A Core-Stateless Buffer Management Mechanism for Differentiated Services Internet

Y. Thomas Hou [†*] Dapeng Wu [‡] Jason Yao [†] Takafumi Chujo [†]

[†] Fujitsu Laboratories of America, Sunnyvale, CA, USA
[‡] Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

The IETF differentiated services (DiffServ) framework achieves scalability by moving complexity out of the core of the network into edge routers which process fewer number of flows. Recently, an end-to-end service called the premium service (PS) has been proposed under the DiffServ model to provide coarse grained guaranteed rate service. This paper presents a buffer management mechanism based on simple FIFO scheduling to support integrated transport of the PS and the traditional best effort (BE) service. A key feature in our buffer management is to perform selective packet discarding from an embedded queue at a shared buffer. We show that such a buffer management mechanism is capable of achieving the following objectives: (1) A core router does not maintain any state information for any flow (i.e., stateless); (2) The bandwidth for an PS flow is guaranteed (in conjunction with a bandwidth broker (BB) for admission control). Simulation results demonstrate that our buffer management mechanism can achieve integrated transport of the PS and the BE services.

1. Introduction

The current Internet offers the so-called best effort (BE) service, which does not make any service quality commitment. Such a simple service model allows IP routers to be stateless: except routing state, which is highly aggregated. As the Internet evolves into a global communication infrastructure, there is a growing demand to support more sophisticated service quality than the traditional BE service.

One solution for Internet QoS is to design a stateful architecture (e.g., the integrated services (IntServ) model [3]). Under such an architecture, each router must maintain per-flow state (e.g., per-flow QoS scheduling state and per-flow reservation state). Performing per-flow management inside the network affects both the core network scalability and robustness. Under such an architecture, bandwidth guarantee are typically achieved by using per-flow queueing mechanism such as virtual clock (VC) [23] or weighted fair queueing (WFQ) [8, 18]. However, the scalability issue related to such mechanisms has been questioned.

Another approach to Internet QoS is to maintain the stateless property of the original IP architecture, e.g., the differentiated services (DiffServ) model [2]. DiffServ provides a coarse level of service differentiation with a small number of traffic classes and offers greater scalability than the IntServ architecture. Under the DiffServ approach, all the routers within a DiffServ domain can be identified as an edge router or a core router. Edge routers maintain per-flow state and mark packets passing through them into specific service classes by marking each packet header. Core routers employ simple scheduling and buffer management mechanisms to process packets based on the marking bits in each packet's header.

Under the DiffServ model, a new service, call the *premium service* (PS) [16] was proposed to provide a coarse grained end-to-end service. The PS is designed to provide a guaranteed rate, low loss, and low delay jitter through a DiffServ domain. Such a service is similar to the service provided by a leased line. Consequently, it is called "virtual leased line" service. Example applications using PS service include Internet telephony and video conferencing.

The BE service will remain under the DiffServ architecture and is expected to make up the bulk of the Internet traffic [16].

This paper presents a buffer management mechanism (in conjunction with a bandwidth broker (BB) for admission control) under the DiffServ model to support integrated transport of the PS and the BE services. On the data plane, we propose a *selective packet discarding* mechanism from an *embedded* queue at a shared buffer. On the control plane, we employ a centralized BB for each DiffServ domain and remove per-flow QoS reservation state information away

*Please direct all correspondence to Y.T. Hou, Fujitsu Laboratories of America, 595 Lawrence Expressway, Sunnyvale, CA 94085, USA. Email: thou@fla.fujitsu.com.

from the routers. Under this architecture, we show that our buffer management mechanism is capable of achieving the following two objectives: (1) A core router does not maintain any state information for any flow (i.e., stateless); and (2) The bandwidth for an PS flow is guaranteed (in conjunction with a BB admission control). Simulation results demonstrate that our buffer management mechanism can achieve integrated support of the PS and the BE services.

Gupta *et al.* [12] and Nichols *et al.* [16] proposed class-based queueing (CBQ) with strict priority (SP) or WFQ among the class-based queueing (CBQ) to support multiple service quality. Under this approach, all the PS flows share the same queue while traffic of other classes share other queues. The PS packets will always depart the routers first (under SP) or will be guaranteed a service rate (under WFQ). While CBQ with SP or WFQ is a feasible solution to support integrated traffic of the PS and the BE flows, this paper proposes an alternative solution based on simple FIFO scheduling. Our approach shows that a simple selective packet discarding mechanism based on FIFO scheduling can also support integrated traffic of the PS and the BE services.

