This is to certify that the

thesis entitled

END-TO-END QOS SUPPORT IN
DIFFERENTIATED SERVICE INTERNET

presented by

Fugui Wang

has been accepted towards fulfillment
of the requirements for

Master's degree in Computer Science
& Engineering

Prasant Mohapatra
Major professor

Date 5/8/00

END-TO-END QOS SUPPORT IN DIFFERENTIATED SERVICE INTERNET

By

Fugui Wang

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science and Engineering

2000

ABSTRACT

END-TO-END QOS SUPPORT IN DIFFERENTIATED SERVICE INTERNET

By

Fugui Wang

Differentiated service (DiffServ) is recently proposed as a scalable solution for providing quality of service (QoS) in the Internet. DiffServ does not maintain per-flow state in Internet core routers. Packet level service differentiation is done in core routers on a per-hop behavior basis which makes the end-to-end QoS assurance more challenging. Assured service and premium service are the two classes of services proposed by IETF for DiffServ.

Assured service uses static service level agreement (SLA) between Internet domains based on statistical estimation. We have proposed an efficient marker called random early demotion and promotion (REDP), which provides better fairness and bandwidth assurance. The fairness of REDP is obtained from the random and early demotion of packets, and the bandwidth assurance is impoved by the promotion capability. The performance of the REDP marker is evaluated through the *ns* simulator. The results prove the fairness and the higher resource utilization of the REDP scheme compared to the previously proposed marking models. Premium service uses dynamic SLA, which introduces the scalability problem for the bandwidth broker (BB). A novel concept of pipe is proposed as a scalable solution for bandwidth management for premium service, which has negligible signaling overheads for BBs and maintains high resource utilization. The advantages of pipe is derived from the hybrid nature of SLA negotiation; the static part is predetermined and the dynamic part is updated periodically. The performance of pipe is evaluated through simulation. The results indicate the effectiveness and low overhead of the implementation of pipe.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Prasant Mohapatra for his guidance in this research. I would also like to thank my committee members Dr. Lionel M. Ni and Dr. Philip K. McKinley for their time to review this work.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

The current Internet uses the best-effort service model. In this model the network allocates bandwidth among all the contending users as best as it can and attempts to serve all of them without making any explicit commitment to rate or any other service quality. With the proliferation of multimedia and real-time applications, it is becoming more desirable to provide certain Quality of Service (QoS) guarantee [15][1] for Internet applications. Furthermore, several enterprises are willing to pay an additional price [7][8][3] to get preferred service in return from the Internet service providers. The Internet Engineering Task Force (IETF) [21] has proposed a few service models and mechanisms to ensure Internet QoS. Notably among them are the Integrated Services (IntServ) [4][11] model and the Differentiated Services (DiffServ) model [22][26].

1.1 Integrated Services

Like the circuit-switched service in current telephone system, IntServ [13][14] could provide per-flow QoS guarantee. IntServ defines two service classes: Gruanteed Service [5] and Predictive Service [6]. IntServ uses RSVP [11] as a signaling protocol for applications to reserve resources. All of the routers in the path keep a soft state of the flow and use the soft state to make further admission control decisions. Service quality is ensured at a per-flow level. IntServ is desirable for flows requiring strict end-to-end QoS. However, IntServ has two major drawbacks [23]. First, the amount of state information increases proportionally with the flow leading to poor scalability

at the core routers. Second, implementation of IntServ requires changes in the Internet infrastructure, which is deemed economically infeasible considering the vast size of the Internet.

1.2 Differentiated Services

Because of the difficulty in implementing and deploying IntServ and RSVP, DiffServ is introduced [23].

DiffServ provides a simple and predefined Per-Hop Behavior (PHB) [26] level service differentiation in the Internet core routers. Functionalities of per-flow or flow aggregate marking, shaping, and policing are pushed to the edge routers. Thus it does not suffer from the scalability problem as IntServ and requires less changes in the Internet infrastructure. The Internet core routers, which have scalability problem in IntServ, will do service differentiation only based on the differentiated service code point (DSCP) [25] of the packet. Only minor changes need to be done in the core routers in order to support DiffServ. Currently IETF defines two service classes for DiffServ: Assured Service and Premium Service. Details of these two service classes are discussed in Chapter 2.

Compared to IntServ, DiffServ requires less implementation changes in the Internet and is more scalable. But without end-to-end resource reservation, DiffServ has more difficulties in providing end-to-end QoS guarantee. This thesis will be focused on solving the problems of end-to-end QoS support in DiffServ model.

1.3 Thesis Organization

Details about the concept and architecture of DiffServ are reported in Chapter 2. Chapter 3 discusses a Random Early Demotion and Promotion (REDP) marker, which is proposed to replace the leaky bucket marker for assured service. Chapter

4 introduces the concept of pipe as an efficient resource management scheme for premium service. Chapter 5 presents the conclusions and future work.

CHAPTER 2 DIFFERENTIATED SERVICES

Today's Internet comprises of multiple interconnected autonomous domains, or administrative domains [24]. Each domain has core routers that are interconnected by backbone networks. End users or the other domains are interconnected to the each other through edge routers. A typical DiffServ architecture is shown in Figure 2.1. Before entering a DiffServ domain, a packet is assigned a DiffServ Code Point (DSCP) [25] by the marker located in the edge router [26]. When the packet reaches a DiffServ aware router, the DSCP of the packet will be checked to determine the forwarding priority of the packet. The DSCP of a packet may be changed when it crosses the boundary of two domains. For example, in Figure 2.1, a packet sent from host A to host B may be marked as high priority DSCP when it enters domain 1. At the boundary of domain 1 and domain 2, if domain 1 has not negotiated enough traffic forwarding rate with domain 2 for that priority, the marker at domain 2 may have to re-mark that packet as a low priority DSCP before it would forward the packet to domain 2.

2.1 Service Classes

Currently, IETF has defined one class for Expedited Forwarding (EF) [27] and four classes for Assured Forwarding (AF) [28].

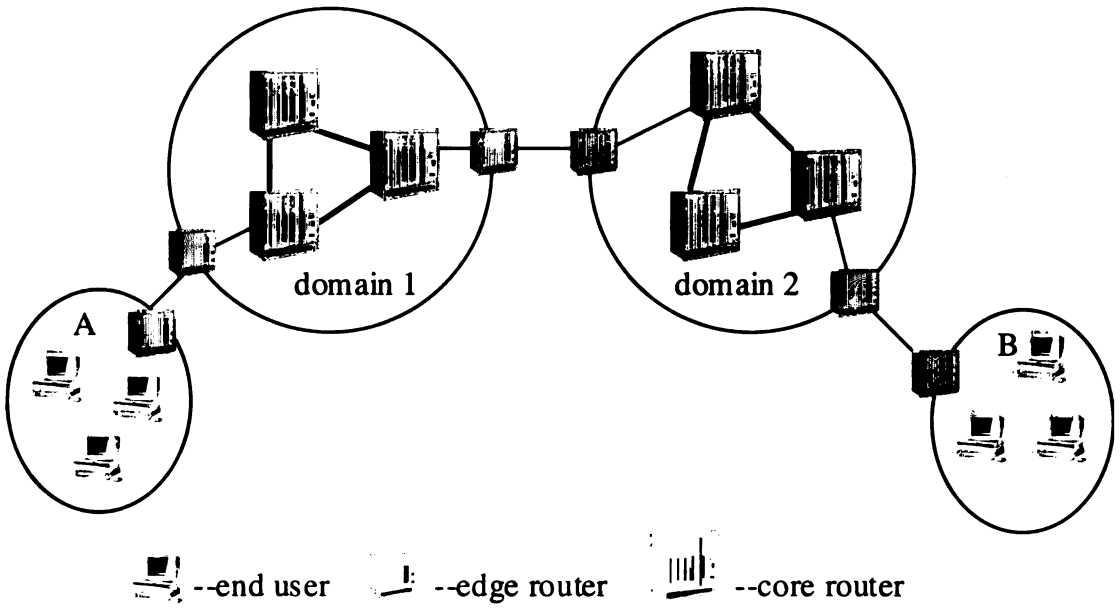


Figure 2.1 A typical DiffServ architecture.

2.1.1 Expedited Forwarding

EF was originally proposed by Jacobson in [29] as Premium Service. Premium service was proposed to meet an emerging demand for a commercial service that requires low loss, low latency, low jitter, and assured bandwidth. It is expected that premium traffic would be allocated only a small percentage of the network capacity and would be assigned to a high-priority queue in the routers. One use of such a service might be to create "virtual leased lines", saving the cost of building and maintaining a separate network. It is ideal for real-time services such as IP telephony, video conferences and the like.

Premium service levels are specified as a desired peak bit-rate for specific flow (or aggregation of flows) [32]. Since it is assigned a high priority over the other traffics, premium service has very small or nonexistent queuing delay.

2.1.2 Assured Forwarding

AF was first proposed by Clark in [30] as Assured Service. Each assured service flow (or aggregation of flows) has an "expected capacity" usage profile that is sta-

tistically provisioned. The assurance that the user of such a service receives is that such traffic is unlikely to be dropped as long as it stays within the expected capacity profile [32]. Originally, it was proposed to use the RED-In/Out (RIO) [31] approach to ensure the "expected capacity" specified by the traffic profile. The basic idea is, upon each packet arrival, if the traffic rate is within the traffic profile, the packet is marked as "In", otherwise, it is marked as "Out". In a DiffServ aware router all the incoming packets are queued in the original transmission order. However, during network congestion the router preferentially drops the packets that are marked as "Out". By appropriate provisioning, if we could make sure that the aggregate "In" packets would not exceed the capacity of the link, the throughput of each flow or flow aggregate could be assured to be at least the rate defined in the traffic profile. To ensure service differentiation, currently, the AF PHB [28] defined by IETF specifies four traffic classes with three drop precedence levels (or three colors) within each class. In all, there are twelve DSCPs for AF PHB. Within an AF class, a packet is marked as one of the three colors, *Green*, *Yellow*, and *Red*, where *Green* has the lowest drop probability and *Red* has the highest drop probability. It is expected that with appropriate negotiation and marking, end-to-end minimum throughput could be assured or at least assured to some extent.

2.2 Support in Internet Core

Within an autonomous domain, the Internet core routers will do service differentiation only based on the DSCP of a packet. No per-flow information is kept here, therefore core routers will not have scalability problem.

2.2.1 Queuing Scheme

Each core router maintains two separate queues, one for EF and one for AF[1]. When a packet arrives the core router, a packet classifier will check the DSCP of the packet to determine which queue the packet should enter. If it is an EF packet, it should enter the EF queue. Otherwise if it is an AF or best-effort packet, it should enter the AF queue. The queues could be implemented by either using priority queue (PQ) or weighted fair queue (WFQ)[2]. In both cases the EF queue should be given higher priority.

2.2.2 RIO

Service differentiation between AF and best-effort packets are implemented through a RED-In/Out (RIO) [31][9] approach. RIO is based on the random early detection (RED) [10] queuing scheme.

RED Scheme

Random early detection (RED) gateways are a simple mechanism for congestion avoidance that could be implemented gradually in current TCP/IP networks with no changes to the transport protocols. The RED gateway calculates the average queue size avg, using a low-pass filter with an exponential weighted moving average. The average queue size is compared to two thresholds, a minimum threshold min_{th} and a maximum threshold max_{th}. When the average queue size is less than the minimum threshold, no packets are dropped. When the average queue size is more than the maximum threshold, every arriving packet is dropped.

[1] If AF supports four classes, then the AF queue will be further divided into four separate queues. Since it is still not clear whether we really need to support four classes of AF, we will only assume one class of AF in the remaining part of this thesis.

[2] The performance of both queuing schemes are studied in Chapter 4

As avg varies from min_{th} to max_{th}, the packet dropping probability p varies linearly from 0 to max_{th}:

$$p = \frac{avg - min_{th}}{max_{th} - min_{th}} max_p \tag{2.1}$$

Most of the traffics in the current Internet are TCP or UDP traffics. Many UDP traffics are periodical, such as real-time audio or video streams. TCP traffics are also periodical, where the period is the Round Trip Time (RTT) of the connection. When these periodical traffics sharing a drop-tail queue, unfairness will be unavoidable during congestions. This phenomenon is called gloable synchronization or phase effect [18]. By dropping packets randomly with a varying probability, the global synchronization problem encountered in the drop-tail scheme is avoided in the RED mechanism.

RIO Scheme

The RED-In/Out (RIO) approach is based on the "expected capacity" framework where the in profile packets are marked as "In" and the out of profile packets are marked as "Out". The RIO algorithm uses two RED algorithms with different parameters for dropping packets: one for "In" packets and the other for "Out" packets. The average queue size of the all of the packets is used for the "Out" packets while the queue size of only the "In" packets will be used for the "In" packets. By choosing the parameters for the respective algorithms differently, RIO is able to discriminate against "Out" packets.

