

Performance Evaluation of Prioritized Scheduling with Buffer Management for Differentiated Services Architectures

Ahmed E. Kamal
Dept. of Electrical & Computer Eng.
Iowa State University
Ames, IA 50011-3060
U.S.A.
Telephone: (515) 294-3580
Fax: (515) 294-1152
E-mail: kamal@iastate.edu

Hossam S. Hassanein
School of Computing
Queen's University
Kingston, ON K7L 3N6
Canada
Telephone: (613) 533-6052
Fax: (613) 533-6513
E-mail: hossam@cs.queensu.ca

Abstract

Differentiated Services (DiffServ) is an architecture for the Internet in which various applications are supported using a simple classification scheme. Packets entering the DiffServ domain are marked depending on the packets' class. In this paper we introduce a versatile service and buffering scheme for different classes in a differentiated services Internet. The objective of this scheme is to introduce relative differentiation in terms of mean delay, throughput. In this scheme both high-buffer-priority (HPC) and low-buffer-priority (LPC) classes have reserved buffers. In addition, they have a shared buffer, where the priority of the shared buffer occupancy is to HPC traffic. Service priorities can be for high-buffer-priority traffic, low-buffer-priority traffic or round robin. An exact performance model for the proposed scheme is introduced. The performance model represents HPC and LPC traffic arrivals by a discrete batch Markov arrival process (D-BMAP). The model is used to obtain loss ratios, packet delays and throughputs for both HPC and LPC traffic. Performance results show that implementing buffer priorities, with a weighted round robin packet scheduling policy, results in low packet loss ratio and/or delay for the high-priority class without starving the low-priority class.

I Introduction

Differentiated Services (DiffServ) [1] is an architecture for the Internet in which various applications are supported using a simple classification scheme. Packets entering the DiffServ domain are marked depending on the packets' class. The DiffServ model was introduced as an alternative to the Integrated Services (IntServ) model [2], which requires resources such as bandwidth and buffers to be explicitly reserved for a given data flow to ensure that the application receives its requested Quality of Service (QoS). The IntServ architecture, however, suffered a scalability problem as it requires routers to maintain per flow information.

In the DiffServ architecture scalability is achieved in two ways. First, per flow service is replaced with aggregate class per hop service. Second, complex processing is moved from the core of the

network to the edges. The DiffServ model aggregates the entire user's requirement for QoS. A user wishing to receive service must first have a Service Level Agreement (SLA) with the service provider. An SLA includes a traffic conditioning agreement that gives detailed service parameters such as service level, traffic profile, marking and shaping. Packet marking is done through the DS field in the IP header in accordance with the requested class of service. Packets are also marked to be in profile or out of profile with respect to the SLA.

Treatment of packets inside the DiffServ domain is known as the Per Hop Behavior (PHB). PHB extended the standard best-effort treatment at routers to include *expedited forwarding* and *assured forwarding*. Expedited forwarding provides low loss, low latency and low jitter, assured bandwidth, end-to-end service through the DiffServ domain. Assured forwarding delivers the aggregate traffic from a user with high assurance as long as the aggregate traffic is within its traffic profile. Assured forwarding is intended for applications that do not require low latency or delay jitter. IETF [3] recommended four assured forwarding classes, each with three drop-precedence values. If there is congestion in a node, the drop precedence of a packet determines the relative importance of the packet.

Two important functions are required to effectively implement the PHB. The first is the packet scheduling policy, which determines packet service priorities at the output link. Priority scheduling can reduce packet delay, delay jitter and loss for high priority traffic. On the other hand, packets with high delay and/or loss tolerance or out of profile packets may be given a low scheduling priority.

Weighted Fair Queuing (WFQ) attempts to serve sessions/classes in proportion to their pre-specified service shares independent of the buffer occupancy of each session/class [4, 5, 6]. WFQ and its variants [7] have been popular due to their capability to address the limitations of the FCFS and fixed priority scheduling mechanisms. Moreover, WFQ adds to the network the capability of providing end-to-end delay guarantees on per flow basis [8]. The use of WFQ type schedulers has been proposed for the implementation of Service Rate Differentiation (SRD) models [9, 10]. SRD assigns a certain rate to each priority class that also depends on its expected arrival rate. High-priority classes have a larger service-to-arrival rate ratio than lower-priority classes. Consequently, higher classes are expected to have a lower queuing delay.

The second function is buffer management, which is responsible for deciding whether a packet belonging to a certain flow/class may be admitted or dropped. Buffers may or may not be shared among the different classes. The advantage of complete buffer sharing is that this technique makes full use of statistical multiplexing. However, a hostile source/link may consume the whole buffer

space. On the other hand, complete per flow/link/class buffer separation usually results in underutilization of buffer space. Recent work addressing the buffer sharing issue can be found in [11, 12, 13]. In addition, it was shown in [14] that the WFQ and buffer management strategies can be combined in order to control multiple performance measures in a proportional manner.

On the other hand, different packet dropping levels were used to handle congestion for various traffic classes in a shared buffer, and Random Early Detection (RED) [15] and RED with IN/OUT (RIO) [16] are such schemes. RED is a threshold-based scheme to detect incipient congestion by computing the average queue size. Unlike Early Random Drop that discards all packets after the queue size exceeds a certain threshold, RED works as follow. When the average queue (buffer) size exceeds the RED threshold, packets are dropped according to some probability which is a function of how full the buffer is. RIO extends RED by allowing different thresholds and dropping probabilities for "In" profile and "Out" of profile packets. RED parameters have significant effects on performance. However, no clear recommendation of some optimal setting of RED parameters has been agreed upon [17].

