STPPE: Source Transparent Packet Pre-Marking Engine in DiffServ Networks

Chih-Heng Ke, Yu Yun Shuai, Ce-Kuen Shieh
EE Department, National Cheng Kung University, Taiwan R.O.C.
Email: {smallko, kaede, shieh}@hpds.ee.ncku.edu.tw

Abstract

In this paper we propose a source transparent packet pre-marking engine (STPPE) to help legacy applications maintain end-to-end throughput in DiffServ networks. By means of Linux Firewall, packets sent from legacy applications will be divert to a specific system port. After that, STPPE uses Linux Divert Socket to receive those packets, modifies the Differentiated Services Code Point (DSCP) fields in the IP headers, and then re-injects the packets into network. In this way, users' traffic can be pre-marked without any modification of original legacy applications to inform the service provider that higher service is needed if their requirements can not be met.

Keyword: DiffServ, STPPE, Divert Socket

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 time- and performance-critical applications, and difficult to provide a better-than-best-effort service when customers are willing to pay more for more bandwidth. Therefore, two different service models have been defined for network Quality of Service (QoS) by IETF [1] (Internet Engineering Task Force: the Integrated Services (IntServ) model [2] and the Differentiated Services (DiffServ) model [3]. IntServ is an architecture that provides service discrimination by explicit allocation and scheduling of resources in the network. However, due to the complexity and scalability problems of IntServ, DiffServ has drawn more attention lately on addressing the QoS issue. Based on a simple model, DiffServ classifies traffic upon entering the network edge into several different behavior aggregates. Each behavior aggregate is a collection of packets with common characteristics and is identified by a single DSCP (Differentiated Service CodePoint). Within the core of a network, packets are forwarded according to the Per-Hop Behavior (PHB) associated with the DSCP. Since the core routers do not need to maintain per-flow state, they can achieve better scalability.

The two basic PHBs defined for DiffServ are the Expedited Forwarding (EF) and the Assured Forwarding (AF) PHBs. The EF PHB is used to provide services that require low delay, low jitter, and low loss, while the AF PHB is to support more elastic services that impose requirements only on throughput without any delay or jitter requirements. The idea behind AF PHB is to give the customer the assurance of a minimum throughput, even during periods of congestion, while allowing him to consume more bandwidth when the network load is low. Thus a connection using the assured service should achieve a throughput equal to the subscribed minimum rate, also called target rate, plus some share of the remaining bandwidth gained by competing with all the active connections. The AF PHB provides four independent classes for the delivery of IP packets, where each class is allocated a certain amount of resources, such as bandwidth and buffer, in each DiffServ node. Within each AF class, IP packets are marked with one of three drop precedence levels (or three colors)---green, yellow, and red---where green has the lowest drop probability and red has the highest drop probability. That is, at the time of congestion, packets with red marking are dropped first, and then packets with yellow marking are dropped next if the congestion condition continues. Finally, packets with green marking are dropped if the congestion persists. In this way, it is expected that with appropriate negotiation and marking, end-to-end minimum throughput could be assured or at least assured to some extent.

Recently, some intelligent markers have been proposed to provide the minimum throughput guarantee [4][5] or to overcome the unfairness problem associated with different RTTs and different target rates [6][7][8] in AF

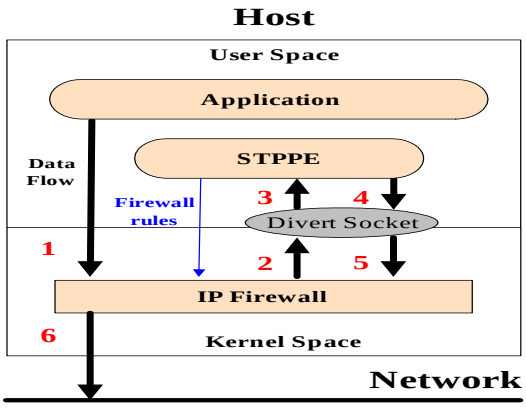
based DiffServ network. But, these researches put more efforts on network than on applications. We argue that applications themselves are also important and need to be evolved. The consideration of customer's preference is an indispensable necessity for supporting QoS within the end-system, as only the customer is able to decide which application is important for him/her and should be preferred. Thus customers can pay more to meet their requirements. Also, service providers would maximize their return on investment in network infrastructure through offering different better-than-best-effort services and charging more money.

However, legacy applications are QoS-unaware which means that these applications do not allow users to choose preferred service. One solution of this problem is to design QoS-aware applications from scratch. But this will cost a lot of effort. Another is to modify the source codes of legacy applications to add the QoS ability. But this needs the source code available. Therefore, we propose a source transparent packet pre-marking engine (STPPE), which is based on Linux Firewall and Linux Divert Sockets [9], to help legacy applications transparently pre-mark the traffic to get better service without any modification of original application source codes. The only thing user need to do before application running is to specify some parameters, such as destination IP address, destination port number, and the wished transfer target rate. Then traffic sent by this application will be pre-marked to inform the service provider that the higher service is needed.

The rest of the paper is organized as follows: Section II presents the implementation of STPPE. Section III validates the effectiveness of STPPE and tests the overhead of Linux Divert Sockets. Then we conclude in section IV.

