QUEUEING ANALYSIS OF BUFFERED SWITCHING NETWORKS

JONATHAN S. TURNER FELLOW, IEEE

Abstract — **This paper provides a method for analyzing the queueing behavior of switching networks constructed from switches that employ shared buffering or parallel bypass input buffering. It extends the queueing models first introduced by Jenq and later generalized by Szymanski and Shaikh to handle these classes of networks. Our analysis explicitly models the state of an entire switch and infers information about the distribution of packets associated with particular inputs or outputs when needed. Earlier analyses of networks constructed from switches using input buffering attempt to infer the state of a switch from the states of individual buffers and cannot be directly applied to the networks of interest here.**

Index Terms — **packet switching networks, ATM networks, queueing analysis**

I. INTRODUCTION

In a widely cited paper [4], Jenq describes a method for analyzing the queueing behavior of binary banyan networks with a single buffer at each switch input. The method, while not yielding closed form solutions, does permit the efficient computation of the delay and throughput characteristics of a switch. A key element of the analysis is the inference of the state of a single switch from the state of its two buffers, based on the assumption that the states of the two buffers are independent. This independence assumption is not valid but does not yield gross inaccuracies in the systems that Jenq studied.

Recently, Szymanski and Shaikh [6] have extended Jenq's method to switching systems constructed from switches with an arbitrary number of inputs and an arbitrary number of buffer slots. They have also applied it to systems with different buffering techniques. While these extensions are useful, it turns out that for many specific choices of system parameters, the independence assumption mentioned above leads to significant inaccuracies.

We extend the previous work to cover switching systems in which the buffer slots in a switch are shared among all the inputs and outputs, rather than being dedicated to either particular inputs or particular outputs. Such systems require an analysis which explicitly models the state of the entire switch rather than the states of individual input or output queues. We can also apply our method to systems using parallel bypass input buffering, a class of systems that cannot be analyzed directly using the previous methods. Our technique can also be applied to the systems studied previously and for some system configurations yields significantly more accurate results.

In section 2, we review the previous results for switching systems with input buffering, in order to motivate the key issues involved in their analysis. In section 3, we show how to analyze a switching system with shared buffering and present a variety of performance curves characterizing such systems. In section 4, we show how our methods can be extended to switching systems with input buffering, including systems supporting bypass queueing. Finally, in section 5, we provide numerical comparisons of the different buffering techniques, describe our computational experience and suggest some possible extensions to our work.

[0]This work was supported by the National Science Foundation (grant DCI 8600947), Bellcore, BNR, DEC, Italtel SIT, NEC, NTT and SynOptics.

Figure 1: Recursive Definition of a Delta Network

II. ANALYSIS OF NETWORKS WITH INPUT BUFFERING

Figure 1 shows the recursive construction of a delta network $D_{n,d}$ with n inputs and outputs, constructed from d-port switches. Such networks provide a single path between any inputs and outputs, and have $\log_d n$ stages of switching. (We use the term *network* here to describe the system as a whole and *switch* to describe the components from which the network is constructed.) The delta network is topologically equivalent to such networks as the banyan and omega networks. The results we describe here are equally applicable to any of these networks. Delta networks are often constructed from switches that contain buffering for a small number of packets, with flow control between successive switches to ensure that the buffers do not overflow. Figure 2 shows the structure of a typical switch in which each switch input has a buffer with a capacity of β packets.

Typically these systems are operated in a time-slotted fashion, with fixed length packets progressing from stage to stage in a synchronous fashion. Consequently, we can think of the system as operating in two phases. In the first phase, flow control information passes through the network from right to left. In the second phase, packets flow from left to right, in accordance with the flow control information. A switch input will allow its predecessor to send it a packet if it has an empty buffer slot currently or if one of the packets in its buffer will leave during the

Figure 2: Switch with Fifo Input Buffer

second phase of the current cycle. This is called *global flow control*, since the flow control decision at a switch potentially depends on all of its successors in the network. *Local flow control* is also possible; in this form, a switch input allows its predecessor to send a packet only if its buffer has an empty slot. While local flow control doesn't make as effective use of a switch's buffers, it is more straightforward to implement, particularly in high speed systems where the propagation time required for global flow control can lead to unacceptable overheads. Note also, that several packets in a switch may contend for the same output, but only one will be allowed to proceed during a given cycle. We assume (as is usual) that in such a case, one of the contending packets is selected at random.

