IMPROVED QUEUEING ANALYSIS OF SHARED BUFFER SWITCHING NETWORKS

GIUSEPPE BIANCHI and JONATHAN S. TURNER FELLOW, IEEE

Abstract — **This paper describes several methods for analyzing the queueing behavior of switching networks with flow control and shared buffer switches. It compares the various methods on the basis of accuracy and computation speed, where the performance metric of most concern is the maximum throughput. The best of the methods accurately predicts throughput for multi-stage networks constructed from large switches (≥ 8 ports).**

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

I. Introduction

In a widely cited paper [2], 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, throughput and packet loss performance of a network. Szymanski and Shaikh [5] extended Jenq's method to switching systems constructed from switches with an arbitrary number of inputs and an arbitrary number of buffer slots.

Turner [6] developed a similar method which can be applied to switching networks with shared buffering. Recently, Pattavina and Monterosso [4] developed a new approach which is based on an exact model of a single shared-buffer switch element. This model, while more accurate is computationally intractable for networks constructed from large switches. In this paper we describe a series of improvements to Turner's method which rival the accuracy of Pattavina and Monterosso's method while maintaining the computational effectiveness of Turner's.

The analysis considered here is for delta networks (see [6]) constructed from switches with d input and output ports. We use n to denote the number of network inputs and outputs and let $k = \log_d n$ denote the number of stages in the network. Each switch used to construct the network is assumed to have a single shared buffer with B buffer slots. Packets from any input can be placed in any available buffer slot and packets can proceed from any buffer slot to the desired output. Packets arriving at the inputs to the network are assumed to be assigned independent random output addresses.

Typically these systems are operated in a time-slotted fashion, with fixed length packets progressing synchronously from stage to stage. Low level flow control mechanisms regulate the flow of packets between stages to prevent packets from being lost due to buffer overflow. Consequently, we can think of the system as operating in two phases. In one phase, flow control information passes through the network from right to left. In the other phase, packets flow from left to right, in accordance with the flow control information. There are several types of flow control that can be used. In *local flow control*, the flow control signals that a switch sends to its upstream neighbors depend only on the number of empty buffer slots in the switch, while in *global flow control*, the flow control signals may also depend on the signals received from downstream neighbors. In *grant flow control*, the flow control signals grant the upstream neighbor permission to send a packet, while in *acknowledgement flow control*, the signals acknowledge successful receipt of a packet and let the sender know that he can safely discard a retained copy. Consider a shared buffer switch employing local/grant flow control and assume that the number of unoccupied buffer slots is x. The switch grants permission to send to $\min\{x, d\}$ of its upstream neighbors at the start of an operation cycle of the network. If $x < d$, we assume that x predecessors are chosen at random. For most of this paper we focus on local/grant flow control, although in Section 6, we will consider the other variants as well. Local/grant control is the easiest to implement and appears to have most practical interest while global/acknowledgement control provides the best performance.

Turner [6] models a network of shared buffer switches by modeling a shared buffer switch as a $B+1$ state Markov chain. We review this result briefly here. Let $\pi_i(s)$ be the steady state probability that a stage i switch contains exactly s packets and we let $\lambda(s_1, s_2)$ be the probability that a switch with s_1 packets during a given cycle contains s_2 packets in the subsequent cycle. In the steady state,

$$\pi_i(s_2) = \sum_{0 \leq s_1 \leq B} \pi_i(s_1)\lambda_i(s_1, s_2)$$

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

[0] This work was supported by the National Science Foundation (grant NCR 8914396), Ascom Timeplex, Bell Communications Research, Bell Northern Research, DEC, Goldstar Information and Communications, Italtel SIT, NEC, NTT and SynOptics.

Figure 1: Throughput of Single Stage Networks

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

$$\lambda_i(s_1, s_2) = \sum_{\substack{0 \le j_1 \le d \\ s_1 + j_1 - j_2 = s_2}} p_i(j_1, s_1) q_i(j_2, s_1)$$

Let a_i be the probability that any given predecessor of a stage i switch has a packet for it. Then,

$$\begin{aligned} p_i(j, s) &= \mathbf{B}(j, \min\{d, B - s\}, a_i) \\ a_i &= \sum_{0 \le s \le B} \pi_{i-1}(s) a_i(s) \qquad (i > 1) \end{aligned}$$

where $\mathbf{B}(j, n, x) = \binom{n}{j} x^j (1 - x)^{n-j}$ and $a_i(s)$ is defined as the probability that a predecessor of a stage i switch has a packet for it, given that the predecessor has s cells. We take a_1 to be the offered load and we approximate $a_i(s)$ by $\hat{a}_i(s) = 1 - (1 - 1/d)^s$ for $i > 1$. Note that this approximation assumes that the destinations of the packets in a switch are independent of one another.

