Buffer Management for Lossless Service in Shared Buffer Switches

Deng Pan
School of Computing & Information Sciences
Florida International University
Miami, FL 33199, USA

Yuanyuan Yang
Dept. of Electrical & Computer Engineering
State University of New York
Stony Brook, NY 11794, USA

ABSTRACT

Fair scheduling and buffer management are two typical approaches to provide differentiated service. Fair scheduling algorithms usually need to keep a separate queue and maintain associated state variables for each incoming flow, which make them difficult to operate and scale. On the contrary, buffer management (in conjunction with FIFO scheduling) needs only a constant amount of state information and processing, and can be efficiently implemented. In this paper, we consider using buffer management to provide lossless service for guaranteed performance flows in shared buffer switches. We study the buffer size requirement and buffer allocation strategies by starting with the single output switch and then extending the analytical results to the general multiple output switch. We present a universally applicable buffer allocation method for assuring lossless service, and validate the correctness of the theoretical results through simulations.

Keywords: Buffer management, shared buffer, differentiated service, lossless service

I. INTRODUCTION

With the rapid development of broadband based multimedia applications, it is necessary for next generation networks to provide differentiated service [1] [2] under various types of network traffic with different requirements. Network service can be broadly classified into two categories: guaranteed performance service and best effort service. For guaranteed performance service, resources are reserved for an allocated transmission rate, and the performance, such as bandwidth, delay and jitter, is bounded within a pre-specified range. Best effort service, as implied by the name, makes use of the available transmission capacity and tries the best to forward packets, but provides no guarantees to the

service quality. The constant bit rate (CBR) service and unspecified bit rate (UBR) service in ATM networks [3] belong to the guaranteed performance category and the best effort category, respectively.

There has been a lot of research in providing differentiated service, focusing on fair scheduling and buffer management. Fair scheduling algorithms [5] [6] [7] [8] organize incoming packets on a per flow basis, and emulate the ideal GPS model [9] [10] to schedule the packets of different flows in a fair manner. By using time stamp based or round robin based approaches, they can provide different levels of fairness guarantees at different costs. However, it has been reported [11] [12] that there exists a fundamental tradeoff between the delay bound that an algorithm can achieve and its computational complexity, which implies that fair scheduling algorithms either suffer from long worst-case delay or high time complexity. Furthermore, fair scheduling algorithms have to keep a separate queue and maintain associated state variables for each flow. This requirement makes them difficult to implement and scale in high speed networks. On the contrary, the traditional first-in-first-out (FIFO) scheduler can perform scheduling fast, by putting packets in a single queue in their arrival order and transmitting packets from the head of the queue. However, the disadvantage of the FIFO scheduler is that it may not schedule the flows in a fair manner, i.e., a flow with a higher input rate naturally obtains a higher output rate. To overcome this problem, buffer management has been adopted in [13] [14] in conjunction with the simple FIFO scheduler to provide differentiated service. (Unfortunately, Proposition 2 and its corresponding results in Section 2 in [13] were incorrect, for which we will provide corrections in this paper.) Buffer management offers protection to guaranteed performance flows as the first defense line, by preventing other flows from injecting an excessive number of packets. The FIFO scheduler and buffer management typically require only a constant amount of processing and state information [13], and are therefore able to work in a high speed environment.

In this paper, we study using buffer management to provide lossless service in shared buffer switches. The scenario that we consider is illustrated in Fig. 1. The shared buffer switch may have a single output or multiple outputs. Each output runs a simple FIFO scheduler, and transmits packets from a shared buffer, where the packets of all the incoming flows are stored. Among the incoming flows, some are guaranteed performance flows that are compliant to specific traffic shaping schemes

and require lossless service. Each packet of these flows is expected to be sent to the destination(s), and should not be dropped due to lack of buffer space. The rest of the incoming flows are best effort flows, which may be aggressive and inject packets to any empty space that it can access. Note that because the proposed buffer management methods are based on the concept of packet flows, they are not applicable when packets are not transmitted as flows.

The shared buffer in Fig. 1 is partitioned into two types of smaller buffers, one for guaranteed performance flows and one for best effort flows. The partition is dynamically performed when a new guaranteed performance flow arrives or an existing guaranteed performance flow leaves. When many guaranteed performance flows come and leave frequently, the cost to calculate the buffer partition might be expensive. There are two solutions to remedy this problem. First, as will be seen in Section II-B, multiple flows can be combined into a single logical flow to simplify the analysis. Second, over-provisioning can be applied to the buffer of guaranteed performance flows, so that a new calculation is only necessary when the buffer space requirement is significantly increased or decreased. In other words, more buffer space than the actually required amount is allocated to the guaranteed performance flows. By doing so, when a few new guaranteed performance flows come, the allocated buffer space is still sufficient to ensure lossless service without recalculating the buffer partition. Once the buffer is divided into smaller buffers, they will not change during packet transmissions, and a specific flow can only store its packets in the designated area.

The multiplexer in Fig. 1 works in a dynamic time division multiple access (TDMA) mode. It dynamically allocates bandwidth to different flows based on their traffic patterns. It is assumed that the multiplexer has sufficient bandwidth and is able to accommodate arbitrary traffic arrivals. The multiplexer schedules packets from the incoming flows in a first-come-first-served (FCFS) manner. For the earliest arrived packet, the multiplexer checks whether the buffer of its flow still has sufficient free space. If there is enough space, the multiplexer accepts the packet and puts it into the buffer of its flow. Otherwise, the multiplexer will simply discard the packet. Similarly, the demultiplexer in Fig. 1 works in the dynamic TDMA mode and removes packets from the shared buffer in a FCFS manner. It always selects the earliest arrived packet, regardless it is from a guaranteed performance flow or a best

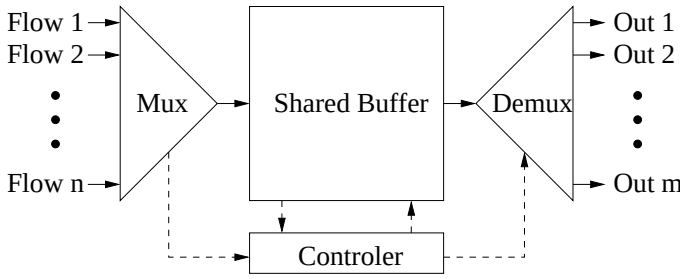


Fig. 1. The shared buffer switch may have a single or multiple outputs. Among the incoming flows, some are guaranteed performance flows that require lossless service.

effort flow, and sends it to the corresponding output link.

There have been some buffer management methods [15] [16] [17] proposed in the literature, but they mostly targeted at improving buffer utilization and minimizing packet loss. Our objective in this paper is to ensure lossless service for the guaranteed performance flows by investigating the buffer size requirement and buffer allocation strategies. Our analysis starts with the single output switch, and extends to the more general case where the switch may have multiple outputs and multicast flows. A universally applicable buffer allocation solution for ensuring lossless service is obtained, and its correctness is verified by simulation.

The rest of this paper is organized as follows. Section II introduces some preliminaries for the work in the paper. Section III analyzes buffer management for lossless service in the single output switch, and Section IV generalizes the results to the multiple output switch. Section V provides some important discussions. Section VI presents simulation results to verify the analytical results obtained in previous sections. Finally, Section VII concludes the paper.

II. PRELIMINARIES

In this section, we give some definitions and properties that will be used in the rest of this paper.

A. Traffic Shaping Schemes

Traffic shaping is necessary for the guaranteed performance flow when lossless service is considered. Clearly, if a flow has an unrestricted input rate or it can have burst arrival of any arbitrary size, there is no way to ensure lossless service. Instead, the guaranteed performance flow should be compliant with or restricted by some traffic shaping scheme. In this paper, we consider two traffic

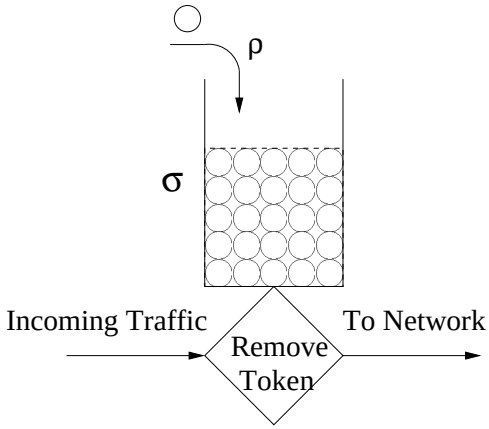


Fig. 2. A leaky bucket has a token bucket of capacity σ with new tokens flowing in at rate ρ.

shaping schemes: the peak rate scheme and the leaky bucket scheme.

We say a flow is peak rate ρ compliant if during any time interval of length t, the amount of traffic it injects into the network is less than or equal to ρt. In other words, the flow has a maximum input rate ρ, and in order to ensure lossless service, reserved bandwidth ρ is required along the transmission path of the flow. Although the peak rate scheme is simple, it is not efficient. Since only the peak transmission rate is indicated and the bandwidth is reserved according to this peak rate, the reserved bandwidth cannot be fully utilized. Nevertheless, the peak rate scheme can be applied to any network flow in reality, since after all, an actual flow is restricted by the maximum transmission rate of the physical link.

