Adaptive Quality of Service (QoS) using OpenFlow

By

Saqib Haleem

2011-NUST-MS PhD-IT-08

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.

(August 2014)

Approval

It is certified that the contents and form of the thesis entitled "**Adaptive Quality of Service (QoS) using OpenFlow**" submitted by **Saqib Haleem** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Adeel Baig**

Signature: _____

Date: _____

Committee Member 1: **Dr. Zawar Hussain Shah**

Signature: _____

Date: _____

Committee Member 2: **Dr. Adnan Khalid Kiani**

Signature: _____

Date: _____

Committee Member 3: **Dr. Junaid Qadir**

Signature: _____

Date: _____

To My Loving Parents

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: **Saqib Haleem**

Signature: _____

Acknowledgment

First of all thanks to **Allah** almighty; for making my task easier for me. WHO gave me courage whenever I was in need.

Thanks to my **Family** for their love, support and prayers all the time in thick and thin. I owe my deepest gratitude to my research advisor **Dr. Adeel Baig** for his support, guidance and help not only in academic area but also related to other issues, I faced during the time I worked with him on my thesis. One simply could not wish for a better or friendlier supervisor than him.

I am also thankful to my Committee members **Dr. Adnan Kiani, Dr. Zawar Hussain** and specially to **Dr. Junaid Qadir** in helping me and sparing time for me.

Last but not least, I am highly obliged and grateful to **NCP** for providing me with the infrastructure support needed for the implementation of Scheme.

Saqib Haleem

Contents

List of Tables

List of Figures

Abstract

Server Virtualization and Cloud based data centre solutions are becoming popular for increasing resource utilization, better manageability and cost reduction. Server virtualization allows us to run multiple independent instances of operating systems i.e. virtual machines (VMs) on a single physical server. Physical resources of server like CPU, memory, storage and network is shared among multiple VMs, and thus it maximizes the resource utilization. But on the other side, advantage of resource utilization may also raise performance issues if these resources are not managed and provisioned in a proper way among VMs. Network resource is crucial for the performance of network based applications. Sometime despite of having enough CPU and memory resource, application service delivers degraded performance to end user just because of in-sufficient provisioned network resource to the VM. Therefore efficient network resource allocation technique is required for application QoS. Network resource can be provisioned proactively to VMs, but it requires advance knowledge of application user load pattern based on statistical analysis. This technique does not work efficiently because application user load is highly dynamic and does not always follow the predictable pattern. We propose an adaptive QoS mechanism using reactive approach, which monitors the application user load in a real time and provision network resources accordingly. More network resources are provisioned to most demanding applications and vice versa. OpenFlow based Software Defined Networking (SDN) technology is used to meet the objective. Our test result shows that Quality of Experience (QoE) targets of maximum user sessions are met which belongs to demanding application.

Chapter 1

Introduction

1.1 Motivation

In last few years advancements in the network and communication technologies has greatly enriched the network applications. High performance network is also becoming the core requirement of organizations because traditional data centre infrastructure is shifting towards virtualization and Cloud based solutions. Server virtualization allows us to run multiple virtual machines on a single physical server. Physical resources of server like CPU, memory, storage and network is shared among VMs/applications. Therefore it provides several benefits including maximum resource utilization, cost reduction and better manageability.

Resource utilization benefit may also create performance issues, if resources are not allocated and shared in efficient way. Virtualized server hosts multiple applications and performance of each application depends on the provisioned CPU, memory and network resource to VM. Network resource plays a key role in the performance of network based application. If insufficient network resource is allocated to application, then it may degrade the end users perceived performance irrespective of having enough CPU and memory resource. Therefore it is necessary to provision network resource in efficient way so that end users perceived performance can be improved.

In virtualized infrastructure by default applications share network resource in best effort manner, because underlying Internet Protocol (IP) is the best effort delivery protocol and it does not provide guaranteed services. However this default behaviour is not desirable all the time, because every network application has different priority and bandwidth requirements. For example real-time voice and video services are delay sensitive applications and require fixed bandwidth. Increasing bandwidth beyond the requirement cannot improve performance. On the other hand elastic application like ftp and http are bandwidth hungry applications and performance improves by increasing bandwidth. Similarly sometime bandwidth resource is allocated to applications according to business requirements. More bandwidth is allocated to high priority applications and least network resources are provisioned for low priority applications. In some cases bandwidth allocation strategy depends on applications popularity or demand factor.

In virtualized environment multiple application services share same limited physical network resource which may increase the chances of network congestion. Therefore Quality of Service (QoS) becomes a necessary requirement to provide differentiated and guaranteed services to selected users or applications. Network bandwidth provisioning factor mainly impacts application's QoS parameters like throughput, loss and delay. Two types of approaches can be used for network resource allocation i.e Proactive and Reactive approach. Existing virtualization products offer proactive network resource management feature, which allows us to statically provision network bandwidth for each VM/application. Static bandwidth provisioning requires advance knowledge of application usage pattern based on some statistical analysis. Proactive resource allocation technique does not work efficiently because application user load may not always follow the pattern according to prediction. Every hosted application has fluctuating user demand, and thus each application requires dynamic network resources to meet application's QoS requirements. Dynamic network resource provisioning for adaptive QoS is becoming an important resource management issue in virtualization environment.

1.2 Problem Statement

The problem statement of the thesis is as below:

"To design and implement adaptive Quality of Service (QoS) mechanism for virtualized environment using openFlow"

1.3 Thesis Contribution

Contribution of this thesis is development of adaptive Quality of Service (QoS) mechanism. Our proposed mechanism dynamically provision network resources according to applications demand factor. Maximum resources are guaranteed to popular applications i.e applications with more number of users and vice versa. This mechanism ensures that maximum users, which belongs to demanding application receives service performance.

1.4 Thesis Organization

There are six chapters in this thesis which are organized in a following way: In Chapter 2, Background of the thesis work is explained. It provides the comparison between traditional data centres and virtualization technology. It also provides the literature review of traditional Quality of Service(QoS) and Quality of experience(QoE) In Chapter 3,problem in current virtualized environment is formulated. In Chapter 4, Design methodology of proposed adaptive QoS solution is discussed Chapter 5 describes the test bed implementation, and discussion on performance results. Chapter 6 includes conclusions regarding the work carried out in this research and proposes some possible future extensions to this work. In the end, references to related work are given.

Chapter 2

Background and Literature Review

This chapter introduces the background of the work presented in this thesis.We first discuss the evolution of data centre technologies. Section 2.1 discusses traditional data centre computing model and issues associated with that model. Then we discuss the latest virtualization and cloud computing concepts and their benefits.Section 2.2 discuss the importance of network resource management for QoS.It discuss tradational QOS models and different QoS solutions and related work for virtualized environment.Section 2.3 discuss the OpenFlow architecture and its capability to provide QoS.In Last section 2.4 Quality of Experience (QoE) and its related work is discussed.

2.1 Evolution of Data centre technologies

2.1.1 Traditional Data centre implementation

With the availability of low cost and high performance computing and network technology, organizations started to build data centers to host their application services in their own premises. In traditional data centre environments mostly application servers are deployed and configured as stove-piped which means every

application service runs independently on dedicated physical server. Figure 2.1 shows the architecture of server in traditional data centre. This type of data centre infrastructure requires many physical servers to host organizations application services which raises certain issues. One major issue is resource under utilization. Resources of physical servers like CPU, memory and storage remain under utilized in this environment; because of the fact that dedicated servers are used for every application service which usually does not consume full resources. Data centers running with this conventional setup ultimately become expensive solution due to cost of purchasing multiple physical servers, and more building space is required to place this growing server farm . Similarly to run this setup more power and air conditioning is required. More network resources are required to provide physical connectivity to all servers. It also increases the complexity of deploying, managing and troubleshooting infrastructure and thus requires more manpower to run this type of data centre environment. Hence traditional data center approach is expensive and in-efficient in terms of resource utilization and power consumption.

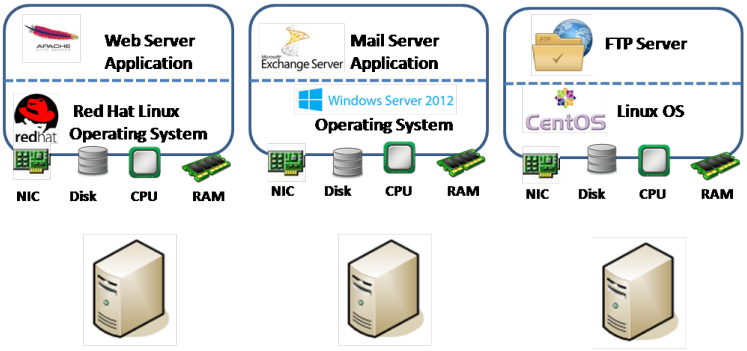


Figure 2.1: Test bed Architecture

2.1.2 Virtualization and Cloud based data centres

To overcome the issues in traditional data centers, Virtualization technology has been introduced in last few years in order to make data centers fully utilized, energy efficient and cost effective. In server virtualization, physical server is virtualized by using software based or firmware based hypervisor. Many Virtualization products are available in market, including VMware [3], Xen Citrix [4], Microsoft

hyper V [5] and KVM. Server virtualization ensures maximum utilization of physical resources, by allowing us to create and run multiple logical containers of Operating systems i.e. virtual machines (VMs) and application instances on a single physical server. CPU, memory, storage and network resource from physical server is shared with VM. Different proportion of physical resource is allocated to each VM.virtualization architecture is shown in figure 2.2

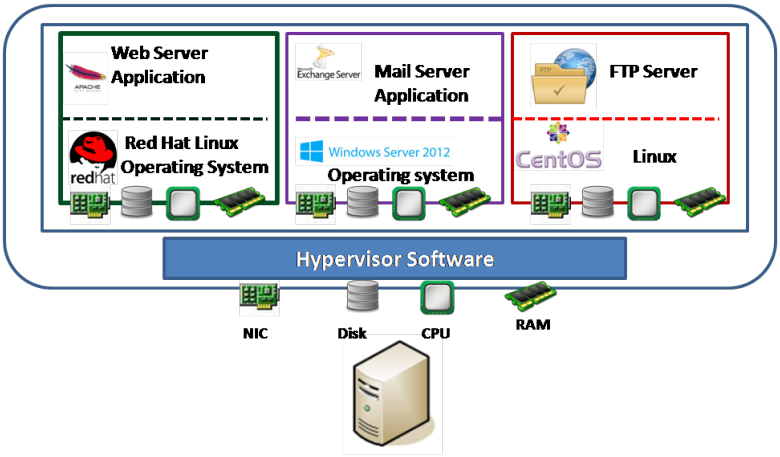


Figure 2.2: Testbed Architecture

It also reduces overall cost, because multiple virtual machines can run on single physical server. Usually server VMs are created on virtualization infrastructure for hosting Application services, but due to emerging Virtual Desktop Environment (VDI) technology organization can also create and allocate virtual desktop machines on virtualization infrastructure for their staff instead of purchasing dedicated physical desktop machine. Thus it minimizes the number of machine requirements, and ultimately occupies less building space, require less power and cooling as compared to traditional data centre. Benefits of virtualization technology have motivated the organizations to migrate their applications on virtual machines.Figure 2.3shows virtualization adoption trends.

