

In the paper by Aleksey Amelyanovic et al. [30] a method is presented for providing QoS in an Software-Defined Wireless Network (SDWN). QoS is provided by using OpenFlow for classifying IP packets tagged with a DSCP priority combined with OpenFlow queueing. Eight DSCP priority values are used to divide traffic flows into different traffic categories. The eight categories were mapped to eight OpenFlow queues, the higher the DSCP priority value, the earlier the transfer. In the experiments, a simulation was made experimenting with the setup using OpenvSwitch in an application called Mininet. A multihop topology was used with three wired hosts (H11, H12 and H13) on one switch S3, and two Access Points (AP4 and AP5) connected to a switch S3. S2 and S3 were connected with a single link.
Three DSCP priority values were assigned to three OpenFlow queues in S2 which had two links connected to AP4 and AP5 with different maximum bandwidth settings. The link on AP4 used two queues with DSCP 46 and 32. The queue with DSCP 46 was assigned a max rate of 2mbps and DSCP 32 a max rate of 1mbps. The link on AP5 was assigned a max rate of 500kbps. To demonstrate the system, the wired hosts generated streams of traffic using the tool Iperf with the same three DSCP values as configured on the OpenFlow queues. The results showed that each stream of traffic was rate-limited according to the rate that was configured on each queue.

The mechanism for controlling bandwidth usage is similar to the one used in this thesis in that rate-limiting is used. Similarly, Iperf is also used for generating traffic from a sender to a receiver. The main difference is that queues are used instead of statically setting a bandwidth as a parameter with Iperf. However, there is no admission control implemented as the traffic is only rate-limited when it passes through the queues.
4 Method

The Systems Development methodology outlined by Nunamaker and Chen [31] was a suitable general research process for developing the system presented in this thesis. The aim was to develop a system with an admission control mechanism in an heterogeneous SDN network using network virtualization. It was selected as a general methodology because it offers a structured approach which aided in developing a system with the desired functionality. The research process is divided into five principal stages as shown in Figure 5.

The first stage is centered around the need for the research questions to be grounded in an appropriate conceptual framework. During this stage, understanding of the processes of the system itself and studying the relevant literature for new ideas is important [31]. To create an initial framework, a literature study was performed focusing on reviewing related works on QoS in heterogeneous SDN networks. This gave further insight into the current state of QoS and SDN which aided in the second and third stage of the research process. In these stages, a set of theorized approaches were prepared with the desired functionality and the research questions and possible limitations in mind. This further led to the last two stages which focused on building a prototype and experimenting, observing and evaluating the built system. The prototype stage allowed for tests which in turn could shine light on advantages and disadvantages with the various approaches resulting in a final design.
As the research questions specifically deal with evaluating methods for maintaining QoS for a heterogeneous SDN network, a quantitative approach was taken to validate the final design. This approach was chosen as QoS is often achieved by manipulating the scheduling of packets. The effect of this manipulation can easily be demonstrated through graphs, thereby showing a proof of concept.
5 System Design

To address the research questions in Section 1.1, several approaches to QoS in both wireless and wired SDN networks were studied. A final design for an admission control mechanism was developed. The method for arriving at the final design primarily focused on experimenting with the hardware used in this thesis. Studying the approaches and the limitations that come with the hardware was important in creating a functional system. The following section discusses the theorized approaches, the final design and its drawbacks.

5.1 Backoff timer

The first approach focused on research around providing QoS by modifying the backoff timer involved in the DCF of the 802.11 protocol using EDCA. Since ACs map onto CoS priority values in EDCA, the TCI field in a VLAN IEEE 802.1Q header that contains the 3-bit Priority Code Point (PCP) field can be used to put packets into different ACs through tagging. In theory, a similar approach as implemented in [12] could provide QoS using EDCA while also implementing admission control. However, EDCA is static in that the values that AIFS use for different ACs are not changeable in runtime without making major changes to the standard itself. There have been papers such as [24] that modified the default operation by adding an algorithm that changed priority values based on the number of clients connected during runtime. This implementation, however, was used in a traditional wireless network. At the time of writing, no implementations using hardware switches and APs, specifically focusing on backoff timer manipulation in an SDN architecture have been found.

