Adaptive Packet Pre-marker for TCP-based Applications in DiffServ Networks

Chih-Heng Ke, Ce-Kuen Shieh, *Yao-Ting Chen,*Wen-Shyang Hwang
Department of Electrical Engineering,
National Cheng Kung University, Taiwan R.O.C.
*Department of Electrical Engineering,
National Kaohsiung University of Applied Sciences, Taiwan R.O.C
smallko@hpds.ee.ncku.edu.tw, shieh@ee.ncku.edu.tw
timoth@wshlab2.ee.kuas.edu.tw, wshwang@mail.ee.kuas.edu.tw

Abstract

This paper proposes a control mechanism named PPE (Packet Pre-marking Engine) in application layer to assist TCP-based connection flow in maintaining end-to-end throughput in DiffServ networks. The application with PPE built in can adaptively adjust the DSCP (DiffServ CodePoint) of user's traffic according to the network traffic load to inform the service provider what kind of service is needed. When the network is light-loaded, the user's requirement can be met and the best service is provided. As the traffic load becomes heavy, the requirement will fail to meet, and the better level of service is used instead. In this paper, we will implement an FTP client program with PPE in a testing DiffServ platform to illustrate the effectiveness of this control mechanism.

Keywords: DiffServ, TCP, PPE, application

I. Introduction

The Internet has historically offered a best-effort delivery service, where all user packets are equally treated in the network. Under this kind of service model, it is insufficient to meet the requirement of emerging real-time multimedia applications, and difficult to provide a better-than-best-effort service when customers are willing to pay more. Therefore, two different service models have been defined for network QoS by IETF [1] (Internet Engineering Task Force): Integrated Services (IS) [2] and Differentiated Services (DS) [3]. IS is an architecture that provides service discrimination by explicit allocation and scheduling of resources in the network. However, the complexity and scalability problems of IS has led DS to draw much attention to address quality of service. DS is based on a simple model where traffic entering a network is classified and possibly conditioned at the boundaries of the network, and assigned to different behavior aggregates that are a collection of packets with common characteristics. Each behavior aggregate is identified by a single DSCP (Differentiated Service CodePoint). Within the core of the network, packets are forwarded according to the Per-Hop Behavior (PHB) associated with the DSCP. Per-flow state does not need to be maintained in the core routers, which leads to increased scalability.

Transmission Control Protocol (TCP) is the most widely used transport layer protocol in the Internet. Most popular applications, such as Web and file transfer use the reliable services provided by TCP. Although the well-developed congestion and flow control mechanism of TCP helps these applications work well in the traditional best-effort service based Internet, customers' need and service providers' expect would not be satisfied. Service providers would expect to maximize their return on investment in network infrastructure through offering different better-than-best-effort services and charging more money. In the other hand, customers would want to pay more to meet their requirement. Therefore, several traffic management and packet marking mechanisms [4-8] have been proposed for improving TCP performance with minimum rate guarantee in a differentiated service network. However, these researches put more efforts on network than on applications. We argue that applications themselves are also important and need to be evolved. The consideration of customer's preference is an indispensable necessity for supporting QoS within the end-system, as only the customer is able to decide which application is important for him/her and should be preferred. This paper assumes network provides two different levels of service according to the Type of Service (TOS) bits of user's traffic, and the usage-based pricing strategy is used to discourage users from continually requesting the higher level of service. We propose a control mechanism, which we call a packet pre-marking engine (PPE), in application layer to help the individual connection flow maintain end-to-end throughput while keeping the service expenditure as low as possible. PPE measures the return speed of acknowledgement packet of TCP that approximately represents network status and then adaptively adjusts ToS bits when sending out customer's traffic. If the network is light-loaded and measured transfer rate is above the requested rate, the best-effort packets are generated. When traffic load becomes heavy and the measured rate falls below the minimum target rate, the ToS bits are set to inform service provider the higher service is needed.

The rest of the paper is organized as follows: Section II gives an overview of the data path from application level to kernel space. Section III presents the function of packet pre-marking engine and Section IV validates the PPE by using FTP with an adaptive packet pre-marking mechanism. We conclude in section V.

