11-21-2002

Simulation and analysis of network traffic for efficient and reliable information transfer

Neelima Boppana
Florida International University

DOI: 10.25148/etd.FI14051194

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

SIMULATION AND ANALYSIS OF NETWORK TRAFFIC FOR EFFICIENT AND RELIABLE INFORMATION TRANSFER

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Neelima D. Boppana

2002

TO: Dean Vish Prasad
College of Engineering

This thesis, written by Neelima D. Boppana, and entitled Simulation and Analysis of Network Traffic for Efficient and Reliable Information Transfer, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Malcolm Heimer

Tadeusz M. Babij

Subbarao Wunnava, Major Professor

Date of defense: November 21, 2002

The thesis of Neelima D. Boppana is approved.

Dean Vish Prasad
College of Engineering

Dean Douglas Wartzok
University Graduate School

Florida International University, 2002

DEDICATION

I dedicate this master's thesis to my parents. Without their constant support, encouragement and above all their love and care, this work would not have been possible.

ACKNOWLEDGMENTS

I wish to thank my committee members Dr. Malcolm Heimer and Dr. Tadeusz M. Babij for their support, patience and encouragement. I wish to express my sincere gratitude to my major professor, Dr. Subbarao Wunnava, who guided me from the beginning and encouraged me to do better all the time. I would like to acknowledge the support I got from Computer Engineering Department especially Pat Brammer, Carolyn and Carmen. Their support and words of encouragement have always been a source of inspiration. I would like to thank all HSTNL lab members for their help and last but not the least I would like to thank my friend, labmate and roommate Bhargavi Kodiparthi for all the moral support she provided throughout my masters degree.

ABSTRACT OF THE THESIS

SIMULATION AND ANALYSIS OF NETWORK TRAFFIC FOR EFFICIENT AND RELIABLE INFORMATION TRANSFER

by

Neelima D. Boppana

Florida International University, 2002

Miami, Florida

Professor Subbarao Wunnava, Major Professor

With the growing commercial importance of the Internet and the development of new real-time, connection-oriented services like IP-telephony and electronic commerce resilience is becoming a key issue in the design of IP-based networks. Two emerging technologies, which can accomplish the task of efficient information transfer, are Multiprotocol Label Switching (MPLS) and Differentiated Services. A main benefit of MPLS is the ability to introduce traffic-engineering concepts due to its connection-oriented characteristic. With MPLS it is possible to assign different paths for packets through the network. Differentiated services divides traffic into different classes and treat them differently, especially when there is a shortage of network resources. In this thesis, a framework was proposed to integrate the above two technologies and its performance in providing load balancing and improving QoS was evaluated. Simulation and analysis of this framework demonstrated that the combination of MPLS and Differentiated services is a powerful tool for QoS provisioning in IP networks.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

1.0 INTRODUCTION

Being one of the largest man-made communication systems in the world, the Internet with its control algorithms is an interesting and challenging area of research for people. The number of users is increasing and there has been a shift in the traffic characteristics: more traffic and shorter transmissions. The most widely used control algorithm is the one in the transfer control protocol, TCP. Its purpose is to control the flow and prevent congestion. The fast growth of the Internet has changed the conditions that TCP was made for. As over provisioning is not a realistic solution, the challenge for researchers is to come up with significant improvements that require as few changes as possible.

When there is a problem of congestion due to increased usage of networks researchers came up with a process called traffic engineering as a solution. Traffic engineering (TE) can be leveraged to significantly enhance the operation and performance of their networks. It matches the network resources with the prevailing demands and helps to utilize the resources in an efficient manner. But until recently very few traffic engineering capabilities existed in IP networks. Chapter 2 discusses the importance of TE and current traffic engineering techniques with their limitations.

All the traffic-engineering techniques address the capacity issue and are designed to provide the required bandwidth. But, in present day networks, providing bandwidth is not the only answer as capacity is not the only issue. Development of different streaming technologies and applications like IP telephony, multimedia requires a level of service assurance besides the bandwidth. Chapters 3 and 4 discuss the issue of Quality of service (QoS) and different QoS techniques available in today's market.

Recently a technology called Multiprotocol label switching (MPLS) is introduced as a result of the efforts to overlap the advantages of different architectures like IP and ATM (Asynchronous Transfer Mode). Chapter 5 discusses this latest technology and its application in the area of traffic engineering.

Based on the research done on different available techniques, a framework work, which can accomplish the task of efficient and reliable information transfer, is proposed in chapter 6. Simulation based analysis of proposed framework is done later in chapter 7 which is followed by conclusions, and future work. In summary, this thesis work not only discussed the issues related to traffic engineering, but also presented a systematic approach for providing QoS in IP networks.

2.0 TRAFFIC ENGINEERING

2.1 Introduction

Congestion in control protocols has been a problem in the networks since the past. Congestion in informal terms is "too many sources sending too much data too fast for network to handle". Internet Protocol (IP) a pure datagram protocol, and Transmission Control Protocol (TCP), transport layer protocol, when used together, are subject to unusual congestion problems caused by interactions between the transport and datagram layers. Improved handling of congestion is now mandatory for successful network operation under load. It can arise in the networks for many different reasons.

2.2 Reasons for Traffic Congestion

Congestion is caused in many situations. Below mentioned are some reasons for the traffic congestion.

1. Flood of traffic destined for the same output line (queue fills up and packets drop). Memory does not necessarily solve this type of congestion problem.
2. Slow processors or inefficient routing software
3. Mismatch between parts of the system (several fast lines and one slow)

At present, streaming media applications are being increasingly deployed on the networks. A recent study reports that about 50 million minutes of long distance traffic now travel over Internet Protocol networks every day [1]. When multimedia applications are used, they require the transmission of real-time streams over a network. These streams often exhibit variable bandwidth requirements, and require high bandwidth and guarantees from the network. In such situation congestion is caused due to insufficient

bandwidth and the inability to prioritize traffic. Congestion may be caused due to many other reasons like lack of backup when there is a link failure and so on.

2.3 Effects of Traffic Congestion

While some services, such email, don't usually suffer from network delays, web services as well as new and sophisticated applications that send audio or video files over IP networks can suffer greatly when network packets get delayed. If a router floods control packets to its neighbors and has no congestion avoidance and control mechanisms, it can overwhelm its neighbors and cause congestion. To prevent this, mechanisms are needed to avoid going into congestion and to recover when congestion occurs. Congestion avoidance prevents the network domain from entering a state of congestive collapse with control traffic. Congestive collapse is a situation where, although the network links are being heavily utilized with an overload of control messages, very little useful work is being done because of the routers are overwhelmed with non-useful work. To successfully manage today's service as well as the next generation of voice and video IP services, network managers need advanced tools to help regulate network traffic and provide a sound basis for decisions from trouble-shooting near-term problems to longer-term improvement in capacity and network design. [2]

2.4 Present Situation

The central challenge facing the network managers is keeping customers happy and sustaining high rates of traffic growth. Meeting these challenges requires a number of circuits of various bandwidths over a geographic area In other words, a physical topology that meets the needs of the customers connected to its network must be deployed. After the network is deployed, customer traffic flows onto the physical topology must be

mapped. In the early 1990s, mapping traffic flows onto a physical topology was not approached in a particularly scientific way. Instead, the mapping occurred as a by-product of routing configuration-traffic flows simply followed the shortest path calculated by the Interior Gateway Protocol (IGP). The limitations of this haphazard mapping were often resolved by over provisioning bandwidth as individual links began to experience congestion. Today as networks grow larger, as the circuits supporting IP grow faster, and as the demands of customers become greater, the mapping of traffic flows onto physical topologies needs to be approached in a fundamentally different way so that the offered load can be supported in a controlled and efficient manner.

2.5 Network Traffic Engineering

The task of mapping traffic flows onto an existing physical topology is called traffic engineering. Traffic engineering tries to match network resources with prevailing demands. Traffic engineering is necessary because network demands are not predictable. Extrinsic events like a web cast of a fashion show can create huge demands. Additionally, intrinsic factors such as equipment failures require traffic engineering to make better use of existing network resources to handle fluctuating demands. If a traffic engineering application implements the right set of features, it should provide precise control over the placement of traffic flows within its routed domain. Specifically, traffic engineering provides the ability to move traffic flows away from the shortest path selected by the IGP and onto a potentially less congested physical path across the network. Figure 1 shows a typical traffic-engineering path.

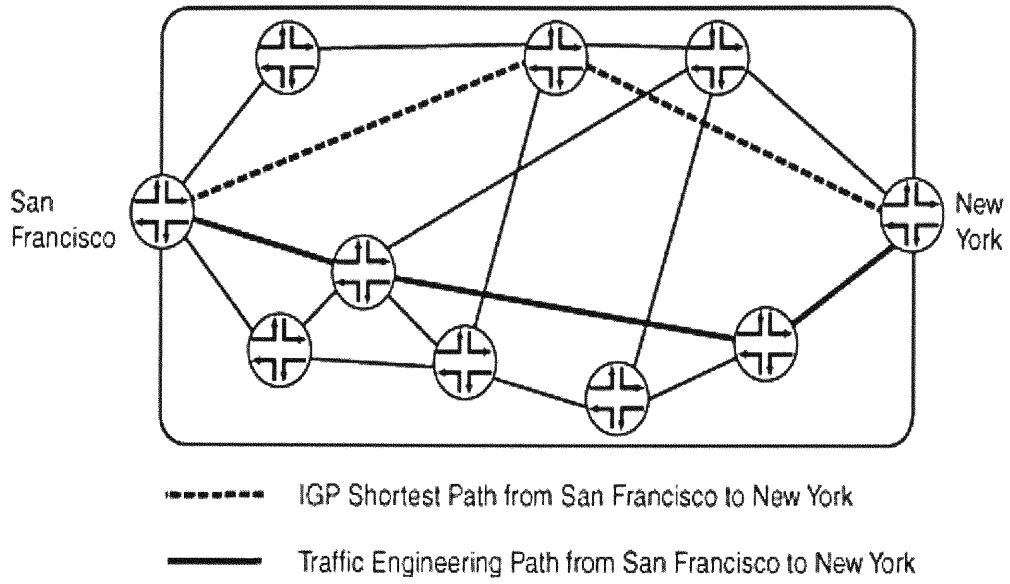

Figure 1: Traffic Engineering Path vs. IGP Shortest Path across a Network [3]

Traffic engineering is a powerful tool that can be used to balance the traffic load on the various links, routers, and switches in the network so that none of these components is over utilized or underutilized. In this way, the economies of the bandwidth that has been provisioned across the entire network can be exploited. Traffic engineering should be viewed as assistance to the routing infrastructure that provides additional information used in forwarding traffic along alternate paths across the network. [3]

2.6 Applications for Traffic Engineering

Traffic engineering has become an important issue because of the unprecedented growth in customer demand for network resources, the critical nature of IP applications, and the increasingly competitive nature of the Internet marketplace. Existing IGPs (Internet gateway protocols) can actually contribute to network congestion because they do not take bandwidth availability and traffic characteristics into account when building their forwarding tables. Traffic engineering can be leveraged to significantly enhance the operation and performance of their networks. [3]

Traffic engineering capabilities can be used to:

- Route primary paths around known bottlenecks or points of congestion in the network.
- Provide precise control over how traffic is rerouted when the primary path is faced with single or multiple failures.
- Provide more efficient use of available aggregate bandwidth by ensuring that subsets of the network do not become over utilized while other subsets of the network along potential alternate paths do not become underutilized.
- Make an ISP (Internet Service Provider) more competitive within its market by maximizing operational efficiency, resulting in lower operational costs.
- Enhance the traffic-oriented performance characteristics of the network by minimizing packet loss, minimizing prolonged periods of congestion, and maximizing throughput.
- Enhance statistically bounded performance characteristics of the network (such as loss ratio, delay variation, and transfer delay) that will be required to support the forthcoming multi services Internet.
- Provide more options, lower costs, and better service to their customers.

2.7 Evolution of Traffic Engineering in IP Networks

Until recently very limited traffic engineering capabilities existed in IP networks to provide differentiated queue management and scheduling services to packets belonging to different classes. In terms of routing control, the IP networks have employed distributed protocols for intra-domain routing. These protocols are highly scalable and

resilient. But they are based on simple algorithms for path selection, which have very limited functionality to allow flexible control of the path selection process.

2.7.1 Traditional IP Routing

When a host (which is not a router) wishes to send an IP packet to a destination on another network or subnet, it needs to choose an appropriate router to send the packet to. According to the IP Architecture, it does so by maintaining a route cache and a list of default routers. Each entry in the route cache lists a destination (IP address) and the appropriate router to use to reach that destination. The host learns the information stored in its route cache through the ICMP Redirect mechanism. The host learns the list of default routers either from static configuration information or by using the ICMP Router Discovery mechanism [4]. When the host wishes to send an IP packet, it searches its route cache for a route matching the destination address in the packet. If one is found it is used; if not, the packet is sent to one of the default routers. [5]

2.7.2 Adaptive Routing in the ARPANET

The early ARPANET recognized the importance of adaptive routing where routing decisions were based on the current state of the network. Early minimum delay routing approaches forwarded each packet to its destination along a path for which the total estimated transit time was the smallest. Each node maintained a table of network delays, representing the estimated delay that a packet would experience along a given path toward its destination. The minimum delay table was periodically transmitted by a node to its neighbors. The shortest path, in terms of hop count, was also propagated to give the connectivity information. One drawback to this approach is that dynamic link metrics tend to create "traffic magnets" causing congestion to be shifted from one

location of a network to another location, resulting in oscillation and network instability.[6]

2.7.3 Early congestion control techniques in IP/TCP networks

In heavily loaded pure datagram networks with end-to-end retransmission, as switching nodes become congested, the round trip time through the net increases and the count of datagrams in transit within the net also increases. This is normal behavior under load. As long as there is only one copy of each datagram in transit, congestion is under control. Once retransmission of datagrams not yet delivered begins, there is potential for serious trouble. [7]

Host TCP implementations are expected to retransmit packets several times at increasing time intervals until some upper limit on the retransmit interval is reached. Normally, this mechanism is enough to prevent serious congestion problems. Even with the better adaptive host retransmission algorithms, though, a sudden load on the net can cause the round-trip time to rise faster than the sending hosts measurements of round-trip time can be updated. Such a load occurs when a new bulk transfer, such a file transfer, begins and starts filling a large window. If the round-trip time exceeds the maximum retransmission interval for any host, that host will begin to introduce more and more copies of the same datagrams into the net. The network is then in serious trouble. Eventually all available buffers in the switching nodes will be full and packets must be dropped. The round-trip time for packets that are delivered is now at its maximum. Hosts are sending each packet several times, and eventually some copy of each packet arrives at its destination. This condition is stable. Once the saturation point has been reached, if the algorithm for selecting packets to be dropped is fair, the network will continue to operate

in a degraded condition. In this condition every packet is being transmitted several times and throughput is reduced to a small fraction of normal. It is possible for round trip time to become so large that connections are broken because the hosts involved time out. Adding additional memory to the gateways will not solve the problem. The more memory added, the longer round-trip times must become before packets are dropped. Thus, the onset of congestion collapse will be delayed but when collapse occurs an even larger fraction of the packets in the net will be duplicates and throughput will be even worse.[7]

2.7.4 Dynamic Routing in the IP networks

The IP networks evolved from the ARPANET and adopted dynamic routing algorithms with distributed control to determine the paths that packets should take en-route to their destinations. The routing algorithms are adaptations of shortest path algorithms where costs are based on link metrics. The link metric can be based on static or dynamic quantities. The link metric based on static quantities may be assigned administratively according to local criteria. The link metric based on dynamic quantities may be a function of a network congestion measure such as delay or packet loss.

It was apparent early that static link metric assignment was inadequate because it can easily lead to unfavorable scenarios in which some links become congested while others remain lightly loaded. One of the many reasons for the inadequacy of static link metrics is that link metric assignment was often done without considering the traffic matrix in the network. Also, the routing protocols did not take traffic attributes and capacity constraints into account when making routing decisions. This results in traffic concentration being localized in subsets of the network infrastructure and potentially

causing congestion. Even if link metrics are assigned in accordance with the traffic matrix, unbalanced loads in the network can still occur due to a number factors including:

- Resources may not be deployed in the most optimal locations from a routing perspective.
- Forecasting errors in traffic volume and/or traffic distribution.
- Dynamics in traffic matrix due to the temporal nature of traffic patterns, BGP policy change from peers, etc.

The inadequacy of the legacy Internet interior gateway routing system is one of the factors motivating the interest in path-oriented technology with explicit routing and constraint-based routing capability such as Multiprotocol Label Switching (MPLS).

2.7.5 ToS Routing

Type-of-Service (ToS) routing involves different routes going to the same destination with election dependent upon the ToS field of an IP packet [8]. The ToS classes may be classified as low delay and high throughput. Each link is associated with multiple link costs and each link cost is used to compute routes for a particular ToS. A separate shortest path tree is computed for each ToS. The shortest path algorithm must be run for each ToS resulting in very expensive computation. [8]

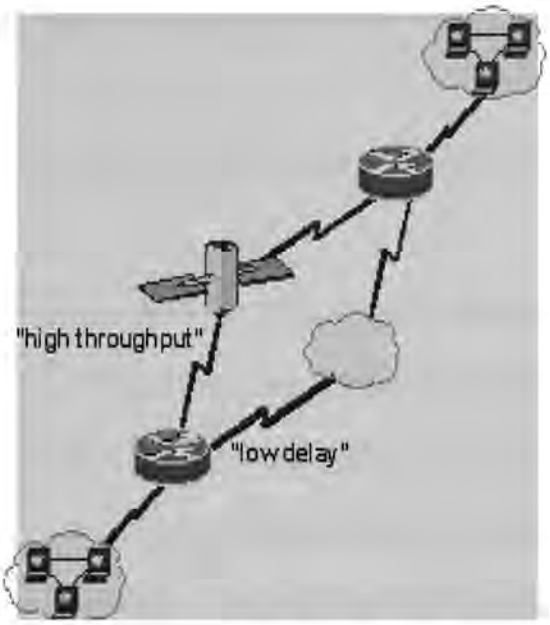

Figure 1: ToS Routing [8]

Effective traffic engineering is difficult to perform in classical ToS-based routing because each class still relies exclusively on shortest path routing which results in localization of traffic concentration within the network. So, Classical ToS-based routing is now made obsolete by replacing the IP header field by Diffserv field. [8]

2.7.6 Equal-Cost Multipath

Equal Cost Multi-Path (ECMP) is another technique that attempts to address the inefficiency in the Shortest Path First (SPF) interior gateway routing systems. In the classical SPF algorithm, if two or more shortest paths exist to a given destination, the algorithm will choose one of them. The algorithm is modified slightly in ECMP so that if two or more equal cost shortest paths exist between two nodes, the traffic between the nodes is distributed among the multiple equal-cost paths.

Traffic distribution across the equal-cost paths is usually performed in one of two ways.

1. Packet-based in a round-robin fashion: It can easily cause out- of-order packets

2 Hash-Threshold: This approach is flow-based, using hashing on source and destination IP addresses and possibly other fields of the IP header The router first selects a key by performing a hash over the packet header fields that identify a flow. The N next-hops have been assigned unique regions in the key space. The router uses the key to determine which region and thus which next-hop to use. Flow-based load sharing may be unpredictable in an enterprise network where the number of flows is relatively small and less heterogeneous (for example, hashing may not be uniform), but it is generally effective in core public networks where the number of flows is large and heterogeneous.

In ECMP, link costs are static and bandwidth constraints are not considered, so ECMP attempts to distribute the traffic as equally as possible among the equal-cost paths independent of the congestion status of each path. As a result, given two equal-cost paths, it is possible that one of the paths will be more congested than the other. Another drawback of ECMP is that load sharing cannot be achieved on multiple paths, which have non-identical costs. [9]

3.0 QUALITY OF SERVICE

3.1 Introduction

Quality of Service (QoS) means providing consistent, predictable data delivery service. In other words, satisfying customer application requirements. QoS is the ability of a network element (e.g. an application, host or router) to have some level of assurance that its traffic and service requirements can be satisfied. To enable QoS, it requires the cooperation of all network layers from top-to-bottom, as well as every network element from end-to-end. Quality of service, with respect to computer networks is a set of service requirements to be met by the network while transporting some network traffic flow. Here, the flow represents a packet stream from source to destination that can be unicast or multi-cast, with associated quality of service requirements Network. QoS is a measurable level of service delivered to network users, characterized by quality of service parameters. QoS does not create bandwidth. It isn't possible for the network to give what it doesn't have, so bandwidth availability is a starting point. QoS only manages bandwidth according to application demands and network management settings, and in that regard it cannot provide certainty if it involves sharing. Hence, QoS with a guaranteed service level requires resource allocation to individual data streams. A priority for QoS designers has been to ensure that best-effort traffic is not starved after reservations are made. QoS-enabled (high-priority) applications must not disable the low-priority Internet applications. [10]

3.2 Need for QoS

Network traffic has increased as the number of users and applications has increased. Over time, the needs and uses of the networks have changed. Rather than the existing "best

effort" paradigm, it is necessary for the Internet to support service-level agreements that guarantee a specified level of good throughput and network reliability, irrespective of the usage level and individual network failures.