Cloud is another buzzword in IT industry. It is the latest technology which has emerged from virtualization. It is basically advanced form of virtualization in which different resources like CPU, Storage and networks are provided as a service. In other words it is utility based computing infrastructure, where resources are

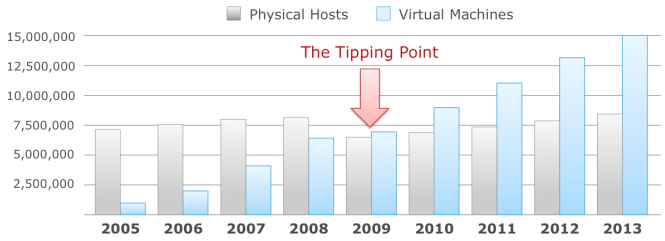


Figure 2.3: Applications on Virtualization infrastructure[1]

provided to tenants on demand basis. This model is much more cost effective and efficient because in this model you have to pay only for what you have used. Physical resources are shared among tenants.

Cloud technology allows organizations to build Virtual Data Centre (VDC), which is basically group of VMs, storage and network. These resources are allocated to organization according to their requirement from pool of resources. In this model, VMs can be easily and quickly deployed. There is no headache of disaster recovery and backup procedures at customer end. System upgrades are automatic and scaling the system up or down is easy.In order to access the Cloud based applications high performance network is mandatory requirement.CLoud technology is also being adopted by organizations and hence it results in more network traffic.Figure 2.4shows impact of cloud technology on network traffic.

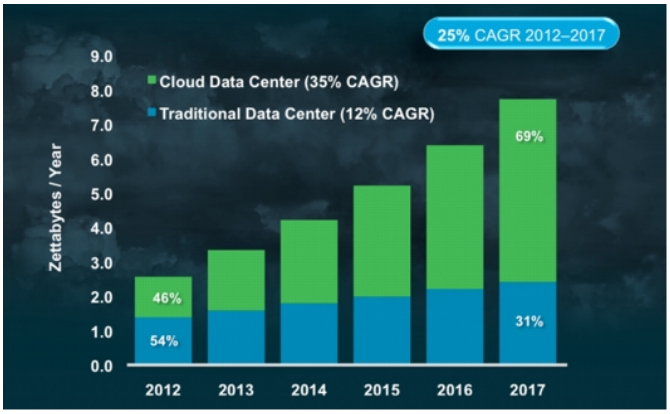


Figure 2.4: Network traffic trend [2]

2.2 Network Resource Management

Resource management is an important task in virtualized environment.It involves adequate provisioning of computing, memory and networking resources to each application or VM , in order to meet the performance goals of an application during dynamic and time varying user load.

Different QoS parameters for network applications including throughput, packet loss, and delay mainly depends on provisioned network resource. There are generally two strategies which can be used for resource allocation. Proactive and reactive. In proactive technique, network resource utilization pattern is learnt in advance and then network resource is provisioned to each application on the basis of that prediction. This technique delivers best performance until prediction works well. However it can fail to deliver performance when traffic pattern do not follow the predictable pattern due to different conditions like flash crowds, noisy traffic, or change in application user load due to some event.another issue with this technique is that it requires too much effort to predict traffic pattern based on statistical analysis.

Reactive approach is another network resource provisioning technique.In this technique network resource is allocated according to demand. Whenever user load is detected towards application, network is provisioned to application accordingly. This approach is simple and attractive because it does not require advance knowledge of application user load.However efficacy of this technique depends on the its timely user load detection and resource allocation ability.

2.2.1 Traditional QoS approaches

Different techniques are developed to maintain performance of application in a limited network. In this section we first discuss traditional QoS techniques which are developed to achieve application's performance goals

Over provisioning is the basic technique to provide QoS in best- effort delivery

networks, because performance is only degraded when resource utilization exceeds from available capacity. Provisioning extra bandwidth between links can avoid from situation of congestion and maintains the network performance. However this model is not cost effective and network bandwidth remain under utilized.

Initially Asynchronous transfer Mode (ATM) technology was introduced in order to provide QoS in legacy SDH/SONET based telecommunication infrastructures. This technology is cell switching technology having fixed cell for fast switching. Point to point and point to multi point connections are made, using concept of virtual channels and virtual paths. ATM provides QoS to certain services like real time Voice and video services using traffic contracts. It provides four types of service contracts which includes Constant bit rate (CBR), variable bit rate (VBR), Available bit rate (ABR), Unspecified bit rate (UBR).These contracts are negotiated between ATM switches at both ends.

ATM technology only provides QoS in backbone network, because QoS contracts can be defined only between ATM capable switches. It cannot ensure end to end QoS, because end user devices use native IP protocol. Another reason is that ATM cannot do anything above layer 2, which means all layer 3 flows which have been aggregated on layer 2 ATM, cannot be differentiated. Thus all the flows end up competing against one another for obtaining network resources. In order to ensure end to end Quality of Service (QoS), it was needed to provide traffic control mechanisms at Layer 3. Therefore IETF introduced two architectures for providing QoS i.e integrated services (Intsrv) and Differentiated Services (Diffsrv)

Integrated service (Intsrv)[6] provides hard QoS, by reserving resources like bandwidth and buffer. Its architecture specifies the elements to guarantee QoS at flow level. Each flow is uniquely identified by source and destination IP address pair, port numbers and protocol type. Resources are reserved by using Resource Reservation Signalling protocol (RSVP). RSVP is used by end host to request specific QoS requirements from network. Routers also use this protocol to distribute QoS requirements by end host in network. It works similar to circuit switched

networks in which connections are established first, before actual data transfer. RSVP is not a routing protocol. It is just a control protocol which inter-operates on routing protocols and performs QoS management of flows which are forwarded according to routing protocol. Different traffic control mechanisms are implemented in Intsrv based router in order to ensure QoS to individual flows, which includes admission control, packet scheduler and packet classifier mechanisms. Admission control component accept and reject traffic flows on the basis of demand factor and available network resources. Packet scheduler forwards the packet by maintaining queues for traffic streams. Traffic classifier component maps incoming traffic to particular class based on traffic type. IntServ architecture has advantage that it provides end to end QoS solution and ensures QoS guarantee to individual flow level. However on the other side it has some drawback. One major issue is the scalability factor i.e it works only on small scale, as it is difficult to keep track of all resources reservation requests and signalling between end hosts on large networks. Complexity and cost of the infrastructure increase, if Intsrv architecture is used on large scale. It also requires all routers in the path to support IntServ, which makes fundamental change on network core. RSVP protocol itself is an additional overhead in IntServ based QoS architecture.

Differentiated Service (DiffSrv) [7] based architecture provides QoS based on aggregated traffic classes. In this architecture packets are classified and marked to receive per hop forwarding behaviour on router and along the path to destination. It uses 6 bit Differentiated Service Code Point (DSCP) field of IP header for classification of packets and each packet is classified upon entry into network. It is scalable as compared to IntSrv because packets are marked and classified at network edges for getting specific preference while passing through router. Each traffic class has different priority level at router, and therefore forwarded accordingly. However it provides soft QoS and cannot assure end to end QoS, due to it's per hop behaviour. It can only provide desired QoS with high probability.

Multi Protocol Label Switching (MPLS) developed by IETF also has leverage

to provide QoS , although MPLS is not a QoS architecture i.e its specification does not include tools for Qos but its fast switching mechanism makes it suitable for providing efficient services especially for delay sensitive applications. It has the combined features of ATM and IP technologies. Fast switching is achieved with its label switching mechanism. In conventional IP based networks IP table lookup is performed for forwarding packet to next hop. It requires long IP header to process. However in MPLS labelled packet are switched after label lookup, without looking into IP lookup table. This process is performed directly into switch fabric, thats why it is fast. MPLS traffic engineering (TE) capabilities and its integration with DiffSrv can provide much better performance to application services.

With the evolution of Virtualization technology, people started to work on network performance issues in cloud based data centers. Different mechanisms are proposed for fair network sharing between VMs, applications and tenants.

2.2.2 Network QoS and resource management in Virtualization

In this section we first discuss the network resource management features available with popular virtualization platforms and then present other QoS related work for Virtualized data centres.

VMwares virtual switch (vSwitch) provides basic QoS feature by offering traffic shaping policy [8]. With this policy network bandwidth of VM can be controlled by configuring average rate, peak rate and burst rates. On standard switch this policy can control only outward traffic. VMware also provides Distributed virtual Switch, which provides advanced traffic control features. In this type of switch user defined network resource pools can be configured. Network resource pool setting allows us to allocate resources for VMs by configuring physical adapter share and by adding QoS priority tags from range 1-7. Physical adapter share for host can have value 100 (unlimited), 50 (normal) and 25 (low). QoS priority tag value 1 represents lower priority and 7 represents highest priority.

OVirt[9] is the virtualization management platform used for KVM based virtualization infrastructure. It provides different resource management features including network QoS. With this tool per VM traffic shaping can be implemented.It allows us to control network bandwidth of VM. Xen based virtualization platform also provides basic traffic shaping facility for virtual interfaces. Most of the commercial products provide basic QoS facility. They are not too much useful and efficient because these are proactive techniques and require prior calculation of bandwidth requirement against each VM.