The remainder of this paper is organized as follows. In Section 2, we discuss related work on buffer management. Section 3 describes our selective packet discarding mechanism in detail. In Section 4, we discuss how to perform admission control using a BB under our DiffServ architecture. Section 5 uses simulation results to demonstrate the performance of our buffer management mechanism. Section 6 concludes this paper.

2. Related work on buffer management

Traditional technique for managing router queues in the BE Internet is the drop-tail mechanism, which drops the incoming packet when there is not enough free buffer space. A key problem associated with drop-tail is that it can bring about global synchronization among the TCP flows traversing the same node, in which case both link utilization and overall throughput can be significantly reduced [4]. Furthermore, the drop-tail mechanism is unable to offer service differentiation between the PS and the BE traffic.

Random Early Detection, or RED [11], is an active queue management algorithm for routers that resolves the TCP synchronization problem associated with drop tail. In contrast to drop tail, which drops packets only when the buffer is full, the RED algorithm drops arriving packets probabilistically before the buffer is full. More specifically, it computes the average queue size and when the average queue size exceeds a certain threshold, it drops each arriving packet with a certain probability, which is a function of the average queue size. The probability of dropping increases as the estimated average queue size grows. Such randomization in packet dropping keeps TCP connections back off at different times, which avoids the global synchronization effect of all connections and maintains high throughput for TCP traffic. Although RED is a viable solution for BE traffic [4], it is not sufficient to achieve service differentiation among the PS and the BE packets under our DiffServ architecture.

RED has many variants and the variants can be classified into the following two classes depending on whether the algorithm maintains per-flow state information. The first class extends the RED algorithm without any per-flow management [7, 9, 10] while the second class employ per-flow management [1, 15, 17] by using either a fixed size table [17] or per active flow accounting [1, 15]. Since this paper focuses on core-stateless architectures, we will only discuss the first class algorithms, i.e., buffer management algorithms without requiring per-flow management.

Due to paper length limitation, we will only discuss RED with In and Out (RIO) by Clark and Fang [7], which we consider the most important extension of RED algorithm within the first class of algorithms. RIO retains all the attractive features of RED and with the added capability of discriminating against out-of-profile packets during congestion. RIO employs two RED algorithms for dropping packets, one for in-profile packets and one for out-of-profile packets. By appropriately choosing the parameters for the respective thresholds for the in-profile and out-of-profile packets, RIO is able to preferentially drop out-of-profile packets. Although RIO can offer service differentiation among different traffic classes, it is not clear how the PS can be supported under the RIO mechanism.

The so-called pushout (PO) packet discarding mechanism allows an incoming packet to enter the buffer by discarding some other packets in the buffer [6, 20]. Compared to other threshold-based packet discarding mechanisms, pushout offers: (1) better buffer utilization since packet discarding only occurs when the buffer is full; (2) higher reliability to certain higher-priority packets. The problem with PO mechanism is that it does not address how to avoid global synchronization problem associated with TCP traffic.

3. A stateless buffer management mechanism

In this section, we present our stateless buffer management mechanism to support the PS and the BE traffic. Figure 1 shows the flow chart of our proposed buffer management mechanism.

According to Fig. 1, when an PS packet arrives at the node, our node mechanism makes every effort to let it enter the buffer by potentially discard BE packets in the buffer. On the other hand, when a BE packet arrives at the buffer, our node mechanism will let it join the buffer only if there

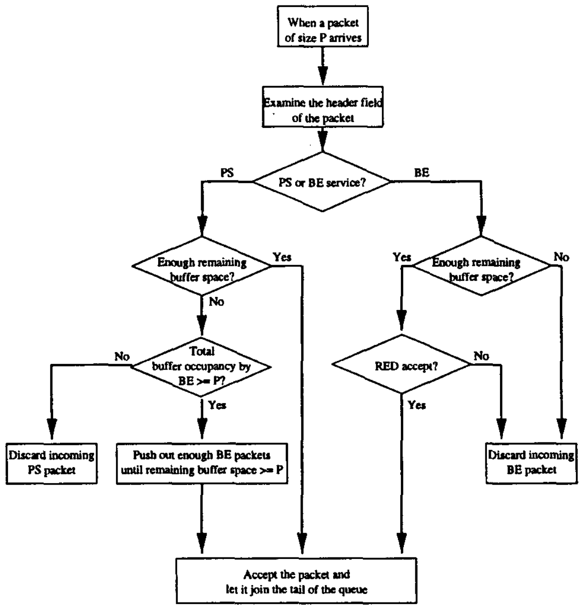

When a packet of size P arrives
Examine the header field of the packet
PS or BE service?
PS
BE
Enough remaining buffer space?
Yes
No
Total buffer occupancy by BE >= P?
Yes
No
Discard incoming PS packet
Push out enough BE packets until remaining buffer space >= P
Enough remaining buffer space?
Yes
No
RED accept?
Yes
No
Discard incoming BE packet
Accept the packet and let it join the tail of the queue

Figure 1. Flow chart of selective packet discarding mechanism.

is enough buffer space *and* RED decides to accept it (with a probability). Therefore, our node mechanism achieves the highest possible loss protection for the PS while providing randomization in packet dropping for the BE packets.

