ANG Progress Report - 12/92

The Advanced Networks Group

The Advanced Networks Group of the Computer and Communications Research Center is concerned with new communication technologies that can support a wide range of different communication applications in the context of large public networks and private. Fast packet or ATM networks promise a far more flexible communications infrastructure than is currently available. The Advanced Networks Group is in particular concerned with systems that are capable of supporting ubiquitous multicast communication, suitable for applications such as video distribution, voice/video teleconferencing and LAN interconnection. We have developed an experimental switching system supporting links operating at 100 Mb/s and have devised economical switch architectures that can support link speeds in excess of a gigabit per second and having total throughput exceeding a terabit per second.

Our work spans a variety of particular topics including switching system design and analysis, performance evaluation of switching systems and networks, multicast connection management, algorithms for multicast routing, buffer and bandwidth management in the presence of bursty traffic, internetworking of high speed networks, design of high speed transport protocols, multimedia applications and image and video compression. The design of computing platforms that are better suited for multi-media applications is a major topic of current interest. Our research program includes a strong experimental component, which includes experimental switching platforms as well as protocol and application software implementations. The experimental work is a crucial element of the overall research program, exposing detailed issues not apparent in higher level studies and providing a strong focus for the other activities.

Faculty
- Jonathan Turner
- Andreas Bovopoulos
- Paul Min
- Guru Parulkar

Doctoral Students
- Akira Arutaki
- Millind Buddhikot
- Charles Cranor
- Zubin Dittia
- Andy Fingerhut
- Fengmin Gong
- Saied Hosseini
- Seyed Mahdavian
- Nader Mirfakhraei
- Christos Papadopoulos
- Gopal Raman
- Ammar Rayes
- Hossein Saidi
- Einir Valdimarsson
- Peter Yan
- Ellen Witte Zegura

Masters Students
- Alex Chandra
- Xiaolin Bi
- Deepak Bhargava
- Apostolos Dailianas
- Rex Hill
- Anurag Maunder

Visitors
- Hossam Afifi (INRIA)
- Giusseppe Bianchi (CERFRIEL)

Our research program enjoys support from the following sponsors.

National Science Foundation
Ascom Timeplex
Bell Communications Research
Bell Northern Research
Digital Equipment Corporation
Goldstar Information and Communications
Italtel SIT
Nippon Electric Corporation
Nippon Telephone and Telegraph
SynOptics Communications

We thank all our sponsors for their collaboration and support.

Executive Summary

1992 has been another busy year for the Advanced Networks Group with good progress in several different areas. During the past year, ANG has produced over ten papers in refereed journal and conference proceedings, over fifteen university technical reports, three MS theses and one doctoral thesis. Washington University presented four papers at Infocom in 1992, more than any other single organization.

During this year, Paul Min has joined ANG. Paul is a professor of electrical engineering and has research activities in several areas including adaptive non-hierarchical routing, SONET network design and switching system analysis. His graduate students are Alex Chandra, Anurag Maunder, Ammar Rayes, Hossein Saidi and Peter Yan. Also, joining the group was Xiaolin Bi. Fengmin Gong completed his doctoral degree and accepted a research position at MCNC in North Carolina. Deepak Bharghava, Millind Buddhikot, Chuck Cranor, Zubin Dittia and Christos Papadopoulos all completed MS degrees this year. While Deepak has left for a position at BARRA, a software development company in Berkely, CA, the others will all be staying to complete doctoral degrees.

This has been an exciting year, as some of ANG's early work is now finding commercial application. SynOptics Communications announced in October its ongoing R&D relationship with ANG and Washington University's Applied Research Laboratory. This collaboration has led to a set of six chips that implement the broadcast packet switch architecture and are expected to be used in commercial products in the near future. The fall InterOp trade show featured several announcements of ATM products for local networks and many more are expected in the spring InterOp. Washington University and Ascom Timeplex also announced a new R&D relationship involving both ANG and ARL. This is a crucial next step in Washington University's plans to create an ATM campus network and we look forward to working with Ascom Timeplex over the next several years. ARL demonstrated our prototype switch, along with a variety of video and image applications at the Radiological Society of North America's annual conference and trade show in Chicago this year, attracting a lot of favorable attention. We have had dozens of groups visiting Washington University to see our demonstration network in action and the work has gotten a lot of notice in the trade press. During the last year, the ATM Forum has developed into a crucial standards body for the emerging local ATM market. ARL has played a leading role in helping the Forum develop multicast signaling standards that will take full advantage of ATM's ability to support flexible multicast communication.