Different solutions are proposed for network resource management in virtualized data centres and Cloud environments. Thomas Voith et al [10] focused on network resource management methods in virtualized data centers in order to ensure QoS. Author proposed idea of creating different QoS classes for different types of traffic. Aggregated traffic flows of each type are assigned to particular class. Different proportion of overall bandwidth is provisioned to those classes according to traffic sensitivity.

Netshare [11] is another approach for predictable bandwidth allocation, high utilization and bandwidth isolation. In this approach hierarchical weighted max-min fair sharing algorithm is used in which bandwidth is assigned to application according to its weight. Weight is assigned to VM either manually by network administrator according to tenant payment factor, or weight can be automatically calculated and assigned according to VM location in data centre. Automatic weigh assignment is based on two factors; first factor is that weight per application varies according to switch port; second factor is the VM placement factor, which is considered in automatic weight assignment. It computes both the downstream/upstream sums of bandwidth assigned to all VMs allocated to application with respect to port. Then weight assigned to that application service is smaller of two values.

In [12] seawall provides performance isolation in virtualized data centres. It provides fairness in resource utilization by different Tenants. Central controller is

used to predict congestion caused by a Tenant VMs and applications, it then limit the network in order to optimize the resource utilization.

SecondNet [13] is another bandwidth Guarantee approach in virtualization environment. In this approach concept of virtual data centre (VDC) is proposed which contains set of virtual machines and SLA. Each VDC has its own IP address space. Three service models were provided, which include best effort services delivery model, Type-1 model with ingress/egress bandwidth guarantees VDC and Type-0 model having bandwidth guarantee between 2 VMs. SecondNet has logically central VDC manager, which performs VDC allocation, bandwidth reservation and Port based source routing. There are several other proposals which address network performance guarantees in virtualized data centers, which include Gatekeeper [14]. It provides predictable performance at the access network by giving illusion to tenants that all of its VMs are connected to non-blocking single switch. It uses hyper visor for rate limit purpose.

Oktopus [15] is also based on Gatekeeper logic, but it extends its functionality by virtual oversubscribed Cluster (VOC) model. Falloc [16] also provides fair distribution of network bandwidth between VMs. It is based on OpenFlow technology and provides bandwidth guarantee to VMs by allocating base bandwidth, and ensure maximum utilization by using weight mechanism. Residual capacity is shared between VMs based on their weight.Weighted Performance centric fairness[17] based resource allocation mechanism is also proposed in order to maximize the application performance. Weights are assigned to applications according to their performance requirements. More Network bandwidth is allocated to flows according to proportion of their flows.

2.3 OpenFlow based software defined Networking (SDN)

Software Defined networking (SDN)[18] is a new concept which has been evolved in last few years. It has radically changed the way networks are configured and managed. This concept brings agility and innovation in network world by allowing us to program a network in our own way according to business requirements. SDN makes network simplified by reducing operational complexity. There are several approaches to achieve software defined networking. One approach for SDN is using virtual overlay networking. In this technique software based virtual network is created on top of physical network. This technique does not require any specialized network hardware. Generic Routing Encapsulation (GRE), Virtual Extensible LAN (VXLANS) and Stateless Transport Tunnelling (STT) protocol based tunnels are used to transport traffic between virtual networks which are segregated by physical network. Therefore it can be implemented on existing network infrastructure as it requires only IP connectivity between switches. Another popular approach for Software Defined networking is to use OpenFlow [19]. It is an open standard and it was developed by researchers at Stanford University. Now its development is managed by open networking foundation (ONF). This technique decouples the control and forwarding functionalities of network devices. Traditional network devices like switches and routers consist of two functional components. First one is the control plane, which handles all the control logic required to take decisions of processing and forwarding packets passing through that device. Flow table is built by control plane. Another component is data plane, which forwards the incoming packets at line rate by looking at flow table entries.

In OpenFlow based approach data plane and control plane of network devices are separated by using central controller. OpenFlow enabled switches only performs data forwarding function, based on flow table entries. All the control logic is

programmed on central server known as controller. This controller provides flexibility in programming. All the instructions and actions from remote controller are communicated to switches and routers using OpenFlow protocol. Isolation of control plane provides freedom and much more flexibility to customize the data forwarding decision on central controller according to ones requirement and without overloading the network devices.

Technically OpenFlow is a protocol which standardizes the communication between data plane and control plane. OpenFlow architecture is shown in figure 4.1. It consists of three components.

Flow table OpenFlow Protocol Secure channel for communication between controller and switch.

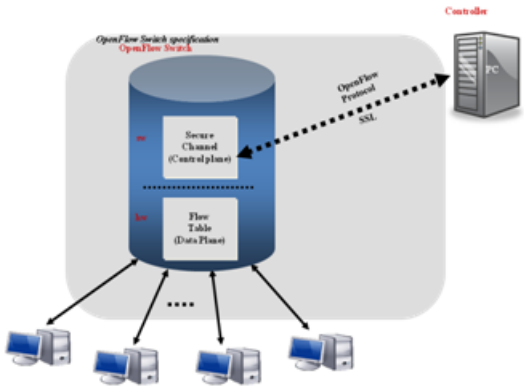


Figure 2.5: OpenFlow Architecture

There are two models to populate flow table entries for packet forwarding using OpenFlow. First model is to use pro-active technique. In this model flow table entries are pre-populated in network devices, based on controller logic. In this way incoming flow does not have to contact central controller for forwarding decision. This approach is fast and efficient but inflexible. Second model is to use reactive-technique. In this technique initially flow table is kept empty .Flow table is built according to arriving flows. Controller is consulted whenever new flow is received. After consultation flow entry is added into switch. There are several use cases of OpenFlow technology in industry, including QoS, Security implementation and

policy based routing.

2.3.1 Quality of Service (QoS) using OpenFlow

Different methods are available to implement Quality of Service (QoS) using OpenFlow in network switches. We will briefly discuss some basic techniques and different proposed solutions.

Separate virtual ports can be created on a switch for different type of traffic classes. Different priority and weight parameters are defined for each virtual port according to requirement. Controller is programmed to forward the traffic flow from particular traffic type to its associated virtual port. By this way bandwidth of different classes can be managed.

OpenFlow is capable to re-write the ToS/DSCP field of IP header. This technique can also be used for marking packet and their classification. However it is not flexible because it has limited set of predefined services.

OpenFlow version 1.2 support en-queue action. Therefore QoS can be implemented by classifying and sending flows to different traffic queues according to their priority. OpenFlow version 1.3 [20] supports per flow rate limiting feature by providing meter table.

Flow visor [21] is an OpenFlow based solution in which network resources are virtualized by slicing physical network into multiple logical networks. It can be integrated with QoS controller and flows of each application can be assigned to each virtual slice for strict performance isolation.

Author in [22] proposed the framework for automated QoS in converged networks using OpenFlow. Proposed solution defined different network slices, having different specifications. Then aggregated traffic flows are automatically assigned to different network slices according to their criticality. But this framework cannot detect application performance requirement.

QoSFlow [23] extends the OpenFlow protocol and provides packet scheduling functionality. Currently it provides control of HTB, SFQ , RED and FIFO based

packet scheduler. It adds new messages in OpenFlow protocol for controlling traffic queues.

2.4 Quality of Experience (QoE) and related work

Quality of Experience (QoE) is another performance parameter, which is considered nowadays by many service providers. QoE is a user centric performance parameter and it refers to overall acceptability of an application or service, as perceived subjectively by the end-user. Quality of experience (QoE) depends on various objective and subjective factors. Objective factors include QoS parameters from network and application point of view. However subjective factor involves user experience level, emotions, service billing, client terminal, and context etc. These components are difficult to measure because it varies from one user to another. Below we discuss most related work and solutions for improving QoE.

Author in [24] proposed the idea of traffic prioritization for home users. It dynamically prioritize the traffic with which user is interacting and rest of the background traffic is set as low priority. It detects user interacting application by identifying active windows which is in focus. By this way user perceived performance is improved. This solution requires additional client agent on user computer. Secondly this solution is viable for only home user machines.

Another survey paper on Quality of Experience (QoE) in cloud networks [25], also concluded by different field experiments that bandwidth is the main factor which impacts QoE. Interactive application are highly dependent on response time and it remain fine, if sufficient bandwidth is provisioned. Author [26] proposed cloud Rank framework, which predicts QoS ranking for cloud services. In order to make decisions about personalized QoS ranking of cloud services for current users, it looks at past user's experiences.

Author [27] proposed the architecture for user controlled Quality of Experience (QoE) using OpenFlow. This architecture basically allocates the network resources

by slicing the access link at customer end. In this model it is considered that ISP is running central controller and customer communicates their resource requirements to ISP via API. User friendly GUI is presented to customer in which customer can set its resource allocation strategies for its application and devices, and then communicate those requirements to ISPs central controller. GUI has options to set device level priorities, and application level shares. Devices with highest priority level are allocated more bandwidth slice. Device priorities and application shares are hierarchically assigned in terms of bandwidth.

Author [28] also proposed the utility function which maps, Quality of service parameters with respect to user satisfaction. Every user type has different utility function based on different characteristics and satisfaction levels. If one can calculate utility function then it will become easy to allocate network resources more optimally. Author provided the direction by analysis of two user types case study, which can be extended to more generic user types analysis.

[29] proposes the fair resource sharing mechanism to maximize user's quality of experience. Author introduced Quality of Experience (QoE) framework (QFF) based on OpenFlow technology, which fairly distribute network resources in multimedia networks. Simple bandwidth allocation strategy based on active users leads to unfairness. Therefore QFF monitors DASH (Dynamic Adaptive Streaming over HTTP)[30] applications and dynamically assign optimal bandwidth according to device requirements based on its resolution, because connected multimedia devices like smart phones, HD TVs have different resolutions and hence require different bandwidth. Its utility function maps video bit rate at particular resolution with respect to QoE by end user.

Remote Virtual Desktop (RVD) is common application which is being used in Cloud infrastructure. Quality of experience (QoE) in RVD sessions are discussed in paper [31]. As RVD is an interactive application therefore response time matters a lot in QoE. Bandwidth shortage and packet loss put negative impact on response time. Author concluded with multiple experiments that sufficient band-

width allocation and low latency is the core requirement for improved QoE in RVD sessions.

