Performance and Cost Analysis of QoS Routing in an Intranet

Zhansong Ma

Helsinki University of Technology
Department of Electrical and Communications Engineering
Laboratory of Telecommunications Technology
October 2000

HELSINKI UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMMUNICATIONS ENGINEERING

ABSTRACT OF THE MASTER'S THESIS

Työn tekijä:	Zhansong Ma	
Työn nimi:	Palvelun laatua tukevan Intranet reitityksen suorituskyky ja kustannus analyysi	
Päivä:	27.10.2000	**Sivumäärä**: 66
Osasto:	Sähkö- ja tietoliikennetekniikan osasto	
Professuuri:	S-38 Teletekniikan laboratorio	
Valvoja:	Professori Raimo Kantola	
Ohjaaja:	TkT Peng Zhang	

QoS reititystä pidetään QoS-pohjaisten Internet-palveluiden tärkeänä evoluution osa-alueena. QoS reititys laskee halutun QoS tason polut ja voi parantaa verkon resurssien käyttöastetta. Edellämainittujen etujen lisäksi QoS reititykseen liittyy kustannuksia. Suurin osa QoS reitityksen kustannuksiata tulee polkujen laskennan, linkkiyhteyden tilan, ja tallennuskapasiteetin kustannuksiata. Ennen kuin QoS reititystä kannattaa harkita käytettävän, suorituskyky ja kustannus suhdetta kannattaa tarkasti analysoida. Tässä diplomityössä on pyritty tutkimaan QoS reitityksen suorituskyky ja kustannus ongelmaa simuloimalla.

Aluksi työssä luodaan yleiskatsaus palvelun laatuun Internetissä, jonka yhteydessä seuraavanlaisia QoS-riippuvia komponentteja esitellään: integroidut palvelut, eriytetyt palvelut, MultiProtocol leimakytkentä ja palvelun laatua tukeva reititys. Seuraavaksi perehdytään QoS reitityksen ongelmiin. Lisäksi joitakin QoS reitityksen toteutukseen liittyviä asioita esitetään ja niitä pohditaan.

Jotta QoS reitityksen suorituskyky ja kustannus suhdetta voitaisiin tutkia, työssä suunniteltiin ja toteutettiin QoS reitityksen simulaattori. Kaksi reititysalgoritmia ja neljä linkkikerroksen tilapäivityksen algoritmia toteutettiin. Simulaattorin avulla QoS reitityksen suorituskyky ja kustannus suhteeseen vaikuttavia tekijöitä analysoitiin. Tuloksena mainittakoon esimerkiksi seuraavat tekijät: QoS reititys on erittäin herkkä verkon koolle, kustannus voidaan merkittävästi vähentää käyttämällä sopivaa linkkiyhteyden tilan päivitysalgoritmia, taajaan tulevat pyynnöt nostavat kustannus, ja että liikenteen luonne vaikuttaa QoS reitiyksen kustannus. Joitakin tässä diplomityössä esitettyjä tuloksia voidaan käyttää QoS IP-reitityksen käyttöönoton ohjeena.

Avainsanat: Quality of Service, IntServ, RSVP, DiffServ, MPLS, QoS reititys, QoS reitityksen algoritmi, Linkkiyhteyden tilan päivitysalgoritmi, QoS reitityksen suorituskyky, QoS reitityksen kustannus

HELSINKI UNIVERSITY OF TECHNOLOGY **DEPARTMENT OF ELECTRICAL AND** **COMMUNICATIONS ENGINEERING**	**ABSTRACT** **OF THE** **MASTER'S THESIS**

Author: Zhansong Ma

Title: Performance and Cost Analysis of QoS Routing in an Intranet

Date: 27.10.2000 **Number of Pages**: 66

Department: Electrical and Communications Engineering
Professorship: S-38 Telecommunication Technology

Supervisor: Professor Raimo Kantola
Instructor: Ph.D. Peng Zhang

QoS routing has been regarded as an important part in the evolution of QoS-based service offerings in the Internet. It computes paths that are subject to QoS requirements and can improve the utilization of network resources. Besides benefits, QoS routing also results in cost. The main cost of QoS routing includes path computation cost, link state update cost and storage cost. Before deploying QoS routing into the Internet, the performance and cost of QoS routing still needs careful study. This thesis aims to investigate this problem through simulation study.

This thesis begins with an overview of QoS in the Internet, in which some QoS related components, e.g., Integrated Services with Resource Reservation Protocol, Differentiated Services, MultiProtocol Label Switching and QoS routing, etc., are introduced. Then, the problem of QoS routing in the Internet is investigated. Also, some issues on implementing QoS routing in the Internet are presented and discussed.

In order to study the performance and cost of QoS routing, we designed and implemented a QoS routing simulator. Particularly, two routing algorithms and four link state update algorithms were implemented. With the help of the simulator, we analyzed the factors that affect the performance and cost of QoS routing. As results, we found that the cost of QoS routing is very sensitive to network size, that the cost can be significantly reduced by the use of suitable link state update algorithms, that frequent requests cause high cost, that traffic pattern is another factor that affects QoS routing cost. Some results presented in this thesis can be used for guiding the deployment of QoS routing in IP networks.

Keywords: Quality of Service, IntServ, RSVP, DiffServ, MPLS, QoS routing, QoS routing algorithm, Link state update algorithm, QoS routing performance, QoS routing cost

ACKNOWLEDGEMENTS

This thesis is done in Telecommunication Laboratory of Helsinki University of Technology. It started in October of 1999 and has lasted as long as 12 months until today that I finally feel ready to close up this task and to be a Master of Science graduate—the highest professional achievement so far in my life.

Even though I have worked as hard as I can to make this happen, I know this thesis would not have been as good as it is now without the contributions from many persons in various forms. I would like to thank them for their generosity to give comments and suggestions to improve this thesis and to share expertise and knowledge to improve my competence.

First of all, my warmest thanks are to my supervisor, professor Raimo Kantola who has guided me in the course of my study during past three years. As my supervisor, he has followed the progress of my thesis work and gave a lot of inspirational comments. His wide vision and precise manner in scientific research let me understand better how to become a successful professional.

I sincerely thank Dr. Peng Zhang who is a brilliant young man and my advisor. Since the day I got to know him, I have been benefiting from his suggestions and discussions. I have also been enjoying his positive attitude and professional competence as well as his friendship.

I sincerely thank Mr. Mika Ilvesmäki who kindly guided me in starting the research in telecommunication field in the summer of 1999. I am also very grateful to all the people in the laboratory who gave me a lot of practical help in the past three years.

Finally, I would like to say thanks to my dearest ones, my wife who shares my feelings and gives me unconditional love and support; my family in China for their encouragement and support; my friends in Finland who have been there with me whenever I needed them.

Zhansong Ma

Espoo, October 2000

CONTENTS

FIGURES

ABBREVIATIONS

BGP	Border Gateway Protocol
BE	Best-effort Traffic
CBQ	Class Based Queuing
DiffServ (DS)	Differentiated Services
DSCP	Differentiated Services CodePoint
ECB	Equal Class Based
ER	Explicit Route
EWMA	Exponential Weighted Moving Average
FTP	File Transfer Protocol
IntServ	Integrated Services
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
LDP	Label Distribution Protocol
LSA	Link State Advertisement
LSP	Label Switched Path
LSR	Label Switched Router
LSU	Link State Update
MPLS	MultiProtocol Label Switching
NBR	Nominal Bit Rate
NP	Non-deterministic Polynomial
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PB	Period Based
PHB	Per-Hop Behavior

PNNI	Private Network Network Interface
PQ	Priority Queuing
PSTN	Public Switched Telephone Network
QoS	Quality of Service
QOSPF	QoS OSPF
QRS	QoS Routing Simulator
RIP	Routing Information Protocol
RT	Real-time Traffic or Realtime Traffic
RFC	Request For Comment
RSVP	Resource Reservation Protocol
SLA	Service Level Agreement
ST	Simple Traffic
SPF	Shortest Path First
TB	Threshold Based
TCA	Traffic Conditioning Agreement
TE	Traffic Engineering
TOS	Type of Service
TTL	Time to Live
UCB	Unequal Class Based
WDM	Wavelength-Division Multiplexing

1 Introduction

With the success of the Internet in recent years, today's Internet armed with best-effort routing is being expected to support various services, not only the traditional services (e.g., email, FTP) but also the upcoming high speed and real-time services (e.g., audio/video real-time transmission, virtual private network). The latter services represent much different traffic characteristics from the former services in terms of bit-rate and burst, and they require fixed assurance of Quality of Service (QoS) in the duration of transmission. However, the current Internet does not support QoS requirements. As a result, a need for a high performance network emerges. Obviously it is almost impossible to build up a new high performance network while abandoning the legacy networks, e.g., public switched telephone network (PSTN) and Internet. The sole solution seems to be to immigrate the existing networks into the future network under a general architecture. Two almost opposite strategies for achieving the goal are being proposed. The first one is that fibers and wavelength-division multiplexing (WDM) will make bandwidth so abundant and cheap that QoS will be automatically delivered. The other one is, according to Moore's Law "as you increase the capacity of any system to accommodate user demand, user demand will increase to consume system capacity [1]", to build up a QoS-based network on the basis of the current Internet by providing classified services with quality requirements guaranteed.

Our view is inclined to the latter strategy. We believe that even if the bandwidth will eventually become abundant and cheap, it is not going to happen soon. At the moment, some techniques related to data transmission still lag far behind the increment of the capacity of the fiber. For example, the network bottleneck usually lies in the switching system instead of transmission system in the public wide-area network. Besides our view is also supported by the fact that all the major router/switch vendors now provide some QoS mechanisms in their high-end products, such as Cisco's 1200 series, Ascend's GRF routers, 3 Com's switches, Lucent's PacketStar 6400 Series.

Great efforts have been put forward to provide guarantees for specific services or customers, e.g., Integrated Services (IntServ) with the Resource Reservation Protocol (RSVP) and the Differentiated Services (DiffServ) architecture. MultiProtocol Label Switching (MPLS) is a forwarding scheme. It can be used together with DiffServ to provide better QoS. Quality of Service routing is to compute paths that are subject to QoS requirements. On the other hand, traffic engineering is concerned with performance optimization of operational networks.

QoS routing as a special case in constraint-based routing has been recognized as an important part in the evolution of QoS-based service offerings in the Internet. Some research results [2][3][4][5][6][7] have pointed out its potential benefits:

- enabling creation of virtual circuit-like services over internet protocol (IP) networks;
- improving user satisfaction by increasing chances of finding a path that meets the QoS requirements;
- improving network utilization by finding alternate paths around congestion spots.

However, these benefits come at the cost of deploying QoS routing protocol, of incurring potentially higher communication, processing and storage overheads. As a result, one major concern facing the deployment of QoS routing is its feasibility, that is whether or not its benefits are worth its cost.

This thesis aims to discuss the design issues of QoS routing; to evaluate network performance and cost of QoS routing; to explain the feasibility of deploying QoS routing into the Internet; and to develop mechanisms for improving QoS routing in the Internet.

Since studying QoS routing in global IP networks is a hard and broad issue, we choose to investigate it in the intranet as the first step to simplify the implementation and help us to clarify the key factors dominating the feasibility of the QoS routing. Besides, due to the complexity of multicast routing, at this stage our study only considers the unicast routing. Further justifying this issue, we consider that studying such a problem in a real network will be quite expensive and probably cause some unexpected effects on the network administration. Therefore we choose simulation-based study. We use a QoS Routing Simulator (QRS) [8] to investigate our interests.

The rest of this paper is arranged as follows. In Chapter 2, we give an overall introduction to Internet quality of service, focusing on Internet QoS architecture, traffic engineering, MPLS and constraint-based routing. We present more details about QoS routing problems in Chapter 3. Issues on QoS routing implementation are discussed in Chapter 4.

We discuss the design issues of a QoS routing simulator in Chapter 5. We begin with an introduction to general design, and then the design of a QoS routing protocol. Particularly, we focus on QoS routing's three core functions, i.e., distribution of routing information, storage of resource information and computation of QoS paths.

In Chapter 6, we first analyze the performance and cost of QoS routing. And then we present the simulation results. All simulations are done in QRS. In simulations, we vary network size, traffic model, link state update (LSU) algorithm in order to study how these parameters influence the performance and cost of QoS routing. Finally conclusions and future work are presented in Chapter 7.

2 Quality of Service in the Internet

2.1 Introduction

With the rapid transformation of the Internet into a commercial infrastructure, it is becoming apparent that several service classes will likely be demanded. One service class will provide predictable Internet services for companies that do business on the Web. Such companies will be willing to pay a certain price to make their services reliable and give their users a fast feel of their services. Another service class will provide low-delay and low-jitter services to the applications such as Internet telephony and videoconferencing. Meanwhile, best-effort service will remain for those customers who only need connectivity. Therefore it is fair to say that the current Internet is expected to become a QoS-based Internet in which various services with QoS requirements will be provided.

QoS refers to the ability of a network to provide better services to selected network traffic over different underlying technologies. QoS features provide better and more predictable network services by:

- Supporting dedicated bandwidth;
- Improving loss characteristics;
- Avoiding and managing network congestion;
- Shaping network traffic;
- Setting traffic priorities across the network.

To configure QoS features throughout a network and to provide QoS delivery, a general QoS architecture as well as some mechanisms such as traffic engineering techniques are needed. So far many different kinds of QoS architectures and mechanisms have been proposed to meet the demands of QoS. This chapter gives an overall discussion of QoS related issues. We begin with the discussion of the demands of QoS. Further sections are organized as follows: Section 2.2 explains first the components of QoS architectures and then gives a brief introduction to two QoS architectural models, i.e., InteServ and DiffServ. The last three sections present general discussions of traffic engineering, MPLS and constraint-based routing.

2.2 Internet QoS Architecture

The objective of a QoS architecture is to provide a framework for the integration of quality of service control and management mechanisms. The following three components are necessary to deliver QoS across a heterogeneous network:

- QoS within a single network element, which includes queuing, scheduling, and traffic shaping features, etc;
- QoS signaling techniques for coordinating QoS from end-to-end between network elements;
- QoS policing and management functions to control and administer end-to-end traffic across a network.

2.2.1 Integrated Services

The fundamental idea of IntServ [9] architecture is to reserve resources such as bandwidth and buffers, and a priori for a given traffic flow to ensure that the QoS required by the flow is satisfied. In addition to best-effort service, IntServ offers two integrated services: guaranteed service [10], and controlled-load service [11].

2.2.1.1 Framework

Figure 1 illustrates the framework of IntServ. It includes a number of components, i.e., signaling protocol, admission control, packet classifier and packet scheduler.

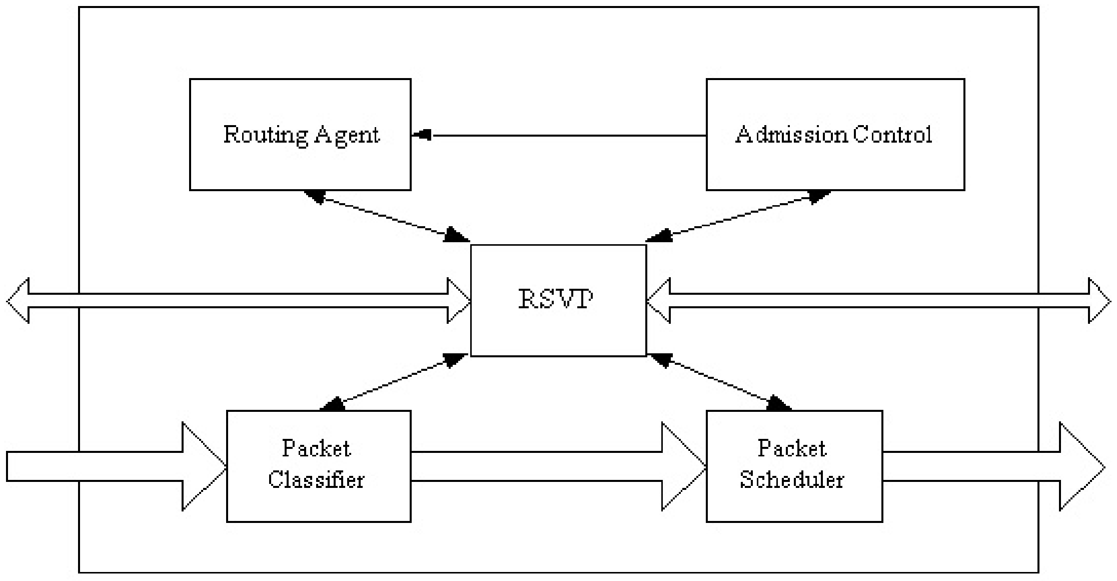

Figure 1 Integrated services architecture

Signaling Protocol

RSVP [12] is invented as a signaling protocol for applications to reserve resources. The signaling process is illustrated in Figure 2. The sender sends a PATH message to the receiver specifying the characteristics of the traffic. Every intermediate router along the path forwards the PATH message to the next hop determined by the routing protocol. Upon receiving a PATH message, the receiver responds with RESV message to request resources for the flow. Every intermediate router along the path can reject or accept the request of RESV message. If the request is accepted, link bandwidth and buffer space are allocated for the flow, and the related flow-state-information will be installed in the router. If the request is rejected, the router will send an error message to the receiver, and the signaling process will terminate.

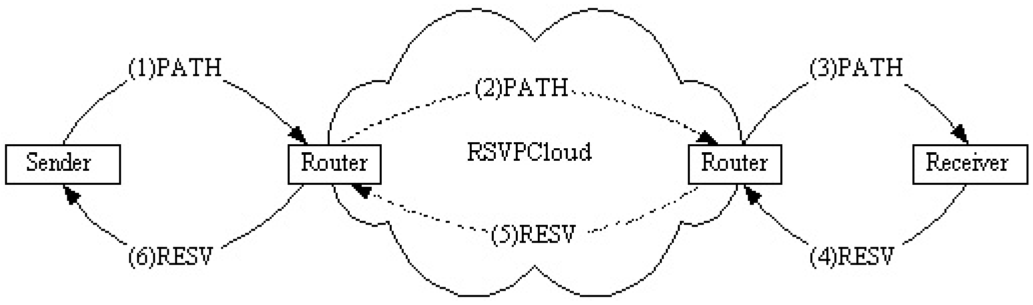

Figure 2 RSVP signaling

RSVP reservation request specifies the amount of resources to be reserved for all, or some subsets, of the packets in a particular session. The resource quantity is specified by a flow specification (FlowSpec), while the packet subset to receive those resources is specified by a filter specification (FilterSpec). Assuming admission control succeeds, the FlowSpec will be used to parameterize a resource class in the packet scheduler, and the FilterSpec will be instantiated in the packet classifier to map the appropriate packets into this class.

Admission Control

Admission control implements the decision algorithm that a router or host uses to determine whether a new flow can be granted without impacting earlier guarantees. Admission control is invoked at each node to make a local accept/reject decision when a host requests a real-time service along some path through the Internet. The admission control algorithm must be consistent with the service model, and it is logically part of traffic control.

In addition to ensuring that QoS guarantees are met, admission control is concerned with enforcing administrative policies on resource reservations. Some policies will demand authentication of those requesting reservations. Besides, admission control plays an important role in accounting and administrative reporting.

Packet Classifier

Packet classifier maps each incoming packet into some classes. Choice of a class may be based upon the contents of the existing packet headers and/or some additional classification number added to each packet.

Packet Scheduler

The packet scheduler manages the forwarding of different packet streams using a set of queues and perhaps other mechanisms like timers. All packets in the same class get the same treatment from the packet scheduler. The packet scheduler must be implemented at the point where packets are queued; this is the output driver level of a typical operating system, and corresponds to the link layer protocol.

2.2.1.2 Service Classes

Guaranteed Service guarantees that datagrams will arrive within the guaranteed delivery time and will not be discarded due to queue overflows, which ensures the flow's traffic stays within its specified traffic parameters. This service is intended for applications which need a firm guarantee that a datagram will arrive no later than a certain time after it was transmitted by its source. Such for example can be some real-time audio/video applications.

Controlled–Load Service intends to support a broad class of applications that were originally developed for today's Internet, but is highly sensitive to network overload. Controlled-load service strives to approximate tightly the behavior visible to applications receiving best-effort service during unloaded conditions. This implies that both the packet loss ratio and minimum delay will remain unchanged regardless of the overall load level within the network.

To ensure that these requirements are fulfilled, subscribers requesting controlled-load service initially provide the intermediate network elements with an estimation of the data traffic that they will generate. If the traffic generated by the subscriber should fall outside of the region described by this descriptive envelope, the QoS provided to the subscriber may deteriorate in terms of delay and packet loss.

2.2.1.3 Problems

IntServ architecture represents a fundamental change to the current Internet architecture, which is founded on the concept that all flow-related state information should be in the end systems. However it meets some problems:

- The amount of state information increases proportionally with the number of flows. This places a huge storage and processing overhead on the routers. Therefore, this architecture does not scale well;
- The requirement on the routers is high. All routers must have RSVP, admission control, packet classification, and packet scheduling;
- Ubiquitous deployment is required for guaranteed service. Incremental deployment of controlled-load service is possible by deploying controlled-load service and RSVP functionality at the bottleneck nodes of a domain and tunneling the RSVP messages over other parts of the domain.

2.2.2 Differentiated Services

DiffServ [13][14] architecture is based on a simple model where traffic entering a network is classified and possibly conditioned at the boundaries of the network, and assigned to different behavior aggregates. Each behavior aggregate is identified by a single Differentiated Services codepoint (DSCP). Within the core of the network, packets are forwarded according to the per-hop behavior (PHB) associated with the DSCP. IETF DiffServ Working Group defines two PHB groups: the expedited forwarding PHB group (EF-PHB) and the assured forwarding PHB group (AF-PHB) [15][16]. Other two possible PHB groups are [17]: the class selector PHB group (CS-PHB) and the dynamic real-time/non real-time (RT/NRT) PHB group (DRT-PHB).

Using those PHBs, several classes of services can be defined using different classification, policing, shaping, and scheduling rules.

2.2.2.1 Framework

DiffServ architecture consists of a number of functional elements implemented in network nodes, including a set of PHBs, packet classification functions, and traffic conditioning functions including metering, marking, shaping, and dropping.

