TECHNICAL REPORT

QoS Routing for DiffServ Networks:

Issues and solutions

(Draft)

Peng Zhang, Raimo Kantola, Samuli Aalto
Networking Laboratory
Helsinki University of Technology
Email: {pgzhang@tct.hut.fi, raimo.kantola@hut.fi, samuli.aalto@hut.fi**}**
October 2nd, 2002

Abstract: This document presents our preliminary study on QoS routing for DiffServ networks. We first present problems and issues in this area. Then, we propose our solutions and make preliminary analysis on the solutions.

Content

QoS Routing for DiffServ Networks: 1
Issues and solutions 1
1 Introduction 3
2 Background 3
2.1 Forwarding classes 4
2.2 Traffic engineering 4
2.3 QoS routing 5
2.4 Intra-domain QoS routing in DiffServ networks 6
2.4.1 QOSPF [18] 6
2.4.2 Traffic engineering extension to OSPF [20] 6
2.4.3 Extensions to OSPF for per-class routing [21] 6
3 Issues on QoS management in DiffServ networks 7
3.1 Issues 7
3.2 Problems 8
4 Proposed solution - PERD 9
4.1 Motivation 9
4.2 PERD mechanism 10
4.3 PERD principles 10
4.4 PERD routing procedures 11
5 Performance study 12
5.1 Simulation Environment 12
5.1.1 Simulation topology - NSFNET Topology 12
5.1.2 Traffic 13
5.1.3 Metrics 13
5.2 Simulations 14
5.2.1 Throughput vs. Load Percentage 14
5.2.2 Delay vs. Load Percentage 16
5.2.3 Cost vs. Hold-timer Value 18
6 Conclusions and future work 20

1 Introduction

With the rapid growth of the Internet, various new services have emerged and are becoming dominant in the Internet. These services mainly include VoIP, VPN (virtual private network) and video conference, which are normally characterized as real-time traffic. These services require Quality of Service in terms of bandwidth and delay. However, the traditional Internet was not designed for these services. Nowadays, it is becoming increasingly important and challenging for network operators to provide quality guaranteed services in the Internet.

In order to meet the demand, Differentiated Services (DiffServ) and Multi-Protocol Label Switching (MPLS) have been proposed [1-3]. The DiffServ provides a scalable QoS architecture that aims to achieve QoS assurance and scalability. It achieves QoS guarantee by differentiating traffic forwarding. And it achieves scalability by providing *coarse* differentiation on traffic, in other words, there are a few forwarding classes in a DiffServ network. The MPLS can provide fine differentiation on traffic flows. In principle, MPLS can be built on the basis of DiffServ architecture, i.e., DiffServ MPLS networks. In this paper, we mainly concentrate on DiffServ networks.

So far, the study on DiffServ mainly relies on traffic engineering (e.g., queuing, shaping, etc), which guarantees QoS only in short-term dimension. There still lacks study on mechanisms that can assure QoS in relatively long-term dimension. In this paper, we present problems in the area. One of the key problems is that by using traditional routing mechanisms (i.e., the shortest path routing) may easily lead to congestion that significantly degrades the performance of all traffic classes, especially low priority traffic class. In the paper, we employ QoS based routing (QoSR) as a tool to address the problem. In particular, we propose a novel routing mechanism, called per-class routing based on per-class dissemination (PERD). The key idea of the PERD is to differentiate routing for differentiated traffic classes. Through simulation study, we prove that the algorithm can improve QoS satisfaction of all traffic classes, especially low priority traffic classes.

The rest of the report is organized as follows. In Chapter 2, we introduce background knowledge and we present the unresolved issues of QoS management in DiffServ networks in Chapter 3. We propose our PERD mechanism in Chapter 4. We investigate the mechanism through simulation study and present the results in Chapter 5 and conclude our study in the final chapter.

2 Background

In this section, we briefly introduce background knowledge used in the paper.

2.1 Forwarding classes

A DiffServ network may contain only two forwarding classes, i.e., expedited forwarding (EF) [4] and assured forwarding (AF) [5]. The former can be used to convey delay sensitive traffic while the latter for other traffic.

For clarification, we list the key points of the classes as follows.

EF class:

- The class is designed for low delay and low jitter traffic.
- The class is generally specified by a configured rate that each core node for this class MUST have such bandwidth from long to short time-scales.
- Traffic that belongs to the class SHOULD not exceed the configured rate, otherwise, extra traffic MUST be dropped.
- Supporting the class requires enough bandwidth but little buffer space.

AF class:

- The class can be used for delay insensitive traffic.
- The class is specified by a rate that is achieved from long to short time scales. It may also be specified by other metrics, e.g., delay, packet loss, etc.
- The class may have subclasses that have different drop precedence.
- However, there is no clear definition of the class yet. Here, “clear definition” refers to mathematical formulas that calculate mean traffic rate in a wide time scale.

Clearly, EF traffic has distinctive QoS characterization from AF traffic. The differentiation between EF and AF can be mainly realized by traffic engineering. In particular, it arises a need of routing differentiation that can assure QoS in a relatively long period. We will discuss the issue in the following chapters.

2.2 Traffic engineering

Traffic engineering refers to a set of mechanisms including classifier, marker, and shaper. In a DiffServ network, edge nodes provide more traffic engineering than core nodes so as to achieve scalability. An edge node carries out traffic classification and marks every coming packet with DHCP that identifies the packet’s forwarding class. A core node mainly carries out forwarding that forwards each packet according to the DHCP of the packet. We show the traffic engineering in a DiffServ network in Figure 1.

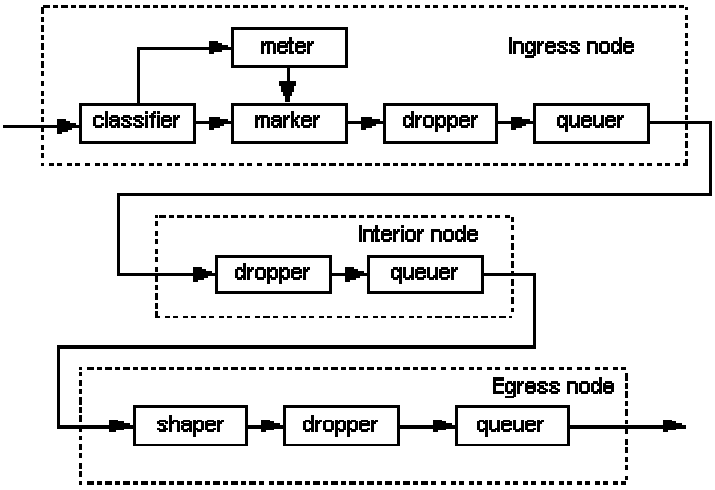

Figure 1. Traffic engineering in a DiffServ domain

2.3 QoS routing

QoS based routing, as well called constraint based routing or class based routing, has been regarded as an essential mechanism for providing QoS guaranteed services in the Internet [7-8]. It works with other technologies (e.g., traffic engineering) for efficiently managing limited network resources of IP networks. It aims to achieve two key goals, i.e., improving network resource utilization and achieving users' QoS satisfaction. Great efforts have been done in this area including such aspects as QoS routing architecture, QoS routing protocols, QoS routing algorithms, and so on [9-12]. Figure 1 shows the DiffServ architecture with QoS routing.