II. Background

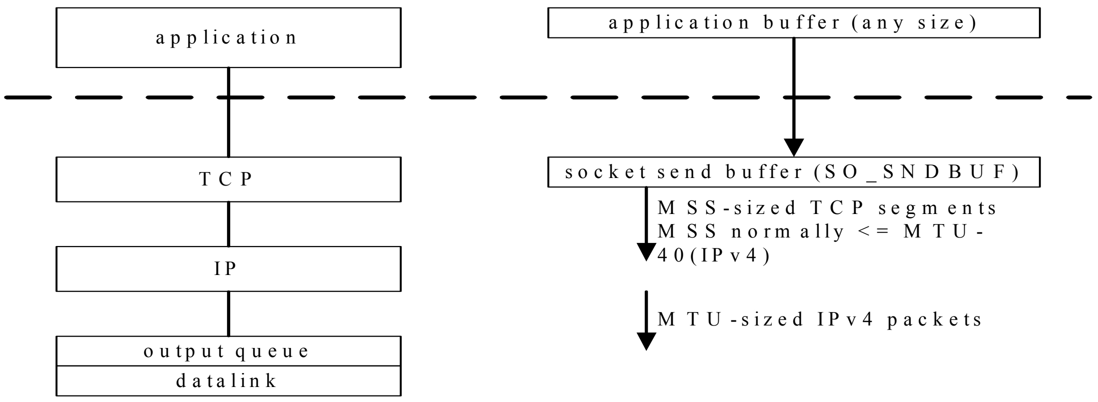
The Internet has two main protocols in the transport layer, TCP and UDP. TCP provides a reliable service and UDP provides an unreliable service. Because there is a distinct discrepancy in the socket send buffer. We can see clearly from Figure 1 and Figure 2. Every TCP socket has a send buffer, but UDP doesn't. In TCP, when an application calls the function ***write()***, the kernel copies all the data from the application buffer into the socket send buffer. If there is insufficient room in the socket buffer for all of the application's data (either the application buffer is larger than the socket send buffer, or there is already data in the socket send buffer), the process is put to sleep. The kernel will return from the function ***write()*** until the final byte in the application buffer has been copied into the socket send buffer. Then TCP takes the data in the socket send buffer and sends it to the peer TCP, based on all the rules of TCP data transmission. The peer TCP must acknowledge the data from the socket send buffer. TCP must keep a copy of the data until the peer acknowledges it. On the contrary, UDP does not have a send buffer. When the application data is copied into a kernel buffer as it passes down the protocol stack, this copy is discarded by the data-link layer after the data is transmitted [9].

III. Packet Pre-marking Engine (PPE)

A packet pre-marking engine (PPE) is mainly made up of two parts. One is the transfer rate monitor and the other part is the packet pre-marking decision-maker. The transfer rate monitor snoops on the application's data copying from user space to kernel space and measures its observed transfer rate. In the following, the copying rate and transfer rate mean the same thing and will be used interchangeably. Then the measured information is then passed to the packet pre-marking decision-maker to decide whether to pre-mark packets or not. If the observed transfer rate is above the requested target rate, the PPE takes the role of a passive monitor. If the measured transfer rate is below its requested target rate, the PPE takes a more active role and starts pre-marking packets. The fraction of pre-marked packets varies from 0 to 1 depending upon algorithm chosen in packet pre-marking decision-maker. Selective upgrading the fraction of packets to the higher priority level will help sustain the transfer rate close to the requested target rate and keep the number of the pre-marked packets as low as possible.

A. The transfer rate monitor

The main task of the transfer rate monitor is used to measure the data's copying rate from user space to kernel space. The measured information implies the end-to-end network traffic load which can be fed into the packet pre-marking decision-maker that has to decide whether to pre-mark packets or not. From section 2, we know that the sender using TCP to transmit data will keep its sending data in the kernel space buffer until the acknowledgements from the receiver is received. But the sending buffer is limited. Therefore when the end-to-end network traffic load is heavy and then the speed of the acknowledgement from the receiver is slow, the sender's copying rate is also slow because the buffer is filled with the unacknowledged data. In the other hand, when the end-to-end network traffic is light-loaded and then the speed of acknowledgement from the receiver is fast, the sender's copying rate is also fast because the unacknowledged data is easy to wipe out. From a network application's point of view,



application
application buffer (any size)
TCP
socket send buffer (SO_SNDBUF)
MSS-sized TCP segments
MSS normally <= MTU-40(IPv4)
IP
MTU-sized IPv4 packets
output queue
datalink

