Buffer Management for Lossless Service in Network Processors

Deng Pan and Yuanyuan Yang

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 in high speed networks. On the contrary, buffer management and FIFO scheduling need 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 network processors. We investigate the buffer size requirement and buffer allocation strategies by starting with the single output network processor and then extending the analytical results to the general multiple output network processor. A universally applicable buffer allocation method for assuring lossless service is obtained, and the correctness of the theoretical results is verified through simulations.

Keywords: Buffer management, network processor, differentiated service, lossless service

I. INTRODUCTION

There has been a lot of research in providing differentiated service, focusing on fair scheduling and buffer management. Fair scheduling algorithms [1] [2] [3] organize incoming packets on a per flow basis, and emulate the ideal GPS [4] model to fairly schedule the packets of each flow according to its reserved bandwidth. By using different methods, such as time stamp or round robin, they can reach different levels of fairness guarantee with different costs. However, it has been reported in [5] that there exists a fundamental tradeoff between the delay bound that an algorithm can achieve and its computational complexity, which means that fair schedulers either suffer from long worst case delay or high 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 scale or implement in high speed networks. The traditional FIFO scheduler can perform scheduling quickly, but may not schedule the flows in a fair way. However, in conjunction with proper buffer management methods, the simple FIFO scheduler can also provide differentiated service [6] [7]. (Unfortunately, Proposition 2 and its consequent results in Section 2 of [6] were wrong, which we provide corrections in this paper.) Buffer management offers protection to guaranteed performance flows as the first defense line, by preventing other flows from injecting excessive number of packets. The FIFO scheduler and buffer management typically require only a constant amount of processing and state information [6], and are able to work in a high speed environment.

On the other hand, network processors [8] [9] [10] have been demonstrating advantages as specialized equipment for efficiently processing packet admission, classification and transmission. Due to their special characteristics, network processors are required to work with high speed and be cheaply implementable and easily scalable. In this paper, we discuss using buffer management to provide lossless service in network processors. The considered network processor 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. The rest incoming flows are best effort flows, which may be aggressive and inject packets to any empty space it can access.

There have been some buffer management methods [11] [12] [13] proposed in the literature, but they mostly target 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 the single output network processor, and extends to the more general case where the network processor may have multiple outputs and multicast incoming flows. A universally applicable buffer allocation solution for ensuring lossless service is obtained, and its correctness is verified by simulation.

II. PRELIMINARIES

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

A. Traffic Shaping Schemes

Traffic shaping is necessary for the guaranteed performance flow when lossless service is considered. It is obvious that if a flow has unrestricted input rate or it can have burst arrival of arbitrary size, there is no way to ensure lossless service. Instead, the incoming guaranteed performance flow should be compliant with or restricted by some traffic shaping scheme. In this paper, we consider two traffic 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 that 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, a reserved bandwidth ρ is required along the transmission path of the flow.

Another more efficient traffic shaping scheme is the leaky bucket scheme [14]. A flow is said to be leaky bucket (ρ, σ) compliant if during any time interval of length t, the traffic that it injects into the network is less than or equal to $\rho t + \sigma$. 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 can use the bucket to hold the burst and requires lower reserved bandwidth.

It is interesting to note that the peak rate scheme can be viewed as a special case of the leaky bucket scheme with the burst size equal to zero.

This research was supported by the U.S. National Science Foundation under grant numbers CCR-0073085 and CCR-0207999.

Deng Pan is with Dept. of Computer Science, State University of New York, Stony Brook, NY 11794, USA.

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

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

B. Logical Traffic Combination

In order to simplify the analysis of 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 the 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 combination of the flows 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 combination of the flows 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 between flows. 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 make full use of the entire buffer.

In order to combine the advantages of both methods, in this paper we partition buffer space into two parts, one for the guaranteed performance flows and the other for the 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 the guaranteed performance flows. On the other hand, the buffer utilization improves by enabling group members to share the common buffer space.

III. Lossless Service in Single Output Network Processor

In this section, we discuss buffer management for providing lossless service in a single output network processor. The considered network processor 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 network processor 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.

A fluid model is adopted 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 is transmitted at the same time as well. Because f_g is regulated and f_e is aggressive, we can safely 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. It is easy to prove by induction that

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

The proof is omitted due to space limitation.

