HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering
Networking Laboratory

Zhou Wenpeng

Provision and Route Optimization in Differentiated Services Networks

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Engineering

Espoo, Finland, 16.9.2002

Supervisor Professor Raimo Kantola

Instructor Ph.D. Peng Zhang

ABSTRACT OF THE MASTER'S THESIS

Author:	Zhou Wenpeng
Title:	Provision and QoS Route Optimization in Differentiated Service Networks
Date:	16.9.2002 **Numbers of pages**: 98
Department:	Department of Electrical and Communications Engineering
Professorship:	S-38 Network Laboratory
Supervisor:	Prof. Raimo Kantola
Instructor:	Ph.D. Peng Zhang

With the occurrence of new applications such as delay-sensitive real-time services, Quality of Service is becoming increasingly important in the Internet. DiffServ is motivated by the desire to improve the overall performance of IP networks. The performance and efficiency of the DiffServ architecture can be enhanced with per-class traffic engineering.

This master's thesis examines the issues of provision in DiffServ networks and a per-class routing approach (PERD). Traffic of different classes can be distributed in accordance with network state and QoS requirements in provisioned networks.

The first part introduces some of the theory and literature related to the topic. The DiffServ architecture and end-to–end QoS are presented. In addition, provision and link sharing are described and discussed.

In the second part, we discuss the traffic engineering based on a per-class approach which we call PERD. With the help of the QRS simulator, we analyzed performance of networks under different routing algorithms. As results, we found that PERD can improve throughput and reduce end-to-end delay of the EF class service as well as lower priority classes. Under the same conditions, we find that the cost with PERD does not increase much compared to the widest bandwidth algorithm.

Keywords: routing, QoS, DiffServ, per-class, Intserv

Acknowledgements

This master's thesis has been written in the Networking Laboratory of Helsinki University of Technology within the IRoNet project.

My special thanks go to my supervisor, Professor Raimo Kantola, for his guidance and support during the writing of the thesis.

I gratefully acknowledge the inspiring ideas and valuable suggestions given by my instructor Ph.D. Peng Zhang.

I sincerely thank Vesa Kosonen and his wife, Kirsti, for their support when I was filled with immense grief.

I would also like to thank the people within the team who have contributed to my thesis with their knowledge and support. I want to thank my colleagues at the lab for an innovative and harmonious atmosphere.

Last, but most importantly, I would like to thank my friends and close relatives for their support and encouragement during my study in Finland, special thanks to my parents, my wife and my brothers, for their unconditional love, continuing support.

Espoo, 16.9.2002

Zhou Wenpeng

Table of Contents

Abbreviations

AF	Assured Forwarding
ATM	Asynchronous Transfer Mode
BA	Behavior Aggregate
BB	Bandwidth Broker
CR-LDP	Constraint Route Based Label Distribution Protocol
CBQ	Class-based Queuing
DiffServ	Differentiated Service
DSCP	Differentiated Service Code Point
DS domain	DiffServ Domain
EF	Expedited Forwarding
IETF	Internet Engineering Task Force
IntServ	Integrated Service
IP	Internet Protocol
ISP	Internet Service Provider
LSA	Link State Advertisement
LSU	Link State Update
MPLS	Multiprotocol Label Switching
OSPF	Open Shortest Path First
PERD	Per-class Routing Based on Per-class Dissemination
PQ	Priority Queuing
PHB	Per-Hop Behaviors
QoS	Quality of Service
QOSPF	QoS Open Shortest Path First
QRS	QoS Routing Simulator
RSVP	Resource Reservation Protocol
SLA	Service Level Agreement
SLS	Service Level Specification
SRRP	Sender-initiated Resource Reservation Protocol
TCA	Traffic-Conditioning Agreement
TCP	Transmission Control Protocol
TCS	Traffic Conditioning Specification
ToS	Type of Service
TLV	Type/length/value
TE	Traffic Engineering
UDP	User Datagram Protocol
VoIP	Voice of IP
WFQ	Weighted Fair Queuing

Index of Figures

Index of Tables

1 Introduction

1.1 Background

The traditional Internet was designed to provide best-effort service to all users. So, the Internet today cannot provide resource reservation and QoS (Quality of Service). In the past, this approach was able to meet users' needs since the applications that made use of the Internet did not require fixed delay bounds (e.g. telnet, ftp, e-mail, etc). However, with the occurrence of new applications such as Internet telephony and video-conferencing, there is an ongoing discussion about realizing QoS in the Internet today because these applications are delay-sensitive and require a fixed end-to-end delay bound.

To address this problem, the IETF (Internet Engineering Task Force) has proposed two philosophies to guarantee end-to-end QoS, namely, IntServ (Integrated Service) [9] and DiffServ (Differentiated Service) [10].

The main idea behind the IntServ is resource reservation per flow. It utilizes the Resource Reservation Protocol (RSVP) [11] as the signaling protocol to reserve resources (e.g., buffer space and link bandwidth) in each intermediate router from source to destination in order to satisfy specific QoS requirements for applications. RSVP is based on the idea of reserving resources for each TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) flow, causing every RSVP capable router to store information about this flow, to allocate resources and to instantiate traffic control components and queuing systems. RSVP is really able to guarantee bandwidth and delay on a per-flow basis, which matches the needs of modern real time applications. However, even though this works fine in small and medium sized networks the use of per-flow state and per-flow processing causes scalability concerns for large networks. In other words, RSVP cannot scale to the Internet backbone.

An alternative concept for a QoS supporting Internet is the so-called DiffServ. The basic idea is the implementation of different traffic classes in the Internet. The differentiation among these classes is done by DSCP (Differentiated Service Code Point) in the ToS (Type of Service) byte of the header of IP (Internet Protocol) packets. In order to overcome scalability problems, only boundary routers process traffic on a per-flow basis. Core routers forward packets based on PHB (Per-Hop Behaviors). According to the DSCP, packets are put into corresponding queues with different priority or dropping algorithms causing different packets to be forwarded. Since there is no need to maintain per flow states in the core routers, the DiffServ is more scalable than IntServ.

In order to maintain the service quality, each ISP (Internet Service Provider) domain needs to control the amount of incoming traffic, which is negotiated through a SLA (Service Level Agreement). In addition, some domains have a BB (Bandwidth Broker) which manages the bandwidth resources within the domain and negotiates the SLA with neighboring domains.

1.2 Objectives of This Thesis

Some researchers have studied end-to-end QoS guarantees in DiffServ networks. Most of the research so far on DiffServ was focused on service specification, service architecture and component definition [44][2][7][26][11]. Few of works discuss how resource management and traffic engineering could be implemented in the DiffServ environment. Without good resource management schemes, DiffServ itself would not provide any QoS. Our emphases will be on the route optimization in provisioned DiffServ networks. Provisioning and route optimization are not two relatively independent issues. They are closely related to each other. Provisioning is a long-term process to optimize network resources, while QoS routing is a relatively short-term process to optimize efficient utilization of network resources against relatively short-term traffic fluctuation. We will discuss them one after another.

In this paper, we will firstly deal with provisioning. Provisioning in DiffServ networks does not only mean determination and allocation of resources necessary at

various points in the network, but also modification of existing resources to be shared dynamically among other traffic classes.

Then, we will discuss routing optimization which is realized by QoS routing. Some researchers suggested a Bandwidth Broker to keep the available resource information of every node and link [2] [11] [12]. However, the available link bandwidth they used is the total residual bandwidth of a single class instead of the available bandwidth of an individually specific class (in provisioned DiffServ networks, each class has its own maximum available bandwidth). This kind of total resource information cannot reflect the real dynamic resource state of networks.

In our work, we refer residual bandwidth to the “residual bandwidth” of a link; in contrast, “available bandwidth” stands for the available bandwidth of a specific class. We will keep this convention throughout this paper.

In order to overcome this problem, in our work, we will propose a method which we call PERD -- (Per-class Routing Based on Per-class Dissemination) [37]. By implementing PERD, we will extend QOSPF (QoS Open shortest Path First) on the principle of per-class processing. In LSA packets (Link State Advertisement), both the residual bandwidth and the available bandwidth of each individual class of a link are broadcast and updated. We specify three classes—EF class, AF1 class and AF2 class in this thesis. With more accurate network state information, we hope that traffic of different classes can find optimal paths accordingly and end-to-end QoS can be guaranteed.

1.3 The Structure of This Thesis

This thesis is divided into seven chapters. The first four chapters cover some of the theory that is significant to this topic. They are mainly based on literature research.

Chapter 2 describes the architecture and components of DiffServ networks, QoS routing and end-to-end QoS guarantee. In addition, we compare advantages and disadvantages between DiffServ and IntServ.

Chapter 3 presents provisioning in DiffServ. We propose an architecture implementing provisioning with a Bandwidth Broker. Moreover, CBQ is discussed.

Chapter 4 describes the traffic engineering and proposes the PERD concept.

Chapter 5 describes our simulations with different routing algorithms.

Chapter 6 concludes this thesis on the basis of simulation results and Chapter 7 suggests some ideas for future work.

2 End-to-end QoS in DiffServ Networks

To get a clear overview of QoS in DiffServ networks, we will present some concepts relative to QoS and DiffServ. We present the DiffServ architecture and the differences between IntServ and DiffServ. We also describe QoS routing and refer to previous works associated with this topic by other researchers.

2.1 Background of DiffServ Network

DiffServ is a comprehensive concept. We mainly describe some aspects in terms of DiffServ architecture and components.

2.1.1 Targets for Differentiated Services

In the Internet, fundamentally different packets are delivered. Some packets pertaining to contracts or bank accounts are vital to someone's business, so the received packets must be exactly same as the originally sent packets. Some packets such as VoIP (Voice of IP) traffic packets must be received within fixed delay limit, or those packets will be discarded even though they can arrive at the destination after their playback time has expired. Those different kinds of packets expect different QoS instead of uniform best effort.

At the same time, more and more hosts are connected to the Internet, so core routers in backbone networks need to store more and more network state information and per-flow state information in order to forward an incoming packet, which belongs to a traffic flow, to its outgoing interface. The size of routing tables is becoming larger and larger, which slows down the routing table lookup and overall delivering speed is reduced. Therefore, scalability is becoming a salient problem for the traditional Internet.

IETF proposes DiffServ aiming to resolve the above problems. Differentiated services refer to a simple service structure that provides quality differentiation mainly by means of packet marking [13].

This definition consists of three parts:

- DiffServ is a target model rather that a specification that contains detailed information about the required implementation.

- From the service perspective, DiffServ provides a moderate level of quality differentiation without strict guarantees.

- The distinctive technical characteristic is that the quality of service is not attained by reserving capacity for each individual flow or connection, but by marking packets at the network boundaries.

2.1.2 Basics of DiffServ Network

A large network is composed of a number of nodes and links. In spite of its complex structure, it usually can be divided functionally into two sections: the access network and the backbone (core) network (Figure 1). The main tasks of the access network are to physically connect customers to the network and to provide appropriate tools, such as pricing capabilities, to manage the relationship between the network operator and the customers.

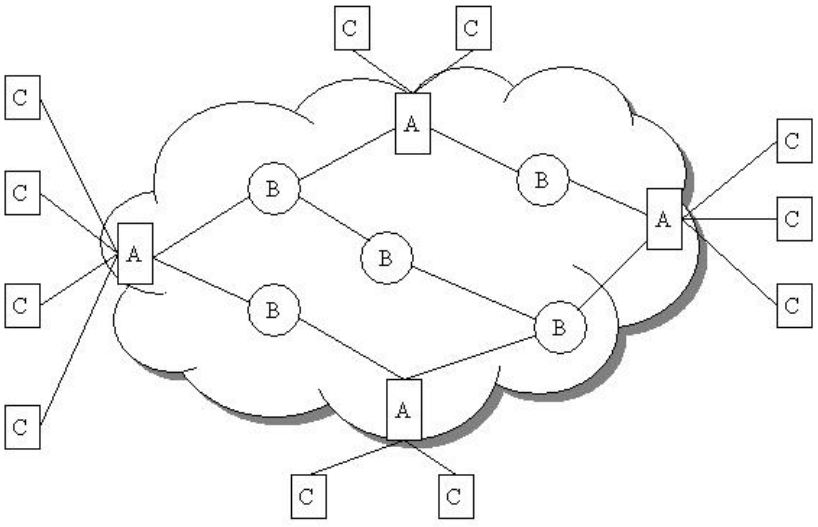

Figure 1 The main element blocks are boundary nodes (A), interior nodes (B), and access equipment (C)

A DS domain (Figure 2) is a core network within a large network. In general, a DiffServ network normally consists of one or more networks under the same administration; for example, an organization's intranet or an ISP. Similarly, a DS domain is made up of many nodes and links. We divide those nodes in a DifferServ network into two categories by their functions, namely, boundary node (DS boundary node in Figure 2) and interior node (DS interior node in Figure 2), according to their characteristics. DS boundary nodes interconnect the DS domain to other DSs or access networks, while DS interior nodes only connect to other DS interior or boundary nodes within the same DS domain. In detail, the boundary node consists of an ingress node and an egress node. These elements are virtual in the sense that one physical node may contain all characteristics of all node types. In other words, each type of node is a collection of characteristics:

- Boundary node: A collection of functions needed to interconnect a DS domain to another DS domain or to a non-DS-capable domain.
- Interior node: A collection of functions needed if a node is connected only to other DS-capable nodes.
- Ingress node: A collection of functions needed to handle incoming traffic streams to a DS domain.
- Egress node: A collection of functions needed to handle outgoing traffic streams from a DS domain.

As we mentioned earlier, boundary nodes in DiffServ include both ingress nodes and egress nodes. In fact, DS boundary nodes act both as a DS ingress node and as a DS egress node for different directions of traffic. Traffic enters a DS domain at a DS ingress node and leaves the DS domain at a DS egress node (Figure 2). A DS ingress node is responsible for ensuring that the traffic entering the DS domain conforms to the TCA (Traffic Conditioning Agreement) between it and the other domains (access networks or other DSs) to which the ingress node is connected. A DS egress node may perform traffic conditioning functions on traffic forwarded to a directly connected peering domain, depending on the details of the TCA between the two domains.

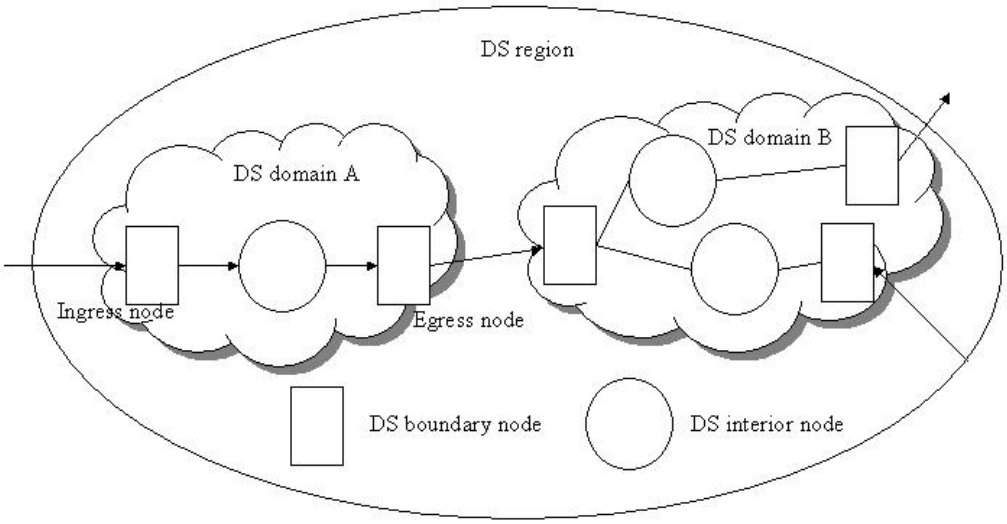

Figure 2 Basic elements of a DiffServ network

There is a definite distinction between boundary functions and interior functions, or boundary nodes and interior nodes (Figure 3).

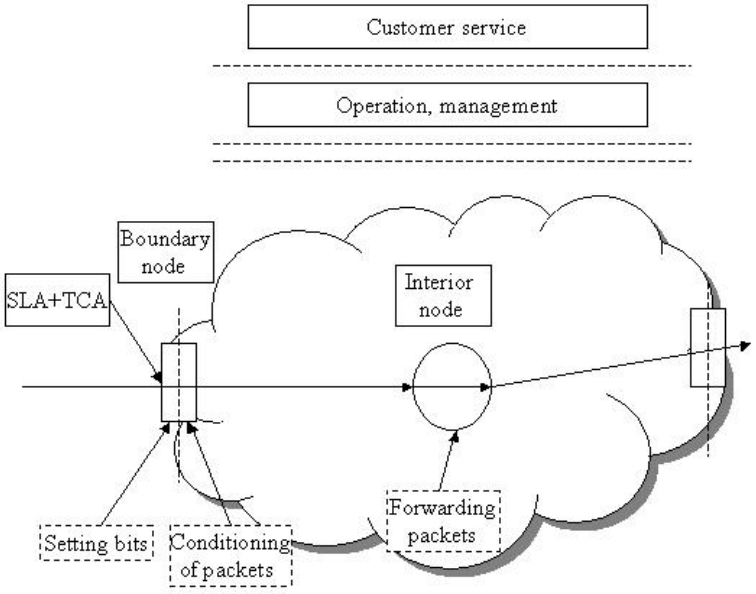

Figure 3 Differentiated service model

In general, the total capacity of a boundary node connecting access and core networks, measured in bit rate, is usually much smaller than that of an interior node. On the other hand, a boundary node has more sophisticated tools that enable it to

control and measure individual flows. Boundary nodes are responsible for classifying packets, setting DS bits in packets, and for conditioning packets. Interior nodes efficiently forward large bundles of aggregate traffic at a high speed.

At each user/provider interface (boundary node) in DiffServ networks, the provided service is defined by means of an SLA (Service Level Agreement). The SLA is a contract, established either statically or dynamically, that specifies the overall performance and features which can be expected by a customer. The subset of the SLA, which provides the technical specification of the service, is referred to as the SLS (Service Level Specification).

A profound subset of the SLS is the TCS (Traffic Conditioning Specification) which specifies detailed service parameters for each service level. These service parameters include service performance parameters (e.g., throughput, latency, drop probability) and traffic profiles corresponding to the requested service. The TCS may define the marking and shaping functions to be provided.

Traffic entering the DS domain must be classified, marked and possibly conditioned according to TCA (Traffic Condition Agreement) and SLA. SLA seems to be rather a customer service and network service concept; TCA on the other hand, should be defined by terms that belong to the traffic handling level. The function components in the DiffServ edge node consist of packet classifier, meter, marker and shaper/dropper (Figure 4).

The DSCP markings are applied either by a trusted upstream node, e.g. a customer, or by edge routers on entry to the Diffserv network [1]. Figure 4 presents a logical structure of traffic classification and conditioning functions. This structure is based on the assumption that classification is made according to the information in the packet header (such as source address and destination address and DS field) and the incoming interface.

A traffic profile is one way to present traffic-conditioning rules. In the simplest model, each packet is either in-profile or out-of-profile based on the metering result at

the arrival time of the packet. In-profile packets obtain better traffic conditioning and forwarding treatment than out-of-profile packets.

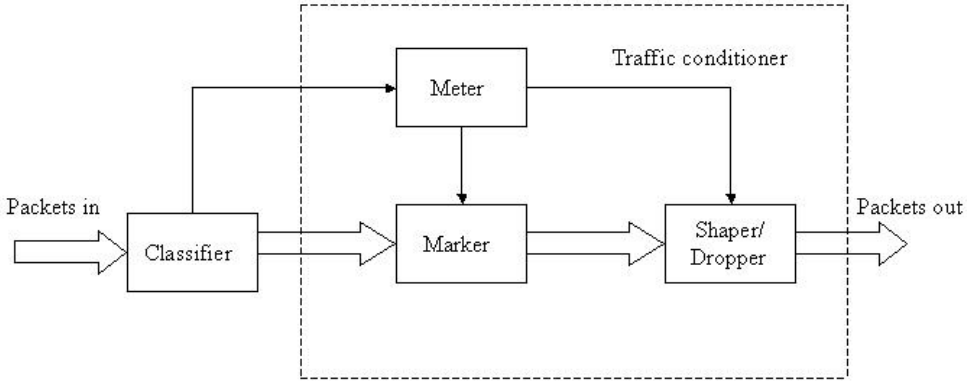

Figure 4 Traffic conditioning and packet classifier

- Packet classifier identifies packets in a traffic stream based on the content of some portions of packet header.

- Traffic meters measure the temporal properties of the stream of packets selected by a classifier against a traffic profile specified in a TCA. A meter passes state information to other conditioning functions to trigger a particular action for each packet which is either in- or out-of-profile (to some extent).

- Packet markers set the DS field of a packet to a particular code point adding the marked packet to a particular DS behavior aggregate. The marker may be configured to mark all packets which are steered to it to a single code point, or may be configured to mark a packet to one of a set of code points used to select a PHB in a PHB group, according to the state of the meter. When the marker changes the codepoint in a packet it is said to have "re-marked" the packet.

- Shapers delay some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. A shaper usually has a finite-size buffer, and packets may be discarded if there is not sufficient buffer space to hold delayed packets.

- Droppers discard some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. This process is known as "policing" the stream. Note that a dropper can be implemented as a special case of a shaper by setting the shaper buffer size to zero (or a few) packets.

2.1.3 Per-hop Forwarding Behavior (PHB)

In this subsection, we describe one of the important characteristics of DiffServ—the Per-hop Forwarding Behavior (PHB). The term PHB is both difficult to comprehend and important for understanding the whole idea of DiffServ. A PHB is a description of the externally observable forwarding behavior of a differentiated services node and is applied to a collection of packets with the same DSCP that are crossing a link in a particular direction. Each service class is associated with a PHB. Figure 5 shows a simplified model for PHB specification that concerns the treatment of an aggregate stream inside a black box.

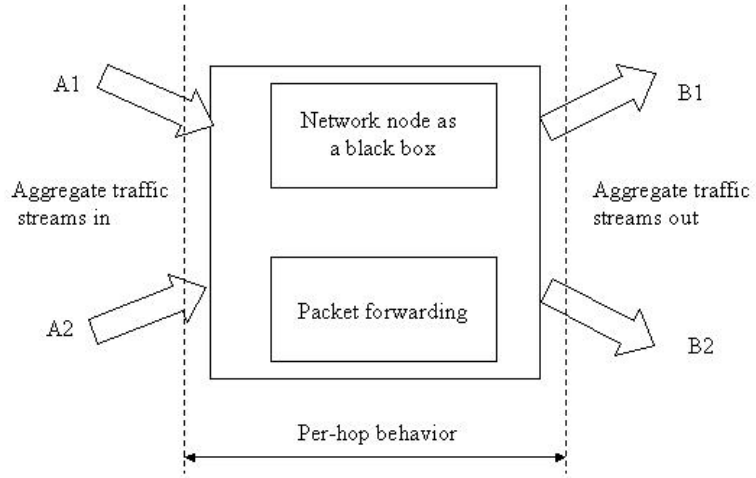

Figure 5 Per-hop behavior (PHB)

PHB is defined in terms of behavior characteristics relevant to service provisioning policies, and not in terms of particular implementations. PHBs may also be specified in terms of their resource priority relative to other PHBs, or in terms of their relative observable traffic characteristics. These PHBs are normally specified as group PHBs and are implemented by means of buffer management and packet scheduling mechanisms.