In our implementation, we maintain two variables Q_{BE} and R (both in unit of bytes) at a buffer as follows.

Q_{BE}: is the sum of packet size (in bytes) of all the BE packets in the buffer. It is used to keep track of the buffer occupancy by all the BE packets.

R: is the remaining free buffer space (in bytes).

We maintain the following data structure in the buffer to achieve our selective packet discarding mechanism. Each data unit in the buffer consists of a physical IP packet and three pointers, of which two pointers are used for doubly linked list L_{Total} and the third is used for linked list L_{BE}.

Linked list L_{Total} is an FIFO-like doubly linked list of all packets (both the PS and the BE services) in the buffer. L_{Total} is updated whenever an incoming packet joins the tail of the queue or a packet is served at the front of the queue by the output link or pushed out in the middle of the queue.

Linked list L_{BE} is the linked list of the BE packets *embedded* in the linked list L_{Total}. L_{BE} is updated whenever an incoming BE packet joins the tail of the queue or a BE packet is either served by the output link *or* discarded by our node mechanism.

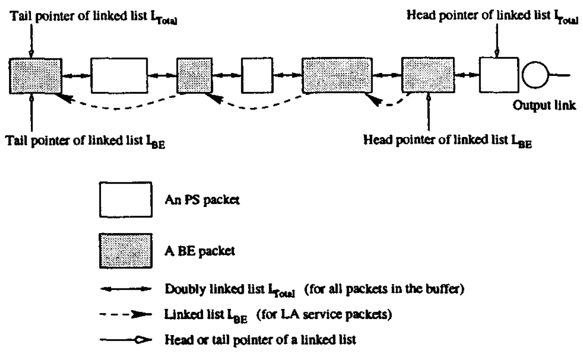

Tail pointer of linked list L_{Total}
Head pointer of linked list L_{Total}
Output link
Tail pointer of linked list L_{BE}
Head pointer of linked list L_{BE}
An PS packet
A BE packet
Doubly linked list L_{Total} (for all packets in the buffer)
Linked list L_{BE} (for LA service packets)
Head or tail pointer of a linked list

Figure 2. Linked list data structure for selective packet discarding mechanism. Linked list L_{BE} is embedded in L_{Total}.

Figure 2 shows the linked list structure for packets in the buffer at a node. Similar to the FIFO queueing mechanism, packets can only be served at the head of linked list L_{Total} and any incoming packet can only join the tail of linked list L_{Total}. A second linked list L_{BE} (embedded in L_{Total}) keeps track of the BE packets in the buffer. In our buffer management mechanism, when an PS packet arrives and the remaining free buffer space cannot accommodate this packet, BE packet(s) will be discarded if such discarding can make sufficient free buffer space to accommodate this incoming PS packet. Should there be enough buffer space for the incoming PS packet after discarding BE packet(s), we discard BE packets from the head of linked list L_{BE} along linked list L_{BE} until there is just enough free buffer space to allow the incoming PS packet to enter the buffer. The reason why we discard BE packets from the head (instead of from the tail) of linked list L_{BE} is that this will make TCP acknowledgment be conveyed to the TCP source earlier than is the case under tail-discarding, which translates into quicker reaction to congestion and considerable performance improvement [14].

Note that a doubly linked list is employed for L_{Total} in Fig. 2. This is because the head of L_{BE} can be anywhere in L_{Total} and only a doubly linked list for L_{Total} can preserve the connectivity of L_{Total} when the packet at the head of L_{BE} is discarded. On the other hand, a singly linked list is sufficient for the BE packets since packet discarding for L_{BE} always takes place at its head.