For corporate customers, there is a range of applications that operate across the network. Many of these applications do not have any strict service-level requirements; however, there are applications that are mission critical, and the service delivery to these applications is crucial. Along with the growing importance of these data services, there is also a change in the types of applications that are available. The traditional range of non-real-time applications (e.g., e-mail and ftp) is being extended to include real-time interactive applications such as voice and video services and the World Wide Web. The real-time nature of these new applications places additional demands on the network. [11]

3.3 Characterization Of Network Applications

Network applications can be characterized in terms of how predictable the data rate is (see Table 1), and how tolerant of delay delivery is (see Table 2). Generally, two-way applications are more sensitive to delay than one-way. [12]

Table 1Terms to characterize application data rates in terms of relative predictability [12]

Rate Type	*Descriptions*
Stream	Predictable delivery at a relatively constant bit rate (CBR). For example, although their rates often fluctuate, audio and video data streams are considered CBR because they have a quantifiable upper bound.

Table 1 continued

Burst	Unpredictable delivery of "blocks" of data at a variable bit rate (VBR). Applications like file transfer move data in bulk that can increase data rate to use all available bandwidth (no upper bound).

Table 2 Terms to characterize application sensitivity to data delivery delays [12]

Delay	***Tolerance***	***Delivery Type Description***
high	Asynchronous	No constraints on delivery time
	Synchronous	Data is time-sensitive, but flexible.
	Interactive	Delays may be noticeable to users/applications, but do not adversely affect usability or functionality.
	Isochronous	Time-sensitive to an extent that adversely affects usability.
low	Mission-Critical	Data delivery delays disable functionality.

3.4 Real-Time and Two-Way Applications

IP telephony is today's premier application. For IP telephony and other real-time or two-way applications, the timing requirements are much more significant than the bandwidth requirements. There is a person at either end of the conversation, and they have immediate and obvious evidence of the quality of a call or lack of it. Dropouts and delays are noticeable and distracting. Round-trip delivery delays above 0.5 second can make them unusable. ATM (Asynchronous Transfer Mode) plays an important role in telephone network backbones, and its salient feature is "quality of service" (QoS) support. By allocating resources to a virtual circuit during connection setup that remain

dedicated for the duration of the connection, ATM can satisfy the real-time (isochronous) delivery requirements of a two-way phone conversation. Performance and traffic management capabilities are the key advantages of ATM-based networks, while scalability and flexibility are the key advantages of IP-based networks. Neither type of network offers the full advantages of the other type of network. So IP networks need a way to map to the QoS of ATM and extend it to the pure-IP portions of the Internet.

3.5 Quality of Service for Different Applications

To operate the network efficiently, every application used on that network must be considered in terms of its requirements to operate effectively. While this may appear to be a daunting task, the reality is that applications typically can be grouped into a relatively small number of classes, with the applications in each class having similar requirements on the network. One class of applications has no requirements beyond that of the traditional "best effort" network. However, other classes of applications introduce new requirements (see Figure 3).

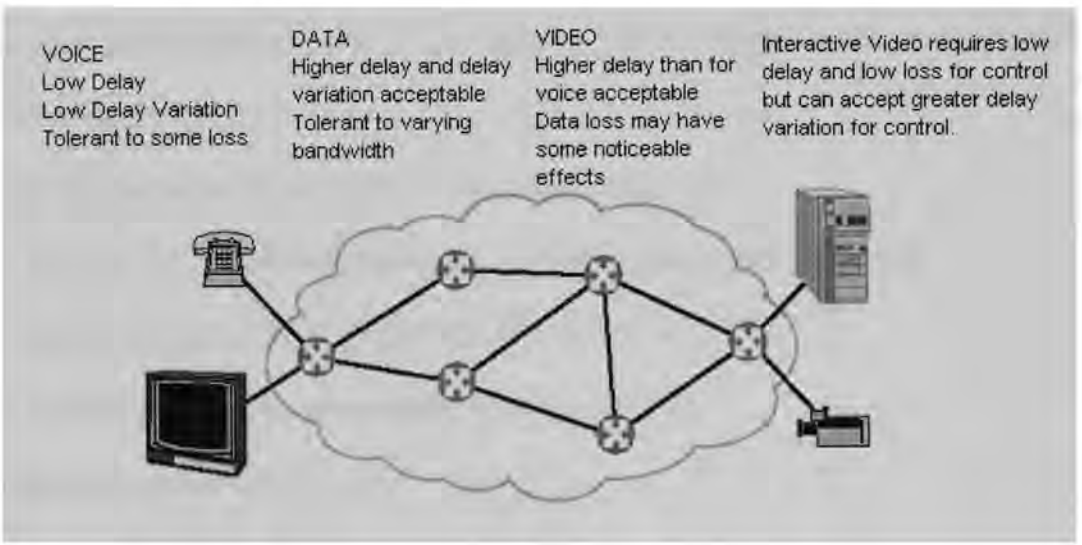

Figure 3 Different applications have different QoS requirements [13]

For these other classes, it is necessary to determine what requirements must be met to ensure the applications perform satisfactorily. [13]

3.6 Quality of Service Parameters

To implement a service level in a network for a customer, service requirements have to be expressed in some measurable QoS parameters.

There are a number of characteristics that qualify QoS, including

- Minimizing delivery delay
- Minimizing delay variations
- Providing consistent data throughput capacity

For wireless networks, jitter, bandwidth, noise, and fading are some of the deciding QoS parameters. The Internet is being used to power both intranets within the enterprise and extranets that enable electronic commerce with business partners. As business is increasingly conducted over the web, it becomes more important that IT managers ensure that these networks deliver appropriate levels of quality. Quality of Service (QoS) technologies provide the tools for IT managers to deliver mission critical business over the public network. [13]

QoS technologies allow IT managers and network managers to:

- Manage jitter sensitive applications, such as audio and video playbacks
- Manage delay-sensitive traffic, such as real time voice
- Control loss in times of inevitable bursty congestion.

3.7 Basic QOS Architecture

The basic architecture introduces the three fundamental pieces for QoS implementation. (See Figure 4)

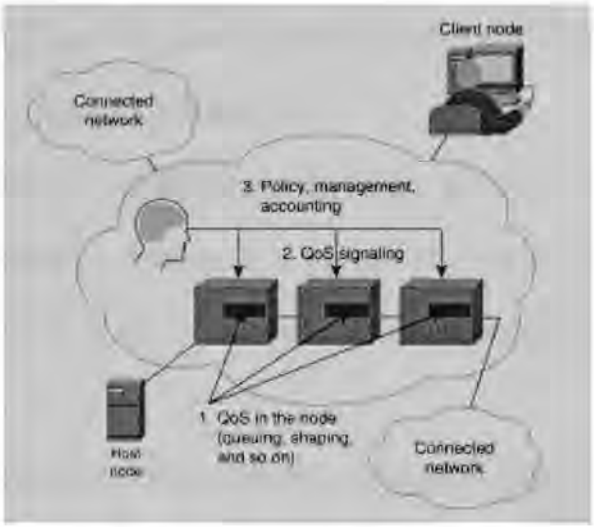

Figure 4 A Basic QoS Implementation Has Three Main Components [14]

- **Queuing, Traffic Shaping, and Filtering**, these QoS technologies are essential to traffic prioritization and congestion control, determining how a router or switch handles incoming and outgoing traffic [14]
- **Quality-of-Service Signaling** how network nodes communicate to deliver the specific end-to-end service required by applications, flows, or sets of users [14]
- **Quality-of-Service Management, Policy Control, and Accounting** provides intelligence for end-to-end monitoring, implementation and control of QoS policy.[14]
- **End-to-End Service** puts the three components together [14]

3.7.1 Queuing, Traffic Shaping, and Filtering

Quality of service (QoS) offer a variety of queuing, traffic shaping, and filtering technologies for implementing traffic priority and controlling congestion end-to-end across the network. The challenge of providing effective end-to-end QoS is that network administrators must serve disparate users, applications, organizations, and technologies,

all at a reasonable cost and effort. QoS technologies must enable administrators to balance service priority levels for user satisfaction with efficient backbone and access utilization to minimize operations expense. [14]

QoS queuing, traffic shaping, and filtering technologies gives network administrators the options they need to prioritize, reserve, and manage network resources to achieve user satisfaction in heterogeneous, multi-application network environments.

Queuing Techniques

As the most basic QoS element, smart queuing techniques efficiently prioritize a variety of traffic types and requirements for mission-critical and time-sensitive multimedia/voice application traffic.

Custom Queuing (CQ), Weighted Random Early Detection (RED or WRED are the examples of Queuing Techniques

Traffic Shaping and Filters

Traffic filters and shaping technologies actively monitor traffic flow, handle congestive conditions, and govern priority decisions among traffic flows or applications according to QoS policies set by the network administrator

Generic Traffic Shaping (GTS) -- uses queuing on an ATM, Frame Relay or other types of network to limit surges, which can cause congestion

Frame Relay Traffic Shaping (FRTS) -- uses queuing on a Frame Relay network to limit surges, which can cause congestion.

3.7.2 QoS Signaling

Quality of service (QoS) signaling is the way that end stations or network nodes implement QoS priorities across the network according to the QoS policy. For instance,

an IP network uses part of the IP packet header to request special handling of priority or time-sensitive traffic. Asynchronous Transfer Mode (ATM) networks establish QoS parameters as one aspect of signaling in ATM connections. The challenge of a robust QoS signaling solution is achieving true Layer 3 QoS service level guarantees end-to-end over heterogeneous network infrastructures.

Queuing and traffic shaping technology uses this information to provide the desired QoS handling at each node. [15]

IP Precedence -- provides the capability to partition traffic into multiple (up to six) classes of service.

RSVP -- allows applications to request a specific QoS for a data stream. Hosts and routers uses RSVP to deliver these requests to routers along the path of a data stream, and to maintain router and host state to provide the requested service, usually bandwidth and latency.

3.7.3 QoS Management Policy Control and Accounting

Network administrators must optimize network resources to attain maximum user satisfaction and investment protection. The challenge of implementing a QoS policy is effectively distributing the effort between backbone and edge devices. At the edge of the network, QoS management and policy control features enable administrators to specify policies that establish traffic classes and service levels; define how network resources are allocated and controlled to handle these traffic classes; efficiently map packets into the traffic classes; apply QoS policies and "high-touch" services to meet application and security requirements; and collect and export detailed network traffic and service resource allocation statistics. [15]

3.7.4 End-to-End QoS Services

The ability of a network to deliver service needed by a specific network applications from end-to end with some level of control over delay, loss, jitter, and/or bandwidth can be categorized into three levels of service:

Best Effort Service -- basic connectivity with no guarantees. The Internet is an example of best effort level of service.

Differentiated Service -- Expedited handling for specific classes of traffic. An example of differentiated services is differentiated Service for SNA on Intranets

Guaranteed Service -- a reservation of network resources to ensure that specific traffic gets a specific level of service it requires. An example of guaranteed service is:

- Advanced QoS Services for the Intelligent Internet
- Cisco IOS Software Quality of Service Solutions

4.0 TECHNOLOGIES FOR QUALITY OF SERVICE

4.1 Introduction

Different technologies like Type Of Service (TOS) and Integrated Service (Int-Serv) were created in an attempt to provide some quality-of-service (QoS) control. One of these early technologies allowed the network to distinguish between network control traffic and user traffic. Based on the TOS byte defined in the IP header, this technique provides a coarse-grained service classification and a small number of service classes. Here, the service classes were defined very early when the exact needs of applications and users were unclear. It has been revealed since that the definition of the classes was not well suited for providing the required range of services. Because of this, the byte is often not fully supported in routers and hosts. The most prominent QoS techniques are the integrated services and the differentiated services.

4.2 Integrated Services

The integrated service model (Int-Serv) [16] is based on a goal to augment the best-effort service model traditionally provided by connectionless IP networks. Int-Serv provides a well-defined, end-to-end service between hosts for both point-to-point and point-to-multipoint applications. In Int-Serv, the application initiates a session on demand with the network using the Resource Reservation Signaling Protocol (RSVP). This session identifies the service requirements of the application, including information such as bandwidth and delay, and the source of the data.

4.2.1 RSVP Overview

RSVP is a resource reservation setup protocol that is used by both network hosts and outers. It is designed to interact with integrated services on the Internet.

Hosts use RSVP to request a specific quality of service (QoS) from the network for particular application flows. Routers use RSVP to deliver QoS requests to all routers along the data path. RSVP also can maintain and refresh states for a requested QoS application flow. RSVP treats an application flow as a simplex connection. That is, the QoS request travels only in one direction—from the sender to the receiver. RSVP is a transport layer protocol that uses IP as its network layer. However, RSVP does not transport application flows. Rather, it is more of an Internet control protocol, similar to ICMP, IGMP, IS-IS, or OSPF.

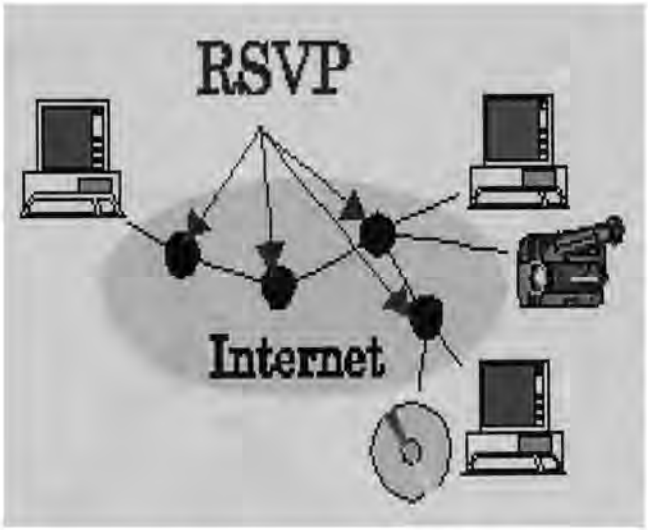

Figure 5 Overview of RSVP [16]

RSVP is not a routing protocol, but rather is designed to operate with current and future unicast and multicast routing protocols. The routing protocols are responsible for choosing the routes to use to forward packets, and RSVP consults local routing tables to obtain routes. RSVP is responsible only for ensuring the QoS of packets traveling along a data path. The receiver in an application flow is responsible for requesting the preferred QoS from the sender. To do this, the receiver issues an RSVP QoS request on behalf of

the local application. The request propagates to all routers in reverse direction of the data paths toward the sender. In this process, RSVP requests might be merged, resulting in a protocol that scales well when there are a large number of receivers. [16]

4.2.2 RSVP Operation

RSVP creates independent sessions to handle each data flow. A session is identified by a combination of the destination address, an optional destination port, and a protocol. Within a session, there can be one or more senders. Each sender is identified by a combination of its source address and source port. An obsolete mechanism, such as a session announcement protocol, is used to communicate the session identifier to all senders and receivers.[17]

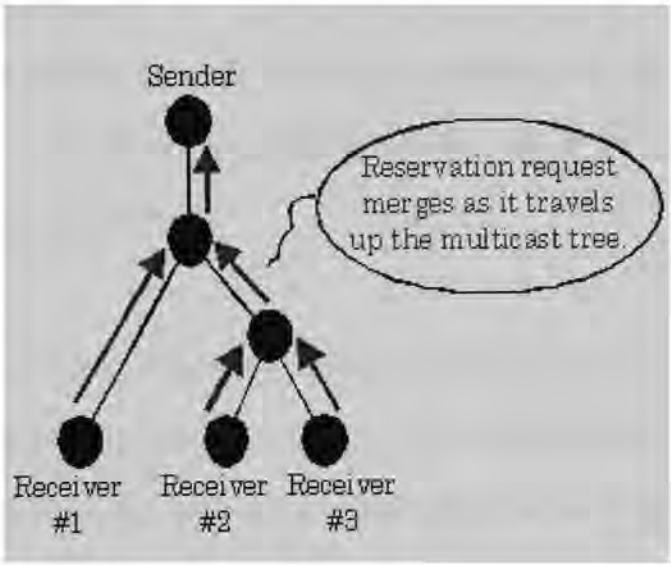

Figure 6 A RSVP Session [18]

A typical RSVP session involves the following sequence of events:

1. A potential sender starts sending RSVP Path messages to the session address.

2. A receiver, wanting to join the session, registers itself if necessary. For example, a receiver in a multicast application would register itself with IGMP.

3. The receiver receives the Path messages.

4. The receiver sends appropriate Resv messages toward the sender. These messages carry a flow descriptor, which is used by routers along the path to make reservations in their link-layer media.

5. The sender receives the Resv message, and then it starts sending application data.

This sequence of events is not necessarily strictly synchronized. For example, receivers can register themselves before receiving Path messages from the sender, and application data can flow before the sender receives Resv messages. Application data that is delivered before the actual reservation contained in the Resv message typically is treated as best effort, nonreal-time traffic with no QoS guarantee. [18]

4.2.3 RSVP Message Types

RSVP uses several types of messages to establish and remove paths for data flows, to establish and remove reservation information, to confirm the establishment of reservations, and to report errors.

Path Messages

Each sender host transmits Path messages downstream along the routes provided by the unicast and multicast routing protocols. Path messages follow the exact paths of application data, creating path states in the routers along the way, thus enabling routers to learn the previous hop and next-hop node for the session. Path messages are sent periodically to refresh path states. The refresh interval is controlled by a variable called the *refresh time*, which is the periodical refresh timer expressed in seconds. A path state times out if a router does not receive a specified number of consecutive Path messages. This number is specified by a variable called *keep-multiplier*. [19]

Path states are kept for ((*keep-multiplier* + 0.5) * 1.5 * *refresh-time*)seconds.

Resv Messages

Each receiver host sends reservation request (Resv) messages upstream toward senders and sender applications. Resv messages must follow exactly the reverse path of Path messages. Resv messages create and maintain a reservation state in each router along the way. Resv messages are sent periodically to refresh reservation states. The refresh interval is controlled by the same refresh time variable, and reservation states are kept for

((*keep-multiplier* + 0.5) * 1.5 * *refresh-time*) seconds.[19]

PathTear Messages

PathTear messages remove (tear down) path states as well as dependent reservation states in any routers along a path. PathTear messages follow the same path as Path messages. A PathTear typically is initiated by a sender application or by a router when its path state times out. PathTear messages are not required, but they enhance network performance because they release network resources quickly. If PathTear messages are lost or not generated, path states eventually time out when they are not refreshed, and then the resources associated with the path are released.[19]

ResvTear Messages

ResvTear messages remove reservation states along a path. These messages travel upstream toward senders of the session. In a sense, ResvTear messages are the reverse of Resv messages. ResvTear messages typically are initiated by a receiver application or by a router when its reservation state times out. ResvTear messages are not required, but they enhance network performance because they release network resources quickly. If

ResvTear messages are lost or not generated, reservation states eventually time out when they are not refreshed, and then the resources associated with the reservation are released.[19]

PathErr Messages-When path errors occur (usually because of parameter problems in a Path message), the router sends a unicast PathErr message to the sender that issued the Path message. Using PathErr messages is advisory; these messages do not alter any path state along the way.

ResvErr Messages-When a reservation request fails, a ResvErr error message is delivered to all the receivers involved. Using ResvErr messages is advisory; these messages do not alter any reservation state along the way.

ResvConfirm Messages-Receivers can request confirmation of a reservation request, and this confirmation is sent with ResvConfirm message. Because of the complex RSVP flow-merging rules, a confirmation message does not necessarily provide end-to-end confirmation of the entire path. Therefore, ResvConfirm messages are an indication of potential success only, with no guarantees.

4.2.4 RSVP Reservation Styles

A reservation request includes options for specifying the reservation style. The reservation styles define how reservations for different senders within the same session are treated and how senders are selected. [20] Two options specify how reservations for different senders within the same session are treated:

Distinct reservation—Each receiver establishes its own reservation with each upstream sender.

Shared reservation—All receivers make a single reservation that is shared among many senders.

Two options specify how senders are selected:

Explicit sender—List all selected senders.

Wildcard sender—Select all senders, which then participate in the session.

The following reservation styles, formed by a combination of these four options, currently are defined:

Fixed filter (FF)—This reservation style consists of distinct reservations among explicit senders.

Examples of applications that use fixed-filter style reservations are video applications and unicast applications, which both require flows that have a separate reservation for each sender.

Wildcard filter (WF)—This reservation style consists of shared reservations among wildcard senders. This type of reservation reserves bandwidth for any and all senders, and propagates upstream toward all senders, automatically extending to new senders as they appear.