Currently there are two proposed PHBs which are briefly described below.

The Expedited Forwarding (EF) PHB is a high priority behavior typically used for network control traffic; for example, routing updates, and delay-sensitive real-time traffic. The EF PHB is defined as a forwarding treatment for a particular differentiated service aggregate where the departure rate of the aggregate's packets from any DS-compliant node must equal or not exceed a configurable rate. Extra traffic must be dropped. The EF traffic should be allocated this rate independently of the intensity of any other traffic attempting to pass the node [27].

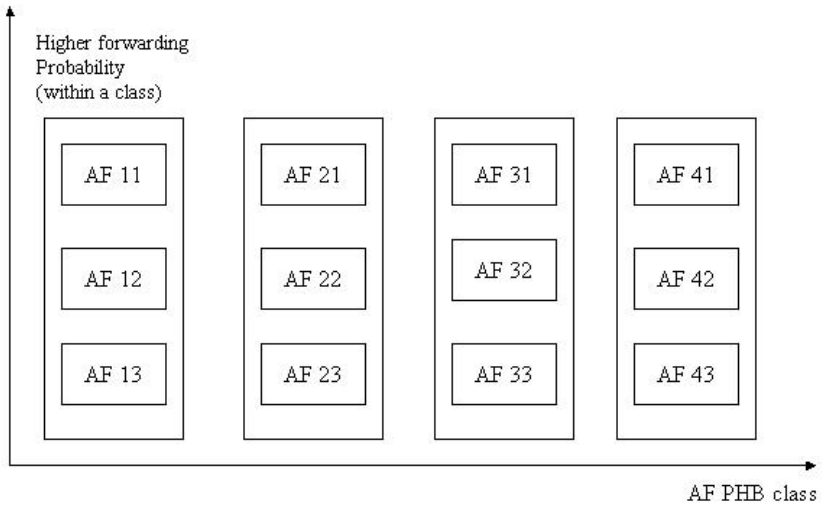

Figure 6 Structure of AF PHB group

The Assured Forwarding (AF) PHB is a means for a provider of differentiated services domain to offer different levels of forwarding assurances for IP packets received from a customer of the differentiated services domain. Four AF classes are defined, where each AF class in each differentiated services node is allocated a certain amount of forwarding resources, e.g., buffer space and bandwidth. Within each AF class, IP packets are marked with one of three possible drop precedence values as shown in Figure 6. In case of congestion, the drop precedence of a packet determines the relative importance of the packet within the AF class [28].

2.1.4 Differences between IntServ and DiffServ Networks

From above descriptions, we can summarize the differences between the IntServ and DiffServ networks:

Since there is only a limited number of service classes indicated by the DS field, 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 network is therefore more scalable.

Sophisticated classification, marking, policing and shaping operations are only needed at boundary of the networks (Figure 7). ISP core routers need only to implement Behavior Aggregate (BA) classification. Therefore, it is easier to implement and deploy Differentiated Services.

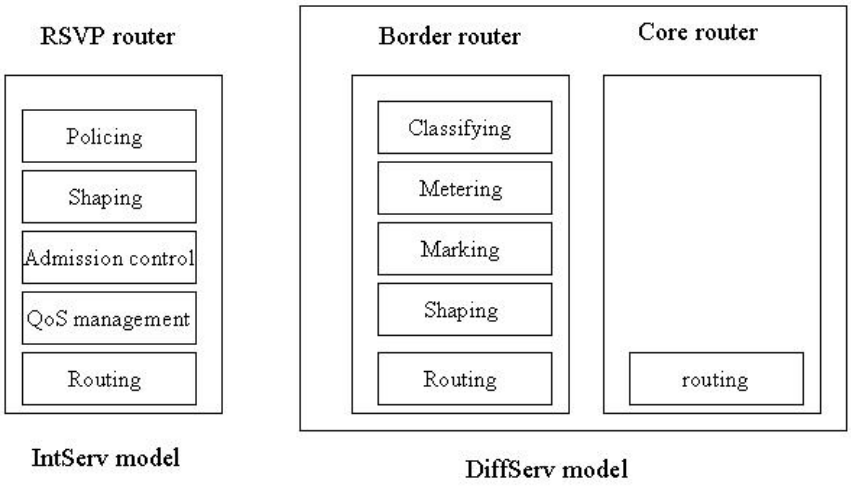

Figure 7 Tasks performed in the network elements of IntServ and DiffServ networks

We know that the advantage of DiffServ model is its scalability because there is no need to maintain per-flow states in the core routers, while Intserv can provide good control granularity on QoS. On the other hand, disadvantages of the DiffServ network are [14]:

- Its service is not flexible, since the application cannot specify the required end-to-end delay. In its current definition [12], DiffServ can only provide a static priority service discipline to the differentiated classes, which cannot be translated into end-to-end delay bounds.

- The issue of admission control has not been defined yet and, therefore, QoS guarantee cannot be provided.

2.2 QoS Routing

Different applications generate different traffic. Therefore, IP traffic is heterogeneous and it has the following features [47]:

- IP traffic is variable-speed.
- IP traffic arrives in sudden bursts.
- Destination address is frequently changed (e.g., net surfing).
- A variety of QoS is needed depending on applications.

Delivering IP traffic is not an easy task. Conventional IP routing attempts to find and follow the shortest path between a packet's current location and its intended destination with the intention of saving network resources. This can lead to congestion in the bottleneck of network—routers and links on the shortest path to many destinations subject to high traffic load. Packet loss rate, latency, and jitter increase as the average load on a router rises. Two solutions exist: faster routers and links, or distributing packets across alternate routes (load balancing). The former solution is out of scope of our study. The latter method is to find different paths for different kinds of traffic depending on their QoS requirements and network resources. This solution is called QoS routing.

QoS routing has been defined in different ways. In [49], 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 [50], 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".

Regardless of different definitions, the basic function of QoS routing is to find feasible paths which have sufficient residual resources to match the QoS requirements of flows while achieving efficiency in network resource utilization. QoS Routing supports traffic with different services, the path is decided on a certain metric or a combination of metrics. Its one noticeable advantage is that it can provide alternate

routes. If the best existing path cannot admit a new flow, the associated traffic can be forwarded on an alternate route that can guarantee its service. In most cases, the goal is not to find the best solution, but a viable solution with acceptable cost. QoS routing algorithms can prevent traffic shifting from one path to another “better” path only if the current path meets the service requirements of the existing traffic. Designing and implementing QoS routing is much more difficult than best-effort routing. Some tradeoffs have to be considered.

2.2.1 Objectives of QoS Routing

In summary, QoS routing is supposed to improve the utilization of network and QoS of applications. The main objectives of QoS routing [49] 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 the network is under heavy load, QoS routing is expected to give better performance (e.g. better throughput) than best-effort routing. By doing this, QoS routing can find a longer and lightly loaded path better than the heavily loaded shortest path. Network traffic is thus distributed more evenly.

2.2.2 Routing Metrics and Path Computation

Routing metrics determine if an optimal path can be found. So routing metrics must represent a network in routing; as such, they are associated with not only the range of QoS requirements that can be supported but also the complexity of path computation.

2.2.2.1 Selection Criteria of Routing Metrics

Normally, defining suitable routing metrics needs to take into account a number of factors [51]:

The metrics must reflect the basic network properties of interest. Such metrics may include the information of residual bandwidth, delay etc., which makes 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 cannot support QoS requirements that cannot 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 both 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 such as bandwidth are computed first and a subset of them are eliminated based on secondary metric (e.g., hop count) and so forth until a single path is found .

Once suitable metrics are defined, a uniform representation of them is required. Particularly, encoding for the maximum, minimum, range, and granularity of the metrics are needed.

2.2.2.2 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, in 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. Nevertheless, 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 $P = (n_1,n_{2\ldots,} n_j,n_k)$, metric m is:

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

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. [51] 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).

2.2.2.3 Bandwidth and Hop-count as Metrics

Among the common routing metrics in QoS routing, bandwidth and hop-count are more useful constrains than delay and jitter, because:

- Although applications may care about delay and jitter bounds, few applications cannot 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, the longer delay it experiences. For example, a 1Mb/s flow that traverses two hops consumes twice as many resources than one that traverses a single hop.

In addition, algorithms for finding paths with bandwidth and hop-count constraints are rather simple [52]: 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 [61].

2.2.2.4 Path Computation Mode

QoS paths can be either computed on-demand or pre-computed for each traffic class. On-demand computation is triggered by the receipt of the QoS request of a flow. It has the benefit of being able to always use the latest network information. However, if requests arrive too frequently, this approach may be 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 in terms of storing if most of the pre-computed paths are not used.

Basically, common approaches to reduce the computation overhead of QoS routing include:

- Using a large-valued timer to reduce the computation frequency.
- Choosing bandwidth and hop-count as metrics.
- Using administrative policy to prune unsuitable links before computing the routing table.

2.2.3 QoS Routing and Other Network Components

Providing QoS guarantees (e.g., EF class) in any kind of networks requires the reservation of enough resources along the path from the source to the destination. Therefore, a path has to be established in advance between the source and destination nodes, and all packets should follow the same path [32]. In order to reach this aim, some network components are needed to implement some functions such as admission control and resource reservation.

In [25] the author elaborated the relationship among QoS routing and some other network components:

- QoS routing is obviously different from the traditional best effort routing. The QoS routing is normally connection-oriented with resource reservation to provide the guaranteed service. The best effort can be either connection-oriented or connectionless with dynamic performance subject to the current availability of shared resources.

- QoS routing and resource reservation are two important and closely related network components. In order to provide the guaranteed service, the required

resources (buffer space, link bandwidth) must be reserved when a QoS connection is established. Before the reservation can be done, a path with the best chance to satisfy the resource requirement must be selected.

- The task of admission control is to determine whether a connection request should be accepted or rejected. Once a request is accepted, the required resources must be guaranteed. If the resource reservation is successfully done along the route(s) selected by the routing algorithm, the connection request is accepted, otherwise, the request is rejected.

- A QoS routing algorithm may fail to find a feasible path for a connection, either because a feasible path does not exist, or because the searching space of a heuristic approach does not cover any existing feasible path. When this happens, the system can either reject the connection or negotiate with the application for a looser QoS constraint. QoS routing can assist the negotiation by finding the best available path and returning the QoS bounds supported. If the negotiation is successful according to the provided bounds, the best available path can be used immediately.

2.2.4 Benefits and Problems of QoS Routing

QoS routing determines routes under both the knowledge of network resource availability and the requirements of flows. As a result, the performance of applications is guaranteed or improved in comparison with that without QoS routing by means of optimizing the resource usage in the network [41][42]. These benefits might be achieved in a number of ways as follows.

- QoS routing selects feasible routes by avoiding congested nodes or links.

- If workload exceeds the limit of existing paths, QoS routing offers multiple paths for transferring additional traffic.

- If network or node failure occurs, QoS routing selects alternative paths for quick recovery without seriously degrading the quality.

- Different traffic classes have different QoS requirements. Traffic aggregates having identical sources and destinations can travel different paths.

However, these benefits of QoS routing also incur the cost of developing new routing protocols or extending the existing ones. Moreover, it potentially increases communication, processing and storage overheads. It brings out a number of problems as follows [43].

- What kinds of resource information can be used for determining feasible routes?

- Which protocols are suitable for distributing route and source information in intra-domain or inter-domain?

- How to select routes across multiple domains?

- How to balance the complexities and benefits of introducing QoS routing into a real network?

- In which ways the cost of running QOS routing in the DiffServ network can be minimized.

2.3 End-to-end QoS Guarantee

The IntServ architecture provides the Internet the ability to deliver end-to-end QoS to applications over heterogeneous networks. Existing approaches for providing IntServ requires routers to manage per-flow states and perform per-flow operations. Such an IntServ network raises the scalability concerns when the network size or the number of flows is significantly large. The DiffServ approach proposes a scalable means to deliver end-to-end QoS guarantee based on aggregate traffic handling.

2.3.1 RSVP and IntServ

In IntServ networks, in order to guarantee the end-to-end QoS, the reservation of enough resources along the path from the source to the destination is required [31]. We know clearly there are signaling protocols, such as RSVP [38] and CR-LDP [56] guaranteeing the end-to-end QoS.

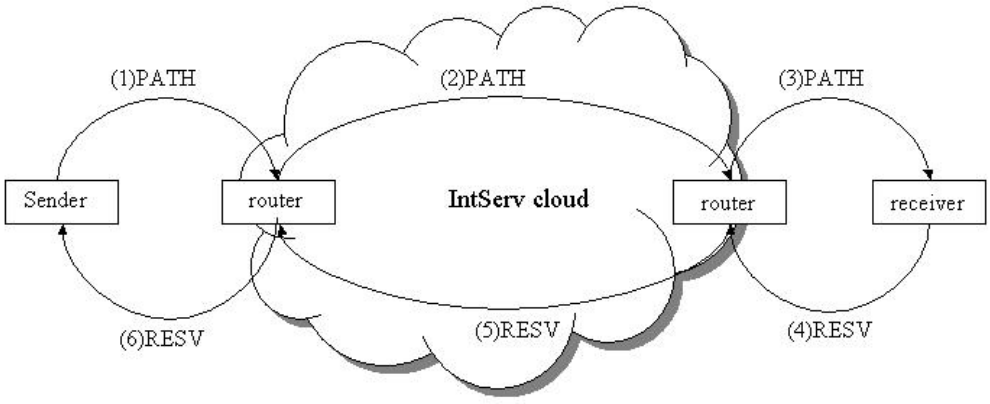

Figure 8 RSVP Protocol

We depicted RSVP protocol briefly in Figure 8. In IntServ networks, each router has the admission control function. When some new flow wants to receive a particular level of service, admission control looks at the Tspec (flow's traffic characteristics) and Rspec (service requested from the network) of the flow and tries to decide if the desired service can be provided to that amount of traffic, given the currently available

resources, without causing any previously admitted flow to receive worse service than it had requested.

Each intermediate router checks the PATH message as it goes past, and it figures out the reverse path that will be used to send reservations from the receiver back to the sender.

Having received a PATH message, the receiver sends a reservation back in a RESV message. This message contains the sender's Tspec and Rspec describing the requirements of this receiver. Each router on the path looks at the reservation request and tries to allocate the necessary resources to satisfy it. If the reservation can be made, the RESV request is passed on to the next router, if not, an error message is returned to the receiver who made the request. If all goes well, the correct reservation is installed at every router between the sender and the receiver.

2.3.2 Proposals for End-to-end QoS in Diffserv Networks

We compared IntServ and DiffServ networks in Section 2.1.4 "Differences Between IntServ and DiffServ Networks". By observing Figure 7, it is obvious that the architecture of RSVP router is different from that of DiffServ router. Since core routers in DiffServ domain are not signal-aware, existing signaling protocol such as RSVP cannot be applied to DiffServ network any more. Hence, the QoS problem arises in DiffServ network.

At the same time, in the pure DiffServ network, the admission control is in away: edge network elements at the ingress to the Diffserv region deal with the traffic in an aggregate manner, not in per-flow traffic police manner [7]. This may be very inefficient in some cases.

We can illustrate those cases with one simple example (Figure 9). The capacity of each link within this DiffServ network is 2Mbps. There are 6 IP video traffic flows (EF class service) on demand originating outside of the DiffServ network region. Each requires 0.3Mbps of EF service from the DiffServ region. So the total bandwidth

requirement is 6*0.3 = 1.8Mbps. On the other hand, we suppose the capacity of lk1-6 is only provisioned to accept 0.9Mbps EF traffic. Then the traffic conditioner on the ingress node will drop half of the incoming traffic (see Section 2.1.2 “Basics of DiffServ Networks”), which is EF aggregates, with no regards of which flow packets belong to. Congestion will take place on lk1-6.

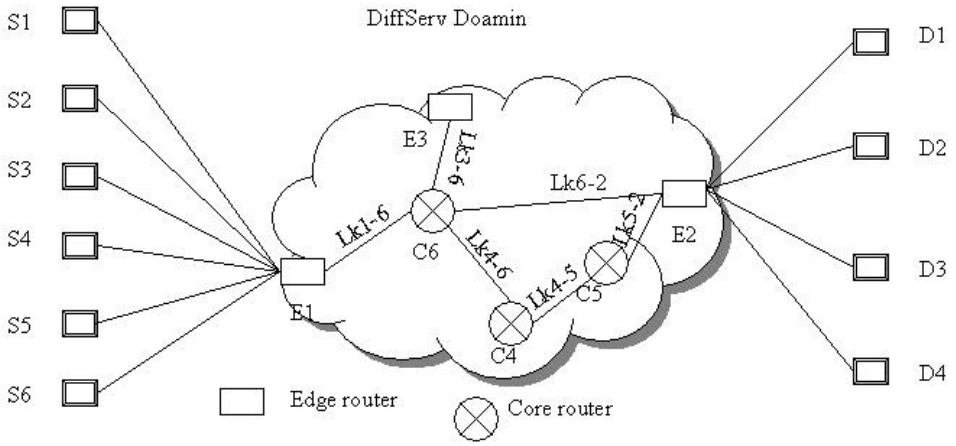

Figure 9 Resource based admission control for DiffServ

A network element (link or buffer) is said to be in congestion if it experiences sustained overload over an interval of time. Congestion is one of the most significant problems in the Internet. If congestion occurs it almost always results in degradation of QoS to end-users. This includes packet loss and delay increase. The result is that no connection could really get satisfactory service. This case is not what customers and ISPs expect, while in fact there are sufficient resources for 3 connections.

Moreover, inside the DiffServ network, when traffic load is heavy, uneven distribution of traffic can cause congestion on bottleneck links. The conventional routing algorithm is the shortest path algorithm. Under the scenario depicted by Figure 9, Lk6-2 is the bottleneck link. Congestion will occur on this link when the traffic load is heavy. Now, we can find congestion happened not only at the edge routers but also in the interior of DiffServ networks. From the functions of both edge routers and core routers (see Section 2.1.2 Basics of DiffServ Networks), we know Diffserv alone cannot solve this problem. QoS routing must be used to avoid such

congestion caused by uneven traffic distribution. Figure 10 illustrates the DiffServ architecture equipped with QoS routing [48].

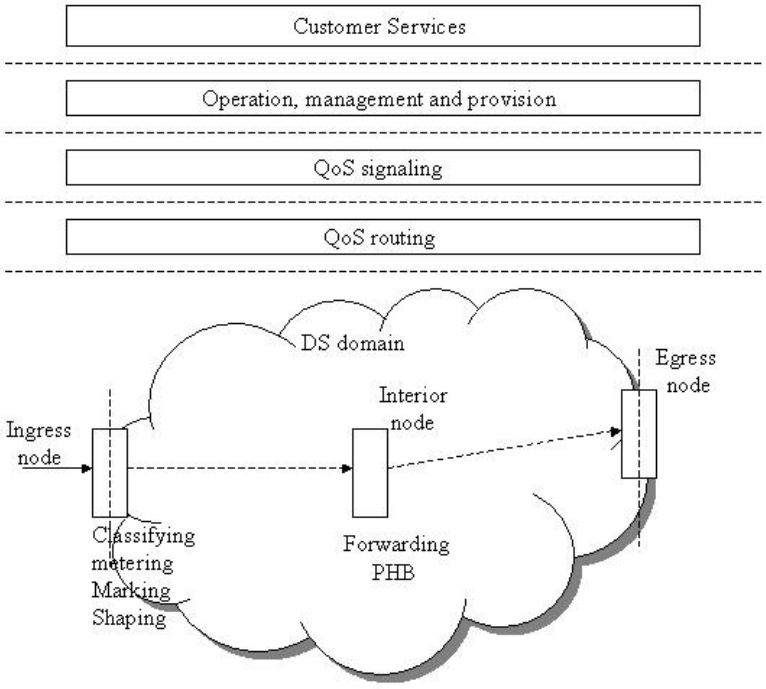

Figure 10 DiffServ architecture with QoS routing

In this architecture, compared with Figure 3 (DiffServ architecture without QoS routing), requirements from customers should be classified first in order to specify customer services. Then, network providers will provision the network resources for supporting these services. To keep reliability and usability of the network, network providers must carry on functions of network operation, management and provision. QoS signaling protocol is needed to broadcast control messages from network manager or exchange interoperation information among network nodes. QoS routing can be made depending on network state information. If a feasible route that satisfies the required QoS cannot be found, this request will be rejected, and vice versa.

Based on this QoS routing architecture, we try to solve those previous problems by using explicit signaling over DiffServ domains. An admission control agent for the DiffServ region of the network could be configured to do explicit signaling. In this case, three connections (S1-S3) could be set up and the other 3 connections (S4-S6)

would be rejected because of limited resources. Those admitted connections could guarantee their QoS.

2.3.2.1 Related Works

Some researchers have studied mechanisms to set up end-to-end QoS guarantees over Diffserv networks. In [6], Y.Bernet described a framework by which Intserv can be supported over Diffserv networks. By using such a framework, one can achieve the advantages of both sides, i.e. providing services as flexible and powerful as Intserv networks, while the architecture itself is as scalable and robust as the Diffserv network. They provide a general model to use Intserv with DiffServ but put much stress on the RSVP (the default) signaling protocol with DiffServ.

In [7], G. Zhang proposed a Sender-initiated Resource Reservation Protocol (SRRP) that can effectively cooperate with the DiffServ network and offer scalable end-to-end services. Ingress nodes in DiffServ domains can use custom policies to allocate resources to specific applications. The author suggested each edge router in the access network connecting to a DiffServ domain consist of two halves. A standard IntServ half which interfaces to the customer's network region and a DiffServ half which interfaces to the DiffServ network. The IntServ half is to identify and process traffic on a per-flow granularity.

The DiffServ half of the edge router reviews the resource requests against the service mapping and resource availability table. Based on the review, the admission could be made. However, it cannot make sure all packets belonging to the same flow pass the same route.

