Performance Evaluation of Input and Output Queueing Techniques in ATM Switching Systems

Enrico Del Re, *Senior Member, IEEE,* and Romano Fantacci, *Senior Member, IEEE*

Abstract— **In this paper, alternatives to model a fast packet switching system are analyzed. A nonblocking switch fabric which runs at the same speed as the input/output links is considered. The performance of the considered approaches have been derived by theoretical analysis and computer simulations. Performance comparison between input queueing approaches with different selection policies are presented. Novel input and output queueing techniques are also proposed. In particular it is shown that, depending on the implementation, the novel input queueing approach studied in this paper achieves the same performance as the optimum (output) queueing alternative, without resorting to a faster packet switch fabric.**

I. Introduction

IN moving toward integrated services broadband communication network (BISDN), efforts are focusing on high-speed packet switching networks [1]–[3] that allow flexible services and on high capacity packet switches for interconnecting a large number of customers lines through a central hub. These requirements lead to a star topology where customers are connected to the hub by means of full-duplex optical fiber lines. Such architecture provides high-speed transmission rates, typically 1–200 Mb/s on each input–output line [3]. To support such transmission rates a high speed is required in the switching operation. An efficient approach to providing a high-speed switching operation seem to be a switching technique which routes packets from inputs to outputs through the use of hardware-based processing. This switching technique has been called fast packet switching (FPS). The basic idea of FPS is to provide simplified connection-oriented packet switching to attain an improvement in the throughput efficiency. In particular, no error or flow control on a link-by-link basis is performed. The use of high-quality optical fiber links makes possible this simplification.

In FPS systems, each information packet, typically called a cell, is labeled with a short header containing address information that is used by the switch fabric to allow the required routing from inputs to outputs at the switch. Congestion can happen if the switch is a blocking network, but also in the case of a nonblocking switch fabric, when two or more packets arrive simultaneously on different inputs requiring to be routed to the same output. One of these contending packets attains switching to the output. Queueing is required for the others to wait for a later route to the output. It is evident that this form of congestion is unavoidable in FPS systems and it often leads to an increased complexity in the switch architecture.

Different approaches for providing the queueing necessary for contending packets have recently appeared in the scientific literature [4]–[5]. Depending on the speed of the switch fabric and its architecture, alternatives to where the queueing is performed are possible. A performance comparison between input and output queueing is presented in [5]. In that paper it is shown that better performance is attained when the queueing is done on output. However, a faster switch fabric with respect to that used when queueing is done on inputs is required. In particular, if N is the number of input/output links, a switch fabric able to run N times faster than the input and output links is necessary. A switch fabric running at the same speed as the input/output links is instead sufficient for input queueing. Different selection policies, also including priority selection are investigated in [5] for input queueing. In particular, the performance comparison between output queueing and input queueing is carried out in term of mean waiting time. Moreover, it is shown that the output queues saturate as the throughput approaches 1, differently from the input queueing, where saturation makes the maximum possible throughput less than unity and dependent on N and on the service policy. For high value of N, the maximum possible throughput is 0.586 resulting when each input queue has always a packet to be routed on output links [5] (and when the traffic load is uniform over the output links).

An alternative to the FIFO discipline to serve the packets in the input queues has been proposed in [6]. With this service discipline, named window service discipline (WD), a packet can be removed form each input queue per time slot, but not necessarily the first packet in the queue. The WD permits a significant increase in the throughput of each output link. With the WD, those input links not attaining the transmission of the first packet in their queues contend with their second packets for access to any idle output link. This procedure is started over again for the first W packets (window width) of each queue. It is evident that, although the overall performance may be improved by using WD, more processing power may be required than with classical queueing on inputs.

Alternatives to input and output queueing in FPS systems are also proposed and analyzed in [4] where it is clearly shown that output queueing is the better solution. In particular, an efficient architecture which provides for output

Paper approved by E. G. Sable, the Editor for Communication Switching of the IEEE Communications Society. Manuscript received July 19, 1990; revised September 6, 1991. This work was supported by the National Research Council (C.N.R.) in the frame of the Telecommunications Project and also supported in part by MURST. This paper was presented in part at IEEE GLOBECOM'92, Orlando, FL, December 6–9, 1992.

The authors are with the Dipartimento di Ingegneria Elettronica, Università di Firenze, Via S. Marta 3, 50139 Firenze, Italy.

IEEE Log Number 9211136.

queueing without resorting to separate buffers for each output, namely completely shared buffering, is proposed. This technique achieves the same performance as the classical output queueing approach reducing the total amount of buffering in the switch, but at the expense of an increase in the size of the switch fabric.

This paper is concerned with novel forms of input and output queueing approaches. The switch fabric is assumed to be nonblocking and running at the same speed of each input/output link. Packets have fixed length and arrive on the N inputs in a time-slotted fashion. The results show that the novel queueing approaches achieve the same performance as classical output queueing without resorting to a faster switch fabric, while maintaining a hardware simplicity typical of input queueing.

II. Queues on Inputs

The performance analysis presented in this section is based on the well-known results for discrete-time queueing system [7], [8]. Let us assume that for each of the N input links no more than one packet may arrive in a time slot, and the arrivals in different time slots are statistically independent. The arrival processes on the N input links may be modeled as N independent Bernoulli processes with the probability of an arrival on one of the N input links equal to p. Each packet has an equal probability to be addressed to any of the possible N output links and successive packets require independent routing. The service statistics are assumed the same for packets arriving at an empty input queue and for packets arriving at a busy input queue. Packets are routed to the appropriate output link strictly in their order of arrival: this means that the order of service is FIFO within each input queue. When a packet arrives at the head of its input queue a request asking for routing to the destination output link is sent immediately to the switch controller. For each output link, a queue of routing requests, called destination queue, is formed by the switch controller. Each new request is placed at the end of the appropriate destination queue and the routing requests are served on the basis of two different selection policies, namely, the first-in first-out (FIFO) selection policy and the Random (RND) selection policy.