Kailong Li et al.[32] proposed solution of QoE improvement in FTTH networks. SDN based solution is presented in which user identifies the application for which he need QoE and put request to SDN controller for bandwidth resource demand against that application. At user end socket client transports the user request and on the other side socket server receives user request and forward it to controller for bandwidth resource provisioning.

Cloud based SDN solution is also proposed for home user quality of experience [33]. In this solution cloud hosted GUI is presented to user.User will be able to communicate their application preferences and QoS requirements to ISP via that GUI.A. Ferguson et al.[34] proposed a framework which allows application to automatically communicate with network devices via controller and set different network configuration parameters according to application demand.

Existing QoS techniques are either complex or require manual and fixed QoS configuration changes, which may become inefficient in terms of resource utilization. Similarly with the emergence of virtualization technology different models are proposed for sharing network resources among VMs. But most of the work is done to provide network fairness according to tenant payment factor, minimum bandwidth guarantee to application and maximum utilization. Very little work has been done to address the user's Quality of Experience issues in virtualized environment, where multiple applications are hosted on single physical server with dynamic user demand. Existing mechanisms lack to provide dynamic and automatic tuning of network resources according to user demand factor.

2.5 Summary

This chapter presents the background and literature review of the thesis. It first discuss the traditional data centre technology and its issues.Then it highlights the

evolution of virtualization technology and its benefits over traditional computing infrastructures.Then network resource management approaches in server virtualization are described. .Following that, background information about network QoS is presented along with current QoS techniques available in virtualization products.This chapter also presents OpenFlow architecture in detail and also discuss different OpenFlow based QoS implementations. Last part of this chapter describe Quality of experience (QoE) and different proposed mechanisms to address the QoE issues.

Chapter 3

Problem Formulation and Design Goal

This chapter presents the problem definition in detail along with Design objectives.Problem is formulated by discussing different scenarios and use cases.

3.1 Problem Definition

As already discussed Quality of Experience (QoE) is based on complex performance metric. QoE depends on several subjective human components including user experience, emotions, and context. Measurement of these subjective components are time consuming, because it may differ from one user to another. However objective component i.e network QoS parameters can easily be managed for delivering Quality of Experience (QoE) to end user.Bandwidth is the main network resource, and its allocation factor has great impact on applications QoS parameters like delay, throughput and packet loss.

Nowadays biggest problem in current virtualized environment is that network resource assigned to application act as one gear bicycle, which means network resources provisioned to VMs/applications remain fixed. They cannot be shifted or re-allocated to next level according to workload or application demand. Thus it impacts the performance of network based applications. Just like multi-geared

bicycle, in which gear is changed according to complexity level, network resource allocation mechanism also must be multi-geared, which should be shifted according to application demand i.e during peak hours, more resources should be allocated so that end user can get improved performance. Studies [35] revealed that performance bottleneck usually occurs at edge network, because network core is usually deployed with more bandwidth, which rarely gets saturated. For virtual machines (VMs) edge network is the physical network interface of host, which usually creates network performance bottleneck.

Existing virtualization products offer proactive network resource management facility. Some proportion of the physical network is provisioned statically at the time of VM creation. Share of physical network is allocated according to advanced VM/application usage predication or application priority. This static network allocation strategy may not work efficiently due to unpredictable application user load. User load may not follow the pattern according to prediction.It may change due to different conditions like flash crowd, increase in some specific application service demand due to occurrence of some event etc.

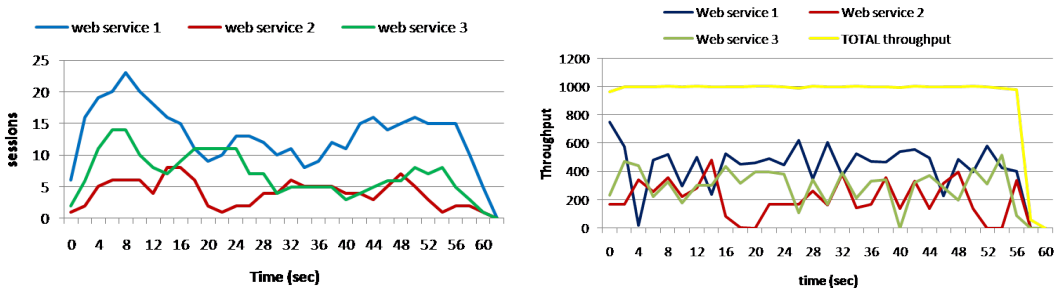
If application is allocated resources dynamically according to its fluctuating consumer demand factor i.e. maximum network resource provisioning to popular application and least resource allocation to least demanding application, then more number of users can receive Quality of Experience (QoE) as compared to static and default configuration.

Let suppose an academic institute is running multiple application services on independent VMs. These applications include official web site, student Learning Management System (LMS), and Email Service. However all of the VMs are hosted on single physical server having limited and shared network connection. Now by default network is shared between these VMs in best effort manner, which may result in bad quality of experience to end user. For instance traffic load on web site increases, whenever results are published on that web site. But user may face degradation in web service because enough bandwidth may not be available

to web service at that time. There may be a reason that few LMS portal users have taken up maximum portion of link bandwidth by downloading heavy files at that time .This is the one case where few users may become reason for Bad Quality of Experience (QoE) against most demanding service at that particular time. It is difficult to predict application usage pattern and network resource allocation in advance accordingly. Every application has fluctuating user load. Network resources can be allocated to applications on the basis of statistical traffic analysis but this method is still in-efficient because bandwidth is reserved statically for application.

Similarly, in most of the organizations scheduled backup job runs to perform backup of databases or other services. Usually backup job is not interactive and thus does not require any strict QoS. However mostly Backup jobs are TCP based i.e elastic, it can consume all the available bandwidth, which may cause degradation if some most demanding application runs at that time.

We performed a test in which three independent web servers were hosted on independent VMs. All three web servers were configured to host same size of data, but with different user load. First web server had traffic inter-arrival rate 500msec, second had 2 second and third had inter-arrival rate of 1 second. With this traffic pattern, first web server had more user demand because inter-arrival time between sessions was 500 msec. However only 496 users i.e 17% of first web server received better QoE out of 2793 users .It showed that rest of the users did not receive performance because of insufficient bandwidth availability. Network bandwidth is not fairly shared between these web server according to their user load. Figure 3.1 shows user load and bandwidth consumption of all web servers. Performance results in figure 3.2 clearly shows that bandwidth is not fairly used by web servers according to user load.



(a) User Sessions (b) Bandwidth Usage

Figure 3.1: User load and Application Throughput

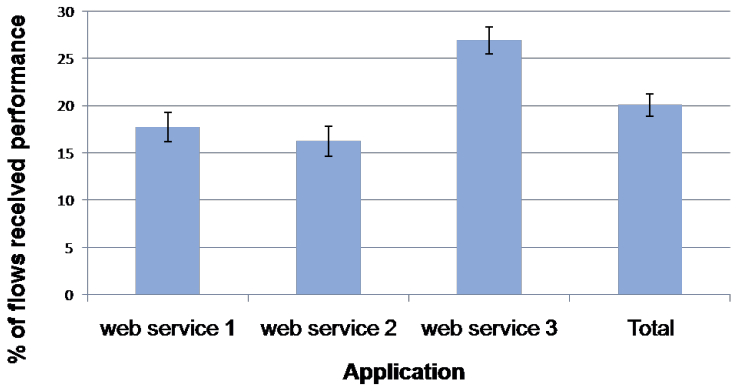


Figure 3.2: Performance Result

3.2 Design Objective

Our Objective is to *"dynamically provision network resources according to application popularity and demand; so that end users Quality of Experience (QoE) can be improved."*

Previous section describe the problem definition with the help of test case example. It concluded that popular or demanding application user may not receive better QoE just because of in-sufficient network resource allocation to application with respect to its user load.If application services are allowed to use network resources without any guarantee then it may result in unfair network resource consumption by applications with respect to its user load. If network resource is provisioned to application according to its popularity or demand factor then

improved end user performance can be achieved.

Therefore some mechanism is required to provide adaptive QoS according to user load, and without requiring manual and static configurations. Our proposed solution can handle these types of situations efficiently. It provides adaptive QoS by automatically provisioning more network resources to demanding applications and restricting bandwidth of low priority non-interactive applications. By efficient allocation of bandwidth we can assure that maximum possible network resource is provisioned to demanding application. It ultimately improves end user QoE.

3.3 Summary

This chapter introduces problem in current network resource allocation mechanisms. It discusses with the help of test example that best effort and proactive network resource allocation mechanisms are not efficient to ensure application performance. Different scenarios are discussed which highlights the problem in proactive network resource allocation approach. At the end design objective is presented for delivering improved QoE. Design objective states that network resource demand for popular applications should be fulfilled first, which means more network resources should be provisioned to popular applications and vice versa.

Chapter 4

Methodology

4.1 Adaptive QoS using OpenFlow

We propose OpenFlow based adaptive Quality of service (QoS) mechanism for virtualized environments where application services are consolidated on a single physical server and thus share the bottlenecked network resource. OpenFlow based solution is used due to its flexible architecture and central controller. Our adaptive QoS program efficiently and dynamically provision this scarce network resource to applications according to their user demand factor. Adaptive QoS results in improved end users perceived performance. Below we discuss the design methodology and working mechanism of our proposed solution.

4.1.1 Adaptive QoS architecture

Our proposed adaptive QoS architecture is shown in figure 4.1. Here we discuss different components of this architecture.