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 network processor 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 to the buffer space requirement of the guaranteed performance flows. We assume that the size of the burst is σ and arrives at time t where $t_{s-1} \le t < t_s$. Comparing with the previous case where all the guaranteed performance flows are peak rate compliant, an immediate result of the burst 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. By the FIFO principle, when the burst is transmitted, it exclusively consumes all the bandwidth R, since the burst was injected instantaneously. Therefore, when the burst is being transmitted, f_e cannot accept new traffic because no existing content is leaving and its buffer is still full. On the other hand, f_g keeps injecting traffic with 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 to the 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 $B_g(t_p) > B_g(t_q)$ 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.,

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

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 arbitrary burst size σ. As discussed earlier, the effect of the burst to the 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 $B_g = \sum_{i=1}^{n}\sigma_i + \frac{\sum_{i=1}^{n}\rho_i}{R}(B - \sum_{i=1}^{n}\sigma_i)$. ■

IV. Lossless Service in Multiple Output Network Processor

In the previous section, we have analyzed buffer management for lossless service in the single output network processor. In this section, we extend the results to the more general situation where the network processor has multiple outputs and a flow may be a multicast flow destined to more than one outputs. The considered network processor 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.

The traffic in the buffer of a multiple output network processor may go to different outputs, and the buffer threshold setting in Section II need 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, no matter which output the flow is destined to, and B_{je} is shared by the best effort flows to out_j. This method offers two folds of protection to the guaranteed performance flows: firstly, a best effort flow cannot grab the buffer space for the guaranteed performance flows, and secondly, it cannot inject large amount of traffic by using the buffer space for the best effort flows to other outputs.

There can be many different ways to allocate buffer space for $B_{1e}, \ldots, B_{me}$. In this paper, we make the buffer size of different logical best effort flow proportional to its 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 some extent of 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 network processor 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 network processor 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

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

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

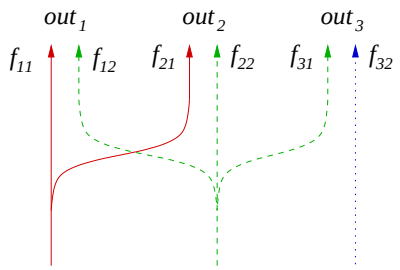


Fig. 1. A physical multicast flow may be labelled as different flows at different outputs.

buffer, which will be transmitted to all its destination outputs. A pointer based queueing scheme, similar to that in [15], can be used to efficiently organize multicast content in the shared buffer.

A multicast flow may be locally labelled 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. 1, $F(f_{11}) = F(f_{21}) = \{out_1, out_2\}$, $F(f_{12}) = F(f_{22}) = F(f_{31}) = \{out_1, out_2, out_3\}$, 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 labelled flow f_{ji}. Since the traffic of a multicast flow is stored only once, the scaled traffic rate for each labelled flow can be viewed as $\frac{1}{|F(f_{ji})|}$ of the original value ρ_{ji}. For example, in Fig. 1, a multicast flow is labelled 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 increase caused by the burst, first to hold the burst itself and then to store the arriving traffic when the burst is scheduled. 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 but still needs to be kept in the buffer for the rest outputs.

$\hat{\sigma}_{ji}$ is the scaled extra buffer space of the labelled 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 labelled flows may need different amount 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 labelled 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. 1, 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 f_{21}, the value is $\frac{\sigma_{2g}}{R_2}\rho_{21}$. Note that $\rho_{11} = \rho_{21}$. Assuming $\frac{\sigma_{1g}}{R_1}\rho_{11} \le \frac{\sigma_{2g}}{R_2}\rho_{21}$, then $\frac{\sigma_{2g}}{R_2}\rho_{21}$ 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 with and its space is able to cover the arrived traffic $\frac{\sigma_{ji}}{R_j}\sum_{|F(f_{jl})|=1}\rho_{jl}$ of the unicast flows to this outputs, 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. 1, f_{32} has $\frac{\sigma_{32}}{R_3}\rho_{32}$ traffic arrived when the 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 the 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 labelled 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 network processor 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 network processor 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

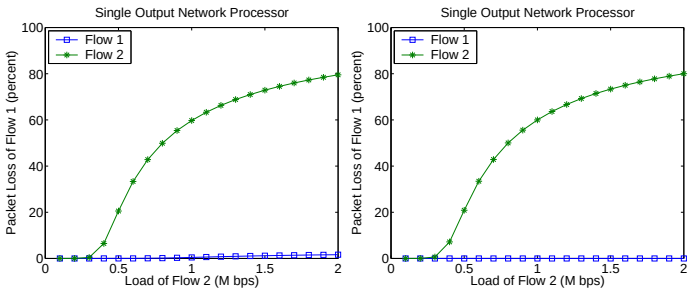
$$B_{jg} = \sigma_{jg} + \frac{\rho_{jg}}{R_j}(B_j - \rho_{jg})$$

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 network processors.



(a) (b)

Fig. 2. 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.