A sample application for wildcard filter reservations is an audio application in which each sender transmits a distinct data stream. Typically, only a few senders are transmitting at any one time. Such a flow does not require a separate reservation for each sender; a single reservation is sufficient.

Shared explicit (SE)—This reservation style consists of shared reservations among explicit senders. This type of reservation reserves bandwidth for a limited group of senders.

A sample application is an audio application similar to that described for wildcard filter reservations.

4.2.5 Limitations Of Integrated Services And RSVP

IntServ is an architecture requiring per-flow traffic handling at every hop along an applications end-to-end path and explicit signaling of each flows requirements using a signaling protocol like RSVP. IntServ suffers from lack of scalability due to the scalability problems with the standard RSVP signaling protocol.

The objective of Int-Serv has dictated the properties of the traditional RSVP protocol, such as the soft-state nature and the merging of resource requests. While these aspects of Int-Serv make it powerful, enabling it to guarantee the minimum requirements of an application will be met, it also imposes a high price in terms of the processing power and signaling required.

Traditional RSVP requires a state machine that includes timers for each session and a classifier in each router, which makes both memory and processing capacity expensive. In an Internet backbone router, there can be many of these sessions with individual users and hosts, and these routers do not have the necessary resources to deal with all of the sessions. This limitation generally restricts Int-Serv deployment to the edge of the network and tunnels it through the backbone. This reduces the effectiveness, since there is now no guarantee that the tunneled part with the remainder of the session meets the end-to-end requirements.

Such limitations in these technologies have restricted their use, and they have been unable to provide a framework for provision of service to meet service-level agreements. It is obvious that something different is required to enable services on the

Internet given today's state-of-the-art router capabilities. As a result, the differentiated services architecture was proposed.

4.3 Differentiated Services

Differentiated Services (DiffServ) is based on an architecture that pushes complex decision making to the edges. This results in less processing load on core routers and, thus, faster operation, due to less signal state processing and storage. According to this architecture, a differentiated services code point (DSCP) is carried in every packet. This is carried in the old IP type of service (ToS) field. Classification, rate shaping, and policing are done at the edge routers and packets are mapped onto service levels. Per-hop queuing and scheduling behaviors, or simply per-hop behaviors (PHBs), are defined through which a number of edge-to-edge services might be built. The expedited forwarding (EF) PHB [21] requests every router along the path to always service EF packets at least as fast as the rate at which EF packets arrive. This entails rate shaping and expedited packet servicing at the routers. As a consequence EF is used to achieve low-loss, low-latency, and low-jitter edge-to-edge services. The assured forwarding (AF) PHB [22] supports more flexible and dynamic sharing of network resources by supporting soft bandwidth and loss guarantees appropriate for bursty traffic. At each transit node, the DSCP is used to select the per-hop-behavior (PHB) that determines the scheduling treatment and drop probability of each packet.

Thus, unlike Int-Serv, the objective of Diffserv is not to provide an end-to-end service for the host/application. Rather, the goal is to create a set of "building blocks" that provide a foundation for building end-to-end services throughout the network. Even though, Diffserv takes an approach similar to TOS, it is better targeted to meet the needs

of today's applications. Diffserv is sufficiently scalable that it can be supported in routers at the core of the Internet since it avoids per-session states. Instead, each packet carries information about the service class.

4.3.1 Working of Diffserv

Diffserv is a strategy for providing QoS across a network through a set of "building blocks" that can be used together in order to achieve an end-customer service. One of these key building blocks is the Per Hop Behavior (PHB) (Figure #7).

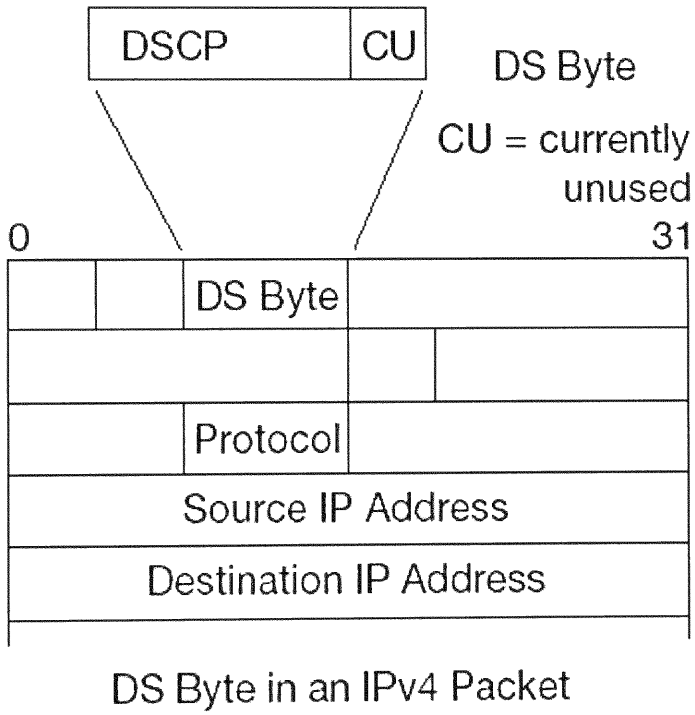

Figure 7 DSCP in the IP header [21]

Diffserv defines a number of data treatments known as a PHB that can be applied to the packets in each node. The PHB is used to identify the treatment that will be given to the packet within the node. This "treatment" includes selection of the queue and scheduling discipline to apply at the egress interface and congestion thresholds. For example, a

treatment can select a queue with a high-scheduling priority but a low threshold for congestion, and a congestion management scheme. If a packet is given a similar treatment at each node throughout the network, then the effect on packets across the network end-to-end can be identified. The packets are marked to identify the treatment that the packets must receive using the DS byte. The DS byte replaces the TOS byte in the IP header. This byte was selected because it originally was intended to be used to indicate service information, but, as mentioned, its use has been limited.[14]

Within the DS byte, a Diffserv CodePoint (DSCP) field has been defined. The value in this field identifies the "behavior" or "treatment" to be applied to the packet within that node in the network. As the packet progresses through the network, each node applies the same "forwarding treatment" to the packet. The DS byte contains 6 bits for the DSCP, plus 2 bits that are currently unused and reserved for the future. The 6 bits are used as an indexed table to identify the PHB, rather than used as bit fields. This allows for 64 independent codepoints. These are mapped in the node to determine the "treatment" to apply. Depending on this mapping, it is possible to have multiple codepoints selecting the same behavior, allowing local use of existing behaviors for different purposes. [14]

4.3.2 The Defined Per Hop Behaviors

The defined PHBs are:

4.3.2.1 Expedited Forwarding (EF)

The EF PHB provides a low-loss, low-jitter, and low-delay handling within the node. To provide the low delay, the EF handling further defines that the maximum aggregated reception rate of data for this PHB at any time must be no greater than the

minimum transmission rate available for this PHB per egress (see Figure 8, For EF PHB, the sum of the ingress rates must not exceed the egress rate in each node). This requirement ensures that there is no queue buildup occurring for this service within the node (apart from synchronous data arrival compared to the packet transmission period), which minimizes the delay and the delay variation. [23]

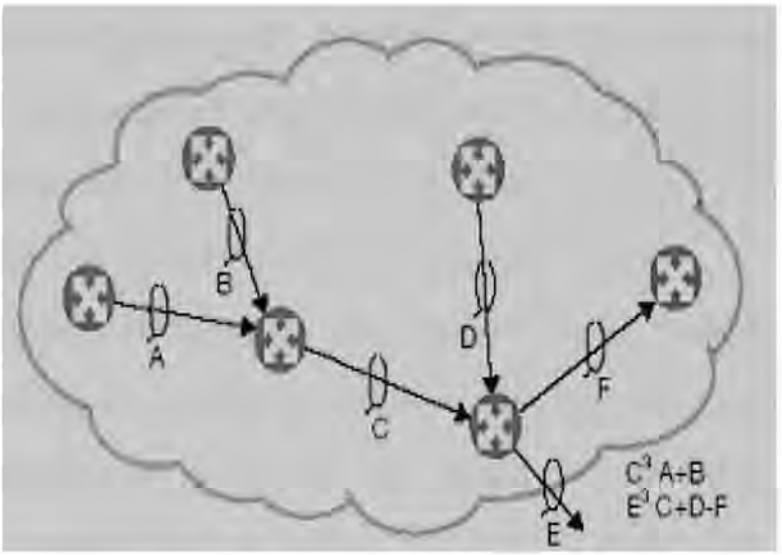

Figure 8 Expedite Forwarding [23]

4.3.2.2 Assured Forwarding (AF)

The PHB group defines N independent forwarding classes (4 defined currently) denoted as AF1 to AFn. Within each of these forwarding classes, there are also M subclasses (3 defined currently) for probability of delivery. The higher level of delivery probability should have a greater probability than the second level of getting data through in times of congestion. Likewise, the second level should have a better probability for delivery than the third level. Each forwarding class within this group is configured

independently for resources such as buffer space and minimum egress capacity that should be ensured by the scheduling mechanism. [23]

Scheduling Mechanisms

The node must decide how packets from different PHBs are to be scheduled out on the link. The scheduling mechanism may consider priority order between classes, but it must control access to link bandwidth for each class. A strict priority mechanism between two or more classes aims to provide the lowest possible delay for the highest priority class. This mechanism sends the data from the highest priority class before sending data for the next class. This could lead to starvation of lower priority classes, so the traffic level must be shaped to limit the used bandwidth. [24]

Weighted Round Robyn (WRR) aims to give a weighted access to the available bandwidth to each class, ensuring a minimum allocation and distribution. The scheduling services each class in a round-robin manner according to the weights. If one or more classes is not using its full allocation, then the unused capacity is distributed to the other classes according to their weighting. A class can be given a lower effective delay by giving it a higher weighting than the traffic level it is carrying. [24]

Weighted Fair Queuing (WFQ) similarly aims to distribute available bandwidth over a number of weighted classes. The scheduling mechanism uses a combination of weighting and timing information to select which queue to service. The weighting effectively again controls the ration of bandwidth distribution between classes under congestion and can also indirectly control delay for underutilized classes. [24]

Class Based Queuing (CBQ) is a more general term for any mechanism that is based on the class. CBQ can allow the unused capacity to be distributed according to a different

algorithm than a minimum bandwidth weighting. For example, there could be a different weighting, which is configured for this excess capacity, or it could be dependent on the traffic load in each class, or some other mechanism such as priority. [24]

4.3.2.3 Default Behavior (DE)

The DE PHB identifies the existing "best effort" traffic. The behavior defines that the node will deliver as many of these packets as possible, as soon as possible. Other defined behaviors have greater requirements on timeliness of delivery. So, the DE definition of "as many as possible" and "as soon as possible" allows deference to these other behaviors.[23]

4.3.2.4 Other PHBs

The Class Selector (CS) codepoints create a set of codepoints for backward compatibility with the precedence field of the IPv4 TOS byte that is now used as the DS byte. These PHBs ensure that routers implementing TOS will provide compatible behavior to routers employing Diff-Serv for these values. Apart from the specific codepoints that have already been defined, there are also spare codepoints. Some spare codepoints have been left for definition of more assured forwarding PHBs, if required in the future. In addition, certain ranges of codepoints have been set aside for local and experimental use. These may be used by a network operator as they see fit or by network equipment vendors to implement additional classes. [23]

5.0 MULTIPROTOCOL LABEL SWITCHING

5.1 What is MPLS and Motivation

As mentioned earlier in chapter 3(Section 3.4), IP networks need a way to map to the QoS of ATM and extend it to the pure-IP portions of the Internet for better Performance and traffic management capabilities.

Multiprotocol Label Switching (MPLS) is a promising effort to provide the kind of traffic management and connection oriented Quality of Service (QoS) support found in Asynchronous Transfer Mode (ATM) networks, to speed up the IP packet-forwarding process, and to retain the flexibility of an IP-based networking approach. The Internet Engineering Task Force (IETF) standardized the Multiprotocol Label Switching (MPLS) architecture to merge the strengths of IP and ATM networks into one standards-based alternative. [25]

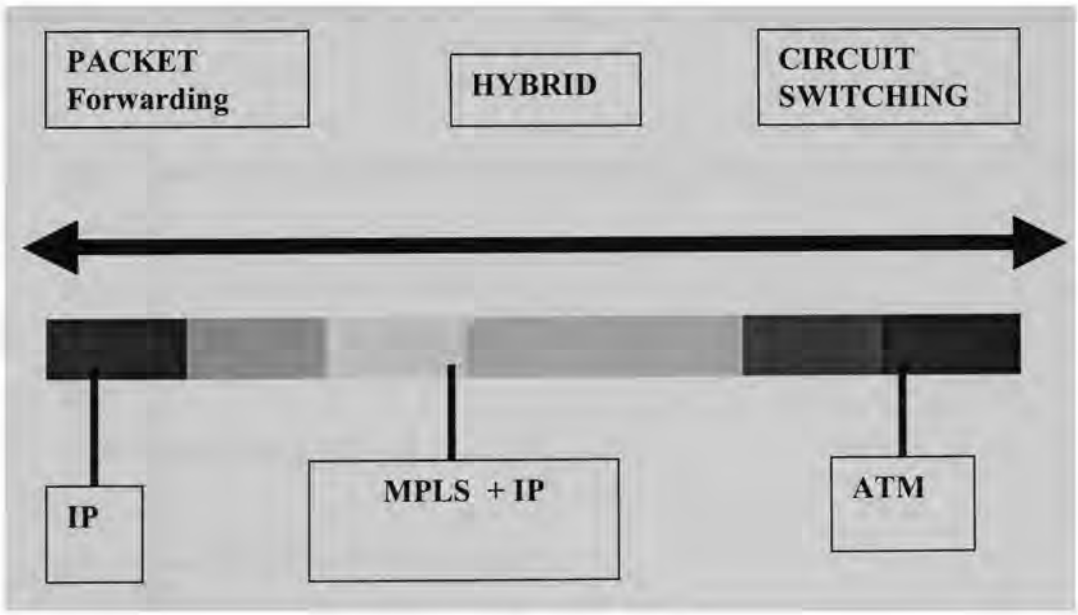

Figure 9 Best of both worlds [26]

5.2 Background

The roots of MPLS go back to numerous efforts in the mid-1990s to combine IP and ATM technologies. The first such effort to reach the marketplace was IP switching, developed by Ipsilon. To compete with this offering, numerous other companies announced their own products, notably Cisco Systems (Tag Switching), IBM (aggregate routebased IP switching), and Cascade (IP Navigator). The goal of all these products was to improve the throughput and delay performance of IP, and all took the same basic approach: Use a standard routing protocol such as Open Shortest Path First (OSPF) to define paths between endpoints; assign packets to these paths as they enter the network; and use ATM switches to move packets along the paths. When these products came out, ATM switches were much faster than IP routers, and the intent was to improve performance by pushing as much of the traffic as possible down to the ATM level and using ATM switching hardware. [26]

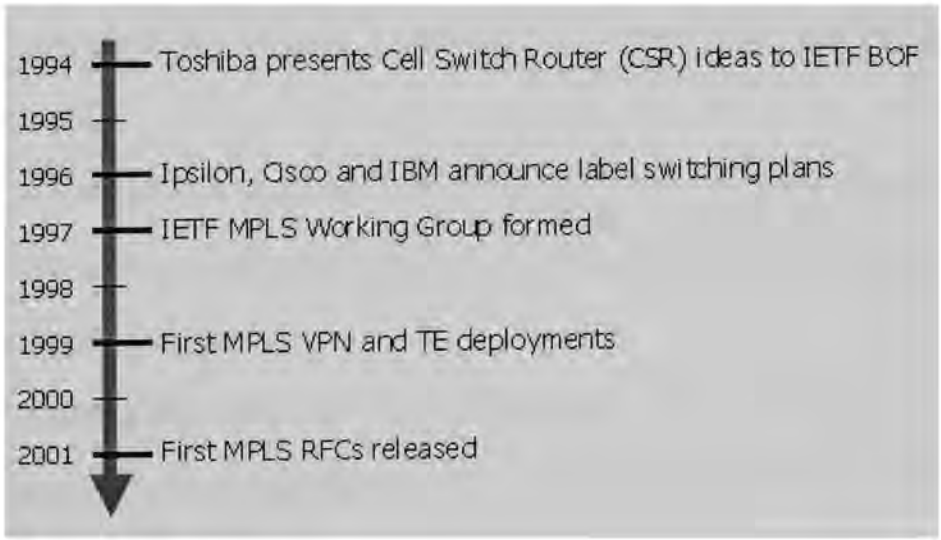

Figure 10: History Of MPLS [26]

In response to these proprietary initiatives, the *Internet Engineering Task Force* (IETF) set up the MPLS working group in 1997 to develop a common, standardized approach. The working group issued its first set of Proposed Standards in 2001. Meanwhile, however, the market did not stand still. The late 1990s saw the introduction of many routers that are as fast as ATM switches, eliminating the need to provide both ATM and IP technology in the same network.

Nevertheless, MPLS has a strong role to play. MPLS reduces the amount of per-packet processing required at each router in an IP-based network, enhancing router performance even more. More significantly, MPLS provides significant new capabilities in four areas that have ensured its popularity: QoS support, traffic engineering, Virtual Private Networks (VPNs), and multiprotocol support. [26]

5.3 MPLS Terminology

Label: A short fixed length physically contiguous identifier, which is used to identify an FEC, usually of local significance.

LDP: Label Distribution Protocol is a specification, which lets an LSR distribute labels to its LDP peers.

LSP: Label Switched Path, a path that MPLS sets to transmit packets based upon the labels. LSP is setup and provisioned prior to actual data forwarding using the label distribution protocols (LDP).

FEC: Forwarding Equivalence Class is a representation of group of packets that share the same requirements for their transport. FEC's are based on service requirements for a given set of packets.

LSR: Label Switching Router is a router that is capable of switching and routing packets on the basis of a label appended to each packet. (Usually located at the core of the network).

LER: Label Edge Router is capable of utilizing the routing information and assigns labels to packets and then forwards them into MPLS domain. (Usually located at the edge of the network).

TTL: In conventional IP forwarding, each packet carries a Time To Live (TTL) value in its header. Whenever a packet passes through a router, its TTL gets decremented by 1; if the TTL reaches 0 before the packet has reached its destination, the packet gets discarded.

5.4 Operation of MPLS

MPLS works through a process of Label Switching, whereby the forwarding path is independent of the data content. This could be clarified further using the following highway analogy, assuming you need to get from point A to point B: [27]

- Have a friend go down the high way ahead of you.
- At every intersection they would posts a big sign that says where you need to go if you are on this lane, and reserve the corresponding lane for you before you even start your trip.
- These road signs are analogous to the labels in MPLS, and the process by which these signs are posted is the label substitution process itself. (This is illustrated in
- the figure 11).

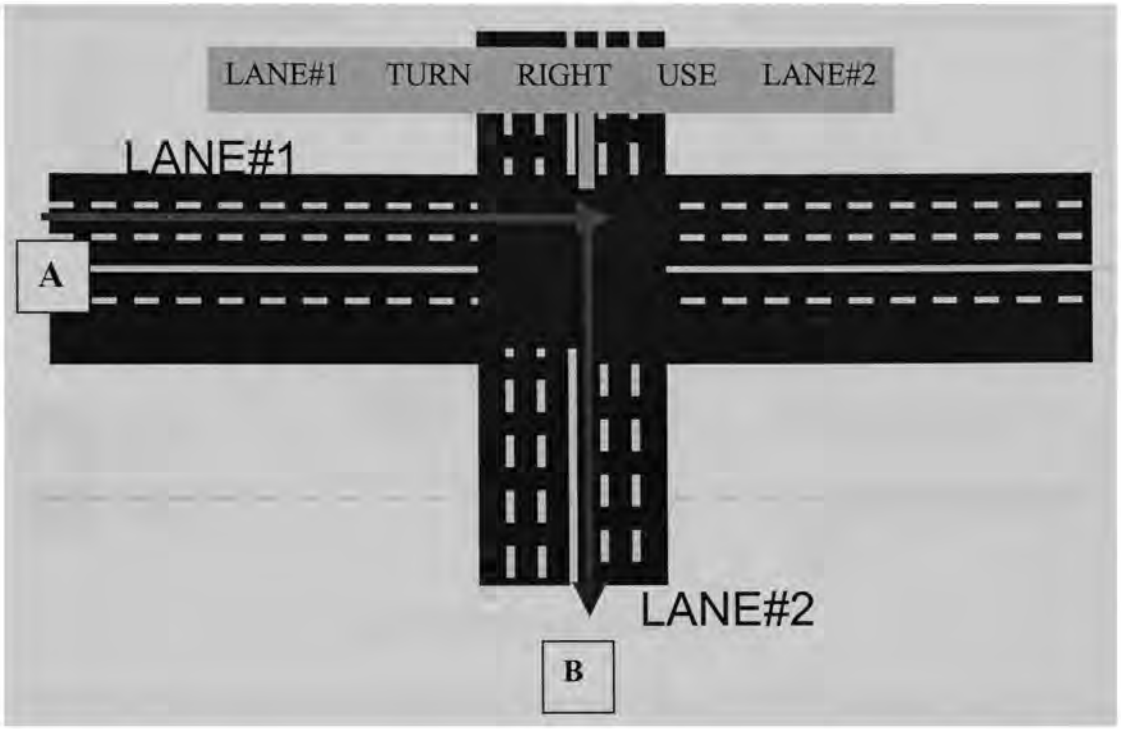

Figure 11 Label Substitution Analogy [27]

An MPLS network or Internet consists of a set of nodes, called Label Switched Routers (LSRs) that are capable of switching and routing packets on the basis of a label, which has been appended to each packet. Labels define a flow of packets between two endpoints or, in the case of multicast, between a source endpoint and a multicast group of destination endpoints. For each distinct flow, called a Forwarding Equivalence Class (FEC), a specific path through the network of LSRs is defined. Thus, MPLS is a connection-oriented technology. Associated with each FEC is a traffic characterization that defines the QoS requirements for that flow. The LSRs do not need to examine or process the IP header, but rather simply forward each packet based on its label value. Therefore, the forwarding process is simpler than with an IP router.