Let b_i be the probability that a successor of a stage i switch provides a grant and let $Y_d(c, s)$ be the probability that a switch containing s packets, contains packets for exactly c *distinct* outputs. An output of a switch is called *active* if the switch contains some packet which is to be sent out using that output, so $Y_d(c, s)$ is the probability that a d port switch with s packets has c active outputs.

$$\begin{aligned} q_i(j, s) &= \sum_{j \le c \le d, s} Y_d(c, s) \mathbf{B}(j, c, b_i) \\ b_i &= \sum_{0 \le s \le B} \pi_{i+1}(s) \min\{1, (B - s)/d\} \end{aligned}$$

We assume that network outputs can always accept packets and model this by letting $b_k = 1$.

We can approximate Y by assuming that all distributions of s packets to the d outputs are equally likely. This is just a classical distribution problem. For the purposes of calculation, we can use the recurrence

$$\hat{Y}_d(r, s) = \frac{r}{d} \hat{Y}_d(r, s - 1) + \frac{d - (r - 1)}{d} \hat{Y}_d(r - 1, s - 1)$$

when $0 < r \le s$, $\hat{Y}_d(r, s) = 1$ when $s = r = 0$ and $\hat{Y}_d(r, s) = 0$ otherwise. 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 $\hat{Y}$ required; the above recurrence is ideal for this purpose. We compute performance parameters by assuming a set of initial values for $\pi_i(j)$, then use these and the equations given above

Figure 2: Packet Loss in Single Stage Networks

to compute $\lambda_i(s_1, s_2)$. These, together with the steady-state equations for the Markov chain are used to obtain new values of $\pi_i(j)$ and we iterate until we obtain convergence. This type of analysis is referred to as the *scalar method*.

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

$$(1/d) \sum_{0 \le j \le d} \sum_{0 \le s \le B} j q_k(j, s) \pi_k(s)$$

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. The packet loss rate is the ratio of the offered load minus the carried load to the offered load.

The scalar method is reasonably fast, even for switches with large values of B and d. The computation is dominated by the time required to compute the transition probabilities λ_i. If m iterations are required for convergence, this is proportional to $m(B/d)d^3 \log_d n$, while the time required for simulation is proportional to $m' n \log_d n$, where m' is the number of simulations steps required to get accurate results. Typically $m << m'$ and the elementary step in the analysis is faster to compute than the elementary step in the simulation (which typically involves making some routing decision on a packet data structure and moving it from one queue to another). Moreover, for networks with more than 3 stages $d^3 < n$, so the analysis' computational advantage grows with network size. 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, and the calculation of a_i rely 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

