Content-Aware Congestion Control over MPLS Networks for Multimedia Transmission

by

Muhammad Rehan Sami

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

In Partial Fulfillment of the
Requirements for the degree

MASTER OF SCIENCE
IN

COMPUTER NETWORKING

King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia

January 2005

King Fahd University of Petroleum & Minerals
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Muhammad Rehan Sami** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER NETWORKING**.

Thesis Committee

_____
Dr. Abdul Waheed (Chairman)

_____
Dr. Muhammad Sadiq Sait (Member)

_____
Dr. Muhammad Farrukh Khan
(Member)

_____
Dr. Muhammad Sadiq Sait
(Department Chairman)

_____
Dr. Mohammad A. Al-Ohali
(Dean of Graduate Studies)

_____
Date

Dedications

I dedicate this work to my parents, for their endless love and prayers for my success, to my fiancée for her companionship, support and patience and to all who love me.

Acknowledgments

All thanks are due Allah first and foremost for His countless blessings. Acknowledgement is due to KFUPM and the COE Department for supporting this research.

I was honored and privileged to have Dr. Abdul Waheed as my advisor. I feel deeply indebted to him for shaping this research with his extensive knowledge and experience. I am thankful to him for bearing with patience my every mistake, correcting it and encouraging at every step. I also wish to thank my thesis committee members, Dr. Sadiq Sait and Dr. Farrukh Khan for their help, support, and contributions.

My thankful admiration goes to my colleague, peer researcher Itrat for positively contributing to my research through constructive criticism and discussions. I specially thank James R. Leu, developer of MPLS-Linux, and Hung-ying Tyan, developer of J-Sim, for their expert advice. I would like to express my gratitude towards Ali Mazhar for his efforts in helping me during the course of this work. Thanks to all my friends and colleagues at KFUPM who provided wholesome companionship throughout my studies at KFUPM. Above all I thank my family for all their support and love throughout my MS studies.

Table of Contents

Acknowledgments iv
List of Tables x
List of Figures xi
Thesis Abstract xiv
TU خلاصة الرسالة xv
Chapter 1 Introduction 1
1.1 Content Service Model 3
1.2 Discrete Wavelet Transform based Compression 4
1.3 Multiprotocol Label Switching 5
1.3.1 Sequence of MPLS Operations 7
1.3.2 MPLS Forwarding Information Base 8
1.3.3 Traffic Aggregation 9
1.3.4 Traffic Engineering 10
1.4 Advantages of MPLS 11
1.5 Quantitative QoS Evaluation 12
1.5.1 Bandwidth 12
1.5.2 End-to-end Delay and Delay Jitter 13
1.5.3 Packet Loss 14
1.6 Problem Statement 14
1.7 Practicality of Proposed Scheme 15
1.8 Contributions 16
1.9 Organization of Thesis 16
1.10 Summary 17
Chapter 2 Literature Survey 18
2.1 Compression using DWT 18
2.2 Congestion Control in MPLS Networks 22
2.3 Using MPLS-TE to Improve QoS 24
2.4 QoS Approach in Ipv4 Networks 27

2.5 Summary 28
Chapter 3 Problem Definition and Solution Methodology 29
3.1 Problem Definition 29
3.2 Solution Methodology 30
3.2.1 Bit Mapping and Quality of Service Routing 30
3.2.2 End-to-end Flow 32
3.2.3 Packet Identification using RTP 33
3.2.4 Packet order and Prioritization 34
3.2.5 EXP bits Encoding Scheme 35
3.2.6 MPLS Router Operations 36
3.2.7 Buffer Management at Router 37
3.3 Summary 38
Chapter 4 Simulation and Analysis 39
4.1 J-Sim Network Simulator 39
4.1.1 The J-Sim Autonomous Component Architecture 39
4.1.2 J-Sim Network Modeling and Simulation 40
4.1.3 The J-Sim Core Service Layer (CSL) 41
4.2 MPLS Support in J-Sim 41
4.2.1 MPLS Model within J-SIM 42
4.3 Implementation of Content-Aware MPLS Routing in J-Sim 43
4.3.1 New Packet Header for DWT Encoded Packets 43
4.3.2 Enhanced MPLS Router with Content-Aware Routing Functionalities . 44
4.3.3 Additional Changes 44
4.4 Simulation Experiments 45
4.4.1 Experimental Design 46
4.4.2 Experimental Parameters 46
4.4.3 Metrics 47
4.4.4 Factors 47
4.4.5 Network Model 48
4.4.6 Simulation-Based Evaluation 49

4.4.6.1 Comparison of Average end-to-end Delay and Jitter 49
4.4.6.2 Comparison of Packet Loss 52
4.4.7 Heterogeneous Network Model 54
4.4.7.1 Comparison of Average end-to-end Delay and Jitter 55
4.4.7.2 Comparison of Packet Loss 57
4.4.8 Service Ratings 58
4.5 Summary 59
Chapter 5 Implementation of Content-Aware MPLS Router 61
5.1 The Linux iptables and netfilter 61
5.1.1 Life of a packet within the Linux Box 62
5.1.2 Matching Packets 64
5.1.3 Targets/Jumps 65
5.2 QoS and Traffic Shaping 67
5.2.1 Traffic Shaping Strategies 68
5.2.1.1 Packet Queues 69
5.2.1.2 QoS Guarantees 70
5.2.2 Queue Disciplines 70
5.2.2.1 Classless Queuing Disciplines 71
5.2.2.2 Classful Queuing Disciplines 71
5.2.2.3 The PRIO qdisc 73
5.2.2.4 Stochastic Fairness Queuing (SFQ) 73
5.3 The MPLS-Linux Project 74
5.3.1 Implementation of FIB in MPLS-Linux 77
5.4 Content-Aware MPLS Router Architecture 81
5.4.1 Ingress Mode 82
5.4.1.1 Marker – Content-Aware Routing Module 82
5.4.1.2 Classifier – QoS Module 84
5.4.1.3 MPLS Module 84
5.4.2 Switch Mode 85
5.4.3 Egress Mode 86

5.5 Summary 87
Chapter 6 Measurement Based Performance Evaluation 89
6.1 Experimental Test-bed 89
6.1.1 Hardware/Software Platform 90
6.1.2 Traffic Generation Tools 91
6.2 Goals and Hypotheses of Experimental Verification 92
6.3 Experimental Design and Parameters 93
6.3.1 Metrics 93
6.3.2 Factors 94
6.3.3 Experimental Scenarios 94
6.4 Analysis of Results 95
6.4.1 Comparison of Packet Loss 96
6.4.2 Comparison of Image Quality and Received File Size 97
6.4.3 Comparison of Average end-to-end Delay and Jitter 102
6.4.4 System level Measurements 106
6.5 Heterogeneous Network Backbone 108
6.5.1 Comparison of Packet Loss 109
6.5.2 Comparison of Image Quality 110
6.5.3 Comparison of Average end-to-end Delay and Jitter 111
6.6 Summary 113
Chapter 7 Conclusion 115
7.1 Limitations and Further Work 115
7.1.1 DWT compressed Video 116
7.1.2 Support for Multicasting 116
7.1.3 Comparison with IPv6 and ATM 116
7.2 Summary and Contributions of Thesis 117
Appendix A: Network Traces 119
Appendix B: MPLS Configuration Files 121
Appendix C: System Level Measurements 122
Bibliography 125

Vita 130

List of Tables

Table 4.1. Comparison of image packet loss between DiffServ, MPLS and content-aware MPLS at $\rho = 1$. 53
Table 4.2. Comparison of video packet loss between DiffServ, MPLS and content-aware MPLS at $\rho = 1$. 54
Table 4.3. Comparison of video packet loss at $\rho = 1$. 57
Table 4.4. Comparison of image packet loss at $\rho = 1$. 58
Table 4.5. Comparison of DiffServ, MPLS and MPLS using proposed scheme as a viable candidate for Multimedia Transportation. 59
Table 5.1. Targets in iptables and their functions. 66
Table 5.2. Queuing Disciplines in Linux. 72
Table 5.3. MPLS-Linux unicast instructions overview. 75
Table 5.4. Implementation of the three MPLS operations with the five MPLS-Linux instructions. 77
Table 6.1. Hardware/Software resources used in test-bed. 91
Table 6.2. Comparison of packet loss between MPLS, MPLS using proposed scheme and DiffServ at $\rho = 1$. 97
Table 6.3. Comparison of packet loss of simulation and measurement based tests when transmitted over content-aware MPLS. 97
Table 6.4. Comparison of packet loss at $\rho = 1$. 109
Table 6.5. Comparison of packet loss of simulation and measurement based tests when transmitted over heterogeneous network backbone. 110

List of Figures

Number *Page*
Figure 1.1: Content Service Model. 4
Figure 1.2: Architecture of a typical MPLS Domain. 6
Figure 1.3: A sequence of operations of forwarding IP packets through an MPLS domain. 7
Figure 3.1: 32-bit MPLS Shim header. 31
Figure 3.2: End-to-end flow of proposed scheme. 33
Figure 3.3: Fields inside an RTP header. 34
Figure 3.4: DWT encoded packet format. 34
Figure 3.5: EXP bits encoding scheme. 36
Figure 3.6: Flow of priority information. 37
Figure 3.7: Priority queue for router buffer management. 38
Figure 4.1: Network Topology for Simulation. 48
Figure 4.2: Comparison of average end-to-end delay of compressed video traffic over DiffServ, MPLS and MPLS using proposed scheme. 50
Figure 4.3: Comparison of average end-to-end delay of compressed images over DiffServ, MPLS and MPLS using proposed scheme. 51
Figure 4.4: Comparison of average jitter of compressed video traffic over DiffServ, MPLS and MPLS using proposed scheme. 51
Figure 4.5: Comparison of average jitter of compressed image traffic over DiffServ, MPLS and MPLS using proposed scheme. 52
Figure 4.6: A heterogeneous network backbone. 54
Figure 4.7: Comparison of average end-to-end delay of video traffic. 55
Figure 4.8: Comparison of average end-to-end delay of DWT encoded images. 56
Figure 4.9: Comparison of average jitter of video traffic. 56
Figure 4.10: Comparison of average jitter of DWT encoded images. 57
Figure 5.1: Traversal of packet within Linux router. 63
Figure 5.2: Processing of a packet in the MPLS layer with MPLS-Linux unicast. 79

Figure 5.3: Modules of proposed content-aware router in ingress mode. 82
Figure 5.4: Modules of proposed content-aware router in switch mode. 86
Figure 5.5: Modules of proposed content-aware router in egress mode. 87
Figure 6.1: Experimental test-bed. 90
Figure 6.2: Comparison of image quality at $\rho = 0.5$. 99
Figure 6.3: Comparison of image quality at $\rho = 0.7$. 100
Figure 6.4: Comparison of image transferred at $\rho = 0.9$ over proposed scheme (image b) and an image transferred at $\rho = 0.7$ over regular MPLS (image a). 100
Figure 6.5: Comparison of image transferred at $\rho = 0.9$ over DiffServ (image a) and proposed scheme (image b). 102
Figure 6.6: Comparison of average jitter. 103
Figure 6.7: Comparison of average end-to-end delay 103
Figure 6.8: Comparison of delay values obtained from simulation and measurement studies while transmitting DWT traffic over content-aware MPLS. 105
Figure 6.9: Comparison of jitter obtained from simulation and measurement studies while transmitting DWT traffic over content-aware MPLS. 105
Figure 6.10: Comparison of active memory of user processes at different values of ρ at ingress router. 107
Figure 6.11: Comparison of interrupts per second at different values of ρ at ingress router. 107
Figure 6.12: Comparison of CPU utilization at different values of ρ at ingress router. 108
Figure 6.13: Comparison of image quality at $\rho = 0.5$. 110
Figure 6.14: Comparison of average jitter. 111
Figure 6.15: Comparison of average end-to-end delay 111
Figure 6.16: Comparison of delay values obtained from simulation and measurement studies while transmitting DWT traffic on heterogeneous network backbone. 112

Figure 6.17: Comparison of jitter values obtained from simulation and measurement studies while transmitting DWT traffic on heterogeneous network backbone. 113

Thesis Abstract

NAME: MUHAMMAD REHAN SAMI

TITLE: CONTENT-AWARE CONGESTION CONTROL OVER MPLS

NETWORKS FOR MULTIMEDIA TRANSMISSION

MAJOR FIELD: COMPUTER NETWORKING

DATE OF DEGREE: JANUARY 2005

Multimedia distribution over the Internet has emerged as a popular application. However, delivering multimedia content poses many non-trivial challenges such as fulfilling Quality of Service (QoS) requirements. This thesis focuses on how to improve QoS of multimedia traffic over Enterprise Data Networks (EDNs) using MultiProtocol Label Switching (MPLS). The reason for choosing MPLS is its support for traffic engineering (TE) and QoS provisioning. A novel content-aware congestion control algorithm is designed, implemented and tested. The proposed scheme enables the MPLS router to discard packets with less important content when there is congestion in the network. By employing such content-aware congestion control, the overall quality of multimedia content is not significantly affected, enabling graceful degradation of quality using a Wavelet based compression technique. The proposed scheme is modeled and simulated using the J-Sim network simulator and implemented in an extended public domain Linux-based MPLS router. Simulation and measurement based testing shows that the proposed scheme maintains low delay, low jitter, and packet loss of multimedia traffic, even under high network congestion when compared with standard MPLS and IP QoS architecture - DiffServ. This thesis contributes a novel scheme to the existing MPLS architecture to improve QoS of Wavelet based compressed multimedia data.

Keywords: *MPLS, QoS, Traffic Engineering, Wavelet Compression, Multimedia, Congestion Control*

Master of Science Degree

King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia
January 2005

خلاصة الرسالة

الاســـــم: محمد ريحان سامي

عنوان الرسالة: التحكم بالازدحام – ذو الدراية بالمحتوي – عبر شبكات إم بي إل إس (MPLS) لإرسال الوسائط المتعددة

التخصص: شبكات الحاسب الآلي

تاريخ التخرج: **يناير** ٢٠٠٥

ظهر توزيع الوسائط المتعددة عبر الشبكة العنكبيوتية (الإنترنت) كتطبيق واسع الانتشار. ولكن توصيل محتوي متعدد الوسائط طرح العديد من التحديات المهمة مثل تلبية متطلبات نوعية الخدمية (QoS). يركز هذا البحث على كيفية تحسين نوعية الخدمة لمرور الوسائط المتعددة عبر مشروع شبكة البيانات (EDN) باستخدام مقسمات ذات بروتوكولات متعددة (إم بي إل إس MPLS). سبب اختيار بروتوكول أم بي إل إس هو دعمه لهندسة المرور وإدارته الوقائية لنوعية الخدمة. تم تصميم وتطبيق واختبار خوارزمية جديدة للتحكم بالازدحام تتعرف على المحتوى. الخطة المقترحة تمكن موجه أم بي إل إس إلى التخلص من الحزم ذات الشأن الأقل أهمية عندما يكون هناك ازدحام في الشبكة، وبتوظيف مثل هذا التحكم فإن النوعية الكلية للرسائل المحتوية على وسائط متعددة لن تتأثر بصورة ملحوظة متيحة ا ال لتدنٍ بسيط في النوعية عند استخدام طريقة ضغط مبنية على مويجة.

تمت نمذجة ومحاكاة الخطة المقترحة باستخدام محاكي الشبكة جي-سيم (J-Sim) وتطبيقها في موجه إم بي إل إس المبني على نطاق لينكس العام والموسع. وأظهرت اختبارات المحاكاة والقياسات أن الخطة المقترحة تحتفظ بمستوى تأخير منخفض ونسبة فقد قليلة للبيانات عند مرور الوسائط المتعددة حتى في حالة وجود ازدحام عالٍ وذلك عند مقارنتها بـ إم بي إل إس القياسي ومعمارية آي بي (IP) لنوعية الخدمة. يساهم هذا البحث بوضع أسلوب جديد عن الأساليب الموجوده لـ أم بي إل إس وذلك لتحسين نوعية الخدمة لبيانات وسائط متعددة مبنية على ضغط المويجة.

كلمات البحث الرئيسية: مقسمات ذات بروتوكولات متعددة (إم بي إل إس MPLS، نوعية الخدمية (QoS)، هندسة الازدحام، الضغط المويجي، التحكم بالازدحام، الوسائط المتعددة.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن، المملكة العربية السعودية

يناير ٢٠٠٥

Chapter 1

Introduction

Multimedia distribution over the Internet has emerged as a popular application. However delivering real-time compressed video and images, which is a major part in multimedia content, poses many non-trivial challenges. Fulfilling the Quality of Service (QoS) requirements for delivering multimedia content over the Internet is an active area of research. As the QoS needs of such applications can not be ignored, the heterogeneous nature of today's "best-effort" Internet makes it difficult to deliver desired quality of such content to different users with different requirements.

Compressing raw video or images and delivering them plays a key role in obtaining desired level of QoS. Compression techniques essentially reduce the size of data to be delivered and hence reduce the amount of bandwidth required to deliver this content. Standard compression techniques (such as Motion Picture Expert Group–MPEG Standard) provide Discrete Cosine Transform (DCT) based lossy compression [1]. Playback of compressed video using DCT does not tolerate losses of parts of the compressed frames due to network congestion and quality of playback becomes unacceptable. Wavelet based techniques provide an alternative video/image compression methodology. These techniques are non-standard but can provide graceful

degradation when parts of compressed data frames are dropped due to congestion. Such techniques can identify parts of the compressed frames that can be dropped under congestion without appreciable loss of quality.

Compression alone cannot improve delivery of good quality multimedia content. The quality of such content also depends on the network characteristics. A network that is aware of QoS requirements of the content can strive to deliver that level of quality compared to today's best-effort Internet. The work focuses on the Multiprotocol Label Switching (MPLS) technology to enhance the QoS provisions of the Internet.

This research involves implementation of a scheme that marks wavelet-based compressed multimedia packets as those containing important or less important contents of compressed and encoded frames. Such markings enable an MPLS router to make decisions according to the packet's priority. At times of congestion, like any other congestion control enabled router, this router will also restrict the rate of outgoing traffic. However, it can restrict low priority traffic to allow high priority streams to maintain a gracefully degraded QoS for compressed multimedia content at times of high network congestion. The proposed congestion control scheme is a load-shedding based congestion control scheme that not only ensures QoS by adding content awareness in devices but also allows efficient utilization of available network resources.

A simulation and measurement based performance study is conducted by compressing raw multimedia content using DWT based compression and transporting it over an MPLS network. QoS is compared with QoS enabled IPv4 (Differentiated Services – DiffServ) network as well as standard MPLS.

1.1 Content Service Model

In today's content driven Internet, a content service model can be defined in terms of three main entities [2]. These three entities are 1) Content Customers 2) Content Providers (CP) and 3) Content Service Providers (CSP). These three entities are interconnected together through Service Level Agreements (SLA).

Content customers are able to get the content from the CPs through CSPs. In current Internet, portal services such as Yahoo, America On-Line (AOL) and Infoseek can be considered as CSPs. CSPs can help customers to access content services efficiently and economically. CP in this model refers to the actual creator and owner of the content.

In case of the Internet, there are several issues that pose non-trivial problems such as scalability, compatibility and interoperability. The proposed scheme is well suited for small and medium sized enterprise networks that comprise of a single managed domain with end-to-end MPLS backbone.

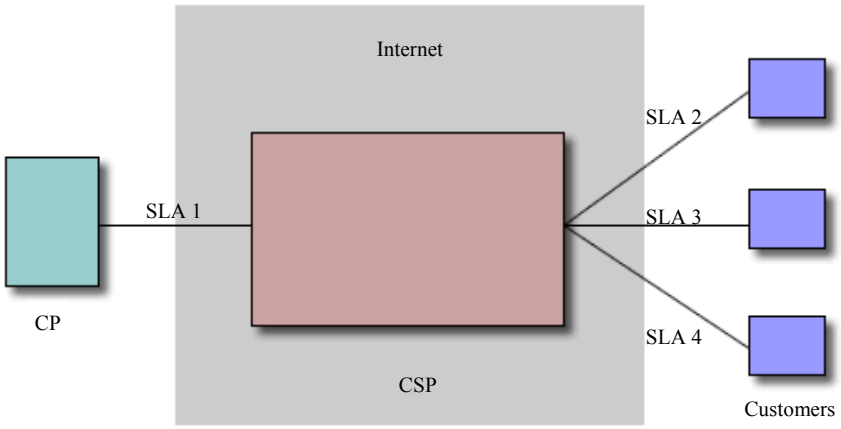


Figure 1.1: Content Service Model.

1.2 Discrete Wavelet Transform based Compression

DWT is similar to DCT as it converts an image into frequency components. The difference between DCT and DWT is that in the wavelet transform, the process is performed on the entire image and the result is a hierarchal representation of the image, where each layer is a frequency band [1] . DWT techniques are more suitable to obtain variable QoS as compared to DCT. For instance, some of the high frequency contents can be dropped with minutely perceptible degradation of decompressed image quality. In the worst case, low frequency bands of compressed image can help restore a coarse grain representation of original image. Due to this characteristic, graceful degradation of image quality using DWT-based compressed video is an ideal candidate to test load shedding based congestion control schemes. Thus this study will focus DWT compression of multimedia traffic only.

The idea of Embedded Image Coding using Zerotrees of Wavelet Coefficients (EZW) was presented by Shapiro [3]. The EZW coder exploits the characteristic that visual information can tolerate some data loss without drastically degrading the visual quality. It applies wavelet transformation and heuristics such that the encoded data is ordered in terms of visual significance. The encoded data is disseminated as an embedded bit-stream with the data in descending order of significance. The bit-stream can be truncated at any point. Thus various compression ratios are possible from a single bit-stream. The degradation of visual quality, as a result of not sending data of low-significance, can be managed so that the resulting decoded image is still useable by the user and the application.

1.3 Multiprotocol Label Switching

MPLS is a layer 3 switching technology that emphasizes on the improvement of packet forwarding performance of backbone routers [57]. The main idea behind MPLS technology is to forward packets based on a short and fixed length identifier termed as a “label” rather than the layer 3 IP address of the packets. The labels are assigned to each packet (PUSH) at the ingress node known as the Label Edge Router (LER) of an MPLS Domain [4]. MPLS routers make forwarding decisions based on these labels (SWAP). The labels are detached (POP) as the packets depart the MPLS domain at the egress

LER. Within the MPLS domain, packets are forwarded using these labels by the core Label Switch Routers (LSRs).