Despite EDCA being a viable approach, it was ultimately scrapped. The initial idea required the backoff timer to be changeable during run time and integrated with the SDN architecture. If this approach were to be taken, what is considered “integration with SDN” also needs to be specified. Southbound APIs such as OpenFlow is supported in wireless networks, but can only be used to modify flow tables of forwarding devices such as APs or switches [5]. However, in an IEEE 802.11 based network, a standard implementation of OpenFlow cannot control the inner workings of the 802.11 protocol to manage the wireless terminals. Modifying the backoff timer during run time by dynamically changing CW boundaries using an SDN controller based on the network state is a more truthful approach to “integration with SDN” but also a highly complex problem.

While the DEDCA mechanism for providing dynamic change of the backoff timer is an interesting approach and did involve an SDN controller, experiments were performed using the emulator Estinet which allowed easy change over the parameters of the 802.11 protocol. As this thesis uses real hardware, modification of the 802.11 protocol cannot be done without a major revamp of the underlying code that supports it. Other limitations include the fact that there is lack of commercial hardware support for DEDCA which increases implementation complexity.

5.2 Traffic Queuing in Access Point

Using queues to provide QoS in an SDN network is a viable approach as it can be used to mimic PHB that DiffServ provides in an OpenFlow network. The initial thought behind using queues as an approach was to rate-limit incoming traffic using multiple queues with different bandwidth settings. Incoming packets would be marked with a DSCP priority value and be mapped to queues to achieve rate-limited traffic with varying data rates for sending hosts. As flow entries in the flow table are processed top-down with the highest priority first, the installed flows in the flow table would be ordered according to highest priority first. A higher priority flow would then always match before a lower priority flow, thus starting transmission earlier for a higher priority node.
However, from experimenting with queues using QoS methods available in Open vSwitch [32], it was made clear that creating multiple queues on the AP did not work as intended. Only one usable queue at a time was functioning. If queues were functioning correctly, a framework for ACM could be developed by polling the AP for OpenFlow statistics and admitting packets up to a certain limit and rejecting subsequent packet flows that do not fall within the specified limit. Ultimately, because of implementation complexity, queues and polling of statistics was dismissed.

5.3 Final design

After dismissing management of the backoff timer and queues as viable approaches due to high complexity and hardware limitations, an alternative approach was taken. When designing the alternative approach, the primary focus was put on providing a system that at the very least satisfied the research questions outlined in Section 1.1, on a basic level. To achieve this, the ACM was moved from the controller to the AP. The usage of SDN is limited to only handling the data plane for traffic using OpenFlow 1.0 with a network hypervisor in a single slice between the AP and switch in the network. The overall goal of the ACM was to provide two main functions; prioritize bandwidth usage for a requested flow from a node based on a priority value and guarantee that the usage of bandwidth is within the limits of a given request.

An algorithm was considered and drafted for scenarios where multiple nodes can request bandwidth from the AP. In Figure 6, a general process for this algorithm is shown. The starting point for a flow is when a node requests bandwidth with a priority. The request will be sent to the AP and there will be a control if the requested bandwidth is available on the AP or not. If the requested bandwidth is available, the ACM will reserve bandwidth for the given request and send an acknowledgement to the client to start a flow. Subsequent requests are continually tracked and checked against the bandwidth limit. This can be seen in Figure 6 as a “Yes” decision on the “Available bandwidth?” statement, which further locks the bandwidth and sends an OK message for transmission to the client.

However, if the requested bandwidth is denied by the first statement (lack of bandwidth), the second statement will be checked in order to continue the request process for the node. The ACM will check the remaining bandwidth that is left on the AP and the priority value of the requested transmission. In this algorithm, priority values have a major role in decision-making, a higher number means a higher priority. There are two options during this stage, based on the priority. These options can be visualized in Figure 6. If the requesting node has a higher priority a stop signal will be sent to a lower priority node. In case the requesting node has a lower priority and there is an ongoing transmission by a higher priority node, the remaining bandwidth will be sent to the requesting client by the AP.