Figure 1 Steps and buffers involved when application writes to a TCP socket

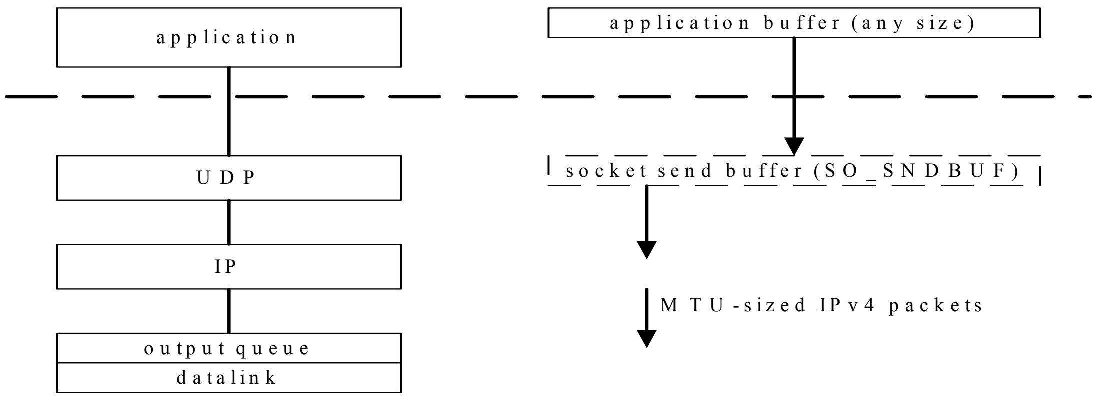


Figure 2 Steps and buffers involved when application writes to a UDP socket

the copying rate is easy to get. Every time the function ***write()*** which is usually used to send data based on TCP is called and the return value which means the actual data bytes copying from user space to kernel space is recorded. Then we can get average transfer rate by dividing the altogether bytes with total transfer time.

B. The packet pre-marking decision-maker

The main purpose of the packet pre-marking decision-maker is to adaptively adjust the packet pre-marking rate based on the measured transfer rate. We will introduce two different algorithms to implement the decision-maker. One is the probabilistic marking scheme proposed in [4] and the other one is 2-bit-states method. In probabilistic marking scheme, packets are randomly pre-marked and the marking probability (prob) is periodically updated depending on measured transfer rate and requested target rate. Figure 3 shows the algorithm when we use the probabilistic marking scheme. Giving an instance to explain the algorithm. Suppose that initially the prob is set to 70% to pre-mark packets when the transfer rate is less than the target rate, and the target rate is 100 Kbytes/sec. If the measured transfer rate is 90Kbytes/sec, the prob will become 73%. If the measured transfer rate is 110Kbytes/sec, the prob will become 27%. One advantage of this algorithm is to decrease the prob quickly when the transfer rate is higher than the target rate. This helps the number of pre-marked packets as low as possible. Once the prob is given, a random number generator is used to generate a number to compare with the prob and then to determine whether to pre-mark packets or not. If it needs to change the sending service, the function ***setsockopt()*** is called to change the ToS bits in the IP header.

But [4] also shows that probabilistic marking will result in potential network instability in the network due to large swings in the number of marked and unmarked packets. So we propose a 2-bit-states method to mitigate the problem mentioned above. Figure 4 shows our method. The basic idea behind this method is that we use some transient states to minimize the changes from unmarked packets to

```
Every update interval
If ( transfer_rate  < target_rate ){
        scale = ( target_rate- transfer_rate ) / target_rate;
        prob = porb + scale * ( 100 - prob );
}
else {
        scale = ( transfer_rate - target_rate ) / transfer_rate;
        prob = ( 100 - prob ) * ( 1 - scale) ;
}
```

