QoS for Real-time applications using OpenFlow in Wireless Network

By
Sayed Qaiser Ali Shah
2011-NUST-MS PhD-IT-24

Supervisor
Dr. Adeel Baig
NUST-SEECS

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(September 2014)

Approval

It is certified that the contents and form of the thesis entitled "**QoS for Real-time applications using OpenFlow in Wireless Network**" submitted by **Sayed Qaiser Ali Shah** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Adeel Baig**

Signature: _____
Date: _____

Committee Member 1: **Dr. Adnan Khalid Kiani**

Signature: _____
Date: _____

Committee Member 2: **Dr. Syed Ali Hassan**

Signature: _____
Date: _____

Committee Member 3: **Dr. Nadeem Ahmed**

Signature: _____
Date: _____

Abstract

Network Convergence is of vital importance due to limited network resources. On one hand it decreases network cost but on the other hand it reduces Quality of Service (QoS). Due to increase in real-time network applications, such as video conferencing and VOIP calls, there is a need to achieve fairness in user demand through QoS. This task is more challenging in an Infrastructure less network due to increased network losses. QoS is not implemented in Wireless Local Area Network (WLAN) using OpenFlow. We have proposed an algorithm to support QoS in WLAN using OpenFlow architecture for real-time traffic, whereby efficiently utilizing bandwidth and reducing flow starvation. We named this algorithm as Adaptive QoS algorithm. This algorithm allocates queues to flows using real time traffic analysis in OpenFlow architecture. A specific queue is allocated to a specific type of traffic and flows are dynamically switched based on increase in traffic demand if other queues are under use. We have implemented our work on a physical test bed to compare it with standard QoS mechanism. The results show bandwidth efficiency over standard QoS mechanism and decrease in Packet loss rate. Adaptive QoS algorithm has increased the overall bandwidth and decreased the flow starvation.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at National University of Sciences & Technology (NUST) School of Electrical Engineering & Computer Science (SEECS) or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Sayed Qaiser Ali Shah**

Signature: _____

Acknowledgment

Up and above everything all glory to **ALMIGHTY ALLAH**. The Beneficent, The most Merciful and Most Compassionate. It's a great blessing from Almighty Allah that gives me the health and strength to do this research work. It is also made possible due to prayers of Parents and their great support and help.

I would like to special thank the **Supervisor** Dr. Adeel Baig and Committee Members

Sayed Qaiser Ali Shah

Contents

List of Abbreviations

Abbreviations	Descriptions
QoS	Quality of Service
QoE	Quality of Experience
WLAN	Wireless Local Area Network
LAN	Local Area Network
IEEE	Institute of Electrical and Electronics Engineers
DCF	Distributed Co-ordinated Function
PCF	Point Co-ordinated Function
TGe	Task Group E
VOIP	Voice over IP
CSMA	Carrier Sense Multiple Access
HCF	Hybrid Coordination function
EDCA	Enhanced Distributed Channel Access
HCCA	HCF Controlled Channel Access
MAC	Medium Access Control
AIFS	Arbitration Inter-frame Spacing
SIFS	Short inter-frame space
TXOP	Transmission Opportunity
IETF	Internet Engineering Task Force
ForCES	Forwarding and Control Element Separation
IP	Internet Protocol
DDOS	Distributed Denial of Service
D-ITG	Distributed Internet Traffic Generator

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Organizations use Wireless Local Area Network (WLAN) to cater mobility in a local area network. WLANs have many problems like limited bandwidth, losses because of collision with other wireless signals, delay etc. Multi users in a network can run multi applications at the same time, which can lead to congestion and starvation of applications because of limited bandwidth. Hence to overcome these issues in wireless network we need some Quality of Service (QoS) mechanism. QoS as a general term refer to overall performance of network. QoS includes bandwidth, bandwidth, packet loss rate, jitter, and delay etc. QoS is also important in computer networks in terms of Quality of Experience (QoE), where users can experience better services. QoS in wireless network has always been a great challenge for researchers as it is difficult to achieve and also different users have different requirements. Lot of research is being carried out on QoS in wired network; However QoS is much more challenging in wireless network due to losses and limited bandwidth where applications can lead to starvation. Many algorithms have been proposed to cater QoS in wireless network. Use of Real-time applications has increased recently. So, while using Real-time Applications like Video Conferencing or VOIP calls, losses are not tolerable. Hence we need some QoS mechanism for better performance and to overcome the problems in networks, especially in wireless network for Real-Time applications. A Lot of work is being done on QoS where queues are created, but the existing QoS mechanisms cannot fulfill today's dynamic requirements. Current QoS mechanisms lack capabilities like adaptive behavior and dynamism. These mechanisms are either static or needs manual configuration. To overcome some of the basic problems in networks, software defined networks are introduced.

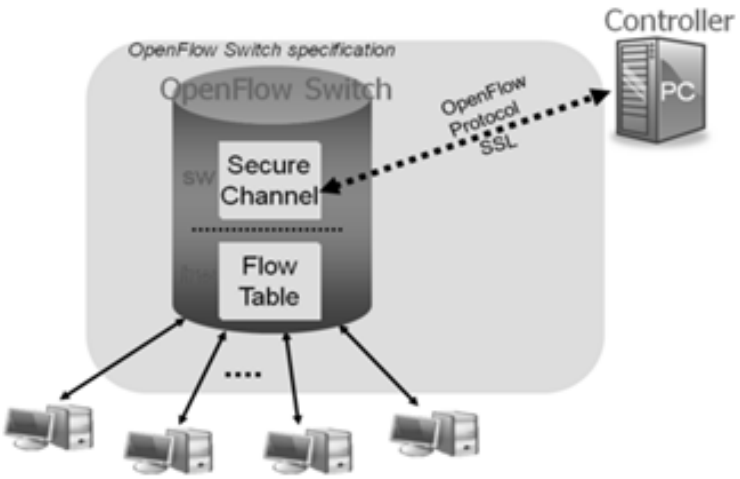

Figure 1.1: OpenFlow Architecture (1)

1.1 OpenFlow

Software Defined Networks use OpenFlow protocol for communication between switch and controller. OpenFlow is still an emerging technology. In a traditional network, switch works for both data and control planes. In case of OpenFlow, switch works only for data plane and control plane is implemented on a separate controller. The architecture is shown in the Fig. 1.1

The controller takes all the control decisions by defining a rule for each flow. The rule is than sent to the switch, which maintains flow tables to save flow entry for an individual flow when a packet arrives to the switch, it first checks flow tables. If a flow entry is present in a flow table the switch takes same action defined in the table for that specific flow, if the entry is not present, then the switch sends that packet to the controller to define a rule for that flow. We select OpenFlow because OpenFlow controller is open source and we can program it accordingly. OpenFlow is used in wired network for QoS. However it is not implemented in wireless network for QoS yet.

1.2 Problem Statement

Bandwidth Wastage and Starvation of Applications due to lack of Adaptive QoS mechanism for real-time applications in WLAN.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 discusses the state of the art related to the QoS, Wireless Network, OpenFlow, QoS in Wireless Network, QoS using OpenFlow, and reviews the relevant literature.

In Chapter 3, we discuss Experimental Setup and Implementation and then proposed methodology is presented.

In Chapter 4, the results are given along with detailed discussions.

In Chapter 5, the conclusion and future work is presented.

Chapter 2

LITERATURE REVIEW

In this chapter, literature and background, related to QoS and OpenFlow is presented.

2.1 QoS in legacy IEEE 802.11

IEEE 802.11 was first designed for best-effort burst traffic. The access function used was Distributed Coordination Function (DCF), which works in distributed manner and handles every wireless station independently. DCF uses contention manner channel access. Due to this contention access manner there were some issues like fairness, high collision, unpredictable jitter and delay etc. which leads to inefficient channel utilization. At that time researchers felt that delay sensitive services like Voice over IP (VOIP) or video flows should be forwarded with minimum delay so, for this purpose Point Coordination Function (PCF) was introduced, which is legacy IEEE 802.11 (4). Main idea was to schedule stations transmissions in round-robin manner by issuing polls for PCF enabled stations. This scheme introduced QoS to some extent but lacks flexibility and versatility. As PCF at that time was having low QoS management in IEEE 802.11 technology so was very limited in IEEE 802.11a/b/g commercial products. Till now IEEE 802.11 a/b/g standards of WLANs are considered only for best-effort traffic i.e. general data transmission. An effort was put by IEEE 802.11 Task Group-E (TGe) resulted in amendment of IEEE 802.11 standard and named it IEEE 802.11e (5).

