

CHALMERS



BITS-Pilani



Investigations into the Design and Implementation of an IPv6-QoS-Aware Grid

Piyush Gupta

Master's Thesis

International Masters Program in

Digital Communications Systems and Technology

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden

Research Advisor

Dr. Rahul Banerjee

Associate Professor: CS & IS Group

*Birla Institute of Technology & Science, Pilani
Rajasthan, India*

Certificate

It is to certify that Mr. Piyush Gupta, a visiting student from Chalmers University (Sweden) has carried out the presented research work at the Centre for Software Development of the Birla Institute of Technology & Science, Pilani under my supervision.

The work done by him was a part of a long -term research project called “Project Grid -One” (<http://discovery.bits-pilani.ac.in/Grid-One/index.htm>) and as part of this work he primarily tried to establish a limited-capability IP test-bed which can be extended to work as a full-fledged IPv6 QoS -aware facility atop which a Grid can be realized. Due to lack of ready support for IPv6 Flow -Label specification (in compliance with the RFC 3697) in several commercial operating systems, he had to li mit part of his work to whatever was achievable under laboratory conditions with simulated WAN within the Intranet itself. In addition, due to the unavailability of native IPv6-QoS enabled WAN setup, tunneling had to be used for the purpose of testing. This, as can be easily predicted, has consequently resulted in partial skew in performance of the IPv6 QoS enabled test-bed in some of the presented cases which must not be misconstrued as the true performance of such a test -bed in a native IPv6 WAN scenario.

The referred scholar has done this work as part of the academic requirements of his Master’s degree programme under official agreement between the Chalmers and BITS.

BITS-Pilani
October 12, 2005

(Rahul Banerjee)
Research Supervisor

Abstract

Grids are distributed systems that differ from other such systems by their geographical scale and focus on increasing utilization of existing computing resources to achieve system functionality. Today, large-scale grids predominantly use the public Internet—an inter-network based on TCP/IPv4 protocol suite—to connect sites that share their computing resources in a grid environment. Some one-way video grids applications, like remote control of instruments, and remote surgery, require certain performance level from the network—in other words, certain quality of the network service—to meet their requirements. Diffserv architecture is the scalable architecture, which is standardized by IETF to deploy various network services on inter-networks based on TCP/IP protocol suite. IPv6 is the latest version of most important network layer protocol in TCP/IP suite. Flow label is a 20-bit field in IPv6 base header. Its purpose is to inform routers that traffic identified by the flow label needs special treatment.

In this work, IPv6-QoS enabled VPN—IP-in-IPv6 encapsulation with encryption and an elevated quality of network service using diffserv architecture and IPv6 flow label—is proposed. Such a VPN connection will be useful to connect sites that want to participate privately in a large-scale multi-media grid. Immutable property of flow label field makes it ideal for carrying site's quality of service requirement to all intermediate diffserv domains along the path to the destination. IPv6-QoS VPN scheme described above could not be implemented in the laboratory because of lack of support for flow label related features on commodity operating systems, and time constraints in performing this work. However, to support interactive one-way video grid applications, a network service—based on differentiated services architecture—is also required. Such a service has been implemented and tested using IPv4. One interesting difference between this IPv4-based service's performance and a similar IPv6-based service's performance in the laboratory environment is the minor increase in one-way delay possibly due to larger size of IPv6 base header as compared to IPv4 header. In a real-life scenario involving a native IPv6-QoS-enabled WAN, tunneling overheads would not exist and links might require to carry a hybrid traffic through larger number of WAN routers in between sources and destinations; and, it is under such situations when the IPv6 header design will actually work to qualitative and quantitative advantage.

In the laboratory, an IPv4/IPv6 network is setup using commodity hardware and software. This setup is part of a future grid, which among other things, will support interactive applications like tele-control of devices, and tele-surgery. This setup will serve as a test-bed for further development and experimentation. On this network, a diffserv domain has been setup. In this diffserv domain, network control, diagnosis, and management traffic is transmitted with highest priority. Real-time traffic of one-way video applications is transmitted with second highest priority. All other traffic is transmitted with lowest priority. The network service offered by this domain to real-time traffic controls the maximum delay experienced by any real-time service packet on every per-hop in the domain. Real-time traffic experiences zero packet loss in the laboratory environment. Jitter is controlled at every hop in the domain by shaping the real-time traffic.



Preface

This master's thesis work is part of a multi-year project at Birla Institute of Technology and Science (BITS), Pilani, India, namely, *Project Grid-One: an IPv6 QoS Aware Grid*. One of the aims of this project is to setup a grid over production and laboratory IPv4/IPv6 network. This grid setup will, among other things, act as a test-bed for experimentation. Currently, Microsoft Research partially funds this project.

Excellent infrastructure and excellent teaching at Chalmers University, the calm and serene life of Gothenburg, and generosity of Swedish government have spoilt me. I am indebted to the Swedish government and the Chalmers University for providing me an opportunity to study in Sweden. I am indebted to the BITS University for providing me an opportunity to work in the university, and providing accommodation in the campus. Life in Pilani has been a great learning experience. Adjusting to the new people and the new environment was challenging.

Some people have helped me immensely in finishing the thesis work. I would like to express my gratitude to Prof. Rahul Banerjee (Computer Science & Information Systems Group, BITS) for granting me permission to work in IBM Lab at BITS University, and his patience. I would like to thank Amit Vyas (MECS student, BITS) and Savan Gupta (MESS student, BITS) for their help in setting up the network in the laboratory and installing the measurement tools. Last, but not the least, I would like to thank Prof. Sven Tafvelin (Computer Engineering Department, Chalmers University) for finding time in his busy schedule to examine the work.

Acronyms and Abbreviations

ATM	Asynchronous Transfer Mode
BB	Bandwidth Broker
Diffserv	Differentiated Services Architecture
DSCP	Differentiated Services Code Point
DS	Differentiated Services
DV	Digital Video (a multimedia standard)
DVTS	Digital Video Transport System
FC3	Fedora Core 3
FCFS	First Come First Serve
FPS	Frames per Second
IETF	Internet Engineering Task Force
Intserv	Integrated Services Architecture
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
MPEG	Motion Picture Experts Group
MPLS	Multi-protocol Label Switching
MTU	Maximum Transfer Unit
NCMIR	National Center for Microscopy and Imaging Research
NIC	Network Interface Card
NTP	Network Time Protocol
NTSC	Never Twice Same Color
PHB	Per-hop Behavior
QoS	Quality of Service
RED	Random Early Detection
RFC	Request for Comments
RSVP	Resource Reservation Protocol
RTP	Real-time Protocol
SLA	Service Level Agreement
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UHVEM	Ultra-High Voltage Electron Microscope
VBR	Variable Bit Rate
VPN	Virtual Private Network
WFQ	Weighted Fair Queuing

Contents

1. INTRODUCTION.....	1
1.1 WHAT IS A GRID?.....	1
1.2 WHAT IS QUALITY OF SERVICE?.....	2
1.3 GRIDS, TCP/IPV6 AND QUALITY OF SERVICE.....	3
1.4 EXAMPLE OF A GRID APPLICATION: ONE-WAY VIDEO OVER IPV6 INTER-NETWORK.....	4
1.5 SCOPE AND REPORT ORGANIZATION.....	6
2. NETWORK QOS AND IPV6-QOS FEATURES	7
2.1 NETWORK QOS PARAMETERS.....	7
2.2 IPV4/IPV6 QOS ARCHITECTURES.....	8
2.2.1 <i>Integrated Services Architecture</i>	8
2.2.2 <i>Differentiated Services Architecture</i>	8
2.2.3 <i>Intserv-over-Diffserv Architecture</i>	9
2.3 IPV6-ONLY QOS FEATURE (FLOW LABEL).....	9
2.4 RELEVANCE OF SUB-IP LAYER TECHNOLOGIES.....	10
2.5 DISCUSSION	10
3. IPV6-QOS SCHEME FOR PRIVATE ONE-WAY VIDEO	11
3.1 HOW DO DIFFSERV ARCHITECTURE AND IPV6 FLOW LABEL FIT IN?.....	11
3.2 USE CASE FOR DIFFSERV AND FLOW LABEL-IPV6-QOS ENABLED VPN.....	12
4. NETWORK SERVICE DESIGN FOR SUPPORTING ONE-WAY VIDEO	14
4.1 SERVICES REQUIRED.....	14
4.2 FLOW LABEL, DSCP AND PHB ID FOR REAL-TIME SERVICE	15
4.3 PACKET SCHEDULING ALGORITHM AT DIFFSERV ROUTER.....	15
4.3.1 <i>How to control maximum delay</i>	15
4.3.2 <i>Choice of Packet Scheduling Scheme</i>	16
4.4 TRAFFIC PROFILE OF AGGREGATE FLOWS.....	17
4.5 TRAFFIC CONDITIONING AT DIFFSERV ROUTER.....	18
4.6 DOMAIN-LEVEL PROBLEMS.....	18
5. IMPLEMENTATION SCOPE, NETWORK SETUP AND MEASUREMENTS	20
5.1 SUPPORT REQUIRED FOR FULL IMPLEMENTATION AND REDUCED IMPLEMENTATION SCOPE.....	20
5.2 NETWORK SETUP AND TEST SCENARIO.....	22
5.3 MEASUREMENT METRICS, TOOLS, AND METHOD.....	23
5.4 TEST RESULTS AND ANALYSIS.....	24
5.4.1 <i>Per-Hop delay for Real-time service at R1</i>	24
5.4.2 <i>Packet Loss</i>	24
5.4.3 <i>Delay for Real-time Service at R3</i>	25
5.4.3.1 <i>Queuing Delay without Interfering Traffic</i>	25
5.4.3.2 <i>Queuing Delay With Interfering Traffic</i>	26
5.4.4 <i>Effect of no QoS Support at Link Layer</i>	27
6. DISCUSSION	28
APPENDIX A: SCRIPTS USED IN SETUP.....	29
A.1 TRAFFIC GENERATION SCRIPT DESIGN	29
A.2 SCRIPTS FOR NODE A.....	29
A.3 SCRIPTS FOR NODE C.....	29
A.4 SCRIPTS FOR ROUTER 3.....	29
A.5 SCRIPTS FOR ROUTER 1.....	34

A.6 SCRIPTS FOR ROUTER 2.....	37
REFERENCES	39
REFERENCES TO WORK IN PROGRESS	41

Figures

FIGURE 1: TCP/IP REFERENCE MODEL [15, COMER 2002]	4
FIGURE 2: TCP/IP SERVICE MODEL [15, COMER 2002]	4
FIGURE 3: INTER-NETWORK USED FOR TELESCEINCE DEMONSTRATION [13, LEE 2003]	5
FIGURE 4: BINARY FORMAT FOR 20-BIT FLOW LABEL FIELD IN IPV6 BASE HEADER [44, BANERJEE 2002]	11
FIGURE 5: SCENARIO TO DESIGN AND IMPLEMENT IN THE LABORATORY	13
FIGURE 6: PLOT OF MAXIMUM DELAY VS. BUCKET SIZE FOR A FLOW SERVICED BY WFQ	16
FIGURE 7: PRIORITY QUEUEING PACKET SCHEDULER FOR EVERY HOP IN DIFFSERV DOMAIN	16
FIGURE 8: PLOT OF BURST SIZE VS. VIDEO APPLICATION DATA RATE.....	18
FIGURE 9: NETWORK SHOWING INTER-CONNECTED DIFFSERV DOMAINS	19
FIGURE 10: NETWORK DIAGRAM OF SETUP IN THE LABORATORY.....	22
FIGURE 11: NETWORK DIAGRAM OF SETUP SHOWING DETAILS OF LINKS.....	23
FIGURE 12: INSTANTANEOUS DELAY WITHOUT INTERFERING T RAFFIC.....	25
FIGURE 13: INSTANTANEOUS DELAY WITHOUT INTERFERING T RAFFIC WITH ZOOM AT BURST INTERVAL.....	25
FIGURE 14: INSTANTANEOUS DELAY WITH INTERFERING TRAFFIC.....	26
FIGURE 15: INSTANTANEOUS DELAY WITH INTERFERING TRAFFIC WITH ZOOM AT BURST INTERVAL.....	26
FIGURE 16: INSTANTANEOUS TRANSMISSION DELAY BETWEEN ROUTERS R3 AND R1 (NO INTERFERING TRAFFIC)....	27
FIGURE 17: INSTANTANEOUS TRANSMISSION DELAY BETWEEN ROUTERS R3 AND R1 (INTERFERING TRAFFIC).....	27

1. Introduction

Many definitions of term *grid* and varied examples of grids exist today. First section reviews different definitions and examples of grids that exist today. Aim is not to define the term *grid*. Aim is to understand what the term *grid* refers to today. Similarly, *quality of service* is an ill-defined term and its meaning changes when looked at from different perspectives. Second section describes application quality of service and its relevance to grids with a few examples. Public Internet is the inter-network that connects computers in large-scale grids. Third section describes importance of internet technology, IPv6, and quality of the network service to grids. Fourth section describes an example of a grid application that requires network layer quality of service over public Internet. Fifth section lays out the scope of work described in this report and report organization.

1.1 What is a Grid?

[1, Foster 1999] defines *grid* as hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. Here, pervasive means access to grid is available within confines of whatever environment grid is designed to support. Grid is not about compute cycles only, but data, sensors, and people also. Since then, many other definitions of grid have been proposed. Definition in [2, Foster 2001] restricts the grid to a scenario that closely resembles what is described in [1, Foster 1999] as *virtual grid*. This definition implies a grid that spans multiple institutions and thus prevents intranet scale grids from being called a grid. [3, Foster 2002] adds further restrictions on definition given in [2, Foster 2001]. All these restrictions have been questioned [4, Gentsch 2002][5, Jonston 2002]. [6, 2002] defines the grid in terms of basic services that should be offered by the software infrastructure that comprises the grid. [7, Freund 2002] states that grid may not be multi-institutional.

Definition and description of a grid in [1, Foster 1999] give an idea of what a grid can be. All other definitions present specific and sometimes conflicting perspectives on grids. No definition clearly tells how to differentiate between a grid and any other system. Lack of such a *defining* attribute of grid in general leads to the question: How can one practically differentiate grids from other systems today?

