ELSEVIER

Computer Networks 32 (2000) 185–209

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

A differentiated services architecture for multimedia streaming in next generation Internet

Yiwei Thomas Hou [a,*], Dapeng Wu [b], Bo Li [c], Takeo Hamada [a], Ishfaq Ahmad [c], H. Jonathan Chao [b]

[a] *Fujitsu Laboratories of America, 595 Lawrence Expressway, Sunnyvale, CA 94086-3922, USA*
[b] *Polytechnic University, Brooklyn, NY, USA*
[c] *Hong Kong University of Science and Technology, Kowloon, Hong Kong, People's Republic of China*

Abstract

This paper presents a Differentiated Services (Diffserv or DS) architecture for multimedia streaming applications. Specifically, we define two types of services in the context of Assured Forwarding (AF) per hop behavior (PHB) that are differentiated in terms of reliability of packet delivery: the *High Reliable* (HR) service and the *Less Assured* (LA) service. We propose a novel node mechanism called Selective Pushout with Random Early Detection (SPRED) that is capable of simultaneously achieving the following four objectives: (1) a core router does not maintain any state information for each flow (i.e., core-stateless); (2) the packet sequence within each flow is not re-ordered at a node; (3) packets from HR service are delivered more reliably than packets from LA service at a node during congestion; and (4) packets from TCP traffic are dropped randomly to avoid global synchronization during congestion. We show that SPRED is a generalized buffer management algorithm of both tail-dropping and Random Early Detection (RED), and combines the best features of pushout (PO), RED and RED with In/Out (RIO) mechanisms. Simulation results demonstrate that under the same link speed and network topology, network nodes employing our Diffserv architecture have substantial performance improvement over the current Best Effort (BE) Internet architecture for multimedia streaming applications. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Differentiated services; Per hop behavior; Scalability; Best effort service; Buffer management; Multimedia streaming; Next generation Internet

1. Introduction

Over the past several years, as the speed of computer increases and multimedia applications proliferate, there is an increasing demand for streaming multimedia applications over the Internet. However, the current Internet only offers the so-called Best Effort (BE) service, which does not make any service quality commitment. Since many streaming applications require better-than-BE delivery, the current Internet is becoming increasingly inadequate to support the service demand from multimedia streaming applications.

[*] Corresponding author. Tel.:+1-408-530-4529; fax: +1-408-530-4515.
E-mail address: thou@fla.fujitsu.com (Y.T. Hou).

Recently, the Internet Engineering Task Force (IETF) has specified the Differentiated Services (Diffserv or DS) framework for the next generation Internet [3,19]. The Diffserv architecture offers a framework within which service providers can offer each customer a range of network services differentiated on the basis of performance. Once properly designed, a Diffserv architecture can offer great flexibility and scalability, as well as meeting the service requirements for multimedia streaming applications.

The IETF Diffserv working group has specified the Assured Forwarding (AF) per hop behavior (PHB) [14]. The AF PHB is intended to provide different levels of forwarding assurances for IP packets at a node, and therefore, can be used to implement multiple priority service classes.

This paper presents a Diffserv implementation architecture, in the context of AF PHB, with the aim of providing different levels of reliability in terms of packet delivery over the Internet. Our Diffserv architecture is targeted at integrated support for both real-time streaming applications and traditional data applications, e.g., TCP-based applications such as file transfer, email, and web browsing. Under our Diffserv architecture, we define two types of services, namely, the *High Reliable* (HR) service and the *Less Assured* (LA) service. The HR service is intended for certain high priority traffic in real-time streaming applications (e.g., foreground video object (VO) and system information in MPEG-4 video [1]) while LA service is for low priority traffic in real-time streaming applications (e.g., background VO in MPEG-4 video) and traditional TCP applications. Packets under HR service are considered critical to overall perceptual quality for a multimedia streaming application and should be delivered as reliably as possible. On the other hand, packets under LA service either have less impact on the perceptual quality (if they belong to real-time streaming applications) or can be retransmitted (if they are traditional TCP-type applications).

We propose a node mechanism, called Selective Pushout with Random Early Detection (SPRED), to perform packet discarding during network congestion and achieve our Diffserv AF PHB. By employing a single shared queue and storing and serving packets in the queue in the order of their arrival, SPRED does not introduce any packet re-ordering at the node. SPRED performs selective packet discarding from an *embedded* queue at a shared buffer and does not maintain any state information for each flow. For HR service, when network is congested and buffer is full, SPRED selectively pushes out LA packets in the buffer to make room for the incoming HR packets. Thus, SPRED offers more reliable delivery for HR service than RED/RIO. For LA service, SPRED employs RED to resolve the global TCP synchronization problems. Our proposed SPRED node mechanism is capable of achieving the following four objectives simultaneously:

Objective 1: A core router does not maintain any state information for each flow (i.e., core-stateless).

Objective 2: The packet sequence within each flow should not be altered at a node.

Objective 3: Packets from HR service should be delivered as reliably as possible.

Objective 4: Packets from TCP traffic should be dropped randomly during congestion to avoid global synchronization.

We show that SPRED is a generalized buffer management algorithm of both tail-dropping and RED. Furthermore, SPRED combines the best features of pushout (PO) [7,28], Random Early Detection (RED) [10], and RED with In/Out (RIO) [8]. Simulation results show that under the same link speed and network topology, network nodes employing our Diffserv/SPRED architecture have substantial application level performance improvement (in terms of perceptual quality) over the current BE Internet architecture for multimedia streaming applications.

Prior efforts on service differentiation include the class-based priority scheduling [13,18]. Such schemes create service classes of different priorities to serve users with different needs. Higher priority packets always depart the routers first. Thus, the effect of priority queueing is to build up a queue of lower priority packets,

[1] We will use MPEG-4 video as an example video streaming application in our simulation study (see Section 2.2).

which will cause packets in this class to be preferentially dropped due to queue overflow. This scheme might be a useful building block for explicit service discrimination among flows, each of which consists packets of the same priority class. But for a flow consisting of both high and low priority packets, out-of-sequence problem will arise if we put packets of different priority from the same flow into different queue and use priority scheduling. Since IETF Diffserv working group explicitly states that it is important that the network does not re-order packets belonging to the same flow [19], separate queueing cannot hereby be employed and we only focus on mechanisms that handle all packets stored and serviced in the same queue. [2]

The remainder of this paper is organized as follows. Section 2.1 gives an overview of the core-statelss Diffserv architecture. Section 2.2 describes multimedia streaming applications using MPEG-4 as an example. In Section 3, we explain our Diffserv architecture in detail and describe the SPRED mechanism to achieve the four design objectives. Section 4 uses simulation results to demonstrate the performance of our Diffserv architecture in supporting multimedia streaming applications. Section 5 concludes this paper.

2. Background

In this section, we provide essential background on core-stateless Diffserv architecture and multimedia streaming to set the stage for later parts of the paper.

2.1. Architecture of core-stateless Diffserv Internet

Our core-stateless Diffserv architecture is based on the following simple model. We identify all the routers within a Diffserv domain and distinguish them between the edge and core routers. Edge routers maintain per flow state; they perform traffic classification and conditioning (marking, policing, and shaping) on each flow. Core routers maintain no per flow state; they use simple scheduling and buffer management for aggregated traffic flows. We call this approach core-stateless Diffserv since the core routers keep no per flow state.

More specifically, a customer maintains a Service Level Agreement (SLA) with its network provider. Based on the SLA, the edge routers perform traffic conditioning functions and assign each packet with a DS codepoint (DSCP) [19]. This value specifies the PHB to be allotted to the packet within the provider's network. Within the core routers inside the network, packets are forwarded according to the PHB associated with the DSCP. PHBs are defined to permit a reasonably granular means of allocating buffer and bandwidth resources at each node among competing traffic streams.

A salient feature of Diffserv framework is its scalability, which allows it to be deployed in very large networks. This scalability is achieved by forcing much complexity out of the core of the network into boundary devices which process smaller volumes of traffic and less number of flows, and by offering services for aggregated traffic rather than on a per flow basis.

A Diffserv architecture can be specified by defining or implementing the following four components:
1. the services provided to a traffic aggregate,
2. the traffic conditioning functions and PHBs used to realize the services,
3. the DSCP used to mark packets under a particular PHB,
4. the particular node mechanism to realize a PHB.

[2] Packet re-ordering can results in jitter in real-time traffic and performance degradation in TCP.

2.2. Multimedia streaming with MPEG-4

Multimedia streaming implies that the content needs not be downloaded in full before it begins playing, but is played out while it is being received and decoded. We choose to use the new international standard, MPEG-4, as a representative multimedia streaming application since MPEG-4 is poised to become the enabling technology for multimedia communications in the next millennium [15]. MPEG-4 builds on elements from several successful technologies such as digital video, computer graphics, and the World Wide Web with the aim of providing powerful tools in the production, distribution, and display of multimedia contents with unprecedented new features and functions. MPEG-4 provides extreme flexibility and efficiency by coding a new form of data called audio-visual object (AVO) (see Fig. 1 for an example of VOs in a video plane). It is foreseen that MPEG-4 will be capable of addressing the emerging truly interactive content-based video services as well as conventional video storage and transmission.

This paper focuses on designing a Diffserv architecture with the aim of providing significant performance improvement over the current BE architecture for multimedia streaming applications. For illustration purpose, we will only discuss the video component of MPEG-4. As it will soon become clear that our Diffserv architecture and SPRED node mechanism discussed in Section 3 are equally applicable to other forms of multimedia streaming (e.g., audio). Such generality is possible due to fact that our Diffserv architecture is designed to offer generic service differentiation (i.e., HR and LA services) regardless the characteristics of the particular streaming application.