4.1.1.1 Application QoE requirement definition

Function of this component is to get the minimum bandwidth requirements for individual application flow based on the fact that every network based application service has some minimum unconstrained throughput requirement for delivering

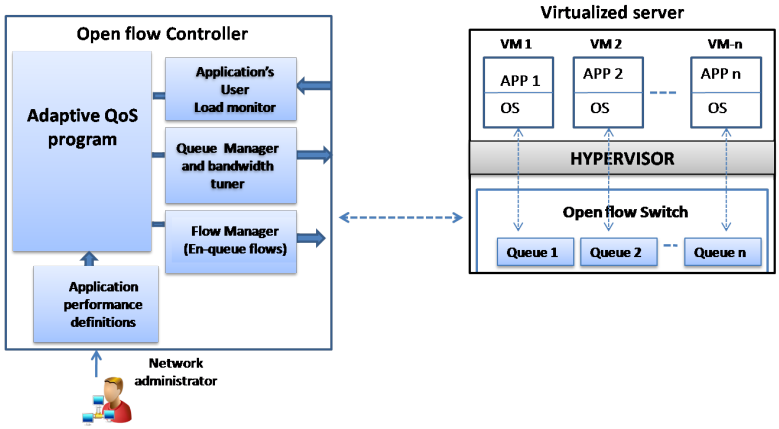


Figure 4.1: Test bed Architecture

optimal performance to end user. These requirements can be either given by application or can be calculated according to some defined Quality of Experience (QoE) settings. Real-time and interactive services have usually fixed bandwidth requirements. For instance, unconstrained throughput requirement of highest quality Netflix streaming movie is about 5 Mb/s, [36]. Bandwidth requirement for voice and video services can be calculated based on codec type. Similarly given an average query page of 20 KB and an average required query response time of 0.25 s [37], the unconstrained throughput for a Google search is about 600 kb/s, or just over 1/10 of Netflix. TCP based video streaming bandwidth requirements like YouTube, can also be calculated, which depends on resolution and end user player [38]. Different tools and techniques are also available to find out the minimum bandwidth requirement for individual sessions. Microsoft has released calculators for calculating bandwidth requirement according to number of users, for office365, Outlook web Access application, Remote Desktop application, VDI clients etc.

This component requires manual input from outside. Network administrator manually adds per session bandwidth requirement for all application services running on virtual machines (VMs). Priority level of all application services are specified by administrator. Priority level can be assigned in a range between (0-3). Low priority application service is assigned value 0, normal priority is represented by 1 and high priority application service is assigned value 2. Per session bandwidth

requirement of only normal priority application is specified. Low priority and High Priority applications are not assigned per session bandwidth. They are treated as best effort services. More proportion of the bandwidth is reserved for high priority application queue and least portion of bandwidth capacity is reserved for low priority application's queue. By this way applications can get different proportion of bandwidth resource according to their priorities.

4.1.1.2 Queue Manager and Bandwidth Tuner

This component manages queues and their bandwidth parameters. Linux Hierarchical Token Buckets (HTB)[39] based QoS queues are used. Every new application service is assigned a separate queue for application's performance isolation. HTB queues require two bandwidth configuration parameters. First parameter is the minimum rate i.e. minimum guaranteed bandwidth settings of the queue. Second parameter is maximum rate i.e. maximum allowed bandwidth. If maximum rate is specified then queue's bandwidth is limited to that value even excess bandwidth is available. Therefore in our proposed solution maximum rate is set to full link capacity, so that network resource utilization can be maximized. For application services with priority level normal, minimum guaranteed bandwidth is provisioned dynamically according to application user load. On the other hand if traffic with high priority is detected, then maximum portion of the bandwidth is guaranteed to that application irrespective of number of sessions. Maximum assignable bandwidth value is specified by administrator. If low priority application traffic is detected, then application queue is restricted to some minimum value irrespective of number of sessions.

Although OpenFlow protocol itself does not support queue bandwidth tuning facility because queue creation and its bandwidth configuration is a switch specific function. But open networking foundation has provided an alternate auxiliary protocol i.e. OF-CONFIG [40] for OpenFlow capable switch configuration function.With the help of this protocol different switch specific features including

QoS, minimum and maximum bandwidth guarantees can be configured. Open vSwitch queue bandwidth parameters can be configured using ovs-vsctl utility.

4.1.1.3 User load monitor

This component monitors per application user sessions at fixed time interval. It gets real time statistics from OpenFlow switch. It also ranks the application services on the basis of popularity. Application service which has more user requests is considered as more demanding and popular. Queue manager allocates guaranteed bandwidth to application according to application demand. Available network resource is allocated first of all to highest demanding application service according to its per session bandwidth demand and so on. Applications user load is highly fluctuating and may vary at different time intervals. Therefore application ranking also changes at different time intervals.

4.1.1.4 Flow Manager

This component is responsible for adding new application flows and forward the application flows to its associated queue, by using OpenFlows en-queue action.

4.2 Working mechanism

Adaptive QoS program, works by taking application bandwidth requirements as an input from administrator. It automatically allocates independent QoS queues to every application based on administrator provided information. Minimum bandwidth of the queue is initially set to some proportion of link bandwidth.This proportion is already defined by administrator for every application service, so that bandwidth starvation of that application service can be avoided. High priority applications are assigned large proportion and low priority service queues are allocated small proportion. When users start to access application services from virtualized server, our adaptive QoS program comes into an action. It gets the

user ranking information form user load monitor module. On the basis of priority and popularity factor it provision the bandwidth for application queue by configuring minimum guaranteed bandwidth rate on queue. Bandwidth is provisioned on the basis of application priority. Network resources are first provisioned to high priority application queue. For normal priority applications bandwidth is first provisioned to application queue with highest number of user session. Bandwidth is allocated to normal priority application queue according to the factor of its number of users and per session minimum required bandwidth. Similarly remaining bandwidth is provisioned to next application queue, which has second rank of popularity and so on. In last low priority application is assigned least proportion of bandwidth. Queue manager performs bandwidth management functions. If application's bandwidth requirement is more than link capacity due to increased user load, then bandwidth guarantee to individual flow will not be assured due to link capacity limitation. However performance maximization will be guaranteed by allocating maximum portion of network to demanding application. Thus more number of user can receive improved performance as compared to default system. In next section algorithm and flow chart shows the working of adaptive QoS program.

4.2.1 Algorithm flow chart

4.2.2 Summary

In this chapter methodology and working mechanism of adaptive QoS solution is discussed. It first presents the architecture of our adaptive QoS solution, which comprises of different components. Then it describe the functionality of each component in detail. In last part of the chapter flow chart is presented which shows working mechanism of adaptive QoS algorithm.

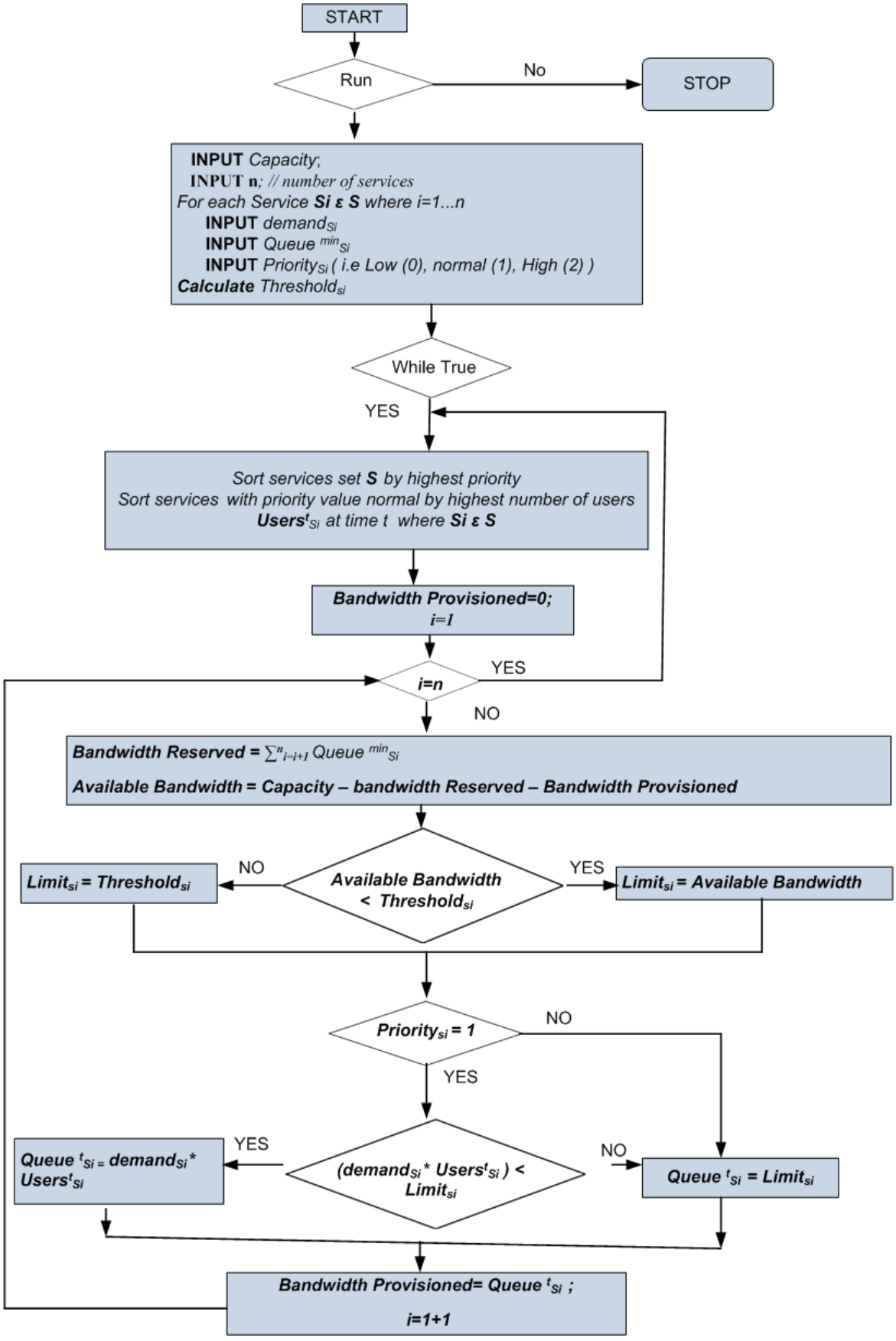


Figure 4.2: Adaptive QoS Working mechanism

Chapter 5

Implementation and Results

In this chapter Implementation details are presented. Section 5.1 describe the test bed setup with details of all the hardware and software components.Next section 5.2 and 5.4 provides detail about traffic generation and data sample collection methodology.In last section 5.5 different simulation scenarios and their performance results are discussed.

5.1 Testbed setup

We implemented test bed setup by using following components

Hardware: Sun X4150 dual processor Quad core, Intel Xeon x5460, 3.14GHz hardware is used.

Virtualization Hypervisor: Kernel based Virtual machine (KVM)[41] module was installed on Centos 6.3 Linux Operating system, because it turns Linux operating system into a hypervisor.