In papers [19] [20], the authors suggested DiffServ networks and MPLS combined. MPLS uses label-switching technology to aggregate a large number of IP flows onto a label at an ingress node. The label-switched path between ingress nodes and egress nodes can be established by a distributed routing and signaling protocol, the constraint route based label distribution protocol (CR-LDP).

In paper [11], the author proposed a lightweight version of RSVP, called DiffServ PHB Reservation Protocol (DPRP), as a way to transport and negotiate the QoS requirement between source and destinations. DPRP is receiver-oriented, it makes PHB reservation, adapts dynamically to changes of receiver requirement as well as to changes of DS domains.

In paper [2], the author proposed an architecture that can guarantee end –to-end QoS for EF class service without per-flow state management at core routers under the DiffServ framework. Its principle is:

A centralized BB performs admission control, resource provisioning, QoS routing and other policy decisions. It decouples the QoS control plane from the data-forwarding plane. The QoS reservation states are stored at and managed solely by the BB in a network domain.

2.3.2.2 Summary of those Proposed Mechanisms

Now, we compare the proposed methods in terms of their advantages and disadvantages.

All of the works we mentioned in the preceding section try to establish a signaling mechanism in a DiffServ domain. The source sends a resource request and this request traverses the path from source to destination. We should bear in our mind that no core routers in DiffServ region are signaling-aware.

In general, there are two ways to implement end-to-end QoS guarantee in DiffServ networks. In the first way, each edge router in DiffServ region makes admission control separately. We call this kind of way distributed admission control. With this method, sometimes, the whole situation of a network cannot be considered and congestion may occur on bottleneck links.

In the second way, other researchers introduced a BB to centrally allocate resources in DiffServ domains. BB is responsible for admission control and QoS routing of a new

application in a domain. It can be seen as a node in an autonomous domain. BB can be an independent node or it can be combined with the edge routers in DiffServ domain. This means is depicted in Figure 16.

All of the above proposals hoped to provide end-to-end QoS guarantee over DiffServ domains, all of them had their own strengths and shortcomings. The common flaws in those works are that they did not give any details on how the resources were allocated and how admission control was actually performed and, in addition, they did not consider the provisioning problem in real networks. Although those researchers tried to distribute the traffic loads evenly within networks it is likely that their methods could result in too much real-time traffic aggregating on some links, which can cause congestion in bottlenecks and increase end-to-end delay and packet loss. QoS will be degraded.

As we stated at the beginning of this paper, the motivation of DiffServ network is to satisfy the real-time traffic. Traffic of different classes is transported within DiffServ networks based on PHB. In general, EF class is used to deliver real-time traffic. EF class can provide low-delay, low-jitter and low loss service. However, some researchers ignored these following factors.

Firstly, each class has its own queue in every router buffer along the chosen path in DiffServ region. If most of the capacity of a link is allocated to the delay-sensitive real time traffic, the queue of EF class in the router buffer will become longer. It is obvious that queuing introduces latency and the potential for packet loss if a queue overflows.

Secondly, it may also be desirable to limit the performance impact of high priority traffic (EF class) on relatively low priority traffic (AF class). In real network world, the space of router buffer is finite. If there is too much EF class traffic passing some routers, packets of AF class traffic will queue at their own queues because EF class has the precedence of using resources. This will cause congestion and overflow resulting in packet loss. For some AF class traffic, the lost packets must be retransmitted. In turn, this process will deteriorate congestion.

Considering factors such as delay, jitter and packet loss, it is undesirable to carry more than a certain percentage of real time traffic on any link [29]. The rest of the available link bandwidth can be used to route other classes corresponding to delay and jitter insensitive traffic (e.g., best effort traffic). Therefore, the DiffServ network needs to be configured beforehand. Each class has its own maximum allocated bandwidth. Traffic belonging to a certain class cannot exceed this specific amount. We call this problem provisioning in DiffServ network. We will study this problem from two directions, edge provisioning and core provisioning in DifffServ networks.

Thirdly, since our study will focus on provisioned DiffServ network in which each class is allocated certain maximum bandwidth, we must consider the individually available bandwidth of each class instead of a single class. This problem will lead to QOSPF extensions and per-class QoS routing.

We try to use the end-to-end delay and throughput assurance as measures of network service performance. The studies examine cases for different network traffic loads and different kinds of applications. The study is carried out by using QoS Routing Simulator (QRS) developed by Networking Laboratory of Helsinki University of Technology [48]. We adopt a reservation mechanism similar to RSVP. We will discuss it in Chapter 4 “QoS Routing in DiffServ Networks”.

The main characteristics of QRS are:

- Unicast QoS routing protocol.
- Intra-domain routing.
- QoS routing for a network where traffic engineering is done per-class.
- Path selection according to metrics such as bandwidth and link cost
- Load balance capability.
- Available bandwidth per-class is advertised in LSU packets.
- 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 [24] to the QoS routing protocol, we call it QOSPF;

3 Provisioning in DiffServ Networks

In DiffServ networks, it is desirable that traffic in both EF class and AF class can get its requested service. EF class traffic can pre-empt resources because of its higher priority. How can we restrict the effect imposed by EF class over other lower priority classes? This is a problem associated with provisioning. We will explain this problem from the point of view of link sharing, traffic mapping principles and the queuing mechanism.

3.1 Network Provisioning

A fundamental challenge in network operation, especially in large-scale public IP networks, is to increase the efficiency of resource utilization while minimizing the possibility of congestion [35]. To fulfil this goal, one of the most critical tasks for service providers is to provision the network properly. Both congestion and starvation can be prevented and avoided to some extent simultaneously. Provisioning is crucial to the cost and revenue of the network providers. With the occurrence of DiffServ, one approach to alleviate the provisioning problems is to make certain that different class services are isolated by using different PHBs [13]. Ulrich compared IntServ networks with DiffServ networks and found that, under the same circumstances, IntServ may need considerable over-provisioning depending on the fraction of real-time traffic in networks [58].

However, because of the DiffServ architecture, efficient capacity provisioning appears more challenging than in circuit-based networks such as Asynchronous Transfer Mode (ATM) and MPLS for two reasons [57]. Firstly, there is a lack of detailed control information (e.g., per-flow state information) and supporting mechanisms (e.g., per-flow queuing) in the network. Secondly, there is a need to provide increased levels of service differentiation over a single global IP infrastructure.

In DiffServ networks, the bandwidth required by a customer for a specific service is specified by an SLA. We assume admission control is made separately at edge routers according to SLA. Packets within DiffServ networks are forwarded only on the basis of PHB. Congestion might occur in the bottleneck links and nodes. There might be three possibilities to deal with this problem:

- DiffServe network providers over-provision resources (link bandwidth, buffer space) so that in the worst case, if all traffic (conforming to SLAs from different customers) heading to one destination (let's say one customer access network), the traffic will not end up in a bottleneck. Of course, this is an extremely inefficient way of resource provisioning. In practice, this way is seldom adopted.

- Based on historically measured traffic information that tells us where the customers' traffic is going to and coming from, DiffServ network providers provision resources on a statistical average basis. The DiffServ provider is not sure that in all cases its resource provision is enough. So in this case, some requests will be rejected.

- DiffServ network providers provision networks according to historical statistics. Moreover, some traffic can be directed through lightly loaded routes with the aid of QoS routing if congestion is likely to occur. The network can avoid congestion to some extent. We will adopt this method in our work.

3.2 Link Sharing

Usually, a link is used to transport packets of both real-time and non-real-time traffic. The capacity of a link should be shared by several kinds of traffic. Link sharing is to allocate a percentage of the overall link bandwidth to each class.

3.2.1 Approaches to Link Sharing

As a rule, packets contend for the use of network resources as they are conveyed through the network. Network is considered to be in congestion if the arrival rate of packets exceeds the output capacity of the resource over an interval of time.

Congestion can result in incremental delay and packet loss, and reduce the predictability of network service. Sometimes, congestion takes place when there is too much real-time traffic which usurps most of resources. So efficient sharing of network resources by multiple traffic streams or by multiple service classes is a basic economic premise for packet switched networks in general and the Internet in particular.

As we know, two PHBs currently standardized by the IETF are Expedited Forwarding (EF) and Assured Forwarding (AF) PHB.

The AF PHB is a means for a Diffserv domain to provide a scale of reliable packet delivery assurance even during the time of network congestion. There are four defined AF classes, and within each class, packets can be marked with one of three drop-precedence levels.

The EF PHB is defined as a forwarding treatment for a particular aggregate where the departure rate of the aggregate's packets from any Diffserv node must equal or exceed the configured rate. The EF PHB is used to build a low loss, low latency, low jitter and assured bandwidth.

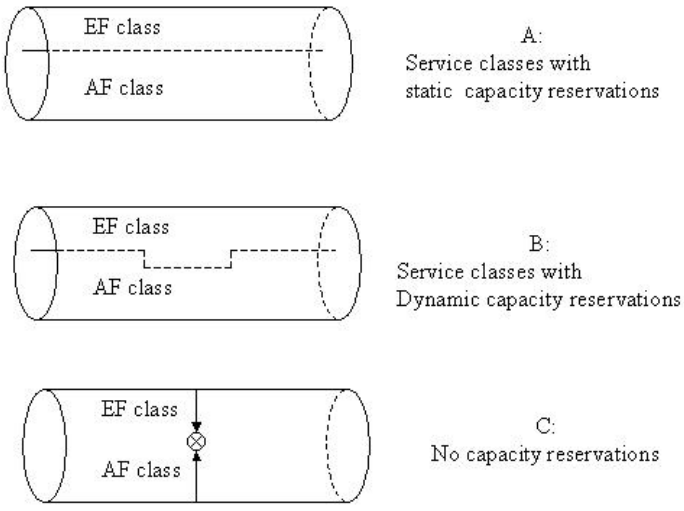

Figure 11 Three approaches to share link resources

There are several provisioning techniques, for instance, fixed and dynamic provisions. Figure 11 illustrates three cases of link sharing.

- In case A, a fixed capacity is allocated for both traffic classes. We call this method static.

- In case B, one possible way to improve the situation is to dynamically adjust the capacity of service classes based on the momentary traffic load of the steams. Some issues limit the usefulness of dynamic provision. Firstly, there is usually an ultimate limit for the total reservation because link capacity seldom can be adjusted dynamically. Secondly, reservations as such consume some resources, and they can never follow every change in traffic demand.

- In case C, a third alternative could be the key to better use of network resources. It is based on solving a conflicting situation when it really occurs rather than on proactive reservations. As long as there is enough capacity for all traffic streams, no special action is needed.

3.2.2 Our Tentative Link Sharing Approach

Some researchers argue that dynamic provisioning techniques are desirable because traffic volumes are likely to change dynamically. Nevertheless, those dynamic methods are too complicated to be implemented in reality. At the same time, if the fraction of real-time traffic is increasing the end-to-end delay and packet loss are increasing accordingly [58]. Therefore, it will be rational that real-time traffic does not surpass “certain” percentage of each link capacity. This “certain” value can be obtained by experiments and is beyond our interest.

Based on the above rationale, we will prefer case A in the preceding section. Within DiffServ networks, each link is virtually split into parts, each with a fixed percentage of total capacity of this link. One part for EF class traffic and the other for AF class traffic. The sum of those two parts is equal to the total capacity of this link.

In Figure 12, we present an example that may not occur in practice but still can serve our purpose. EF class allocation amounts to 10-20% of the capacity of a link and AF class is 80-90%. In some cases, AF class can consist of a couple of sub-classes such as AF1 and AF2. Under this assumption, AF1 class is assigned 30-40% of the total capacity of a link and AF2 about 50%. The number of classes will be no more than three in our simulations. Whatever allocations, the sum of percentage of all service classes is equal to 100 percent. We classify Best Effort into AF2 to make sure that the BE class is never completely blocked by the higher priority classes.

EF and AF are two distinct traffic classes with different priorities (in Figure 12). EF class precedes AF class. All arriving packets at the core DiffServ router are assigned to one of the leaf classes (such as video and voice); the interior classes (such as EF and AF) are used to designate guidelines about how “excess ” bandwidth should be allocated.

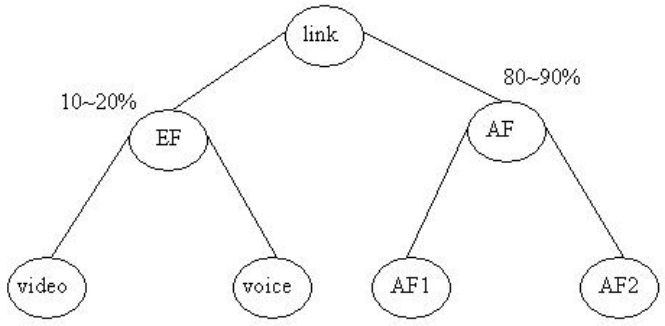

Figure 12 link sharing structure

3.3 Traffic Mapping Principles

By considering the diversity of traffic specifications and QoS requirements and limitation of service classes in DiffServ, it is essential to map incoming traffic to Diffserv classes by some rules.

3.3.1 Traffic Delivery Requirement

Increasingly, the Internet will have to function in the presence of different classes of traffic with different service requirements. The advent of DiffServ makes this requirement particularly acute. Thus packets might be grouped into behavior aggregates such that each behavior aggregate may have a common set of behavior characteristics or a common set of delivery requirements, in practice, the delivery requirements of a specific set of packets may be specified explicitly or implicitly. Two of the most important traffic delivery requirements are capacity constraints and QoS constraints.

Capacity constraints can be expressed statistically as peak rates, mean rates, burst sizes, or as some deterministic notions of effective bandwidth. QoS constraints can be expressed in terms of packet loss, delay and delay variation (Table 1).

	Traffic specs			QoS specs		
	Mean rate	Burst	Peak rate	Delay	Jitter	Packet loss
Video	High	Low/ Medium	High	Low/ Medium	Low	Low/ Medium
VoIP	Medium	Low	Low/ Medium	Low	Low	Medium
FTP	Low	High	Low/ High	Low/ High	Low/ High	Low
Web	Low/ Medium	High	Low/ Medium	Medium	Low/ High	Low
Email	Low	Medium	Low/ Medium	Low/ High	Low/ High	Low

Table 1 Typical Traffic specification and QoS specification [62]

In table 1, “low” stands for small numerical values and “high” stands for large numerical values. For example, if the delay for voice is greater than 150ms QoS starts

to degrade and we show “low” in table 1. However, for either FTP or web, the user’s tolerance to delay is about 2s and we show it “low/high” in table 1.

3.3.2 Traffic Mapping Policy

In real scenarios, two stream types compose the real time traffic: video streams and audio streams (VoIP). Although both streams have similar network requirements, they have different demands regarding delay and traffic specification (Table 1). Moreover, the video stream is transmitted continuously for the whole duration of a videoconference, while the audio stream is transmitted only during the periods that a user speaks. Therefore, audio sources can be modelled as an ON/OFF CBR (Constant Bit Rate) source.

Apart from the real-time traffic, there is still non-real time traffic, for instance, ftp, web and email. Their traffic specifications and QoS specifications are also listed in Table 1.

By observing Table 1, different types of traffic have different traffic specifications and QoS requirements. In order to implement provisioning in DiffServ networks, we must map incoming traffic to DiffServ classes. Traffic mapping pertains to the assignment of traffic workload onto pre-established paths to meet certain requirements. Thus, while QoS based routing deals with path selection, traffic mapping deals with the assignment of traffic to established paths which may have been selected by QoS based routing or by some other means [35].

An important aspect of the traffic mapping function is the ability to establish multiple paths between an originating node (ingress node) and a destination node (egress node), and the capability to evenly distribute the traffic between the two nodes across the paths according to some policies. A pre-condition for this scheme is the existence of flexible mechanisms to partition traffic and then assign the traffic partitions onto the parallel paths. In our work, we use our proposed mechanism PERD (Section 4.3 “Proposed solution”) to distribute traffic of the same class onto different routes.

In our simulations, we map real time traffic (voice and video) to EF class, and other non-real time traffic (ftp and email, etc.) to AF class.

3.4 Provision and Allocation Policies

We need to take into account the edge provisioning and interior provisioning simultaneously. In this section, we present a two-tier architecture to illustrate roles of Bandwidth Brokers for provisioning.

3.4.1 The Goal of Provision in DiffServ

Determination of resources required at each node for every service class needs the estimation of volume of traffic that will traverse each network node. The boundary provisioning is the easy part of the issue, in particular in the direction from the boundary node to the core network since the admission control is made at the boundary nodes. On the other hand, the much harder question is the interior provisioning because the traffic volumes cannot be anticipated with 100% accuracy, especially, if an explicit route has not been selected. Moreover, there is a possibility that bottleneck links exist within networks.

In order to overcome congestion and utilize resources efficiently, provisioning in a DiffServ network does not only mean determination and allocation of resources at various points in the network, but also modification of existing resources to be shared dynamically among various classes. Both EF class service and AF class service are required to be provisioned at the network boundaries and in the network interior.

Customers have SLAs with their ISPs. The SLAs will specify the amount of bandwidth allocated for customers. Customers are responsible for deciding how their applications share that amount of bandwidth. Given the SLAs, ISPs must decide how to configure their boundary routers so that they know how to handle the incoming traffic.

The simple situation is for static SLAs. Routers can be manually configured with the classification, policing and shaping rules by administrators. Resources are therefore

statically allocated for each customer. Unused resources can be or cannot be shared by other customers according to provision policies.
The situation will be more complex for dynamic SLAs. Resource allocation is closely related to the signaling process [26]. The BB in the customer domain uses signaling (e.g., RSVP) to request for resources from its ISP. At the ISP side, the admission control can be made either in a distributed manner by the boundary routers or in a centralized manner by a BB. If boundary routers are directly involved in the signaling process, they are configured with the corresponding classification, policing and shaping rules when they grant a request. If a BB is involved rather than the boundary routers, then the BB must configure the boundary router when it grants a request.

3.4.2 Roles of Bandwidth Brokers for Provisioning

In order to provide end-to-end QoS, we refer to the two—tier network model [39]. Each domain has a BB that manages resources within this domain. We prefer BB because it is not only capable of performing dynamic end-to-end admission control but also capable of managing and provisioning network resources of a separately administered DS domain and cooperating with other similar domains [40].

In our architecture, the local admission decisions are made independently at the edge routers of each domain. The BB in each domain will be responsible for periodically updating the available allocation of resources inside the domain, according to some measurements of the traffic load at the edge routers (Figure 13). The QoS reservation states are stored at and managed solely by BB in a network domain. When all allocation of resources is completed, all the edge routers will be able to make instantaneous and independent admission control decisions for new connection requests.

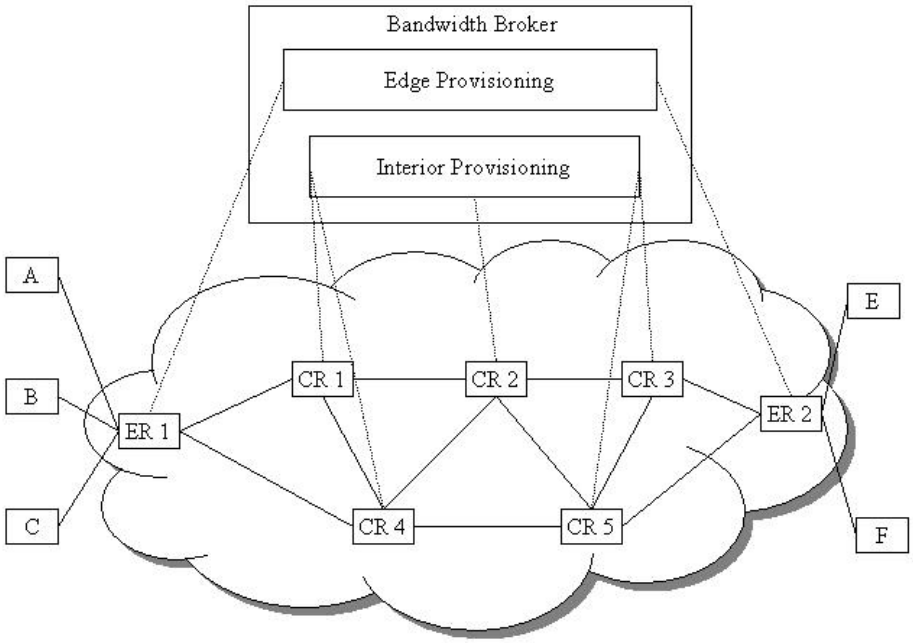

Figure 13 Layered provisioning view of DiffServ network

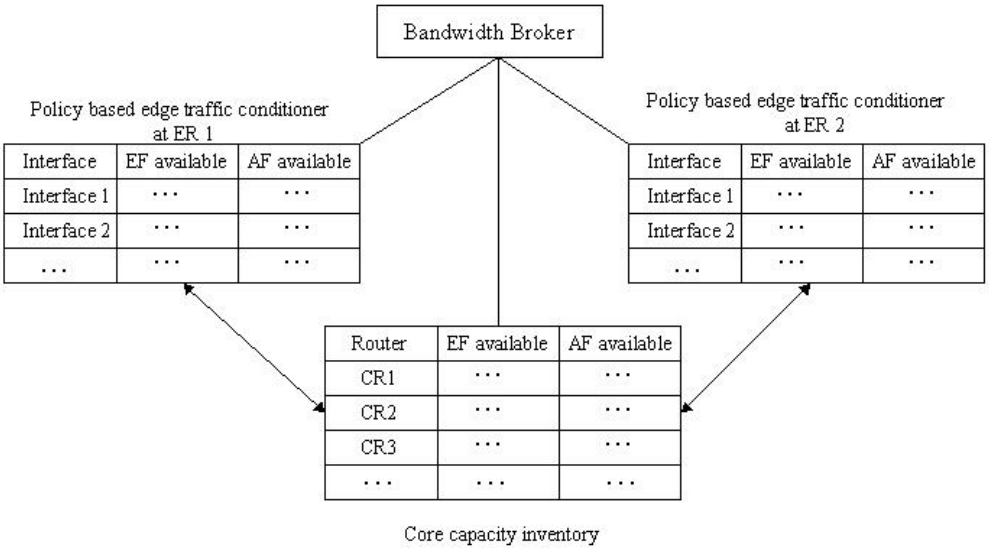

Figure 14 Resource table kept in BB

Based on the basic needs of provisioning a DiffServ network to support different class services, we consider the provisioning as a two layered model --the top layer responsible for edge provisioning and driving the bottom layer which is in charge of interior provisioning (Figure 13).

In DiffServ networks, edge provisioning drives interior provisioning since SLAs are contracted at the boundaries (Figure 14). These are coupled with each other to a high degree in a way that each has direct influence on the other and it would not make much sense to offer guarantees only at the edges which are not met in the interior.

An edge router selects an explicit route and signals the path through the network. Router interfaces along these routes are pre-configured to serve a certain amount of EF or AF class traffic.