For streaming MPEG-4 video over the Internet, on the sender side, raw bit-stream of live video is encoded by an MPEG-4 encoder. After this stage, the compressed video bit-stream is first packetized at the sync layer and then passed through the RTP/UDP/IP layers before entering the Internet. Packets may be dropped at a router/switch due to congestion. For packets that are successfully delivered to the destination, they first pass through the RTP/UDP/IP layers in reverse order before being decoded at the MPEG-4 decoder.

Fig. 2 shows the protocol stack for MPEG-4 video streaming [29]. The right half of Fig. 2 shows the processing stages at an end system. At the sending side, the compression layer compresses the visual information and generates elementary streams (ESs), which contain the coded representation of the VOs. The ESs are packetized as SL-packetized (SyncLayer-packetized) streams at the sync layer [29]. The SL-packetized streams provide timing and synchronization information, as well as fragmentation and random access information. The SL-packetized streams are multiplexed into a FlexMux stream at the TransMux Layer, which is then passed to the transport protocol stacks composed of RTP, UDP and IP. The resulting IP packets are transported over the Internet. At the receiver side, the video stream is processed in the reversed manner before its presentation. The left half of Fig. 2 shows the data format at each layer.

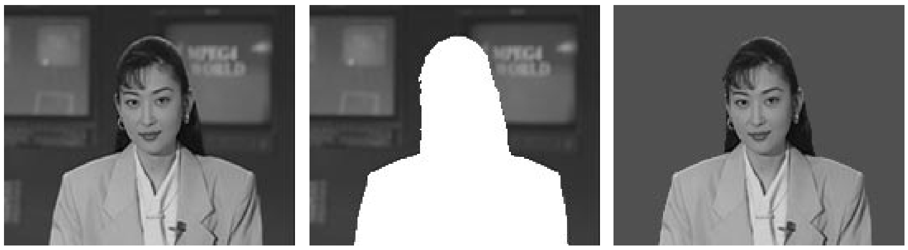


Fig. 1. An example of VO concept in MPEG-4 video. A video plane (left) is segmented into two VO planes where VO1 (middle) is the background and VO2 (right) is the foreground.

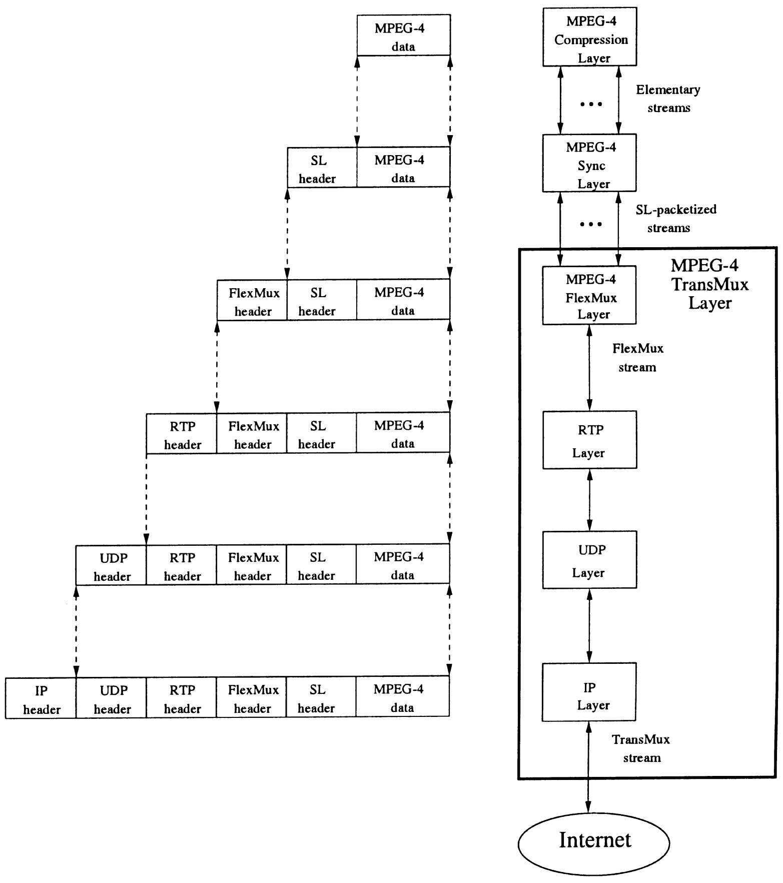


Fig. 2. Data format at each processing layer at an end system.

A key requirement for Internet video streaming is the reliable transport of certain critical information (e.g., system information, header information) at all times. Such information is considered critical for decoding at the receiver side to maintain satisfactory perceptual quality. The BE service model of today's Internet is not able to offer such reliable real-time delivery since there is no service differentiation among all the packets at a node. Thus, it is essential to design a Diffserv architecture for the next generation Internet that is capable of offering service differentiation to user traffic and providing application level performance improvement (i.e., perceptual quality) over the current BE service Internet for multimedia streaming applications.

3. An implementation architecture

We organize this section as follows. In Section 3.1, we define the services, PHB and DS codepoint for our Diffserv architecture. Section 3.2 presents the SPRED node mechanism to achieve our Diffserv PHB, which is the main contribution of this paper. In Section 3.3.5, we discuss extensions of our Diffserv architecture.

3.1. Services, PHB, and DS codepoint definitions

We define two types of services in the context of AF for our Diffserv architecture, namely, the HR service and the LA service. Packets under HR service are considered critical to overall perceptual quality at receiver for streaming application and should be delivered as reliably as possible. On the other hand, packets under LA service either have less impact on the application level perceptual quality (if they belong to multimedia streaming applications) or can be retransmitted (if they are traditional data applications).

We assume that end hosts are capable of marking packets into HR and LA services since they have complete knowledge about the source applications. There can be different mix of HR and LA packets even within the same flow. We also assume that all the edge routers have traffic conditioning functions (i.e., marking, shaping, and policing). At the core routers inside the Diffserv domain, we do not separate traffic from different users into different queues. As discussed in Section 1, class-based queueing with priority scheduling such as [13,18] cannot be employed since packets within the same application flow but of different priority classes may be put into different queues and are served out of sequence (violating Objective 2). With such consideration, we aggregate the packets of all users into one shared queue and packets are served in the order of their arrival, just as today's Internet. Unlike the current BE Internet, the PHB and node mechanism under our Diffserv architecture offers service differentiation in terms of delivery reliability to HR and LA packets.

We first define the PHB of our Diffserv architecture as follows.

Definition 1. (PHB). Packets from HR service should experience lower loss ratio than packets from LA service at a node during congestion. An incoming HR packet shall not be discarded if there are LA packets in the buffer and discarding of such LA packets can leave enough buffer space for the incoming HR packet.

According to the above PHB definition, HR packets have exclusive buffer access and are not interfered by LA packets when the buffer is full. Therefore, our PHB provides the highest possible reliability to HR packets during congestion.

It is straightforward to match our PHB with a DSCP in the IP header. As an example, we may use AF11 = '001010' and AF21 = '010010' under AF PHB for HR and LA services, respectively [14].

3.2. Node mechanism

As discussed previously, we will employ a common shared queueing architecture for all traffic streams at a node to achieve scalability and maintain packet sequence. Under such architecture, an arriving packet may be allowed to enter the buffer only when there is enough remaining buffer space. Otherwise, we have to either discard the incoming packet or discard some other packet(s) in the buffer in order to make room for the incoming packet.

In the following, we first give a brief summary of current existing node mechanisms under a shared single queueing architecture. We find that none of these mechanisms are able to meet all four design objectives (see Section 1) simultaneously. Then we present our SPRED node mechanism, which is capable of meeting all four design objectives.

3.2.1. Previous work

Buffer management mechanisms under a single queueing architecture can be categorized into 'stateful' or 'stateless' node mechanism. Stateful mechanisms such as Flow RED (FRED) [17], Balanced RED (BRED) [1] and Stabilized RED (SRED) [20] all require per-active-flow accounting. Since these node mechanisms require to maintain state information for a flow, they do not meet the first design objective (i.e., core-stateless). In the following, we only discuss node mechanisms that do not require any state information for each flow.

The traditional technique for managing router queue in the BE Internet is the so-called 'tail-dropping' mechanism, which drops the incoming packet when there is not enough remaining buffer space. A key problem associated with tail-dropping is that it can bring about global synchronization among TCP flows traversing the same node, in which case both link utilization and overall throughput can be significantly reduced (violates Objective 4) [5]. Furthermore, the tail-dropping mechanism is unable to offer service differentiation under our PHB (violates Objective 3).

RED is an active queue management algorithm for routers that resolves the TCP synchronization problem associated with tail-dropping [10]. In contrast to tail-dropping, 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. This avoids the global synchronization effect of all connections and maintains high throughout for TCP traffic in the routers. Although RED is a viable solution for traditional data traffic [5], it is not sufficient to achieve service differentiation (HR and LA services) among the packets that is essential for multimedia streaming applications. That is, RED is unable to meet Objective 3.

In [8], a dropping mechanism called RIO was proposed to perform preferential dropping of out-of-profile packets over in-profile packets. 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 ins and one for outs. By choosing the parameters for respective algorithms differently, RIO is able to preferentially drop out-of-profile packets.