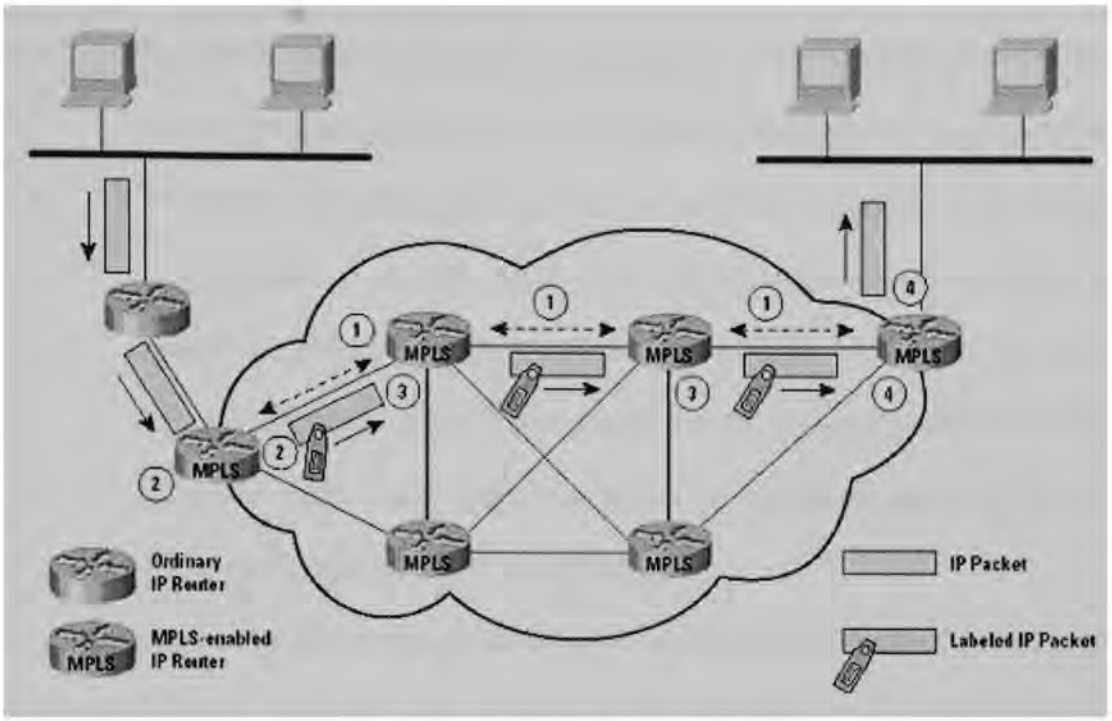

Figure 12 Operation Of MPLS [28]

Figure depicts the operation of MPLS within a domain of MPLS-enabled routers. The following are key elements of the operation.

1. Prior to the routing and delivery of packets in a given FEC, a path through the network, known as a Label Switched Path (LSP), must be defined and the QoS parameters along that path must be established. The QoS parameters determine (1) how many resources to commit to the path, and (2) what queuing and discarding policy to establish at each LSR for packets in this FEC. [28]

To accomplish these tasks, two protocols are used to exchange the necessary information among routers:

a. An interior routing protocol, such as OSPF, is used to exchange reachability and routing information.

b. Labels must be assigned to the packets for a particular FEC. Because the use of globally unique labels would impose a management burden and limit the number of usable labels, labels have local significance only, as discussed subsequently. A network operator can specify explicit routes manually and assign the appropriate label values. Alternatively, a protocol is used to determine the route and establish label values between adjacent LSRs. Either of two protocols can be used for this purpose: the Label Distribution Protocol (LDP) or an enhanced version of RSVP.

2. A packet enters an MPLS domain through an ingress edge LSR where it is processed to determine which network-layer services it requires, defining its QoS. The LSR assigns this packet to a particular FEC, and therefore a particular LSP, appends the appropriate label to the packet, and forwards the packet. If no LSP yet exists for this FEC, the edge LSR must cooperate with the other LSRs in defining a new LSP.

3. Within the MPLS domain, as each LSR receives a labeled packet, it:

a. Removes the incoming label and attaches the appropriate outgoing label to the packet.

b. Forwards the packet to the next LSR along the LSP.

4. The egress edge LSR strips the label, reads the IP packet header, and forwards the packet to its final destination.

5.5 Modes of Operation

5.5.1 Label-Controlled ATM

In this model the ATM Forum switching is replaced by IP/MPLS but still uses the ATM hardware to perform switching, such that the LSR can be viewed as two distinct

functional components - a control component and a forwarding component (see Figure 12).

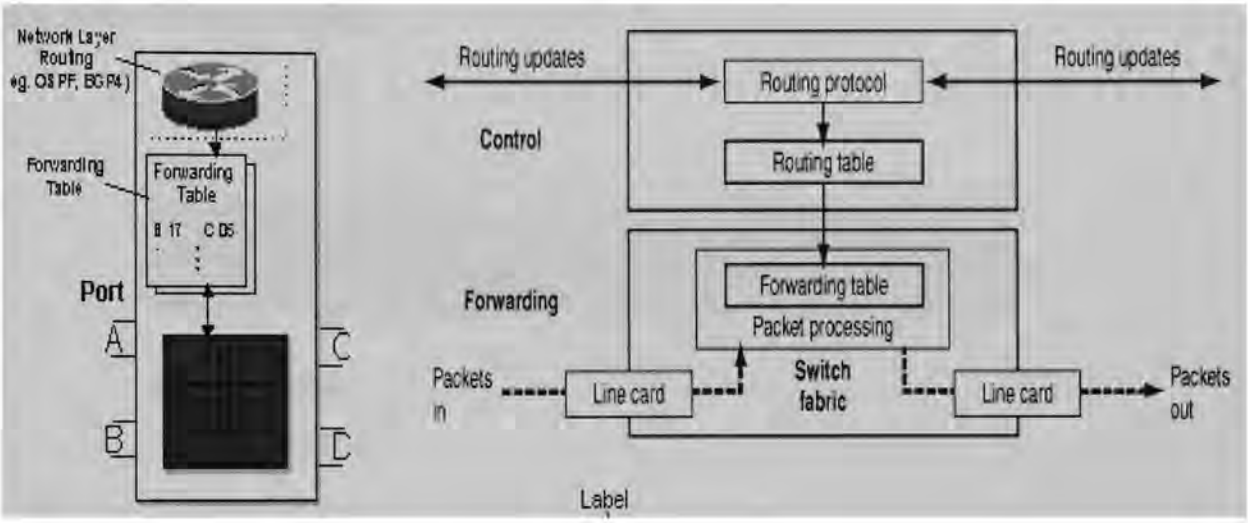

Figure 13 Block diagram of LSR [29]

The control component uses standard routing protocols (OSPF, IS-IS, and BGP-4) to exchange information with other routers to build and maintain a forwarding table. When packets arrive, the forwarding component searches the forwarding table maintained by the control component to make a routing decision for each packet. Specifically, the forwarding component examines information contained in the packet's header, searches the forwarding table for a match, and directs the packet from the input interface to the output interface across the system's switching fabric. [29]

5.5.2 Ships in the night with ATM:

ATM label switch simultaneously functions as a conventional ATM switch and an MPLS LSR. This allows a single device to simultaneously operate as both an MPLS LSR and an ATM switch independently of each other. ATM Forum and MPLS control planes both run on the same hardware but are isolated from each other, i.e. they do not interact;

thus the ships in the night analogy. This mode is important for migrating MPLS into an ATM network [29]

5.6 MPLS Header Format

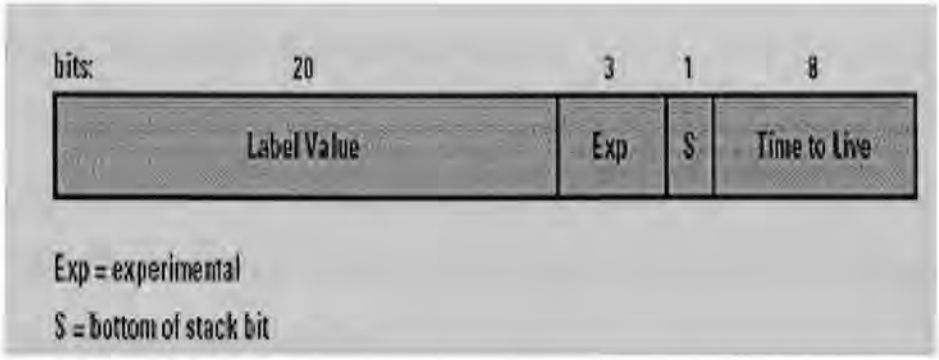

Figure 14: MPLS header format [30]

The MPLS Label is formatted as follows:

The label consists of a 32-bit MPLS label, which is located after the Layer 2 header and before the IP header. The MPLS label contains the following fields:

1 The label field (20-bits) carries the actual value of the MPLS label, in the range 0 through 1048575.

a) 0 through 15--Reserved and have special semantics.

b) 16 through 1023--Unused and unassigned by the software, a feature that is specific to the JUNOS software. You can use labels to manually configure static LSPs, and ensure that there are no conflicts with labels that are dynamically assigned by the software.

c) 1024 through 99,999--Reserved for future applications.

d) 100,000 through 1,048,575--Automatically negotiated, assigned,

released, and reused by the software. Typically, per-box labels are assigned in the 100,000-799,999 range and per-interface labels are assigned in the 800,000-1,048,575 range.

2 *Exp*: 3 bits reserved for experimental use; for example, these bits could communicate DS information or PHB guidance.

3 The Stack (S) field (1-bit) supports a hierarchical label stack. It is set to one for the oldest entry in the stack, and zero for all other entries.

4 The TTL (time-to-live) field (8-bits) provides conventional IP TTL functionality. This is also called a "Shim" header.

5.6.1 Special Labels

Some of the reserved labels (in the 0 through 15 range) have well-defined meanings. [30]

- 0, IPv4 Explicit Null Label--This value is legal only when it is the sole label entry (no label stacking). It indicates that the label must be popped upon receipt. Forwarding continues based on the IPv4 packet.
- 1, Router Alert Label--When a packet is received with a top label value of 1, it is delivered to the local software module for processing.
- 2, IPv6 Implicit Null Label--This value is legal only when it is the sole label entry (no label stacking). It indicates that the label must be popped upon receipt. Forwarding continues based on the IPv6 packet.
- 3, Implicit Null Label--This label is used in the control protocol (LDP, RSVP) only to request label popping by the downstream router. It never actually appears in the encapsulation. Labels with a value of 3 should not be used in the data packet as a real label. No payload type (IPv4 or IPv6) is implied with this label.

- 4 through 15--Unassigned.

Special labels are commonly used between the egress and penultimate routers of an LSP. If the LSP is configured to carry IPv4 packets only, the egress router might signal the penultimate router to use 0 as a final hop label. If the LSP is configured to carry IPv6 packets only, the egress router might signal the penultimate router to use 2 as a final hop label.

The egress router might simply signal the penultimate router to use 3 as the final label, which is a request to perform penultimate hop label popping. This means an egress router will not process a labeled packet; rather, it receives the payload (either IPv4, IPv6 or others) directly. This reduces one MPLS lookup at egress.

For label-stacked packets, the egress router will receive an MPLS label packet with its top label already popped by the penultimate router. The egress router cannot receive label-stacked packets using label 0 or 2. [31]

5.6.2 Time-to-Live Processing

A key field in the IP packet header is the TTL field (IPv4), or Hop Limit (IPv6). In an ordinary IP-based Internet, this field is decremented at each router and the packet is dropped if the count falls to zero. This is done to avoid looping or having the packet remain too long in the Internet because of faulty routing. Because an LSR does not examine the IP header, the TTL field is included in the label so that the TTL function is still supported. The rules for processing the TTL field in the label are as follows:

1 When an IP packet arrives at an ingress edge LSR of an MPLS domain, a single label stack entry is added to the packet. The TTL value of this label stack entry is set to the value of the IP TTL value. If the IP TTL field needs to be decremented, as part of the IP

value of the IP TTL value. If the IP TTL field needs to be decremented, as part of the IP processing, it is assumed that this has already been done.

When an MPLS packet arrives at an internal LSR of an MPLS domain, the TTL value in the top label stack entry is decremented [31]. Then:

a. If this value is zero, the MPLS packet is not forwarded. Depending on the label value in the label stack entry, the packet may be simply discarded, or it may be passed to the appropriate "ordinary" network layer for error processing (for example, for the generation of an Internet Control Message Protocol [ICMP] error message).
b. If this value is positive, it is placed in the TTL field of the top label stack entry for the outgoing MPLS packet, and the packet is forwarded. The outgoing TTL value is a function solely of the incoming TTL value, and is independent of whether any labels are pushed or popped before forwarding. There is no significance to the value of the TTL field in any label stack entry that is not at the top of the stack.

2 When an MPLS packet arrives at an egress edge LSR of an MPLS domain, the TTL value in the single label stack entry is decremented and the label is popped, resulting in an empty label stack. Then:

a. If this value is zero, the IP packet is not forwarded. Depending on the label value in the label stack entry, the packet may be simply discarded, or it
b. May be passed to the appropriate "ordinary" network layer for error processing.

c. If this value is positive, it is placed in the TTL field of the IP header, and the IP packet is forwarded using ordinary IP routing. Note that the IP header checksum must be modified prior to forwarding.

5.6.3 Label Stack

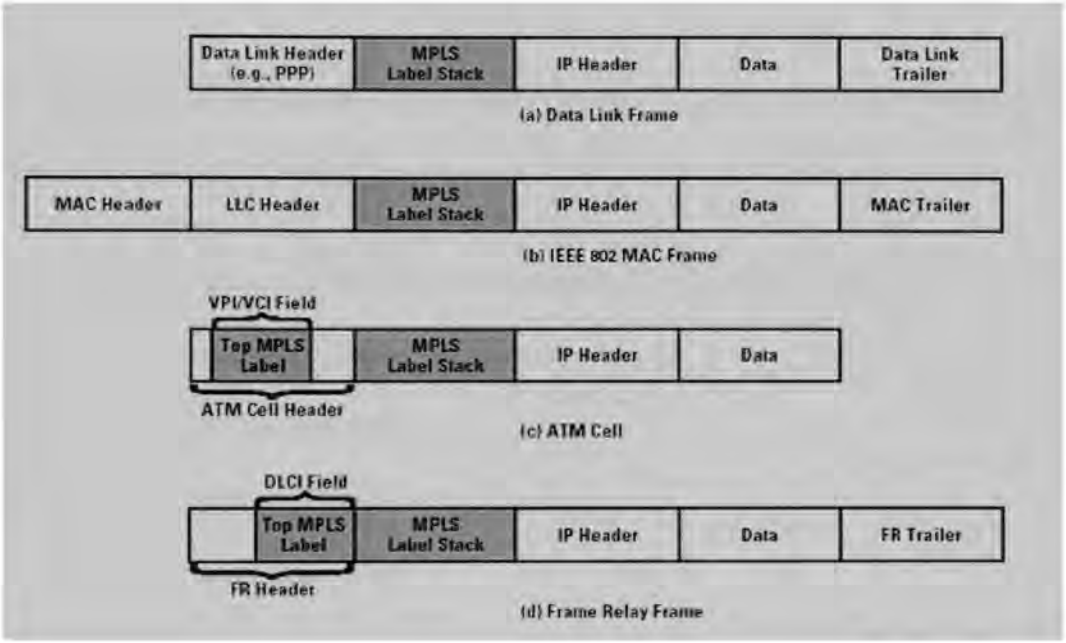

Figure 15 Position Of MPLS Label [32]

The label stack entries appear after the data link layer headers, but before any network layer headers. The top of the label stack appears earliest in the packet (closest to the network layer header), and the bottom appears latest (closest to the data link header). The network layer packet immediately follows the label stack entry that has the S bit set.

In a data link frame, such as for the Point-to-Point Protocol (PPP), the label stack appears between the IP header and the data link header (Figure 15a). For an IEEE 802 frame, the label stack appears between the IP header and the Logical Link Control (LLC) header (Figure 15b). [32]

5.7 Key features of MPLS

1 An MPLS domain consists of a contiguous, or connected, set of MPLS-enabled routers. Traffic can enter or exit the domain from an endpoint on a directly connected network. Traffic may also arrive from an ordinary router that connects to a portion of the Internet.

2 One or more of a number of parameters, as specified by the network manager can determine the FEC for a packet. Among the possible parameters:

- Source or destination IP addresses or IP network addresses
- Source or destination port numbers
- IP protocol ID
- Differentiated services codepoint
- IPv6 flow label

3. Forwarding is achieved by doing a simple lookup in a predefined table that maps label values to next-hop addresses. There is no need to examine or process the IP header or to make a routing decision based on destination IP address.
4. A particular *Per-Hop Behavior* (PHB) can be defined at an LSR for a given FEC. The PHB defines the queuing priority of the packets for this FEC and the discard policy.

5 Packets sent between the same endpoints may belong to different FECs. Thus, they will be labeled differently, will experience different PHB at each LSR, and may follow different paths through the network. Similarly packets destined to different points can be mapped to a same path. (See figure 16)

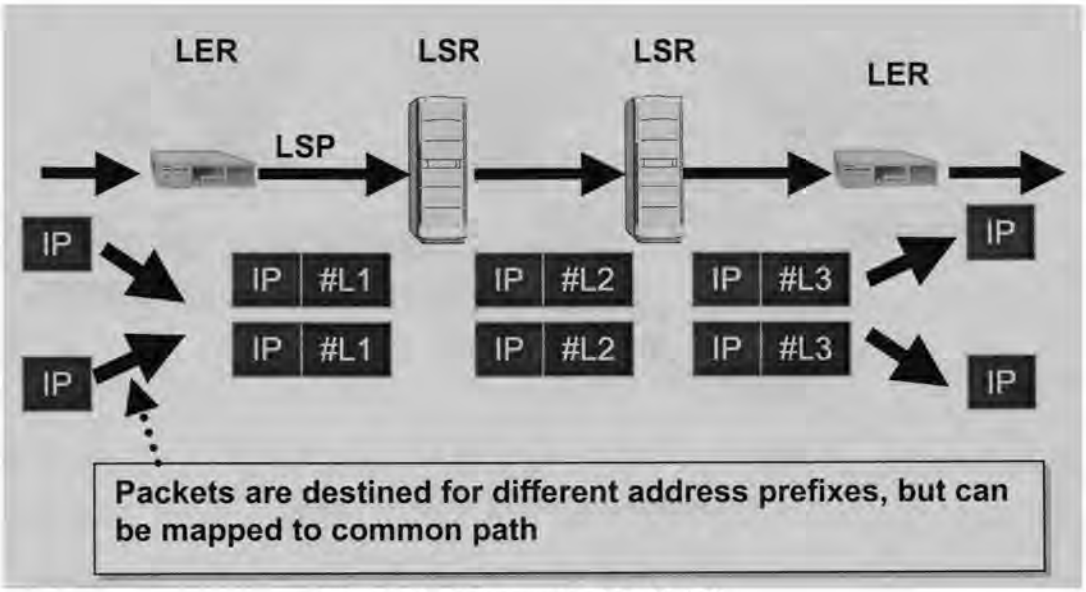

Figure 16 Assignments of FECs [33]

6 Label-Handling And Label-Forwarding

Figure 16 shows the label-handling and label-forwarding operation in more detail.