When a request is selected to be served, the switch controller makes the necessary connection between the input and the destination link. No more than one request can be satisfied on each time slot per output link. It follows that each destination queue may be modeled as a discrete Geom/D/1/N/N queueing system with customers served according to the FIFO or RND selection policies: Bernoulli input process (Geom), constant service time process (D), one server with maximum system memory size equal to N and customer population equal to N.

A. FIFO Selection Policy

Focusing our attention on a packet at the head of an input queue, i.e., the tagged packet, its service time is equal to the total time which must elapse until its routing request (the tagged routing request) leaves the appropriate destination queue (the tagged destination queue). Note that any routing request leaves its destination queue when the transmission of the associated packet on the appropriate output link is completed. Clearly, the maximum value of the service time is N slots corresponding to the event that all input queues have packet at their heads which require routing to the same output link.

From the above, it is possible to model each input queue as a discrete Geom/G/1 queueing system, with service time given by the total delay spent in the destination queue.

Focusing on a particular input queue, the imbedded Markov chains approach developed for the continuous $M/G/1$ model is applicable here also to derive the mean total delay per packet [8].

The probability generating function of the number of packets in the input queue, assuming equilibrium, is

$$Q(z) = \frac{Q_o A(z)(z-1)}{z - A(z)} \tag{1}$$

where Q_o is the probability of having an idle queue and $A(z)$ is the probability generating function of the number of arrivals during a service period of a customer.

In particular, it is possible to show that:

$$A(z) = (1 - p + pz)G(1 - p + pz) \tag{2}$$

where $G(z)$ is the probability generation function of the total time (in number of cells) spent by any routing request waiting to reach the head of the destination queue. In deriving $A(z)$ it should be considered that the service periods of successive packets arriving at the input queues are dependent. In particular, the service period of a packet is dependent on the destination of the packet just before it in the same input queue. Moreover, it should be necessary to distinguish between packets that arrive at busy and idle input queues. Nevertheless, with the aim of simplifying our analysis we have assumed that the service periods of successive packets are independent. In particular, under this assumption there is no difference between the service periods of packets arriving at idle or busy input queues. It is evident that we are considering an approximate analytical approach. The goodness of this approximation will be discussed later by comparing the analytical results thus obtained with those derived by computer simulations.

In deriving an expression for $G(z)$ it must be taken into account that in the considered case, the routing requests may arrive at the appropriate destination queue in batches of random size [8]. We assume that routing requests which arrive at the destination queue at same instant are served in a random order. However, the routing requests arriving in earlier instants are served first on the basis of the FIFO discipline. Therefore, the total time spent in the destination queue waiting for service by a routing request is due to the sum of two contributions, e.g., w_1 and w_2. The term w_1 takes into account the time necessary to serve all the routing requests arrived before and just waiting in the queue at the arrival instant. The second term w_2, is an additional delay due to the service of the routing requests which arrive at the same instant and randomly selected to be served first.

Focusing on the tagged destination queue, and assuming that k packets are already waiting for routing, the probability

that the tagged routing request arrives in a batch of size i is given by

$$P(i|k) = \frac{i}{(N-k)\alpha}\binom{N-k}{i}\alpha^i(1-\alpha)^{N-k-i}, \quad i = 1, 2, \cdots, N-k \tag{3}$$

where α is the probability of having a packet from one of the free input queues requesting routing to the tagged output link.

In deriving an expression for α we define:

- A_m as the overall number of routing requests arrived at all the destination queues at the mth time slot;
- F_m as the number of free input queues at the mth time slot.

Note that an input queue is free at the mth time slot if it is idle or if the packet at its head has been selected to be routed during the $(m-1)$th time slot. It is evident that an arrival at a destination queue must come only from a free input queue. Starting from the previous considerations, it is straightforward to verify that

$$P\{A_m = h\} = \binom{F_m}{h}(N\alpha)^h(1-N\alpha)^{F_m-h}. \tag{4}$$

Therefore, the mean number of arrivals $\overline{A}_m(F_m)$ conditioned on the occurrence that there are F_m free input queues is

$$\overline{A}_m(F_m) = F_m N\alpha\,. \tag{5}$$

Letting B_m be the number of routing requests in all the destination queues that are waiting for service we have

$$F_m = N - B_m\,. \tag{6}$$

In a steady-state condition we have

$$\overline{F} = N - \overline{B} \tag{7}$$

where $\overline{F}$ denotes the mean number of free input queues and $\overline{B}$ is the mean number of routing requests in all the destination queues.

By assuming equilibrium we also have

$$\overline{A} = [N - \overline{B}]N\alpha = Np\,. \tag{8a}$$

For the last expression of (8a) it must be noted that the mean number of routing requests which arrive at all the destination queues per time slot is equal to the mean number of packets routed from input queues to the output links, and is also equal to the mean number of packets which arrive at all the input queues.

It follows from (8a) that:

$$\alpha = \frac{p}{N-\overline{B}}\,. \tag{9}$$

Although it is mathematically pleasing to have closed-form expressions, for the case we are studying it is very difficult to derive $\overline{B}$ in a closed form [21]. Therefore, to simplify our analysis, the following approximate expression for α is used

$$\alpha = p/N\,. \tag{10}$$

In deriving (10) it has been assumed $\overline{B}$ is negligible with respect to N. By deriving $\overline{B}$ through simulation, it is possible to compare the values of α given by (9) and (10), respectively. The result is shown in Fig. 1. This figure points out that (10) is a good estimation of α. A slight underestimation only arises for high-load conditions. Nevertheless, it will be discussed later that the estimation accuracy of our analysis is not appreciably influenced by this approximation.

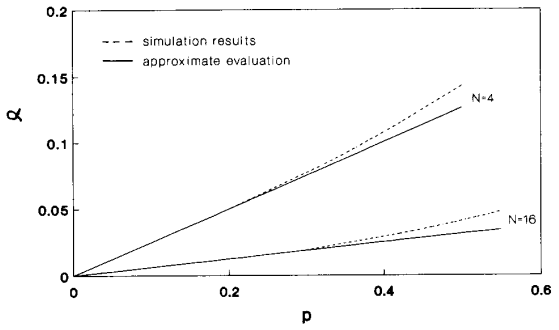

Fig. 1. The parameter α as a function of p.