RIO is able to offer service differentiation between HR and LA services, if we treat HR as in-profile and LA as out-of-profile and set the two RED algorithms for them such that LA packets are dropped more aggressively than HR packets. But under our Diffserv architecture, HR packets are primarily from real-time streaming applications (instead of TCP) and these packets should be delivered as reliably as possible. In particular, HR packets should not be dropped before buffer is full (as in RIO). Furthermore, according to our PHB definition (Definition 1), should the network be congested and buffer is full, an incoming HR packet should still be allowed to enter the buffer by discarding some LA packets in the buffer (if there is any). However, such high reliability for HR packet delivery, as defined by our PHB, is not achievable under the RIO mechanism since RIO also drops packets with high priority before the buffer is full.

The so-called pushout (PO) packet discarding mechanism allows an incoming packet to enter the buffer by discarding some other packets in the buffer [7,28]. Compared to other threshold-based packet discarding mechanisms, pushout offers: (1) better buffer utilization since packet discarding only occurs when there is insufficient remaining buffer space to store an incoming packet; (2) higher reliability to certain incoming packets of high priority. The problem with PO mechanism is that it does not address how to avoid global synchronization problem associated with TCP traffic, i.e., unable to meet Objective 4.

3.2.2. SPRED node mechanism

To achieve the four design objectives and the PHB under our Diffserv architecture, we present a node mechanism called SPRED. Fig. 3 shows the flow chart of the SPRED mechanism. According to Fig. 3, when an HR packet arrives at the node, SPRED makes every effort to let it enter the buffer by potentially pushout LA packets in the buffer. On the other hand, when an LA packet arrives at the buffer, SPRED will let it join the buffer only if there is enough buffer space and RED decides to accept it (with a probability).

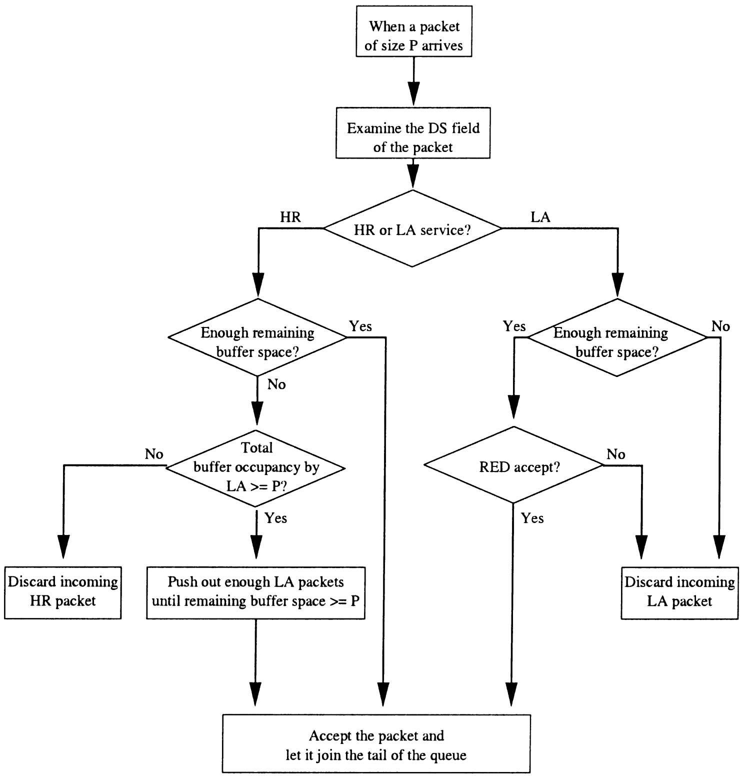

When a packet of size P arrives
Examine the DS field of the packet
HR or LA service?
HR
LA
Enough remaining buffer space?
Yes
No
Total buffer occupancy by LA >= P?
Yes
No
Discard incoming HR packet
Push out enough LA packets until remaining buffer space >= P
Enough remaining buffer space?
Yes
No
RED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

Fig. 3. Flow chart of SPRED node mechanism.

Therefore, SPRED achieves the highest possible loss protection for HR service (Objective 3) while resolving global synchronization problem associated with TCP traffic (Objective 4). [3]

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

- Q_{LA} is the sum of packet size (in bytes) of all LA service packets in the buffer. It is used to keep track of the buffer occupancy by all the LA 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_{LA} as follows:

- *Linked list*, L_{Total}, is an FIFO-like doubly linked list of all packets (both HR and LA 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.
- *Linked list*, L_{LA}, is the linked list of LA service packets *embedded* in the linked list L_{Total}. L_{LA} is updated whenever an incoming LA service packet joins the tail of the queue or an LA service packet is either served by the output link *or* discarded by pushout mechanism.

[3] We implicitly define all TCP traffic under the LA service in our Diffserv architecture. If a TCP connection requires some other type of service, we may put such TCP connection under other node mechanism to meet its service requirement (see Section 3.3.5).

Fig. 4 shows the linked list structure for packets in the buffer at a node. Similar to 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_{LA} (embedded in L_{Total}) keeps track of the LA service packets in the buffer. In our SPRED mechanism, when an HR service packet arrives and the remaining free buffer space cannot accommodate such packet, LA service packet(s) will be discarded if such discarding can make sufficient free buffer space to accommodate this incoming HR service packet. Should there be enough buffer space for the incoming HR service packet after discarding LA service packet(s), we discard LA service packets from the head of linked list L_{LA} along linked list L_{LA} until there is just enough free buffer space to allow the incoming HR service packet to enter the buffer. The reason why we discard LA packets from the head (instead of from the tail) of linked list L_{LA} is that this will make TCP acknowledgment to 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 [16].

Note that a doubly linked list is employed for L_{Total} in Fig. 4. This is because the head of L_{LA} is identified by a pointer and can be anywhere in L_{Total}. Since packet discarding starts with the packet pointed by this pointer, only a doubly linked list for L_{Total} can keep track of the packet immediately preceding the packet subject to discarding in the linked list L_{Total}. That is, only a doubly linked list for L_{Total} can preserve the connectivity of L_{Total} when the packet at the head of L_{LA} is discarded. On the other hand, a singly linked list is sufficient for LA packets since packet discarding for L_{LA} always takes place at its head.

Remark 1. We point out that our SPRED mechanism generalizes both tail-dropping and RED node mechanisms. To see this, let an incoming packet be with probability p of HR service and $1 - p$ of LA service. When $p = 1$, i.e., all packets are of HR service, the SPRED simply behaves like a tail-dropping mechanism since there is no LA packets to be pushed out. When $p = 0$, i.e., all packets are of LA service, the SPRED becomes RED. When $0 < p < 1$, which is the most common case of practical interest, HR packets have complete access of buffer and have better loss protection than RED since there is no dropping for HR packets before buffer is full, while LA packets are subject to both being pushout by an incoming HR packet when buffer is full *and* random dropping by RED before buffer is full.

The following algorithm provides detailed description of the SPRED node mechanism, with R being initialized to the total buffer space.

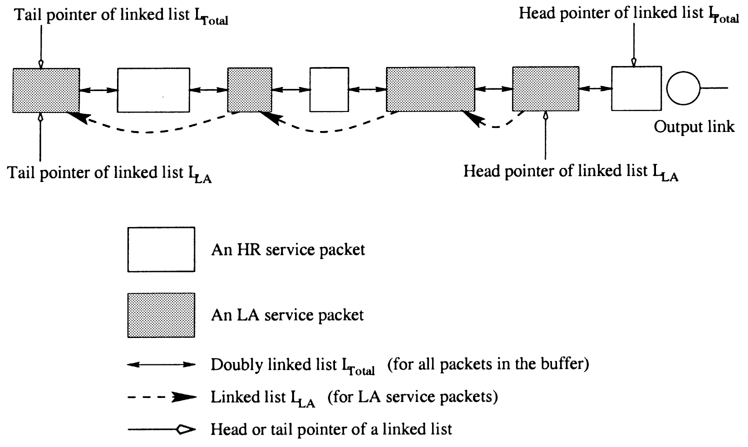

Tail pointer of linked list L_Total
Head pointer of linked list L_Total
Output link
Tail pointer of linked list L_LA
Head pointer of linked list L_LA
An HR service packet
An LA service packet
Doubly linked list L_Total (for all packets in the buffer)
Linked list L_LA (for LA service packets)
Head or tail pointer of a linked list

Fig. 4. Linked list data structure for selective packet discarding under SPRED node mechanism. Linked list L_{LA} is embedded in L_{Total}.

Algorithm 1. Node mechanism with SPRED

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

```
if (the departing packet belongs to LA service) {
  update linked list $L_{LA}$;
  $Q_{LA} := Q_{LA} - P$;
  }
```

3.3. Discussions

3.3.1. Implementation consideration

We would like to point out that it is entirely feasible to implement our SPRED mechanism in hardware for a router. Since the largest IP packet size is 1500 bytes and the smallest is 64 bytes (under Ethernet), in the worst-case, the incoming packet with the largest packet size will pushout at most 24 packets with the smallest packet size. Unlike ATM where there is a cycle time constraint (e.g., 2.83 μs for OC-3), there is no such cycle time for an IP router and the processing time of a packet is basically proportional to the duration of the packet. The longer the packet, the more time there will be available to do pushout. Therefore, our pushout scheme will not have a timing constraint bottleneck in IP switch hardware implementation.

3.3.2. Deployment issue

Unlike Integrated Services (Intserv) framework [4,23], where per flow based QoS guarantees require universal deployment of a node mechanism (e.g., weighted fair queueing (WFQ) [9,21] and its many variants [2,12,24–27,30]) for all routers, there is no such requirement for deploying our Diffserv architecture over the Internet. An incremental deployment of our Diffserv architecture can still have clear benefits to multimedia steaming applications, since the approach for Diffserv architecture is for per hop qualitative service differentiation, not end-to-end quantitative QoS guarantee.