II. Source Transparent Packet Pre-Marking Engine (STPPE)

The details how STPPE works are shown in Fig. 1. Before running legacy application, user have to set up some parameters, such as destination IP address, destination port number, and the desired transfer target rate, through the graphical user interface of STPPE shown in Fig. 2 which is implemented by GTK. After that, STPPE will use the information given by the user to setup the Linux Firewall rules. When application begins to send packets, all the packets matched the rules will be redirected to a specific system port. Then STPPE use Linux Divert Sockets to get all the redirected packets, modify the DSCP in the IP header if needed, and finally re-inject them into network. In current state, we only implement Time Siding Window Two Color Marker (TSW2CM) [10] in STPPE. When the average sending rate is below the desired target rate, the packets will be pre-marked with low drop probability. When above the target rate, the packets will be pre-marked with normal drop probability. STPPE will monitor the transmitting rates of the applications and shows them in the Statistics area (see Fig 2).



. Figure 1. The packet processing by STPPE

Figure 2. The graphical user interface of STPPE

III. Experimental Result

Two experiments are conducted in order to evaluate the feasibility of the STPPE. The first experiment verifies the effectiveness of the STPPE, while the second experiment measures its overhead.

A. Effectiveness of the STPPE

Fig. 3 shows the experimental environment. There are three hosts and one router. The two sending hosts are named Henry and Liza. Bob is the destination for both traffic streams. Henry uses FTP client program to send traffic, and sets target rate to 500Kbytes/sec through STPPE, while Liza uses default best-effort service to send packets. Grace is a Linux-based pc router, which enables General Random Early Detection mechanism [11]. GRED is a queuing mechanism that generalizes CISCO's DWRED and Dave Clark's RIO. The RED parameters {min_{th}, max_{th}, P_{max}} used are :{ 40, 80, 0.02} for low drop probability packet; and {4, 8, 0.1} for normal packet. All interfaces of Grace are 10 Mbps point-to-point Ethernet links.

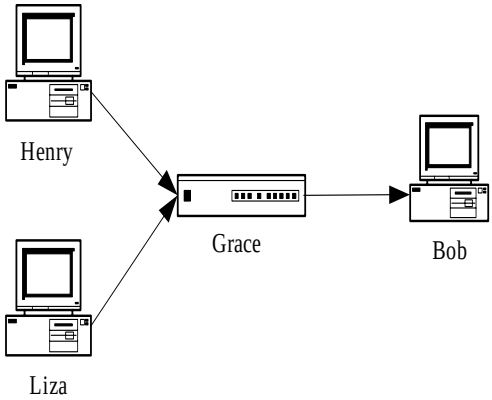


Figure 3. The experimental environment

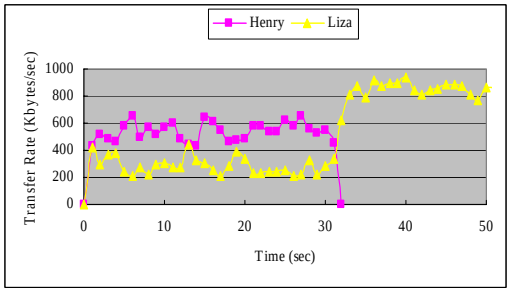


Figure 4. The transfer rate of Henry and Liza (After 32 second, Henry has transferred all the testing data)

Fig. 4 shows the transfer rate of Henry and Liza. The average transfer rate of Henry's flow is 512Kbytes/sec which meets the desired target rate (i.e., 500Kbytes/sec). Therefore it shows the effectiveness of STPPE.

B. overhead of Linux Divert Sockets

The interception and re-injection of IP-packets, which is the essential part of the STPPE, induces an overhead. To evaluate the influence of Linux Divert Sockets, the average delay time with variable packet size is estimated. As Fig. 5 shows, the Host A sends echo request to Host B, and waits for the echo response. Each echo response would be redirected by Linux Divert Sockets for packet marking before transmission. Then the round trip time (RTT) is measured in Host A to compare the time difference of echo responses with and without redirection.

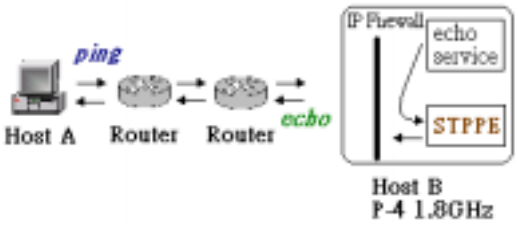


Figure 5. Overhead Testbed

Table 1. Divert socket average delay time

ICMP Packet Size (bytes)	RTT Without marking (ms)	RTT with marking (ms)	Time Difference (ms)	Time Difference / RTT without marking
56	0.975	0.981	0.006	0.61538%
512	2.579	2.587	0.008	0.31021%
1024	4.344	4.356	0.012	0.27614%
2048	6.849	6.864	0.015	0.21901%

Table 1 shows the experimental result. From the ratio of Time Difference to RTT without marking, the overhead of Linux Divert Sockets for packet processing can almost be neglected compared to the delay of packet transmission.

IV. Conclusion and Future Work

In this paper, we have proposed a source transparent packet pre-marking engine that can transparently modify the DSCP in the IP header. Hence the existing Internet applications on Linux platform can be pre-marked without modification or recompilation of source codes.

In the future, we will plan to add more adaptive packet pre-marking algorithms in STPPE and test these algorithms on more complicated Diffserv networks. We will also find some methods that can help legacy Microsoft Windows based applications become QoS-aware without any modification to their source codes

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

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

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

[8] B. Nandy, N. Seddigh, P. Pieda, and J. Ethridge, "Intelligent traffic conditioners for assured forwarding based differentiated services networks," IFIP High Performance Networking, June 2000.

[9] Divert Sockets for Linux: http://www.anr.mcnc.org/~divert/index.html

[10] J. Heinanen and R. Guerin, "A two rate three color marker", RFC 2698,September 1999

[11] GRED queuing discipline: http://www.opalsoft.net/qos/DS-27.htm