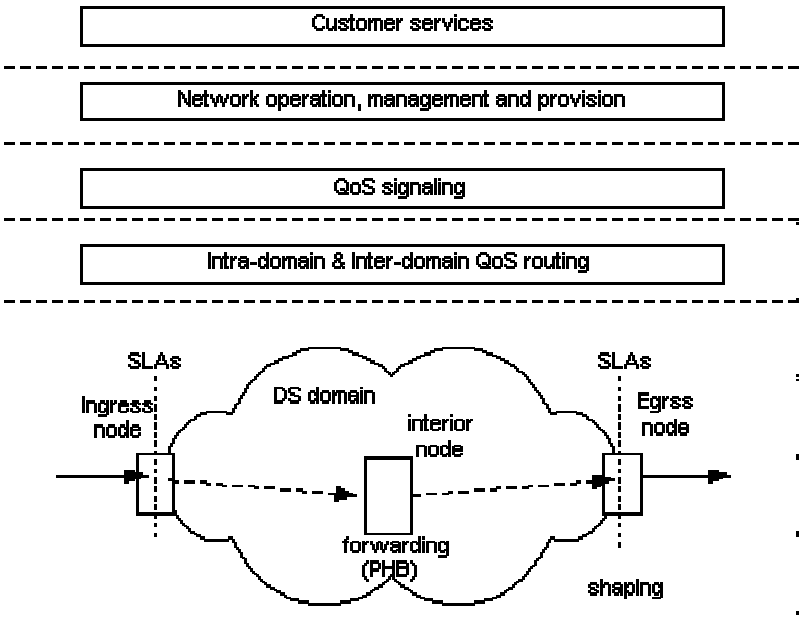

Figure 2. DiffServ architecture with QoS routing

On the other hand, most previous work only considers single class, in other words, there are no routing differentiation in DiffServ networks.

2.4 Intra-domain QoS routing in DiffServ networks

In this subsection, we introduce recent progress on QoS routing in DiffServ networks.

2.4.1 QOSPF [18]

QOSPF is based on OSPF and extends it with QoS metrics and QoS routing algorithms. It provides the capability to compute bandwidth constraint paths but without class differentiation. However, Oliveira et al, utilizes a differentiation function to computes routes for each class in DiffServ framework [19].

2.4.2 Traffic engineering extension to OSPF [20]

In this draft, OSPF is extended to support traffic engineering by using opaque LSAs. The extensions provide 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, which includes the following sub-TLVs.

1 - Link type (1 octet)
2 - Link ID (4 octets)
3 - Local interface IP address (4 octets)
4 - Remote interface IP address (4 octets)
5 - Traffic engineering metric (4 octets)
6 - Maximum bandwidth (4 octets)
7 - Maximum reservable bandwidth (4 octets)
8 - Unreserved bandwidth (32 octets)
9 - Resource class/color (4 octets)

The draft only proposes format but lacking consideration on its applications.

2.4.3 Extensions to OSPF for per-class routing [21]

Based on [20], the draft proposes corresponding extension to OSPF for support of Traffic Engineering on a per-Class-Type basis. The draft add new sub-TLVs:

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 reservable bandwidth) is referred to "Maximum Reservable Aggregate Bandwidth"
sub-TLV 8 (unreserved bandwidth) is referred to "Unreserved Bandwidth for Class-Type 0".

Similar to [20], the proposal lacks considerations on the applications.

From observing the above proposals, we notice that there are two approaches to achieve per-class routing: One is based on single-class dissemination [18-20] and the other is based on per-class dissemination[21]. It is worthwhile to compare these two approaches and study the latter especially.

3 Issues on QoS management in DiffServ networks

In this section, we briefly present key issues and solutions on QoS management in DiffServ networks.

3.1 Issues

QoS management in DiffServ networks aims to address two key issues:

- Provision network resources based on traffic demand.
- Adjust network resources against network congestion.

The first issue contains the following subtopics as illustrated in Figure 3:
Given network and traffic demand,

1. Classify traffic and map traffic to DiffServ forwarding classes.
2. Map DiffServ classes to network components with the help of routing.
3. Map QoS to network components and provision network resources at each node.

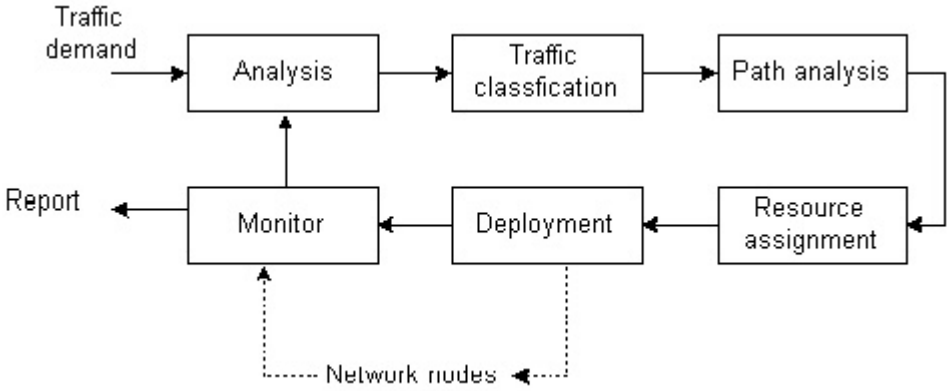

Figure 3. Provision procedures in QoS management

The second issue is to manage network resources dynamically against network congestions. This contains such sub-topics as follows as illustrated in Figure 4.

1. Enforce the provision of network resources (with the help of traffic engineering).
2. Measure traffic performance and monitor network congestion.
3. Adjust network resources allocation against relatively long term congestions with the help of routing.
4. Re-provision network resources against long-term congestions.