Grids that use idle cycles of geographically distributed computers to support applications that can be broken into independent tasks are known as *scavenging grids*. Examples of scavenging grids are SETI@home [33] and World Community Grid [34]. Offerings like *on-demand services* from IBM and *utility computing* services from SUN Microsystems Inc use grid capabilities to meet short-term requirements for resources. These applications are driven by cost-performance [1, Foster 1999]. Grids that support applications that enhance human-to-human interactions are known as *collaboration grids*. This may also involve shared use of resources such as data archives, simulations, and remote instruments [1, Foster 1999]. An example of a collaboration grid is AccessGrid [35]. Grids that synthesize new information from data that is maintained in geographically distributed repositories, digital libraries, and databases are known as *data grids*. Such applications are often computationally and communication intensive as well

[1, Foster 1999]. Examples of data grids are Sloan Digital Sky Survey (SDSS) [36] and TerraServer [37]. Some governments have started *national grid* initiatives. Such grids could help in increasing utilization of high-end facilities, developing a strategic computing reserve and enable collaboration among scientists. Such grids are characterized by large scale of distribution and high-end resources [1, Foster 1999]. Examples of *national grid* initiatives include GrangeNet [38, 2002], and TeraGrid [39, 2002]. Categorization of grids by some vendors is based on how corporations and institutions are organized today [9, Joseph, 2004][40, SUN][41, SUN]. A grid that brings together computational resources in a department of an organization is known as *cluster grid* or *infra grid*. A grid that brings together computational resources in the same organization is known as *enterprise grid*, *campus grid*, and *intra-grid*. A grid that brings together computational resources spread across organizations is known as *global grid*, *inter-grid*, *extra-grid*, or *partner grid*.

If one looks at examples of grid system externally, applications like on-demand services from IBM, utility computing services from SUN Microsystems Inc, national grid initiatives, AccessGrid and Internet-scale idle cycle scavenging applications distinguish grids from other systems. Looking at grids internally, focus on using existing underutilized resources (underutilized instruments, idle CPU cycles, disk space) to achieve system functionality differentiates grids from other systems. When grids are associated with scientific communities, *scale of these systems* (physical area spanned by these systems, resources spread across organizations, massive amount of data) is enough to distinguish between grids and other systems. Here, grids span across countries and even continents. With commercialization, grids can no longer be differentiated based on scale only. Here, grids primarily help in *increasing utilization of existing resources* and thus creating high aggregate capacity. Grid is a distributed system. Goal of this system, like any other distributed system, is to coordinate use of resources to provide transparent, dependable and consistent access to system functionality. Like any distributed system, it is also an aggregate of software and hardware infrastructure interconnected by digital communications layer [8, Banerjee 2004]. *Performance requirement of grid applications* and *ownership of grid resources* are two important factors that influence the construction of any grid.

1.2 What is Quality of Service?

End users (human and non-human) interact with a grid by using the applications that run on the grid infrastructure. An application running on a grid can be thought of as a service being offered by the grid to the user. In simple terms, quality of a service refers to the degree of *goodness* of the service with respect to some desired goal in mind. User's expectations, nature of the task the user performs by using the application, and the type of data (multimedia data or non-multimedia data) used for communication, are some factors that determine the quality of the service an application should provide to its users [42, Miras 2002]. Applications in which user takes an action and then waits for a response are known as interactive applications [42, Miras 2002]. Two examples of interactive applications are video conferencing, and remote control of devices. In video conferencing, remotely located humans use computers to communicate with each other by using live voice feedback and live video feedback from the remote party. AccessGrid is an example of a grid that supports video conferencing application. In applications like tele-

microscopy, remote surgery etc, a human performs his or her task by sending commands to the remote computer that controls the device and viewing the live video feedback from the remote device. Tele -microscopy by Telescience [12, Akiyama 2003] is an example of a grid application that supports remote control of devices.

Remote communication takes time—there is a delay between the transmission and reception of data . This delay is not necessarily constant, i.e., there can be a variation in the delay. Variation in delay is called jitter. Delay experienced when data moves from source to destination is known as *one-way delay* . Delay affects the quality of interactive applications that transport audio-video data over a distance. Normal human -to-human conversation requires synchronization between the lip movements and corresponding spoken words. The synchronization between lip movements (video data stream) and spoken words (voice data stream) is called *lip sync* . In order to meet this requirement, video data and accompanying voice data should be played within 80 milliseconds of each other at the receiver [42, Miras 2002] . Interaction between humans is not natural when response of the one human (verbal reply and visual gestures) is delayed by more than 150 milliseconds [42, Miras 2002] .

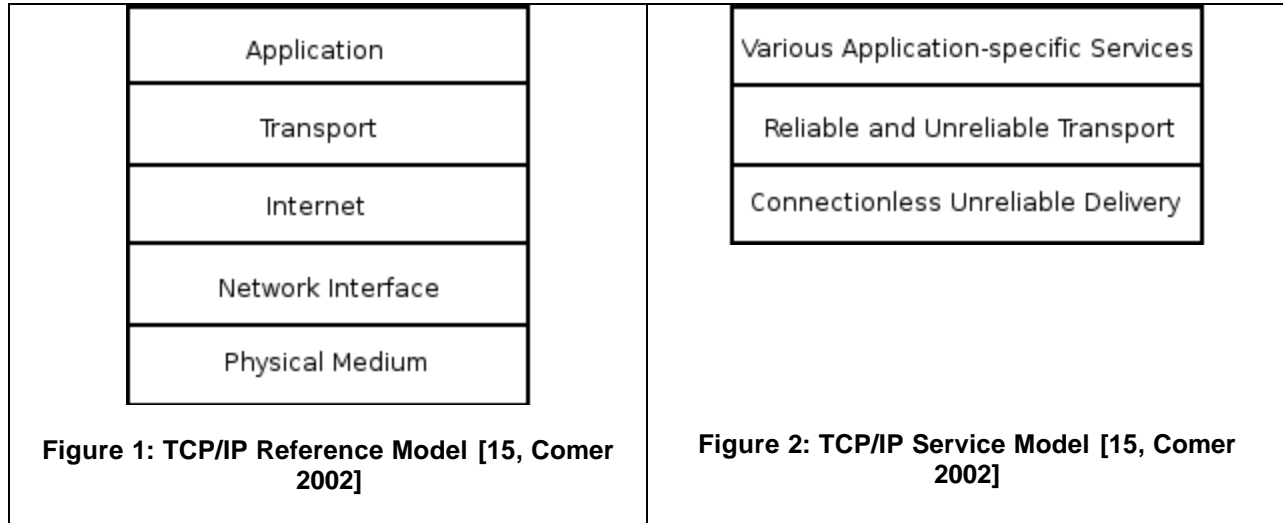
Certain applications require large volumes of audio -video data to be transported over a distance. In critical applications like tele -surgery, surgeon expects live high -quality video feedback from the remote operation room with no deterioration in quality at all. Here, high -quality video means that video must show high level of detail. This requires that source computer must send more video data to the receiving computer. Both tele -surgery and tele -microscopy require that the one -way delay experienced by the control data (like instructions to move or rotate the equipment) should be very low and this data should never be lost in the way. Any deterioration in video quality, audio quality, and voice quality is always annoying to the user.

1.3 Grids, TCP/IPv6 and Quality of Service

When looking at grids from data networking point of view , grid applications belong to the application layer of the OSI model [15, Comer 2002] . Internet technology is the dominant networking technology that interconnects computer networks spread across the world today. Computers in the grids, which span cities, countries, and continents, are very likely to use TCP/IP protocol suite for data communication. Conceptually, Internet technology software is organized in layers [15, Comer 2002] , as shown in figure 1 .

TCP/IP protocol suite uses software to hide the details of various physical networking technologies from the applications. TCP/IP protocol suite is primarily concerned with software in application layer, transport layer, internet layer, and how to integrate various types of links with the internet layer. Conceptually, each layer offers a service to the layer above it , as shown in figure 2. Internet layer uses the services provided by the data link layer below and offers a connectionless unreliable packet delivery service to the transport layer. This means that packets can be lost or reordered in the network, processing of a packet in the internet layer does not depend upon processing of any other packet. In most of the public Internet today, all packets, which are forwarded by a router , are put in the same logical queue and transmitted on the outgoing link on a first come first serve basis. This network service is known as *best effort service* . Such queuing implies

unpredictable variable queuing delay for a sequence of packets. Such a model of processing the packets is not suitable to meet the delay requirements of applications like video conferencing, remote control of devices, etc.



Delay experienced by the end user is not due to the network software and hardware only. Processing delay in operating systems, protocol stack implementations, computer bus speeds, etc encountered in all nodes along the path to the destination also contribute to the variable delay. In order to determine whether application's requirements can be met or not, software and hardware of all intermediate computers needs to be taken into account. Software in Internet layer processes the application's packets at all intermediate nodes. If this layer provides better than best effort service to packets generated by applications like video conferencing, and remote control of devices, it can be possible to run such applications over the internet and grid also.

IPv4 is the most important protocol in the internet layer. IPv6 [14, Deering 1998][16, Davies 2003] is the next version of IPv4. It will be widespread some day in the future. IPv6 provides 128-bit address space to solve the problem of shortage of IPv4 addresses. For grid systems to *scale geographically* and operate over the data networks in the future, grids need to be ready for IPv6. For grids to support applications like video conferencing, and remote control of devices, features of IPv6 that are relevant to providing quality of service support are important.

1.4 Example of a Grid Application: One-way Video over IPv6 Inter-network

At iGrid 2002 workshop, Telescience demonstrated tele-microscopy and subsequent data analysis as a grid application over an intercontinental IPv6 network [13, Lee 2003]. In this grid application, the electron microscope called UHVEM, data generated by the experiment, computers needed to process the data, and researchers who conducted the experiment were distributed across continents, as shown in figure 3. Tele-microscopy is the process of remotely controlling a high-energy electron microscope. Two one-way audio-video streams were transported from Osaka University, (Japan)

one to NCMIR at San Diego (West Coast of USA) and second to the workshop venue at Science park in Amsterdam (Netherlands in Europe). These streams correspond to the dynamic images of a physical specimen placed under UHVEM at Osaka University. People in San Diego examined the specimen. People look at the video of the specimen and use hardware knobs on a co-located computer to examine the specimen from various angles. When hardware knobs move, data is sent back to the computer system controlling the UHVEM and dynamic images of the specimen from a new angle are sent back to San Diego. DVTS system [10, Ogawa 2000] was used to transport the multimedia streams to both remote locations.

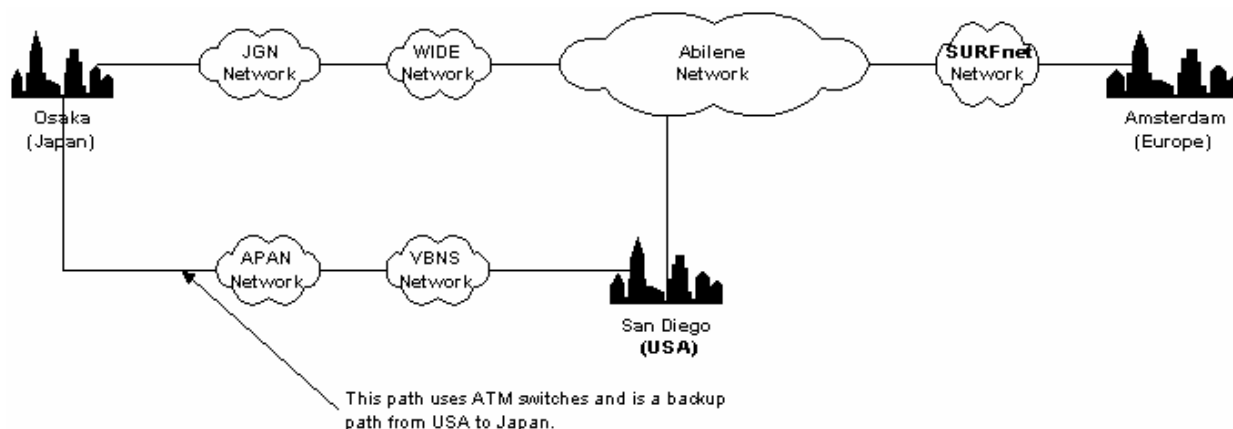


Figure 3: Inter-network used for Telescience demonstration [13, Lee 2003]

Tele-microscopy is a visually intensive process, which is not tolerant of high end-to-end delay or frame loss and benefits from the highest possible quality video [13, Lee 2003]. DVTS system is capable of transporting DV format audio-video stream by using RTP protocol over IPv4 as well as IPv6 protocol. RTP protocol is an application layer protocol that can use either UDP or TCP transport layer protocols. These protocols in turn use either IPv4 or IPv6 network layer protocols to deliver data to remote computers [15, Comer 2002]. DVTS system is compliant with NTSC television transmission standard. NTSC standard requires application-level data rate of 30 frames per second [11, Sharda 2002]. When DVTS transmits data at 30 frames per second (also called full frame rate), its audio-video streams require roughly 30 Mbps bandwidth.

DVTS system uses application adaptation techniques to maintain quality of video at the receiver. Receiver of this system uses receiver-side frame buffering to tackle the variable end-to-end delay. Because the playback of the received audio-video does not start until the receiver buffer is full, large playback delay is expected. If required end-to-end network throughput is not available, DVTS system can be configured to operate at reduced frame rate. DVTS system achieves this by not sending the video data to the receiver. This implies loss of the quality of video at the receiver because sender does not send the actual images to the receiver at all. [10, Ogawa 2000] states that *RTP does not ensure the packet's reachability to the destination host*. This implies that UDP transport protocol was used with RTP. Packets that are lost in the network will cause degradation in quality of received audio-video.

[12, Akiyama 2003] states that ATM switches were used to connect Japan and US. [13, Lee 2003] states that an Ethernet infrastructure was used in the US, and the inter-continental IPv6 network that carried the audio-video data comprised of 1Gb to 10Gb links. Before the demonstration, 300 Mbps bandwidth was available from Osaka University to Amsterdam and 400 Mbps bandwidth was available from US to Amsterdam [12, Akiyama 2003].

There is no measurement data available on the bandwidth utilization, packet loss and latency —when the demonstration was going on —at the network layer. In addition, there is no information available on the perceptible quality of video received in Amsterdam during the whole demonstration.

1.5 Scope and Report Organization

Grids that support network-based multimedia applications like visual remote control of high-end scientific instruments, human-to-human interaction using multimedia facilities [35], and remote access to data archives like multimedia libraries can take advantage of new quality of service features of IPv6. From grid's point of view, a grid application requires end-to-end quality of service for its application flow(s). From network point of view, grid applications are no different from other applications that need end-to-end quality of service.

This work focuses on network quality of service aspects of grids, which are designed to support multimedia applications with soft real time requirements, like the one demonstrated by Telescience. Aim is to understand the network quality of service requirements of such grid applications, understand the standardized architectures available to create a network service that can meet the requirements, design such a network service with available standards, implement the service over an experimental network in the laboratory, and verify the service.

This report is organized as follows. In chapter 2, network quality of service, and current internet technology quality of service standards, namely, Diffserv [17, Nichols 1998][18, Blake 1998][20, Grossman 2002] and Intserv [21, Jha 2002] architecture, are described. In chapter 3, diffserv architecture and IPv6 flow label are discussed. In chapter 4, design of a diffserv domain is discussed. In chapter 5, scope of implementation, network setup in the laboratory and measurements are described. Chapter 6 is an epilog to this work.

2. Network QoS and IPv6-QoS Features

In this chapter, first section describes the key parameters used to describe quality of the service offered by a network. Second section describes the architectures or architecture's standardized by public Internet's standardization body, IETF, to build a variety of network services using Internet technology. IPv6, next version of IP protocol—the key network layer protocol—provides a 20-bit field in its base header for different network services. Third section describes the purpose of this field and efforts made to use this field. Forth section describes the relevance of quality of the services offered by technologies like ATM, and MPLS [21, Jha 2002], from TCP/IP reference model point of view.