If a stop signal is sent by the AP to a node, the ongoing transmission for that particular node will be stopped and bandwidth occupied by that transmission will be released to the ACM. This will give the node with higher priority, prioritized access to the bandwidth. When the node with higher priority gets access to the bandwidth, the remaining bandwidth of the AP will be accessible and usable by a node with lower priority. During the process when a stop signal is sent to a client, the ACM will be placed in a wait-state (overhead) and the bandwidth used will be released for higher prioritized nodes. These states can be seen in Figure 6.
Figure 6: Flowchart of the ACM.
6 Implementation

Based on the final design in Section 5.3, a collection of hardware and software was put together for an implementation to evaluate the system. How experiments are done with the implementation is explained in Section 7. The general hardware setup used for the implementation is the following:

- 1 Lenovo T420 Laptop
- 1 Cisco 2960 switch
- 1 Zodiac FX switch
- 1 Zodiac WX AP
- 3 Raspberry Pi Gen 1 boards with Edimax EW-7811UN Wireless Nano USB Adapter

6.1 Admission Control Mechanism

As dealing with socket programming in lower level languages such as C can be tedious, the chosen language for the implementation of the ACM uses Python 2.7 as it easy to write and has good support for modules. For handling the communication of control messages between the AP and senders, raw TCP sockets are used. The code for the implemented ACM is split into two parts; one part in the wireless sender nodes and the second part in the AP.

The wireless sender nodes make use of the “socket” module to establish a TCP connection towards the listening AP on a port. After a connection is established the node sends an initial bandwidth request and the set priority in the form of a string. If the initial bandwidth request does not exceed the set bandwidth limit with respect to the current bandwidth usage, the transmission is started immediately. Otherwise, two conditional statements are used to handle two types of control messages that is received from the AP; NEW_REQUEST and TERMINATE. The NEW_REQUEST message is used to handle the case when the AP detects that a requested data rate exceeds the bandwidth limit and a new data rate must be used. Depending on the case it will react to this message in two different ways. If the AP detects that a higher priority node already has an ongoing transmission, no transmission restarts are required and the lower priority requesting node uses the newly received data rate for the transmission. If the AP detects that a lower priority node already has an ongoing transmission during the sent request from a higher priority node, a TERMINATE message is sent to the lower priority node. The lower priority node with an ongoing transmission reacts to this message immediately and terminates the current transmission and restarts it with the new data rate. The higher priority node will always get the higher data rate.

Similarly on the code implemented on the AP, the socket module is used to handle TCP connections. In addition to this, the “thread” module is used to provide multithreading to handle multiple nodes at a given time. However, as this implementation is a proof of concept, code was implemented to support two nodes for simplicity.

To deal with transmissions, Iperf3 is used and is executed with the “subprocess” module which spawns the Iperf3 process on the sender node towards the receiver. Various parameters are also supplied to the receiver during this stage. Polling is used to continually check if the process is running and also checks for received messages from the AP at the same time. After transmission is completed or aborted a message is sent to the AP declaring that the sender does not use the medium anymore and the AP frees up the used bandwidth.
6.2 Zodiac FX Switch

There are a variety of OpenFlow-compliant switches on the market from manufacturers such as HP, Juniper Networks, Brocade and more. A general trend with commercial SDN switches is the fact that the pricing is quite high and there is no guarantee that the source code is fully changeable. Because of this the Zodiac FX SDN switch was chosen as it is fully open sourced, affordable and portable. The Zodiac FX switch is made by Northbound Networks [33] and is used for the wired network. The switch supports four ports. By default, three of the ports are for forwarding flows and the fourth is a controller port which connects to a SDN controller. The Zodiac FX switch runs a fully open source firmware and can be updated using the official firmware and modified if necessary. In the thesis, a standard firmware running version 0.84 is used with no modifications. The configuration of the switch can be done through a web-based GUI or through the terminal using the micro-USB interface. There are some limitations with the switch as OpenFlow queues are not supported due to lack of available memory [34]. See Figure 7 for an overview of the switch.