Figure 3 provides a detailed description of our packet discarding mechanism, with R being initialized to the total buffer space.

```
When a packet of size $P$ arrives at the output port of a switch:
    examine the packet header field;
    if (BE packet) {
        if ($R \geq P$) { /* i.e., sufficient remaining buffer space */
            use RED to decide whether or not to accept the incoming BE packet;
            if (RED accepts the incoming BE packet) {
                let the incoming BE packet join the tail of linked list $L_{Total}$;
                update linked list $L_{Total}$;
                $R := R - P$;
                update linked list $L_{BE}$;
                $Q_{BE} := Q_{BE} + P$;
                }
            else /* i.e., RED does not accept the incoming BE packet */
                discard the incoming BE packet;
            }
        else /* i.e., $R < P$, insufficient remaining buffer space */
            discard the incoming BE packet;
        }
    else /* i.e., PS packet */ {
        if ($R \geq P$) {
            accept the incoming PS packet and let it join the tail of $L_{Total}$;
            update linked list $L_{Total}$;
            $R := R - P$;
            }
        else /* i.e., $R < P$ */ {
            if ($Q_{BE} + R < P$)
            /* i.e., insufficient buffer space even if all BE packets are discarded */
                discard the incoming PS packet;
            else {
            /* i.e., there is enough free buffer space available if some BE packets are discarded */
                discard BE service packets (with a total of $x$ bytes) from the head of linked list $L_{BE}$
                until ($R + x > P$);
                update linked list $L_{BE}$;
                $Q_{BE} := Q_{BE} - x$;     $R := R + x$;
                accept the incoming PS packet and let it join the tail of linked list $L_{Total}$;
                update linked list $L_{Total}$;
                $R := R - P$;
                }
            }
        }

When a packet of size $P$ departs from the head of linked list $L_{Total}$ at the output port of a switch:
    update linked list $L_{Total}$;
    $R := R + P$;
    if (the departing packet belongs to BE service) {
        update linked list $L_{BE}$;
        $Q_{BE} := Q_{BE} - P$;
        }
```

Figure 3. Selective packet discarding algorithm.

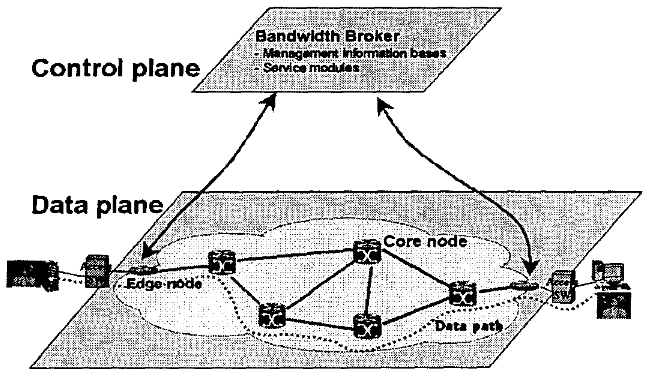


Figure 4. A schematic of bandwidth broker on the control plane to manage resources for an IP network domain.

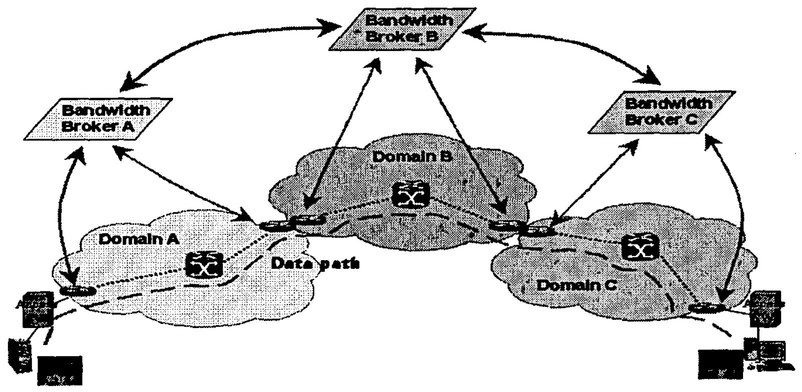


Figure 5. A schematic of multiple bandwidth brokers on the control plane, one for each IP network domain.

4. Admission control for the PS flows

Note that our buffer management mechanism only controls data plane QoS and itself would not be sufficient to support the PS flows without appropriate admission control on the control plane.

We promote to use a centralized bandwidth broker (BB) that maintains topology as well as the some state information of all nodes in the network [16]. A BB is configured with the organizational policies and manages the resources of a DiffServ domain (see Fig. 4).

Admission control is implemented on the BB, eliminating the need for maintaining distributed reservation state. Such a centralized approach is ideal for the PS since such flows are relatively long lived, and set-up and tear-down events are relatively on a much larger time scale than round trip time.

For an PS flow initiates and terminates within a single DiffServ domain, it is straightforward to use a BB to perform admission control. The sender requesting PS first sends a message to the BB indicating such request. The BB makes an admission control decision based on the network topology and traffic volume on the network. If the request is denied, the BB sends a message to the sender. If the request is accepted, the BB will send a message to the edge routers so that they can set up the traffic conditioning functions for the PS flow (e.g., classification, shaping, and marking). Finally, the BB sends a message to the sender and the sender can start transmitting the PS packets.