Switch: Open vSwitch[42] version 1.7 was installed and attached with KVM hypervisor .Open Vswitch is an open source software switch and supports OpenFlow protocol. It is typically used with hypervisors and it provides connectivity between virtual machines within the host and outside the host. It provides several features including Quality of Service (QoS).

OpenFlow controller: OVS-controller is used as an OpenFlow controller. It

is a default OpenFlow controller which comes with Open vSwitch package.

Adaptive QoS program: It is written in Bash shell scripting language along with following following command line utilities.ovs-ofctl tool is used for managing flow entries in OpenFlow switch.different applications flows are forwarded to different queues using this tool. Queue usage statistics are alos gathered using this tool. ovs-dpctl tool is used to collect application user statistics at run time.ovs-vsctl tool is another command line tool for managing Open vSwitch. We used this tool for dynamic configuration and bandwidth management of queues.

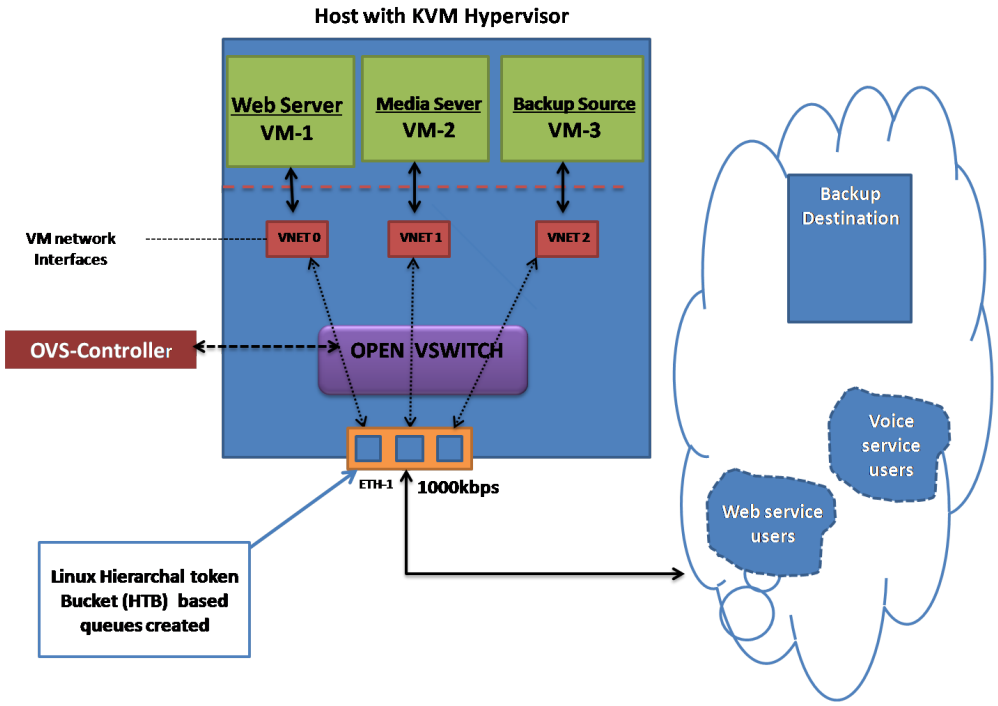


Figure 5.1: Testbed Architecture

5.2 Traffic simulation

In test bed setup as shown in figure 5.1, we installed three independent Virtual machines with Linux operating system. Apache Web server was installed and configured on first VM and sample website was hosted on that VM. Second VM was also installed with Linux OS and it was configured to act as media server hosting voice media files. Third VM was setup as a source of transferring heavy file

to another outside destination host. Bandwidth of physical network is restricted to only 1000kbps so that link capacity can be fully choked by generating traffic on smaller scale.Three independent HTB queues are created for individual VM's traffic.

Real-time application traffic was generated from outside KVM host using different tools to access application services running on these three VMs.

httperf [43]tool was installed on a separate physical host. It was used to generate web request towards web server hosted on virtual machine. Inter-arrival rate of web requests was set to random and exponentially distributed.

Distributed Internet Traffic generator (D-ITG) [44] tool was installed on another independent physical host and voice traffic with G.711.2 codec was generated toward media server virtual machine. Inter-arrival rate of voice sessions was set as random and exponentially distributed, and duration of each voice session was uniformly distributed.

As third Virtual machine was setup to transfer heavy file. Therefore Single TCP session was generated from third VM using D-ITG tool to transfer 100MB of data to destination outside KVM box. Inter-departure rate of packets was set to constant rate of 1000 packets/sec and packet size was uniformly distributed between 500-1000 bytes.

5.3 Application Performance criteria

Every application service must meet some minimum performance criteria. Providing performance beyond that minimum limit is considered as service degradation and ultimately results in bad quality of experience by end user. Therefore following performance criteria of individual flow is defined.

http flow is set as an error if http response time is greater than 1 second. If http response time is less than or equal to 1 second, then that flow is considered to meet the desired QoS. For individual voice session, less than 2 % packet loss is

considered as minimum QoS requirement. Backup job traffic is left as best effort delivery job because usually schedule backup does not require fix time limits to complete.

5.4 Data Collection

Data for result is collected by performing each test for the duration of 60 seconds. Two separate types of tests were performed for each simulation scenario. First the scenario was tested with default configurations i.e without applying any QoS. Then same scenario was tested by running our adaptive QoS program. In order to validate the results each test was performed more than 50 times and 95% confidence interval of performance result is also calculated. Performance results were generated to show percentage of application flows, which have met out performance criteria.

5.5 Simulation scenarios and result discussion

Four different scenarios were created and tested to evaluate our adaptive QoS solution. In web traffic simulations web response size is usually random and follows Pareto distribution. However for simplicity in our simulation scenario we assumed that single web page of size 40KB is hosted on web server and therefore web response size is always 40KB. On the basis response size, minimum bandwidth requirement for single web session is set as 320kbps so that it can be loaded in 1second without any network constraint. 40 KB web object size is chosen because this is an optimal size which can choke configured network capacity. Setting web object with less size cannot fully utilize link capacity and therefore adaptive QoS scenarios can not be properly tested. Voice session is using G.711.2 codec and therefore requires approximately 88kbps fixed bandwidth for providing desired performance. Backup job is considered as best effort; therefore no minimum bandwidth requirements are defined.

Four simulation scenarios were created to evaluate our adaptive QoS program with maximum possible traffic trends. In test-1 elastic traffic i. e TCP based backup traffic is introduced along with web and voice sessions. In second test case we introduced more voice traffic as compared to http. In third test case we injected more web traffic as compared to voice. In test-4 three independent web services were hosted and inter-arrival rate of web request were kept different on each web server. So by this way we have tried all of the possible combinations. Below table 5.1 shows the test scenarios with having different inter-arrival rates of traffic.

TEST No.	**Application Traffic**	**Mean Inter-arrival time**	**Performance requirement for QoE**
TEST-1	WEB SERVICE	1 second	Response time <=1 second
	Voice	3 second	Packet loss <2 %
	Backup	100MB transfer	Best effort
TEST-2	WEB SERVICE	3 second	Response time <=1 second
	VOICE	1 second	Packet loss <2 %
TEST-3	WEB	0.5 second	Response time <=1 second
	VOICE	2 second	Response time <=1 second
TEST-4	WEB SERVICE -1	0.5 second	response time <=1 second
	WEB SERVICE -2	2 second	Response time <=1 second
	WEB SERVICE -3	1 second	Response time <=1 second

Table 5.1: Simulation Scenarios

5.5.1 Test 1:Web and Voice traffic along with single TCP session

In test 1 scenario web requests, voice sessions and backup traffic was generated .Mean inter-arrival time of web request was set to 1 seconds. Mean inter-arrival time of voice session was configured as 3 seconds. Backup session was generated from VM-3 to transfer 100MB data to outside host. Session inter-arrival rate is shown in figure 5.2. Initially Test was performed with default settings i.e without applying any QoS and then same case was tested by applying our adaptive QoS program on controller. In our adaptive QoS program, per web

flow 320kbps bandwidth is allocated because it was the minimum unconstrained throughput requirement for web session, to full fill required performance criteria. voice session's minimum bandwidth requirement was set as 100 kbps. Minimum guaranteed bandwidth for WEB, and voice traffic queue is set to 200kbps, in order to avoid starvation of these services at time of congestion. However minimum guaranteed bandwidth limit for backup queue is set to 100kbps, as it is not a priority job. Figure 5.3 shows the bandwidth usage graph with default settings and with adaptive QoS. Figure 5.3 (a) shows that backup traffic consumed much more bandwidth capacity because it is TCP based elastic type of traffic and consume maximum available bandwidth. Therefore it impacted the performance of other critical traffic i.e web and voice sessions.Only 26.72% web flows and 46.83% web flows received performance But when adaptive QoS is applied, bandwidth of low priority backup traffic was automatically restricted to 100kbps as shown in figure 5.3 (b), and more bandwidth was allocated to web and voice traffic because they had more user sessions. By dynamically allocating more network resources to web and voice queues, more number of web and voice flows received performance according to our defined performance criteria. Performance results are based on 95% confidence interval. Figure 5.4 shows that overall higher number of Web and voice flows received QoS, when applied Adaptive QoS algorithm.