Figure 3 probability marking scheme

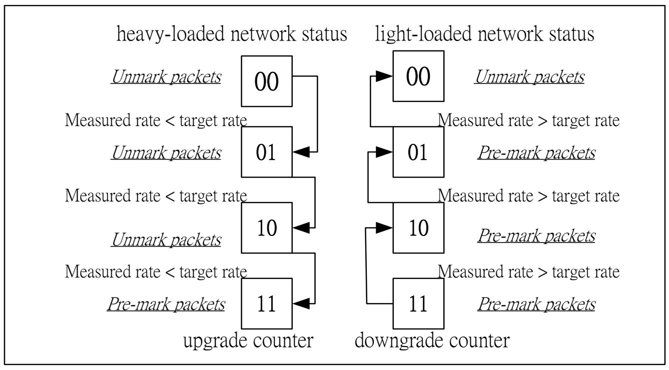

heavy-loaded network status
light-loaded network status
Unmark packets
00
Unmark packets
00
Measured rate < target rate
Measured rate > target rate
Unmark packets
01
01
Pre-mark packets
Measured rate < target rate
Measured rate > target rate
Unmark packets
10
10
Pre-mark packets
Measured rate < target rate
Measured rate > target rate
Pre-mark packets
11
11
Pre-mark packets
upgrade counter
downgrade counter

Figure 4 2-bit-states method

pre-marked packets or vice versa so frequently. We use two counters in our algorithm, and each counter has 4 different states, i.e. 00, 01, 10, and 11. Upgrade counter is used when network traffic becomes heavy and measured rate is below the target rate. Downgrade counter is used when network status becomes better and measured rate is above target rate. 00 represents the default state and best effort service is used. 11 state means that the current best-effort service is not sufficient to meet the user's requirement, so we have to pre-mark packets to indicate the ISP that higher priority service is needed. 01 and 10 are the transient states. Initially, the state is 00. When the measured rate is less than the target rate, state will change from 00 to 01. But we don't pre-mark packets immediately. The network may be congested just for a short period. We just start to pre-mark packets when the measured rate is below the target rate successively observed 3 times. Likewise, when the network status becomes better, we don't use best-effort service packets until the state changing from 11 to 00.

IV. Experimental Results

To verify our PPE and compare the performance of different methods chosen in packet pre-marking decision-maker, we use Cisco routers to construct a DiffServ testbed. Figure 5 shows the experimental environment. We will run FTP client programs with PPE built in on PC sender 1 and PC sender 2. PC Receiver is the destination of traffic sent from these two senders and has a FTP server program on it. We also use a library ***libpcap*** on Linux platform to implement the Packet Monitor 1 and Packet Monitor 2 to record the information of data transmission. The network itself consists of 4 Cisco routers. The egress routers, Router 1 and Router 2, and the ingress router, Router3, are Cisco 1700 routers. The core router, Router 4, is a Cisco 2621 router which has Weighted Fair Queueing enabled. The bottleneck link of capacity is 250 Kbytes/sec and lies between Router 4 and Router 3. Other links of capacity is 10 Mbps.

In the first experiment, we set the target transfer rate of flow 1 on PC sender 1 to 150Kbytes/sec and of flow 2 to 50Kbytes/sec on PC sender 2. The transmitted file length is 10598400 bytes for both senders. First we use probability marking based packet pre-marker on PC sender 1 and PC sender 2 to transmit files and then record the observation of flow 1 on PC monitor 1 and of flow 2 on PC monitor 2. Following, we use 2-bit-states marking based packet pre-marker. Figure 6 and Figure 7 show the observations. From these two figures, we can clearly