3.3.3. QoS performance

QoS under Diffserv can be defined either quantitatively or qualitatively. This paper follows a qualitative QoS approach to implement Diffserv architecture. Furthermore, the proposed Diffserv architecture focued only on the delivery reliability, not the delay constraint. This is because for real-time streaming applications, the complication associated with delay can be easily dealt with by adding playout buffer at the receiver side to absorb the potential delay variation (e.g., jitter) in the network.

3.3.4. Resource provisioning

Under our Diffserv architecture, TCP traffic is placed under LA service and HR has strictly higher priority over LA, there exists a potential starvation for TCP traffic under heavy load condition. To resolve this problem, appropriate resource control mechanism must be in place in order to limit the total amount of HR service traffic in the network and to provide reasonable amount of network resource for LA service. This paper focuses only on the data plane QoS mechanism (i.e., SPRED) and leaves the detailed mechanism on control plane for future study.

3.3.5. SPRED as a Diffserv module

Our Diffserv architecture focuses on reliable transport of multimedia streaming applications. As discussed in [3], it is likely that more than one PHB group may be implemented on a node. PHB groups are defined such that the proper resource allocation between groups can be inferred, and integrated mechanisms can be implemented which can simultaneously support two or more groups [3]. Our PHB and the SPRED mechanism can be employed as a building block at a node for a more sophisticated Diffserv architecture offering a broader range of services (or PHBs). Fig. 5 illustrates that SPRED is used as a Diffserv

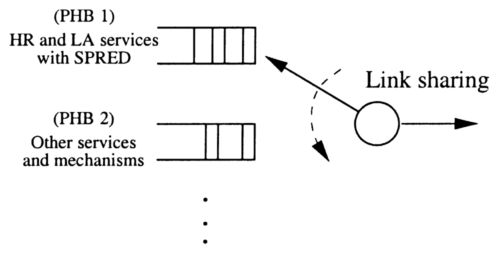


Fig. 5. SPRED as a service module under a more sophisticated Diffserv architecture where there are multiple PHBs at the node.

module under a hierarchical link sharing architecture for a more sophisticated Diffserv architecture at a node [11].

3.3.6. *In-profile and out-of-profile packets*

A traffic profile specifies the temporal properties of a traffic stream selected by a classifier. It provides rules for determining whether a particular packet is in-profile or out-of-profile [3,8]. So far we have only considered packets that are all in-profile. This is valid as long as traffic shapers are deployed in Diffserv boundary nodes and therefore all packets entering the Diffserv domain are shaped to conform traffic profile.

In the case that traffic shapers are not available or it is inappropriate to shape certain type of traffic, a marker can be employed at the Diffserv boundary to tag packets within a traffic stream into *in-profile* and *out-of-profile* packets [3].

We point our that it is straightforward to extend our Diffserv architecture to handle both *in-profile* and *out-profile* traffic. When there is *out-of-profile* HR and LA traffic present, we can incorporate the RIO mechanism (described in [8]) on top of our SPRED algorithm to handle out-of-profile packets (while still use SPRED for in-profile HR and LA traffic).

4. Simulation results

In this section, we implement both the BE Internet (FIFO with tail-dropping) and our Diffserv/SPRED architectures on our network simulator. We perform simulations of integrated traffic of real-time multimedia streaming applications and traditional TCP/UDP traffic over various benchmark network configurations under the BE and our Diffserv/SPRED architectures. We use MPEG-4 video described in Section 2.2 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 Diffserv/SPRED architecture can provide substantial performance improvement over the BE service Internet for multimedia streaming applications.

4.1. *Simulation settings*

The network configurations that we use are the *peer-to-peer* (Fig. 6), the *parking lot* (Fig. 13), and the *chain* (Fig. 16) network configurations.

We use MPEG-4 video as an example multimedia streaming application. At the source side, we use the standard raw video sequence ‘Akiyo’ in QCIF format for the MPEG-4 video encoder. The encoder performs MPEG-4 coding described in [6]. The encoded MPEG-4 bit-stream is packetized and classified into HR and LA service packets before being sent to the network. In particular, we classify the foreground VO

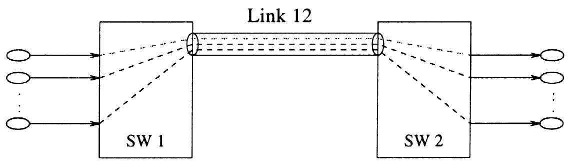


Fig. 6. A peer-to-peer network.

(right of Fig. 1) and important system information as HR service and background (middle of Fig. 1) as LA service. 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 [15].

In addition to MPEG-4 video streaming, we also use TCP/UDP traffic to represent traditional data applications and classify such traffic under LA service. We assume all TCP sources are persistent during the simulation run. For UDP connections, we use an exponentially distributed on/off model with average $E(T_{\text{on}})$ and $E(T_{\text{off}})$ 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

$$r_{\text{p}} \cdot \frac{E(T_{\text{on}})}{E(T_{\text{on}}) + E(T_{\text{off}})}.$$

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) [22].

Table 1
Simulation parameters

End system	MPEG-4	MaxPL	526 bytes
		Aggregate rate	20 Kbps
		VO1 (background) rate	6.8 Kbps
		VO2 (foreground) rate	13.2 Kbps
		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	$E(T_{\text{on}})$	100 ms
		$E(T_{\text{off}})$	150 ms
		r_{p}	100 Kbps
		Packet size	576 bytes
Switch		Buffer size	10 Kbytes
		Packet processing delay	4 μs
Link	End system to switch	Link speed	10 Mbps
		Distance	1 km
	Switch to switch	Distance	1000 km

For the RED mechanism used for LA 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 [10]. The $\min_{\mathrm{th}}$ and $\max_{\mathrm{th}}$ parameters are set to 5 and 15 packets, respectively.

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

4.2. Peer-to-peer configuration

The simulation results under the peer-to-peer network (Fig. 6) are organized as follows. As a first case (Case 1), we show the performance of a MPEG-4 video streaming under BE and Diffserv/SPRED where there is sufficient network bandwidth. Under this scenario, both BE and Diffserv/SPRED should have the same application level performance (in terms of perceptual quality). Then we show the cases when there is a shortage of network bandwidth (Case 2) and interaction with competing TCP/UDP traffic (Case 3). Under both Cases 2 and 3, we find substantial performance improvement of our Diffserv/SPRED over the BE architecture for video streaming application. We elaborate each case as follows.

4.2.1. Case 1: Abundant bandwidth (congestion free)

We activate only one MPEG-4 source under the peer-to-peer configuration (Fig. 6) without any other TCP/UDP traffic. The capacity for Link12 is set to 25 Kbps, which is higher than the MPEG-4 aggregate rate of 20 Kbps (VO1 and VO2).

We observe that the link utilization is 80% and there is no packet loss, under both BE and SPRED architectures, indicating that there is no congestion.

The peak signal-to-noise (PSNR) can be used as a measure for application level performance (perceptual quality) for video application. PSNR calculates the difference between the original source video sequence and the received video sequence. Fig. 7 shows the PSNR of Y component of the MPEG-4 video at the receiver under the BE Internet (FIFO with tail-dropping) and our Diffserv/SPRED architectures. As expected, there is no difference in terms of PSNR performance for each VO between the two architectures, since there is abundant network bandwidth for the MPEG-4 video connection under both architectures.

To examine the perceptual quality of the MPEG-4 video, we play out the decoded video sequence at the receiver. Fig. 8 shows a sample video frame at the receiver under BE and Diffserv/SPRED architectures,

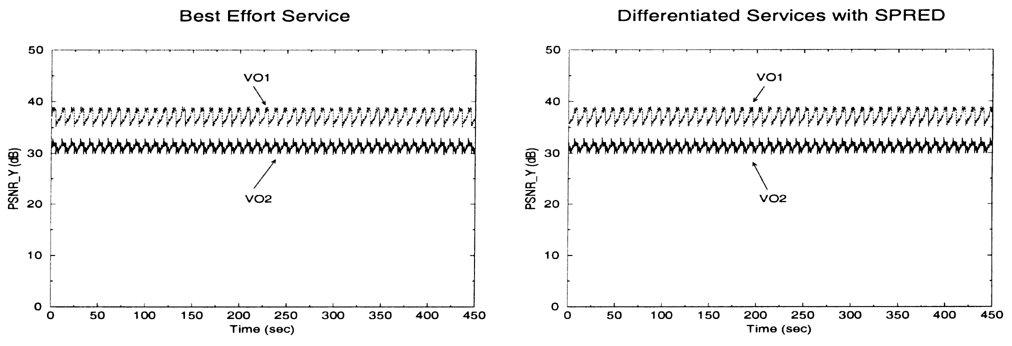


Fig. 7. PSNR of VOs at the receiver under BE and DS/SPRED architectures for the peer-to-peer network. Case 1: abundant link bandwidth.

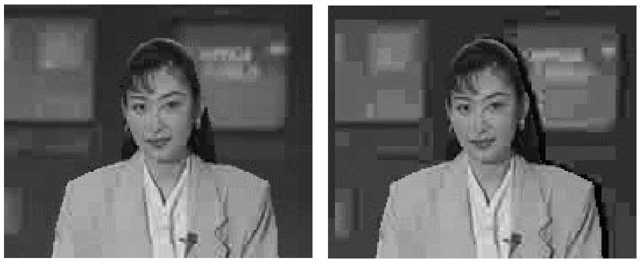


Fig. 8. Sample frame at the receiver under BE Internet (left) and DS/SPRED (right) for the peer-to-peer network.

respectively. The pictures in Fig. 8 all show the same frame. We find that the perceptual quality is the same since there is no packet loss under both architectures. [4]

The simulation results under Case 1 for BE and Diffserv/SPRED shows the best possible PSNR performance for each VO at the receiver (due to over-supply of bandwidth and zero packet loss) and these PSNRs will be used as references for subsequent simulations, where there is a shortage of bandwidth or congestion.