Our plans for Project Zeus continue to develop. The Computer Science Department has been awarded a grant of $250,000 to create a multimedia ATM network in the department. While this falls short of the original request, it should still allow us to provide ATM connectivity to every faculty member's desktop workstation, a step that we feel is crucial to stimulating the development of novel, multimedia applications. It will also permit the University's campus computing organization to develop the necessary expertise to deploy, operate and maintain local ATM networks across the campus. Fiber is being put in place on both campuses to link various groups into the developing Zeus network. The departments of Radiology, Earth and Planetary Sciences, Biology and Architecture will be among the first to join Computer Science in the Zeus network.

The Advanced Networks Group's research agenda continues to include (1) design and analysis of switching systems, (2) internetworking and end-to-end protocol issues and (3) network performance and traffic engineering. We are now also putting a lot of attention into multimedia applications and the design of computing platforms that incorporate cell switching concepts internally, in order to facilitate handling of real-time applications. The short articles that follow cover a variety of specific topics. More detailed accounts of these activities appear in the appendix.

Our Industrial Partnership Program currently has eight members. Acom Timeplex joined in September and Goldstar Information and Communications, one of Korea's most important electronics companies joined at the end of 1992. We welcome them both and look forward to a productive relationship. The annual funding provided by the IPP is now $400,000 and our NSF funding is about $150,000. While our funding continues to be adequate, the need to develop new projects and funding sources continues. We have, together with the Applied Research Laboratory, applied for a planning grant from the Corporation for National Research Initiatives to develop a detailed proposal for a gigabit testbed. In addition, we have applied for a grant from the Defense Advanced Research Projects Agency for gigabit technology development. While both of these are still pending, they have survived the initial screening stage giving us some cause for optimism.

Refinements to the Broadcast Packet Switch

Jonathan Turner

During the last year we have made several refinements to the broadcast packet switch architecture in order to improve its performance and cost-effectiveness in large configurations. One of these refinements centers around the possibility of congestion arising in a routing network due to a poor assignment of multicast copies to Broadcast Translation Circuits. This is illustrated in Figure 1, which shows a broadcast packet switch constructed with four port switch elements and carrying one point-to-point connection from input 5 to output 3 with a data rate equal to 48% of the internal switch data rate, and one multicast connection from input 11 to outputs 2,5,7,12 and 15 with a data rate of 64% of the switch data rate. The numbers adjacent to selected links in the networks show the load on those links. Note that in the first stage of the copy network, traffic distribution spreads the load as evenly as possible. The numbers shown within the Broadcast Translation Circuits (BTC), at the center of the figure, give the destinations for the multicast cells passing through each specific BTC. A different assignment of destinations to BTCs can result in very different loading in the routing network. In particular, if the cells for outputs 15 and 5 are interchanged, some of the routing network link loads increase from 32% to 64%. There are connection configurations that can make the load on the internal links equal to $\sqrt{n}$ times the load on the external links, where n is the size of the networks.

In the original architecture, this link overloading was avoided by including distribution stages in the routing network to spread the traffic evenly. While this approach solves the problem, it increases the cost of the routing network substantially. An alternative approach is to assign the copies of cells to multicast connections intelligently, in order to avoid internal link overloading. We have developed an algorithm for assigning copies which relies on certain properties of the routing network's topology to minimize traffic imbalances within the routing network. We describe it here for a connection with fanout f equal to a power of two, although it can be readily extended to arbitrary fanouts.

First we note that for any given copy of a multicast cell, the alternate paths that the copy can take on its way to the output pass through BTCs that are n/f apart from one another. This means that each individual copy is distributed in the ideal fashion. The distinct copies of a particular original multicast cell appear at f consecutive BTCs. The destination assignment algorithm seeks to maintain disjoint paths between these different copies to the largest extent possible. This can be done as follows.

- Let $R(x, j)$ be the number obtained by representing x as a base d number with j digits, then reversing the digits (here, d is the dimension of the switches used in the routing network).
- Let $y_0, \ldots, y_{f-1}$ be the routing network outputs that are to receive copies of the multicast cell, where $y_i < y_{i+1}$ for all i.
- Within each group of f BTCs where the different copies of a multicast original can appear, assign y_i to BTC $R(i, \lceil \log_d f \rceil)$.