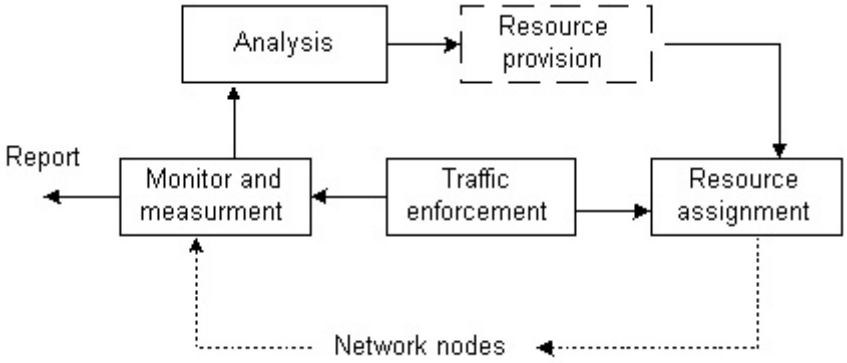

Figure 4. Traffic management in QoS management

3.2 Problems

In the subsection, we present key problems of above issues in the area and give brief explanation to each problem.

- Problem 1: Lacking efficient mechanisms for traffic classification.

  Traffic classification is a key component of traffic engineering in DiffServ networks. A biased traffic classification may significantly degrade traffic performance. A traffic flow may be classified based on:
  - TCP/UDP
  - TCP/UDP port number
  - Source/destination IP address
  - Pricing information
  - Measurement based

  Nevertheless, it is important to develop efficient mechanisms for traffic classification.

- Problem 2: Lacking guidance for mapping traffic to forwarding classes.

  There are many ways for mapping traffic to DiffServ forwarding classes. For example, a DiffServ network may support:
  - Only EF classes
  - EF for realtime traffic and/or AF for non-realtime traffic
  - AFx classes for various traffic
  - EF class for UDP, AFx classes for TCP
  - EF class for IP addresses whose users buy EF service

  On the other hand, there lacks guidance on when and how the DiffServ forwarding classes are used in order to support a set of services.

- Problem 3: Lacking optimized algorithms for resource provisions.

  Differentiated services may have different traffic characteristics as well as different QoS requirements. Supporting differentiated services need to

provision network resources (e.g., bandwidth, buffer). Efficient and optimized algorithms for resource provision are still missing.

- Problem 4: Few mechanisms for scheduling aggregated traffic flow.

  In DiffServ networks, small-volume traffic flows are aggregated into big-volume flows. This requires mechanisms for scheduling aggregated traffic that can guarantee both QoS of inter-flows and that of intra-flows.

- Problem 5: Lacking mechanisms in response to network congestions.

  Network congestions result in performance degradation of differentiated services. This requires mechanisms for adjusting network resources allocation in response to network congestions. Network congestions may be resulted from
  - Unsuitable resource allocation
  - Significant traffic variation

Besides, in order to address the above problems, there are in need of following mechanisms.

- Mechanisms for routing traffic
- Mechanisms for congestion detection

Unfortunately, there lack deep investigation on the above topics. In this paper, we focus on QoS routing mechanisms for addressing the above issues and problems.

4 Proposed solution - PERD

In this section, we propose a routing mechanism, called per-class routing based on per-class dissemination (PERD) for routing in DiffServ networks.

4.1 Motivation

As we presented in previous chapters, in DiffServ networks, delay sensitive traffic (e.g., real-time traffic) and loss sensitive traffic (e.g., email) can be conveyed by EF class and AF class, respectively. Normally, EF class has higher priority than AF traffic so that without special care on both classes, EF traffic may preempt resources for AF traffic. On the other hand, EF traffic has different QoS requirements from AF traffic, that is, EF class aims to achieve minimum delay while AF class aims to achieve maximum throughput. Therefore, it is necessary to designate per-class traffic engineering mechanisms that can guarantee QoS of each class. In this paper, we propose PERD that is based on per-class traffic engineering. The PERD algorithm is based on and extends the proposals in Chapter 2.4.

4.2 PERD mechanism

The PERD mechanism realizes routing differentiation for different DiffServ forwarding classes. In order to guarantee the required QoS of each class, the PERD applies different (QoS) routing algorithms as well as routing metrics for each class in order to meet different QoS requirement of the classes.

The PERD mechanism assumes that the resources of a DiffServ network is provisioned, for example, given a link, 20% of the link's bandwidth is allocated for EF class, and the rest is allocated for AF class. In the PERD, network resources for each class are maintained in an extended link state database that contains link resources for each class. When the resources of a class change on a link, new link state is advertised to the network. Route computation is carried out for each class and results in route entries for each class.

Figure 5 gives the example implementation of PERD based on OSPF.

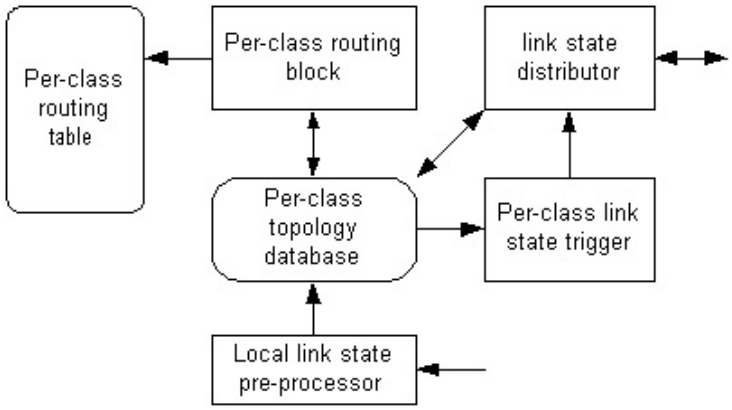

Figure 5. Example implementation of PERD

In this architecture, the "local link state pre-processor" determines when and how to input local link state information into "per-class topology database". One of the key functions of the processor is to transform various local link state information into the variables that can be used by routing. The "per-class link state trigger" determines when to broadcast the link state change of each class. The "per-class topology database" stores the link state information of each class. The "per-class routing block" computes paths for each class and stores paths into the "per-class routing table".

4.3 PERD principles

The PERD algorithm includes the following key principles.

- In a DiffServ network, a class in PERD can represent a PHB (Per-hop behaviour) or a group of PHBs. This simplifies routing. For example, traditional routing is the special case of PERD.

- Network resources for each class are maintained in an extended link state database that contains link resource for each class. When the resources of a

class change on a link, new link state is advertised to the network. In addition, a trigger mechanism can be applied in order to reduce advertisement cost.

- 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 (e.g., CBQ, WFQ, PQ, RIO). 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 (referred to delay) 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 network as a whole.

- Nevertheless, it is well known that route computation with more than one constraint is normally NP-hard. Hence, heuristics algorithms should be developed.