2.3 Supports at Internet Edge

Service differentiation only at the core routers may not be able to ensure the service quality of EF and AF flows. There must be some admission control approaches at the edges of the domains to limit the amount of EF and AF packets entering the domains. In DiffServ, each stub domain (or local domain) negotiates an SLA

with its Internet service provider (ISP)[12]. Individual host negotiates with the local bandwidth broker (BB) to determine how much EF or AF bandwidth it could acquire. Transit domains (domain 1 and domain 2 in Figure 2.1) also negotiate SLA between each other. Once an SLA has been set up, the BB will configure a marker for the traffic, using the negotiated traffic profile. If the traffic conforms the negotiated SLA profile, the packets will be marked with appropriate DSCP. Out of profile packets will be marked as lower priority DSCP. With appropriate signaling and negotiation, the edge routers could control the amount of packets of each service level entering the domain.

2.3.1 Marker

Usually the marker is implemented using a leaky bucket. Tokens fill the bucket with a rate equal to the negotiated rate. When a packet passes by the leaky bucket, it gets marked with the preferred DSCP only if it can grab a token from the leaky bucket. Otherwise, if the leaky bucket is empty, it should be marked as a lower priority DSCP. There are two types of markers:

1. *Leaf marker:* Leaf markers are configured for individual TCP or UDP flows in the stub domain. It could reside on the edge router or be embedded with the IP layer of the hosts. Based on the traffic profile and the actual traffic rate, the leaf marker assigns a DSCP for the packet.

2. *Intermediate marker:* Intermediate markers usually reside on the edge routers between any two domains. It usually works on the aggregate level and monitors whether the aggregate traffic from one domain to another conforms the SLA. Before entering the intermediate marker, a packet already has its DSCP. If the total amount of a certain service level packets exceeds the negotiated rate, the out of profile part will be remarked as a best-effort packet or other lower priority.

2.3.2 Bandwidth Broker

Bandwidth broker(BB) is a functional component in each of the domain that manages the resources in the domain and negotiate SLAs with neighboring domains. It is still an open topic whether BB should be implemented as a centralized or distributed component. If we use static SLA, then BB has relatively less work to do. However if we use dynamic SLA, then BB has to cooperate with neighboring BBs to update SLAs periodically or on demand.

2.4 Providing End-to-End QoS In Differentiated Services

Although the deployment of DiffServ paradigm is simpler and scalable than the Integrated Services (IntServ), it makes end-to-end quality of service guarantee more challenging. The problem lies in the fact that DiffServ operates on the per-hop behavior (PHB) without any end-to-end guarantees. Unlike RSVP [11], it does not perform an end-to-end resource reservation before a connection is set up.

For assured service (AF), most researchers agree that a static SLA would be more feasible. In order to keep a reasonable resource utilization, static SLA is usually negotiated based on statistical estimation. But without end-to-end resource reservation, a flow may experience deficient or surplus SLA on its way. A random early demotion and promotion marker is proposed in Chapter 3 to address this problem.

For premium service (EF), a dynamic SLA is desirable because the resources for EF is more expensive and EF usually need a better end-to-end quality guarantee. However, dynamic updating SLA may cause scalability problem for BBs. In Chapter 4, we propose pipe as a solution for the resource management of premium service.

CHAPTER 3 A RANDOM EARLY DEMOTION AND PROMOTION MARKER FOR ASSURED SERVICES

An Internet connection can span through a path involving one or more network domains. If we want to guarantee the end-to-end minimum throughput of the connection, we have to make sure that the aggregate traffic along the path does not exceed any of the interdomain negotiated service level agreements (e.g., the traffic rate) after this flow joins. This is very hard to ensure since the interdomain service agreement is not usually renegotiated at the initiation of each new connection. For assured service, the interdomain traffic rates are usually negotiated statically or updated periodically to avoid signaling overhead and scalability problem [22]. The negotiation is usually based on statistical estimation. So, the instantaneous aggregate flow rate may be higher or lower than the negotiated rate. In case of higher incoming flow rate, the intermediate marker demotes some of the "In" packets to "Out" so that aggregate rate of "In" packets conform to the negotiated rate. The demotion, although exercised at an aggregate flow level should affect all the connections proportional to their current usage (i.e., fair demotion). On the other hand, if the incoming flow is lower, ideally the intermediate marker should reallocate the excess capacity and promote a "previously-demoted" packet. This promotion should be fair across all connections as well.

In this chapter we propose a new technique for the marking process at the edge routers. The proposed process is motivated by the observation that some of the "In" packets may get marked as "Out" at nodes where the aggregate incoming traffic

rate exceeds the available bandwidth. However, later in the connections' path, the available bandwidth may be enough to route these "Out" packets (that were originally "In"). Therefore, there is a need to identify these demoted packets instead of clubbing them together with the packets that were marked "Out" at the point of origination. Our technique addresses two important aspects of the marking process at the edge routers. First, in the case of demotion, it ensures that the proportion of packets demoted for each micro-flow is fair (with respect to their rates). Second, it proposes a mechanism to identify the demoted "In" packets and promotes them fairly across connections when a domain has excess capacity. The fairness is achieved by early and randomly making marking decisions on the packets. Identification of a previously demoted packet is ensured using the AF PHB specified packet markings. In order to support this, the marker uses a three color (Red, Yellow, and Green) marking process, where Yellow is used as an indicator for temporary demotion. We have experimented with the marker on the *ns* [2] simulator. Our results show the effectiveness of the technique for both TCP and UDP traffic. The marking scheme is very fair in demoting and promoting packets and provides better performance to the in-profile traffic compared to the traditional leaky bucket scheme and the RIO scheme.

The rest of the chapter is organized as follows. Section 3.1 discusses the demotion and possible promotion at the boundary of two domains. The benefit of providing promotion is also discussed in detail in this section. Section 3.2 proposes a REDP marker to fairly demote and promote packets at the domain boundary. Section 3.3 studies the performance of REDP marker using ns network simulator. Section 3.4 discusses how to further improve the benefit from promotion by supporting three drop precedences in the core routers. Parameter sensitivity of the proposed marker is discussed in Section 3.5.

3.1 Interdomain Marking

A packet in the Internet travels from a source to its destination by getting routed through one or more network domains. According to the architecture of DiffServ defined by the IETF, neighboring domains negotiate Service Level Agreement (SLA) with each other, which specifies how much traffic of each service level could be passed from one domain to the other domain. More technical details, such as the committed rate, maximum burst size, etc., are specified by the Traffic Conditioning Agreement (TCA). Traffic Conditioners (TC) are implemented at the edge routers to ensure that the aggregate traffic of any level should not exceed the traffic profile of the TCA. A simple example of TC is a leaky bucket marker used for RIO as shown in Figure 3.1. The TCA between the upstream domain and the downstream domain specifies that r bits/sec "In" traffic from the upstream domain could enter the downstream domain with a maximum burst size of b. The leaky bucket is fed with a constant rate of r bits/sec. When a packet arrives from the upstream domain, if the packet has been marked as "Out", TC simply forwards it as "Out". If the packet has been marked as "In", TC checks the leaky bucket to see whether there is enough tokens for this packet. If there is, the packet is forwarded as "In" and the packet size worth of tokens are deducted from the leaky bucket. Otherwise, the "In" packet is demoted to "Out" and forwarded.

A more sophisticated Two Rate Three Color Marker (TR-TCM) [16] based on a similar mechanism was proposed recently as a possible candidate for the three color AF. The common idea behind the marker is that when the aggregate traffic of certain service level exceeds the rate defined in the traffic profile, the packet is demoted to a lower service level. However, in this model if the traffic rate of the service level is lower than the rate defined in the traffic profile, lower service level packets are not promoted to higher level. The scheme does not promote a packet because of the problem in identifying the packets to promote. For example, assume that flow-1 has subscribed

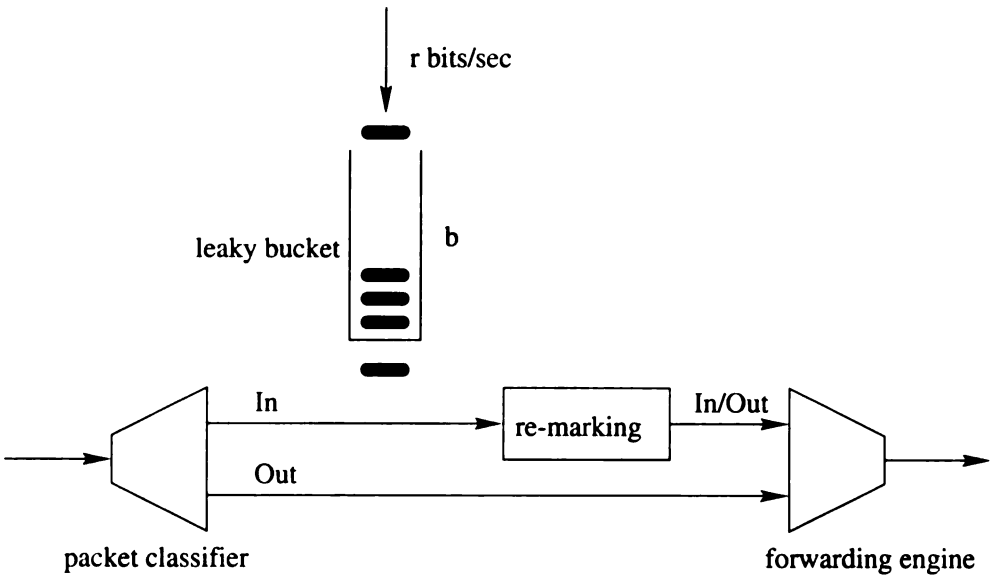


Figure 3.1 A leaky bucket intermediate marking model.

for certain throughput of assured service, and flow-2 has subscribed for best effort service. Both of them pass through several domains. Assume that some of the "In" packets of flow-1 are demoted while crossing the first domain. While crossing the second domain, if the TC has some extra "In" tokens and if promotions were allowed, the best effort traffic and the demoted traffic of flow-1 compete for getting promoted. In all fairness, the demoted packets of flow-1 should be promoted first. However, there are no identification marks in these demoted packets. By promoting the packets of both flows randomly, the assurance of the in-profile packets cannot be improved. The simulation result in Section 4.2 supports this argument.

Usually, an end-to-end connection would cross multiple DiffServ domains. For assured service, static TCA based on statistical estimation is preferred for simplicity and ease of pricing. Since there is no end-to-end signaling and negotiation, demotion is unavoidable. If we use the marking model as has been proposed in the literature, once an "In" packet is demoted, it will be treated as an "Out" packet for all of the remaining domains. Assuming the number of domains along the end-to-end connection is n, and the probability that a packet gets demoted through each domain boundary is p, the

end-to-end demotion probability of a packet would be $1-(1-p)^n$. However, some of the demotion decisions could be reverted if we can identify the demoted packets and promote them as soon as we have excess resources available in the downstream domain. Based on these motivations, we propose a three color demotion-promotion scheme in the following section.

3.2 REDP Marker

In this section, we propose a new technique called Random Early Demotion and Promotion (REDP) for managing the interdomain flow control. Main aspects of the REDP scheme are the provision of promotion of the demoted packets and making the demotion/promotion process fair. The provision of promotion enhances the performance of the assured traffic, whereas the early randomness in packet marking decision ensures the fairness of the proposed scheme. These performance measures are quantified and justified in the next section. Here we describe the marking process and the framework of the REDP scheme. Notice that the initial marking of packets at the host markers can be done on a per-flow basis. However, the intermediate marking must be done on the aggregate level for the ease of scalability.

We use a variation of the tricolor marking model for the REDP scheme. Therefore, each packet can be marked as *Green*, *Yellow* or *Red*. Suppose an end user submits an expected rate r. Initially, the local domain configures a leaf marker for the flow. A packet from this flow is marked as *Green* if it is in-profile and *Red* if it is out-of-profile. None of the packets is marked as *Yellow*. Intermediate markers are implemented in the TC of domain boundaries. While crossing a domain boundary, a *Green* packet is demoted to *Yellow* if the aggregate *Green* packet rate[1] exceeds the negotiated rate at the intermediate marker. A *Yellow* packet is promoted to *Green* if the aggregate

[1]Many micro-flows may pass from the upstream domain to the downstream domain through the intermediate marker. Aggregate *Green* packet rate is the sum of the rate of all of the *Green* packets of these micro-flows.

Green packet rate is lower than the negotiated rate. A *Yellow* packet is never demoted to *Red* and a *Red* packet is never promoted to *Yellow*. Thus, *Yellow* is specifically used to memorize the demoted *Green* packets. When we are able to promote, we only try to promote the *Yellow* packets. In other words, we would only promote the assured packets that were demoted. The motivation for reserving the *Yellow* packets to remember the previous state of a high priority packet came from the fact that different traffic classes, not the three color scheme per traffic class, gives effective isolation between TCP and UDP flows [20]. Two colors per class is enough for service differentiation within a class [20]. The third color can be used more effectively to record the demoted packets. The state diagram of the demotion-promotion algorithm is shown in Figure 3.2.