DS Domain

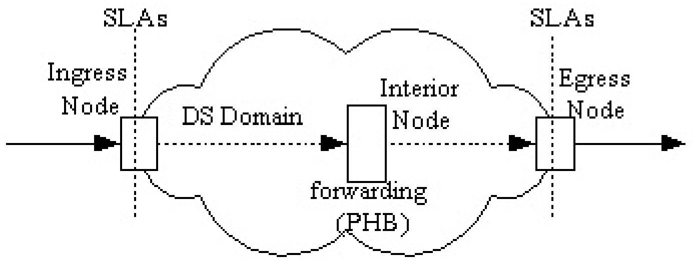

Figure 3 Differentiated services domain

A DS domain is a contiguous set of nodes which operate with a common service provisioning policy and set of PHB groups implemented on each node. It consists of DS boundary nodes and DS interior nodes as shown in Figure 3. DS boundary nodes act both as a DS ingress node and as a DS egress node for different directions of traffic. DS boundary nodes interconnect the DS domain to other DS or non-DS capable domains, while DS interior nodes only connect to other DS interior or boundary nodes within the same DS domain. A boundary node could contain functionality for both logical interconnection of domains and controlling traffic streams. The interior node could have certain limited traffic conditioning capabilities, e.g. for degrading the importance level of a multicast packet at each hop.

In addition, DS boundary nodes classify and possibly condition ingress traffic to ensure that packets transiting the domain are appropriately marked to select a PHB from one of the PHB groups supported within the domain.

DS Region

A DS region is a set of contiguous DS domains. It is capable of offering differentiated services over paths across its DS domains.

The DS domains in a DS region may support different PHB groups internally and different DSCP→PHB mappings. However, to permit services which span across the domains, each peering DS domain must establish a peering service level agreement (SLA) which defines a traffic conditioning agreement (TCA). A TCA specifies how transit traffic is conditioned at the boundary between the two DS domains.

Traffic Classification

The packet classification policy identifies the subset of traffic which may receive differentiated services by being conditioned and/or mapped to one or more PHB groups within the DS domain.

The classification is performed by packet classifiers which select packets in a traffic stream based on the content of some portion of the packet header. In general, two types of classifiers are widely discussed. The behavior aggregate (BA) classifier classifies packets based on the DSCP only. The multi-field (MF) classifier selects packets based on the value of a combination of one or more header fields, such as source address, destination address, DS field, protocol ID, source port and destination port numbers, and some other information.

Traffic Conditioning

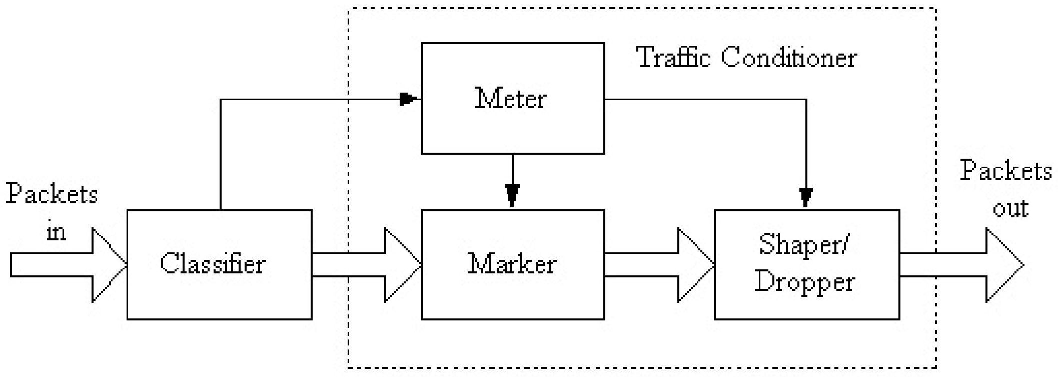

Figure 4 The packet classifier and traffic conditioner

Traffic conditioning is to enforce rules specified in a TCA. It is done by a traffic conditioner which contains the following elements: meter, marker, shaper and dropper. The meter measures each traffic stream and informs the marker, shaper and dropper elements of the status of the stream. The marker sets the importance level of the packet according to the given status of the stream. The shaper is used to smooth the traffic stream at a particular aggregate level. Dropper makes discarding decisions based on the content of the service level and TCAs.

Per-hop behavior

The PHB plays a significant role in DiffServ architecture. The term PHB refers to a set of rules that allows for the treatment of packets in a specific and unambiguous way inside the network. These rules should be easily comprehensible, as they may also be used as a base for discussion between service providers, backbone operators and vendors. In practice, PHB defines the service that the packet receives at each hop as it is forwarded through the network.

2.2.2.2 PHB Groups

The Expedited Forwarding PHB

EF-PHB aims to provide low-loss, low-latency, low-jitter and assured bandwidth end-to-end service through DS domains. It can be also described as premium service. There are no different service classes inside the EF-PHB. The purpose is that all EF packets inside the network are forwarded as quickly as possible. To make this possible there must be tight traffic control in the ingress node to ensure that no source exceeds the maximum agreed bite rate. There is then a problem that because no reservations are done in the network, how the ingress node knows whether to accept the EF packet or not. Due to strict border traffic-control, a congestion situation within the network is considered erroneous; thus any excessive packets will be discarded.

The Assured Forwarding PHB Group

AF-PHB group aims to provide reliable service even on times of network congestion. It may comprise a number of PHB classes, each with several importance levels. The current specification defines four classes with three importance levels. Packets must be forwarded independently from packets in another AF class so at least the minimum amount of resources for each of the classes must be assigned. In case of congestion packets with a lower importance level within the class will be discarded first. It should be noted that AF-PHB group is not an end-to-end service model, but rather a collection of tools to build services. The relation between different AF classes is still open.

The Class Selector PHB Group

CS-PHB group intends to provide backward compatibility with the present use of type of service (TOS) field defined in IPv4. A CS-PHB should give packets a probability of timely forwarding that is not lower than that given to packets marked with lower CS-PHB, under reasonable operating conditions and traffic loads. This is the first goal of CS-PHB, and is called timely forwarding requirement. In addition, network nodes may wish to limit the amount of resources of each PHB, i.e., to impose bandwidth enforcement on a PHB. Moreover, the CS-PHB may be used to provide a grade of service differentiation, in respect to delay, importance or bandwidth.

To achieve all three goals simultaneously, the traffic load within each individual PHB must be tightly controlled. Particularly, timely forwarding and bandwidth enforcement may conflict during congestion. This can be resolved by prioritizing the timely forwarding requirement, so that bandwidth enforcement is considered merely a tool to achieve timely forwarding.

The Real-time and Non Real-time PHB Group

The main target of DRT-PHB group is to provide a consistent and clear framework for building differentiated services. This group defines a system with two PHB classes, each of which comprises six different levels of importance. The two PHB classes offer distinctively different delay characteristics: the real-time class is for flows requiring real-time service, while the non-real-time class is for flows without strict delay requirement. The six levels of importance offer wide dynamics for implementing diverse traffic-control mechanisms and pricing schemes. The framework may be modified if necessary. For example, a real-world implementation could consist of a larger number of delay classes or importance levels, but decreasing these values may dilute the usefulness of the DRT-PHB group.

One of the fundamental concepts of the DRT-PHB group is the nominal bit rate (NBR). The NBR defines the relative amount of resources that a certain entity is allowed to obtain from the network. In this context, the entity could be a flow, part of a flow, a customer, a group of customers, or perhaps an entire organization.

2.2.3 Differences between IntServ and DiffServ

DiffServ is significantly different from IntServ at two points:

- First, differentiated service is allocated in the granularity of a class, the amount of state information is proportional to the number of classes rather than the number of flows. DiffServ is therefore more scalable;
- Second, sophisticated classification, marking, policing, and shaping operations are only needed at the boundary of the network. Core routers need only to have behavior aggregate classification. Therefore, it is easier to implement and deploy differentiated services.

2.3 Traffic Engineering

Traffic engineering is concerned with performance optimization of operational networks. In general, it consists of the application of technology and scientific principles to the measurement, modeling, characterization, and control of the Internet traffic. The major goal of traffic engineering is to facilitate efficient and reliable network operations while simultaneously optimizing network resource utilization and traffic performance.

Historically, effective traffic engineering has been difficult to achieve in public IP networks, due to the limited functional capabilities of conventional IP technologies. But recent developments in MPLS open new possibilities to address some of the limitations of IP systems concerning traffic engineering [18][19][20][21]. Constraint-based routing is an important tool for making the traffic engineering process automatic [22].

2.3.1 Process Model

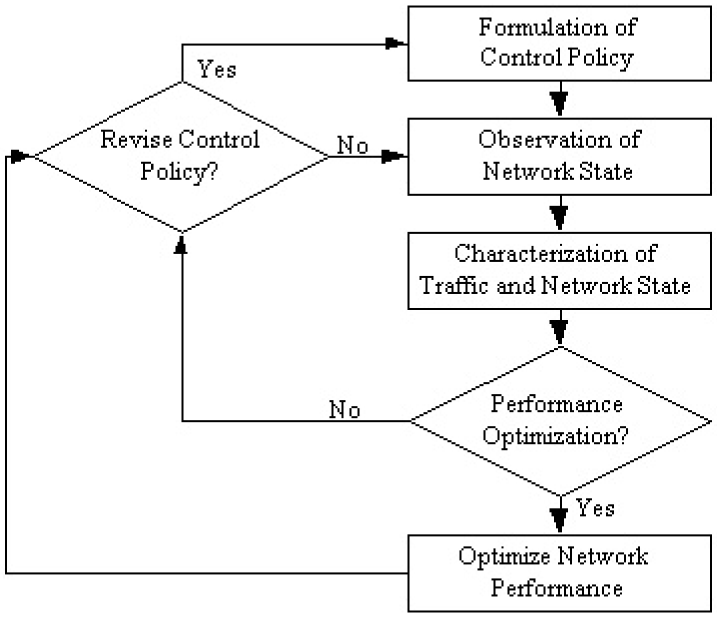
Formulation of Control Policy
Yes
Revise Control Policy?
No
Observation of Network State
Characterization of Traffic and Network State
No
Performance Optimization?
Yes
Optimize Network Performance

Figure 5 Traffic engineering process model

As illustrated in Figure 5, traffic engineering process model consists of four practical functions.

The formulation of a control policy: The control policy depends on the network context, cost structure, revenue or utility model, operating constraints, and success criteria.

The observation of the network state: This is done through a set of monitoring functions. It is the feedback component of the traffic engineering process. It may include preprocessing activities such as data reduction and data transformation.

The characterization of traffic and analysis of the network state: This function can be accomplished by various qualitative and quantitative techniques. The target is to identify the bottlenecks and pathologies that impede network performance.

Optimization of network performance: This is achieved by applying control actions, if necessary, to drive the network to a desired state according to the control policy.

2.3.2 Objectives

A practical function of traffic engineering in IP networks is the mapping of traffic onto the network infrastructure to achieve specific performance objectives. It requires precise control over the routing function to achieve the objectives. Therefore an essential requirement for traffic engineering in IP networks is the capability to compute and establish a forwarding path from one node to another. This path must fulfill some requirements, while also satisfying network capacity and policy constraints. Generally, performance objectives can be traffic-oriented, resource-oriented and operation-oriented.

Traffic-Oriented Performance Objectives

Traffic-oriented performance objectives relate to the improvement of the QoS provisioned to Internet traffic. Traffic-oriented performance metrics include packet loss, delay, delay variation, and throughput. The effectiveness of traffic-oriented policies can also be measured in terms of the relative proportion of offered traffic achieving their performance requirement. In case that SLAs are involved, it is a very important part of traffic-oriented performance objectives to protect traffic streams that comply with their SLAs from those non-compliant with their SLAs.

Resource-Oriented Performance Objectives

Resource-oriented performance objectives relate to the optimization of the utilization of network assets. Efficient resource allocation is the basic approach to secure resource-oriented performance objectives. In general, a traffic engineering system can be recognized as 'rational' only when it can address traffic-oriented performance problems at the same time utilizing network resources efficiently.

Operation-Oriented Performance Objectives

Operation-oriented performance objectives relate to improving the reliability of network operation. Multiple failure recovery scenarios must be devised to ensure continuity following network impairments. Adequate capacity for service restoration must be provided. Therefore the operational capability must exist to expeditiously reroute traffic through the redundant capacity when faults occur. Re-optimization may be required following restoration to make more effective use of the residual posse-fault capacity. It may be advantageous to utilize subsets of the redundant capacity to improve network performance and efficiency when the network is fault-free.

2.4 Multiprotocol Label Switching

MPLS[23][24] is a forwarding scheme. Packets are assigned labels at the ingress of an MPLS-capable domain. Subsequent classification, forwarding, and services for the packets are based on the labels. MPLS is evolved from Cisco's Tag Switching. In the open systems interconnection (OSI) seven-layer model, it is between link layer and network layer.

2.4.1 Concepts of MPLS

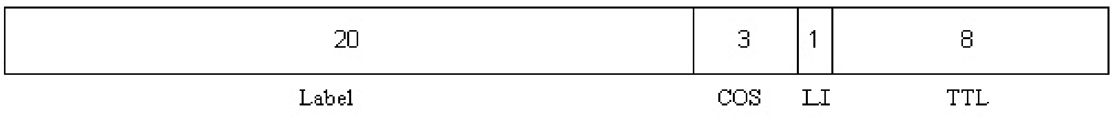

Figure 6 MPLS packet header

Each MPLS packet has a header as shown in Figure 6. The header contains a 20-bit label, a 3-bit class of service (COS) field, a 1-bit label stack indicator (LI) and a 8-bit time to live (TTL) field. The MPLS header is encapsulated between the link layer header and the network layer header. A MPLS-capable router is called the label switched router (LSR). It examines only the label when forwarding the packet. The network protocol can be IP or others. This is why it is called multiprotocol label switching.

MPLS needs a label distribution protocol (LDP) to distribute labels for setting up label switched paths (LSPs). MPLS labels can also be piggybacked in routing protocols. MPLS LSRs use LDPs to negotiate the semantics of each label, that is, how to handle a packet with a particular label from the peer. LSP set-up can be control-driven (i.e., triggered by control traffic such as routing update) or data-driven (i.e., triggered by the request of a flow or traffic trunk). In MPLS, a traffic trunk is an aggregation of flows with the same service class that can be put into one LSP. The LSP between two routers can be the same as the network layer hop-by-hop route, or the sender LSR can specify an explicit route (ER) for the LSP. The capability to set up ERs is one of the most useful features of MPLS. A forwarding table indexed by labels is constructed as the result of label distribution. Each forwarding table entry specifies how to process packets carrying the indexing labels.

Packets are classified and routed at the ingress LSRs of an MPLS-capable domain. MPLS headers are then inserted. When an LSR receives a labeled packet, it will use this label as the index to look up the forwarding table. The packet is processed as specified by the forwarding table entry. The incoming label is replaced by the outgoing label, and the packet is switched to the next LSR. Inside an MPLS domain, packet forwarding, classification, and service quality are determined by the labels and the COS fields. This makes core LSRs simple. Before a packet leaves an MPLS domain, its MPLS label is removed.

2.4.2 MPLS VPN

When an LSP has been established, it can be seen as a tunnel through the network. A packet's path can be completely determined by the label assigned by the ingress LSR. There is no need to enumerate every intermediate router of the tunnel. Comparing to other tunneling mechanisms, MPLS is unique in that it can control the complete path of a packet without explicitly specifying the intermediate routers. Hence one attractive application of MPLS is building up Virtual Private Networks (VPNs) [25].

2.4.3 MPLS DiffServ

Because MPLS is path-oriented, it can potentially provide faster, reliable service than connectionless-oriented DiffServ. MPLS can be used together with DiffServ to provide better QoS [26]. In such an architecture, LSPs are first configured between each ingress-egress pair. For LSP(LSR1→ LSR2) and LSP(LSR2→LSR1), their intermediate LSRs need not be reciprocal. It is likely that for each ingress-egress pair, a separate LSP is created for each traffic class.

The operation of the routers is basically the same as in the DS-field-based architecture described in 2.2.2. In the processing of a packet, there are three differences:

- At the ingress, in addition to all the processing in the DS-field-based architecture, an MPLS header is inserted into the packet;
- Core routers process the packet based on its label and COS field rather than its DS field;
- At the egress, unless inter-domain LSPs are configured, the MPLS header is removed.

With such a scheme, the MPLS effect is confined within the ISPs that use MPLS. Whether a particular ISP's architecture is DS-field-based or MPLS-based, it is transparent to other ISPs. Therefore, the DS-field-based and MPLS-based architecture can easily interoperate.

2.4.4 MPLS Traffic Engineering

MPLS is strategically significant for traffic engineering, because it can potentially provide most of the functionality available from the overlay model, in an integrated manner, and at a lower cost than the currently competing alternatives.

The applicability of MPLS to traffic engineering can be attributed to the following factors:

- Explicit LSPs which are not constrained by the destination based forwarding paradigm can be easily created through manual administrative action or through automated action by the underlying protocols;
- LSPs can potentially be efficiently maintained, and traffic trunks can be instantiated and mapped onto LSPs;
- A set of attributes can be associated with resources which constrain the placement of LSPs and traffic trunks;
- MPLS allows for both traffic aggregation and disaggregation whereas classical destination only based IP forwarding permits only aggregation;
- It is relatively easy to integrate a constraint-based routing framework with MPLS;
- A good implementation of MPLS can offer significantly lower overhead than competing alternatives for traffic engineering.

2.5 Constraint-Based Routing

Constraint-based routing is to compute paths for traffic flows with multiple constraints including QoS constraints (requirements) and policy constraints. To determine a path, constraint-based routing considers not only network topology, but also requirements of the flow, resources availability of the links, and possibly other policies specified by the network administrator.

Constraint-based routing is recognized as an essential enabling mechanism for a variety of emerging network services such as virtual private networking and QoS support. A lot of work has been done to investigate its significance and its operation aspects. In particular, most work focused on its special case–QoS routing, which only considers QoS requirements when determining routes. We believe implementing QoS routing is the primary step to implement constraint-based routing with the full set of characteristics. Hence, in the rest of this thesis, we will only focus on QoS routing.

3 QoS Routing in the Internet

3.1 Introduction

The notion of QoS has been proposed to capture the qualitatively or quantitatively defined performance contract between the service provider and the user applications. The QoS requirements of a connection are given as a set of constraints which can be link constraints, path constraints or tree constraints [6]. A link constraint specifies a restriction on the use of links, for example, a bandwidth constraint of a unicast connection requires that links composing the path must have certain amount of free bandwidth available. A path constraint specifies the end-to-end QoS requirements on a single path; a tree constraint specifies the QoS requirements for the entire multicast tree, for example, the delay constraint of a multicast connection requires that the longest end-to-end delay from the sender to any receiver in the tree must not exceed an upper bound.

QoS routing has been defined in different ways. In [2], it is defined as “a routing mechanism under which paths for flows are determined based on some knowledge of resource availability in the network as well as the QoS requirements of flows”. In [27], it is defined as “a dynamic routing protocol that has expanded its path-selection criteria to include QoS parameters such as available bandwidth, link and end-to-end path utilization, node resource consumption, delay and latency, and induced jitter”.

However no matter which definition is concerned, the basic function of QoS routing is to find feasible paths which have sufficient residual resources to satisfy the QoS requirements of flows while achieving efficiency in network resource utilization. Designing and implementing QoS routing is much more difficult than best-effort routing. Some tradeoffs have to be made. And in most cases, the goal is not to find the best solution, but a viable solution with acceptable cost.

In this chapter, we give a survey on this subject. The next section presents the objectives of QoS routing. Section 3.3 presents the requirements of QoS routing at intra-domain level and inter-domain level. QoS routing problems and algorithms designed to solve those problems are described in section 3.4 and 3.5, respectively. In section 3.6, we summarize the current status of QoS routing research and discuss the future research directions.

3.2 Objectives of QoS Routing

Current Internet routing protocols such as OSPF, RIP, and BGP are best-effort routing protocols. They use a single objective optimization algorithm which considers only one metric such as bandwidth, hop-count or cost to find a “shortest” path for each traffic. Therefore, even if there are some alternate paths existing, they are not used as long as they are not the “shortest” ones. Obviously, one drawback of this kind of scheme is that it may lead to the congestion of some links, while some other links are not fully used.

In addition, during data transmission, whenever the new "shortest" path is found, the best-effort routing will shift the traffic to this path from the earlier defined "shortest" path. Thus, traffic may be routed back and forth between different paths. This kind of shift is undesirable because it will bring routing oscillations when the routing is based on metrics such as available bandwidth. These oscillations may change rapidly from time to time, and then increase the variation in the delay and jitter experienced by the end users.

QoS routing is supposed to solve or avoid the problems mentioned above. The main objectives of QoS routing [2] are:

- To meet the QoS requirements of end users. In case there are several feasible paths available for a given flow, a path is selected dynamically or may be selected subject to some policy constraints such as path cost, provider selection, etc.
- To optimize the network resource utilization. This is an objective from service providers' point of view. Every service provider wants to maximize the utilization of its current network facilities. Besides, this is also a requirement from network engineering's perspective. QoS routing is expected to direct network traffic in an efficient way to maximize the total network throughput;
- To gracefully degrade network performance when things like congestion happen. When network is in heavy load, QoS routing is expected to give better performance (e.g. better throughput) than best-effort routing, which degrades the performance dramatically.

3.3 Intra-domain and Inter-domain QoS Routing

QoS routing is expected to be applied into IP networks at both intra-domain level and inter-domain level. At intra-domain level, it is intended to accommodate many different routing schemes in one domain. The network managers should have the freedom to use whatever QoS routing scheme inside the domain, which is independent of the QoS routing used in other domains. Also, diversity is encouraged at this level. However, an intra-domain QoS routing should at least satisfy the following requirements:

- The routing scheme must route a flow along a path that can accommodate its QoS requirements, if such a path exists. Otherwise it indicates that the flow cannot be admitted;
- The routing scheme must accommodate best-effort flows without any resource reservation requirements. That is, present best-effort applications and protocol stacks do not have to be changed to run in a QoS routing domain;
- The routing scheme must indicate disruptions to the current path of a flow due to topological changes and build an alternative path if one exists;
- The routing scheme has the capabilities to optimize the utilization of network resources;
- The routing scheme is expected to have the option of supporting multicast QoS routing with receiver heterogeneity and shared reservation styles.

In addition, some other capabilities are also recommended, for example the implementation of higher level admission control procedure to limit the overall resource utilization by individual flows. Further requirements along these lines may be specified. The requirements should capture the consensus view of QoS routing, but should not preclude particular approaches from being implemented. Thus the intra-domain QoS routing requirements are expected to be rather broad.