4.2.2. Case 2: Bandwidth shortage

In this simulation, we activate only one MPEG-4 source (still without any TCP/UDP traffic) and set the bandwidth of Link12 to be 18 Kbps, which is higher than the rate of MPEG-4 foreground VO2 (13.2 Kbps), but lower than the aggregate rate (20 Kbps).

We observe that the link utilization is 100% and there is packet loss under both the BE and our Diffserv/SPRED architectures. Under the BE architecture, due to shortage of bandwidth, the respective average packet loss ratio for VO1 and VO2 are 12.6% and 17.2%. On the other hand, under our Diffserv/SPRED architecture, the average packet loss ratio for VO1 (under LA service) is 29.4% and there is no packet loss for VO2 (under HR service). This shows that our Diffserv/SPRED architecture offers much higher reliability to VO2 than the BE architecture.

Fig. 9 shows the PSNRs for VO1 and VO2 under the BE and our Diffserv/SPRED architectures, respectively. Comparing with Fig. 7, both VO1 and VO2 under the BE architecture have substantial performance degradation in terms of PSNR. On the other hand, under our Diffserv/SPRED architecture, only VO1 (under LA service) has significant PSNR degradation while the PSNR for VO2 (under HR service) is not affected.

To examine the perceptual quality of the MPEG-4 video, we play out the decoded video sequence at the receiver. Fig. 10 shows a sample video frame at the receiver under BE and Diffserv/SPRED architectures, respectively. The pictures in Fig. 10 all show the same frame. For a VOP with packet loss, we use error concealment to obtain that VOP rather than freezing the frame or replay the previous frame. The picture under BE architecture has lower quality due to error propagation, i.e., loss of one packet will affect all the following P-frames. Fig. 10 clearly demonstrates that our Diffserv/SPRED offers better application level performance improvement (in terms of perceptual quality) over the BE architecture under the same link bandwidth and network topology.

[4] Note that the pictures shown in Fig. 8 are of less quality than the left picture in Fig. 1. This is because the video shown in Fig. 1 is the original raw video before compression, which can be as high as 8 Mbps. On the other hand, the video frame shown in Fig. 8 is compressed with overall output of only 20 Kbps.

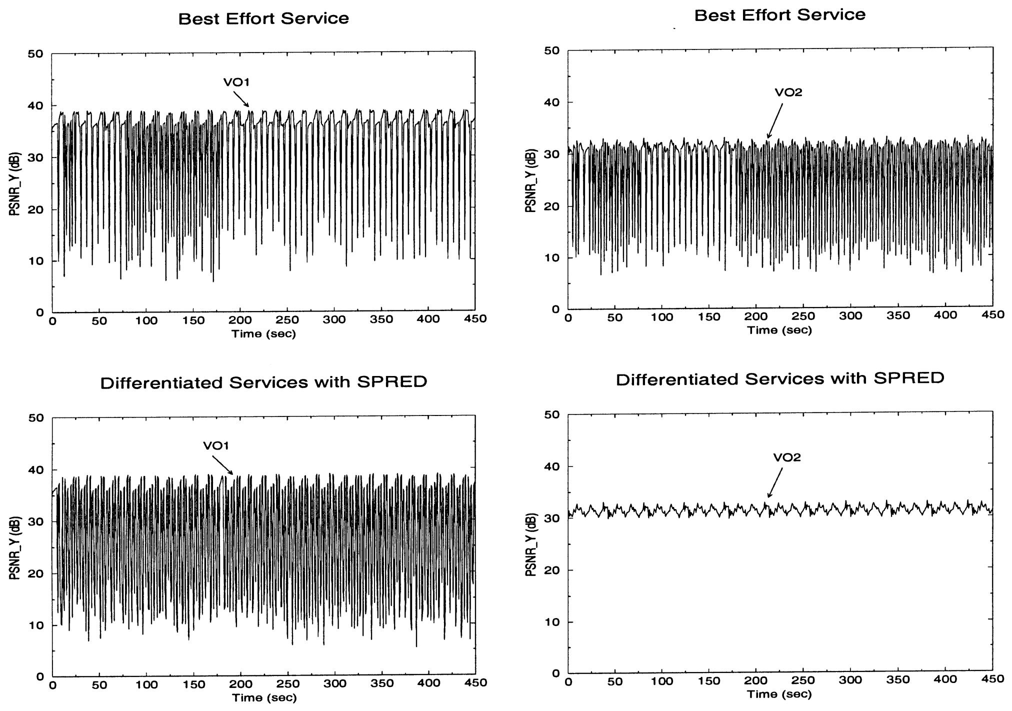


Fig. 9. PSNR of VOs at the receiver under BE and DS/SPRED architectures for the peer-to-peer network. Case 2: bandwidth shortage.

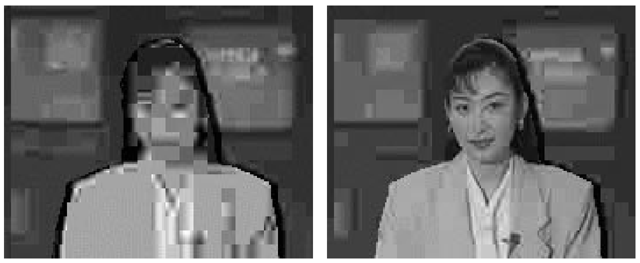


Fig. 10. Sample frame at the receiver under BE Internet (left) and Diffserv/SPRED (right) for the peer-to-peer network.

4.2.3. Case 3: Interaction with competing TCP and UDP traffic

We set the capacity of Link12 to be 200 Kbps (Fig. 6). In addition to one MPEG-4 video source, we also activate 5 TCP and 5 UDP connections to compete with the MPEG-4 video for the link bandwidth.

Fig. 11 shows the link utilization of Link12 under both the BE and Diffserv/SPRED architectures. We observe that Link12 is heavily utilized under both architectures. Under the BE architecture, the packet loss ratio are 7.18% for VO1 and 7.62% for VO2, respectively, while under the Diffserv/SPRED architecture, the packet loss ratio is 9.46% for VO1 and there is no packet loss for VO2, which shows that our Diffserv/SPRED architecture offers much higher reliable transport to VO2 than the BE architecture.

Fig. 12 shows the PSNR for VO1 and VO2 under both BE and our Diffserv/SPRED architectures. Comparing with Fig. 7, under the BE architecture, both VO1 and VO2 have substantial performance

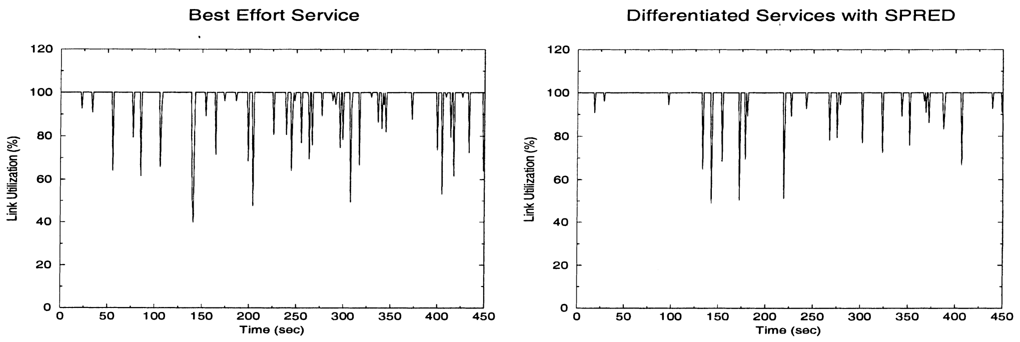


Fig. 11. Link utilization under BE and DS/SPRED architectures for the peer-to-peer network. Case 3: interacting with TCP/UDP traffic.

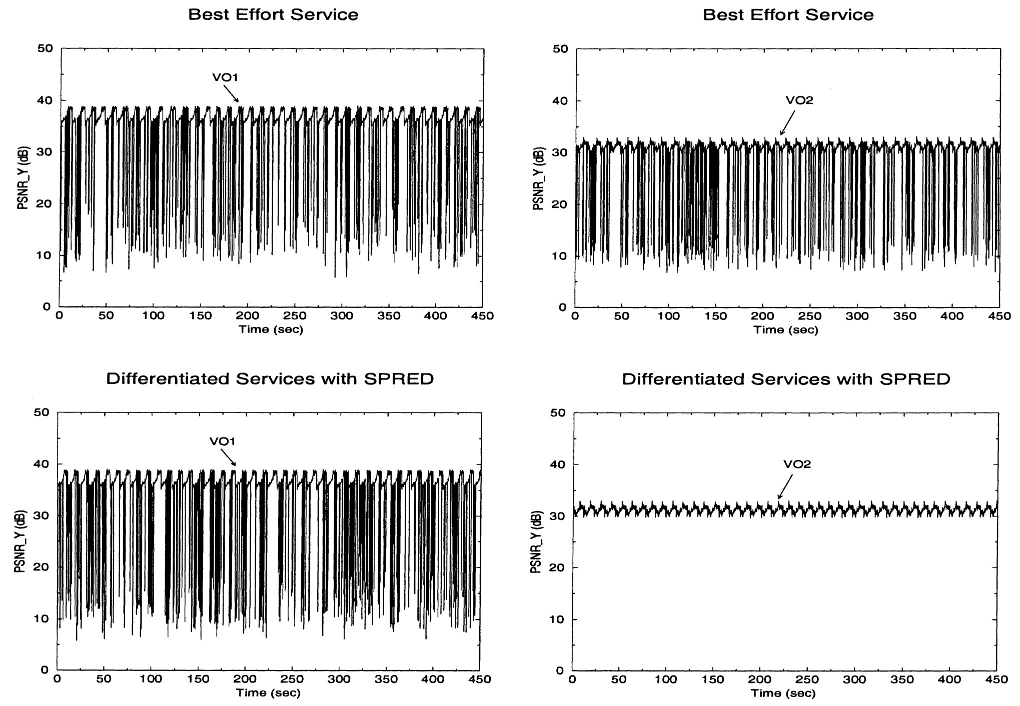


Fig. 12. PSNR of VOs at the receiver under BE and DS/SPRED architectures for the peer-to-peer network. Case 3: interacting with TCP/UDP traffic.

degradation in terms of PSNR. However, under the Diffserv/SPRED architecture, only VO1 (under LA service) has significant degradation in PSNR while the PSNR for VO2 (under HR service) is not affected, indicating that HR packets are well protected under our SPRED mechanism.