We can show that for $d = 2$ and fanouts that are powers of 2, this algorithm maintains a load of $\leq 2\beta$ on each routing network link and conjecture that in general, the load is bounded by $\beta + \delta$, where β is the maximum load on the external links supported by the switch and δ/β is the fraction of the overall traffic due to multicast traffic.

The second area we have sought improvements in the broadcast packet switch architecture is in reducing the amount of memory required in the BTCs. In the original architecture, because the

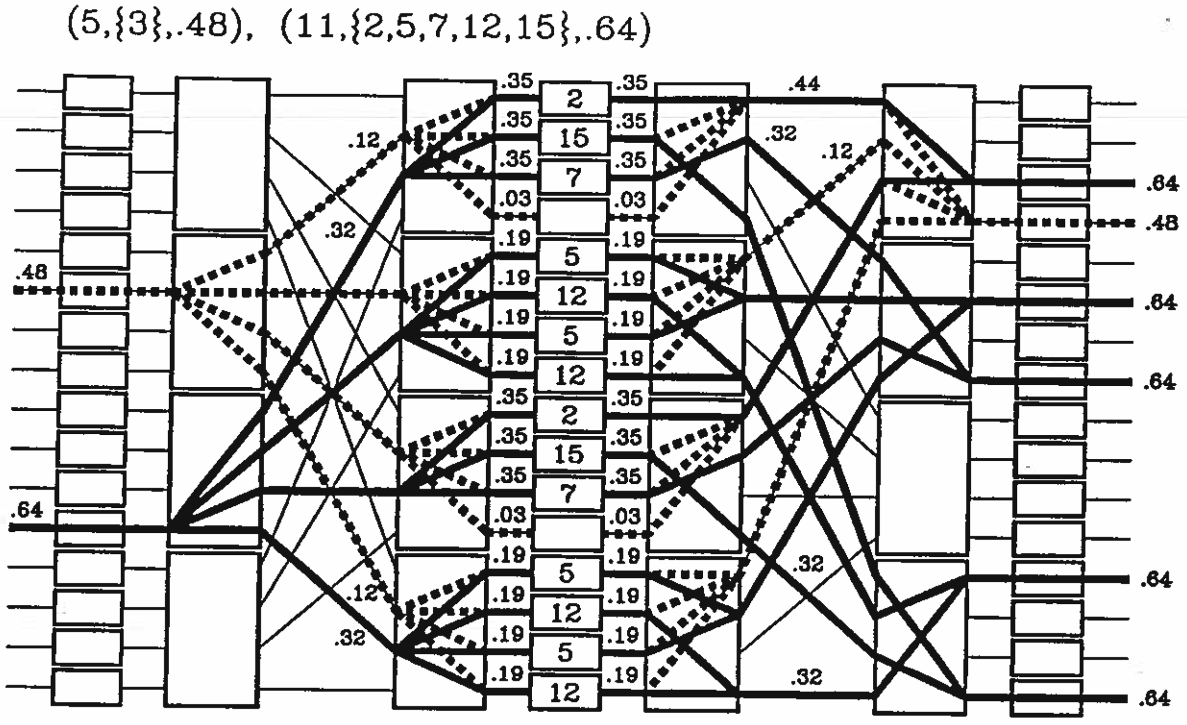

Figure 1: Loading in Broadcast Packet Switch

copy network distributes load when not copying, copies of a small fanout connection can appear at any copy network output and hence require an entry in every BTC. If we route copies of cells to a particular set of BTCs instead of distributing them randomly, a given connection consumes memory only in those BTCs. This allows corresponding entries in different BTCs to be used for different connections.

Let $f_0 = 1 < f_1 < f_2 < \cdots < f_r = n$. We say a connection belongs to fanout class i if its fanout exceeds f_{i-1} but not f_i. If the copy network routes connections in class i to a specified range of f_i outputs, a given broadcast channel number can be used for n/f_i different class i connections. If i maximizes $f_i/(f_{i-1}+1)$, all connections have fanout $f_{i-1}+1$ in the worst-case. In this case, we can have $mn/(f_{i-1}+1)$ different connections and we use f_i BTC entries for each one. To handle the worst-case then, $mn(f_i/f_{i-1})$ total entries are enough. (In the original architecture, there is only one class and we need $mn^2/2$ BTC entries to handle the worst-case situation.) If f_i/f_{i-1} is the same for all i, then the number of classes is $\log_{(f_i/f_{i-1})} n$.