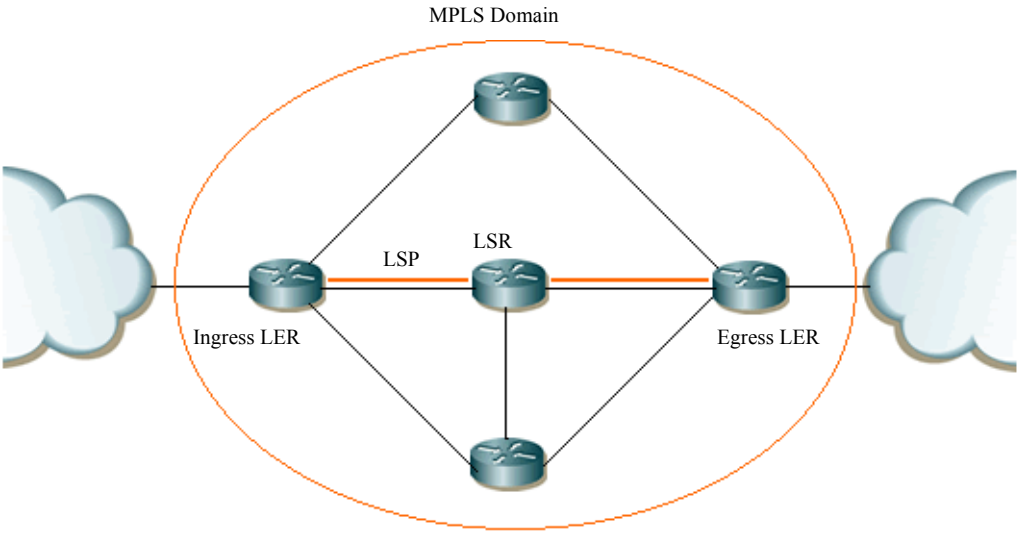


Figure 1.2: Architecture of a typical MPLS Domain.

The IP packets are switched through pre-established Label Switched Paths (LSP) by signaling protocols or done statically. LSPs are determined at the ingress LER and are unidirectional (from ingress to egress). Packets with the identical label follow the same LSP and are categorized into a single Forwarding Equivalence Class [4] (FEC). FEC can be defined as a group of IP packets which are forwarded in the same manner along the same LSP.

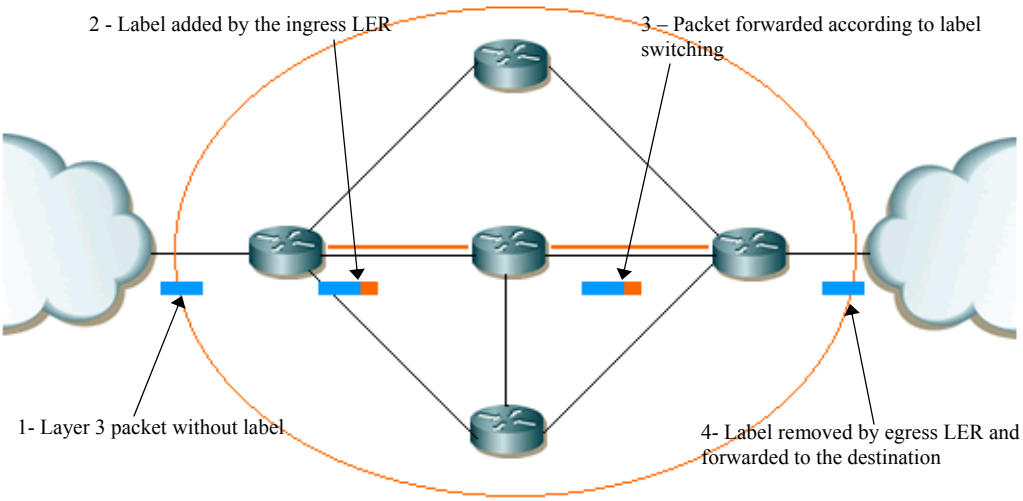


Figure 1.3: A sequence of operations of forwarding IP packets through an MPLS domain.

1.3.1 Sequence of MPLS Operations

Consider a traffic flow from cloud A to cloud B in Figure 1.3. Both clouds A and B are pure IP networks. When a packet from A enters the MPLS domain and reaches the ingress LER, the source and destination IP addresses of the packet are analyzed and the packet is classified in a FEC. All packets within the same FEC use the same LSP. Suppose an LSP has already been established for the FEC of the packet sent by A to B, then the ingress LER inserts or pushes an MPLS header on the packet. Subsequent routers of the MPLS domain update the MPLS header by swapping the label. Finally, the last router of the LSP, called egress LER, removes or pops the MPLS header,

so that the packet can be handled by subsequent MPLS-unaware IP routers or hosts inside cloud B.

1.3.2 MPLS Forwarding Information Base

MPLS routers push, swap and pop MPLS headers according to rules contained in a forwarding table called Forwarding Information Base (FIB) that is distinct for each MPLS router. The FIB can contain three different types of entries. A Next Hop Label Forwarding Entry (NHLFE) contains the information necessary to forward a packet for which a label has already been assigned. A NHLFE contains two pieces of information: the packet's next hop address, and whether the MPLS header of the packet must be swapped or popped. If the MPLS header of the packet must be swapped, then the NHLFE also contains the new label of the packet. The Incoming Label Map (ILM) contains the mappings between labels carried by incoming packets and NHLFE entries. Last, the FEC-to-NHLFE (FTN) contains the mappings between incoming packet FECs and NHLFE entries. MPLS routers use their FIB as follows. Suppose a packet with no label arrives at an MPLS router. The MPLS router first determines the FEC for the packet, and then looks up in the FIB for the FTN that matches the FEC of the packet. This FTN contains a label and a NHLFE which in turn contains the next hop for the packet. The MPLS router pushes an MPLS header that contains the label read in the FTN and forwards the packet according to the information

contained in the NHLFE. Now suppose that a labeled packet arrives at an MPLS router. The MPLS router searches in the FIB for an ILM that matches the label of the packet and reads the associated NHLFE. The NHLFE can either indicate that the MPLS header must be swapped against a new label, or popped. In the former case, the MPLS router swaps the MPLS header and forwards the packet to the next hop specified in the NHLFE. In the latter case, the MPLS router pops the label and forwards the packet to the next hop specified in the NHLFE.

1.3.3 Traffic Aggregation

A main point of interest with FECs in a traffic engineering context is that they support aggregation. All packets from different sources but entering the MPLS domain through the same LER, and bound to the same egress LER can be assigned to the same FEC and therefore the same virtual circuit. In other words, there is no need to establish a new virtual circuit for each (source, destination) pair read in the headers of incoming packets. Once ingress LER has determined the FEC of a packet, the ingress LER assigns a virtual circuit to the packet via a label number. Also, FEC definitions can take into consideration IP packet sources in addition to destinations. Two packets that enter the MPLS domain through the same LER and going to the same destination can use different sets of links so as to achieve load

balancing, that is, put the same amount of traffic on all links hereby distributing the load of traffic on each link.

1.3.4 Traffic Engineering

Traffic engineering deals with the performance of a network in supporting users' QoS needs. Traffic engineering for MPLS networks involves the measurement and the control of traffic. The objectives of traffic engineering in the MPLS environment are related to two performance functions [5]:

1. Traffic oriented performance which includes QoS operations.
2. Resource oriented performance objectives which deal with networking resources to contribute to the realization of traffic oriented objectives.

The aim of traffic engineering is to find mechanisms to satisfy the growing need of users for bandwidth; thus, the efficient management of the available bandwidth is the essence of traffic engineering. MPLS plays an important role in engineering the network to provide efficient services to its customers. RFC 2702 specifies the requirements of traffic engineering over MPLS and describes the basic concepts of MPLS traffic engineering like traffic trunks, traffic flows and LSPs [20]. The advantages of MPLS for traffic engineering include:

1. Label switches are not limited to conventional IP forwarding by conventional IP-based routing protocols

2. Traffic trunks can be mapped onto label switched paths
3. Attributes can be associated with traffic trunks
4. MPLS permits address aggregation and disaggregation (IP forwarding permits only aggregation)
5. Constraint-based routing is easy to implement
6. MPLS hardware is less expensive than ATM hardware.

Proper traffic engineering techniques are the basis of providing good QoS support to MPLS networks [41][44].

1.4 Advantages of MPLS

Three main factors determine the performance of any network application, namely the supporting system, network and the protocol. Considering the architectural philosophy of IP, it imposes a restriction in that it does not support a fixed data flow model upon the network's application set. It does not take into consideration the "type" of traffic to route. It merely routes the traffic based on simple routing algorithms (mostly shortest path). In such cases, delay sensitive traffic can have an impact due to excessive injection of delay insensitive traffic as mentioned in [6], because all types of traffic follow the same shortest path. This can be a drawback for video traffic due to its time stringent nature. Another disadvantage of conventional IP routing is the amount of processing the core routers have to perform in order to forward

a packet to its destination. In L3 (IP) routing, the network does not maintain state, the data packets following the first packet in a flow are unaware of its routing. Hence the route is calculated independently for every packet although they have the same destination address. MPLS improves these shortcomings in IP routing by combining L3 routing with L2 switching. It uses the first data packet to establish the LSP, and distributes a label to the FEC that the first data packet belongs to. If the data packets following the first one belong to the same FEC as the fist data packet, then MPLS uses the same label to encapsulate them. Forwarding decisions are made on the basis of the fixed length short labels rather than the variable length IP header and longest prefix matches. This considerably reduces the processing time and in turn increases the core network's packet forwarding performance.

1.5 Quantitative QoS Evaluation

In order to transport desired quality multimedia content, there are certain quantitative requirements that should be fulfilled; these include: bandwidth, delay and delay jitter, and loss. Each of these is considered in more detail in the following.

1.5.1 Bandwidth

Minimum bandwidth is required to achieve an acceptable quality for streaming multimedia content. However keeping in mind today's Internet,

there is no explicit bandwidth reservation mechanism, which is deployed everywhere for such applications. Furthermore, excessive traffic can cause congestion in the network, which can further degrade the quality of such applications. Hence congestion control must be introduced. For video streaming, congestion control takes the form of rate control; that is, adapting the sending rate to the available bandwidth in the network. Compared with non-scalable DCT based video compression, scalable DWT based video compression is more adaptable to the varying levels of available bandwidth in the network.

1.5.2 End-to-end Delay and Delay Jitter

In contrast to data transmission, multimedia content requires bounded end-to-end delay and delay jitter. That is, every multimedia packet must arrive at the destination in time to be decoded and displayed. This is because real-time compressed video must be played out continuously. If the video packet does not arrive in time, the playout process will pause, which is annoying to the human eye. The video packet that arrives beyond a time constraint is useless and can be considered lost. Although real-time compressed video requires timely delivery, the best-effort Internet does not offer such a delay or jitter guarantee. In particular, the congestion in the Internet could incur excessive delay. Since the Internet introduces time-varying delay, to provide continuous playout, a buffer at the receiver is usually introduced to remove

jitter. The average delay for real-time applications should not exceed more than 400ms [65] [66]. An acceptable range lies from 150 to 400ms where anything below 150ms is guaranteed high quality [25].

1.5.3 Packet Loss

Loss of packets can potentially make the presentation displeasing to human eyes, or, in some cases, make the presentation impossible. Thus, multimedia applications typically impose some packet loss requirements. Specifically, the packet loss ratio is required to be kept below a threshold to achieve acceptable visual quality. However, the best-effort Internet does not provide any loss guarantee. In particular, the packet loss ratio could be very high during network congestion, when number of packets queued at the router's buffer exceed the capacity of the buffer causing the router to drop packets. This in turn results in severe degradation of video quality. Thus, it is desirable that a multimedia stream be tolerant to packet loss.

1.6 Problem Statement

An acceptable quality of multimedia content is essential at the receiver's end even under high network congestion. This requires the network devices to be intelligent enough to discard only those packets that contain less important content. The crux of the underlying problem is to design and implement a mechanism in MPLS router to distinguish between high priority and low

priority packets containing DWT encoded content. Furthermore the MPLS device must also encode this priority information in the MPLS header for QoS routing (QoSR)

1.7 Practicality of Proposed Scheme

Based on the generalized content service model described in Section 1.1, a crisp practicality of the proposed scheme can be defined here. In case of the Internet, there are several issues that pose non-trivial problems such as scalability, compatibility and interoperability. The proposed scheme is well suited for small and medium sized enterprise networks that comprise of a single managed domain with end-to-end MPLS backbone. Hence, Enterprise Data Networks (EDNs) are most suitable candidates for implementing the content-aware congestion control since the CP and CSP are managed and administered by same authorities. Data applications can generate DWT compressed multimedia content and the MPLS backbone can be made content-aware congestion control enabled to utilize the compression technique and provide EDN users with optimum quality content even when the network backbone is congested due to other application data on the backbone.

1.8 Contributions

This thesis contributes to the state of the art a novel content-aware congestion control method in MPLS routers that enables the routers to forward packets based on the content in the packet's payload. It implements a client/server architecture that has a server application sending DWT/EZW encoded images and performs application level prioritization of packets and a client application receiving DWT/EZW encoded images, decoding the image and displaying it on the picture viewer at the client's machine. The modeling and simulation of the proposed scheme required modifications in the current MPLS implantation of J-Sim. This work adds extension to the J-Sim MPLS classes by adding content-aware routing, new packet format, and delay counters. The proposed content-aware congestion control scheme is implemented in software based MPLS routers and is experimentally verified and tested on a Linux based MPLS test-bed.

1.9 Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 gives an extensive literature survey of DWT encoded schemes, congestion control in MPLS networks and QoS in IP networks. Chapter 3 presents the implementation methodology of the proposed scheme. Chapter 4 presents simulation based study and provides critical analysis of the results obtained from this study.

Implementation of content-aware congestion control techniques in Linux based software MPLS router is described in detail in Chapter 5. Chapter 6 describes the experimental test-bed, gives the experimental validation and results of these experiments. Chapter 7 provides the conclusion and further work in this direction.

1.10 Summary

This is an introductory chapter outlining fundamental concepts of content modeling, basic operations of MPLS networks, traffic engineering, advantages of MPLS networks and the QoS parameters understudy in a typical QoS related research. The chapter crisply defines the problem statement and describes the expediency of the proposed scheme as a performance enhancement scheme for DWT encoded multimedia traffic for end-to-end MPLS enabled EDNs. At the end the chapter presents the key contributions made by this thesis work and a brief organization of the thesis.

Chapter 2

Literature Survey

This chapter reviews some of the recent work on improving QoS for delivering multimedia content over the Internet. Many approaches have been reported to achieve high quality of streaming video. The chapter is organized as follows: In Section 2.1 a review of recent video and image compression efforts using Discrete Wavelet Transform is presented. Different congestion control mechanisms in MPLS networks are discussed in Section 2.2, Section 2.3 elaborates on QoS issues related to the delivery of rich multimedia on the Internet and finally Section 2.4 elaborates QoS techniques in IPv4 networks.

2.1 Compression using DWT

Video compression is necessary in terms of achieving better quality video as it optimizes the video quality over a given bit rate range. As our study focuses on DWT compression techniques, we will consider a few contributions made in this area.

Reza Adhami [7] has presented a video compression technique based on the use of wavelet transform and a differential image compression technique. The basic idea behind the algorithm is to perform a wavelet transform on the first frame of the video images; concurrently quantize, encode and store the

wavelet transforms coefficients. All subsequent frames are treated similarly. Experimental results showed the highest compression ratio without visible degradation in the video quality to be 72:1 (better in some cases than JPEG).

The idea of Embedded Image Coding using Zerotrees of Wavelet Coefficients was presented by Shapiro [3]. The EZW algorithm generates bits in the bit stream in order of importance. Thus the resulting code is fully embedded. Embedded coding allows the encoder to terminate encoding at any point thereby allowing achieving a target rate exactly. Since embedded code contains all the lower rate codes embedded at the beginning of the bit stream, effectively, the bits are "coded in importance".

The EZW algorithm discussed above has the following main features as described by the author:

1. The EZW uses a discrete wavelet transform that provides a compact multiresolution representation of the image.
2. It implements zerotree coding that provides a compact multiresolution representation of significance maps that are binary maps indicating the positions of the significant coefficients.
3. It uses Successive Approximation that provides a compact multiprecision representation of the significant coefficients and helps in embedded coding.

4. It makes use of a prioritization protocol that determines the order of importance. That is it decides the importance of wavelet coefficients by considering parameters such as precision, magnitude, spatial location and scale.
5. Adaptive multilevel arithmetic coding is also used to provide a fast and efficient method for entropy coding.
6. The EZW runs in a sequential manner and stops whenever a target bit rate or a target distortion is met.

Experimental studies of this algorithm showed that the compression performance was competitive with virtually all known techniques. A distinct advantage of this algorithm is that it achieves a precise rate control. It allows the user to choose any bit rate and encode the image to exactly the desired bit rate. Furthermore, since no training of any kind is required, EZW is fairly general and runs smoothly with most types of images.

The DWT approach is also useful when integrated with standards such as MPEG. The authors present a scalable coding scheme based on DWT and MPEG coding [8]. This technique uses the hierarchal pyramid structure that provides multiple resolutions. DWT decomposes the image into several bands. Each band is then subjected to MPEG coding technique. One of the advantages of this scheme is that it reuses the widely available MPEG

hardware and software. Simulation results also show that this approach provided significantly improved results than the original MPEG coding.

A very low-bit rate video coding scheme is designed using DWT [9]. The authors claim that the approach reveals that the coding process works more efficiently if the quantized wavelet coefficients are preprocessed by a mechanism exploiting the redundancies in the subband structure of the wavelet. They introduce a new precoding technique termed as PACC (partitioning, aggregation and conditional coding). Experimental studies have been carried out to compare the PACC approach with MPEG4 both for coding of intraframes and residual frames of typical MPEG4 test sequences. The algorithm mainly consisted of: motion estimation and compensation, wavelet representation and quantization, the PACC precoding framework, and arithmetic coding. By comparing the PACC codec with MPEG4 coding technique at very low bit rates showed better performance of PACC using the four MPEG4 test sequences Akiyo, Hallmonitor, News, and Foreman.

There are many other codec standards that are out of the scope of this research study.

2.2 Congestion Control in MPLS Networks

This section highlights some of the research that has been done in improving congestion control in MPLS networks. An Active Traffic and Congestion Control Mechanism in MPLS (MPLS ATCC) was proposed by Zhiqan Zhand et al [10]. The authors propose an integrated model that combines both MPLS and active networks. This mechanism moves the endpoint congestion control algorithm to the core network. ATCC uses Active Networking technology to enable router participation in both congestion detection and congestion recovery. The feedback congestion control system is extended from the ingress routers to the core LSRs. Congestion is detected at the core router, which immediately begins reacting to congestion by changing the traffic that has already entered the network. Performance studies were conducted using simulation techniques to compare the MPLS ATCC with TCP congestion control. The results showed that the proposed approach was much better in terms of round trip delay and overall throughput of the network.

A reactive congestion control scheme is proposed known as the Fast Acting Traffic Engineering (FATE) to control congestion in MPLS networks [11]. The ingress LER and the LSRs react to information received from the network regarding flows experiencing significant packet losses, by taking

appropriate remedial action, i.e., by dynamically routing traffic away from a congested LSR to the downstream or upstream underutilized LSRs.

The above mentioned schemes are focused on controlling congestion in MPLS networks. There is a general idea among the research groups that if by appropriately incorporating TE in MPLS networks the congestion point might not be reached. Proper traffic engineering in a MPLS network is itself a congestion avoidance scheme. A traffic engineered MPLS network less likely requires a congestion control mechanism. Or in other words, when congestion avoidance is done properly, congestion control may be of less significance. General patterns of response time and throughput of a network are described in [12] as the network load increases. If the load is small, the throughput generally keeps up with the load. As the load increases, throughput increases. After the load reaches the network capacity throughput stops increasing. This point is called the knee. If the load is increased any further, the queues start building, potentially resulting in packets being dropped. Throughput may suddenly drop when the load increases beyond this point. This point is called the cliff because the throughput falls off rapidly beyond this point.

A scheme that allows a network to operate at the knee is known as a congestion avoidance scheme as distinguished by a congestion control

scheme, which tries to keep the network operating in the zone to the left of the cliff [12].

An application-specific congestion control method is yet to be implemented. The proposed scheme emphasizes on employing appropriate congestion control by providing the ingress router application-specific information to assign priorities.

2.3 Using MPLS-TE to Improve QoS

"Traffic engineering is the process of arranging how traffic flows through the network so that congestion caused by uneven network utilization can be avoided" [13]. As mentioned earlier, QoS for multimedia content relates to terms like bandwidth, end-to-end delay and packet loss. End-to-end delay and packet loss are mainly caused by congestion in the network. Congestion is caused when there is more traffic on a link than the link can actually handle. Generally speaking, for any time interval, the total sum of demands on a resource is more than the capacity of the resource, the resource is said to be congested for that interval. Mathematically

$$\sum \text{Demands} > \text{Available Resources}$$

Hence in a congested network, traffic gets delayed or is even dropped at the routers. This sort of congested links are due to one of the major

characteristics of today's routing protocols. Almost all the routing protocols use a single objective optimization i.e. shortest path to the destination and omit all the paths that are not the shortest. Second, today's "best-effort" routing will shift the traffic from one path to "better" path whenever the "better" path is found. This happens even if the current used path meets the service requirements of the traffic. This kind of shift is undesirable because it will bring routing oscillations when the routing is based on metrics like available bandwidth, which changes rapidly from time to time. The traffic will be routed back and forth between alternate paths. Even worse, this kind of oscillations can increase the variation in the delay and jitter experienced by the end users.

MPLS offers the capability of efficiently utilizing network resources to improve the delivery of packets on the network by implementing Traffic Engineering. Traffic engineering reroutes traffic to paths that are not used by regular routing protocols. This can be defined as QoS routing (QoSR). This phenomenon of MPLS can prove to be very handy in terms of improving QoS of multimedia content. Different LSPs can be set up according to the traffic requirements of these LSPs. For example, different LSPs can be used for different classes of traffic with different priorities, resulting in the logical partitioning of the network. Each logical partition of the network will transport one class of traffic. The end result being that the premium traffic

(in our case DWT encoded video and images) will use the more resources of the network as compared to the best effort traffic [14].

Improvement in QoS using MPLS-TE is becoming a very hot area of research nowadays. Researchers have conducted simulation based experiments in order to study the impact on QoS using proper traffic engineering techniques. In [6], the authors report the results they generated from conducting a simulation based study by comparing the services received by TCP and UDP traffic flows when they share a link or a MPLS traffic trunk. Since TCP flow is congestion sensitive, it suffers at the hands of UDP flow that is congestion insensitive. The authors show that by properly engineering the different traffic flows using MPLS trunks on LSPs, the overall throughput of the TCP flow improves significantly. What they did was that they isolated the TCP traffic flow on a different trunk. This way even if they increased the UDP flow to quite some extent, it had no effect on the TCP flow that was using a different path.

Performance of time critical applications over MPLS enabled networks has been tested [15]. The authors conduct a simulation based study, which incorporates TE and QoSR of the MPLS to support Internet-Based Distance Learning (I-DL). This service model is simulated using OPNET and compared to existing IP routing algorithms. Using MPLS approach,

dedicated LSPs were used for different application types, each serving for an application of different priority. Optimal results were obtained with I-DL end-to-end delay of only 3.5ms.