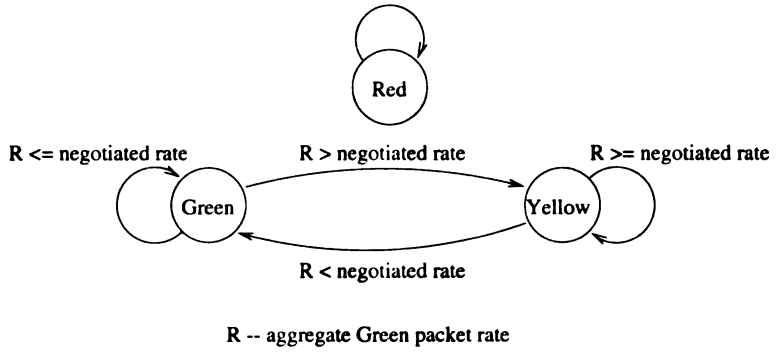


Figure 3.2 State diagram of demotion and promotion within three colors.

The leaky bucket is a deterministic flow control network element that can be used as a traffic marker. Like the drop-tail queue, a simple leaky bucket demotes all packets that arrive when there are no tokens available. As argued in [18], much of the Internet traffic is highly periodic, either because of periodic sources (e.g., real-time audio or video) or because window flow control protocols have a periodic cycle equal to the connection round trip time (e.g., a network-bandwidth limited TCP bulk data transfer). This phase effect could bring unfairness in the demotion and promotion among different micro-flows as addressed in [17]. The following (concocted) example

explains the unfairness of the phase effect (or synchronization). Suppose all packets of two streams are originally marked as *Green*. They have the same rate and same packet size and are aggregated in the marker. Suppose the packet from the streams (1 and 2) are interleaved in the following pattern, 1212121212..., and the marker has to demote 50% of the packets from the aggregate flow, i.e., every other packet must be demoted. Then all the packets from one flow will be demoted while all the packets from the other flow will remain unaffected. Phase effect could also bring about unfairness in promotion. Detail discussions on the phase effect have been reported in [18]. Introducing randomness in the packet selection process of the flow control mechanism could solve the problem. An example is the Random Early Detection (RED) gateway [10] that reduces the unfairness of the drop-tail queue. We apply a similar concept to the leaky bucket marker by introducing randomness and early decisions on the packet marking process. In addition, as discussed earlier, we allow the promotion of the *Yellow* packets based on bandwidth availability. We call this marker the Random Early Demotion and Promotion (REDP) marker.

An REDP marker is implemented using a leaky bucket. A promotion threshold is set in the leaky bucket. If the tokens in the leaky bucket exceed the promotion threshold and an arriving packet is *Yellow*, it is promoted to *Green*. Similarly a demotion threshold is used in the leaky bucket. If the number of tokens in the leaky bucket is less than the demotion threshold, an arriving *Green* packet is demoted to *Yellow*. Using this scheme, we can also detect whether the aggregate rate of the arrival of *Green* packets is lower or higher than the negotiated rate.

The marking model is shown in Figure 3.3. Two thresholds, T_L and T_H divide the leaky bucket into three regions, *demotion region*, *balanced region*, and *promotion region*. Initially, the token count is set within the *balanced region*. Three situations can arise during the marking process:

1. *Balanced:* If the arriving rate of *Green* packets is equal to the token filling rate r, the token consumption rate is same as the token filling rate. Therefore, the

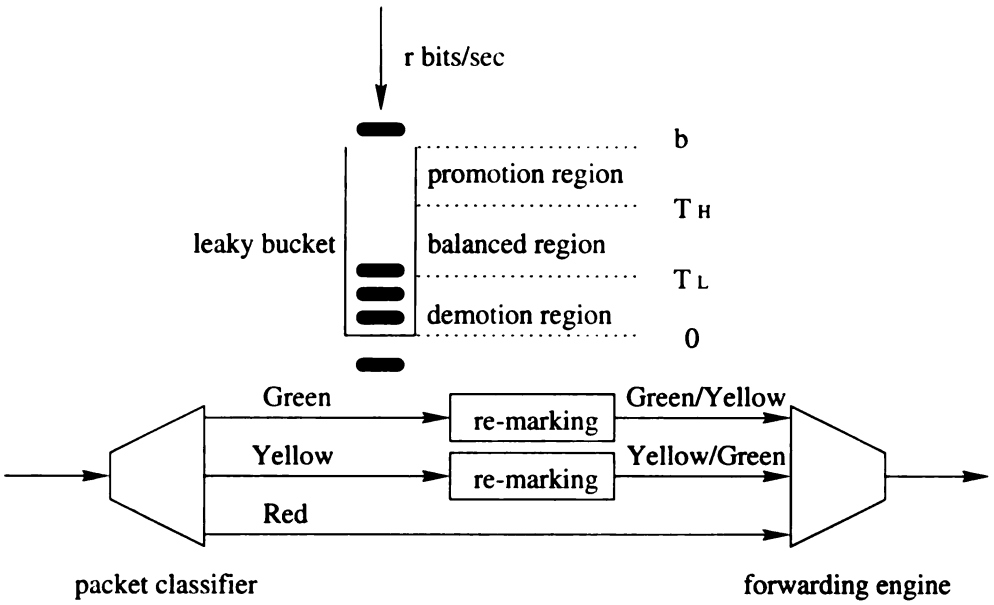


Figure 3.3 REDP Marker.

number of token in the bucket remains in the *balanced region.* Each packet is forwarded without changing the color.

2. *Demotion:* If the arriving rate of *Green* packets exceeds r, the token consumption rate exceeds the token filling rate. The number of tokens decreases and the token level falls into the *demotion region.* In the *demotion region*, each arriving *Green* packet is randomly demoted to *Yellow* with a probability of P_{demo}, where P_{demo} is a function of the token count (TK_{num}). A simple linear example of the function could be:

$$P_{demo} = (T_L - TK_{num})MAX_{demo}/T_L,$$

where MAX_{demo} is the maximum demotion rate. When the leaky bucket runs out of token, each arriving *Green* packet is demoted to *Yellow.*

3. *Promotion:* If the arriving rate of *Green* packets is less than r, the token filling rate exceeds the token consumption rate. The number of tokens increases and

the level reaches the *promotion region*. In the *promotion region*, each arriving *Green* packet will still be forwarded as *Green*, consuming a certain number of tokens. Each arriving *Yellow* packet will be randomly promoted to *Green* with a probability of P_{promo}, where P_{promo} is a function of the token count in the leaky bucket (TK_{num}). A linear example of the function is:

$$P_{promo} = (TK_{num} - T_H)MAX_{promo}/(b - T_H).$$

The REDP scheme removes the phase effect of periodical flows by detecting the arriving rate of the *Green* packets early and by promoting or demoting packets randomly. During demotion, it keeps the number of demoted packets of each flow approximately proportional to the number of *Green* packets of that flow. Similarly during promotion, it keeps the number of promoted packets of each flow approximately proportional to the number of *Yellow* packets of that flow.

The DiffServ core routers could support either two or three drop precedences. If it supports two drop precedences (e.g. RIO), *Green* is deemed as "In". Both *Yellow* and *Red* are deemed as "Out". If it supports three drop precedences, *Green* has the lowest drop probability and *Red* has the highest drop probability.

3.3 Performance Study

In this section, we analyze the performance and effectiveness of the REDP scheme. In the previous section, we claimed that the REDP has two major advantages. First, the demotion and promotion performed by REDP is fair across the connections. Second, by allowing the promotion of demoted packets, REDP improves the performance of the assured traffic. We quantify these two measures in this section through experiments using the *ns* simulator. Both UDP and TCP sources are analyzed to show the performance improvement.

3.3.1 Fairness of Demotion and Promotion

Note that the demotion and promotion algorithm employed in the REDP marker uses the same mechanism (i.e., random and early decisions) to ensure fairness. Here, to avoid repetition we only show the fairness of demotion. Figure 3.4 depicts the simulation topology used to study the fairness of demotion. Host H1, H2, H3, H4 each has a leaf marker implemented inside. Each of the hosts has a 0.5 Mbps assured service profile. So initially each host could have up to 0.5 Mbps packets marked as *Green*. The remaining packets are marked as *Red*. Each flow originates from a host and passes through multiple domains and terminates at CR2. After successfully crossing one[2] or several domains the packets reach the edge router ER. ER is at the boundary of the two domains. Suppose at ER, we don't have enough SLA to pass all the *Green* packets to the downstream domain. Then some of the *Green* packets need to be demoted to *Yellow*. The goal of our experiment is to evaluate the fairness of different marking schemes. In other words, we investigate if equal proportion of the *Green* packets of each flow would be demoted by the different marking schemes. In the following discussions, we analyze and compare the fairness of the following three schemes using our simulation: Leaky Bucket, REDP, and per flow marking[3].

All the marking schemes are implemented in the ER of Figure 3.4. The token filling rate of the leaky bucket is α Mbps, where $\alpha < 2$ Mbps. After being re-marked by the marker, a packet is forwarded to the core router CR1 and then terminates at core router CR2. The link capacity between ER and CR1 is larger than the aggregate bandwidth of the four flows. The link capacity between CR1 and CR2 is exactly α

[2]In the real world, it is unlikely that a *Green* packet will get demoted when it reaches the first intermediate marker because the leaf markers are configured based on the capacity of the first intermediate marker. In this experiment, in order to simplify the simulation topology, we assume that the demotion happens when packets reach the first intermediate marker, that is, when it reaches edge router ER in Figure 3.4.

[3]Per flow marking is implemented in the following way: Assume that all the intermediate markers know the original submitted rate of each flow. Tokens assigned to each flow are proportional to its original submitted rate. This model, although should be the fairest among these three, needs per flow monitoring and signaling. It may not be practical as an intermediate marker because of the scalability problem. Here we only use it as an ideal case to evaluate the fairness of REDP marker.

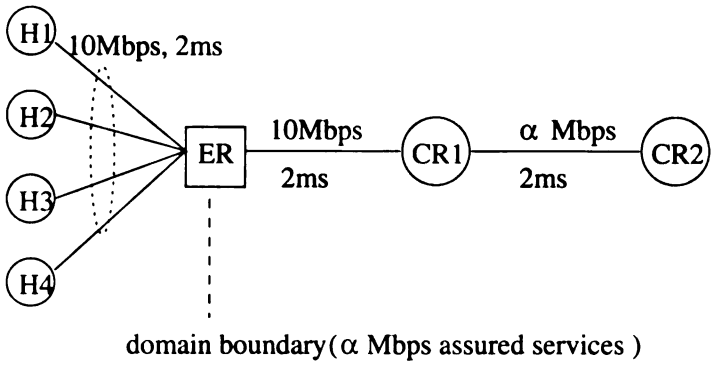


Figure 3.4 Simulation topology used to study the fairness of demotion.

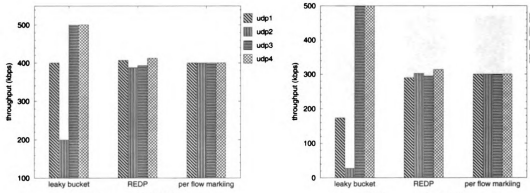
Mbps. Assured service is implemented in CR1 through the RIO scheme. In the core routers, all the *Green* packets are treated as "In", both *Red* and *Yellow* packets are treated as "Out". This configuration ensures that the aggregate *Green* packets from CR1 to CR2 is exactly the link capacity. So almost all the *Green* packets would be forwarded and almost all the *Yellow* and *Red* packets would be dropped. By computing and analyzing the throughput of each flow at CR2, we derive the fairness of the demotion for different markers. Theoretically, if the demotion is fair, each flow should get approximately the same throughput, that is, $\alpha/4$. We have used both UDP and TCP sources in our simulation to demonstrate the effectiveness of REDP for these two popular transport-level protocols.

Fairness of demotion for UDP sources

In this simulation, we have assumed four UDP sources, udp1, udp2, udp3, udp4 starting from hosts H1, H2, H3, and H4, respectively. The sending rate of each flow is 0.6 Mbps. Originally, 0.5 Mbps is marked as *Green* and the remaining 0.1 Mbps is marked as *Red*. In the first simulation, we choose $\alpha = 1.6$ Mbps. So at edge router ER, 2 Mbps *Green* packets arrive but only 1.6 Mbps of them could be marked as *Green* before entering the downstream domain. If the marker implemented in ER is ideally fair, each flow should have 400 kbps packets forwarded as *Green* and

100 kbps packets demoted as *Yellow*. Because the bandwidth of the bottleneck link, from CR1 to CR2 is exactly 1.6 Mbps, only the *Green* packets could pass this link. All the *Yellow* and *Red* packets will be dropped here. So ideally, each flow should get 400 kbps throughput. The simulation is executed for 50 secs and we use the last 40 secs to calculate the throughput of each flow. Calculation of the throughput is done in the same way for all the other results in this chapter.