The probability generating function of the total delay time spent in the tagged destination queue, conditioned on the occurrence that k requests are waiting for routing in the tagged destination queue and on the event that the tagged request arrives in a batch of size i is

$$G(z|i,k) = \sum_{0}^{i-1} {}_j \frac{z^{j+k}}{i} = \frac{1-z^i}{i(1-z)}z^k. \tag{11}$$

Therefore, $G(z|k)$ is given by

$$G(z|k) = \sum_{1}^{N-k} {}_i\, G(z|i,k)P(i|k) = \frac{1-(1-\alpha+\alpha z)^{N-k}}{(N-k)\alpha(1-z)}z^k. \tag{12}$$

The probability $P_R(k)$ of having k routing requests $0 \le k \le N-1$ in the destination queue can be obtained numerically by an application of the Markov chain balance equations. The final result is

$$P_R(1) = P_R(0)\frac{(1-a_{o,o}-a_{o,1})}{a_{1,o}} \tag{13}$$

$$P_R(k) = \frac{1-a_{k-1,1}}{a_{k,o}}P_R(k-1) - \sum_{i=2}^{k}\frac{a_{k-i,i}}{a_{k,o}}P_R(k-i), \quad 2 \le k \le N-1 \tag{14}$$

with $P_R(0)$ determined by the following condition:

$$\sum_{0}^{N-1} {}_k\, P_R(k) = 1 \tag{15}$$

and the terms $a_{i,j}$ given by

$$a_{i,j} = \binom{N-i}{j}\alpha^j(1-\alpha)^{N-i-j}. \tag{16}$$

Therefore, the probability generating function of the total time spent in the tagged destination queue $G(z)$ can be derived as

$$G(z) = \sum_{0}^{N-1} {}_k\, G(z|k) P_R(k) = \sum_{0}^{N-1} {}_k \frac{\left[1-(1-\alpha+\alpha z)^{N-k}\right]}{(N-k)\alpha(1-z)} P_R(k). \quad (17)$$

By means of standard discrete queueing systems theory results it is straightforward to derive (2) from (17) [8].

The mean delay spent by a packet in an input queue is

$$T = \frac{Q'(1)}{p} = \frac{A'(1)}{p} + \frac{A''(1)}{2p[1-A'(1)]} \quad (18)$$

where $A'(1)$ and $A''(1)$ are obtained by differentiating once and twice, respectively, $A(z)$ with respect to z and taking the limit as z approaches 1. The final result is given in (19) below. Fig. 2 shows T (normalized with respect to the cell duration time), as a function of p for different values of N. In the same figure the results derived by computer simulations are reported to validate the analytical results. From this figure, it can be seen that the analytical results agree with the simulation results for small and medium values of p.

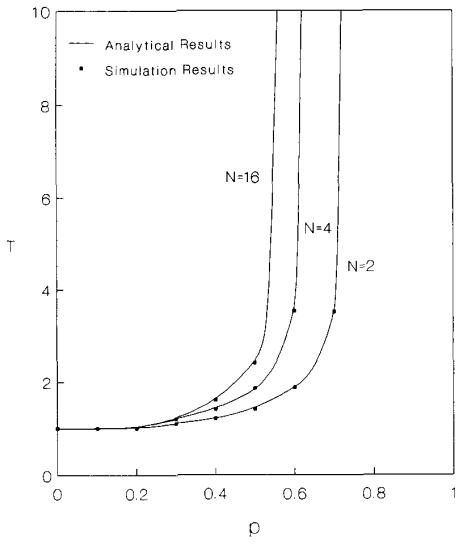

Fig. 2. The mean normalized total delay for input queueing with FIFO selection policy for different values of N.

B. RND Selection Policy

An alternative to the FIFO selection policy could be the RND selection policy here considered. In this case, when k routing requests are waiting for service in a destination queue, one of the k is chosen at random. Each routing request can be selected with equal probability $1/k$. The unselected routing requests wait for the next time slot when a new random selection takes place. The use of the RND selection policy in FPS systems with input queueing has been considered in [5]. However, our analysis differs from that presented in [5] mainly in that a finite value of N is considered.

In this case, the mean packet delay can be easily derived as

$$T = E[s] + \frac{p\{E[s^2] - E[s]\}}{2\{1-pE[s]\}} \quad (20)$$

where s denotes the random service time with mean value $E[s]$ and mean squared $E[s^2]$. The probability distribution of s is obtained from the results given in [5] as

$$P\{s=m\} = \sum_{1}^{N} {}_k\, P_{m,k} \sum_{0}^{k-1} {}_n \binom{N-1-n}{k-n-1} \cdot \alpha^{k-n-1}(1-\alpha)^{N-k} P_R(n) \quad (21)$$

with parameter $P_{m,k}$ denoting the probability that the remaining delay is m time slots until the considered packet leaves the input queue, conditioned on k packets waiting in the destination queue. The parameter P_m, k is defined in (22) below. Therefore, from (21) and (22) it is possible to derive $E[s]$ and $E[s^2]$ recursively and to obtain T from (20) in terms of all known parameters.