2.4 QoS Approach in Ipv4 Networks

The best effort service cannot provide effective QoS support to real-time multimedia applications because it treats all packets the same way, without taking into consideration the constraints these applications have. IETF introduced “Differentiated Services” [16] as a simple and scalable method for providing QoS over IP networks.

DiffServ provides scalable and “better than best-effort” QoS. DiffServ routers are stateless and do not keep track of individual microflows, making it scalable to be deployed in the Internet. The DiffServ Code Point (DSCP) in the Differentiated Services (DS) field of the IP header identifies the Per Hop Behavior (PHB) associated with the packet, which is used to specify queuing, scheduling, and drop precedence.

The motivations for DiffServ with MPLS [17] include user demands for consistent QoS guarantees, efficient network resource requirements by network providers, and reliability and adaptation of node and link failures. DiffServ provides scalable edge-to-edge QoS, while MPLS performs traffic

engineering to evenly distribute traffic load on available links and fast rerouting to route around node and link failures.

The proposed scheme adds the “better than best-effort” QoS characteristic of DiffServ into the MPLS technology. With this scheme, the Class of Service (CoS) architecture of DiffServ is eliminated and priority information can be directly mapped to EXP bits. In this way, a QoS aware network may no not require integrating two different technologies; instead MPLS alone can serve the purpose.

2.5 Summary

This chapter presents some of the recent work on improving QoS for delivering multimedia content over the Internet. It highlights key achievements in the area of DWT compression for multimedia content, congestion control in MPLS networks, and application of MPLS-TE to improve QoS. It also describes DiffServ, architecture for QoS support in IP networks. It emphasizes on the point presented in literature that both MPLS and DiffServ are being implemented in a hybrid manner to achieve better QoS and that the proposed scheme will be able to do that with MPLS alone.

Chapter 3

Problem Definition and Solution Methodology

This chapter discusses the problem definition and describes in detail the solution methodology of the proposed scheme.

3.1 Problem Definition

Transmitting multimedia content like video and images over IP networks using the Internet Protocol (IPv4), suffers from many drawbacks due to the best-effort nature of IP. The unreliable performance of IP is the primary motivation to integrate MPLS traffic engineering functionality in transmitting multimedia content over the Internet. Different compression techniques have been proposed and implemented that compress raw content in efficient ways to conserve network bandwidth. High quality multimedia content over the Internet requires bounded delay and jitter, minimal packet loss and economical use of available link bandwidth. Some standard video compression techniques have been tried over MPLS networks [19] but these studies do not incorporate content-aware differentiation among various types of incoming network traffic. Although these approaches show better performance in terms of path establishment, they provide insignificant improvement of QoS. Moreover, they are restricted to standard compression

techniques. A DWT encoded data transmission framework delivered over MPLS (QoS aware network) is not implemented so far. The primary goal is to improve the overall quality of multimedia content and to maintain a certain level of QoS even when the network is congested. As seen in literature, MPLS is appropriate for delay sensitive traffic and also provides TE and QoS capabilities. DWT based compression should achieve graceful degradation of quality in a congested network.

3.2 Solution Methodology

This section comprehensively describes the details of the various parts of the solution methodology and their design. It also describes why each part is used in the proposed solution.

3.2.1 Bit Mapping and Quality of Service Routing

One of the problems related to MPLS, as defined in [20], is how to map packets onto forwarding equivalence classes. Figure 3.1 shows the MPLS Shim Header. This header contains the actual label that is inserted between the layers 2 and 3 headers. The header is 32 bits long. It is worth noting here the 3 experimental bits, marked as EXP. These bits may be used to map certain priority information encoded within the video packets onto FECs. The question here lies in the fact that how to map this information from the

DWT encoded packet to the MPLS header so that the MPLS router has ample information and can decide on the basis of this information whether to discard a packet at times of congestion or not.

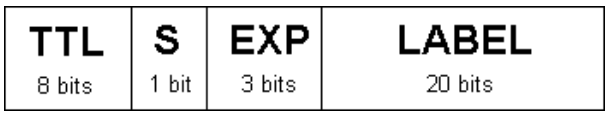


Figure 3.1: 32-bit MPLS Shim header.

The task was to design a scheme that maps the output bits of the DWT decoder to these 3 Exp bits in such a way that there could be a clear distinction between high priority packets and low priority packets. This will provide a means for routing these frames in a different manner according to the priority of the data to be routed. There can be 8 (2^3) different priority levels of image frames. By having knowledge of the importance level of different image frames, the ingress router should be capable of determining which frames can be dropped at time of network congestion without degrading the overall video quality.

An MPLS router performs three main functions: pushing of a label (encapsulating a regular IP packet with MPLS header), swapping the label (changing incoming label to outgoing label according to preconfigured switching table) and popping label (removing label at the edge of the network). The proposed algorithm adds diminutive but significant changes to

these three functions of an MPLS router in such a way that it makes the router intelligent enough in making forwarding decisions under network congestion. The scheme is implemented to serve a wide range of loss-tolerant, real-time applications as well as images whose content can be dropped under congestion (graceful degradation). However, this adds a constraint to the algorithm's applicability that it can be used only for DWT encoded data.

3.2.2 End-to-end Flow

The proposed content-aware congestion control scheme has end-to-end flow. At the sending end, application generates raw multimedia content that is encoded using EZW technique. Packetization is performed by encapsulating the content with appropriate headers and packet priority is assigned at the application and sent on to the network via the MPLS router. The MPLS router checks the packet's priority and encodes priority information in the MPLS header with the relevant EXP bit value. At the receiving end, again the MPLS router (egress) removes the MPLS header and sends the packet to the upper layers for de-encapsulation. Headers are removed and the decoder decodes the content and finally the application displays the content. Figure 3.2 depicts the end-to-end flow of the scheme. This scheme works well with a pure end-to-end MPLS network. The same methodology can be applied to

a mix of IP and MPLS network where packet encapsulation/de-encapsulation will be done at every demarcation point between an MPLS domain and an IP network.

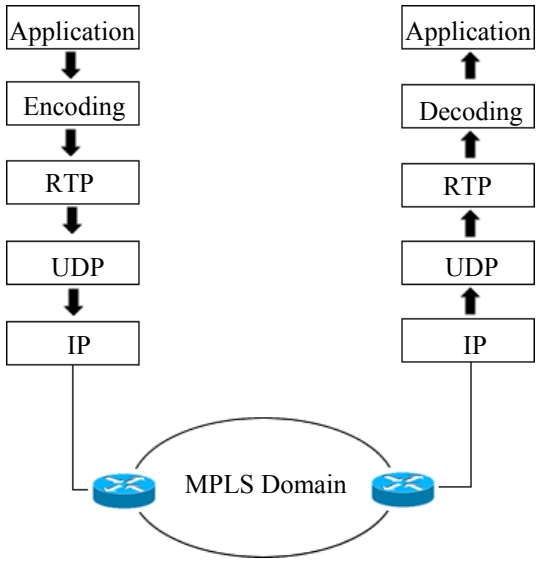


Figure 3.2: End-to-end flow of proposed scheme.

3.2.3 Packet Identification using RTP

Since the main idea is to treat a particular class of traffic differently, there must be some sort of packet identification scheme built within the router in order to classify packet on the basis of type. Given that multimedia streaming is a real-time application and that the compression algorithm encodes bits ordered in importance, we use the Real Time Protocol (RTP) [18] as the transport protocol. Apart from the real-time nature of most of the multimedia

content, another motivation is to use UDP as the underlying transport protocol since UDP has proven to be a better solution in transferring DWT encoded images [58]. The payload type field in the RTP header is used to uniquely identify DWT encoded files. The sending application marks high priority packets with a unique payload type value and the low priority packets with another.

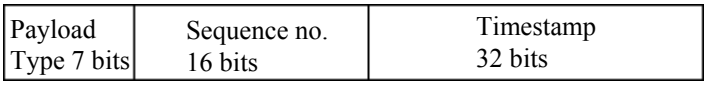


Figure 3.3: Fields inside an RTP header.

The fields in use and the ones that can be used for more complicated real-time applications are shown in the figure above. The total size of the DWT encoded packet is 1468 bytes from which 20 bytes consist of IP header, 4 bytes of UDP header and 16 bytes of RTP header. The size of the payload may vary from 576 bytes to 1468 bytes. The maximum size is kept to follow the Ethernet standard of a maximum of 1500 bytes for a packet.

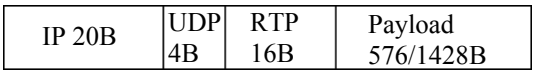


Figure 3.4: DWT encoded packet format.

3.2.4 Packet order and Prioritization

Since the DWT compression component decomposes an image into different frequency bands and generates bits ordered in importance, sequence numbers

assigned to packets will represent this order. A video clip may contain many frames, and within each frame there could be many frequencies (both important as well as less significant). So for an average long video clip, if there are n frames each with f different frequency components, where f_h are the high priority frequency bands and f_l are the low priority bands, then n can be defined as a set of f_h and f_l and a video clip is a sequence of n f_h and f_l. From this it can be stated that sequencing of packets will replicate this order of importance and after every complete set of f_h and f_l, the sequencing of packets will restart for the next n.

From the above description, it is clear that the sending application will follow the generation of packets as a prioritization scheme. In a total of n packet generation, first $n/2$ packets can be considered high priority packets and the rest low priority.

3.2.5 EXP bits Encoding Scheme

An EXP bit encoding scheme has been designed that will be used for encoding packet information onto the 3 EXP bits in the MPLS header. If the first bit is set to 0, then there is no need to check further as the packet is not a DWT encoded packet. If it is 1, then check for the next bit. If the second bit is 0, then the packet is of low priority and is suitable to drop under congestion. If it is 1, then the packet is of high priority and cannot be

dropped even under severe congestion. If the third bit is 0, the packet is delayed and is of no use any longer. This implies that it can be dropped at any time. If it is 1, then the packet is in time and has to be forwarded. Keeping this bit scheme in mind, there are only two values for the EXP field that will be used for packet prioritization by the router: 7 and 5.

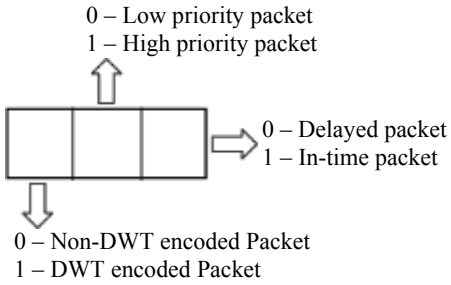


Figure 3.5: EXP bits encoding scheme.

3.2.6 MPLS Router Operations

The ingress router determines the priority of a particular DWT encoded packet when it pushes the MPLS header at the edge. Router analyzes the header and assigns EXP field value 7 to high priority packets and 5 to lower ones. At the core backbone, the label switch routers need not read the transport layer header, instead they will just check the EXP field value in the MPLS header and queue them according to their respective priorities. Similarly at the receiving edge, the router is intelligent enough to save

bandwidth of the subsequent IP network by discarding all delayed video packets (assuming there could be still some delay within the MPLS cloud).

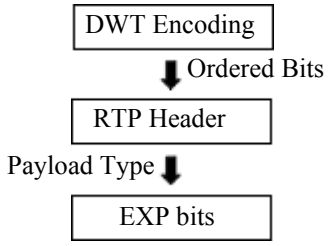


Figure 3.6: Flow of priority information.

3.2.7 Buffer Management at Router

Priority queues are used instead of the regular FIFO queues for buffer management at the router's interface. A priority queue is an m-level queue where m is configurable. Specifically, it consists of m FIFO queues. The level 0 queue has the highest priority while the level (m-1) queue has the lowest. When a packet arrives, the packet is classified into one of the m levels and is put in that FIFO queue.

For this implementation, m = 3, level 0 queue is the highest priority queue and all DWT encoded packets with EXP bits set to 7 are assigned to this level. Level 1 is the next higher priority queue to which all DWT encoded packets with EXP bits set to 5 are assigned. All other packets are assigned to the level 2 queue that maintains the lowest priority. At times of congestion packets belonging to queue 0 have the privilege to be forwarded to the

outgoing interface whereas packets belonging to queues 1 and 2 are treated with lesser importance.

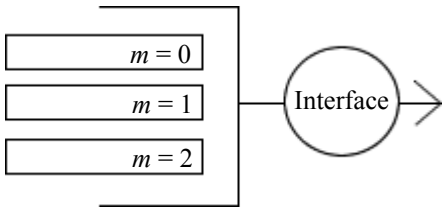


Figure 3.7: Priority queue for router buffer management.

In order to maintain fairness among the three levels of the priority queue, Stochastic Fairness Queue is also implemented. Traffic is separated into conversations and packets are dequeued in a round-robin fashion ensuring that no single conversation completely swamps the queue.

3.3 Summary

This chapter defines the problem under study and presents a comprehensive and detailed description of the solution methodology that is adopted in order to solve the problem. The bit mapping scheme at the MPLS router along with packet identification, ordering and prioritization of packets is discussed in further detail. The chapter sheds light on the key aspects of operation of the proposed MPLS router that include bit encoding scheme and packet queuing and buffer management.

Chapter 4

Simulation and Analysis

Simulation based analysis was conducted to evaluate the performance of the proposed scheme. This chapter goes in the details of the simulation study and provides critical analysis on the results obtained from this study.

4.1 J-Sim Network Simulator

J-Sim network simulator [21] was used to study an MPLS network. J-Sim (formerly known as JavaSim) is a component-based compositional simulation environment. It has been built upon the notion of the autonomous component programming model similar to COM/COM+, JavaBeansTM, or CORBA.

4.1.1 The J-Sim Autonomous Component Architecture

The basic entity in J-Sim is components, but unlike the other component-based software packages/standards, components in J-Sim are autonomous and are realization of software ICs [22]. The autonomous component architecture mimics the IC design architecture in the closest possible way. The behavior of J-Sim components are defined in terms of contracts and can be individually designed, implemented, tested and incrementally deployed in

a software system. A system can be composed of individual components in much the same way a hardware module is composed of IC chips. Moreover, components can be plugged into a software system, even during execution.

4.1.2 J-Sim Network Modeling and Simulation

For the purpose of network modeling and simulation, a generalized packet switched network model is defined on top of the autonomous component architecture. The model defines the generic structure of a node (either an end host or a router) and the generic network components, both of which can then be used as base classes to implement protocols across various layers. Although the model is derived by featuring out the common attributes of network entities in the current best-effort Internet, it is general enough to accommodate other network architectures, such as the IETF differentiated services architecture, the mobile wireless network architecture, and the WDM-based optical network architecture.

J-Sim has been developed entirely in Java. This, coupled with the autonomous component architecture, makes J-Sim a truly platform-neutral, extensible, and reusable environment. J-Sim also provides a script interface to allow integration with different script languages such as Perl, Tcl, or Python. In the current release, J-Sim is fully integrated with a Java implementation of the Tcl interpreter (with the Tcl/Java extension), called

Jacl. So, similar to ns-2, J-Sim is a dual-language simulation environment in which classes are written in Java (for ns-2, in C++) and "glued" together using Tcl/Java. However, unlike ns-2, classes/methods/fields in Java need not be explicitly exported in order to be accessed in the Tcl environment. Instead, all the public classes/methods/fields in Java can be accessed (naturally) in the Tcl environment.

4.1.3 The J-Sim Core Service Layer (CSL)

In J-Sim, a node is a composite component which consists of applications, protocol modules, and a core service layer (CSL).

The CSL is an abstract component which encapsulates the functions of the network layer and the layers beneath the network layer. It provides network services and events to protocols, in the form of inter-component contracts. Each service port is in charge of one or more CSL services.

4.2 MPLS Support in J-Sim

To support MPLS inside J-Sim, developers have done some modifications inside the simulator [24]. Two components have been added: a forwarding table component and a MPLS component. The forwarding table component keeps all information about configured labels. It associates an IP prefix or an incoming label with an outgoing interface and an outgoing label.

4.2.1 MPLS Model within J-SIM

To create an MPLS compliant node, a specific Core Service Layer (CSL) builder termed MPLSBuilder has also been developed. The forwarding table keeps information about labels: it links a label or an IP prefix to an outgoing interface and an operation list. This list contains an operator (SWAP, PUSH or POP) and a label as argument. These operators are applied on label carried in the packet.

A new type of packet has also been defined: the Label Switched Path packet (called LSP in the following). It is used to carry MPLS information like a stack of labels and another packet (an IP packet for instance). The stack of labels is used with the operator to store appropriate label inside the packet. These packets are used by the MPLS component of a node to bypass normal routing.

The MPLS component is located between the down port of the CSL and the packet dispatcher. This component receives packets from other nodes. According to the received packet and the configuration of the forwarding table, this component decides where the packet is sent. If it is an LSP packet, the MPLS component looks up inside the forwarding table and forwards the packet according to the record found. If it receives an IP packet, it also looks inside the forwarding table to see if it needs to encapsulate this packet inside

a new LSP (that's why the forwarding table can associate a label to an IP prefix). If no record is found for a given IP prefix, the packet is sent to the packet dispatcher and routed normally.

4.3 Implementation of Content-Aware MPLS Routing in J-Sim

In order to implement the idea of content-aware MPLS routing, certain changes are made to the existing implementation of the MPLS Class and network packet implementation.

Basically, a new packet format is designed as described in Section 3.2.3 and an enhanced MPLS router that incorporates the content-aware routing functionalities as described in Sections 3.2.6 and 3.2.7.

This section describes the implementation of the proposed scheme and the changes that are made to the current MPLS code in J-Sim in order to achieve the above.

4.3.1 New Packet Header for DWT Encoded Packets

By default, J-Sim only supports one type of standard IP packet encapsulated in a transport layer header with no unique identification. For the MPLS router to differentiate among regular packets and packets containing DWT encoded data, a unique identification is required along with other header fields that implement the real-time services such as timestamp and sequence

number as needed for RTP implementation. The required packet format is designed. This is done in the DwtPacket Class. A new feature is added that allows the user to tell any traffic generator in J-Sim to use which packets at the time of network modeling. By doing so, generating a mix of traffic that had both DWT encoded data as well as regular Internet traffic is made possible for the test runs.

4.3.2 Enhanced MPLS Router with Content-Aware Routing Functionalities

As described in Section 4.2, the MPLS implementation in J-Sim allows an MPLS compliant node to perform PUSH, SWAP and POP operations. In order to make the MPLS router smart enough to distinguish between regular packets and packets containing DWT encoded data and then assign routing priorities as described in Section 3.2.6, the proposed algorithm is integrated with the PUSH, SWAP and POP operations in the MPLS Class. Backward compatibility is maintained by adding a feature to select MPLS mode (regular MPLS or MPLS with Content-Aware Routing) for comparison purpose.

4.3.3 Additional Changes

J-Sim provides with tools for counting packets on interfaces and plotting results. The experiment runs required a mechanism to calculate average end-

to-end delay of packets traversing the MPLS network. However, the Simulator lacks in providing a delay counting tool. For this purpose, a delay counter is developed that counts delay on each packet received on the interface by subtracting current time from the value obtained in the timestamp field of the packet header. The counter is also capable of calculating the average delay of all the packets and displaying the result on the console. This is implemented in the DelayCounter Class.

4.4 Simulation Experiments

As mentioned in Section 3, the proposed scheme can serve a wide range of real-time, loss-tolerant applications as well as compressed images; both video streaming and compressed still image transfer over RTP with and without proposed and enhanced MPLS routers, is simulated. Delay characteristics of the DWT encoded traffic carried using proposed mechanism is observed and are compared with that carried over regular MPLS network and DiffServ enabled IPv4. Along with this, the proposed scheme is also tested in a hybrid network backbone consisting of MPLS and IP clouds.

DWT encoded traffic is simulated along with background IP traffic in order to analyze the impact on end-to-end delay, jitter and packet loss under normal circumstances as well as congested conditions. In order to simulate

the proposed algorithm, it is assumed that packets with sequence number 0 to *n* are high priority packets carrying essential frequency bands resulting from a DWT compression.

4.4.1 Experimental Design

The simulation experiments are based on the $2^k r$ Factorial Designs with Replications approach [67]. Each experiment consists of k factors and r independent runs for each data point to get it within 90% confidence interval of the mean. This allows estimation of experimental errors.

4.4.2 Experimental Parameters

As mentioned in Section 3.2.3, the size of a DWT encoded packet varies from 576 to 1468. For video packets, a packet size of 1468 bytes is chosen. This size is chosen to keep jitter values is low as possible. In general, local jitter increases as the maximum packet size decreases. This phenomenon is explained by the fact that more packets are generated for the smaller maximum packet size during the same time interval. More packets with a variety of inter-arrival times result in a greater likelihood of inter-arrival time variation and hence, jitter.

For still images, the average packet size is kept 576 bytes. An average packet size of background IP traffic is arbitrarily picked as 1000 bytes, which is sent at different bit rates in order to gradually congest the network and then

analyze average end-to-end delay, jitter and packet loss for DWT encoded packets. Another main reason for selecting these packet sizes is to avoid fragmentation.

4.4.3 Metrics

Following are the response variables (metrics)

- Average End-to-end delay
- Average Jitter (inter-arrival time variation of two consecutive packets [74])
- Packet loss

4.4.4 Factors

The factors affecting the response variables are:

- The congestion level quantity is defined as

  $$\rho = \lambda/\mu$$

  Where λ = rate of incoming background traffic

  μ = the capacity of the network

- Type of network backbone : MPLS or IP or MPLS with IP
- Type of QoS technique: MPLS, MPLS with proposed scheme or IP with DiffServ

4.4.5 Network Model

Figure 4.1 illustrates the network topology used for simulation. Nodes h0 and h1 are traffic sources generating different types of traffic destined to nodes h2 and h3. Node n4 is the ingress edge router of the MPLS cloud whereas node n9 is egress edge router. The core cloud consists of four switch routers; nodes n5, n6, n7 and n8.

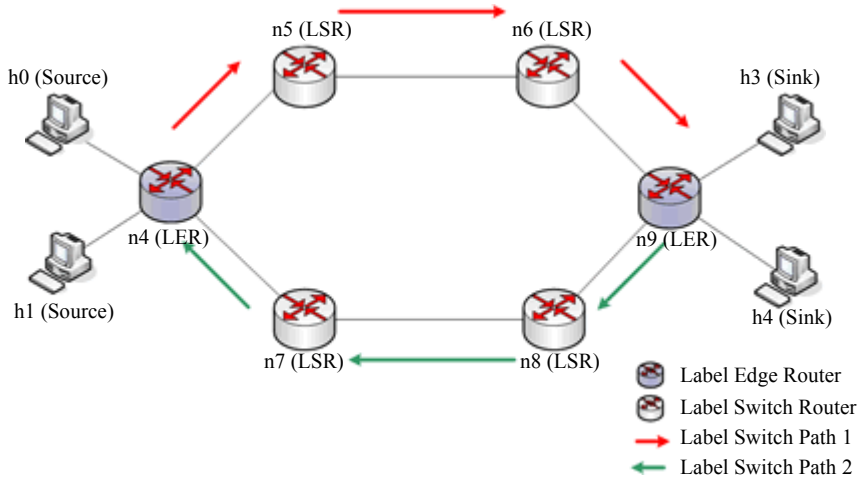


Figure 4.1: Network Topology for Simulation.