The throughput for different flows using different makers is shown in Figure 3.5(a). If we use a leaky bucket marker in ER, the throughput of the four flows are highly biased. Flow2 only get about 200 kbps while flow3 and flow4 get about 500 kbps each. This is because of the synchronization problem. The four UDP flows have the same rate and are sending data periodically. Most of the time when a *Green* packet of flow2 comes, the leaky bucket happens to run out of tokens. If we use a per-flow based marker in ER, each flow gets close to 400 kbps. If we use our REDP marker in ER, each flow also gets approximately 400 kbps.



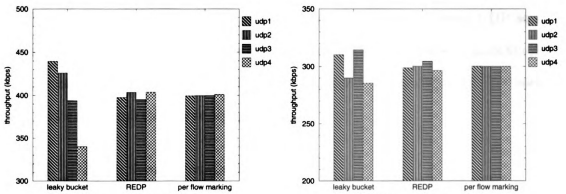
(a) $\alpha = 1.6Mbps$ (b) $\alpha = 1.2$ Mbps

Figure 3.5 Demotion fairness comparison: UDP sources, four flows have same sending rate.

Figure 3.5(b) shows another set of result with a different demotion ratio. Here, $\alpha = 1.2$ Mbps, so ideally 300 kbps of each flow could be passed as *Green* and 200 kbps should be demoted as *Yellow*. From the result we can observe that the throughputs

are highly biased if we use leaky bucket marker in the ER. However, the REDP scheme removes the synchronization or phase effect and is very fair as is demonstrated by comparing its results with the ideal per-flow marking process.

Depending on the sending rate of each flow, the phase effect could be less or more serious. Next, we change the sending rate of each flow to 0.79 Mbps, 0.73 Mbps, 0.53 Mbps, and 0.61 Mbps respectively and repeated the simulation. Note that from each flow, 0.5 Mbps is marked as *Green* and the rest is marked as *Red*. Figure 3.6 shows the results. Figure 3.6(a) and Figure 3.6(b) shows that the throughputs of the four flows are still biased in the case of the leaky bucket marker. However, the degree of variance is higher. In both the cases, the REDP scheme achieves better fairness over the leaky bucket marker. The fairness of REDP is almost as good as the per-flow marking while it works on the aggregate flow level and thus does not have any scalability problem.



(a) $\alpha = 1.6$ Mbps (b) $\alpha = 1.2$ Mbps

Figure 3.6 Demotion fairness comparison: UDP sources, four flows have different sending rates.

Phase effect is very common for UDP sources. We have done several simulations and have observed this effect frequently. Depending on the rate of each flow, the SLA, and the packet size, etc., the effect could vary. By using REDP marker, we incorporate a random component in the path. This randomness removes the deterministic phase effect so that it does fair demotion and promotion.

Fairness of demotion for TCP sources

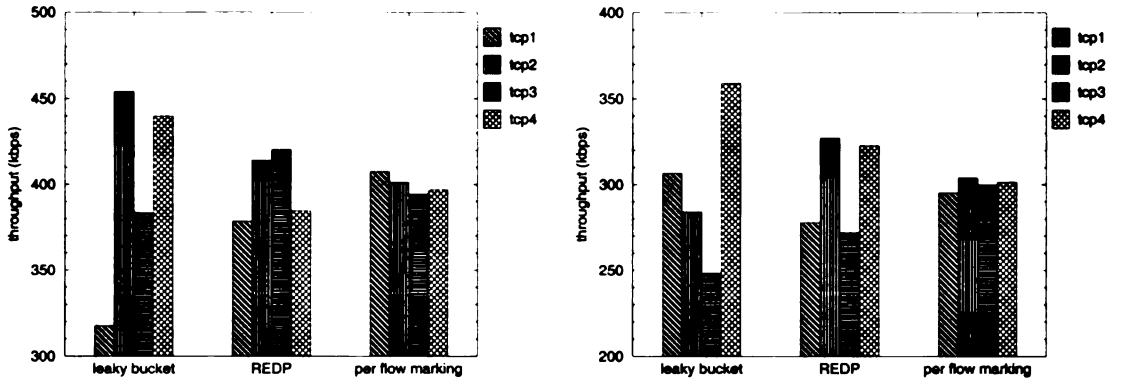
According to the analysis in [18], TCP sources also have phase effect because of its sliding window flow control algorithm. A TCP source will not send the next burst of packets until it receives the ACK of the current burst of packets. So, the period is the round trip time (RTT) of the connection. In [18], the authors have shown that flows with similar RTT could get biased throughputs if they share a common link.

The topology of Figure 3.4 is again used for this simulation, where the four UDP sources are changed to four TCP sources. The delay of the link between H1 and ER is changed to 1ms, between H3 and ER is changed to 1ms, and between CR1 and CR2 is changed to 10ms. So the RTT of each flow is 26ms, 28ms, 26ms, 28ms, respectively. The throughput of each flow is shown in Figure 3.7. From the figure we could observe the phase effect when we use the leaky bucket marker. Both per flow marking and REDP could increase the fairness of demotion. However, the fairness improvement of REDP marker over leaky bucket marker is not as obvious as using UDP sources. This is because TCP has its own flow control and congestion control algorithm [19] [4]. Adding random component along the path could improve the fairness of TCP sources, but would not completely solve the problem.

3.3.2 Benefit from Promotion

Depending on the actual network traffic, a packet demoted at the boundary of a domain may or may not get dropped in that domain. If it is not dropped in that domain, it is preferable to promote it as soon as there are excess tokens in any of the downstream edge markers. This ensures that a packet does not get dropped under minor and transient congestions in the downstream domains. The proposed REDP marker could do both demotion and promotion.

[4] We believe if the TCP sources are modified according to [19], a "better" fairness can be obtained through REDP. We did not carry out the experiments since the source does not remain traditional TCP compliant.



(a) $\alpha = 1.6$ Mbps (b) $\alpha = 1.2$ Mbps

Figure 3.7 Demotion fairness comparison: TCP sources.

The topology shown in Figure 3.8 is simulated to study the benefit of promotion. ER1 and ER2 are two edge routers, each of which has a marker implemented in it. CR1, CR2, and CR3 are core routers with built-in RIO mechanism for flow control. Similar to the previous simulation, each of H1, H2, H3, H4 has a flow starting from it and sinking at CR3. Each flow crosses two domain boundaries. At the first domain boundary defined by ER1, there is not enough SLA to forward all the *Green* packets as *Green*. Some of the *Green* packets are demoted. Let us assume that at the second domain boundary, ER2, there is some excess SLA. So we have three choices:

1. *No promotion:* We only use two colors, *Green* and *Red* (or "In" and "Out"). In case of deficient SLA, *Green* is demoted to *Red*. In case of excess SLA, *Red* is not promoted to *Green*.

2. *Two color promotion:* We only use two colors. In case of deficient SLA, *Green* is demoted to *Red*. In case of excess SLA, *Red* is promoted to *Green*. In this case, there is no distinction between a packet that is originally marked as *Red* and a demoted packet that is also marked *Red*.

3. *Three color promotion:* We use three colors. In case of deficient SLA, *Green* is demoted to *Yellow*. In case of excess SLA, *Yellow* is promoted to *Green*. No

demotion or promotion is done between *Yellow* and *Red*. In the core routers, *Green* is treated as "In", both *Yellow* and *Red* are treated as "Out".

We implement all these three alternatives and compare the performance in the following experiments.

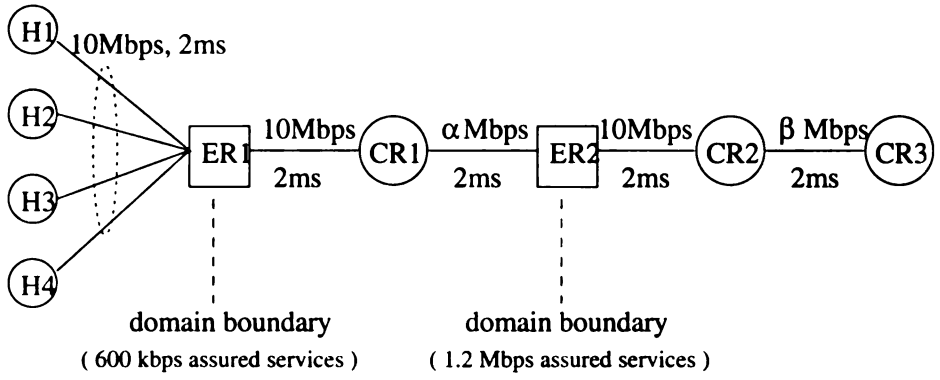


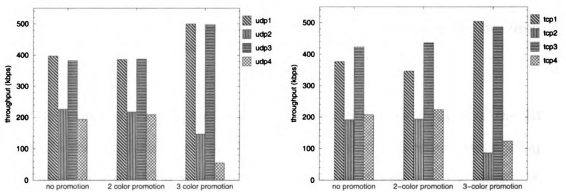
Figure 3.8 Simulation topology used to study the benefit from allowing promotion.

UDP sources

For the four hosts, assume that H1 and H3 each negotiate 500 kbps assured service, and H2 and H4 use best-effort service. Four UDP flows, udp1, udp2, udp3, udp4 start from H1, H2, H3, H4, respectively and sink at CR3. The rate of each flow is set at 500 kbps. So initially udp1 and udp3 each has 500 kbps packets marked as *Green*, udp2 and udp4 each has 500 kbps packets marked as *Red*. At ER1, up to 600 kbps *Green* packets are allowed to be forwarded to the next domain. So 40% of the *Green* packets are demoted here. We set α = 2 Mbps. So no congestion happens in this domain and the demoted packets will not be dropped in this domain. At ER2, up to 1.2 Mbps *Green* packets are allowed to be forwarded to the next domain. If we choose promotion, we could promote some of the *Yellow* packets to *Green*. We set β = 1.2 Mbps. So within this domain, some of the *Red* or *Yellow* packets will be

dropped. The link between CR2 and CR3 is the bottleneck. Figure 3.9(a) shows the throughput of each flow under different marking schemes.

Without promotion, a demoted packet is treated as "Out" for all of the remaining domains. So some of the packets will be dropped at the bottleneck link. Each of udp1 and udp3 gets about 400 kbps throughput although they submitted 500 kbps. If we use the 2-color promotion as described above, we can promote some of the *Red* packets at ER2. However, since we cannot tell which one is initially marked as *Red* and which one is demoted to *Red*, both of them could be promoted to *Green*, which would not improve the throughput assurance of udp1 and udp3. The simulation results support this point. We cannot improve the throughput assurance of flow1 and flow3 through 2-color promotion. In the 3-color promotion, we use *Yellow* to memorize the demoted *Green* packets. In ER2, only the *Yellow* packets are promoted to *Green*. So we could improve the bandwidth assurance of udp1 and udp3. Each of them gets a throughput of about 500 kbps. This is where the REDP scheme benefits the most.



(a) UDP sources (b) TCP sources

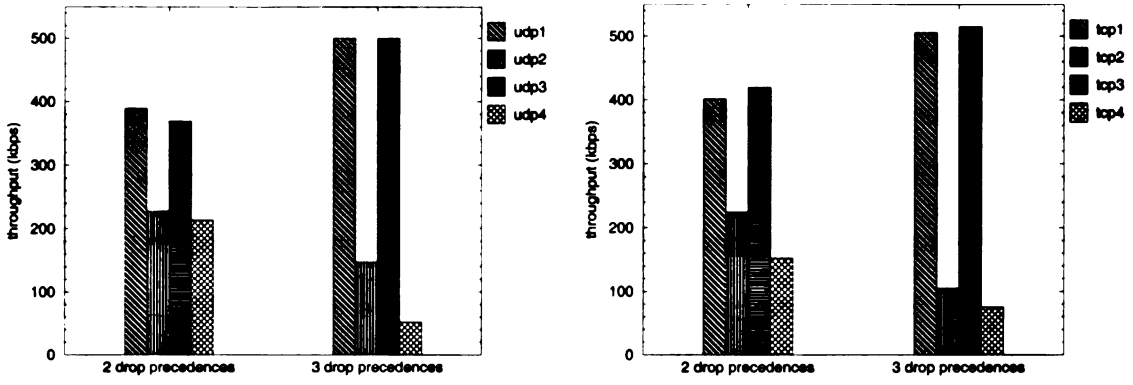
Figure 3.9 The benefit from promotion.

For TCP sources.

Now we change the four UDP sources to four TCP sources, keeping all of the other parameters unchanged. The simulation result is shown in Figure 3.9(b). The result is

similar to the previous simulation. No promotion and 2-color promotion have similar performance while the 3-color promotion improves the throughput assurance of tcp1 and tcp3. Thus the concept of promotion used in the REDP scheme benefits both TCP and UDP traffics.

3.4 Supporting Three Drop Preferences to Improve Assurance



(a) UDP sources (b) TCP sources

Figure 3.10 Benefit from promotion: three drop precedences vs. two drop precedences.