A number of schemes have been proposed for packet scheduling and buffer management in the Differentiated Services architecture. May et al. [18] evaluated the performance of the assured service scheme and the premium service scheme through analytical modeling. For the Assured Service a single buffer, in which both "tagged" (In) and "non-tagged" (out) packets are queued is used. RIO was used to control the number of in and out of profile packets. The model for Premium Service uses two separate buffers, a finite buffer for the "In" packets and an infinite one for the "Out" packets. Tagged packets get service first until the finite tagged queue is empty, then non-tagged packets get service. For ease of mathematical analysis, the arrival process was assumed to be Poisson and the service time was assumed exponential.

Kohler and Schafer [19] performed a comparative simulation study of the Performance of different class-and-drop treatments of data in DiffServ IP Networks. They investigated "Two-Bit Differentiated Services" with "Two Rate Three Color Marking with Three Drop Precedence". The Two-Bit DS model containing three classes, premium, assured and best effort, has been implemented to estimate the impact of different classes Separate queues were used for each service class, with priority scheduling. This led to more significant variations of the throughput of assured and best effort connections at different loads than for premium connections. The simulations also showed that premium and assured connections are extensively protected from best effort traffic.

Sahu et al. [20] studied two router mechanisms for DiffServ architectures, namely Threshold

Dropping (TD) and Priority Scheduling (PS). They performed a comparative study of the loss and delay behaviors of the TD and PS under the edge-discarding (ED) and edge-marking (EM) packet marking mechanisms. TD and PS distinguish between two classes of packets, preferred and non-preferred packets with the assumption that preferred class receives "preference" over the packets in the non-preferred class. Two analytical Markov models have been considered, a first model for TD coupled with ED, and second one for PS coupled with ED. In the first model, both preferred and non-preferred packets are queued into a single FIFO buffer. While, in the second model, preferred and non-preferred packets are kept in two separate finite buffers. Buffers are selected for service according to "strict priority scheduling", i.e., non-preferred packets receive service only when the preferred queue is empty. They concluded that preferred packets are not affected by the behavior of lower priority packets with the PS router mechanism. However, using pure priority scheduling results in starvation for the lower priority class. It was shown in [21] that starvation can be avoided only if the higher priority class is both burst and bandwidth controlled. On the other hand, [20] also shows that there is no mechanism to provide lower expected delays to preferred packets with the TD scheme without substantially increasing the loss rate of lower priority packets.

De Rezende [22] conducted a study to evaluate the performance of TCP and UDP traffic flows crossing DS domains in which network nodes implement AF forwarding. It is well known that TCP traffic is adaptive to traffic congestion by decreasing its sending rate, while UDP traffic is not. Without proper buffer management, UDP (non-adaptive traffic) may actually fill the buffers and it becomes hard to assign a fair portion on the buffer to TCP traffic. On the other hand, if UDP traffic is marked as a low priority traffic and/or is dropped from the buffer, this may cause serious performance degradation. It is, therefore, essential to devise mechanisms that can guarantee priority for high-priority classes yet do not unnecessarily degrade the loss or delay performance of lower-priority traffic.

This paper presents a versatile service and buffering scheme for different classes in a differentiated services Internet. In this scheme both high-priority and low-priority classes have reserved buffers. In addition, they have a shared buffer, where the priority of the shared buffer occupancy is to the higher buffer priority class. Service priorities may or may not be the same as the buffer occupancy priorities. The service priority can be fixed, or dynamic in a manner consistent with rate allocation, delay bounds, etc. For example, it can be done according to a weighted fair queuing approach. The proposed mechanism falls within the class of relative differentiation. However, by properly adjusting the parameters of service and buffer allocation, certain guarantees can be

provided, such as proportional delay differentiation, as well as throughput guarantees. More will be said about this issue in Section II.C.

This paper is organized as follows. The scheme is presented in the following section. A performance model of the proposed scheme is presented in Section 3. Numerical results for loss ratios and packet delays for the different classes is given in Section 4. Finally, Section 5 summarizes the findings of this paper.

II Proposed Scheme

In this section we describe a generic service and buffer management scheme, which is motivated by the following observations:

1. While priority scheduling achieves low loss and delay for the higher priority class, it causes performance degradation for the lower priority class. In many cases, the buffer space reserved for the higher priority class is not used and wasted. Allowing the lower priority class to use some of the buffer space would reduce the packet loss resulting from the use of priority scheduling policy.
2. To handle short-term congestion due to simultaneous burst arrivals, the higher priority class must have a reserved buffer set aside. Under normal network load, the higher priority class can actually share a buffer with other traffic classes.
3. The differentiated services Internet is intended to support a wide range of traffic classes, viz premium, assured and best effort. It is, therefore, desirable to use a service and buffer management scheme that is versatile and can be utilized for service differentiation among various traffic classes with different QoS requirements.
4. Service and buffer sharing priorities need not be the same. A loss-sensitive, delay-tolerant traffic class would require higher priority in buffer sharing, but not necessarily priority scheduling.

In our proposed scheme both high-priority and low-priority classes have reserved buffers. In addition, they have a shared buffer, where the priority of the shared buffer occupancy is to the higher priority class. The service priority can be fixed, or dynamic in a manner consistent with rate allocation, delay bounds, etc. We illustrate our scheme here using two classes only, but the

extension to multiple classes is straightforward. The two classes have different priorities in buffer occupancy, and service.

II.1 Buffer Management

We will refer to the class with the higher priority in buffer occupancy as the *Higher Priority Class* (HPC), while the other class is the *Lower Priority Class* (LPC). The service priority will be discussed later, but is not necessarily the same as the buffer occupancy priority. Both classes have reserved buffers, whose sizes are parameters that determine the performance of the respective classes. In addition, they have a shared buffer, where the priority of the shared buffer occupancy is to the HPC class. The buffering strategy is as follows:

LPC buffer strategy: LPC traffic is given a lower priority in using the shared buffer:

- LPC traffic always attempts to use its dedicated buffer first, and it is allowed to use the shared buffer only if its dedicated buffer is full, and there is an available space in the shared buffer. This also means that older packets will be queued in the LPC dedicated buffer.
- LPC traffic is also forced to release any space it occupies in the shared buffer if space becomes available in its dedicated buffer because an LPC packet is served.