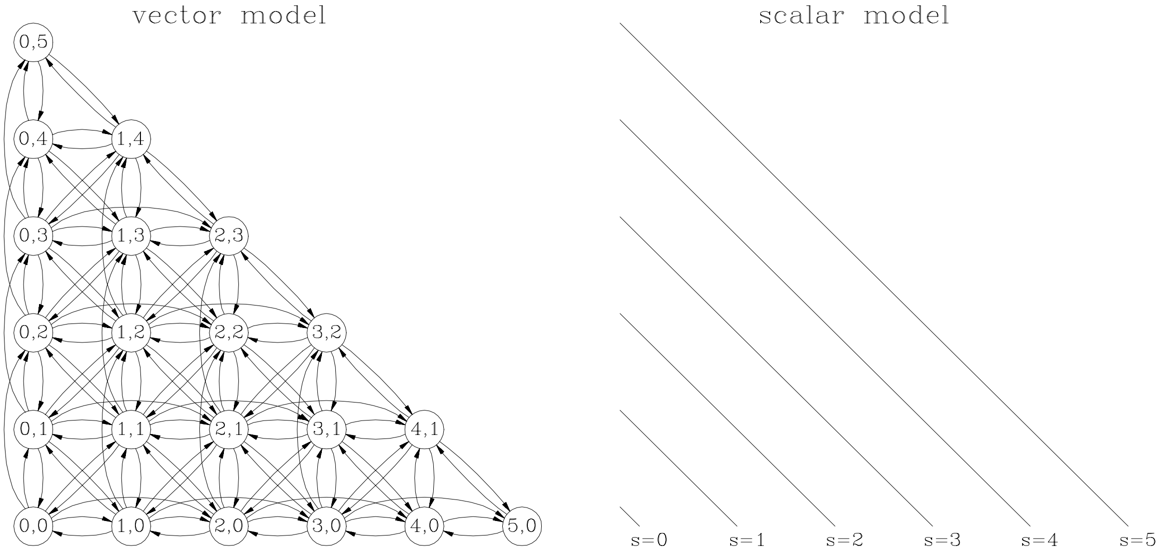

Figure 3: Comparison of Vector and Scalar Methods for Binary Switches

outputs. The correlations are strongest when d is small and B large. The independence assumption causes the analysis to overestimate the maximum throughput. This is illustrated in Figure 1 which shows the maximum load that can be achieved by single stage networks for various values of d and B. Note that the throughput predicted by the scalar method can exceed that of the analysis by more than 10%. Figure 2 shows the packet loss experienced in several single stage networks as a function of offered load. Note that for larger amounts of buffering, the scalar method can grossly under-estimate the loss rate when compared to simulation.

One comment regarding our use of results for single stage networks. While in practice, we are mainly interested in applying our analytical methods to multistage networks, we use results for single stage networks to make detailed comparisons among methods. This allows us to separate the issues surrounding the modeling of individual switches from the inter-stage issues that affect all the models in the same way. This will become clear in the final section of the paper where we give results for multistage networks.

Monterosso and Pattavina [4] have developed an exact model of a shared buffer switch which they call the *vector method*. In this approach, the state of a switch is represented by a vector $[s_1, \ldots, s_d]$, where s_j denotes the number of stored packets for output j. The drawback of this method is of course that the number of states grows exponentially with d and the number of transition probabilities is roughly the square of the number of states. While Pattavina and Monterosso have taken care to avoid explicit representation of redundant states, a 16 port switch with 32 buffer slots still has about 30 thousand states and there are more than 10 million transition probabilities. In the scalar model, on the other hand, the same switch has 33 states and fewer than 1000 transition probabilities. These considerations limit the applicability of the vector method to networks constructed from small switches ($d \leq 4$). The objective of this paper is to develop an analytical method which rivals the accuracy of the vector method while retaining the computational effectiveness of the scalar method. We describe three techniques. The first is a variant of the scalar method, which we call the *uniform scalar* method. The second method models the state of a switch using two variables and is called the *bidimensional method*. As we will see, the bidimensional method is more accurate but also more computationally expensive. Consequently, we have developed a third method called the *interval method* which is intermediate in both accuracy and speed.

II. The Uniform Scalar Method

The scalar method of queueing analysis computes the number of active outputs that a switch has using the assumption that if s packets are stored in a switch's buffer, that the outputs those packets are to take are independent of one another. To understand the implications of this assumption, it's helpful to compare the vector and scalar methods for binary switches ($d = 2$). Figure 3 shows the Markov chain corresponding to the vector model of a two port switch, with five buffer slots. In the illustration, a state (i, j) represents a switch in which i packets are destined for output 0 and j are destined for output 1. In the scalar method, a state corresponds to the sum $s = i + j$, so that the states of the scalar method correspond to sets of states in the vector method that lie along a common diagonal, as shown in the figure.

If a switch is in state (i, j) (vector method) where i or j but not both are equal to zero, then there is one active output. We refer to these as boundary states. The scalar method calculates the probability of these boundary states relative to the non-boundary states along a given diagonal by assuming that the packet destinations

Figure 4: Comparison of Probabilities for Diagonal States

are independent. This can be interpreted as assigning probabilities to the states along a given diagonal according to a binomial distribution. In particular, given that there are s packets in a switch, the probability assigned to state (i, j) is $\binom{s}{i}(1/2)^s$ (where $s = i + j$).

Figure 4 plots these probabilities as a function of i, when $s = 14$ (the solid curve with the peak at 7). The dashed curve in the figure is a measured frequency distribution taken from a simulation of a 2 port switch with a 16 packet buffer and an offered load of 100%. In this simulation, the frequency of the various states of the vector method were recorded, and the relative frequencies were plotted for those states (i, j) with $i + j = 14$. (When the offered load is 100%, these states account for almost all of the probability.) Note that the binomial assignment grossly under-estimates the probability of the boundary states ($i = 0$ and $i = 14$). Also, note that the relative frequencies of the states on the diagonal appear to be well-approximated by a uniform distribution, that is equal values for all probabilities along a given diagonal. An intuitive justification for this can be found by considering the steady-state situation for a heavily loaded switch. In this case, one would expect that the state of a switch would move back and forth along a diagonal as packets come and go. While there would be some movement across diagonals as well, the predominant movement would be along a diagonal. Since there is equal probability of moving in either direction along a diagonal, one would expect the distribution along a diagonal to be approximately uniform. This reasoning leads to the *uniform scalar method* in which, when determining the number of active outputs that a switch has, we assume that the set of states of the underlying vector method corresponding to a given state of the scalar method are equally likely.