Two LSPs in the MPLS cloud are configured. Path 1 (n4 → n5 → n6 → n9) is the LSP from source to sink whereas LSP 2 (n4 → n7 → n8 → n9) is the path carrying traffic back to the source. Both LSPs support 100Mbps.

In order to compare the proposed scheme with regular MPLS and also with DiffServ, several experiments were conducted with the same simulation parameters and compared the average end-to-end delay, jitter and packet loss

of multimedia traffic on each of the three networks. Similar model is used for the DiffServ enabled IP backbone with all routers operating as DiffServ enabled devices.

4.4.6 Simulation-Based Evaluation

Transferring DWT encoded content over regular MPLS, MPLS with proposed scheme and DiffServ enabled IP network results in interesting and varying results. Under high network congestion, i.e. high values of ρ, both MPLS and MPLS with proposed scheme perform better in terms of delay and jitter. On the other hand, MPLS lags behind both MPLS with proposed scheme and DiffServ in terms of packet loss due to absence of any packet classification and marking scheme. The reason being that the routers cannot distinguish between packets of different applications and start dropping them randomly when the buffers reach a maximum.

4.4.6.1 Comparison of Average end-to-end Delay and Jitter

It is obvious from the results in Figure 4.2 that under higher values of ρ, the average end-to-end delay of video traffic is much high compared to that when transmitted using proposed scheme on the same link under same congestion level. Results also show that although DiffServ supports QoS, the delay of traffic is high. This is because DiffServ relies on IP routing algorithms that are slower compared to fast MPLS switching.

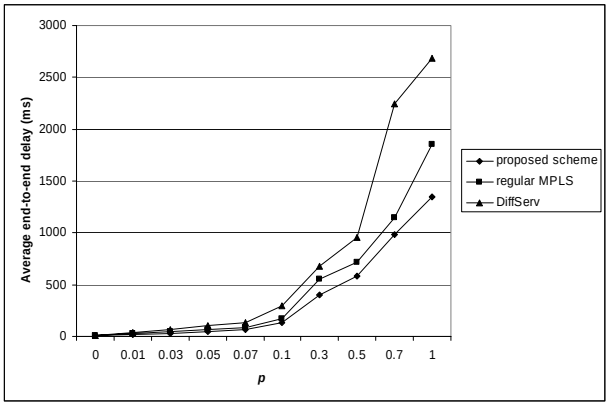


Figure 4.2: Comparison of average end-to-end delay of compressed video traffic over DiffServ, MPLS and MPLS using proposed scheme.

Similar trend is seen when simulating compressed image transfer over RTP. Here again, MPLS routers using proposed scheme excel regular MPLS devices and DiffServ enabled IP routers and maintain low end-to-end delay of RTP traffic. Results of these experiments are illustrated in Figure 4.3. Even at lower levels of congestion, content-aware congestion control enabled MPLS shows improvement in delay and jitter over DiffServ and regular MPLS, although all the three network backbones lie within the acceptable range [66]. At higher levels, only content-aware congestion control enabled MPLS maintains the toll quality whereas both DiffServ and regular MPLS approach the unacceptable region [66].

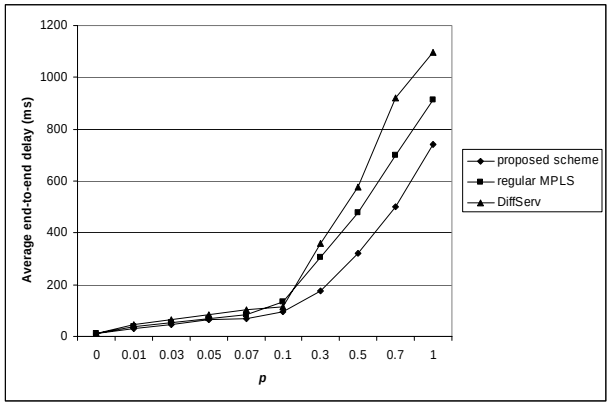


Figure 4.3: Comparison of average end-to-end delay of compressed images over DiffServ, MPLS and MPLS using proposed scheme.

Simulation results show that the proposed scheme maintains comparatively low jitter values. Figure 4.4 illustrates the difference in jitter values among DiffServ, MPLS and MPLS with content-aware congestion control when simulating DWT compressed video traffic.

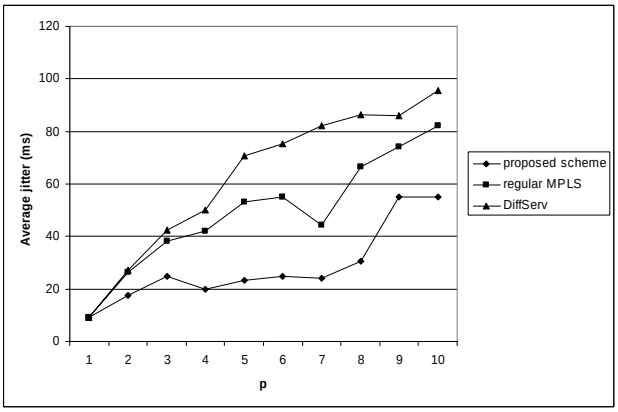


Figure 4.4: Comparison of average jitter of compressed video traffic over DiffServ, MPLS and MPLS using proposed scheme.

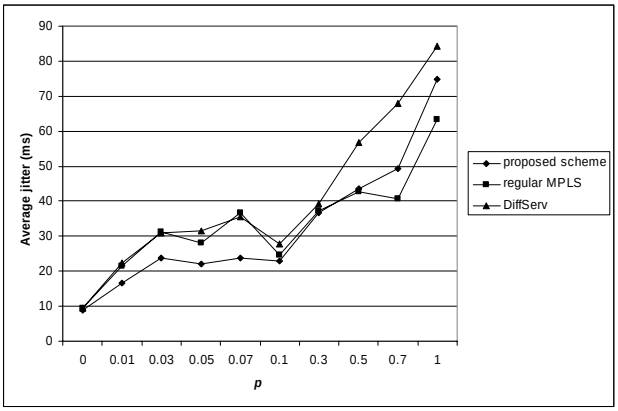


Figure 4.5: Comparison of average jitter of compressed image traffic over DiffServ, MPLS and MPLS using proposed scheme.

A similar trend is seen for DWT compressed images, where MPLS and IP routers fail to provide lower jitter values compared to the proposed scheme.

4.4.6.2 Comparison of Packet Loss

Packet loss in all the three cases is compared. Table 4.1 summarizes the results obtained when transmitting DWT encoded images at ρ = 1, i.e., at 100Mbps. It is clear from the results shown that DiffServ drops least number of DWT encoded data packets due to the fact that it is marking them as premium traffic and is not allowing background traffic to be injected into the network at times of congestion. On the other hand, in regular MPLS, since there is no traffic prioritization and all traffic (DWT encoded as well as background traffic), is treated similar, DWT encoded data packets are dropped along with background traffic.

Table 4.1. Comparison of image packet loss between DiffServ, MPLS and content-aware MPLS at $\rho = 1$.

Network Type	Packet Size (B)	Packets Sent	Packets Received	% Packets Received
DiffServ	576	116	75	64 %
MPLS	576	116	15	13 %
MPLS with proposed scheme	576	116	59	50.8 %

The packet loss of multimedia traffic on the congested LSP decreases using content-aware MPLS compared to regular MPLS. The reason being that the content-aware MPLS router identifies and prioritizes packets containing DWT encoded data. Also it divides and prioritizes DWT encoded data into two classes; hence the DWT encoded data packets that are dropped by content-aware MPLS are those of lower priority.

It can be seen from Table 4.1 that less packets are dropped in case if DiffServ compared to the proposed scheme. This phenomenon can be explained by the fact that DiffServ is not able to distinguish one single stream of DWT encoded data into two: high priority and low priority. Hence under high network congestion DiffServ can not drop low priority packets containing less important DWT content. This results in poor utilization of available network resources and also causes background traffic to starve. Table 4.2 summarizes the results obtained from simulating video traffic.

Table 4.2. Comparison of video packet loss between DiffServ, MPLS and content-aware MPLS at $\rho = 1$.

Network Type	Packet Size (B)	Packets Sent	Packets Received	% Packets Received
DiffServ	1468	2500	2355	94.2 %
MPLS	1468	2500	1945	77.8 %
MPLS with proposed scheme	1468	2500	2125	85 %

4.4.7 Heterogeneous Network Model

A real world scenario of a heterogeneous network backbone consisting of ingress and egress MPLS clouds along with an intermediate IP cloud is modeled and simulated to verify the performance of the proposed scheme. The logical view of the backbone is depicted in Figure 4.6.

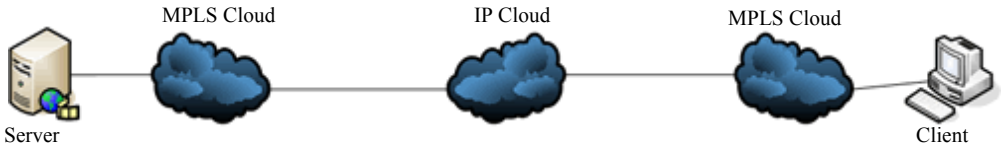


Figure 4.6: A heterogeneous network backbone.

In the scenario above, DWT encoded content reaches the client after passing through an intermediate IP cloud that does not support content-aware congestion control. Thus when the packets leave the first MPLS cloud and enter the subsequent IP cloud, they loose their respective priorities and are treated like any other data packet while traversing the IP cloud. When they reach the next MPLS cloud, only then the content-aware congestion control enabled MPLS routers re-prioritize and mark packets for QoS.

4.4.7.1 Comparison of Average end-to-end Delay and Jitter

Average end-to-end delay and jitter values were carefully examined at different values of ρ. It can be seen from the figures that an end-to-end content-aware MPLS backbone performs much better in terms of delay and jitter than a backbone with intermediate IP hops that do not support content-awareness and QoS.

Results show that the proposed scheme does not work well in such a case since the packets loose there respective priorities once out of the MPLS domain. The IP network may randomly drop packets due to congestion which may very well include high priority DWT data. This in turn increases delay as well as number of packets lost.

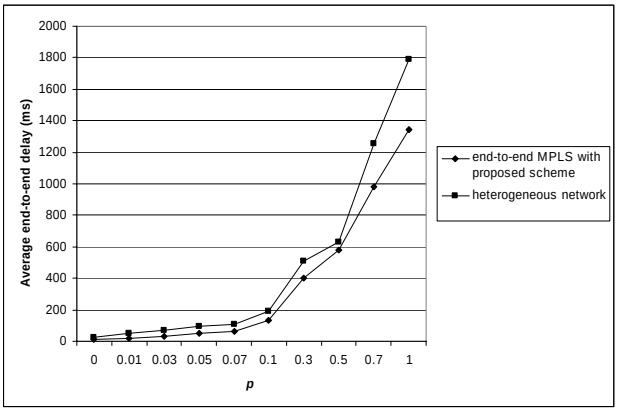


Figure 4.7: Comparison of average end-to-end delay of video traffic.

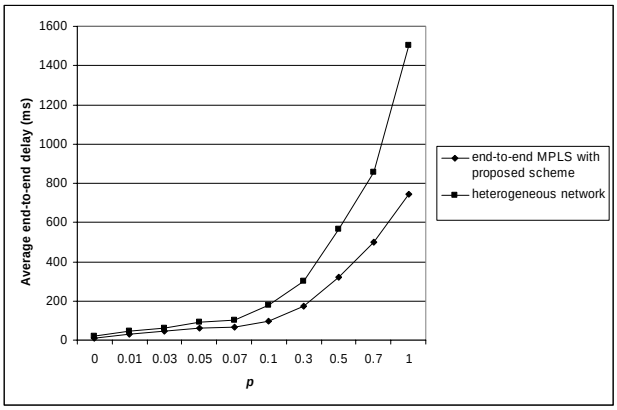


Figure 4.8: Comparison of average end-to-end delay of DWT encoded images.

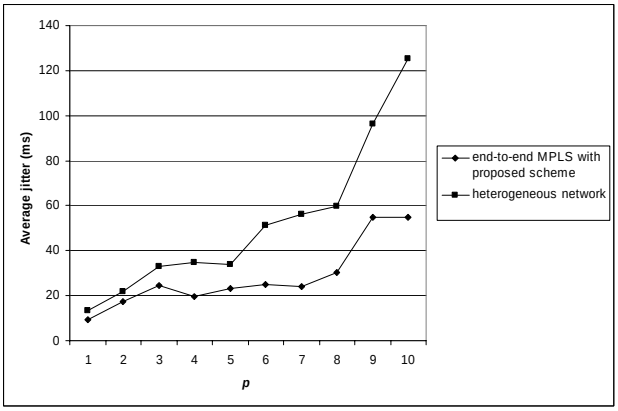


Figure 4.9: Comparison of average jitter of video traffic.

The delay and jitter graphs in Figures 4.7 - 4.10 show that the delay and jitter values are higher in case of heterogeneous network backbone when compared with an end-to-end MPLS backbone with content-aware support. This test supports the proposed applicability of the scheme as a viable QoS solution for end-to-end MPLS based EDNs managed by a single domain.

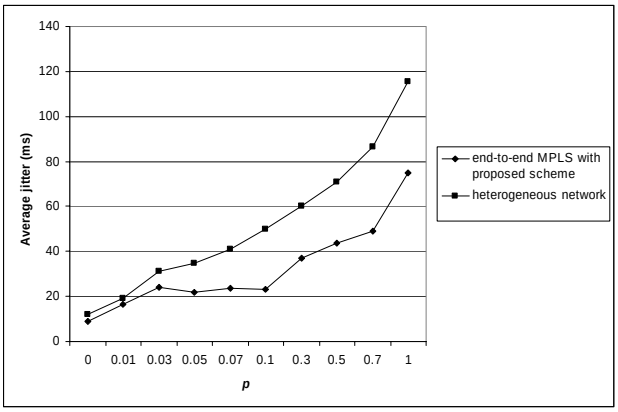


Figure 4.10: Comparison of average jitter of DWT encoded images.

4.4.7.2 Comparison of Packet Loss

It can be seen from Tables 4.3 and 4.4 that a large number of multimedia packets are being dropped by the heterogeneous network although these packets are being marked at the first MPLS cloud.

Table 4.3. Comparison of video packet loss at $\rho = 1$.

Backbone Type	Packet Size (B)	Packets Sent	Packets Received	% Packets Received
End-to-end MPLS	1468	2500	2125	85 %
Heterogeneous	1468	2500	960	38.4 %

Table 4.3 shows poor results in case of heterogeneous network. Only 38.4 % of the total traffic transmitted is received at the receiver's end. Whereas, under the same congestion level, 85 % of traffic is received when transmitted using an end-to-end content-aware MPLS backbone. Similar results are shown for DWT encoded images in Table 4.4

Table 4.4. Comparison of image packet loss at $\rho = 1$.

Backbone Type	Packet Size (B)	Packets Sent	Packets Received	% Packets Received
End-to-end MPLS	576	116	59	50.8 %
Heterogeneous	576	116	35	30 %

4.4.8 Service Ratings

Simulation results show that by employing the content-aware congestion control scheme into the MPLS router, the average end-to-end delay and jitter of DWT encoded traffic is maintained quite low even under high network congestion. From these results, the three services (DiffServ, MPLS, and MPLS using proposed scheme) can be rated in terms of viable solutions for transporting DWT encoded multimedia content. DiffServ is not suitable in terms of delay and jitter when compared to the fast switching capability of MPLS. MPLS further enhances the performance and maintains QoS when integrated with the proposed content-aware technique. The proposed scheme not only guarantees QoS but also manages and utilizes available resources in a more efficient manner allowing non-DWT traffic a fair chance at the network bandwidth. Table 4.5 summarizes the analysis. Despite of reasonable and acceptable results obtained by employing content-aware congestion control to MPLS networks, the proposed scheme only works well in an end-to-end pure MPLS backbone as reported from the results obtained in Section 4.4.7.

Table 4.5. Comparison of DiffServ, MPLS and MPLS using proposed scheme as a viable candidate for Multimedia Transportation.

Service	Content-Awareness	Traffic Engineering	Delay	Jitter	Packet Loss
DiffServ	No	No	Highest	Highest	Lowest
MPLS	No	Yes	High	High	Highest
Proposed MPLS	Yes	Yes	Lowest	Lowest	Low

4.5 Summary

This chapter presents modeling and simulation of the proposed scheme using J-Sim network simulator. It outlines structural details of the simulator and highlights the contribution made to the existing implementation in order to incorporate content-aware congestion control in MPLS networks. The chapter elaborates the experimental designs of the simulation and analysis of the model. It points out the metrics under study such as delay, jitter, packet loss and the factors that affect these metrics like type of network backbone and the type of QoS support. A detailed presentation and subsequent analysis of results obtained from the simulation experiments are presented followed by concluding remarks and observation. Results show that proposed scheme works better than regular MPLS and DiffServ in terms of delay and jitter although DiffServ shows better packet loss ratio. This in turn proves that proposed scheme utilizes available network bandwidth and also maintains

fairness among different flows of traffic. Another experiment proves that the proposed scheme works well with an end-to-end MPLS backbone with no intermediate IP cloud.

Chapter 5

Implementation of Content-Aware MPLS Router

This chapter discusses the prototype router developed for content-aware routing in MPLS networks. It also describes the Linux based MPLS implementation that is used to build the testbed and validate the router. The chapter also describes some of the key features of the Linux operating system that aided in the design and development of the content-aware router.

The key features of the router are as explained in Chapter 3. This chapter details how these features have been implemented on a Linux based public domain router.

5.1 The Linux iptables and netfilter

"netfilter" and "iptables" are building blocks of a framework inside the Linux 2.4.x and 2.6.x kernel [26]. This framework enables packet filtering, network address [and port] translation (NA[P]T) and other packet mangling. It is the re-designed and heavily improved successor of the previous Linux 2.2.x ipchains and Linux 2.0.x ipfwadm systems [27].

"netfilter" is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered

callback function is then called back for every packet that traverses the respective hook within the network stack.

iptables is a generic table structure for the definition of rule sets. Each rule within an IP table consists out of a number of classifiers (iptables matches) and one connected action (iptables target).

netfilter, iptables and the connection tracking as well as the NAT subsystem together build the whole framework.

Following are the main features of the framework.

1. Stateless packet filtering (IPv4 and IPv6)
2. Stateful packet filtering (IPv4)
3. All kinds of network address and port translation (NAT/NAPT)
4. Flexible and extensible infrastructure
5. Multiple layers of API's for 3rd party extensions
6. Large number of plug-in/modules kept in “patch-o-matic” repository [28]

5.1.1 Life of a packet within the Linux Box

Basically there is a chain for packets leaving, entering, or passing through the computer. Any packet entering the computer goes through the INPUT

chain. Any packet that the computer sends out to the network goes through the OUTPUT chain. Any packet that the computer picks up on one network and sends to another goes through the FORWARD chain. The chains are half of the logic behind iptables themselves.

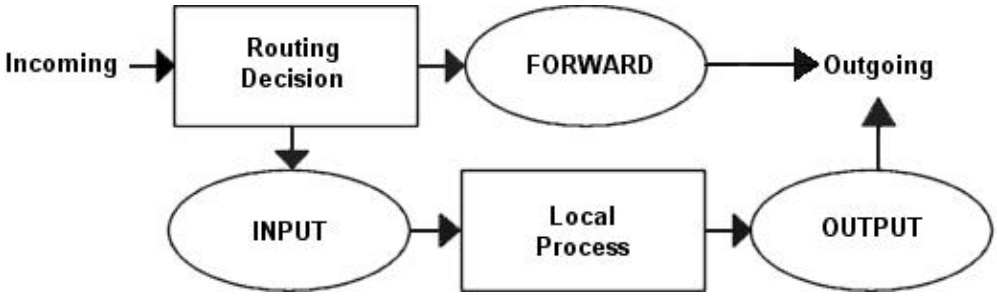


Figure 5.1: Traversal of packet within Linux router.

Now the way that iptables works is that certain rules are set in each of these chains that decide what happens to packets of data that pass through them. For instance, if the computer was to send out a packet to www.yahoo.com requesting an HTML page, it would first pass through the OUTPUT chain. The kernel would look through the rules in the chain and see if any of them match. The first one that matches will decide the outcome of that packet. If none of the rules match, then the policy of the whole chain will be the final decision maker. Then whatever reply Yahoo sent back would pass through the INPUT chain.

5.1.2 Matching Packets

Matching packets is, of course, the most important aspect of packet filtering setup. The two most basic match conditions are source and destination address of the packet, the first of which is discussed above. These can either be individual IP addresses or a whole subnet, depending upon what is to be achieved. If all packets heading to 192.168.1.2 from anything on the 10.0.0.0/8 network are to be blocked, the following command will perform this action.

```
iptables -A INPUT -s 10.0.0.0/8 -d 192.168.1.2 -j DROP
```

Match can also be based on protocol used, such as TCP, UDP, ICMP and so forth, as well as the specific port or service type used by that protocol. As an example, a common usage is to block connections to port 113 via TCP.

```
iptables -A INPUT -p tcp --dport 113 -j REJECT –rejectwith tcp-reset
```

Of course, protocol and source or destination address can be mixed into one whole rule, as appropriate.

```
iptables -I INPUT -p tcp --dport 113 -s 10.0.0.0/8 -j ACCEPT
```

When a protocol is specified, either the abbreviated name can be used, such as tcp, udp or icmp, or its numeric reference, 6, 17 and 1 respectively. If for some reason iptables complains about a protocol name, ensure that it is

defined in /etc/protocols, as the system uses that file to associate protocol names with the numeric assignments.

5.1.3 Targets/Jumps

The target/jumps tell the rule what to do with a packet that is a perfect match with the match section of the rule. There are a couple of basic targets, the ACCEPT and DROP targets. The jump specification is done in exactly the same way as in the target definition, except that it requires a chain within the same table to jump to. To jump to a specific chain, it is of course a prerequisite that that chain exists.

Targets on the other hand specify an action to take on the packet in question. Jumping to targets may incur different results, as it were. Good examples of such rules are DROP and ACCEPT. Rules that are stopped will not pass through any of the rules further on in the chain or in superior chains. Other targets, may take an action on the packet, after which the packet will continue passing through the rest of the rules. A good example of this would be the LOG, ULOG and TOS targets. These targets can log the packets, mangle them and then pass them on to the other rules in the same set of chains. Some targets will accept extra options (what TOS value to use etc), while others don't necessarily need any options. Table 5.1 summarizes different types of targets in iptables/netfilter.

Table 5.1. Targets in iptables and their functions.