Each LSR maintains a forwarding table for each LSP passing through the LSR. When a labeled packet arrives, the LSR indexes the forwarding table to determine the next hop. For scalability, as was mentioned, labels have local significance only. Thus, the LSR removes the incoming label from the packet and attaches the matching outgoing label before forwarding the packet. The ingress-edge LSR determines the FEC for each incoming unlabeled packet and, on the basis of the FEC, assigns the packet to a particular LSP, attaches the corresponding label, and forwards the packet. [34]

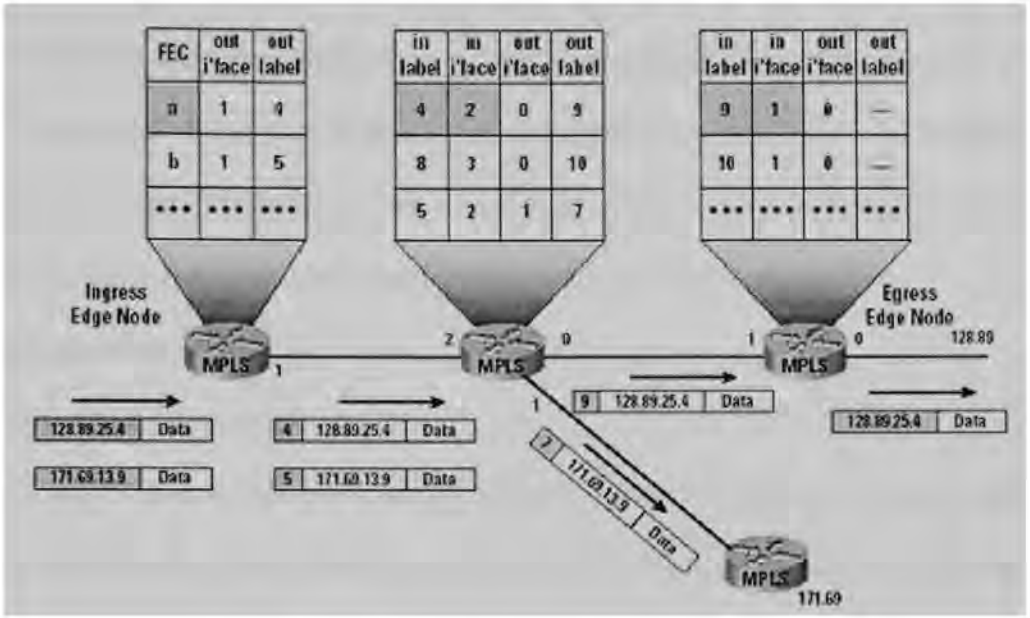

Figure 17 MPLS Packet Forwarding [34]

Label Stacking

One of the most powerful features of MPLS is *label stacking*. A labeled packet may carry many labels, organized as a last-in-first-out stack. Processing is always based on the top label. At any LSR, a label may be added to the stack (push operation) or removed from the stack (pop operation). Label stacking allows the aggregation of LSPs into a single LSP for a portion of the route through a network, creating a *tunnel*. At the beginning of the tunnel, an LSR assigns the same label to packets from a number of LSPs by pushing the label onto the stack of each packet. At the end of the tunnel, another LSR pops the top element from the label stack, revealing the inner label. This is similar to ATM, which has one level of stacking (virtual channels inside virtual paths), but MPLS supports unlimited stacking. Label stacking provides considerable flexibility. An enterprise could establish MPLS-enabled networks at various sites and establish numerous LSPs at each site. The enterprise could then use label stacking to aggregate

multiple flows of its own traffic before handing it to an access provider. The access provider could aggregate traffic from multiple enterprises before handing it to a larger service provider. Service providers could aggregate many LSPs into a relatively small number of tunnels between points of presence. Fewer tunnels mean smaller tables, making it easier for a provider to scale the network core. [35]

5.8 Applications of MPLS

5.8.1 Traffic Engineering

MPLS makes it easy to commit network resources in such a way as to balance the load in the face of a given demand and to commit to differential levels of support to meet various user traffic requirements. The ability to dynamically define routes, plan resource commitments on the basis of known demand, and optimize network utilization is referred to as traffic engineering.

With the basic IP mechanism, there is a primitive form of automated traffic engineering. Specifically, routing protocols such as OSPF enable routers to dynamically change the route to a given destination on a packet-by-packet basis to try to balance load. But such dynamic routing reacts in a very simple manner to congestion and does not provide a way to support QoS. All traffic between two endpoints follows the same route, which may be changed when congestion occurs. MPLS, on the other hand, is aware of not just individual packets, but flows of packets in which each flow has certain QoS requirements and a predictable traffic demand. With MPLS, it is possible to set up routes on the basis of these individual flows, with two different flows between the same endpoints perhaps following different routers. Further, when congestion threatens, MPLS paths can be rerouted intelligently. That is, instead of simply changing the route on a

packet-by-packet basis, with MPLS, the routes are changed on a flow-by-flow basis, taking advantage of the known traffic demands of each flow. Effective use of traffic engineering can substantially increase usable network capacity.[32]

5.8.2 Connection-Oriented QoS Support

Network managers and users require increasingly sophisticated QoS support for numerous reasons. The following are key requirements:

- Guarantee a fixed amount of capacity for specific applications, such as audio/video conference
- Control latency and jitter and ensure capacity for voice
- Provide very specific, guaranteed, and quantifiable service-level agreements, or traffic contracts
- Configure varying degrees of QoS for multiple network customers

A connectionless network, such as in IP-based internetwork, cannot provide truly firm QoS commitments. A Differentiated Service (DS) framework works in only a general way and upon aggregates of traffic from numerous sources. An Integrated Services (IS) framework, using the Resource Reservation Protocol (RSVP), has some of the flavor of a connection-oriented approach, but is nevertheless limited in terms of its flexibility and scalability. For services such as voice and video that require a network with high predictability, the DS and IS approaches, by themselves, may prove inadequate on a heavily loaded network. By contrast, a connection-oriented network has powerful traffic management and QoS capabilities. MPLS imposes a connection-oriented framework on an IP-based Internet and thus provides the foundation for sophisticated and reliable QoS traffic contracts. [32]

5.8.3 VPN Support

MPLS provides an efficient mechanism for supporting VPNs. With a VPN, the traffic of a given enterprise or group passes transparently through an internet in a way that effectively segregates that traffic from other packets on the internet, proving performance guarantees and security. [33]

5.8.4 Multiprotocol Support

MPLS, which can be used on many networking technologies, is an enhancement to the way a connectionless IP-based Internet is operated, requiring an upgrade to IP routers to support the MPLS features. MPLS enabled routers can coexist with ordinary IP routers, facilitating the introduction of evolution to MPLS schemes. MPLS is also designed to work in ATM and Frame Relay networks. Again, MPLS-enabled ATM switches and MPLS-enabled Frame Relay switches can be configured to coexist with ordinary switches. Furthermore, MPLS can be used in a pure IP-based Internet, a pure ATM network, a pure Frame Relay network, or an Internet that includes two or even all three technologies. This universal nature of MPLS should appeal to users who currently have mixed network technologies and seek ways to optimize resources and expand QoS support. [33]

6.0 A FRAMEWORK FOR INFORMATION TRANSFER IN IP NETWORKS

As already mentioned [section 3.1] to enable QoS, it requires the cooperation of all network layers from top-to-bottom, as well as every network element from end-to-end. In this thesis, it is argued that only one of the available technologies is not sufficient for reliable and efficient information transfer. It requires the overlap of different schemes like traffic engineering, quality of service and load balancing at different layers of network to achieve the goal of reliable data transfer in IP networks. Based on this argument a framework is proposed as described in later sections.

6.1 Network Layer Approach

Diffserv essentially provide differentiated performance degradation (or no degradation) for different traffic when there is network congestion. If congestion can be avoided, performance of all traffic will be good even without Diffserv. Traffic engineering can avoid congestion caused by uneven traffic distribution. In such cases traffic engineering is useful.

There is also a problem that Diffserv does not solve. That is, even though policing and shaping are done at the edge, uneven distribution of traffic in the network may still cause concentration of high priority traffic at some routers. This can excessively affect performance of low priority traffic, and can even cause performance degradation of high priority traffic at those routers. Traffic engineering must solve this problem. Therefore, traffic engineering is also necessary for reliable data transfer in the network.

6.2 Application Layer Approach

Load balancing uses multiple servers (geographically close to each other) and distributes

requests among them. The purpose is to increase service availability and to reduce the load of each server.

6.3 Framework

The framework can be summarized as follows.

Table 3 Proposed framework for data transfer in IP networks

Layer	Scheme	Mechanism	Purpose Of Mechanism
Application layer	Load balancing	Load balancing	Direct traffic away from congesting part of a network or server
Transport/Network layer	Quality of service	Differential services	Provide differentiated services for different classes of traffic, especially during network congestion.
Network Layer	Traffic engineering	Multiprotocol label switching	Avoid congestion in the network

6.3.1 Integration Of MPLS and Differentiated Services

Differential services based on a simple model where traffic entering a network is classified, policed, and possibly conditioned at the edges of network, and assigned to different behavior aggregates. Each behavior is identified by a single DS codepoint (DSCP). At the core of the network packets are fast forwarded according to the per hop behavior (PHB) associated with DSCP. By assigning traffic of different classes to different DSCPs, the Diffserv network provides different levels of QoS. [36]

QoS schemes try to provide differentiated services under overload condition. They differ little from best-effort service if the load is light. There are two reasons for network overloading or congestion: usage demand exceeding the available network resource, or uneven distribution of traffic. In the first case, we can either increase the network capacity or limit usage by QoS mechanisms. In the second case, load distribution and balancing may help. Traffic engineering arranges traffic flow so that congestion caused by uneven network utilization can be avoided. The allocated bandwidth for a particular traffic flow can be improved by the degradation of other users service level. It is possible with the right traffic engineering, to improve the service level for all users or a particular user. [37]

MPLS is a layer 3 switching technology proposed by the IETF that provides fast packet forwarding and traffic engineering. It integrates the label swapping forwarding paradigm with network layer routing. The combination of DiffServ and MPLS, presents a very attractive strategy to backbone network service providers with scalable QoS and traffic engineering capabilities using fast packet switching technologies. The motivations for DiffServ + MPLS include user demands for consistent QoS guarantees, efficient network resource requirements by network providers, and reliability and adaptation of node and link failures. MPLS + DiffServ provides solutions to these problems. 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. When MPLS is combined with Diffserv, they become powerful tools for QoS provision in IP backbone networks. [38]

6.3.2 Load Balancing Among Servers

No matter what sophisticated mechanisms are provided in the backbone, if the data servers fail to handle the requests, QoS cannot be delivered. For example, FreeFlow directs requests for embedded objects away from the original server to the dynamically picked servers. This speeds up the delivery of content-rich Web pages. However, because the initial requests are still sent to the original server, if the original server fails, the FreeFlow approach cannot helps. Besides, although the FreeFlow approach significantly reduces the load on the original servers, for busy Web sites, the number of requests on the original server is still every high. Therefore, other approaches are needed to improve the availability of the original servers and reduce their load. Such an approach is presented in this section. [39]

6.3.3 A load balancing approach

Traditionally, a single Web server is connected to a router that is connected to the Internet, as showed in Fig. 18. In order to reduce the load of the server, and improve availability of the Web site, a load balancer and multiple servers can be used. In this scenario, the servers are connected to the load balancer, which is in turn connected to the router, as showed in Fig. 19.

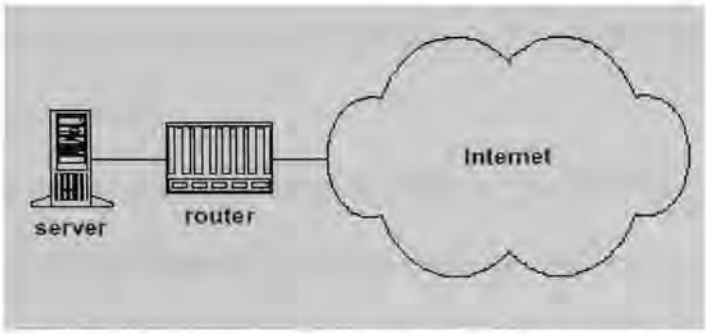

Figure 18 Server connection without a load balancer [39]

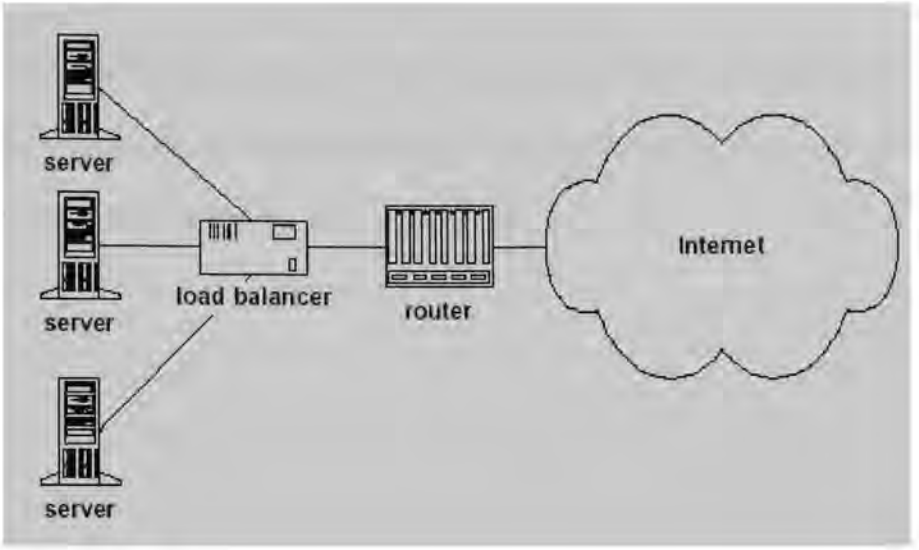

Figure 19 Server connection with a load balancer [39]

The load balancer intercepts each user request, and assigns it to a particular server based on some kind of criteria. The load-balancing scheme can be round robin or weighted round robin based on some statistics. The load balancer also performs application-layer health check for each server to make sure that it is not only reachable but also serving user requests properly (instead of returning "404 Objects Not Found"). The load balancer can also do packet filtering, source tracing and idle connection reaping. These are useful for protecting the servers from denial of service (DoS) attacks. Obviously, the load balancer can also be used to provide load balancing and protection for other application servers such as email servers.

6.3.4 Relationship Between Load Balancing, Diffserv And Traffic Engineering

Since load balancing is for reducing the load on each particular server, and increasing the availability of the servers, it is complementary to Diffserv and traffic engineering. Load

balancing techniques are concerned with the original servers while Diffserv and traffic engineering are concerned with the delivery of traffic in the backbone. No solution is complete without load balancing among the servers.

7.0 NETWORK MODELING

7.1 Need to model a Network

Modern dynamic systems including information systems, computer networks, hardware and software, and business processes behave in complex ways that are difficult to understand. This difficulty typically leads to many errors and inadequacies in developing such systems and planning for their use. Since the cost of delaying the correction of these errors grows rapidly with time, the value of early problem detection and correction is high. Simulation provides a practical way to detect such problems and allow early correction. Avoiding the use of simulation substantially increases the risk of failure. [40]

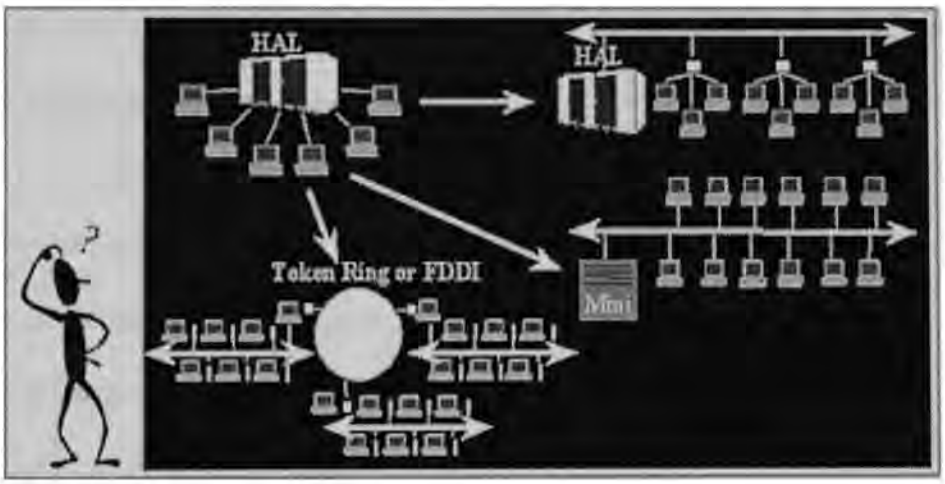

Figure 20 Modern Dynamic Systems Are Hard to Understand [40]

Modeling and simulation is a proactive approach of solving problems with complex dynamic systems those, which evolve in complex ways over time. Potential problems with complex dynamic systems involve performance, cost, correctness, completeness, consistency, reliability, security, time-to-market requirements, usability, and maintainability. Approaches to solving problems with complex dynamic systems include

simulation, measurement, benchmarking, rules of thumb, analytic modeling, and reactive problem solving. [40]

7.2 Different Approaches For Network Modeling And Their Limitations

7.2.1 Analytical Modeling.

Analytic modeling is a mathematical representation of computing systems for the purpose of performance analysis. Analytic modeling allows single hosts or single nodes to be easily and quickly modeled. The arrival distribution from a large number of terminals to a single system can be approximated in a way that is easily solved mathematically. Utilization and response times can be quickly calculated for single system cases. The accuracy of response time calculations depends on having a large number of users delivering a single, "average" transaction load to a single processor server. Applications and software systems are represented at a single level with analytic models. These models work well for average transactions, where detail representation of files, locks and data sizes is not required. Analytic modeling can be effective for non-complex Mainframe and very simple client/ server environments.

7.2.2 Limitations Of Analytical Modeling

Analytic modeling cannot accurately model complex environments but is effective for monitoring and performing trend analysis on individual systems. For complex environments analytic modeling is not enough to assure the performance of end-to-end systems.

- Systems with transient loads, single spikes or varying loads cannot be analyzed. Bulk arrivals and defections (users that give up after 30 seconds) cannot be estimated. Only a single steady state response time can be computed.

- System with concurrent processes or concurrently executing systems cannot be represented in analytic models.
- Systems with simultaneous resource usage, like CPU and memory or CPU and disk, cannot be accurately estimated with analytic models.
- When percentile calculations of a low priority function are needed such as 90% of a response time, they cannot be accurately estimated with analytic models.
- Performance of applications with blocking at database locks or semaphores cannot be accurately estimated using analytic modeling.

7.2.3 Simulation Modeling

Simulation modeling is the ability to create a model of the entire customer environment complete with computer systems, networks, routers, bridges, hubs, applications, databases, and users, and then perform simulations of the interactions between all of these components. [41]

7.2.4 Simulation Modeling Capabilities

Simulation modeling has the following capabilities:

- Simulation modeling is capable of scaling infinitely, maintaining accuracy appropriate for the complexity of the system.
- Simulation modeling can be equally applied to third party software or custom applications such as DBMS, MQSeries, CICS, MS SQL Server, Tuxedo, MS Exchange, etc.
- Simulation modelers can represent software and hardware at the appropriate level of detail, including system modifications. High-level capacity planning questions can be addressed as well as detailed low-level application performance questions.

- Simulation can represent the minute-by-minute workload fluctuations common in IT applications, including bulk arrivals and defections. Spikes and varying loads can be represented.
- The concurrency of the real world can be simulated within a single system (multiple threads and processors) and within multiple parallel systems.
- Application flow with simultaneous resource use can be simulated. Database locking and semaphore blocking can be easily represented in a simulation model.

A simulation model has no bounds; it can accurately model ANY software, hardware and network, no matter how complex. [41]

7.2.5 Load Testing With Simulation Modeling

Load testing is a technique used to stress test existing systems. The main purpose of load testing is to observe system behavior under expected user loading. Load testing tools create virtual users, agents that mimic real client applications, imitating the processes performed by *real* users. Load tests can be costly and time consuming. Scripts describing the virtual user actions and data entry need to be constructed and carefully tested. This takes time and effort by qualified personnel. Test hardware and software must be installed. Problems found during the testing are usually costly to fix and will inevitably delay system delivery dates.

7.2.6 Added value of simulation modeling

Load testing and simulation modeling are complementary technologies. Load testing will tell how well an existing system meets its performance expectations but it takes simulation to tell how to grow the system to meet future needs.

The most common benefits of using both technologies together are:

- Significant cost savings by not paying the licensing fees for increasing load test concurrent users.
- Ability to extrapolate load-testing data points with good confidence.
- Ability to investigate architectural options in very less time(approximately minutes)
- Ability to identify primary, secondary, and tertiary bottlenecks.
- Determines the impact on end user response time and system throughput in fixing bottlenecks.

7.3 Different Network Simulators Available

7.3.1 MIT's NETSIM

NETSIM is an **event driven simulator** for **packet-switched networks** designed at the MIT LCS Advanced Network Architecture group.