HPC buffering strategy: HPC traffic always attempts to use the shared buffer before using its dedicated buffer. Therefore,

- If an HPC packet arrives and finds an availability in the shared buffer, it is queued in the shared buffer; otherwise, it is queued in the dedicated HPC buffer, if an availability exists.
- HPC traffic packets in the dedicated buffer move to shared buffer if space becomes available in the shared buffer. This happens when an HPC packet occupying the shared buffer is served. It can also happen if an LPC packet that is occupying the LPC dedicated buffer is served, therefore causing an LPC packet in the shared buffer to move to the dedicated buffer (as explained above)¹

¹This is required in order to preserve the FCFS service strategy among packets of the same class.

It should be noted that this scheme is different from the space priority scheme proposed in [23]. The low priority class here is guaranteed a minimum buffer space, and once it occupies a buffer it cannot be pushed out.

II.2 Packet Scheduling

The packet scheduling (service) priority can be fixed, or dynamic. Service priorities may or may not be the same as the buffer occupancy priorities. The following service priorities can be implemented:

- Round robin scheduling, with no preference to either class and no load-dependence or buffer occupancy dependence. This scheduling policy, however, cannot be used to assure QoS.
- Weighted round robin, which is buffer occupancy dependent. This scheduling policy would give implicit service priority to HPC packets, since they have higher buffer occupancy priority. However, it would not lead to starving LPC traffic, as LPC packets are served when the number of HPC packets is less than the size of the LPC traffic dedicated buffer. This will be later demonstrated in the performance results.
- HPC service priority, which then gives HPC traffic both buffer occupancy and scheduling priority. Such service priority can be used for premium service, or assured service, with LPC traffic being for best effort service.
- LPC service priority, which results in HPC traffic having higher priority in shared buffer occupancy, while LPC having higher priority in service. This can be used to serve the LPC as an expedited service class, with the HPC as an assured service class.

It is to be noted that, with proper packet scheduling, and buffer dimensioning, a low loss rate can be guaranteed for HPC packets. This is mainly due to the introduction of the shared buffer, and the higher buffer priority of HPC packets. On the other hand, a small delay of LPC traffic is possible since it be offered higher service priority, in addition to the fact that its buffer size is limited, which prevents packets from queueing for a long time.

II.3 Service Guarantees

The proposed mechanisms fall within the framework of relative differentiation. However, they are versatile enough that, with the proper controls, they are able to provide quantitative guarantees.

Table 1: Loss ratios and throughputs under different load conditions

	Service	Loss ratio		Throughput	
		LPC	HPC	LPC	HPC
Heavy LPC and light HPC	RR	$1-(1-\lambda_H)/\lambda_L$	0	$1-\lambda_H$	λ_H
	LPC first	$1-1/\lambda_L$	1	1	0
	HPC first	$1-(1-\lambda_H)/\lambda_L$	0	$1-\lambda_H$	λ_H
Light LPC and heavy HPC	RR	0	$1-(1-\lambda_L)/\lambda_H$	λ_L	$1-\lambda_L$
	LPC first	0	$1-(1-\lambda_L)/\lambda_H$	λ_L	$1-\lambda_L$
	HPC first	1	$1-1/\lambda_H$	1	λ_H
Heavy LPC and heavy HPC	RR	$1-\rho_L/(\lambda_L(\rho_L+\rho_H))$	$1-\rho_H/(\lambda_H(\rho_L+\rho_H))$	$\rho_L/(\rho_L+\rho_H)$	$\rho_H/(\rho_L+\rho_H)$
	LPC first	$1-1/\lambda_L$	1	1	0
	HPC first	1	$1-1/\lambda_H$	0	1

For example, the buffering mechanism guarantees that the HPC will be allocated more buffers than the LPC. This does not by itself guarantee any differentiation in the loss ratio, since the loss ratio depends very much on the offered load, and the service strategy. Therefore, the service strategy can be used to impose certain guarantees on the loss ratios. In Table 1, it is shown how certain loss ratios can be achieved under different load conditions (in this example it is assumed that the service capacity is unity). It is also shown what levels of throughput will be achievable in this case. Note that light and heavy load correspond to $\lambda \approx 0$ and $\lambda > 1$, respectively.

Using a different strategy, and different objectives, different guarantees can be provided. For example, using the waiting time priority service strategy proportional delay differentiation can be implemented [9]. Given the mean delay proportional differentiation, and the total mean delay, the mean delay per class can be calculated. Then, the queueing discipline results in certain average queue lengths for the two classes, \bar{q}_H and \bar{q}_L . For example, under heavy load

$$\bar{q}_L = B_L \quad \text{and} \quad \bar{q}_H = B_H + B_S$$

Since, by Little's result, throughput = \bar{q}/\bar{d} , where \bar{d} is the mean delay, then the buffers can be dimensioned so that the throughput guarantees can be determined, and hence the loss ratio.

II.4 Implementation Issues

In this subsection, we briefly comment on the implementation complexity of the proposed mechanism.

In [24] the authors proposed a switch architecture where the input buffer is divided into two areas, which they called high priority queue and low priority queues. The high and low priority queues are formed on the basis of the occupancy of the output buffer not on the urgency of the individual cells. In order to maintain the cell sequence, they have to simultaneously lookup the two buffers and swap cells between them. It was shown that the complexity of such a scheme is similar to the shared memory buffer architecture. The proposed scheme is also related to a shared-memory ATM switching scheme whose is described in [25]. Therefore, a similar implementation can be used for our scheme, which should have the same complexity (time and space). The only difference is that our search (queue lookup) for next packet to schedule spans three buffer areas, while in [25] they search only in one shared memory. However, from time point of view, the 3 searches (lookup) in our model could be done simultaneously, since they are in three disjoint areas. From a space point of view, the complexity is the total area which is the same (disregarding whether they are in three disjoint chunks or one big chunk).