In contrast, an inter-domain QoS routing is expected to be as simple as possible. Stability and scalability are the most important issues at this level. Therefore, the routing can not be based on highly dynamic network state information. The exchange of QoS routing information between different routing domains should be relatively static. However, an inter-domain QoS routing should at least satisfy the following requirements:

- Making determination of whether a destination is reachable;
- Avoiding routing loops;
- Supporting address aggregation;
- Making determination of whether the QoS requirements can be supported on the path to a destination;
- Making determination of multiple paths to a given destination optionally, based on service classes;
- Mapping routing policies (e.g. monetary cost, usage and administrative factors) onto flow metrics.

Most proposed QoS routing algorithms concentrate on one particular problem and have different assumptions of network conditions. They can hardly work together. Hence, a common framework that can accommodate different kinds of algorithms is needed. IETF QoS routing group [2] provides such a framework which consists of intra-domain level and inter-domain level.

In addition, most existing QoS routing algorithms belong to the intra-domain level. In a recent work [29], Zhang, P. and Kantola, R. gave a detailed discussion about the needs and problems of QoS routing at inter-domain level. They also provided an inter-domain QoS routing model and proposed some mechanisms for operating inter-domain QoS routing in a DiffServ network.

3.4 QoS Routing Problems

3.4.1 A Network Model

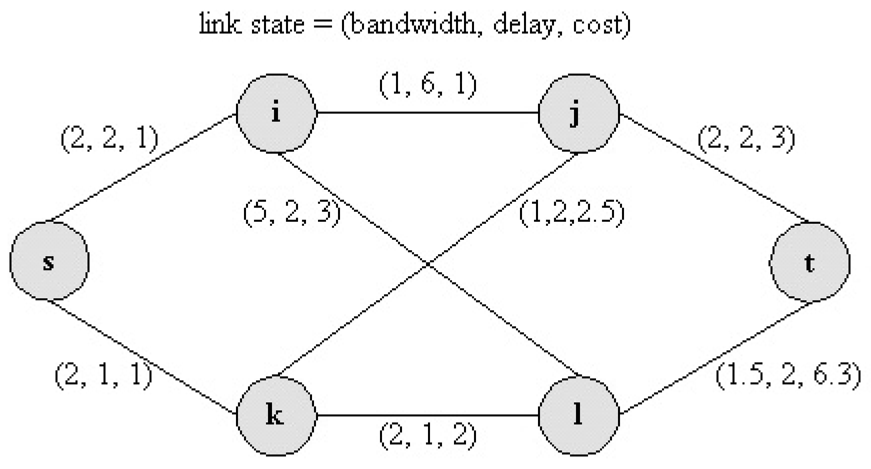

Figure 7 Network model with link state

As far as QoS routing is concerned, a network is usually modeled as a graph *G=(N,E)*, where *N* is a set of nodes representing switches, routers, and hosts and *E* is a set of communication links. |*N*| and |*E*| denote the number of nodes and the number of links in the network respectively. For most real networks the communication links are asymmetric, hence each link is represented by two directed edges in the opposite directions. The edges are undirected only if the links are always symmetric.

Every link has a state measured by the QoS metrics of concern such as residual bandwidth, delay, delay jitter, cost, residual buffer space and so on. In the network model as shown in Figure 7, the link state is a triple consisting of residual bandwidth, delay, and cost. The cost of a link can be defined in dollars or as a function of the buffer or bandwidth utilization. Moreover, each node has a state of node resources, e.g., CPU bandwidth. The state of a node can be considered in conjunction with the link state.

3.4.2 Classification of QoS Routing Problems

QoS routing problems can be classified into two major classes: unicast routing and multicast routing [30]. Unicast routing problem is defined as follows: given a source node *s*, a destination *t*, a set of QoS constraints *C*, and possibly a policy *P*, find the best feasible path from *s* to *t* which satisfies *C* and complies with *P* if it is implied. Multicast routing problem is defined as follows: given a source code *s*, a set *R* of destination nodes, a set of QoS constraints *C*, and possibly a policy *P*, find the best feasible tree covering *s* to all nodes in *R* which satisfies *C* and complies with *P* if it is applied. Multicast routing can be viewed as the generalization of unicast routing in many cases. Routing problems concerning multicast routing and unicast routing can be partitioned into subclasses according to the QoS based metrics

3.4.2.1 Unicast Routing Problems

In a unicast routing domain, concerning metrics of bandwidth and residual buffer space, the routing problems can be divided into two subclasses:

- link-optimization routing: An example is bandwidth-optimization routing which is to find a path that has the largest bandwidth on the bottleneck link.
- link-constrained routing: An example is bandwidth-constrained routing which is to find a path whose bottleneck bandwidth is above a required value.

Concerning other metrics such as delay, delay jitter and cost, the routing problems can be divided into two subclasses:

- path-optimization routing: An example is the least cost routing which is to find a path which has the minimum cost.
- path-constrained routing: An example is the delay-constrained routing which is to find a path whose delay is bounded by a required value.

Many composite routing problems can be derived from the above four basic problems. Among them, there are two NP-complete problem classes:

- path-constrained path-optimization routing: An example is delay-constrained least-cost routing which is to find the least-cost path with bounded delay.
- multi-path-constrained routing: An example is delay-delay-jitter-constrained routing which is to find a path with both bounded delay and bounded delay jitter.

3.4.2.2 Multicast Routing Problems

Multicast routing is different from unicast routing. In multicast routing, an optimization or a constraint must be applied to the entire tree instead of a single path. There are several well-known NP-complete multicast routing problems:

- tree-optimization routing: An example is least-cost multicast routing which is to find the least-cost tree. It is also known as the Steiner tree problem.
- tree-constrained tree-optimization routing: An example is delay-constrained least-cost routing which is to find the least-cost tree with bounded delay. It belongs to constrained Steiner tree problem classes.
- multi-tree-constrained routing: An example is delay-delay-jitter-constrained multicast routing which is to find a tree with both bounded delay and bounded delay jitter.

A detailed discussion of QoS multicast routing problems can be found in [31].

3.5 QoS Routing Algorithms

3.5.1 Requirements for QoS Routing Algorithms

There are some requirements for a QoS routing algorithm:

- Generality: Multimedia applications tend to have diverse QoS requirements on bandwidth, delay, delay jitter, cost, and so on. From a network design's point of view, it would be beneficial to develop a generic routing algorithm instead of implementing different routing algorithms for different types of QoS requirements independently. The generic algorithm captures the common messaging and computational structure;
- Extensibility: As the network infrastructure evolves and capacity increases, new applications become possible. Therefore, it is important to design extensible algorithms with adaptabilities to new applications. Because with the networks becoming increasingly complex, the deployment of new routing algorithms afterwards will be costly;
- Simplicity: The simplicity of a routing algorithm in terms of time/logical complexity often allows efficient implementation, debugging and evaluation. Simplicity also makes the algorithm easier to be understood, maintained, and upgraded;
- Efficiency and scalability: QoS routing should be efficient and scalable.

It is noted that some requirements conflict with each other. On the one hand, efficient algorithms are needed and the algorithms should be scalable enough that they can be used in the Internet. On the other hand, these algorithms should not be too complicated. Therefore, some tradeoffs must be made between them.

3.5.2 Strategies of Routing Algorithms

Like the traditional routing, QoS routing involves two basic tasks: (1) collecting the state information and keeping it up to date; (2) searching the state information for a feasible path in unicast routing or a feasible tree in multicast routing. In order to find an optimal path/tree which satisfies the constraints, the state information about the intermediate links between the source and destination(s) must be known. Searching for the feasible path/tree greatly depends on how the state information is collected and where the information is stored. According to how the state information is maintained and how the search of a feasible path/tree is carried out, there are three QoS routing strategies [30]: source routing strategy, distributed routing strategy and hierarchical routing strategy. Each strategy has its strengths and weaknesses.

3.5.2.1 Source Routing Strategy

In source routing, each node in a network maintains a complete global state including the network topology and state information of every link. A feasible path/tree that satisfies a set of constraints and possibly an optimization goal is locally computed at the source node based on its global state. Then a control message is sent out along the selected path/tree to inform the intermediate nodes of their precedent and successive nodes. A link-state protocol is used to update the global state at every node.

Source routing achieves its simplicity by transforming a distributed problem into a centralized one. By maintaining a complete global state, the source node calculates the entire path locally. It avoids dealing with distributed computing problems such as distributed state snapshot, deadlock detection, and distributed termination. It guarantees loop-free routes. Many source routing algorithms are conceptually simple and easy to implement, evaluate, debug, and upgrade.

However, source routing has some problems. First, the global state maintained at every node has to be updated frequently enough to cope with the dynamics of network parameters such as bandwidth and delay. This makes the communication overhead excessively high for large-scale networks. Second, the link state algorithm can only provide approximate global state due to the overhead concern and non-negligible propagation delay of the state messages. As a consequence, QoS routing may fail to find a feasible path/tree due to the imprecision in the global state used. Third, the computation overhead at the source is excessively high. This is especially true in the case of multicast routing or when multiple constraints are involved.

3.5.2.2 Distributed Routing Strategy

In distributed routing, the path/tree is computed by a distributed computation. Control messages are exchanged among the nodes, and the state information kept at nodes is collectively used for the path/tree search. Most distributed routing algorithms need a distance vector protocol to maintain a global state in term of distance vector at each node.

Distributed routing achieves its scalability and makes the routing response time shorter by distributing the path/tree computation among the intermediate nodes between the source node and destination node(s). Moreover, in distributed routing, searching paths in parallel for a feasible one is made possible, which increases the chance of success.

Distributed routing algorithms which depend on the global state share more or less the same problems as source routing algorithms have. The distributed algorithms which do not need any global state tend to send more messages. It is difficult to design efficient distributed heuristics for the NP-complete routing problems, especially in the case of multicast routing, because there is no detailed topology and link-state information available. In addition, when the global states at different nodes are inconsistent, loops may occur. A loop can easily be detected when a routed message is received by a node for the second time. However, loops generally make the routing fail because the distance vector principle does not provide sufficient information for an alternative path.

3.5.2.3 Hierarchical Routing Strategy

In hierarchical routing, nodes are clustered into groups which are further clustered into high-level groups recursively, creating a multilevel hierarchy. Each physical node maintains an aggregated global state. This state contains detailed state information about the nodes in the same group and aggregated state information about the other groups. Source routing is used to find a feasible path/tree on which some nodes are logical nodes representing groups. A control message is then sent along this path/tree to establish the connection.

Hierarchical routing scales well because each node only maintains a partial global state where groups of nodes are aggregated into logical nodes. The size of such an aggregation state is logarithmic in the size of the complete global state. Well-studied source routing algorithms can be directly used on each hierarchical level to find feasible paths based on the aggregated state. Hence, hierarchical routing remains many advantages of source routing. It also has some advantages of distributed routing because the routing computation is shared by many nodes.

However, because the network state is aggregated additionally, imprecision is introduced, which has a negative impact on QoS routing [32].

3.5.3 QoS Routing Algorithms

There are many proposed QoS routing algorithms. We give a list of them with their main features. The more detailed content can be found in [30][31] and their references.

3.5.3.1 Unicast Routing Algorithms

Table 1 Unicast routing algorithms

Algorithm	Solving Routing Problem	Routing Strategy	Time Complexity	Communication Complexity	
				Maintaining State	Routing
Wang-Crowcroft [33]	Bandwidth-delay-constrained	Source	*O(vlogv+e)*	Global	*Zero*
Ma-Steenkiste [34]	Bandwidth-constrained	Source	*O(vlogv+e)*	Global	*Zero*
	Multi-constrained	Source	*O(kve)*	Global	*Zero*
Guerin-Orda [32]	Bandwidth-constrained	Source	*O(vlogv+e)*	Global	*Zero*
	Delay-constrained	Source	*Polynomial*	Imprecise Global	*Zero*
Chen-Nahrstedt [35]	Bandwidth-cost-constrained	Source	$O(xve)^3$	Imprecise Global	*Zero*
Wang-Crowcroft [33]	Bandwidth-optimization	Distributed	*O(ve)*	Global	*O(v)*
Salama et al. [36]	Delay-constrained least-cost	Distributed	$O(v^3)$	Global	$O(v^3)^4$
Sun-Landgengorfer [37]	Delay-constrained least-cost	Distributed	*O(v)*	Global	*O(v)*
Cidon et al. [38]	Generic	Distributed	*O(e)*	Global	*O(e)*
Shin-Chou [39]	Delay-constrained	Distributed	*O(e)*	Local	*O(e)*
Chen-Nahrstedt [40]	Generic	Distributed	*O(e)*	Local	*O(e)*
PNNI [41]	Generic	Hierarchical	*Polynomial*	Aggregated	*O(v)*

Note: *v* is the number of nodes; *e* is the number of edges;
x is a constant in the algorithm. A large *x* results in a higher probability of finding a feasible path and a higher over head.
k in the time complexity is the number of all possible residual bandwidth that a link may have.

3.5.3.2 Multicast Routing Algorithms

Table 2 Multicast routing algorithms

Algorithm	Solving Routing Problem	Routing Strategy	Time Complexity	Communication Complexity	
				Maintaining State	Routing
MOSPF [42]	Least-delay	Source	*O(vlogv)*	Global	*Zero*
Kou et al. [43]	Least-cost	Source	$O(gv^2)$	Global	*Zero*
Takahashi-Matsuyama [44]	Least-cost	Source	$O(gv^2)$	Global	*Zero*
Kompella et al. [45]	Delay-constrained Least-cost	Source	$O(gv^3d)$	Global	*Zero*
Sun-Landgengorfer [37]	Delay-constrained Least-cost	Source	*O(vlogv+e)*	Global	*Zero*
Widyono [46]	Delay-constrained Least-cost	Source	*Exponential*	Global	*Zero*
Zhu et al. [47]	Delay-constrained Least-cost	Source	$O(kv^3logv)$	Global	*Zero*
Rouskas-Baldine [48]	Delay-constrained Least-cost	Source	$O(klgv^4)$	Global	*Zero*
Kompella et al [45]	Delay-constrained Least-cost	Distributed	$O(v^3)$	Global	$O(v^3)$
Chen-Nahrstedt [40]	Generic	Distributed	*O(ge)*	Local	*O(ge)*
Note: *v* is the number of nodes; *e* is the number of edges; *g* is the number of destinations; *d* is the delay requirement; *k* and *l* are constants in the algorithm. A large *k* (or *l*) results in a higher probability of finding a feasible tree and a higher overhead.					

3.6 Developments and Research Targets on QoS Routing

The study of QoS routing attracts more and more attention. Two IETF RFCs had been done, one is RFC2386: A Framework for QoS-based Routing in the Internet, and another one is RFC2676: QoS Routing Mechanisms and OSPF Extensions. There are also many routing algorithms available as presented in section 3.5.3. Besides, Apostolopoulos. G.'s group has done a lot of work on it.

3.6.1 Apostolopoulos, G.'s Research on QoS Routing Implemention and Feasibility

3.6.1.1 About QoS Routing Implementation

Apostolopoulos, G. et al. [49] proposed a set of additions to OSPF [50] routing protocol to support unicast QoS routing in IP networks:

- Reclaiming the 'T' bit in OSPF option field as an indicator of router's QoS routing capability and referring to it as the 'Q' bit;

- Adding information about link available bandwidth into link state database and updating it through link state advertisement (LSA);
- Encoding information about link available bandwidth using a new type of services (TOS) field.

Meanwhile, they provided an implementation framework of QoS routing extensions to OSPF in their work [51]. However, a completely functional QoS routing framework which could be used for QoS routing study is still missed.

3.6.1.2 About QoS Routing Feasibility

Apostolopoulos, G. et al. [51] evaluated the overhead incurred by QoS routing extensions needed in OSPF. The main results are:

- The increased processing cost of QoS routing is not excessive, and remains well within the capabilities of medium-range modern processors;
- LSA generation and reception cost is a major cost component in QoS routing;
- Bandwidth consumption associated with LSA traffic is only a small fraction of link bandwidth;
- Memory management of QoS routing table has potential to be improved;
- RSVP can be extended further to support better control of QoS paths.

In another work [52], Apostolopoulos, G. et al. investigated how QoS routing cost components are affected by various parameter settings. They focused on a link state based intra-domain QoS enabled routing protocol which they regarded likely to represent the first setting in which QoS routing is deployed.

They found that the processing of the increased update traffic that QoS routing generates is the dominant contributor to the cost of QoS routing. If left unchecked, it has the potential to severely limit the scalability of QoS routing, and therefore the range of environments where it would prove useful. To cope with this problem, authors used a number of techniques, e.g., coarse update thresholds and hold-timers, to control the volume of update traffic, and therefore its processing cost. Their effect on both cost reduction and performance were explored too. The latter aspect is significant as it determines the tradeoff involved when controlling update traffic. Further cost reduction can be achieved through the use of path pre-computation. They also explored the efficiency of this approach and, in particular, illustrated that while pre-computation typically incurs a small drop in performance when compared to on-demand routing, this difference is small when methods to control the volume of update traffic are also used.

Overall, for the set of configurations that were tested, they found that the cost of QoS routing is reasonable and, within the capabilities of a moderately powerful CPU. This holds true even for fairly large networks which have 400 nodes.

3.6.2 Apostolopoulos, G.'s Research on Improving QoS Routing

3.6.2.1 Higher Level Admission Control

Excessive alternate routing can reduce routing performance in conditions of high load. In order to address this problem, Apostolopoulos, G. et al. [53] proposed a trunk reservation approach as the higher level admission control, which may result in rejecting requests routed over alternate paths during the resource reservation phase, even when there are sufficient resources to satisfy the request. When a reservation is attempted through a node, the quantity $(b_i^{available}\text{-}b_{request})/b_i^{capacity}$ is calculated for the outgoing link *i*, where $b_i^{capacity}$ is the capacity of link *i* and $b_i^{available}$ is the amount of available or residual bandwidth on link *i*. Here, available bandwidth is defined as the difference between the link capacity and the reserved bandwidth on the link. This definition is applied also to the available bandwidth in the rest of this paper. The resource reservation for the request is allowed to continue only if this fraction is larger than a trunk reservation level which depends on the length of the path. If the test fails, the request is rejected. Computing the trunk reservation level based on the request's requirements and the residual capacity of the link allows to reject requests only when they really would have resulted on overloading a link. Having different trunk reservation levels for increasingly longer paths allows to penalize longer path more and better control alternate routing.

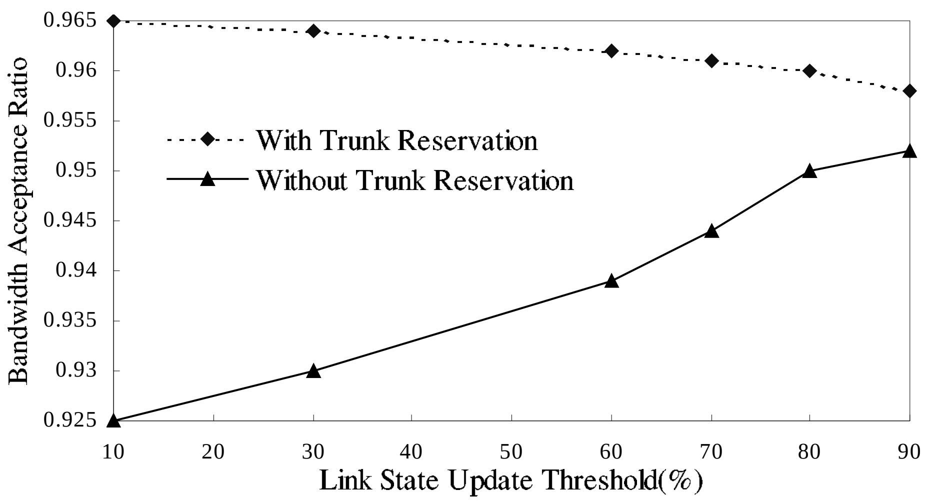

Figure 8 Effects of trunk reservation in the mesh topology [53]

Their experiment (Figure 8) shows that when using higher level admission control routing performance is improved and there is no un-intuitive dependency on the threshold setting. For this experiment, they use bandwidth acceptance ratio as the measure of routing performance, which is defined as the fraction of the offered bandwidth that is successfully routed. Link state will be updated when the relative difference between the current and the previously advertised link state exceeds the threshold. The trunk reservation levels were set to 5% for one hop long paths, 10% for two hop long paths and 20% for all paths longer than two hops.

3.6.2.2 Path Caching

On-demand QoS path computation leads to a simple implementation and reduces storage requirements compared with QoS path pre-computation. But its processing overhead is also high. In order to solve this problem, Apostolopoulos, G. et al. [54] proposed a path caching scheme that can reduce the processing cost of on-demand path computation.

The fundamental idea of this mechanism is to store already discovered paths in a cache. Future requests are routed on-demand only if they can not be routed using the contents of the path cache. This mechanism is controlled by three cache policies:

- cache selection policy: selecting a feasible path from cached paths for a request;
- cache replacement policy: determining which cached path will be replaced by a newly arrived path when the cache is full;
- cache update policy: determining how the cached paths are updated to make sure that the cached paths reflect the current network state to a reasonable degree.

Their work gave a detailed discussion of different cache policies and measures the performance of path caching under different cache policies through simulation-based study. The main results are:

- Path caching is a viable method for reducing the cost of on-demand QoS path computation, at least when a widest-shortest selection policy is used;
- Topology can play an important role in both the processing cost and the routing performance of the different caching policies;
- Small cache sizes are sufficient for achieving good routing performance and processing savings.

3.6.3 Research Targets

Much research work shows the benefits of QoS routing. However, there are still a lot of work for careful study before deploying QoS routing into the Internet. Here are some of the problems that must be solved:

- Performance and cost analysis: The benefits of QoS routing come with the increased cost. Whether or not the benefits are worth the increased cost needs to be answered.
- Efficient routing algorithms: We still lack simple and efficient QoS routing algorithms for the IP networks.
- QoS routing with other QoS techniques: QoS routing is not alone to provide QoS in the Internet. It must work together with other QoS techniques.

In this thesis, we intend to study these problems. We investigate the problem of QoS routing and the relationships between QoS routing and some other QoS components. In particular, we focus on studying the performance and cost of QoS routing through simulation study.

4 Issues on QoS Routing Implementation

In this chapter, we discuss the issues of QoS routing implementation. First we present the position of QoS routing in the QoS framework in section 4.1. Then we discuss the issues of QoS routing implementation in section 4.2.