Another more efficient traffic shaping scheme is the leaky bucket scheme [18]. A flow is said to be leaky bucket (ρ, σ) compliant if during any time interval of length t, the amount of traffic it injects into the network is less than or equal to $\rho t + \sigma$. As illustrated in Fig. 2, a stream of tokens is generated at a constant rate ρ and flows into a bucket with capacity σ. The newly generated tokens are added to the bucket if it is not full, or discarded otherwise. For the flow to inject traffic into the network, the same amount of tokens have to be available and removed from the bucket. As indicated by the formula, ρ represents the long term average transmission rate of the flow, while σ defines the maximum size of an instantaneous burst. Since real network traffic is usually in a burst mode, the leaky bucket scheme is more efficient than the peak rate scheme in the sense that it is able to utilize the bucket to hold the

burst and requires lower reserved bandwidth. By varying the average rate and burst size, the leaky bucket scheme can be used to characterize a broad range of real traffic.

It is interesting to point out that the peak rate scheme can be viewed as a special case of the leaky bucket scheme with the burst size equal to zero. By using this property, the results for the leaky bucket scheme can be directly applied to the peak rate scheme. Thus we have the following property.

Property 1: A peak rate ρ compliant flow is leaky bucket $(\rho, 0)$ compliant.

B. Logical Traffic Combination

In order to simplify the analysis for a group of flows destined to the same destination, we define the combination of multiple flows to be a logical flow whose traffic is the sum of the traffic of each individual physical flow, i.e., the outgoing (incoming) packets of this logical flow is the outgoing (incoming) packets of all its member flows. We have the following properties regarding the combination of peak rate compliant flows or leaky bucket compliant flows.

Property 2: Assume that flow $f_1, \ldots, f_n$ are peak rate $\rho_1, \ldots, \rho_n$ compliant, respectively. The logical combined flow of these flows is peak rate $\sum_{i=1}^{n} \rho_i$ compliant.

Proof. During any time interval of length t, f_i has maximum arrived traffic of $\rho_i t$, and the logical combined flow has maximum arrived traffic of $\sum_{i=1}^{n} \rho_i t = (\sum_{i=1}^{n} \rho_i)t$. By definition, the logical combined flow is peak rate $\sum_{i=1}^{n} \rho_i$ compliant. ■

Property 3: Assume that flow $f_1, \ldots, f_n$ are leaky bucket $(\rho_1, \sigma_1), \ldots, (\rho_n, \sigma_n)$ compliant, respectively. The logical combined flow of these flows is leaky bucket $(\sum_{i=1}^{n} \rho_i, \sum_{i=1}^{n} \sigma_i)$ compliant.

Proof. During any time interval of length t, f_i has maximum arrived traffic of $\rho_i t + \sigma_i$, and the logical combined flow has maximum arrived traffic of $\sum_{i=1}^{n} (\rho_i t + \sigma_i) = (\sum_{i=1}^{n} \rho_i)t + \sum_{i=1}^{n} \sigma_i$. By definition, the logical combined flow is leaky bucket $(\sum_{i=1}^{n} \rho_i, \sum_{i=1}^{n} \sigma_i)$ compliant. ■

Similarly, a group of best effort flows, which may be unregulated, can also be viewed as a logical combined best effort flow.

C. Buffer Threshold Setting

Buffer threshold setting methods define the way that each individual flow utilizes the buffer space. The simplest method is complete sharing, in which the incoming packets of all the flows are put into the same buffer, and a new packet can be accepted as long as there is space in the common buffer. Complete sharing enables efficient buffer usage, but cannot provide isolation among flows. If an unregulated flow keeps injecting packets into the common buffer, all other flows will eventually have no space to accept new packets, and consequently receive no bandwidth. On the contrary, complete partitioning permanently divides the entire buffer space among different flows, and each flow can only use its assigned share. Complete partitioning prevents different flows from affecting each other, but may not fully utilize the entire buffer. It is possible that while the buffers of some flows are empty, other flows suffer from packet loss due to lack of buffer space.

In order to combine the advantages of both methods, in this paper we partition the buffer space into two parts, one for guaranteed performance flows and the other for best effort flows, and each part is shared by all the flows in that group. Since best effort flows are likely to be aggressive, assigning them an exclusive buffer offers protection to guaranteed performance flows. On the other hand, buffer utilization is improved by enabling buffer sharing within a group.

III. Lossless Service in Single Output Switch

In this section, we discuss buffer management for providing lossless service in a single output switch. The considered switch has buffer space B and a single output, which has bandwidth R and runs a FIFO scheduler. The incoming flows include a set of guaranteed performance flows $f_1, \ldots, f_n$, which will be treated as a logical combined flow f_g, and a set of best effort flows, which are combined as f_e. As indicated in Section II, the buffer space B is partitioned into two parts B_g for f_g and B_e for f_e, where $B_g + B_e = B$. All the guaranteed performance flows share B_g while all the best effort flows share B_e. The objective is to assure lossless service for the guaranteed performance flows under any traffic arrival.

A. Peak Rate Compliant Flows

First, we consider the case where all the guaranteed performance flows are peak rate compliant.

Theorem 1: Assume that flow $f_1, \ldots, f_n$ are peak rate $\rho_1, \ldots, \rho_n$ compliant, respectively. In order for the single output switch to assure lossless service, the guaranteed performance flows should be assigned a buffer with size proportional to the sum of their peak rates, i.e.,

$$B_g = \frac{\sum_{i=1}^{n} \rho_i}{R} B$$

Proof. By Property 2, f_g, the logical combination of $f_1, \ldots, f_n$, is peak rate $\rho_g = \sum_{i=1}^{n} \rho_i$ compliant.

We adopt a fluid model to effectively analyze the behavior of flows, in which the traffic of a flow arrives and leaves on an infinitesimal bit basis. We define a set of critical time points $t_0, t_1, \ldots, t_p, \ldots$. $t_0 (= 0)$ is the initial state, and t_{p+1} is the time that the last bit at t_p in B_g and B_e leaves the buffer, or in other words, the buffered content at t_p clears from the buffer at t_{p+1}. Since the output schedules traffic in a FIFO manner and the flows are served on an infinitesimal bit basis, traffic arriving at B_g and B_e at the same time leaves at the same time as well. Because f_g is regulated and f_e is aggressive, we can assume that B_g is empty and B_e is full at t_0.

Define $B_g(t)$ to be the amount of the actually buffered content of f_g at time t, and $B_g = \max\{B_g(t)\}$ is the buffer size of f_g in order to ensure lossless service. Next, we prove the following equation by induction.

$$B_g(t_p) = B_e \sum_{q=1}^{p} \left(\frac{\rho_g}{R}\right)^q$$

Base case: During $[t_0, t_1)$, f_e fills its buffer at rate R, and $B_g(t_1) = B_e \frac{\rho_g}{R}$.

Since at t_0 only f_e has buffered content, during $[t_0, t_1)$ only f_e is served at rate R, or it exclusively consumes all the bandwidth of the output. On the input side, f_g receives traffic at its peak rate ρ_g, and f_e behaves aggressively and fills its buffer at a rate equal to its output rate R. As a result, $t_1 - t_0 = \frac{B_e}{R}$, and at t_1, f_g has accumulated $B_g(t_1) = \rho_g(t_1 - t_0) = B_e \frac{\rho_g}{R}$ content in its buffer.

Inductive hypothesis: During $[t_{p-1}, t_p)$, f_e fills its buffer at rate $\frac{R}{\sum_{q=0}^{p-1}\left(\frac{\rho_g}{R}\right)^q}$, and $B_g(t_p) = B_e \sum_{q=1}^{p} \left(\frac{\rho_g}{R}\right)^q$.

During interval $[t_{p-1}, t_p)$, f_g keeps injecting traffic at its peak rate ρ_g, while the input rate of f_e is $\frac{R}{\sum_{q=0}^{p-1}\left(\frac{\rho_g}{R}\right)^q}$ as given in the inductive hypothesis. Following the FIFO scheduling principle, the output rates of f_g and f_e in $[t_p, t_{p+1})$ are proportional to their input rates in $[t_{p-1}, t_p)$. In other words,

$$\text{output rate of } f_e \text{ during } [t_p, t_{p+1}) = \frac{\frac{R}{\sum_{q=0}^{p-1}\left(\frac{\rho_g}{R}\right)^q}}{\rho_g + \frac{R}{\sum_{q=0}^{p-1}\left(\frac{\rho_g}{R}\right)^q}} R = \frac{R}{\sum_{q=0}^{p}\left(\frac{\rho_g}{R}\right)^q}$$

Since f_e behaves aggressively, it has the same input rate during $[t_p, t_{p+1})$ as its output rate. On the other hand, f_g continues to fill its buffer at rate ρ_g and

$$B_g(t_{p+1}) = \rho_g(t_{p+1} - t_p) = \rho_g \frac{B_e}{\frac{R}{\sum_{q=0}^{p}\left(\frac{\rho_g}{R}\right)^q}} = B_e \sum_{q=1}^{p+1} \left(\frac{\rho_g}{R}\right)^q$$

Thus, we have completed the proof for the inductive case that, during $[t_p, t_{p+1})$ f_e fills its buffer at rate $\frac{R}{\sum_{q=0}^{p}\left(\frac{\rho_g}{R}\right)^q}$, and $B_g(t_{p+1}) = B_e \sum_{q=1}^{p+1}\left(\frac{\rho_g}{R}\right)^q$.