Use

All kinds of networks made of components that send messages to one another can be modeled.

The package

The simulator framework only provides the means to schedule events and to communicate with the user. It provides a simple X window graphic user interface (GUI) representation capability to display the topology of the network and the data of its operation. The GUI displays graphic objects and information objects (parameters and queues). Some component parameters may be changed during runtime by clicking in the relevant window, this affecting the simulation.

The package also comprises an event manager, I/O routines, various structural tools such as queues and lists used to build components, and a toolkit. The toolkit is a library of C functions that eases the manipulation of components and allows to create, save, load and display the network configuration, and to associate information windows with the components.

In order to run a simulation, the user has to write new components in C, to modify a few files and to compile and link them. He then creates the simulation with the toolkit.

Implementation of the simulator

Components are things like links, hosts, switches and network connections. Every component has a class (e.g. link) and a type and has to handle a set of events. They are stored in a doubly linked list while Netsim is running and are represented by a data structure and an action routine. The action routine is called whenever an event is sent to the component. Events handled by a component are of three types: command, regular and private. Private events are events a component sends to itself, regular events are those, which are concerned with the actual running of the simulation, and commands are those, which perform housekeeping actions. The action routine has to respond to a predefined set of events besides user-defined ones since the management of components (creation, deletion, etc,.) has to be coded by the user in the action routine. The action of components is therefore entirely defined by the user and written in C and is not affected by the framework of the simulator.

Parameters are information that characterizes a component and need to be displayed on the screen or saved in a file (inputs given by the user and outputs produced by the simulator). Input parameters may be specified at the time of component creation or

may be modified during run time. They are stored in a singly linked list and may as well directly be referenced by a pointer in the data structure of the component, this avoiding the action routing to search through the list.

The simulator implements static routing by the use of components of connection. type A route is a list of adjacent components, starting and ending with a connection component. Components available in the current version are an Ethernet link, a point-to-point link, a switch (switches packets between several links), a host (about the same as a switch), and Purdue's implementation of TCP, data supplier and consumer from TCP, a simple Poisson traffic source and a packet sink.

Since the simulator was designed for packet-switched networks, it includes a packet data type, but its data structure is not constrained to any particular format. The user may therefore implement new data types like cells or any other convention to communicate between components, but components have to be rewritten in order to correspond to the new data structure. There are two implementations of list. One kind allocates a chunk of memory where to store a pointer once a new element is added to the list. Adding and removing an event is slower, but an element can be placed on several lists. The other requires that, the element being placed on the list contains the pointer to link it into the list. No extra memory is needed to link it into the list, but it can only be on one list at a time.

Simulation Engine

It is a single process written in C. It merely provides the means to schedule events. Events may include a packet and are inserted in an event queue and ordered according to the fire time. The event queue is therefore neither a FIFO nor a LIFO. Events that should

have occurred in the past (i.e. with a fire time smaller than the clock) are not allowed to be inserted in the queue. There is no particular order for concurrent events scheduled to fire at the same time. The scheduler extracts the event at the head of the queue, it sets the logical time of the clock to that of the fire time of the event if it is greater and calls the action routine of the component related to the event. The action routing produces other events and defines the time at which they should fire.

Time is maintained in an unsigned 32-bit value. The simulator can then run for almost 12 hours of simulated time.

7.3.2 REAL 5.0

REAL (REalistic And Large) is a network simulator written at Cornell University

Use

REAL is intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched data networks (namely TCP/IP).

The package

REAL provides 30 modules written in C that emulate flow-control protocols such as TCP, and 5 scheduling disciplines such as FIFO, Fair Queuing, DEC congestion avoidance and Hierarchical Round Robin. The description of the network topology, protocols workload and control parameters are transmitted to the server using a simple ascii representation called *NetLanguage* where the network is modeled as a graph.

Implementation of the simulator

The REAL code has been rewritten to make it less general, cleaner and faster. REAL is implemented as a client-server program. The code is freely available to anyone willing to modify and improve it.

Node functions implement computation at each node in the network whereas queue management and routing functions manage buffers in nodes and packet switching. *Routing is static* and is based on Dijkstras's shortest path algorithm. A node could be a source, a router or a sink. Source nodes implement TCP-like transport layer functionality. Routers implement the scheduling disciplines, while the sinks are universal receivers that only acknowledge packets.

REAL sends out a timer packet from a source back to itself to return after some specified time, but timers cannot be reset using this method.

7.3.3 NS version 2.0

NS is an *object-oriented discrete-event* simulator for networking research based on REAL.

Use

For the time being, NS is well suited for packets switched networks and is used mostly for small-scale simulations of queuing algorithms, transport protocol congestion control, and some multicast related work. It provides support for various implementations of TCP, routing, multicast protocols, link layer, MAC, ... It currently has memory limitations in the face of large simulations.

The package

The current non-beta version of NS implements key protocol modules for unicast and multicast routing, reservation, transport and session protocols.

NS is written in C++. The package provides a compiled class hierarchy of objects written in C++ and an interpreted class hierarchy of objects written in OTcl (MIT's object extension to Tcl - Tool Command Language), which are closely related to the compiled

ones. The user creates new objects through the OTcl interpreter. A corresponding object in the compiled hierarchy closely mirrors new objects.

Tcl procedures are used to provide flexible and powerful control over the simulation (start and stop events, network failure, statistic gathering and network configuration). The Tcl interpreter has been extended (OTcl) with commands to create the network topology of links and nodes and the agents associated with nodes.

Implementation of the simulator

The simulation is configured, controlled and operated through the use of interfaces provided by the OTcl class Simulator. The class provides procedure to create and manage the topology, to initialize the packet format and to choose the scheduler. It stores the references to each element of the topology, internally. The user creates the topology using OTcl through the use of the standalone classes, node and link that provide a few simple primitives. The function of a node is to receive a packet, examine it and map it to the relevant outgoing interfaces. A node is composed of simpler classifier objects. Each classifier in a node performs a particular function, looking at a specific portion of the packet and forwarding it to the next classifier.

Links are characterized in terms of delay and bandwidth. They are built from a sequence of connectors' objects. The data structure representing a link is composed by a queue of connector objects, its head, the type of link, the ttl (time to live), and an object that processes link drops. Connectors receive a packet, perform a function, and send the packet to the next connector or to the drop object. Various kinds of links are supported, e.g. point-to-point, broadcast, wireless. The output buffers attached to a link in a ``real'' router in a network are modeled by queues. In NS, queues are considered as part of a link.

NS allows the simulation of various queuing and packet scheduling disciplines. C++ classes include drop-tail (FIFO) queuing, Random Early Detection (RED) buffer management, CBQ (priority and round-robin), Weighted Fair Queuing (WFQ), Stochastic Fair Queuing (SFQ) and Deficit Round-Robin (DRR).

Traffic generation in NS looks rather basic in the current implementation. For the purpose of TCP, only FTP and Telnet traffic can be generated; otherwise, NS provides an exponential on/off distribution and it allows generating traffic according to a trace file.
In order to analyze results, NS provides classes to trace each individual packet as it arrives, departs or is dropped, and to record any kind of counts, applied on all packets or a per-flow basis. The trace can be set or unset as desired by the user. The user has to specify the routing strategy (static, dynamic) and the protocol to be used. This is done by a procedure in the class simulator. Supported routing features include asymmetric routing, multipath routing, Distance Vector algorithm, multicast routing (PIM, etc.,).

Simulation Engine

The simulation engine is extensible, configurable and programmable. The current implementation is single-threaded (only one event in execution at any given time). It does not support partial execution of events.

Events are described by a firing time and a handler function. The type of event scheduler used to drive the simulation can be chosen among the four presently available: a simple linked-list (default), heap, calendar queue, and a special type called real-time. Each one is implemented using a different data structure. The simple linked-list scheduler provides a list of events kept in time-order, from the earliest to the latest. This requires scanning the list to find the appropriate entry upon insertion or deletion. The entry at the

head is always executed first. Entries with the same simulated time are extracted according to their order in the list.

The heap scheduler code is borrowed from the MARS-2.0 simulator (that itself borrowed the code from MIT'S NETSIM). This implementation is superior to the linked list scheduler when the number of events is large. Insertion and deletion times are in O (logn) for n events. The real-time scheduler is still under development and is currently a subclass of the list scheduler. It is well suited when events arrive with a relatively slow rate. Execution of events should occur in real time.

Forecoming changes to NS

NS has scaling constraints in terms of storage requirements of the routing tables: each node maintains a route to all other nodes in the network, resulting in aggregate memory. This grows with the number of nodes in the network. VINT proposes to replace NS's flat addressing conventions by implementing efficient hierarchical routing table look-ups in the nodes objects. NS represents an underlying transmission link and its corresponding scheduling and queuing algorithms in a single object, but the VINT simulation framework will require their separation in order to flexibly combine different scheduling algorithms with different underlying link technologies. In the current implementation of NS, each module that implements a scheduling discipline need be changed when adding modules to support a new link type.

Another modification is the performance improvement to the event scheduler. The linear search insertion algorithm should be replaced with a heap or a calendar queue. Coexistent scheduling algorithms can be derived from base class abstraction and are easily implemented in NS due to its C++ implementation.

Large Scale Issues

Since the simulation is likely to generate enormous amounts of data, visualization will have to play a key role and will be added to all phases of the simulation process, from inputs to outputs. A tool that generates random topologies according to user-specified parameters will help in describing very large topologies; a tool for the manipulation of data packets at the bit level allows detailed tests. For the study of protocol interaction and behavior at significantly larger scale, the simulator will provide two levels of abstraction.

The detailed level simulator is the one currently built. It allows a fine abstraction of the distinct modules of the simulation and will later include an emulation interface that will allow incorporating a real network node as a component of the simulation. The session level simulator will give a coarse-grain abstraction. Data packets will be represented by flows instead of individual packets. This will reduce the number of events and state required, at the cost of lost detail in the simulation. As an instance, instead of tracing each packet through each router and link, the simulator only calculates the time for that packet to be received by the sink according to the path used.

7.3.4 OPNET

OPNET provides a comprehensive development environment supporting the modeling of communication networks and distributed systems. Both behavior and performance of modeled systems can be analyzed by performing discrete event simulations. The OPNET environment incorporates tools for all phases of a study, including model design, simulation, data collection, and data analysis.

Simulation Package Overview

The Sim *(Simulation)* package is a collection of kernel procedures concerned with providing simulation-wide services. Many protocols and algorithms have the need to determine time values, so that they can measure time differences and schedule future events. For instance, self interrupts are scheduled for absolute times, which are usually computed as the sum of fixed intervals and the current simulation time. The KP ***op_sim_time()*** returns the current value of the global simulation clock.

There are several ways in which a simulation can terminate. Automatic termination, which stops a simulation when the global simulation clock reaches the specified value of the environment attribute "duration", is the most common strategy. Simulations can also be terminated based on dynamic conditions. When an executing process determines that the simulation stopping condition has been attained, it can invoke the KP o***p_sim_end()***, which immediately (but cleanly) terminates the simulation. As simulations execute, they can communicate information to users using a variety of mechanisms, including: (1) standard screen I/O features of the host operating system; (2) standard file I/O features of the host operating system; (3) calling procedures from graphics libraries. When short textual messages need to be printed to indicate the progress of a simulation, the most convenient mechanism is the KP ***op_sim_message()***, which prints a three line message onto the standard output de-vice.

Developing simulations usually entails a debugging phase, in which special tools are used to confirm the proper execution of process logic. These tools include the debug mode of the simulation and special print statements, which provide information about dynamic execution. The KP ***op_sim_debug()*** combines both tools by permitting debug

print statements to be permanently embedded into simulations, they will only print when the simulation is executing under the debug mode.

Key System Features

OPNET is a vast software package with an extensive set of features designed to support general network modeling and to provide specific support for particular types of network simulation projects.

•**Object orientation:** Systems specified in OPNET consist of objects, each with configurable sets of attributes. Objects belong to "classes" which provide them with their characteristics in terms of behavior and capability. Definition of new classes is supported in order to address as wide a scope of systems as possible. Classes can also be derived from other classes, or "specialized" in order to provide more specific support for particular applications.

• **Specialized in communication networks and information systems:** OPNET provides many constructs relating to communications and information processing, providing high leverage for modeling of networks and distributed systems.

•**Hierarchical models:** OPNET models are hierarchical, naturally paralleling the structure of actual communication networks.

•**Graphical specification:** Wherever possible, models are entered via graphical editors. These editors provide an intuitive mapping from the modeled system to the OPNET model specification.

• **Flexibility to develop detailed custom models:** OPNET provides a flexible, high-level programming language with extensive support for communications and distributed systems.

This environment allows realistic modeling of all communications protocols, algorithms, and transmission technologies.

•**Automatic generation of simulations:** Model specifications are automatically compiled into executable, efficient, discrete-event simulations implemented in the C programming language. Advanced simulation construction and configuration techniques minimize compilation requirements.

•**Application-specific statistics:** OPNET provides numerous built-in performance statistics that can be automatically collected during simulations. In addition, modelers can augment this set with new application-specific statistics that are computed by user-defined processes.

• **Integrated post-simulation analysis tools:** Performance evaluation, and trade-off analysis require large volumes of simulation results to be interpreted. OPNET includes a sophisticated tool for graphical presentation and processing of simulation output.

•**Interactive analysis:** All OPNET simulations automatically incorporate support for analysis via a sophisticated interactive "debugger".

• **Animation:** Simulation runs can be configured to automatically generate animations of the modeled system at various levels of detail and can include animation of statistics as they change over time. Extensive support for developing customized animations is also provided.

•**Application program interface (API):** As an alternative to graphical specification, OPNET models and data files may be specified via a programmatic interface. This is useful for automatic generation of models.

8.0 SIMULATION AND ANALYSIS OF MPLS+DIFFSERV

The central idea behind the architecture proposed in this thesis and described in the previous chapters is the integration of MPLS and DiffServ in the backbone networks. In this chapter, performance analysis of DiffServ networks, MPLS networks, and integrated MPLS and DiffServ networks are presented using simulation experiments in OPNET. This is done to demonstrate the effectiveness of MPLS and DiffServ integration under specific circumstances.

8.1 Simulation Aims

The main function of this simulation is to quantitatively compare performance achieved by network traffic before and after MPLS+ Diffserv, which is done in three different network configurations:

- Neither MPLS nor DiffServ is enabled,
- Only MPLS is enabled,
- Both MPLS and DiffServ are enabled.

This comparison aims to support the most important claim for the proposed architecture, namely that there is benefit for the current Internet traffic provided by the co-existence of MPLS and DiffServ in backbone networks. Link throughput and queuing delay happen to be the most commonly considered parameters in a network. If the comparison shows better link throughputs and low link delays for the case where both MPLS and DiffServ are enabled, the claim is demonstrated.

8.2 Network Topology

Two servers Yahoo.com and Google.com are trace routed from a workstation at High speed Tele Networking Lab at Florida International University (FIU) as shown below.

C:\>tracert yahoo.com

Tracing route to yahoo.com [66.218.71.198]

over a maximum of 30 hops:

1 <10 ms <10 ms <10 ms 131.94.163.1

2 <10 ms <10 ms <10 ms 131.94.192.49

3 <10 ms <10 ms <10 ms 131.94.205.2

4 16 ms <10 ms <10 ms up-br1-fe100.net.fiu.edu [131.94.191.2]

5 250 ms 266 ms 250 ms 172.25.97.209

6 156 ms 172 ms 187 ms 205.152.144.139

7 141 ms 187 ms 203 ms 65.83.237.8

8 157 ms 140 ms 125 ms 65.83.236.18

9 219 ms 219 ms 250 ms POS1-0.hsipaccess2.Miami1.Level3.net [64.156.8.21]

10 203 ms 234 ms 235 ms ge-6-1-1.mp2.Miami1.level3.net [64.159.1.101]

11 297 ms 281 ms 297 ms so-2-0-0.mp1.SanJose1.Level3.net [209.247.9.114]

12 * 234 ms * gige9-1.ipcolo3.SanJose1.Level3.net [64.159.2.73]

13 * * 359 ms unknown.Level3.net [64.152.69.30]

14 344 ms 297 ms 265 ms w1.rc.vip.scd.yahoo.com [66.218.71.198]

Trace complete.

C:\>Tracert google.com

Tracing route to google.com [216.239.51.100]

over a maximum of 30 hops:

1 <10 ms <10 ms <10 ms 131.94.163.1

2 <10 ms 15 ms <10 ms 131.94.192.49

3 <10 ms <10 ms <10 ms 131.94.205.2

4 <10 ms <10 ms <10 ms up-br1-fe100.net.fiu.edu [131.94.191.2]

5 250 ms 234 ms 219 ms 172.25.97.209

6 187 ms 188 ms 203 ms 205.152.144.139

7 234 ms 219 ms 172 ms 65.83.237.8

8 171 ms 204 ms 218 ms 65.83.236.16

9 172 ms 140 ms 156 ms 0.so-0-0-0.gw8.mia4.alter.net [65.208.86.153]

10 140 ms 157 ms * 0.so-1-3-0.XL1.MIA4.ALTER.NET [152.63.84.54]

11 187 ms 203 ms 172 ms 0.so-4-0-0.XL1.ATL5.ALTER.NET [152.63.86.190]

12 265 ms 250 ms 250 ms POS6-0.BR2.ATL5.ALTER.NET [152.63.82.197]

13 * 187 ms * atl-brdr-03.inet.qwest.net [205.171.4.241]

14 * 203 ms 156 ms atl-core-02.inet.qwest.net [205.171.21.101]

15 188 ms 172 ms 187 ms dca-core-02.inet.qwest.net [205.171.8.154]

16 157 ms 187 ms 219 ms dca-core-03.inet.qwest.net [205.171.9.50]

17 203 ms 219 ms * dcx-edge-01.inet.qwest.net [205.171.9.162]

18 187 ms 188 ms 187 ms 65.118.39.170

19 172 ms 187 ms 188 ms 216.239.47.126

20 203 ms 172 ms 187 ms 216.239.47.102

21 * * 187 ms 216.239.51.100

Trace complete.

The above results for routing the trace to different servers shows that all traffic from FIU server to them are following the same path for half the number of hops. Though the exact amount of other traffic in that path can't be assessed, it shows that if bulk traffic is

suddenly to be transferred, then there is always congestion in that path. So a network based on such a situation is selected as the baseline network for this simulation experiment. OPNET Modeler 8.0 is used for the simulation analysis using OPNET's MPLS models available in the OPNET library.

8.3 Baseline network: Neither MPLS nor Diffserv enabled

The following network is the baseline network for this simulation experiment.

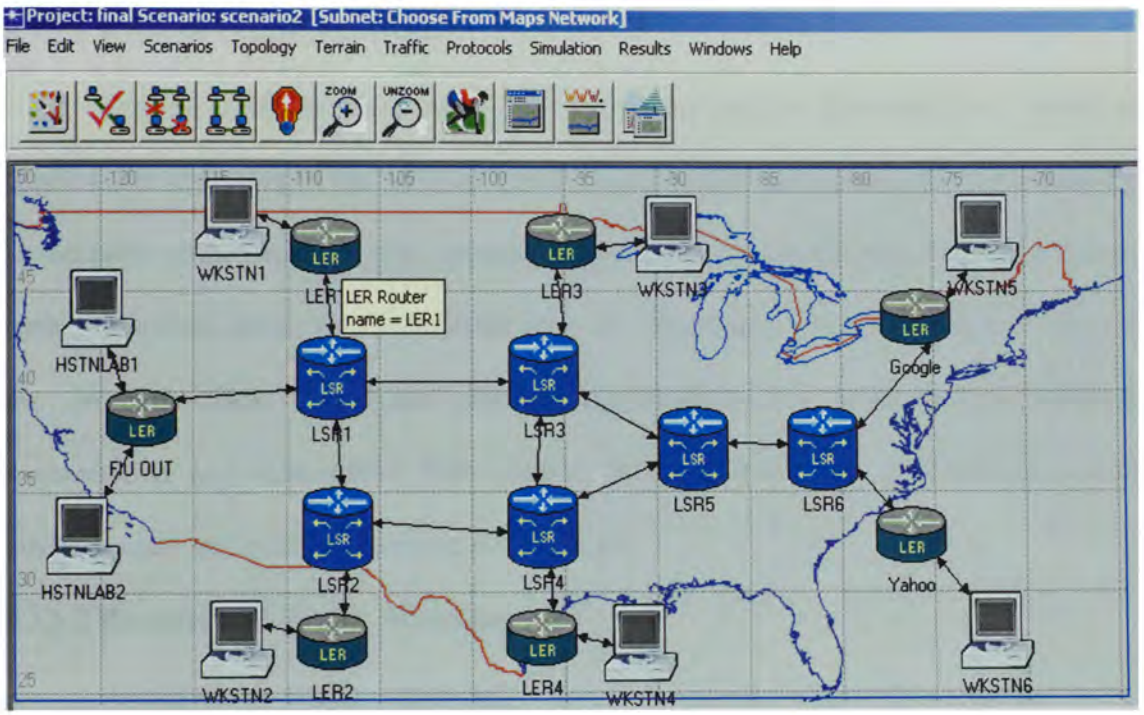

Figure 21 Baseline network topology

All routers (LERs and LSRs) in the above topology are MPLS enabled. The have been configured such that their label mapping and switching algorithms are enabled only when LSPs are defined in this network. With no LSPs defined, these routers will use the routes advertised by the dynamic routing protocol running on their interfaces (RIP in this case).