2.2 QoS in IEEE 802.11e

Carrier Sense Multiple Access (CSMA) seemed efficient in IEEE 802.11 DCF for burst traffic but was not suitable for predictable traffic. However, PCF is contrary of DCF and it is not suitable for burst traffic but is suitable for predictable traffic. Another coordination function named Hybrid Coordination function (HCF) was introduced in IEEE 802.11 e which combines both DCF and PCF which makes transmission efficient in both burst and constant bit rate traffic. Enhanced Distributed Channel Access (EDCA) and HCF Controlled Channel Access (HCCA) was introduced by enhancing DCF and PCF MACs. Both EDCA and HCCA used HCF which exists in QoS enabled AP. EDCA defines access categories and prioritize different traffic streams using different channel access times and is called as Arbitration Inter-frame Spacing (AIFS). To achieve efficient channel utilization if collision appears than packets having high priority is forwarded to Physical layer. Other significant improvement was in case of transmission of frame bursts where frames are separated by Short inter-frame space (SIFS). The improvement was that each packet in a burst didnt require contention so, saved large amount of bandwidth hence improved channel utilization whereas IEEE 802.11 e defines Transmission Opportunity (TXOP). TXOP value defines time limit which transmits any number of packets in a burst that belongs to same access category and fits in that TXOP. TXOP values in EDCA are described through beacon frames. In case of HCCA HC calculates individual TXOP values for each polled packet from AP but making sure not to increase delay for other flows. The algorithm which performs this function is scheduler dependent. In EDCF IEEE 802.11 e researchers (6) show that using TXOP in contention-free burst improves system performance but in case of certain type of traffic delay increases. Simulation results in (7) show priority efficiency in EDCA but in case of high network load flows with low priority suffers starvation or in some cases complete flow blockage. In (8) researchers show that EDCA is not efficient in case of real time applications because EDCA cannot ensure QoS in high network load for high priority flows, which results in random delay because collision probability is high. In (9) researchers show voice capacity analysis using prioritizing channel access mechanisms in WLANs and they show that delays cannot be fully controller in EDCA.

2.3 QoS with HCCA

According to IEEE 802.11e both contention and contention-free access mechanisms in EDCA are combined, like in legacy IEEE802.11 contentions free

and contention periods were controlled by DCF and PCF. Similarly HCF combines HCCA and EDCA modes. Just like PCF super-frame in legacy 802.11, each super-frame in HCF starts with beacon frame and consists of alternation contention and contention-free periods. In contention-free period, access point polls stations for data as in PCF. QoS enabled stations receive CF poll frames having TXOP allocations. These frames indicate burst length of stations data to be sent to AP. Just after contention-free period, contention period starts and allow stations contending for the channel access. This period works on the basis of EDCA mechanism, handling virtual collisions and providing priorities for 8 different packet queues. To achieve desired QoS, access point needs scheduler which can handle constant and burst flows, transmission without collision, ensure fairness and can use spectrum efficiently. Very few schedulers can handle delay sensitive multimedia traffic. One of the efforts was J. Roys et al. proposal. The scheduler propped by J. Roys et al handles uplink scheduler for multimedia traffic flows, capable of achieving the QoS requirement (10). The results show improvement in bandwidth, delay and channel utilization. Later on researcher tried to achieve QoS for delay sensitive real-time traffic but here is no such scheduler or algorithm up to date which can achieve desired results for real-time applications in wireless network.

2.4 Software Defined Networks

Limitations in hardware components compel researchers to think about software based networks and for this purpose many proposals were presented. The most famous software defined networks are SoftRouter(11), Forwarding and Control Element Separation (ForCES) (12) and OpenFlow (1). The SoftRouter architecture (11) was proposed for network layer devices that allow dynamic binding among network elements running control elements, which are software based and data plane. Similarly ForCES (12) was project of Internet Engineering Task Force (IETF) and the main purpose of ForCES was to standardize communication between network elements and control elements. Unfortunately ForCES didnt become interest of vendor community worldwide. Based on same data and control plane separation mechanism and standardizing communication, OpenFlow (1) architecture was introduced. OpenFlow basically describes the way how software applications program flow tables of switches. OpenFlow touched hearts of researchers and became very hot area of research. Researchers in (13) documented differences between OpenFlow and ForCES. According to (13) both OpenFlow and ForCES standards decouple control plane from data plane. There is

one major difference between ForCES and OpenFlow. ForCES describes networking element and forwarding element and the way how they communicate? So using ForCES network architecture doesnt change. While using OpenFlow network architecture modifies in a sense that data plane element becomes simple device that forwards packets according to rules defined by control element. Another difference between ForCES and OpenFlow is that ForCES allows decentralize approach for controlling whereas OpenFlow uses centralize approach for control plane.

2.4.1 OpenFlow Specification

OpenFlow specification describes open standard which allows software application to program flow table of the switch. OpenFlow architecture has three main components i.e. an OpenFlow switch, Controller and secure channel as described in fig. 1.1. Switches maintain flow tables to forward packets. A flow table has flow entries. Each flow entry contains match fields, instructions and counters. When packet arrives the switch it is compared with match field. Certain action will be performed on packet if match is found. Counters keep packets statistics. Before sending packet to controller packet will be encapsulated. An OpenFlow controller is software program which manipulates flow table in a switch with the help of OpenFlow protocol. The interface through which controller and switch communicates is called secure channel. There are different versions of OpenFlow protocol. The very first released version of OpenFlow was 0.2.0. Right after the first version, 0.8.0 and 0.8.1 versions were introduced followed by version 0.8.2. Version 0.8.2 includes Echo Request and Echo Reply messages. Additional statistic information, IP netmasks and many other updates were included in version0.8.9. Than OpenFlow version 0.9 was introduced and finally the most widely deployed version 1.0.0 was released in 2009. Currently used versions are v1.0.0 (14), v1.1.0 (15), v1.2 (16) and v1.3.0 (17). OpenFlow version 1.0.0 flow table entry contains the following match fields given in Tab. 2.1 OpenFlow version 1.1.0 has some more fields in flow table other than the version 1.0.0 given in Table. 2.2 and also unlike version 1.0.0, version 1.1.0 supports multiple flow tables instead of just single flow table. The major improvements in OpenFlow version 1.2 were IPv6 addressing and IPv6 source address and destination address matching. Another major improvement is connectivity of single switch to multiple controllers which helps controllers to communicate and can also perform hand overs. In case of failure fast recovery will be ensured because of multiple controllers and can also perform load balancing.

The final version of OpenFlow specification i.e. v1.3 includes control packet rate, adding cookies to packets. Table. 2.3 shows changes between

Table 2.1: OpenFlow v1.0.0 match fields

Ingress Port
Ethernet Src
Ethernet Dst
Ethernet type
VLAN id
Vlan priority CoS
IP src
IP dst
IP Proto
IP ToS bits
TCP/UDP src port
TCP/UDP dst port

versions 1.0.0, 1.1.0, 1.2 and 1.3.0 presented in (17). In order to manipulate flow table we need to run software applications on controller. So, we need network operating system. It works as intermediary layer between user application and OpenFlow switch. Some of the examples of Network Operating Systems are NOX (18) (C++ based), POX (19) (Python based), Beacon (20) and Maestro (21) (both Java based). Recently BigSwitch released open Java based Floodlight (22) controller. Another researcher Foster et al (23) proposed network programming language named Frenetic to simplify applications development. Similarly Trema (24) was proposed by NEC to develop applications using C and Ruby and finally DreamersLab introduced Node.flow (25), which is used to build JavaScript controller using Node.js (26).

After the advent of OpenFlow researcher worked a lot in the areas of Optical Networks and some of the contribution in case of Optical Networks is shown in (27) (28).

Similarly in Network security to detect Distributed Denial of Service (DDOS) attack detection OpenFlow is also used and is shown in (29). In (29) DDOS attack is detected by Traffic analysis and centralized control, where statistic information of flow tables are used to classify normal or ma-

Table 2.2: OpenFlow v1.1.0 match fields

Ingress Port
Metadata
Ether Src
Ether Dst
Ether type
VLAN id
VLAN priority
MPLS label
MPLS EXP traffic class
IPv4 src
IPv4 dst
IPv4 proto/ ARP opcode
IPv4 ToS bits
TCP/UDP/SCTP src port. ICMP Type
TCP/UDP/SCTP dst port. ICMP Type

licious traffic. Similarly another research paper (30) in Network security is published. In (30) source address validation problem is discussed and the problem is solved by calculating flow path and traffic analysis using OpenFlow controller.