Since $B_g(t_{p+1}) > B_g(t_p)$ and $B_g(t_p)$ has a limit when p goes to infinity, we have

$$B_g = \max\{B_g(t)\} = \lim_{p\to\infty} B_g(t_p) = B_e \frac{\rho_g}{R - \rho_g}$$

By $B_g + B_e = B$, we can obtain $B_g = \frac{\rho_g}{R} B$, or $B_g = \frac{\sum_{i=1}^{n}\rho_i}{R} B$. ■

As indicated by Theorem 1, when all the guaranteed performance flows are peak rate compliant, $B_g = \frac{\sum_{i=1}^{n}\rho_i}{R} B$ is sufficient and also necessary to guarantee lossless service for the guaranteed performance flows. It is sufficient because f_g will never have more than $\frac{\rho_g}{R} B$ buffered content. On the other hand, it is necessary because $B_g(t)$ may infinitely approach this value.

B. Leaky Bucket Compliant Flows

We now look at the situation where the guaranteed performance flows are leaky bucket compliant.

Theorem 2: Assume that flow $f_1, \ldots, f_n$ are leaky bucket $(\rho_1, \sigma_1), \ldots, (\rho_n, \sigma_n)$ compliant, respectively. In order for the single output switch to assure lossless service, the guaranteed performance flows should be assigned a buffer of size

$$B_g = \sum_{i=1}^{n} \sigma_i + \frac{\sum_{i=1}^{n}\rho_i}{R}\left(B - \sum_{i=1}^{n}\sigma_i\right)$$

Proof: By Property 3, f_g, the logical combination of $f_1, \ldots, f_n$, is leaky bucket $(\rho_g = \sum_{i=1}^{n}\rho_i, \sigma_g = \sum_{i=1}^{n}\sigma_i)$ compliant.

A leaky bucket compliant flow is different from a peak rate compliant flow in the way that it may have an instantaneous burst. In the following, we analyze the effect of a burst on buffer space

requirement of guaranteed performance flows. We assume that the size of the burst is σ and arrives at time t where $t_{s-1} \leq t < t_s$. Comparing with the previous case where all the guaranteed performance flows are peak rate compliant, an immediate result is that the buffered content of f_g at t_s should take the burst into consideration $B_g(t_s) = B_e \sum_{q=1}^{s} \left(\frac{\rho_g}{R}\right)^q + \sigma$.

During $[t_s, t_{s+1})$, the transmission of the traffic is the same as that with the peak rate compliant flows except for the burst. Since the burst was injected instantaneously, by the FIFO principle, when it is transmitted, it exclusively consumes all the bandwidth R. Therefore, when the burst is being transmitted, f_e cannot accept new traffic because no existing content has left and its buffer is still full. On the other hand, f_g keeps injecting traffic at rate ρ_g, and during the time interval that the burst is being transmitted, f_g has $\frac{\sigma}{R}\rho_g$ new content to be buffered. Thus, $B_g(t_{s+1}) = B_e \sum_{q=1}^{s+1} \left(\frac{\rho_g}{R}\right)^q + \sigma\frac{\rho_g}{R}$. We can see that the effect of the burst on buffer space requirement diminishes as time goes by. It is easy to prove that

$$B_g(t_p) = \begin{cases} B_e \sum_{q=1}^{p} \left(\frac{\rho_g}{R}\right)^q, & t_p < t_s \\ B_e \sum_{q=1}^{p} \left(\frac{\rho_g}{R}\right)^q + \sigma \left(\frac{\rho_g}{R}\right)^{p-s}, & t_p \geq t_s \end{cases}$$

Since $B_g(t_q) < B_g(t_p)$ for any $t_q < t_s \leq t_p$, in order to obtain B_g, we only need to consider $B_g(t_p)$ where $p \geq s$. Given $p \geq s$,

$$\frac{dB_g(t_p)}{dp} = \left(\frac{\rho_g}{R}\right)^p \ln\frac{\rho_g}{R} \left(\frac{\sigma}{\left(\frac{\rho_g}{R}\right)^s} - \frac{B_e\rho_g}{R-\rho_g} \right)$$

and $\left(\frac{\rho_g}{R}\right)^p \ln\frac{\rho_g}{R} < 0$. Therefore, $B_g(t_p)$ for $p \geq s$ may be an increasing, decreasing, or equivalent function depending on the values of σ and s, i.e., the size of the burst and its arriving time. In any case, the maximum value of $B_g(t_p)$ is obtained when either $p = s$ or $p = \infty$, i.e.,

$$\begin{aligned} B_g &= \max\{B_g(t_p)\} = \max\{B_g(t_s), \lim_{p\to\infty} B_g(t_p)\} \\ &= \max\left\{ B_e \sum_{q=1}^{s} \left(\frac{\rho_g}{R}\right)^q + \sigma, \frac{B_e\rho_g}{R-\rho_g} \right\} < \frac{B_e\rho_g}{R-\rho_g} + \sigma \end{aligned}$$

Thus, $B_g = \frac{B_e\rho_g}{R-\rho_g} + \sigma$ is sufficient to assure lossless service when f_g has a burst of size σ. It is also necessary. Considering that the burst comes at a time when s is approaching infinity,

$$\lim_{s\to\infty} B_g(t_s) = \lim_{s\to\infty} B_e \sum_{q=1}^{s} \left(\frac{\rho_g}{R}\right)^q + \sigma = \frac{B_e\rho_g}{R-\rho_g} + \sigma$$

The above analysis applies to any arbitrary burst of size σ. As discussed earlier, the effect of the burst on buffer space requirement diminishes as time goes by. Thus, if the burst arrives as different parts at different time, say, σ' coming at t' and σ'' at t'' and $\sigma' + \sigma'' = \sigma$, it is easy to see that $B_g = \frac{B_e \rho_g}{R - \rho_g} + \sigma$ is still sufficient to ensure lossless service. For the logical guaranteed performance flow f_g, it has a maximum possible burst of size σ_g, and therefore $B_g = \frac{B_e \rho_g}{R - \rho_g} + \sigma_g$. Since $B_g + B_e = B$, we have $B_g = \sigma_g + \frac{\rho_g}{R}(B - \sigma_g)$, or $B_g = \sum_{i=1}^{n} \sigma_i + \frac{\sum_{i=1}^{n} \rho_i}{R}(B - \sum_{i=1}^{n} \sigma_i)$. ■

C. Buffer Size for Best Effort Flows

It can be noticed that there is no requirement to the total buffer size, except that when all the flows are peak rate compliant, it should be larger than the sum of the bursts of all the flows. As long as the entire buffer (in the case that the guaranteed performance flows are peak rate compliant) or the entire buffer minus the portion for the bursts (in the case that the guaranteed performance flows are leaky bucket compliant) is partitioned proportionally, lossless service can be assured for guaranteed performance flows. However, in practice, the buffer size has to be larger than a specific value, because the lengths of real packets cannot be infinitely small. Furthermore, while lossless service is always assured for guaranteed performance flows, the total buffer size is critical to the packet loss ratios of the best effort flows and the throughput of the switch. If the total buffer size becomes larger, by a proportional partition, the buffer allocated for best effort flows is correspondingly larger, leading to smaller packet loss ratios and higher throughput.

In addition, the buffer size for best effort flows is also critical to the congestion control of the network. Network congestion occurs when too many flows attempt to send packets at high rates. In a congested network, packets experience extremely long delays due to the long queue lengths. Moreover, as the buffer space is always limited, when too many packets are received, some of them have to be dropped. In order to compensate for the lost packets due to buffer overflow, the sender has to retransmit them. When a packet is dropped along a path, the transmission capacity used at each of the upstream routers to forward the packet to the point at which it is dropped is wasted. It also may happen that although an acknowledgment is returned, it is significantly delayed, and the sender has

retransmitted the packet before receiving it, resulting in bandwidth wasting on forwarding unnecessary duplicate copies of the same packet.

The TCP/IP protocol stack has been the standard in the current Internet, and its congestion control mechanism is widely deployed. TCP uses a window based approach for congestion control [18]. When the transmitted packets are lost due to buffer overflow and no acknowledgment is returned to the sender, the sender would assume that there is a congestion in the network, and would reduce the size of the window. TCP uses an additive-increase, multiplicative-decrease (AIMD) algorithm to adjust the window size. The initial congestion window size is set to one, and is increased exponentially if all the sent TCP segments are positively acknowledged. When it reaches a specific threshold, the congestion window size grows linearly rather than exponentially. Due to the bandwidth limitation of the physical links, the window size cannot increase infinitely, and packet loss will occur eventually. When a timeout for the lost packet occurs, the value of the threshold is set to half the value of the current congestion window size, and the congestion window size is reset to one.