To examine the perceptual quality of the MPEG-4 video, we play out the decoded video sequence at the receiver. The sample frames are similar to those shown in Fig. 10, which demonstrates that our Diffserv/SPRED offers better application level perceptual quality over BE architecture for video streaming.

Finally, we observe that there is no synchronization behavior among the TCP connections under the SPRED mechanism. This is due to random dropping of LA packets under SPRED.

4.3. Parking lot configuration

This configuration and its name is derived from theater parking lots, which consists of several parking areas connected via a single exit path. The specific parking lot network that we use is shown in Fig. 13, where path G1 consists of multiple flows and traverse from the first switch (SW1) to the last switch (SW5), path G2 starts from SW2 and terminates at the last switch (SW5), and so forth. Clearly, Link45 is the potential bottleneck link for all flows.

In this simulations, path G1 consists of one MPEG-4 source, three TCP connections and three UDP connections, while paths G2, G3 and G4 all consist of three TCP connections and three UDP connections, respectively. We set the link capacity between the switches to be 400 Kbps.

Fig. 14 shows the link utilization of Link45 under BE and SPRED architectures, respectively. Under the BE architecture, the respective average packet loss ratio for VO1 and VO2 are 4.85% and 3.14%, while, under our Diffserv/SPRED architecture, the average packet loss ratio for VO1 is 6.48% and there is no packet loss for VO2. This shows that our Diffserv/SPRED architecture offers much higher reliable transport to VO2 than the BE architecture.

Fig. 15 shows the PSNR for VO1 and VO2 under the BE and our Diffserv/SPRED architectures, respectively. Comparing with Fig. 7, under the BE architecture, both VO1 and VO2 have performance degradation for PSNR. On the other hand, under the Diffserv/SPRED architecture, only VO1 has significant degradation in PSNR while the PSNR for VO2 is not affected.

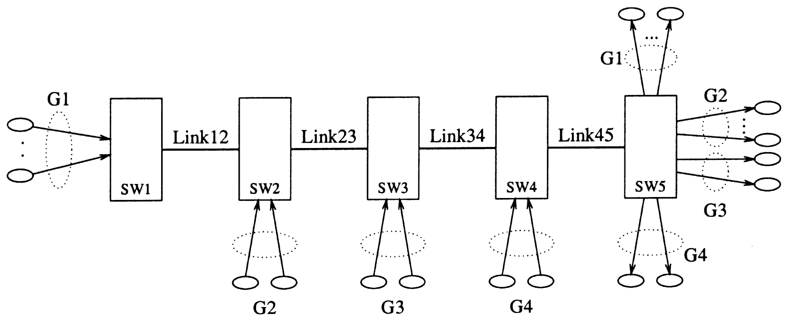


Fig. 13. A parking lot network.

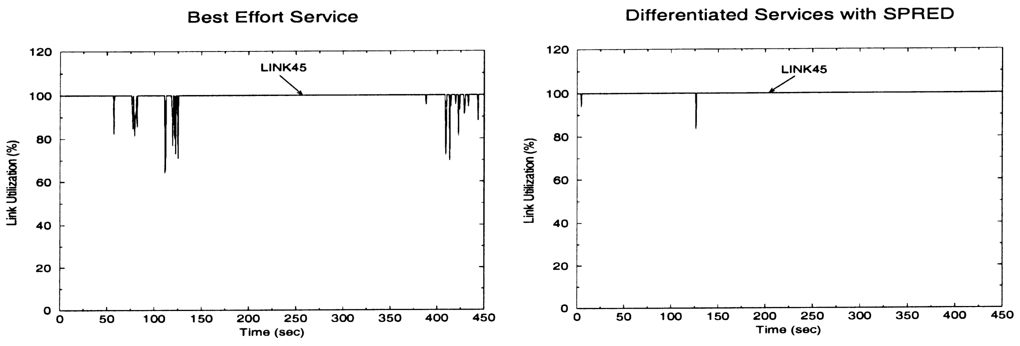


Fig. 14. Link utilization under BE and DS/SPRED architectures for the parking lot network.

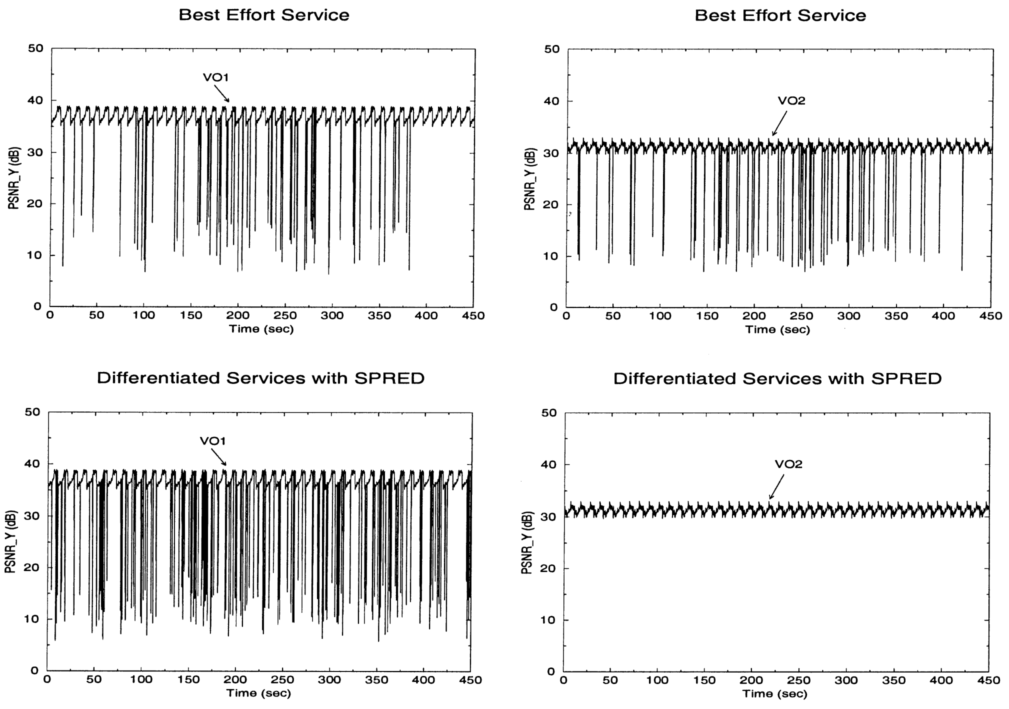


Fig. 15. PSNR of VOs at the receiver under BE and DS/SPRED architectures for the parking lot network.

To examine the perceptual quality of the MPEG-4 video, we play out the decoded video sequence at the receiver. The sample frames are similar to those shown in Fig. 10, which demonstrates that our Diffserv/SPRED offers better application level service quality than BE architecture for streaming applications.

We also observe that there is no synchronization among the TCP connections under the SPRED mechanism. This is due to random dropping of LA packets under SPRED.

4.4. Chain configuration

This is a benchmark network configuration commonly used to examine traffic behavior under the impact of other traversing interfering traffic. The specific chain configuration that we use is shown in Fig. 16 where

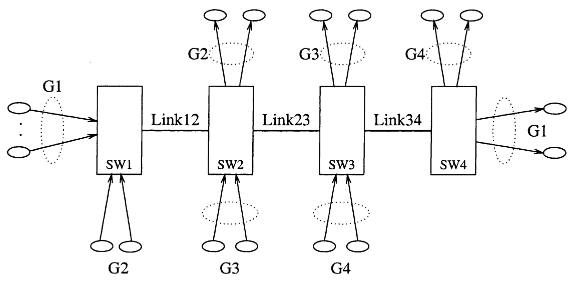


Fig. 16. A chain network.

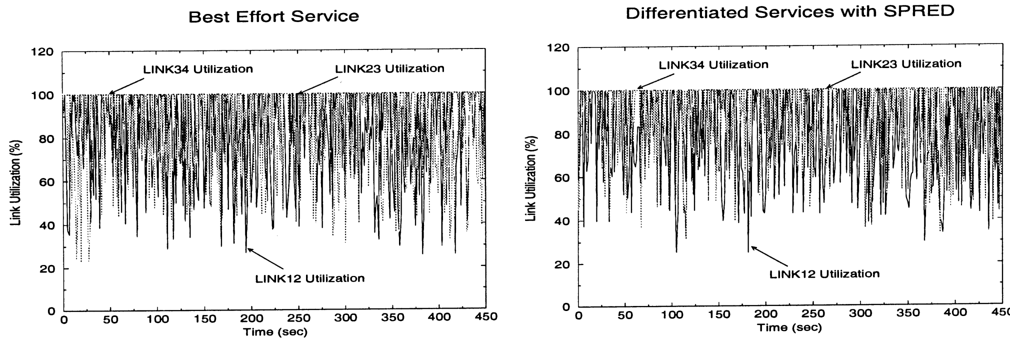


Fig. 17. Link utilization under BE and DS/SPRED architectures for the chain network.

path G1 consists 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 Kbps on Link12, Link23, and Link34.