To handle user demands and dynamic requirement work on QoS in wired network using OpenFlow is also done. In (31) researchers have designed an architecture that can detect malicious packets that are not allowed to use QoS. Whereas QoS packets will be allowed to use fastest path in a network but in case of congestion a problem arises and all the flows from switch are deleted by in the solution which increases overhead. Researchers in (32) have proposed OpenFlow architecture for multiple services load balancing. In (32) researchers used multiple OpenFlow controllers to button the load of multiple services and hence services loads are divided among controllers.

Some of the contribution is done in almost every area of computer networks including QoS in wired network. As compared to wired network there

Table 2.3: Comparison of OpenFlow Versions

Specification	**1.0.0**	**1.1.0**	**1.2**	**1.3.0**
Widely deployed	Yes	No	No	No
Flow table	Single	Multiple	Multiple	Multiple
MPLS matching	No	Yes	Yes	Yes, bit added at bottom of stack
Group table	No	Yes	Yes	Yes, more flexible
IPv6	No	No	No	Yes, new field added
Simultaneous communication with multiple controllers	No	No	Yes	Yes, auxiliary connections established

is lot of problems in wireless networks like limited bandwidth, wastage of bandwidth, high delay, large number of losses etc. So, there is need to introduce some QoS mechanisms in Wireless network to minimize bandwidth wastage and starvation of applications due to lack of Adaptive QoS mechanism for real-time applications.

Chapter 3

IMPLEMENTATION

This chapter explains tools used, experimental setup and practical implementation of our research on physical testbed in detail.

3.1 Tools to be used

To implement our research we worked on a physical Test bed because of OpenFlow based wireless scenarios limitations in simulators and emulators. For our test bed we used TP Link WR1043ND V1.8 Switch (36) and installed OpenWRT firmware(37) on it which is described as Linux distribution for embedded systems. We then installed OpenFlow package on the switch to make it OpenFlow enabled. The version of OpenFlow used is OpenFlow 1.0(1). We used POX platform (19) for controller and the version is POX carp.

3.2 Experimental Setup Architecture

After Test bed configuration we implement our work in WLAN IEEE 802.11. In our implementation we attached switch with controller via wired medium and attached Laptops to switch via wireless medium. Using Distributed Internet Traffic Generator (D-ITG)(39) we generated both real-time (VOIP and Video) and best-effort traffic in the scenario given in Fig. 3.1

VOIP and Video traffic requirements are given in Table. 3.1. We use G.711 codec for VOIP and H.264 for Video traffic. Bandwidth requirement for Video Application is given in Table. 3.2

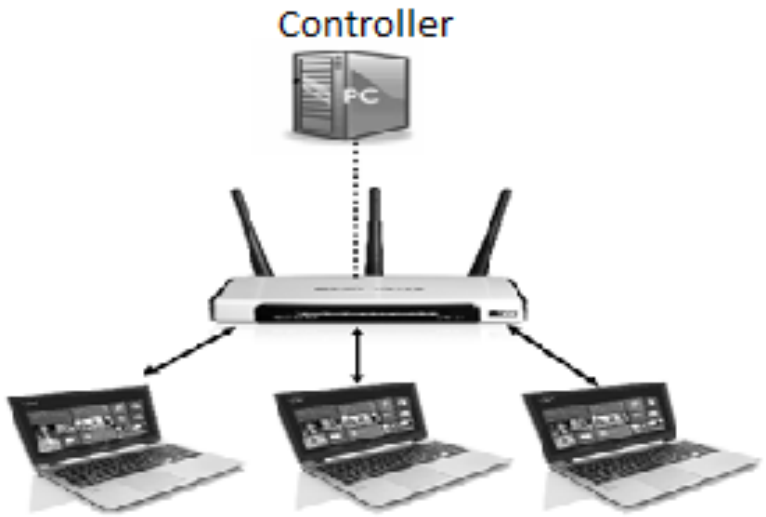

Figure 3.1: Test bed Architecture

Codec Information				Bandwidth Calculations					
Codec & Bit Rate (Kbps)	Codec Sample Size (Bytes)	Codec Sample Interval (ms)	Mean Opinion Score (MOS)	Voice Payload Size (Bytes)	Voice Payload Size (ms)	Packets Per Second (PPS)	Bandwidth MP or FRF.12 (Kbps)	Bandwidth w/cRTP MP or FRF.12 (Kbps)	Bandwidth Ethernet (Kbps)
G.711 (64 Kbps)	80 Bytes	10 ms	4.1	160 Bytes	20 ms	50	82.8 Kbps	67.6 Kbps	87.2 Kbps
G.729 (8 Kbps)	10 Bytes	10 ms	3.92	20 Bytes	20 ms	50	26.8 Kbps	11.6 Kbps	31.2 Kbps
G.723.1 (6.3 Kbps)	24 Bytes	30 ms	3.9	24 Bytes	30 ms	33.3	18.9 Kbps	8.8 Kbps	21.9 Kbps
G.723.1 (5.3 Kbps)	20 Bytes	30 ms	3.8	20 Bytes	30 ms	33.3	17.9 Kbps	7.7 Kbps	20.8 Kbps
G.726 (32 Kbps)	20 Bytes	5 ms	3.85	80 Bytes	20 ms	50	50.8 Kbps	35.6 Kbps	55.2 Kbps
G.726 (24 Kbps)	15 Bytes	5 ms		60 Bytes	20 ms	50	42.8 Kbps	27.6 Kbps	47.2 Kbps
G.728 (16 Kbps)	10 Bytes	5 ms	3.61	60 Bytes	30 ms	33.3	28.5 Kbps	18.4 Kbps	31.5 Kbps
G722_64k(64 Kbps)	80 Bytes	10 ms	4.13	160 Bytes	20 ms	50	82.8 Kbps	67.6Kbps	87.2 Kbps
ilbc_mode_20(15.2Kbps)	38 Bytes	20 ms	NA	38 Bytes	20 ms	50	34.0Kbps	18.8 Kbps	38.4Kbps
ilbc_mode_30(13.33Kbps)	50 Bytes	30 ms	NA	50 Bytes	30 ms	33.3	25.867 Kbps	15.73Kbps	28.8 Kbps

Table 3.1: Bandwidth requirements for Audio Applications

3.3 Priority Queue Configuration

To ensure flows out of starvation priority queues are created on switch. Three queues are created on switch for real-time traffic and best-effort traffic shown in the Fig. 3.2 Queue-1 is for Video traffic, Queue-2 is for VOIP traffic and

Video codec	Resolution and aspect ratio	Maximum video payload bitrate (Kbps)	Minimum video payload bitrate (Kbps)
H.264	320x180 (16:9) 212x160 (4:3)	250	15
H.264/RTVideo	424x240 (16:9)) 320x240 (4:3	350	100
H.264	480x270 (16:9) 424x320 (4:3)	450	200
H.264/RTVideo	640x360 (16:9) 640x480 (4:3)	800	300
H.264	848x480 (16:9)	1500	400
H.264	960x540 (16:9)	2000	500
H.264/RTVideo	1280x720 (16:9)	2500	700
H.264	1920x1080 (16:9)	4000	1500
H.264/RTVideo	960x144 (20:3)	500	15
H.264	1280x192 (20:3)	1000	250
H.264	1920x288 (20:3)	2000	500

Table 3.2: Bandwidth requirements for Video Resolution

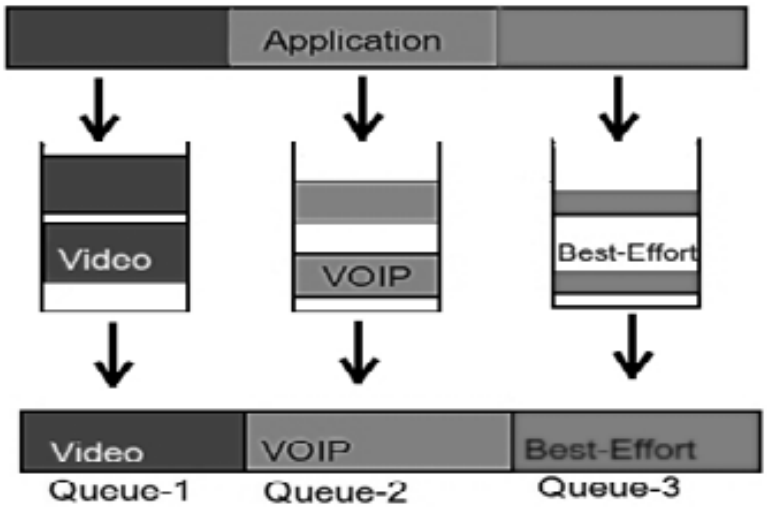

Figure 3.2: Queuing Configuration

Queue-3 is used for Best-effort traffic.Queues are created using dpctl command.