Fig. 3 shows T as a function of p for different values of N achieved under input queueing with the RND selection policy.

$$T = 1 + \sum_{0}^{N-1} {}_k \frac{(N+k-1)\alpha}{2} P_R(k) + \frac{p \sum_{0}^{N-1} {}_k \left[k(k-1) + k(N-k-1)\alpha + (N-k-1)(N-k-2)\alpha^2/3\right] P_R(k)}{2\left\{1 - p\left[1 + \sum_{0}^{N-1} {}_k \frac{(N+k-1)\alpha}{2} P_R(k)\right]\right\}}. \quad (19)$$

$$P_{m,k} = \begin{cases} 1/k & \text{if } m = 1, \text{ all } k; \\ 0 & \text{if } m = 1 \text{ and } k = 1; \\ (1-1/k) \sum_{0}^{N-k-1} {}_j\, P_{m-1,k-1+j} \binom{N-k-1}{j} \alpha^j (1-\alpha)^{N-k-1-j}. \end{cases} \quad (22)$$

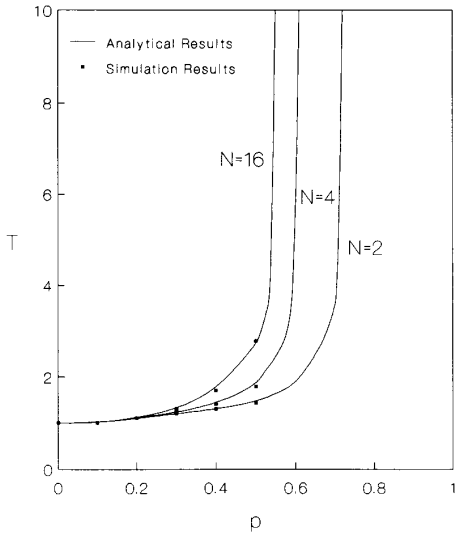

Fig. 3. The mean normalized total delay for input queueing with RND selection policy for different values of N.

The same considerations raised for Fig. 2 can be applied here also.

III. A Multiple Queueing Approach

In the analysis presented above, it has been assumed that none of the packets waiting for routing in the source queue can leave the queue until the packet at the head has completed service. An efficient alternative to the previously considered queueing approach is the multiple queueing approach, which always permits routing for packets whose destination queues are empty.

In the multiple queueing approach, each input queue is split into N separate queues, one for each possible output link.

When a packet arrives on an input, it is routed to a queue associated with that packet destination outlink. Note that the buffering of packets occurs after the routing–switching so the multiple queueing approach is basically an output queueing architecture.

Packets at the head of input queues for the same output link contend for routing. Both FIFO and RND selection policies can be used to select the packet to be routed.

The service discipline in each input queue is FIFO. Note that despite the switching system is now essentially an output queued fabric, the analytical procedure developed in Section II for the classical input queueing approach can be used here again in performing our analysis. However, in this case a closed-form expression for α can be derived.

The multiple queueing approach is basically implemented also by the bus matrix switch (BMS) proposed in[17]. In the BMS the queueing of the packets is performed at the cross points of the switch. The service discipline within each cross point buffer (XPM) is FIFO. Secondary packet distributors (SPD), one for each outgoing link, are used to solve the contention among packets waiting for routing to the same output. The bus matrix switch implements a selection policy similar to the roll-call procedure [13], [14]. Each SPD scans every XPM for the appropriate output and removes packets at the heads of the corresponding queues. Differently from [17], routing requests queues (named destination queues) are formed in the proposed switch by each output controller. The FIFO and random selection policies are considered as alternatives to select the routing requests to be satisfied per time slot. Moreover, it is important to point out that in [17] only simulation results are presented.

In deriving α we focus on a particular output link (the tagged output link). The arrival process at each queue for the tagged output link is Bernoulli with probability p/N that a packet arrives in any time slot. Moreover, it is assumed that no more than one packet can be transmitted on the output links per time slot.

Letting Q_m denote the number of routing requests in the tagged destination queue, we can write

$$F_m = N - Q_m . \tag{23}$$

Therefore, in a steady-state condition

$$\overline{F} = N - \overline{Q} \tag{24}$$

where the mean number of routing requests in the tagged destination queue can be derived as

$$\overline{Q} = \sum_{0}^{N-1}{}_n \, nP_R(n) . \tag{25}$$

It is straightforward to obtain in this case an expression similar to (8a) that now reads as

$$\left(N - \overline{Q}\right) N\alpha = Np . \tag{8a}$$

Hence,

$$\alpha = \frac{p}{N - \sum_{0}^{N-1}{}_n \, nP_R(n)} . \tag{26}$$

$$T = \frac{NA'(1)}{p} + \frac{NA''(1)}{2p[1-A'(1)]} = 1 + \sum_{0}^{N-1}{}_{k} \frac{(N+k-1)\alpha}{2} P_R(k) + \frac{p \sum_{0}^{N-1}{}_{k} \left[k(k-1) + k(N-k-1)\alpha + (N-k-1)(N-k-2)\alpha^2/3\right] P_R K}{2\left\{N - p\left[1 + \sum_{0}^{N-1}{}_{k} \frac{(N+k-1)\alpha}{2} P_R(k)\right]\right\}}. \quad (27)$$

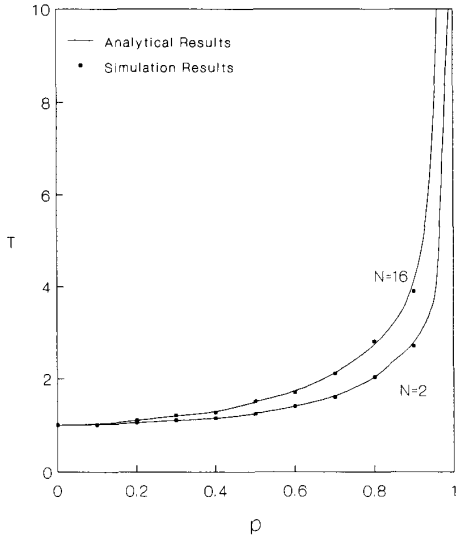

Fig. 4. The mean normalized total delay for the multiple queueing approach with FIFO selection policy for different values of N.