The equations for the uniform method are similar to those for the original scalar method. In fact, the only things we need change are our approximations $\hat{a}_i(s)$ and $\hat{Y}_d(c, s)$. Define, $\sigma(s, d)$ to be the number of ways d distinct non-negative integers can add up to s. For example $\sigma(2, 3) = 6$ because there are 6 different ordered triples whose sum is 2. It is easy to show that

$$\sigma(s, d) = \binom{s + d - 1}{d - 1}$$

There are $\sigma(s, d-1)$ states of the underlying vector model in which a switch with s cells contains no cell for some particular output. Hence, by the uniform assumption, a switch with s cells in its buffer has no cell for some particular output with probability $\sigma(s, d-1)/\sigma(s, d)$, and $\hat{a}_i(s) = 1 - \sigma(s, d-1)/\sigma(s, d)$. Similarly, if we fix some subset A of the outputs of a switch, then if $|A| = c$, the number of states in the underlying vector model in which A is the set of active outputs is $\sigma(s - c, c)$. So by the uniform assumption, the probability that A is the set of active outputs is $\sigma(s - c, c)/\sigma(s, d)$ and

$$\hat{Y}_d(c, s) = \binom{d}{c} \frac{\sigma(s - c, c)}{\sigma(s, d)}$$

The values of σ can be pre-computed and stored in a table, so the amount of computation required for the uniform method is essentially the same as for the original scalar method. Figure 1 shows the maximum throughput for single stage switches computed using the uniform method. The uniform method *under-estimates* the maximum throughput of these networks by a substantial margin and so is only marginally more useful than the original method. Nonetheless, as we shall see in the next section, it provides the basis for another method that is considerably better.

III. The Bidimensional Method

The two scalar methods attempt to use a single number to summarize the state of a switch. To obtain better accuracy, the bidimensional model keeps track of both the number of packets in a switch and the number of active outputs.

Let the state of a switch be a pair (s, c) where s is the number of packets and c is the number of active outputs. Then

$$\pi_i(s_2, c_2) = \sum_{(s_1,c_1)} \pi_i(s_1, c_1)\lambda_i((s_1, c_1), (s_2, c_2))$$
$$\lambda((s_1, c_1), (s_2, c_2)) = \sum_{\substack{0 \le j_1 \le d \\ s_1 + j_1 - j_2 = s_2}} p_i(j_1, s_1) q_i(j_2, c_1) R(s_1, c_1, j_1, j_2, c_2)$$

where $R(s_1, c_1, j_1, j_2, c_2)$ is the probability that if a switch is initially in state (s_1, c_1) and then j_1 packets arrive and j_2 leave, that the new state has c_2 active outputs. Note that in the bidimensional method, the the number of cells that leave (q_i) depend not on the number of stored cells but just on the number of active outputs.

To compute R, we consider an imaginary intermediate state following the departure of the j_2 packets leaving the switch, but before the arrival of the j_1 entering packets. The probability that there are exactly c active outputs in this intermediate state is $R(s_1, c_1, 0, j_2, c)$ and the probability that we go from an intermediate state with c active outputs to a final state with c_2 active outputs is $R(s_1 - j_2, c, j_1, 0, c_2)$. This leads to the recurrence.

$$R(s_1, c_1, j_1, j_2, c_2) = \sum_{0, c_1 - j_2 \le c \le c_1} R(s_1, c_1, 0, j_2, c) R(s_1 - j_2, c, j_1, 0, c_2)$$