For an PS flow traversing multiple DiffServ domains, we need to deploy a BB for each domain and coordinate among the BBs to perform admission control (see Fig. 5). More specifically, after receiving a request from the sender, the BB within the same domain will first check to see if there is sufficient network resource to accommodate the PS flow within its own domain. Then it communicates with the BB in the next DiffServ domain (in the direction towards to the receiver) to see if there is sufficient resource to accommodate this flow, and so forth. Only when the BBs at all the DiffServ domains between the sender and receiver have sufficient network resource to accommodate the flow and agree to accept the new PS flow, the BBs will send messages to the edge routers so that appropriate traffic conditioning functions can be set up. Finally, the BB within the sender's domain sends a message to the sender indicating admission or rejection.

Note that the BB still needs to maintain states for the PS flows so that the resources in the network will not be overly subscribed. Work is currently underway to design flow aggregation algorithm to alleviate the storage and processing requirements at BBs.

5. Simulation results

In this section, we implement our selective packet-discarding buffer management mechanism on our network simulator. We perform simulations of integrated traffic

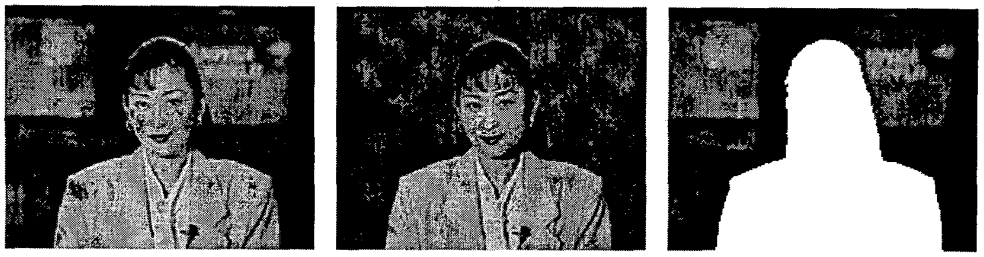


Figure 6. An example of video object (VO) concept in MPEG-4 video. A video plane in (a) is segmented into two VO planes in (b) and (c), where VO1 (b) is the foreground and VO2 (c) is the background.

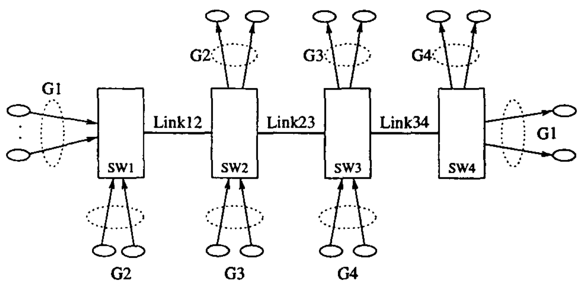


Figure 7. A chain network.

of real-time video streaming[1] application and traditional TCP/UDP traffic under our DiffServ architecture. We use MPEG-4 video as our real-time streaming application and use application level perceptual quality as performance measure. The purpose of our simulation study is to demonstrate that our buffer management mechanism can provide integrated support of the PS and the BE services.

5.1. Simulation settings

The network configuration that we use is the chain network shown in Fig. 7, where path G1 consisting of multiple flows and traverses from the first switch (SW1) to the last switch (SW4), while all the other paths traverse only one hop and "interfere" the flows in G1. In our simulations, G1 consists of one MPEG-4 source, three TCP connections and three UDP connections while G2, G3 and G4 all consist of three TCP connections and three UDP connections, respectively. The link capacity between the switches is 200 kb/s on Link12, Link23, and Link34.

For MPEG-4 video, at the source side, we use the standard raw video sequence "Akiyo" in QCIF format for the MPEG-4 video encoder [21]. The encoder performs MPEG-4 coding described in [5]. The encoded MPEG-4 bit-stream is packetized and classified into the PS and the BE service packets before being sent to the network. In particular, we identify the foreground VO as VO1 (Fig. 6(b)) and the background object as VO2 (Fig. 6(c)). For arriving packets, the receiver extracts the packet content to form the bit-stream for the MPEG-4 decoder. To prevent error propagation due to packet loss, we let the source encoder encode an Intra-VOP every 100 frames [13].

We assume all TCP sources are persistent during the simulation run. For UDP connections, we use an exponentially distributed on/off model with average $E(T_{on})$ and $E(T_{off})$ for on and off periods, respectively. During each on period, the packets are generated at peak rate r_p. The average bit rate for a UDP connection is, therefore, $r_p \cdot \frac{E(T_{on})}{E(T_{on})+E(T_{off})}$.