2.1 Network QoS Parameters

The *network quality of service* (or *network QoS*) term refers to a set of network performance characteristics. The table below lists the important parameters.

Category	Parameters
Timeliness	Delay, Jitter
Bandwidth	Data rate
Reliability	Packet loss rate

Table 1: Important network quality of service parameters [21, Jha 2002]

A frequently term in literature related to quality of service is *flow*. There are many definitions of the term *flow*. Generally, *flow* refers to a sequence of packets that are related to each other in some way. An *application flow* is a sequence of packets that originate from a particular application and expect similar forwarding treatment from the network. For example, one instance of a client program of an instant messaging application like Yahoo chat can generate one application flow for text chat data, one application flow for audio-only chat data, and one application flow for audio-video chat data (i.e., same flow includes both audio and video data). An aggregate flow is a set of application flows that expect similar forwarding treatment from the network.

Delay means the end-to-end delay suffered by packets. End-to-end delay affects applications that require interactivity. *Jitter* means variation in end-to-end delay. *Bandwidth* means throughput that an application can expect to receive for its flow. Multimedia applications need to maintain an average data rate with an allowance for some bursts of packets so that quality of the presentation at the receiving end can be maintained. *Packet loss rate* means number of packets lost enroute to the destination. Causes of packet loss include network congestion.

In order to provide some network QoS to a certain flow (a sequence of related packets), source needs to provide the *traffic profile* of this sequence of packets to the network and request its timeliness requirements and reliability requirements for this particular flow [21, Jha 2002]. Many multimedia applications generate traffic at variable data rate. For a variable rate source, traffic profile is bounded by peak bit rate, average bit rate and burst size. Here, burst size refers to the data (in bytes) generated by the source at the

peak rate. For variable rate interactive applications, it is necessary for the network to allow the burst of data to reach destination. This is required to maintain interactivity between the sender and receiver.

Every network node enroute to the destination (routers, switches, bridges) needs to be setup in order to meet network QoS requirements for this flow. Two key resources in the network nodes are bandwidth and buffer size [21, Jha 2002]. Every node needs to be able to identify the flow, provide sufficient buffer size to this flow to avoid packet loss and accommodate bursts. This buffer is referred to as a queue. In order to meet the bandwidth requirements of the flow, this queue needs to be served at a rate at least equal to the average bandwidth requirement of the flow. As intermediate network nodes will be processing many application flows, each with its own profile, each flow's packets are separated into different (logical and maybe, physical also) queues and each queue is served with a different criteria. The algorithm that decides which packet from many queues gets its turn to be transmitted on the link is called packet scheduling algorithm (or packet scheduler) [21, Jha 2002].

2.2 IPv4/IPv6 QoS Architectures

Internet today mostly offers only one service to packets in an application flow, namely, best-effort service, which means that packets are served on a first-come-first-serve (FCFS) basis). In order to provide more network services, two quality of service architectures have been standardized by Internet Engineering Task Force (IETF), namely, Diffserv and Intserv.

2.2.1 Integrated Services Architecture

Intserv architecture can provide guaranteed end-to-end quality of service on a per-flow basis. In Intserv architecture, RSVP protocol is used to deliver an application flow's quality of service parameters like bandwidth, delay etc to all intermediate routers. After the reservation is confirmed, source node sends the packets. All intermediate routers need to maintain per-flow state, which affects router's scalability. Due to this fact, intranets are suitable for deploying network QoS based on Intserv architecture [21, Jha 2002].

2.2.2 Differentiated Services Architecture

Diffserv architecture provides quality of service to aggregate-flows and not individual application flows. Therefore, it cannot provide any quantitative guarantees to an application flow and hence, offered service cannot be measured quantitatively. There is no need to maintain per-flow state in the routers as is the case with Intserv. Diffserv architecture specifies building blocks that can be used to implement different network services. In a typical scenario, an ISP creates a diffserv domain by setting up an ingress router with classifiers that identify application flows and marks the packets with a pre-defined value, namely, Diffserv Code Points. Diffserv code point actually means a specific per-hop behavior (PHB) that packet must receive on every router along the data path in that diffserv domain. There are two PHBs currently standardized, namely, Expedited Forwarding and Assured Forwarding. Diffserv architecture is scalable and is suitable to be deployed in the public Internet. However, what services are offered by ISPs, how these services are defined and how these different services will be mapped across

ISPs (or diffserv domains) to achieve end-to-end quality of service for an application flow have not been standardized in Diffserv architecture.

To achieve end-to-end quality of service using Diffserv architecture, service level agreements (SLAs) are required between intermediate ISPs so that a packet gets a certain minimum qualitative level of service. DSCP field is mutable. Different ISPs may choose to mark aggregate flows using different DSCP. Some of the problems with Diffserv are as follows [21, Jha 2002]:

- * End users and access networks have no way of specifying what per-hop behavior they want from all intermediate diffserv domains enroute to the destination
- * Multi-field classifier used in Diffserv to identify flows uses fields in transport PDU (like port number). In order to use this classifier, transport header needs to be parsed by intermediate nodes. This increases processing overhead. When an IP packet is encrypted and sent in a tunnel, transport header fields will not be available to the classifier.
- * A logical entity, which will keep track of resource allocation within a diffserv domain, and participate in inter-domain SLA negotiations, has not been standardized. However, it has been named as Bandwidth Broker (BB) and described.
- * Protocols for automatic negotiation of SLAs across diffserv domains have not been standardized.

Some problems with Diffserv architecture faced while building a Premium network service in Qbone in Internet2 network have been documented in [45, Teitelbaum 2002]. One of these problems is that mutability of DS field burdens all edge routers with the need to re-mark traffic. Other problems include provisioning across diffserv domains in order to provide end-to-end network QoS service and dynamic admission control.

2.2.3 Intserv-over-Diffserv Architecture

Intserv-over-Diffserv architecture combines the best of Intserv and Diffserv architectures to provide end-to-end QoS over public Internet [21, Jha 2002]. In this architecture, Intserv is used in access networks where number of application flows going through the routers is certainly not high. ISPs deploy Diffserv architecture based network services. Guaranteed service of Intserv is mapped to Expedited Forwarding PHB of Diffserv. Controlled load service of Intserv is mapped to Assured Forwarding service of Diffserv.

2.3 IPv6-only QoS feature (Flow Label)

IPv6 provides a 20-bit field called *flow label* in its base header, which can be used to provide various qualities of service. Per its specification in [25, Rajahalme 2004], purpose of flow label is to provide an efficient means of classification of packet flows at intermediate nodes. Classification using flow label is considered efficient because classifiers currently used in routers need access to fields in transport layer segments (like port number). This involves parsing the IPv6 extension headers and transport header, which increases processing overhead. Flow label is set in an IPv6 packet by the source node. Flow label cannot be changed by intermediate routers. Hence, it can be depended upon to be delivered to all routers with the same value. Source address, flow label and destination address tuple is defined to uniquely identify a flow. 20-bit flow label field does not have any binary encoding and is a meaningless number. Per the explanations provided by authors of the

standard in [46], flow label is not meant for network QoS purposes only. Network QoS can be one of uses of flow label. Immutability of flow label is important because it will allow intermediate nodes to get the same flow after classification based on the flow label classifier. Adding structure (or binary encoding) to this 20-bit field limits the purposes for which flow label based classifier can be used.

2.4 Relevance of Sub-IP Layer Technologies

Sub-IP layer technologies include all technologies that encapsulate IP packets and transmit it over the physical medium. These include physical networking technologies like Ethernet, networking technologies like ATM, and techniques like MPLS. Frame processing done at sub-IP layer, especially, queuing, also introduces delay. Hence, quality of service support is also required at the sub-IP layers. ATM is a networking technology that provides quality of service support at the network layer. Telephone companies and ISPs use IP over ATM network to provide quality of service support in their IP network. ISPs also use MPLS technology to provide quality of service support in their IP network. Both ATM and MPLS are local to an ISP or a telephone carrier. IP protocol is designed to connect networks. Hence, IP Quality of Service architectures like Diffserv and Intserv are not designed with specific sub-IP technologies in mind. IP protocol header fields are not created to support technologies that exist in certain local networks. Due to this reason, these sub-IP layer technologies will not be discussed in this report.

2.5 Discussion

Due to Intserv's architectural limitations, Intserv is deployable in access networks. Diffserv is suitable designed with ISP networks in mind. Intserv-over-Diffserv architecture is the most suitable architecture —among the currently standardized architectures—to build end-to-end network services over public Internet. Definition of flow label field is very general. Network QoS is not the only purpose for which flow label field. Its key property, immutability, makes it attractive for end-to-end network QoS purposes. The next chapter describes how flow label field can complement the diffserv architecture and a practical scenario where flow label can be useful.

3. IPv6-QoS Scheme for Private One-way Video

In this chapter, first section describes prior work done in standardization of flow label field. It also describes uses of flow label field with diffserv architecture described in the literature. Second section describes one of the prior works in detail and a scheme to provide private, one-way video over an IPv6 inter-network. Third section describes the real-life scenario where this scheme can be useful. Forth section describes the difficulties in implementation of this scheme as it is in the laboratory. This section also describes the scope of implementation work in a laboratory given the time constraints and resource constraints. Hence, forth section sets the context for the design and implementation work described in the next chapters.

3.1 How do Diffserv Architecture and IPv6 Flow Label Fit in?

Prior to its standardization, many papers and internet drafts were submitted suggesting ways of using flow label. [22, Fgee 2003] treats flow label and source address pair as a unique flow identifier and uses traffic class field for prioritizing the flow. Per [22, Fgee 2003], using the unique flow label and source address pair for routing—like MPLS—minimizes the lookup time (longest destination address match method) in a router. [24, Fgee 2004] proposes a new QoS management scheme using traffic class and flow label fields in IPv6 base header. It treats flow label and source address pair as a unique identifier. It uses the pair as an index in the resource reservation table in a domain manager entity for fast lookup. It uses the pair to perform forwarding—like MPLS. It uses traffic class field as an indicator of priority and not service class—as per the diffserv specification. [23, Tang 2003] proposes end-to-end QoS scheme using the flow label field, and a new flow label option for IPv4 to make IPv4 protocol compatible with IPv6 protocol's flow label field. It uses the flow label format proposed in [44, Banerjee 2002] to integrate Diffserv architecture with flow label field. Some internet drafts that are applicable to Diffserv are [43, Conta 2001] and [44, Banerjee 2002]. A generic approach, which summarizes most of these efforts and a new scheme using hop-by-hop extensions header is presented in [44, Banerjee 2002]. Advantage of this scheme is that it can support both Intserv and Diffserv architectures and it still leaves the flow label usage for further re-definitions. In one of these schemes, flow label can have the structure shown in figure 4 for using this field with Diffserv architecture.



Figure 4: Binary format for 20-bit flow label field in IPv6 base header [44, Banerjee 2002]

Here, the first three bits specify the meaning of flow label. When the first three bits are 0x3, the meaning of flow label is per-hop behavior.

An immutable DSCP field may not be a substitute for flow label. To make things clear, consider classifiers as criteria based on data in a packet's fields and/or policy (like time of day). Criteria are applied to every packet and decision of the criteria is a number, namely, DSCP. The meaning of DSCP is PHB. The decision is stored in every packet (resulting in remarking if

necessary). Diffserv achieves scalability because of its per -packet processing at all nodes in the DS domain. Core routers of DS domain classify efficiently because per-packet decision is already available in the packet header itself. It is necessary to keep a mutable field in base header for storing the local decision of Diffserv domain. By using flow label as mentioned above, DSCP based functionality as defined in the Diffserv architecture is not affected at all.

3.2 Use Case for Diffserv and Flow Label-IPv6-QoS enabled VPN

Flow label specification, in its current form, is very general. If *source address, flow label, destination address* 3-tuple is to be used to identify flows originating from end nodes, then as far as network QoS is concerned, flow label suits usage with Intserv architecture. However, in this case, immutability of the flow label loses its relevance because Intserv is not scalable enough for public Internet. As Diffserv works with aggregate flows, identification of micro -flow specified with 3 -tuple above is not of much use. However, in case of a VPN connection (IPv4/IPv6 -IPv6 encapsulation and encryption) between edge routers of two access networks, flow label with embedded PHB can be useful with Diffserv.

Benefits of this scheme are the following:

- * Access networks have a way of specifying the per -hop-behavior they need to all intermediate diffserv domains.
- * Classification performed by at the ingress routers of diffserv domains to identify micro -aggregates of flows using the flow label and source address or source prefix will be efficient because transport headers need not be parsed.
- * Since the packets going over the VPN connection is encrypted, flow label helps in identification of aggregate flows.

This scheme has the following limitations:

- * Public Internet is assumed IPv6-enabled, when deployment of IPv6 has been slow for many years.
- * Edge routers of access networks are assumed IPv6 enabled.
- * Public Internet is assumed composed of interconnected diffserv domains with appropriate SLAs based on flow labels, when such an interconnection is widespread today. Flow label field is still experimental.
- * Flow label functionality is assumed, when today in many common operating systems like MS Windows, various linux distributions and commonly available routers from Cisco, flow label is ignored [Banerjee 2005]. This is probably because of lack of a standard and well-defined use-case for flow label field.
- * [45, Teitelbaum 2002] suggests that provisioning of Diffserv domains, their configuration, inter -domain SLA definition and enforcement and service verification using measurements is a complex task.

3.3 Scenario to Design and Implement in the Laboratory

Scenario to design and implement in the laboratory would be to set up two intserv enabled access networks connected by ipv6 enabled public internet — comprising of multiple diffserv enabled ISP networks with proper SLAs —by using VPN connection between edge routers of the access networks. This is shown in figure 5. VPN connection would use the flow label provided by the first hop diffserv domain and will be used by each intermediate domain for

classification and mapping to appropriate PHB supported by the diffserv domain. Each intermediate domain will use DSCP field for per -packet marking —relevant internal to the domain.