4.1 The Position of QoS Routing in the QoS Framework

QoS routing is only one component for providing QoS in the Internet. It needs to work together with many other techniques. QoS routing has a close relationship with resource reservation, admission control, link sharing, traffic engineering, IntServ, DiffServ and MPLS.

4.1.1 QoS Routing and Resource Reservation

QoS routing is usually used in conjunction with some form of resource reservation or resource allocation mechanisms. Resource reservation protocols provide the method of requesting and reserving network resources, but they do not provide any mechanisms for determining a path/tree that has adequate resources to accommodate the requested QoS. Conversely, QoS routing can determine a path/tree that has good chance to accommodate the requested QoS, but it does not include any mechanisms to reserve the requested resources.

However, there are opposite views on whether QoS routing and resource reservation should be integrated or separated. The IntServ architecture separates routing from resource reservation, and uses RSVP for receiver-initiated resource reservations for unicast/multicast data flows. By separating routing from resource reservation, the task of network management is eased at the expense of the path/tree located by routing possibly not being QoS-satisfying. On the other hand, combing routing with the resource reservation is argued that it is more likely to locate QoS-satisfying path/tree at reduced connection setup latency. For example, a protocol such as RSVP may be used to trigger QoS routing calculations to meet the needs of a specific flow.

4.1.2 QoS Routing and Admission Control

As mentioned in 3.2, one goal of QoS routing is to optimize the utilization of the network resources and improve the total throughput of the network. Considering this, simply routing a flow to a path that meets the QoS requirements of this flow is not enough. Both the required resources of this flow and the total available resources along the QoS-satisfying path need to be taken into account. If this flow needs too many resources, it may be rejected even if the network has the capability to accept it. A related problem is fairness. Larger flows tend to need more resources while small flows need less. Thus small flows always have a better chance to be accepted. To be fair, a control is needed to guarantee that larger flows can get a certain level of acceptance rate. As a result, admission controls that can help QoS routing to achieve better performance are needed.

4.1.3 QoS Routing and Link Sharing

QoS routing is expected to support best-effort routing as well. This means these two routing schemes should be able to coexist. One question behind is how to allocate network resources between them. Normally, QoS traffic has priority over best-effort traffic. Given a fixed traffic load with QoS requirements, resources available to best-effort traffic will be influenced by the routing of QoS traffic. If a QoS routing scheme ignores the best-effort traffic load or even ignores the presence of best-effort traffic, it will optimize the throughput of QoS traffic but ignore the performance of best-effort traffic. Considering that the best-effort traffic is likely to continue to be the dominant traffic class in the network in the near future, it is not appropriate to improve the performance of QoS traffic while ignoring the performance of best-effort traffic. To solve this problem, some link sharing approaches have been proposed. Among them are static link sharing approach, semi-dynamic link sharing approach and dynamic link sharing approach [55].

4.1.3.1 Static Link Sharing Approach

Static link sharing approach, also called class based queuing (CBQ) [56], proposes that the capacity of a link is statically divided between QoS and best-effort traffic. The key problem is how to determine what fraction of the link capacity should be used for each traffic class because the ratio of QoS traffic over best-effort traffic changes over time. Allocating too much bandwidth for best-effort traffic can cause a high blocking rate for QoS traffic, while allocating too little bandwidth for best-effort traffic can cause serious congestion. Moreover, the traffic in each class may not be evenly distributed. QoS traffic may be concentrated in one part of the network, while best-effort traffic is concentrated in another part of the network. Static sharing policies can not adapt to such an imbalance. However, because of its simplicity, this approach is attractive from the implementation point of view.

4.1.3.2 Semi-Dynamic Link Sharing Approach

Semi-dynamic link sharing approach may be employed to overcome the problems of static link sharing. Under this approach, the measured link utilization for different traffic classes is used to update periodically what fraction of the link capacity is assigned to each traffic class. While this provides some degree of adaptation, this strategy may not work well if there are sudden changes in the utilization. Often these changes are caused by external phenomena which are difficult or impossible to predict and whose effects are acute. Because the semi-dynamic algorithms are reactive rather than pro-active, latencies associated with their adaptive operation may make their reaction time too long.

4.1.3.3 Dynamic Link State Approach

Another approach is to use the measured link utilization of both QoS traffic and best-effort traffic as the link state for both traffic classes. This means that the routing algorithm for QoS traffic considers the load conditions of best-effort traffic, and is likely to route QoS traffic along the links with low utilization. However, link utilization is not always a good indicator of how much bandwidth is available to reserve. For example, best-effort traffic with big bursts can consume all link bandwidth, which leads to high utilization of the link, even though the link in question has reservable bandwidth. The routing algorithm for QoS traffic may thus reject a request even though plenty of reservable bandwidth is available.

4.1.4 QoS Routing and Traffic Engineering

Traffic engineering is the process of arranging how traffic flows through the network to avoid network congestion caused by uneven network utilization. It includes a lot of different mechanisms which are based either on static control or on dynamic control. QoS routing is an important part of traffic engineering, which may find a longer but lightly loaded path better than a heavily loaded shortest path to make network traffic distributed more evenly. Hence it can help make the traffic engineering process automatic.

4.1.5 QoS Routing and QoS Architectures

QoS routing can be used in QoS architectures to better deliver QoS. For example in DiffServ, as discussed in 2.2.2, packets are marked just at the edge routers in a DS domain, it can not solve the congestion inside the domain. As a result, a lot of flows in the same class can be routed through the same link and cause a congestion problem when the link does not have enough resources. QoS routing will be very helpful in this case. Figure 9 illustrates the DiffServ architecture equipped with QoS routing [57].

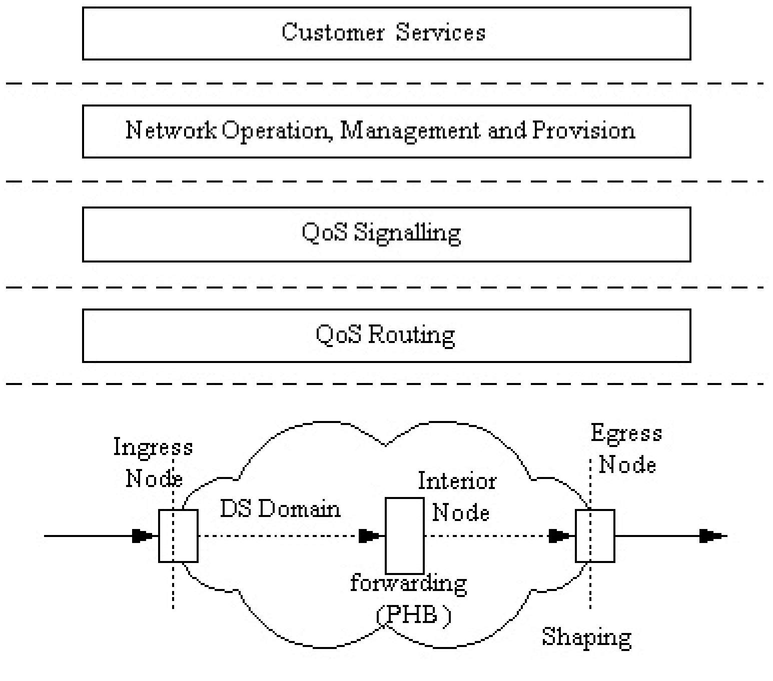

Figure 9 DiffServ architecture with QoS routing

IntServ architecture equipped with QoS routing will be discussed in detail in Chapter 5.

4.1.6 QoS Routing and MPLS

MPLS as a forwarding scheme and QoS routing as a routing scheme are mutually independent of each other. QoS routing determines the path between two nodes based on resource information and topology information. It is useful with or without MPLS. Meanwhile, given a path, MPLS uses its label distribution protocol to set up the LSPs. It does not care whether the paths are determined by QoS routing or other routings. However, when MPLS and QoS routing work together, they make each other more useful. MPLS makes it possible to do QoS routing at traffic trunk granularity without introducing multifield classification to the core routers. MPLS's per-LSP statistics provide QoS routing with precise information about the amount of traffic between every ingress-egress pair. Given such information, QoS routing can better compute the paths for setting up LSPs. In combination, MPLS and QoS routing provide powerful tools for supporting QoS. Zhang, P. and Kantola, R. [25] point out "QoS routing is likely to be used for constructing efficient and high performance MPLS VPNs." Benefits from building MPLS VPNs with QoS routing capability might be achieved in a number of ways:

- QoS routing selects feasible paths by avoiding congested nodes or links;
- If the workload exceeds the limit of existing paths, QoS routing offers multiple paths for transferring additional traffic;
- If a link or node failure occurs, QoS routing selects alternate paths for quick recovery without seriously degrading the quality.

4.2 QoS Routing Implementation

Regarding QoS routing implementation, one important issue is how to collect and update network state information. The performance of QoS routing directly depends on how well this task is solved. How to define QoS routing metrics and when to execute the routing action are the other two factors that can influence the effects of QoS routing. In addition, the level of granularity of QoS routing decision is also an important factor that need to be chosen carefully to further enhance the performance of QoS routing. The target of this section is to elaborate these issues.

4.2.1 Collecting and Updating Network State Information

In a network, each node is assumed to maintain its up-to-date local state, including the queuing and propagation delay, residual bandwidth and the availability of other resources.

The combination of the local states of all nodes in a network is called global state. Every node is able to maintain the global state by either a link-state protocol or a distance-vector protocol, which exchange the local states among the nodes periodically. Link-state protocols broadcast the local state information of every node to every other node so that each node knows the topology of the network and the state of every link. Distance-vector protocols exchange distance vectors among adjacent nodes. A distance vector has an entry for every possible destination, consisting of the property of the best path and the next node on the best path. An example of a distance vector is shown in Table 3, which represents the global state in a distance vector at node s in the network shown in Figure 7.

Table 3 Global state in the form of distance-vector

Node s	Destination	i	j	k	l	t
	Bandwidth	1.5	2	2	2	1.5
	Next Hop	i	k	k	i	i
	Destination	i	j	k	l	t
	Delay	2	3	1	2	4
	Next Hop	i	k	k	k	k
	Destination	i	j	k	l	t
	Cost	1	2	1	3	5
	Next Hop	i	i	k	k	i

The global state information kept by a node is an approximation of the current network state information due to the delay of propagating local states. As the network size grows, the imprecision increases and the size of global state increases also, which leads to a scalability problem.

A common way to achieving scalability is to reduce the size of the global state by aggregating state information according to the hierarchical structure of large networks. In a hierarchical network, nodes are clustered recursively into several level groups. In each level group, a node is selected to represent the group and store the higher-level state information. Thus, each physical node maintains an aggregated network image. The link state algorithm can be extended to collect the aggregated state information for every node [28]. However, aggregating state information achieves scalability while it brings imprecision problem since the imprecision is also aggregated as the state information is aggregated.

Link state protocols and distance vector protocols update state information periodically. The main disadvantage of this approach is that it can not ensure timely propagation of significance changes, and therefore can not ensure providing accurate information for path computation. Ideally, nodes should have the most current view of the network state. Updating state information whenever it changes provides the most accurate information for computing paths. But if the state information changes very quickly from time to time, updating state information for each change will cause a great burden for the network links and routers–consuming much network bandwidth and routers' CPU cycles. One way to solve this problem is to set a threshold to distinguish significant changes from minor changes. And the state information updating is triggered when a significant change occurs. Alternatively, network resources can be partitioned into ranges or classes, the state information updating is triggered for each class boundary crossing. Such methods provide some control on the tradeoff between information accuracy and volume of updates. However, periods of rapid traffic fluctuations may trigger frequent updating and, as a result, cause transient control overloads. To alleviate this problem, a hold-timer can be invited to complement the threshold and the class based triggering methods to enforce a minimum spacing between consecutive updates. One focus of this thesis is to investigate the impact of different state information updating methods on the performance and cost of QoS routing. We will illustrate four different updating methods in detail in Chapter 5.

4.2.2 Routing Metrics and Path Computation

Routing metrics are the representation of a network in routing; as such, they have major implications not only on the range of QoS requirements that can be supported but also on the complexity of path computation.

4.2.2.1 Selection Criteria of Routing Metrics

Normally, defining suitable routing metrics needs to take into account a number of factors [4]:

- The metrics must reflect the basic network properties of interest. Such metrics may include the information of residual bandwidth, delay etc., which make it possible to support basic QoS requirements. Since QoS requirements of flows have to be mapped onto path metrics, the metrics define the types of QoS guarantees that the network can support. Alternatively, QoS routing can not support QoS requirements that can not be meaningfully mapped onto a reasonable combination of path metrics;
- The path computation based on a certain metric or a combination of metrics must not be too complex as to make it impractical. Theoretically, it is hard to compute paths based on certain combinations of metrics (e.g., delay and jitter). Thus the allowable combination of metrics must be determined while taking into account the complexity of computing paths based on these metrics and the QoS needs of flows. A common strategy to allow flexible combinations of metrics while at the same time reducing the path computation complexity is to utilize "sequential filtering", this means that paths based on a primary metric are computed first and a subset of them are eliminated based on secondary metric and so forth until a single path is found;
- Once suitable metrics are defined, a uniform representation of them is required. Particularly, encoding of the maximum, minimum, range, and granularity of the metrics are needed.

4.2.2.2 Single Mixed Metric and Multiple Metrics

A possible routing metric can be one single mixed metric which is defined as the function of multiple parameters. The idea is to mix various pieces of information into a single measure and use it as the basis for routing decisions. For example, a single mixed metric M may be produced by bandwidth B, delay D and loss probability L with a formula $f(p)=B(p)/D(p)L(p)$. A path with a large value of $f(p)$ is likely to be a better choice in terms of bandwidth, delay and loss probability. However, single mixed metric does not contain sufficient information to assess whether QoS requirements can be met or not, hence, it can be only used as a reference indicator.

Multiple metrics can provide more accurate information for routing decisions. But, the problem is that finding a path subject to multiple constraints is not easy, and sometimes, even impossible For example, finding a least-cost path with a delay constraint is regarded as NP-complete [30].

The path computation complexity is primarily determined by the routing metrics which can be divided into three classes.

Let $m(n_1,n_2)$ be a metric for link (n_1,n_2). For any path $P=(n_1,n_2{...}, n_j,n_k)$, metric m is:

- Additive, if $m(P)=m(n_1,n_2)+ m(n_2,n_3)+ ...+m(n_j,n_k)$;
- Multiplicative, if $m(P)= m(n_1,n_2)*m(n_2,n_3) ...*m(n_j,n_k)$;
- Concave, if $m(P)=min\{m(n_1,n_2), m(n_2,n_3),..., m(n_j,n_k)\}$.

Common routing metrics are delay, delay jitter, cost, hop-count, reliability, and bandwidth. It is obvious that delay, delay jitter, cost and hop-count are additive, reliability(1-loss rate) is multiplicative, and bandwidth is concave.

Wang, Z. and Crowcroft, J.[4] proved that finding a path subject to constraints on two or more additive and multiplicative metrics in any possible combination is NP-complete. As a result, the computation that uses any two or more of delay, delay jitter, hop-count, cost, reliability in any combinations as metrics is NP-complete. The computationally feasible combinations of metrics are bandwidth and one of five(delay, delay jitter, hop-count, cost, reliability).

4.2.2.3 Bandwidth and Hop-count as Metrics

Among the common routing metrics in QoS routing, many people believe that bandwidth and hop-count are more useful. This is because:

- Although applications may care about delay and jitter bounds, few applications can not tolerate occasional violation of such constraints. Therefore, there is no obvious need for routing flows with delay and jitter constraints. Besides, since delay and jitter parameters of a flow can be determined by the allocated bandwidth and the hop-count of the path, delay and jitter can be mapped to bandwidth and hop-count constraints if needed;
- Many real-time applications will require a certain amount of bandwidth. The bandwidth metric is therefore useful. The hop-count metric of a path is important because the more hops a flow traverses, the more resources it consumes. For example, a 1Mb/s flow that traverses two hops consumes twice as many resources as one that traverses a single hop.

In addition, algorithms for finding paths with bandwidth and hop-count constraints are rather simple [58]: Bellman-Ford's algorithm or Dijkstra's algorithm can be used. For example, to find the shortest path between two nodes with bandwidth greater than 1Mb/s, all links with residual bandwidth less than 1Mb/s can be pruned first. Then Bellman-Ford's algorithm or Dijkstra's algorithm can be used to compute the shortest path in the pruned network.

4.2.2.4 Path Computation Mode

QoS paths can be either computed on demand or pre-computed. An on-demand approach computes the path for each new request when it arrives. It has the benefit of being able to always use the most recent information. However, if requests arrive too frequently, this approach may show costly even if the algorithm is of relatively low complexity. Pre-computation approach is to compute a QoS routing table in advance. However, since the resource requests are not known in advance, such a routing table needs to pre-compute and store multiple alternative paths to each destination, potentially for all possible values of resource requests. This may show inefficient both in terms of processing and storing if most of the pre-computed paths are not used.

Each computation approach shows its strengths and drawbacks, it is hard to judge which one is better. Both need more investigations.

4.2.3 Routing Granularity

QoS routing can be destination-based, source-destination-based, class-based, traffic trunk-based, or flow-based [22][59].

Destination-based routing, e.g., routing in the current Internet, is based only on the destination address of a packet, then an intermediate router will route all flows between different sources and a given destination along the same path. This is acceptable if the inquired path has adequate capacity, otherwise there will be a congestion problem even though there may be another path that has enough capacity.

Source-destination-based routing, e.g., proposed in [60], which is based on both the source and destination of a packet, implies that all traffic between a given source and destination travels down the same path. It also has congestion problem when the path does not have adequate capacity. In addition, the amount of routing state increases since the routing tables must include source/destination pairs instead of just the destination.

Flow-based routing is based on individual flows, and each flow with QoS requirements is routed separately between any source and destination. Flow-based routing can avoid the congestion problem that other two routing schemes have. An existing flow-based routing is in the IntServ architecture with RSVP. Routing on different granularity levels leads to different performance and cost, routing with coarse granularity is simple, but not efficient in terms of resource utilization. Routing with finer granularity is more flexible, and thus more efficient in terms of resource utilization. However, it incurs a tremendous cost in terms of the routing state.

5 Design of a QoS Routing Simulator

In this chapter, we present our development of a QoS routing simulator (QRS). We begin with making some pre-decisions before the design. Then in section 5.2, we present the general design of our QoS routing framework. Section 5.3 explains the detailed design of a QoS routing protocol. We describe the QRS in section 5.4.

5.1 Design pre-decisions

We intend to design a QoS routing simulator under IntServ architecture, which means all routers in a network have RSVP, admission control or resource management, a packet classifier and a packet scheduler. The design is based on the following pre-decisions/assumptions:

- Working at intra-domain level;
- Support for unicast routing;
- Support for on-demand path computation;
- Using bandwidth as the QoS routing metric and providing delay as an option;
- Providing selectable routing algorithms and link state update (LSU) algorithms;
- Extending existing OSPF as proposed in [49] to the QoS routing protocol, we call it QOSPF;
- Implementing flow-based routing;
- Implementing a simplified RSVP instead of a full set of RSVP;
- Using RSVP to trigger QoS routing calculations;
- Assigning best effort packets into routing table and QoS traffic packets into flow table;
- Creating a simple resource management which only considers QoS requirements;
- Creating a simplified class based queuing (CBQ) [56] to implement packet scheduling.

5.2 QoS Routing Framework under IntServ Architecture

Figure 10 illustrates the framework of QoS routing under IntServ architecture which we intend to implement into the QRS. We explain the basic process of each component and their interrelations in following subsections. But, in the current implementation, the functions of policy control are not specified and are unclear yet. Some proposals can be found in [8].

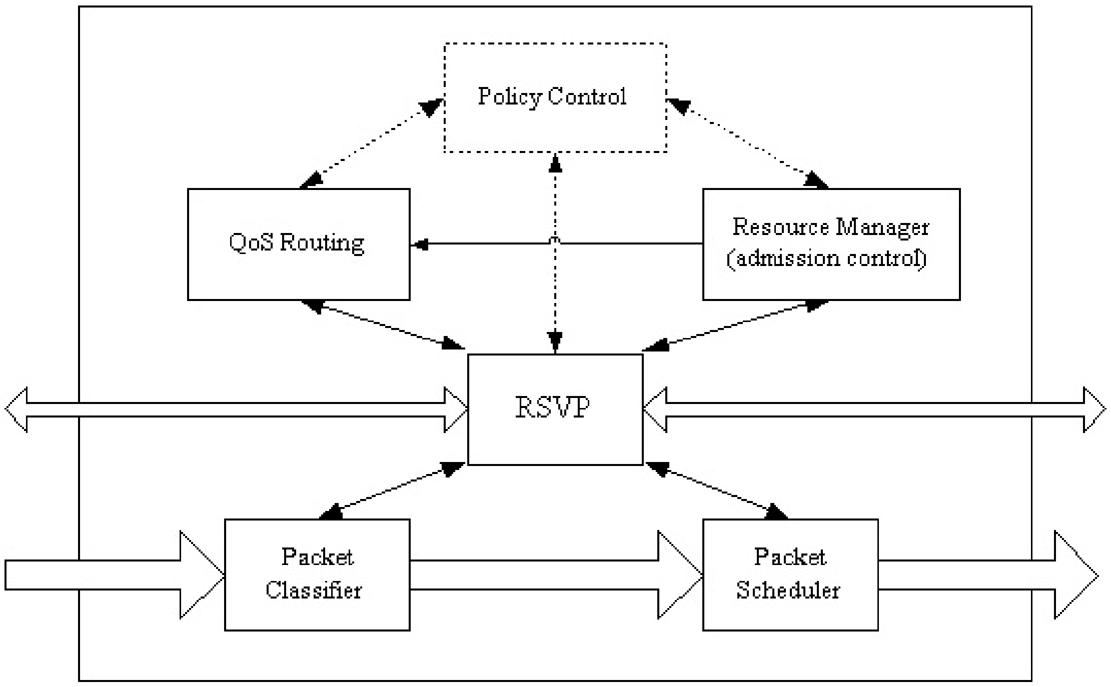

Figure 10 Framework of QoS routing

5.2.1 QOSPF

- QOSPF responds to path request from RSVP, determines the path under the knowledge of network resource availability and flow requirements; and replies the result to RSVP;
- If a message from RM is received, QOSPF updates the local topology database and recalculates paths in the normal routing table, and decides whether to broadcast link state or not; if yes, broadcasts link state to other nodes;
- If a valid LSU message from another node is received, QOSPF updates the local topology database and recalculates paths in the normal routing table, then broadcasts this message to other nodes. Especially, if an adjacent link is down, QOSPF checks up whether any existing flow connections in the flow table use this link, if yes, informs RSVP with an error message.