![Figure 7: Northbound Networks Zodiac FX switch [33].](image-url)
6.3 Zodiac WX Access Point

A Zodiac WX AP is used for the wireless part of the network topology. Zodiac WX is manufactured by Northbound Networks and powered by OpenWRT which is an operating system targeting embedded devices [35]. OpenWRT is an extensible and stable operating system supporting 3000+ standardized applications through its package manager which was one of the main deciding factors for deciding on Zodiac WX. Northbound Networks provides an OpenWRT firmware build that comes with preinstalled packages for OpenFlow support. Starting from firmware 1.20, OpenFlow support is provided by OpenvSwitch [36]. OpenvSwitch is an open source multilayer software switch supporting a variety of standards for security, QoS, automation and monitoring purposes in networks [37]. On the AP, two physical Gigabit Ethernet ports are available for use [38]. The configuration of the AP can be done with a CLI through SSH or GUI with LuCI [39].

Figure 8: Northbound Networks Zodiac WX AP [38].
6.4 Floodlight SDN Controller

Floodlight is an open source, Apache licensed Java-based SDN controller that supports a collection of applications that are built upon it. The applications are separated into two categories: Java modules that are compiled together with Floodlight and applications built over the Floodlight REST API.

Floodlight is able to perform various functionalities for the OpenFlow network. Besides acting as a controller in the OpenFlow network to handle the forwarding of packets, the application modules can be used to define how the switches will act. A switch can, for example, act as a learning switch, load balancer, hub or a firewall using the compiled Floodlight modules. See Figure 9 for a set of core applications that come with Floodlight.

Floodlight supports two types of OpenFlow entry insertions: reactive and proactive. With reactive entry insertion, incoming packets are initially sent from the switch to the controller for evaluation. After evaluation, it adds the appropriate flow entries to the switch. With proactive entry insertion, flow entries are inserted before packets arrive at the switch [40]. For the implementation, Floodlight 1.2 is used with reactive entry insertion and is provided by the “Learning Switch” module which mimics a L2 switch but uses OpenFlow for forwarding.

![Figure 9: A diagram of the Floodlight architecture.](image)
6.5 FlowVisor

To provide network virtualization in the OpenFlow network a semi-transparent proxy controller called FlowVisor is used. FlowVisor manages the network slices based on the concept of flowspaces which contains a subset of rules based on the available header fields for matching in OpenFlow 1.0. The flow entries in OpenFlow is used to identify and map network flows to a specific slice [16]. The slices are created through slice requests that are passed to FlowVisor. Each slice has its own view of the network topology and the network nodes that reside within it. In this implementation FlowVisor acts as a proxy together with Floodlight to provide network virtualization for the network in a single slice. As a single slice is used, all flow entries in the switch and AP is mapped to the slice.
7 Experiments

Four experiments are presented in this thesis, two focusing on measuring the overall throughput and packet jitter in the system and two experiments focusing on measurements using the ACM. For the specifics on the parameters and tools used to obtain the results for the experiments, see Section 7.3-7.5. This section will present the experiments and the topology for experiment.

7.1 Proposed cases for Proof of Concept

As mentioned in Section 5.3, The ACM is designed for general cases with scenarios where multiple nodes can simultaneously request bandwidth and the ACM should be able to handle the requests. Two cases were used as a simple proof of concept for the experiments and show how a typical exchange between the AP, senders and receiver looks like.