V. Simulation Results

In this section, we conduct simulations to verify the analytical results obtained in the previous sections. Since most realistic networks are packet switching based, the simulations are conducted in a packet switched network. The packet length is uniformly distributed in the range $[100, 300]$ bytes. (The simulations are also conducted with constant and normally distributed packet length, and similar results are obtained, which are not presented in this paper due to space limitation.)

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. 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 Network Processor

The purpose of the first simulation is to verify Theorem 1. We set up a single output network processor 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 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. 2(a). Unfortunately, flow 1 still suffers packet loss, although its packet loss ratio is much smaller than that of flow 2.

The reason for the inconsistency between the analytical results and the simulations results is that, while the analysis is based on a fluid model, the simulation is conducted in a packet switched network. To ensure lossless service, buffer allocation has to be adjusted in a packet switched network due to this packet fragmentation.

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, only one flow can be served at any instant. Thus, when a best effort flow is transmitting a packet, the guaranteed performance flows can not release any buffer. The maximum time interval that no packet of the guaranteed performance flows arrives is $\frac{L}{\max\{\rho_l\}}$, where L is the maximum packet length. This is also the the maximum time interval that best effort flows can continue sending packets. During this interval, 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 outline, the corresponding buffer space is still occupied. With a 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 use A_i to denote the adjustment for f_i in a single output network processor, 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 network processor with multicast incoming flows, the general expression for the adjustment of f_{ji} is given as follows:

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

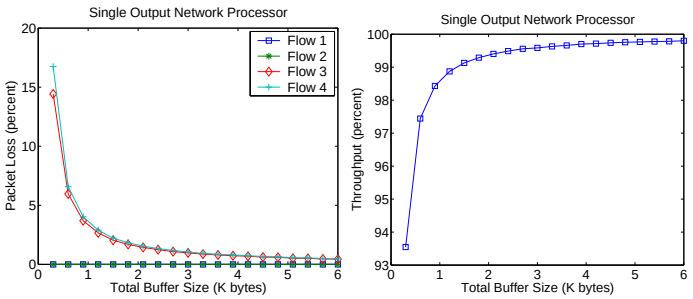
The explanation is to choose the largest value among those of all the labelled flows and use the scaled rate to replace the actual rate. As a result, the universally applicable formulas in Theorem 4 should be modified as follows to reflect the adjustment for packet fragmentation.

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

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

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. 2(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 network processor 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



(a) (b)

Fig. 3. Effect of total buffer size to packet loss and throughput. (a) Packet loss decreases as the total buffer size increases. (b) Throughput increases as the total buffer size increases.

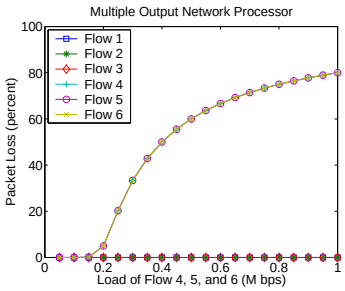


Fig. 4. By saving a multicast packet as a single copy in the share buffer, the buffer space requirement for ensuring lossless service is much smaller.

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 in Fig. 3(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 network processor is given in Fig. 3(b). As the increase of the total buffer size, the throughput of the network processor is steadily increasing, and finally approaching 100%.

B. Multiple Output Network Processor

In the following simulation, we verify Theorem 4. The simulation is set up based on the example in Fig. 1. The network processor has three output 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 (labelled 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, and 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. 4. 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.

VI. 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 in high speed and with low cost. In this paper, we have studied using buffer management to ensure lossless service for guaranteed performance flows in network processors. By adopting the discussed buffer management methods, they can efficiently provide guaranteed performance service in a high speed network.

Our analysis started with the simpler case where the considered network processor has only one output. Then we extended the results to the more general situation for multiple output network processors with multicast flows, and obtained a universally applicable buffer allocation solution for assuring lossless service. We then 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 presented 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 network processors well.

References

[1] 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.

[2] 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.

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

[4] 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.

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

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

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

[8] Intel, "Intel ixp2800 network processor," http://www.intel.com/design/network/products/npfamily/ixp2800.htm

[9] IBM, "The network processor: Enabling technology for highperformance networking," 1999.

[10] Motorola, "Motorola c-port corporation: C-5 digital communications processor," http://www.cportcom.com/solutions/docs/c5brief.pdf, 1999.

[11] D. Lin and R. Morris, "Dynamics ofrandom early detection," *ACM SIGCOMM 1997*, pp. 127-l37, Sophia Antipolis, France, Sep. 1997.

[12] 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.

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

[14] J. Kurose and K. Ross, "Computer networking: a top-down approach featuring the Internet," *Addison Wesley*, 3rd edition, May 2004.

[15] 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.