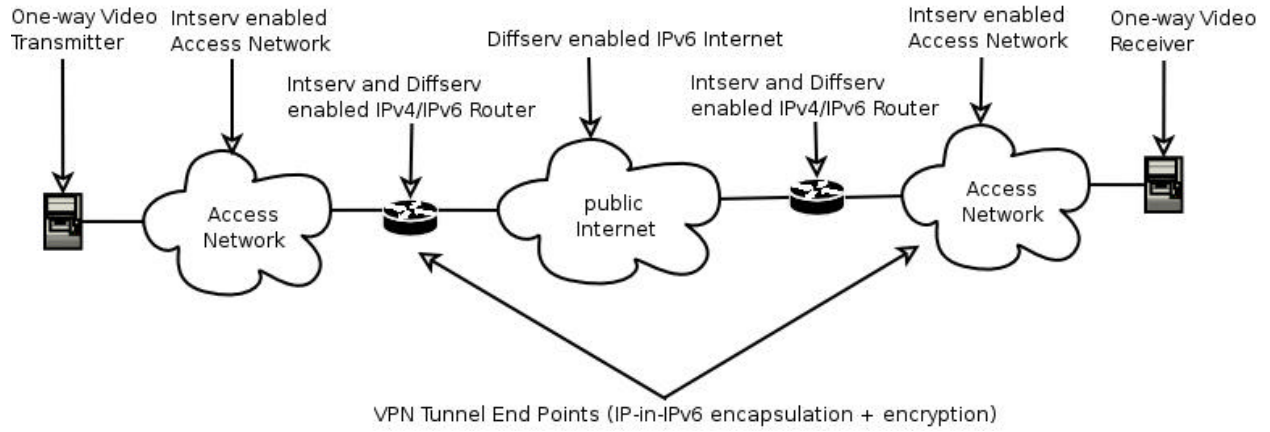


Figure 5: Scenario to design and implement in the laboratory

4. Network Service Design for Supporting One-way Video

The basic block of IPv6-QoS enabled VPN is a diffserv domain offering a network service to meet quality of service requirements of interactive one-way video applications. In this chapter, first section describes all network services that are necessary in order to introduce one special delivery service for one-way video applications. Second section describes the flow label value that video traffic should carry in order to be identified as traffic that needs the real-time service and the DSCP value for real-time service used locally within the diffserv domain. Third section describes how priority-scheduling algorithm can be used in diffserv-enabled routers to implement all network services. Forth section describes a simple way of roughly calculating the traffic profile of an aggregate flow. Fifth section describes the traffic conditioning that should be performed on traffic destined to each service's queue. Sixth section describes the traffic classifier specification required to identify the real-time traffic and the problem of providing domain-level network service by using diffserv-enabled routers.

4.1 Services Required

A diffserv domain is created to provide differentiated network services to different traffic. A differentiated network service can be an elevated service that provides a service better than the best effort network service commonly available on public Internet today. It can also be a non-elevated service that provides worse than best effort service. Downside of designing an elevated service with public Internet in mind is that to meet application's requirements, all intermediate diffserv domains should provide an elevated service. A network service is designed to support network related requirements of similar applications. This work focuses on providing an elevated service to meet network requirements of interactive one-way video applications. Such applications typically use UDP transport protocol to transport multimedia streams to unicast destinations. Network should meet the average bandwidth, one-way delay and packet loss requirements of these applications, as described in chapter 1. Audio and video codecs commonly used for compressing video and data use lossy compression techniques that produce variable bitrate (VBR) traffic. Due to high compression, traffic generated by these applications is susceptible to packet loss and bit errors. Since, UDP transport does not provide reliable delivery, it is necessary to reduce packet loss as far as possible.

Since an elevated quality of service is about providing some assurance (if not guarantees) on service level to the end user, it is important to make the service as good as possible so that end users can perceive the benefit of paying for such a service. A tight quantitative requirement for such a network service based on diffserv architecture is guarantee on average bandwidth, controlled maximum per-hop delay (close to a bound), zero packet loss and minimized jitter. Such a network service will be referred to as *real-time service* in this report. This service will transport the one-way video data over the diffserv domain. *Best-effort service* will carry all the remaining traffic. In order to maintain the network, certain protocol messages like ARP, routing system messages, SNMP messages, NTP messages also need a network service that is much more critical than any other network service. This

service should not be accessible outside the diffserv domain. This service is called *network control service* in this report.

4.2 Flow Label, DSCP and PHB Id for Real-time Service

An experimental 6-bit DSCP for real-time service is chosen as defined in [17, Nichols 1998].

$DSCP = 000111$

(When $0xFC$ mask is applied to DS field, result will be $0x1c$)

(When $0xFC$ mask is applied to DS field and right -shifted by 2, result will be $0x7$)

If flow label were to be used, the experimental 16-bit PHB-Id corresponding to above DSCP per [19, Black 2001] is:

$PHB-Id = 0001110000000001 = 0x1c01$

Flow label corresponding to this PHB-Id is chosen per [44, Banerjee 2002] where last reserved bit is 0:

$Flow\ label = 0110001110000000001\ 0 = 0x63802$

4.3 Packet Scheduling Algorithm at Diffserv Router

4.3.1 How to control maximum delay

When weighted fair queuing (WFQ) packet scheduler is used to serve a flow at rate R_L , which has been shaped per a token bucket specification, say B_{sz} and average rate R , and $R_L > R$ then maximum queuing delay experienced by any packet in the flow is bounded by $D_{max} = B_{sz}/R_L$ [21, Jha 2002]. Above plot visually shows the same for various service rates. In case of Diffserv, flow is an aggregate and not a micro-flow. When priority packet scheduler is used to serve a flow that has been shaped by a token bucket specification and this flow has the highest priority, maximum queuing delay experienced by the any packet in the flow can be approximated by the above expression where service rate will be the link speed. This is so because highest priority queue is always served before any other queue. If this flow belongs to a queue with less priority, then above delay measure cannot be used. However, if the queues with higher priority have very low traffic, then figure 6 can be a useful in getting a close idea of maximum queuing delay.

Figure 6 gives information about how choices can be made in order to configure a router in a diffserv domain in order to control the queuing delay introduced at the egress interface of that router.

In order to maintain a certain maximum delay at the router, the token bucket size of the aggregate flow needs to be controlled. Admission control module of the diffserv domain should check the token bucket size and average rate of the new *micro-aggregate* flow so that $R_L > R$ and B_{sz} is still below its threshold value corresponding to the required to meet the maximum queuing delay requirement at the router. In order to meet worst-case scenario, new values of B_{sz} and R can be calculated by simply adding the token bucket specification of the new micro-aggregate to the current values of B_{sz} and R parameters.

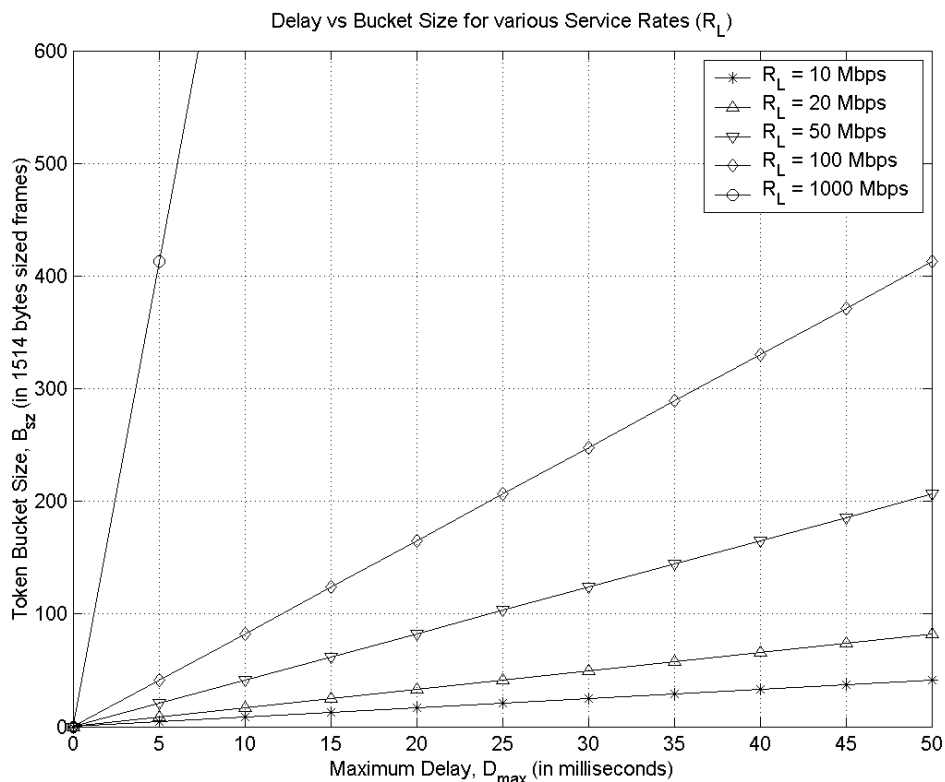


Figure 6: Plot of maximum delay vs. bucket size for a flow serviced by WFQ

4.3.2 Choice of Packet Scheduling Scheme

Diffserv architecture specifies the use of building blocks like shapers, policers, markers, meters, and packet schedulers etc to implement a per-hop behavior. To implement the required services, differentiation between traffic is required and priority packet scheduler enables us to do so, as shown in Figure 7.

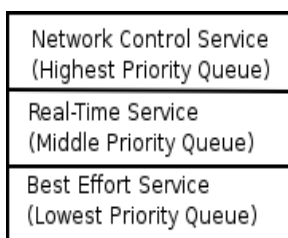


Figure 7: Priority queuing packet scheduler for every hop in Diffserv domain

Priority scheduler enables us to prioritize the traffic, control the maximum delay bound on real-time service and service the best-effort traffic. Since, network control service will not be servicing heavy traffic, it will not cause starvation of real-time service and best effort service. Because the bandwidth requirements of the network control service are very less, maximum delay at every hop in real-time service's queue will not deviate much from the maximum delay bound as described in the previous section. FCFS queuing

will be used for real-time service queue to avoid packet re-ordering when packets leave this queue. Real-time service queue can be the cause of starvation of traffic in best effort service. During the provisioning, the arrival rate at real-time service should be less than departure rate. Best effort traffic will get the leftover bandwidth but with a large delay. By setting a large queue length for best-effort service, and requesting the best effort flows to reduce their data rate, adverse affects on the end applications can be reduced. Best effort traffic primarily comprises of TCP flows [28, Thompson 1997][47, Banerjee 2005], which reduce their rate when packets are lost. Queuing management technique, Random Early Detection (RED), will be used to drop the packets in best effort queue to help TCP flows adjust their data rate. UDP flows, which form a small portion of best effort traffic [28, Thompson 1997][47, Banerjee 2005], are unlikely to adapt their rate in case of packet loss because UDP transport does not implement congestion control mechanisms.

4.4 Traffic Profile of Aggregate Flows

The term micro-aggregate flow is used in this report for aggregate flows that originate from a particular access network and is destined to a particular destination network for which access network needs network quality of service. The term micro-flow is used in this report to refer to a constituent multi-media stream of a particular micro-aggregate flow. Source access network needs to define the traffic profile of every micro-aggregate flow in the form of a token bucket specification, which refers to average rate and burst size of the flow. Traffic profile of a micro-aggregate flow also depends upon the end users, who initiate the micro-flows comprising the micro-aggregate flow.

In order to provision for a certain number of simultaneous micro-flows and in order to handle the worst-case scenario, token bucket of the each micro-aggregate flow can be calculated by adding the token bucket parameters of constituent micro-flows. Token bucket parameters of each micro-flow can be looked up from the figure 8. Number of micro-flows that constitute a micro-aggregate has to be guessed if information on the user activity is not available. To calculate the maximum burst size of a micro-flow, maximum number of average sized frames per burst has been assumed and is shown the figure 8.

Maximum frames per burst, $N_f = 2, 3, 5$ frames/burst (an assumption)

Video data rate, $R_a = 1.5$ Mbps for MPEG-1 and 3-6 Mbps for MPEG-2

Frames per second, $fps = 25$ frames/second

Max frame size, $F_{max} = 1514$ bytes

Max application PDU size assuming RTP/UDP/IPv6, $P_{max} = 1514 - 12 - 8 - 40 = 1454$ bytes

Max average packet rate, $R_p = R_a / (P_{max})$

*Max packets per burst, $B_{sz} = N_f * R_p / fps$*

This approach to calculating the token bucket specification of micro-aggregates is taken in the DS domain setup.

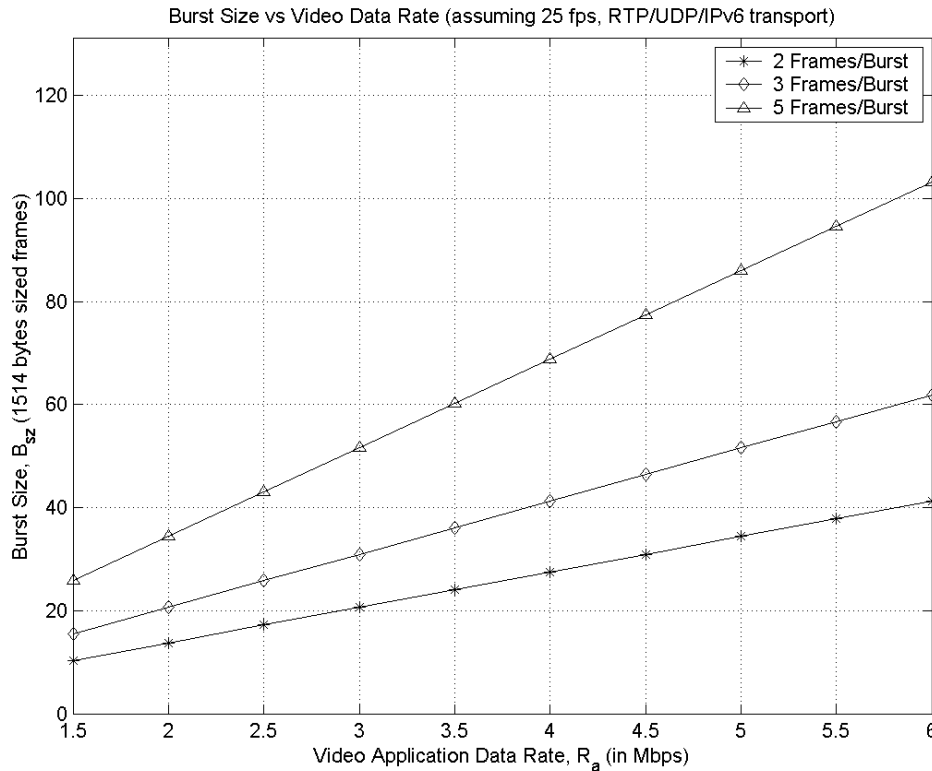


Figure 8: Plot of burst size vs. video application data rate

4.5 Traffic Conditioning at Diffserv Router

Traffic going out of network -control queue will not be shaped because shaping introduces delay and traffic is urgent. Since the provisioning for real-time service is done specifically to control the maximum delay, each micro - aggregate flow that avails this service will be policed against its traffic profile and out-of-profile packets will be dropped at the ingress router only. Out-of-profile packets will be dropped because in case of multi -media applications, limited loss is tolerable and a too -much-delayed packet is as good as lost. Aggregate flow leaving the real -time service queue will be shaped to control the maximum delay for this queue at each intermediate router's egress interfaces in the DS domain. Best effort traffic will not be shaped, as there is no need to do so.