In the following section we present a mixed service priority/buffer management mathematical model that represents the generic buffer management and service disciplines described above. It should be noted that the model may be modified to incorporate other service disciplines.

III The Performance Model

The system is modeled as a time-homogeneous, discrete time, finite-state Markovian chain. We assume that time is slotted, and the slot duration is equal to the packet transmission time.

III.1 Definitions

We make the following definitions for HPC and LPC traffic:

- B_L LPC dedicated buffer size.
 - B_H HPC dedicated buffer size.
 - B_S Shared buffer size.
 - L_{LD} Number of LPC packets in the LPC dedicated buffer.
 - L_{LS} Number of LPC packets in the shared buffer.
 - L_L Total number of LPC packets in the system = $L_{LD} + L_{LS}$.
 - L_{HD} Number of HPC packets in the HPC dedicated buffer.
 - L_{HS} Number of HPC packets in the shared buffer.
 - L_H Total number of HPC packets in the system = $L_{HD} + L_{HS}$.
- It should be noted that the following relations hold:

$$\begin{aligned}
L_{LD} &= \min(L_L, B_L) \\
L_{LS} &= \max(0, L_L - B_L) \\
L_{HS} &= \min(L_H, B_S - L_{LS}) \\
L_{HD} &= \max(0, L_H - L_{HS})
\end{aligned}$$

III.2 Assumptions

- Packet size is fixed, and the packet transmission time is assumed to be unity.
- Time is slotted, and the slot duration is equal to the packet transmission time.
- HPC traffic packets in the dedicated buffer move to shared buffer if space becomes available in the shared buffer. LPC traffic uses shared buffer only if its dedicated buffer is full, and there is an available space in the shared buffer. This assumption is made in order to keep the model tractable by just tracking the total number of the packets from each class in the buffer, instead of keeping track of the number of the packets in the dedicated buffer, or in the shared buffer.
- We will use a probabilistic packet service approach, since using an auxiliary variable to indicate service priority will result in increasing the state space. So we serve either the LPC traffic with a certain probability ρ_L , or serve the HPC traffic with probability $\rho_H = 1 - \rho_L$. The choice of these probabilities can be load dependent (e.g., queue size priority² $\rho_L = \frac{L_L}{L_L + L_H}$,

²If $\rho_L(\rho_H)$ is chosen to be $\rho_L = \frac{L_L/\lambda_L}{L_L/\lambda_L + L_H/\lambda_H}$ ($\rho_H = \frac{L_H/\lambda_H}{L_L/\lambda_L + L_H/\lambda_H}$), then this corresponds to the implementation of waiting time priority scheduling.

and $\rho_H = \frac{L_H}{L_L + L_H}$. It can also be load independent which implements a certain strict priority service order, or a combination of the two. The choice of $\rho_L = \rho_H = 0.5$ approximates round robin scheduling, while $\frac{\rho_H}{\rho_L} = \infty$ gives HPC traffic service priority as well. Setting $\frac{\rho_H}{\rho_L} = 0$ can be used to give higher service priority to LPC traffic. As such, HPC has higher priority in shared buffer occupancy, while LPC has higher priority in service.

III.3 Model Description

HPC and LPC packet arrivals are modeled as independent traffic streams. Each of the two traffic streams generates traffic according to a discrete batch Markov arrival process (D-BMAP):

1. The LPC D-BMAP process has r_L phases, and the probability of s arrivals from the LPC stream is governed by the matrix $D_L(s) = [d_{L_{i,j}}(s)]$, where

$$d_{L_{i,j}}(s) = \Pr(s \text{ arrivals, next phase } = j | \text{current phase } = i)$$

for $i, j \in \{0, 1, \dots, r_L - 1\}$. The maximum number of arrivals from the LPC process per slot time is denoted by M_L .

2. The HPC D-BMAP is defined similarly, but with r_H phases, a $D_H(s)$ matrix, and a maximum of M_H arrivals per slot.

We define the system state at the end of a slot, and is described in terms of a four tuple: (i, j, m, n) , where

- i is the value of L_L .
- j is the value of L_H
- m is the phase of the LPC arrival process.
- n is the phase of the HPC arrival process.

We define the matrix P to be the transition probability matrix of the Markov chain. P is a block matrix of dimension $((B_L + B_S + 1) \times (B_H + B_S + 1))$ by $((B_L + B_S + 1) \times (B_H + B_S + 1))$. Each entry in the P matrix, $p_{(i,j),(i'',j'')}$, is the probability of making a transition from state $(L_L = i, L_H = j)$ to state $(L_L = i'', L_H = j'')$. $p_{(i,j),(i'',j'')}$ is itself a matrix of dimension $(r_L \times r_H)$ by $(r_L \times r_H)$ which corresponds to the probabilities of phase changes in the arrival processes.

The state transition is done in two steps:

1. In the first step, we consider the service process, in which one packet, if any, is served from the buffer(s). In this case, a transition takes place from state (i, j, m, n) , to state (i', j', m, n) .
2. In the second step, we consider the arrivals, together with the arrival process phase change. The state changes from (i', j', m, n) to (i'', j'', m'', n'') .

III.3.i First transition

A transition from state (i, j, m, n) , to state (i', j', m, n) is independent of m and n , and takes place with probability $q_{(i,j),(i',j')}$. This probability depends on the values of i, j, i' and j' . There are four cases to consider:

1. If $i = 0$, and $j = 0$, then $i' = j' = 0$ with probability $q_{(i,j),(i',j')} = 1$.
2. If $i = 0$, and $j > 0$, then $i' = 0$ and $j' = j - 1$ with probability $q_{(i,j),(i',j')} = 1$.
3. If $i > 0$, and $j = 0$, then $i' = i - 1$ and $j' = 0$ with probability $q_{(i,j),(i',j')} = 1$.
4. If $i > 0$, and $j > 0$, then
 - with probability $q_{(i,j),(i',j')=\rho_L}$, $i' = i - 1$, and $j' = j$.
 - with probability $q_{(i,j),(i',j')=\rho_H}$, $i' = i$, and $j' = j - 1$.