One way to analyze the queueing behavior of a buffered delta network is to explicitly model the state of a single input buffer by a discrete time birth-death process and then model the state of an entire switch by assuming that the states of its various input buffers are independent. A Bernoulli arrival process is assumed and packets are independently assigned random destination addresses upon entry to the system. This analytical technique is described in [6]. We briefly review it here for completeness.

Let $\pi_i(j)$ be the steady state probability that an input buffer in stage i of the network (stages are numbered from left to right starting with 1) contains exactly j packets, where $0 \leq j \leq \beta$. Let a_i be the probability that a packet is available to enter a stage i buffer and let q_i be the probability that the first packet (assuming there is a packet) in a stage i buffer can leave during a given cycle. With these definitions, the transition probabilities for the stage i buffer are as shown below.

(Here, $\overline{a}_i = 1 - a_i$ and $\overline{q}_i = 1 - q_i$; we use the overline throughout to indicate the "complement" of the given probability.) The reasoning is straightforward. If the queue contains j packets where $0 < j < \beta$, then the probability that during the next cycle the queue contains $j+1$ packets is just the probability that a new packet is available to enter the queue and the packet at the head of the queue does not leave; this is $a_i\overline{q}_i$, assuming that arrivals and departures are independent of one another. Similarly, the probability that during the next cycle the queue contains $j-1$ packets is just the probability that no new packet is available to enter the queue and the packet at the head of the queue does leave; that is, $\overline{a}_i q_i$.

If we knew a_i and q_i then, we could easily compute the state probabilities $\pi_i(j)$. The trouble of course is that a_i and q_i depend on the state probabilities of the buffers in the neighboring switches. This leads to an iterative computational method in which we assign arbitrary initial values to the state probabilities, then compute a_i and q_i for all i, use these values together with the balance equations for the Markov chain to compute new state probabilities, and so forth.

We calculate a_i using the following equation

$$a_i = 1 - (1 - \overline{\pi}_{i-1}(0)/d)^d$$

The reasoning is that a packet is available to enter a particular input buffer of a stage i switch if at least one of the d buffers in the predecessor is non-empty and has a first packet for the particular stage i switch of interest. Note that the states of the predecessor's d buffers are assumed to be independent.

Define b_i to be the probability that a successor of a stage i switch can accept a packet. Then,

$$b_i = \begin{cases} 1 - \pi_{i+1}(\beta)\overline{q}_{i+1} & \text{for global flow control} \\ \overline{\pi}_{i+1}(\beta) & \text{for local flow control} \end{cases}$$

and

$$\begin{aligned} q_i &= \sum_{j=0}^{d-1} \frac{b_i}{j+1}\binom{d-1}{j}(\overline{\pi}_i(0)/d)^j(1-\overline{\pi}_i(0)/d)^{(d-1)-j} \\ &= \frac{b_i}{\overline{\pi}_i(0)}\sum_{j=1}^{d}\binom{d}{j}(\overline{\pi}_i(0)/d)^j(1-\overline{\pi}_i(0)/d)^{d-j} \\ &= \frac{b_i}{\overline{\pi}_i(0)}(1-(1-\overline{\pi}_i(0)/d)^d) = b_i a_{i+1}/\overline{\pi}_i(0) \end{aligned}$$

The first equality above is based on the observation that the first packet in a stage i buffer can leave if the successor it is destined for can accept it and it wins any contention that may occur between it and the other input buffers in the same switch. There are $d-1$ other input buffers that might contend with it, the probability that any one does contend is $\overline{\pi}_i(0)/d$, and the probability that the given input buffer wins, when it has to contend with j others is $1/(j+1)$.