4.6 Domain-Level Problems

In figure 9, T1, T2, T3, T4, T5, T6, T7 represent micro- aggregates. Each traffic aggregate originates from a particular access network and is destined to another access network. All traffic aggregates need same per -hop behavior from all intermediate diffserv domains. Routers R1, R2 and R3 form the core of diffserv domain, say, A. Routers R4, R5 and R6 form the core of diffserv domain, say B. Arrows show the path taken by aggregates through the diffserv domains. From figure 9, it is clear that bandwidth and buffer size allocation at each egress interface of all routers in the diffserv domain may not be the same. In order to setup routers R1, R2 and R3, DS domain A

should be aware of the aggregate flows that pass through each of these routers (or hops) and their traffic profile. Each aggregate can be identified by using a classifier defined as $\langle \text{source network address, flow label, destination network address} \rangle$. This classifier and its associated traffic profile could be used by all diffserv domains, including the ones owned by tier -1 ISPs, if they are along the path to destination. Domain setup also depends upon the routing system. Example in the figure 9 assumes that aggregates follow a fixed path inside the diffserv domain and across diffserv domains. This may not be true in public Internet. This adds further complexity to the provisioning problem. In the experimental setup, routes have been fixed by adding static entries in the routing tables.

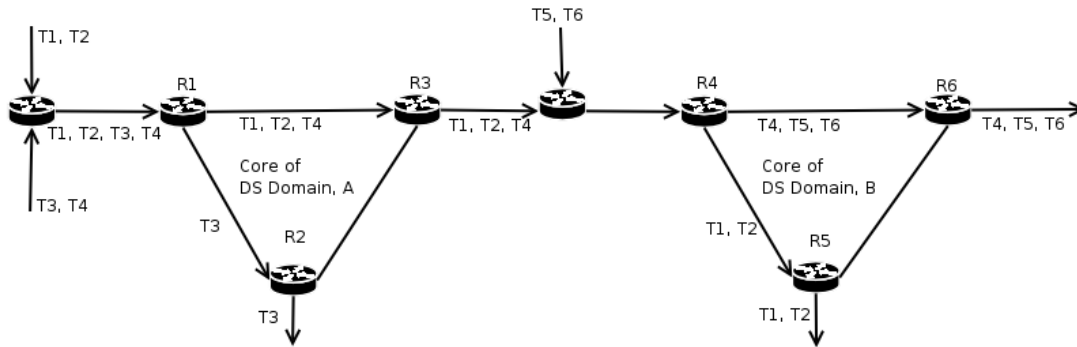


Figure 9: Network showing inter-connected diffserv domains

5. Implementation Scope, Network Setup and Measurements

In this chapter, first section describes the network setup in the laboratory, real-life scenario that we think of when setting up the network, and actual scenario—corresponds to the real -life scenario in a limited way—that is measured.

5.1 Support Required for Full Implementation and Reduced Implementation Scope

Public Internet is a complex system. In a typical scenario of public Internet, there can be ISPs that may not support Diffserv based services, thus creating *QoS holes* in the path to the destination. Heterogeneous hardware and software can throw up interoperability problems. This problem came up in the laboratory during this work. Public Internet comprises of many kinds of links each with its own bandwidth. On top of this, routing failures cause packet loss, packet reordering and even loss of connectivity. Given the complexity of public Internet and limitations of the scheme described above, this thesis work aims at creating an IPv4/IPv6 test-bed with similar hardware and software. Readily available computers and networking equipment in the laboratory are used for implementation. 10/100 Mbps ethernet NICs are readily available in the laboratory. Computers with PC hardware and commodity operating systems, namely, fedora core 3 distribution (fc3), and Microsoft's windows xp sp2 (herewith, referred to as windows), are readily available in the laboratory. fc3 and windows provide production quality IPv4 and IPv6 support [52][53]. Focus has been on using windows because student performing this work is familiar with it. fc3 is used when windows could not be used.

In both these systems, support for network QoS functions—for example, classification using filters, shaping, policing, marking, various scheduling algorithms, queue management algorithms—is available in the form of a traffic control module. This module includes a kernel -level module that implements QoS functions, application programming interface that application programs—programs that run in user mode as opposed to kernel mode—utilize to take advantage of kernel -level functionality, and ready -made tools, which ease the task of configuring the operating system with no need to modify the application programs. Windows supports an IPv4 traffic control module. It does not provide an IPv6 traffic control API and configuration tools. Status of kernel -level support for IPv6 traffic control functions in this operating system is unknown. It does not support intserv architecture with both IPv4 and IPv6. It supports diffserv architecture with IPv4. It does not provide an IPv6 enabled API and configuration tools for supporting diffserv architecture, as is the case with IPv4.

IPv6 enabled plain sockets API [29, Gilligan 1999] and advanced sockets API recommended by IETF [30, Stevens 2003] provides a way to specify a flow label value and traffic class value. Both APIs does not explicitly state how flow label value can be read using plain sockets. TCP/IP protocol stacks of both fc3 and windows do not actually read and write flow label value in the packet header using plain sockets. Windows does not support the IPv6 socket option to set and read traffic class. Cisco routers ignore flow label field and

Juniper routers provide a proprietary implementation for using flow label field [43, Banerjee 2005]. Windows does not support any of the three ways — socket options recommended by IETF, traffic control API and configuration tools—to set the traffic class field in the IPv6 base header. These are necessary to configure diffserv related functions. This support is available with IPv4. However, windows supports a non-standard IPv6 socket option that can be used to write the whole IPv6 packet, including base header with fields like traffic class and flow label. By using this feature and the layered service provider feature of windows networking architecture [31, Jones 2000][52][54], it is possible to write a configuration tool —fully in user mode as opposed to kernel-mode—, which will "trap" an application's UDP socket calls, and write flow label and traffic class fields without affecting the application. However, no way of reading the flow label value, and traffic class value could be found on windows. Because IPv6 IPsec support is implemented in the kernel, it is not clear how the user-mode flow label marking capability described above can be used to mark the outer IPv6 packet in an IPsec tunnel-mode configuration. Due to flexibility in the windows networking architecture, a certain application (for example, MGEN traffic generator [48]) may not load the service provider described above. To make such applications work, it becomes necessary to understand the application's logic for choosing the appropriate service provider, and possibly modify the application to make it work.

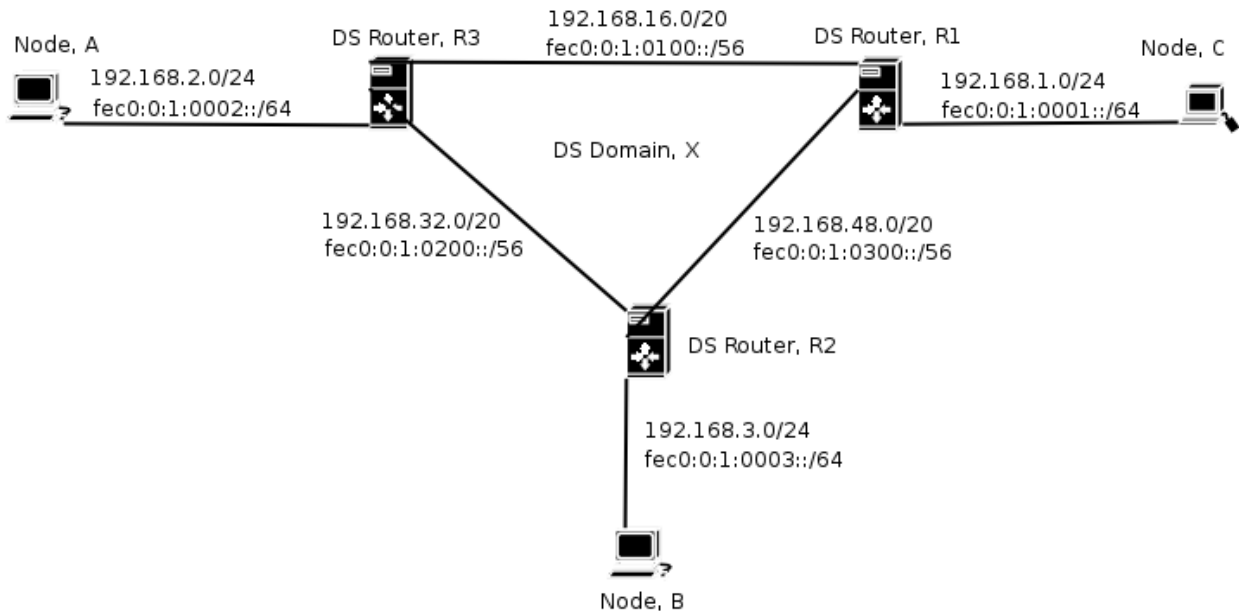
The approach of adding missing support to the operating systems and developing configuration tools, and then making the widely used network tools (like MGEN) work, has been found to be a time consuming approach (from the point of view of the student performing this work). For this reason, only the part of the IPv6-QoS enabled VPN scheme, which can be implemented by using readily available configuration tools, has actually been implemented in this work. In short, IPv6-QoS enabled VPN scheme has not been implemented for lack of support for flow labeling in general, and lack of support for marking of outer IPv6 packet—in an IPsec tunnel-mode configuration—with flow label, to be specific. An integral part of the IPv6-QoS enabled VPN scheme is an IPv6 inter-network that comprises of diffserv domains—each offering a real-time network service—interconnected with appropriate SLAs. This is required to ensure that elevated quality of network service is available to the traffic flowing through the VPN connection. A minimum of two diffserv domains are required to demonstrate the usefulness of flow label in the IPv6-QoS enabled VPN scheme. Due to lack of support of flow label, number of readily available computers and network cards required to setup two diffserv domains, and amount of work involved in setting up these computers, only one diffserv domain has been setup in the laboratory. From delay metric point of view, the primary difference between an IPv4 based implementation of real-time network service and its IPv6 counter-part in the laboratory environment is the larger size of IPv6 base header (20 octets). Since, this overhead is very small as compared to ethernet MTU (1500 octets), this difference is ignored and real-time network service is implemented and tested using IPv4. Icmp3 based software routers are used in the diffserv domain because it provides extensive support for IPv4/IPv6 enabled traffic control as compared to windows [32, Yin 2003][51].

Because tunneling and encryption is not implemented, the processing delay and computational load at routers is less. As flow label functionality is not implemented, *the advantages of using flow label with diffserv cannot be measured and demonstrated*. However, this does not dilute the advantages of

using flow label with diffserv for achieving end-to-end network quality of service for VPN connections between sites.

5.2 Network Setup and Test Scenario

Figures 10 and 11 show the network setup done in the laboratory. A and B are hosts, which run Windows XP SP2 operating system. C is a host, which runs FC3 operating system. R1, R2, and R3 are software routers, which run FC3 operating system. Each router has three 10/100 Mbps ethernet network interface cards with no quality of service support. Hosts-to-router links are 100 Mbps point-to-point links. Router-to-router links are 10 Mbps point-to-point links (reason for slow speed link is explained later). Private unicast IPv4 addresses and deprecated IPv6 unicast site-local addresses [] are used because of the convenience these address spaces offer in setting up an experimental network. Hosts use default routing to send traffic to the first-hop routers. Routers use static routes to direct traffic (reason is explained later). IPv6 auto-configuration is used on R1, R2, and R3 to provide network prefixes to the hosts.



All nodes have 1GHz processor and 128MB RAM.

Figure 10: Network diagram of setup in the laboratory

For implementation of real-time network service and diffserv domain, nodes in the above setup represent certain entities in a diffserv domain. Routers R1, R2, and R3 represent the core routers of the diffserv domain, X. In this small domain, these routers also double-up as the edge routers of the domain. Nodes A and C, represent edge routers of the access networks, N_A and N_C respectively. Node A generates real-time traffic micro-aggregate flow T_{AC} with token bucket specification $\langle R_{AC}, B_{AC} \rangle$ and classifier $\langle A_{NA}, 0x63802, A_{NC} \rangle$, where $0x63802$ is the flow label value. To make sure that N_A conforms to T_{AC} , N_A uses windows IPv4 traffic control to shape the outgoing traffic. R3

polices the traffic identified by $\langle A_{NA}, 0x63802, A_{NC} \rangle$ to check its conformance with T_{AC} .

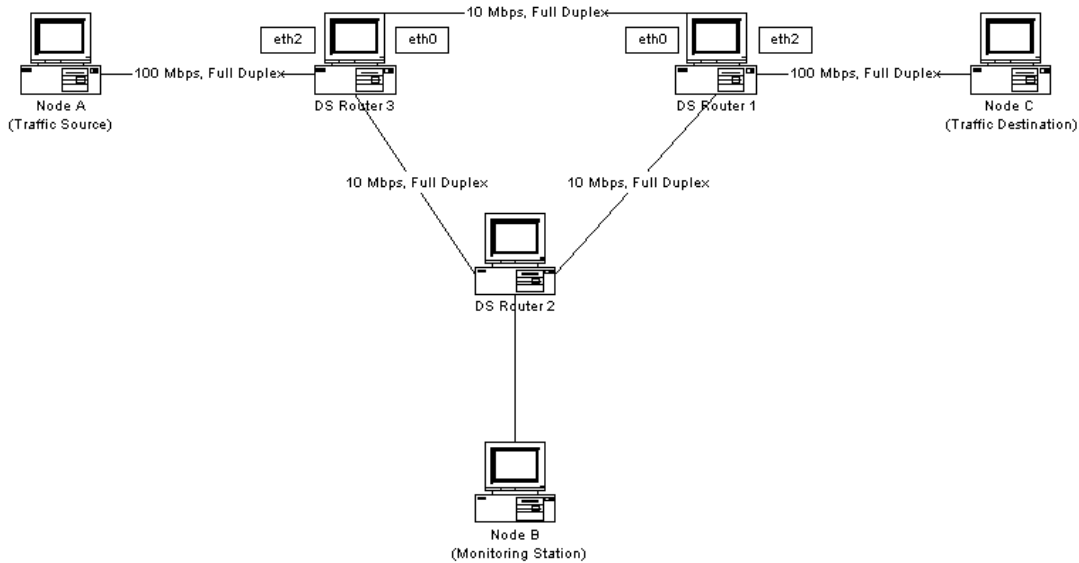


Figure 11: Network diagram of setup showing details of links

In order to test whether the real-time network service design as described in the previous chapter works, diffserv domain is configured to meet per-hop maximum delay of 63 msec (reason is explained later) for traffic profile, T_{AC} . Scripts are available in the Appendix A. T_{AC} is chosen as below:

$$R_{AC} = 3.250 \text{ Mbps}$$

$$B_{AC} = 52 \text{ full-sized ethernet frames}$$

Node B is used for controlling, and monitoring the setup, and data analysis.

5.3 Measurement Metrics, Tools, and Method

In the setup, each router uses priority packet scheduling and arrival rate of data is less than the service rate. There is sufficient bandwidth available to meet the requirements of all flows. Therefore, bandwidth used by flows is not measured. Other network performance parameters relevant to one-way video applications with interactivity requirements are one-way delay and packet loss. IETF has defined one-way delay (owd) metric [26, Almes 1999] and one-way packet loss metric [27, Almes 1999] for measuring these parameters in IP networks. Two reasons common to choice of all tools are that they are freely available and flexible. *OpenIMP* tool [49] is used to measure one-way delay because it follows the IETF measurement standards. *MGEN* tool [48] is used to generate traffic flows. Packet loss has been measured using *tc* tool [51] and *ifconfig* tool available in fedora core 3 distribution [50].