3.4 IMPLEMENTATION

To tackle bandwidth wastage problem, high packet loss rate, flow starvation problem and lack of adaptive behavior we design a Control System for QoS

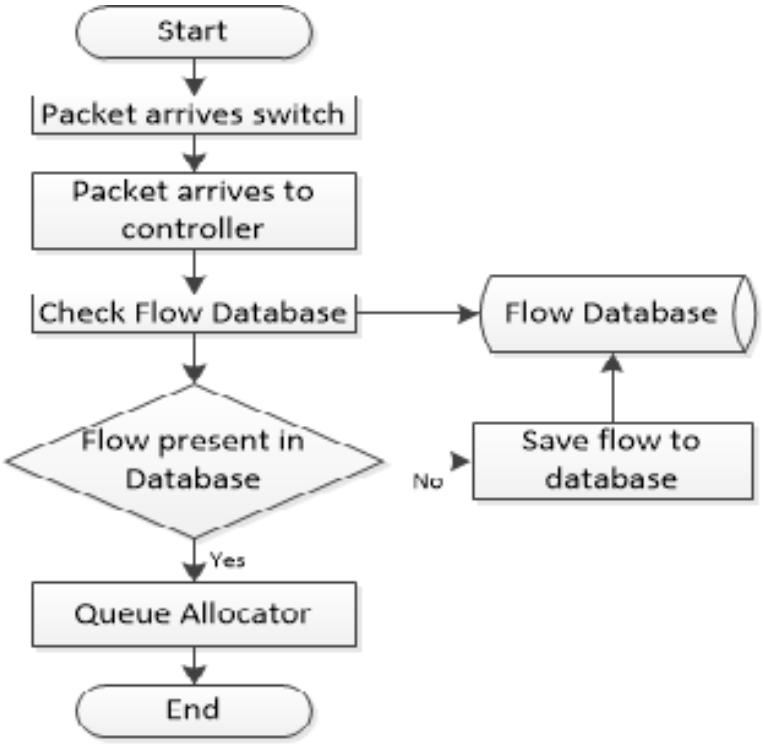

Figure 3.3: Flow Chart of the System

and introduce OpenFlow for QoS in wireless network. The design goals of Adaptive QoS architecture are as follow:

Automated Control: The designed controller is able to perform traffic analysis and then apply best configurations automatically without any manual intervention.

Dynamic workloads: Traditional approach follows class-based static priorities (33) (34) whereas Adaptive QoS controller works on adaptive basis where QoS configuration set to utilize maximum bandwidth.

Efficient resource usage: Traditional approach follows distributed control, whereas OpenFlow approach uses centralized control. Due to this centralized control, network is provided with a global view. Flow Chart of Control system for proposed model is shown in Fig. 3.3 .

3.4.1 QoS API

To work with QoS in wireless network, we integrate QoS API with OpenFlow. OpenFlow (35) is an open specification; it provides large number of APIs to control the flow of packets in a network. On packet arrival at OpenFlow

switch, the switch doesn't forward the packet instead sends the packet to the controller. Controller performs deep packet inspection and then defines certain rule for packets of particular flow. The rule is than send to switch and switch maintain flow table. After that when a packet of that flow reaches the switch, instead of sending packet to controller the switch forwards the packet on the network.

.

3.4.2 Flow database

After deep packet inspection controller saves all the information contained in that packet, within flow database module. These information are maintained on a controller for some specific time. This module helps in identifying number of flows, type of flows i.e. whether the flow belongs to real-time traffic or best-effort traffic.

.

3.4.3 Dynamic Flow Allocator

This is most important module of the system. Administrator must define number of queues manually. This module dynamically assigns priority queues to flow depending on type of flows and available bandwidth of priority queues. If desired priority queue for certain type of traffic (flow) is full and does not have the capability to pass another flow then the flow is switched on another priority queue.

The Flowchart for our proposed solution is given in Fig. 3.4

The flowchart shows overall mechanism of our research which is shown in the algorithm

1: Start
2: Data: *Threshold*, *qi*, *i*, *Fi*, *ViF*, *VoF*, *BEF*, *VideoFi*, *VoiceFi*, *TotalViFi*, *TotalVoFi*
3: Variable Declaration
4: Threshold: Total number of flows a queue can accommodate
5: qi : Queues {Limit i: 1 **to** 3 //q1 for Video, q2 for VOIP, q3 for BE Traffic.}
6: *Fi* : Number of flow
7: *ViF*: Video Flag
8: *VoF*: VOIP Flag
9: *BEF*: Best Effort Flag
10: *VideoFi*: Number of video Flow
11: *VoiceFi*: Number of Voice Flow
12: *TotalViFi*: Total Number of Video Flows

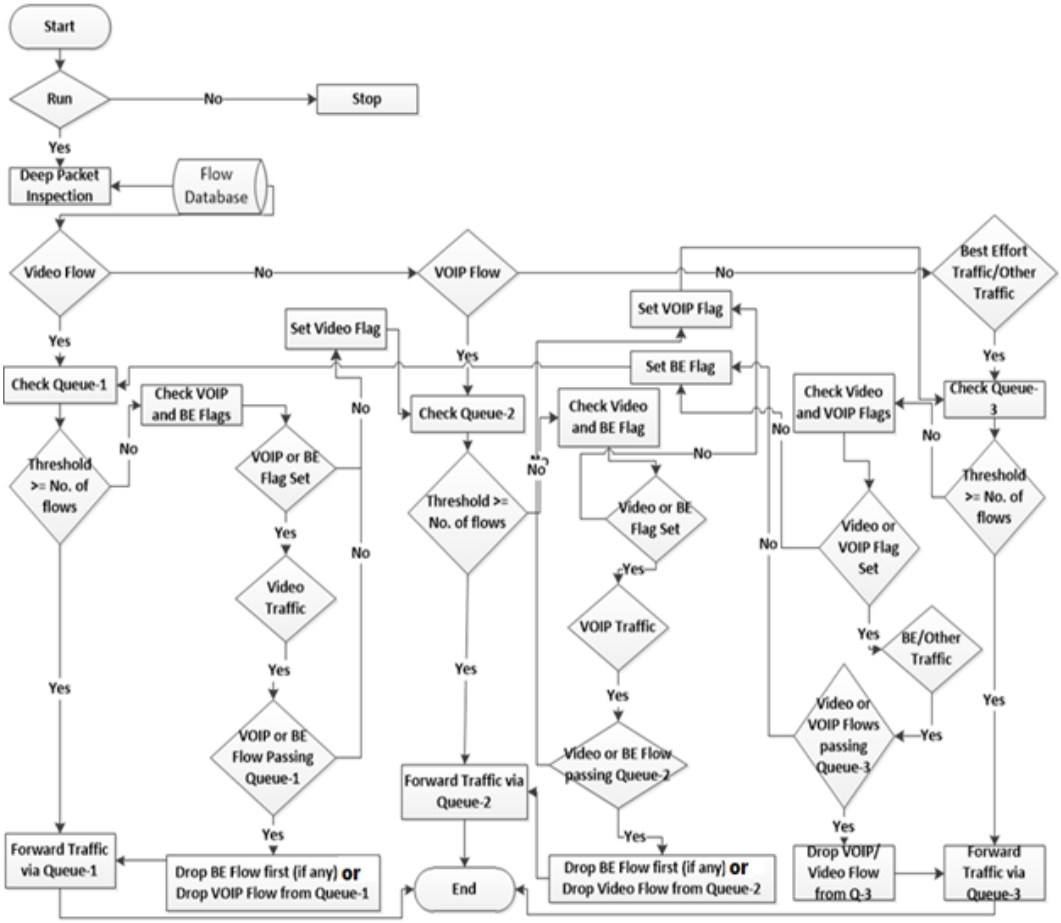

Figure 3.4: Flow Chart

13: *TotalVoFi*: Total Number of VOIP Flows
14: Variable Initialization
15: *ViF* not set
16: *VOF* not set
17: *BEF* not set
18: *Threshold* equals queue bandwidth/packet size
19: *Fi* equals 0
20: *VideoFi* equals 0
21: *VoiceFi* equals 0
22: *TotalViFi* equals 0
23: *TotalVoFi* equals 0 {Repeat the steps for each packet received by Controller}
24: Deep packet Inspection

```
25: if flow ← Video then
26:   if Threshold ≥ TotalViFi then
27:     Enqueue Fi in q1
28:   else if Threshold ≤ TotalViFi then
29:     Set ViF
30:   else if Threshold < TotalViFi then
31:     if Threshold ≥ TotalVoFi then
32:       Enqueue Fi in q2
33:     else if If One Video/Audio Flow not passing q3 then
34:       Enqueue Fi in q3
35:     else if VOF and BEF is set then
36:       Drop BE (if any) if not than drop VoiceFi (if any)
37:       Enqueue Fi in q1
38:     else
39:       Wait
40:     end if
41:   end if
42: end if
43: if flow ← VOIP then
44:   if Threshold ≥ TotalVOFi then
45:     Enqueue Fi in q2
46:   else if Threshold < TotalVOFi then
47:     Set VOF
48:   else if Threshold < TotalVOFi then
49:     if Threshold ≥ TotalViFi then
50:       Enqueue Fi in q1
51:     else if One Video/Audio Flow not passing q3 then
52:       Enqueue Fi in q3
53:     else if ViF and BEF is set then
54:       Drop BE (if any) if not than drop VoiceFi (if any)
55:       Enqueue Fi in q2
56:     else
57:       Wait
58:     end if
59:   end if
60: end if
61: if flow ← BE then
62:   if Flows can accommodate then
63:     Enqueue Fi in q3
64:   else
65:     Set BEF
```

```
    end if
    if Threshold < TotalVOFi then
        Enqueue Fi in q2
    else if Threshold ≥ TotalViFi then
        Enqueue Fi in q1
    else
        Wait
    end if
end if
```