Thus, we can cut the amount of memory needed to $2mn$, using $\lg n$ levels. Since, mn words are required in any case to describe an arbitrary collection of multicast connections mn total destinations, this is approaching the optimum. Unfortunately, fanout f connections now require consecutive sets of BTCs with sufficient unused bandwidth. To prevent blocking at the BTCs, we need an additional speed advantage proportional to the number of fanout classes.

The trade-off between the number of BTC entries and the network complexity (resulting from the increased speed advantage) is illustrated for several examples in the following table.

r	BTC entries	network complexity
1	$O(mn^2)$	$O(n \log_d n)$
2	$O(mn^{3/2})$	$O(n \log_d n)$
$\lg\lg n$	$O(mn^{1+1/\lg\lg n})$	$O(n(\log_d n)(\lg\lg n))$
$\lg n$	$O(mn)$	$O(n(\log_d n)(\lg n))$

For $n = 2^{16}$, $\lg\lg n$ is 4 and $n^{1/\lg\lg n}$ is 16.

In practice most multicast connections have small fanout and we can get better performance in practice by designing for this case. So how much memory is needed to handle the 'expected case' rather than the worst-case? To answer this we need a probability distribution for the frequency of connections with different fanouts. There are two natural distributions that are worth considering. First, assume that for any i, the number of connections with fanout in the interval $[2^{i-1}+1, 2^i]$ is twice the number in the interval $[2^i+1, 2^{i+1}]$. The justification for this is that because the connections in the larger fanout class use roughly twice the output bandwidth of connections in the other class, there is "only room" for half as many of them.

Because smaller fanouts are more common, we can get better memory utilization by making the boundaries defining the small fanout classes closer together at the expense of wider spacing for the larger fanout classes. For example, if fanout classes are defined by the intervals $\left[2^{2^{i-1}}+1, 2^{2^i}\right]$, we can handle the expected case with $O(mn)$ table entries and network complexity of $O(m(\log n)(\lg\lg n))$.

Another fanout distribution is obtained by assuming that each of the mn output VCIs selects with equal probability from among the mn input VCIs (where each input VCI can be chosen multiple times). It's straightforward to show that under this assumption, when mn is not too small, the expected number of connections with fanout $> \lg mn$ is less than 1. Hence, two fanout classes with a boundary at $\lg mn$ requires $O(mn \log mn)$ BTC table entries with a network complexity that is $O(n \log n)$. By going to more levels, we can reduce the number of BTC entries required. In particular, with $\lg\lg\lg mn$ levels we need $O(mn)$ BTC entries and network complexity $O(n(\log n)(\lg\lg\lg mn))$

In practice, we expect switching networks to support a few hundred 'public service' broadcast sources with large fanouts and the remainder will be small. It seems likely that the second of the fanout distributions described could be a good model for the small fanout connections, suggesting that two fanout classes are indeed sufficient. Under this assumption, it can be shown that the worst-case load in the copy network is at most $\beta + \delta' + B$, where δ'/β is the fraction of the total traffic due to multicast connections whose fanouts are not powers of 2 and B is the maximum rate of any connection. Multiplexing several copy network outputs through a common BTC can improve memory efficiency of small fanout connections further, reducing the memory requirement by up to the multiplexing factor.

Advances in Multipoint Switching

Jonathan Turner and Ellen Witte Zegura

In [31], Jordan and Masson derived conditions under which a Clos network is nonblocking for multipoint circuit-switched connections. While the resulting networks are asymptotically cheaper (in terms of crosspoint count) than a crossbar, they are generally too complex to be of much practical interest. In [59], Yang and Masson showed that the number of crosspoints can be drastically reduced if one drops the requirement that existing connections can be augmented without rearrangement. Multipoint networks that are nonblocking for *new connections*, but may block when augmenting existing connections, are called *nearly nonblocking* [38]. Because augmentations can always be handled by rearranging the affected connection (without disturbing anything else), nearly nonblocking networks can be of practical value. The key idea that Yang and Masson use to reduce the crosspoint count is to restrict the branching of multipoint connections in the first half of the network to prevent excessive early branching from blocking new connection requests.