Since real network traffic is usually in bursts, a larger buffer size for best effort flows can help reduce packet loss caused by buffer overflow and improve the network congestion control. Assume that best effort flows have a small buffer. When a burst of packets comes, it may not be able to fit into the buffer and will be dropped, although the bandwidth is sufficient for the average traffic rate. Due to those dropped packets, the sender would assume that there is a congestion in the network and reduce the TCP congestion window size. However, this may not be necessary and can be avoided if the best effort flows are allocated a larger buffer. Traditionally, the buffer size for routers and switches has been calculated as the product of the round trip time (RTT) and the bandwidth [19]. The principle is to buffer all the sent packets before the first acknowledgment is received, which takes the round trip time to arrive after the packet transmission. In several recent works [20] [21] [22], it is reported that a much smaller buffer is sufficient for backbone routers.

There are many queueing based approaches in the literature to improve congestion control. A well known example is Random Early Detection (RED) [23]. It randomly discards packets in the earlier stage based on statistical probabilities to avoid TCP global synchronization. Those schemes

can be combined with the buffer management strategies proposed in this paper to improve congestion avoidance.

D. Multiple Output Queues with Complete Partitioning

The results obtained in the above are based on the assumption that there is only a single queue for all the guaranteed performance flows, i.e., the complete sharing buffer threshold setting method is used and B_g is shared by $f_1, \ldots, f_n$. We discuss below the situations when there are multiple output queues, each for a different guaranteed performance flow. In other words, the complete partitioning method is used, and each guaranteed performance flow f_i is assigned an exclusive buffer, denoted as B_i.

We first look at the case when all the performance guaranteed flows are peak rate complaint. If flow f_i is peak rate ρ_i compliant, by a similar analysis to that in Section III-A, we can draw the conclusion that a buffer of size $B_i = \frac{\rho_i}{R} B$ is sufficient and necessary for f_i to assure lossless service. Note that the total buffer space required for all the guaranteed performance flows in this case, i.e., $\sum_{i=1}^{n} B_i$, is the same as the corresponding B_g in Section III-A. Thus, complete sharing and complete partitioning make no difference when all the guaranteed performance flows are peak rate complaint.

Next, we will show that a single shared FIFO queue saves buffer space when the guaranteed performance flows are leaky bucket compliant. Assume that each flow has a separate queue and the complete partitioning method is used. If flow f_i is leaky bucket (ρ_i, σ_i) compliant, the buffer space necessary for f_i to ensure lossless service is

$$B_i = \sigma_i + \frac{\sum_{j \neq i} \sigma_j}{R} \rho_i + \left(B - \sum_{i=1}^{n} \left(\sigma_i + \frac{\sum_{j \neq i} \sigma_j}{R} \rho_i \right) \right) \frac{\rho_i}{R}$$

The effect caused by the burst σ_i of flow f_i on buffer space requirement includes two parts. On the one hand, σ_i more space is immediately needed to hold the burst. On the other hand, when the burst is being transmitted, all the guaranteed performance flows have a total of $\frac{\sigma_i}{R} \rho_g$ new traffic arrived that needs to be buffered. Since the two parts of traffic do not coexist and σ_i is larger than $\frac{\sigma_i}{R} \rho_g$, if complete sharing among guaranteed performance flows is adopted, σ_i is sufficient for the space requirement of both parts. However, when using complete partitioning, f_i needs σ_i space to buffer its own burst, and

needs $\frac{\sum_{j\neq i}\sigma_j}{R}\rho_i$ space in the worst case to buffer the traffic arrived when the bursts of all other flows are scheduled. It is easy to see that $\sum_{i=1}^{n} B_i$ is a larger value than the corresponding B_g in Section III-B, which means that a single queue is more efficient than multiple queues in this case.

Fair scheduling algorithms, such as WF2Q [4] and DRR [6], also use multiple output queues to buffer the packets of different flows, however, the above results for multiple queues cannot be directly applied to them, because those results are obtained under the assumption of a FIFO scheduling algorithm. Fair scheduling algorithms arrange the departure order of the buffered packets of different flows to ensure that each flow receives the same amount of service as what it should receive in the ideal Generalized Processor Sharing (GPS) model.

In fact, the buffer space requirement for lossless service of a fair scheduling algorithm is easy to analyze if its fairness guarantee is known. The fairness guarantee is the maximum difference between the amount of service that a flow should receive in the ideal GPS model and that it receives in the specific fair scheduling algorithm. For example, the fairness guarantee of WF2Q is L [4], where L is the maximum packet length. This indicates that the maximum service difference between GPS and WF2Q for any flow is at most the length of a single packet.

If the allocated bandwidth of flow f_i in GPS is ρ_i, then the average traffic rate of f_i should also be ρ_i. Otherwise, if the average traffic rate is too high, the arrived traffic will be infinitely accumulated; if the average traffic rate is too low, the leftover bandwidth is wasted. Under this assumption, we will analyze the buffer space requirement of a flow when the adopted fair scheduling algorithm has a fairness guarantee of θ. First, if f_i is peak rate ρ_i complaint, the buffer space needed for f_i to ensure lossless service is θ. The reasoning is straightforward. According to the definition of the peak rate scheme, during any interval of length t, f_i has maximum arrived traffic of $\rho_i t$. During the same interval, the service f_i should receive in the ideal GPS model is $\rho_i t$ [9], and with a fairness guarantee of θ, the service f_i receives in the fair scheduling algorithm is at least $\rho_i t - \theta$. As a result, the maximum amount of traffic stored in the buffer during the interval is θ, which is also the buffer space to ensure lossless service. Similarly, if f_i is leaky bucket (ρ_i, σ_i) compliant, the buffer space for lossless service

is $\theta + \sigma_i$, and the additional σ_i space is used to store the burst.

IV. Lossless Service in Multiple Output Switch

In the previous section, we have analyzed buffer management for lossless service in the single output switch. In this section, we extend the results to the more general situation where the switch has multiple outputs and a flow may be a multicast flow destined to more than one output. The considered switch has buffer space B, and m outputs $out_1, \ldots, out_m$, each of which runs a FIFO scheduler. For a specific output out_j, it has bandwidth R_j, shared by a set of guaranteed performance flows $f_{j1}, \ldots, f_{jn_j}$, which are leaky bucket $(\rho_{j1}, \sigma_{j1}), \ldots, (\rho_{jn_j}, \sigma_{jn_j})$ compliant respectively, and a set of best effort flows. The guaranteed performance flows destined to out_j are analyzed as a logical combined flow f_{jg}, and by Property 3, f_{jg} is leaky bucket $(\rho_{jg} = \sum_{i=1}^{n_j} \rho_{ji}, \sigma_{jg} = \sum_{i=1}^{n_j} \sigma_{ji})$ compliant. The best effort flows to out_j are analyzed as a logical combined flow f_{je}, and we define $\rho_{je} = R_j - \rho_{jg}$, which is the leftover bandwidth for all the best effort flows to out_j.

The traffic in the buffer of a multiple output switch may go to different outputs, and the buffer threshold setting in Section II needs to be extended to manage the shared buffer. To be specific, the entire buffer is partitioned into $m+1$ parts B_G and $B_{1e}, \ldots, B_{me}$. B_G is shared by all the guaranteed performance flows, regardless which output the flow is destined to, and B_{je} is shared by the best effort flows to out_j. The reason for the setting is that, since all the guaranteed performance flows are leaky bucket compliant, sharing the buffer among them improves the space utilization and lowers the operation complexity. On the other hand, because the best effort flows are not regulated and may be aggressive to fill any empty space they can access, assigning an exclusive buffer for the best effort flows to each output offers protection to the guaranteed performance flows. The protection is two-fold: first, a best effort flow cannot grab the buffer space for the guaranteed performance flows, and second, it cannot inject a large amount of traffic by using the buffer space for the best effort flows to other outputs.

There can be different ways to allocate buffer space for $B_{1e}, \ldots, B_{me}$. We have known from Section III that, as long as B_{je} is proportional to the bandwidth share ρ_{je}/R_j of the logical best

effort flow f_{je}, lossless service can be assured for the guaranteed performance flows to the same output. Since there is no requirement for the actual buffer size, the buffer allocation can be quite flexible. In this paper, we make the buffer sizes of different logical best effort flows proportional to their bandwidth, i.e.,

$$\frac{B_{1e}}{\rho_{1e}} = \frac{B_{2e}}{\rho_{2e}} = \cdots = \frac{B_{me}}{\rho_{me}}$$

which simplifies the buffer allocation, and also achieves fairness among the best effort flows to different outputs.

We are now ready to present the theorem for the pure unicast scenario.