Note that $R(s_1, c_1, 0, j_2, h)$ is independent of j_1 and that $R(s_1 - j_2, h, j_1, 0, c_2)$ is independent of both s_1 and j_2. So if we define $S(s, c_1, j, c_2) = R(s, c_1, 0, j, c_2)$ and $T(c_1, j, c_2) = R(x, c_1, j, 0, c_2)$, the sum above is equal to

$$\sum_{0, c_1 - j_2 \le c \le c_1} S(s_1, c_1, j_2, c) T(c, j_1, c_2)$$

To determine $S(s, c_1, j, c_2)$, we introduce the term *minimally active* to describe an output for which there is exactly one packet and note that if exactly c of the c_1 active outputs are minimally active that we will end up with c_2 active outputs if and only if exactly $c_1 - c_2$ of the j packets that leave belong to minimally active outputs. This occurs with probability

$$\binom{c}{c_1 - c_2}\binom{c_1 - c}{j - c_1 + c_2} \Big/ \binom{c_1}{j}$$

Let A be any fixed subset of the set of active c_1 active outputs. If $|A| = c$, then there are $\sigma(s - 2c_1 + c, c_1 - c)$ states of the underlying vector model in which A is the set of minimally active outputs. Since there are $\sigma(s - c_1, c_1)$ states of the underlying vector model in which the c_1 outputs are active, the probability that A is the set of minimally active outputs is

$$\sigma(s - 2c_1 + c, c_1 - c)/\sigma(s - c_1, c_1)$$

These observations yield the approximation

$$\hat{S}(s, c_1, j, c_2) = \sum_{0, 2c_1 - s \le c \le c_1} \binom{c_1}{c} \frac{\sigma(s - 2c_1 + c, c_1 - c)}{\sigma(s - c_1, c_1)} \binom{c}{c_1 - c_2}\binom{c_1 - c}{j - c_1 + c_2} \Big/ \binom{c_1}{j}$$

We can compute T exactly, using the recurrence

$$\begin{aligned} T(c_1, j, c_2) &= \frac{c_2}{d} T(c_1, j - 1, c_2) \\ &\quad + \left(1 - \frac{c_2 - 1}{d}\right) T(c_1, j - 1, c_2 - 1) \end{aligned}$$

when $c_1 < c_2 \le c_1 + j$ and the boundary case $T(c, j, c) = (c/d)^j$. It remains only to give the equations for p and q.

$$\begin{aligned} p_i(j, s) &= \mathbf{B}(j, \min\{d, B - s\}, a_i) \\ a_i &= \sum_{s,c} (c/d)\pi_{i-1}(s, c) \\ q_i(j, c) &= \mathbf{B}(j, c, b_i) \\ b_i &= \sum_{s,c} \pi_{i+1}(s, c) \min\{1, (B - s)/d\} \end{aligned}$$

For computational purposes, $\hat{S}$ and T can be pre-computed and stored in a table. $\hat{S}$ is the larger of the two and consumes 2 Mbytes for $d = 16$, $B/d = 4$. In fact, one can go further and pre-compute R. This will save time during the iterative computation of the steady-state probabilities but requires substantially more memory (32 Mbytes for $d = 16$, $B/d = 4$).

The maximum throughput predicted by the bidimensional method is shown in Figure 1. Notice that it closely tracks the simulation results, over-estimating slightly for small d and under-estimating slightly for large d. Figure 2 shows the loss predicted by the bidimensional method. Again note the close match with the simulation results.

IV. Interval and Threshold Methods

While the bidimensional method offers a big improvement in accuracy over the scalar methods, it has a substantially higher computational cost. We can reduce the computational cost by keeping less detailed information about the number of active outputs. The *interval method* divides the range $[0, d]$ into two or more intervals and keeps track of the probability that the number of active outputs is in a particular interval.

Let (s, C) denote a switch state where $C = [\ell, h]$ is an interval. Then,

$$\pi_i(s_2, C_2) = \sum_{(s_1,C_1)} \pi_i(s_1, C_1)\lambda_i((s_1, C_1), (s_2, C_2))$$

Figure 5: Running Times of Analysis Methods

$$\lambda_i((s_1, C_1), (s_2, C_2)) = \sum_{c_1 \in C_1} Z(c_1, s_1, C_1) \sum_{\substack{0 \leq j_1 \leq d \\ s_1 + j_1 - j_2 = s_2}} p_i(j_1, s_1) q_i(j_2, c_1) R'(s_1, c_1, j_1, j_2, C_2)$$

where $Z(c, s, C)$ is the probability that there are exactly c active outputs, given that the state is (s, C) and where $R'(s_1, c_1, j_1, j_2, C_2)$ is the probability that if a switch initially has s_1 packets and c_1 active outputs and then j_1 packets arrive and j_2 leave, that the number of active outputs in the new state is in C_2. We can compute R' as follows.