We have generalized Yang and Masson's result to the multirate environment [58], in which different connections can be multiplexed on a switch's internal data paths, so long as the total data rate of the connections does not exceed the internal data path speed. This allows Yang and Masson's ideas to be directly applied to ATM switching systems being developed by several companies, including NEC, Fujitsu and France Telecom. In the process, we have reformulated Yang and Masson's proof to make it more transparent and reveal the central role played by an embedded set covering problem.

The key ideas can be understood by considering the three stage case. To set up a new connection from a given input x to a set of outputs Y, one starts by identifying the middle stage switches that are accessible from x (that is, the paths between them and x have sufficient unused bandwidth to accommodate the new connection). For each of these middle stage switches, we identify the set of accessible third stage switches connected to outputs in Y. To set up the connection, we need to select middle stage switches whose sets of accessible third stage switches *cover* the set of third stage switches with outputs in Y. To avoid excessive branching in the first stage switch, we want to do this using as few middle stage switches as possible. When the Clos network has sufficient expansion in the first and third stage switches, each third stage switch is accessible from a large number of second stage switches, making it possible to find a suitable covering using a simple greedy covering algorithm. It is shown in [58] that the three stage Clos network with n inputs and outputs, $d \times m$ crossbars in the first stage and $m \times d$ crossbars in the last stage is nearly nonblocking when

$$m > \frac{\beta}{1-\beta}(d-1)(f+(n/d)^{1/f})$$

where f is a restriction on the fanout allowed in the first stage switches and $1/\beta$ is the speed advantage of the network's data paths, relative to the external links. By selecting a value of f to minimize the above expression, we can reduce the requirement on m. Alternatively, if we prefer to fix m we can rewrite the inequality as

$$1/\beta > 1 + \frac{d-1}{m}(f+(n/d)^{1/f})$$

and select f to minimize the required speed advantage. For example, with $n = 256$, $m = d = 16$, and $f = 3$ we find that a speed advantage of 6.5 is sufficient, as opposed to 16 if we do not restrict the first stage fanout. The result is easily extended to asymmetric networks and with more difficulty to networks with an arbitrary number of stages. See [58] for details.

In [38], it was shown that a nearly nonblocking network can also be constructed by cascading a pair of Beneš networks, with branching restricted to the second of the two networks. The resulting network requires the same speed advantage as the component Beneš networks require to be nonblocking for point-to-point traffic. For example, a 256 port network constructed from two three-stage Beneš networks with 16 port switches requires a speed advantage of three. This network has five stages, since we can combine the last stage of the first Beneš network and the first stage of the second one. We have recently discovered that we can eliminate one more stage from this network, while maintaining the nearly nonblocking property at the cost of some increase in the required speed advantage. The cost of the resulting network turns out to be only slightly lower than the original, assumng cost is proportional to speed advantage, but could be advantageous in situations where the number of stages required by the original network is problematical.

To make the four stage network nearly nonblocking, we need a speed advantage that is large enough to ensure that from any input x, there is some accessible second stage switch from which all the fourth stage switches are accessible. We call a second stage switch *heavily loaded* if the total traffic on its output ports exceeds $(1+\epsilon)\beta d$ where $d=\sqrt{n}$ is the size of the switches and ϵ is a constant to be determined. Because the total traffic exiting all the second stage switches is at most βn, the number of heavily loaded switches is limited. The network is nearly nonblocking if the speed advantage is large enough to ensure that every first stage switch can reach some lightly loaded second stage switch and that every lightly loaded second stage switch can reach all of the fourth stage switches. If we restrict branching of connections to the last three stages, this occurs when $1/\beta \geq 3+\epsilon$ and $1+\epsilon \geq (1-\beta)/(1-2\beta)$. By combining these inequalities and solving for β we find that a speed advantage of $2/(1-\sqrt{1/5}) \approx 3.62$ is enough to make the network nearly nonblocking.

In the general case, a nearly nonblocking network is constructed by cascading a delta network and a Beneš network with branching occurring only in the Beneš network. For n port networks constructed from d port switches, the resulting network has $3k-2$ stages (assuming we merge the last stage of the delta network with the first stage of the Beneš network), where $k=\log_d n$. For comparison, the nearly nonblocking Beneš network with fanout restriction requires $2k-1$ stages and the pair of cascaded Beneš networks requires $4k-3$ stages. The required speed advantage drops with the number of stages but the $3k-2$ stage network gives the lowest cost, assuming cost is proportional to speed advantage.