Like edge nodes, only a specific amount of bandwidth will be allocated to a request in each interior node. If a connection is accepted at the edge but it does not find enough resources provisioned for EF class in the network, the connection will be finally rejected.

It might seem that like IntServ approach, a connection is established by sending a signaling message to reserve resources for the new flow at each hop along the path, but capacity reservation states are actually stored in a BB capacity inventory and not in the core routers. Therefore, unlike the traditional IntServ, which has the fundamental scalability limitations because of the responsibility to manage each traffic flow individually on each of its traversed routers.

3.5 Queuing Mechanism in Diffserv Networks

In a provisioned DiffServ network, each class service has its allocated resources. Moreover, each class service has its own packet queue in the buffer. Therefore, a mechanism is needed to schedule packets from different classes.

3.5.1 Comparisons of Different Queuing Mechanisms

As a rule, mechanisms that perform the traffic mapping functions should aim to map the traffic into the network infrastructure to minimize congestion. If the total traffic load cannot be accommodated, or if the routing and mapping functions cannot react fast enough to changing traffic conditions, then a traffic mapping system may rely on short time scale congestion control mechanisms (such as queue management,

scheduling, etc.,) to mitigate congestion. Thus, mechanisms that perform the traffic mapping functions should be complemented with existing congestion control mechanisms.

In general, service disciplines can be classified as either work-conserving or nonwork-conserving [15]. In the former, the packet transmission in never idle as long as there is a packet in the queue. While in the latter, the packet transmission may hold even there are packets waiting for departure. The work-conserving discipline features larger delay variation (jitter) than nonwork-conserving discipline, but it provides lower delay and higher bandwidth utilization.

Several work-conserving disciplines have been well discussed in the literatures. Different queuing mechanisms have been proposed starting from simple ones as Priority Queuing (PQ) to more advanced as Weighted Fair Queuing (WFQ) [16] and Class-Based Queuing (CBQ) [17].
PQ is a basic scheme that simply allows high priority traffic first access to available bandwidth; it provides no means of controlling the allocation of bandwidth. In addition, PQ often results in all but the highest priority applications being completely starved of bandwidth.

WFQ overcomes PQ's limitations by providing for each of a small number of traffic classes a fixed proportion of bandwidth. Its drawbacks are that classification is complex and limited; it does not scale in number of granularity of classes and does not ensure explicit rate control for traffic classes [60].

CBQ was developed as a progression of earlier efforts such as WFQ referenced in the IETF's DiffServ. CBQ will be used as the queuing mechanism in this paper.

3.5.2 CBQ Mechanism

The CBQ mechanism is based on the notion of controlled link-sharing. Moreover, the user traffic is organized into a tree, or hierarchy, of classes. A class can be an individual flow or an aggregate of flows representing different applications, users, organizations, or protocol families. A class is defined by standard IP information such

as address, protocol (e.g., TCP and UDP) and application (e.g., ftp, telnet, etc.). Each traffic class is then assigned a committed bandwidth rate and a priority value. At the same time a separate queue is maintained for each traffic class to ensure individual traffic class service requirements are satisfied (Figure 15).

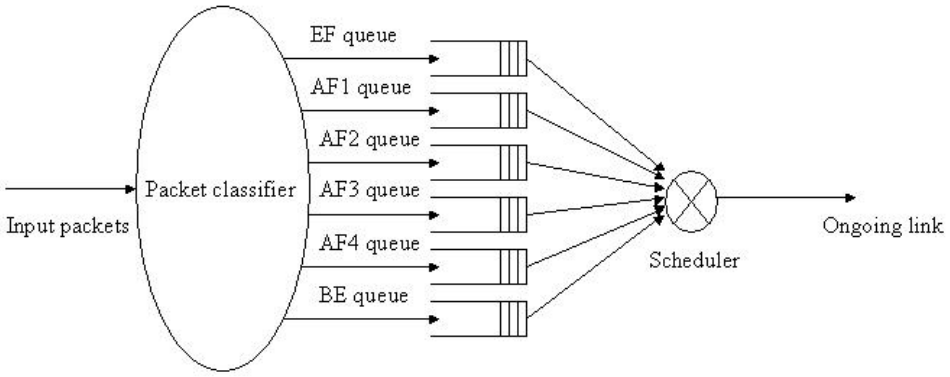

Figure 15 CBQ scheduler

CBQ unifies a number of essential elements such as packet classification, packet scheduling and bandwidth management. The packet scheduling of CBQ is decomposed into two types of scheduler, the general scheduler and the link-sharing scheduler. The general scheduler is a priority-based scheduler and determines exclusively the scheduling of the packets in the absence of congestion. On the contrary, in the presence of congestion the link-sharing scheduler controls the scheduling of the packets from different priority classes. The aim of link-sharing scheduler is twofold:

- Firstly, to assure that each class will receive its allocated bandwidth over appropriate time intervals.
- Second, to distribute any "excess" bandwidth following some reasonable guidelines.

The overall working of CBQ can be described as follows.

When a packet arrives, it is firstly classified. If the traffic class has not yet used all of its bandwidth (underlimit class), the packet flows immediately to the outbound link

and will only experience store and forward router-like latencies by the general scheduler. If a packet arrives and the class is attempting to use more than its committed bandwidth rate (overlimit class), the link-sharing scheduler is activated and there are two possibilities.

Firstly, the packet is placed in its queue and then it is either rate shaped onto the outbound link or it is allowed to “borrow” from the currently idle bandwidth of any other traffic class. Borrowing is a unique feature of CBQ that provides bandwidth cost savings with extremely efficient sharing of link bandwidth. Each class can be specifically authorized to borrow, or not borrow, from any idle bandwidth on the link. If a class is allowed to borrow, it can burst above its committed rate to meet periodic bursts on demand.

4 Per-class QoS Routing in DiffServ Networks

In DiffServ networks, there is a limited set of traffic classes and packets are classified by DSCP. Core routers forward packets based on PHB instead of per traffic flow. For resource utilisation efficiency in provisioned DiffServ networks, not only the total residual bandwidth but also per-class available bandwidth of any link must be taken into consideration. Hence, OSPF is extended to support per-class traffic engineering.

4.1 Per-class Routing Concept

DiffServ is becoming prominent in providing scalable networks supporting multiple classes of services. It is necessary to perform traffic engineering at a per-class level instead of an aggregation level in order to further enhance networks in performance and efficiency. Per-class TE (traffic engineering) is independently proposed by Le Faucheur, et. al. [54] in which they referred the scheme as DS-TE.

4.1.1 Traffic Engineering in DiffServ Environments

Traffic engineering is concerned with performance optimization of operational networks. In general, it encompasses the application of technology and scientific principles to the measurement, modelling, characterization, and control of Internet traffic, and application of such knowledge and techniques to achieve specific performance objectives.

4.1.1.1 Traffic Engineering Performance Objectives

The key performance objectives associated with traffic engineering can be classified as being either [55]:

- Traffic oriented or
- Resource oriented.

Traffic oriented performance objectives include those aspects that enhance the QoS of traffic streams. In a single class best effort Internet service model, the key traffic

oriented performance objectives include: minimization of packet loss, minimization of delay, maximization of throughput, enforcement of service level agreements and so on.

Resource oriented performance objectives include the aspects pertaining to the optimization of resource utilization. Efficient management of network resources is the vehicle for the attainment of resource oriented performance objectives. In particular, it is generally desirable to ensure that subsets of network resources do not become over-utilized and congested. On the other hand, other subsets along alternate feasible paths remain under-utilized. Bandwidth is a crucial resource in contemporary networks. Therefore, a central function of traffic engineering is to efficiently manage bandwidth resources.

Minimizing congestion is a primary performance objective of traffic and resource oriented traffic engineering. The interest here is on congestion problems that are prolonged rather than on transient congestion resulting from instantaneous bursts. Congestion typically manifests under the following two scenarios:

- When network resources are insufficient or inadequate to accommodate offered traffic load.

- When traffic streams are inefficiently mapped onto available resources; causing subsets of network resources to become over-utilized while others remain under-utilized.

The first type of congestion problem can be addressed by either:

- Expansion of capacity, or
- Application of classical congestion control techniques, or
- Both.

The second type of congestion problems, namely those resulting from inefficient resource allocation, can usually be addressed through traffic engineering.

In general, congestion control resulting from inefficient allocation can be reduced by adopting load balance policies. The objective of such strategy is to minimize maximum congestion or alternatively to minimize maximum resource utilization, through efficient resource allocation. When congestion is minimized through efficient resource allocation, packet loss decreases, end-to-end delay decreases, and aggregate throughput increases. Thereby, the perception of network service quality experienced by end users becomes significantly enhanced.

4.1.1.2 Available Resources Based on Per-class

In most previous works, researchers mainly considered the entire residual bandwidth of any link regardless of per-class level. This resulting admission control is simple. At the edge router or bandwidth broker, the total residual bandwidth is checked if it satisfies the requirement of the applications. If satisfied, this request is admitted. If not, this request is rejected. Nevertheless, in provisioned DiffServ networks, this method is neither optimal nor efficient because each class has its maximum allocation bandwidth. This method cannot reflect the individually available bandwidth per-class.

In provisioned DiffServ networks, each class has its own queue in every router buffer along the chosen path. EF class is the highest priority and it can preempt resources. Delay-sensitive real-time traffic is delivered with EF class service. However, if most of the capacity of a link is allocated to the delay-sensitive real time traffic, the queue of EF class in the router buffer will become longer. It is obvious that queuing introduces latency and the potential for packet loss if a queue overflows.

Moreover, if there is too much high priority traffic (e.g., EF class traffic) passing some routers, packets of low priority traffic (e.g., AF class traffic) may result in long queues because EF class has the precedence of using the resources. In real network world, the space of router buffer is finite. This will result in congestion and some packets will be discarded. For some AF class traffic, the lost packets must be retransmitted. In turn, this process will increase congestion.

Like in edge nodes, only a specific amount of bandwidth will be allocated to each class service in each interior node. If a connection is accepted at the edge node but it

does not find enough resources at any of the interior nodes, the connection request will be finally given up.

We can illustrate this problem with a simple example with reference to Figure 9.

We assume there is an EF class request for 0.3Mbps at the ingress node E1. The residual bandwidth at that moment is like this: link6-2 is 1.5Mbps and link4-6 is 1.0Mbps. However, the available bandwidth of EF class within those two links is different. The values are 0.2 Mbps and 0.5Mbps in link6-2 and link4-6 respectively. If the widest bandwidth algorithm (WB) is used and the path is chosen depending on the entire residual bandwidth, then link6-2 is the optimal. But in this case, it will not guarantee the required QoS. If the routing algorithm chooses the route on the basis of the individual residual bandwidth per-class, then link4-6 is better and it can satisfy the bandwidth requirement.

Furthermore, in a dynamic network, traffic is connected or disconnected from time to time. The available resources (both entire residual bandwidth and the individual available bandwidth per class) are varying along the amount of traffic. This kind of fluctuation of both traffic and residual resources makes QoS routing more complex.

4.1.1.3 Traffic Engineering on Per-class Basis in DiffServ Networks

Traffic engineering can be used as a complement to DiffServ mechanisms to improve utilization of network resources. It can be operated on an aggregate basis across all service classes or on a per-service class basis. The former is used to provide better distribution of the aggregate traffic load over the network resources [53]. The latter case is specific to the DiffServ environment, with so-called DiffServ-aware traffic engineering [54].

The first option is typically used when aggregate traffic engineering is deployed using current MPLS TE [30] mechanisms. The MPLS architecture allows aggregation. In that case, traffic from all aggregates is routed collectively according to a single shared set of constrains and will follow the same path.

Further aggregation may cause problems in bottleneck links within a network. Most of the capacity of a link may be allocated to the delay-sensitive real time traffic. The queue of EF class in the buffer of the bottleneck router will become longer. Queuing introduces latency and the potential for packet loss if a queue overflows.

In contrast with the first option, the second option is to split traffic from the different aggregates into multiple traffic paths. In other words, traffic, from a given source node to a given destination node, is split into multiple traffic routes on the basis of the available resources per class and the QoS requirements of each traffic class. Those split traffic can potentially follow a different path through the network. In so doing, DS-TE [29] can take into account the specific requirements of the traffic class transported on each route (e.g., bandwidth requirements). Moreover, DS-TE can take into account specific engineering constraints to be enforced for these sets of traffic trunks (e.g., limit all traffic trunks transporting a given set of aggregates to x% of link capacity, see Section 3.2 “Link sharing”). In brief, DS-TE achieves per-class constraint based routing with paths that tightly match the specific objectives of the traffic class.

Through our example and explanation, we know that for some DiffServ networks, it may be desirable to control the performance of some service classes by enforcing certain relationships between the traffic workload contributed by each service class and the amount of network resources allocated or provisioned for that service class. For example, such relationships between demand and resource allocation can be enforced using a combination of methods [35]:

- Traffic engineering mechanisms on a per-service class basis that enforce the desired relationship between the amount of traffic contributed by a given service class and the resources allocated to that class.

- Mechanisms that dynamically adjust the resources allocated to a given service class to relate to the amount of traffic contributed by that service class.

It may also be desirable to limit the performance impact of high priority traffic on relatively low priority traffic. This can be achieved by, for example, controlling the

percentage of high priority traffic that is routed through a given link. Another way to accomplish this is to increase link capacities appropriately so that lower priority traffic can still achieve adequate service quality. The latter method is beyond our discussion.

In summary, it is necessary to perform traffic engineering at a per-class level instead of an aggregate level. Performing traffic engineering on a per-class basis may require certain per-class parameters to be distributed (we will discuss this in Section 4.2 "QoS-enabled OSPF routing QOSPF"). Note that it is common to have some classes to share some aggregate constraint (e.g. maximum bandwidth requirement) without enforcing the constraint on each individual class. These classes then can be grouped into a class-type so as to improve scalability. It also allows better bandwidth sharing between classes in the same class-type. A class-type is a set of classes that satisfy the following two conditions:

- Classes in the same class-type have common aggregate requirements to satisfy required performance levels.

- There is no requirement to be enforced at the level of individual class in the class-type [35].

4.2 QoS-enabled OSPF Routing QOSPF

In QOSPF, routers advertise network resource information as well as topology information. A route is calculated based on topology, network resource information, and QoS requirements.

4.2.1 The OSPF Protocol

Open Shortest Path First (OSPF) [33] is a widely deployed link state routing protocol. The most important characteristic of link state routing protocols is that each router maintains the full topology of the network in a link state database. The OSPF standard specifies that routers implementing the protocol run the shortest path Dijkstra computation on their local link state database, and determine the shortest paths to all

other nodes in the network. The database is constructed and updated by means of link state advertisements, which are generated by each router and propagated to all other routers using reliable flooding.

The flooding procedure utilizes a variety of packet types: Link State Update (LSU) packets contain information about changes in the topology, and are used to carry multiple Link State Advertisements (LSAs). Link State Acknowledgement packets are used to acknowledge receipt of link state advertisements.

4.2.2 QoS Routing Extensions

QoS-enabled OSPF (QOSPF) protocol is being standardized in IETF, extending the OSPF protocol to support QoS link state parameters. In QOSPF, routers advertise network resource information as well as topology information. The network resource information includes residual link resources on a router as well as existing link resource reservation on the router.

The QoS routing extensions to OSPF are based on two key ideas [34]:

- Enhancing the link state advertisements and the topology database to include network resource information (for example, available bandwidth).

- Using an alternate route computation algorithm to compute routes that take this resource information into account.

Two encoding schemes for OSPF QoS extensions have been proposed: [18]: type-of-service (TOS)-metrics-based encoding [21] and opaque-LSA (link state advertisement) encoding [22]. Although the TOS-metrics-based encoding can support backward compatibility, it restricts the encoding of extended parameters and does not have sufficient flexibility to accommodate future extensions. We therefore chose the opaque LSA encoding for the proposed QoS LSA.

QOSPF is designed to collect and maintain the QoS topology map used for QoS path computation. The resource information maintained by a BB or a router can be shown

in Table 2, where $C\ (i,k)_{residual}$ denotes the unreserved capacity of router i's kth interface.

	Interface 1	Interface 2	……	Interface k
Router 1	$C\ (1,1)_{residual}$	$C\ (1,2)_{residual}$		$C\ (1,k)_{residual}$
Router 2	$C\ (2,1)_{residual}$	$C\ (2,2)_{residual}$		$C\ (2,k)_{residual}$
……				
Router m	$C\ (m,1)_{residual}$	$C\ (m,2)_{residual}$		$C\ (m,k)_{residual}$

Table 2 Resource Table in a BB

The information in the QoS resource table is used to identify paths capable of satisfying the bandwidth requirements of new requests. This is requested by a new flow to the available bandwidth in successive entries in the row associated with the flow's destination. The search stops at the first entry with an available bandwidth value corresponding to implemented routing algorithm, at which point the corresponding next hop is returned and used to determine the next hop on which to forward the request.

Based on this information, a BB can make admission control by querying routers in the topology map. The end-to-end admission control can be presented as follows:

For all the nodes along the candidate route $P = \{R_1,\ldots,R_m\}$:

If $(C\ (m,k)_{residual}) >= C_{request}$ *){*

accept connection request;

$C\ (m,k)_{residual} = C\ (m,k)_{residual} - C_{request}$ *;*

}

else

Reject connection request;

4.2.3 Per-class QOSPF Extensions

In [23], OSPF is extended to support traffic engineering by using opaque LSAs. The extension provides a way of describing the traffic engineering topology (including bandwidth and administrative constraints). The draft defines a new LSA, i.e., traffic engineering LSA. The LSA contains link TLV (Type/length/value), which describes a single link and includes the following sub-TLVs.

- Link type (1 octet)
- Link ID (4 octets)
- Local interface IP address (4 octets)
- Remote interface IP address (4 octets)
- Traffic engineering metric (4 octets)
- Maximum bandwidth (4 octets) specifying the maximum bandwidth that can be used on this link in this direction. This is the link capacity.
- Maximum available bandwidth (4 octets) specifying the maximum bandwidth that can be reserved on this link in this direction. The default value should be the Maximum bandwidth.
- Unreserved bandwidth (32 octets) specifying the amount of bandwidth not yet reserved at each of the eight priority levels. The values correspond to the bandwidth that can be reserved with priority of 0 through 7, arranged in increasing order with priority 0 occurring at the start of the sub-TLV and priority 7 at the end of the sub-TLV.
- Resource class/colour (4 octets)

The draft [24] proposes corresponding extensions to OSPF for support of traffic engineering on a per-class type basis. The draft adds new sub-TLVs on the basis of [23]:

- TBD1-Unreserved bandwidth for class-type 1 (32 octets)
- TBD2-Unreserved bandwidth for class-type 2 (32 octets)
- TBD3-Unreserved bandwidth for class-type 3 (32 octets)

Meanwhile, sub-TLV 7 (maximum available bandwidth) is referred to “Maximum Available Aggregate Bandwidth”. Sub-TLV 8 (unreserved bandwidth) is referred to “Unreserved Bandwidth for class-type 0”.

4.3 Proposed Solution

In this section, we propose a routing mechanism named per-class routing based on per-class dissemination (PERD).

4.3.1 PERD Principles

In a DiffServ network, a class in PERD can represent a PHB or a group of PHBs, for example, given 2 PHBs, PERD may use one class to represent the two PHBs.

Network resources for each class are maintained in an extended link state database that contains link resources for each class. When resources of a class change on a link, new link state is advertised to the network.

If one class represents a group of PHBs, the network resources of the class represent the aggregate of network resources.

In DiffServ networks, PHBs are realized by traffic scheduling mechanisms. Hence, the resources of PHBs can map to the parameters of the scheduling mechanisms, for example, resources can refer to maximum bandwidth, available bandwidth, and buffer space of queues.

Route computation is carried out for each class and results in route entries for each class. For different classes, different route computation mechanisms can be applied but the computation should be able to reach the optimization of the network as a whole.

Nevertheless, it is well known that route computation with more than one constraint is normally NP-hard, heuristic algorithms should be developed [36].

4.3.2 Our Proposed Model

As we described, some papers proposed a BB to manage the resources within each DiffServ domain and make local admission control decisions. Although the centralized approach removes the burden of admission control from core routers, there might be some scalability considerations if the BB has to process hundreds and thousands of requests per second. Moreover, this approach has certain disadvantages that are inherent to any centralized architecture.

- The links around the BB will become very congested when the traffic load from the signaling messages is high.

- The BB must maintain per-flow information about every flow that is currently active inside this domain.

- The BB is a single point of failure (i.e. undesirable due to reliability considerations).

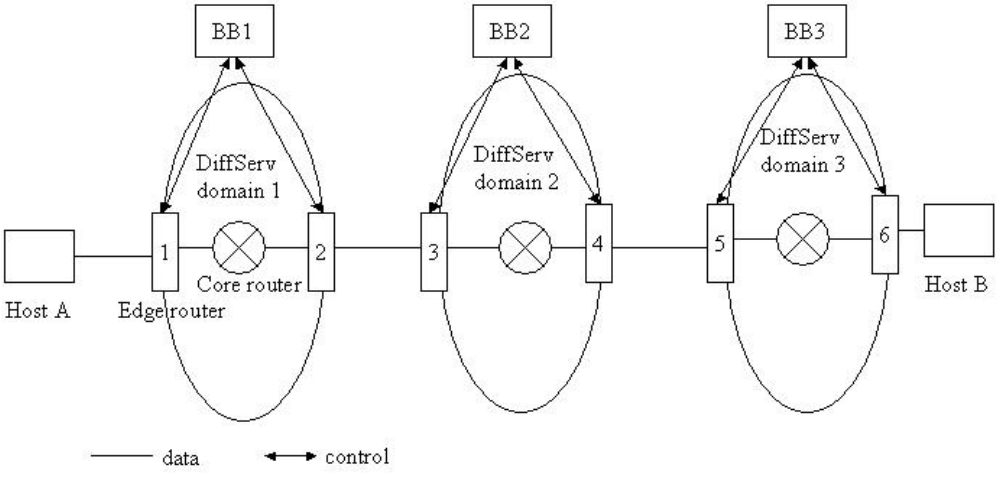

Figure 16 Proposed QoS DiffServ architecture

In this paper, we will adopt an alternative distributed approach like paper [32]. Throughout this paper we will assume that the Internet consists of several

independently administered DiffServ domains, which are interconnected in order to provide global connectivity.

Our proposed model is in Figure 16. Each domain consists of a BB, core and edge routers. The BB will exchange control messages with the edge routers for the purpose of resource management.