Chapter 4

RESULTS AND DISCUSSION

This chapter shows performance of the system in terms of bandwidth, Packet loss, Delay and Jitter. Although the main theme of the research is to implement QoS in WLAN using OpenFlow but to evaluate performance of the system we first implemented standard QoS model in WLAN using OpenFlow.

We implement different scenarios by limiting bandwidth using TC. For better analysis we allocate bandwidth to queue-1 and queue-2 in a way so that same number of VOIP and Video flows can pass through queues. We call maximum number of flows that a queue can accommodate as threshold. We consider bandwidth and packet loss parameters of our proposed model to minimize flow starvation.

We find packet loss rate using following formula

Packet Loss = Total packets sent in a second – Total packets received in a second.

4.1 Test-1

We take best case scenario with parameters shown in Table. 4.1

Threshold 20 means 20 Video flows can pass via queue-1 and 20 VOIP flows can pass via queue-2 in ideal condition and 1 Video or VOIP flow can pass via queue-3 which is for best effort traffic. Using proposed model and standard QoS model we get bandwidth results shown in Fig. 4.1. Average Bandwidth experienced for proposed model is 3684.596986 Kbit/s and for standard QoS the Average bandwidth value is 3324.9453 Kbit/s. While analyzing the results we notice that 20 VOIP flows and 21 Videos flows are successful and three flows drop. Similarly we experience the packet loss rate shown in Fig. 4.2 The average packets loss rate for proposed model is 23516 packets (7.07 %) and packet loss rate using standard QoS model is 36724

Table 4.1: Scenario 1 Parameters

Duration	200 Seconds
Threshold	20 Flows
No. of Flows	45
No. of Video Flows	23
No. of VOIP Flows	22

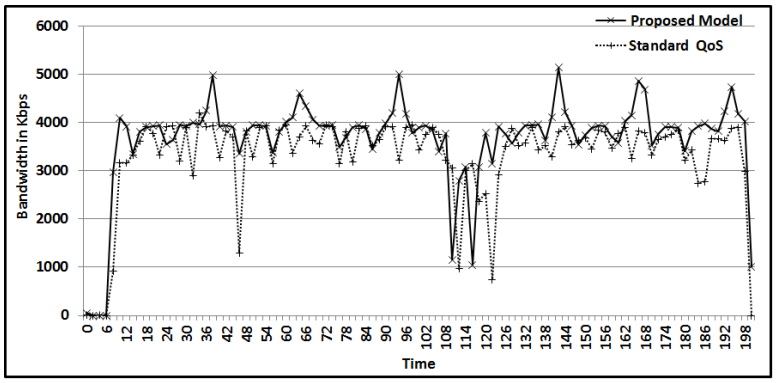

Figure 4.1: Bandwidth Scenario 1

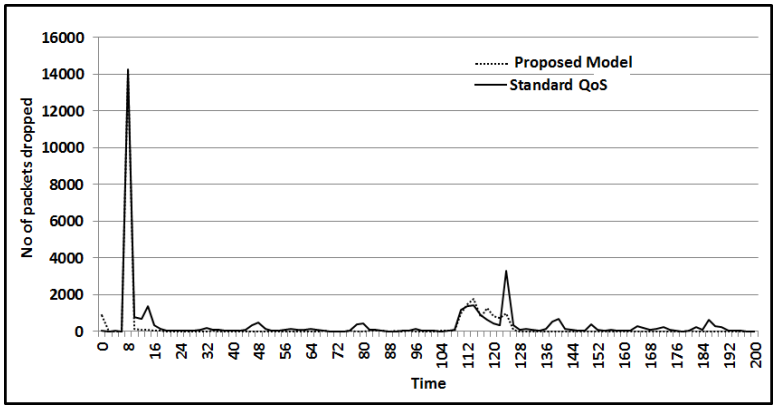

Figure 4.2: Packet Loss Rate Scenario 1

packets (11.65 %). The initial loss at about 8 seconds is because of adding flows to OpenFlow switch. The above graphs show improvements in Bandwidth and Packet loss rate of proposed model compared to Standard QoS model but in case of Delay and Jitter standard QoS model performs better than our proposed model because standard QoS model works only on param-

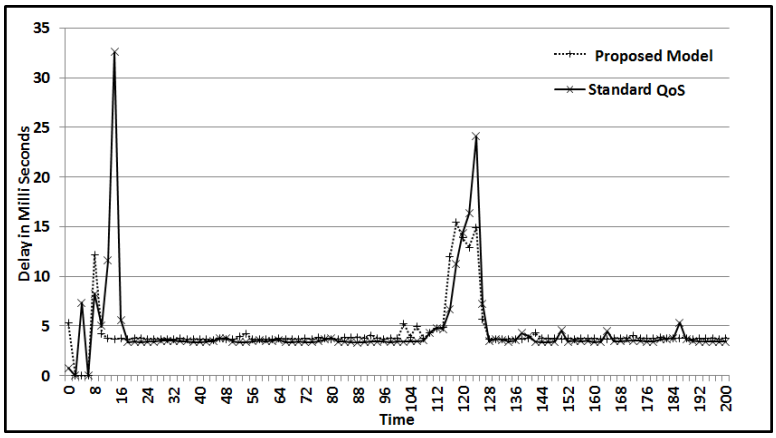

Figure 4.3: Delay Scenario 1

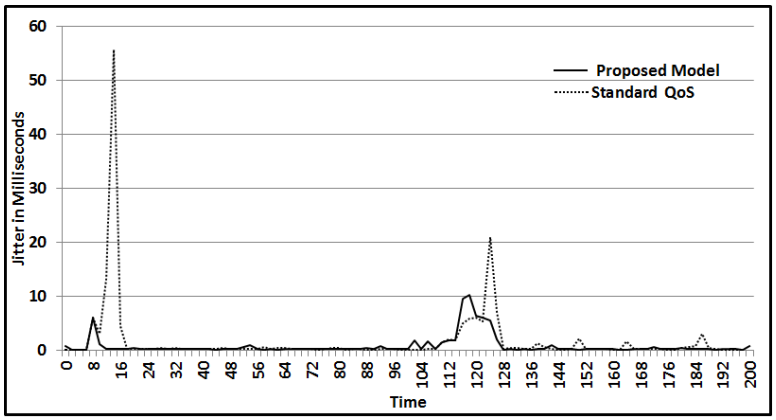

Figure 4.4: Jitter Scenario 1

eter comparisons whereas proposed model works on parameter comparison and queuing status. Proposed model has high delay and jitter as compared to standard but that delay and jitter is tolerable. Delay for Scenario-1 is given Fig. 4.3 The delay graph in Fig. 4.3 shows that Average Delay for proposed model is 0.019997 second whereas Average Delay for standard QoS model is 0.015889 second. The difference between proposed model and Standard QoS model delay is 0.004108 seconds. Jitter for proposed model and Standard QoS model is shown in Fig. 4.4 The Fig. 4.4 shows that average jitter for proposed model is equal to 0.006827 seconds whereas jitter for standard QoS model is equal to 0.026674 seconds. The difference between proposed model and Standard QoS model is 0.019847 seconds.