Analysis of Blocking in Multipoint Networks

Ellen Witte Zegura

Analysis of blocking probability in circuit switching networks with point-to-point connections has been studied for years and is fairly well-understood. In many situations, a simple model such as that proposed by Lee [34] in 1955 is adequate. More precise results can be obtained using an exact analysis such as the one developed by Pippenger [46] for uniform series-parallel networks. While these analyses do not capture dynamic traffic effects or the impact of routing algorithms on blocking probability, they are useful for understanding the impact on blocking of various architectural choices and for making comparisons among competing configurations.

During the last year, we have formulated a general model for analyzing blocking probability in circuit switching networks that allows us to derive approximate blocking probabilities for series-parallel networks. The analysis is exact for series networks (networks like the delta network which have just one path between any input and output), but is approximate for the general series-parallel case. This appears to be the first attempt to analyze blocking probability for multipoint connections.

Figure 2 illustrates the series and parallel operators for constructing networks. The series operator constructs a network $N_1 \times N_2$ from two component networks and the parallel operator constructs a network $N_1 \bowtie N_2 \bowtie N_3$ from three component networks. Many practically interesting networks can be constructed by repeated applications of the series and parallel operators. Such networks are called *series-parallel* networks.

We can determine the blocking probability in series-parallel networks, by showing how, given the blocking probability for the component subnetworks, we can calculate the blocking probability for networks using the series and parallel operators. Pippenger [46] did this for point-to-point connections. His "traffic model" is based on two key assumptions; first, that each output of the overall network is busy with some fixed probability p and that these probabilities are independent; second, that if a component crossbar in the network has r busy outputs, all possible ways that these can be connected to the crossbar's inputs are equally likely. A consequence of these assumptions is that the busy-idle status of two links in the same stage of any uniform series-parallel network are independent.

For multipoint connections, it is natural to make essentially the same assumptions, but now in each crossbar, multiple outputs can be connected to the same input. We think of each busy output of a crossbar selecting at random from among the crossbar's inputs, without regard to whether they have been previously selected or not. Unfortunately, under these assumptions the busy-idle status of two links in the same stage are no longer independent of one another. This can be shown by a simple example. Consider a three stage Beneš network constructed from 2×2 crossbars and assume that all the outputs are busy ($p = 1$). If one input of a third stage switch is idle, then the other must be busy so their status is clearly not independent. Unfortunately, there appears no way to capture the dependencies among links short of a complete enumeration over the states of the network, so we have proceeded to approximate the blocking probability by *assuming* that the states of different links in the same stage are independent, then exploring the effect of this assumption on the accuracy of the resulting estimate by comparison with simulation.

The analysis is similar in spirit to Pippenger's analysis for the point-to-point case. We show how the series and parallel constructions affect the blocking probability, giving a straightforward recursive procedure for computing blocking probability estimates. We have also generalized the model to permit more