The local admission decisions are made independently at the edge routers of each domain. The BB in each domain will be responsible for periodically updating the allocation of resources inside the domain, according to some measurements of the traffic load at the edge routers and provision polices. When all allocation of resources is completed, all the edge routers will be able to make instantaneous and independent admission control decisions for new connection requests. QOSPF is used to broadcast link state changes, e.g., changes of available bandwidth of a class on a link.

Based on the earlier discussions about per-class QOSPF extensions and PERD routing, the resource information maintained by an edge router or a BB can od shown in Table 3.

	Interface 1	Interface 2	……	Interface k
Router 1	$C(1,1)_{EFavailable}$	$C(1,2)_{EFavailable}$		$C(1,k)_{EFavailable}$
	$C(1,1)_{AFavailable}$	$C(1,2)_{AFavailable}$		$C(1,k)_{AFavailable}$
Router 2	$C(2,1)_{EFavailable}$	$C(2,2)_{EFavailable}$		$C(2,k)_{EFavailable}$
	$C(2,1)_{AFavailable}$	$C(2,2)_{AFavailable}$		$C(2,k)_{AFavailable}$
……				
Router m	$C(m,1)_{EFavailable}$	$C(m,2)_{EFavailable}$		$C(m,k)_{EFavailable}$
	$C(m,1)_{AFavailable}$	$C(m,2)_{AFavailable}$		$C(m,k)_{AFavailable}$

Table 3 Generalized Resource Table for end-to-end connection Admission Control

$C(m,k)_{EFavailable}$ denotes the EF class available bandwidth of router *m*'s *k*th interface. Similarly, $C(m,k)_{AFavailable}$ denotes the AF class available bandwidth of router *m*'s *k*th

interface. $C(m,k)_{EFavailable}$ and $C(m,k)_{AFavailable}$ corresponds to "Unreserved bandwidth" in Section 4.2.3 "Per-class QOSPF Extensions".

We compare Table 3 with Table 2. We replace residual bandwidth in Table 2 with available bandwidth per class in Table 3. When a request arrives at an edge router, this router judges which service class this request belongs to from the PATH message and makes admission control. It selects an explicit route and signals the path through the network. If this request is for EF class service, this edge router will check if the available bandwidth of EF class in the resource table from the ingress node to the egress node can satisfy the requirement. If the available resources can meet the requirement, the connection will be established. If not, the request will be rejected. If this request is for AF class service similar processes will occur.

Supposing source host A in domain-1 in Figure 16 wants to establish a connection to the destination host B in domain-3. Then the intra-domain admission control at domain-1 will take place as follows:

The source node will send a PATH message to edge router-1which will include the required amount of bandwidth *B* and class service index of *X*, for instance, EF class or AF class.

Edge router-1 will know that the destination node is in domain-3, so it will check whether there are enough resources to carry this class service towards edge router-2.

In particular, for all the nodes along the route $P = \{ER_1 \ldots R_m \ldots ER2\}$:

If $((C(m,k)_{Xavailable}) >= B)$ *{*

Accept connection request;

$(C(m,k)_{Xavailable}) = (C(m,k)_{Xavailable}) - B;$

}

else

Reject connection request;

X stands for EF or AF class, in our simulation, we use numbers 0 and 1 to substitute EF and AF respectively.

Having introduced the intra-domain admission control, we move on to illustrate briefly how to perform admission control over multiple domains. We start from the point where the PATH message is sent to edge router-2 of domain-1.

- Edge router-2 will forward the PATH message to edge router-3 in domain-2.
- Edge router-3 will perform the intra-domain admission control in a way identical to the one described previously.
- If the request is accepted, this PATH message will be forwarded to the next domain until to the destination.
- The destination node will send the RESV message or reject message to the source node.
- While the RESV message travels back to the source node, all the intermediate edge routers will configure their traffic shapers and markers to account for the new connection.

4.3.3 Example Implementation of PERD

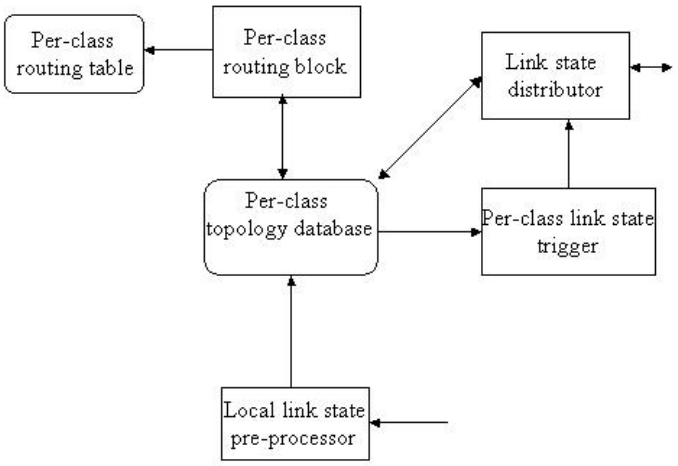

Figure 17 Implementation of PERD

Figure 17 gives an example of PERD based on a link state protocol--per-class extended OSPF [62].

Each edge router has the function blocks such as “local link state pre-processor”, “per-class routing block”, “per-class routing table” and “per-class topology database”. When a request comes, the edge router will be responsible for finding the feasible route for this coming request. One of the key functions of the “local link state pre-processor” is to transform various local link state information into the variables that can be used by routing. “Per-class routing block” tries to find the feasible route for this request by checking the available resources. If possible, a per-class routing table is created.

“Per-class topology database” keeps the network topology information based on per-class. “Per-class link state trigger” determines when “per-class topology database” advertises its state information. In addition, a trigger mechanism can be applied in order to reduce advertisement costs. There are four LSU algorithms: period based (PB), threshold based (TB), equal class based (ECB) and unequal class based (UCB) updating [63].

4.3.4 PERD Routing Algorithms

There are various routing algorithms for a single class routing (e.g., WB, LC), which can be naturally used in PERD.

For example, we introduce two simple routing algorithms: Lowest Cost (LC) algorithm and Widest Bandwidth (WB) algorithm [51].

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,…,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 WB 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.*

Then, we present the lowest cost algorithm as follows:

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 when either node *d* is included by *L* or the cost exceeds the threshold before reaching node *d.*

In addition, the WB algorithm is given below**:**

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 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.

Step1 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 addition, there is likely to be a need of differentiating routing algorithms for different classes. This is because different classes have different QoS demands. Hence, one example is to apply a separate routing algorithm for each of EF class, AF class. For example, delay-sensitive real-time traffic can adopt the shortest-path or WB routing algorithm. For AF class, it can use the WB algorithm.

5 Simulations

In this section, we will describe simulations we have run with different network topologies. First, we briefly introduce simulation environments; then we will perform our simulations with several steps. During each step, we will present a different network topology. The network topology will become slightly more complicated from the beginning to the end. In the last part of this section, we will compare performances of different routing algorithms—(SP-Shortest path algorithm, WB-Widest bandwidth algorithm and PERD).

5.1 Simulation Environments

Before simulations, we first describe the QRS simulator and metrics we will use to judge performance.

5.1.1 Traffic Flows

In the QRS simulator, traffic flows are generated in traffic sources connected to source nodes and they sink in traffic destinations connected to destination nodes. We can adjust the transmitting rate by setting different inter-departure times between consecutive packets. QRS simulator can generate traffic of different classes in accordance with our intentions; moreover, we can prescribe the period of duration of traffic, packet size and starting time of traffic flows.

There are three independent traffic classes: EF, AF1 and AF2. At the ingress node, traffic packets are classified based on the class number specified in our configuration files. After classification, packets will be put into corresponding queues based on their class numbers. In our simulations, we simply regard the class number as the DSCP making.

5.1.2 Metrics

We use the end-to-end delay and throughput assurance as measures of network service performance. The studies examine cases for different network traffic loads and different kinds of end users/applications.

The length of load packet is 512 bytes and the header is 32 bytes long. When we calculate the throughput we regard the length of the packet as 544 bytes. We record the number of received packets at the destination. The unit of throughput is bps.

End-to-end delay is the time duration a packet travels from the source node to the destination node. For real-time traffic, only throughput is not enough, we must take end-to-end delay into consideration. The unit of delay in this paper is ms.

We calculate end-to-end delay by: Delay = packet's receiving time-packet's sending time. We record the packet's sending time when the first bit of a packet leaves a source. Similarly, we record the packet's receiving time when the last bit of this packet arrives its intended destination. It is composed of propagation delay, transmission delay and queuing delay.

We assume that the propagation delay is a constant and is equal to 10ms. In addition, Transmission delay = packet size/bandwidth. Queuing delay occurs inside the network, since routers generally need to store packets for some time before forwarding them on an outbound link. Queue delay varies with the length of queues in buffers. Transmission delay can be viewed as a constant if we define the packet size and link bandwidth. Therefore, the only variable factor is queuing delay. We hope to reduce the queuing delay by optimization of networks to a great degree.

5.2 Simulations

In our simulations, we always run the simulator for 1000s. We will present our work in several steps. There are always three classes—EF, AF1 and AF2 class. In our configuration, we always assume EF class as real-time traffic. Both AF1 and AF2 class represent simple traffic: FTP or email, but they have different priority.

5.2.1 Step 1: A Simple Topology

In this step, we will simulate with the same traffic through different routing algorithms. The first is the shortest path routing algorithm; the second is the WB routing algorithm and the third is PERD. Then we draw performance (throughput and delay) graphs based on the recorded data. We will analyze those different performance results by comparing the figures.

5.2.1.1 Network Topology and Configuration of Network

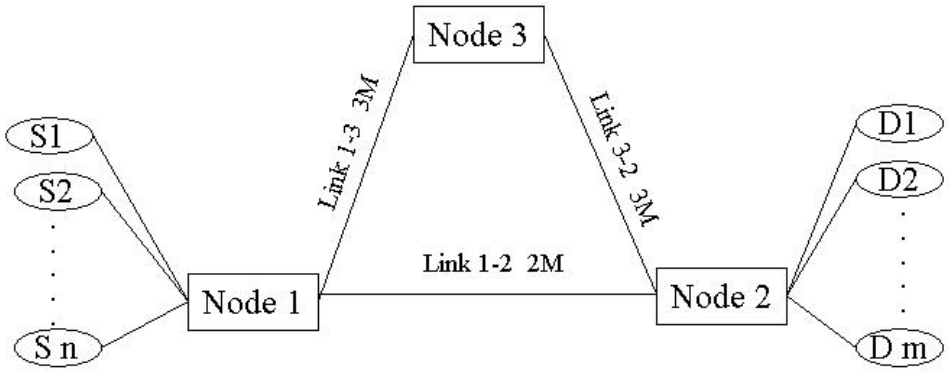

Figure 18 Simple topology of a DiffServ network

In Figure 18, S stands for traffic source and D is the acronym of destination. Numbers n and m are the indices of traffic sources and traffic destinations. The capacity of link1-2 is 2Mbps and the capacity of other links is 3Mbps.

Node 1 is the ingress router and node 2 is the egress router in our provisioned DiffServ network. Node 3 is the only core node.

We list the configuration of this network in Table 4. Bandwidth unit is Bps.

Link name	Bandwidth		
	EF	AF1	AF2
Link 1-2	0.4M	0.6M	1M
Link 1-3	0.6M	0.9M	1.5M
Link 3-2	0.6M	0.9M	1.5M

Table 4 Configuration of simple DiffServ Network

Link 1-2 has the capacity of 2M. We split the capacity of this link into three parts. The first part is 0.4M for EF class. The second part is 0.6M for AF1 class and the third part 1M for AF2 class traffic. This separation complies with our policy, EF class amounts to 10-20% of the capacity of link, AF1 class 30~40% and AF2 class 50%.
In contrast, both link 1-3 and link 3-2 have the capacity of 3M. Similarly, we divide their capacity in accordance with the ratios we mentioned earlier. We can see bandwidth allocation in Table 4.

Of course, the size of buffer space in routers is finite. For EF queue is 1000 byte long and for AF class queue length is 5000 bytes. If the queue length exceeds this limit, the coming packet will be lost. Moreover, if a buffer overflows, packets are discarded regardless of which flows they belong to.

5.2.1.2 Traffic parameters

The traffic flows are initiated every 40s in order as we listed in Table 5. In Figure 18, we can see that there are two paths from the source node 1 to the destination node 2. The total capacity of those paths is 5M and it consists of EF 1M, AF1 1.5M and AF2 2.5M.

No.	Class	Rate	Starting time	Total
Flow1	EF	0.1M	0s	5M
Flow2	AF1	0.2M	40s	EF/1M
Flow3	AF2	0.2M	80s	AF1/1.5M
Flow4	EF	0.1M	120s	AF2/2.5M
Flow5	AF1	0.2M	160s	
Flow6	AF2	0.2M	200s	
Flow7	EF	0.1M	240s	
Flow8	AF1	0.2M	280s	
Flow9	AF2	0.2M	320s	
Flow10	EF	0.1M	360s	
Flow11	AF2	0.2M	400s	
Flow12	AF2	0.2M	440s	
Flow13	AF1	0.2M	480s	
Flow14	EF	0.1M	520s	
Flow15	EF	0.1M	560s	
Flow16	AF1	0.2M	600s	
Flow17	EF	0.1M	640s	
Flow18	EF	0.1M	680s	
Flow19	AF1	0.2M	720s	
Flow20	EF	0.1M	760s	
Flow21	EF	0.1M	800s	
Flow22	AF1	0.2M	840s	
Flow23	AF1	0.1M	880s	
Flow24	AF2	0.8M	920s	
Flow25	AF2	0.7M	960s	

Table 5 Traffic parameters

5.2.1.3 Simulation with the Shortest Path Routing

In this step, we take the shortest path routing algorithm. This algorithm is the basic one. We explained it in Section 2.2 “QoS Routing”.

We can see in Figure 18 that all traffic will flow through link 1-2. The capacity of link1-2 is 2M. When simulation begins, traffic flows come in order mentioned in Table 5. Because of the limit of link bandwidth, only 12 traffic flows, flow1 to flow12, can be established. All other traffic will be rejected since there is no available bandwidth any more along this road.

The performance results--throughput and delay--are depicted in Figure 19 and Figure 20. The x-axis in Figure 19 shows time and y-axis shows the throughput of each class

and total throughput. Similarly, the x-axis in Figure 20 shows time and y-axis shows the average delay of each class.

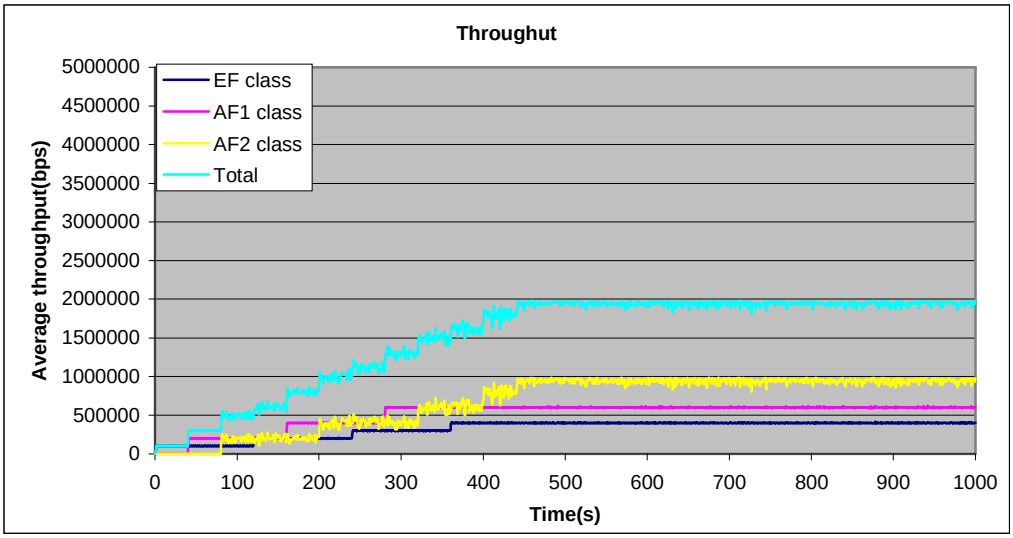

Figure 19 Throughput of three classes--EF AF1 and AF2 at node 2

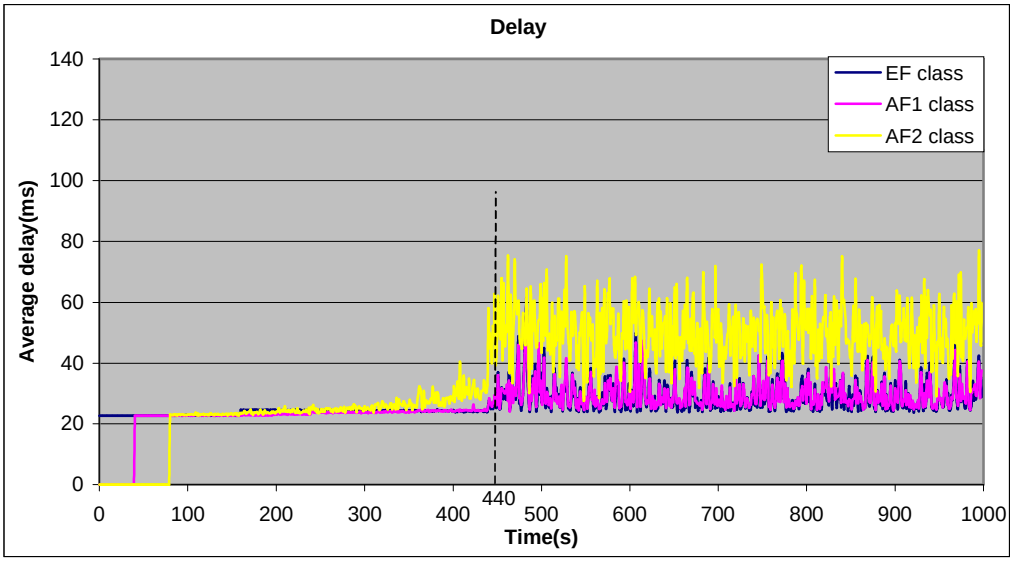

Figure 20 Average delays of three classes

Although the capacity of the path from node1through node3 to node2 is larger than that of path node1 to node2, the traffic flows through the second route instead of the first one. This can be shown by the fact that the total throughput of three classes is 1.95Mbps, approximately equal to the capacity of link 1-2. The network efficiency is

1.95/5 = 39% and the efficiency is too low. This reflects one of the flaws of the shortest path algorithm. Some links may be heavily loaded; on the other hand, other links may be lightly loaded or even stay idle.

In Figure 20, delay of each class increases with the increase of traffic. When all of the bandwidth is used up, for example, at about 440s as we indicate this point with the dashed line, delay increases sharply. Under this situation, congestion occurs. A lot of packets are lost. Some lost packets need to be retransmitted. In turn, this worsens the situation. The overall QoS is degraded.

Although the bandwidth of link1-2 is exhausted and congestion occurs, this case is far away from efficient utilization. The alternative path node1->node3->node2 stays idle all along. Therefore, we hope to distribute some traffic to this route. By doing so, more traffic flows can be accepted and congestion may be removed, at the same time, the required QoS can be guaranteed in the long term.

We can summarize conclusions about this case with the shortest path algorithm.

- All traffic competes on the shortest path. However, other feasible routes even stay idle. Although this algorithm is widely used it is not efficient from the viewpoint of traffic engineering. Moreover, it is not optimal from the point of view of routing optimization.

- This method may cause more requests to be rejected because there are not enough resources on the shortest path.

- On the shortest path, even the established connections may suffer because of congestion.

In order to improve the efficiency of networks, we hope that traffic can be evenly distributed within networks. We simulate with the WB routing algorithm to achieve this goal.

5.2.1.4 Simulation with the Widest Bandwidth Algorithm

In this step, we try the WB routing algorithm. This algorithm always finds the link with the maximum residual bandwidth. This way, traffic load can be distributed within networks and congestion may be avoided. If a route matching the requirement can be found, this request can be connected, if not, then it will be rejected. Comparing with the shortest path algorithm, the utilization efficiency can be improved.

This time, we will again use the traffic listed in Table 5 to simulate. However, we use the WB algorithm rather than the shortest path algorithm. The procedure is similar to the one in the earlier section. The performance results are shown in Figure 21, Figure 22 and Figure 23. Figure 21 shows the throughput. Figure 22 and Figure 23 are graphs about delay. Figure 22 depicts the delay of every class on different paths. Figure 23 is the average delay of each class service. In Figure 23, we calculate average delay like this: average delay = total delay of all packets in a specific class/number of packets in this class.

This time, traffic traverses not only the path node1->node2, but also the path node1->node3 ->node2, because one of those two routes has wider bandwidth than the other one alternately. We can testify this by comparing the total throughput in Figure 21 with the total throughput in Figure 19. We can find that the total throughput increases. The total throughput rises step by step as the time goes on. After all of the 25 traffic flows are connected, the average total throughput is 4.672M. Thereby the efficiency of the network is: 4.672/5 = 93.4%. The network efficiency is improved compared with the shortest path routing algorithm in the preceding section.