$$R'(s_1, c_1, j_1, j_2, C) = \sum_{c \in C} R(s_1, c_1, j_1, j_2, c)$$

where R is computed as in the previous section. To compute $Z(c, s, C)$, we note that in the underlying vector model, there are exactly $\sigma(s - c, c)$ states in which some particular set of size c contains all the active outputs and that if $C = [\ell, h]$, there are $\binom{d}{c}$ sets of size c to choose from. Hence, using the uniform assumption we get the following estimate for Z.

$$\hat{Z}(c, s, C) = \frac{\binom{d}{c} \sigma(s - c, c)}{\sum_{\ell \leq c' \leq h, s} \binom{d}{c'} \sigma(s - c', c')}$$

(An earlier short version of this paper [1] contained an error in the equation for $\hat{Z}$.) The equations for p and q are similar to the ones for the bidimensional method.

$$\begin{aligned} p_i(j, s) &= \mathbf{B}(j, \min\{d, B - s\}, a_i) \\ a_i &= \sum_{s, C, c \in C} Z(c, s, C)(c/d)\pi_{i-1}(s, c) \end{aligned}$$

$$\begin{aligned} q_i(j, c) &= \mathbf{B}(j, c, b_i) \\ b_i &= \sum_{s, C} \pi_{i+1}(s, C) \min\{1, (B - s)/d\} \end{aligned}$$

If the number of intervals is not too large R' can be pre-computed and stored. We define the *threshold method* to be the particular case of the interval method in which there are just two intervals. In this case, the table for R' requires 4 Mbytes for $d = 16$, $B/d = 4$.

Figure 1 shows the maximum throughput predicted by the threshold method. To produce these results, a variety of thresholds were tried and the results for the thresholds that produced the closest match to the simulation results were plotted. In general, the threshold method yields results that lie between the uniform scalar and bidimensional methods.

V. Complexity Analysis

In this section, we look at the running time required for analysis via the various methods and compare to the running time for simulation. For an n port network, let $k = \log_d n$ be the number of stages. Consideration of the equations for each of the methods studied shows that for large values of d and $B/d \geq 2$, the running time can be approximated by $Cmk\left(\frac{B}{d}\right)d^r$ where C is the time for a basic step in the computation, m is number of iterations required for convergence and $r = 3$ for the scalar methods, $r = 4$ for the threshold method and $r = 6$ for the bidimensional model, assuming that S and T are pre-computed. From experimental measurements, we've found that C is typically around 10 μs on a SparcStation 2 and m is typically around 100.

Figure 6: Comparison of Analysis and Simulation Run Times

Figure 5 compares the measured run times for the different analytical methods to one another and to that predicted by the above equations, as a function of d. Note that the times have all been divided by d^3 to compress the vertical axis and simplify comparisons. We note that over the range of values shown, the running times indicated by the above analysis give a crude, but still useful approximation. For the bidimensional case, the running time matches $Cmk(B/d)d^5$ more closely than the predicted $Cmk(B/d)d^6$. We expect that for larger values of d, the predicted asymptotic behavior would be observed, but have not verified that.

We can do a similar analysis of the running time for a switch simulation. The simulation time takes the form $C'm'kd^k$ where C' is the time to simulate one time step at one switch element divided by d and m' is the number of iterations required for the simulation. For our simulation, we have found that C' is around 50 μs and m' can be anywhere from 10^4 to 10^6 depending on the data of interest. (This of course, is for uniform random traffic with Bernoulli arrivals.) For throughput measurements, we have found $m' = 10^4$ to be adequate for obtaining statistically reliable results.

The analysis is most attractive in comparison to simulation for large networks. This can be seen in a concrete way in Figure 6 which compares the running time estimates for the analysis and simulation for networks with up to 16,384 ports. These comparisons were made using the values of $C = 10$ μs, $m = 100$, $C' = 50$ μs and $m' = 10,000$. Note that in this comparison, the bidimensional method can be more expensive than simulation, but it still retains an advantage for the largest networks and for smaller values of d.

VI. Closing Remarks

Figure 7 shows the maximum throughput for multistage networks constructed from switches of various sizes. These curves show that as the number of stages in a network increases, all the analysis methods overestimate the throughput. This can be explained by the fact that none of the methods account for correlations between the states of switches in different stages. This is a common failing for this entire class of methods, including the vector method of Monterosso and Pattavina. These correlations are strongest for networks with lots of stages and small switches. For networks constructed from larger switches, the inter-stage correlations are relatively weak, allowing more reliable throughput predictions. In particular, we see that the bidimensional method gives acceptable results for switches with $d \geq 8$ and even the threshold method provides reasonable accuracy.