Table 1 lists the parameters used in our simulation. We use 576 bytes for the path MTU. Therefore, the maximum payload length, MaxPL, for MPEG-4 is 526 bytes (576 bytes minus 50 bytes of overhead) [19].

For the RED mechanism used for the BE service, we use a linear probability function for p_a where $\max\{p_a\} = 0.1$. The parameter w_q is used to calculate the average queue size avg and is set to 0.02 [11]. The min_{th} and max_{th} parameters are set to 5 packets and 15 packets, respectively.

We run our simulation for 450 seconds for all configurations. Since there are only 300 continuous frames in "Akiyo" sequence available, we repeat the video sequence cyclically during the 450-second simulation run.

5.2. Results

We organize our simulation results as follows. As a first case (Case 1), we mark both VO1 and VO2 packet streams under BE service and interact with other TCP/UDP (also marked as BE service). This is the case under the traditional BE Internet where there is no service differentiation among all the packets. We expect to see packet loss from both VO1

[1]Video streaming implies that the content need not be downloaded in full before being played, but is played out while it is being received and decoded [22].

Table 1. Simulation parameters.

End system	MPEG-4	MaxPL	526 bytes
		Aggregate rate	20 kb/s
		VO1 (foreground) rate	13.2 kb/s
		VO2 (background) rate	6.8 kb/s
		Buffer size	1 Mbytes
	TCP	Mean packet processing delay	300 μs
		Packet processing delay variation	10 μs
		Packet size	576 bytes
		Maximum receiver window size	64 Kbytes
		Default timeout	500 ms
		Timer granularity	500 ms
		TCP version	Reno
	UDP	$E(T_{on})$	100 ms
		$E(T_{off})$	150 ms
		r_p	100 kb/s
		Packet size	576 bytes
Switch		Buffer size	10 Kbytes
		Packet processing delay	4 μs
Link	End system to switch	Speed	10 Mb/s
		Distance	1 km
	Inter-switch	Distance	1000 km

and VO2 packet stream. In the second case (Case 2), we mark only VO1 under the PS service and thus is guaranteed with 13.2 kb/s while VO2 is marked under BE service (together with other interfering TCP/UDP traffic). Under this case, we expect that there is no packet loss from VO1 stream while there may be packet loss from VO2 stream. Finally, in Case 3, we mark both VO1 and VO2 under the PS and require 20 kb/s (13.2 + 6.8) guaranteed bandwidth. Under this scenario, we do not expect any packet loss from VO1 and VO2 video stream under our DiffServ architecture. Only TCP/UDP traffic (marked under the BE service) may be subject to loss.

Case 1: BE Service Internet

Under the BE architecture, the packet loss ratio are 2.88% for VO1 and 2.68% for VO2, respectively. Figure 8 shows the peak signal to noise ratio (PSNR) for VO1 and VO2 under the BE architecture. Note that both VO1 and VO2 have wide oscillations of PSNR, which translates into substantial perceptual degradation. To examine the perceptual quality of the MPEG-4 video, we play out the decoded video sequence at the receiver. A sample frame is shown in Fig. 11(a).

Case 2: DiffServ Internet with VO1 under the PS

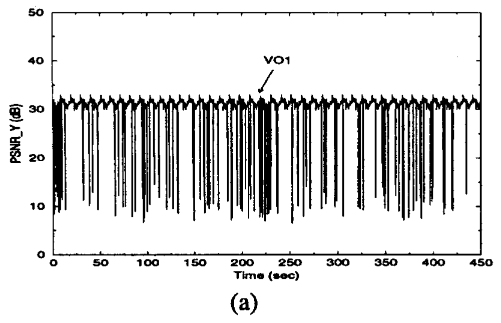
In this case, we find that there is no packet loss for VO1 and the packet loss ratio for VO2 is 3.77% under our DiffServ architecture. Figure 9 shows the PSNR for VO1 and VO2 under the our DiffServ architecture. Note that only VO2 have substantial performance degradation in terms of PSNR while the PSNR for VO1 is excellent, indicating that the PS service offers guaranteed rate for such packets. To examine the perceptual quality of the MPEG-4 video, we play out the decoded video sequence at the receiver. A sample frame is shown in Fig. 11(b).