Targets	Operation
ACCEPT	This target needs no further options. As soon as the match specification for a packet has been fully satisfied, and we specify ACCEPT as the target, the rule is accepted.
DROP	The DROP target does just what it says, it drops packets dead and will not carry out any further processing.
DNAT	The DNAT target is used to do Destination Network Address Translation.
SNAT	The SNAT target is used to do Source Network Address Translation.
LOG	The LOG target is specially designed for logging detailed information about packets.
MARK	The MARK target is used to set Netfilter mark values that are associated with specific packets.
MASQUERADE	The MASQUERADE target is used basically the same as the SNAT target, but it does not require any --to-source option.
QUEUE	The QUEUE target is used to queue packets to User-land programs and applications
REDIRECT	The REDIRECT target is used to redirect packets and streams to the machine itself
REJECT	The REJECT target works basically the same as the DROP target, but it also sends back an error message to the host sending the packet that was blocked.
RETURN	The RETURN target will cause the current packet to stop traveling through the chain where it hit the rule.
TOS	The TOS target is used to set the Type of Service field within the IP header
TTL	The TTL target is used to modify the Time To Live field in the IP header
ULOG	The ULOG target is used to provide user-space logging of matching packets

From all of the targets mentioned in Table 5.1, QUEUE is most important in terms of this research. The QUEUE target is used to queue packets to User-land programs and applications. It is used in conjunction with programs or

utilities that are extraneous to iptables and may be used, for example, with network accounting, or for specific and advanced applications which proxy or filter packets. In this case, it is the QUEUE target that will basically contribute the most in adding content-awareness to the router. This is explained in more detail in the latter sections.

5.2 QoS and Traffic Shaping

Simply put, traffic shaping is an attempt to control network traffic in order to optimize or guarantee performance, low-latency, and/or bandwidth [30]. Traffic shaping deals with concepts of classification, queue disciplines, enforcing policies, congestion management, quality of service (QoS), and fairness.

Despite (or maybe because of) the open and cooperative nature of the Internet, competition for available network resources tend to be unfair or selfish. Given that network bandwidth is a limited resource, traffic shaping allows prioritizing and managing network services.

Intelligently managed, traffic shaping improves latency, service availability and bandwidth utilization without any drawback (theoretically). Imagine a typical business that needs to connect to their headquarters for a mission critical financial application, but application performance is too slow because

of local users browsing the web or downloading multimedia content. Imagine the other scenario where a typical home user is running a P2P application and his roommates complain because web browsing is now unacceptably slow due to the saturated uplink.

Traffic shaping can provide:

1. Granular control of network services
2. More efficient use of limited/shared resources
3. Guaranteed QoS

5.2.1 Traffic Shaping Strategies

Generally speaking, egress (outgoing) traffic is more important than ingress (incoming) traffic. This is true for a couple of reasons. First, the network bottleneck on ingress traffic typically sits at the ISP (capped bandwidth). Second, while it's possible to have more or less completely control of the traffic sent out, the reverse is not true. While protocols like TCP may have flow control features, it's not always possible to utilize that to a full advantage.

A typical network today is running at least around 100mbits/s (bits) while a T1 connection is 1.5mbits/s. This means that when traffic leaves the network, it's going from a lot to a little - putting the bottleneck at the egress packet

queue. Whereas the reverse, the ingress queue, is likely never used because of the bandwidth disparity. Besides, when policing ingress policies, perfectly good packets (already received) gets dropped and have to be retransmitted. Because of this, ingress queues are still rather controversial.

5.2.1.1 Packet Queues

A packet queue is basically a buffer. When the amount of packets leaving exceeds the gateway router's ability to send them, it typically queues up packets until it's possible. If the packet queue overflows, then the packets are silently dropped. If a timeout occurs because a packet sits in the queue for too long, the packet gets resent making the queue even more likely to overflow (necessitating even more resends and their associated timeouts).

All network devices utilize a packet queue. Linux by default has a packet queue length of 100, meaning that it can buffer up to 100 packets before it starts to silently drop packets. Most ISPs also configure packet queues to be significantly larger, in order to avoid resends. DSL and cable modems have their own packet queues as well.

What this means that a packet might take several seconds to make it through all of the egress queues before it actually reaches the first hop of Internet. The sum result of this is very high latency when the network link becomes congested. If the latency becomes sufficiently high, timeouts may occur and

necessitate packet resends making a bad situation worse. Packets already sent and received could appear to have timed-out (not acknowledged in time due to the latency). Resends further deepens the packet queues.

5.2.1.2 QoS Guarantees

Other considerations such as minimum guaranteed bandwidth for specific network services, or a maximum bandwidth cap for certain departments or clients might also be important when it comes to traffic shaping policies.

Suppose the accounting department runs Citrix clients in order to update financial records at the headquarters, but the network is completely swamped by the sales people downloading the latest Apple iPod commercials on Kazaa. Traffic shaping can set aside a minimum guaranteed bandwidth and latency for the accounting department, but allow sales to utilize the rest of the bandwidth when the pipe lays idle.

5.2.2 Queue Disciplines

A queue discipline is a strategy for managing a "queue." Imagine standing in line in the post office vs. waiting in the emergency room. Both have "items" in the queue that needs to be cleared in some manner, but have very different strategies. Post offices typically use a first in first out strategy (classless FIFO). Customers are served in the order that they've arrived in the queue. Other the other hand, this is an unacceptable strategy for managing an

emergency room (prioritized classful). Someone in a critical condition requires urgent attention regardless of their order of arrival. Suppose 10 people all show up at the same time, and there are only enough resources to deal with two people, what needs to happen? First the queue (or line) needs to be sorted into classes (maybe critical, urgent, non-urgent, can-wait-indefinitely). Then the queue is emptied based on priority of the different classes.

In Linux, queuing disciplines can be divided into two main groups: Classless Queuing and Classful Queuing.

5.2.2.1 Classless Queuing Disciplines

Classless queuing disciplines are those that, by and large accept data and only reschedule, delay or drop it. These can be used to shape traffic for an entire interface, without any subdivisions. By far the most widely used discipline is the pfifo_fast qdisc - this is the default.

5.2.2.2 Classful Queuing Disciplines

Classful queues have the ability to classify and prioritize traffic. Since this study mainly focuses on QoS, classful queues are explained more in detail.

When traffic enters a classful qdisc, it needs to be sent to any of the classes within - it needs to be 'classified'. To determine what to do with a packet, the

so called 'filters' are consulted. It is important to know that the filters are called from within a qdisc, and not the other way around.

The filters attached to that qdisc then return with a decision, and the qdisc uses this to enqueue the packet into one of the classes. Each subclass may try other filters to see if further instructions apply. If not, the class enqueues the packet to the qdisc it contains.

Besides containing other qdiscs, most classful qdiscs also perform shaping. This is useful to perform both packet scheduling and rate control. There are 4 main classful qdiscs in Linux. For this implementation, the PRIO qdisc is used along with the SFQ discipline to prioritize and classify traffic flows and to keep fairness among different priority levels.

Table 5.2. Queuing Disciplines in Linux.

Qdisc	Type	Description
pfifo_fast	Classless	First-in, First-out. No classification of traffic
Token Bucket Filter (TBF)	Classless	passes packets arriving at a rate which is not exceeding some Administratively set rate
Stochastic Fairness Queue (SFQ)	Classless	implementation of the fair queuing algorithms family
PRIO	Classful	Classful queuing discipline that contains an arbitrary number of classes of differing priority
Class Based Queuing (CBQ)	Classful	CBQ is a classful qdisc that implements a rich link sharing hierarchy of classes
Hierarchical Token Bucket (HTB)	Classful	Multiplexes a fixed amount of bandwidth into different classes, guaranteeing each class a specified amount of bandwidth

5.2.2.3 The PRIO qdisc

The PRIO qdisc doesn't actually shape, it only subdivides traffic based on the configuration of filters. When a packet is enqueued to the PRIO qdisc, a class is chosen based on the filter commands. By default, three classes are created. These classes by default contain pure FIFO qdiscs with no internal structure, but can be replaced by any qdisc available.

Whenever a packet needs to be dequeued, class 1 is tried first. Higher classes are only used if lower bands all did not give up a packet.

This qdisc is very useful in case certain kind of traffic is to be prioritized without using only TOS-flags but using all the power of the tc filters. It can also contain more or all qdiscs.

5.2.2.4 Stochastic Fairness Queuing (SFQ)

This is a useful queue discipline when dealing with very full queues, and especially combined with a classful queue discipline. Traffic is separated into conversations and packets are dequeued in a round-robin fashion ensuring that no single conversation completely swamps the queue. It is a classless queuing discipline.

SFQ is a simple implementation of the fair queuing algorithms family. It's less accurate than others, but it also requires fewer calculations while being almost perfectly fair.

The key word in SFQ is conversation (or flow), which mostly corresponds to a TCP session or a UDP stream. Traffic is divided into a pretty large number of FIFO queues, one for each conversation. Traffic is then sent in a round robin fashion, giving each session the chance to send data in turn. This leads to very fair behavior and disallows any single conversation from drowning out the rest. SFQ is called 'Stochastic' because it doesn't really allocate a queue for each session; it has an algorithm which divides traffic over a limited number of queues using a hashing algorithm.

5.3 The MPLS-Linux Project

The Linux based MPLS implementation is used for the proposed prototype and its measurement based evaluation [31]. MPLS for Linux is a project to implement a MPLS stack for the Linux kernel, and portable versions of the signaling protocols associated with MPLS.

There are several reasons to choose the Linux based version, one of the more important reasons is that Linux provides an extensive set of traffic control functions and mechanisms [32]. Since the experiments are closely coupled to QoS, flexibility was a major concern.

MPLS-Linux is a recent implementation of MPLS for PCs running the Linux operating system. MPLS-Linux is freely modifiable under the GNU license

and conforms to the MPLS specifications. Other MPLS implementations for PCs have been proposed in the past [33][34], but are not maintained by their authors. Thus, MPLS-Linux is chosen to implement the content-aware routing mechanism on PCs. Before explaining how MPLS-Linux is extended, some background information is provided on the existing MPLS-Linux implementation. MPLS-Linux is implemented as a layer between Ethernet and IP. Ethernet is a MAC layer protocol which encapsulates IP packets in frames. In Sections 1.3.1 and 3.2, an overview of the three operations that MPLS routers can perform on packets (push, swap and pop) are described and in Section 1.3.2 description of the Forwarding Information Base (FIB) is provided which contains the rules according to which MPLS routers forward packets. This section describes how the MPLS operations and the FIB are implemented in MPLS-Linux.

Table 5.3. MPLS-Linux unicast instructions overview.

Instruction	Input Layer	Output Layer	Description
PUSH	IP	MPLS	Adds a shim header to an IP packet
SET	MPLS	Ethernet	Passes an MPLS unicast packet to an Ethernet interface.
POP	Ethernet	MPLS	Removes a shim header from an Ethernet frame.
FWD	MPLS	MPLS	Calls PUSH for a packet coming from POP.
DLV	MPLS	IP	Passes an MPLS packet to the IP layer.

MPLS-Linux defines five instructions to implement shim header pushing, swapping and popping. Each of these instructions can be applied to IP packets or Ethernet frames in the MPLS layer as they are being processed by the Linux kernel. Overview of these five instructions is provided in Table 5.3 and description on how they implement the three MPLS operations in Table 5.4. The PUSH instruction adds an MPLS shim header to a packet which comes from the IP layer. The SET instruction passes an IP packet with a shim header from the MPLS layer to the Ethernet layer and tells the Ethernet layer on which Ethernet interface the MPLS packet should be forwarded. Together, the PUSH and SET instructions implement the MPLS ``push'' operation. The POP instruction removes the shim header of a packet that comes from the Ethernet layer. Packets processed by POP must be subsequently processed by either FWD or SET. The FWD instruction takes as an input a packet processed by POP and calls the PUSH instruction. Together, the POP, FWD, PUSH and SET instructions implement the MPLS ``swap'' operation. Last, the DLV instruction takes as an input a packet processed by POP and passes it to the IP layer. The POP and DLV instructions implement the MPLS ``pop'' operation.

Table 5.4. Implementation of the three MPLS operations with the five MPLS-Linux instructions.

MPLS Operation	Corresponding sequence of instructions in MPLS-Linux
Push	PUSH, SET
Swap	POP, FWD, PUSH, SET
Pop	POP, DLV

5.3.1 Implementation of FIB in MPLS-Linux

In MPLS-Linux, the FIB is split into three tables: the MPLS input and output tables, and the IP routing table. MPLS-Linux defines a Forwarding Equivalence Class (FEC) with a prefix and a prefix length. A prefix is a 32-bit IP address and a prefix length is a number comprised between 1 and 32. A packet with destination IP $\boldsymbol{IP_d}$ matches the FEC $\boldsymbol{P/P_{len}}$ constituted by the prefix $\boldsymbol{P}$ and the prefix length $\boldsymbol{P_{len}}$ if and only if the first $\boldsymbol{P_{len}}$ bits of $\boldsymbol{IP_d}$ and $\boldsymbol{P}$ are the same. A requirement of MPLS-Linux is the presence in the IP routing table of a specific entry for each FEC that is defined at MPLS ingress LER. It is not possible to define a FEC if no matching entry exists in the routing table. Indeed, MPLS-Linux relies on the IP routing table to determine the FEC of an IP packet. In MPLS-Linux, IP routing table entries are extended and contain FEC to Next Hop Label Forwarding Entry (FTN) mappings in addition to the IP routing information. Both the MPLS input and output table contain Next Hop Label Forwarding Entries (NHLFEs), while the MPLS input table implements the Incoming Label Map (ILM).

Figure 5.2(a) shows how a shim header is pushed on an incoming Ethernet frame by ingress LER. The Ethernet layer of the LER receives a frame with a protocol field in the Ethernet header set to 0x0800, which is the protocol code for IPv4. The Ethernet layer passes the incoming frame to the IP layer. The MPLS router searches for an entry in the IP routing table to make the routing decision, but since this entry matches a FEC it has been modified so that the packet is passed to the MPLS layer instead of being routed by the IP layer. The additional information contained in the IP routing table is a FTN, that is, a pointer to an MPLS output table entry. This output table entry is a NHLFE that contains two instructions. A PUSH instruction defines the label number of the packet, and a SET instruction defines on which interface the packet should be sent on.

The MPLS layer adds at the beginning of the packet an MPLS header which contains the label found in the NHLFE, and passes the packet to the Ethernet layer. The Ethernet layer generates a frame with the protocol field set to the code assigned to MPLS unicast packets (0x8847) and sends the frame over the wire. Consider now Figure 5.2(b) which shows how a label is swapped by a LSR.

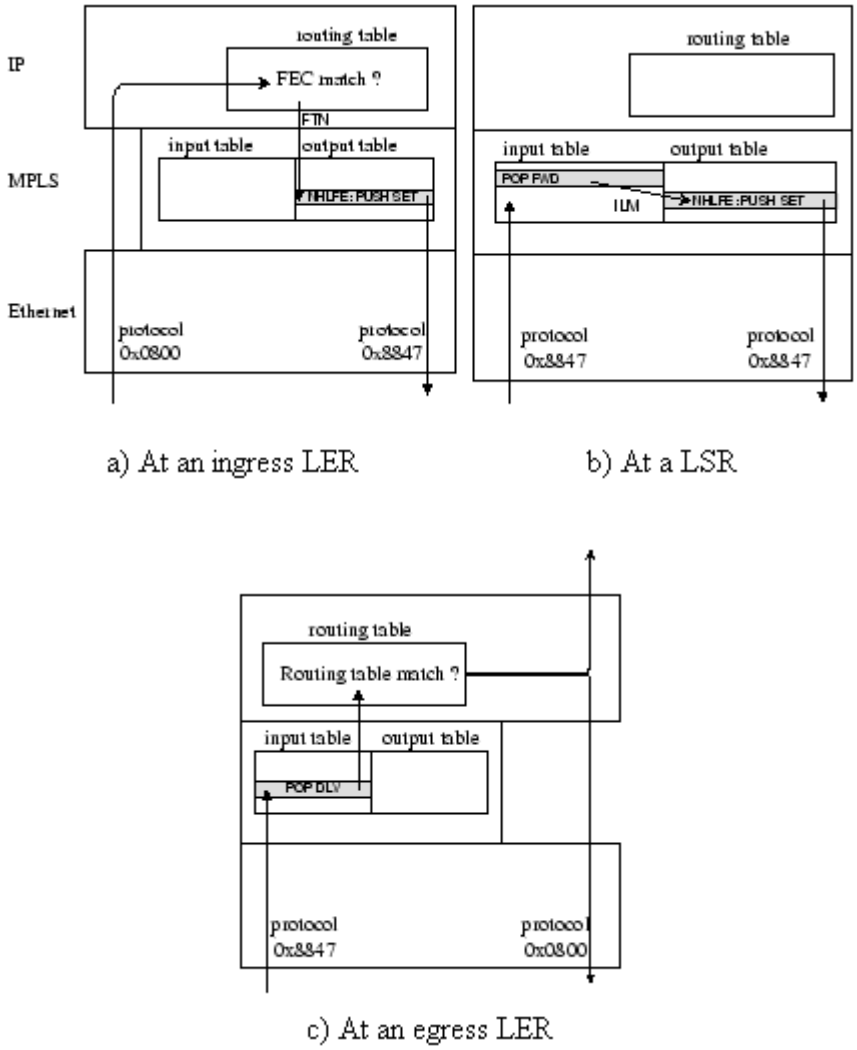


Figure 5.2: Processing of a packet in the MPLS layer with MPLS-Linux unicast.

The Ethernet layer of the LSR receives a frame with a protocol field in the Ethernet header set to 0x8847. Since 0x8847 is the code assigned to MPLS unicast packets encapsulated in Ethernet frames, the Ethernet layer passes the frame to the MPLS layer of the LSR. The MPLS layer searches in the MPLS

input table for the entry that matches the label embedded in the shim header of the packet. The input table implements the ILM and tells the MPLS layer what to do with the packet. The input table entry contains two instructions. The POP instruction tells the LSR to remove the MPLS header, and the FWD instruction points to an entry of the MPLS output table. This entry in turn contains two instructions: the PUSH instruction contains the new label for the packet and tells the LSR to add a shim header on the packet with this new label, while the SET instruction tells the LSR on which Ethernet interface the packet should be sent. The Ethernet layer then builds a frame with a protocol field of 0x8847 and sends it over the wire. By definition, the NHLFE tells an MPLS router whether a header must be popped or swapped. In MPLS-Linux the SWAP operation is implemented by successively popping and pushing a shim header, and the instructions required to pop and push a label are located in each of the MPLS tables. In this case, the NHLFE is contained at the same time in the input table and the output table.

Last, consider Figure 5.2(c) which shows how a label is popped by an egress LSR. The Ethernet layer of the LSR receives a frame with a protocol field in the Ethernet header set to 0x8847 and therefore passes the frame to the MPLS layer. The MPLS input table entry that matches the label of the packet contains two instructions. The POP instruction tells the LER to remove the shim header from the packet, and the DLV instruction tells the LER to pass

the packet to the IP layer where it will be processed like any other IP packet. In this case, the NHLFE is fully contained in the input table entry and tells the packet to pop the shim header.

Labelspaces define the scope of forwarding rules. If two interfaces of the same MPLS router belong to the same labelspace, then they apply the same set of forwarding rules to MPLS packets. For example, if interfaces ``2'' and ``4'' are part of the same labelspace, then two packets with the same label arriving one on interface ``2'' and the other on interface ``4'' will follow the same forwarding rule. On the other hand, if multiple interfaces do not belong to the same labelspace then the incoming MPLS packets follow different forwarding rules. In our implementation, we do not use labelspaces and for each Ethernet interface we set the labelspace to be equal to the interface index assigned by the kernel.

5.4 Content-Aware MPLS Router Architecture

So far, in Linux, QoS is incorporated within the implementation integrating DiffServ with MPLS [64]. The architecture of the proposed content-aware MPLS router is independent and does not involve assimilation of other technology. Just like any other MPLS router, the proposed content-aware MPLS router has 3 modes of operation: as ingress, switch or egress router. In all of these 3 modes, the router is carefully programmed to differentiate,

prioritize, and then apply specific MPLS operations on the packets based on classification and prioritization. This section will describe in detail the design and function of the architecture of the proposed router in all of the above mentioned operational modes.

5.4.1 Ingress Mode

In the ingress mode, the router has three main modules that perform content-aware MPLS routing. Figure 5.3 shows the internal logical structure of the router and the different modules and their relationship.

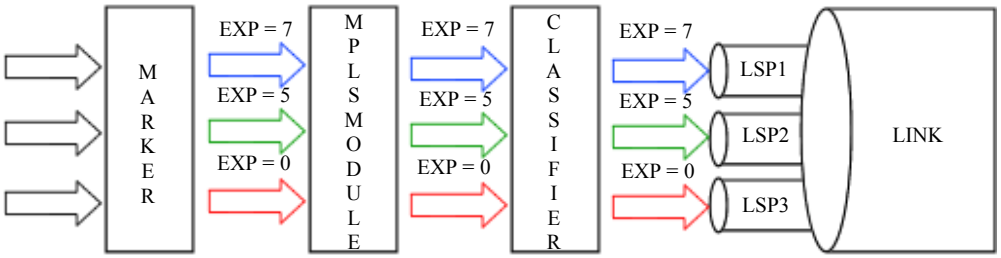


Figure 5.3: Modules of proposed content-aware router in ingress mode.

5.4.1.1 Marker – Content-Aware Routing Module

The marker is the module that takes all incoming packets from the incoming interface and then analyzes packet header and marks packets on the basis of some predefined policy.

By default, in Linux, all incoming packets at a particular network interface are taken up by the kernel and routed according to defined routing policies. This is done using iptables and netfilter as explained in the beginning of this

chapter. In the current implementation of iptables and netfilter, there is no support for RTP/RTSP and since the multimedia client and server communicate over RTP, there is a need to incorporate some mechanism into the Linux router to understand RTP packets. For this purpose the extended version of the libipq library is used. This library is available with the iptables to design userspace packet identification and marking applications [35]. By using this library, an application is developed that takes all incoming packets from the kernel and then analyzes their header and payload. For the purpose, packets with RTP version number and payload type are matched for identification since the DWT application server adds a different and fixed value to the payload type field while prioritizing packets. The pseudo code that implements this module is given below.

For all incoming packets, check the header and verify the following conditions:
if RTP packet with payload type 95
set nfmark 1 and ACCEPT
else if RTP packet with payload type 96
set nfmark 2 and ACCEPT
else
set nfmark 3 and ACCEPT

The nfmark above corresponds to the mark on the packet as described in Table 5.1.

The Marker basically looks at the value of the payload type field in the RTP header and then marks packets and passes them to the next module.

5.4.1.2 Classifier – QoS Module

This module performs all the queuing functions at the egress interface of the router. It implements the PRIO and SFQ queuing discipline as described in Section 5.2.2.3 and 5.2.2.4. The module then classifies packets on the basis of the mark values returned by the Marker module.

The Classifier creates three priority queues, one for each flow of traffic. The module implements packet classification rules that directs packets carrying high priority DWT packets to the queue with highest priority to dequeue packets, low priority DWT packets to the queue with second highest priority and rest of the traffic to the queue with the least priority.