HSTNLAB1 and HSTNLAB2 represent the systems at FIU and are used to create traffic going out of FIU.FIUOUT represents the router that routes all the internet traffic going out of FIU. The edge routers Google and Yahoo represent the routers connecting the outside network to respective servers.

8.3.1 Traffic Pattern

In OPNET, there are two types of traffic modeling methods. They are (1) Explicit traffic modeling and (2) Background traffic modeling.

8.3.1.1 Explicit Traffic Modeling

Explicit traffic is composed of the individual packets generated by a subset of applications and users in the network. These are the typical or critical users for whom the IT decision maker seeks highly accurate information. Using discrete event simulation, each application transaction is modeled between clients and servers, and at each layer of the protocol stack, important protocol mechanisms are implemented, such as: segmentation and reassembly; flow control; timeouts, back offs and retransmissions; media access; and quality of service prioritization.

8.3.1.2 Background Traffic Modeling

Background traffic represents high-level information concerning end-to-end traffic volume between pairs of network devices. Typically, background traffic is gathered over long periods of time, ranging from tens of minutes to multiple days. Background traffic does not keep track of individual packets, but merely counts packets and bytes to characterize the amount of traffic that is flowing in the network.

The selected routing protocol dictates the path taken from one edge of the network to another. Both the explicit traffic as well any background traffic flow will be

subject to the same set of routing decisions. In this case, considering the volume of the traffic to be created background traffic modeling is used. Background traffic is characterized as below in table 4. Figure 22 shows the Traffic conversation pair browser window with HSTNLAB1 as the source.

Table 4 Traffic Pairs in baseline network

Source	Destination
HSTNLAB1	WKSTN 3
	WKSTN 5
	WKSTN 6
HSTNLAB2	WKSTN 4
	WKSTN 5
	WKSTN 6
WKSTN 2	WKSTN 5
	WKSTN 6
WKSTN 3	WKSTN 2
WKSTN 4	WKSTN 1

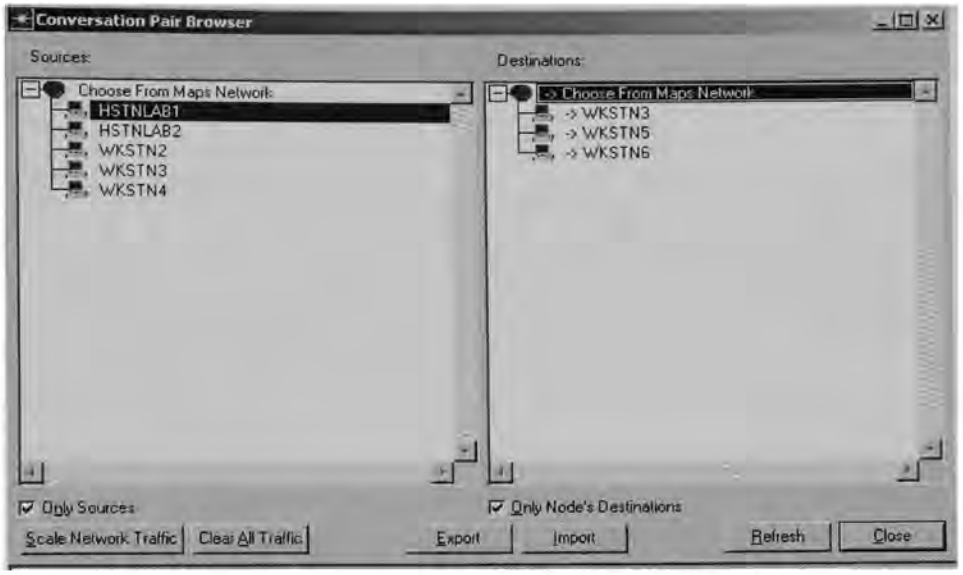

Figure 22 Traffic Conversation pair with HSTNLAB1 as the source

8.3.2 Links

In the above scenario the links connecting the LERs and the workstations are referred to as edge links and the links connecting the LERs and LSRs are referred to as core links. In the base line network, the edge links used are PPP-DS3 links and the core links are OC3.

8.3.3 Link throughput And Utilization

The baseline network is simulated for four hours duration with RIP as the routing protocol. The following graph (figure 23) shows the throughput of different links in the network. Very high link throughputs on links such as links between LSR1-LSR3 and LSR3-LSR5 show that they are over utilized. Very low throughput on some links such as links between LSR2-LSR4 and LSR4-LSR5 show that they are underutilized. This mismatch in the utilization of the links is due to the routing protocol used and causes network congestion The graph in figure 24 shows the percentage utilization of the above mentioned links. The x-axis shows the simulation time in minutes and y-axis shows the respective parameters.

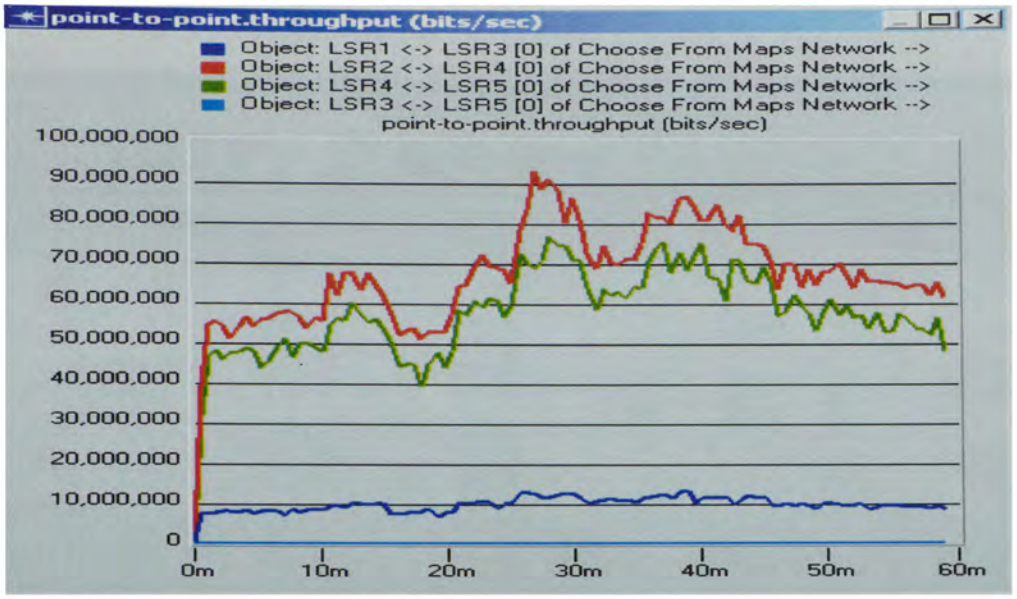

Figure 23 Link Throughputs in baseline network

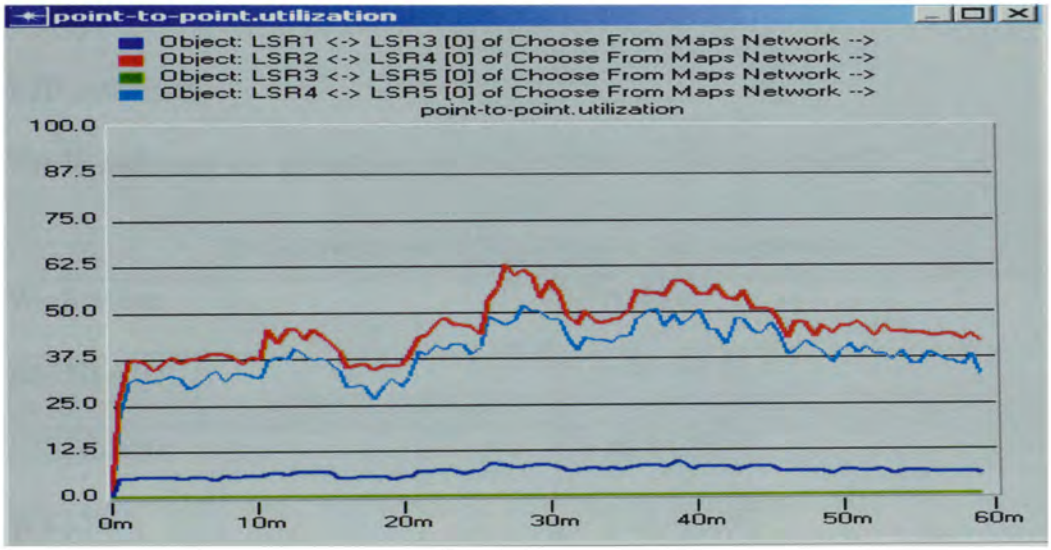

Figure 24: Link utilizations in baseline network

8.4 MPLS Deployed network: MPLS enabled, Diffserv disabled

In the baseline network some links are overloaded and some are not at all utilized. In this configuration, MPLS is deployed to balance the load on the links and improve the percentage of link utilization.

The network diagram after defining the MPLS paths is shown in figure 25.

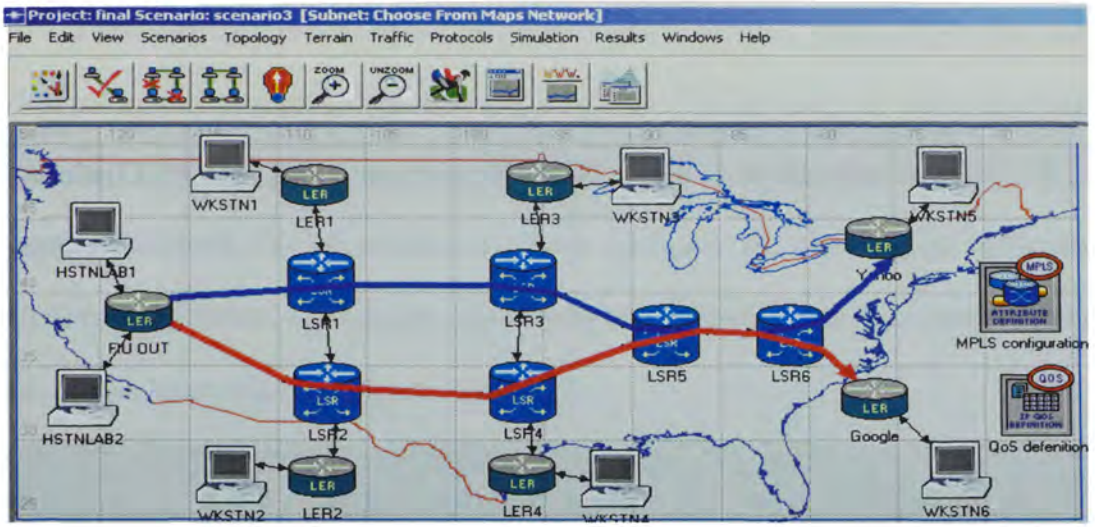

Figure 25 Network Diagram after deploying MPLS

Following are network characteristics after deploying MPLS.

8.4.1 IP addresses

The IP addresses are assigned to the workstations as shown in table 5

Table 5 Assignment of IP addresses to the workstations

Workstation	IP Address
HSTNLAB1	131.94.178.93
HSTNLAB2	131.94.178.94
WKSTN 1	172.25.97.209
WKSTN 2	205.152.144.203
WKSTN 3	172.25.98.210
WKSTN 4	205.152.145.204
WKSTN 5	64.58.79.230
WKSTN 6	216.239.51.100

8.4.2 Label Switched Paths

Label switched path is a path that MPLS sets to transmit packets upon the labels. LSP is set up and provisioned prior to actual data forwarding using the label distribution protocols (LDP). Two paths, one from FIUOUT to Yahoo and the other from FIUOUT to Google are defined. Defining these two paths facilitates the diversion of traffic from FIUOUT to the Yahoo and Google servers into two different paths. The path details for these two paths are shown in figures 26 and 27.

Figure 26 Path details for the path FIUOUT-Google

Figure 27 Path Details for FIUOUT-Yahoo

8.4.3 FEC

Forward equivalence class is a representation of group of packets that share the same requirements for their transport. FEC are based on service requirements for a given set of packets. Two FECs, in corresponding to the two-switched paths are created based upon their destination addresses as shown in table 6.

Table 6 Forwarding Equivalence Classes defined in the network

FEC	ToS	Protocol	Source Address	Destination address	Source Port	Destination port
Yahoo	Unassigned	Unassigned	Unassigned	64.58.79.230	Unassigned	Unassigned
Google	Unassigned	Unassigned	Unassigned	216.239.51.100	Unassigned	Unassigned

By defining these classes, the traffic destined to yahoo server follows the path FIUOUT-Yahoo and packets destined to Google server follows the path FIUOUT-Google. This happens only when the classes are appropriately bound to respective paths.

8.4.4 Traffic Trunk

Differential service support is not yet added to the network. So the traffic trunk is a default trunk for all classes and is as shown in the figure 28.

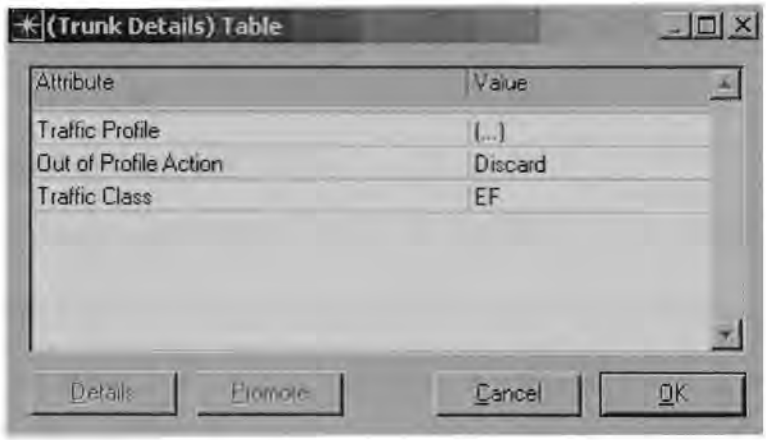

Figure 28 Default traffic trunk

8.4.5 LER Configuration

Once LSPs, FECs and traffic trunks are defined in the network, the traffic engineering (TE) bindings that govern which packets are sent to which LSPs are to be created. In this scenario, FIUOUT should be configured to bind the traffic destined to yahoo server to FIUOUT-Yahoo and traffic destined to Google server to the path FIUOUT-Google. The traffic-engineering configuration for FIUOUT can be seen in table 7.

Table 7 Traffic engineering configuration for FIUOUT

Interface In	FEC	Traffic Trunk	Primary LSP
0(HSTNLAB1) 1(HSTNLAB2)	Yahoo	Default Trunk	FIUOUT-Yahoo
0(HSTNLAB1) 1(HSTNLAB2)	Google	Default Trunk	FIUOUT-Google

From the table it can be seen that, the two interfaces HSTNLAB1 and HSTNLAB 2 are bound to same FEC, as there is no differential service support.

8.4.6 Link throughput and Utilization

The following graph (figure 29) shows the throughputs of different links after deploying MPLS. It shows that after defining separate paths for different traffic flows the load on different links in the network is balanced .The utilization graph in figure 30 shows that the links, which were previously not used at all, are utilized to some extent and there is a flow of packets through them.

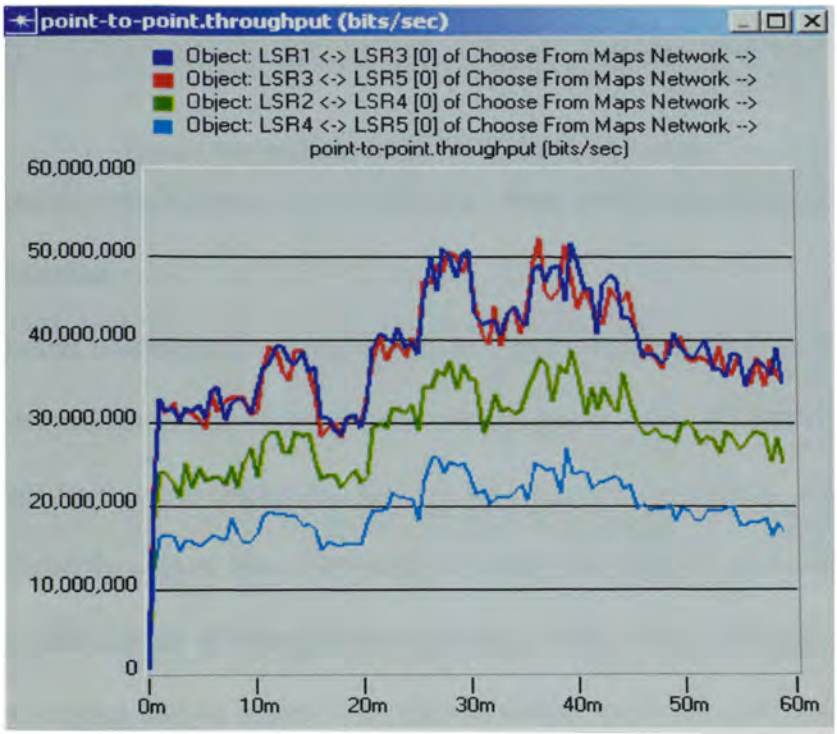

Figure 29 Throughput of links after deploying MPLS

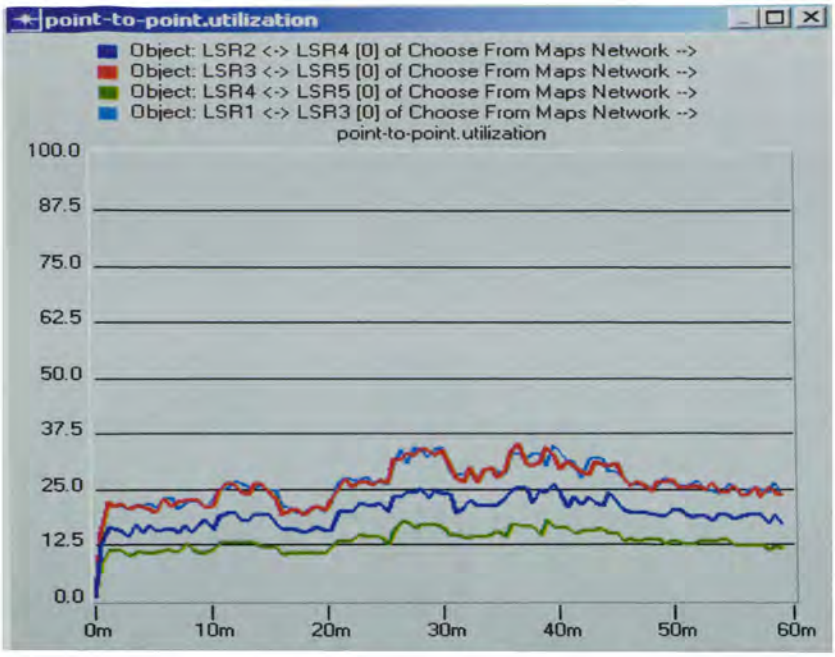

Figure 30 Utilization of links after deploying MPLS

8.5 Differential Service Support for A Network : Both MPLS and Diffserv enabled.

8.5.1 Configuration

This configuration illustrates the use of MPLS to classify different flows in the LSP, and serve them with different QoS levels. In this configuration, a file of 150MB is uploaded to yahoo server from two workstations HSTNLAB1 and HSTNLAB2.As an MPLS path is already defined for a flow from FIUOUT to Yahoo, this kind of traffic is diverted to that particular path instead of being routed according to the routing protocol (RIP, in this case).Diffserv support can be added to assign the traffic from different workstations to different Diffserv codes and treat them with different priority levels. The network disgram after adding differentiated services is as shown in figure 31.