Theorem 3: Assume that any flow is only destined to one output. In order for the multiple output switch to assure lossless service, the guaranteed performance flows should be assigned a buffer of size

$$B_G = \sum_{j=1}^{m}\sum_{i=1}^{n_j}\sigma_{ji} + \frac{\sum_{j=1}^{m}\sum_{i=1}^{n_j}\rho_{ji}}{\sum_{j=1}^{m}R_j}\left(B - \sum_{j=1}^{m}\sum_{i=1}^{n_j}\sigma_{ji}\right)$$

and the best effort flows to out_j should be assigned a buffer of size

$$B_{je} = \frac{R_j - \sum_{i=1}^{n_j}\rho_{ji}}{\sum_{j=1}^{m}R_j}\left(B - \sum_{j=1}^{m}\sum_{i=1}^{n_j}\sigma_{ji}\right)$$

Proof: We first consider the guaranteed performance flows to out_j, and define

$$B_{jg} = \sigma_{jg} + \frac{\rho_{jg}}{\sum_{j=1}^{m}R_j}\left(B - \sum_{k=1}^{m}\sigma_{kg}\right)$$

If we view out_j as a single output switch with R_j bandwidth and $B_j = B_{jg} + B_{je}$ buffer space, shared by f_{jg} and f_{je}, it is easy to see that

$$B_{jg} = \sigma_{jg} + \frac{\rho_{jg}}{R_j}\left(B_j - \rho_{jg}\right)$$

By Theorem 2, we know that B_{jg} is sufficient and necessary to ensure lossless service for f_{jg}. Since we consider only unicast flows, the flows to different outputs have no interference with each other. Thus, $B_G = \sum_{j=1}^{m} B_{jg}$ is sufficient and necessary to ensure lossless service for all the guaranteed performance flows. ■

Next, we will generalize the results by adding multicast flows into consideration. In order to save buffer space, the traffic of a multicast flow is usually stored as a single copy in the shared buffer, which

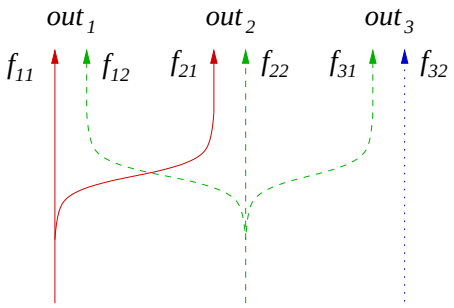


Fig. 3. A physical multicast flow may be labeled as different flows at different outputs.

will be transmitted to all its destination outputs. A pointer based queueing scheme, similar to that in [24], can be used to efficiently organize multicast content in the shared buffer.

A multicast flow may be locally labeled as different flows at different outputs. We define the following functions to represent the fanout property of a multicast flow.

$$F(f_{ji}) = \{\ out_k |\ out_k \text{ is one of the destinations of the physical (multicast) flow that } f_{ji} \text{ corresponds to}\}$$

For the example in Fig. 3, $F(f_{11}) = F(f_{21}) = \{out_1, out_2\}$, $F(f_{12}) = F(f_{22}) = F(f_{31}) = \{out_1, out_2, out_3\}$, and $F(f_{32}) = \{out_3\}$.

Then, update the expression of ρ_{ji} and σ_{ji} by adding multicast information,

$$\hat{\rho}_{ji} = \frac{\rho_{ji}}{|F(f_{ji})|}$$

$$\hat{\sigma}_{ji} = \frac{1}{|F(f_{ji})|}\left(\max\left\{\frac{\sigma_{kg}}{R_k}\rho_{ji} | out_k \in F(f_{ji})\right\} + \sigma_{ji} - \min\left\{\frac{\sigma_{ji}}{R_k}\sum_{|F(f_{kl})|=1}\rho_{kl} | out_k \in F(f_{ji})\right\}\right)$$

$\hat{\rho}_{ji}$ is the scaled rate for the labeled flow f_{ji}. Since the traffic of a multicast flow is stored only once, the scaled traffic rate for each labeled flow can be viewed as $\frac{1}{|F(f_{ji})|}$ of the original value ρ_{ji}. For example, in Fig. 3, a multicast flow is labeled as f_{11} at out_1 and f_{21} at out_2. ρ_{11} and ρ_{21} are the actual rate of the physical flow, and $\hat{\rho}_{11} = \hat{\rho}_{21} = \frac{\rho_{11}}{2} = \frac{\rho_{21}}{2}$.

The burst σ_{ji} of flow f_{ji} causes the increase of buffer space requirement in two aspects: to hold the burst itself, and to store the arriving traffic of all the guaranteed performance flows destined to the same output when the burst is transmitted. For a unicast flow, since the two parts of traffic never come together and the former is always larger than the latter, a total of σ_{ji} extra space is enough to

cover the buffer space increase caused by the burst: first to hold the burst itself and then to store the arriving traffic when the burst is transmitted. However, for a multicast flow, the two parts of traffic may simultaneously exist in the buffer, such as the situation that the burst has only been transmitted to part of the destinations of the multicast flow and still needs to be kept in the buffer for the remaining outputs.

$\hat{\sigma}_{ji}$ is the scaled extra buffer space of the labeled flow f_{ji} to smooth the bursts. There are two parts in the expression of $\hat{\sigma}_{ji}$. The first part is the buffer space for f_{ji} to buffer the arriving traffic when the bursts are transmitted. Different labeled flows may need different amounts of buffer space for the first part. Since the traffic of a multicast flow is buffered only once, the largest one among those of all the labeled flows, i.e., $\max\left\{\frac{\sigma_{kg}}{R_k}\rho_{ji}|out_k \in F(f_{ji})\right\}$, is counted. For example, in Fig. 3, f_{11} may have up to $\frac{\sigma_{1g}}{R_1}\rho_{11}$ traffic arrived when the burst σ_{1g} of the logical combined guaranteed performance flow f_{1g} is transmitted. For the other labeled flow f_{21} of the same multicast flow, the value is $\frac{\sigma_{2g}}{R_2}\rho_{21}$. Note that $\rho_{11} = \rho_{21}$. Assuming $\frac{\sigma_{1g}}{R_1}\rho_{11} \leq \frac{\sigma_{2g}}{R_2}\rho_{21}$, $\frac{\sigma_{2g}}{R_2}\rho_{21}$ should be counted. The second part of the expression is the space that f_{ji} needs to buffer its own burst σ_{ji}. It should be noted that, when the burst is transmitted by its last destination output, it will definitely not coexist and its space is able to store the arrived traffic $\frac{\sigma_{ji}}{R_j}\sum_{|F(f_{jl})|=1}\rho_{jl}$ of unicast flows to this output, which has been counted in the first part. In order to ensure sufficiency, we deduct the smallest one among those of all the outputs of a multicast flow, i.e. $\sigma_{ji} - \min\left\{\frac{\sigma_{ji}}{R_k}\sum_{|F(f_{kl})|=1}\rho_{kl}|out_k \in F(f_{ji})\right\}$. For example, in Fig. 3, f_{32} has $\frac{\sigma_{32}}{R_3}\rho_{32}$ traffic arrived when burst σ_{32} is transmitted. Since the arrived traffic $\frac{\sigma_{32}}{R_3}\rho_{32}$ has been counted in the first part of the expression, and it will not coexist with burst σ_{32}, we can safely deduct it. Finally, multiply the sum of the two parts by $\frac{1}{|F(f_{ji})|}$, and we obtain the scaled extra buffer space $\hat{\sigma}_{ji}$ of the labeled flow f_{ji} for the bursts.

Since best effort flows are always assumed to be aggressive to fill any empty buffer space, a multicast best effort flow has no difference in our analysis compared to a unicast best effort flow.

Theorem 4: In order for the multiple output switch to assure lossless service, the guaranteed performance flows should be assigned a buffer of size

$$B_G = \sum_{j=1}^{m}\sum_{i=1}^{n_j}\hat{\sigma}_{ji} + \frac{\sum_{j=1}^{m}\sum_{i=1}^{n_j}\hat{\rho}_{ji}}{\sum_{j=1}^{m}(\sum_{i=1}^{n_j}\hat{\rho}_{ji}+\rho_{je})}\left(B - \sum_{j=1}^{m}\sum_{i=1}^{n_j}\hat{\sigma}_{ji}\right)$$

and the best effort flows of out_j should be assigned a buffer of size

$$B_{je} = \frac{\rho_{je}}{\sum_{j=1}^{m}(\sum_{i=1}^{n_j}\hat{\rho}_{ji}+\rho_{je})}\left(B - \sum_{j=1}^{m}\sum_{i=1}^{n_j}\hat{\sigma}_{ji}\right)$$

Proof: Similar to the proof of Theorem 3, we view out_j as a single output switch shared by f_{jg} and f_{je}, with R_j bandwidth and $B_j = B_{jg} + B_{je}$ buffer space, where B_{jg} is defined as

$$B_{jg} = \sum_{i=1}^{n_j}\sigma_{ji} + \frac{\sum_{i=1}^{n_j}\rho_{ji}}{\sum_{j=1}^{m}(\sum_{i=1}^{n_j}\hat{\rho}_{ji}+\rho_{je})}\left(B - \sum_{j=1}^{m}\sum_{i=1}^{n_j}\hat{\sigma}_{ji}\right)$$

It is easy to see that

$$B_{jg} = \sigma_{jg} + \frac{\rho_{jg}}{R_j}(B_j - \rho_{jg})$$

By Theorem 2, B_{jg} is sufficient and necessary to ensure lossless service for f_{jg}. Thus $\sum_{j=1}^{m} B_{jg}$ should be sufficient to ensure lossless service of all the guaranteed performance flows. However, because there are multicast flows and the traffic of a multicast flow is buffered only once, the required buffer space is smaller. Following the above analysis, replacing ρ_{ji} by the scaled rate $\hat{\rho}_{ji}$ and σ_{ji} by the scaled burst $\hat{\sigma}_{ji}$ in the above formula, we obtain

$$\hat{B}_{jg} = \sum_{i=1}^{n_j}\hat{\sigma}_{ji} + \frac{\sum_{i=1}^{n_j}\hat{\rho}_{ji}}{\sum_{j=1}^{m}(\sum_{i=1}^{n_j}\hat{\rho}_{ji}+\rho_{je})}\left(B - \sum_{j=1}^{m}\sum_{i=1}^{n_j}\hat{\sigma}_{ji}\right)$$

and $B_G = \sum_{j=1}^{m}\hat{B}_{jg}$ is sufficient and necessary for providing lossless service. ■

It can be noticed that, for a unicast flow, $\hat{\rho}_{ji} = \rho_{ji}$ and $\hat{\sigma}_{ji} = \sigma_{ji}$, and therefore Theorem 4 also holds under the pure unicast scenario. In fact, it is the universally applicable solution for providing lossless service in shared buffer switches.