In the previous section, we analyzed the benefits of promotion in the REDP scheme. In the simulation, we chose $\alpha = 2$ Mbps so that congestion does not happen in the first domain. What will happen if instead we choose $\alpha = 1.2$ Mbps? Would we still get the same throughput assurance for flow1 and flow3? The answer is "no". Since in the core router, we only support two drop precedences, the *Yellow* packets are treated the same as the *Red* packets in the core router. If congestion happens in the first domain, some of the *Yellow* packets will be dropped before they reach ER2. Promotion will not help to improve the throughput assurances of flow1 and flow3. The simulation result shown in Figure 3.10 supports this answer. It is desirable to assign a lower drop probability to the *Yellow* packets compared to the *Red* packets,

so that the *Yellow* packets will be protected during network congestion. Thus, the core router needs to support three drop precedences. *Green* packets have the lowest drop probability and the *Red* packets have the highest. Figure 3.10 shows the assurance gain by adding one more drop preference in the core routers. For both UDP and TCP sources, if we set $\alpha = \beta = 1.2$ Mbps, three drop precedences in the core router could greatly improve the throughput assurance of flow1 and flow3. Each of them get a throughput of about 500 kbps as shown in the figure.

3.5 Parameter Sensitivity

For a leaky bucket marker, the only variable parameter is the size of the leaky bucket, b, which is also the maximum burst size. However, a REDP marker has additional parameters that determine the demotion and promotion processes. In this section, we briefly discuss how to select these parameters and their impact on the performance of REDP.

T_L and MAX_{demo} are two parameters which determine the fairness of the demotion process. If $T_L = 0$, the demotion is same as the demotion of a leaky bucket marker. In order to ensure enough randomness for the demotion process, T_L need to be large enough. However, increasing T_L may result in a larger b, which will increase the maximum burst size of the output traffic. So we should select an appropriate T_L so that we can have both good fairness and acceptable burst size.

The range of MAX_{demo} is between 0 and 1. If $MAX_{demo} = 0$, a *Green* packet will not get demoted until the bucket runs out of token. So the demotion process is the same as the demotion in case of a leaky bucket marker. Given the fact that T_L could not be set very large, selecting a large enough MAX_{demo} could improve the utilization of the *demotion region*, thereby improving the randomness of demotion.

Figure 3.11 shows the fairness of demotion under different (T_L, MAX_{demo}) selections, where the unit of T_L is in packets. We have used the same topology shown

in Figure 3.4 for the simulation, but with a varying series of (T_L, MAX_{demo}). The overall demotion ratio is set to 40%. Ideally, each UDP flow should have 200 kbps *Green* packets demoted and 300 kbps forwarded. It can be observed that when $MAX_{demo} = 0.5, 1.0$, the demotion fairness increases with the increase in T_L. The results are unfair for all values of T_L when $MAX_{demo} = 0.2$. The following inferences could be derived from Figure 3.11:

- The degree of fairness increases with the increase in MAX_{demo}. The fairness improves with the increase in T_L.
- For low values of MAX_{demo}, the results are unfair irrespective of the variations in T_L.

Based on these observations, we suggest that MAX_{demo} and T_L should be set reasonably high. The process of promotion is symmetrical to that of demotion. So we could choose $b - T_H = T_L$, $MAX_{promo} = MAX_{demo}$.

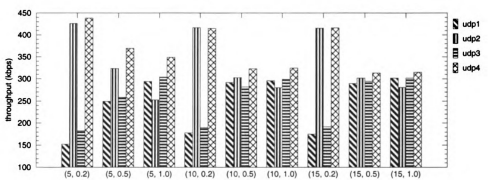


Figure 3.11 Demotion fairness under different (T_L, MAX_{demo}).

($T_H - T_L$) determines the size of the *balanced region*, which also should be selected large enough. If the *balanced region* is too small, the leaky bucket may oscillate between the *demotion region* and the *promotion region*. This may cause unnecessary demotion and promotion. Again, large ($T_H - T_L$) may increase the bucket size b, and will also delay the demotion and promotion processes. It is hard to configure a simple

simulation to determine the best value for $(T_H - T_L)$. However, from our simulation experiences, we find $(T_H - T_L) = 10$ performs pretty good. These discussions may provide broad guidelines for parameters selection.

CHAPTER 4 PIPE: A SCALABLE RESOURCE MANAGEMENT APPROACH FOR PREMIUM SERVICES

Most of the current research on DiffServ are focused on service specification, service architecture and components definition[30][31][26]. Few of them [24] discuss how resource management, especially dynamic SLA, could be implemented in the DiffServ environment. Without good resource management schemes, DiffServ itself would not provide any quality of service (QoS) assurances or guarantees. Usually, assured service only need a soft QoS guarantee. We agree that the SLA could be negotiated statically based on statistical estimation or through a learning process. Premium service, however, are more critical and may demand QoS guarantees. So the SLA negotiation should be more precise. As proposed in [32], a dynamic signaling process could negotiate the exact amount of SLA for the premium service traffic. In this scheme, when a new flow need to enter the domain, the BB will receive a signaling message. It will then check the resource database to see whether it can update the corresponding SLA. This will cause even worse scalability problem compared to RSVP [11] because in RSVP the signaling and reservation is distributed among all routers within the domain. In order to solve the scalability problem for BB, the SLAs for premium service should not be updated upon the join/leave of each connection. A comprising approach can be used by aggregating the SLA updating requests and periodically signaling the BB for SLA updating. We can limit the amount of updating

requests sent to BB. Since we update the SLA periodically, the resource utilization should be better than that of the static SLA scheme.

In this chapter, we introduce the concept of "pipe" as a solution for resource management within a DiffServ domain. A pipe is a destination-aware SLA from the ingress router to a specific egress router. It specifies the amount of premium traffic that could flow from the ingress router to the egress router, hence forms a virtual channel with a certain bandwidth between an ingress and an egress router. When a new connection joins/leaves the pipe, it only signals the ingress router. BB only get involved when the pipe capacity needs to be updated. Thus, we offload and distribute parts of the resource management work to edge routers so that the BB can handle the scalability problem. We have reported the implementation details of pipe and have used voice traffic to evaluate its performance. The result shows that the use of pipe could greatly reduce the signaling overhead on BBs. Also, the utilization is comparable to the per-flow-based signaling schemes. In our simulations using IP telephony traffic, when the average number of calls in the pipe ranges from 100 to 1000, the updates would be only 10^{-2} to 10^{-4} times of the completely dynamic SLA scheme, while the utilization could still be maintained as high as 80 percent. Thus, without sacrificing any significant utilization we can greatly reduce the updating overheads.

The rest of the chapter is organized as follows. Section 4.1 discusses the concept of pipe. Section 4.2 studies different implementation schemes of pipe. Section 4.3 discusses how to improve pipe utilization through updating.

4.1 Concept of Pipe

In DiffServ model, a small fraction of the resources are allocated for premium service. Typically, real-time applications (e.g. voice over Internet, video over Internet)

are going to be the users of this service. These applications mandates end-to-end resource reservation for proper operation.

In this chapter we propose the use of "pipes" to establish end-to-end QoS guarantee in the DiffServ environment. The concept of pipe, as described earlier, can be used to guarantee ingress-to-egress bandwidth availability without the overheads of per-flow state maintenance. We have demonstrated the implementation and effectiveness of the usage of "pipe" for IP telephony applications. IP telephony is used as a representative of real-time applications requiring premium service in the DiffServ Internet model.

4.1.1 Definition of a Pipe

A pipe is defined as a logical path between two end points on the network, having a pre-defined capacity. The choice of end points depends on many factors:

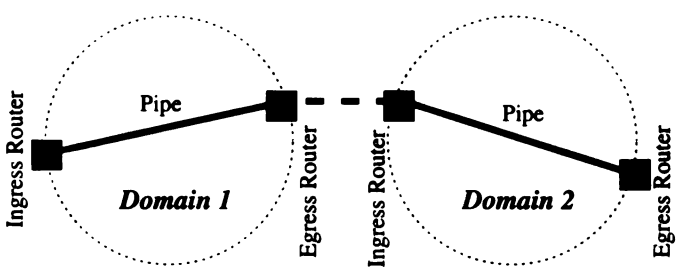


Figure 4.1 Pipes in DiffServ domains.

Service providers' point of presence: Technically, pipe could be within an ISP domain or extend through multiple domains as shown in Figure 4.1. However, it is preferable to have pipe within one ISP domain for convenience of management. Usually a pipe starts from an ingress router and end at an egress router.

Traffic between regions: The purpose of pipe is to reduce the signaling and computation overhead of BB. Usually, for the same type of traffic (eg. voice traffic),

the aggregate traffic gets smoother as the total amount of traffic increases. Depending on the average traffic amount between an ingress router and an egress router pair, it may or may not be necessary to construct a pipe between this pair. If the total traffic between the two points exceed a certain threshold, we can construct a pipe. Since the aggregate traffic is relatively smooth, pipe capacity need not to be updated frequently. So the BB need not get involved frequently, thus minimizing its load and thereby enhancing scalability. If a connection need to cross an ingress-egress router pair without a pipe, it has to use the dynamic signaling process through the BBs.

Dynamism in the traffic pattern: The choice of end points may also depends on the dynamism of the traffic pattern. The smoother the aggregate traffic between the end points, the higher is the benefit obtained by constructing a pipe between the two points.

4.1.2 Relationship between Pipe and SLA

Pipe is just a destination-aware SLA. According to the definition in [22], the scope of SLA could be one of the three situation:

1. all traffic from ingress point A to any egress point
2. all traffic between ingress point A and egress point B
3. all traffic from ingress point A to a set of egress points

Pipe belongs to the second category. All of the pipes' configurations within the domain are stored in the domain's BB. BB could also store the topology of the domain and the link capacity between each pair of nodes. When a pipe needs to update its capacity, it talks to the BB and the BB decides whether the request should be granted or rejected. If the request is granted, the BB will notify the ingress router to reconfigure its traffic conditioner (TC), and thereby the SLA is updated.

4.1.3 Using Pipes to Construct an End-to-end QoS Guaranteed Connection

When an end host need to make a connection with another end host using premium service, it contacts all the pipes it need to use in its path one by one. If the connection are admitted by all of the pipes, it will be set up successfully. Note that it only needs to signal those ingress routers. If one or more of the pipe do not have enough capacity to admitted the new connection, the ingress router could have two choices:

1. reject the connection
2. contact the corresponding BB to increase the pipe capacity to admit this connection, which in turn could be accepted or declined.

By concatenating multiple pipes, an end-to-end QoS guaranteed connection can be established.

4.2 Implementation of Pipe

In this section we study how a pipe can be implemented within the constraints of current Internet architecture and the DiffServ paradigm. Since expedited forwarding should not be delayed (or experience bounded small delay) per-hop, it is desirable to have it implemented on a priority queuing environment.

Unlike the virtual channel in ATM, a pipe in the DiffServ is just a destination-aware SLA. It is configured in the markers and policers in the edge routers only for admission control purpose. Once a packet enters the domain core, we cannot tell which pipe it comes from. The BB of the domain stores the pipe information. So BB knows whether the domain is able to increase a pipe capacity or construct a new pipe. There is no per-flow or per-pipe state information stored in the domain core routers. Depending on the topology of the domain, multiple pipe may share some

common paths. The forwarding behavior in the core routers is only determined by the DiffServ Code Point (DSCP) [26] of the packet, which retains the per-hop behavior in the core. The service quality of the premium service is ensured through the following methods:

1. premium service packets are queued separately and treated preferentially. The premium service queue could be implemented as a high priority queue over the RIO queue [31]. It could also be implemented as a WFQ with the RIO queue and the premium queue could be given higher weight compared to its actual traffic.

2. Pipe would ensure that the aggregate traffic through it would not exceed its capacity. Since each pipe is destination oriented, BB could make sure that for each link, the aggregate traffic of all the pipes through it would not exceed a certain percentage of the link capacity. (Usually, premium service would not occupy more than 20 percent of the total capacity of a link.) So even there is no per-pipe level service guarantee in the core routers, the QoS of each pipe could still be ensured.

We set up the following experiment to show the service quality of the aggregate scheme under different queuing disciplines. We use the *ns* [2] simulator to implement different queuing disciplines which include priority queuing (PQ) and different WFQs. Figure 4.2 shows the network topology that is simulated. Four nodes: n0, n1, n2, n3 are part of the network core. The link bandwidth of n0-n1, n1-n2, n2-n3 are 10Mbps, 5Mbps, and 10Mbps, respectively. A pipe with capacity of 10 voice streams exists between n0 and n3. A pipe with capacity of 15 voice streams also crosses link n0-n1. Similarly, a pipe with capacity of 3 voice streams and a pipe with capacity of 15 voice streams cross link n1-n2, n2-n3, respectively. These three pipes (n0-n1, n1-n2, and n2-n3) are used as cross traffic. All of the voice traffic use the premium service

through pipes. The remaining capacity of the links are used by the best-effort TCP traffic. The premium service queue are implemented in all of the core routers n0, n1, n2, and n3.