We have applied the bidimensional analysis method to determine packet loss in a network with mixed results. In realistic system configurations, a multistage network is typically preceded by a collection of input buffers with flow control between the network and the input buffers. Packet loss can occur at these input buffers as the load offered to the network approaches its maximum throughput. The bidimensional method can accurately predict packet loss for single stage networks with no input queues (as shown in Figure 2) or very small input queues. However, when larger input queues are added (≥ 8 slots), the accuracy rapidly deteriorates. This is unfortunate since the ability to accurately predict packet loss with a fast analytical method would offer great advantages over simulation. We have considered two possible explanations for the inaccuracy of the bidimensional method in the presence of input queues. The first is that the extension

Figure 7: Throughput of Larger Networks

of the bidimensional method to model the input queues, treats the different queues feeding a single switch as independent of one another, where in fact their states are strongly correlated. Unfortunately, an alternative model of the input queue which captures these correlations provides only a marginal improvement in the prediction of packet loss. The second (and more important) explanation was that the analysis ignores time correlations that become increasingly significant as the amount of input buffering is increased. Another model was developed to attempt to capture the effects of these time correlations and while the results were far more accurate, the complexity of the model makes it of little practical value.

All the analysis methods presented above can be extended to handle networks that use different flow control methods. We present here the equations for the bidimensional methods. The scalar and interval methods can be extended to handle the different flow control options in a similar fashion. For *local acknowledgement* flow control, the equations for π_i and λ_i are unchanged but the equations for p_i and q_i require some modification.

$$p_i(j,s) \;=\; \begin{cases} 0 & \text{if } j > B-s \\ \mathbf{B}(\geq B-s, d, a_i) & \text{if } j = B-s \\ \mathbf{B}(j,d,a_i) & \text{if } j < B-s \end{cases}$$

$$a_i \;=\; \sum_{s,c} (c/d)\pi_{i-1}(s,c)$$

where we use the notation $\mathbf{B}(\geq B-s, d, a_i)$ to denote $\sum_{j\geq B-s}\mathbf{B}(j,d,a_i)$. In the context of acknowledgement flow control, b_i is defined as the probability that a downstream switch acknowledges a packet sent to it. If the downstream switch is in state (s,c) when a packet is sent to it and there are r packets arriving on other inputs that are contending for the $B-s$ available slots in the downstream switch's buffer, then the given packet is acknowledged with probability 1 if $B-s>r$ and with probability $(B-s)/(r+1)$ otherwise. The probability that r packets are contending is $\mathbf{B}(r,d-1,a_{i+1})$, so

$$\begin{aligned} q_i(j,c) &= \mathbf{B}(j,c,b_i) \\ b_i &= \sum_{s,c}\pi_{i+1}(s,c) \\ &\qquad \sum_{0\leq r\leq d-1}\mathbf{B}(r,d-1,a_{i+1})\min\left\{1,\frac{B-s}{r+1}\right\} \end{aligned}$$

For *global grant* flow control, the number of packets that enter a switch depend, not on the number initially stored, but rather on the number in the buffer after the packets that are leaving have done so. Hence, in the global flow control context, we define $p_i(j,x)$ to be the probability that j packets arrive and are retained at a switch given that x is the difference between the number of packets originally present and the number leaving. This requires changing the equation for λ_i.

$$\lambda_i((s_1,c_1),(s_2,c_2)) = \sum_{\substack{0 \le j_1 \le d \\ s_1+j_1-j_2=s_2}} p_i(j_1, s_1-j_2) q_i(j_2,c_1) R(s_1,c_1,j_1,j_2,c_2)$$

To compute b_i, we note that if a downstream switch is in state (s,c) and h packets leave, that a given upstream neighbor receives a grant with probability 1 if $B-s+h \ge d$ and with probability $(B-s+h)/d$ otherwise. Since the probability of h packets leaving is simply $q_{i+1}(h,c)$,

$$b_i = \sum_{s,c} \pi_{i+1}(s,c) \sum_{0 \le h \le d, c} q_{i+1}(h,c) \min\{1, (B-s+h)/d\}$$

All the other equations are identical to the local grant case.