5.2.2 RSVP

- If a new flow request from source or from another hop is received, RSVP inquires QOSPF about the next hop or informs the destination if the message arrives at its destination;
- If a valid next hop from QOSPF is returned, RSVP sends PATH message to next hop; otherwise, it sends PATH_ERR upstream;
- If a new acknowledgement from destination or another hop is received, RSVP requests RM to reserve resources;
- If a valid success message from RM is returned, RSVP sends RESV message upstream or informs source if message arrives at source; otherwise sends RESV_ERR downstream;

- If a valid PATH or RESV update message is received, RSVP updates PATH or RESV refresh timer;
- If PATH timer expires, RSVP sends PATH_TEAR message downstream or PATH_ERR message upstream and RESV_ERR message downstream;
- If source RSVP receives PATH_ERR, RSVP sends PATH_TEAR downstream;
- If destination RSVP receives PATH_ERR, RSVP sends PATH_TEAR upstream;
- Source RSVP periodically sends PATH messages and destination RSVP periodically sends RESV messages. The period is varied with Gauss function with a mean value.

5.2.3 RM

- If RM receives a reservation request from RSVP, it checks if there is enough available bandwidth in the link, replies the result and informs QOSPF that the amount of available resources has changed if resources are reserved;
- If RM receives a tear down message from RSVP, it updates the available bandwidth and informs QOSPF that the amount of available resources has changed.

On the current stage, RM implements a simple algorithm:

```
If (current available bandwidth – required bandwidth > 0)
    reserve required bandwidth
else reject the request
```

In order to achieve efficient resource utilization, new RM algorithms are expected. For example, RM may consider not only if enough bandwidth is available but also a certain level of available bandwidth remained. An alternative algorithm is:

```
If ((current available bandwidth-required bandwidth)/link bandwidth>threshold)
    reserve required bandwidth
else reject the request
```

The value of threshold can be defined by the user or calculated depending on other factors as discussed in section 3.6.2.

5.2.4 Packet Classifier

When a data packet arrives at a node, it goes into the packet classifier before being forwarded. If it belongs to a specified flow and its flow connection exists, it is indexed in the flow table for next hop, otherwise it is indexed in the routing table for the next hop. Then the packet queues at the buffer to the corresponding link. According to the class the packet belongs to, it queues at the corresponding buffer in CBQ. And it is served by CBQ and transmitted to the next hop.

5.2.5 Packet Scheduler

We design a CBQ traffic scheduling in the following way.

- It derives from Priority Queuing (PQ) in which the queue with higher priority is always served first;
- One queue is served only if other queues with higher priority are empty or regulated;
- Once a packet in the queue is served, the CBQ performs Exponential Weighted Moving Average (EWMA) and determines the regulated state and the under limited state;
- If the buffer is full, the consecutive packets belonging to this class will be discarded directly;
- A hierarchical link sharing structure with three levels of traffic classes as shown in Figure 11 is deployed.

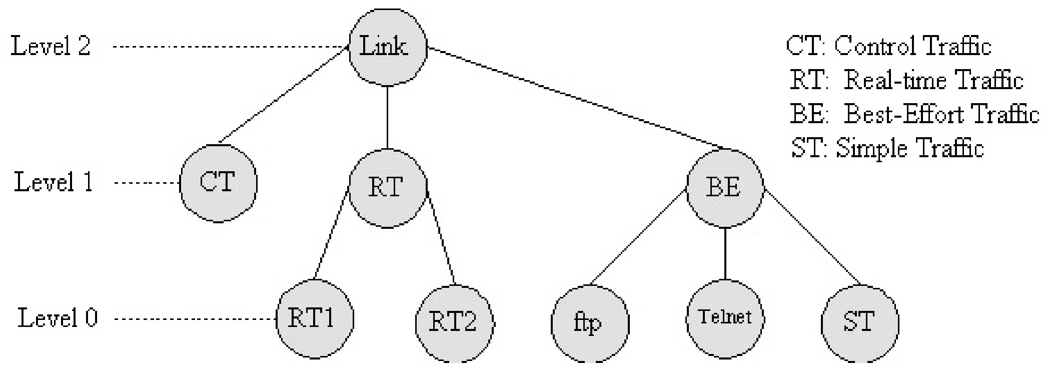

Figure 11 Levels of traffic classes in CBQ

A CBQ can be configured by setting its parameters, i.e., allocated bandwidth, buffer size, EWMA weight and EWMA threshold. A detailed discussion of CBQ can be found in [56].

5.3 Core Functions of QOSPF

A typical QoS routing protocol should at least consist of three core functional components as shown in Figure 12:

1. Distribution of resource availability information ;
2. Topology database with resource information;
3. QoS path computation.

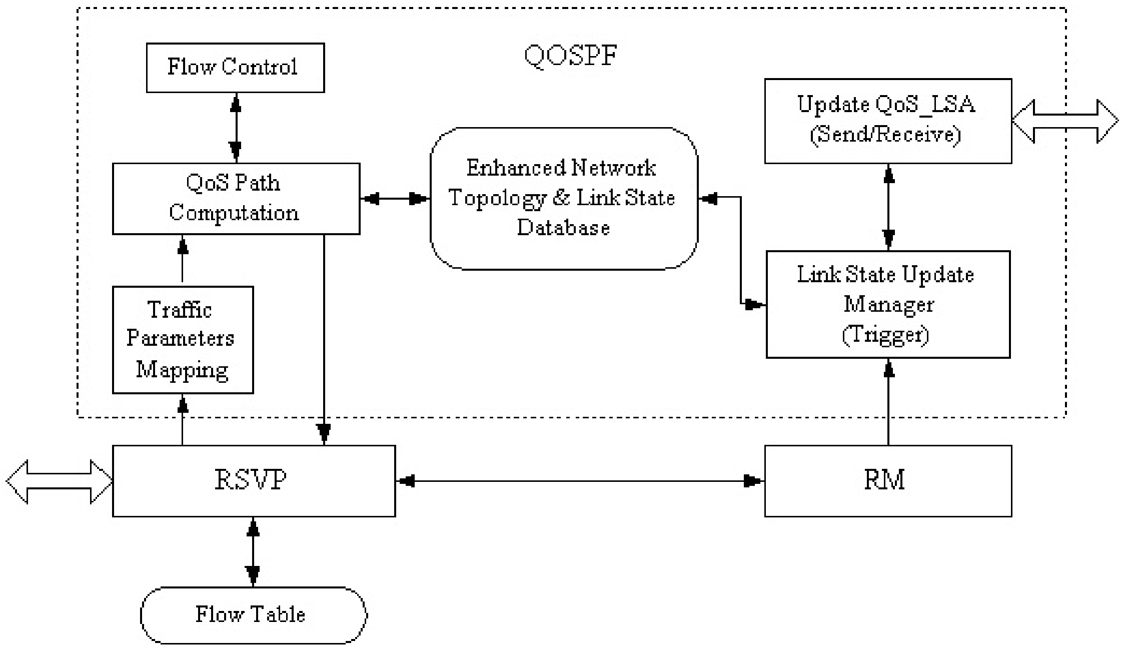
QOSPF
Flow Control
QoS Path Computation
Traffic Parameters Mapping
Enhanced Network Topology & Link State Database
Update QoS_LSA (Send/Receive)
Link State Update Manager (Trigger)
RSVP
RM
Flow Table

Figure 12 Design of QOSPF

5.3.1 Distribution of Resource Availability Information

One basic requirement of QoS routing is tracking the network state information (i.e., available link bandwidth) so that the state information is available to the path computing. Link State Advertisements (LSAs) flooded by OSPF already carry administrative cost metrics for each link, and there is a provision for advertising multiple cost metrics using Type of Service (TOS) fields. TOS fields can be effectively used to encode information such as available bandwidth and propagation delay associated with a particular link [49]. However, while the distribution of updates can be accomplished with minor modifications to OSPF, additional mechanisms are still needed to determine when updates should be sent. In particular, routers need not only to track available bandwidth on their interfaces, but also determine at what time the bandwidth has changed sufficiently to warrant a new path. The latter one is particularly important as it plays a major role in both the protocol overhead and the performance of QoS routing.

The decision of when to broadcast LSUs is the responsibility of a triggering function. The design of a triggering function involves various tradeoffs between performance and cost as discussed in 4.2.1. So far, many LSU algorithms have been proposed, in this intra-domain QoS routing protocol we provide four algorithms, they are period based (PB), threshold based (TB), equal class based (ECB), unequal class based (UCB).

The basic idea of PB algorithm is to update the whole topology periodically. A constant timeout is set to determine when the network states should be updated. This algorithm provides a direct control over the communication overhead, but does not ensure timely propagation of significant changes especially when the timeout is assigned a big value.

The idea of TB, ECB and UCB algorithms is that the scope of a node's update extends to all its incident links, that available bandwidth values for all the interfaces of the node are advertised even when the update is triggered by just one link. It is also in compliance with the behavior of routing protocols such as OSPF [56] that only generate LSUs on all the links attached to a node. In addition, TB, ECB and UCB attempt to trigger an update only when the current available bandwidth of a link differs significantly from the previously advertised value.

- **TB**: In this algorithm, a constant threshold value (*th*) is set. For an interface *i* of a node, let bw_i^o be the last advertised value of available bandwidth, and bw_i^c is the current value, an update is triggered when $(|bw_i^o - bw_i^c| / bw_i^o) > th$ for $bw_i^o > 0$. For $bw_i^o = 0$, an update is always triggered. This algorithm tends to provide more detailed information when operating in the low available bandwidth range and becomes progressively less accurate for larger value of available bandwidth.

- **ECB**: In this algorithm, a constant *B* is set, which is used to partition the available bandwidth operating region of a link into multiple equal size classes: *(0,B),(B,2B),(2B,3B),...,etc.* An update is triggered when the available bandwidth on an interface changes so that it belongs to a class that is different from the one to which it belonged at the time of the previous update. It has the same degree of accuracy for all ranges of available bandwidth.

- **UCB**: In this algorithm, two constants *B* and *f* $(f>1)$ are set, which are used to define unequal size classes: $(0,B)$, $(B,(f+1)B)$, $((f+1)B, (f^2+f+1)B)$, $((f^2+f+1)B, (f^3+f^2+f+1)B)$,..., etc. Unlike the equal class based algorithm, the class sizes grow geometrically by the factor *f*. An update is triggered as before, i.e., when a class boundary is crossed. This policy has fewer and larger classes in the high available bandwidth operating region and more and smaller classes when available bandwidth is low. Consequently, it tends to provide a more detailed and accurate state description for the low bandwidth region.

Besides, in TB, ECB and UCB, we also install a periodical LSU trigger in case of deadlock, and provide an option of hold-timer that can be used to enforce a minimum spacing between consecutive updates. For example, a hold-timer may be combined to TB, under this combination, an update is triggered only when the degree of link state changing is more than the threshold and the difference between the current time and last link state advertising time is more than the value of hold-timer. This mechanism avoids the high frequency of update in a very dynamical environment.

5.3.2 Topology Database with Resource Information

The standard OSPF protocol provides each router with a complete network map which is stored in a topology database. We simply enhance the local topology database by including available resource information i.e., available bandwidth of each link, and keep the best-effort metrics unchanged in the topology database, so that the best-effort and QoS routing can readily coexist within the same router.

5.3.3 QoS Path Computation

As discussed earlier, QoS paths can be either on-demand computed or pre-computed. We choose on-demand computation in our intra-domain QoS routing protocol. Because firstly, on-demand computation is simple since it only involves traversing the topology database and determines a single QoS-satisfying path. Secondly on-demand computation can provide accurate QoS-satisfying paths since it is able to always use the most recent information. In addition, we note that there is not yet any implemented prototype of on-demand QoS routing and we could not find an overall study on its performance and cost in the literature. Our goal is to fill this blank in this study.

We implement two alternative QoS routing algorithms: lowest cost algorithm and widest bandwidth algorithm. Consider a directed graph $G=(N,E)$ with numbers of nodes N and numbers of edges E, in which edge (i,j) is weighted by two parameters: b_{ij} as the available bandwidth and c_{ij} as the cost. Let $b_{ij}=0$ and $c_{ij}=\infty$ if edge (i,j) does not exist in the graph. Given any directed path $p=(i,j,k,\ldots,l,m)$, define $b(p)$ as the bottleneck link bandwidth of the path, i.e., $b(p)=min[b_{ij}, b_{jk},\ldots, b_{lm}]$ and define $c(p)$ as the sum of the cost, i.e., $c(p)=c_{ij}+c_{jk}+\ldots+c_{lm}$. Given two nodes s and d of the graph and two constraints B and C, to the lowest cost algorithm, the QoS routing problem is to find a path p^* between s and d so that $b(p)\geq B$ and $c(p)\leq C$; to the widest bandwidth algorithm, the QoS routing problem is to find a path p^* between s and d so that $b(p)\geq B$ and the path has the widest bandwidth and if there are more than one widest paths, the path with the lowest cost is selected.

Let C_i be the estimated cost of the path from node s to node i and let B_i be the estimated bandwidth of the path from node s to node i.

- **Lowest cost algorithm:**

```
Step 1: Set $c_{ij}=\infty$, if $b_{ij}<B$;
Step 2: Set $L=\{s\}$, $C_i=c_{si}$ for all $i\neq s$;
Step 3: Find a node $k\notin L$ so that $C_k=min_{i\notin L}C_i$;
        If $C_k>C$, no such a path can be found and the algorithm terminates
        If $L$ contains node $d$, a path is found and the algorithm terminates
        $L:=L\cup\{k\}$
Step 4: For all $i\notin L$, set $C_i:=min[C_i, C_k+c_{ki}]$;
Step 5: Go to Step 3.
```

Step 1 eliminates all links that do not meet the bandwidth requirement by setting their cost to ∞. Steps 2-5 find the least cost path from node s to node d using Dijkastra's algorithm. We do not have to find the least cost paths to all nodes. The algorithm can be terminated either when node *d* is included by *L* or the cost exceeds the threshold before reaching node *d*.

- **Widest bandwidth algorithm:**

```
Step 1: Set $b_{ij}=0$, if $b_{ij}<B$;
Step 2: Set $L=\{s\}$, $B_i=b_{si}$ for all $i \neq s$;
Step 3: Find set $K$, $K \cap L = \Phi$, so that $B_k = max_{i \notin L} B_i$;
Step 4: If $K$ has more than one element
            Find a node $k \in K$, so that $cost\ of\ path(s,\ldots,k,i) = min_{j \in K}[C_{(s,\ldots,k,i)}]$
            $L := L \cup \{k\}$
        If $L$ contains all nodes, the algorithm is completed
Step 5: For all $i \notin L$, set $B_i := max[B_i, min[B_k, b_{ki}]]$;
Step 6: Go to Step 3.
```

Stepl eliminates all links that do not meet the bandwidth requirement by setting their available bandwidth to 0. Steps 2-6 find the widest bandwidth path from node *s* to node *d* by using a variation of Dijkstra's algorithm.

In the lowest cost algorithm, the link cost is computed by a link cost function which is the same as in [61]. In the current version, four link cost functions are supported:

1. Hop-count: An operational link cost is 1, otherwise is infinity;
2. Utilization: It is a fraction of the time that the outgoing link queue is not empty in the last link cost update period;
3. Delay: It represents the average delay (over all packets transmitted in the last cost update period) suffered by a packet on the outgoing link queuing and in transmission (including the processing time at the node and the propagation delay of the link);
4. Hop-normalized Delay: It is a utilization measure calculated from the average queuing delay per packet and the average transmission delay per packet over the last link cost update period.

5.4 The Implementation of the QoS Routing Simulator

A discrete-event QoS Routing Simulator (QRS) is developed in [8] from Maryland Routing Simulator (MaRS) [62][63] by adding and modifying some components for handling QoS routing. Specifically, RSVP, Resource Management, and Realtime Traffic Source/Sink components are added, SPF component is extended to QOSPF and Node and Link components are modified. QRS can be used to investigate issues of QoS routing in the Internet.

5.4.1 Architecture

The QRS architecture is shown in Figure 13. The functions of QOSPF, RSVP and RM are the same as we described in section 5.2. Functions of packet classifier and traffic scheduler described in section 5.2 are performed by node and link components respectively. Realtime Traffic Source/Sink is used to model QoS traffic with bandwidth requirements.

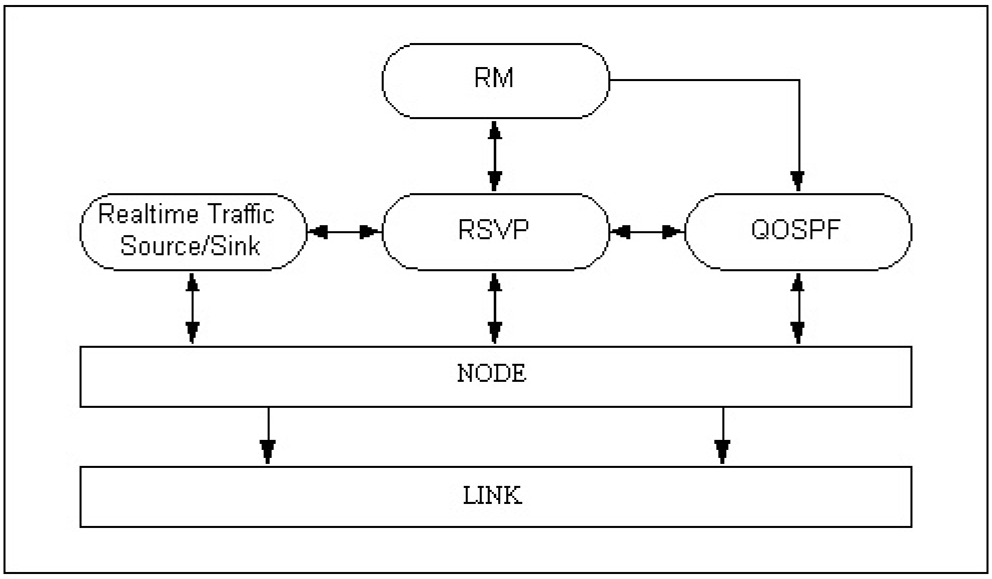

Figure 13 Design and interactions between components of QRS

During a successful connection setup of a flow, the interactions among these components are:

At the beginning of the connection, Realtime Traffic Source requests RSVP for the flow connection setup, RSVP then queries QOSPF for the next hop, if a valid next hop is returned, RSVP sets the PATH state in the flow table and sends a PATH message through Node and Link to RSVP at the next hop. Next RSVP also asks QOSPF for the next hop, when the PATH message finally arrives at RSVP at the destination, RSVP informs Realtime Traffic Sink about the flow using the PATH message.

If Realtime Traffic Sink wants to set up the flow, it answers RSVP to set up the flow. Then, RSVP will first ask RM if enough resources are available. If RM replies yes, RSVP sets the RESV state in the flow table and sends a RSVP RESV message upstream to the previous hop. RSVP at the previous hop also asks local RM for resource reservation.

Finally, when the RESV message arrives at RSVP at the source, RSVP replies Realtime Traffic Source the success of the connection setup. Then, Realtime Traffic Source starts to send data packets through Node and exact Link according to the next hop in the flow table.

When an intermediate node receives the data packets, it first checks the class type of the data packet. If it belongs to a flow for which a connection exists, the data packet will be forwarded according to the next hop in the flow table; otherwise, it is forwarded according to the path in the normal routing table. After the data packet arrives at the destination, the Realtime Traffic Sink consumes it and updates statistics.

If a request of a flow connection setup fails, this request will be repeated after a certain period of time. This period value can be defined by the user.

5.4.2 Service Classes

In QRS, we specify four kinds of flows and assign them to different workload types as shown in Table 4. Class A is the highest priority for control and signaling traffic, i.e., RSVP traffic and route traffic; class B and C are for real-time traffic services, they have higher and lower priority respectively; Class D is the lowest priority for best-effort traffic services, i.e., FTP, Telnet and simple traffic workload. Traffic with higher priority will be served before traffic with lower priority in accordance with the CBQ scheduling algorithm.

Table 4 Service classes in QRS

	Class A	Class B	Class C	Class D
Workload	RSVP, Route	Realtime Traffic	Realtime Traffic	FTP, Telnet, Simple Traffic
Route Selection	RSVP: Flow Table Route: Topology Table	Flow Table	Flow Table	Routing Table
Priority	Highest	Higher	Lower	Lowest

5.4.3 Realtime-Traffic Source/Sink

Realtime Traffic Source/Sink pair is used to generate QoS traffic used in simulation. By configuring its parameters, various QoS traffic flows can be modeled. The main parameters are [64]:

- Flow index: Each pair of Realtime Traffic Source/Sink is indexed by the flow index number. In current version of QRS, the flow index should be an integer within [0, 100]. Different pairs of Realtime Traffic must have different flow index numbers;

- Class type: The parameter corresponds to the class of the traffic, i.e., Class A, B, C, and D. Value1 refers to Class B while value 2 refers to C. The class type of Realtime Traffic can only be 1 or 2;
- Flow rate [bytes/sec]: The parameter specifies the bandwidth requirements of the Realtime Traffic flow;
- Flow delay: The parameter specifies the delay requirements of the Realtime Traffic flow. Currently, it is not used;
- Flow Starting: The parameter specifies the time between the simulator starts up and Realtime Traffic begins to request for path setup;
- Flow Producing: The parameter specifies the active period (ON) of the Realtime Traffic;
- Flow Pausing: The parameter specifies the inactive period (OFF) of the Realtime Traffic.

If a Realtime Traffic is configured to x seconds of flow starting, y seconds of ON, and z seconds of OFF, it works during simulation time as follows:

It requests a connection setup at x seconds after simulation started. After the success of the setup, it sends packets for y seconds, then terminates. After waiting z seconds, it requests connection setup again. This process repeats till the end of the simulation. In the case the connection setup fails, it will re-request after a certain period of time that can be defined by the user.

6 QoS Routing Performance and Cost Analysis

In this chapter, we analyze the QoS routing performance and cost. First we give a general discussion. Section 6.1 presents the parameters which can possibly affect QoS routing performance, and the significance of the effect from each parameter. In section 6.2, we explain path calculation cost, link state update cost and storage cost, as well as their contribution to QoS routing cost. In section 6.3, we use QRS to simulate on-demand QoS routing aiming to study its performance and cost under different network conditions set by varying network topology or size, LSU algorithm and network traffic model. In simulations, we first collect the values of variables we are interested in, and then process them. The processed data is used to investigate the performance and cost of QoS routing.