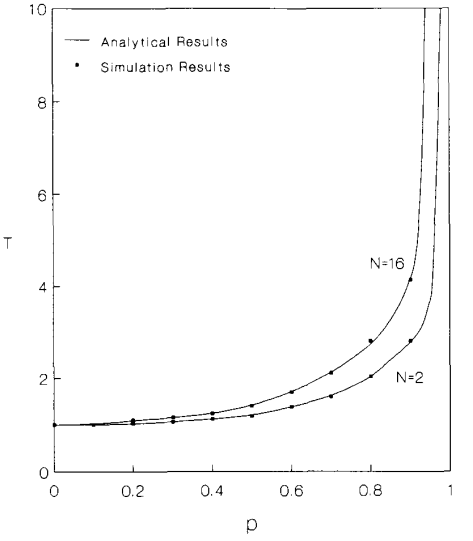

Fig. 5. The mean normalized total delay for the multiple queueing approach with RND selection policy for different values of N.

Equation (26) defines a nonlinear equation in α. By solving this equation numerically it is possible to determine α.

Starting from the previous considerations, it is straightforward to derive the mean delay per packet T as in (27) above for the FIFO selection policy and as

$$T = E[s] + \frac{p\{E[s^2] - E[s]\}}{2\{N - pE[s]\}} \quad (28)$$

for the RND selection policy. The terms $E[s]$ and $E[s^2]$ in (28) can be obtained as outlined in Section II-B.

Figs. 4 and 5 show T as a function of p for different values of N. These figures show that the maximum attainable throughput now approaches 1 as p approaches 1 for both the FIFO and RND selection policies. These results are consistent with the classical results for output queueing [5].

IV. Implementation Considerations

The switch fabric with input queueing studied in Section II is internally fully accessible meaning that every input has an available path to every output. A switch control is necessary to select among all packets waiting at the heads of different input queues to be routed to the same output link. Different implementation alternatives are described in [1], [10]. One solution is to make use of a centralized control. This control handles routing requests from all busy input queues. Suitable technology and efficient techniques, such as pipelining, can be used to build very high-speed switch fabrics. The drawback of this approach is that the control overhead strongly increases with the size of the switch. The way to reduce the control overhead is to use a switch fabric architecture with distributed control functions. This approach is also discussed in [10]. In this case each output link has its individual control, which allows only one packet from the input queues to be routed to an output link at the same time by a suitable selection policy.

This paper deals with a two-stage approach to packet switching. In stage one, the packet header (i.e., the routing request) is transmitted, whereas in stage two the packet itself is transmitted. The proposed fast packet switching approach is particularly appropriate in the case of an ATM packet format (CCITT Recommendation I121). However, the fast switching approach under consideration performs well also for different packet formats. It is evident that the switching processing time increases with the header dimension.

The architecture proposed for the classical input queueing approach is shown in Fig. 6. Each input queue is a simple FIFO queue. The headers (i.e., the routing requests) of packets at the heads of such FIFO queues are broadcast over buses (one per input port) to all the output controllers (arbiters).

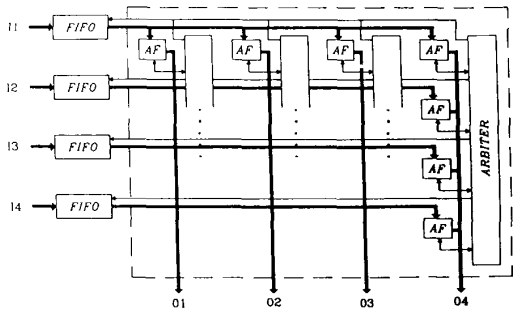

Fig. 6. Switch fabric architecture for the input queueing approach ($N = 4$).

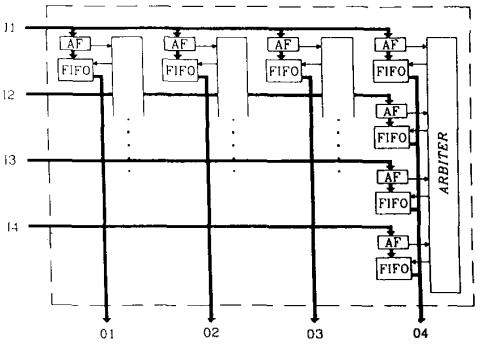

Fig. 7. Switch fabric architecture for the multiple queueing approach ($N = 4$).

By means of routing request (address) filters (AF) a routing request arrives only at the input of the desired arbiter, as a routing request can only pass through the filter whose address matches the routing request destination address. The arbiter handles the received request according to a suitable selection policy (FIFO or RND).

The architecture for the switching fabric with multiple queueing (Sect. III) is shown in Fig. 7. Whenever, a new packet arrives at an input port its address is immediately broadcast over the input bus. Collisions on an input bus are impossible because at the most one packet may arrive per slot at each input port. By means of address filters each output controller receives only the routing request for its outport and packets wait for service in N separate FIFO buffers. An improvement of the architecture shown in Fig. 7 may be the use of input shared buffers, which permits a reduction of the buffering requirements. Of course, this approach leads to an increasing of the control overhead. This solution implements basically an input queueing because the buffering of packets occurs before the routing/switching. The multiple queueing with input shared buffers will be studied in details in Section VI.

The proposed architecture for the input queueing switch fabric can be also adapted to permit use in optical networks [11], [12].

V. Comparison with the Classical Queueing on Outputs Approach

An alternative to the approaches stated above could be a FPS system in which queueing is done on each output link. The incoming packets are routed to the appropriate output queue by means of their headers independently of the other $N-1$ input links. As described in [1], [2], [5] this operation requires a switch fabric running N times faster than in the other cases or having a degree of internal connectivity N times higher (e.g., Knockout) [16].

Focusing on a particular output queue, it can be noted that the number of packet arrivals at a given instant has a binomial distribution. Let p/N be the probability that a packet arrives at the queue coming from any input link, the probability generating function of arrivals is

$$B(z) = [1 - (p/N)(1 - z)]^N. \tag{29}$$

Assuming that packets are removed from the output queue on the basis of the FIFO discipline, the total time spent by any packet in the FPS system is the sum of the packet transmission time (one cell) and the waiting time given by two independent contributions x_1, x_2. The first contribution x_1 is due to the number of packets waiting for service in the queue at the arrival instant. The second contribution x_2 is due to the random selection of the packet from the batch of arrivals.