Case 3: DiffServ Internet with both VO1 and VO2 under the PS

Under our DiffServ architecture, there is no loss for both VO1 and VO2 packets. Figure 10 shows the PSNR for VO1 and VO2 under the DiffServ architecture. Note that only PSNR for VO2 have substantial performance improvement (over that under Case 2) while the PSNR for VO1 is the same as that under Case 2, indicating that the PS offers guaranteed rate for both VO1 and VO2 packets. To examine the perceptual quality of the MPEG-4 video, we play out the decoded video sequence at the receiver. A sample frame is shown in Fig. 11(c).

6. Conclusion

This paper presented a buffer management mechanism under core-stateless DiffServ architecture for integrated transport of the PS and the BE services. A key feature in our buffer management is to perform selective packet discarding from an embedded queue under a simple FIFO scheduler. We showed that such buffer management mechanism is capable of achieving the following objectives: (1) A core router does not maintain any state information for any flow (i.e., stateless); (2) The bandwidth for an PS flow is always guaranteed (in conjunction with the BB's admission control). Simulation results demonstrated that our buffer management mechanism can achieve integrated support of the PS and the BE services.



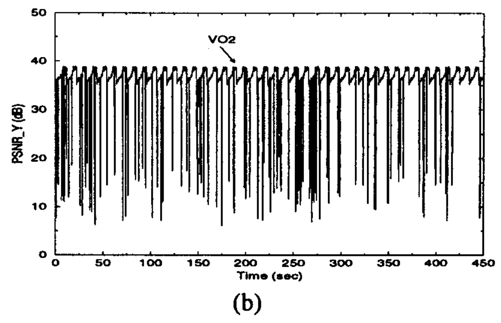
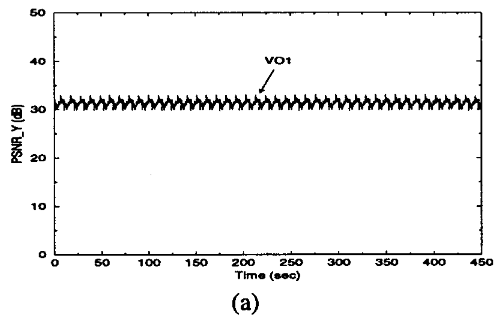


Figure 8. PSNR of (a) VO1 and (b) VO2 at the receiver under Case 1.



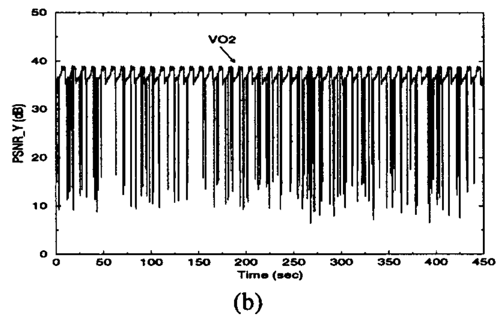
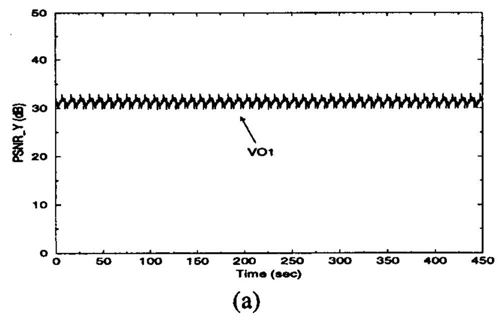


Figure 9. PSNR of (a) VO1 and (b) VO2 at the receiver under Case 2.



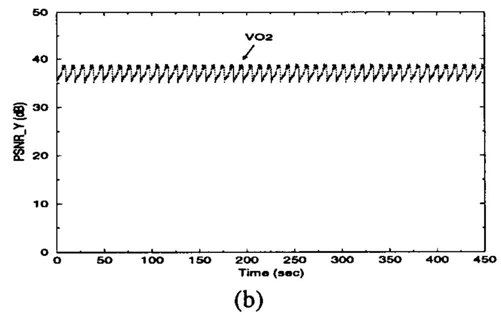


Figure 10. PSNR of (a) VO1 and (b) VO2 at the receiver under Case 3.

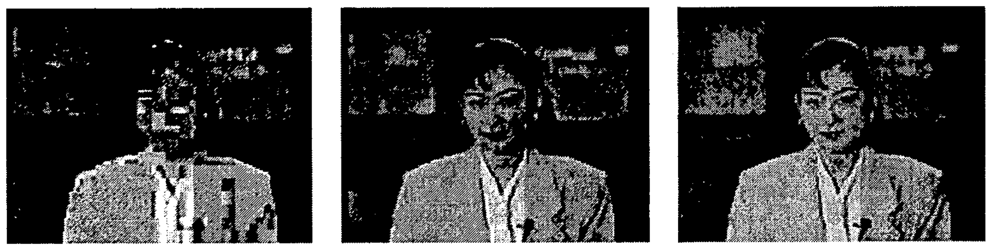

(a) (b) (c)

Figure 11. Sample frames at the receiver. (a) Case 1: both VO1 and VO2 under the BE service; (b) Case 2: VO1 under the PS but VO2 under the BE service; and (c) Case 3: both VO1 and VO2 under the PS.