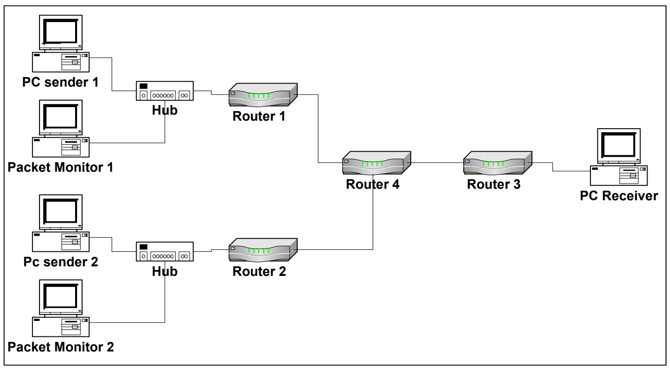


Figure 5 Experimental Environment

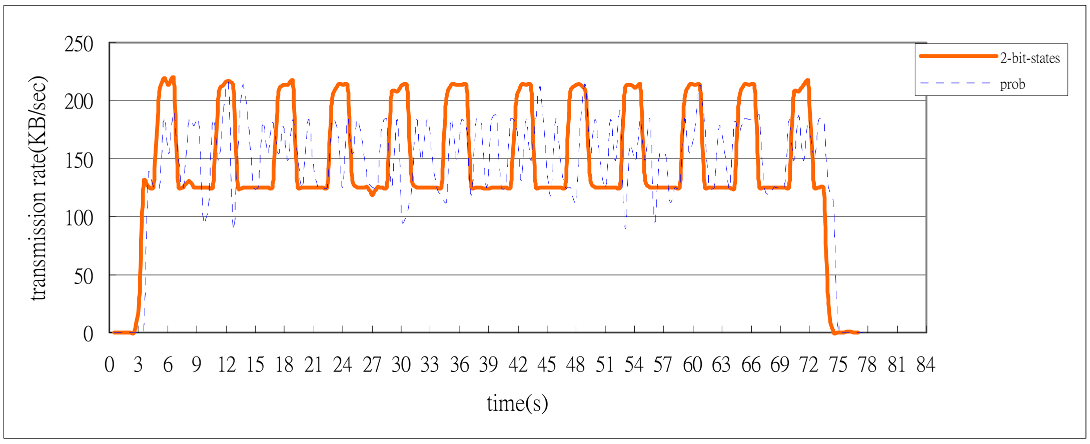


Figure 6 the transmission rate of flow 1 observed by packet monitor 1

see that both probability and 2-bit-states methods can achieve the target rate, 150 Kbytes/sec and 50 Kbytes/sec respectively. It also can be shown that the transfer rate of 2-bit-states method is more stable than that of probability method. But it is interesting to note that flow 2 achieves about 100Kbytes/sec transmission rate when flow 1 is on. Unlike other research [7] which aims to make all flows get a share of excess bandwidth proportional to their target rates in an under-subscribed network. Our goal is to keep the number of pre-marked packets as low as possible when the target rate of every flow can be achieved. So flow 1 just needs to get some bandwidth from flow 2 through pre-marking some packets to achieve 150Kbytes/sec. Then flow 2 gets the remaining bandwidth and achieves the transmission rate of 100Kbytes/sec when flow 1 is on. Take a closer look in this experiment, we shows the number of unmarked and pre-marked packets in Figure 8 and Figure 9. We can find that the number of pre-marked packets of 2-bit-states method is less than that of probability method. This is because the transient states smooth out the variation and then reduce the number of pre-marked packets.

Following experiment, we set the target transfer rate of flow 1 on PC sender 1 to 200Kbytes/sec and of flow 2 to 150Kbytes/sec on PC sender 2. This will cause the aggregate target rate above the capacity of bottleneck link between Router 4 and Router 3. Figure 10 and Figure 11 show that both methods can not meet the requested rate. So we propose to use two thresholds in packet pre-marking decision-maker to alleviate this problem. One is the target transfer rate threshold which is the user's requirement. The other one is the minimum acceptable sending rate threshold which means that user is willing to start paying more money to compete for better-than-best-effort service. Therefore we do the experiment again but with minimum acceptable sending rate set. The target transfer rate and minimum acceptable sending rate of flow 1 is set to 200Kbytes/sec and 150Kbytes/sec respectively. And the target transfer rate and minimum acceptable sending rate of flow 2 is set to 150Kbytes/sec and 100Kbytes/sec respectively.