The probability generating function of packets waiting in the queue can be derived as

$$P(z) = \frac{P_o(z-1)}{z - B(z)} \tag{30}$$

where P_o denotes the probability of having an empty queue. Recalling that the packet transmission time is one slot, the mean value of the first contribution is

$$\overline{x}_1 = P'(1) = \frac{(N-1)p^2}{2N(1-p)}. \tag{31}$$

Let the probability that a packet arrives in batches of size i be given by

$$P_r(i) = \binom{N}{i} \frac{i(p/N)^i(1-p/N)^{N-i}}{p}. \tag{32}$$

Hence, the probability generation function of the additional delay due to arrival in batches of random size is

$$X_2(z) = \frac{1 - B(z)}{(1-z)p}. \tag{33}$$

The mean value $\overline{x}_2$ of the second contribution is obtained by differentiating $X_2(z)$ with respect to z and taking the limit as z approaches 1. The final result for T is:

$$T = 1 + \overline{x}_1 + \overline{x}_2 = 1 + \frac{(N-1)p}{2N(1-p)}. \tag{34}$$

The maximum throughput possible by using queueing on outputs has been derived in [5] and it approaches 1 as p approaches 1. Fig. 8 shows T as a function of p for the queueing on outputs. In the same figure the parameter T attained by using the queueing on inputs and the proposed multiple queueing approach is reported for comparison purposes. It is evident in this figure that the proposed multiple

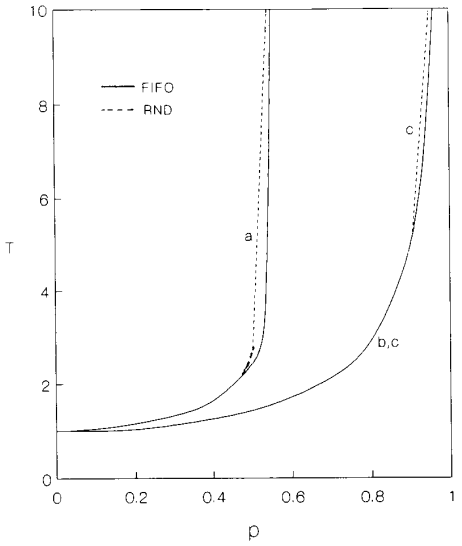

Fig. 8. Mean normalized total delay as function of p for $N = 16$. (a) queueing on inputs; (b) queueing on outputs (FIFO); (c) proposed multiple queueing.

queueing approach achieves the same performance as the classical output queueing approach without resorting to a factor switching fabric operating at a faster speed or having a higher degree of internal connectivity.

VI. A Multiple Queueing Approach with Input Shared Buffers

As stated in the Section IV, in the switch fabric with multiple queueing whenever a packet arrives on an input it is immediately routed to the appropriate destination queue. Therefore if buffering is required, it occurs after the routing operation so that this approach is basically an output queueing approach (Fig. 7).

Nevertheless, the switch architecture of Fig. 7 can be readily modified to perform queueing on inputs by using an implementation architecture in which all the arriving packets at each input are queued in a shared buffer.

Also in this case the packet switching is assumed to be performed in two stages. In stage one, the routing request, associated with each packet, is analyzed while in stage two the packet itself, is removed from the input shared buffer and transmitted onto the output link whenever the associated routing request reaches the head of the appropriate destination queue (FIFO) or it is randomly selected to be served (RND). Note that the routing request may arrive at the destination queues in batches of random size. By considering the FIFO selection policy, we have to queue in the worst case N routing requests simultaneously within a time slot. However, the problem of a speed up typical of the output queueing implementation architecture, does not arise here because the routing request are formed by few bits. It is important to point out that this problem can be avoided by using the RND selection policy.

Architectures for input queueing switching systems based on shared buffers have been recently proposed in [18], [19]. In the switch fabric proposed in [18] a schedule algorithm is used to solve contentions among packets for the same output. This algorithm has two phases: request phase and arbitration phase. In the request phase, whenever a packet arrives at an input a routing request is sent to the appropriate contention control module. In the arbitration phase each contention control module updates the scheduling table for the associated output link by assigning the transmission time slot to each requesting packet. For each input queue a sending table is kept and updated whenever the scheduling time slot for the requesting packet arrives. If the sending table has already been reserved at the assigned time slot the packet must be returned to the request phase in the next time slot. This has been called "buffer blocking" in [18] and it is a primary factor in the achievement of an efficiency of about 0.74 under uniform traffic load conditions. A better efficiency (of about 0.9) can be attained if several input ports share a common queue. Unfortunately, in this case faster memories are required.

The switch fabric with shared buffers proposed in this paper differ from that proposed in [18] because it is based on different selection policies (i.e., scheduling algorithm) which permit to achieve a throughput approaching 1 as the probability of having an arrival per slot approaches 1 (the same as the output queueing). In our case the "buffer blocking" is avoided by means of a shared buffer architecture which permits that the input and outputs can operate independently and more than one packet (with different output destinations and therefore, different memory addresses) can be read out per time slot. A similar shared buffer architecture is proposed in [20] for the output queuing approach. Such an architecture can be modified accordingly.

In [19] a switch fabric similar to that described in [18] is presented. The "buffer blocking" problem is reduced in this case by performing both input and output queueing. This is the main point of difference of this switching system over the novel multiple queueing switch proposed in this paper which is only based on the input queueing approach. In addition to this, better throughput performance (the same as the output queueing) can be attained by means of the proposed switch.

The multiple queueing fabric with input shared buffer differs from that discussed in Section III only for the different way to allow buffering of the incoming packets. In particular this means that the same throughput and, depending on the shared buffer implementation [20], delay performance as the separate buffer case are achieved by the novel multiple queue fabric presented in this section. However, to highlight the advantages of using input shared buffers over classical separate buffering approach, it is important to note that in any practical implementation the size of the buffers (shared or separate) is finite and therefore, packets may be loss.