Case 1 - Low priority node is started first and high priority node after:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>Bandwidth limit set to support a data rate of 10 Mb/s</td>
</tr>
<tr>
<td>Low node</td>
<td>Sends request to the AP for a data rate of 6 Mb/s, priority 100</td>
</tr>
<tr>
<td>AP</td>
<td>AP checks if bandwidth is available, if available send OK to the requesting node</td>
</tr>
<tr>
<td>Low node</td>
<td>If OK message received from AP, start packet flow towards receiver with a data rate of 6 Mb/s</td>
</tr>
<tr>
<td>High node</td>
<td>Sends request to the AP for a data rate of 6 Mb/s, priority 200</td>
</tr>
<tr>
<td>AP</td>
<td>Low node is running and current node requests 6 Mb/s but has a higher priority, send message to low node to terminate packet flow and restart with a data rate of 4 Mb/s</td>
</tr>
<tr>
<td>Low node</td>
<td>Restart packet flow with a data rate of 4 Mb/s towards receiver</td>
</tr>
</tbody>
</table>

Table 1: Case 1 – Low priority first

Case 2 - High priority node is started first and low priority node after:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>Bandwidth limit set to support a data rate of 10 Mb/s</td>
</tr>
<tr>
<td>High node</td>
<td>Sends request to the AP for a data rate of 6 Mb/s, priority 200</td>
</tr>
<tr>
<td>AP</td>
<td>AP checks if bandwidth is available, if available send OK to the requesting node</td>
</tr>
<tr>
<td>High node</td>
<td>If OK message received from AP, start packet flow with a data rate of 6 Mb/s</td>
</tr>
<tr>
<td>Low node</td>
<td>Sends request to the AP for a data rate of 6 Mb/s, priority 100</td>
</tr>
<tr>
<td>AP</td>
<td>High node is running and current node requests 6 Mb/s but has lower priority, send message to low node to start with a data rate of 4 Mb/s</td>
</tr>
<tr>
<td>Low node</td>
<td>Start packet flow with a data rate of 4 Mb/s</td>
</tr>
</tbody>
</table>

Table 2: Case 2 – High priority first

7.2 Experimental Setup

There will be two main topologies in this thesis for the experiments. Topology 1 is used to test the achievable throughput and packet jitter in the network without the ACM. Topology 2 is used for testing the ACM, using both cases mentioned in Section 7.1.
7.2.1 Heterogeneous SDN Network

Topology 1, as shown in Figure 10, is a single sliced topology that has one Zodiac WX AP supporting wireless clients and one Zodiac FX switch for supporting wired clients. On the switch, Port 1 is connected to the AP, bridging the network. Port 2 is connected to the Receiver and Port 4 is connected to the L2 switch, which has a laptop with FlowVisor and Floodlight 1.2 running. On the AP side, the WAN port is used to connect the switch and the LAN port to connect the L2 switch. Client (Sender) was connected to the AP.

![Figure 10: Topology 1 - Heterogeneous SDN Network.](image-url)
7.2.2 ACM for Heterogeneous SDN Network

Topology 2, shown in Figure 11, is similar to Topology 1 except that two clients are connected to the AP. Also, the ACM is implemented on the AP to provide QoS for wireless clients with two different priority values.

![Topology 2 - Heterogeneous SDN Network with ACM.](image)

7.3 Measurements

The presented experiments in this thesis use Iperf3 which is a tool for performing network bandwidth measurements. Iperf3 was selected as a measurement tool as it is easy to use and is widely used for bandwidth testing. Iperf3 can test bandwidth capability using TCP, UDP or SCTP transport protocols. Iperf3 functions by starting the receiver in server mode listening on a port and starting a sender in client mode to initiate a connection to the port [41]. When a client connects, test parameters are exchanged between the nodes. The test data is sent from the client to the server and measures the upload speed of the client. The results are displayed line by line based on a specified interval which states how much data was transferred and the average bitrate for each interval. If Iperf3 is started in UDP mode, packet jitter can also be measured. To collect the data for plotting, the sender and receiver used bash I/O redirection to redirect stdout to a file [42].
In the experiments, TCP and UDP transport protocols were chosen to evaluate the system. TCP is used to measure the throughput and UDP to measure the packet jitter. While UDP can also be used to measure throughput, TCP was chosen as it can also simultaneously collect TCP retransmissions which is a good indicator for judging the stability of the system. TCP retransmissions occur when an ACK is not received for a sent packet within an estimated 2x round-trip-time (RTT) [43]. When an ACK is not received the wait time is doubled before attempting retransmission again and the process repeats. The wait time is referred to as the retransmission timeout (RTO). On systems using the Linux kernel, if an ACK is not received within the RTO for a packet, a “tcp_retry” counter is incremented. A maximum amount of 15 retries is allowed before the TCP session is teared down. According to [44] this is equal to 924.6 seconds if no ACKs are received for 16 consecutive retransmissions attempts.