References

[1] F.M. Anjum and L. Tassiulas, "Fair bandwidth sharing among adaptive and non-adaptive flows in the Internet," in *Proc. IEEE INFOCOM,* pp. 1412–1420, March 1999, New York, NY.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *RFC 2475,* Internet Engineering Task Force, Dec. 1998.

[3] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," *RFC 1633,* Internet Engineering Task Force, July 1994.

[4] B. Braden, D. Black, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the Internet," *RFC 2309,* Internet Engineering Task Force, April 1998.

[5] T. Chiang and Y.-Q. Zhang, "A new rate control scheme using quadratic rate distortion model," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 7, no. 1, pp. 246–250, Feb. 1997.

[6] I. Cidon, L. Georgiadis, R. Guerin, and A. Khamisy, "Optimal buffer sharing," *IEEE Journal on Selected Areas in Communications,* vol. 13, no. 7, pp. 1229–1240, Sept. 1995.

[7] D.D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Trans. on Networking,* vol. 6, no. 4, pp. 362–373, Aug. 1998.

[8] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulations of a fair queueing algorithm," in *Proc. ACM SIGCOMM'89,* Austin, TX, 1989, pp. 1–12.

[9] W. Feng, D. Kandlur, D. Saha, and K.G. Shin, "Adaptive packet marking for providing differentiated services in the Internet," in *Proc. IEEE International Conference on Network Protocols (ICNP'98),* October 1998.

[10] W. Feng, D.D. Kandlur, D. Saha, and K.G. Shin, "BLUE: A new class of active queue management algorithms," *Tech. Report CSE-TR-387-99,* Univ. of Michigan, April 1999.

[11] S. Floyd and V. Jacobson, "On random early detection gateways for congestion avoidance," *IEEE/ACM Trans. on Networking,* vol. 1, no. 4, pp. 397–413, Aug. 1993.

[12] A. Gupta, D. Stahl, and A. Whinston, "Priority pricing of integrated services networks," in *Internet Economics,* L. McKnight and J. Bailey, Eds., MIT Press, Cambridge, MA. 1997, pp. 253–279.

[13] ISO/IEC JTC 1/SC 29/WG 11, "Information technology - coding of audio-visual objects, part 1: systems, part 2: visual, part 3: audio," FCD 14496, Dec. 1998.

[14] T. V. Lakshman, A. Neidhardt, and T. J. Ott, "The drop from front strategy in TCP and in TCP over ATM," in *Proc. IEEE INFOCOM'96,* pp. 1242–1250, San Francisco, CA, March 1996.

[15] D. Lin and R. Morris, "Dynamics of random early detection," in *Proc. ACM SIGCOMM'97,* Sept. 1997, Cannes, France.

[16] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," *RFC 2638* Internet Engineering Task Force, July 1999.

[17] T.J. Ott, T.V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proc. IEEE INFOCOM,* pp. 1346–1355, March 1999, New York, NY.

[18] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control — the single node case," in *Proc. IEEE INFOCOM'92,* pp. 915–924, May 1992.

[19] H. Schulzrinne, D. Hoffman, M. Speer, R. Civanlar, A. Basso, V. Balabanian, and C. Herpel, "RTP payload format for MPEG-4 elementary streams," *Internet Draft,* Internet Engineering Task Force, March 1998, work in progress.

[20] L. Tassiulas, Y.C. Hung, and S.S. Panwar, "Optimal buffer control during congestion in an ATM network node," *IEEE/ACM Trans. on Networking,* vol. 2, no. 4, pp. 374–386, Aug. 1994.

[21] D. Wu, Y.T. Hou, W. Zhu, H.-J. Lee, T. Chiang, Y.-Q. Zhang, and H.J. Chao, "On end-to-end architecture for transporting MPEG-4 video over the Internet," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 10, no. 6, Sept. 2000.

[22] D. Wu, Y.T. Hou, and Y.-Q. Zhang, "Transporting real-time video over the Internet: challenges and approaches," *Proceedings of the IEEE,* vol. 88, no. 12, Dec. 2000.

[23] L. Zhang, "VirtualClock: A new traffic control algorithm for packet switching networks," *ACM Trans. Computer Systems,* vol. 9, pp. 101–124, May 1991.