In order to maintain fairness among the three levels of priority, the Classifier also implements the Stochastic Fairness Queue discipline. The SFQ discipline prevents low priority flows from starvation as it checks after a fixed amount of time whether any higher priority flow is eating up all the resources.

5.4.1.3 MPLS Module

Finally, the MPLS module creates LSPs on the outgoing interface and assigns appropriate labels and EXP bits values to outgoing traffic. As

mentioned in Section 3.2.6, high priority DWT application data packets will be encapsulated in MPLS header with EXP bits set to value 7, whereas packets containing less important data will be assigned EXP bits equal to 5, this module implements this core feature of the content-aware router. Once the packets are marked and classified by the first two modules, the MPLS module knows exactly what packets will be assigned what EXP value and forwarded on which LSP. All the packets marked 1 (high priority DWT application data packets) by the Marker module and classified by the Classifier are encapsulated with MPLS header carrying EXP bits set to 7 and forwarded on the appropriate LSP.

5.4.2 Switch Mode

The switch mode is kept simple and scalable to maintain the useful fast forwarding paradigm of MPLS. In this mode there are only two modules that perform content-aware MPLS switching. In fact the switch router is unaware of the content of the packets it has to switch. It only relies on the information provided to it by the ingress or the previous switch router in the network.

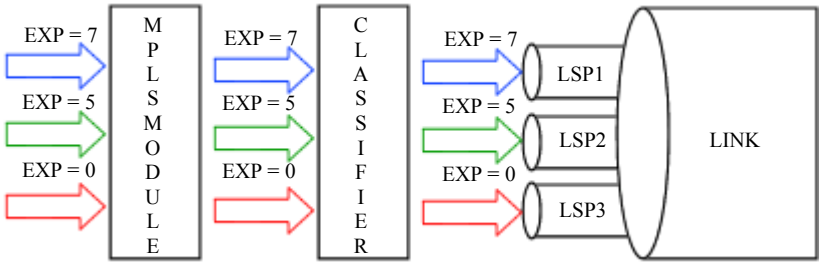


Figure 5.4: Modules of proposed content-aware router in switch mode.

The MPLS module, in the figure above, analyzes, for every incoming packet, the MPLS header and reads the EXP bits value. According to the value received, for every pre-defined flow, the module marks the FEC and then delivers the packets to the next module. The function of the Classifier Module is exactly the same as the one in ingress mode. Baaed on the mark value received from the MPLS Module, the classifier classifies and queues packets in accordance with their respective priorities.

It is clear from the figure above that the EXP bits value is maintained between the two modules, but that is only logical. In actual, the Classifier reads the mark on the FEC and enqueues packets.

5.4.3 Egress Mode

At the egress, the router finally removes the MPLS header and forwards the packet to the IP link. But before doing so, the router maintains the priorities of the FECs. This is the last mile of the end-to-end MPLS QoS using

content-awareness. A congested IP link beyond this point will not maintain the priorities assigned by the MPLS ingress.

In the egress mode, just like the switch mode, there are also two modules. the only difference is that now, the router removes MPLS header and forwards packet based on destination IP address instead of assigning label and maintaining EXP bits value.

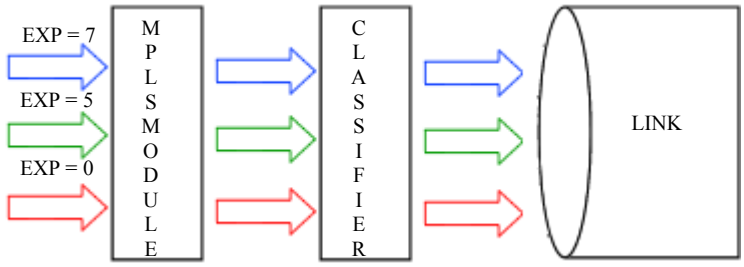


Figure 5.5: Modules of proposed content-aware router in egress mode.

5.5 Summary

This chapter presents in depth description of the implementation of the content-aware congestion control enabled MPLS router over a public domain Linux based IP router. The chapter explains IP routing functionalities in Linux such as iptables and netfilter. It also highlights some key aspects of Linux in terms of traffic shaping, packet queuing and QoS.

The chapter presents details of the MPLS-Linux project, on top of which the content-aware congestion control scheme is built. Finally, implementation of

the prototype router is presented with explanation of its different modes, how each mode operates and interacts with other modes. Description of the ingress, switch and egress modes of the proposed MPLS router is presented with the help of illustrations.

Chapter 6

Measurement Based Performance Evaluation

In this chapter, an evaluation of the proposed content-aware congestion control routing scheme is presented. The proposed scheme is experimentally validated. For the purpose of experimental evaluation, an MPLS enabled network test-bed is installed. The test-bed consists of label edge and label switch routers along with a content server that provides DWT encoded data on request by the client. Section 6.1 describes the experimental MPLS test-bed configuration, hardware and software platform used for the test-bed and the client/server application. Section 6.2 elaborates the goals and hypothesis of the experiments, details like traffic characteristics, metrics to be measured and experimental design are described in Section 6.3. Sections 6.4 and 6.5 provide in-depth analysis of experimental results obtained from different tests and compare the results to those presented in Chapter 4.

6.1 Experimental Test-bed

The MPLS test-bed consists of Linux based MPLS enabled software routers and Microsoft Windows based client and server. Figure 6.1 illustrates the topology of the test-bed.

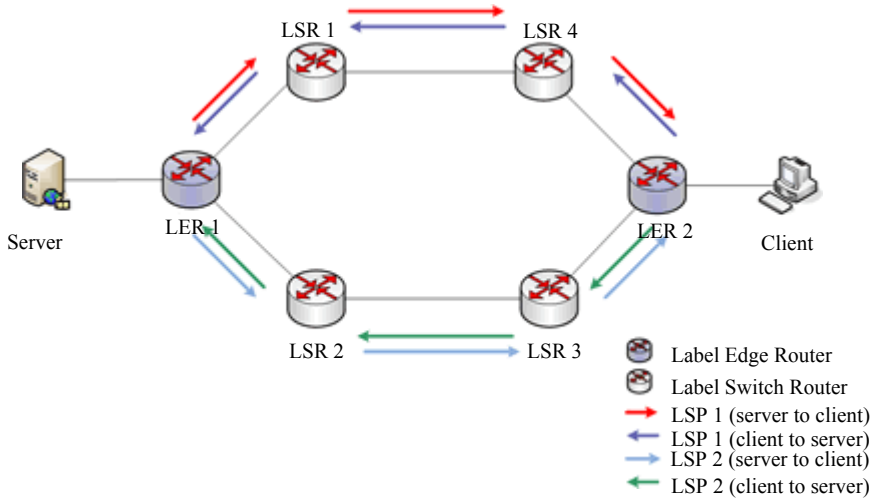


Figure 6.1: Experimental test-bed.

All the MPLS routers in the figure above are running both the regular MPLS with no congestion control capabilities as well as with the content-aware congestion control technique.

6.1.1 Hardware/Software Platform

The experimental test-bed consists of 8 PCs. Two PCs are used for sending and receiving DWT encoded traffic as well as generating background traffic for emulating a congested network. The PC based routers are all Pentium III hosts with 600 MHz CPU, 128 MB RAM running RED HAT Linux 9.0 with kernel 2.6.1. MPLS is enabled through patching the kernel along with iptables and iproute files. Table 6.1 summarizes the hardware and software resources utilized for the experiments.

Table 6.1. Hardware/Software resources used in test-bed.

Processor	Memory	Operating System	Functional Status	Number
PIII – 600MHz	128 MB	RED HAT Linux 9.0	MPLS Router (LER and LSR)	6
PIII – 600MHz	128MB	Windows 2000 Server	Client	1
PIV – 2.0GHz	512MB	Windows XP Professional	Content Server	1

6.1.2 Traffic Generation Tools

Traffic on the MPLS test-bed is mainly of two types; DWT encoded data from server to client and background network traffic to emulate network congestion. IPERF [36] is used as background traffic generator due to its capability of producing large amount of traffic with characteristics similar to that of traffic on the Internet. It is generally used to test the effective bandwidth of a network.

For DWT encoded traffic, a dedicated client/server application is developed that transfers DWT encoded images over RTP as the transport protocol. The client requests a known content server for compressed images. The server application, on receiving client request, starts to send the images. The application is capable of assigning priorities to different packets as it sends packets on the network. The design of the application is made such that it understands the EZW approach and assigns high priority to packets that are high in order and are sent first on the wire. It uses the payload type field in

the RTP header to assign a value that reflects the priority of the packets. It breaks the stream into two and assigns value 95 to the first half (high priority packets) and 96 to the rest. These values are unreserved for experimental purposes [37].

On the receiving end, the client application receives the compressed image and hands it over to an EZW decoder [38] that decodes the image and displays it on the client's screen.

6.2 Goals and Hypotheses of Experimental Verification

The goal of this experimental validation is to verify whether the proposed content-aware congestion control scheme over MPLS maintains reasonably acceptable image quality even under high network congestion, i.e. achieving graceful degradation compared to regular MPLS network.

The hypotheses of this validation can be stated as follows:

- Hypothesis H1: Image quality will be comparatively better under high network congestion in case of content-aware congestion control over MPLS.
- Hypothesis H2: Only important parts of the image will be preserved and rest dropped by the router in case of high congestion.

- Hypothesis H3: Content-aware congestion control enabled MPLS will show less delay, jitter and packet loss compared to regular MPLS and IP with DiffServ.

The main metrics to be calculated for the experimental verification of the proposed scheme are packet loss, received file size, delay and jitter. Image quality is the only non-numeric measure of the efficacy of any congestion control scheme.

6.3 Experimental Design and Parameters

As illustrated in Figure 6.1, the client is connected to the content server via an MPLS backbone network. It requests for a DWT encoded image of size 64KB. This simple file transfer is conducted under several different levels of network congestion by gradually increasing the rate of background IP traffic. Characteristics of background traffic are kept very similar to those used for the simulation based analysis. Packet size of background traffic is 1000B, which is injected at a gradually increasing rate starting from 128 kbps up to 100 Mbps. Packet loss, jitter, delay and image quality are carefully measured under each congestion level and then compared among different scenarios.

6.3.1 Metrics

Following are the response variables (metrics)

- Packet Loss
- Received File Size
- Average end-to-end delay
- Average jitter

6.3.2 Factors

The factors affecting the response variables are:

- The congestion level quantity is defined as

  $$\rho = \lambda/\mu$$

  Where λ = rate of incoming background traffic

  μ = the capacity of the network

- Type of network backbone : MPLS or IP or MPLS with IP
- Type of QoS technique: MPLS, MPLS with proposed scheme or IP with DiffServ

6.3.3 Experimental Scenarios

In order to present a comprehensive experimental comparison, a number of scenarios were carefully designed. The proposed content-aware congestion control technique is compared for the above mentioned metrics with regular MPLS and with DiffServ. Delay, jitter, packet loss and image quality is

compared to test the performance of the proposed scheme and verify results obtained from the simulation based analysis presented in Chapter 4.

6.4 Analysis of Results

Transferring DWT encoded images over regular MPLS network results in poor image quality when the network is experiencing high congestion. A router cannot distinguish between packets of different applications and start dropping them randomly when the buffers reach a maximum. In turn, the important parts of the image are lost (packets that are first in the stream), which results in poor quality image or even a corrupt file due to very high packet loss rate. On the other hand, since DiffServ is classifying and prioritizing DWT data packets, the overall packet loss rate is much less in this case and hence a good quality of image is maintained. One drawback of this, however, is that DiffServ at all times provides premium service to DWT encoded data, which in turn leads to starvation for regular network traffic. Also, delay and jitter values are more in case of DiffServ when compared to MPLS and content-aware congestion control enabled MPLS due to the fact that slower IP address lookups are the basis of routing in a pure DiffServ network. Transferring the same file at the same rate over content-aware congestion control enabled MPLS backbone shows that even under high congestion, although the packet loss increases but overall image quality is

comparatively better than the previous case. This phenomenon can be explained by the fact that under content-aware routing, the routers are capable of distinguishing packets from the information within the header. Thus under high congestion all packets that contain less important parts of the image are dropped and only high priority packets are allowed into the network. This results in achieving graceful degradation. It also allows background traffic a fair chance at the network resources and efficient usage of available bandwidth.

6.4.1 Comparison of Packet Loss

Packet loss of DWT traffic is calculated at each point by increasing the value of ρ. Table 6.2 shows packet loss in all the three cases at the highest value of ρ, when the background traffic is injected at a rate of 100 Mbps. At this point, regular MPLS network halts and all packets are being dropped. Since both DiffServ and MPLS with proposed scheme implement queuing for QoS, DWT traffic is getting premium service and hence lesser packets are dropped. However, since the proposed scheme is performing classification and queuing on the basis of the content of the packet, a feature that is not present in DiffServ, more packets are dropped after a point because the packets with the less important DWT encoded data is prioritized low than background traffic. This prevents from starvation of rest of the network

traffic. DiffServ is unable to divide one single stream of DWT encoded data into two based on packet content.

Table 6.2. Comparison of packet loss between MPLS, MPLS using proposed scheme and DiffServ at $\rho = 1$.

Network Type	Packet Size (B)	Packets Sent	Packets Received	% Packets Received
DiffServ	576	116	68	58 %
MPLS	576	116	0	0 %
MPLS with proposed scheme	576	116	36	31 %

The results of packet loss in case of MPLS with proposed scheme presented in Table 6.2 are compared to the ones obtained from simulation tests. This comparison shows that lesser packets were being dropped in simulation. This phenomenon can be explained by the fact that simulation based routing and queuing is faster compared to software routing and queuing. Also, hardware and software platform of software routers play an important role in this aspect.

Table 6.3. Comparison of packet loss of simulation and measurement based tests when transmitted over content-aware MPLS.

Test Type	Packet Size (B)	Packets Sent	Packets Received	% Packets Received
Simulation	576	116	59	50.8 %
Measurement	576	116	36	31 %

6.4.2 Comparison of Image Quality and Received File Size

The standard Barbara image in the pgm (portable grey map) format is used in the experiments. As mentioned in Section 6.3, the size of the file transferred

is 64KB. Figure 6.2 illustrates the difference in quality and file size at the receiver's end at $\rho = 0.5$. The reason to choose this rate was that at this point the regular MPLS backbone started to drop packets as can be seen from the figure below (image a). Image b is the file transferred over MPLS network using proposed scheme without any packet loss and hence a perfect image quality at this point is achieved. Same is the case with DiffServ (image c). At $\rho = 0.7$, nearly equal number of packets are dropped by content-aware scheme as by regular MPLS network. In this case, as depicted in Figure 6.3, only low priority packets are dropped by content-aware routers (less important parts of the image). Hence a blur but comparatively better image quality is maintained. It can also be seen that the sizes of the received files in case of MPLS and content-aware MPLS are almost same but the quality is much better in case of content-aware MPLS. On the other hand, DiffServ still maintains a 100% image quality at the expense of background traffic and a full 64KB file is received at the receiver's end.

a - 43KB Received via MPLS

b - 64KB Received via MPLS with proposed scheme

c - 64KB Received via DiffServ

Figure 6.2: Comparison of image quality at ρ = 0.5.

It is clear from the images in Figure 6.3 that under high congestion, regular MPLS routers drop packets randomly and hence important packets never reach the receiver (image a).

At higher values of ρ, where regular MPLS chokes out, the proposed scheme still continues the content-aware routing process and still maintains a better image quality (image b). Figure 6.4 shows the difference in quality of 20KB file received via content-aware MPLS (image b) and 40.2KB file received via regular MPLS (image a).

a - 40.2KB Received via MPLS

b - 44.6KB Received via MPLS with proposed scheme

c - 64KB Received via DiffServ

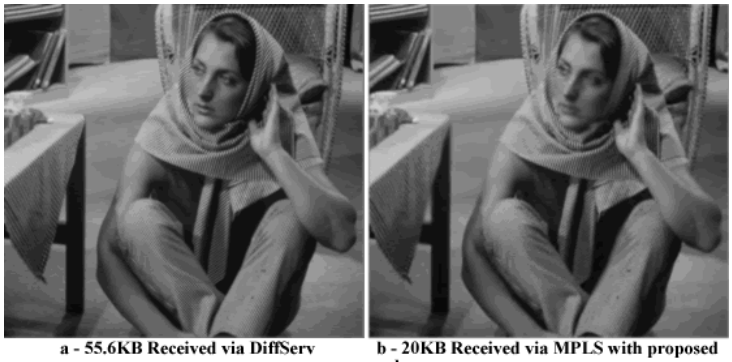
Figure 6.3: Comparison of image quality at $\rho = 0.7$.

a - 40.2KB Received via MPLS at $p = 0.7$

b - 20KB Received via MPLS with proposed scheme at $p = 0.9$

Figure 6.4: Comparison of image transferred at $\rho = 0.9$ over proposed scheme (image b) and an image transferred at $\rho = 0.7$ over regular MPLS (image a).

At ρ = 0.9, regular MPLS network saturates and packet loss reaches a maximum. The file size received through content-aware congestion control MPLS at this point is much smaller than that which was received through regular MPLS at ρ = 0.5. The reason for this behavior is that the proposed scheme makes the router capable of intelligently using available resources and route only those packets that are of extreme importance. Thus, although the file size received is much smaller, still the image quality is relatively better. At this point, in terms of image quality, DiffServ outperforms content-aware MPLS because in DiffServ, the Type of Service ToS/DSMark field is used for packet marking and QoS and DWT data is marked as Expedited Forwarding (EF) class [73]. However, it can be seen that the received file size in case of DiffServ is more than double of that received via proposed scheme. There are both pros and cons of this result. Even under high levels of congestion in the network, the quality of DWT content is not affected. Less important parts of the image could have been dropped allowing other data traffic to pass through. This results in starvation of other application data and inefficient use of available bandwidth. This is certainly avoided by content-aware MPLS.



a - 55.6KB Received via DiffServ

b - 20KB Received via MPLS with proposed scheme

Figure 6.5: Comparison of image transferred at ρ = 0.9 over DiffServ (image a) and proposed scheme (image b).

6.4.3 Comparison of Average end-to-end Delay and Jitter

The average end-to-end delay and jitter is compared at different values of ρ. Due to the fact that DiffServ is based on traditional IP routing with longest prefix match lookups, there is a certain amount of delay during file transfer. Moreover, packet marking, setting up of the EF class and configuring and maintaining per-hop-behavior for the EF class also degrades performance in terms of higher latency. This is one other point where the proposed scheme excels DiffServ. Fast MPLS forwarding results in lesser jitter and delay values. Figure 6.6 and Figure 6.7 illustrate the comparison.

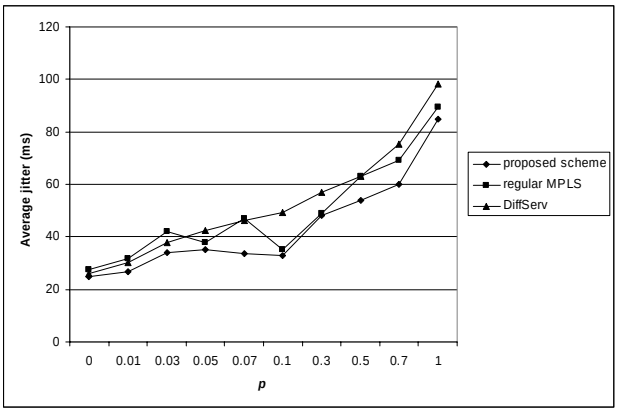


Figure 6.6: Comparison of average jitter.

The difference between the three schemes becomes obvious as the network gets congested. At higher values of ρ, content-aware congestion control enabled MPLS maintains comparatively low jitter and delay values. Comparison with DiffServ shows that in terms of quality, there is not much difference in performance from the proposed scheme.

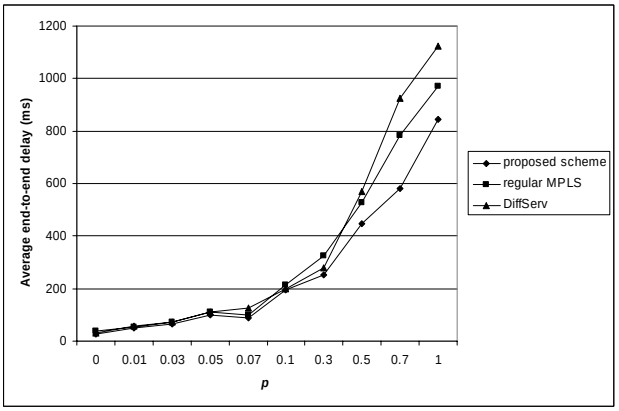


Figure 6.7: Comparison of average end-to-end delay

In fact DiffServ maintains comparatively better quality at higher congestion. But in context of efficient bandwidth utilization and maintaining fairness

among different traffic flows, content-aware congestion control technique in MPLS provides better and comparable results.

The results obtained from the simulation and measurement based experiments show similar trend. Average end-to-end delay, jitter of DWT encoded content and overall packet loss is maintained at a low level when transmitted over MPLS with proposed scheme. In both the cases, content-aware congestion control enabled MPLS restricts the metrics well within the acceptable region as explained in [65][66]. The difference in the results of simulation and measurement based experiments is in the numerical values of the data sets obtained since measurement based testing was conducted using slower hardware platform and software routing. Figure 6.8 clearly illustrates the similarity in trend but difference in the values of average end-to-end delay of DWT encoded data packets transferred over proposed scheme enabled MPLS backbone.

Figure 6.9 shows the difference in values of average jitter at different values of ρ in both simulation and measurement based tests. Here again the trend of the graph is similar, i.e., the values rise as ρ increases.

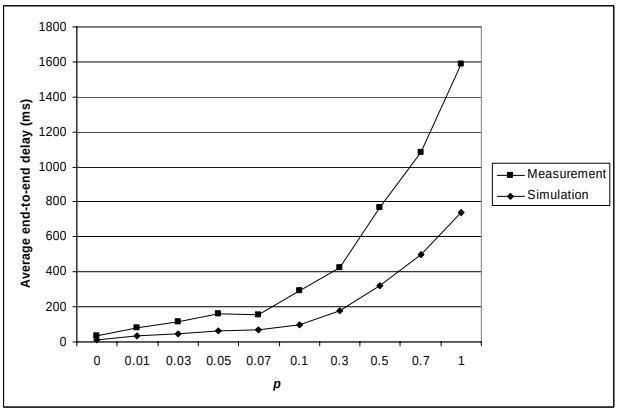


Figure 6.8: Comparison of delay values obtained from simulation and measurement studies while transmitting DWT traffic over content-aware MPLS.