As retransmissions are triggered by the sender, this data was collected on the sending node. Packet jitter was chosen as a metric as the variation in delay between two received packets from point to point is useful for evaluating network congestion and timing drift. In real-time communication, a high packet jitter can cause artifacts due to variations in latency which results in the arrival of packets out of order. Low jitter is especially important in IP telephony or video conferencing where connectionless protocols such as UDP is used.

In Section 8.3 an estimated time specifying the overhead for the node to restart the TCP flow was given. This estimated time is based on the sum of the execution time for the specific code block which deals with this mechanism and the time for a 3-way TCP handshake to be completed for the restarted TCP connection. Wireshark was used to collect these packets and the time was obtained as each captured packet is time-stamped in Wireshark as it is received [45]. For timing the execution, the time.clock Python method was used.

All experiments are reproducible if the parameters are the same. Results of performance metrics such as throughput and packet jitter may vary from experiment to experiment due to a multitude of uncontrolled variables such as the wireless medium and unknown instabilities in the hardware and software that is used. TCP retransmissions also has a negative impact on the performance. The core function of the implemented admission control and the pattern shown in the experiments are reproducible. Core functions in this case refers to being able to maintain a set data rate that respects a defined bandwidth limit for two sending wireless nodes. This includes mechanisms as described in Section 7.1 for Case 1 and Case 2.

Expected functionality of the ACM is consistent as long as the defined bandwidth limit is not exceedingly low or high. The bandwidth limit should be below the average measured throughput achieved in the experiments which evaluate the measured throughput of the system. This guarantees that the requested resources are always available as setting a too high or too low bandwidth limit that cannot be achieved defeats the purpose of the ACM.

7.4 Parameters

For Iperf3 parameters, a default TCP window size of 29Kbytes and a read/write buffer length of 512 bytes was passed to the receiver node. Rate-limiting was achieved by Iperf3’s implementation of the token bucket filter [46]. At the time of writing, a set buffer length is known to maintain the stability of the rate-limiting performed by Iperf3 [47] which is why it is used in the measurements.

7.5 Limitations

The presented measurements in this thesis did not control for interference. The measured achievable throughput can therefore vary depending on the environment. There was also no specialized tuning done to the TCP connections [48]. Changing the TCP window size could improve performance.
8 Results

8.1 Heterogeneous SDN Network - Measured Throughput

In the following experiment, Topology 1 is used to benchmark the throughput of the network from a sender to a receiver. A TCP transmission was started from wireless Client A to wired Receiver B in the network. The test ran for 600 seconds. As can be seen in Figure 12, several large dips and peaks in throughput is prominent across the whole time interval, near the 550 second mark, measured throughput almost reaches zero. There is also a high number of TCP retransmissions for every passed second throughout the interval as seen in Figure 13. As a TCP retransmission is synonymous with a packet that does not receive an ACK based on the time of a 2x RTT, packet drops contribute to a declining throughput. As the tcp_retry has a max limit of 15, tests that run for longer periods of time risk TCP connection teardown if retransmissions are not resolved. As the retransmission count is consistently above 0 for the entire measured interval, this is highly probable for measurements with a longer interval.

Figure 12: Measured throughput for Heterogeneous SDN Network.