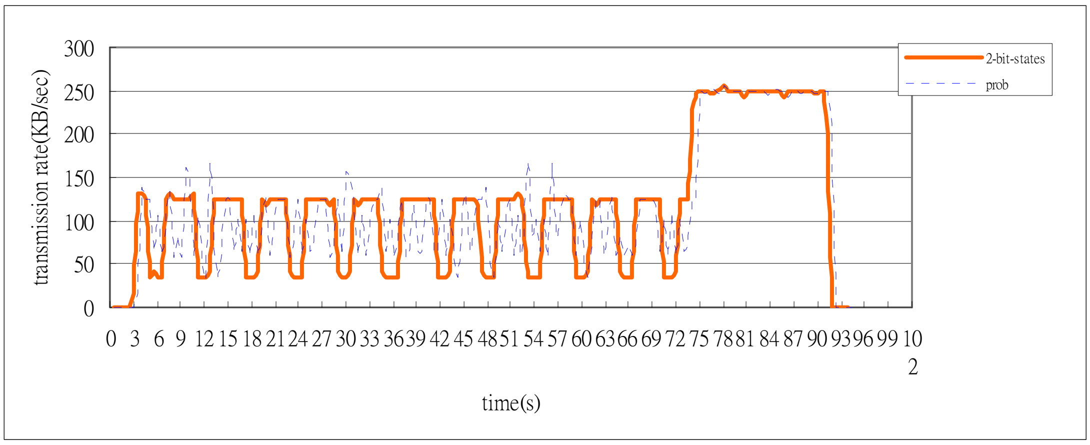


Figure 7 the transmission rate of flow 2 observed by packet monitor 2

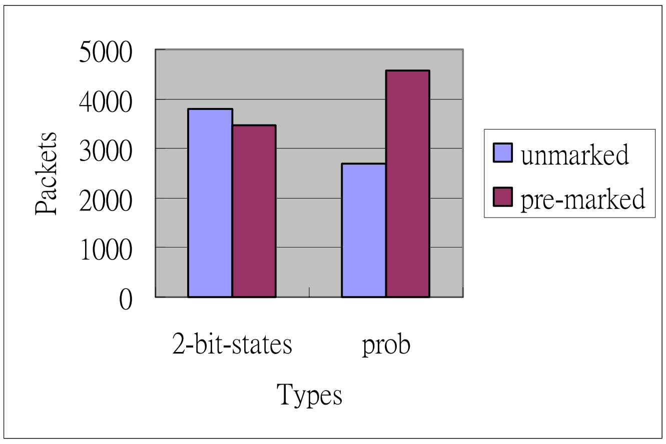


Figure 8 the number of unmarked and pre-marked packets in flow 1

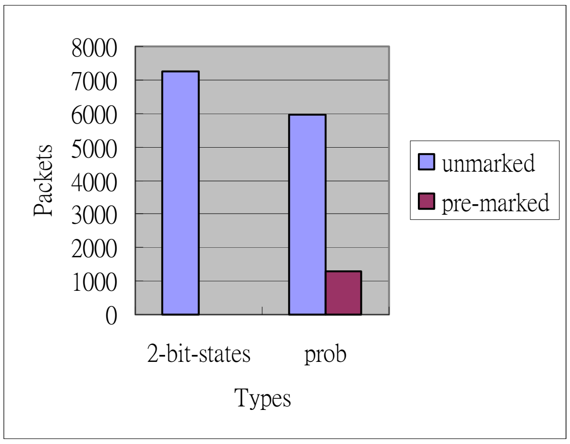


Figure 9 the number of unmarked and pre-marked packets in flow 2

Figure 12 and Figure 13 show the results. When flow1 still fails to achieve the minimum acceptable sending rate, flow 1 gives up the competition for bandwidth. Then the flow 2 will get the bandwidth and achieve the target rate, 150Kbytes/sec.

V. Conclusion and Future Work

In this paper, we have presented two important components of the packet pre-marking engine (PPE): the transfer rate monitor and the packet pre-marking