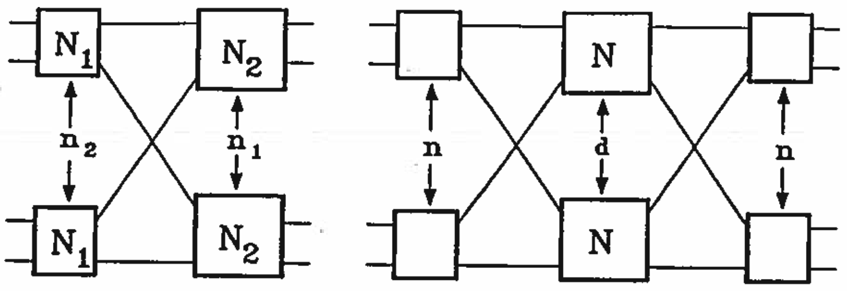

Figure 2: Series and Parallel Construction Operators

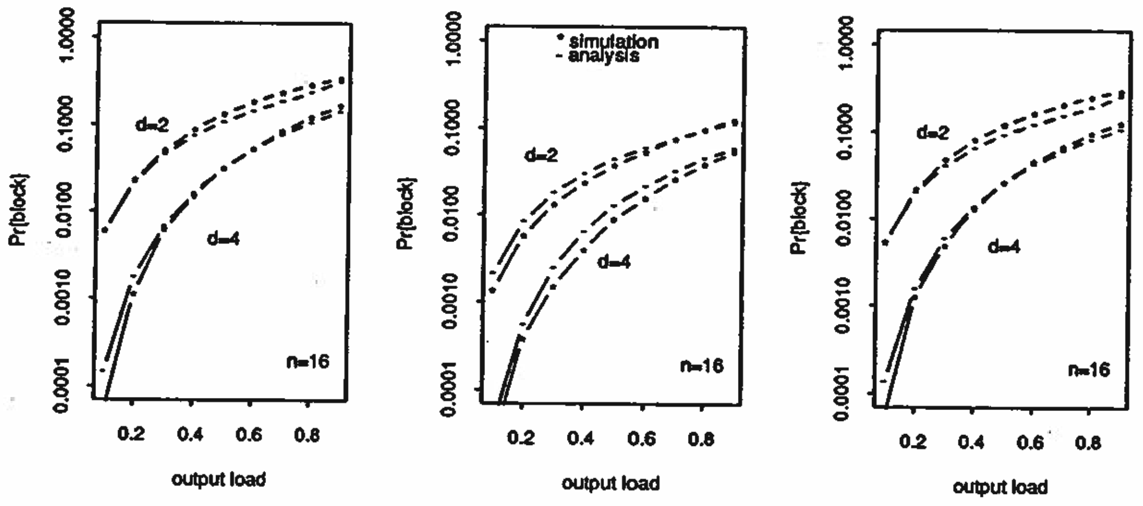

Figure 3: Multipoint Blocking Probability in Beneš Networks

control over the way branching occurs in networks. This involves the introduction of a fanout function $f(i)$. The fanout function affects the way branching occurs at a crossbar. We think of the crossbar's busy outputs being sequentially connected to inputs, selecting from among the busy inputs with probability $f(i)$, where i is the number of busy inputs at the time a selection is made. Selecting $f(i) = i/d$ gives the basic model described above, $f(i) = 0$ gives Pippenger's model, and $f(i) = \alpha i/d$ for $0 < \alpha < 1$ gives a family of models with varying amounts of branching.

We have evaluated the blocking analysis by comparing it to simulation for Beneš networks of various size. Some typical results are shown in Figure 3. These results are for Beneš networks with 16 ports, constructed from crossbars with $d = 2$ and $d = 4$. Networks with small crossbars are particularly difficult, as the independence assumption introduces the greatest error in these cases. The left hand plot shows the probability of blocking when connecting to an idle input, the middle plot shows the probability of blocking when connecting to a busy input and the right hand plot shows the probability of blocking when connecting to an arbitrary input. We note that the analysis tracks the simulation results closely. The crossovers reflect the fact that the approximations introduced by the independence assumption are in some cases optimistic and in other cases pessimistic.

Improved Analysis of Shared Buffered Networks

Giuseppe Bianchi and Jonathan Turner

In a widely cited paper [30], 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 [53] extended Jenq's method to switching systems constructed from switches with an arbitrary number of inputs and an arbitrary number of buffer slots.

Turner [56] developed a similar method which can be applied to switching networks with shared buffering. Recently, Pattavina and Monterosso [39] 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. We have recently developed 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 method.

The analysis is for delta networks (see [56]) 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. We have obtained results for several different methods of flow control; local and global grant flow control as well as local and global acknowledgement flow control.

Pattavina and Monterosso refer to Turner's method as the *scalar method*, since it represents the state of a shared buffer switch, by a single number giving the number of packets that are stored in the switch's buffer. This leads to a computationally efficient model, but does require the introduction of an assumption concerning the distribution of addresses of the stored packets. Pattavina and Monterosso, on the other hand, maintain a state vector that records the number of packets addressed to each output, leading to a much a larger state that precludes the application of their method to networks constructed from large switches (say larger than 4×4).

The scalar method of queueing analysis computes the number of outputs for which a switch has packets (called the *active outputs*) 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 4 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 are independent. This can be interpreted as assigning probabilites 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