6.1 QoS Routing Performance

QoS routing performance is usually described in terms of the utilization of network resources or the network throughput achieved by network traffic. It is affected by such parameters:

- Routing algorithm: Comparing with simple routing algorithms, complicated routing algorithms can consider more requirements and provide more accurate and efficient results, but they may consume more time.
- LSU algorithm: QoS paths are computed based on the routing information. With more accurate routing information, the routing algorithm can produce more accurate QoS paths. The choice of LSU algorithm directly affects the accuracy of routing information, and therefore affects the performance of QoS routing.
- Network traffic model: QoS routing attempts to improve network utilization by diverting traffic to paths that could not be discovered by best-effort routing. This may be impossible if all paths are equally loaded. QoS routing might be more useful and more effective in the environments where traffic and network capacity are mismatched and alternative paths with lower load exist.
- Network topology and size: Certain network topologies and sizes may better suit with some routing algorithms than with some other algorithms, thereby affect their performance.
- High level admission control: High level admission control may reject a request if admitting the request will lead to inefficient use of network resources even when a feasible path has been found. Therefore the choice of high level admission control may affect the QoS routing performance significantly.

6.2 QoS Routing Cost

QoS routing cost mainly includes path computation cost, LSU cost and storage cost.

6.2.1 Path Calculation Cost

QoS paths can be on-demand computed or pre-computed. On-demand path computation involves traversing topology database and determining a single QoS path. The cost of a single path computation depends mainly on the network topology and the relative distance between the source and the destination. Size of the request and the levels of available bandwidth on network links will affect the number of computational steps, and therefore the cost. Overall, the total cost of on-demand path computation depends on both the cost of individual computations and their frequency, i.e., the arrival rate of new requests.

In contrast, path pre-computation is mostly insensitive to the frequency of new requests, and primarily depends on how often the QoS routing table is recomputed, which, as opposed to the arrival rate of new requests, is a parameter that the router can control. Clearly, frequent re-computations provide more accurate QoS paths, and therefore better network performance. But this comes at the cost of a substantial increase in processing load. Building a complete QoS routing table is typically more complicated than computing a single path and it further involves the additional cost of deallocating and reallocating memory. In addition, when paths are pre-computed, an additional step of path selection is required to retrieve a suitable path when an incoming request needs to be routed. The suitable path is retrieved from the QoS routing table by searching column by column. As the routing table gets bigger, the cost of retrieving becomes more expensive.

In general, no matter which kind of path computation mode is concerned, the computation cost is influenced by the following two factors:

- Path computation criteria: e.g., lowest cost, widest bandwidth path, etc. Using multiple path computation criteria leads to more sophisticated computing and therefore more cost than using single computation criteria. The tradeoff is between the cost which mainly comes from finding QoS-satisfying paths and the computational complexity of the routing algorithm;
- Flexibility in supporting alternate path selection choices: e.g., maintaining equal cost choices and selecting among them, accounting for inaccuracy in network state information, etc. In general, the unavoidable inaccuracy in the network state information has implications for the path computation process. In particular, it is desirable to maintain and alternate between several choices in order to avoid being stuck with a single bad choice caused by inaccurate information. Similarly, relying on strict cost minimization criteria may not be justified when cost information is not accurate. In such cases, relaxing the optimization criteria of the path selection process may produce more robust solutions. However, allowing such options can affect the overall computational cost.

6.2.2 Link State Update Cost

LSU cost includes two parts: one is the cost of generating and receiving LSAs, the other one is the traffic cost, that is bandwidth consumption associated with LSA traffic.

Generating and receiving an LSA involves accessing the link state database to extract information or insert newly received information. In addition, the generation of an LSA requires that a packet is assembled and transmitted. And transmitting the LSA packet consumes link bandwidth. Both kinds of cost primarily depend on how often the LSA is triggered. LSA trigger algorithms described in 5.3.1 are used to control the frequency of LSA, thus to control LSU cost.

Besides, network topology and the characteristics of requests influence the volume of update traffic also. The former determines the actual number of LSA packets that are exchanged on each link, the latter affects the level of activity in the network, i.e., number and duration of requests, and consequently the frequency of update generation.

Research work [57] shows that cost of generating and receiving LSAs is a major cost component in QoS routing and bandwidth consumption associated with LSA traffic is only a small fraction of link bandwidth even with the more frequent updates that QoS routing requires. Therefore, the rest of this paper ignores the LSU traffic cost.

6.2.3 Storage Cost

Comparing to best-effort routing, QoS routing imposes additional storage requirements. First, the topology database needs to store more information such as link resource availability information. Second, when path pre-computation is used, the QoS routing table itself requires additional storage. The magnitude of this storage overhead depends to a large extent on specific implementation details such as the exact data structures used. However, it is also affected by parameters of QoS routing, such as the operation of triggering policy, which can influence the number of distinct paths that the path computation algorithm generates. For example, there may exist, for each destination, a large number of distinct paths with incrementally different bandwidth values, hence contributing to a larger QoS routing table.

However, it is not expected that QoS routing will drastically change the storage requirements of a router. And it is found [58] that QoS routing has only a small impact on this factor, in particular, the increase in size of the topology database is minor. And although the QoS routing table, when used, can be large, it is not a problem for the storage capabilities of modern systems. As a result, we ignore the storage cost in the rest of this paper. A detailed discussion of the storage cost of a specific implementation of QoS routing, and its comparison to that of best-effort routing, can be found in [57].

In addition to above three QoS routing costs, there are some other costs related to the implementation of QoS routing such as signaling cost, software cost, operation cost and maintenance cost, which are not covered in this thesis.

6.3 Simulation Study

6.3.1 Simulation Objectives

We simulate QoS routing in different network environments in order to:

- examine the basic features of QoS routing, i.e., finding paths that meet QoS requirements; accommodating best-effort traffic; improving the network throughput; indicating disruptions to the current path of a flow and building a new path if one exists;
- find the factors that affect performance and cost of QoS routing;
- investigate how to improve the performance of QoS routing while keeping the cost of QoS routing on an acceptable level;
- gain practical experience for designing and implementing totally functional QoS routing.

6.3.2 Measurement

Performance

We use the average network throughput achieved by real-time traffic with bandwidth requirements to represent the network performance. Theoretically, the larger the average network throughput is, the better the network performance should be. To get the average network throughput, we log the number of received packets in real-time traffic sinks during the simulation, then calculate the average throughput: $\Sigma_i(N_i*L_i)/t$, where N is the number of packets which are received by real-time traffic sinks, L is the packets' size, t is the simulation time and i represents the type of a packet.

Cost

We use total processing time consumed by QOSPFs during the simulation time to represent the cost of QoS routing. The longer the processing time is, the higher the cost will be. To get the cost of QoS routing, we log the time consumed by QOSPF in every node during the simulation time, and then simply calculate the sum.

The processing time of each action of QOSPF in a node is set as shown in Table 5.

Table 5 Cost of each QOSPF action

No.	Cost (us)	Action
1	1500	Find the next hop which can accept the required bandwidth
2	100	Check a message from RSVP and decide what to do next
3	1500	Compute the QoS path
4	500	Update the local topology database
5	200	Broadcast the link state information
6	100	Broadcast a message packet
7	1000	Compute normal routing table for best effort traffic

The time consumed by action 1, 3 and 7 specified in Table 5 should not be a constant, it depends on the network size. But in our work, we define it to be a constant for limiting the amount of programming work and for observing easily the impact of LSU on QoS routing cost, which is regarded as the major cost component in QoS routing.

6.3.3 Simulation Results for QoS Routing Performance

In order to study the QoS routing performance in different network environments, we simulate QoS routing in three different network topologies, i.e., a simple tree topology with four nodes, a matrix topology with nine nodes and NSFNET backbone topology. In all the simulated networks, every link is symmetric, and its propagation delay is 1 millisecond, its capacity is 6Mb/s. All nodes have adequate buffer space for buffering packets awaiting processing and forwarding.

6.3.3.1 Simulation 1

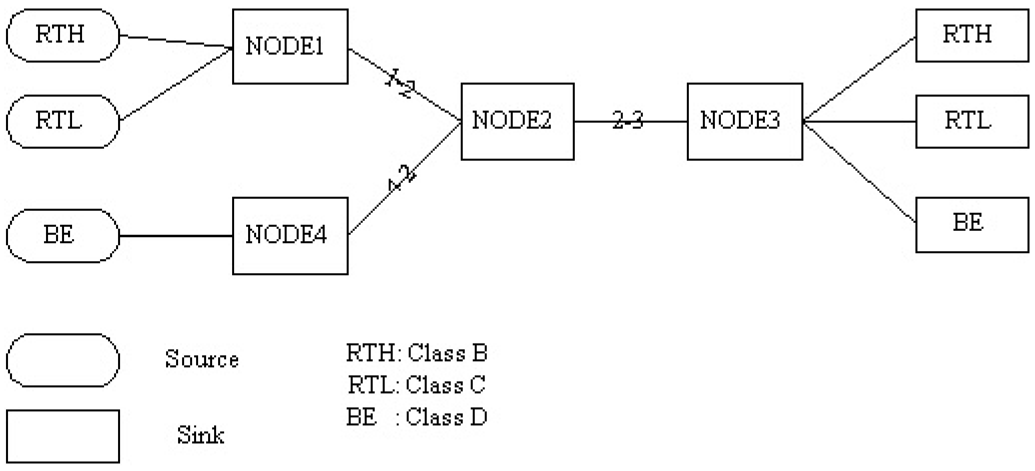

Figure 14 Tree topology

We construct a network with a tree topology, as shown in Figure 14, with four nodes and three links. We configure three pairs of workload: one pair of Realtime Traffic (RT) with Class B, called RTH; one pair of RT with Class C, called RTL and one pair of simple traffic (ST) which presents best-effort traffic (BE). Both RTH and RTL are injected into node 1 and ST is injected into node 4 and both escape from node 3. They all go through link 2-3. The rate of RTH is 2Mb/s, ON is 30s, OFF is 15s. The rate of RTL is 3Mb/s, both ON and OFF are 20s. The rate of BE is set at 6Mb/s. Thus, the total rate of all workloads is larger than the bandwidth of link 2-3, 6Mb/s.

We run the simulation for 100 seconds and log the throughput of each type of workload and then calculate the total throughput. The result is shown in Figure 15, the total throughput is nearly equal to the link bandwidth, that is, the link efficiency is very high. Meanwhile, RTH and RTL can obtain the bandwidth they required, while BE utilizes the remaining bandwidth of the link.

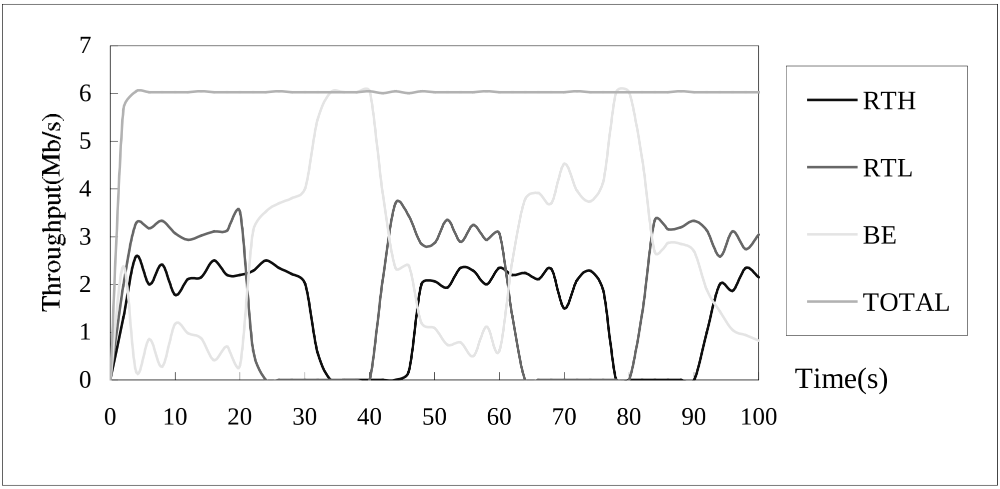

Figure 15 Throughput of real-time traffic and best effort traffic

6.3.3.2 Simulation 2

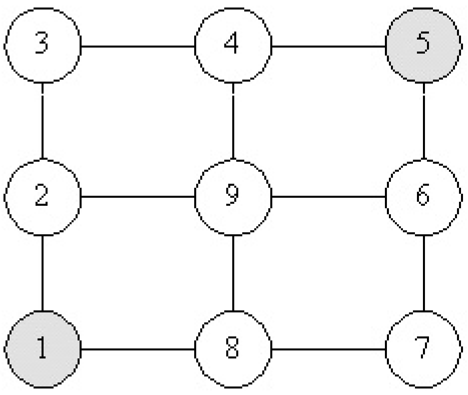

Figure 16 Matrix topology with 9 nodes

On the matrix topology with nine nodes as shown in Figure 16, we install four pairs of RT workload from node 1 to node 5. Their rates are 1Mb/s, 2Mb/s, 3Mb/s, 4Mb/s respectively, then the total rate is 10Mb/s, more than a single link bandwidth. That means they can not be transmitted through one link at the same time. Their OFFs are set to zero.

We run the simulation for 100 seconds. The throughput of each RT pair and their total throughput are shown in Figure 17. This result shows that all RT pairs get the bandwidth they required and that the total throughput is nearly 10Mb/s. That means traffic is being transmitted through more than one path.

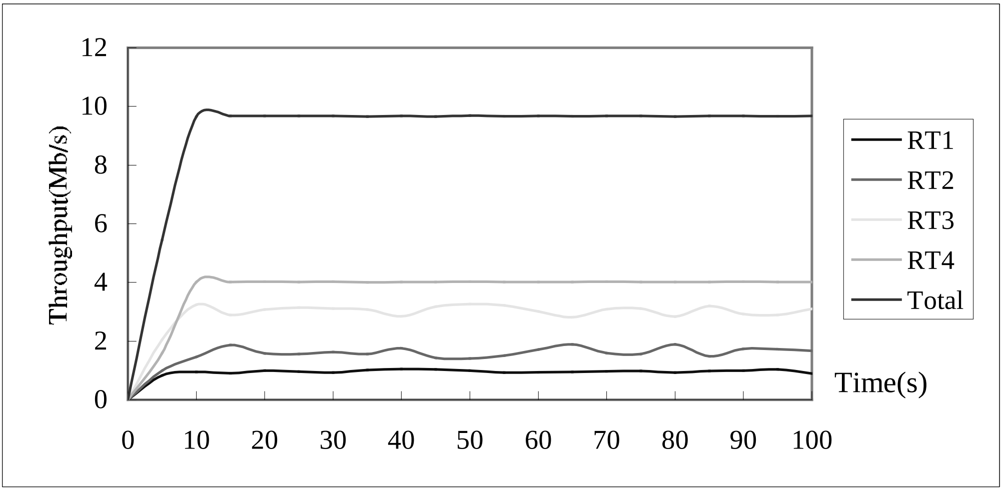

Figure 17 Throughput of real-time traffic with bandwidth requirements

In the next step, we use four pairs of ST with the same rates to replace the pairs of RT respectively. We run the simulation for 100 seconds. Each ST pair's throughput and their total throughput are shown in Figure 18.

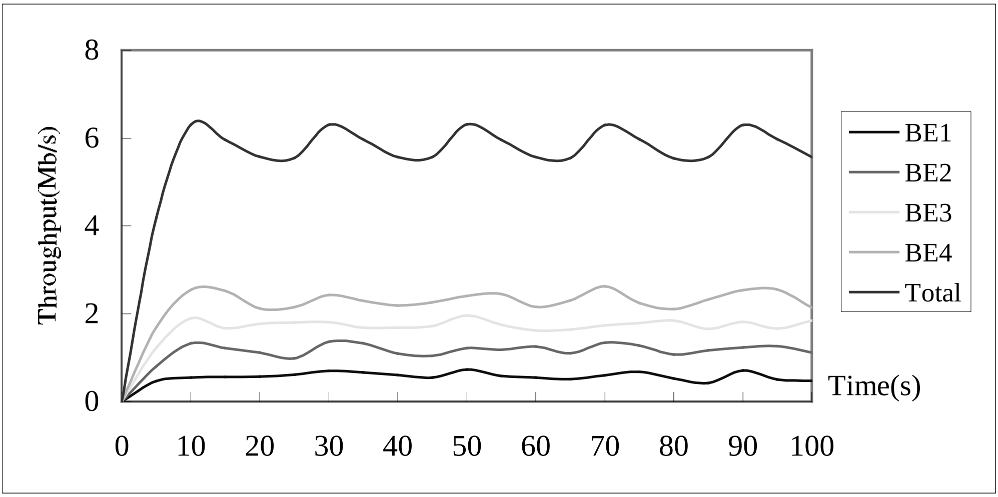

Figure 18 Throughput of simple traffic without bandwidth requirements

This result shows that not all ST pairs get the bandwidth that they need and that the total throughput is only around 6Mb/s.

In the above two simulations, we also record the link utilization of link 4-5(U_{4-5}) and the link utilization of link 6-5(U_{6-5}). The results show that in the first simulation, U_{4-5} and U_{6-5} are stable all the time, but in the second simulation they oscillate. Specifically in the second simulation, when one link utilization changes from 1 to 0, another one always changes from 0 to 1. This result shows that RTs are routed into two paths. STs are routed into one path and shifted back and forth between the two paths. We run the two simulations several times and get similar results. The difference is that U_{4-5} and U_{6-5} for RTs have different values in different simulations. This is caused by the feature that QoS routing dynamically determines the path. For example, RTs with 1Mb/s and 4Mb/s may be routed into one path or two different paths.

6.3.3.3 Simulation 3

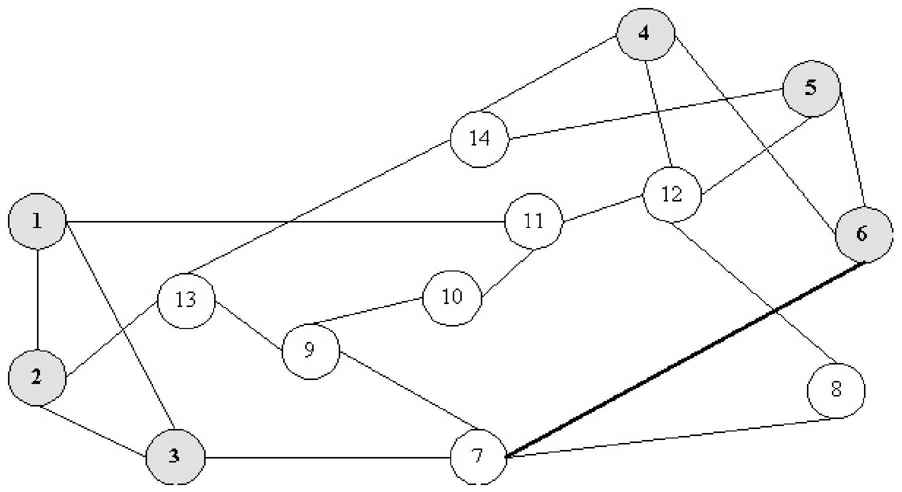

Figure 19 NSFNET-T1 backbone topology

In this simulation, we use NSFNET-T1 backbone as the network topology as shown in Figure 19. We configure: one RT workload from node 1 to each node of 4, 5 and 6; one RT workload from node 2 to each node of 4, 5 and 6; one RT workload from node 3 to each node of 4, 5 and 6. Totally there are 9 RT workloads. In addition, we configure other three ST workloads between 1 and 4; 2 and 5; 3 and 6. The average rate of each workload is set to 2Mb/s. Link 7-6 (dark line in Figure 19) represents the failed link during the simulation time. In our configuration, each RT source starts up randomly. Once the previous path connection fails, each RT source requests for setting up a new path after a mean value of 100ms. Thus, the path for each workload might change each time. We set the link failure at second 4 and link repair at second 7, then run the simulation for 10 seconds. Furthermore, we define the recovery time for a RT workload as the time between that the link fails and that the workload finds a new path and establishes a connection again. We trace the path for each RT workload and record the recovery time for the workload whose path contains the failed link.

The result is illustrated in Figure 20, two pairs of workloads (3-4 and 3-6) are able to establish the new paths and send packets again around second 6 after link failure at second 4. We run the simulations for several times and get the mean value of the recovery time, which is nearly 2 seconds.

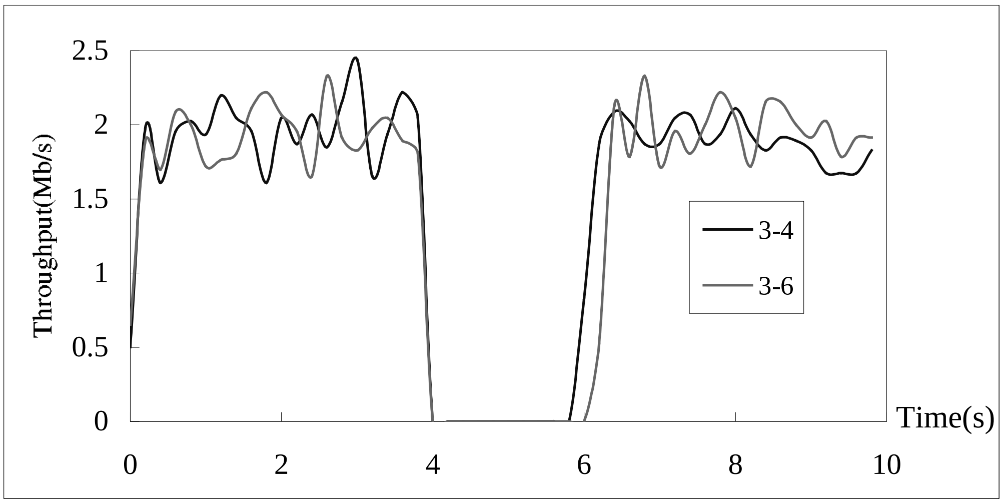

Figure 20 Path recovery in Simulation 3

6.3.3.4 Summary of Simulation Results for QoS Routing Performance

These simulation results show that QoS routing can:

- find paths for flows, which meet the requirements of flows;
- accommodate BE flows without any resource reservation requirements;
- improve the network throughput by alternative path routing;
- indicate disruptions to the current path of a flow due to topological changes and build a new path if one exists.