Fig. 17 shows the link utilization of Link12, Link23 and Link34 under both the BE and SPRED architectures. Under the BE architecture, the packet loss ratio are 2.88% for VO1 and 2.68% for VO2, respectively, while under the Diffserv/SPRED architecture, the packet loss ratio is 3.77% for VO1 and there is no packet loss for VO2, indicating that our Diffserv/SPRED architecture offers much higher reliable transport to VO2 than the BE architecture.

Fig. 18 shows the PSNR for VO1 and VO2 under both BE and our Diffserv/SPRED architectures. Comparing with Fig. 7, both VO1 and VO2 have substantial performance degradation in terms of PSNR under the BE architecture. However, under our Diffserv/SPRED architecture, only VO1 (under LA service) has significant PSNR degradation while the PSNR for VO2 (under HR service) is not affected, which shows that that HR packets are well protected under our SPRED mechanism.

To examine the perceptual quality of the MPEG-4 video, we play out the decoded video sequence at the receiver. The sample frames are similar to those shown in Fig. 10, which demonstrates that our Diffserv/SPRED offers better application level service quality than the BE architecture.

Finally, we find that there is no synchronization among TCP connections under the SPRED mechanism. This is due to random dropping of LA packets under SPRED.

Remark 2. We summarize the packet loss ratio (PLR) from the above simulations in Table 2. Note that under all simulations, the output rate of the MPEG-4 video encoder is 20 Kbps (6.8 Kbps for VO1 and 13.2 Kbps for VO2, see Table 1).

- Under the Diffserv/SPRED architecture, since the PLR for VO2 are all zero, the perceptual quality for VO2 is, therefore, the same as the VO2 shown in the right picture of Fig. 10. On the other hand, the PLR for VO1 varies a great deal under different simulations (e.g., 29.4%, 3.77%). Thus, the perceptual quality for VO1 has variation under different simulation, but all has similar degradation pattern as shown in VO1 in the right picture in Fig. 10.
- Under the BE architecture, both VO2 and VO1 have packet loss under different simulation settings. The performance degradation for VO2 and VO1 all follow the similar pattern to those shown in the left picture of Fig. 10, with some degree of variation of course.

Based on our extensive simulation results, we conclude that, under the same link bandwidth and network topology, our Diffserv/SPRED architecture offers significant application level performance improvement

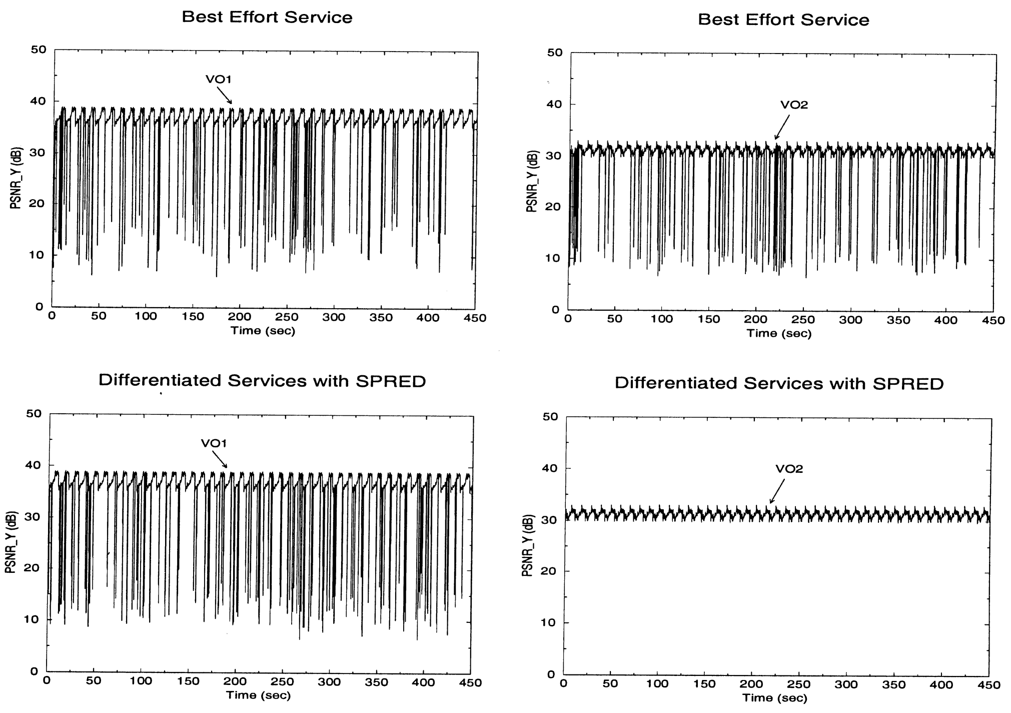


Fig. 18. PSNR of VOs at the receiver under BE and DS/SPRED architectures for the chain network.

Table 2
Packet loss ratio (PLR) of the VOs for the MPEG-4 video sequence 'Akiyo' under different network configurations

Network configuration	BE		Diffserv	
	VO1 (%)	VO2 (%)	VO1 (%)	VO2 (%)
Peer-to-peer (Case 2)	12.6	17.2	29.4	0
Peer-to-peer (Case 3)	7.18	7.62	9.46	0
Parking lot	4.85	3.14	6.48	0
Chain	2.88	2.68	3.77	0

over the BE service architecture for transporting real-time multimedia streaming applications. The trade-off lies in the fact that the proposed Diffserv/SPRED architecture can intelligently discard low priority packets while preserving the high priority packets which are critical for the perceptive quality of the streaming application.

5. Concluding remarks

As multimedia streaming applications proliferate, the current BE service Internet is becoming increasingly inadequate to meet the service requirements from streaming applications. This paper presented a core-stateless Diffserv architecture in the context of Assured Forwarding PHB with the aim of imporving the performance of multimedia streaming. We defined two types of services differentiated in terms of reliability: the HR service and the LA service. Our main contribution is a novel node mechanism called SPRED to achieve the service differentiation. We showed that the SPRED node mechanism is a generalized form of buffer management with both tail-dropping and RED as its special cases. It combines the best features of pushout and RED/RIO and is well suited for multimedia streaming applications. More important, SPRED is capable of achieving all of our four design objectives and PHB requirement simultaneously.

- SPRED does not require core routers to maintain any state information for each flow and therefore is highly scalable.
- By employing single shared queue and storing/servicing packets in the order of arrival, the packet sequence within each flow is preserved at each node.
- Packets from HR service have much better loss protection than packets from LA service at a node during congestion. In particular, an incoming HR packet will not be discarded if there are LA packets in the buffer and discarding of such LA packets can leave buffer space for the incoming HR packet (our Diffserv PHB).
- By incorporating randomization of packet dropping for TCP connections (i.e., RED), our SPRED mechanism avoids the global synchronization problem associated with TCP.

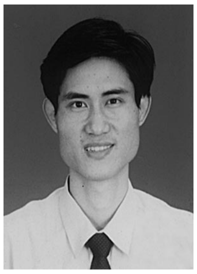
Our simulation results conclusively demonstrated that under the same link speed and network topology, network nodes employing our Diffserv/SPRED architecture has substantial performance improvement over the current BE architecture for real-time multimedia streaming applications.

References

[1] F.M. Anjum, L. Tassiulas, Fair bandwidth sharing among adaptive and non-adaptive flows in the Internet, in: Proceedings of the IEEE Infocom, New York, March 1999, pp. 1412–1420.
[2] J.C.R. Bennett, H. Zhang, WF2Q: worst-case fair weighted fair queueing, in: Proceedings of the IEEE Infocom, San Francisco, CA, March 1996, pp. 120–128.
[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An architecture for differentiated services, RFC 2475, Internet Engineering Task Force, December 1998.
[4] R. Braden, D. Clark, S. Shenker, Integrated services in the Internet architecture: an overview, RFC 1633, Internet Engineering Task Force, July 1994.
[5] 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, L. Zhang, Recommendations on queue management and congestion avoidance in the Internet, RFC 2309, Internet Engineering Task Force, April 1998.
[6] T. Chiang, Y.-Q. Zhang, A new rate control scheme using quadratic rate distortion model, IEEE Trans. Circuits Systems Video Technol. 7 (1997) 246–250.
[7] I. Cidon, L. Georgiadis, R. Guerin, A. Khamisy, Optimal buffer sharing, IEEE J. Selected Areas Commun. 13 (1995) 1229–1240.
[8] D.D. Clark, W. Fang, Explicit allocation of best-effort packet delivery service, IEEE/ACM Trans. Networking 6 (1998) 362–373.
[9] A. Demers, S. Keshav, S. Shenker, Analysis and simulations of a fair queueing algorithm, in: Proceedings of the ACM SIGCOMM, Austin, TX, 1989, pp. 1–12.
[10] S. Floyd, V. Jacobson, On random early detection gateways for congestion avoidance, IEEE/ACM Trans. Networking 1 (1993) 397–413.
[11] S. Floyd, V. Jacobson, Link-sharing and resource management models for packet networks, IEEE/ACM Trans. Networking 3 (1995) 365–386.
[12] S.J. Golestani, A self-clocked fair queueing scheme for broadband applications, in: Proceedings of the IEEE Infocom, Toronto, Canada, April 1994, pp. 636–646.