To test whether the actual one-way delay is close to the calculated maximum one-way delay, bursty traffic needs to be generated. To simulate a burst that can fill the router output queue quickly, arrival rate of traffic destined to a particular router's output queue needs to exceed the service rate of the output queue by a large factor. In the setup, speed of links in the diffserv domain is set lower than speed of access links. Source node generates the burst by transmitting at full link rate, which is ten times the service rate of the router's

queue. Because commodity software and hardware is used, number of packets generated during the burst is not constant. Shaping of the traffic at the source node helps in controlling this variation. In the setup, duration of burst is arbitrarily chosen because the aim is to verify the way the network service can be designed for a given burst size.

To measure one-way delay, time setup on the nodes is important. In the setup, NTP protocol is used to synchronize the clocks of all nodes. Time synchronization in the setup is far from perfect. The relative offset between the measurement nodes is in order of tens of microseconds, which increases to hundreds of microseconds every minute or so. Because commodity hardware and software is used, skew in the clocks is not negligible. Skew is controlled by synchronizing the clocks very often (every one second) at measurement nodes. NTP traffic is not routed through the measurement path, and so, it does not affect the measurements. Expected one-way delay for the measurement exercise is intentionally kept in tens of milliseconds. This is to make it possible to verify the one-way delay calculation with measurement. Since, the measurement metric is one-way delay—a time difference, relative offset between the measurement nodes is in order of tens or hundreds of microseconds, and expected value of one-way delay is in tens of milliseconds, the effect of poor synchronization in the clocks can be ignored.

To test whether the actual packet loss is close to the expected packet loss, number of packets dropped by the NIC and number of packets dropped at the real-time service queue is checked before start of the test run and after end of the test run. Due to limitation of the tool used to monitor packet loss, policing of traffic destined to real-time service queue is disabled during the test run.

To take measurements with stable network paths, diffserv domain uses static routing.

5.4 Test Results and Analysis

5.4.1 Per-Hop delay for Real-time service at R1

Negligible queuing delay is observed at R1 because the service rate of real-time service is 100 Mbps and peak arrival rate for this service can only be 10 Mbps.

5.4.2 Packet Loss

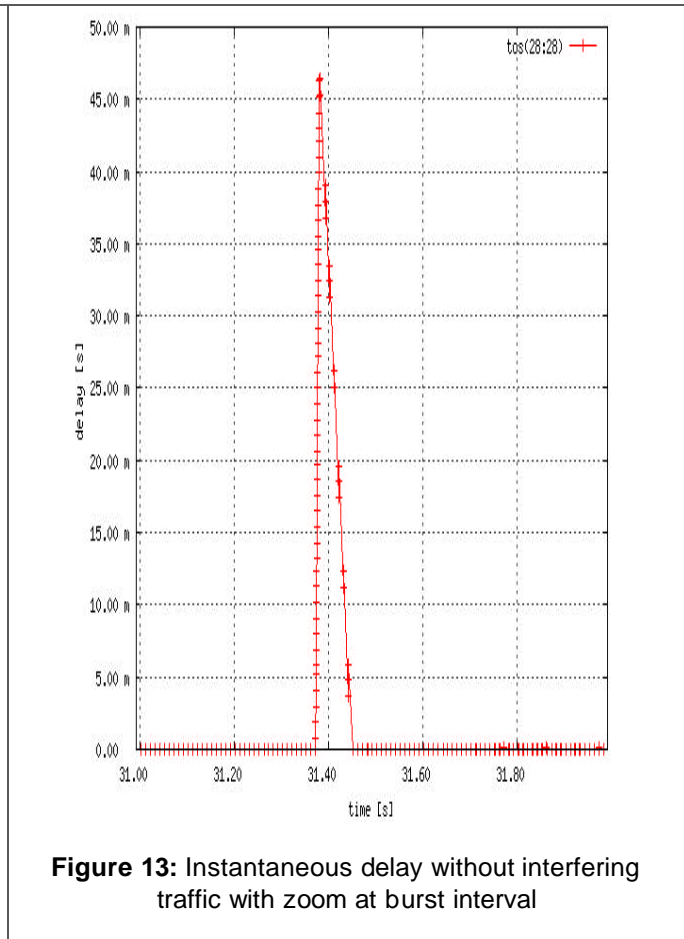
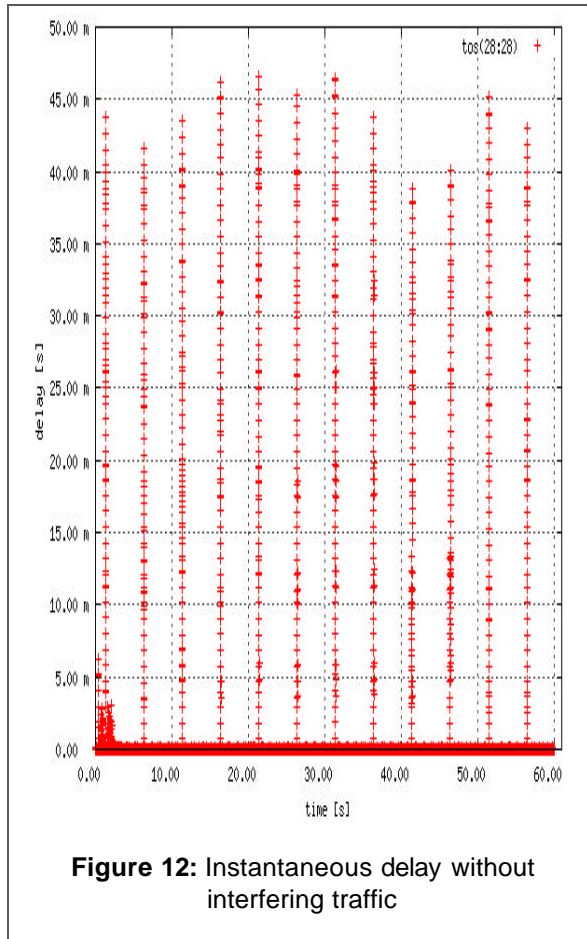
OpenIMP shows packet losses. However, no packet loss is reported by *tc* and *ifconfig* tool. *Tc* shows losses on a per-queue basis. *ifconfig* shows losses on per-card basis. However, packet drops by the policer on T_{AC} in R3 are not captured by *tc* tool. Therefore, policer on T_{AC} was removed and tests are re-run. Again, *OpenIMP* shows packet losses. However, no packet loss is reported by *tc* and *ifconfig*. It seems reasonable to conclude that measured packet loss is zero. Zero packet loss for real-time service queue is expected because network control traffic is minimal and the real-time traffic profile T_{AB} is shaped at source N_A .

5.4.3 Delay for Real-time Service at R3

Processing delay is observed to be much less than 1 millisecond. So, only transmission delay and queuing delay are considered in the analysis.

5.4.3.1 Queuing Delay without Interfering Traffic

Figure 12 shows that the queuing delay without any interfering traffic is well within the bound of 63 milliseconds. Figure 13 shows how the queuing delay suddenly jumps when a burst arrives from node A.



Bound on per-hop delay at R3 is calculated by assuming fixed transmission delay. At 10 Mbps, minimum transmission delay is expected to be 1.21 milliseconds. As will be seen later, this may not be the case when NIC card does not support QoS. Since node A takes at least 6.3 milliseconds to transmit the last bit to R3, R3 would have already transmitted packets for 6.3 milliseconds. Therefore, maximum queuing delay that can be encountered at R3 is 56.7 milliseconds ($= 63 - 6.3$). If node A takes longer, maximum queuing delay at R3 will further reduce. Point to note here is that in order to simulate a burst that will cause full maximum delay possible, node A needs to transmit much faster (100Mbps here) than service rate of real-time queue (10Mbps here).

5.4.3.2 Queuing Delay With Interfering Traffic

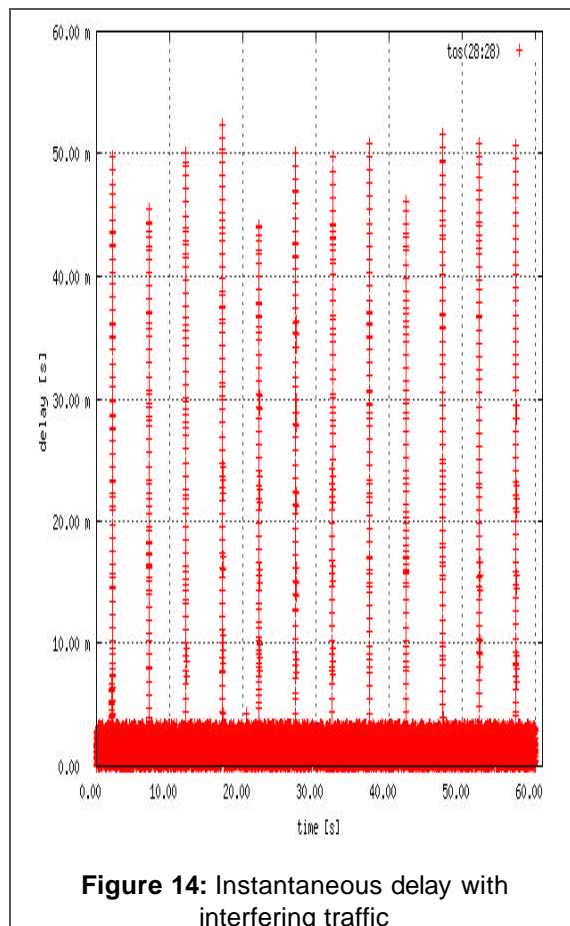


Figure 14: Instantaneous delay with interfering traffic

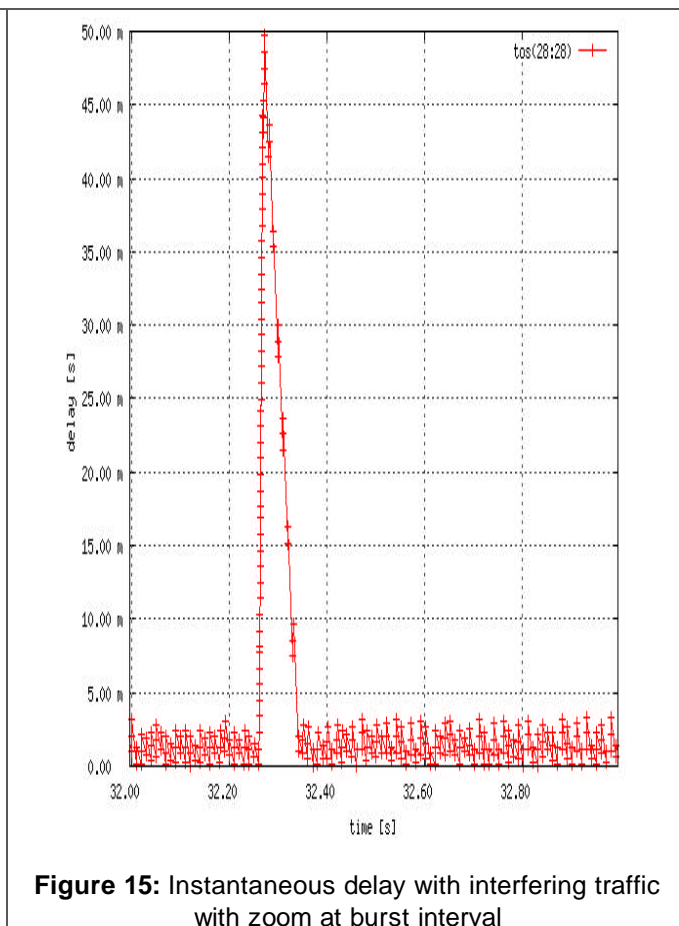


Figure 15: Instantaneous delay with interfering traffic with zoom at burst interval

Figure 14 shows that the queuing delay without any interfering traffic is well within the bound of 63 milliseconds. Figure 15 shows how the queuing delay suddenly jumps when a burst arrives from node A.

5.4.4 Effect of no QoS Support at Link Layer

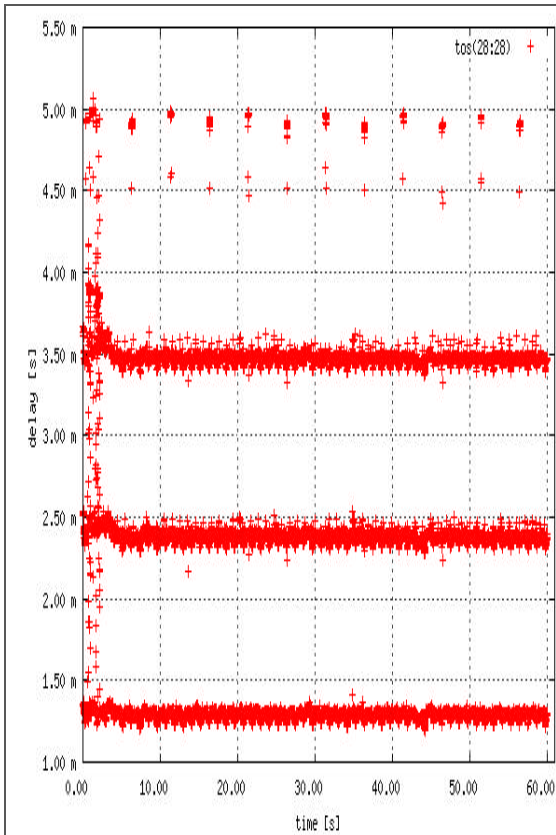


Figure 16: Instantaneous Transmission Delay between Routers R3 and R1 (No Interfering Traffic)

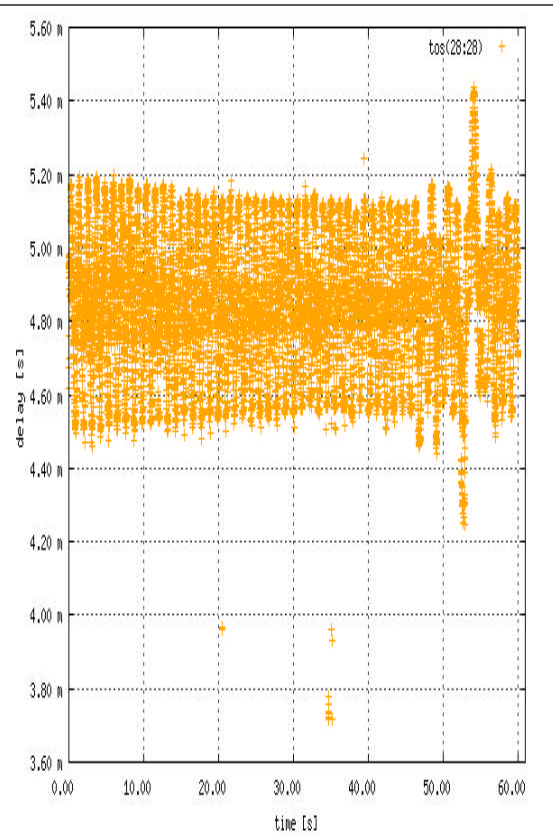


Figure 17: Instantaneous Transmission Delay between Routers R3 and R1 (Interfering Traffic)

Figures 16 and 17 show instantaneous transmission delays experienced by packets when transmitted by R3 and received by R1. As propagation delay is very negligible (cable length less than 3 meters) and processing delay is negligible (routers have 64 MB free memory and CPU utilization is much less than 100%), figures 16 and 17 show the affect of lack of QoS support at ethernet cards. Cards used in the routers do not have 802.1q support. At 10 Mbps, minimum transmission delay is expected to be 1.21 milliseconds.