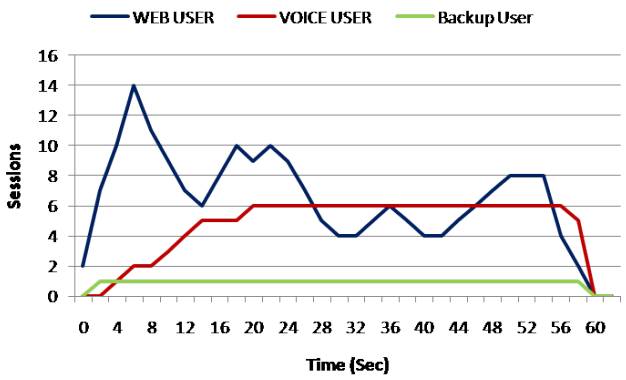
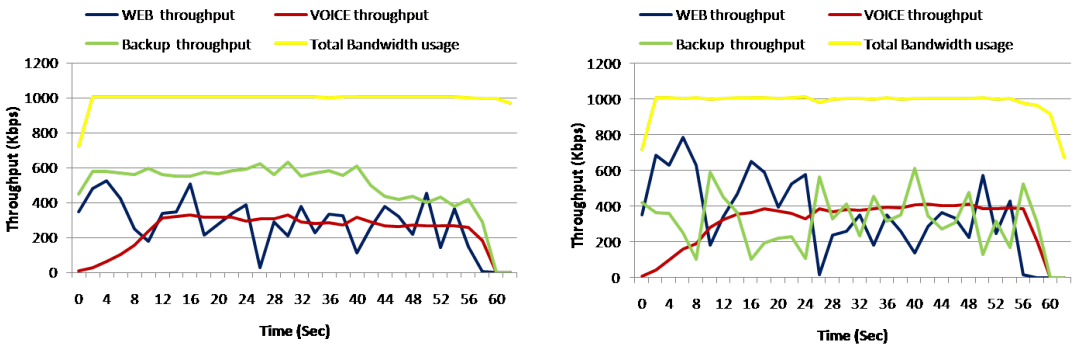


Figure 5.2: Test 1 sessions



(a) No QoS (b) Adaptive QoS

Figure 5.3: Test 1 Bandwidth usage plot

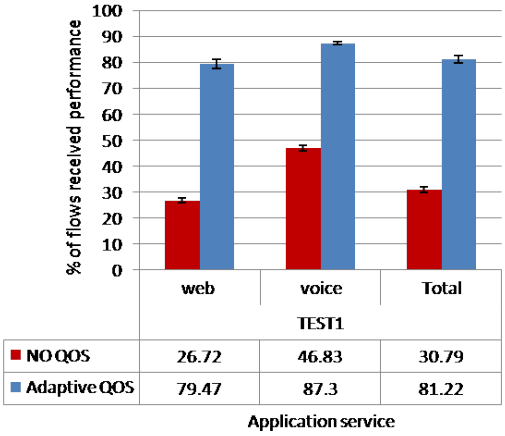


Figure 5.4: Test 1 performance Result

5.5.2 Test 2:Web and voice traffic with voice service popularity

In Test 2 scenario only Web and Voice traffic was generated with mean inter-arrival time of 3 seconds and 1 second respectively. Figure 5.5 shows that during test duration voice session remained high in demand. Without running our adaptive QoS program, only 15% voice sessions received performance. However by applying our adaptive QoS mechanism more demanding service i.e voice service was allocated more bandwidth at runtime. Therefore approximately 39% of flows received performance as shown in figure 5.7, which is significant improvement. Bandwidth usage plot in figure 5.6 (b), clearly shows that more number of voice

flows received performance just because more network resources are provisioned to voice application queue with adaptive QoS program as compared to default scenario shown in figure 5.6 a),where comparatively low demanding web service consumed more bandwidth.

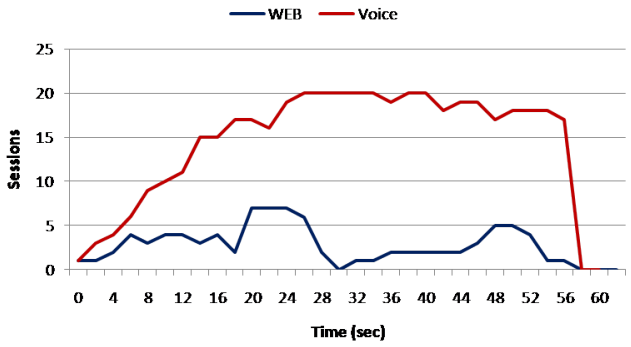
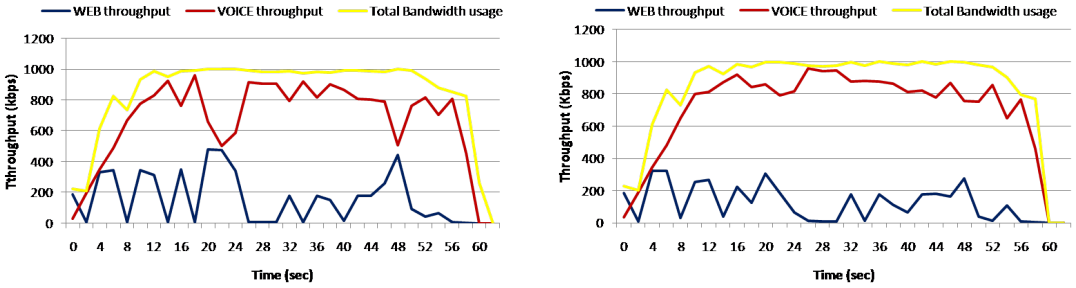


Figure 5.5: Test 2 sessions



(a) No QoS

(b) Adaptive QoS

Figure 5.6: Test 2 Bandwidth usage plot

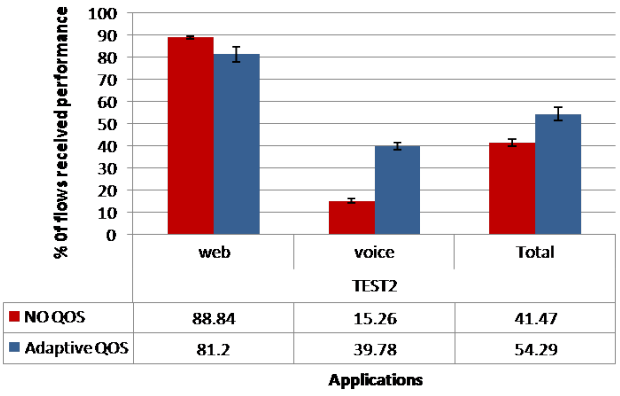


Figure 5.7: Test 2 performance Result

5.5.3 Test 3:Web and voice traffic with web service popularity

Test 3 was performed by generating web and voice flows, but this time mean inter-arrival rate for web traffic was set to 500 msec and for voice traffic 2 seconds. In this test scenario overall demand for web sessions remained high as shown in figure 5.8. Result shows that by applying adaptive Qos algorithm, 52% of Web traffic flows met performance criteria as compared to default configurations where only 31% of web flows received performance. Bandwidth usage plots with both type of tests are shown in figure 5.9. In both plots link was fully utilized. Throughput of the Web traffic remained high because of more number of web sessions. Plot also shows that by applying QoS, minimum bandwidth guarantee of voice service was switched to 200kbps to avoid that service from starvation, and rest of bandwidth was assured to Web service because web service had more user sessions. At time interval 20, user sessions of voice and web became equal, and then voice service has allocated more resources because of its priority over web service. Performance result of test 3 in figure 5.10 that our adaptive QoS performs better and deliver improved performance to maximum users.

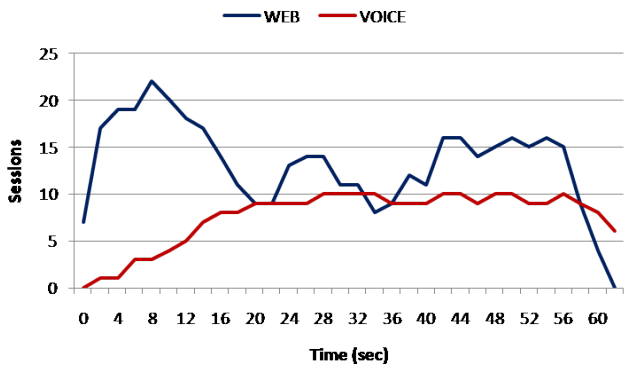
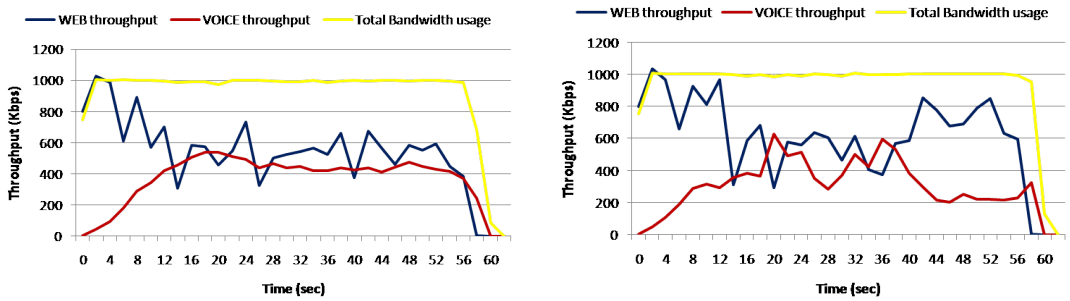


Figure 5.8: Test 3 sessions



(a) No QoS (b) Adaptive QoS

Figure 5.9: Test 3 Bandwidth usage plot

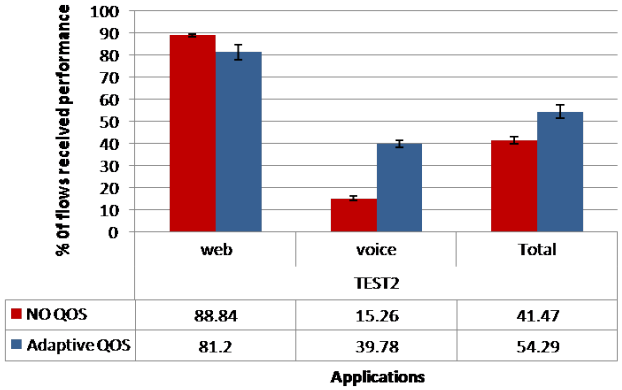


Figure 5.10: Test 3 performance Result

5.5.4 Test 4:Multiple Web servers with different popularity

In test-4, three web servers were configured on independent virtual machines. Each web server hosted a web page of 40KB. Random web requests were generated to access the web pages from these web servers. HTTP requests were generated towards web server 1, with exponentially distributed inter-arrival time and mean value of 500msec. On second web server random requests were generated with same distribution and mean inter-arrival time of 2 seconds. On third server mean inter-arrival time of request was set to 1 second. Overall sessions for web server 1 remained high throughout the test duration because its mean inter-arrival time was 500 msec. similarly web server 2 had lowest number of sessions, because mean inter-arrival time was 2 seconds. Session rate is shown in figure 5.11. Results in

figure 5.13 showed that without running our adaptive QoS program, less number of web server-1 flows received performance despite of having more popularity. Same test was performed again with our adaptive QoS program, and result showed improvement. Most demanding application i.e web server -1 received more network resources and therefore more number of flows received desired performance. Significant improvement has been observed in overall performance result in adaptive QoS test. With adaptive QoS program 30 % of total flows received performance as compared to default configuration settings, where overall only 20% flows received performance. Bandwidth usage plots in figure 5.12 depict that improvement in result was achieved by allocating more bandwidth to demanding application service and restricting bandwidth of low demanding application service.