In realistic systems, each input to the network is supplied with a buffer that is typically much larger than those in the switches. We can model such a buffer using the Markov chain shown below, where β' is the number of buffer slots, b_0 is the probability that a stage 1 switch can accept a packet offered to it (computed according to the equation for b_i given above) and ρ is the offered load, that is the probability that a packet is available to enter the buffer.

Finally, we note that a_1 is computed not according to the general equation given above but is equal to the probability that the input buffer is nonempty; also, we assume that the output of the network can always accept a packet meaning that $b_k = 1$, where $k = \log_d n$.

Given the above quantities, we can easily obtain the common performance metrics of interest. The carried load, for example, is the probability that a buffer in the last stage is non-empty and given that it is non-empty, that it is able to transmit a packet; that is, $\overline{\pi}_k(0)q_k$. The average delay through the network can be calculated by summing the average delays at each stage. The average

Figure 3: Throughput of Networks with Fifo Input Buffering (S&S Analysis)

delay at stage i is calculated using Little's Law. For the network using global grants we calculate the delay using

$$\frac{1}{a_i(1-\pi_i(B))\overline{q}_i}\sum_{j=0}^{B} j\pi_i(j)$$

In this expression, the quantity in the denominator of the initial fraction is the average arrival rate at stage i and the summation is the average queue length. For local grants, we just substitute $a_i\overline{\pi}_i(B)$ for the expression in the denominator. The delay in the input buffer can be calculated in a similar fashion.

The performance curves shown in Figure 3 were computed with this method. The leftmost and center pairs of plots show the maximum obtainable throughput as a function of network size for networks comprising switches of different sizes and varying amounts of buffering. The rightmost plots show the effect of varying the amount of buffering for switches with 256 inputs. The curves on the left show the throughput in the case of local flow control and those on the right are for global flow control. It's interesting to note that the networks constructed from larger switches have lower throughput when n is large. This appears to be caused by two mechanisms. First, because the networks constructed from larger switches have fewer stages for a given value of n, they have less buffering overall. Secondly, the head-of-line blocking that occurs in these networks has a greater effect on the networks made up of large switches, since a blocked packet can affect packets with a wider range of destination addresses in this case.

Figure 4: Switch with Shared Buffering

III. Analysis of Networks with Shared Buffering

It's well known that switching networks in which buffers are shared among the inputs can yield better performance than those in which buffers are dedicated either to inputs or outputs. Figure 4 shows a switch in which packets arriving at any of d inputs are placed in available buffer slots from a pool containing B slots. Packets are routed from the shared buffer to the appropriate outputs. An implementation of such a switch would require a $d\times B$ crossbar to distribute arriving packets to buffers and a separate $B\times d$ crossbar to route packets from buffers to outputs.

As in the input buffered switch, one can use either local or global flow control, but we analyze only the case of local flow control. There are two additional possibilities for implementing local flow control which we refer to as the *grant* and *acknowledgement* methods. In the grant method of flow control, a switch with x empty buffer slots, grants permission to send a packet to min $\{x,d\}$ of its upstream neighbors at the start of an operation cycle of the switch. If $x<d$, we assume that x predecessors are chosen at random. Notice that in this method, a switch supplies grants to upstream neighbors without knowing which of them has packets to send. This can result in sending a grant to a neighbor that doesn't have a packet, while a neighbor that does have a packet may not receive a grant. The acknowledgement method of flow control remedies this fault by allowing all predecessors

with packets to send them. The receiving switch stores as many as it can in its buffer and acknowledges their receipt by means of a control signal. Unacknowledged packets are retransmitted during a subsequent cycle. The acknowledgement method requires that the predecessors hold a copy of a packet pending an acknowledgement, but allows better buffer utilization overall.