High variability in transmission delay can cause per-hop delay at a router to cross the bound. Luckily, this has not been the case in this experiment, but this needs to be handled in order to provide a per-hop delay bound.

6. Discussion

For large-scale multimedia grids to meet performance requirements of the multimedia applications and scale geographically, IPv6 QoS is important. IP-based VPN connections are used to connect sites privately over the public Internet. Multimedia application's requirements from the network depend upon a number of factors, which can also be subjective. A particular set of standardized network services are not appropriate to serve the needs to variety of multimedia applications. It is necessary that end networks have the capability to inform all the internet service providers enroute to the destination network what network service they require. By using the immutable flow label field in IPv6 to carry access network's requirements and by integrating this field with the scalable diffserv architecture, IPv6-QoS enabled VPN connections between sites could be offered. These connections can form the building block for a achieving end-to-end quality of service over an IPv6-enabled public Internet. Building such a VPN connection in a laboratory environment is difficult today. However, building a diffserv domain, which can provide interactive one-way video applications an appropriate real-time network service, is possible in a laboratory environment.

Future work includes the following:

- * building at least one more diffserv domain that can support real-time network service, and connecting it with existing diffserv domain by using inter-domain SLA negotiation protocols,
- * enabling quality of service in access networks by deploying intserv architecture,
- * integrating intserv-enabled access networks with interconnected diffserv domains by deploying intserv-over-diffserv architecture,
- * adding support on commodity operating systems for flow label marking on VPN connections.

Appendix A: Scripts Used in Setup

A.1 Traffic Generation Script Design

To generate T_{AB} ($R_{AB} = 3.250$ Mbps, $B_{AB} = 52$ full-sized ethernet frames), 3.0 Mbps constant rate traffic is added with bursts, regularly at every 5 seconds, at 100 Mbps of duration 6.3 milliseconds. MGEN tool generates this traffic from node A and receive it at node C. Node A uses traffic control feature of Microsoft Windows XP SP2 to shape this traffic to the mentioned profile.

To generate interfering traffic, 2 Mbps UDP data is injected from node A to node C using MGEN tool. Additionally, FTP file transfer from node A to node C of file sized over 1GB is started at the same time.

A.2 Scripts for Node A

MGEN script to send interfering UDP traffic

```
# 165 1514 bytes sized frames means 2.0 Mbps
0.0 ON 2 UDP DST 192.168.1.2/5009 PERIODIC [165 1472]

# terminate the flow 600.0 sec after script execution
3600000.0 OFF 2
```

MGEN script to simulate video UDP traffic

```
0.0 ON 3 UDP DST 192.168.1.2/5008 PERIODIC [258 1472]
5.0 ON 4 UDP DST 192.168.1.2/5008 BURST [REGULAR 5.0 PERIODIC [7998 1472]
FIXED 0.0063]

3600000.0 OFF 3
3600000.0 OFF 4
```

A.3 Scripts for Node C

MGEN script to receive simulated video and interfering UDP traffic

```
# Monitor some ports for UDP traffic
0.0 LISTEN UDP 5007-5012

# stop listening
3600000.0 IGNORE UDP 5007-5012
```

FTP commands to pull 1.2 GB file from node A to node C is not available

A.4 Scripts for Router 3

Setting Ethernet card

```
ethtool -s eth1 speed 10 duplex full autoneg off
ethtool -s eth0 speed 10 duplex full autoneg off
```

/etc/sysconfig/network

```
NETWORKING=yes
FORWARD_IPV4=yes
```

```
HOSTNAME=qosrouter3
```

```
IPV6INIT=yes  
NETWORKING_IPV6=yes  
IPV6FORWARDING=yes
```

```
/etc/sysconfig/network-scripts/ifcfg-eth2
```

```
# Accton Technology Corporation SMC2-1211TX  
DEVICE=eth2  
ONBOOT=yes  
#BOOTPROTO=dhcp  
HWADDR=00:10:B5:EE:87:E9  
IPADDR=192.168.2.1  
NETMASK=255.255.255.0
```

```
IPV6INIT=yes  
#IPV6_AUTOCONF=yes  
IPV6ADDR=fec0:0:1:2::1/64
```

```
/etc/sysconfig/network-scripts/ifcfg-eth1
```

```
# Accton Technology Corporation SMC2-1211TX  
DEVICE=eth1  
ONBOOT=yes  
BOOTPROTO=static  
HWADDR=00:10:B5:A4:09:A5  
IPADDR=192.168.32.1  
NETMASK=255.255.240.0
```

```
IPV6INIT=yes  
#IPV6_AUTOCONF=yes  
IPV6ADDR=fec0:0:1:200::1/56
```

```
/etc/sysconfig/network-scripts/ifcfg-eth0
```

```
DEVICE=eth0  
ONBOOT=yes  
#BOOTPROTO=dhcp  
IPADDR=192.168.16.1  
NETMASK=255.255.240.0
```

```
IPV6INIT=yes  
#IPV6_AUTOCONF=yes  
IPV6ADDR=fec0:0:1:100::1/56
```

```
/etc/sysconfig/network-scripts/route-eth1
```

```
192.168.3.0/24 via 192.168.32.2  
fec0:0:1:3::/64 via fec0:0:1:200::2  
fec0:0:1:300::2/128 via fec0:0:1:200::2
```

```
/etc/sysconfig/network-scripts/route-eth0
```

```
192.168.1.0/24 via 192.168.16.2  
fec0:0:1:1::/64 via fec0:0:1:100::2  
fec0:0:1:300::1/128 via fec0:0:1:100::2
```

```
Router Advertisement Setting
```

```
interface eth2  
{  
    AdvSendAdvert on;
```

```

MinRtrAdvInterval 20;
MaxRtrAdvInterval 30;
AdvLinkMTU 1500;
AdvCurHopLimit 255;
AdvManagedFlag off;
AdvOtherConfigFlag off;
UnicastOnly off;
AdvSourceLLAddress off;
prefix fec0:0:1:2::/64 {
    AdvOnLink on;
    AdvAutonomous on;
    AdvValidLifetime 604800;
    AdvPreferredLifetime 601200;
};
};

interface eth1
{
    AdvSendAdvert on;
};

interface eth0
{
    AdvSendAdvert on;
};

```

TC Script for eth0

```

#queue setup
#-----
#bands are numbered 0, 1, 2 and corresponding classes are numbered :1, :2 and :3
tc qdisc add dev eth0 root handle 1: prio \
    bands 4 priomap 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

#only for arp. do not remark. kernel gives error messages and tuns ppl off
tc qdisc add dev eth0 parent 1:1 handle 20: pfifo limit 5

tc qdisc add dev eth0 parent 1:2 handle 3:0 dsmark indices 2 default_index 1
#remark
tc class change dev eth0 classid 3:1 dsmark mask 0x03 value 0x3c
#set the queue, kbit in kilo bit/s, burst, limit in bytes
tc qdisc add dev eth0 parent 3:1 handle 30: pfifo limit 10

tc qdisc add dev eth0 parent 1:3 handle 4:0 dsmark indices 2 default_index 1
#remark to new dscp code
tc class change dev eth0 classid 4:1 dsmark mask 0x3 value 0x1c
#set the queue for aggregate flow (rate in kilobit/s, burst, limit in bytes).
tc qdisc add dev eth0 parent 4:1 handle 40: tbf rate 3270kbit burst 83270
limit 83270 mpu 64

tc qdisc add dev eth0 parent 1:4 handle 5:0 dsmark indices 2 default_index 1
#remark to zero
tc class change dev eth0 classid 5:1 dsmark mask 0x03 value 0x00

```

```
tc qdisc add dev eth0 parent 5:1 handle 50: pfifo limit 1000

#filters setup
#-----
#####filter for arp queue#####
#arp (will cause error becoz dsmark will try to mark arp packet)
tc filter add dev eth0 parent 1:0 protocol 0x806 prio 1 u32 \
    match u8 0x00 0x00 at 0 flowid 1:1

#####filters for network control queue#####
#ntp-ip-udp-sport
tc filter add dev eth0 parent 1:0 protocol ip prio 2 u32 \
    match u8 0x11 0xff at 9 \
    match u16 0x007b 0xffff at 20 \
    flowid 1:2
#ntp-ip-udp-dport
tc filter add dev eth0 parent 1:0 protocol ip prio 3 u32 \
    match u8 0x11 0xff at 9 \
    match u16 0x007b 0xffff at 22 \
    flowid 1:2
#ntp-ip-tcp-dport
tc filter add dev eth0 parent 1:0 protocol ip prio 4 u32 \
    match u8 0x06 0xff at 6 \
    match u16 0x007b 0xffff at 22 \
    flowid 1:2
#ntp-ipv6-udp-sport
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 5 u32 \
    match u8 0x11 0xff at 6 \
    match u16 0x007b 0xffff at 40 \
    flowid 1:2
#ntp-ipv6-udp-dport
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 6 u32 \
    match u8 0x11 0xff at 6 \
    match u16 0x007b 0xffff at 42 \
    flowid 1:2
#ntp-ipv6-tcp-dport
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 7 u32 \
    match u8 0x06 0xff at 6 \
    match u16 0x007b 0xffff at 42 \
    flowid 1:2

#snmp-ip-udp-sport
tc filter add dev eth0 parent 1:0 protocol ip prio 8 u32 \
    match u8 0x11 0xff at 9 \
    match u16 0x00a1 0xffff at 20 \
    flowid 1:2
#snmp-ip-udp-dport
tc filter add dev eth0 parent 1:0 protocol ip prio 9 u32 \
    match u8 0x11 0xff at 9 \
    match u16 0x00a1 0xffff at 22 \
    flowid 1:2
#snmp-ip-tcp-dport
tc filter add dev eth0 parent 1:0 protocol ip prio 10 u32 \
    match u8 0x06 0xff at 6 \
    match u16 0x00a1 0xffff at 22 \
    flowid 1:2
```

```
#snmp-ipv6-udp-sport
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 11 u32 \
    match u8 0x11 0xff at 6 \
    match u16 0x00a1 0xffff at 40 \
    flowid 1:2
#snmp-ipv6-udp-dport
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 12 u32 \
    match u8 0x11 0xff at 6 \
    match u16 0x00a1 0xffff at 42 \
    flowid 1:2
#snmp-ipv6-tcp-dport
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 13 u32 \
    match u8 0x06 0xff at 6 \
    match u16 0x00a1 0xffff at 42 \
    flowid 1:2

#snmp-trap-ip-udp-sport
tc filter add dev eth0 parent 1:0 protocol ip prio 14 u32 \
    match u8 0x11 0xff at 9 \
    match u16 0x00a2 0xffff at 20 \
    flowid 1:2
#snmp-trap-ip-udp-dport
tc filter add dev eth0 parent 1:0 protocol ip prio 15 u32 \
    match u8 0x11 0xff at 9 \
    match u16 0x00a2 0xffff at 22 \
    flowid 1:2
#snmp-trap-ip-tcp-dport
tc filter add dev eth0 parent 1:0 protocol ip prio 16 u32 \
    match u8 0x06 0xff at 6 \
    match u16 0x00a2 0xffff at 22 \
    flowid 1:2
#snmp-trap-ipv6-udp-sport
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 17 u32 \
    match u8 0x11 0xff at 6 \
    match u16 0x00a2 0xffff at 40 \
    flowid 1:2
#snmp-trap-ipv6-udp-dport
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 18 u32 \
    match u8 0x11 0xff at 6 \
    match u16 0x00a2 0xffff at 42 \
    flowid 1:2
#snmp-trap-ipv6-tcp-dport
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 19 u32 \
    match u8 0x06 0xff at 6 \
    match u16 0x00a2 0xffff at 42 \
    flowid 1:2

#####filters for real-time service queue#####
#do not police icmp as this might increase measured delay
tc filter add dev eth0 parent 1:0 protocol ip prio 20 u32 \
    match u8 0x01 0xff at 9 \
    flowid 1:3

#do not police icmp as this might increase measured delay
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 21 u32 \
    match u8 0x3a 0xff at 6 \
    flowid 1:3
```

```
tc filter add dev eth0 parent 1:0 protocol ip prio 23 u32 \  
    match ip src 192.168.2.0/16 \  
    match ip dst 192.168.1.0/16 \  
    match u16 0x001c 0x00fc at 0 \  
        police rate 3250kbit burst 80242 drop \  
    flowid 1:3  
  
#####filters for best effort queue#####  
tc filter add dev eth0 parent 1:0 protocol ip prio 31 u32 \  
    match u8 0x40 0xf0 flowid 1:4  
  
tc filter add dev eth0 parent 1:0 protocol ipv6 prio 32 u32 \  
    match u8 0x60 0xf0 flowid 1:4
```

A.5 Scripts for Router 1

Setting Ethernet card

```
ethtool -s eth1 speed 10 duplex full autoneg off  
ethtool -s eth0 speed 10 duplex full autoneg off
```

/etc/sysconfig/network

```
HOSTNAME=qosrouter1  
NETWORKING=yes  
FORWARD_IPV4=yes
```

```
IPV6INIT=yes  
NETWORKING_IPV6=yes  
IPV6FORWARDING=yes
```

/etc/sysconfig/network-scripts/ifcfg-eth2

```
# Accton Technology Corporation SMC2-1211TX  
DEVICE=eth2  
ONBOOT=yes  
BOOTPROTO=static  
HWADDR=00:10:B5:EE:83:94  
IPADDR=192.168.1.1  
NETMASK=255.255.255.0
```

```
IPV6INIT=yes  
#IPV6_AUTOCONF=no  
IPV6ADDR=fec0:0:1:1::1/64
```

/etc/sysconfig/network-scripts/ifcfg-eth1

```
# Accton Technology Corporation SMC2-1211TX  
DEVICE=eth1  
ONBOOT=yes  
BOOTPROTO=static  
HWADDR=00:10:B5:A3:F1:A6  
IPADDR=192.168.48.1  
NETMASK=255.255.240.0
```

```
IPV6INIT=yes  
#IPV6_AUTOCONF=yes  
IPV6ADDR=fec0:0:1:300::1/56
```

/etc/sysconfig/network-scripts/ifcfg-eth0

```
# Accton Technology Corporation SMC2-1211TX
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
HWADDR=00:10:B5:EE:89:A8
IPADDR=192.168.16.2
NETMASK=255.255.240.0
#GATEWAY=192.168.192.254
```

```
IPV6INIT=yes
#IPV6_AUTOCONF=yes
IPV6ADDR=fec0:0:1:100::2/56
```