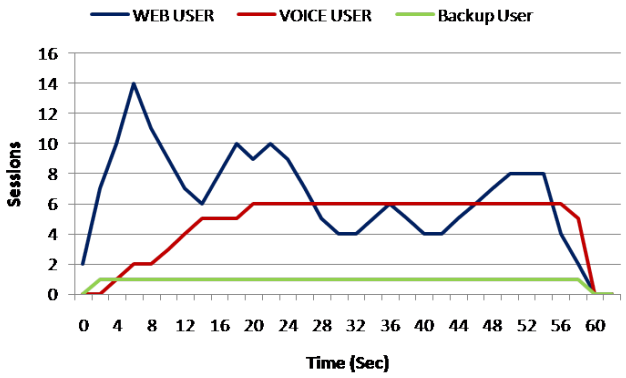
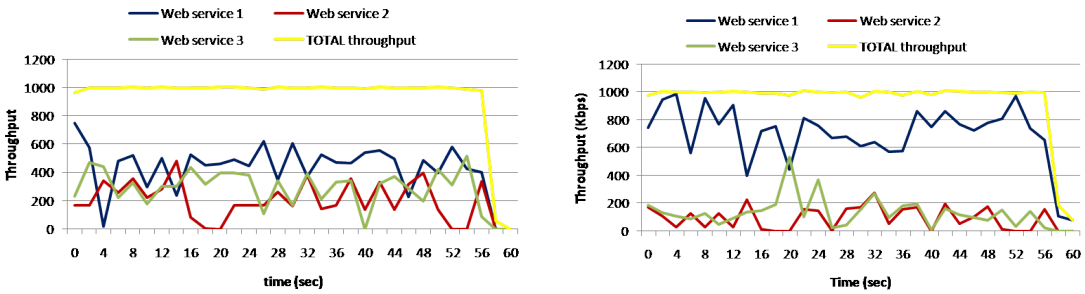


Figure 5.11: Test 4 sessions



(a) No QoS

(b) Adaptive QoS

Figure 5.12: Test 4 Bandwidth usage plot

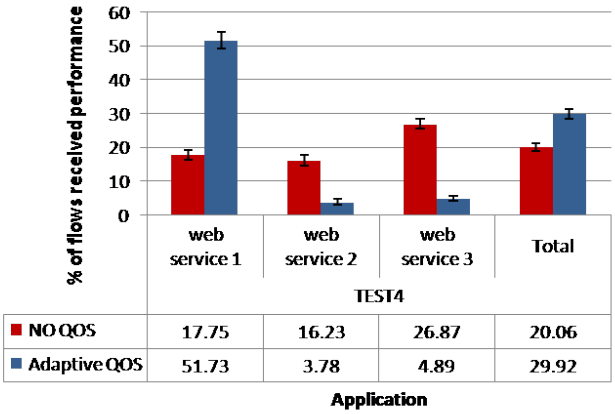


Figure 5.13: Test 4 performance Result

5.6 Summary

In this chapter Test bed setup and implementation details are discussed. KVM based virtualization plateform and Open vSwitch is used in test bed. OVS-controller is used as a central controller for managing OpenFlow enabled Open Vswitch. Traffic is generated with httperf and D-ITG tools. Each simulation test is performed multiple times and performance results are calculated using confidence interval. Simulation result shows that adaptive QoS mechanism provides much better QoE to end user as compared to default best effort network setup.

Chapter 6

Conclusion & Future Work

6.1 Conclusion

We presented one of the use cases of OpenFlow technology which provides an adaptive QoS according to application popularity. Network resources are provisioned to applications according to their demand factor. More bandwidth is reserved for more demanding application and vice versa. Application priority parameter further improve performance by confining the bandwidth of low priority and best effort application services like scheduled backups e.t.c. Similarly it can allocate maximum bandwidth to application queue, if it detects high priority traffic like VM migration traffic from one physical server to another. Our proposed solution is scalable, because OpenFlow based solution is adopted. OpenFlow's flexible architecture with central controller facility allow us to extend our solution to manage and control network traffic from multiple servers.

We have evaluated our adaptive QoS algorithm by performing multiple tests on different patterns of random traffic. Analysis of tests revealed that dynamic provisioning of bandwidth according to application user load results in improved performance. By applying our adaptive QoS solution more number of demanding application's flows receive desired QoS.

Our proposed mechanism is useful in organizations, where services are hosted

on virtual infrastructure, and share bottlenecked network resource. Network administrator has just to define priority and per session bandwidth requirements of all hosted applications according to their application QoE targets. Our adaptive QoS program will then automatically adjust network resources to meet that those performance targets.

6.2 Future Work

Our proposed solution is currently tested on a single switched environment, but it can be extended to Cloud based data centres where VMs are connected through multiple switches. In our proposed work some proportion of the bandwidth is reserved for least demanding services, so that starvation of those services can be avoided. Currently this proportion is defined manually by network administrator. However in future work optimal value will be automatically calculated on the basis of application priority and usage factor for further improvement in results.

Currently only packet loss parameter is considered for delay sensitive and real time applications like voice and video services. Delay parameter is not considered because we used Hierarchal token Bucket (HTB) queuing mechanism in our work, which does not provide latency control. However Linux Hierarchal Fair Service Curve Scheduler (HFSC) is capable to provide prioritized services to delay sensitive traffic, but currently Open vSwitch does not provide direct facility to configure latency parameters for HFSC queues. Therefore latency issue will be catered, once Open vSwitch release updates.

End user Quality of experience (QoE) is currently improved by tuning application's bandwidth parameter only. However in future work more optimized approach will be developed in which OpenFlow controller will keep track of different other parameters at user end.Bandwidth will be provisioned in more optimized way by considering end user device type and supported functionalities like codec etc in client application.

Bibliography

[1] "Idc," 2013. [Online]. Available: https://www.idc.com

[2] "Cisco global cloud index." Cisco, 2013. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf

[3] "Vmware," Jun. 2009. [Online]. Available: http://wwww.vmware.com

[4] "Xen," 2014. [Online]. Available: http://www.citrix.com

[5] "Hyper-V," 2014. [Online]. Available: http://www.microsoft.com/en-us/servercloud/solutions/virtualization.aspx

[6] J. Wroclawski, "The use of rsvp with ietf integrated services," 1997.

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

[8] "vsphere network i/o control." [Online]. Available: http://pubs.vmware.com/vsphere-51/topic/

[9] O. Masad, "Network qos," 2014. [Online]. Available: http://www.ovirt.org/Features/Network_QoS

[10] T. Voith, K. Oberle, and M. Stein, "Quality of service provisioning for distributed data center inter-connectivity enabled by network virtualization," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 554–562, 2012.

[11] S. Radhakrishnan, R. Pan, A. Vahdat, G. Varghese *et al.*, "Netshare and stochastic netshare: predictable bandwidth allocation for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 3, pp. 5–11, 2012.

[12] A. Shieh, S. Kandula, A. Greenberg, and C. Kim, "Seawall: performance isolation for cloud datacenter networks," pp. 1–1, 2010.

[13] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," p. 15, 2010.

[14] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," *USENIX WIOV*, 2011.

[15] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 242–253.

[16] J. Guo, F. Liu, H. Tang, Y. Lian, H. Jin, and J. Lui, "Falloc: Fair network bandwidth allocation in iaas datacenters via a bargaining game approach," pp. 1–9, 2013.

[17] L. Chen, Y. Feng, B. Li, and B. Li, "Towards performance-centric fairness in datacenter networks."

[18] "Software defined networking a new norm for networks." [Online]. Available: https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf

[19] 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.

[20] O. networking foundation, "openflow switch specification version 1.3," 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf

[21] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[22] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, "Automated and scalable qos control for network convergence," *Proc. INM/WREN*, vol. 10, pp. 1–1, 2010.

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

[24] J. Martin and N. Feamster, "User-driven dynamic traffic prioritization for home networks," pp. 19–24, 2012.

[25] P. Casas and R. Schatz, "Quality of experience in cloud services: Survey and measurements," *Computer Networks*, 2014.

[26] Z. Zheng, X. Wu, Y. Zhang, M. R. Lyu, and J. Wang, "Qos ranking prediction for cloud services," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1213–1222, 2013.

[27] H. Kumar, H. H. Gharakheili, and V. Sivaraman, "User control of quality of experience in home networks using sdn."

[28] T. Yamazaki and T. Miyoshi, "Resource allocation method based on qoe for multiple user types," pp. 13–18, 2014.

[29] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards

network-wide qoe fairness using openflow-assisted adaptive video streaming," pp. 15–20, 2013.

[30] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over http dataset," pp. 89–94, 2012.

[31] P. Casas, M. Seufert, S. Egger, and R. Schatz, "Quality of experience in remote virtual desktop services," pp. 1352–1357, 2013.

[32] K. Li, W. Guo, W. Zhang, Y. Wen, C. Li, and W. Hu, "Qoe-based bandwidth allocation with sdn in ftth networks," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014, pp. 1–8.

[33] H. H. Gharakheili, J. Bass, L. Exton, and V. Sivaraman, "Personalizing the home network experience using cloud-based sdn."

[34] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory networking: An api for application control of sdns," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 327–338.

[35] I. Society, "Bandwidth management," 2012.

[36] K. Florance, "Netflix technology blog," 2011. [Online]. Available: http://techblog.netflix.com2011/01/netflix-performance-on-top-ip-networks.html

[37] Google, "Google mountain view CA,USA,," 2012. [Online]. Available: http://www.google.com/corporate/tech.html

[38] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," p. 25, 2011.

[39] "Hierarchal token bucket (htb)," 2014. [Online]. Available: http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm

[40] O. N. Foundation, "Open flow config," 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf

[41] "Kvm," 2014. [Online]. Available: http://www.linux-kvm.org

[42] "Openvswitch." [Online]. Available: http://openvswitch.org

[43] D. Mosberger and T. Jin, "httperf a tool for measuring web server performance," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 3, pp. 31–37, 1998.

[44] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.