We first analyze a network using the grant method of flow control. We model each switch as a $B+1$ state Markov chain. We let $\pi_i(s)$ be the steady state probability that a stage i switch contains exactly s packets and we let $\lambda_i(s_1, s_2)$ be the probability that a stage i switch contains s_2 packets in the current cycle given that it contained s_1 packets during the previous cycle.

Let $p_i(j,s)$ be the probability that j packets enter a stage i switch that has s packets in its buffer and let $q_i(j,s)$ be the probability that j packets leave a stage i switch that has s packets in its buffer. Then

$$\lambda_i(s_1,s_2) = \sum_{0, s_2-s_1 \le h \le d, s_2, B-s_1} p_i(h,s_1) q_i(h-(s_2-s_1), s_1) \tag{1}$$

Let a_i be the probability that any given predecessor of a stage i switch has a packet for it. Then if we let $m = \min\{d, B-s\}$,

$$\begin{aligned} p_i(j,s) &= \binom{m}{j} a_i^j (1-a_i)^{m-j} && (2) \\ a_i &= \sum_{0 \le j \le B} \pi_{i-1}(j)\left[1-(1-1/d)^j\right] && (3) \end{aligned}$$

Let b_i be the probability that a successor of a stage i switch provides a grant and let $Y_d(r,s)$ be the probability that a switch that contains s packets, contains packets for exactly r *distinct* outputs. Then

$$\begin{aligned} q_i(j,s) &= \sum_{j \le r \le d,s} Y_d(r,s) \binom{r}{j} b_i^j (1-b_i)^{r-j} && (4) \\ b_i &= \sum_{0 \le h \le B-d} \pi_{i+1}(h) + \\ & \quad \sum_{0 \le h \le d-1} \pi_{i+1}(B-h) h/d && (5) \end{aligned}$$

Y is easily calculated, assuming all distributions of s packets to the d outputs are equally likely. This is just a classical distribution problem. For the purposes of calculation, the following recurrence is all we require.

$$Y_d(r,s) = \begin{cases} 1 & s=r=0 \\ 0 & (s>0 \wedge r=0) \vee s<r \\ \frac{r}{d} Y_d(r,s-1) + \frac{d-(r-1)}{d} Y_d(r-1,s-1) & \\ & 0 < r \le s \end{cases}$$

Note that $Y_d(r,s)$ is independent of the stage of the switch in the network. For computational purposes, it is most convenient to merely precompute a table with the values of Y required; the above recurrence is ideal for this purpose. As in the earlier analysis, we compute performance parameters by assuming a set of initial values for $\pi_i(j)$, then use these and the equations given above to compute $\lambda_i(s_1,s_2)$. These, together with the balance equations for the Markov chain are used to obtain new values of $\pi_i(j)$ and then we iterate until we obtain convergence. While convergence is not guaranteed, our experience has shown convergence to be fairly rapid except when the offered load is approximately equal to the network's maximum throughput; when the offered load is below this critical point, convergence is obtained in fewer than 100 iterations, above the critical point convergence typically requires several hundred iterations and in the vicinity of the critical point, it may require several thousand iterations.

Notice that the calculation of $Y_d(r,s)$ given above relies on the assumption that the addresses of the packets stored within a switch's buffer are independent. This is not in fact the case. While it is true that the addresses of packets arriving at a switch are independent (given the input traffic assumptions), buffered packets are correlated as a result of having contended for outputs. The correlations are strongest when d is small and B large.

We can now easily obtain the performance metrics of interest. The carried load is given by

$$\sum_{j=0}^{d} \sum_{s=0}^{B} j q_k(j,s) \pi_k(s)$$

The average delay at stage i is given by

$$\frac{\sum_{s=0}^{B} s\pi_i(s)}{\sum_{j=0}^{d} \sum_{s=0}^{B} j p_i(j,s) \pi_i(s)}$$

In this expression, the quantity in the numerator is the average queue length in the stage i buffer and the denominator is the average arrival rate. Total delay is obtained by summing the per stage delays.