Table 4.2: Scenario 2 Parameters

Duration	200 Seconds
Threshold	15 Flows
No. of Flows	31
No. of Video Flows	30
No. of VOIP Flows	1

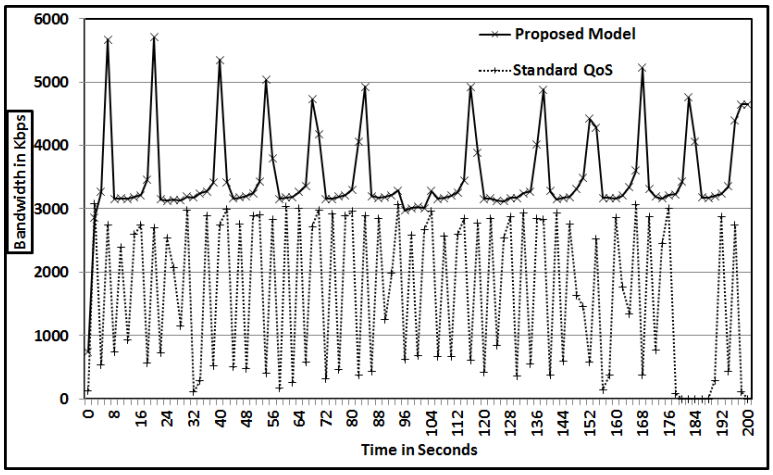

Figure 4.5: Bandwidth Scenario 2

4.2 Test-2

Similarly for better evaluation we implement worst case scenario with parameters shown in Table 4.2. . The results are better in case of our proposed model instead of standard QoS model even in worst case when there are flows of only one type of traffic. From scenario-2 we get the bandwidth results shown in Fig.From the scenario defined in Table. 4.2we get the bandwidth Result shown in Fig. 4.5 The bandwidth graph in Fig. 4.6 shows that average bandwidth for Adaptive QoS algorithm is 3478.830125 Kbit/s while average bandwidth for Standard QoS model is 1658.806 Kbit/s. Similarly Fig. 4.6 shows Packet loss rate graph. The packet loss rate graph in Fig. 4.6 shows that Adaptive QoS algorithm has 4648 (2.57%) packet loss rate while standard QoS has 10460 (10.75%) packet loss rate. This is a handsome difference.

The graph in Fig. 4.7 shows that average delay for proposed model is 0.019997 seconds and average delay for standard QoS model is 0.015889 sec-

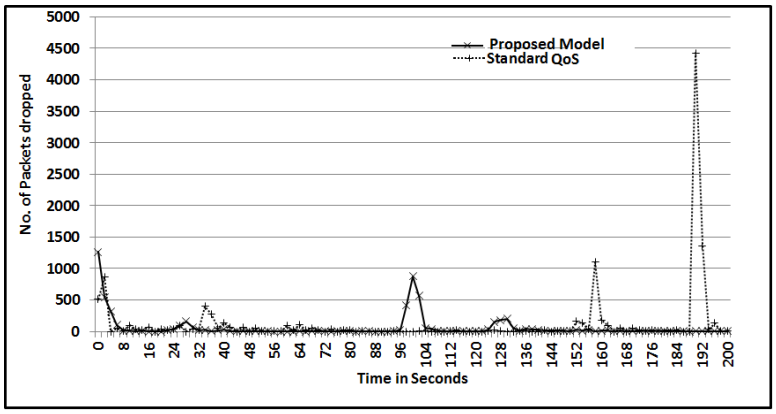

Figure 4.6: Packet Loss Rate Scenario 2

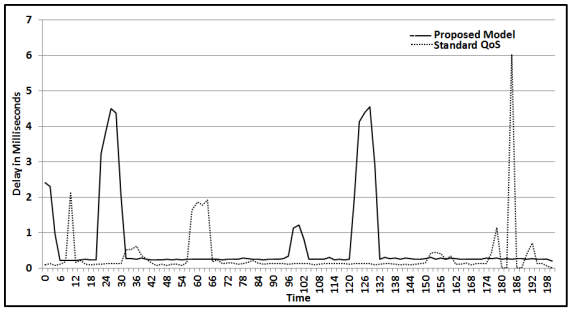

Figure 4.7: Delay Scenario 2

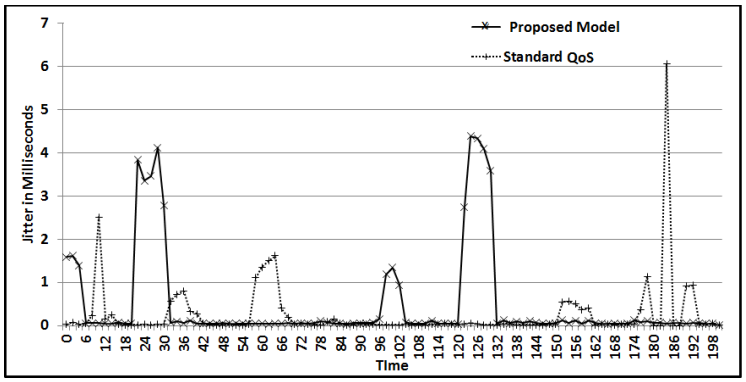

Figure 4.8: Jitter Scenario 2

onds. Hence difference between QoS and standard QoS model is 0.004108 seconds. The graph in Fig. 4.8 shows jitter. The graph trend shows that average jitter of proposed model is equal to 0.006488 seconds and average

Table 4.3: Scenario 3 Parameters

Duration	200 Seconds
Threshold	10 Flows
No. of Flows	21
No. of Video Flows	9
No. of VOIP Flows	12

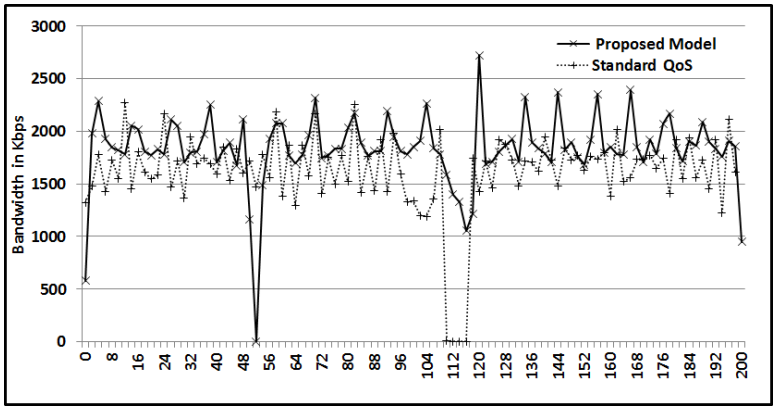

Figure 4.9: Bandwidth Scenario 3

standard QoS model jitter is equal to 0.010336 seconds. Difference between QoS and standard QoS model is equal to 0.003848 seconds.

4.3 Test-3

For further verification we implement another scenario with parameters shown in Table. 4.3 . By running the scenario for 200 seconds we get 1841.585771 Kbps average bandwidth for proposed model and whereas standard proposed model shows 1602.1054 Kbps average bandwidth the trend is shown in Fig. 4.9. Hence proposed model shows 239.48Kbps improvement instead of standard QoS model.

The graph in Fig. 4.10shows Packet loss rate for both QoS and standard QoS model. The graph shows total number of packets dropped is 5436 packets which is 3.15 % in case of proposed model whereas standard QoS model shows 31219 packets loss which is 18.90 %. The results show quite handsome improvement in case of proposed model. Similarly delay for scenario-3 is shown in Fig. 4.11. The delay graph shows that average delay for proposed

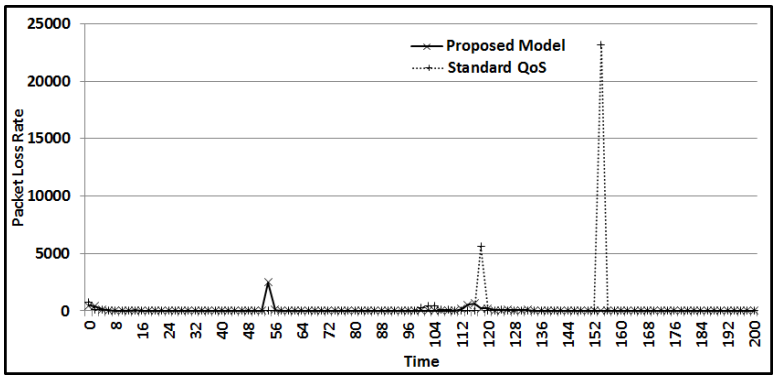

Figure 4.10: Packet Loss Rate Scenario 3

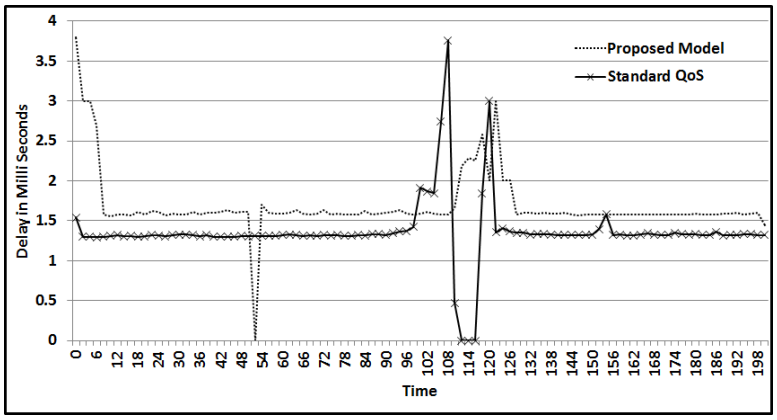

Figure 4.11: Delay Scenario 3

model is equal to 0.089023 seconds whereas average delay for standard QoS model is equal to 0.077058 seconds. The difference between standard and proposed model is equal to 0.011965 seconds.