Unfortunately, in the case of multiple queueing approach with input shared buffers the analytical evaluation of the packet loss probability leads to a too complex queueing problem to be solved in a closed form [21]. In what follows, an approximate evaluation of the packet loss probability is derived and validated by comparing analytical and simulation results.

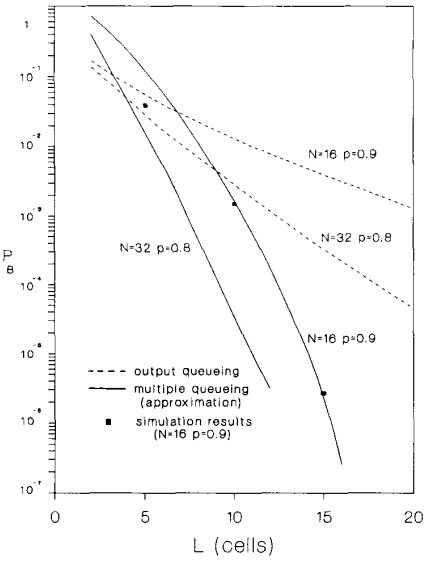

Fig. 9. Packet loss probability comparison.

We start our analysis by considering input shared buffers of infinite size. For such buffers, the number of packets stored at a particular input having the same output destination and therefore, forming the queue for that output is not independent of the number of packets stored in the same shared buffer forming the queues for the other $N-1$ outputs [21]. Nevertheless, to simplify our analysis we carry out an approximate approach by assuming the overall number of packets n_t queued in a shared buffer as the sum of N independent identically distributed (i.i.d.) random variables n_i, n_i denoting the number of packets stored in the queue for output i $(1 \leq i \leq N)$. Note that our approach is consistent with that outlined in [4] in deriving the performance of a switch fabric with a completely shared (output) buffering. It follows that the probability generating function for n_t is given by

$$C_t(z) = \sum_0^\infty{}_k\, z^k c_t(k) = \left[\sum_0^\infty{}_k\, z^k c(k)\right]^N = C^N(z) \quad (35)$$

with $C(z)$ the probability generation function of the number of packets in one of the N (logically separated) input queues, $c_t(k)$ the probability of having k packets stored in the shared input buffer and $c(k)$ the probability of having k packets in an input queue defined as:

$$c(1) = c(0)\frac{1-a_0-a_1}{a_1} \quad (36)$$

$$c(k) = \frac{1-a_{k-1}}{a_k}c(k-1) - \sum_2^k{}_i \frac{a_{k-i}}{a_k}c(k-i) \quad \text{for } k>1\,, \quad (37)$$

with

$$a_i = \sum_1^\infty{}_m \binom{m}{i}(p/N)^i(1-p/N)^{m-i}p(m) \quad (38)$$

where $p(m)$ denotes the probability for the packet at the head of the queue to have a service time equal to m (slots). These probabilities can be derived numerically from (2) and (10).

With a finite buffer size, it is evident that the random variables n_i cannot be considered as independent. We refer to (35) in that follows however, since it is a good approximation for low packet loss probabilities which are typically the only values of interest. The unknown term $c(0)$ in (36), (37) is defined to verify the following equation:

$$\sum_0^\infty{}_k\, c(k) = 1\,. \quad (39)$$

Therefore, it is possible to derive numerically the probability of having x packets stored in the shared input buffer (i.e., $c_t(x)$) [14].

In our approach, we approximate the packet loss probability P_B for a shared buffer of size L (in cells) as the probability P_L of having stored in a shared buffer of infinite size a number of packets greater than L, i.e.,

$$P_L = \sum_{i=L+1}^\infty c_t(i)\,. \quad (40)$$

Fig. 9 shows P_L in comparison with P_B achieved by using the classical output queueing approach as a function of L. The parameter P_B in the case of a classical output queueing can be defined as:

$$P_B = 1 - \frac{1-a_o c(0)}{p}\,. \quad (41)$$

This figure clearly shows that the proposed queueing approach with shared buffers outperforms the classical output queueing approach. A good agreement between our approximation and simulation results is also evident in the figure, in particular, for low values of the packet loss probability, (tail of the distribution) which, typically, are the values of interest.

VII. Conclusion

The performance of classical and novel approaches for queueing on inputs (or outputs) in a FPS system have been derived. An $N \times N$ nonblocking switch fabric running at the same speed as the input/output links has been assumed. The attained results clearly show that the multiple queueing approach achieves the same performance as the classical output queueing approach without having to resort to a speed up in the switching operations.

An important result is that the novel multiple queueing approach with shared input buffers achieve the same throughput/delay performance as the classical output queueing jointly with a reduction in the buffer size requirement, without using a switch fabric which runs N times faster as the input and output links.

It is important to recall that the increasing in the switching fabric speed operations can be avoided by introducing internal parallelism as in the Knockout switch [16]. However, the implementation complexity of the Knockout switch seems to be higher with respect to the proposed multiple queueing approach. This is principally due to the use of the Knockout concentrator/shifter at each output. The basic function of the Knockout concentrator is to select U packets out of N possible contenders by mean of an algorithm analogous to a knockout tournament. For a concentrator with N inputs and U outputs, there are U rounds of competition. The basic building block of the concentrator is a 2×2 switching element which randomly selects one input packet as the winner and, of course, the other as loser. The losers try to win the next rounds to became winners themself. After the last round the losers are lost. Delay lines must be included in the concentrator to keep the competition synchronous. The shifter permits to store sets of at the most U winners in U separate buffers according to the order of their arrival, therefore implementing the FIFO selection policy. The Knockout concentrator permits to reduce the number of separate buffers keeping low the packet loss probability (packets can be loss in the case of an infinite buffer size if they arrive in batches with more than U elements).

The Knockout switch philosophy permits indeed to save memory but it could be evident that it gives rise to an increase in the implementation complexity of the switch.