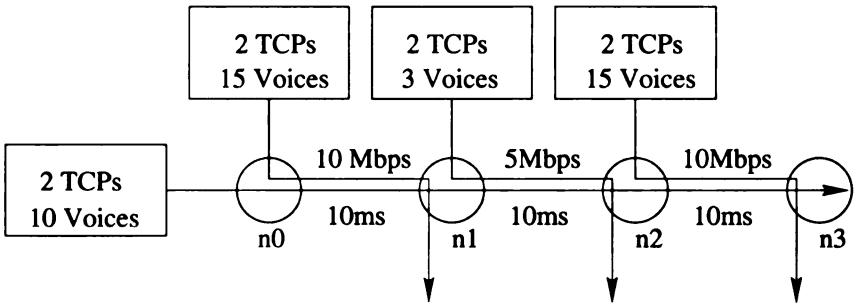


Figure 4.2 Simulation topology.

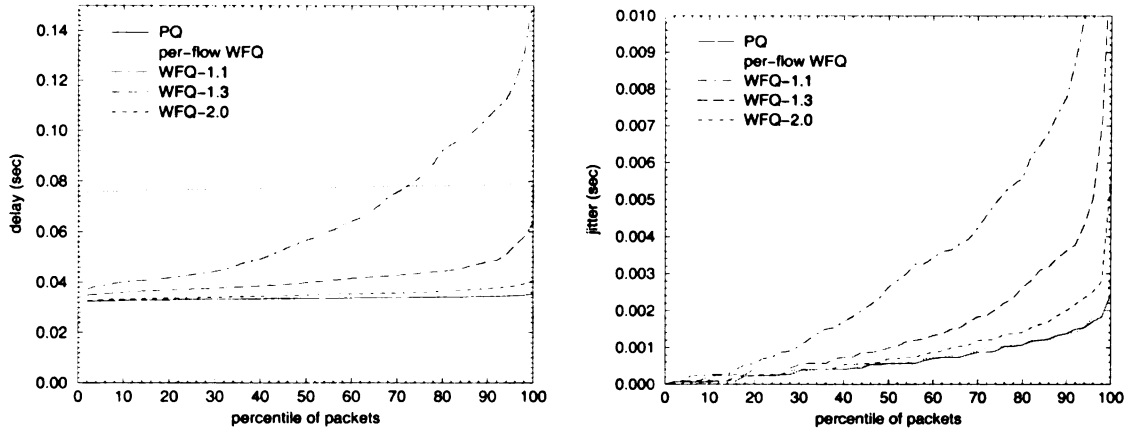
We pick up one voice stream from the pipe between n0 and n3 to study the voice packet delay and jitter. In this experiment, we used several queuing disciplines:

1. *Priority Queue (PQ):* There are two queues in the router, one for the premium service and the other for best-effort service. The premium service queue has higher priority over the best-effort service queue. Thus, all the packets of the premium service queue are served ahead of the best-effort service queue.

2. *Weighted Fair Queue (WFQ):* Same as PQ except that the two queues are weighted fair queues. In order to ensure the quality of premium service, the weight assigned to the premium service queue is more than the actual amount of traffic. For example, WFQ-2.0 means if the average premium service traffic is 1Mbps, we assign a weight equal to 2Mbps capacity to the premium service queue.

3. *Per-flow WFQ:* Each micro-flow has its own queue and is given the weight equal to the peak rate of the voice stream. This queuing scheme is not advisable for DiffServ. We use the result to compare with the first two schemes used in DiffServ, and show how well the aggregate schemes work.

In this simulation, we use the ITU G.711 PCM [33] VoIP traffic as our voice source. The bandwidth of each voice during talk spurt is 87.2 kbps, including the relative protocol headers. Each packet size is 218 bytes. The average burst time is 0.4 second. Average idle time is 0.6 second. It is an exponential ON/OFF model. The packet size of TCP flow is 1000 bytes.



(a) Delay (b) Jitter

Figure 4.3 Delay and jitter in a pipe.

The simulation result in terms of delay and jitter for one of the voice stream from n0 to n3 is shown in Figure 4.3. Figure 4.3(a) is the plot of delay variation and Figure 4.3(b) is the plot for jitters. Observe from the graph that PQ has the best quality, and WFQ-2.0 has almost the same performance as PQ. The performance of WFQ-1.3 is also acceptable. In term of delay, PQ, WFQ-2.0, and WFQ-1.3 work even better than per-flow WFQ. The reason for this could be intuitively explained as follows. Per-flow WFQ could be deemed as a service in which each voice stream uses a thin link and the thin links are separated from each other. PQ, WFQ-2.0 and WFQ-1.3 use a fat link for all of the voice streams. The thin link will stretch individual packet, hence introduce longer delay. The fat link could transmit individual packet much faster, even though the link utilization are similar in both cases. However, per-flow WFQ has almost a fix delay. So the jitter of per-flow WFQ is as good as PQ. The following conclusions could be drawn from these results:

1. Premium service could be implemented with a PQ or an aggregate level WFQ if the relative weight is given higher than 1.3. The performance is comparable to per-flow level WFQ.

2. With appropriate policing at the entrance of pipe, aggregating multiple pipes would not have much side effect on the quality of premium service.

4.3 Improving Pipe Utilization Through Updating

Depending on the burstiness of the aggregate traffic through the pipe, pipe capacity could be set as static or dynamic. If it is static, then all of the admission controls are done at the entrance of the pipe, BB will not receive any updating message from the pipe. However, in order to limit the rejection rate, we will have to reserve the peak of the traffic as the pipe capacity. This, of course, will make the utilization very low given the fact that the traffic could vary greatly during different time of a day and probably vary during different days. On the other hand, we can make the pipe capacity completely dynamic, that is, upon each new call arrival, we update the pipe capacity. In this case, the utilization could be up to 100 percent. However, BB will receive one updating message upon each call arrival/departure, which increases the load and thus defies the benefit of building pipes. So there is always a trade-off between the utilization and the updating overhead. One possible solution is, instead of using static or completely dynamic pipe, a hybrid scheme can be used by updating the pipe capacity periodically. The period is set to several seconds or minutes so that the updating overhead is negligible or acceptable. At the same time, the utilization could still be kept at a high level. In the remaining parts of this section, we explore several updating methods and show how well they work.

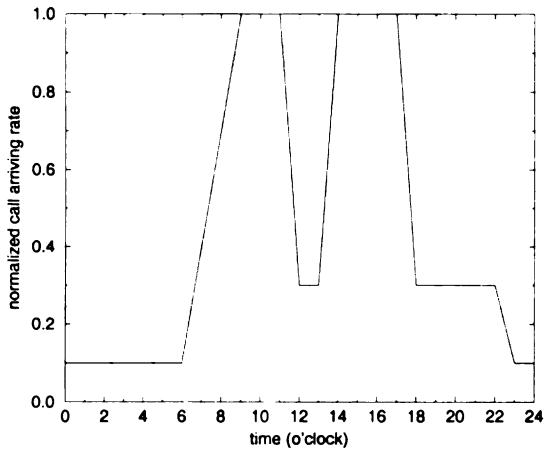
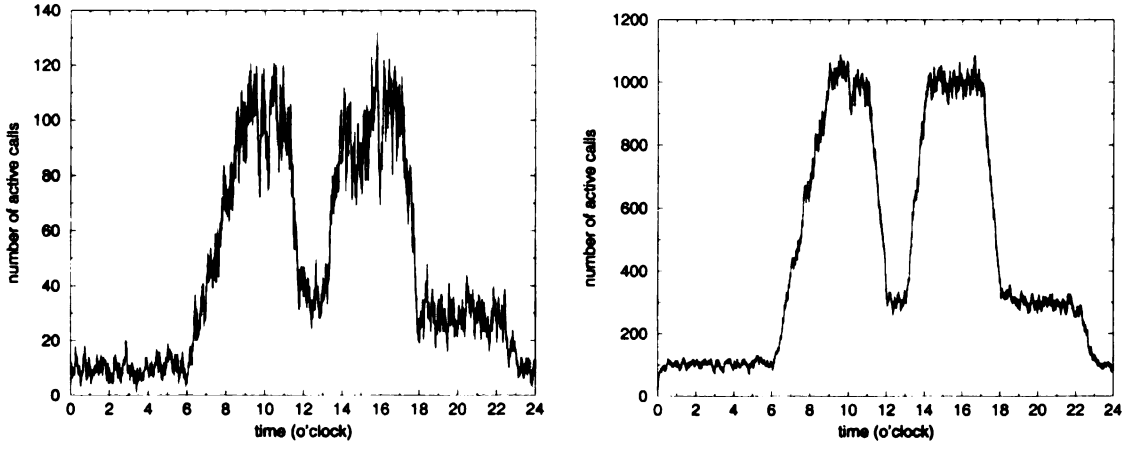


Figure 4.4 Call arrival distribution during a day.

4.3.1 Traffic model

Premium service is designed for real-time traffic like VoIP, Video Conference, etc. Here we select voice traffic as a representative of real-time traffic for our study. VoIP is most likely to become the first user of pipe because of its popularity and the bandwidth limitations of the Internet. The intensity of telephone calls vary greatly during different time of a day. The British Telecommunications (BT) published the average call numbers in their network. Figure 4.4 shows the call arrival pattern where the peak rate (during 9am-11am and 2pm-5pm) is set as 1.0. Actually, the call arrival during a certain period, say, 9am-11am, is not flat as we show in Figure 4.4. In short term, the call arrival could be approximated as a Poisson process. In order to show the detail traffic and see how well different updating methods works for pipe, we build a simple simulator to mimic this arrival/departure process. The length of each call is exponential with an average of 5 minutes. Figure 4.5 is the traffic during a day from the output of our traffic simulator. In Figure 4.5(a), the peak arrival rate is set as 20 calls/min so that on average there are 100 calls in the pipe during the peak period (i.e. 9am-11am, 2pm-5pm). In Figure 4.5(b), the peak arrival rate is set as 200 calls/min so that on average there are 1000 calls in the pipe during the peak period. In the rest

of this chapter, for convenience, we simply say that the pipe with traffic shown in Figure 4.5(a) has 100 calls and the pipe with traffic shown in Figure 4.5(b) has 1000 calls, although 100 or 1000 calls are only the average number of calls in the pipes during the peak period. From the graph we can observe that the relative burstiness of Figure 4.5(a) is larger than that of Figure 4.5(b). For a Poisson arrival/departure process, the variance decreases as the average number of calls increases. Our goal is to find a good prediction method to update the pipe capacity so that we can have both high utilization and acceptable updating overhead.



(a) 100 calls (b) 1000 calls

Figure 4.5 Number of active calls in the pipe during different time of a day.

4.3.2 Prediction Model

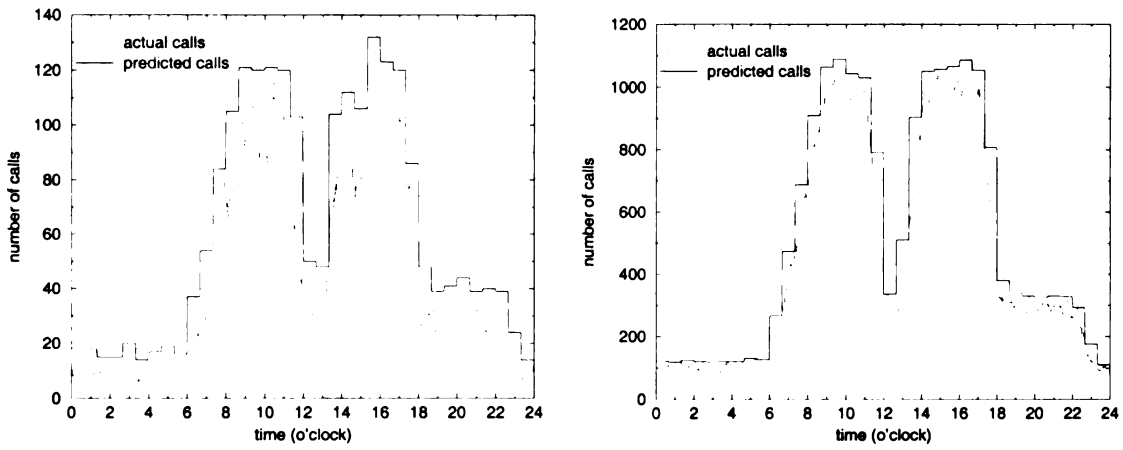
The variance of the number of calls is subject to two main factors: a long term factor which is caused by different usage of phone during different time slot; a short term factor which is caused by the Poisson call arrival process. The long term factor may be due to the variation during a day, or week, or month, or season. The short term factor is due to the independent behavior of users. Without updating, a pipe has to reserve its capacity according to the usage during the peak time. Here we propose a threshold-based prediction scheme to update the pipe capacity. In order

to see how well this scheme performs, we first study the ideal case which we call as ideal prediction.