Figure 13: TCP Retransmission for measured throughput.
8.2 Heterogeneous SDN Network - Jitter

For measuring jitter, Topology 1 was used and an UDP transmission was started from Client A to Receiver B and the test ran for 600 seconds. For the measurements a datagram count of 16 was sent for each second to provide a good sample for packet delay variation. The jitter is measured in ms for each second interval as seen in Figure 14. Jitter fluctuates quite heavily with large peaks. As an irregular jitter is correlated with high network congestion this is likely the cause for the fluctuations.

Figure 14: Jitter for the SDN network.
8.3 Admission Control for Heterogeneous SDN Network - Case 1

For ACM in Case 1, Topology 2 was used for the experiments. In Figure 15, The AP was set with a bandwidth limit to support a data rate of 10 Mb/s. Client 1 requests a transmission with a data rate of 6 Mb/s towards the receiver with a priority of 100. After approximately 2 minutes, Client 2 also sends a request for 6 Mb/s with a priority of 200 towards the receiver which exceeds the bandwidth limit of 10 Mb/s. Because Client 2 has a higher priority, the AP tells Client 1 to step down from 6 Mb/s to 4 Mb/s.

The overhead of the restart mechanism was determined to be approximately 0.164 seconds based on the sum of the execution time for the code block that deals with the mechanism and the time for a TCP 3-way handshake to be completed. The limit of 10 Mb/s is also shown to never be exceeded in the line that combines the throughput of Client 1 and 2. Similar to the results in Section 8.1, a large amount of TCP retransmissions occur as seen in Figure 16. Although it does not exceed a maximum count of 10 across the entire interval, the amount of data sent for each second is less than the measurements in Section 8.1 so this result is to be expected. The stability of the rate-limiting appears to be affected by the retransmissions as explained by the dips in throughput, although it does not exceed the set limit.

The restart of Client 1 can also be seen in the figure as the retransmission count temporarily goes down between the 200 and 300 second interval. Dips in throughput should be followed with higher retransmission counts as the loss in throughput is affected by packet loss. For the interval between 0-180 seconds before Client 2 is started this is the case. However when two clients are transmitting in the network simultaneously, heavy fluctuations in retransmissions occur. The exact cause for the fluctuations is hard to determine without further investigation.

![Figure 15: Throughput for ACM Case 1.](image-url)
8.4 Admission Control for Heterogeneous SDN Network - Case 2

For ACM in Case 2, Topology 2 was used. In Figure 17, the AP has a set bandwidth limit to support a data rate of 10 Mb/s. Client 1 requests a transmission with a data rate of 6 Mb/s and priority of 200. After approximately 2 minutes, Client 2 requests a transmission with a data rate of 6 Mb/s and priority of 100 which exceeds the bandwidth limit of 10 Mb/s. The ACM controls this in the AP and directly tells Client 2 to start the transmission at 4 Mb/s. The graph in Figure 17 show that no restart of the transmission is needed as in Case 1 and the combined line does not exceed the bandwidth limit. As previous measurements focusing on throughput, a large amount of TCP transmissions can be seen in Figure 18. Compared to in Case 1, both Client 1 and Client 2 is more unstable and suffers throughput losses that dip below 2 Mb/s. Retransmissions also seem to fluctuate more heavily in Case 2 because of this, as the retransmission count is rarely below zero for any second in the interval.

Figure 16: TCP Retransmission for ACM Case 1.

Figure 17: Throughput for ACM Case 2.
Figure 18: TCP Retransmissions for ACM Case 2.
9 Discussion

Based on the presented research and experiments in this thesis, conclusions can be made in relation to the research questions.

RQ 1: How can the wireless medium be controlled to achieve QoS using SDN for wireless communication?

For wireless communication in particular, QoS with SDN can be achieved with backoff timer manipulation using the EDCA mechanism. As was discussed in Section 5.1, DEDCA is a proposed mechanism for integrating EDCA with the SDN architecture and used simulations to evaluate its performance. However, integrating DEDCA with the used hardware in this thesis work poses some major challenges as changing parameters related to the 802.11 protocol in hardware is a complex task and available hardware support for EDCA is limited.