Most of the above analysis carries over to networks that use the acknowledgement method of flow control. The only changes required are in the equations for $p_i(j,s)$ and b_i. In particular, we have

$$p_i(j,s) = \begin{cases} 0 & B-s<j \\ \binom{d}{j} a_i^j (1-a_i)^{d-j} & B-s>j \\ \displaystyle\sum_{j \le h \le d} \binom{d}{h} a_i^h (1-a_i)^{d-h} & B-s \ge j \end{cases} \tag{6}$$

and

$$\begin{aligned} b_i \;=\; & \sum_{0 \le h \le B-d} \pi_{i+1}(h) + \sum_{1 \le h \le d-1} \pi_{i+1}(B-h) \\ & \left[\sum_{0 \le r \le h-1} \binom{d-1}{r} a_{i+1}^r (1-a_{i+1})^{(d-1)-r} + \right. \\ & \left. \sum_{h \le r \le d-1} \frac{h}{r+1} \binom{d-1}{r} a_{i+1}^r (1-a_{i+1})^{(d-1)-r} \right] \end{aligned} \tag{7}$$

Figure 5: Carried Load Curves for Networks with Shared Buffering (solid: analysis, dashed: simulation)

Figure 5 shows curves of offered load vs. carried load for networks with 256 inputs and outputs and varying switch and buffer dimensions. In the plots $\beta = B/d$ is the number of buffer slots per switch input, the solid lines are the analytical results, while the dashed lines are simulation results. We note that for shared buffer networks, large switches usually perform just slightly better than small ones with the same values of β. The advantage of the acknowledgement method of flow control is most pronounced when the number of buffer slots is limited, although one would expect a greater benefit in the presence of unbalanced traffic patterns.

The analysis is optimistic in the sense that it predicts higher carried loads than the simulation. This is typical of such analytical techniques. Notice that the analytical results are most accurate when the switch size is largest and the buffering is smallest. Haifeng Bi [1] has traced the source of the discrepancy to the independence assumption mentioned above. Because the analysis neglects the correlations among the destination addresses for packets buffered in a given switch, it overestimates the number of distinct outputs for which packets are present in a given state. This in turn, leads to an overestimate of the number of packets leaving a switch in a given state. As an experiment, Bi ran modified simulations in which correlations among packets in a switch were systematically eliminated by randomly reassigning their addresses at the start of each simulation cycle. The simulation results obtained in this way were virtually identical with the analytical results, meaning that the crucial direction for further refinement of the analytical models lies in capturing the effects of correlations among packets.

The simulation results given above, are taken from an extensive simulation study described in reference [1]. The simulation results consistently reveal the same characteristics mentioned above for all the queueing models we study; that is, the analysis overestimates the maximum carried load and its accuracy is best for networks comprising large switches with limited buffering. The simulation and analysis do rank the different buffering techniques consistently making it possible to compare different buffering techniques qualitatively using the analytical methods. We include no further simulation results here. Interested readers can find further details in [1].

Figure 6 shows curves of average delay. The curves that become constant for large load give the delay through the network itself. The curves that rise steeply for large loads include the delay through the input buffer in addition to the network delay. We note that for offered loads below the maximum capacity of a given network the total delay is generally between 1 and 2 times the number of stages in the network, yielding an advantage for networks with large switches. We also note that the maximum network delay for a given configuration is generally smallest for $\beta = 1.5$ or 2. Figure 7 gives maximum throughput curves for networks with shared buffering of varying size and buffer capacities.