Ideal Prediction

In this scheme, the pipe capacity is updated periodically. Assume that upon each updating point, we can precisely predict the peak bandwidth of the next period so that we can set the pipe capacity to that value. In this scheme, we can ensure that no call gets rejected and the utilization is as high as possible. However, the utilization is not 100 percent because, during the period between two neighboring updating points, the traffic still varies. Ideal prediction cannot be implemented in real world since we are not aware of the future events. However in our simulation we first count the number of calls during different times and use that information as our "prediction", thus using the future knowledge. Figure 4.6 shows the updating processes. The solid line is the pipe capacity and the doted line is the actual used bandwidth. The updating interval is 40 minutes. From the graph we can find, for the same updating interval, Figure 4.6(b) has higher utilization than Figure 4.6(a). The reason is: Figure 4.6(b) has 10 times more calls than Figure 4.6(a), so the traffic in Figure 4.6(b) is smoother than that in Figure 4.6(a).

Since we choose the peak rate of the period as the pipe capacity, using smaller update interval could increase the pipe utilization. This result is observed in Figure 4.7. From Figure 4.7, we can also observe that the pipe utilization could be higher than 80 percent as the update interval is less than 30 minutes. However, if there is no update during the 24 hours, the utilization would be as low as 40 percent. So updating scheme could improve the utilization by a factor of 2 or more. This result coincides with the result shown in [34] where the traffic trace from AT&T telephone network was used for a similar analysis. For the same update interval, the pipe with 100 calls always have lower utilization than the pipe with 1000 calls. So more the traffic between two end points, more the benefit we can get by building a pipe between them.



(a) 100 calls (b) 1000 calls

Figure 4.6 Ideal prediction (Update Interval=40min).

Threshold-Based Prediction

Ideal prediction is not realistic since we cannot know the future events. Our goal is to find a realistic prediction algorithm which has a performance comparable to the ideal prediction. A linear prediction is not good for this case because of the Poisson call arrival process. Here we propose a simple prediction scheme called *threshold-based prediction.* The motivation for the threshold-based prediction scheme is very simple: if we use the per-flow updating scheme, we need to increase the pipe capacity by one for each call arrival and decrease the pipe capacity by one for each call departure. This could make sure that the pipe has the highest resource utilization, but will incur very high updating overhead. However, if we increase the pipe capacity by δ ($\delta > 1$) when a new call arrives and the pipe is full, then we do not have to update the pipe capacity for every incoming arrival as long as the total number of calls does not exceed the new pipe capacity. By reserving more than we actually need right now, we loose a bit utilization, but we may be able to save a lot of updates, given the fact that the total number of calls in the pipe will not change drastically during a short period. The value for δ is selected based on the trade-off analysis between overhead and utilization. When the number of calls in the pipe drops below a threshold, we can

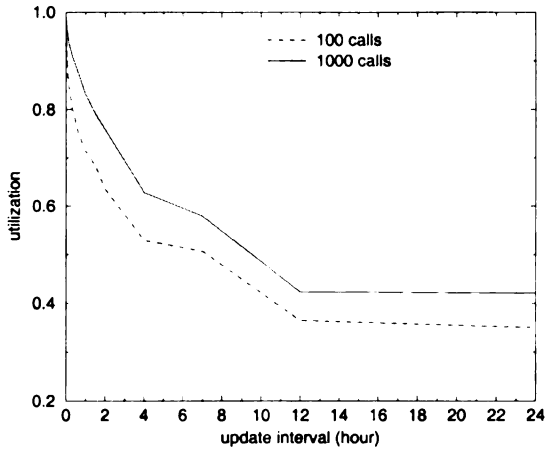


Figure 4.7 Ideal Predication: Utilization vs. Update Interval.

decrease pipe capacity in order to increase the utilization. But we do not decrease the pipe capacity to the exact amount of calls. We can keep it higher than the actual number of calls so that the new coming calls will not trigger a new updating. The algorithm can be stated as follows:

1. Upon a call arrival, if the number of calls reaches the *pipe_capacity*, then *pipe_capacity* is increased by δ.

2. Upon a call departure, if the number of calls is under $pipe\_capacity - 2 \times \delta$, then *pipe_capacity* is decreased by δ.

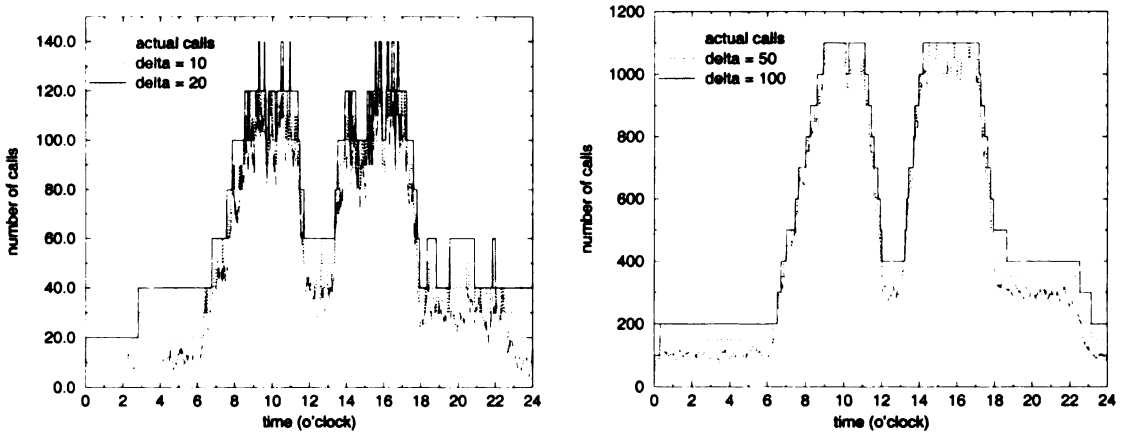
If $\delta = 1$, then this scheme is same as the per-flow updating scheme. Usually, δ is selected larger than 1. The larger δ is, the less frequently the pipe capacity is updated. By doing this, we could ensure:

$$pipe\_capacity - 2 \times \delta \leq number\_of\_calls < pipe\_capacity \tag{4.1}$$

So, the utilization has a lower bound:

$$utilization \geq \frac{pipe\_capacity - 2 \times \delta}{pipe\_capacity} \tag{4.2}$$

The results of the threshold-based updating are shown in Figure 4.8. Figure 4.8(a) is the plot for the pipe with 100 calls. Here the gray line is the actual number of calls in the pipe. The black solid line is the pipe capacity for $\delta = 20$ and the black dotted line is the pipe capacity for $\delta = 10$. With the smaller δ, we have higher utilization however the updating is also more frequent. Figure 4.8(b) is the plot for the pipe with 1000 calls. The pipe capacity for $\delta = 100$ and $\delta = 50$ are shown there. We can observe that the pipe capacity predictions in Figure 4.8(b) are closer to the actual traffic compared to that in Figure 4.8(a) because the traffic are smoother. Instead of updating the pipe capacity periodically, we only update the pipe capacity when the actual number of calls changes out of the region between the upper and lower thresholds. So in Figure 4.8, the updates are less frequent during 0am to 6am, but are more frequent during 6am to 9am.



(a) 100 calls (b) 1000 calls

Figure 4.8 Threshold prediction (δ=delta).

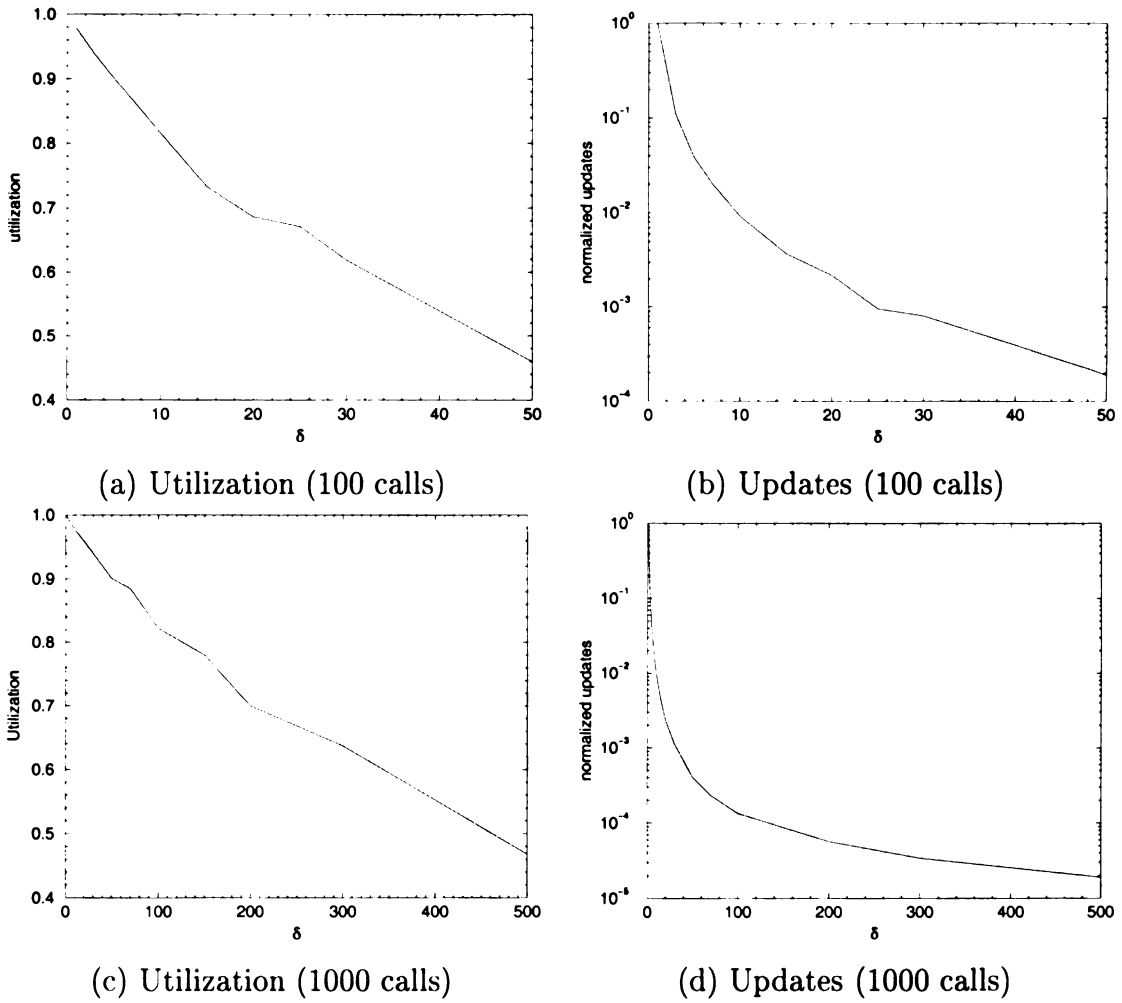
From equation (2) we infer that the utilization could be controlled through δ. Figure 4.9 shows the relationship between utilization and δ. For the pipe with 100 calls (shown in 4.9(a)), if we select $\delta = 10$, the utilization is about 80 percent. For the pipe with 1000 calls (shown in 4.9(c)), if we select $\delta = 100$, the utilization is about 80 percent. For both of them, utilization increases as δ decreases. The corresponding

updating overheads are also shown in the right side. Here we use the normalized updates as the updating overhead. As we know, if we choose $\delta = 1$, i.e. upon each call arrival or departure, we update the pipe capacity, then for each call we have two updates. If we choose $\delta > 1$, we will have less updates. The normalized updates is defines as the actual number of updates during the simulation period divided by double the number of calls, i.e,

$$normalized\_updates = \frac{number\_of\_updates}{2 \times number\_of\_calls} \tag{4.3}$$

Equation (3) could be used to evaluate how much updating overhead can be saved by using pipe. For example, in Figure 4.9(b), if we select $\delta = 10$, the normalized updates is 10^{-2}. So we removed 99% of the updating overhead. In Figure 4.9(d), if we select $\delta = 100$, the normalized updates is 10^{-4}, which means we removed 99.99% of the updating overhead! Thus, by sacrificing a little utilization, we could save a lot of updating overhead, especially when the aggregation degree in the pipe is high (e.g. for the pipe with 1000 calls).