Similarly Fig. 4.12 shows that average jitter of proposed model is equal to 0.005286 seconds whereas average jitter of standard QoS model is equal to 0.004124 seconds. The difference between QoS and standard QoS model is equal to 0.001162 seconds.

4.4 Test-4

To analyze the trend we get another scenario i.e. Test-4 with 16 Video flows and 7 VOIP flows running in random time with threshold 10 given below. Dark gray lines show VOIP flows whereas light gray lines show Video Flows running in random time. The flows start and end timing is shown in

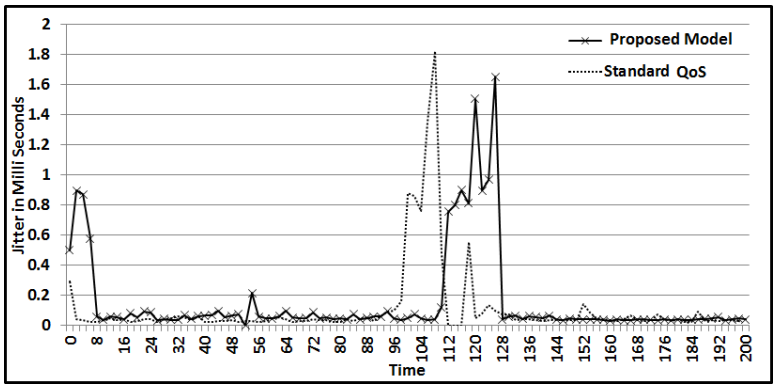

Figure 4.12: Jitter Scenario 3

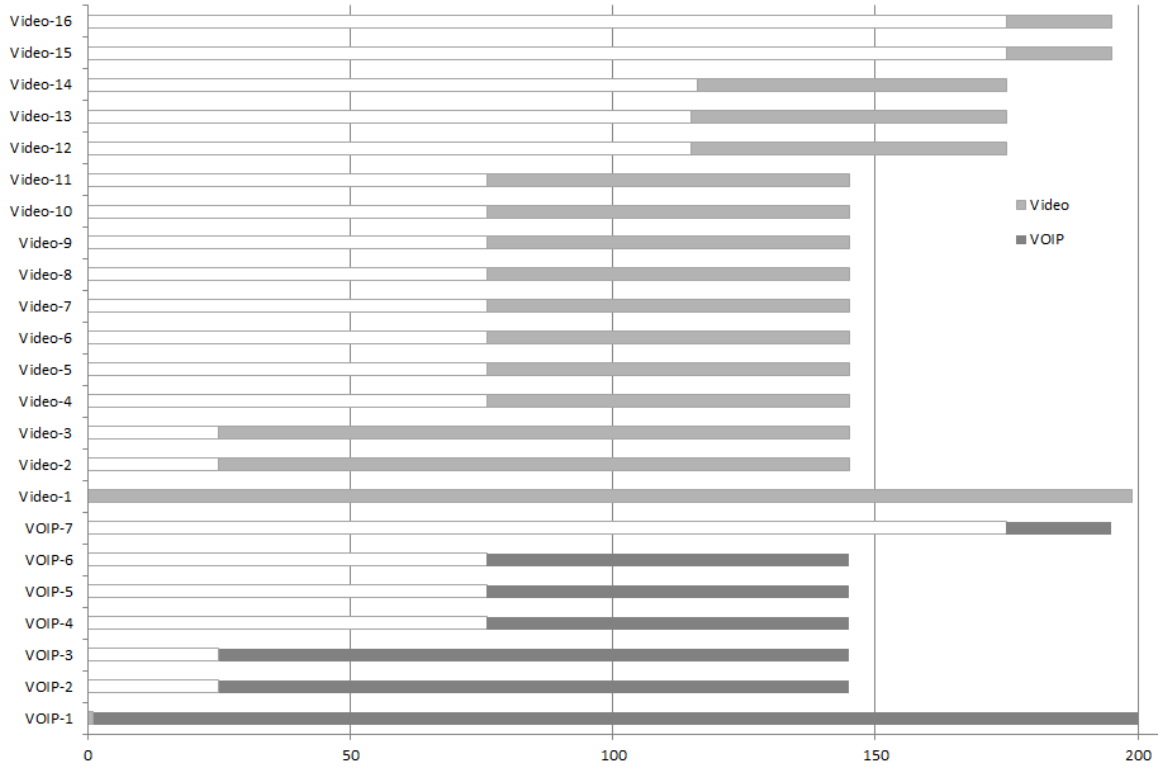

Figure 4.13: Overlapping flows in random time Scenario-4

Fig. 4.13. After running this scenario we got Bandwidth shown in Fig. 4.14. The bandwidth graph shows 961.2659 Kbps bandwidth for Adaptive QoS algorithm. Fig. 4.15 shows bandwidth graph for standard QoS Model and the average bandwidth identified is 842.3772Kbps. By analyzing above graphs Figure-9 and Figure 10 shows that both algorithms work same till 75 seconds.

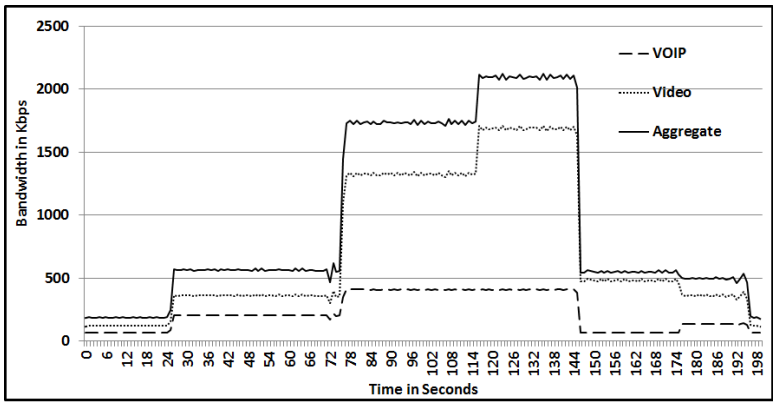

Figure 4.14: Bandwidth for Adaptive QoS algorithm Scenario-4

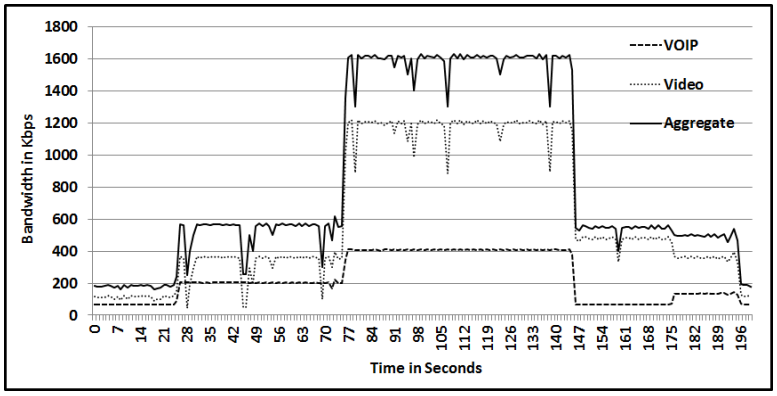

Figure 4.15: Bandwidth for Standard QoS algorithm Scenario-4

Till 75 seconds there are 3 VOIP and 3 Video flows but at 76 seconds 3 VOIP and 8 Video flows arrived so the number of Total VOIP flows reach 6 and Video flows reach 11. As the threshold is 10 so Queue-1 i.e. Video queue is overflow hence 1 flow will be dropped in standard QoS model and Adaptive QoS algorithm will switch 1 Video flow to queue-2. At time 115 seconds 3 more Video flows arrived. So, total number of VOIP flows running is 6 and total number of Video flows becomes 14. Standard QoS model is not capable to accommodate further Video flow as threshold is already met but there is still room in Queue-2 and 3 for flows in Adaptive QoS algorithm. Currently number of active flows in queue-2 i.e. VOIP queue are 7. VOIP queue can accommodate 3 more flows so newly arrived three Video flows will be switched and will pass via queue-2 that is why the graph shows high bandwidth after 115 seconds in Adaptive QoS algorithm. The packet loss rate graph in Fig. 4.16 shows 167(0.24%) packets dropped in Adaptive QoS