6.3.4 Evaluation Environment for Performance and Cost of QoS Routing

We aim to evaluate the performance and cost of QoS routing under various network environments. Particularly, we focus on a certain period of time of operational network, in which a number of requests are handled. By recording the average network throughput and the consumed time, we can observe how the performance and cost changes with different factors. We are interested in four factors: network size, LSU algorithm, traffic model and routing algorithm. But, in this thesis, we only study the first three factors, and not study routing algorithm even though we already implemented two alternative routing algorithms as discussed in section 5.3. This is because studying routing algorithms is a big task and could be an independent topic.

6.3.4.1 Network Topology

We simulate QoS routing in two different topologies: matrix topology and ISP topology. In both topologies, all link propagation delays are 1 millisecond, all links are symmetric and have the same bandwidth of 6Mb/s, and all nodes have adequate buffer space for buffering packets awaiting processing and forwarding. Besides, non of the links fail during simulation.

Matrix Topology

A key factor that affects performance and cost of QoS routing is network size (the number of nodes and links in the network). In order to assess how performance and cost varies with network size, we relay on a sample matrix topology with 4 nodes. By systematically changing the sample matrix size, we can measure the corresponding change of performance and cost. Specifically, we use four matrix topologies whose sizes vary from 2*2 to 5*5 as shown in Figure 21.

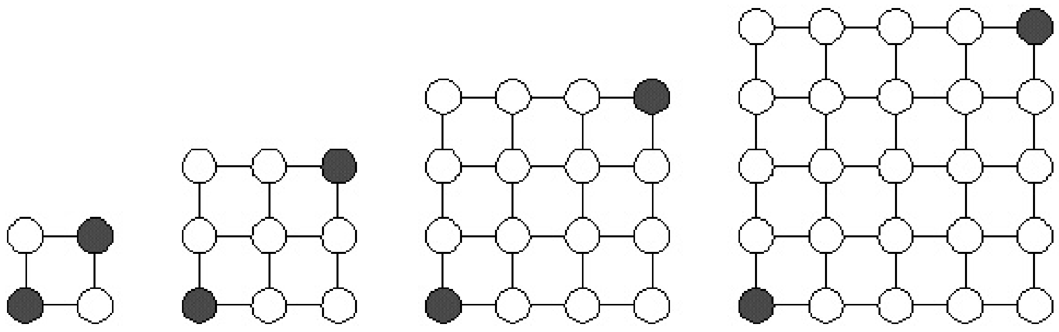

Figure 21 Matrix topologies

ISP Topology

The matrix topology has a number of characteristics that are not really representing the typical topologies. So in order to better assess the impact of different LSU algorithms and traffic models on routing performance and cost, we use more realistic ISP topology as shown in Figure 22, which is widely used in the study of QoS routing [53] .

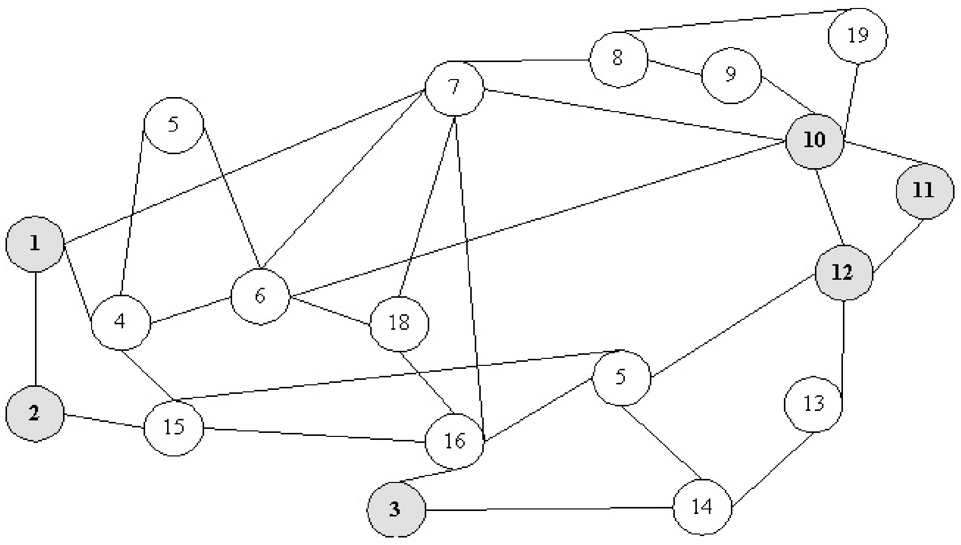

Figure 22 ISP topology

6.3.4.2 Traffic Model

We use Realtime Traffic Source/Sink to model real-time traffic (RT) in terms of requests for setting up connections with specific bandwidth requirements as the QoS traffic. A request is characterized by its source, destination, requested bandwidth, active period (ON), inactive period (OFF), etc., as described in 5.4.3. The requested bandwidth is set uniformly or non-uniformly from 0.1Mb/s to 3Mb/s in different simulations. We will state the main parameters of a RT pair in each simulation.

We use ST, FTP and Telnet to model traffic without resource reservation requirements as the background traffic as opposed to RT. Background traffic is installed to be able to fill all incident links of the concerned nodes when there is no RT.

6.3.4.3 Routing Algorithm

Either lowest cost algorithm or widest bandwidth algorithm can be configured into the simulator. The link cost can be defined in different ways. Here, we only use lowest cost algorithm with the hop-normalized delay cost function.

6.3.4.4 LSU algorithm

There are four LSU algorithms and one option on the hold-timer which can be used together with those algorithms described in 5.3.1. We use different algorithms and a variety of combinations of the threshold based algorithm and the hold-timers in our simulations in order to observe how different LSU algorithms and the hold-timer can influence the performance and cost of QoS routing.

6.3.5 Simulation Results on Performance and Cost of QoS Routing

As discussed earlier the total cost of on-demand QoS path computation mainly depends on the frequency of connection setup requests. High request frequency may lead to a "worst case scenario"–much higher QoS routing cost. We concentrate on this kind of scenario, since the cost produced by such cases could be an upper limit of the cost with other simple QoS routing solutions. The "worst case scenario" is achieved by setting smaller values to ON and OFF of each RT in our simulations.

In all the following simulations, a number of RT pairs and background traffic are installed into the simulated networks. Every RT pair has a unique index, and a class type of either B or C which is set randomly. All RT pairs start requiring connection setup randomly. All simulations are run for 100 seconds. During the simulation, if a request of connection setup fails, it will re-request after 100ms.

6.3.5.1 Impact of Network Size

For each matrix topology in Figure 21, we install 27 RT pairs between the diagonal nodes (black nodes). All pairs have the same source and destination. Each flow rate is set to 0.5 Mb/s. The ONs and OFFs of flows are randomly set to a value from 0.1s to 0.3s. Then the average workload of the RT pairs in a network is about 7Mb/s. In the worst case that all pairs start requesting connection setup at the same time, there will be a need of 13.5Mb/s, therefore some requests will be refused since the maximum bandwidth from the source to the destination is 12Mb/s.

LSU algorithms used in simulations are: PB(100ms), TB(20%), ECB(10%) and UCB(10%/2). We show the simulation results in Figure 23 and Figure 24.

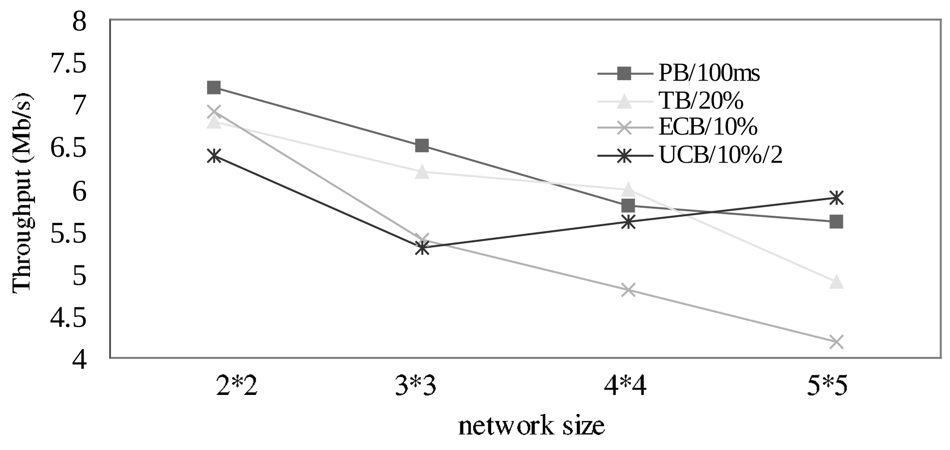

Figure 23 Performance in matrix topologies under different LSU algorithms

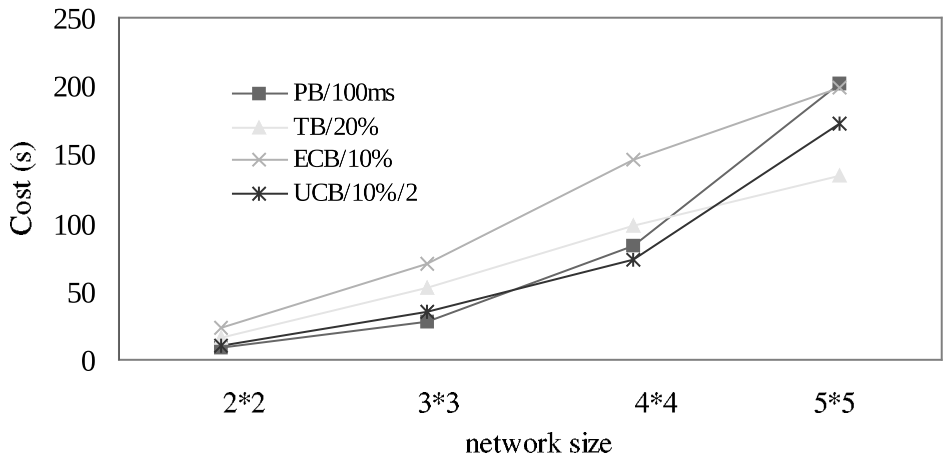

Figure 24 Cost in matrix topologies under different LSU algorithms

The results show that no matter which kind of LSU algorithm is used, the cost of QoS routing increases rapidly along with the increase of network size. The result also show that the way of different LSU algorithms effecting on performance and cost differ with network size. For example, from Figure 23 and Figure 24, we can see that when network size is 2*2 and 3*3, PB (100ms) involved simulation provides the best result with least cost and best performance comparing with other three algorithms. However when network size reaches 4*4, 5*5, neither best performance nor least cost is produced by PB, in particular, it is the most expensive one under size 5*5.

Another phenomenon of interest to us is that even though the cost increases with network size growing in all simulations, the degree of network size impact is significantly different with different algorithms. Figure 24 shows TB/20% involved simulation presents the modest curve , which means that comparing with other three algorithms, TB/20% is least sensitive to network size under the simulation environment we set.

Moreover, the results show that the performance generally has a modestly decreasing trend with network size increasing. A special case is from UCB involved simulation from which the best performance result is under network size 2*2, then 5*5, 4*4, the worst one is under the size 3*3. We repeat the simulation several times with UCB algorithm, but the results we get are always different. At this stage, we regard that this presents the complexity of UCB which is operated by two variables B and f instead of only one as in other algorithms. Further study is needed in order to get more knowledge of the characteristics of this algorithm.

6.3.5.2 Impact of LSU Algorithm

In the ISP network as shown in Figure 22, we configure: 7 RT pairs from node 1 to each node of 10, 11, 12; 7 RT pairs from node 2 to each node of 10, 11, 12; 7 RT pairs from node 3 to each node of 10, 11, 12. Totally there are 63 RT pairs. From the ISP network, we can see that the minimal cut (7-8, 7-10, 6-10, 5-12 and 13-12) has five links with total capacity 30Mb/s(5×6Mb/s). Obviously, the total network throughput achieved by RT should be at most 30Mb/s.

All RT pairs' ONs and OFFs are set randomly from 1s to 3s and 0.1 to 0.3 respectively. If a request of connection setup fails, it will re-request after 100ms. We construct two different traffic models, one is called uniform traffic (UT) model in which all RT pairs have the same flow rate, the other one is called non-uniform traffic (NT) model in which not all RT pairs have the same flow rate. For the UT, the workload of every real-time traffic flow is set to be 0.5Mbps. The average of total workload is about 27Mb/s. For the NT, the workload of each real-time traffic flow is distributed randomly from 0.1Mbps to 3Mpbs. The average of total workload is about 28Mb/s.

We repeat the simulation with different algorithms and sort the results into a chart.

Simulation results under different PB values in the ISP network

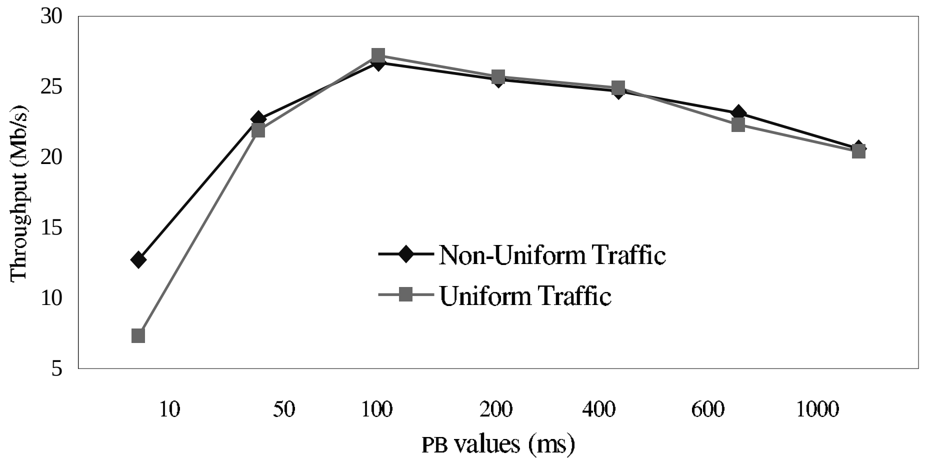

Figure 25 Performance under NT/UT with different PB values

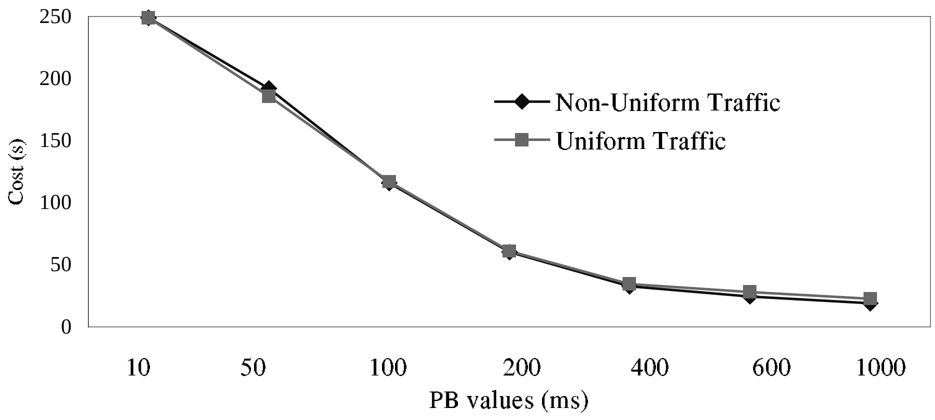

Figure 26 Cost under NT/UT with different PB values

The results recorded in Figure 25 and Figure 26 illustrate that with the PB algorithm, the traffic model has a minor effect on performance and almost has no effect on the cost. This is because, PB updates the whole topology states periodically, not according to the actual situation of network traffic. Therefore, in the same network topology, as long as the PB value is kept unchanged, no matter which traffic model the network has, the QoS routing cost keeps unchanged.

The results also show that small PB values (e.g.,10ms and 50ms) lead to bad performance and high cost. This is because the small PB values lead to high frequency of LSU which directly leads to a high cost. On the other hand due to the delay of the propagation of broadcasting LSU information, the high frequency of LSUs causes the state information to become unstable for path computation, which can be the reason of bad performance.

However, referring to Figure 25, with a suitable PB value (100ms), the performance is almost perfect and achieves the average rate of RTs. Furthermore, the QoS routing cost drops significantly with the increased PB values (e.g., from 100ms to 400ms) while the performance drops smoothly. Further increase of the PB value (e.g., from 400ms to 1000ms) causes both cost and performance to drop smoothly.

Simulation results under different TB values in the ISP network

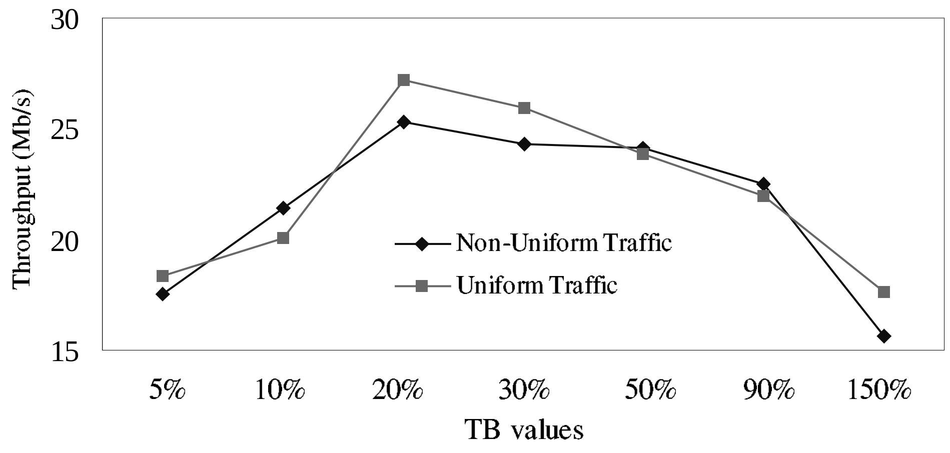

Figure 27 Performance under NT/UT with different TB values

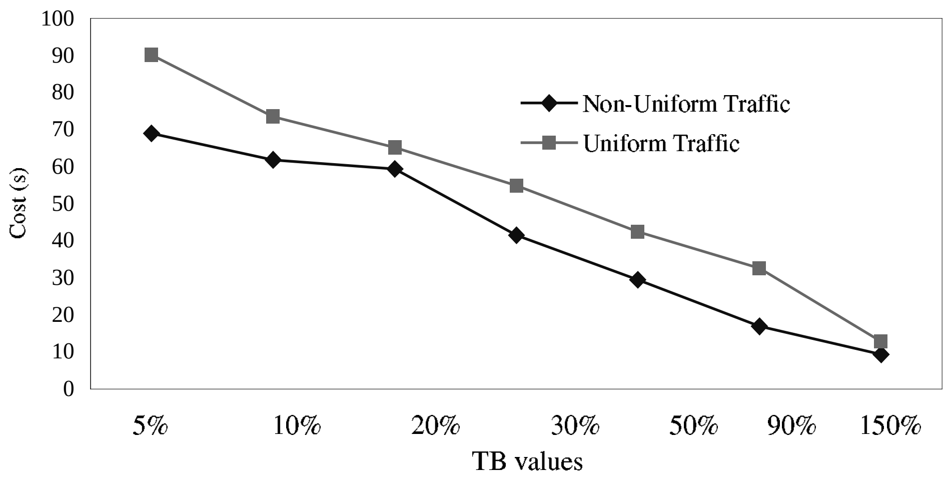

Figure 28 Cost under NT/UT with different TB values

Figure 27 and Figure 28 show that with the TB algorithm, different traffic models result in different performance and cost. The cost in UT is higher than in NT, but there is no exact pattern to compare their performance differences.

Under both UT and NT models, the cost drops sharply with the increase of the TB value, but the best performance on the other hand is not associated with the highest cost, it happens at 20% of TB.

Simulation results under different 20% TB hold-timers in the ISP network

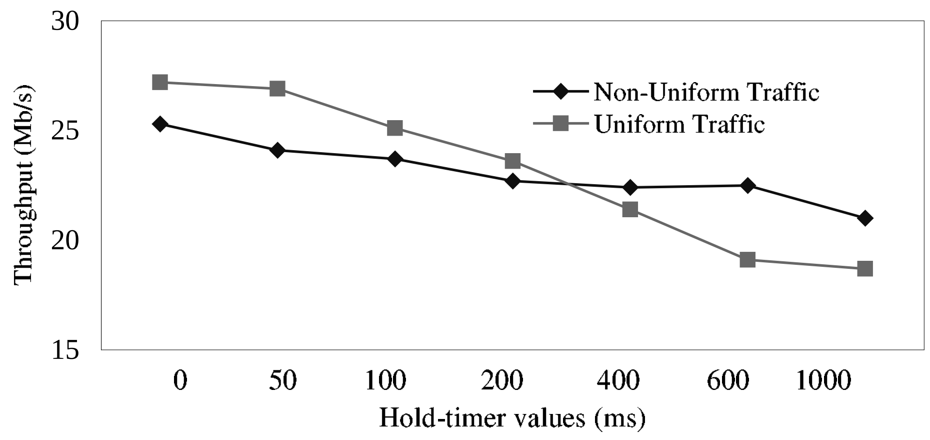

Figure 29 Performance under NT/UT with different 20% TB hold-timers

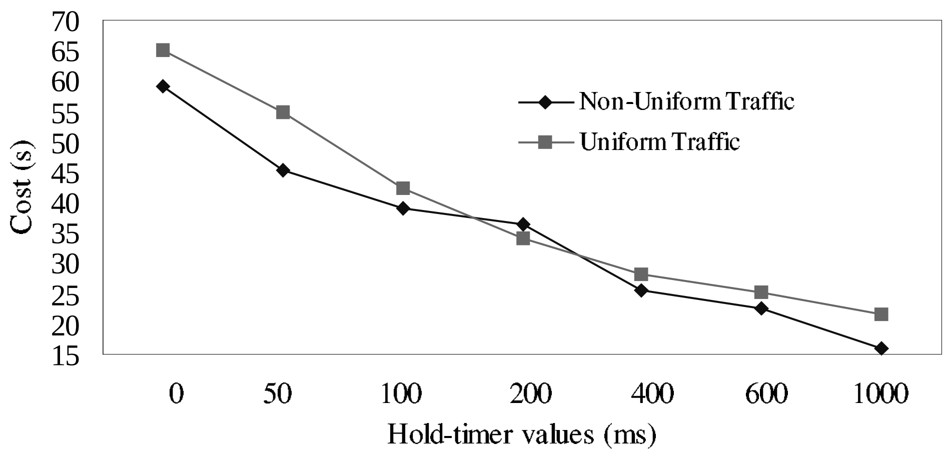

Figure 30 Cost under NT/UT with different 20%TB hold-timers

Figure 29 and Figure 30 imply that with 20%TB algorithm, the hold timer value has great influence on cost, but minor impact on performance. With the increase of hold-timer value, the cost drops sharply under both NT and UT models. This result reflects the role of hold-timer in the cost of QoS routing.