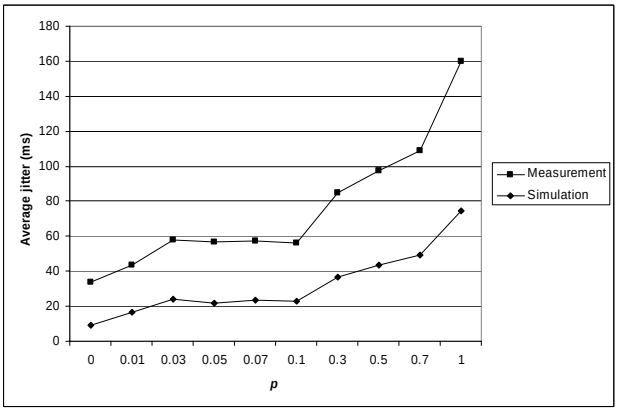


Figure 6.9: Comparison of jitter obtained from simulation and measurement studies while transmitting DWT traffic over content-aware MPLS.

It can be seen from the figures above that under measurement based testing, the observed metrics are much higher than that calculated under simulation based experiments. The main reason for this performance degradation of the system under study is lack of a dedicated processor for networking activities and an operating system kernel that is overwhelmed with interrupts termed as

noise process [68]. Other than these two main factors, network interface card's speed and system memory also play an important role in software router performance.

6.4.4 System level Measurements

This difference in the results obtained from the two modes of testing can be supported by the system level measurements that were taken in order to analyze metrics such as CPU utilization, interrupts per second and active virtual memory during a complete file transfer. Results obtained from the vmstat tests show that these metrics are highest in case of DiffServ compared to content-aware congestion control enabled MPLS and regular MPLS. This is because of IP address lookups from routing tables and PHB setups at the DiffServ nodes. Results show that regular MPLS performs slightly better than proposed scheme at the ingress and switch routers but is approximately equal at the egress node. This is because in MPLS with proposed scheme, the ingress and switch routers are performing extra processing as the crux of the algorithm is implemented in these routers.

Figure 6.10 illustrates active virtual memory of user processes during a file transfer at different values of ρ in all the three types of networks. It can be seen from the figure below that in case of DiffServ, active virtual memory of user processes in the system is highest. Refer to Appendix C for similar

results for active virtual memory of user level processes at LSR/core routers and egress nodes.

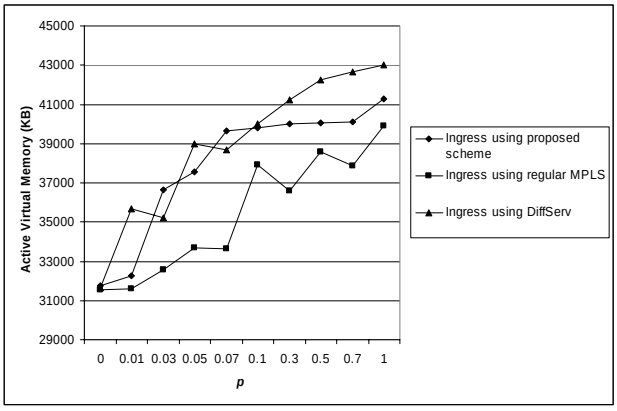


Figure 6.10: Comparison of active memory of user processes at different values of ρ at ingress router.

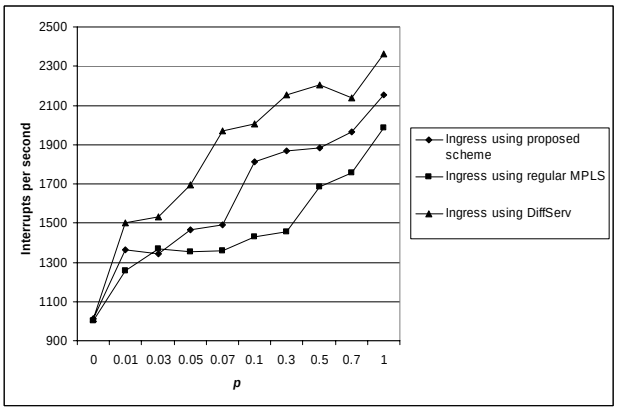


Figure 6.11: Comparison of interrupts per second at different values of ρ at ingress router.

Figure 6.11 shows that in case of regular MPLS, the total number of interrupts per second are least compared to content-aware congestion control

enabled MPLS and DiffServ. Refer to Appendix C for similar results for interrupts per second at LSR/core routers and egress nodes.

CPU utilization at ingress node during a single file transfer in all the three cases is depicted in Figure 6.12. Refer to Appendix C for similar results for CPU utilization at LSR/core routers and egress nodes. System level measurements at obtained at MPLS LSRs and DiffServ enabled core routers verify the fast forwarding attribute of label switching.

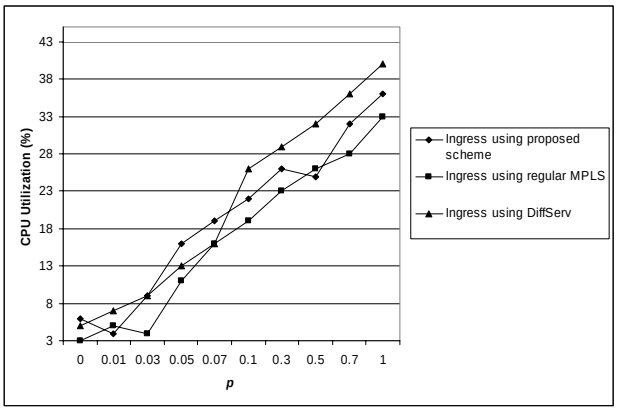


Figure 6.12: Comparison of CPU utilization at different values of ρ at ingress router.

6.5 Heterogeneous Network Backbone

The real world scenario presented in Section 4.4.7 is emulated on the test-bed to verify the performance of the proposed scheme in an hybrid backbone by designing and configuring a heterogeneous network backbone that includes MPLS along with IP routers as intermediate hops to the destination.

As shown in Figure 4.6. Results obtained from this scenario are compared with those while using and end-to-end pure content-aware MPLS backbone.

6.5.1 Comparison of Packet Loss

Table 6.4 compares the packet loss between a file that is transferred over a homogeneous content-aware congestion control enabled MPLS backbone and that over a heterogeneous backbone with intermediate IP hops. It is obvious from the results that in case of the heterogeneous backbone, due to intermediate IP cloud that does not support any QoS or the proposed scheme, packets are randomly dropped at times of congestion. This random dropping of packets also includes the high priority "marked" packets since the IP cloud does not recognize the marks on such packets.

Table 6.4. Comparison of packet loss at $\rho = 1$.

Backbone Type	Packet Size (B)	Packets Sent	Packets Received	% Packets Received
End-to-end MPLS	576	116	36	31 %
Heterogeneous	576	116	0	0 %

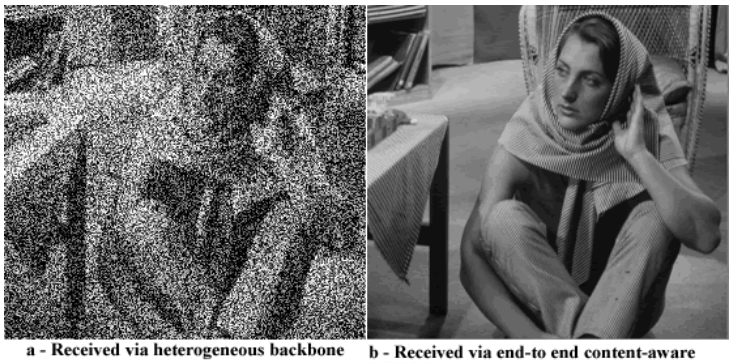
The results of packet loss in case of a heterogeneous network backbone presented in Table 6.4 are compared to the ones obtained from simulation tests. This comparison shows that lesser packets were being dropped in simulation as simulation does not account for the hardware platform performance bottlenecks. Table 6.5 explains the comparison.

Table 6.5. Comparison of packet loss of simulation and measurement based tests when transmitted over heterogeneous network backbone.

Test Type	Packet Size (B)	Packets Sent	Packets Received	% Packets Received
Simulation	576	116	35	30 %
Measurement	576	116	0	0 %

6.5.2 Comparison of Image Quality

Results show that the image quality is adversely affected since packets are randomly dropped once entered into the IP cloud. Figure 6.13 demonstrates the difference in the qualities in both the cases. Image a is received via heterogeneous backbone at ρ = 0.5 whereas repeated experimentation showed that at higher values of ρ nearly 75% of the time, a corrupt file was received with no representation of the image. On the other hand, image b is received via proposed scheme enabled MPLS backbone with 100% representation of the image.



a - Received via heterogeneous backbone

b - Received via end-to end content-aware MPLS backbone

Figure 6.13: Comparison of image quality at ρ = 0.5.

6.5.3 Comparison of Average end-to-end Delay and Jitter

The average end-to-end delay and jitter is compared at different values of ρ. Due to intermediate IP hops that do not support QoS and reprioritization and reclassification at the subsequent MPLS ingress router, there is a certain amount of delay during file transfer over the heterogeneous backbone. Figure 6.14 and 6.15 illustrate the comparison of delay and jitter respectively.

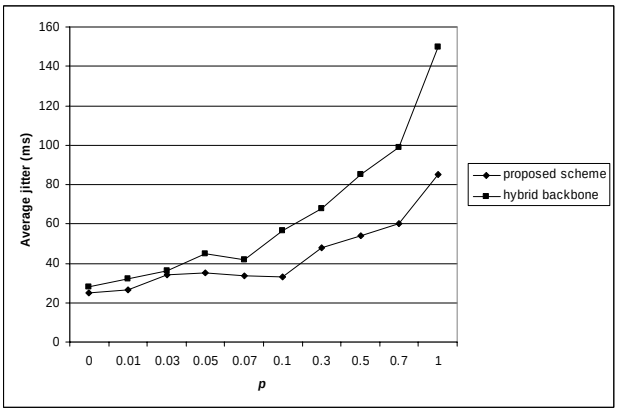


Figure 6.14: Comparison of average jitter.

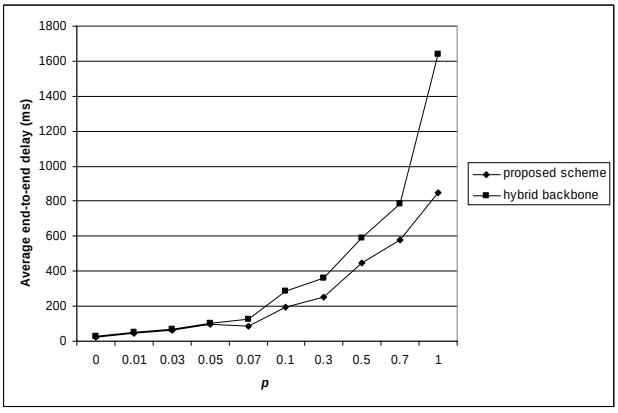


Figure 6.15: Comparison of average end-to-end delay

Results obtained from this scenario support the proposal stated in Section 1.7 that the proposed content-aware congestion control scheme over MPLS networks is well suited for EDNs administered by a single domain and managed by one management authority with and end-to-end pure MPLS network backbone. It does not show persuasive results in case of a network backbone constituting of multiple carrier technologies and protocols with intermediate hops that do not support content-awareness.

When measurement based results are compared with those obtained from simulations, again similar trend is seen in this case as well. Figure 6.16 and 6.17 illustrate the difference in delay and jitter respectively of multimedia traffic when simulated and measured.

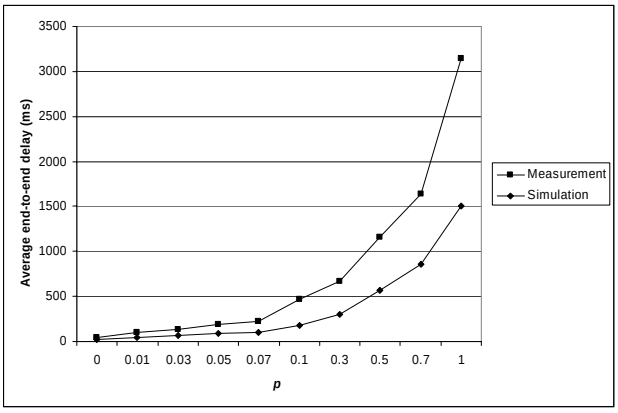


Figure 6.16: Comparison of delay values obtained from simulation and measurement studies while transmitting DWT traffic on heterogeneous network backbone.

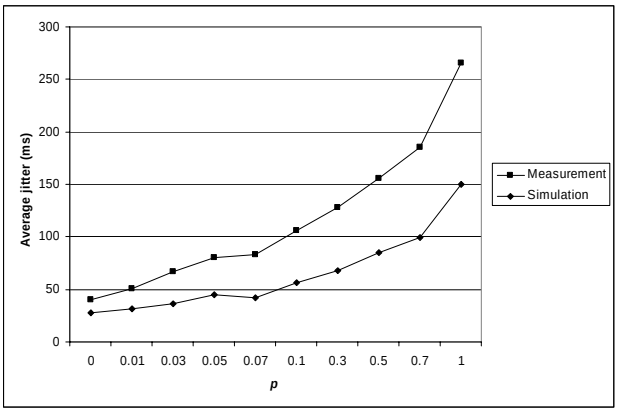


Figure 6.17: Comparison of jitter values obtained from simulation and measurement studies while transmitting DWT traffic on heterogeneous network backbone.

6.6 Summary

This chapter presents experimental validation of the proposed content-aware congestion control scheme. It describes the experimental designs, metrics to be measured, and the factors affecting these metrics. It provides in depth analysis on results obtained from different experimental scenarios and compares them with those obtained from simulation studies. Results show that proposed scheme works better than regular MPLS and DiffServ in terms of delay and jitter although DiffServ shows better packet loss ratio and hence best image quality. This in turn proves that proposed scheme utilizes available network bandwidth and also maintains fairness among different flows of traffic. Another experiment proves that the proposed scheme works well with an end-to-end MPLS backbone with no intermediate IP cloud.

These results confirm to the findings of the simulated based analysis. The difference only comes in the numeric values of the data sets obtained due to the fact that simulation does not take into account hardware bottlenecks and performance degradation.

Chapter 7

Conclusion

In this chapter, a summary of the work is presented and ways of improvement in future are suggested. The proposed content-aware congestion control scheme over MPLS networks can prove useful for content providers and content service providers alike. Multimedia content providers can use such non-standard compression techniques and service providers can implement the proposed scheme at their backbones for a better and QoS aware network infrastructure. The proposed scheme is simple to implement and eliminates the complexities and dependencies introduced by DiffServ for QoS. It is also flexible to allow service providers to add more service classes based on the needs and requirements of the customers. In short, the proposed scheme can provide vast opportunities to content providers in increasing their clientele. It enhances business for new carrier network providers as well as the industry gurus [45].

7.1 Limitations and Further Work

This section provides some of the directions where this work can be taken to and prove useful.

7.1.1 DWT compressed Video

The experimental verification of the proposed scheme was conducted by transferring only DWT encoded still images. A better case study and persuasive results can be provided by transferring DWT encoded video over content-aware congestion control enabled MPLS backbone. This will require an application that uses DWT compression to compress raw video, packetize it, prioritize packets according to the order described in the EZW algorithm and send the packets to the network. On the receiving end there must be a decoder and a video player. So far other compression techniques have been used for this purpose [48][49][50][51].

7.1.2 Support for Multicasting

The current setup works only with unicast transfer between client and server. A better approach would be to introduce multicasting techniques especially when incorporating video transmission [39].

7.1.3 Comparison with IPv6 and ATM

A good analysis would be to compare the proposed scheme with IPv6 [40] and QoS giant ATM.

7.2 Summary and Contributions of Thesis

A method to improve QoS of multimedia traffic over the Internet using MPLS is proposed. Design and implementation of a scheme that enables the MPLS router to discard packets containing less important content when there is congestion in the network is presented. By employing content-aware congestion control, the overall quality of multimedia content is not affected significantly even at times of congestion, enabling graceful degradation of quality using a wavelet based compression technique.

This technique is modeled and simulated in J-Sim network simulator as well as implemented as an extension to a public domain Linux-based MPLS router. Simulation and measurement based testing is used to evaluate the performance and QoS impact of this application-aware congestion control scheme.

The contributions of this thesis can be summarized as follows:

- Proposed a content-aware congestion control method in MPLS routers.
    - Enables the MPLS router to make routing decisions based on the content of the packet.
- Modeled and simulated the proposed scheme by modifying J-Sim.

  - Added content-aware routing module.
  - Added new packet header format.
  - Added delay counter module.

- Implemented a server application that sends DWT/EZW encoded images and performs application level prioritization of packets.
- Implemented a client application that receives DWT/EZW encoded images, decodes the image and displays it on the picture viewer.
- Implemented the proposed content-aware congestion control scheme in software based MPLS routers.
- Experimentally verified and tested the performance of the software based content-aware congestion control enabled MPLS routers.

Appendix A: Network Traces

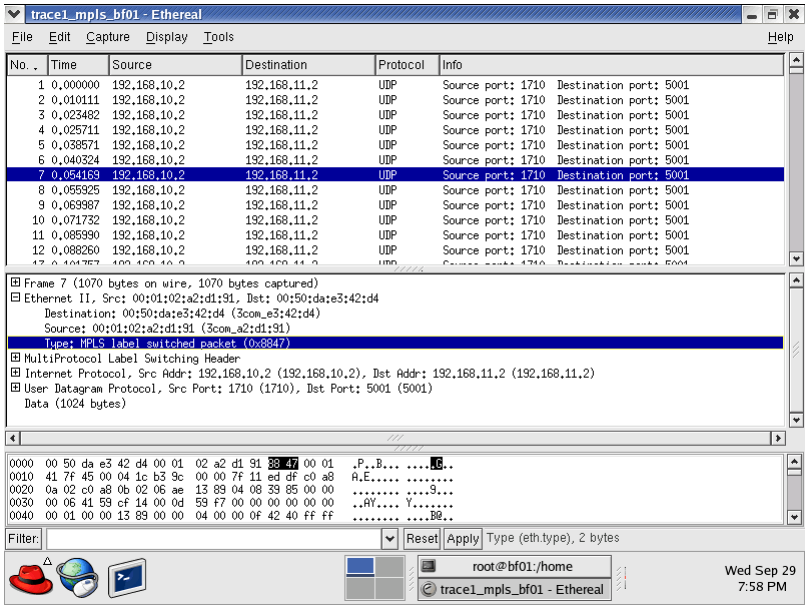


Figure A. 1: Network trace at the ingress router (packet type: MPLS).

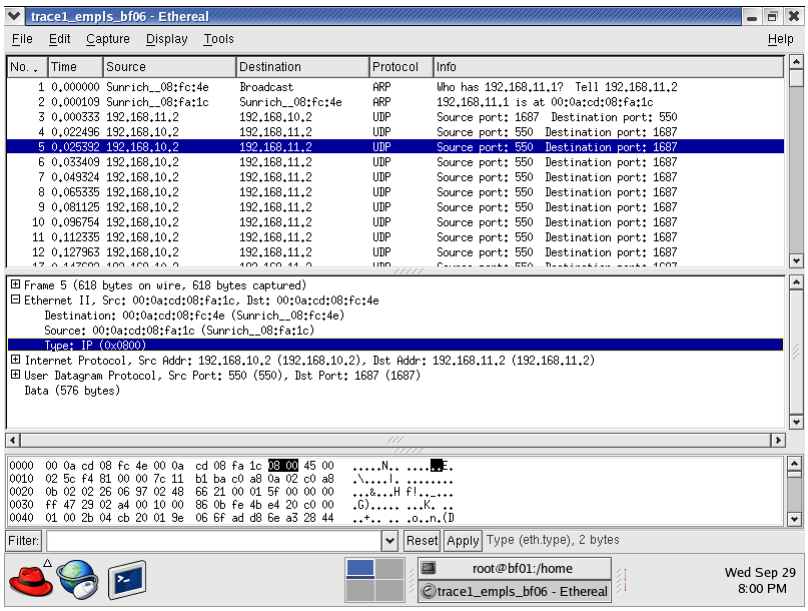


Figure A. 2: Network trace at the egress router (packet type: IP).

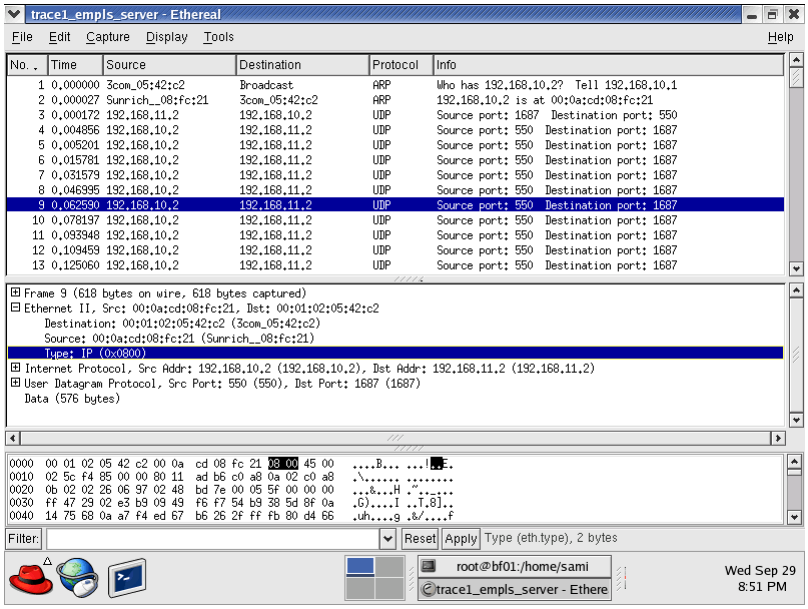


Figure A. 3: Network trace at server (packet type: IP).

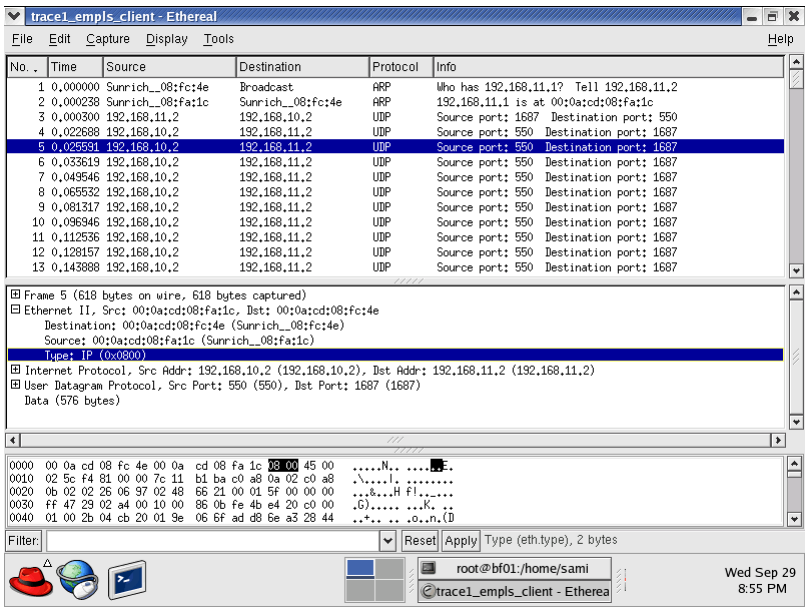


Figure A. 4: Network trace at client (packet type: IP)

Appendix B: MPLS Configuration Files

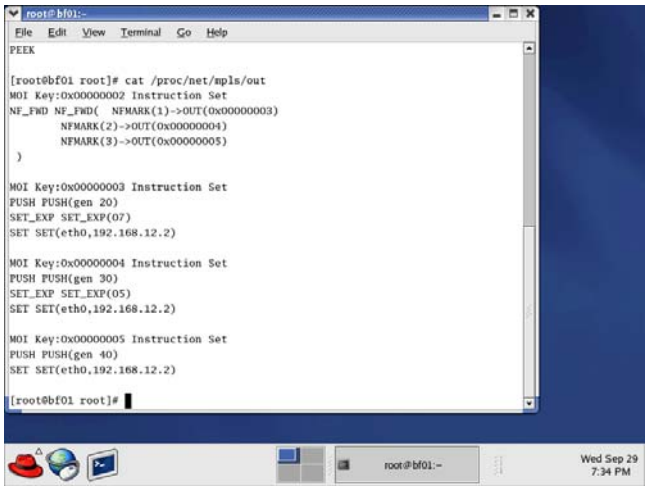


Figure B.1: MPLS configurations file at ingress. Values of EXP bits are set to 7 for high priority and 5 for low priority DWT packets.