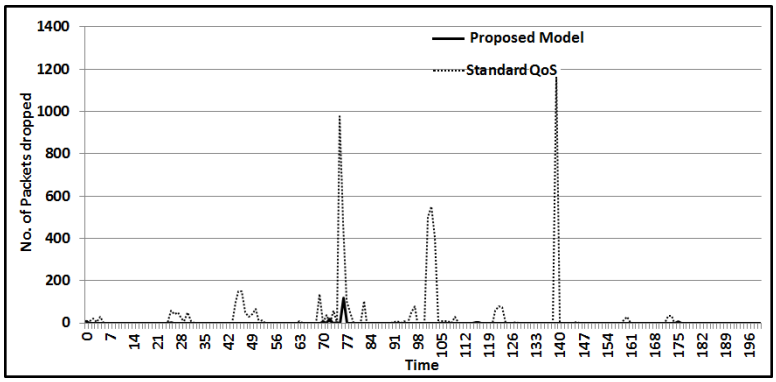

Figure 4.16: Packet Loss Rate Scenario-4

algorithm where as in standard QoS algorithm 6087(8.74%) packet loss is observed which is handsome difference.

Chapter 5

CONCLUSION AND FUTURE WORK

In this chapter, the conclusion with a summary of the research findings along with future directions is presented.

5.1 Conclusion and Future Work

QoS is a crucial requirement in Wireless Networks due to limited bandwidth, losses and delays. Current QoS mechanisms are either static or require manual configuration. Lots of algorithms and protocols are developed but we need some dynamic mechanisms which work according to traffic requirements. We implement our algorithm on physical testbed using TP-Link access point, install OpenWRT firmware and enable OpenFlow on OpenWRT. Hence we design Adaptive QoS algorithm in WLANs using OpenFlow, which allocate queues to flows dynamically depending on queues available bandwidth and hence minimize flow starvation and packet loss due to congestion in a network. The bandwidth wastage was minimized in Adaptive QoS algorithm because does not work like class-based static priority algorithm. In class-based algorithm, traffic belonging to same class will be forwarded to the defined queue. If desired queue is full that flows will be dropped. So, in case of adaptive QoS algorithm i.e. current scheme queues are defined using Linux TC. If queue is defined for certain type of traffic than traffic of that type will be forwarded via that queue if that queue is full than other queues will be checked and on availability of bandwidth in another queue remaining flows of that type will be forwarded through another queue. Priorities are also assigned to type of traffic. Video traffic is given high priority, VOIP traffic is given medium priority and best-effort traffic is given low priority in

other queue.

In future we will work to reduce latency. We are experiencing latency due to two reasons. First is Packet loss rate and other is during decisions taking by controller. So we will modify controller so that latency during taking decisions can reduced. The latency which we are experiencing is not that much greater but to get much better results latency can be reduced by changing decision making criteria on controller.

Bibliography

[1] 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.

[2] M. Ali, H. Qureshi, and M. S. Akhtar (2013), *Analysis of growth in Students Intake and Degree Awarding Contribution: A Comparison of Stanford and MIT*, MLDM 2013: International Conference on Machine Learning and Data Mining, in press.

[3] Cisco, "Wireless lan. www.cisco.com."

[4] L. S. Committee *et al.*, "Ansi/ieee std 802.11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Computer Society*, 1999.

[5] "Ieee standard 802.11e-2005 medium access control (mac) quality of service enhancements (amendment to ieee standard 802.11 1999 edition (reaff 2003) // standard, ieee: 2005." IEEE, 2003.

[6] S. Choi, J. Del Prado, N. Sai Shankar, and S. Mangold, "Ieee 802.11 e contention-based channel access (edcf) performance evaluation," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1151–1156.

[7] D. He and C. Q. Shen, "Simulation study of ieee 802.11 e edcf," in *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, vol. 1. IEEE, 2003, pp. 685–689.

[8] J. Yu and S. Choi, "Comparison of modified dual queue and edca for voip over ieee 802.11 wlan," *European transactions on telecommunications*, vol. 17, no. 3, pp. 371–382, 2006.

[9] X. Ling, Y. Cheng, X. Shen, and J. W. Mark, "Voice capacity analysis of wlans with channel access prioritizing mechanisms," *IEEE Communications Magazine*, vol. 46, no. 1, p. 82, 2008.

[10] J. Jackson Juliet Roy, V. Vaidehi, and S. Srikanth, "A qos weight based multimedia uplink scheduler for ieee 802.11 e wlan," in *Signal Processing, Communications and Networking, 2007. ICSCN'07. International Conference on.* IEEE, 2007, pp. 446–451.

[11] T. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, "The softrouter architecture," in *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, vol. 2004, 2004.

[12] A. Doria, R. Gopal, H. Khosravi, L. Dong, J. Salim, and W. Wang, "Forwarding and control element separation (forces) protocol specification," 2010.

[13] S. Hares, "Analysis of comparisons between openflow and forces," *Analysis*, 2012.

[14] OpenFLow Specification, "Version 1.0. 0 (wire protocol 0x01)," 2009.

[15] B. Pfaff *et al.*, "Openflow switch specification version 1.1. 0 implemented (wire protocol 0x02)," 2011.

[16] OpenFlow Specification, "Version 1.2 (wire protocol 0x03)."

[17] OpenFlow Specification, ", version 1.3.0 (wire protocol 0x04)."

[18] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[19] "Pox controller. http://www.github.com/noxrepo/pox."

[20] D. Erickson, "Beacon," *URL: https://openflow. stanford. edu/display/Beacon/Home. Online*, 2013.

[21] Z. Cai, A. L. Cox, and T. E. N. Maestro, "A system for scalable openflow control," Technical Report TR10-08, Rice University, Tech. Rep., 2010.

[22] "Floodlight. [online]. available: http://floodlight.openflowhub.org."

[23] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," in *ACM SIGPLAN Notices*, vol. 46, no. 9. ACM, 2011, pp. 279–291.

[24] "Trema, full-stack openflow framework in ruby and c. [online]. available: http://trema.github.com/trema/."

[25] "Node.flow. [online]. available: https://github.com/dreamerslab/node.flow."

[26] "Node.js. [online]. available: http://nodejs.org/."

[27] L. Liu, H. Y. Choi, R. Casellas, T. Tsuritani, I. Morita, R. Martínez, and R. Muñoz, "Demonstration of a dynamic transparent optical network employing flexible transmitters/receivers controlled by an openflow–stateless pce integrated control plane [invited]," *Journal of Optical Communications and Networking*, vol. 5, no. 10, pp. A66–A75, 2013.

[28] A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "Openflow and pce architectures in wavelength switched optical networks," in *Optical Network Design and Modeling (ONDM), 2012 16th International Conference on*. IEEE, 2012, pp. 1–6.

[29] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. IEEE, 2010, pp. 408–415.

[30] G. Yao, J. Bi, and P. Xiao, "Source address validation solution with openflow/nox architecture," in *Network Protocols (ICNP), 2011 19th IEEE International Conference on*. IEEE, 2011, pp. 7–12.

[31] F. Zeng, "Design and implementation qos system based on openflow," in *Anti-Counterfeiting, Security and Identification (ASID), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–5.

[32] M. Koerner and O. Kao, "Multiple service load-balancing with openflow," in *High Performance Switching and Routing (HPSR), 2012 IEEE 13th International Conference on*. IEEE, 2012, pp. 210–214.

[33] S. Wang, D. Xuan, R. Bettati, and W. Zhao, "Providing absolute differentiated services for real-time applications in static-priority scheduling networks," *Networking, IEEE/ACM Transactions on*, vol. 12, no. 2, pp. 326–339, 2004.

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

[35] "The openflow switch consortium. openflowswitch.org."

[36] "Tp-link: The reliable choice. www.tp-link.com."

[37] "Openwrt: Wireless freedom. https://openwrt.org."

[38] "Building openwrt. [online]. available: "http://archive.openflow.org/wk/index.php/pantou_:_openflow_1.0_for_openwrt"."

[39] A. Botta, W. de Donato, A. Dainotti, S. Avallone, and A. Pescapé, "D-itg 2.8. 1 manual," 2013.