IV. Improved Analysis of Networks with Input Buffering

We now return to the study of networks comprising switches using input buffering. In addition to switches that use fifo buffers, we are interested in switches that use *bypass buffering* to avoid the head-of-line blocking effects that limit the performance of systems with fifo buffer-

Figure 6: Delay Curves for Networks with Shared Buffering

ing. Two types of bypass buffering are possible. In *serial bypass*, the first packets in a switch's input buffers first contend for outputs, then the losing input buffers that contain a second packet are allowed to contend a second time, those that lose in the second round and have a third packet are allowed to contend a third time, and so forth. In *parallel bypass*, all packets in a switch contend in a single round with the winners proceeding to the outputs. This allows more than one packet from a given input to proceed during a single cycle, allowing potentially higher performance, although of course each output can carry at most one packet per cycle. In high speed systems, parallel bypass is actually somewhat easier to implement, as one does not have the overhead of multiple contention rounds. For this reason and because it is more straightforward to analyze, we concentrate here on parallel bypass.

The analysis of a network with parallel bypass input buffering is similar to that for a network with shared buffers using the grant method of flow control. In particular, we need only alter the equations for b_i and $p_i(j,s)$. As previously, B is the total number of buffers in a switch and $\beta = B/d$. Let $X_d^\beta(j,s)$ be the probability that a given input buffer has j packets given that the switch as a whole contains s. Then,

$$b_i = \sum_{0 \le s \le B} \pi_{i+1}(s)(1 - X_d^\beta(\beta, s)) \tag{8}$$

Next, let $W_d^\beta(r,s)$ be the probability that exactly r input buffers are not full given that a switch contains exactly s packets. Then,

$$p_i(j,s) = \sum_{j \le r \le d} W_d^\beta(r,s) \binom{r}{j} a_i^j (1-a_i)^{r-j} \tag{9}$$

X and W are easily computed, assuming that when the switch contains s packets, all distributions of those packets among the input buffers are equally likely. This assumption is not really correct, as it neglects correlations among packets in a given switch, resulting from prior contention for outputs. Bi [1] presents simulation results quantifying the discrepancy caused by this assumption; the results are similar to those cited above. Let $z_d^\beta(s)$ be the number of ways to distribute s distinct objects (packets) among d distinct containers (input buffers), under the restriction that each container may contain at most β objects. Also, let $x_d^\beta(r,s)$ be the number of ways to distribute s objects among d containers of capacity β, so that a particular container receives exactly r objects. Then

$$X_d^\beta(r,s) = x_d^\beta(r,s)/z_d^\beta(s)$$

Similarly, if $w_d^\beta(r,s)$ is the number of distributions that leave exactly r containers with fewer than β objects, then

$$W_d^\beta(r,s) = w_d^\beta(r,s)/z_d^\beta(s)$$

We compute z, x and w as follows,

$$z_d^\beta(s) = \begin{cases} 1 & s = 0 \\ 0 & s > d\beta \\ \displaystyle\sum_{0 \le i \le \beta, s} \binom{s}{i} z_{d-1}^\beta(s-i) & \\ & 0 < s \le d\beta \end{cases}$$

Figure 7: Throughput Curves for Networks with Shared Buffering

$$
\begin{aligned}
x_d^\beta(r,s) &= \binom{s}{r} z_{d-1}^\beta(s-r) \\
w_d^\beta(r,s) &= \binom{d}{r}\binom{s}{(d-r)\beta} \times \\
&\qquad z_r^{\beta-1}(s-(d-r)\beta)\, z_{d-r}^\beta((d-r)\beta)
\end{aligned}
$$

Using these equations, it is straightforward to compute tables containing the requisite values of X and W.

We now return to the case of an input buffered network with fifo buffers. Most of the analysis for bypass input buffering carries over to this case. The two equations requiring modification are those for a_i and $q_i(j,s)$. Let $Y_d^\beta(r,s)$ be the probability that exactly r input buffers contain at least one packet, given that the switch contains s packets. Then, assuming that when a switch contains s packets, all distributions of the packets among the outputs are equally likely, and that the destination addresses of all packets are independent,

$$
\begin{aligned}
a_i &= \sum_{0\le s\le B} \pi_{i-1}(s) \times \\
&\qquad \sum_{0\le r\le d,s} Y_d^\beta(r,s)(1-(1-1/d)^r) \qquad (10) \\
q_i(j,s) &= \sum_{j\le h\le d,s} Y_d^\beta(h,s) \times \\
&\qquad \sum_{j\le r\le h} Y_d(r,h)\binom{r}{j} b_i^j (1-b_i)^{r-j} \qquad (11)
\end{aligned}
$$