In *global acknowledgement* flow control, once again, the number of packets that enter a switch depends on number stored in the switch after the packets that are leaving have done so.

$$\lambda_i((s_1,c_1),(s_2,c_2)) = \sum_{\substack{0 \le j_1 \le d \\ s_1+j_1-j_2=s_2}} p_i(j_1, s_1-j_2) q_i(j_2,c_1) R(s_1,c_1,j_1,j_2,c_2)$$

To compute b_i, we note that if a downstream switch is in state (s,c) when a packet is sent to it and r packets arrive on the other inputs while h packets leave, that the given upstream neighbor receives an acknowledgement with probability 1 if $B-s+h > r$ and with probability $(B-s+h)/(r+1)$ otherwise. Hence,

$$b_i = \sum_{s,c} \pi_{i+1}(s,c) \sum_{\substack{0 \le r \le d-1 \\ 0 \le h \le d,\, c}} \mathbf{B}(r, d-1, a_{i+1}) q_{i+1}(h,c) \min\left\{1, \frac{B-s+h}{r+1}\right\}$$

The remaining equations are identical to the local acknowledgement case.

The equations for acknowledgement flow control can be trivially modified to handle systems without flow control. In particular, we can model non-flow control systems in which the number of arriving packets accepted by a switch is limited to the number of available buffer slots at the start of the cycle, by using the equations for the local acknowledgement case, but letting $b_i = 1$. Similarly, we can model non-flow control systems in which arriving packets can move into buffer slots being vacated by departing packets, by using the equations for the global acknowledgement case, with $b_i = 1$. For systems without flow control, there are no correlations between stages and the accuracy for multistage networks is similar to that for single stage networks.

In summary, we have developed several methods of analyzing networks constructed from shared buffer switches. The bidimensional method yields excellent predictions of throughput for networks constructed from large switches. While we've only tested the interval method with two intervals, we find that even this case yields respectable accuracy. We have concentrated on maximum throughput, as this is the performance metric of most importance in switching system design. In high speed switching applications, so long as a network is operated below its maximum throughput, the queueing delay will be satisfactory and packet loss can be reduced to acceptable levels by engineering the input buffers appropriately. Methods that can accurately predict packet loss for networks preceded by input buffers could be useful, but in practice, conservative engineering rules for these buffers can be applied with little impact on system cost.

These methods do not model inter-stage correlations which become significant in large networks constructed from small switches. This is one possible direction for future research. Another is in extending the methods for nonuniform traffic as has been done for other types of analysis in [3]. Improving the computational performance of these methods would be very useful. The bidimensional method, in particular, can be very time-consuming. One way to improve the computational performance is to modify the basic iterative algorithm. The basic algorithm computes the transition probabilities λ_i from the current state probabilities (π_i) and then computes new state probabilities from the previous values by a single application of the balance equations for π_i. By applying the balance equations multiple times each time we compute values of λ_i, we can reduce the number of iterations significantly. We have found that this variation of the basic method can reduce the overall computation time by an order of magnitude.

We have not fully explored the possibilities of the interval method. We suspect that by using three or four intervals one could match the accuracy of the bidimensional method with a lot less computational effort. There remains the problem of selecting the intervals. Our approach of using simulation to "tune" the method is not fully satisfactory. The interval method can also be extended so that the intervals are dependent on the number of packets in a switch. For example, when just two intervals are used (the threshold method), we would expect better results if the threshold increases as the number of stored packets increases. Ideally, the probability associated with the two intervals for any given value of s should be approximately equal. This objective can be approximated by selecting the interval so as to divide the corresponding states of the underlying vector model equally. That is, for each value of s, we select a threshold t for which

$$\sum_{c<t} \sigma(s,c) \approx \sum_{c\geq t} \sigma(s,c)$$

The best approach is probably to adjust the threshold dynamically to make these probabilities as nearly equal as possible.

Acknowledgements. The authors wish to thank Achille Pattavina and Alberto Monterosso for stimulating discussions and Einir Valdimarsson for pointing out the error in the equation for Z in the interval method. Also, thanks to the referees for their thoughtful comments and corrections.

References

[1] Bianchi, Giuseppe and Jonathan S. Turner. "Improved Queueing Analysis of Shared Buffer Switching Networks," *Proceedings of Infocom 93*, 3/93.

[2] 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.

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

[4] Monterosso, Alberto, and Achille Pattavina. "Performance Analysis of Multistage Interconnection Networks with Shared Buffered Switch Elements for ATM Switching," *Proceedings of Infocom 92*, 5/92.

[5] 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.

[6] Turner, Jonathan S. "Queueing Analysis of Buffered Switching Networks," *Proceedings of the International Teletraffic Congress*, June, 1991.