Instead of focusing on EDCA, the direction of the research was moved into looking at traffic shaping mechanisms. QoS using traffic shaping is distinctly different compared to EDCA. EDCA directly targets the mechanism for the probability of a wireless node accessing the medium while traffic shaping is used to control the volume of traffic being sent into a network. However, as admission control was also implemented in the network, there is a level of prioritization for how wireless nodes use the medium.

It is unfortunate that the queue mechanism that OpenFlow supports was not usable in this particular implementation due to implementation complexity. If it was usable, traffic shaping could be implemented on a per-hop basis instead of statically set in the client as was done using the Iperf3 application. Furthermore, if queues did function, scheduling disciplines outside TBF could have been explored to see if other scheduling algorithms can add new layers of prioritization. However, as an alternative approach was implemented and an algorithm was designed that is applicable to wireless nodes, the general problem was addressed.

RQ 2: How can admission control using SDN in a network slice be applied to wireless communication?

Although network virtualization is only used for one slice, past related works have evaluated SDN in multi-slice environments. Thus, a proof of concept for admission control with prioritization of wireless nodes was developed and tested in one slice. Implementing traffic shaping was a prerequisite for the ACM in order to guarantee a set data rate for transmission. Although the implemented ACM presented in this thesis evaluated the algorithm for two wireless nodes in the network as a proof of concept, the implementation can be generalized to support more nodes. Even if the control is achieved in the AP instead of the SDN controller, research and evaluation of QoS approaches for SDN with respect to the used hardware is valuable information for future work. SDN is also used in the sense that OpenFlow handles the forwarding of packets in the network.

RQ 3: How will the heterogeneous SDN network perform in a multihop network with respect to throughput and jitter?

All presented experiments that evaluate throughput in the system has severe problems with packet loss as evidenced by the amount of TCP retransmissions explained in detail in Section 8.1. The exact cause of this instability could not be determined and be resolved during this thesis work. We believe that the cause is related to configuration parameters of the wireless interfaces or some mismatch when interconnecting the AP and switch. In the experiments focusing on jitter in Section 8.2, the jitter is also seen to be highly irregular with large dips and rises for the measured 600 second interval which further confirms the instability of the system.
10 Conclusions

The objective of this thesis was to provide QoS in the form of an Admission Control Mechanism for wireless nodes in a heterogeneous multihop SDN network in a single slice. Various approaches were studied and evaluated against hardware limitations and viability which resulted in a final design and algorithm for admission control. The overall goal of the ACM was to provide two main functions; prioritize bandwidth usage for a requested flow from a wireless node based on a priority value and guarantee that the usage of bandwidth is within the limits of a given request.

An implementation based on the design using two sending wireless nodes and one wired receiving node was made to show a proof of concept. The implementation was evaluated with experiments. Experiments for the ACM showed that the AP could limit bandwidth utilization of the wireless channel and add a layer of prioritization for bandwidth requests from wireless nodes thus showing that the algorithm worked. Experiments on the performance of the network arrived at the conclusion that the implemented system using the Zodiac WX AP and Zodiac FX switch had severe issues with network stability and performance as evidenced by a high TCP retransmission count and instable throughput over the time interval for the experiments.

The presented research of QoS as it relates to the used hardware, software and controller is useful for readers of this thesis for future troubleshooting and implementation purposes. The main contribution of this thesis is the admission control algorithm and research into various approaches for providing QoS in SDN networks.
11 Future Work

Possible directions of research that future works can go into is to implement and make improvements to the algorithm by fully integrating and implementing it with an SDN controller for dynamic control. Hosts could start transmissions and abort them without relying on applications such as Iperf3 for rate-limiting as this can be done in the controller or through queues. If the particular hardware and software is used in future works, certain functionalities such as queues and the issue of packet loss can also be examined in more detail and be resolved.
References