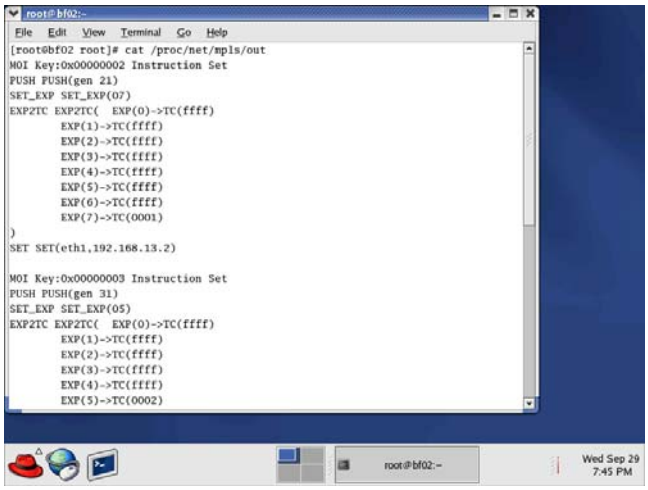


Figure B.2: MPLS configurations file at LSR. Values of EXP bits are set to 7 for high priority and 5 for low priority DWT packets.

Appendix C: System Level Measurements

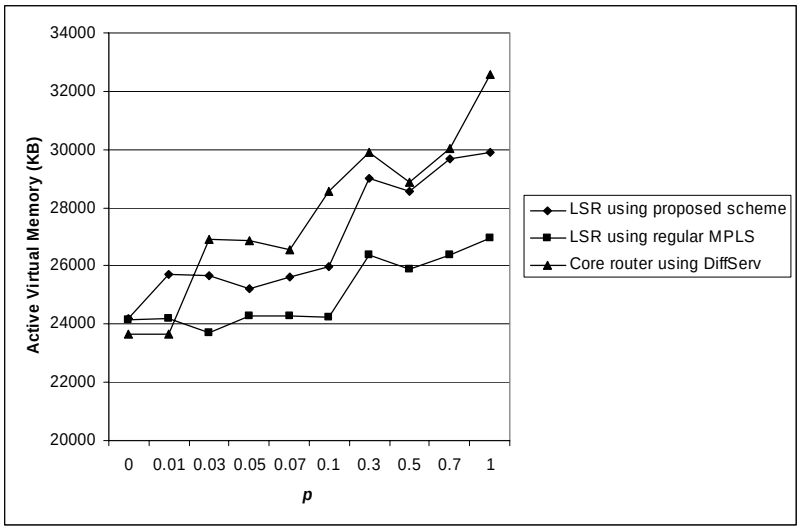


Figure C. 1: Comparison of active memory of user processes at different values of ρ at LSR/core router.

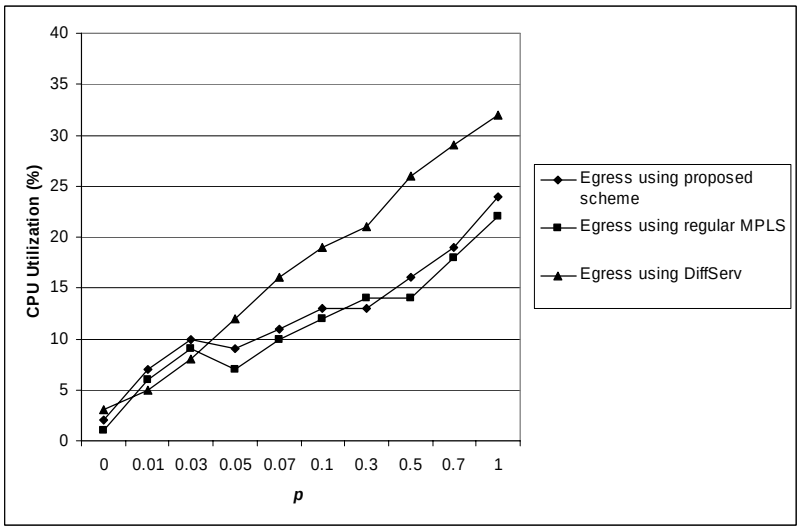


Figure C. 2: Comparison of active memory of user processes at different values of ρ at egress router.

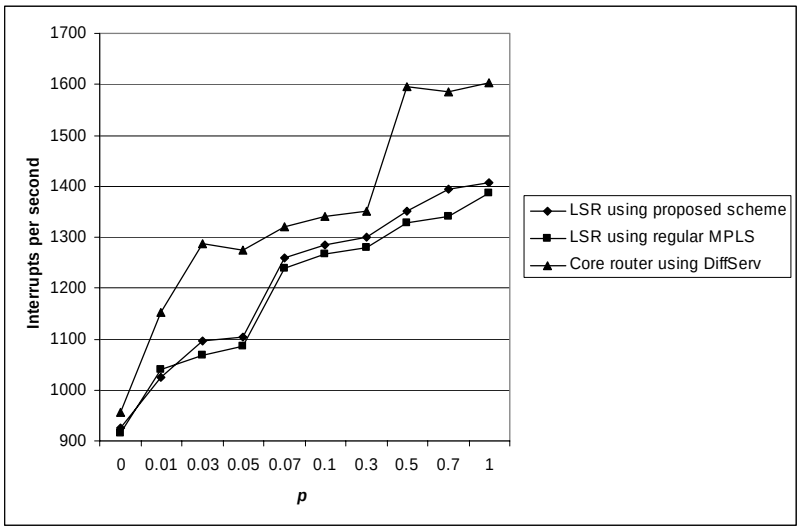


Figure C. 3: Comparison of interrupts per second at different values of ρ at LSR/core router.

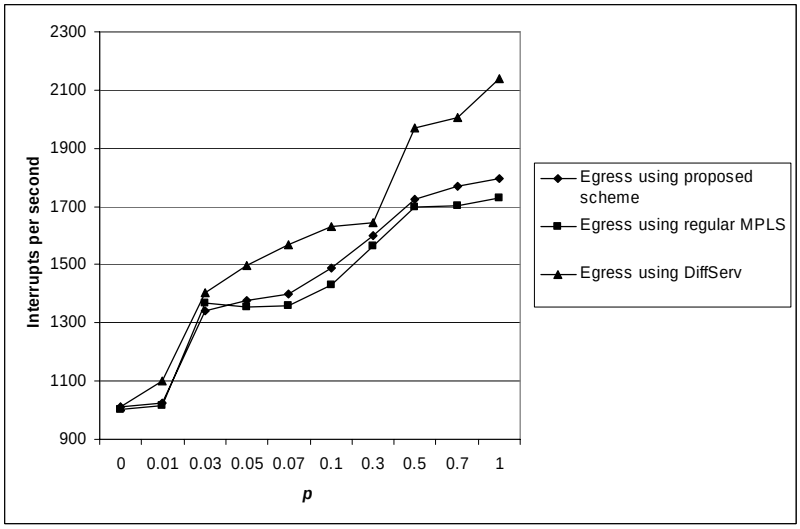


Figure C. 4: Comparison of interrupts per second at different values of ρ at egress router.

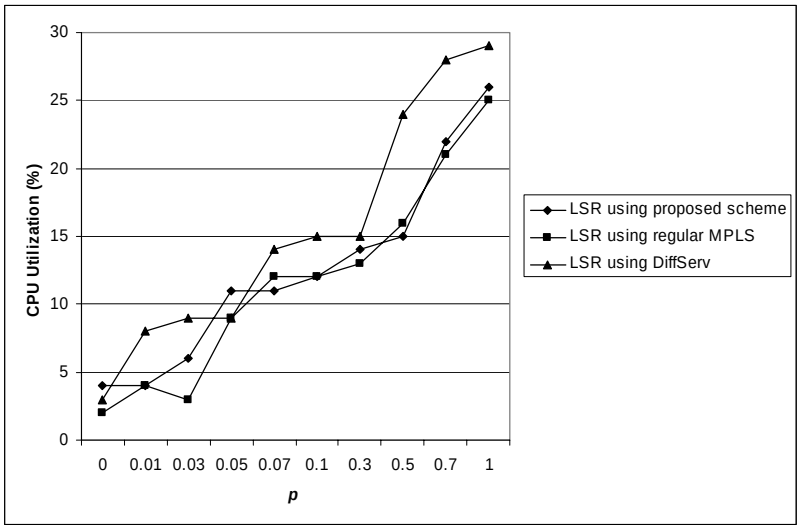


Figure C. 5: Comparison of CPU utilization at different values of ρ at LSR/core router.

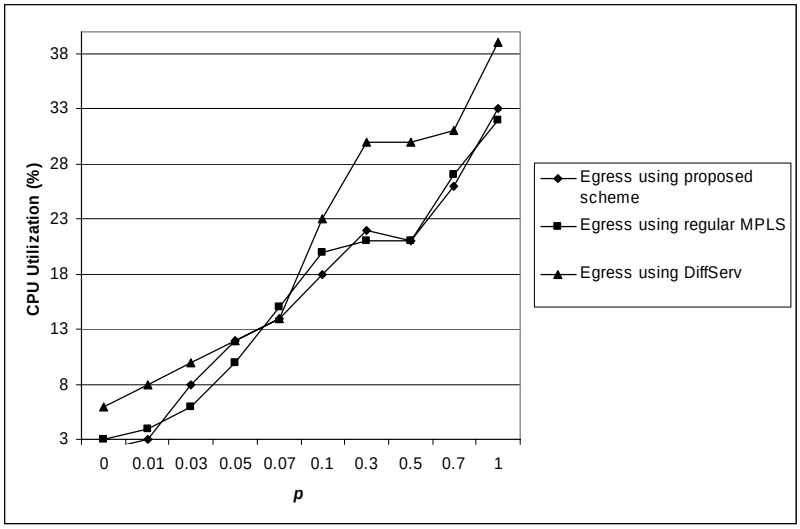


Figure C. 6: Comparison of CPU utilization at different values of ρ at egress router.

Bibliography

[1] Andrew B. Watson, "Image Compression using the Discrete Cosine Transform", Mathematica Journal, Volume: 4, Issue: 1, page(s): 81-88, 1994.J. Resnick and D. Halliday. Fundamentals of Physics. Chapter 5, John Wiley and Sons, New York, third edition, 1988.

[2] Barani Subbiah and Zartash A. Uzmi, "Content Aware Networking in the Internet: Issues and Challenges", IEEE International Conference on Communications, Volume: 4, Page(s): 1310 -1315, 11-14 June 2001.

[3] Jerome M. Shapiro, "Embedded Image Coding using Zerotrees of Wavelet Coefficients", IEEE Transaction on Signal Processing, Volume: 41, Issue: 12, December 1993.

[4] "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[5] Uyless Black, "MPLS and Label Switching Networks". Prentice Hall, 2001.

[6] Wei Sun et al., "Quality of Service using Traffic Engineering over MPLS: An Analysis", Proceedings of the 25th Annual IEEE Conference on Local Computer Networks, Page(s): 238 -241, 8-10 November, 2000.

[7] Reza Adhami, "Video Compression Technique using Wavelet Transform", Proceedings of the IEEE Aerospace Applications Conference, Volume: 4, Page(s): 449 -455, 3-10 February 1996.

[8] Pao Chi Chang and Ta-Te LU, "A Scalable Video Compression Technique based on Wavelet Transform and MPEG Coding", IEEE Transaction on Consumer Electronics, Volume: 45, Issue: 3, August 1999.

[9] Detlev Marpe and Hans L. Cycon, "Very Low Bit-Rate Video Coding using Wavelet-Based Techniques", IEEE Transaction on Circuitsand Systems for Video Technology, Volume: 9, Issue: 1, Page(s): 85 -94, February 1999.

[10] Zhiqan Zhang et al., "MPLS ATCC: An Active Traffic and Congestion Control Mechanism in MPLS", Proceedings of the International Conferences on Info-tech and Info-net, Volume: 5, Page(s): 205 -210, October – November, 2001.

[11] F. Holness and C. Philips, "Dynamic Congestion Control Mechanism for MPLS Networks", In SPIE's International Symposium on Voice,

Video and Data Communications, Internet Performance and Control Network Systems, Page(s): 1001-1005, November, 2000.

[12] Raj Jain, "Congestion Control in Computer Networks: Issues and Trends", IEEE Network Magazine, Page(s): 24-30, May 1990.

[13] Xipeng Xiao and Lionel M. Ni, "Internet QoS: The Big Picture", IEEE Network Magazine, pages 8-18, March/April 1999.

[14] Xipeng Xiao et al., "Traffic Engineering with MPLS in the Internet", IEEE Network Magazine, Volume: 14, Issue: 2, March-April 2000, Page(s): 28 -33.

[15] Mohammad M. Shahsavari and Adnan A. Al-Tunsi, "MPLS Performance Modeling using Traffic Engineering to improve QoS Routing on IP Networks", Proceedings of the IEEE Southeast Conference, Page(s): 152 -157, 5-7 April 2002.

[16] "An architecture for Differentiated Services", RFC 2475, December 1998.

[17] Tarek Saad et al., "Diffserv Enabled Adaptive Traffic Engineering over MPLS", Proceedings of the International Conferences on Info-tech and Info-net, Volume: 2, Page(s): 128 -133, October-November, 2001.

[18] "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[19] Geng-Sheng Kuo and C.T. Lai, "A new Architecture for Transmission of MPEG-4 Video over MPLS Networks", IEEE Communications Magazine, Volume: 40, Issue: 12, Page(s): 114 -119, December, 2002.

[20] "Requirements for Traffic Engineering over MPLS", RFC 2702, September 1999.

[21] "J-Sim Homepage", www.j-sim.org

[22] "Autonomous Component Architecture", www.j-sim.org/whitepapers/aca.html

[23] "Inet Tutorial", www.j-sim.org/drcl.inet/inet_tutorial.html

[24] "MPLS for J-Sim", www.info.ucl.ac.be/~bqu/jsim/mpls_desc.html

[25] "SKC Communications" www.skccom.com/1.888.734.4438/polyvideo/h323networking.pdf

[26] "netfilter/iptables project homepage", www.netfilter.org

[27] "Linux IP Firewalling Chains", www.people.netfilter.org/~rusty/ipchains/

[28] "netfilter/iptables patch-o-matic system", www.netfilter.org/patch-o-matic/index.html

[29] "iptables Tutorial 1.1.19", www.iptables-tutorial.frozentux.net/iptables-tutorial.html

[30] "Linux Traffic Shaping", www.knowplace.org/shaper

[31] "MPLS for Linux", www.sourceforge.net/projects/mpls-linux
[32] W. Almesberger, "Linux Network Traffic Control − Implementation Overview", February 2001.
[33] "NetX: Networking research without a better home", University of Cambridge, UK, www.cl.cam.ac.uk/research/srg/netos/netx/
[34] "NIST Switch - NIST MPLS research platform", National Institute of Standards and Technology, information technology laboratory, www.snad.ncsl.nist.gov/nistswitch/
[35] "iptables/libipq", www.cvs.netfilter.org/iptables/libipq
[36] "Iperf", http://dast.nlanr.net/Projects/Iperf/
[37] "RTP Parameters" www.iana.org/assignments/rtp-parameters
[38] "An Implementation of EZW", www.pesona.mmu.edu.my/~msng/EZW.html
[39] Hong Man and Yang Li, "Multi-Stream Video Transport over MPLS Networks", IEEE Workshop on Multimedia Signal Processing, Page(s): 384 -387, 2002.
[40] El-Bahlul Fgee et al., "Implementing QoS capabilities in IPv6 networks and comparison with MPLS and RSVP", IEEE Canadian Conference on Electrical and Computer Engineering, Volume: 2, Page(s): 851 -854, May, 2003.
[41] Tamrat Bayle et al., "Performance Measurement of MPLS Traffic Engineering and QoS", Technical Report, Hiroshima University, Japan, 2001.
[42] "MPLS Label Stack Encoding", RFC 3032, January, 2001.
[43] Petr Pavlu, "Basics of MPLS Technology and VPN Applications", Technical Presentation, Cisco Systems, 1999.
[44] Robert Pulley and Peter Christensen, "A Comparison of MPLS Traffic Engineering Initiatives", NetPlane Systems Inc., 2000.
[45] "IP Traffic Engineering for Carrier Networks: Using Constraint-Based Routing to Deliver New Services", White paper, Nortel Networks.
[46] Victoria Fineberg, "QoS Support in MPLS Networks", MPLS/Frame Relay Alliance White Paper, May, 2003.
[47] Wei Sun, "QoS/Policy/Constraint Based Routing", Survey paper, Computer and Information Science Department, Ohio State University, July, 2000.
[48] Gregory J. Conklin et al., "Video Coding for Streaming Media Delivery on the Internet", IEEE Transactions on Circuits and Systems for Video Technology, Volume: 11, Issue: 3, Page(s): 269 -281, March 2001.

[49] Borko Furht et al., "Multimedia Broadcasting over the Internet: Part II-Video Compression", IEEE Multimedia Magazine, Volume: 6, Issue: 1, Page(s): 85-89, January – March, 1999

[50] Bernd Girod et al., "Recent Advances in Video Compression", IEEE International Symposium on Circuits and Systems, Volume: 2, Page(s): 580 -583, 12-15 May 1996.

[51] Dapeng Wu et al., "MPEG-4 Compressed Video over the Internet", Proceedings of the IEEE International Symposium on Circuits and Systems, Volume: 4, Page(s): 327-331, May 30–June 2, 1999.

[52] H. Sharif and B. Chen, "End-to-End QoS requirements for real-time video streaming in Internet2", Technical Report, Department of Computer and Electronics Engineering, University of Nebraska-Lincoln, 2001.

[53] Manish Mahajan and Manish Parashar, "Managing QoS for Multimedia Applications in Differentiated Services Environment", Proceedings of the 4th IEEE Annual International Workshop on Active Middleware Services, 2002.

[54] Victoria Fineberg et al., "An End-to-End QoS Architecture with the MPLS-Based Core", IEEE Workshop on IP Operations and Management, Page(s): 26 -30, 2002.

[55] "Simulation-Based Analysis of MPLS Traffic Engineering", Model Research and Development, OPNET Technologies Inc.

[56] Dapeng Wu, "Streaming Video over the Internet: Approaches and Directions", IEEE Transaction on Circuits and Systems for Video Technology, Volume: 11, Issue 3, March 2001.

[57] Sean Harnedy, "An introduction to MPLS", Prentice Hall PTR, November 2001

[58] Robert Prandolini, "Use of UDP for Efficient Imagery Dissemination", DSTO C3 Research Centre, IT Division, Defense Science and Technology Organization, Department of Defense, Australia, 2001.

[59] G. Rosenbaum, S. Jha and M. Hassan, "Empirical study of traffic trunking in Linux-based MPLS test-bed", International Journal of Network Management, Page(s): 277-288, 2003.

[60] Lin Y-D, Hsu N-B, Hwang RH, "Granularity of QoS routing in MPLS networks", International Workshop on Quality of Service (IWQoS). Page(s): 140–154, June 2001.

[61] Viswanathan A, Feldman N, Wang Z, Callon R, "Evolution of Multiprotocol Label Switching", IEEE Communications Magazine, Page(s): 165–173, May 1998.

[62] Muhammad Rehan Sami and Abdul Waheed, "Content-Aware Congestion Control in MPLS Networks for Video and Compressed Images", to appear in the 3rd ACS/IEEE International Conference on Computer Systems and Applications, Cairo, Egypt, January 3-6, 2005.

[63] Sanda Dragos and Radu Dragos, "Bandwidth Management in MPLS Networks", School of Electronic Engineering – DCU, Broadband Switching and Systems Laboratory, November 2001.

[64] P. Van Heuven, S. Van den Berghe, J. Coppens, P. Demeester, "RSVP-TE daemon for DiffServ over MPLS under Linux", www.dsmpls.atlantis.rug.ac.be

[65] James F. Kurose and Keith W. Ross, "Computer Networking – A Top-Down Approach Featuring the Internet", Pearson Education, 2003.

[66] Cox et al., "On the Applications of Multimedia Processing to Communications", Proceedings of the IEEE: 86(5), May, 1998.

[67] Raj Jain, "The Art of Computer Systems Performance Analysis", Wiley, April, 1991.

[68] Oscar-Ivan Lepe-Aldama and Jorge Garcia-Vidal, "A Performance Model of a PC based IP Software Router", IEEE International Conference on Communications, Volume: 2, Page(s):1230 – 1235, 28 April-2 May, 2002.

[69] Dimitri Bertsekas and Robert Gallager, "Data Networks", Second Edition, Prentice Hall, 1992.

[70] Evi Nemeth, Garth Snyder and Trent R. Hein, "Linux Administration Handbook", Prentice Hall, 2002.

[71] Alberto L. Garcia and Indra Widjaja, "Communication Networks – Fundamental Concepts and Key Architectures", McGraw Hill, 2004.

[72] "Differentiated Services on Linux", www.diffserv.sourceforge.net

[73] Hoon Lee et al., "Guaranteeing Multiple QoSs in Differentiated Services Internet" Seventh International Conference on Parallel and Distributed Systems: Workshops, Page(s):233 – 238, 4-7 July, 2000.

[74] "Characterization of Multimedia Streams of an H.323 Terminal", www.developer.intel.com/technology/itj/q21998/articles/art_5.htm

Vita

Muhammad Rehan Sami is a Graduate student of COE Department pursuing Masters Degree in Computer Networking. He did his Bachelors of Science in Computer Engineering from Sir Syed University of Engineering and Technology (SSUET), Karachi, Pakistan in 2002. He has served as an internee at the Cisco Network Lab of SSUET and holds Cisco Certified Network Associate (CCNA) title. As Research Assistant in KFUPM, he worked with the CCSE Network group on network upgrade and trouble shooting projects and in Research Institute on network research/consultancy projects. His major areas of research are Network Design, Network Management and QoS issues in the Internet.