/etc/sysconfig/network-scripts/route-eth1

```
192.168.3.0/24 via 192.168.48.2
fec0:0:1:3::/64 via fec0:0:1:300::2
fec0:0:1:200::2/128 via fec0:0:1:300::2
192.168.32.2/32 via 192.168.48.2
```

/etc/sysconfig/network-scripts/route-eth0

```
192.168.2.0/24 via 192.168.16.1
fec0:0:1:2::/64 via fec0:0:1:100::1
fec0:0:1:200::1/128 via fec0:0:1:100::1
192.168.32.1/32 via 192.168.16.1
```

Router Advertisement Setting

```
interface eth1 {
    AdvSendAdvert on;
};

interface eth2 {
    AdvSendAdvert on;
    MinRtrAdvInterval 3;
    MaxRtrAdvInterval 10;
    AdvLinkMTU 1500;
    AdvManagedFlag off;
    AdvOtherConfigFlag off;
    UnicastOnly off;
    AdvCurHopLimit 255;
    AdvSourceLLAddress off;
    prefix fec0:0:1:1::/64 {
        AdvOnLink on;
        AdvAutonomous on;
        AdvValidLifetime 604800;
        AdvPreferredLifetime 601200;
    };
};

interface eth0 {
    AdvSendAdvert on; #sawan
};
```

TC Script for eth2

```
#queue setup
```

```

#-----
#bands are numbered 0, 1, 2 and corresponding classes are numbered :1, :2 and
:3
tc qdisc add dev eth2 root handle 1:0 prio \
    bands 4 priomap 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

#only for arp. do not remark. kernel gives error messages and turns ppl off
#set the queue, kbit in kilo bit/s, burst, limit in bytes (=1 frame size)
tc qdisc add dev eth2 parent 1:1 handle 20: pfifo limit 5

#set the queue, kbit in kilo bit/s, burst, limit in bytes (=10 frame size)
tc qdisc add dev eth2 parent 1:2 handle 30: pfifo limit 10

#queue for aggregate flow
tc qdisc add dev eth2 parent 1:3 handle 40: tbf rate 3270kbit burst 83270
limit 83270 mpu 64

#set red queue (burst = 347*1000 bytes)
tc qdisc add dev eth2 parent 1:4 handle 50: pfifo limit 1000

#filters setup
#-----
#####filter for arp queue#####
#arp (will cause error becoz dsmark will try to mark arp packet)
tc filter add dev eth2 parent 1:0 protocol 0x806 prio 1 u32 \
    match u8 0x00 0x00 at 0 flowid 1:1

#####filters for network control queue#####
tc filter add dev eth2 parent 1:0 protocol ip prio 2 u32 \
    match u16 0x003c 0x00fc at 0 \
    flowid 1:2

tc filter add dev eth2 parent 1:0 protocol ipv6 prio 3 u32 \
    match u16 0x03c0 0x0fc0 at 0 \
    flowid 1:2

#####filters for real-time service queue#####
#do not police icmp as this might increase measured delay
tc filter add dev eth2 parent 1:0 protocol ip prio 10 u32 \
    match u8 0x01 0xff at 9 \
    flowid 1:3

#do not police icmp as this might increase measured delay
tc filter add dev eth2 parent 1:0 protocol ipv6 prio 11 u32 \
    match u8 0x3a 0xff at 6 \
    flowid 1:3

tc filter add dev eth2 parent 1:0 protocol ip prio 12 u32 \
    match u16 0x001c 0x00fc at 0 \
    flowid 1:3

tc filter add dev eth2 parent 1:0 protocol ipv6 prio 13 u32 \
    match u16 0x01c0 0x0fc0 at 0 \
    flowid 1:3

#####filters for best effort queue#####

```

```
tc filter add dev eth2 parent 1:0 protocol ip prio 21 u32 \  
    match u8 0x40 0xf0 flowid 1:4
```

```
tc filter add dev eth2 parent 1:0 protocol ipv6 prio 22 u32 \  
    match u8 0x60 0xf0 flowid 1:4
```

A.6 Scripts for Router 2

Setting Ethernet card

```
ethtool -s eth1 speed 10 duplex full autoneg off  
ethtool -s eth0 speed 10 duplex full autoneg off
```

/etc/sysconfig/network

```
NETWORKING=yes  
HOSTNAME=qosrouter2  
IPV4_FORWARD=yes
```

```
IPV6INIT=yes  
NETWORKING_IPV6=yes  
IPV6FORWARDING=yes
```

/etc/sysconfig/network-scripts/ifcfg-eth2

```
DEVICE=eth2  
BOOTPROTO=dhcp  
ONBOOT=yes  
TYPE=Ethernet  
BOOTPROTO=static  
DHCP_HOSTNAME=qosrouter2  
IPADDR=192.168.3.1  
NETMASK=255.255.255.0
```

```
IPV6INIT=yes  
#IPV6_AUTOCONF=yes  
IPV6ADDR=fec0:0:1:3::1/64
```

/etc/sysconfig/network-scripts/ifcfg-eth1

```
DEVICE=eth1  
BOOTPROTO=static  
ONBOOT=yes  
TYPE=Ethernet  
#DHCP_HOSTNAME=qosrouter2  
HOSTNAME=qosrouter2  
IPADDR=192.168.48.2  
NETMASK=255.255.240.0
```

```
IPV6INIT=yes  
#IPV6_AUTOCONF=yes  
IPV6ADDR=fec0:0:1:300::2/56
```

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0  
BOOTPROTO=static  
#BOOTPROTO=dhcp  
ONBOOT=yes  
TYPE=Ethernet  
#DHCP_HOSTNAME=qosrouter2
```

```
HOSTNAME=qosrouter2
IPADDR=192.168.32.2
NETMASK=255.255.240.0
```

```
IPV6INIT=yes
#IPV6_AUTOCONF=yes
IPV6ADDR=fec0:0:1:200::2/56
```

```
/etc/sysconfig/network-scripts/route-eth1
192.168.1.0/24 via 192.168.48.1
fec0:0:1:1::/64 via fec0:0:1:0300::1
fec0:0:1:0100::2/128 via fec0:0:1:0300::1
192.168.16.2/32 via 192.168.48.1
```

```
/etc/sysconfig/network-scripts/route-eth0
192.168.2.0/24 via 192.168.32.1
192.168.16.1/32 via 192.168.32.1
fec0:0:1:2::/64 via fec0:0:1:0200::1
fec0:0:1:0100::1/128 via fec0:0:1:0200::1
```

Router Advertisement Setting

```
interface eth1 {
    AdvSendAdvert on;
};

interface eth2 {
    AdvSendAdvert on;
    MinRtrAdvInterval 3;
    MaxRtrAdvInterval 10;
    AdvLinkMTU 1500;
    AdvManagedFlag off;
    AdvOtherConfigFlag off;
    UnicastOnly off;
    AdvCurHopLimit 255;
    AdvSourceLLAddress off;
    prefix fec0:0:1:3::/64 {
        AdvOnLink on;
        AdvAutonomous on;
        AdvValidLifetime 604800;
        AdvPreferredLifetime 601200;
    };
};

interface eth0 {
    AdvSendAdvert on; #sawan
};
```

References

- [1] I. Foster, C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan-Kaufman, chap. 2, 1999.
- [2] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International J. Supercomputer Applications*, vol. 15, no. 3, pp. 200-222, 2001.
- [3] I. Foster, "What is the Grid? A Three Point Checklist," *Grid Today*, vol. 1, no. 6, 2002.
- [4] W. Gentsch, "Response to Ian Foster's What is the Grid?," *Grid Today*, vol. 1, no. 8, 2002.
- [5] W. E. Johnston, "A different perspective on the question of what is a grid?," *Grid Today*, vol. 1, no. 8, 2002.
- [6] "Grid Computing — Today and Tomorrow: Another view," *Grid Today*, vol. 1, no. 9, 2002.
- [7] R.F. Freund, "How does one really characterize grid computing?," *Grid Today*, vol. 1, no. 10, 2002.
- [8] R. Banerjee, "Grid-One: An IPv6-QoS-aware Grid Computing Architecture," *EuroIndia 2004 Summit*, Mar. 2004.
- [9] J. Joseph, M. Ernest, C. Fellenstein, "Evolution of grid computing architecture and grid adoption models," *IBM Systems Journal*, vol. 43, no. 4, pp. 624, 2004.
- [10] A. Ogawa et al., "Design and Implementation of DV based video over RTP," *Packet Video 2000*, May 2000.
- [11] N.K. Sharda, *Multimedia Information Networking*, Prentice-Hall, 2002.
- [12] T. Akiyama et al., "Telecontrol of Ultra-High Voltage Electron Microscope over Global IPv6 Network," *Applications and the Internet Workshops*, pp.184 – 187, 27-31 Jan. 2003.
- [13] D. Lee et al., "Global Telescience featuring IPv6 at iGrid2002," *Future Generation Computer Systems*, vol. 19, no. 6, pp. 1031-1039, Aug. 2003.
- [14] S. Deering, R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, IETF RFC 2460, Dec. 1998; <http://www.rfc-editor.org/rfc/rfc2460.txt>.
- [15] D.E. Comer, *Internetworking with TCP/IP Vol 1: Principles, Protocols and Architectures*, Prentice Hall of India, 2002.
- [16] J. Davies, *Understanding IPv6*, Microsoft Press, 2003.
- [17] K. Nichols et al., *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, IETF RFC 2474, Dec. 1998; <http://www.rfc-editor.org/rfc/rfc2474.txt>.
- [18] S. Blake et al., *An Architecture for Differentiated Services*, IETF RFC 2475, Dec. 1998; <http://www.rfc-editor.org/rfc/rfc2475.txt>.
- [19] D. Blacket et al., *Per Hop Behavior Identification Codes*, IETF RFC 3140, June 2001; <http://www.rfc-editor.org/rfc/rfc3140.txt>.
- [20] D. Grossman, *New Terminology and Clarifications for Diffserv*, IETF RFC 3260, Apr. 2002; <http://www.rfc-editor.org/rfc/rfc3260.txt>.

References

- [21] S. Jha, M. Hassan, *Engineering Internet QoS*, Artech House, 2002.
- [22] El-Bahlul Fgee et al., "Implementing QoS capabilities in IPv6 networks and comparison with MPLS and RSVP, " *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 851-854, 4-7 May 2003.
- [23] X. Tang et al., "QoS Provisioning Using IPv6 Flow Label In the Internet, " *Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications & Signal Processing and Fourth Pacific-Rim Conference on Multimedia*, vol. 2, pp. 1253 - 1257, 15-18 Dec. 2003.
- [24] El-Bahlul Fgee et al., "Implementing an IPv6 QoS management scheme using flow label & class of service fields, " *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 1049 -1052, 2-5 May 2004.
- [25] J. Rajahalme et al., *IPv6 Flow Label Specification*, IETF RFC 3697, Mar. 2004;
<http://www.rfc-editor.org/rfc/rfc3697.txt> .
- [26] A. Almes et al., *A Oneway Delay Metric for IPPM*, IETF RFC 2679, Sep. 1999;
<http://www.rfc-editor.org/rfc/rfc2679.txt>.
- [27] A. Almes et al., *A Oneway Packet Loss Metric for IPPM*, IETF RFC 2680, Sep. 1999;
<http://www.rfc-editor.org/rfc/rfc2680.txt>.
- [28] K. Thompson, G. J. Miller, R. Wilder, "Wide-area Internet traffic patterns and characteristics, " *IEEE Network*, vol. 11, no. 6, pp. 10-23, Nov./Dec. 1997.
- [29] R. Gilligan et al., *Basic Socket Interface Extensions for IPv6*, IETF RFC 2553, Mar. 1999;
<http://www.rfc-editor.org/rfc/rfc2553.txt> .
- [30] W. Stevens et al., *Advanced Sockets Application Program Interface (API) for IPv6*, IETF RFC 3542, May 2003; <http://www.rfc-editor.org/rfc/rfc3542.txt> .
- [31] A. Jones, J. Ohlund, *Network Programming for Microsoft Windows*, Microsoft Press, 2000.
- [32] F. Yin, "IPv6 Diffserv Study and Test Report, " *STC Internal Report*, Jan. 2003.

References to Work in Progress

- [33] *SETI@Home* home page; <http://setiweb.ssl.berkeley.edu/>
- [34] *World Community Grid* home page; <http://www.worldcommunitygrid.org/>
- [35] *AccessGrid* home page; <http://www.accessgrid.org>
- [36] *Sloan Digital Sky Survey (SDSS)* home page; <http://cas.sdss.org/dr3/en/>
- [37] *TerraServer-USA* home page; <http://www.teraservice.net/>
- [38] *GRid And Next GEneration NETwork (GRANGENET)* home page, Apr. 2002; <http://www.grangenet.net/about/services.html>
- [39] *TeraGrid* home page; <http://www.teragrid.org/about/>
- [40] *Sun Powers the Grid: An Overview of Grid Computing from Sun*, SUN Microsystems; <http://www.sun.com/software/grid/powerofgrid.pdf>
- [41] *Grid Technology - Overview*, SUN Microsystems; <http://www.sun.com/software/grid/overview.xml>
- [42] D. Miras et al., *Network QoS Needs of Advanced Internet Applications - A Survey*, Internet2 QoS working group draft, work in progress, Nov. 2002.
- [43] A. Conta, B. Carpenter, *A proposal for the IPv6 Flow Label Specification*, Internet draft, work in progress, July 2001.
- [44] R. Banerjee, S.P. Malhotra, M. Mahaveer, *A Modified Specification for use of the IPv6 Flow Label for providing an efficient Quality of Service using a hybrid approach*, Internet draft, work in progress, Apr. 2002.
- [45] B. Teitelbaum and S. Shalunov, *Why Premium IP Service Has Not Deployed (and Probably Never Will)*, Internet2 QoS working group draft, work in progress, May 2002.
- [46] IPv6 and IPng mailing list; <http://www.atm.tut.fi/list-archive/ipng-2005/thrd4.html>
- [47] *Discussions with Prof. Rahul Banerjee*, Feb. 2005 - July 2005.
- [48] *MGEN-The Multi-Generator Toolset* homepage; <http://mgen.pf.itd.nrl.navy.mil/>
- [49] *6QM - IPv6 QoS Measurement* webpage; <http://www.6qm.org/downloads.php>
- [50] *Fedora Project* homepage; <http://fedora.redhat.com/>
- [51] B. Hubert et al., *Linux Advanced Routing & Traffic Control HOWTO* home page; <http://www.lartc.org/>
- [52] *Microsoft Developer Network (MSDN) Library* homepage; <http://msdn.microsoft.com/library/>
- [53] *Microsoft TechNet* homepage; <http://technet.microsoft.com/>
- [54] *MSDN Magazine* homepage; <http://msdn.microsoft.com/msdnmag/>