V. SOME DISCUSSIONS

In this section, we first discuss why and how the buffer allocation strategies should be adjusted for real packet switched networks, and then compare in detail the proposed buffer management schemes with fair scheduling algorithms.

A. Buffer Allocation Adjustment for Packet Switched Networks

So far all our discussions have been based on a fluid model, which greatly simplifies the analysis but is not practical for real packet switched networks. In the fluid model, traffic comes and leaves on an infinitesimal bit basis like a fluid flow. However, in the packet switched network, packets are the scheduling and transmission units. In order to ensure lossless service in the packet switched network, buffer allocation strategies obtained with the fluid analytical model have to be adjusted due to the packet fragmentation.

We first calculate the adjustment for the single output switch and then generalize the results to the multiple output switch. There are two differences between the fluid model and the packet switched network. First of all, in the fluid model, all flows to the same output are served simultaneously, while in the packet switched network, flows are served one by one, i.e., only one flow can be served at any instant. Thus, when a best effort flow is transmitting a packet, the guaranteed performance flows cannot release any buffer. Because the output uses a FIFO scheduler, best effort flows can continue sending packets in a time period only if no packet of the guaranteed performance flows arrive. The maximum length of such a period is $\frac{L}{\max\{\rho_l\}}$, where L is the maximum packet length. During this period, the guaranteed performance flow f_i sends no packet in the packet switched network, but can transmit up to $\frac{L}{\max\{\rho_l\}}\rho_i$ traffic in the fluid model, which should be compensated in the adjustment. As to the second difference, in the fluid model, flows are served on an infinitesimal bit basis, i.e., a bit is immediately released from the buffer after it has been transmitted. On the contrary, in the packet switched network, a packet will not be removed from the buffer until it has been completely transmitted. Even part of the packet has been sent to the output, the corresponding buffer space is still occupied. With the maximum packet length of L, a packet can be transmitted for as long as $\frac{L}{R}$. During this interval, f_i cannot release any occupied buffer in the packet switched network, but is able to obtain up to $\frac{L}{R}\rho_i$ free space in the fluid model, which should also be considered in the adjustment. If we use A_i to denote the adjustment for f_i in a single output switch, we have the following expression of A_i by adding the above two components.

$$A_i = \frac{L}{\max\{\rho_l\}}\rho_i + \frac{L}{R}\rho_i$$

In a multiple output switch with multicast incoming flows, the general expression for the adjustment of f_{ji} is given as follows:

$$A_{ji} = \max\left\{\frac{L}{\max\{\rho_{kl}\}}\hat{\rho}_{ji} | out_k \in F(f_{ji})\right\} + \max\left\{\frac{L}{R_k}\hat{\rho}_{ji} | out_k \in F(f_{ji})\right\}$$

The idea is to choose the largest value among those of all the labeled flows and use the scaled rate to replace the actual rate. As a result, the universally applicable formulas in Theorem 4 should be modified as follows to reflect the adjustment for the packet fragmentation.

$$\begin{aligned} B_G &= \sum_{j=1}^{m}\sum_{i=1}^{n_j}(\hat{\sigma}_{ji} + A_{ji}) + \frac{B - \sum_{j=1}^{m}\sum_{i=1}^{n_j}(\hat{\sigma}_{ji} + A_{ji})}{\sum_{j=1}^{m}(\sum_{i=1}^{n_j}\hat{\rho}_{ji} + \rho_{je})}\sum_{j=1}^{m}\sum_{i=1}^{n_j}\hat{\rho}_{ji} \\ B_{je} &= \frac{B - \sum_{j=1}^{m}\sum_{i=1}^{n_j}(\hat{\sigma}_{ji} + A_{ji})}{\sum_{j=1}^{m}(\sum_{i=1}^{n_j}\hat{\rho}_{ji} + \rho_{je})}\rho_{je} \end{aligned}$$

B. Comparison with Fair Scheduling Algorithms

In this subsection, we compare in detail the two approaches for differentiated service: buffer management schemes and fair scheduling algorithms. The basic idea of differentiated service is to provide isolation between the flows and guarantee the amount of bandwidth that each flow should receive. Buffer management schemes, as the first defense line, restrict the amount of traffic that different flows can inject into the buffer and thus indirectly control the bandwidth consumption. Fair scheduling algorithms, on the other hand, assume that each flow always has packets in the buffer, and decide the transmission order of the buffered packets to ensure that each flow receives its desired amount of bandwidth.

Due to their low computational overheads, buffer management schemes can provide differentiated service at high speed and low hardware cost. The main advantage of buffer management schemes over fair scheduling algorithms is that there is no extra computation associated with each packet transmission. Buffer management schemes only need to recalculate the buffer allocation when a new flow arrives or an existing flow leaves. For packet transmissions, the FIFO scheduler simply continuously removes the head packet of the queue and sends it out. Since there is no need to decide which packet

to transmit next, the buffer management schemes in conjunction with the FIFO scheduler can transmit packets at high speed. In addition, the FIFO scheduler does not need the support of special hardware and can achieve low implementation cost. On the contrary, in order to accelerate the scheduling process, fair scheduling algorithms usually need the support of special devices. For example, WFQ needs two registers for each flow to store the virtual start time and virtual finish time of its head packet, and it needs comparators to find the smallest time stamp; DRR also needs a register for each flow to store its remaining transmission quota, and a priority encoder to implement the round robin selection.

It should be noted that, without the help of proper buffer management schemes, fair scheduling algorithms alone are difficult to provide guaranteed performance service, as will be seen in the simulation results in Section VI. The reason is that fair scheduling algorithms assume that each flow always has buffered packets, and their job is to decide the transmission order of those packets. Without a proper buffer management scheme, it may happen that when it is the turn of a flow to transmit a packet, it has no packet available in the buffer. In this case, the fair scheduling algorithm has to select a packet of another flow due to the work-conservation property, meaning that the fair scheduling algorithm will transmit a packet as long as there is one in the buffer. Thus, when guaranteed performance flows and best effort flows share the same buffer, it is possible for a malicious flow to inject a lot of packets and occupy most of the buffer space, so that the fair scheduling algorithm can send only its packets. In this sense, when the buffer size is finite in a practical scenario, fair scheduling algorithms can only provide guaranteed performance service when there are supporting buffer management schemes.

On the other hand, the trade-off for buffer management schemes is that the performance guarantees they provide are not as tight as fair scheduling algorithms. Fair scheduling algorithms are able to achieve good worst-case performance guarantees, and are suitable for applications with tight bandwidth or delay requirements.

VI. Simulation Results

In this section, we present our simulation results to verify the analytical results obtained in the previous sections, and compare the proposed methods with fair scheduling algorithms.

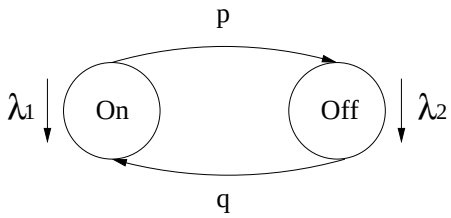


Fig. 4. In a Markov modulated Poisson process, the intensity of the Poisson process depends on the state of the Markov chain.

Since most realistic networks are packet switching based, our simulations are conducted in a packet switched network. We assume that the packet length is uniformly distributed in the range $[100, 300]$ bytes. In other words, use $l(x)$ to denote the length distribution density function, and we have

$$l(x) = \begin{cases} \frac{1}{201}, & 100 \leq x \leq 300 \\ 0, & \text{otherwise} \end{cases}$$

Because the guaranteed performance flow is leaky bucket compliant, we emulate it with a constant bit rate (CBR) flow, and the burst may arrive at any time during the simulation run. On the other hand, the best effort flow is emulated by the Markov modulated Poisson process, to reflect the burst nature of real network traffic, as illustrated in Fig. 4. In a Markov modulated Poisson process, the intensity of a Poisson process is defined by the state of a Markov chain. The Markov chain has two states: on and off. In the on state, the intensity of the Poisson process is λ_1, and in the off state the intensity is λ_2. The probability to switch from the on state to the off state is p, and the probability to switch from the off state to the on state is q. In the simulations, we set $p = q = 0.2$ and $\lambda_2 = 0$, and change the value of λ_1 to adjust the load of the best effort flow. Each simulation run lasts for 10^5 simulation seconds in order to obtain stable statistics.