III.3.ii Second transition

After the first transition, a transition from state (i', j', m, n) to state (i'', j'', m'', n'') takes place. There are also four cases to consider. In all of these cases, we must have $i'' \geq i'$ and $j'' \geq j'$.

1. If

Either

- $i'' < B_L + B_S$,
- $j'' < B_H + B_S$, and
- $\max(0, i'' - B_L) + j'' < B_S$,

Or

- $i'' < B_L$, and

- $j'' < B_H + B_S$

then there is no contention, and no loss will take place. $p_{(i,j),(i'',j'')}$ is therefore given by:

$$p_{(i,j),(i'',j'')} = q_{(i,j),(i',j')} [D_L(i'' - i') \otimes D_H(j'' - j')] \quad (1)$$

where \otimes is the Kronecker product operator.

2. If

- $i'' \geq B_L$,
- $i'' + j'' \geq B_L + B_S$ (shared buffer is full), and
- $i'' - B_L + j'' < B_S + B_H$ (HPC buffer is not full),

then some loss from the LPC traffic may have occurred.

In this case,

$$p_{(i,j),(i'',j'')} = q_{(i,j),(i',j')} \sum_{k=i''-i'}^{M_L} [D_L(k) \otimes D_H(j'' - j')] \cdot \alpha(i'' - \max(i', B_L), j'' - j' | i'' - \max(i', B_L), j'' - j', \max(B_S - \max(i' - B_L, 0) - j', 0)) \quad (2)$$

where

$\alpha(l, h | L, H, f) = \Pr(\text{accepting } l \text{ packets from the LPC stream and } h \text{ packets from the HPC stream, both in the shared buffer, given } L \text{ arrivals from the first stream, and } H \text{ arrivals from the latter stream may occupy the shared buffer, and } f \text{ free locations in the shared buffer}).$

3. If

- $i'' < B_L$, and
- $j'' = B_S + B_H$,

then there is a possible loss from the HPC stream.

In this case,

$$p_{(i,j),(i'',j'')} = q_{(i,j),(i',j')} \sum_{l=j''-j'}^{M_H} [D_L(i'' - i') \otimes D_H(l)] \quad (3)$$

4. If

- $B_L \leq i'' \leq B_L + B_S$,
- $B_H \leq j'' \leq B_H + B_S$, and
- $B_S = (i'' - B_L) + (j'' - B_H)$,

then possible loss may have occurred from both the LPC and the HPC streams.

In this case,

$$p_{(i,j),(i'',j'')} = q_{(i,j),(i',j')} \sum_{k=i''-i'}^{M_L} \sum_{l=j''-j'}^{M_H} [D_L(k) \otimes D_H(l)] \cdot \phi \quad (4)$$

where

$$\phi = \begin{cases} \alpha(i'' - \max(i', B_L), B_S - i'' + \max(B_L, i') - j') \\ \quad k - \max(i', B_L), l, \max(B_S - \max(i' - B_L, 0) - j', 0) & \text{for } \max(i' - B_L, 0) + j' < B_S \\ \alpha(i'' - \max(i', B_L), 0 | k - \max(i', B_L), l, 0) & \text{for } \max(i' - B_L, 0) + j' \geq B_S \end{cases} \quad (5)$$

The derivation of $\alpha(l, h | L, H, f)$ depends on how the packets are selected. Three possibilities can be considered:

Random selection:

$$\alpha(l, h | L, H, f) = \frac{\binom{L}{l} \binom{H}{h}}{\binom{L+H}{l+h}} \quad (6)$$

Priority to HPC packets:

$$\alpha(l, h | L, H, f) = \begin{cases} 1 & \text{if } h = \min(H, f) \text{ and } l = \min(L, f - h) \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Priority to LPC packets:

$$\alpha(l, h | L, H, f) = \begin{cases} 1 & \text{if } l = \min(L, f) \text{ and } h = \min(H, f - l) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

III.4 Performance Measures

Let $\vec{\Pi}$ be the steady state probability (row) vector of the above Markov chain, which can be obtained by solving

$$\vec{\Pi}P = \vec{\Pi} \quad \text{and} \quad \vec{\Pi} \cdot \vec{1} = 1$$

where $\vec{1}$ is an appropriately dimensioned column vector whose elements are all equal to 1. The vector $\vec{\Pi}$ is given by

$$\vec{\Pi} = \{ \bar{\pi}^{0,0}, \bar{\pi}^{0,1}, \dots, \bar{\pi}^{i,j}, \dots, \bar{\pi}^{B_L+B_S, B_H+B_S} \}$$

where i and j are the numbers of LPC and HPC packets in the buffer, respectively. The vector $\bar{\pi}^{i,j}$ has $r_L \times r_H$ components corresponding to the phases of the arrival processes. That is,

$$\bar{\pi}^{i,j} = \left(\pi_{0,0}^{i,j}, \pi_{0,1}^{i,j}, \pi_{0,2}^{i,j}, \dots, \pi_{m,n}^{i,j}, \dots, \pi_{r_L, r_H}^{i,j} \right)$$

Once the steady state probability vector is found, performance measures such as the probability of packet loss, and the mean packet delay can be found as follows:

Arrival rate: The arrival rate of a class LPC packets are given by:

$$\lambda_L = \sum_{i,j} \sum_s s \bar{\pi}^{i,j} D_L(s) \vec{1}$$

Throughput: The throughput of class LPC is given by:

$$\sigma_L = \sum_{i>0} \sum_j \rho_L(i, j) \bar{\pi}^{i,j} \vec{1}$$

Loss ratio: The loss ratio of class LPC is given by:

$$l_L = \frac{\lambda_L - \sigma_L}{\lambda_L}$$

Mean number of packets on departure: The mean number of packets left in the queue by a departing LPC packet is given by:

$$Q_L = \sum_{i>0} \sum_j (i-1) \rho_L(i, j) \bar{\pi}^{i,j} \vec{1}$$

Mean waiting time: The mean time spent in the system by an LPC packet is obtained from Little's result, and is expressed as:

$$W_L = Q_L / \sigma_L$$

Similar results exist for the HPC class.