Figure 4.10 shows the relationship between normalized updates and utilization directly so that we can evaluate the performance of the threshold-based updating more clearly. It also makes it possible for us to compare the performance with that of the ideal prediction. The dashed line is the ideal prediction and the solid line is our threshold-based prediction. Obviously, the ideal prediction performs better than our threshold prediction. However, the difference is not large. Given the fact that the threshold-based prediction is very simple and could be easily implemented, it is a very good scheme. In Figure 4.10(a), if we want to keep the utilization to be 80%, for both of the prediction schemes, the normalized updates overhead is about 10^{-2}. In Figure 4.10(b), if we want to keep the utilization to be 80%, for both of the prediction schemes, the normalized updates overhead is about 10^{-4}. The actual number of calls in the later case is ten times of the former case. So the absolute number of updates in the later one is only about 10% of the former one. The number of updating messages



(a) Utilization (100 calls)

(b) Updates (100 calls)

(c) Utilization (1000 calls)

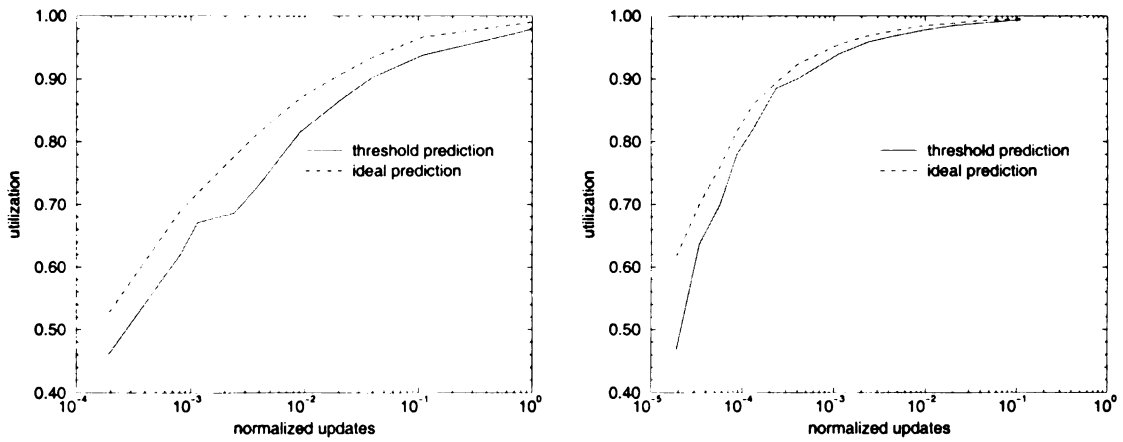
(d) Updates (1000 calls)

Figure 4.9 Relationship between Utilization, Updates and δ.

receive by BB decreases as the total number of calls increases. However, without the proposed pipe implementation, upon each call arrival/departure, the BB will receive an signaling message. The signaling message overhead received by BB is proportional to the number of calls, which may inhibit scalability. This scalability problem is solved by using pipes.

The normalized updates shown in Figure 4.10 is the average value over the 24 hours' simulation. The average update interval could be calculated as the following:

$$average\_update\_interval = \frac{24\ hours}{normalized\_updates \times number\_of\_calls \times 2} \tag{4.4}$$

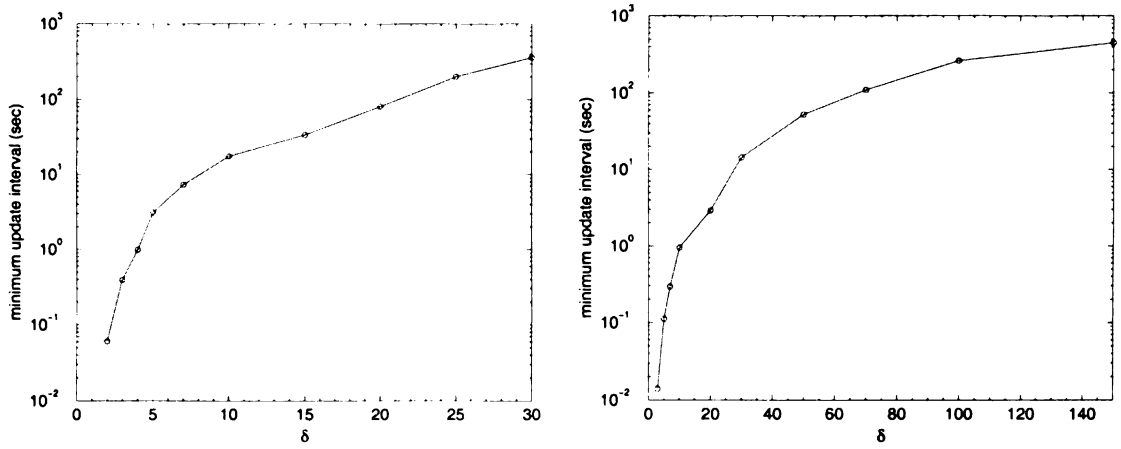


(a) 100 calls (b) 1000 calls

Figure 4.10 Performance comparison of threshold-based prediction and ideal prediction.

Since the threshold-based prediction does not use periodical updating, the update interval during the worst case should be much shorter than the average update interval. The average value is important because one BB may be in charge of many pipes. It is unlikely that all of the pipes will be in the worst case at the same time. Statistically, when one pipe is frequently updated during a period, other pipes may be in a stable state and need not be updated often. However, our simulation shows that even in the worst case, the updating interval is pretty long. The result is shown in Figure 4.11. For the pipe with 100 calls during the peak period in Figure 4.11(a), if we choose $\delta = 10$, then minimum update interval is about 20 seconds. The utilization corresponding to $\delta = 10$ for this pipe is about 80% (see Figure 4.9(a)). For the pipe with 1000 calls during the peak period in Figure 4.11(b), if the we choose $\delta = 100$, then minimum update interval is about 300 seconds. The utilization corresponding to $\delta = 100$ for this pipe is about 80% (see Figure 4.9(c)). The updating interval bounds are long enough. The updating overhead is acceptable even in the worst case.

The proposed threshold-based updating uses a fixed δ. In our simulation, we use the current number of calls in the pipe as a trigger for the pipe capacity updating. In the real implementation, we may use the pipe utilization as a trigger instead of using



(a) 100 calls (b) 1000 calls

Figure 4.11 Minimum update interval for different δ.

the number of calls. This will make it easier to implement in the DiffServ environment because the DiffServ edge router may or may not keep the per-flow state information. The utilization could be achieved through counting the wasted tokens in the leaky bucket. Then we do not have to keep the soft state of each flow at the edge routers (i.e. entrance of the pipe).

CHAPTER 5 CONCLUSION

5.1 Summary

Differentiated services was recently proposed as a scalable solution for the Internet QoS. However, building an end-to-end QoS on a per hop behavior service model is challenging. In order to support an acceptable QoS without restricting the scalability and simplicity, we must have a good resource management scheme.

In the previously proposed studies of differentiated services in the Internet, the marking models in the edge routers demote packets when the available bandwidth is inadequate for the aggregate traffic flow. The demoted packets retain their new marking across all the domains they travel before reaching their destination. A Random Early Demotion and Promotion (REDP) scheme is proposed that can be used for supporting differentiated services through an efficient marking process at the edge routers (the routers between the Internet domains). The primary features of the proposed REDP scheme are the provision of promotion of packets after getting demoted, and the fairness in the promotion and demotion processes. The promotion process is facilitated through a three-color marking process. The fairness is ensured through random and early decisions on the packets. We have simulated the REDP scheme using the *ns* simulator. Results indicate that the marker of the REDP scheme is very fair compared to a leaky bucket marker. The performance in terms of bandwidth allocation for assured traffic is significantly better than the leaky bucket and the previously proposed RIO scheme that uses a two-color marking scheme. We have also analyzed the benefits of promotion by using three drop precedences instead of two

drop precedences and have shown that the assured service gets better service guarantees with three drop precedences compared to the two drop precedences. All the results were obtained for both TCP and UDP traffics to demonstrate the wide applicability of the results and the REDP scheme. A parameter sensitivity study is also reported, results of which could be used as guidelines in determining the parameters for the REDP markers.

The concept of "pipe" was introduced as a solution for resource management of premium service. Pipe could be implemented in an aggregate level as a destination-aware SLA. We use VoIP traffic as an example and show the QoS of a voice steam under different queuing schemes of premium service. Since pipe is relatively static compared to the per-flow dynamic resource management scheme, it greatly reduces the signaling overhead on bandwidth brokers. In order to improve the utilization of pipe, we proposed a threshold-based updating scheme. Through simulation, we have shown that this updating scheme incurs very little overhead while providing high utilization. The threshold-based updating scheme could provide a performance comparable to the ideal prediction scheme, which uses the knowledge of future events.

5.2 Future Work

As discussed in Chapter 2, REDP marker does not work very well for TCP traffic. This is so because TCP's congestion control protocol does not cooperate well with the marker. The same problem exists for other leaky bucket markers. In the current model, TCP has no knowledge about the marker. Embedding the leaf marker with the TCP congestion control scheme should be able to resolve this problem. This enhancement and a prototype implementation of the REDP maker in a laboratory environment is being considered for future work.

Bibliography

[1] R. Comerford, "State of the Internet: Roundtable 4.0," IEEE Spectrum, Oct. 1998.

[2] [Online] Available http://www-mash.cs.berkeley.edu/ns/.

[3] D. Ferrari and L. Delgrossi, "Charging For QoS," IEEE/IFIP IWQOS '98 keynote paper, Napa, California, May 1998.

[4] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," Internet Draft, RFC 1633, June, 1994.

[5] S. Shenker, C. Partridge and R. Guerin, "Specification of Guaranteed Quality of Service," Internet Draft, RFC 2212, Sep. 1997.

[6] J. Wroclawski, "Specification of the Controlled-Load Network Element Service," Internet Draft, RFC 2211, Sep. 1997.

[7] D. Clark, "Combining Sender and Receiver Payments in the Internet," Telecommunications Research Policy Conference, October, 1996.

[8] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang, "Pricing in Computer Networks: Motivation, Formulation, and Example," IEEE/ACM Trans. On Networking, pp. 614-627, December, 1993.

[9] A. Basu, Z. Wang, "A Comparative Study of Schemes for Differentiated Services," Tech. Report, Bell Lab, Lucent Technologies, August 1998.

[10] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, pp397-413, August 1993.

[11] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource Reservation Protocol." IEEE Networks, September 1993.

[12] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, K. Nichols, M. Speer, "A Framework for Use of RSVP with Diff-serv Networks," IETF Internet Draft, November 1998.

[13] D. Ferrari, D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," IEEE JSAC, vol. 8, no. 3, Apr. 1990, pp. 368-379.

[14] A. Banerjea and B. Mah, "The Real-Time Channel Administration Protocol," Proceedings of the 2nd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'91), Springer-Verlag, Heidelberg, Germany, pp. 160-170.

[15] P.Ferguson and G. Huston, "Quality of Service," John Wiley & Sons, 1998.

[16] J. Heinanen and R. Guerin, "A Two Rate Three Color Marker," Internet Draft, <draft-heinanen-diffserv-trtcm-01.txt>, May 1999.

[17] H. Kim, *"A Fair Marker,"*, Internet Draft, <draft-kim-fairmarker-diffserv-00.txt>, Apr. 1999.

[18] S. Floyd and V. Jacobson, *"On Traffic Phase Effects in Packet-Switched Gateways,"* Internetworking: Research and Experience, V.3 N.3, Sep 1992, p.115-156.

[19] W. Feng, D. Kandlur, D. Saha, K. Shin, *"Understanding and Improving TCP Performance over Networks with Minimum Rate Guarantees,"* IEEE/ACM Transactions on Networking, Vol. 7, No. 2, pp. 173-187, Apr. 1999.

[20] N. Seddigh, B. Nandy and P. Pieda, *"Study of TCP and UDP Interactions for the AF PHB,"* Internet Draft, <draft-nsbnpp-diffserv-tcpudpaf-00.txt>, Jun. 1999.

[21] IETF Home Page, [Online] Available http://www.ietf.org/.

[22] Y.Bernet, J. Binder, S. Blake, M. Carlson, B. E. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Berma, "A Framework for Differentiated Services," Internet Draft, [Online] Available http://www.ietf.org/internet-drafts/draft-ietf-diffserv-framework-02.txt, Feb. 1999.

[23] X. Xiao and L. M. Ni, "Internet QoS: the Big Picture," IEEE Network, March/April 1999.

[24] F. Reichmeyer, L. Ong, A. Terzis, L. Zhang, R. Yavatkar, "A Two-Tier Resource Management Model for Differentiated Services Networks," Internet Draft, Nov. 1998.

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

[26] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services," Internet Draft, RFC 2475, Dec. 1998.

[27] V. Jacobson, K. Nichols, K. Poduri, "An Expedited Forwarding PHB," Internet Draft, RFC 2598, Jun. 1999.

[28] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group," Internet Draft, RFC 2597, Jun. 1999.

[29] V. Jacobson, "Differentiated Services Architecture," Talk in the Int-Serv WG at the Munich IETF, Aug. 1997.

[30] D. D. Clark, "Adding Service Discrimination to the Internet," Tech. Report, MIT Laboratory of Computer Science, Sep. 1995.

[31] D. D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," Tech. Report, MIT Laboratory of Computer Science, 1998.

[32] K. Nichols, V. Jacobson and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," Internet Draft, Nov. 1997. [Online] Available ftp://ftp.ee.lbl.gov/papers/dsarch.pdf.

[33] International Telecommunication Union (ITU), [Online] Available http://www.itu.int/.

[34] P. Goyal, A. Greenberg, C.R. Kalmanek, W.T. Marshall, P. Mishra, D. Nortz, K.K. Ramakrishnan, "Integration of Call Signaling and Resource Management for IP Telephony," IEEE Network Magazine, Vol 13, No. 3, May/June 1999, pp-24-32.