A. Single Output Switch

The purpose of the first simulation is to verify Theorem 1. We set up a single output switch with 1M(10^6) bps bandwidth and 5K(10^3) bytes buffer space. There are two flows, flow 1 is a guaranteed performance flow, which is peak rate 600K bps compliant, and flow 2 is a best effort flow, with load varying from 100K bps to 2M bps.

First, we allocate buffer space according to Theorem 1, i.e., $B_g = 5000 \times 0.6 = 3000$ bytes for

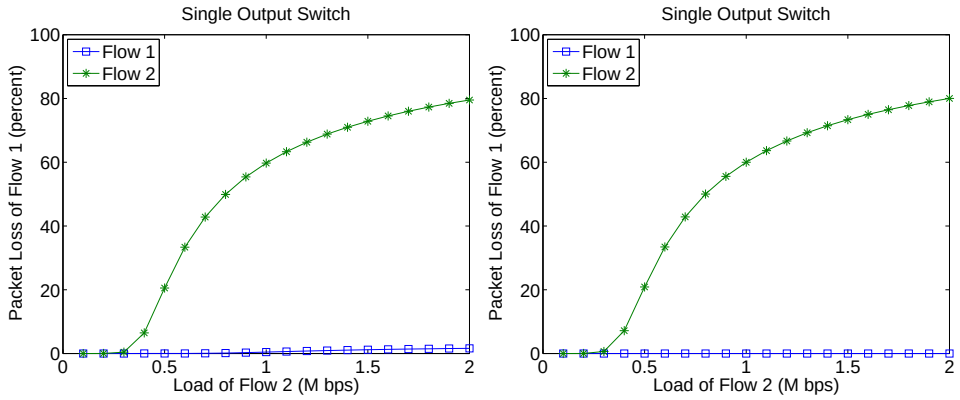
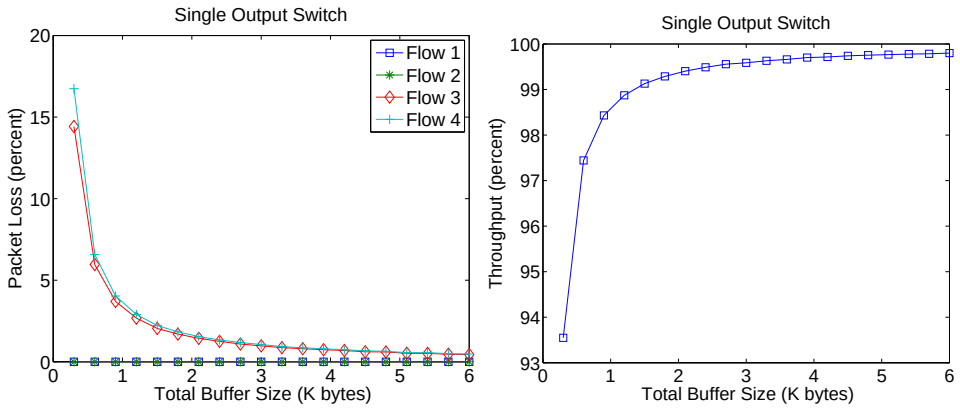


Fig. 5. The buffer allocation for lossless service in the packet switched network needs to be adjusted due to packet fragmentation. (a) Packet loss before adjustment. (b) Packet loss after adjustment.

flow 1 and $B_e = 5000 - 3000 = 2000$ bytes for flow 2. The packet loss of flow 1 and flow 2 is plotted in Fig. 5(a). Unfortunately, flow 1 still suffers packet loss, although its packet loss ratio is smaller comparing with that of flow 2. Furthermore, the packet loss ratio of flow 1 increases gradually as the load of flow 2 increases.

The reason for the inconsistency between the analytical results and the simulation results is that, while the analysis is based on a fluid model, the simulation is conducted in a packet switched network. In Section V-A, we have analyzed how to adjust buffer allocation for packet switched networks. In this case, the adjustment is $300 + 180 = 480$ bytes for flow 1. After the adjustment, $B_g = 480 + (5000 - 480) \times 0.6 = 3192$ bytes and $B_e = 5000 - B_g = 1808$ bytes, and the packet loss of flow 1 and flow 2 is given in Fig. 5(b). It can be seen that flow 1 has zero packet loss now.

Next, we exam the relationship between the total buffer size and the packet loss ratio of the best effort flows. We consider a more complex scenario. The single output switch has 1M bps bandwidth and buffer size varying from 3K to 60K bytes. There are two guaranteed performance flows. Flow 1 is peak rate 200K bps compliant and flow 2 is leaky bucket (400K bps, 4K bits) compliant. There are two more best effort flows, where flow 3 has a load of 100K bps and flow 4 has a load of 300K bps. The buffer is allocated according to Theorem 2 with the adjustment for packet fragmentation. The adjustment for flow 1 and flow 2 is 210 bytes and 420 bytes respectively. Thus, $B_g = 4K/8 + 630 + (B - 4K/8 - 630) \times 0.6$ and $B_e = B - B_g$. The packet loss ratios of the four flows are plotted



(a) (b)

Fig. 6. Effect of total buffer size on packet loss and throughput. (a) Packet loss decreases as the total buffer size increases. (b) Throughput increases as the total buffer size increases.

in Fig. 6(a). As can be seen, lossless service is assured for flow 1 and flow 2. On the other hand, flow 3 and flow 4 have similar packet loss ratios, and the values drop as the total buffer size increases. The throughput of the switch is given in Fig. 6(b). As the increase of the total buffer size, the throughput of the switch is steadily increasing, and finally approaching 100%.

B. Multiple Output Switch

In this simulation, we verify Theorem 4. The simulation is set up based on the example in Fig. 3. The switch has three outputs out_1, out_2 and out_3. The total buffer size is 15K bytes, and the bandwidth for each output is 1M bps. There are three guaranteed performance flows. Flow 1 is a multicast flow to out_1 (labeled as f_{11}) and out_2 (as f_{21}), and is leaky bucket (200K bps, 2K bits) compliant. Flow 2 is a multicast flow to out_1 (as f_{12}), out_2 (as f_{22}) and out_3 (as f_{31}), and is leaky bucket (400K bps, 4K bits) compliant. Flow 3 is a unicast flow to out_3 (as f_{32}), and is peak rate 200K bps compliant. There are also three best effort unicast flows, destined to the three outputs respectively. We let them have the same load, which varies from 100K bps to 2M bps.

The buffers are allocated according to Theorem 4 with adjustment:

$\hat{\sigma}_{11} = \hat{\sigma}_{21} = 400/2$

$\hat{\sigma}_{12} = \hat{\sigma}_{22} = \hat{\sigma}_{31} = 800/3$

$\hat{\sigma}_{32} = 100$

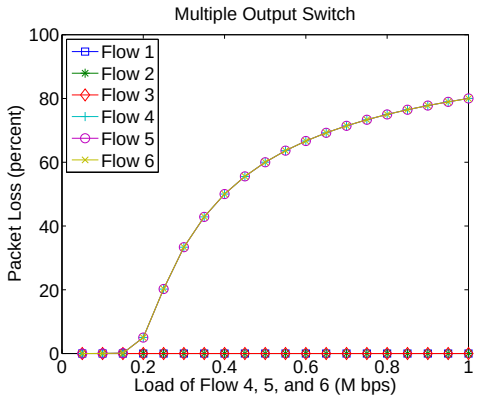


Fig. 7. By saving a multicast packet as a single copy in the share buffer, the buffer space requirement for ensuring lossless service is much smaller.

$$A_{11} = A_{21} = 210/2$$

$$A_{12} = A_{22} = A_{31} = 420/3$$

$$A_{32} = 210$$

Thus, we can obtain $B_G = 2140 + (15000 - 2140) \times 0.4 = 7284$ bytes and $B_{1e} = B_{2e} = B_{3e} = 2572$ bytes. The packet loss ratio of each flow is shown in Fig. 7. As can be seen, lossless service is assured for the three guaranteed performance flows, and the best effort flows lose more packets as their load increases. On the contrary, if a multicast packet is saved as multiple unicast packet copies, B_G has to be greater than 9000 bytes to guarantee lossless service, because for all the three outputs, the input rate of the guaranteed performance flows is 60% of the output bandwidth.

C. Comparison with Fair Scheduling Algorithms

In this simulation, we compare our methods with two typical fair scheduling algorithms. Weighted Fair Queuing (WFQ) [10] and Deficit Round Robin (DRR) [6] are considered, which are representatives of time stamp based algorithms and round robin based algorithms, respectively.

We set a single output switch with 1M bps bandwidth and 5K bytes buffer space. There are four unicast flows. Flow 1 is leaky bucket (100K bps, 1K bits) compliant, flow 2 is peak rate 200K bps compliant, and flow 3 is leaky bucket (300K bps, 3K bits) compliant. Flow 4 is a best effort flow, and we let its load increase from 100K bps to 2M bps.