IV Numerical Results

Using the analysis in Section 3, we now present numerical results for loss ratios and expected delays incurred by HPC and LPC packets. We model bursty arrivals by specializing the general D-BMAP arrival process to represent a number of on-off sources. An on-off source toggles between an on period, with mean t_{ON} , and an off period, with mean t_{OFF} . When a source is in the on state, it transmits at a peak rate λ_p . An on-off source is silent during the off state. The mean rate, λ , of an on-off source is given by $\frac{t_{ON} \times \lambda_p}{t_{ON} + t_{OFF}}$. We define the burstiness factor (BF) of an on-off source to be the ratio of the on and off periods ($BF = t_{ON}/t_{OFF}$)³. For a given λ , a larger BF value would indicate higher burstiness.

The number of sources, on period, off period, burstiness factor and mean rate for LPC traffic is given by N_{LPC} , ON_{LPC} , OFF_{LPC} , BF_{LPC} and λ_{LPC} , respectively. Similarly, we can define N_{HPC} , ON_{HPC} , OFF_{HPC} , BF_{HPC} and λ_{HPC} for HPC traffic. Buffer sharing is controlled through the parameters BL, BH and BS. The function α is set to random selection. Priority scheduling is controlled through the parameters ρ_L and ρ_H , and can be set to weighted round robin, priority scheduling to HPC traffic or priority scheduling to LPC traffic.

We first study the effect of the buffer management scheme independent of the packet scheduling policy. Therefore, we study the performance of both HPC and LPC traffic at different arrival rates and buffer sizes, with weighted round robin packet scheduling. Figures 1 and 2 respectively plot the loss ratio and packet delay for both HPC and LPC traffic versus the HPC arrival rate (λ_{HPC}), for shared buffer sizes of 6 and 10. The parameter settings used were $N_{HPC} = N_{LPC} = 4$, $BF_{HPC} = BF_{LPC} = 1$, $\lambda_{LPC} = 0.25$ and $BL = BH = 2$. We first observe that the loss ratio for HPC traffic is always lower than that for LPC traffic, see Figure 1. This is due to the preferred use of the shared buffer for HPC traffic. However, it should be noted that since LPC traffic is also allowed access to the shared buffer, its loss ratio is maintained at a reasonable level, compared to HPC traffic. It should also be noted that the delay of LPC sources is low even under heavy load. In fact, by considering the results of Figure 1 and 2, we can see that we can provide (assure) 90% of arriving LPC packets (with the given LPC rate) a delay lower than one unit, under a total arrival rate⁴ below 0.85. On the other hand, the delay for HPC traffic is affected by the presence of LPC

³Although the burstiness factor is usually defined as the proportion of epochs in which the source rate exceeds the average rate, for an on-off source we choose to use the above definition. This is because when the source is in the ON period, it always transmits at the peak rate, which exceeds the average rate.

⁴This corresponds to an LPC rate of 0.25, and an HPC rate of 0.6.

traffic, especially under heavy load. By using HPC priority scheduling (not shown here), the delay for HPC traffic is significantly reduced. Indeed, for the settings above, the delay is maintained below 2.5 units, even under heavy load.

We next study the effect of buffer management on loss ratios for HPC traffic with different burstiness factors (BF_{HPC}), see Figure 3. Weighted round robin packet scheduling is used. The total buffer size is 14, which can be partitioned as (B_L, B_H, B_S) of $(2, 2, 10)$, $(3, 3, 8)$, $(4, 4, 6)$ or $(5, 5, 4)$. The offered load was chosen to be 0.92 (heavy load) to emulate network congestion. HPC traffic rate is set to either 0.67 or 0.25 with BF_{HPC} being 0.5, 1 or 1.5. We note that for the lower HPC arrival rate (0.25), the HPC loss ratio is relatively not affected by LPC traffic, which is high in this case (0.67). The loss ratio for HPC traffic is always less than 1%, even for highly bursty traffic ($BF_{HPC}=0.5$). For high HPC arrival rate (0.67), the larger the shared buffer size, the lower the loss ratio. This is expected as a smaller shared buffer (BS) would lead to a larger dedicated LPC buffer (BL) not used by HPC traffic. The effect of traffic burstiness is more evident. HPC traffic can only be assured a loss ratio less than 10% for $BF_{HPC}=1$ only if $BS \geq 6$, whereas for $BF_{HPC}=1.5$, such a loss ratio can always be assured. For $BF_{HPC}=0.5$, no such assurance cannot be made for such a total buffer size regardless of how buffers are managed. However, using HPC priority scheduling with HPC buffer priority, results in loss ratios below 10% for all values of BS and BF_{HPC} .

In Figure 4, we study the average delay of LPC traffic with LPC priority scheduling, under the same settings used in Figure 3. Results indicate that the LPC packet delay is not affected by HPC traffic characteristics, but rather its own. Since the shared buffer is mainly occupied by HPC packets, delay for LPC traffic only increase as a result of higher values of its dedicated buffer (BL). For a lower LPC rate (0.25), LPC buffer occupancy is low, and with LPC priority scheduling, the delay is maintained regardless of the size of BL. LPC priority scheduling with a small BL, can then be used for expedited service with the assured service being treated as HPC traffic.