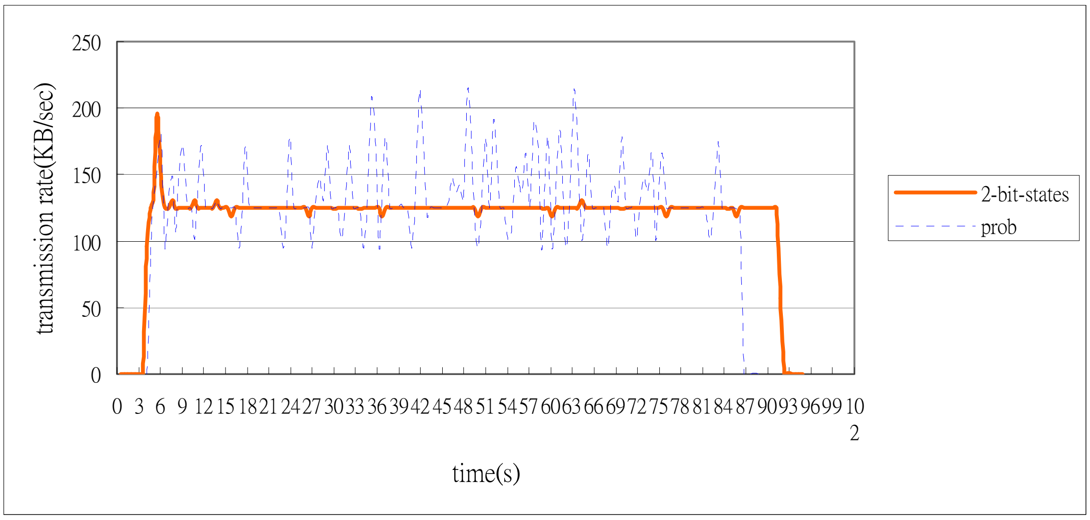


Figure 10 the transmission rate of flow 1

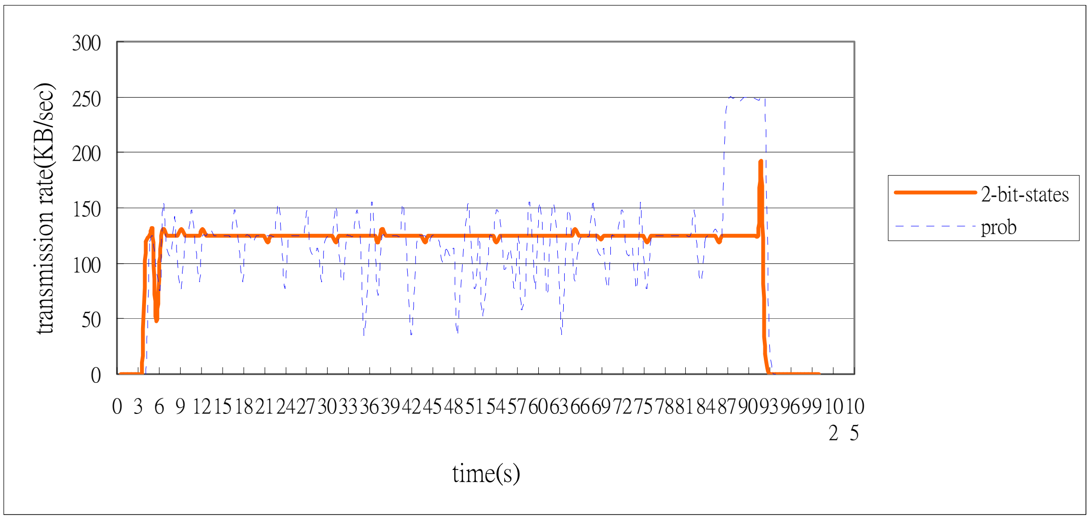


Figure 11 the transmission rate of flow 2

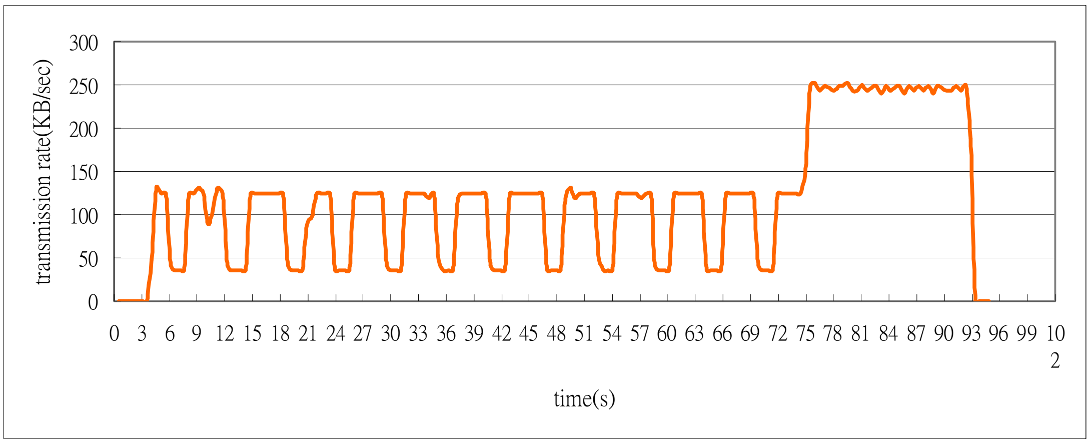


Figure 12 the transmission rate of flow 1 with minimum acceptable sending rate set