4.4 PERD routing procedures

As shown in Figure 6, the PERD mechanism consists of a number of procedures.

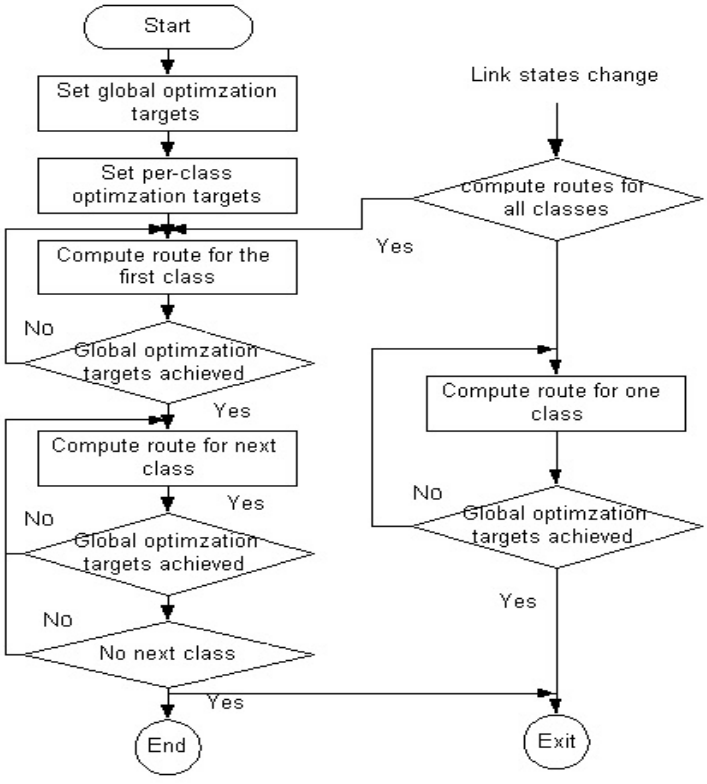

Figure 6. Routing procedures of PERD

In this figure, per-class route computation performs to achieve not only per-class optimization targets but also global optimization targets. One kind of global optimization targets is consisted by weighting per-class optimization targets.
In particular, the PERD mechanism implies two cases of using routing algorithms:

- A single routing algorithm for all classes
- Different routing algorithms for different classes

There are various routing algorithms for a single class routing, which can be naturally used in the PERD, for example, Widest-Bandwidth (WB) and Lowest-Cost (LC)[7]. In addition, there is likely a need of differentiating routing algorithms for different classes. For example, the WB algorithm may be suitable for AF class through balancing traffic and the LC algorithm may be suitable for EF class through searching for the lowest delay path. Yet, this needs further study.

5 Performance study

In this section, we investigate the performance of the PERD through a set of simulations. First, we briefly introduce simulation environments; then we perform our simulations to compare the performance (throughput, delay and cost) of PERD with other two routing algorithms, the shortest path (SP) algorithm and the widest bandwidth (WB) algorithm. The two algorithms and more simulation results can be found in [17].

5.1 Simulation Environment

We use QoS Routing Simulator (QRS) as our simulation platform [13].

5.1.1 Simulation topology - NSFNET Topology

In our simulations, we use an ISP network as our study topology. As shown in Figure 7, the network consists of 14 nodes and 22 links. We assume that both node 1 and node 4 are boundary routers and all traffic enter into the network through node 1 and leave the network via node 4. The bandwidth of every link along the shortest path from node 1 to node 4 (i.e., node1->node11->node12->node4) is 3M. The capacity of other links is 2M. Notably, the link bandwidth is rather randomly selected without any preference and the traffic is simple so as to make preliminary study.

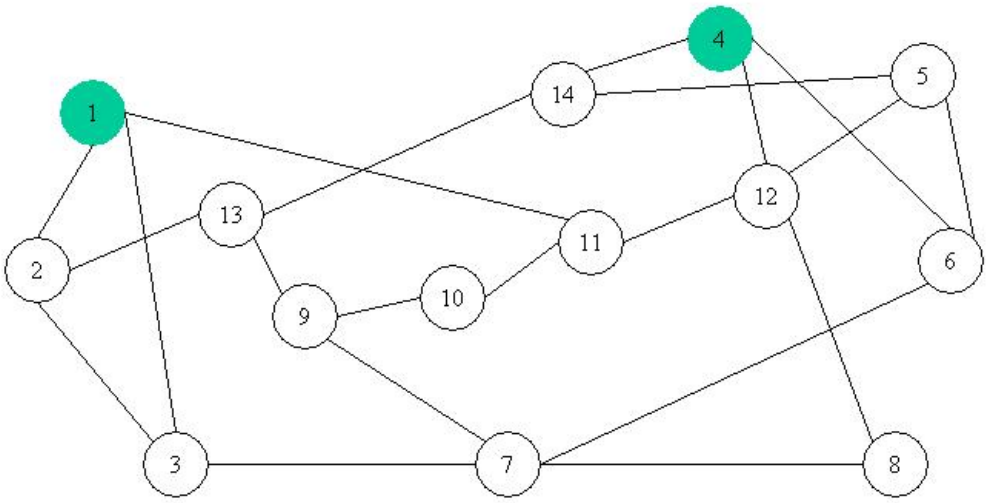

Figure 7 NSFNET-T1 backbone topology

In the network, we notice that the minimal cut between node 1 and node 4 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. In particular, we follow a provisioning policy that the bandwidth of each link is provisioned as 20% for EF class, 30% for AF1 class and the rest for AF2 class. Therefore, the total throughput between node 1 and node 4 consists of 1.4M EF traffic, 2.1M traffic and 3.5M BE traffic.

5.1.2 Traffic

In our simulations, we use three forwarding classes—EF, AF1 and AF2 class for real-time traffic, web traffic, and best-effort (BE) traffic, respectively. We classify BE traffic into AF2 class in order to assure that BE traffic can have a minimum rate even in the worst case.

In order to study the performance of PERD verus traffic load, we vary the traffic load of the network by increasing 0.7Mb/s (10% of full utilization) each time until full 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 1 Traffic load

5.1.3 Metrics

We use average throughput and end-to-end delay as measures of network service performance. Notably, it is important to investigate delay especially for real-time traffic.

For the first metric, we record the number of received packets at the destination as the throughput. We regard the length of the packet as 544 bytes including 512-byte payload and 32-byte header.

For the second metric, we calculate the time duration that a packet travels from the source node to the destination node, that is, delay is packet's receiving time minus 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. The end-to-end delay is composed of propagation delay, transmission delay and queuing delay. In our simulations, we assume that the propagation delay is a constant and is equal to 10ms. It is known that transmission delay is packet size divided by bandwidth. In addition, queuing delay occurs inside the network since routers generally need to store packets for some time before forwarding them on an outbound link. Queuing 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.