V Conclusions

In this paper we have introduced a versatile service and buffering scheme for different classes in a differentiated services Internet. In this scheme both high-priority (HPC) and low-priority (LPC) classes have reserved buffers. In addition, they have a shared buffer, where the priority of the shared buffer occupancy is to the higher buffer priority class. Service priorities can be for HPC

traffic, LPC traffic or round robin. An exact two-class performance model for the proposed scheme was derived. The performance model represents LPC and HPC traffic arrivals by a discrete batch Markov arrival process (D-BMAP). The model was used to obtain loss ratios, packet delays and throughputs for both HPC and LPC traffic.

Performance results have shown that implementing buffer priorities, with a weighted round robin packet scheduling policy, results in low HPC packet loss ratios without starving LPC traffic. To guarantee low packet loss and delay under network congestion, HPC priority scheduling should be used as well. Implementing LPC priority scheduling results low delay for LPC traffic, while maintaining a reasonably low loss ratio for HPC traffic.

References

- [1] Blake, S., Black D., Carlson, M., Davies, E., Wang, Z., and Weiss, W., "An Architecture for Differentiated Services", RFC 2475, Dec. 1998.
- [2] Braden, R., Clark, D. and Shenker, S., "Integrated Services in The Internet: Architecture and Overview", IETF RFC 1633, June 1994.
- [3] Heinanen, J., Baker, F., Weiss, W., and Wroclawski, J., "Assured Forwarding PHB Group", IETF RFC 2597, 1999.
- [4] Demers, A., Keshav, S., and Sheker, S., "Analysis and Simulation of a Fair Queuing Algorithm", Journal of Interworking: Research and Experience, Vol. 1, January 1990, pp. 3-26.
- [5] Blanquer, J. and Ozden, B., "Fair Queuing for Aggregated Multiple Links", ACM SIGCOM, August 2001.
- [6] Parekh, A. and Gallager, R., "A Generalized Processing Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case", IEEE/ACM Trans. on Networking, Vol. 1, No. 3, June 1993, pp. 334-356.
- [7] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", IEEE Proceedings, Vol. 83, No. 10, Oct. 1995.
- [8] J.Zhao, H.Hassanein, Jieyi Wu, and Guanqun Gu, End-to-end routing framework in differentiated services networks, Computer Communications, 26 (2003) pp. 566-578.

- [9] Dovrolis, C., and Stiliadis, D., "Relative Differentiated Services in the Internet: Issues and Mechanisms", ACM SIGMETRICS, May 1999.
- [10] Dovrolis, C., Stiliadis, D., and Ramanathan, P., "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling", ACM SIGCOMM, Sept. 1999.
- [11] Cheung S., and Pencea C., "Pipelined Sections: A New Buffer Management Discipline for Scalable QoS Provision", IEEE INFOCOM, April 2001.
- [12] Wu-chang, F., Kandlur, D., Saha, D., and Shin, K., "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness", IEEE INFOCOM, June 2002.
- [13] Kesselman, A., and Mansour, Y., "Harmonic Buffer Management Policy for Shared Memory Switches", IEEE INFOCOM, June 2002.
- [14] Sankaran, S. and Kamal, A. E., "A Combined Delay and Throughput Proportional Scheduling Scheme for Differentiated Services", to appear in the proceedings of IEEE Globecom 2002
- [15] Floyd, S. and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Vol. 1, No. 4, pp. 397-413, 1993.
- [16] Clark, D. and Wroclawski, J., "An Approach to Service Allocation in the Internet", Internet Draft, draft-clark-diff-svc-alloc-00.txt, 1997.
- [17] Holot, C., Misra, V., Towsley, D., and Gong, W-B., "A Control Theoretic Analysis of RED", IEEE INFOCOM, April 2001.
- [18] May, M., Bolot, J-C., Jean-Marie, A., and Diot, C., "Simple Performance Models of Differentiated Services Schemes for the Internet", IEEE INFOCOM, Mar. 1999.
- [19] Kohler, S. and Schaffer, U., "Performance Comparison of Different Class-and-Drop Treatment of Data and Acknowledgments in DiffServ IP Networks ", Report No. 237, Institute of Computer Science, University of Wurzburg, Aug. 1999.
- [20] Sahu, S., Towsley, D. and Kurose, J., "A Quantitative Study of Differentiated Services for the Internet", Proc. IEEE Global Internet, Dec. 1999.
- [21] Hou, M. and Mouftah, H. T., "Investigation of premium service using differentiated services IP", Computer Communications, vol. 22, pp. 1283-1295, 1999.

- [22] De, Rezende, J.F., "Assured Service Evaluation", Proc. IEEE GLOBECOM, 1999, pp. 100-104.
- [23] G. Hebuterne and A. Gravey, "A Space Priority Queueing Mechanism for Multiplexing ATM Channels", Comp. Net. and ISDN Sys., Vol. 20, Dec. 1990, pp. 37-43.
- [24] M. A. Aboelaze, A. B. Mnaour, and A. Elnaggar " Dynamic Cell Allocation to Input Queues in a Combined I/O Buffered ATM Switch" Proceedings of Internet Computing IC2001, June 2001.
- [25] Y-S. Lin, S-C. Yang, S-J. Fang and C.B. Shung, "VLSI Design of a Priority Arbitrator for Shared Buffer ATM Switches," IEEE International Conference on Circuits and Systems, 1997, pp. 2785-2788.