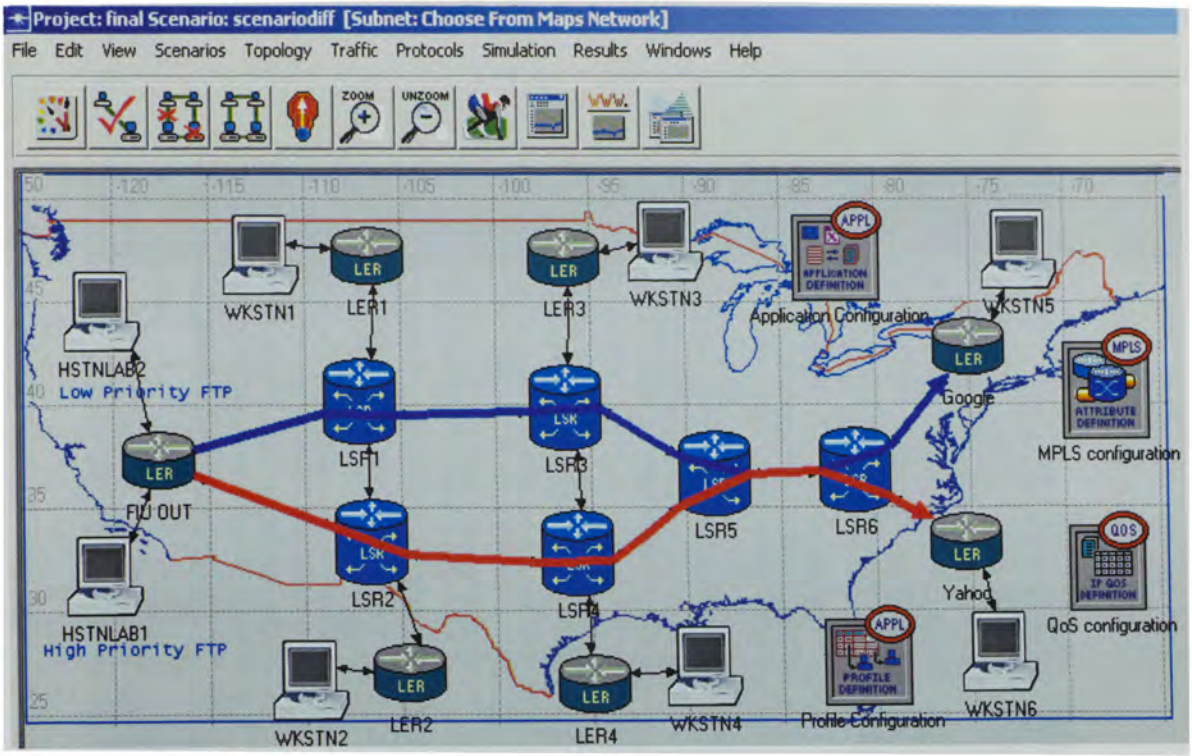

Figure 31 Network diagram with differentiated services

Two trunk profiles have been specified with different codes. Traffic flow from HSTNLAB 1 is given higher priority. Routers are configured to perform weighted Fair Queuing (WFQ).

The graph in figure 32 shows the client FTP traffic sent from both stations. The overlap of both traces show that same size of file is transferred from both stations.

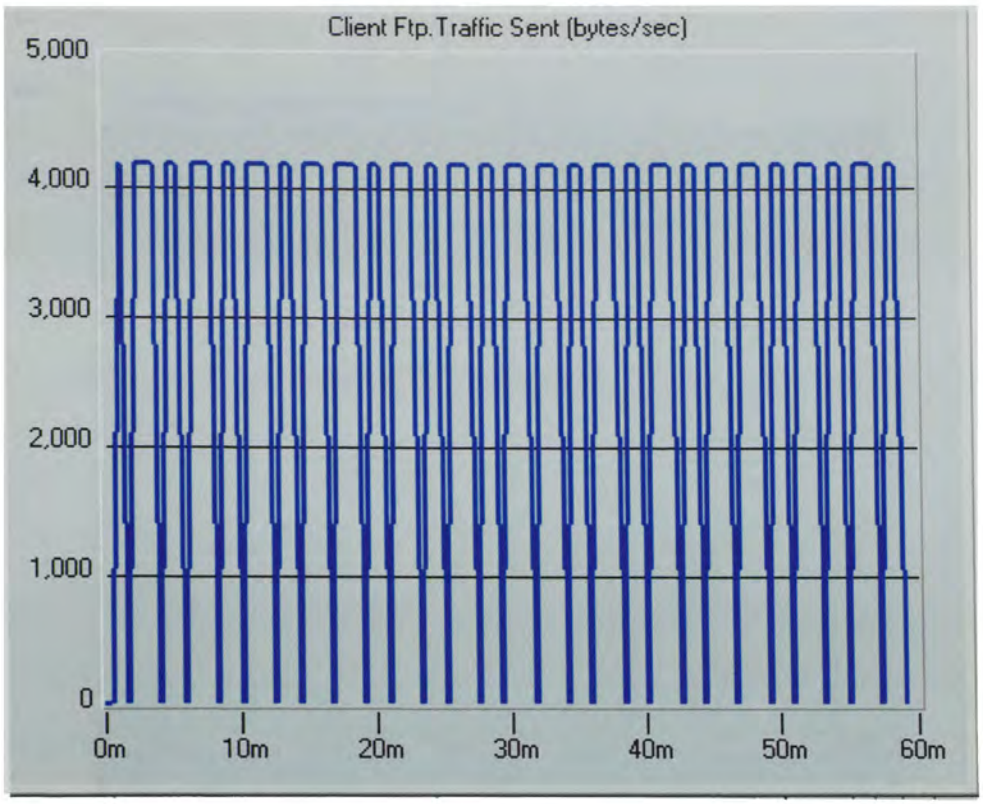

Figure 32 Client Ftp Traffic

8.5.2 Upload response time and Queuing delay

Upload response time and queuing delay are the most considered parameters when prioritizing the traffic. The following graph gives these traces for both workstations and it is shown that flows with higher diffserv code has less upload response time.

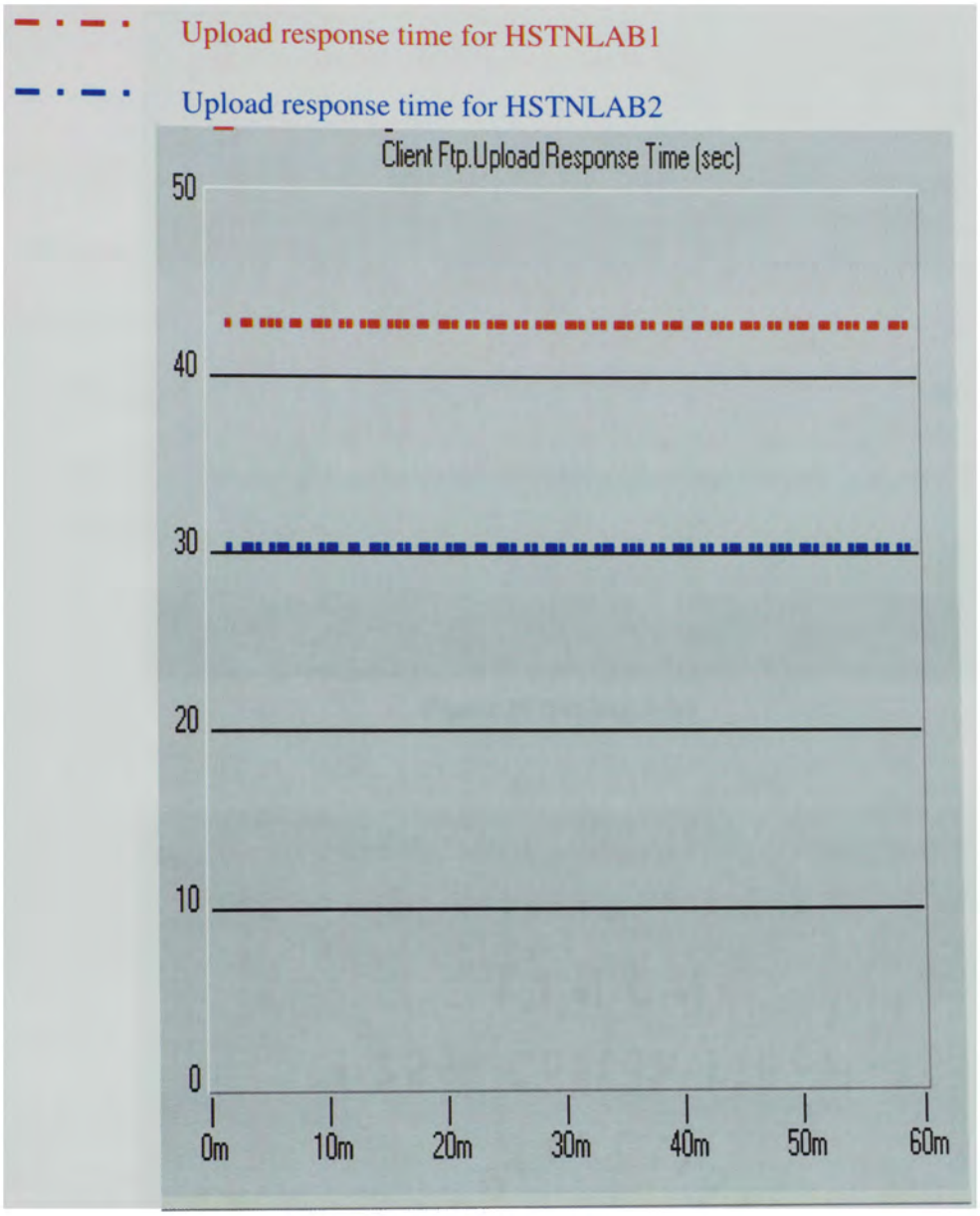

Figure 33 Upload response time

Even though two similar traffic flows are flowing to the same destination but as different traffic trunk profiles have been assigned to these flows, traffic with high priority has less queuing delay and less buffer usage. This can be made evident from the graphs in figures 34 and 35.

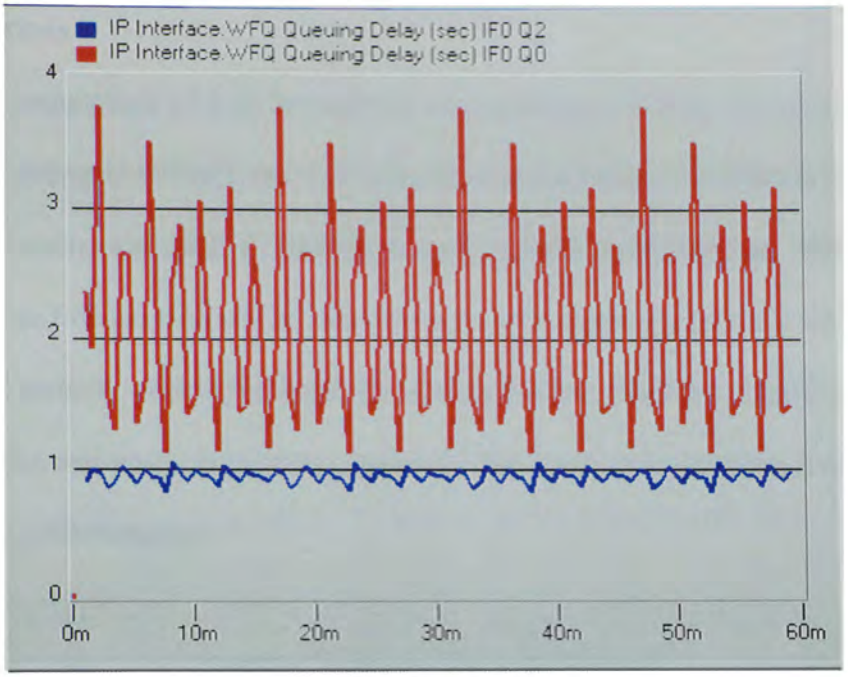

Figure 34 Queuing delay

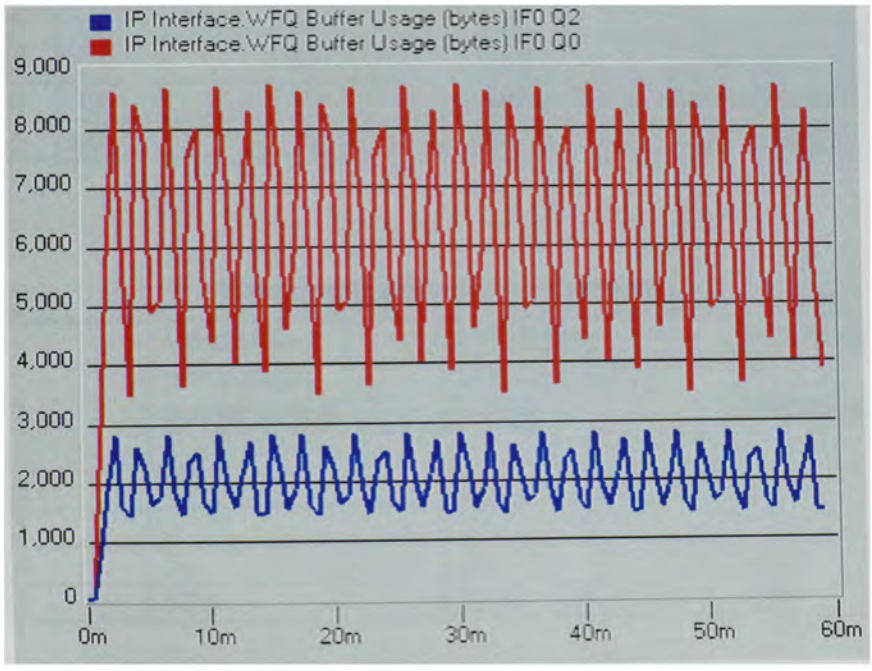

Figure 35 Buffer Usage

8.6 Conclusions

The comparison of link throughputs and utilizations before and after deployment of MPLS is shown in tables 8 and 9. It is shown that the load on the links is distributed in a balanced manner such that all links are active and efficiently utilized. Without MPLS, some links are flooded with traffic, depending upon the routing protocol used. There is no traffic in potentially alternative links. By adding MPLS, paths are created using all the available links and traffic is diverted into paths that are most suitable for them still using the IP routing functionality.

Table 8 Comparison of link throughputs

Link	Throughput before MPLS $*10^6$ bits/sec	Throughput after MPLS $*10^6$ bits/sec	Increase/ Decrease $*10^6$ bits/sec	Coments
LSR1-LSR3	10	40	30	Increase by four times
LSR3-LSR5	0	40	40	Unused link is utilized
LSR2-LSR4	70	30	40	Decrease by 2.3 times
LSR4-LSR5	60	20	40	Decrease by 3 times

Table 9 Comparison of Link utilizations

Link	Utilization percentage before MPLS	Utilization percentage after MPLS	Percentage of Increase/ Decrease
LSR1-LSR3	5	25	20
LSR3-LSR5	0	25	25
LSR2-LSR4	50	25	25
LSR4-LSR5	37.5	12.5	25

This answers the capacity issue with a touch of quality of service. It is important to incorporate traffic-engineering aspects into a network to optimally utilize resources

across the network, and to respect the service level agreements for congestion-sensitive traffic flows.

The following are the simulation results for uploading a 150MB file to a server.

Table 10 Comparison of differential service parameters

Parameter	High priority	Low priority
Upload response time	30 seconds	43 seconds
Queuing delay	0.8 seconds	2.25 seconds
Buffer usage	2 bytes	6 bytes

According to the simulation results, the flow with high priority has low response time, low queuing delay, and low buffer usage. The simulation experiments demonstrated that QoS level for a particular user can be changed with the use of Traffic engineering with MPLS. It was also shown that the overall network performance could be improved by combining MPLS and Diffserv.

8.7 Future Work

Extensions to this work can be done by taking into account node failures, multiple links or multiple node failure, or the computation of several backup paths to improve the resilience of the multicast routing tree. A mechanism for RSVP support can be developed for the applications, which make an extensive use of integrated services.

REFERENCES

[1] V. S. Victor, *ISPhone Inc.*, "Limits and Possibilities: White paper on VoIP Quality," June 2002.
URL: http://www.isphone.net/quality/html

[2] A. Jerry, *et al.* "Congestion Avoidance & Control for OSPF Networks," Internet Draft: draft-ash-manral-ospf-congestion-control-00.txt April, 2002.
URL: http://ietfreport.isoc.org/ids/draft-ash-manral-ospf-congestion-control-00.txt

[3] S. Chuck, *Juniper Networks, Inc.*, "Traffic Engineering for the New Public Network," May 31, 2002.

[4] S. Deering, "ICMP Router Discovery Messages", RFC 1256, Xerox PARC, September 1991.

[5] R. Braden, *USC/Information Sciences Institute*, Internet Engineering Task Force, "Requirements for Internet Hosts -- Communication Layers", RFC 1122 , October 1989.

[6] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and Principles of Internet Traffic Engineering", RFC 3272, May 2002.

[7] J. Nagle, "Congestion control in IP/TCP internetworks", RFC 896, January 1984.

[8] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[9] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992 November 2000.

[10] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A Framework for QoS-based Routing in the Internet", RFC 2386, August 1998.

[11] J. Heinanen, F.Baker, W.Weiss, and J.Wroclawski, "Assured Forwarding PHB", Febrauary 1999.

[12] Stardust.com Inc., "The need for QoS", presented at qosforum, July1998.

[13] S. Raghavan, "An MPLS based Quality of Service Architecture for Heterogenous Networks", M.S Thesis, Virginia Tech, Blacksburg, Virginia, U.S, 2001.

[14] M. Eder, H. Chaskar, and S. Nag, "Considerations from the Service Management Research Group (SMRG) on Quality of Service (QoS) in the IP Network", *IEEE Communications Magazine*, vol. 19, pp. 564-579, September 2002.

[15]Y. Bernet, *et al.* "A Framework for Integrated Services Operation over Diffserv Networks", *IEEE Communications Magazine*, vol. 46, pp. 695-701, November 2000.

[16] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", *IEEE Network*, vol. 16, pp. 10-15, June 1994.

[17] *Juniper Networks*, "RSVP-Overview", Juniper Technical Publications Center, Tech. 28, 2002.

[18] R. Braden *et al.*, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[19] F. Tommasi, and S. Molendini, University of Lecce, "Some Extensions to Enhance the Scalability of the RSVP Protocol", Internet Draft: draft-tommasi-rsvp-enhan-scalab-01.txt, January 2000.
URL: http://netlab.unile.it/id/draft-tommasi-rsvp-enhan-scalab-01.txt

[20] S. Berson, R. Lindell, and R. Braden," An Architecture for Advance Reservations in the Internet", ISI Technical Report, 1999.

[21] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB", RFC 2898, June 1999

[22] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB", *IEEE Communications Magazine*, vol. 23, pp. 33-65, June 1999.

[23] B. Williams, E. Australia, "Quality of Service, Differentiated Services and Multiprotocol Label Switching", *IEEE Communications Magazine*, vol. 27, pp. 53-58, March 2000.

[24] S. Sahu, D. Towsley, and J. Kurose, "A Quantitative Study of Differentiated Services for the Internet", *IEEE Network*, vol. 24, pp. 20-23, Apr 1999.

[25] U. Black, "*MPLS*", Prentice Hall, Second Edition, April 2002

[26] Nortel Networks, Documentations
URL:http://www.nortelnetworks.com/corporate/events/2001b/mpls_eseminar/collateral/55053_25-04-01.pdf

[27]Rick Gallaher, *Telecommunications Technical Services Inc.*, "An Introduction to MPLS"
URL:http://www.convergedigest.com/Bandwidth/archive/010910TUTORIAL-rgallaher4.htm

[28] B. Daive, et *al.*, "MPLS using LDP and ATM VC Switching", RFC 3035, Jan 2001.

[29] L. Anderson, *et al.*, "LDP Specification", RFC3036, Jan 2001.

[30] B. Thomas, *et al.*, "LDP Applicability" RFC3037, Jan 2001

[31] K. Nagami, *et al.*, "VCID Notification over ATM link for LDP", RFC3038 , Jan2001

[32] W. Stallings, "MPLS", *Internet Protocol Journal*, vol 4, pp. 3-6, September 2001.

[34] A. Banerjee, J. Drake, J. P. Lang, and B. Turner, "Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements", *IEEE Communications Magazine*, vol 5, pp 12-16, January 2001.

[35] P. Brittain, and A. Farrel, Data Connection Limited, "MPLS Virtual Private Networks", November 2000.

[36] E. Taylak, "The use of Differentiated Services with MPLS", *IEEE Network*, vol 34, pp 29-36, June 2002.

[37] X. Xiao and L. M. Ni, "Internet Qos: A big picture", *IEEE Network*, vol 13(2), pp.8–18, March/April 1999

[38] Y. D Lin, N. B. Hsu, and R.H. Hwang,"QoS routing granularity in MPLS Networks", *IEEE communications Magazine*, vol 26, pp.8–18, June 2002

[39] *X. Xiao, Providing quality of Service in the Internet*, New York: McGraw-Hill, 2000

[40], D. M. Neuse, SES Inc., "Why Simulate", *Capacity Management Review*, vol.36,pp. 2,February 1998

[41], R. L. Gimarc and A. C. Spellmann, HyPerformix, Inc., "Modeling the Customer Experience", *CMG Journal of Computer Resource Management*, vol 102, pp. 53-59, Spring 2001