The unit of delay in this paper is ms. We run each simulation for 1000s.

5.2 Simulations

In this subsection, we present the performance study of PERD in comparison with SP and WB algorithms.

5.2.1 Throughput vs. Load Percentage

We present the simulation results of throughput from Figure 8 to 11. For each algorithm, Figure 8 shows the throughput of all forwarding classes while Figure 9-11 shows the throughput of EF, AF2 and AF2 class, respectively.

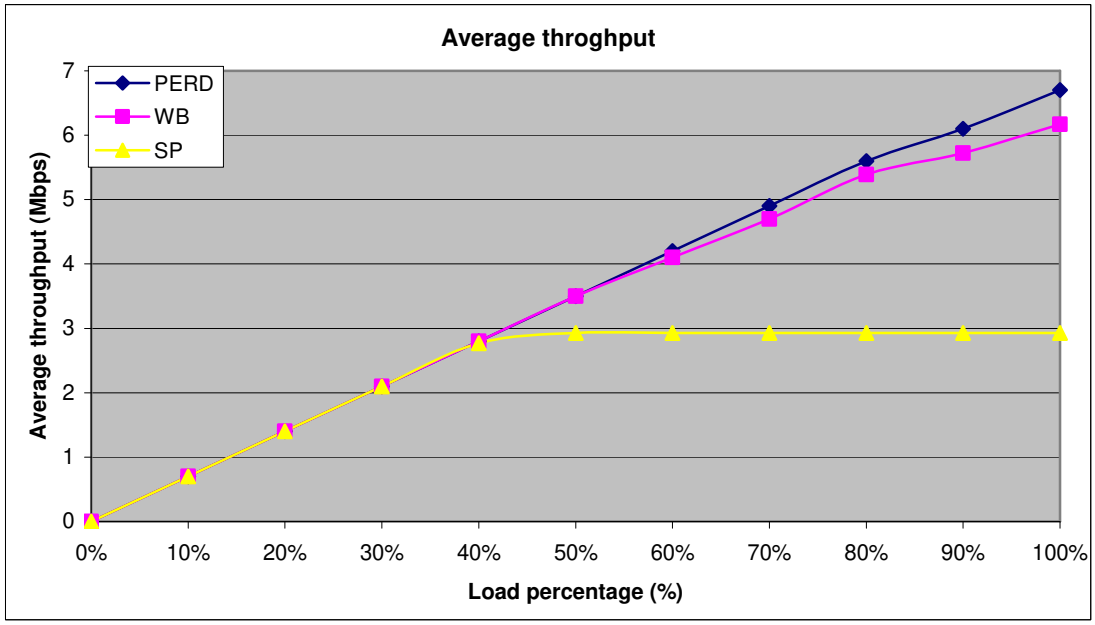

Figure 8 Throughput under different routing algorithms

As shown in Figure 8, when the traffic load is lower than 40%, all algorithms achieve similar performance. However, when the load increases more, PERD and WB achieve

much better throughput than SP. Clearly, this is because SP only selects the shortest path while PERD and WB are able to select multiple paths. Moreover, when the traffic load is becoming higher, for example, more than 60% of the total network capacity, PERD shows advantage over WB. In the case of full load, the total throughput of PERD is 6.73M and 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%. This proves that PERD achieves better throughput than WB algorithm.

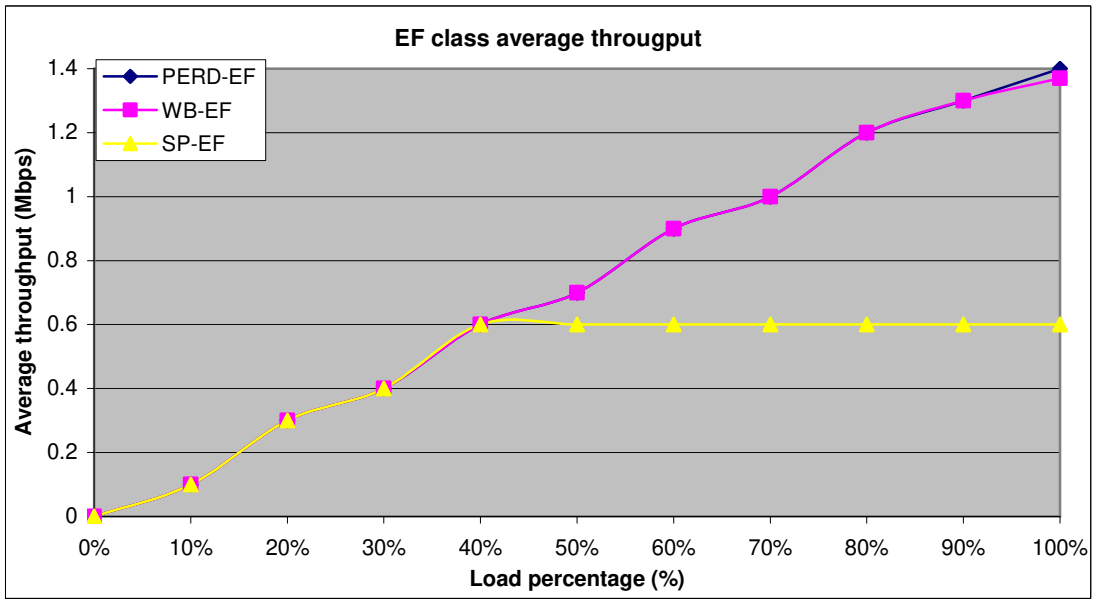

Figure 9 EF throughput under different routing algorithms

Figure 9 shows the throughput of EF class. From the figure, we notice that both PERD and WB achieve much better performance than SP, same as in Figure 8. On the other hand, PERD and WB achieve similar performance of EF class

.