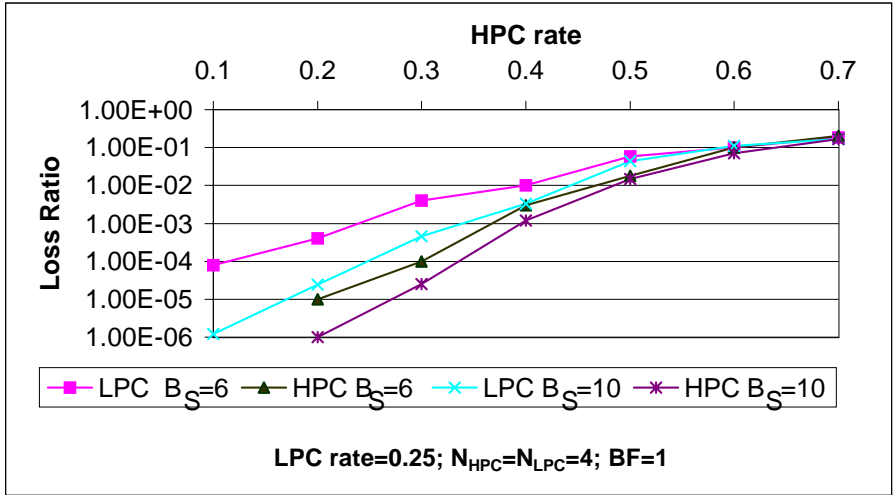


Figure 1 Loss ratio vs. HPC offered load (weighted round robin)

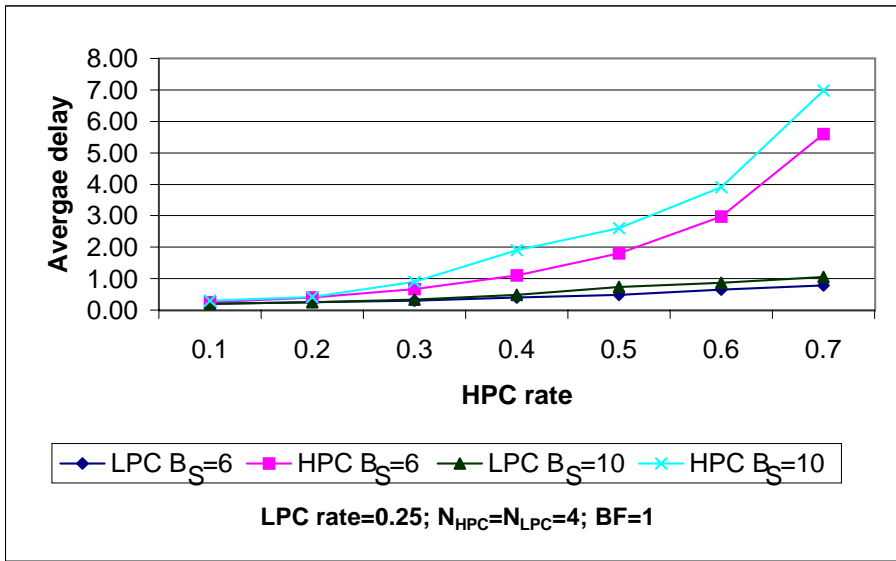


Figure 2 Packet waiting time vs. HPC offered load (weighted round robin)

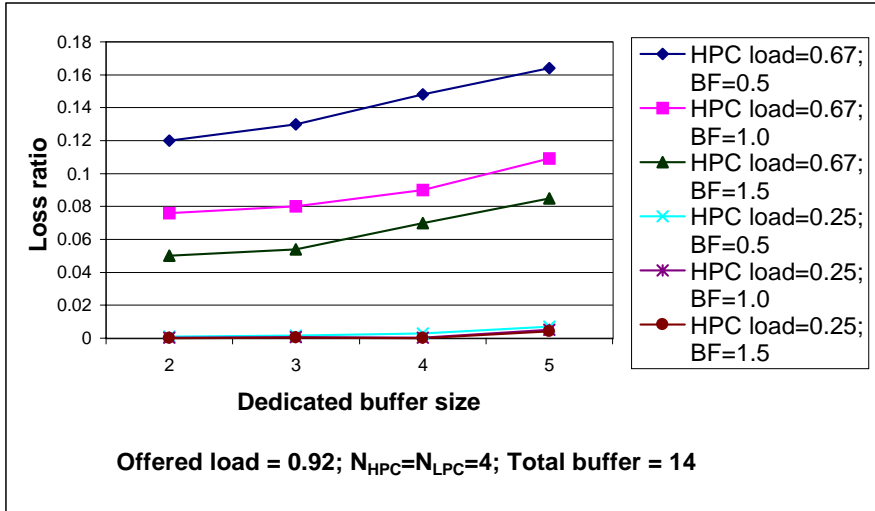


Figure 3 Effect of buffer sharing on HPC loss ratio (weighted round robin)

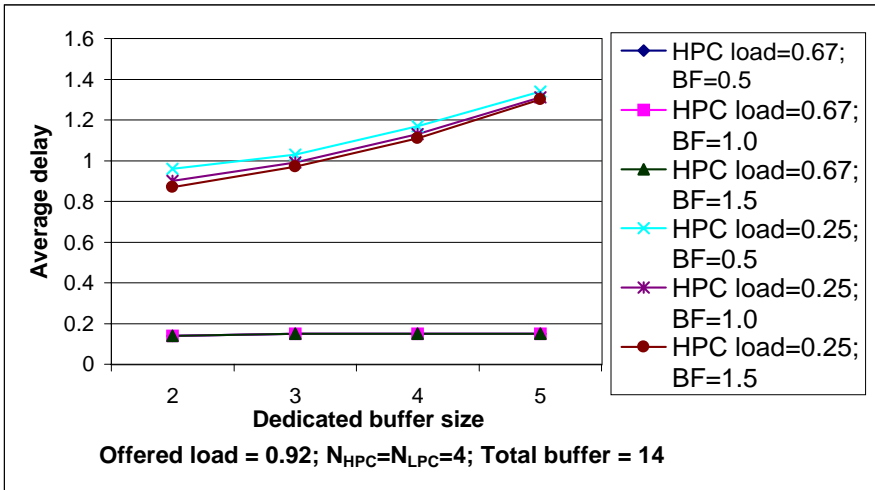


Figure 4 Effect of buffer sharing on LPC packet waiting time (LPC priority scheduling)