Simulation results under different UCB values in the ISP network

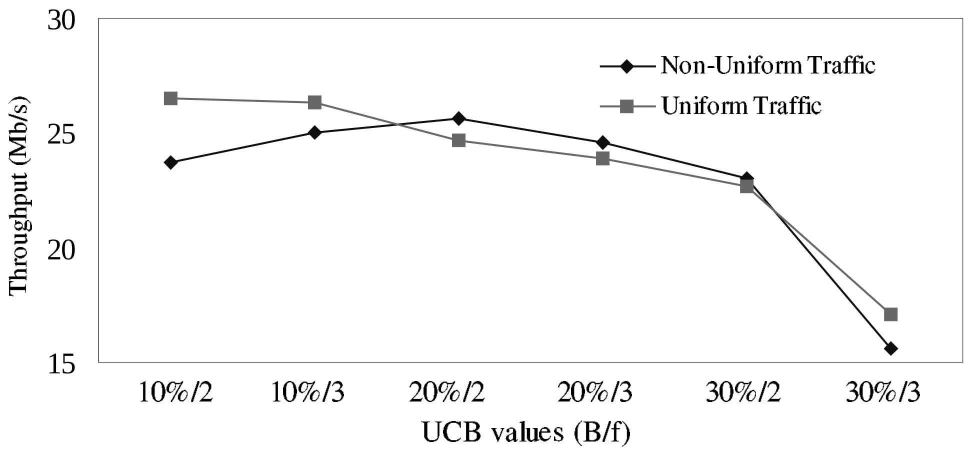

Figure 31 Performance under NT /UT with Different UCB Values

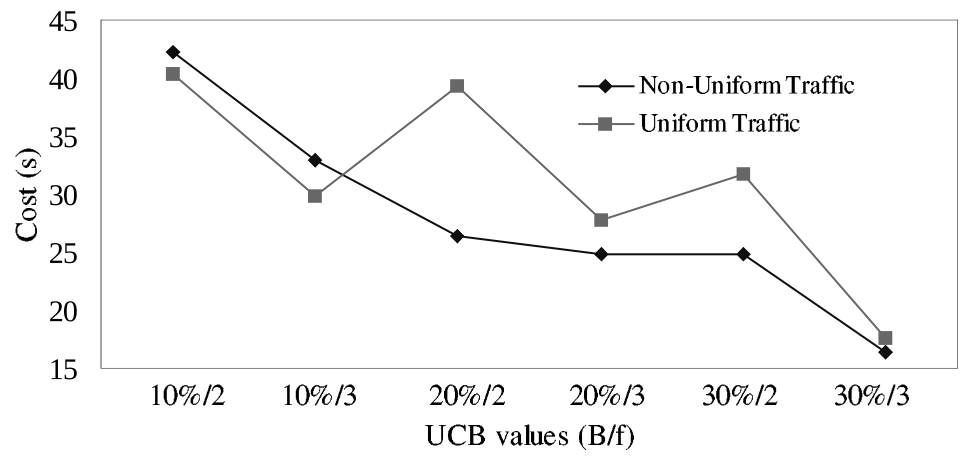

Figure 32 Cost under NT /UT with Different UCB Values

Figure 31 and Figure 32 show that with the UCB algorithm, changing the values of B and f, both the performance and cost show irregular changes. We regard that this shows again the complexity of the UCB algorithm with two controlling parameters.

By further comparing the UCB with the TB, we observe a similar performance result. (For example the performance with TB 20% in Figure 27 and UCB 10%/3 in Figure 31), but the cost produced by UCB is much lower than the cost produced by TB (Figure 28 and Figure 32). This implies that UCB with two controlling variables is more flexible than TB with only one adjustable factor. We deduce that with the optimum B and f values, UCB will possibly be able to achieve the best result in performance and cost combination comparing with other algorithms. But more study is needed before we could make a more convincing conclusion.

From the above four simulation results, we find the best performance of QoS routing achieved with each LSU algorithm is above 25Mb/s. Since the capacity of each link is 6Mb/s, so this result means that at least 5 paths are used to transport real-time traffic simultaneously during the simulations. By analyzing the ISP network, we are convinced that this result can not be achieved by a best effort routing scheme.

Reference to the ISP network topology shown in Figure 22, if we use a best effort routing scheme, for example SPF, the shortest path from node 1 to nodes 10, 11, 12 is on links 1-7 and 7-10, the shortest path from node 2 to nodes 10, 11, 12 is either on links 2-1, 1-7, 7-10 or on links 2-15, 15-5, 5-12; the shortest path from node 2 to nodes 10, 11, 12 is either on links 13-16, 16-5, 5-12 or on links 3-14, 14-13, 13-12. Totally, there are only three paths that can be used simultaneously, and the total best throughput can be only 18Mb/s, far below 25Mb/s QoS routing achieved in our simulations.

6.3.5.3 Impact of the Size of Requests

In the ISP network as shown in Figure 22, we configure: some pairs of RT from node 1 to each node of 10, 11 and 12; some pairs of RT from node 2 to each node of 10, 11 and 12; some pairs of RT from node 3 to each node of 10, 11 and 12. We use TB LSU algorithm and the value of TB is set to 30%. If a request of the connection setup fails, it will re-request after 100ms.

The offered request rate and the real network throughput are shown in Figure 33. The cost is shown in Figure 34.

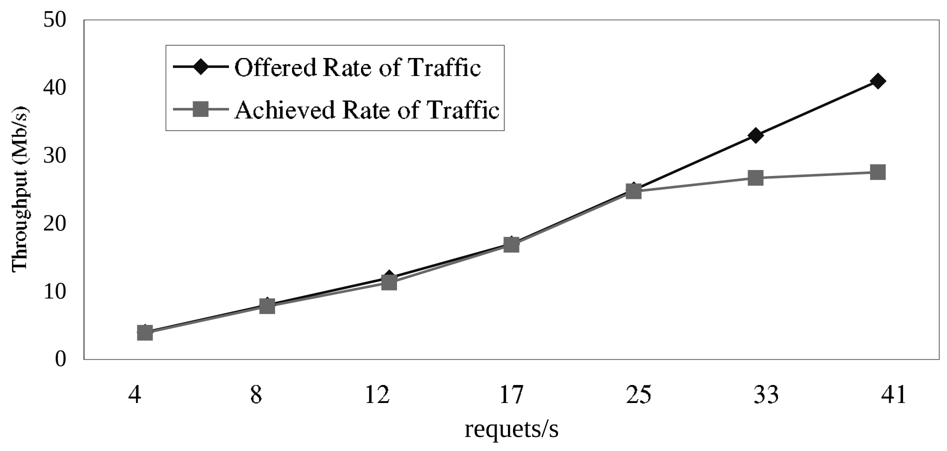

Figure 33 Performance under different request frequencies

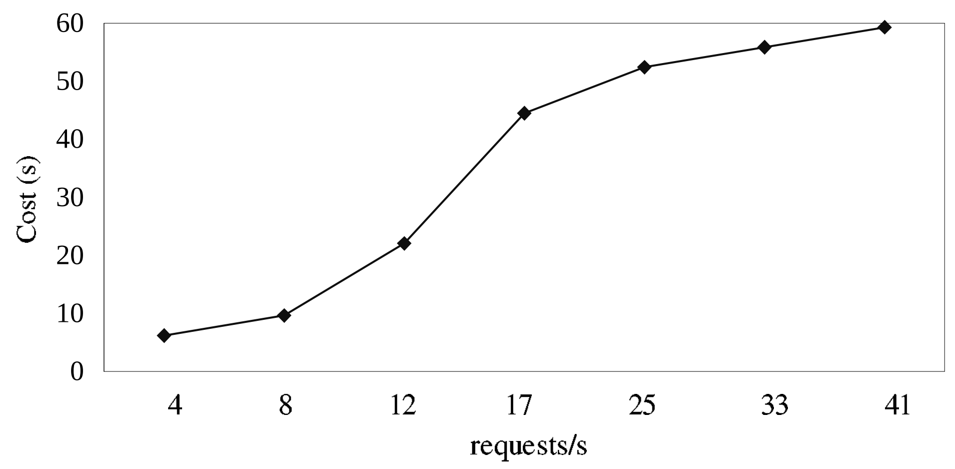

Figure 34 Cost under different request frequencies

Figure 33 and Figure 34 show that when the offered rate of traffic is below total link capacity, the request can be satisfied and the desired performance can be achieved. But if the offered rate of traffic is higher than link capacity, the request can only be partly satisfied according to link capacity. On the other hand, the cost increases with the increase of the number of requests, but the increase becomes smoother after the request load reaches a certain value.

6.3.5.4 Summary of Simulation Results on Performance and Cost of QoS Routing

The simulation results show:

- QoS routing cost increases and the performance decreases with the increase of the network size. The degree of the both cost increase and performance decrease varies with different LSU algorithms;
- In PB, the cost is reduced with the increase of the PB value, small and big PB values result in bad performance;
- In TB, the cost is reduced with the increase of the threshold value; small and big threshold values result in bad performance;
- In UCB, the performance and cost are affected by two variables, by setting suitable values, this algorithm can produce a good combination of performance and cost;
- QoS routing cost increases with the increase of the frequency of requests;
- QoS routing cost can be reduced by introducing the hold-timer.

7 Conclusions and Future Work

7.1 Conclusions

In this thesis, we presented the recent developments of QoS routing in the Internet; discussed the design issues of QoS routing with the aim of summarizing the latest developments and giving the reader a comprehensive understanding about the discussed issues. Besides, in this thesis, we implemented two QoS routing algorithms and four link state update algorithms in QRS which is developed by our lab, and demonstrated a preliminary simulation-based study on performance and cost of on-demand QoS routing under certain network conditions. As results, we found that the cost of QoS routing is very sensitive to network size, that the cost can be significantly reduced by the use of suitable link state update algorithms, that frequent requests cause high cost, that traffic pattern is another factor that affects QoS routing cost. The results we got from simulations offer some indications of QoS routing performance and cost in different network environments. These results could be useful when considering the implementation of QoS routing in the Internet in which most of the time, different network domains may use different routing algorithms and different link state update algorithms according to the characteristics of traffic travelling on it.

7.2 Future Work

Even though traffic with QoS requirements is expected to grow dramatically and even become dominant in the future, the traffic without special QoS requirements and routed by current best-effort routing is predicted to exist for the time being. Technically, implementing QoS routing without influencing best-effort traffic too much is regarded possible and will not cost too much. Especially the extra memory and processing needed by QoS routing is not a problem for the current processors. However, implementing QoS routing is a broad issue. We need to consider both its technical benefits and its economic value. The requirements set by the Internet providers and the service users are in fact different. While the service providers want a high utilization of network resources and less cost on processing and using memory, the users want a fast and guaranteed service and expect to pay less money. Obviously, it is a pair of contradictions. Studying the feasibility of implementing QoS routing is therefore not anymore a purely technical issue, it should be also based on deep knowledge of the needs from both parties involved. Only so, it is possible to work out some reasonable tradeoffs to reach an acceptable and realistic solution.

As a continuous project, our major interests will remain in studying the performance and cost of QoS routing. We will further make simulations with different network topologies and traffic models which are close to the real situation. And possibly our next step could be studying the signaling cost which comes from. Besides, we also intend to further develop QRS by adding more features e.g., high level admission control and policy control, and implement DiffServ MPLS QoS routing at the inter-domain level.

References

[1] Ferguson, P. & Huston, G., "Quality of Service-Delivering QoS on the Internet and in corporate networks", New York : John Wiley, cop. 1998.

[2] Crawley, E. et al., "A Framework for QoS-based Routing in the Internet", RFC 2386, August 1998.

[3] Ahmadi, H. & Chen, J. S.-C & Guerin, R., "Dynamic Routing and Call Control in high speed networks", In proceeding of Workshop Sys.Eng., ITC'13, Copenhagen, Denmark, June 1991.

[4] Wang, Z. & Crowcroft, J., "Quality of service routing for supporting multimedia applications", IEEE Select. Area Communication, Vol.14, no.7, September 1997.

[5] Widyonon, R., "The design and evaluation of routing algorithms for real-time channels", Technical ReportTR-94-024, University of California at Berkely, June 1994.

[6] Lee, W. C. & Hluchyj, M. & Humblet, P., "Routing subject to quality of service constraints in integrated communication networks", IEEE Networks, July/August 1995.

[7] Kompella, V. P. & Pasquale, J. C. & Polyzos, G. C., "Two distributed algorithms for the constrained Steiner tree Problem", In Proceedings of 2nd International Conference on Computer Communication and Networking, 1993.

[8] Zhang, P. & Kantola, R & Ma, Z., "Design and Implementation of QRS (QoS Routing Simulator)", Accepted by SPECTS'2000, February 2000. URL: http://www.tct.hut.fi/~pgzhang/papers.html.

[9] Braden, R. & Clark, D. & Shenker, S., "Integrated Services in the Internet Architecture: an overview", Internet RFC 1633, June 1994.

[10] Shenker, S. & Patrtridge, C. & Guerin, R., "Specification of Guaranteed Quality of Service", RFC 2212, September 1997.

[11] Wroclawski, J., "Specification of the Controlled-based Network Element Service", RFC 2211, September 1997.

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

[13] Blake, S. et al., "An Architecture for Differentiated Services", RFC 2475, December 1998.

[14] Bernet, Y. et al., "A Framework for Differentiated Services", Internet Draft, draft-ietf-diffserv-framework-02.txt, February 1999.

[15] Nichols, K. & Jacobson, V. & Zhang, L., "A Two-Bit Differentiated Services Architecture for the Internet", Internet Draft, draft-nichols-diff-svc-arch-00.txt, November 1997.

[16] Heinaen, J. et al., "Assured Forwarding PHB Group", Internet Draft, draft-ietf-diffserv-af-03.txt. November 1998.

[17] Valtonen, T. P., " Quality of Service in the 3rd Generation Transmission Network and the Internet", Helsinki University of Technology, May 2000.

[18] Awduche, D. O. et al., "Requirements for Traffic Engineering Over MPLS", RFC 2702, September 1999.

[19] Awduche, D. O. et al., "A Framework for Internet Traffic Engineering", Internet Draft, fraft-ietf-tewg-framework-02.txt.

[20] Awduche, D. O., “MPLS and traffic engineering in IP networks”, IEEE Communications Magazine, Volume: 37 12 , December 1999 , Page(s): 42 –47.

[21] Xipeng, X. et al., “Traffic engineering with MPLS in the Internet”, IEEE Network Volume: 14 2 , March/April 2000, Page(s): 28 –33.

[22] Xiao, P. & Ni, L.M., “Internet QoS: a Big Picture”, IEEE Network Volume: 13 2, March/April 1999, Page(s): 8 –18.

[23] Rosen, E. & Viswanathan, A. & Callon, R., “ Multiprotocol Label Switching Architecture”, draft-ietf-mpls-arch-06.txt, August 1999.

[24] Callon, R. et al., “A Framework for Multiprotocol Label Switching”, Internet Draft, draft-ietf-mpls-framework-05.txt, September 1999.

[25] Zhang, P. & Kantola, R. “Building MPLS VPNs with QoS Routing”, Accepted by IW'2000, April 2000.
URL: http://www.tct.hut.fi/~pgzhang/papers.html.

[26] Wu, L. et al., “MPLS Support of Differentiated Services”, Internet Draft, draft-ietf-mpls-diff-ext-06.txt.

[27] QoS Forum, “Quality of Service-Glossary of Terms”, May 1999.
URL: http://www.qosforum.com/white-papers/qos-glossary-v4.pdf.

[28] ATM Forum, “Private Network Network Interface (PNNI)”, version 1.0 Specifications, May 1996.

[29] Zhang, P. & Kantola, R., "Mechanisms for Inter-Domain QoS Routing in Differentiated Service Networks", March 2000. Accepted by QofIs'2000.
URL: http://www.tct.hut.fi/~pgzhang/papers.html.

[30] Chen, S. & Nahrsted, K., “An Overview of Quality of Service Routing for Next Generation High-speed Networks: Problems and Solutions”, IEEE Network, Volume: 12 6 , November/December 1998 , Page(s): 64 –79.

[31] Wang, B. & Hou, J. C., “Multicast Routing and its QoS Extension: Problems, Algorithms, and Protocols”, IEEE Networks, Volume: 14 1, January/February 2000 , Page(s): 22 –36.

[32] Guerin, R. & Orda, A., “QoS based routing in networks with inaccurate information: theory and algorithms”, INFOCOM '97. April 1997.

[33] Wang, Z. & Crowcroft, J., “QoS Routing for Supporting Resource Reservation”, IEEE JSAC, September 1996.

[34] Ma, Q. & Steenkiste, P., “Quality of Service Routing with Performance Guarantees”, Proc. 4th int’l. IFIP Wksp. QoS May 1997.

[35] Chen, S. & Nahrsted, K., “ On Finding Multi-constrained Paths”, IEEE ICC’98, June 1998.

[36] Salama, H. F. & Reeves, D. S. & Viniotis, V., “A Distributed Algorithm for Delay Constrained Unicast routing”, IEEE INFOCOM’97, April 1997.

[37] Sun, Q. & Langendofer, H., “ A New Distributed Routing Algorithm with End-to-End Delay Guarantees,” 2th Wksp. Protocols Multimedia Sys. October , 1995, pp.452-458.

[38] Cidon, R. & Rom, R. & Shavitt, Y., “Multi-path Routing Combined with Resource Reservation”, IEEE INFOCOM’97, April 1997.

[39] Shin, K. G. & Chou, C. C., “A distributed Route-Selection Scheme for Establishing Real-Time Channel”, 6th IFIP int’l Networking, September 1995.

[40] Chen, S. & Nahrstedt, K. “Distributed quality-of-service routing in high-speed networks based on selective probing”, Proceedings., 1998. LCN '98, 23rd Annual Conference on Local Computer Networks, 1998 , Page(s): 80 -89

[41] ATM Forum, Private Network Network Interface (PNNI) v1.0 Specifications, May 1996.
[42] Moy, J., "Multicast Extensions to OSPF", Internet Draft, September 1992.
[43] Kou, L. & Markowsky, G. & Berman, L., " A Fast Algorithm for Steiner Tree", Acta Informatica 15, 1981, pp.141-145.
[44] Takahashi, H. & Matsuyama, A.," An Approximate Solution for the Steiner Tree Problem in Graphs", Mathematica Japonica, 1980
[45] Kompella, V. P. & Pasquale, J. C.& Polyzos, G. C., "Multicast Routing for Multimedia Communication", Networking, IEEE/ACM Transactions on Volume: 1 3 , June 1993, Page(s): 286 –292.
[46] Widyono, R., " The Design and Evaluation of Routing Algorithms for Real Time Channels", Technical Report, ICSI TR-94-024, Univ. CA at Berkeley Int'l. Comp. Sci. Inst., June, 1994.
[47] Zhu, Q. & Parsa, M.& Garcia-Luna-Aceves, J. J., "A Source-based Algorithm for Delay-constrained Minimum-cost Multicasting", INFOCOM '95. IEEE , 1995 , Page(s): 377 -385 vol.1.
[48] Touskas, G. N. & Baldine, I., " Multicast Routing with End-to-End Delay and Delay Variation Constraints", IEEE JSAC, vol. 15, April, 1997, pp. 345-356.
[49] Apostolopoulos, G. et al., "QoS Routing Mechanisms and OSPF Extensions", RFC 2676, August, 1999.
[50] Moy, J., "OSPF version 2", RFC 2178, April 1998.
[51] Apostolopoulos, G. & Guerin, R. & Kamat, S., "Implementation and Performance Measurements of QoS Routing Extensions to OSPF". URL: http://www.seas.upenn.edu/~guerin/ .
[52] Apostolopoulos, G. et al., "Intra-Domain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis", Special Issue of IEEE Networks on Integrated Services and Differentiated Services for the Internet, September 1999. URL: http://www.seas.upenn.edu/~guerin/.
[53] Apostolopoulos, G. & Guerin, R. & Tripathi, S. K., "Quality of Service Based Routing: A Performance Perspective", In Processing of SIGCOMM, September 1998.
URL: http://www.acm.org/sigcomm/sigcomm98/tp/abs_02.html.
[54] Apostolopoulos, G. et al., "On Reducing the processing Cost On-Demand QoS Path Computation", Journal of High Speed Networks, Vol. 7, no. 2, pp.77-98.
[55] Ma, Q. & Steenkiste, P. "Supporting dynamic inter-class resource sharing: a multi-class QoS routing algorithm", INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE, Volume: 2 , 1999 , Page(s): 649–660.
[56] Floyd, S. & Jacobson, V., "Link-sharing and resource management models for packet networks", IEEE/ACM Transactions on networking, Volume: 3 4 , August 1995 , Page(s): 365 –386.
[57] Zhang, P. & Kantola, R., " Designing A New Routing Simulator for DiffServ MPLS Networks", February 2000. Internal Report,
URL: http://www.tct.hut.fi/~pgzhang/papers.html.
[58] Guerin, G. et al., "QoS Routing Mechanisms and OSPF Extensions", Internet Draft, draft-guerin-QoS-routing-ospf-03.txt, January 1998.
[59] Aukia, P., "QoS Routing Within an OSPF Area", Master thesis, Helsinki University of Technology, Department of Computer Science Engineering, September 1998.
[60] Zhang, Z. et al., "QoS Extensions to OSPF", work in Progress.

[61] Alaettinoglu, C. et al., "MaRS (Maryland Routing Simulator)-Version 1.0. User's Manual.
URL: http://www.isi.edu/~cengiz/publications/.

[62] Alaettinoglu, C. et al., "Design and Implementation of MaRS: A Routing Testbed".
URL: http://www.isi.edu/~cengiz/publications/.

[63] Alaettinoglu, C. et al., "MaRS (Maryland Routing Simulator)-Version 1.0. Programmer's Manual".
URL: http://www.isi.edu/~cengiz/publications/.

[64] Zhang, P .& Kantola, R., "QoS Routing Simulator: User's Manual (Version 1.1)", February 2000.
URL: http://www.tct.hut.fi/~pgzhang/papers.html.