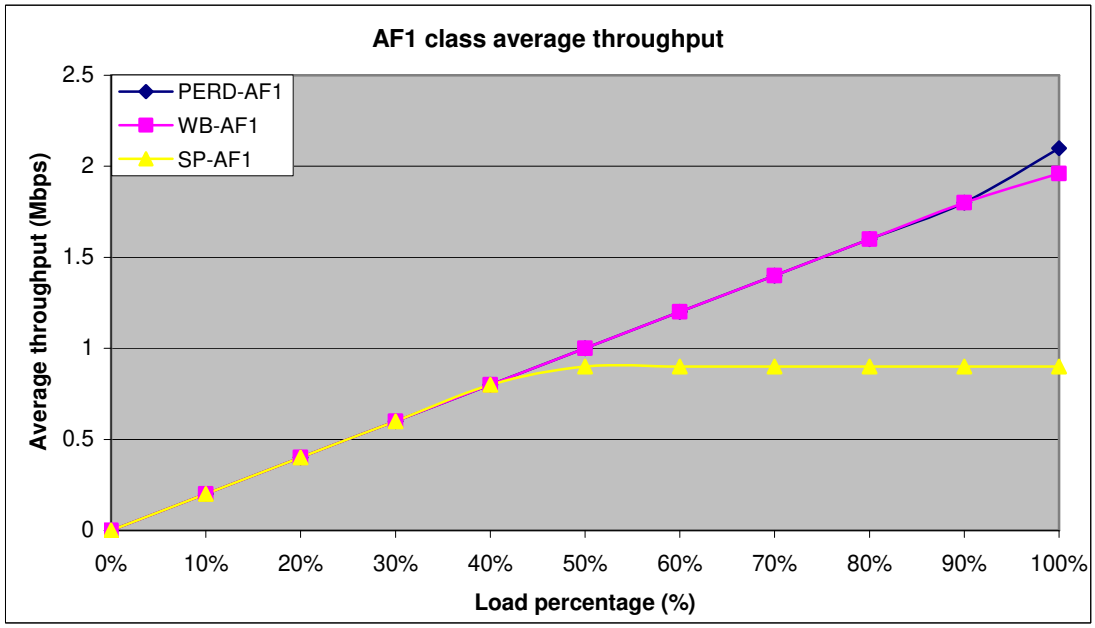

Figure 10 AF1 throughput under different algorithms

As shown in Figure 10, we find the same result that PERD and WB achieves better performance than SP. Besides, we notice that PERD performs slightly better than WB when traffic load is full. In such case, with PERD the throughput of AF1 class is 2.1M while that with WB is 1.96M.

The throughput of AF2 class is shown in Figure 11. It shows that PERD achieves better performance than WB and much better performance than SP. In other words, PERD can guarantee the performance of low priority traffic.

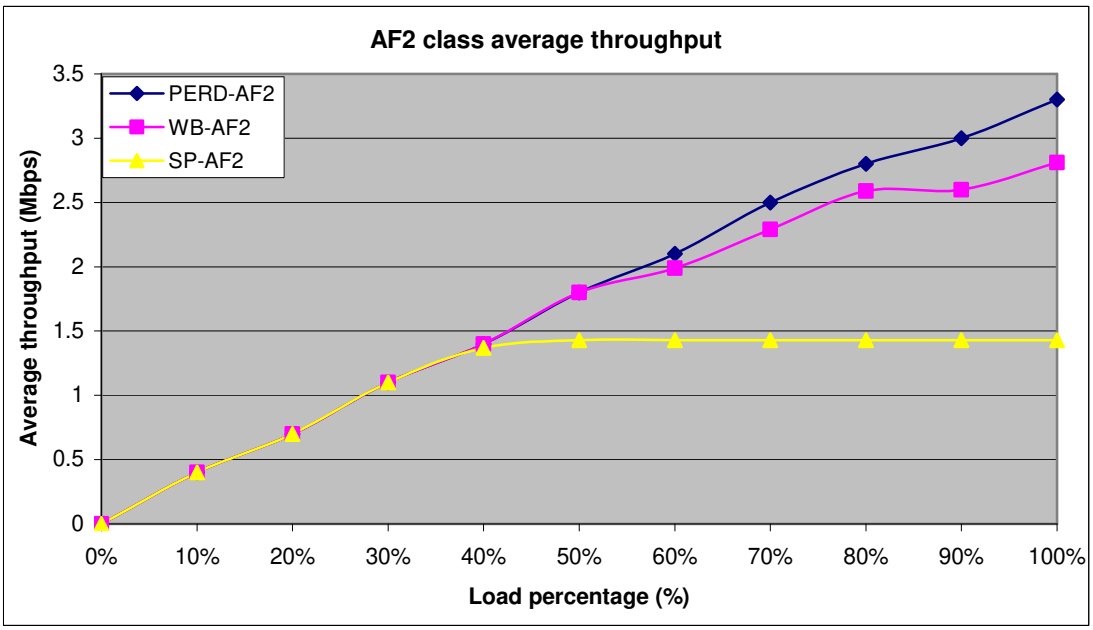

Figure 11 AF2 throughput under different algorithms

By observing Figure 8 to Figure 11, we find that PERD can improve the total throughput as well as per-class throughput. The most significant difference is PERD is able to achieve better performance of low priority traffic (e.g., AF2 class) than WB and SP. The main reason behind this result is that PERD performs better load balancing. For example, PERD is able to evenly distribute EF traffic over the network.

5.2.2 Delay vs. Load Percentage

We present delay results of each class in Figure 12-Figure 14. In Figure 12, we show the average delay of EF class with the increase of traffic load. As shown in the figure, the average delay of SP is the smallest. This is because SP selects the shortest path that has minimum hops. PERD results in more delay than SP, but clearly less than WB algorithm. Since EF class is designed for delay sensitive traffic, PERD shows advantage over WB in term of delay.