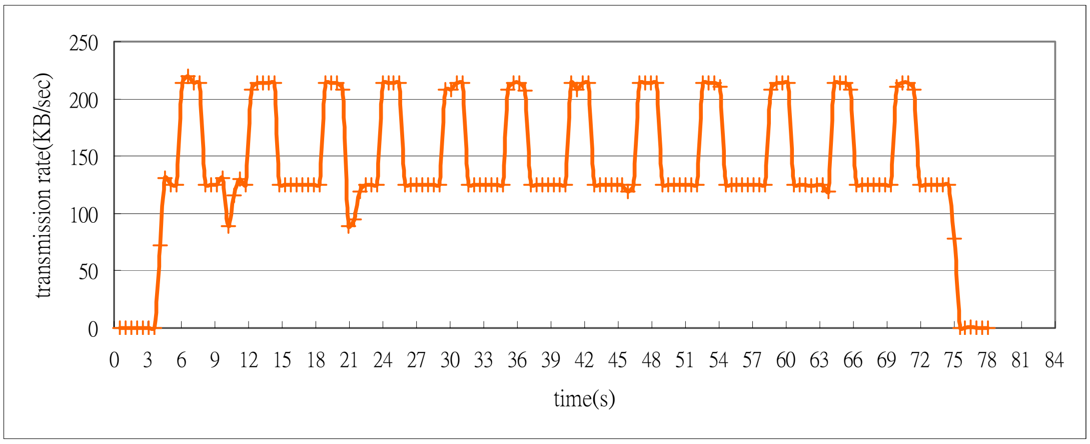


Figure 13 the transmission rate of flow 2 with minimum acceptable sending rate set

decision-maker. The FTP client program with PPE built in is used to study different requested transfer rate. The experimental results show the effectiveness of our PPE and the 2-bit-states method is a better choice to implement the pre-marking decision-maker.

In the future, we will extend the two-priority ToS scheme to multiple priorities. We will also validate the PPE on more complicated network topology.

References

[1] "IETF home page," http://www.ietf.org/.

[2] R. Braden, L.Zhang, S. Berson, S. Herzong, and S. Jamin, " Resource ReSerVation protocol (RSVP)－Version 1 functional specification," RFC 2205, Sept. 1997.

[3] Y. Bernet, J. Binder, S. Blake, M. Carlson, B. E. Carpenter, S. Keshav, E. Davies, B. Ohlman, and D. Berma, "A framework for differentiated services," Internet Draft, Feb. 1999

[4] Wu-Chang Feng, Dilip D. Kandlur, Debanjan Saha, and Kang G. Shin, "Adaptive Packet Marking for Maintaining End-to-End Throughput in a Differentiated-Services Internet," IEEE/ACM Transactions on Networking Vol. 7, No. 5, October 1999.

[5] David D. Clark, and Wenjia Fang, "Explicit allocation of best-effort packet delivery service," IEEE/ACM Transactions on Networking, Vol. 6, No. 4, August 1998

[6] Xiaoning He, Hao Che*,* " Achieving end-to-end throughput guarantee for TCP flows in a differentiated services network," Computer Communications and Networks, 2000.

[7] Mohamed A. El-Gendy, Kang G. Shin, "Equation-Based Packet Marking for Assured Forwarding Services," IEEE INFOCOM, 2002

[8] K.R. Renjish Kumar, A.L. Ananda, Lillykutty Jacob, "TCP-friendly traffic conditioning in DiffServ networks: a memory-based approach," Computer Networks, 2002

[9] W. Richard Steven, "Unix Network Programming," Volume 1, Networking APIs:Sockets and XTI, second edition, Prentice Hall Inc., 1998