We would like to point out, however, that the switch fabrics with multiple queueing studied in this paper, make use of a different switching approach. Differently from the Knockout switch, the packet switching is performed in two stages. In stage one the routing requests associated with the packets at the heads of the input queues are analyzed, while in stage two the packets can be transmitted into the output links. The contention among the routing requests for the same output are handled by the appropriate output controller, which selects the winner slot by slot according to the FIFO or RND selection policy. It is evident that the connection request may arrive in batches of random size. In the FIFO worst case we may have to store N routing requests simultaneously within a time slot. However, the problem of a speed up does not arise here because the connection requests are formed by few bits (those strictly necessary to identify without ambiguity the output destination and the input queue). Note that this problem doesn't arise by using the RND selection policy. In both cases (FIFO or RND) the use of the knockout concentrator and of the shifter is avoided and therefore, a notable reduction in the implementation complexity is achieved.

Even if, depending on where the queueing is performed (input or output) more control is required, the multiple queueing approach seems to have some attractive features with respect to previously proposed switching fabric architectures and can be based on both electronic and optical implementations.

Acknowledgment

The authors would like to thank Dr. E. G. Sable and the reviewers for their careful reading of the manuscript and for their very helpful comments and suggestions.

References

[1] H. Armadi and W. F. Denzel, "A survey of modern high performance switching techniques," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 1091–1103, Sept. 1989.
[2] F. A. Tobagi, "Fast packet switch architectures for broad-band integrated services digital networks," *IEEE Proc.*, vol. 78, pp. 133–167, Jan. 1990.
[3] J. Y. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1264–1273, Oct. 1987.
[4] M. G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1587–1597, Dec. 1988.
[5] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. COM-35, pp. 1347–1356, Dec. 1987.
[6] M. K. Menmet-Ali, J. F. Hayes, and A. K. Elhakeem, "Traffic analysis of a local area network with a star topology," *IEEE Trans. Commun.*, vol. 36, pp. 703–712, June 1988.
[7] T. Meisling, "Discrete-time queueing theory," *Oper. Res.*, vol. 6, pp. 99–105, Jan.–Feb. 1958.
[8] H. Kobayashi and A. G. Konheim, "Queueing models for computer communications system analysis," *IEEE Trans. Commun.*, vol. COM-25, pp. 2–28, Jan. 1977.
[9] P. J. Burke, "Delays in single-server queues with batch input," *Oper. Res.*, vol. 23, pp. 830–833, July–Aug. 1975.
[10] F. A. Tobagi, "Fast packet switch architectures for broadband integrated service digital networks," *IEEE Proc.*, vol. 78, pp. 133–167, Jan. 1990.
[11] K. Y. Eng, "A photonic knockout switch for high-speed packet networks," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1107–1116, Aug. 1988.
[12] A. S. Acampora and M. J. Karol, "An overview of lightwave packet networks," *IEEE Network*, vol. 3, pp. 23–41, Jan. 1989.
[13] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis.* Reading, MA: Addison Wesley, 1987.
[14] J. F. Hayes, *Modeling and Analysis of Computer Communication Networks.* New York: Plenum, 1984.
[15] L. Kleinrock, *Queueing Systems, Vol. 1: Theory.* New York: Wiley, 1975.
[16] Y. Y. Yeh, M. C. Hluchyj, and A. S. Acampora, "The knockout switch: A simple, modular architecture for high-performance packet switching," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1274–1283, Oct. 1987.
[17] S. Nojima, E. Tsutsui, H. Fukuda, and M. Hascimoto, "Integrated services packet network using bus matrix switch," *IEEE J. Select. Areas Commun.*, vol. 10, pp. 1284–1292, Oct. 1992.
[18] H. Obara, "Optimum architecture for input queueing ATM switches," *Electron. Lett.*, vol. 27, pp. 555–557, Mar. 28, 1992.
[19] H. Obara, S. Okamoto, and Y. Hamazumi, "Input and output queueing ATM switch architecture with spatial and temporal slot reservation control," *Electron. Lett.*, vol. 28, pp. 22–24, Jan. 2, 1992.
[20] T. Kazaki, N. Endo, Y. Sakurai, O. Mutsubara, M. Mizukami, and K. I. Asamo, "32×32 shared buffer type ATM switch VLSI's for B-ISDN's," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1239–1247, Oct. 1991.
[21] A. E. Eckberg and T-C. Hou, "Effect of output buffer sharing requirements in an ATDM packet switch," in *IEEE INFOCOM'88*, pp. 459–466.

Enrico Del Re (M'78–SM'84) was born in Florence, Italy, in 1947. He received the Dr. Ing. degree in electronics engineering from the University of Pisa, Pisa, Italy, in 1971.

Until 1975 he was engaged in public administration and private firms, involved in the analysis and design of the telecommunication and air traffic control equipment and space systems. Since 1975 he has been with the Department of Electronics Engineering of the University of Florence, Florence, first as a Research Assistant, then as an Associate Professor, and presently as a Full Professor. During the academic year 1987–1988 he was on leave from the University of Florence for a nine-month period of research at the European Space Research and Technology Centre of the European Space Agency, The Netherlands. His main research interest are digital signal processing, digital transmission techniques, and communication networks, on which he has published more than 100 papers, in international journals and conferences. He is the Co-editor of the book *Satellite Integrated Communication Networks* (North-Holland, 1988) and one of the authors of the book *Data Compression and Error Control Techniques with Applications* (Academic, 1985). Presently, he is the Chairman of the European Project Cost 227 "Integrated Space/Terrestrial Mobile Networks."

Dr. Del Re is a member of AEI and European Association for Signal Processing (EURASIP).

Romano Fantacci (S'84–M'88–SM'91) received the Dr. Ing. degree in electronics engineering from the University of Florence, Florence, Italy, in 1982, and the Ph.D. in electronics engineering in 1987.

In 1982 he joined the Department of Electronics Engineering of the University of Florence, Florence, Italy, where currently he is Associate Professor of Telecommunication Networks. His research interests involve digital communications, queueing systems, multiple access systems, coding theory, and digital signal processing.