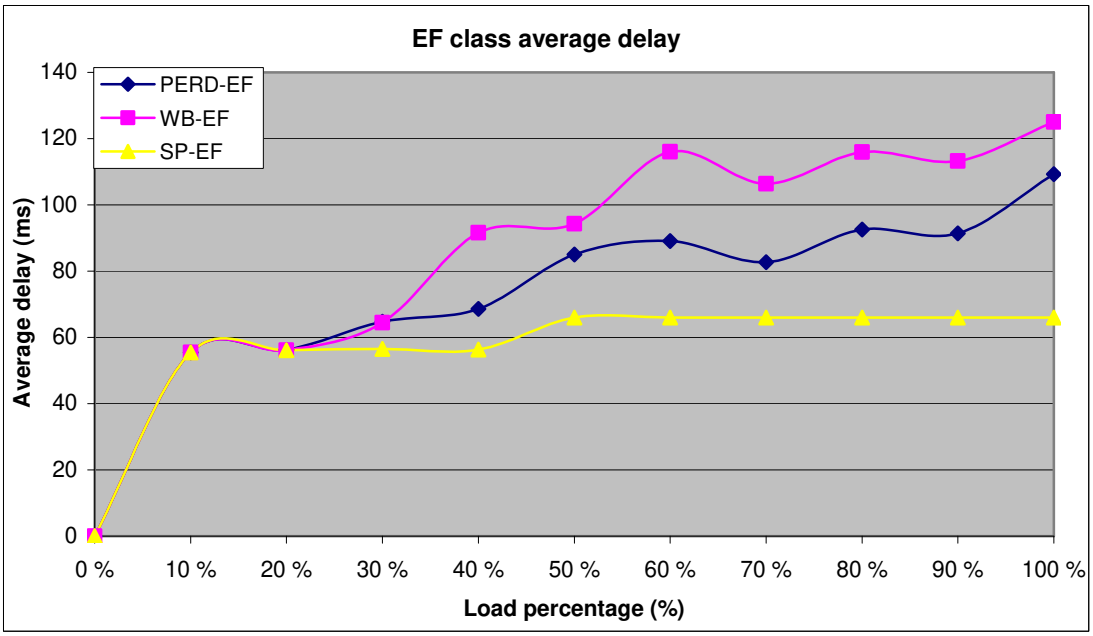

Figure 12 EF class average delay under different algorithms

Figure 13 presents the delay performance of AF1 class under three routing algorithms. For AF1 class, SP achieves best delay performance and PERD achieves slightly better delay performance than WB.

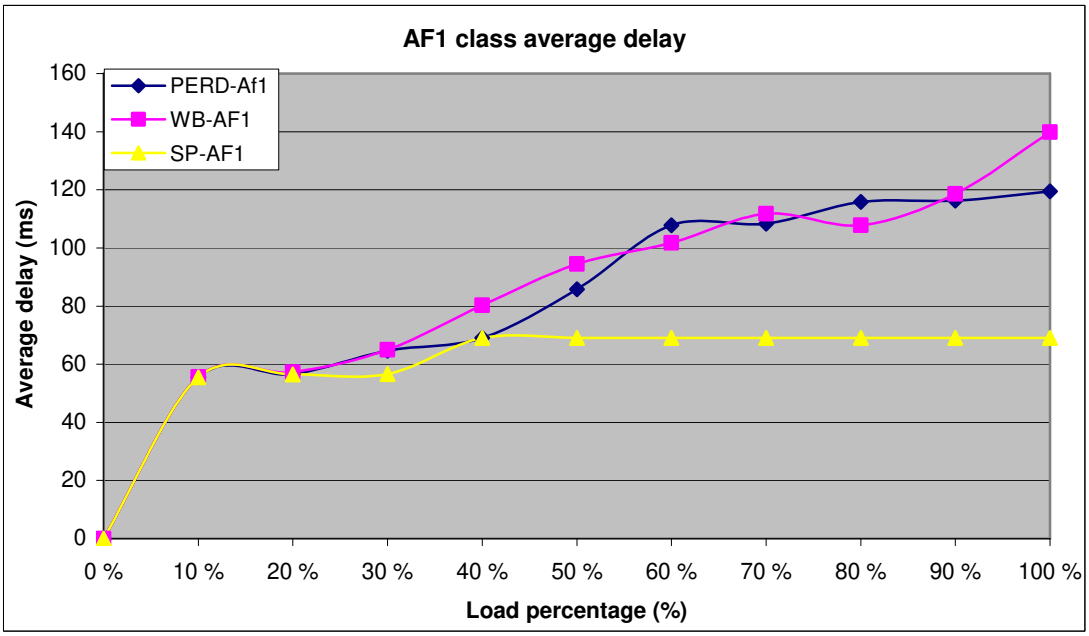

Figure 13 AF1 class average delay under different algorithms

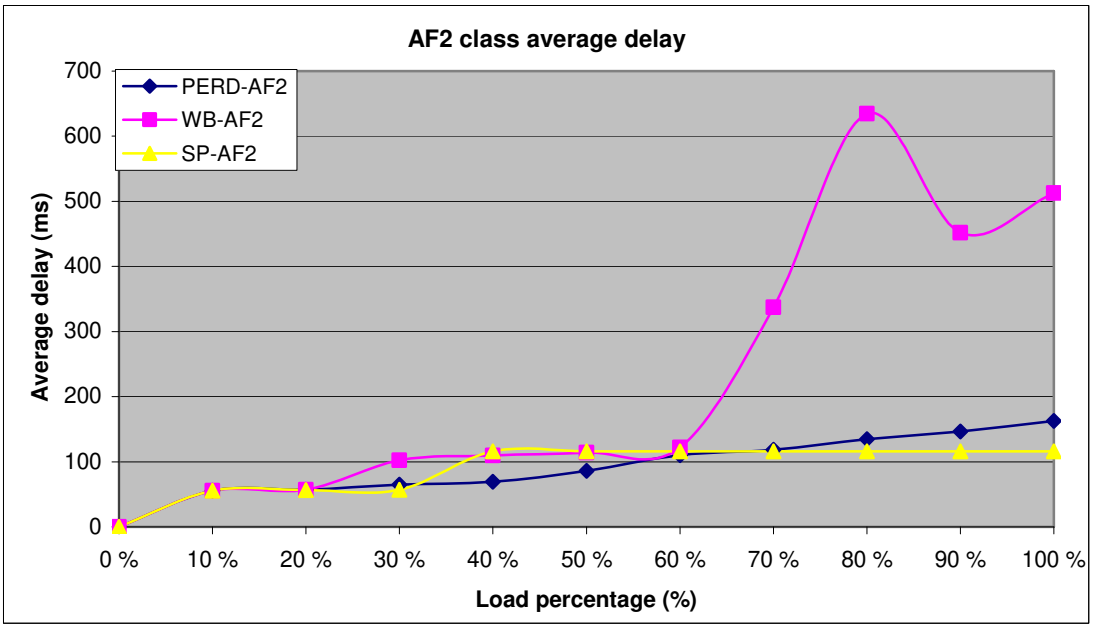

Figure 14 AF2 class average delay under different algorithms

In Figure 14, we show the delay performance of AF2 class. We notice that PERD achieves similar delay performance as SP algorithm and significantly better than WB algorithm. This is mainly because PERD can reduce delay by distributing real-time traffic based on the per-class available bandwidth.

In summary, PERD can improve network throughput as well as delay, especially for low priority traffic.

5.2.3 Cost vs. Hold-timer Value

In previous subsections, we prove that PERD can help to improve throughput and reduce delay by reasonably distributing traffic within networks. However, those benefits do not come for free. In the section, we study the relationship between performance and cost of PERD in comparison with WB.

We use total processing time consumed by QOSPF during the simulation time to represent the cost of PERD. The processing time of each action of QOSPF in a node is set as shown in Table 2 [11].

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 2 Cost of each QOSPF action

In this section, we use TB (threshold based) link state update 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.