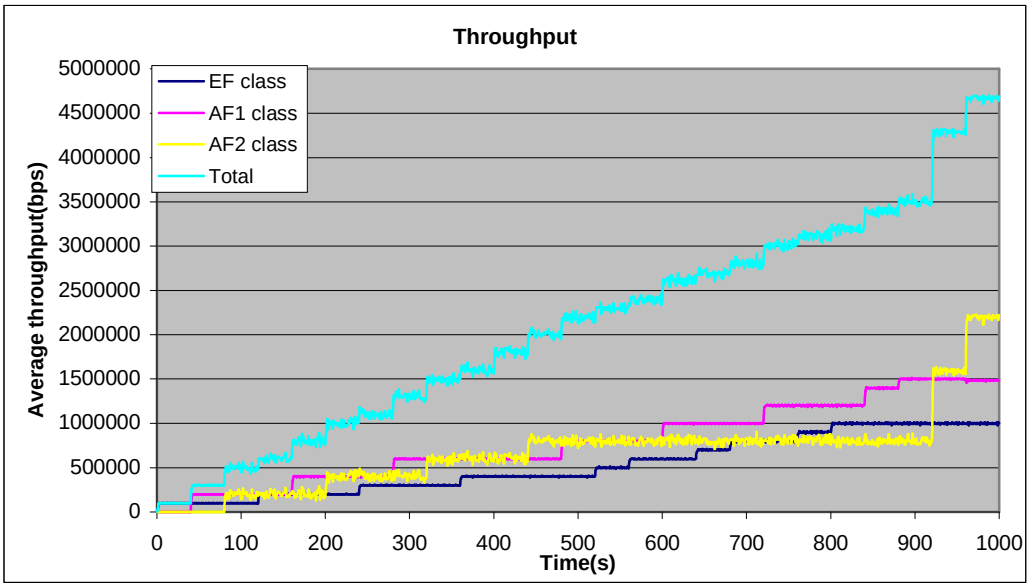

Figure 21 Throughput of three classes at node 2

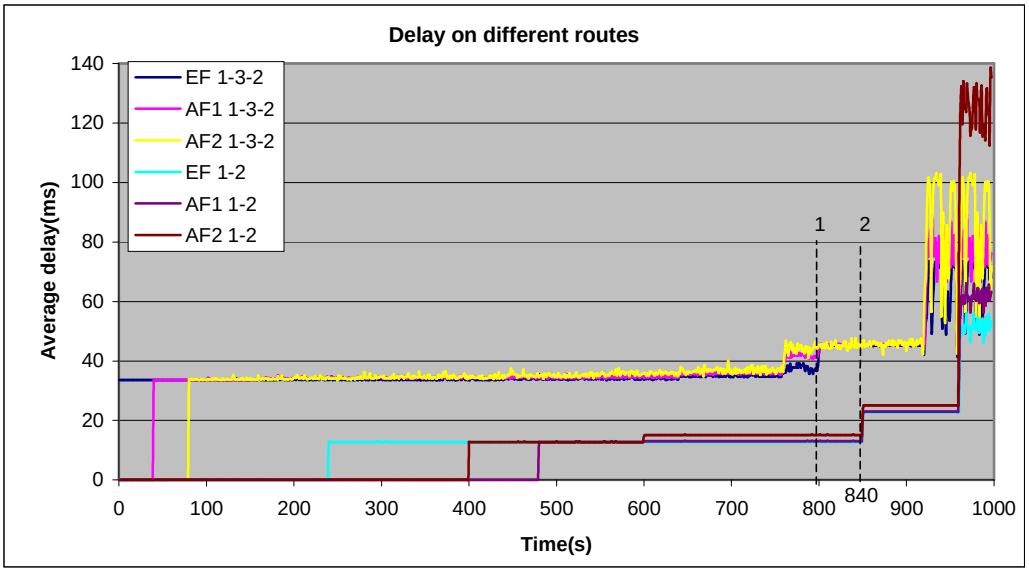

Figure 22 average delays of different classes via different routes

In Figure 22, we distinguish delay on route node1->node2 from that on route node1->node3->node2. Because delay is closely related to the number of hops, the average delay on route node1->node2 is different from the average delay on route node1->node3->node2. No matter which way, the average delay is in this order: EF is the minimum, AF1 is medium and AF2 is the maximum.

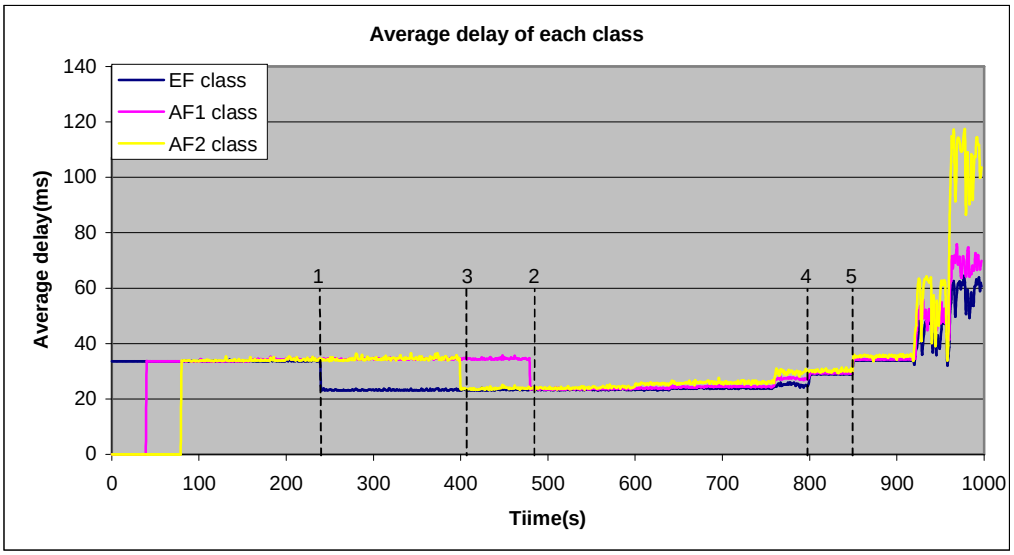

Figure 23 Average end-to-end delay of each class

In Figure 23, delay is shown on the basis of each class instead of different routes. We can see that EF average delay plummets at about 240s as indicated by dashed line1. Before 240s, EF class traffic only flows through route node1->node3->node2. After 240s, some parts of EF class traffic flows through route node1->node2. The average delay is reduced. This kind of phenomenon can be also found in AF1 class and AF2 class. We show them with dashed lines 2, and 3, respectively.

Referring to Table 5, the overall volume of EF class traffic is 1.0 M. By checking our logging files, we find 0.9M EF class traffic passes through route node1->node3->node2 because this path has wider bandwidth than that of route node1->node2 when those EF flows start.

In Figure 22, at around 800s (we show this point with dashed line 1), EF class delay on route node1->node3->node2 increases abruptly. This happened after EF class flow21 was accepted. At that moment, there is still residual bandwidth on this path. Why did this phenomenon happen? This took place because excessive real-time traffic concentrates on the path node1->node3->node2. At this moment, there is 0.9M EF traffic on this route. In addition, we find that delay on path node1->node2 also increases at about 840s as shown with dashed line 2,

In DiffServ networks, EF class has higher priority than other classes. EF class can pre-empt resources because EF class traffic is more important than other classes. In our paper, EF class traffic including real-time traffic, QOSPF messages and controlling signals. If excessive real-time traffic congregates on some links, real-time traffic contends for resources by itself, and impedes other classes consequently.

Therefore, the WB routing algorithm cannot solve congestion within networks if we only distribute oncoming traffic on feasible paths. Moreover, we must prevent real-time traffic from concentrating on some links. The WB algorithm does not consider per-class available bandwidth, and so it routes traffic based on the total residual bandwidth. If there is enough residual bandwidth the request will be granted no matter which class it belongs to. We need to cope with this problem with other methods.

So in the next section, we hope to establish an admission control mechanism that takes care of the per-class available bandwidth, of course, with the new method, the network performance should at least stay the same as in case of the WB routing algorithm.

5.2.1.5 Simulation with PERD (per-class)

In this step, we hope that the real-time traffic can be distributed on different routes. We will adopt PERD (per-class) routing. This algorithm not only considers the total residual bandwidth of a link but also takes the available bandwidth per-class into account. The admission control and routing are made on the basis of the available bandwidth per class.

Again, we run the QRS simulator for 1000s with the traffic listed in Table 5. The performance in terms of the throughput and delay is shown in Figures 24, 25 and 26.

We found that Figure 24 is very similar to Figure 21 of the WB algorithm. However, the total throughput has increased somewhat. By calculating, the total throughput is 4.913M. The efficiency of this network is: 4.913/5 = 98.3%. The total throughput increases from 4.672M to 4.913M by comparison with the WB algorithm.

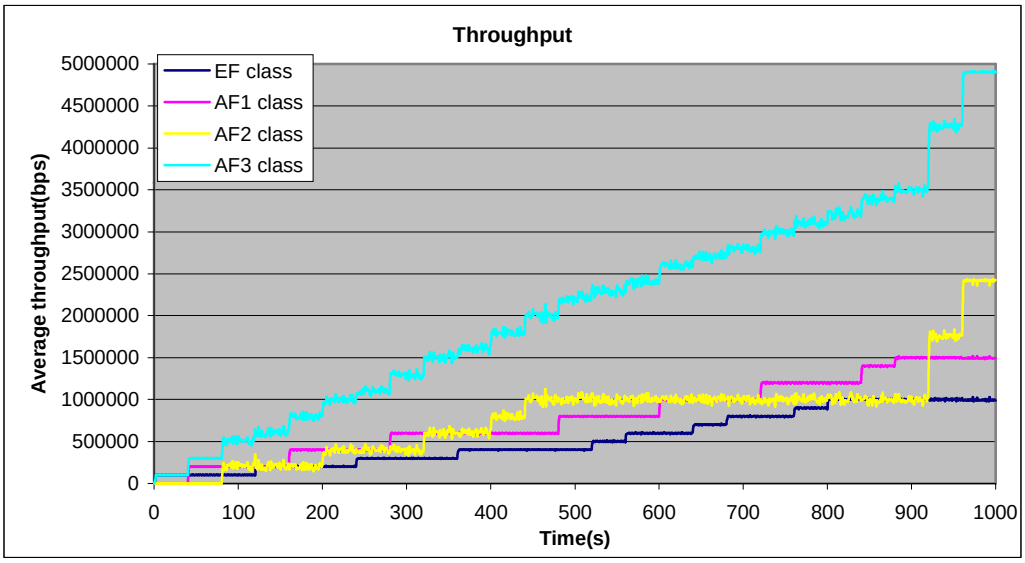

Figure 24 Throughput of three classes at node 2

For real-time traffic, we are also concerned about delay. We compare Figure 25 with Figure 22 and find that there is an obvious difference between those two pictures. In Figure 25, at about 800s denoted by dashed line 1, dramatic increase of EF class delay on route node1->node3->node2 disappears. Delay on route node1->node3->node2 is increasing gradually until the capacity is exhausted. By checking our logging files, we find 0.6M EF class traffic flows through path node1->node3->node2 and the remainder passes route node1->node2. Similar phenomenon also happens on path node1->node2. With PERD, we can distribute real-time traffic and other class traffic by considering per-class available bandwidth on each link. When bandwidth is used up, delay increases noticeably in both figures. This happens at about 940s.

Similarly, we present the average delay on the basis of each class in Figure 26. From 400s to 920s, delay of each class increases smoothly. On the other hand, the average delay in Figure 23 increases at the points we show with dashed line 4 and dashed line 5 from 400s to 920s. After resources are used up, delay rises significantly.

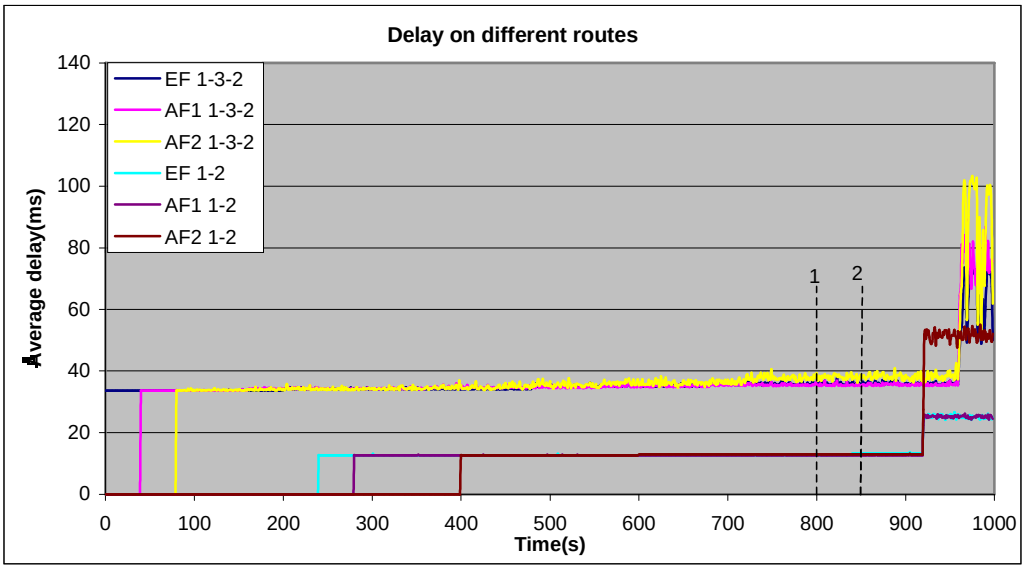

Figure 25 Average delays of different classes via different routes

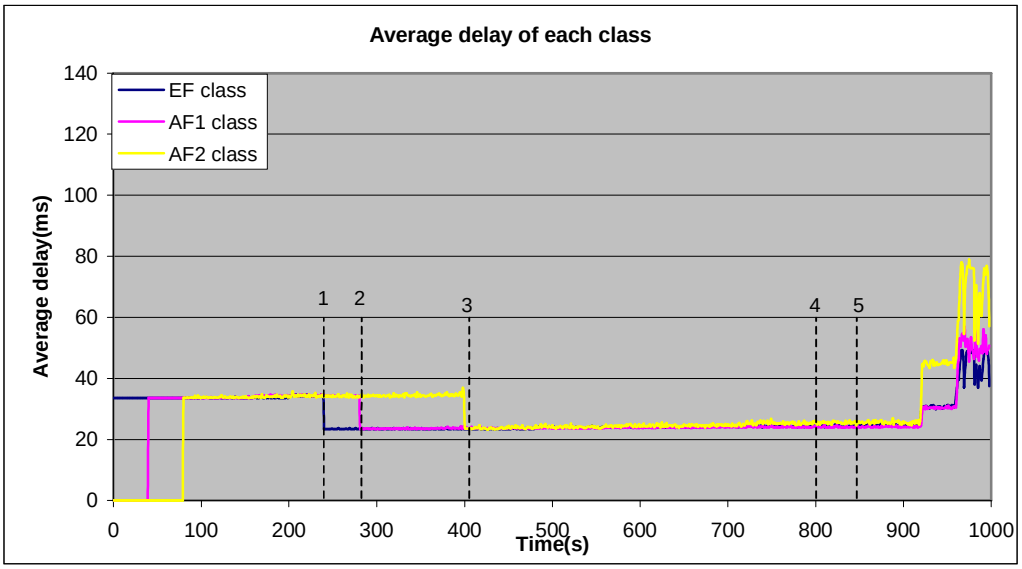

Figure 26 Average end-to-end delay of each class

With the network topology in Figure 18, we can find that the throughput can be increased and delay can be reduced with PERD. Therefore, we can make a conclusion that PERD has an advantage over the WB algorithm.

In addition, we find that whenever a flow comes it always finds the widest available class bandwidth corresponding to its class. Flows can take different routes even

though they belong to the same class and have the same source and destination. Each class calculates its optimal routes independently without any impact of other classes. In our example, EF class traffic flows find routes only considering its available bandwidth. We can observe this process by checking the class routing table at node1.

Destination	Nexthop	EFbandwidth	Destination	Nexthop	AF1bandwidth
Node1	Null	0M	Node1	Null	0M
Node2	Lk1-3	0.6M	Node2	Lk1-3	0.9M
Node3	Lk1-3	0.6M	Node2	Lk1-3	0.9M

Table 6 At the simulation starting

Destination	Nexthop	EFbandwidth	Destination	Nexthop	AF1bandwidth
Node1	Null	0M	Node1	Null	0M
Node2	Lk1-2	0.4M	Node2	Lk1-3	0.7M
Node3	Lk1-2	0.4M	Node2	Lk1-3	0.7M

Table 7 After EF class traffic flow4 is connected

5.2.2 Step 2: A Matrix Topology

In step1, we compared PERD with other routing algorithms such as the shortest path and the WB. We did our simulations with a relatively simple network. In this step, we will use the matrix topology in Figure 27 in order to test our proposed PERD further.

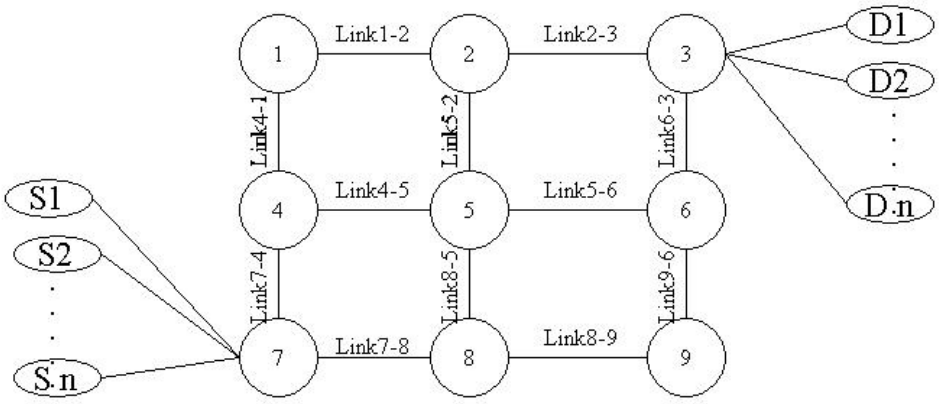

Figure 27 3*3 matrix topology

In Figure 27, node 7 is the source and node 3 is destination. There are six different paths from source node 7 to destination node 3. We enumerate them as follows:

- Node7->node4->node1->node2->node3;
- Node7->node4->node5->node2->node3;
- Node7->node4->node5->node6->node3;
- Node7->node8->node5->node6->node3;
- Node7->node8->node9->node6->node3;
- Node7->node8->node5->node2->node3;

In this topology, the capacity of each link along route node7➔node4➔node1->node2➔node3 is 3M. The capacity of other links is 2M. We provision this network with the same principle as the previous section.

The capacity of each link is shown in Table 8.

Link name	Capacity bandwidth		
	EF	AF1	AF2
Link 1-2	0.6M	0.9M	1.5M
Link 2-3	0.6M	0.9M	1.5M
Link 4-1	0.6M	0.9M	1.5M
Link4-5	0.4M	0.6M	1M
Link5-2	0.4M	0.6M	1M
Link5-6	0.4M	0.6M	1M
Link6-3	0.4M	0.6M	1M
Link7-4	0.6M	0.9M	1.5M
Link7-8	0.4M	0.6M	1M
Link8-5	0.4M	0.6M	1M
Link8-9	0.4M	0.6M	1M
Link9-6	0.4M	0.6M	1M

Table 8 Configuration of matrix topology

We describe the traffic in Table 9. Flow 1 is main EF traffic flow. In addition, other 0.1M traffic flows are connected one after another every 70 seconds.

No.	Class	Rate	Starting time	Total
Flow1	EF	0.4M	0s	EF 1.1M AF1 0.6M AF2 1.2M
Flow2	EF	0.1M	70s	
Flow3	EF	0.1M	140s	
Flow4	EF	0.1M	210s	
Flow5	EF	0.1M	280s	
Flow6	EF	0.1M	350s	
Flow7	EF	0.1M	420s	
Flow8	EF	0.1M	490s	
Flow9	AF1	0.6M	100s/ON150s/OFF200s	
Flow10	AF2	1.2M	50s/ON150s/OFF200s	

Table 9 Traffic parameters

We can see the performance results in Figure 28 and Figure 29. We obtain Figure 28 by calculating the throughput of each individual class at destination node 3. Figure 29 shows average delay of each individual class.

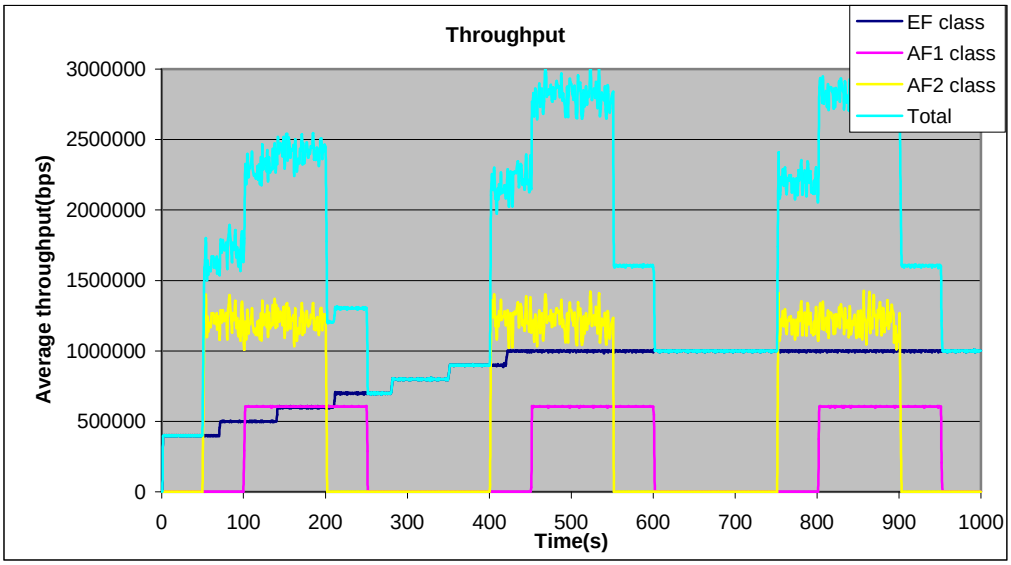

Figure 28 Throughput of three classes at node3

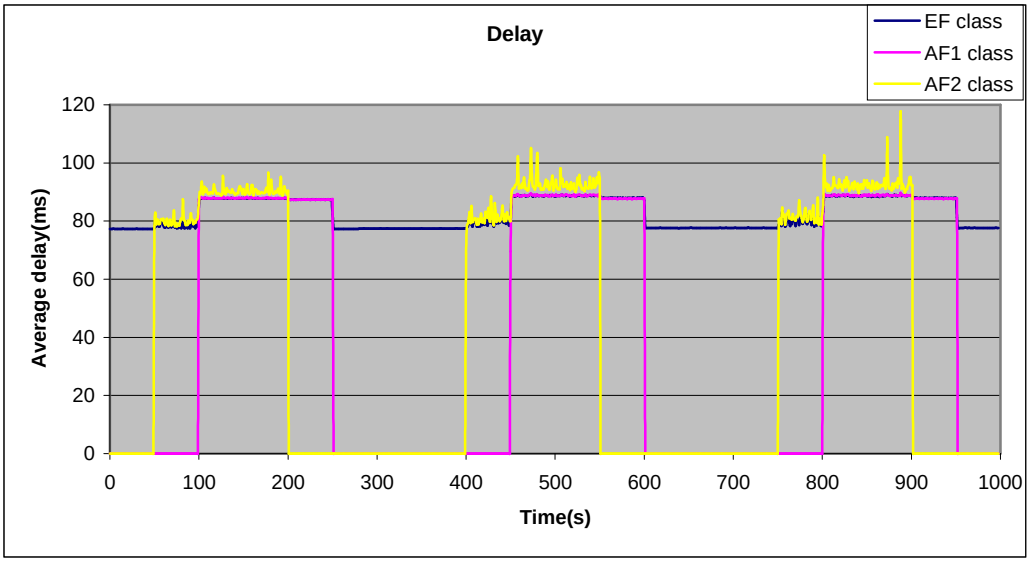

Figure 29 Average delays of different classes via different routes

The total capacity for EF class in Figure 27 is 1.0M although we provide 1.1M EF class traffic. Only flows from 1 to 7 can be connected successfully. This can be demonstrated by the total EF class throughput in Figure 28. Its maximum value is 1.0M. Flow8 is rejected although there is enough residual bandwidth on each route. This case reflects our PERD principle: whether a coming flow is accepted or rejected is determined by its corresponding class available bandwidth instead of residual

bandwidth. PERD provides a way to avoid too much real-time traffic congregating on some links. To some extent, congestion can be prevented and QoS can be guaranteed.