(Once again, the assumption made above neglects correlations among packets, caused by prior contention.) If we let $y_d^\beta(r,s)$ be the number of ways to distribute s objects among d containers so that exactly r containers receive one or more objects, then

$$Y_d^\beta(r,s) = y_d^\beta(r,s)/z_d^\beta(s)$$

and $y_d^\beta(r,s)$ is computed using the recurrence

$$
\begin{aligned}
y_d^\beta(r,s) &= y_{d-1}^\beta(r,s) + \\
&\qquad \sum_{1\le i\le \beta,\, s-(r-1)} \binom{s}{i} y_{d-1}^\beta(r-1, s-i)
\end{aligned}
$$

when $0 < r \le s \le d\beta$ and $r \le d$; $y_d^\beta(r,s) = 1$ when $r = s = 0$ and $y_d^\beta(r,s) = 0$ when $s < r$ or $d < r$ or $d\beta < s$ or $r = 0 < s$.

Figure 8 gives curves of maximum throughput for networks comprising switches with both fifo and parallel bypass input buffering, of varying size and buffer capacity. We note that bypass buffering gives a very substantial improvement over fifo buffering and that larger buffers yield a greater improvement in the case of bypass buffering. It's also worthwhile to note the differences between the curves in the top row of Figure 8 to the corresponding curves in the top row of Figure 3 that were obtained using Szymanski and Shaikh's analysis. The prior analysis is consistently more optimistic than the method described here. However in many cases the improvement obtained with the new method is slight.

V. Conclusions

Figure 9 lists the numbers of the equations used to compute the key quantities for each of the four buffering

Figure 8: Throughput Curves for Networks with Input Buffering

methods analyzed here. Figure 10 compares the maximum throughput obtained with the various buffering methods and networks of varying size, switch dimension and buffer capacity. We show curves for shared buffering using both the grant and acknowledgement methods of flow control. We show curves for input buffering using local flow control, with bypass queueing and fifo queueing. We note that shared buffer switches offer clearly superior performance for a given amount of buffering, but bypass input buffering performs impressively as well. Fifo input buffering, performs rather poorly in comparison to the other methods, but may be acceptable in certain applications. Interestingly, variation in switch size yields only small changes in maximum throughput for networks with the same values of β, but the reduction in the number of stages obtained with larger switches yields a significant economy in implementation, as well as lower delays. We note that the acknowledgement method of flow control yields only modest improvements over the grant method when we have uniform random traffic with Bernoulli arrivals. This appears surprising until one realizes that in the presence of heavy uniform random traffic, all the predecessors of a given switch are likely to have one or more packets to send it at any one time. We would expect a greater difference in the face of non-uniform bursty traffic, since in this case, there can be substantial differences in the instantaneous traffic from the upstream neighbors.

	Shared Buffering		Input Buffering	
	grant	ack	bypass	fifo
λ_i	1	1	1	1
p_i	2	6	9	9
a_i	3	3	3	10
q_i	4	4	4	11
b_i	5	7	8	8

Figure 9: Equations Used in Various Queueing Models

Our computational experience with the method described here is quite favorable. In collecting the data for all the curves shown in this paper, we computed approximately 900 data points and used a total of 40 hours of CPU time on a Sun Sparcstation I, with 16 Mbytes of memory, yielding an average of about 2.67 minutes per data point. The program required to compute the results for input fifo buffering consists of about 8 pages of C++ code. The memory requirements for this program are under 3 Mbytes when dimensioned for networks with up to 12 stages, switches with up to 32 inputs and a total of up to 100 buffer slots. The other programs are a little smaller in both code and memory usage. The switch dimension and buffer capacity both have a strong influence on the running time and memory requirements. We have computed results for switches with $d = 32$ and $\beta = 3$, but this is about as far as one can reasonably push the method with typical workstations. Fortunately, this covers the cases of greatest interest, as larger switches are difficult to implement. Also, the relative insensitivity of the results on switch dimension allows one to extrapolate to networks of larger switches with a good deal of confidence. Since the running time is relatively insensitive to the total size of the network, it is far superior to simulation when modeling networks with hundreds or thousands of inputs. We note that, the analysis also supports computation of delay distributions, and packet loss rates in addition to average delay and throughput.