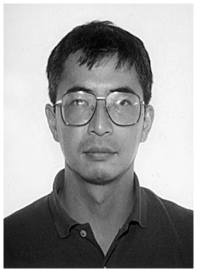
[13] A. Gupta, D. Stahl, A. Whinston, Priority pricing of integrated services networks, in: L. McKnight, J. Bailey (Eds.), Internet Economics, MIT Press, Cambridge, MA, 1997, pp. 253–279.
[14] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, Assured forwarding PHB group, RFC 2597, Internet Engineering Task Force, June 1999.
[15] 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, December 1998.
[16] T.V. Lakshman, A. Neidhardt, T.J. Ott, The drop from front strategy in TCP and in TCP over ATM, in: Proceedings of the IEEE Infocom, San Francisco, CA, March 1996, pp. 1242–1250.
[17] D. Lin, R. Morris, Dynamics of random early detection, in: Proceedings of the ACM SIGCOMM, Cannes, France, September 1997.
[18] K. Nichols, V. Jacobson, L. Zhang, A two-bit differentiated services architecture for the Internet, Internet Draft, Internet Engineering Task Force, November 1997.
[19] K. Nichols, S. Blake, F. Baker, D. Black, Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers, RFC 2474, Internet Engineering Task Force, December 1998.
[20] T.J. Ott, T.V. Lakshman, L.H. Wong, SRED: Stabilized RED, in: Proceedings of the IEEE Infocom, New York, March 1999, pp. 1346–1355.
[21] A.K. Parekh, R.G. Gallager, A generalized processor sharing approach to flow control – the single node case, in: Proceedings of the IEEE Infocom, Florence, Italy, May 1992, pp. 915–924.
[22] H. Schulzrinne, D. Hoffman, M. Speer, R. Civanlar, A. Basso, V. Balabanian, C. Herpel, RTP payload format for MPEG-4 elementary streams, Internet Draft, Internet Engineering Task Force, March 1998.
[23] S. Shenker, C. Partridge, R. Guerin, Specification of guaranteed quality of service, RFC 2212, Internet Engineering Task Force, September 1997.
[24] M. Shreedhar, G. Varghese, Efficient fair queueing using deficit round robin, in: Proceedings of the ACM SIGCOMM, September 1995, pp. 231–242.
[25] D. Stiliadis, A. Varma, A general methodology for designing efficient traffic scheduling and shaping algorithms, in: Proceedings of the IEEE Infocom, Kobe, Japan, April 1997, pp. 326–335.
[26] D. Stiliadis, A. Varma, Rate-proportional servers: a design methodology for fair queueing algorithms, IEEE/ACM Trans. Networking 6 (1998) 164–174.
[27] D. Stiliadis, A. Varma, Efficient fair queueing algorithms for packet-switched networks, IEEE/ACM Trans. Networking 6 (1998) 175–185.
[28] L. Tassiulas, Y.C. Hung, S.S. Panwar, Optimal buffer control during congestion in an ATM network node, IEEE/ACM Trans. Networking 2 (1994) 374–386.
[29] D. Wu, Y.T. Hou, W. Zhu, Y.-Q. Zhang, H.J. Chao, MPEG-4 compressed video over the Internet, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'99) Orlando, FL, 30 May–2 June, 1999.
[30] L. Zhang, VirtualClock: A new traffic control algorithm for packet switching networks, ACM Trans. Comput. Syst. 9 (1991) 101–124.

Yiwei Thomas Hou obtained his B.E. degree (*Summa Cum Laude*) from the City College of New York in 1991, the M.S. degree from Columbia University in 1993, and the Ph.D. degree from Polytechnic University, Brooklyn, New York, in 1997, all in Electrical Engineering. He was awarded a National Science Foundation Graduate Research Traineeship for pursuing Ph.D. degree in high speed networking, and was recipient of the Alexander Hessel award for outstanding Ph.D. dissertation during 1997–1998 academic year from Polytechnic University. While a graduate student, he worked at AT&T Bell Labs, Murray Hill, New Jersey, during the summers of 1994 and 1995, on implementations of IP and ATM internetworking; he also worked at Bell Labs, Lucent Technologies, Holmdel, New Jersey, during the summer of 1996, on fundamental problems on network traffic management.Since September 1997, Dr. Hou has been a Researcher at Fujitsu Laboratories of America, Sunnyvale, California. His current interests are in the areas of quality of service (QoS) support for transporting multimedia applications over the Internet, and scalable architecture, protocols, and implementations for differentiated services. Dr. Hou is a member of the IEEE, ACM, and Sigma Xi.



Dapeng Wu received the B.E degree from Huazhong University of Science and Technology, and the M.E. degree from Beijing University of Posts and Telecommunications in 1990 and 1997, respectively, both in Electrical Engineering. Since July 1997, he has been working towards his Ph.D. degree in Electrical Engineering, Polytechnic University, Brooklyn, New York.During the summer of 1998 and most part of 1999, he conducted research at Fujitsu Laboratories of America, Sunnyvale, California, on architectures and traffic management algorithms for integrated services (Intserv) networks and differentiated services (Diffserv) Internet for multimedia applications. His current interests are in the areas of next generation Internet architecture, protocols, implementations for integrated and differentiated services, and rate control and error control for video streaming over the Internet. He is a student member of the IEEE and the ACM.

Bo Li received the B.S. (*cum laude*) and M.S. degrees in Computer Science from Tsinghua University (Beijing) in 1987 and 1989, respectively, and the Ph.D. degree in Computer Engineering from University of Massachusetts at Amherst in 1993. Between 1994 and 1996, he worked on high performance routers and ATM switches in IBM Networking System Division, Research Triangle Park, North Carolina. He joined the faculty of the Computer Science Department of the Hong Kong University of Science and Technology in January 1996.Dr. Li has been on editorial board for *ACM Mobile Computing and Communications Review* and *Journal of Communications and Networks*. He will be serving as an editor for *ACM/Baltzer Journal of Wireless Networks*. He has been co-guest editing special issues for *IEEE Communications Magazine*, *IEEE Journal on Selected Areas in Communications* and the upcoming *SPIE/Baltzer Optical Networks Magazine*. He has been involved in organizing many conferences such as IEEE Infocom, ICDCS and ICC. He will be the international vice-chair for IEEE Infocom'2001.Dr. Li's current research interests include wireless mobile networking supporting multimedia, voice and video (MPEG-2 and MPEG-4) transmission over the Internet and all optical networks using WDM.



Takeo Hamada graduated from the University of Tokyo with B.E. and M.E. degrees in Electrical Engineering in 1984 and 1986, respectively. He received Ph.D. in Computer Science from UCSD in 1992 for research in physical VLSI design. He has been with Fujitsu since 1986. From 1995 to the end of 1997, he engaged in research on service and resource management architecture in Telecommunication Information Network Architecture (TINA) and was with the TINA-C core-team at Red Bank, New Jersey. Since 1998, he has been with Fujitsu Laboratories of America, Sunnyvale, California. His current research interests include network management, service management issues in IP networks, and policy-based networking.

Ishfaq Ahmad received a B.S. degree in Electrical Engineering from the University of Engineering and Technology, Lahore, Pakistan, in 1985. He earned his M.S. degree in Computer Engineering and Ph.D. degree in Computer Science, both from Syracuse University in 1987 and 1992, respectively. At present, he is an Associate Professor in the Department of Computer Science at the Hong Kong University of Science and Technology (HKUST). He is also Director of Multimedia Technology Research Center at HKUST. The center is engaged in industrial collaboration and a number of research projects related to information technology, in particular in the areas of video coding and interactive multimedia systems in a distributed environment. His additional research interests are parallel programming tools, and scheduling and mapping algorithms for scalable high-performance architectures. He has published over 100 technical papers in refereed archival journals and conference proceedings. He has received a number of research and teaching awards, including the Best Student Paper Award at Supercomputing'90 and Supercomputing'91, and Teaching Excellence Award by the School of Engineering at HKUST. He has served on the program committees of numerous international conferences, and has guest-edited several journals. He serves on the editorial boards of *IEEE Transactions on Circuits and Systems for Video Technology*, *IEEE Concurrency*, and *Cluster Computing*. He is a member of the IEEE Computer Society.

H. Jonathan Chao received the B.S.E.E. and M.S.E.E. degrees from National Chiao Tung University, Taiwan, in 1977 and 1980, respectively, and the Ph.D. degree in Electrical Engineering from The Ohio State University, Columbus, OH, in 1985.He is a Professor in the Department of Electrical Engineering at Polytechnic University, Brooklyn, NY, which he joined in January 1992. His research interests include large-capacity packet switches and routers, packet scheduling and buffer management, and congestion flow control in IP/ATM networks. From 1985 to 1991, he was a Member of Technical Staff at Bellcore, NJ, where he conducted research in the area of SONET/ATM broadband networks. He was involved in architecture designs and ASIC implementations, such as the first SONET-like Framer chip, ATM Layer chip, and Sequencer chip (the first chip handling packet scheduling). He received Bellcore Excellence Award in 1987.He served as Guest Editor for *IEEE Journal on Selected Areas in Communications* special issue on 'Advances in ATM Switching Systems for B-ISDN' (June 1997) and special issue on 'Next Generation IP Switches and Routers' (June 1999). He is currently serving as an Editor for *IEEE/ACM Transactions on Networking*.