In Figure 29, delay fluctuates with connection and release of flow9 and flow10.

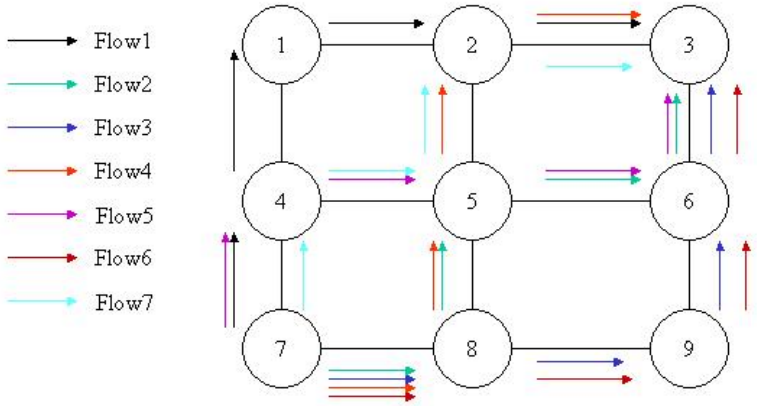

Figure 30 Different paths

Apart from Figures 28 and 29, we draw Figure 30 to demonstrate the distribution of traffic. During the period from 250s to 400s, both flow 9 and flow10 are in the state of release. Route node7->node4->node1->node2-node3 has the maximum residual bandwidth. Flow 5 comes at 280s and it takes the route node7->node4->node5->node6->node3 instead of the path of node7->node4->node1->node2-node3. Similarly, flow 6 takes the path of node7->node8->node9->node6->node3.

5.2.3 Step 3: NSFNET Topology

Based on simulations in prior sections, we can see that our method works very well in simple networks. In this step, we compare the performance (throughput, delay and cost) of different algorithms (the shortest path, the WB and PERD) when traffic load increases from 10% to 100%.

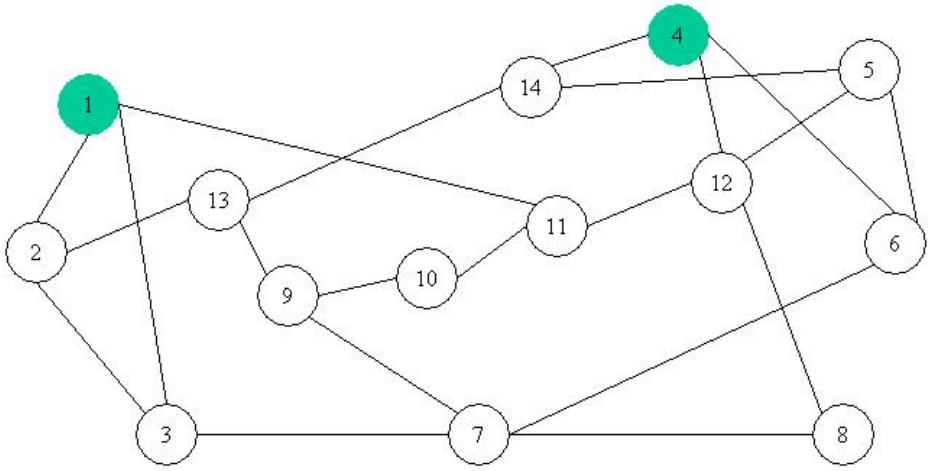

Figure 31 NSFNET-T1 backbone topology

We take a bit more complicated network topology—NSFNET-T1. We depict this network topology in Figure 31 with 14 nodes. We assume that both node 1 and node 4 are boundary routers. We connect traffic sources to node 1 and traffic sinks to node 4.

We allocate link bandwidth in a similar way as in as step 2. This time, we assign the capacity of each link along path node1->node11->node12->node4 as 3M, which is the shortest path. The capacity of other links is 2M.

For NSFNET-T1, we see that the minimal cut has three links (link4-14, link4-12 and link4-6) with the total capacity of 7M (2M+3M+2M). Obviously, the total network throughput should be at most 7M. Considering provisioning policy, EF class allocation bandwidth, AF1 class allocation bandwidth and AF2 class allocation bandwidth are 1.4M, 2,1M and 3.5M respectively.

In our simulations, we increase traffic load by 10% each time, namely, we provide 0.7M traffic in the initial step and 1.4M traffic in the second step, and so on. This process lasts until 100% traffic load. We show the traffic load in Table 10.

Class	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
EF	0.1M	0.3M	0.4M	0.6M	0.7M	0.9M	1.0M	1.2M	1.3M	1.4M
AF1	0.2M	0.4M	0.6M	0.8M	1.0M	1.2M	1.4M	1.6M	1.8M	2.1M
AF2	0.4M	0.7M	1.1M	1.4M	1.8M	2.1M	2.5M	2.8M	3.2M	3.5M
Total	7M/EF 1.4M/AF1 2.1M/AF2 3.5M									

Table 10 Traffic load

5.2.3.1 Throughput and Delay vs. Load Percentage

Figures from Figure 32 to 35 show average throughput. Figure 32 shows the trend of total average throughput under three different routing algorithms. Individual class throughput under different routing algorithms is shown in Figures 33 to 35.

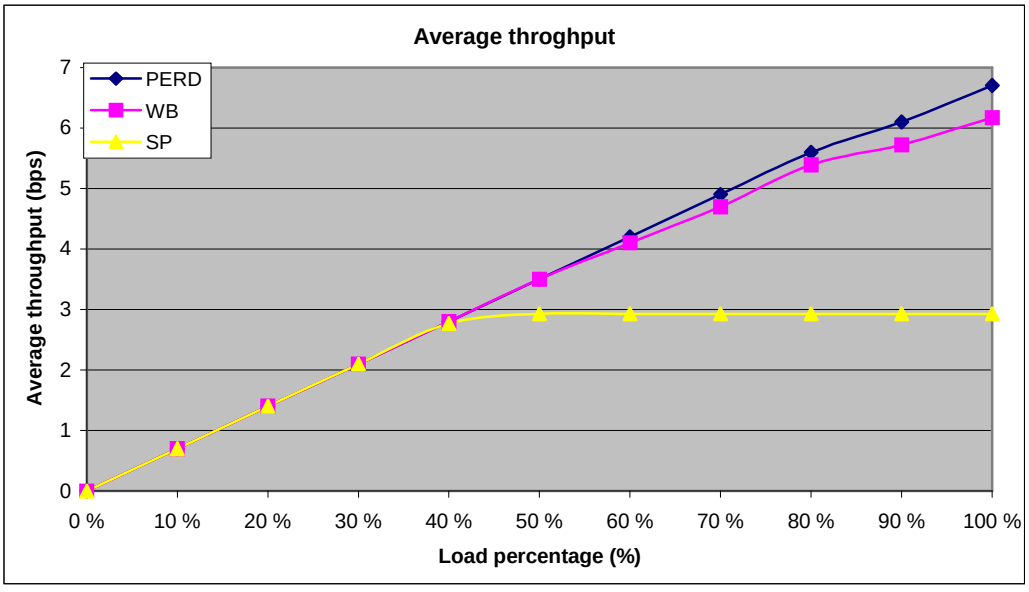

Figure 32 Throughput under different routing algorithms

In Figure 32, the total throughput of the shortest path algorithm is limited by the capacity of the shortest path. We mainly focus on the WB and PERD algorithms. When the traffic load is relatively light, for example, less than 60% of the total

network capacity, there is no obvious difference between PERD and the WB. The advantage of PERD shows up along the increase of traffic load. The total throughput of PERD is 6.73M. The efficiency is 6.8/7 = 97.1%. In contrast, the total throughput of the WB is 6.17M and the efficiency is 6.17/7 = 88.1%.

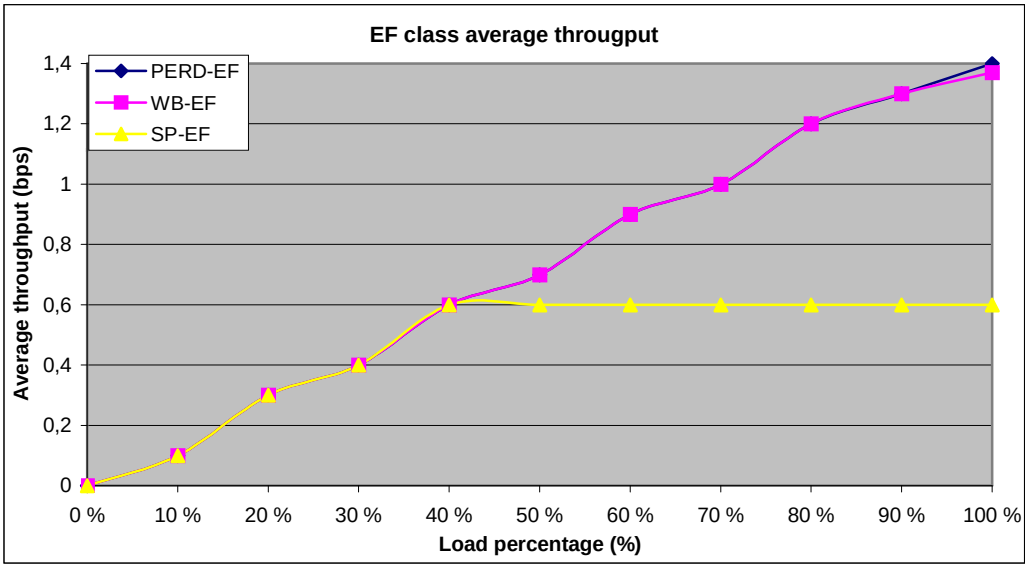

Figure 33 EF throughput under different routing algorithms

In Figure 33, EF class throughput is presented. With PERD, EF class throughput is 1.4M. Under the WB algorithm, EF class throughput is 1.37M.

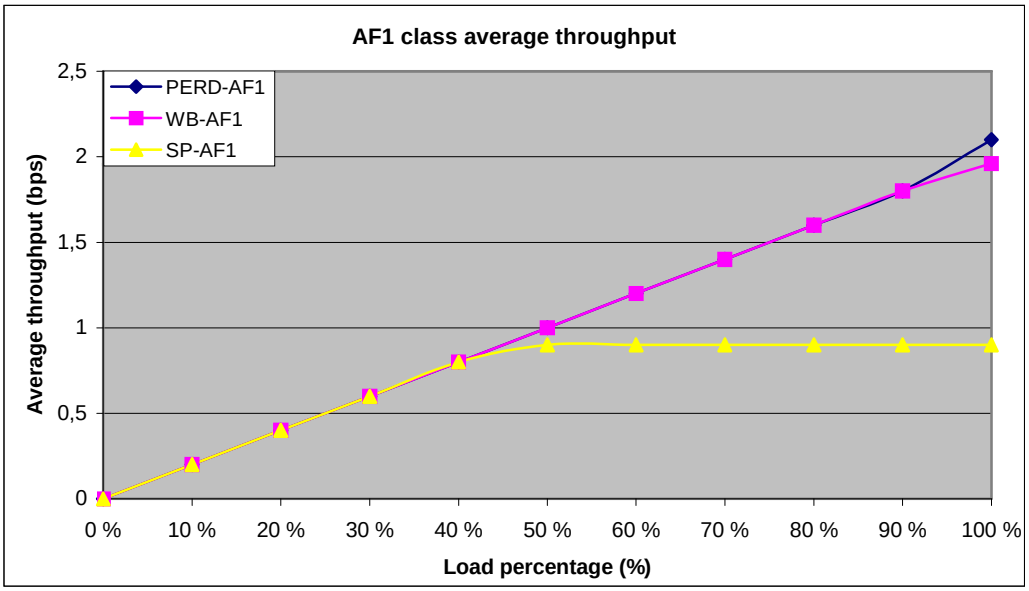

Figure 34 AF1 throughput under different algorithms

AF1 class allocation bandwidth is 2.1M in Figure 31. Figure 34 shows AF1 class throughput under those three different routing algorithms. With PERD, AF1 class throughput is 2.1M. By comparing, AF1 class throughput with the WB is 1.96M.

AF2 class allocation bandwidth is 3.5M in Figure 31. Figure 35 shows AF2 class throughput under three different routing algorithms. With PERD, AF1 class throughput is 3.3M. In contrast, AF1 class throughput with the WB is 2.81M.

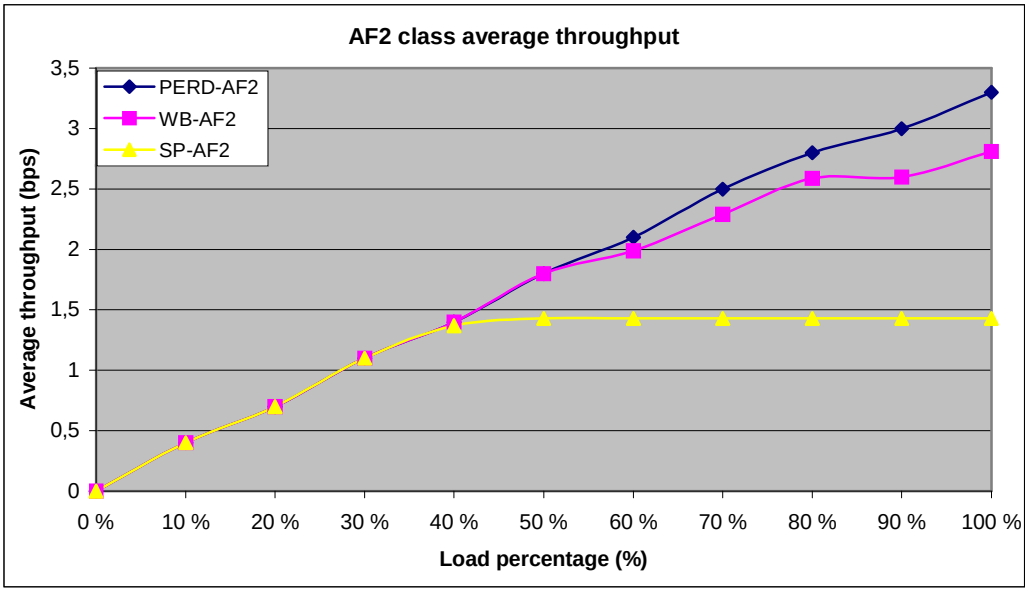

Figure 35 AF2 throughput under different algorithms

By observing Figures 32 to 35, we find that PERD can improve the total throughput and per-class throughput as well. The most significant difference is AF2 class throughput between PERD and the WB routing algorithm. The lower priority AF2 class can use more resources if real-time traffic is distributed within the network in accordance with certain rules.

Delay is also an important metric. For a complicated network, there may be several feasible routes from a source to a destination. The number of hops along routes plays an important role to determine the delay. The number of hops may be different on different routes. Therefore, delay might vary from time to time because of different routes.

In order to compare the delay that packets of different classes experience we resort to two methods.

The first is that we calculate the average delay of each class by the formula: sum of delays of all packets in a class/number of packets in the class. Figures 36 to 38 show average delay of each class under three different routing algorithms.

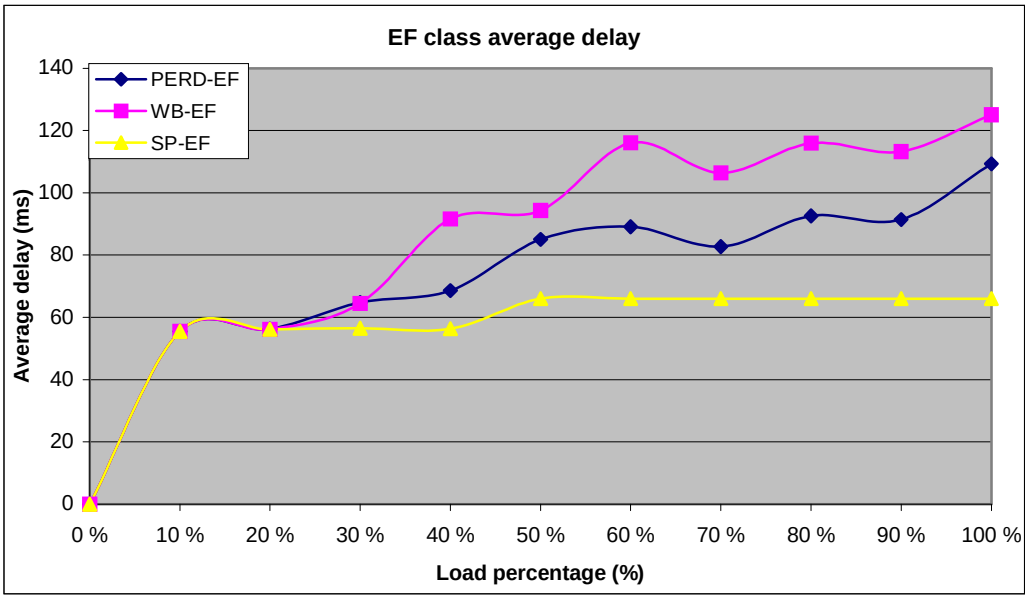

Figure 36 EF class average delay under different algorithms

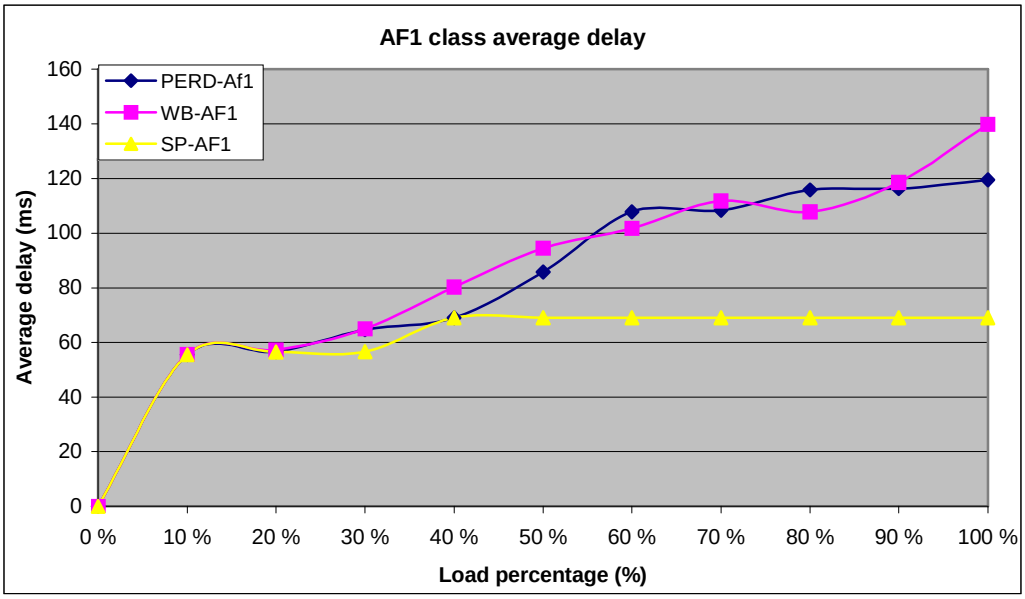

Figure 37 AF1 class average delay under different algorithms

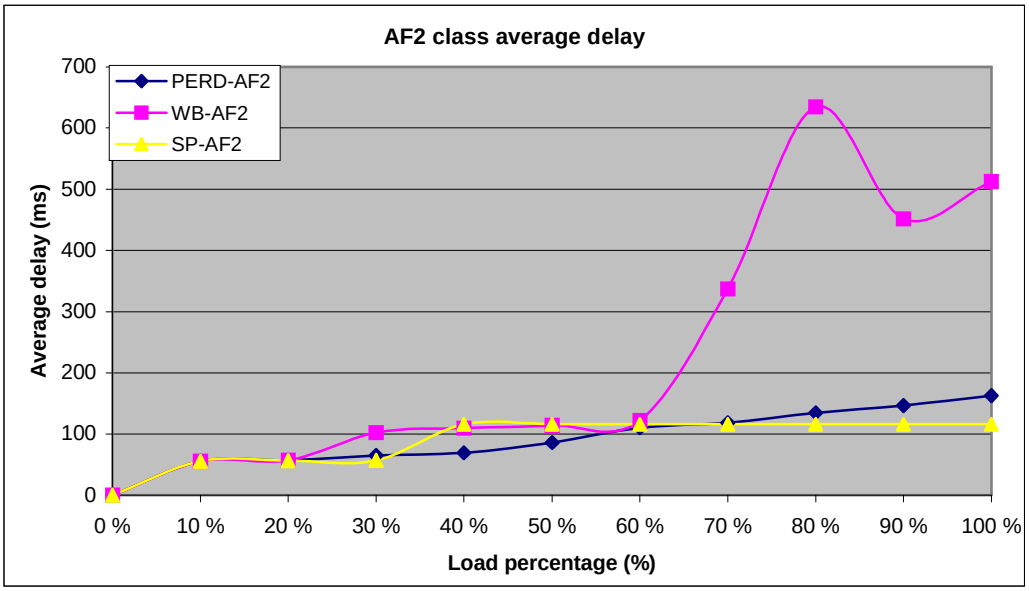

Figure 38 AF2 class average delay under different algorithms

Figure 36 shows the EF class average delay. Curves fluctuate when traffic load increases. Nevertheless, the trend is increasing as a whole. In Figure 31, there are several routes from source node 1 to sink node 4. Different routes have different numbers of hops. As we mentioned earlier, the number of hops can affect delay. No matter which class, PERD is better than the WB, especially, for AF2 class delay in Figure 38.