In our simulations, we set the threshold value as 10%. We can express this relation as:
If $|bw^0 - bw^c| / bw^0 > 10\%$. (1)

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 a link state update packet carries four parameters: total residual bandwidth, EF class available bandwidth, AF1 class available bandwidth and AF2 class available bandwidth.

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 3 Traffic parameters

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 3 and run the simulator for 1000s. We record the time consumed by QOSPF in every node during the simulation time, and then calculate the sum.

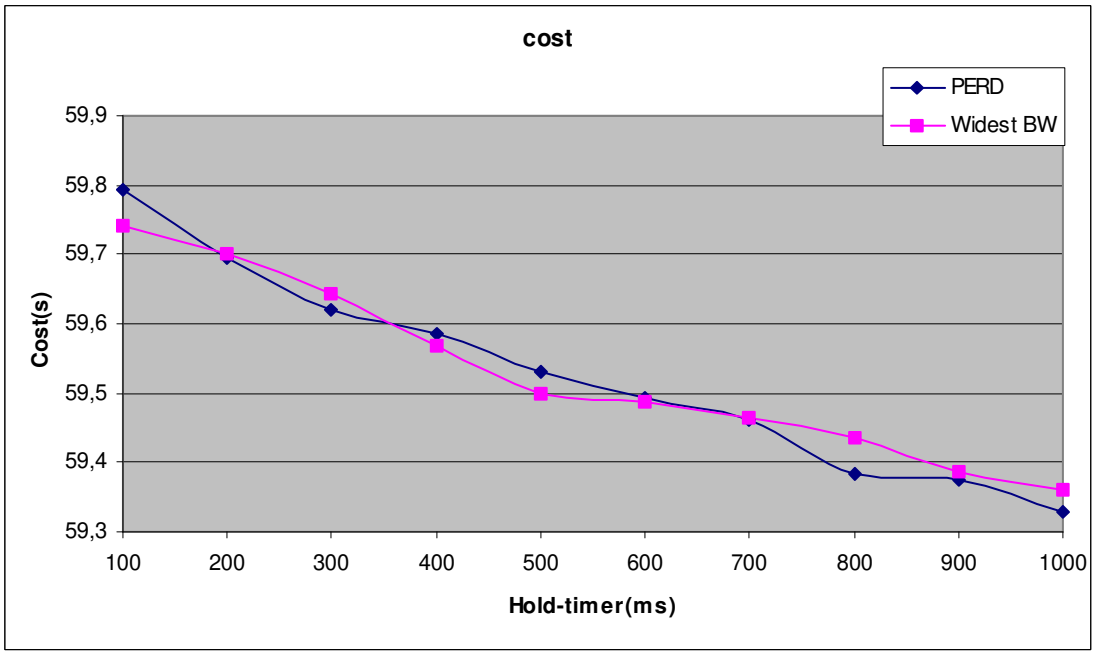

Figure 15 Cost under different hold-timers

We run the simulation for 1000s. As shown in Figure 15, we find that with the increase of the hold-timer value, the cost drops under both PERD and the WB routing algorithm. We can see that PERD results in similar cost as WB, which means that PERD does not increase cost in comparison with WB algorithm.

6 Conclusions and future work

In this paper, we discussed routing problems in DiffServ networks. We focused on routing optimization and proposed the PERD mechanism in order to achieve maximum performance of each class. The key idea of the mechanism takes the available bandwidth of each individual class into consideration and selects routes based on the available resources of each class. Through simulation study, we proved that the mechanism is able to improve the performance of each class, especially for low priority classes. Moreover, it does not significantly increase cost in comparison with other QoS routing algorithms.

In our future work, we will investigate the algorithm with various networks and traffic. We will also combine the mechanism into MPLS networks. In particular, we aim to develop efficient algorithms that can achieve maximum performance of each class as well global optimization. Another topic for future work is to develop a framework and algorithms that can provision network resource upon traffic changes.

References

DiffServ MPLS networks

[1] S. Blake, et al. An Architecture for Differentiated Services. RFC2475. December 1998.

[2] D. O. Awduche, et al. Requirement for Traffic Engineering over MPLS. RFC 2702.September 1999.

[3] D. O. Awduche, et al. Requirement for Traffic Engineering over MPLS. RFC 2702. September 1999.

[4] V. Jacobson, et al. An Expedited Forwarding PHB. RFC 2598. June 1999. September 1997.

[5] J. Heinanen, et al. Assured Forwarding PHB Group. RFC 2597. June 1999.

[6] B. Gleeson, et al. A Framework for IP Based Virtual Private Networks. IETF RFC2764 (2000)

Fundamental documents

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

[8] S. Chen and K. Nahrstedt. An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions. IEEE Networks. November/December 1998.

Performance and cost analysis of QoS routing

[9] G. Apostolopoulos, et al. Intra-Domain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis. IEEE Network Sep./Oct. 1999

[10] G. Apoatolopouloa, R. Guerin, S. Kamat. Implementation and Performance Measurements of QoS Routing Extensions to OSPF. Proceedings of INFOCOM'99, New York, NY, March 21-25, 1999.

[11] Z. Ma, P. Zhang, R. 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), Dallas, Texas USA, May 29-31 2001

[12] P. Zhang, R. Kantola. April 2000. Building MPLS VPNs with QoS Routing Capability Fifth International Symposium on Interworking (Interworking'2000). October 3-6, 2000, Bergen, Norway.

QoS Routing Simulator

[13] P. Zhang, R. Kantola, Z. Ma. Design and Implementation of A New Routing Simulator. 2000 SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'2K). July 16-20, 2000, Vancouver, Canada.

[14] QoS Routing Simulator. http://www.tct.hut.fi/~pgzhang/QRS/index.html

Inter-domain routing:

[15] IETF WG Inter-domain Routing (idr). http://www.ietf.org/html.charters/idr-charter.html

[16] P. Zhang, R. Kantola. Mechanisms for Inter-Domain QoS Routing in Differentiated Service Networks. 1st International workshop on quality of future Internet Services (QofIS'2000). September 25-26, 2000, Berlin, Germany.

PERD

[17] W. Zhou. Provision and route optimizations in Differentiated Services networks. Master thesis. Espoo Finland 2002

Traffic engineering extension to OSPF

[18] R. Guerin, et al. QoS routing mechanisms and OSPF extensions. IETF 2676. Aug 1999

[19] M. Oliveira, et al. Quality of service routing in the Differentiated Services framework. 2000

[20] D. Katz, et al. Traffic engineering extension to OSPF. IETF draft. September 2002
[21] F. Faucheur, et al. Extensions to OSPF for support of Diff-Serv-aware MPLS traffic engnieering. IETF draft. Feb 2001