As mentioned above, Haifeng Bi [1] has made a careful evaluation of the analytical techniques described here by comparing them with simulation and has both quantified the discrepancies and identified the crucial contributing factors. Bi also studied networks comprising switches with output buffering and made a systematic comparison of output buffering with input buffering. His work demonstrates that the difference commonly noted between input buffering and output buffering is less significant than commonly assumed. What most authors overlook is that in a switch with output buffering, the internal crossbar or bus required to provide access to the outputs requires greater capacity than the crossbar required in a switch using fifo input buffering. In conventional output buffering, each buffer can receive up to d packets per cycle, whereas in fifo input buffering, each buffer can transmit only one packet per cycle. If one compares conventional output buffering to bypass input buffering, where each input buffer can transmit multiple packets in a given cycle, the difference between the two disappears. Bi has compared generalized forms of input and output buffering. In his study, there is a parameter r that limits the number of packets that can be transmitted from an input buffer in a given cycle or received at an output buffer. His results show that there is very little difference between input and output buffering when they operate under the same restrictions with respect to crossbar access; interestingly in fact, input buffering enjoys a slight advantage due to "boundary effects" at the first and last stages of the network.

An interesting extension of this work would be to modify our methods to allow modeling of systems with global flow control. This appears difficult and may be of only academic interest, but nonetheless it would be interesting to compare with the earlier analyses. Extension of our models to networks with uneven traffic distribution would also be worthwhile. We expect this could be done following the pattern established in [5]. Networks that perform distribution and/or packet replication are of substantial practical interest currently [7]. The extension of our models to cover distribution networks appears straightforward; the case of replication is more challenging but may prove tractable.

Finally, these techniques are directly applicable to the study of several switching system architectures that are under development for ATM networks [2, 3, 7]. A detailed comparison of these systems using the tools we have developed could have an important practical impact on the development of emerging networks.

Acknowledgements. My thanks to Hiafeng Bi for providing the simulation results reported here and for identifying several errors in an earlier version of the paper. Also, I thank Ankira Arutaki and an anonymous reviewer for their careful reading and incisive comments.

References

[1] Bi, Haifeng. "Queuing Analysis of Buffered Packet Switching Networks," MS thesis, Washington University Computer Science Department, 8/90.

[2] Coudreuse, J. P. and M. Servel. "Prelude: An Asynchronous Time-Division Switched Network," *International Communications Conference*, 1987.

[3] De Prycker, M., and J. Bauwens. "A Switching Exchange for an Asynchronous Time Division Based Network," *International Communications Conference*, 1987.

[4] Jenq, Yih-Chyun. "Performance Analysis of a Packet Switch Based on a Single-Buffered Banyan Network," *IEEE Journal on Selected Areas in Communications*, vol. SAC-1, no. 6, 12/83, 1014–1021.

[5] Kim, Hyong Sok and Alberto Leon-Garcia. "Performance of Buffered Banyan Networks under Nonuniform Traffic Patterns," *Proceedings of Infocom 88*, 4/88.

[6] Szymanski, Ted and Salman Shaikh. "Markov Chain Analysis of Packet-Switched Banyans with Arbitrary Switch Sizes, Queue Sizes, Link Multiplicities and Speedups," *Proceedings of Infocom 89*, 4/89.

[7] Turner, Jonathan S. "Design of a Broadcast Packet Network," *IEEE Transactions on Communications*, June 1988.

Figure 10: Comparison of Buffering Methods