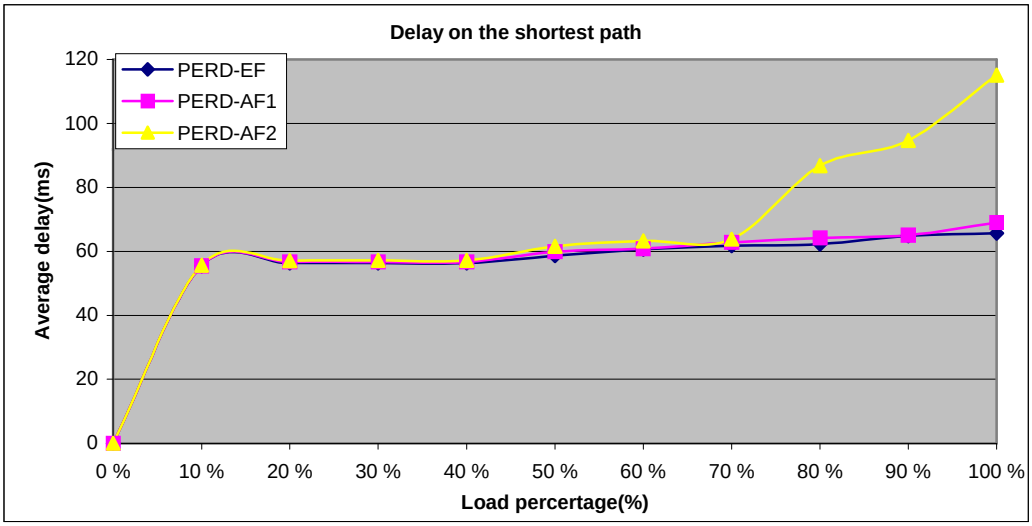

Figure 39 Average delay vs. load percentage with PERD on the shortest path

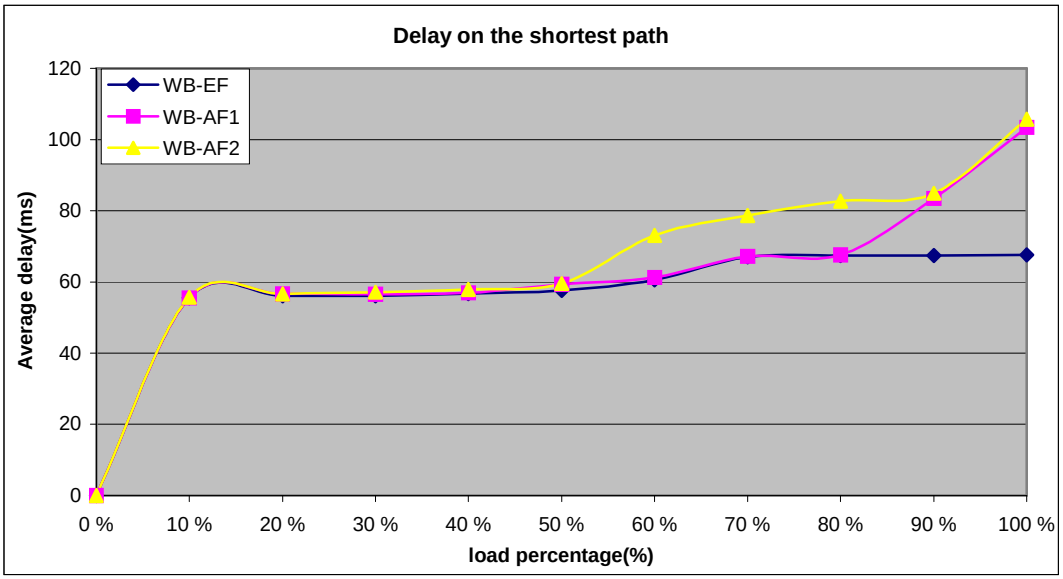

Figure 40 Average delay vs. load percentage with widest bandwidth on the shortest path

For further comparison, we choose the shortest path as the reference route. We calculate the average delay of each class via this same route under three different routing algorithms. Pictures from Figure 39 to Figure 41 show results with this method.

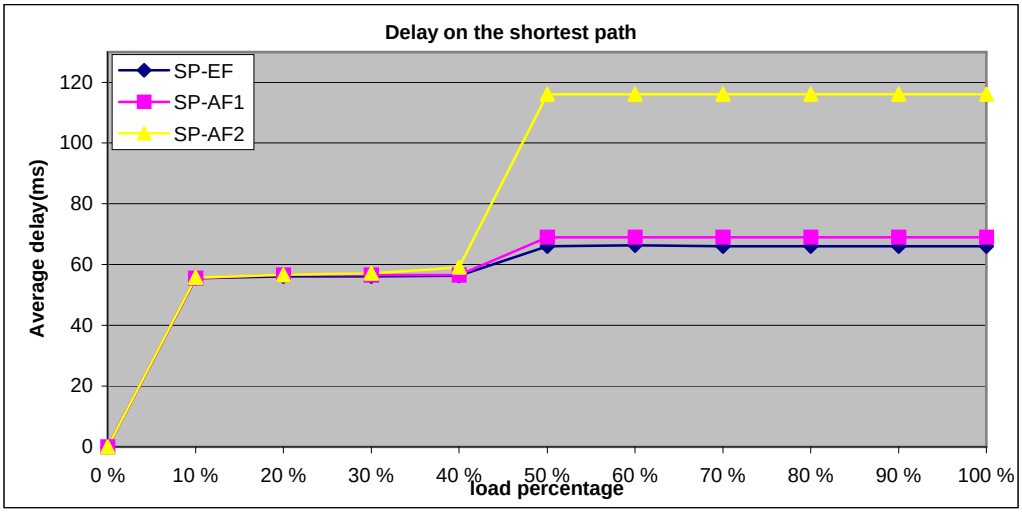

Figure 41 Average delay vs. load percentage with SP on the shortest path

From Figures 39 to 41, we can compare delay on the same route. EF class delay is the minimum, and AF2 class delay is the maximum in those three graphs.

EF class delay in Figure 39 is less than EF class delay in Figure 40. In Figure 40, AF1 class delay rises when the traffic load is 80% and increases with a steep slope.

In summary, no matter which method we adopt to measure delay, PERD has advantage over the other two routing algorithms. PERD can reduce delay by distributing real-time traffic based on the per-class available bandwidth.

5.2.3.2 Cost vs. Hold-timer Value

Now, we already know PERD can help to improve throughput and reduce delay by reasonably distributing traffic within networks. However, those benefits do not come for free. We will study the relationship between performance and cost.

We define cost as follows:

We use total processing time consumed by QOSPF during the simulation time to represent the cost of QoS routing. The processing time of each action of QOSPF in a node is set as shown in Table 11[59].

No	Cost(us)	Action description
1	1500	Find a next hop that can accept the required bandwidth
2	100	Check a message from RSVP and decide next step
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

Table 11 Cost of each QOSPF action

In this section, we use TB (threshold based) LSU algorithm. The basic idea of TB is that the scope of a node's update extends to its entire 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. In addition, TB attempts to trigger an update only when the current available bandwidth of a link differs noticeably from the previous advertised value.

We set the threshold value as 10%. We can express this relation as:

$$\text{If } |bw^0 - bw^c| / bw^0 > 10\%. \quad (1)$$

An update is triggered. Where bw^0 is the last advertised value of available bandwidth, and bw^c is the current bandwidth. bw^c can be residual bandwidth. When (1) holds, available bandwidth of every class is updated, and at the same time, the total residual bandwidth is updated, too. This is because our LSU packets include four parts: total residual bandwidth, EF class available bandwidth, AF1 class available bandwidth and AF2 class available bandwidth.

The hold-timer values vary from 100ms to 1000ms. The effect of the hold timer is that even if (1) holds while the timer is running an update will not be triggered. When the timer has expired and (1) holds the triggering takes place. We still use traffic listed in Table 5 and run the simulator for 1000s. In Figure 31, there are 14 nodes. We record the time consumed by QOSPF in every node during the simulation time, and then calculate the sum.

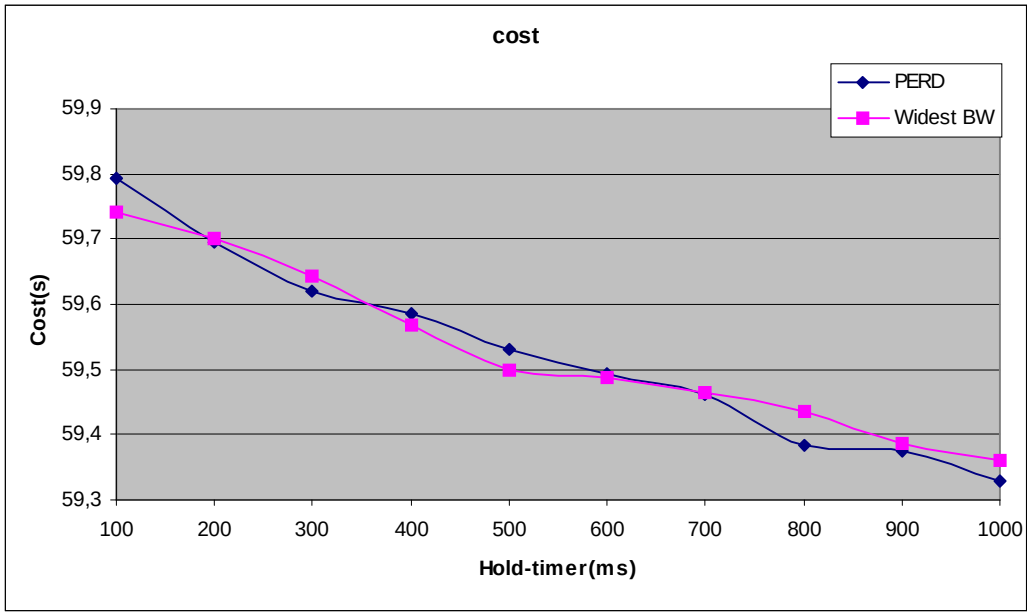

Figure 42 Cost under different hold-timers

We present our results in Figure 42. With the increase of the hold-timer value, the cost drops under both PERD and the WB routing algorithm. We cannot tell which curve is better in Figure 42 because those two curves cross together several times. Their relative positions change alternatively. This means that PERD does not increase cost.

We can make the conclusion that PERD can optimize network performance, but cost does not increase significantly compared with the WB algorithm.

6 Conclusion

In this thesis, we discussed a few problems in provisioned DiffServ networks. We focused on routing optimization. Based on some previous works, we proposed the PERD (per-class) principle. The approach takes the available bandwidth of each individual class (EF class and AF class) into consideration and selects routes based on available resources of each class. Each class has its own routing table independently of other classes. We can find the following conclusions by simulations with the QRS simulator.

(1) PERD can guarantee QoS of the EF class for both throughput and delay.

(2) Low priority classes can use more resources with PERD. AF2 class throughput is improved compared with WB.

(3) Delay increases with the increment of traffic load. It is desirable that real-time traffic cannot exceed a certain percentage of a link's capacity.

(4) By using PERD, each class has its own routing table. Traffic having the same originating node and destination node can have different paths. Traffic can be distributed on different routes based on the QoS requirement and resource state.

(5) PERD does not increase cost significantly with the upgrading performance compared to the widest bandwidth algorithm.

(6) In provisioned DiffServ network, we can use CBQ as a scheduling mechanism. It can guarantee each class' allocation bandwidth.

There are some limitations about PERD. For instance, if the available bandwidth of a specific class is used up or not enough for an incoming request the request of this class may be rejected. However, the residual bandwidth is enough to accept this request. Thereby, the block probability likely increases.

7 Future Work

We see substantial potential advantages in using PERD as a routing method in provisioned DiffServ networks. There is, however, still much research to do before real implementations are possible.

In our work, PERD is used to find the optimal route within networks. As we know, delay is associated with the number of hops along the chosen route. Therefore, it is reasonable that we take the number of hops into consideration. Hop-count can be chosen as a second metric for EF class traffic.

In order to keep the routing table stable and pin down the chosen routes, we hope to combine our current work with MPLS in our future work.

Another topic for future work is how the excess bandwidth can be distributed among classes. In our work, we assumed static allocation of link capacity. Sometimes, this can result in request rejection although there is still enough residual bandwidth. Thereby, dynamic allocation is ideal.

We found that the provisioning of networks plays an important role in optimizing networks. Nevertheless, how to appropriately provision networks still needs further investigation.

Appendix

In our simulator, we use Exponential Weighted Moving Average (EWMA) to compute the limit status for a class. EWMA form of Meter is easy to implement in hardware [1].

Initialization:
borrow_mark=0
borrowing_mark=0
aver=0
aver_1=0
last_packet_time=0
last_packet_time_1=0
for each class queue in one link
if the class $_i$ *is nonempty*
calculate the aver $_i$ *:*
diff=current_time-last_packet_time $_i$ *-transmitting_time*
aver $_i$ *= w*aver* $_i$ *+(1-w)*diff*
if aver $_i$ *<0*
borrow_mark $_i$ *=0*
for each class queue in one link
if borrow_mark $_j$ *==1 in class* $_j$
calculate aver_1 $_j$ value of this queue
diff=current_time-last_packet_time_1 $_j$ -transmitting_time
if aver_1 $_j$ *>0*
distribution of excess bandwidth
class_regulated_state $_i$ *=off*
borrowing_mark $_i$ *=1*
break
if class== max
class_regulated_state $_i$ *=on*
continue
else
class_regulated_state $_i$ *=off*
borrowing_mark $_i$ *=0*
if borrow_mark $_i$ *==0*
update aver $_i$
else
update aver_1 $_j$
if aver>2.0
borrow_mark $_i$ *=1*

```
if class_regulated_state_i =off
  serving packet
  if borrowing_mark_i =1
    last_packet_time_1_j =current_time
  else
    last_packet_time_i =current_time
```

REFERENCES

[1] Y. Bernet, S. Blake, D. Grossman, A. Smith, “An Informal Management Model for Diffserv Routers”, Internet draft <draft-ietf-diffserv-model-06.txt >”, February 2001.

[2] Yongxing Jia and Ming Chen, “A Novel Architecture of Providing Guaranteed Services for Differentiated Services Network”, EUROCON'2001, Trends in Communications, International Conference on. , Volume: 2, 2001 Page(s): 492 -495 vol.2.

[3] Ibrahim Khalil, Torsten Braun, “Edge Provisioning and Fairness in VPN-Diffserv Networks”, Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on, 2000 Page(s): 424 –431.

[4] Y.Bernet et. al, “An Architecture for Differentiated Services”, RFC2475, December 1998.

[5] Stocia and H. Zhang, “Providing Guaranteed Service Without per Flow Management”, ACM Computer Communication Review, vol 29, no.4, pp 81-94, October 1999.

[6] Y.Bernet et. al, “A Framework For Integrated Services Operation Over Diffserv Network ”, RFC2998, November 2000.

[7] G.Zhang, H.T.Mouftah, “End-to-End QoS Guarantees Over Diffserv Networks”, Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on, 2001 Page(s): 302 –309.

[8] Balmer, R., Baumgarter, F., Braun, T., Gunter, M., “A Concept for RSVP Over Diffserv”, Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on, 2000 Page(s): 412 –417.

[9] R.Braden, D.Clark, S.Shenker, “Integrated Services in the Internet Architecture: An Overview”, Internet RFC 1633, June 1994.

[10] S. Blake, D. Black, M. Carlson, et al, “An Architecture for Differentiated Service”, Internet RFC 2475, December 1998.

[11] Wang Qian; Wu Jing; Cheng Shiduan; Ma Jian, “A PHB reservation mechanism in DiffServ network**”,** Communication Technology Proceedings, 2000. WCC - ICCT 2000. International Conference on, Volume: 1, 2000, Page(s): 424 -427 vol.1.

[12] K. Nichols, Van Jacobson and L. Zhang. “A Two-bit Differentiated Service Architecture for the Internet”, RFC2638, July 1999.

[13] Kalevi Kilkki, “Differentiated Service for the Internet”.

[14] Bakiras, S.; Li, V.O.K. **“**Quality of service support in differentiated services packet networks**”**, Communications, 2001. ICC 2001. IEEE International Conference, Volume: 8, 2001, Page(s): 2370 -2374 vol.8.

[15] Dovrolis,C.andRamanathan,P., “A case for relative differentiated services and the proportional differentiation model” IEEE Network , Volume: 13 Issue: 5 , Sept.-Oct.1999. pp. 26 –34.

[16] J. Davin, A. Heybey, “A Simulation Study of Fair Queuing and Policy Enforcement”, Comput. Commun. Rev., vol.20, October 1990.

[17] S. Floyd and V. Jacobson, “Link-sharing and Resource Management Models for Packet Networks”, IEEE/ACM Trans. Networking, Vol. 3, August 1995.

[18] A. Iwata, N. Fujita, “A hierarchical multilayer QoS routing system with dynamic SLA management”, Selected Areas in Communications, IEEE Journal on, Volume: 18 Issue: 12, Dec. 2000 Page(s): 2603 –2616.

[19] D. Awduche, “MPLS and traffic engineering in IP networks ”, IEEE Commun. Mag., pp 42-47, Dec.1999.

[20] G.Swallow, “MPLS Advantages for traffic Engineering”, IEEE Commun. Mag., pp 54-57, Dec. 1999.

[21] G. Apostolopoulos et al., “Implementation and performance measurements of QoS Routing Extensions to OSPF”, in proc. Infocom’99, Apr. 1999, pp680-688.

[22] R.Colton, “The OSPF Opaque LSA Option”, IETF RFC, RFC 2370, July 1998.

[23] Dave. k, et al., “Traffic Engineering Extensions to OSPF”, Internet draft, April 2002.

[24] F.Faucheur, et al., “Extensions to OSPF for Support of Diff-Serv-aware MPLS Traffic Engineering”, IETF draft, Feb 2001.

[25] Shigang Chen; Nahrstedt, K., “An overview of quality of service routing for next-generation high-speed networks: problems and solutions”, IEEE Network, Volume: 12 Issue: 6, November/December 1998, Page(s): 64 –79.

[26] Xipeng Xiao; Ni, L.M., “Internet QoS: a big picture”, IEEE Network, Volume: 13 Issue: 2, March-April 1999, Page(s): 8 –18.

[27] V. Jacobson, et al., “An Expedited Forwarding PHB”, IETF Internet Draft, work in progress, DiffServ WG, February 1999.

[28] J. Heinanen, et al., “Assured Forwarding PHB Group”, IETF Internet Draft, work in progress, DiffServ WG, February 1999.

[29] Le Faucher, et.al, “Requirement for Diff-Serv-aware Traffic Enginnering”, Internet draft, Apr 2002.

[30] E. Rosen, “Multiprotocol Label Switching Architecture”, Internet draft, Jan 2001.

[31] Moh, M.; Wei, B.; Zhu, J.H., “Supporting differentiated services with per-class traffic engineering in MPLS”, Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on, 2001, Page(s): 354 –360.

[32] Bakiras, S.; Li, V.O.K., “Efficient resource management for end-to-end QoS guarantees in diffserv networks**”,** Communications, 2002. ICC 2002. IEEE International Conference on, Volume: 2, 2002, Page(s): 1220 –1224.

[33] j. Moy, “OSPF Version 2”, Internet Request for Comments, RFC 2178, July 1997.

[34] Apostolopoulos, G., Guerin, R., Kamat, S. **“**Implementation and Performance measurements of QoS routing extensions to OSPF**”**, INFOCOM '99., Eighteenth Annual joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Volume: 2, 1999 Page(s): 680 -688 vol.2.

[35] Daniel O. A., Angela c., et al, “Overview and Principles of Internet Traffic Engineering”, < draft-ietf-tewg-principles-02.txt >, Internet draft, November 2001.

[36] 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.

[37] Peng Zhang, Raimo Kantola, “QoS Routing for DiffServ Networks: Issues and Solutions”, Technical report, May 2001.

[38] L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource Reservation Protocol (RSVP)—Version 1 Functional Specification”, RFC 2205, September 1997.

[39] L. Ong, F.Reichmeyer, A. Terzis, etal, “A Two-tier Resource Management Model for Differentiated Services Networks”, Internet draft, Nov 1998.

[40] Benjamin Teitelbaum and et al, “Internet2 Qbone: Building a Testbed for Differentiated Service”, IEEE Network, September/October 1999.

[41] G. Aprostolopoulos, et al, “Qaulity of Service Based Routing: A Performance Perspective”.

[42] G. Apostolopoulos, et al, “Intra-Domain QoS Routing in IP Networks: A feasibility and Cost/Benefit Analysis”.

[43] E. Crawley, et al, “A Framework for QoS-based Routing in the Internet”, IETF RFC2386, 1998.

[44] D.D. Clark and W. Fang, “Explicit Allocation of Best Effort Packet Delivery Service”, Tech. Report, MIT Laboratory of Computer Science.

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

[46] QoS Forum, “Quality of Service-Glossary of Terms”, May 1999.
URL: http://www.qosforum.com/white-papers/qos-glossaryv4.pdf.

[47] Lily C.,Yang C., “A Framework for Internet Network Engineering”, <draft-cheng-network-engineering-framework-01.txt>, Internet draft, July 2001.

[48] Zhang, P. & Kantola, R., “ Designing A New Routing Simulator for DiffServ MPLS Networks", 2001 SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'2001), July 2001.
URL:http://www.tct.hut.fi/~pgzhang/papers.html.

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

[50] QoS Forum, “Quality of Service-Glossary of Terms”, May 1999. URL: http://www.qosforum.com/white-papers/qos-glossaryv4.pdf

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

[52] Guerin, G. et al., “QoS Routing Mechanisms and OSPF Extensions”, Internet Draft, draft-guerin-QoS-routing-ospf-03.txt, January 1998.

[53] F. Le Faucheur, et al, “MPLS Support of Differentiated Services”, work in progress, February 2001.

[54] F. Le Faucheur, et al, “Requirements for supporting of Diff-Serv-aware MPLS Traffic Engineering”, work in progress, may 2001.

[55] D.Awduche, J. Malcolm, et.al, “Requirements for Traffic Engineering Over MPLS”, Internet draft, September 1999.

[56] P. Ashwood-Smith, B. Jamoussi, et al, “Improving Topology Data Base Accuracy with LSP Feedback in CR-LDP ”, < draft-ietf-mpls-te-feed-04.txt >, Internet draft, May 2002.

[57] Raymond R, F. Liao, “Dynamic core provisioning for quantitative differentiated service”, IWQOS 2001, LNCS 2092, pp. 9-26, 2001.

[58] Ulrich F., Polly H., Bernhard P., “Towards Provisioning DiffServ Intra-nets”.

[59] Zhansong Ma, Peng Zhang, Raimo Kantola, “Influence of link State Updating on the Performance and Cost of QoS Routing in an Intranet”, 2001 IEEE Workshop on High Performance Switching and Routing (HPSR 2001). May, 2001.

[60] Mamais, G., Markaki, M., Politis, G., Venieris, I.**S, “**Efficient buffer management and scheduling in a combined IntServ and DiffServ architecture: a performance study”, ATM, 1999. ICATM '99. 1999 2nd International Conference, 1999, Page(s): 236 –242.

[61] Zhangsong Ma, “Performance and Cost Analysis of QoS Routing in an Intranet”, Master’s thesis, October 2000.

[62] Peng Zhang, Raimo Kantola, “QoS Routing for DiffServ Networks: Issues and Solutions”, Technical Report, May 2001.

[63] George Apostolopoulos , Roch Guérin , Sanjay Kamat , Satish K. Tripathi, “Quality of Service Based Routing: A performance Perspective”, ACM SIGCOMM Computer Communication Review,1998.