The reserved bandwidth of each guaranteed performance flow is its input rate, i.e., 100K bps for

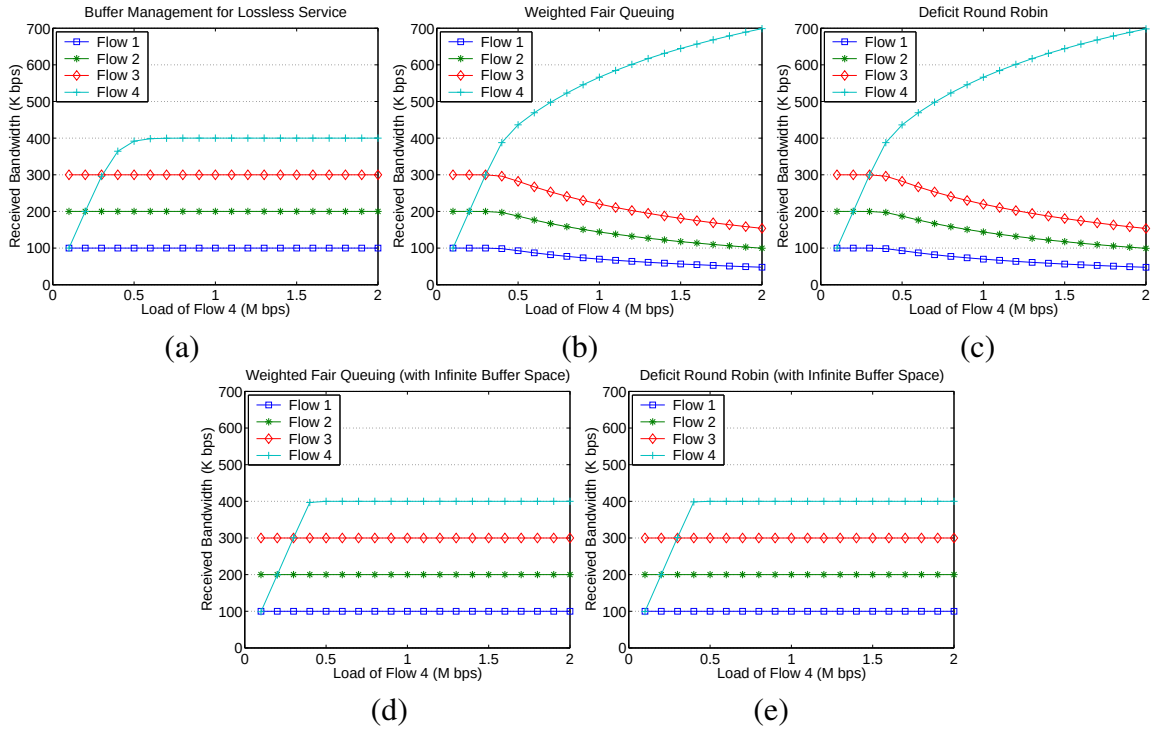


Fig. 8. Actually received bandwidth of each flow with different methods. (a) Buffer management for lossless service. (b) Weighted Fair Queuing. (c) Deficit Round Robin. (d) Weighted Fair Queuing (with Infinite Buffer Space). (e) Deficit Round Robin (with Infinite Buffer Space).

flow 1, 200K bps for flow 2 and 300K bps for flow 3 and the leftover bandwidth 400K bps is regarded as the reserved bandwidth of the best effort flow (flow 4). For the fair scheduling algorithms, all the four flows share the entire buffer without isolation between guaranteed performance flows and best effort flows.

The actually received bandwidth of each flow under buffer management for lossless service is shown in Fig. 8(a). We can see that, regardless of the load of flow 4, the reserved bandwidth of flow 1, flow 2 and flow 3 is guaranteed. The simulation results under WFQ and DRR are plotted in Fig. 8(b) and Fig. 8(c) respectively. As can be seen, although the received bandwidth of flow 1, flow 2 and flow 3 is still approximately proportional to their reserved bandwidth, it is severely affected by the load of flow 4, and drops dramatically as the load of flow 4 increases. The results indicate that, without proper buffer management, fair scheduling algorithms alone are difficult to provide bandwidth guarantees. This is consistent with the analysis in Section V-B. On the other hand, when the buffer size is infinite, the bandwidth of different flows under WFQ and DRR is shown in Fig. 8(d) and Fig.

8(e), respectively. As can be seen, both WFQ and DRR achieve perfect fairness in this case.

VII. Conclusions

Comparing with fair scheduling algorithms, buffer management does not need to keep a separate queue for each flow and maintain associated state variables, and thus can provide differentiated service at high speed with low cost. In this paper, we have studied using buffer management to ensure lossless service for guaranteed performance flows in shared buffer switches. By adopting the discussed buffer management methods, the considered switches can efficiently provide guaranteed performance service in a high speed network.

Our analysis started with the simpler case where the considered switch has only one output. We extended the results to the more general case for multiple output switches with multicast flows, and obtained a universally applicable buffer allocation solution for assuring lossless service. We conducted simulations to verify the analytical results, and discovered the buffer requirement difference between the fluid analytical model and the packet switched network due to packet fragmentation. A general formula was then provided for the adjustment, and after adjusting the buffer allocation for packet fragmentation, the simulation results demonstrated consistency with the analytical results. This indicates that our analytical results in conjunction with the packet fragmentation adjustment can model buffer management for lossless service in shared buffer switches well. We also compared our methods with two typical fair scheduling algorithms by simulation, and the results showed that our methods provide better bandwidth guarantees than the fair scheduling algorithms alone.

Acknowledgments

This research work was supported in part by the U.S. National Science Foundation under grant numbers CCR-0207999 and CCF-0744234.

References

[1] S. Blake et. al., "A framework for differentiated services," *RFC 2475*, Dec. 1998.

[2] J. Song, M. Chang, S. Lee, and J. Joung, "Overview of ITU-T NGN QoS Control," *IEEE Communications Magazine* vol. 45, no. 9, pp. 116-123, Sep. 2007.

[3] M. Garett, "A service architecture for ATM: from applications to scheduling," *IEEE Network Magazine*, pp. 6-14, May 1996.

[4] H. Zhang, "WF2Q: worst-case fair weighted fair queueing," *IEEE INFOCOM'96*, pp. 120-128, San Francisco, CA, Mar. 1996.

[5] P. Valente, "Exact GPS simulation with logarithmic complexity, and its application to an optimally fair scheduler," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1454-1466, Dec. 2007.

[6] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 375-385, Jun. 1996.

[7] D. Pan and Y. Yang, "Credit based fair scheduling for packet switched networks," *IEEE INFOCOM'05*, pp. 843-854, Miami, FL, March 2005.

[8] X. Yuan and Z. Duan, "Fair round robin: A low complexity packet scheduler with proportional and worst-case fairness," *IEEE Transactions on Computers*, vol. 58, no. 3, pp. 365-379, Mar. 2009.

[9] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 344-357, Jun. 1993.

[10] A. Demers, S. Keshav and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM'89*, vol. 19, no. 4, pp. 3-12, Austin, TX, Sep. 1989.

[11] J. Xu and R. Lipton, "On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms", *ACM SIGCOMM'02*, Pittsburgh, PA, Aug. 2002.

[12] Q. Zhao and J. Xu, "On the computational complexity of maintaining GPS clock in packet scheduling", *IEEE Infocom 2004*, Hong Kong, Mar. 2004.

[13] R. Guerin, S Kamat, V. Peris and R. Rajan, "Scalable QoS provision through buffer management", *ACM SIGCOMM'98*, pp. 29-40, 1998.

[14] S. Cheung and C. Pencea, "Pipelined sections: a new buffer management discipline for scalable QoS provision", *IEEE INFOCOM'01*, pp. 1530-1538, Anchorage, Alaska, Apr. 2001.

[15] D. Lin and R. Morris, "Dynamics of random early detection," *ACM SIGCOMM'97*, pp. 127-l37, Sophia Antipolis, France, Sep. 1997.

[16] A. Romanow and S. Floyd, "Dynamics of TCP traffic over ATM networks," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 4, pp. 633-641, May 1995.

[17] J. Turner, "Maintaining high throughput during overload in ATM switches," *IEEE INFOCOM'96*, pp. 287-295, San Francisco, CA, Apr. 1996.

[18] J. Kurose and K. Ross, *Computer Networking: a Top-down Approach Featuring the Internet,* Addison Wesley, 3rd edition, May 2004.

[19] C. Villamizar and C. Song, "High performance TCP in ANSNET," *ACM Computer Communications Review*, vol. 24, no. 5, pp. 45-60, 1994.

[20] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown and T. Roughgarden, "Routers with very small buffers," *IEEE INFOCOM'06*, Barcelona, Spain, April 2006.

[21] D. Wischik and N. McKeown, "Part I: Buffer sizes for core routers," *ACM/SIGCOMM Computer Communication Review*, vol. 35, no. 3, July 2005.

[22] G. Appenzeller, I. Keslassy and N. McKeown, "Sizing router buffers," *SIGCOMM'04*, pp. 281-292, New York, 2004.

[23] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413, Aug. 1993.

[24] D. Pan and Y. Yang, "FIFO based multicast scheduling algorithm for VOQ packet switches," *IEEE Trans. Computers*, vol. 54, no. 10, pp. 1283-1297, October 2005.