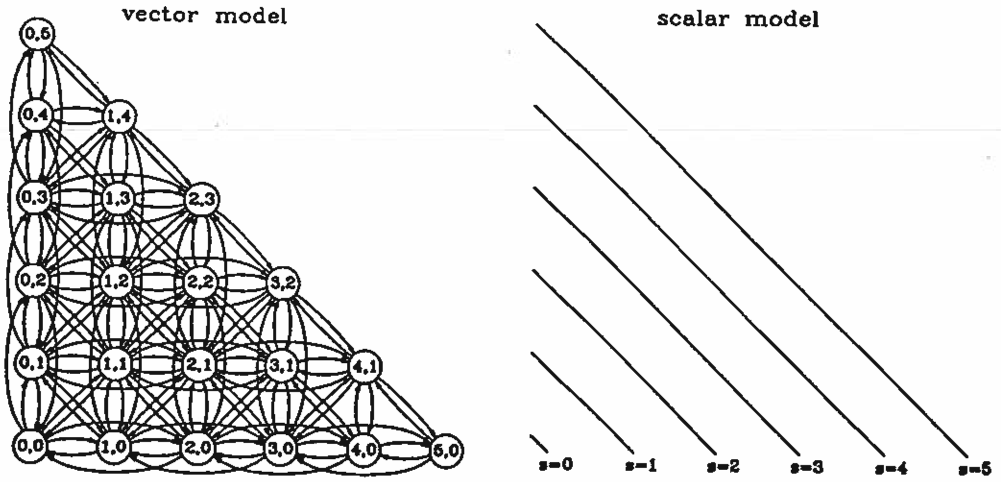

Figure 4: Comparison of Vector and Scalar Methods for Binary Switches

assigned to state (i,j) is $\binom{s}{i}(1/2)^i$ (where $s = i + j$).

Closer study reveals that the distribution of the probabilites of states along a diagonal are more closely approximated by a uniform distribution. This makes intuitive sense if we consider the steady-state situation for a heavily loaded switch. In this case, one expects that the state of the 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 accuracy of the uniform scalar method turns out to be only slightly better than that of the original scalar method, although the uniform method is generally conservative whereas the original is generally optimistic. However, the uniform assumption plays a role in the more sophisticated methods we turn to next. The first of these, called the *bidimensional method*, expands the state of a switch to include a second variable giving the number of active outputs that the switch has. One still must make an assumption regarding packet destinations in order to compute the state transition probabilities, but the resulting method is highly accurate, albeit at a higher computational cost.

The final method we have studied is called the *interval method*. In this method, the state consists of the number of packets in the buffer together with an approximate representation of the number of active outputs. More specifically, we keep track of which of several disjoint intervals contains the number of active outputs. In the simplest case, there are just two intervals and the method tracks whether the number of active outputs is above or below a given threshold. Surprisingly, this threshold method can yield results that are competitive with the bidimensional method in accuracy, while being nearly as fast as the scalar methods. The one drawback is that the accuracy is highly dependent on the choice of threshold. We have used simulation to determine the best choice of threshold for a variety of different switches, but would prefer a less ad hoc method of determining the best choice. We are considering a variant of the threshold method in which the threshold is determined adaptively as the analysis is carried out. The threshold would be adjusted so as to make the probability of being above the threshold roughly equal to the probability of being below the threshold. The same idea could be applied to the more general interval method.

More details on this work can be found in [3].

Near Optimal Configuration of ATM Networks

Andy Fingerhut and Jonathan Turner

The problem of configuring low cost communication networks that satisfy a given set of traffic requirements has been studied in depth, in the context of telephone and conventional data networks. The choices that network managers make when purchasing and deploying communication equipment can have a profound effect on the cost of the network and the performance delivered to the network's users. The emergence of ATM networks in the last several years has introduced two important new considerations in network design. First, because ATM networks offer the ability to configure multipoint virtual circuits, network designers need to take this into account when selecting an appropriate network configuration. Second, because ATM networks are designed to handle arbitrary mixtures of traffic types, there is little usable statistical information that a network designer can bring to bear. We have developed a framework in which to address ATM network design problems and have obtained a number of useful results that we plan to incorporate into a network design tool.

There is a variety of ways one can formulate the network design problem. Different formulations focus attention on different aspects of the problem, bringing certain features to the fore, while suppressing others. For ATM network design, we prefer formulations in which we design networks that can handle an arbitrary set of user requests meeting specified constraints. That is, with respect to network traffic we take a worst-case point of view, rather than a probabilistic one. This is motivated by our observation that in ATM networks, the diversity of the applications and our limited understanding of how they will be used makes it difficult to obtain usable statistcal predictors.

One formulation of the problem is as follows. We are given a complete graph, $G = (V, E)$, where each vertex represents a switching system. For each vertex pair (u, v), we have a value $c(u, v)$ that represents the *cost per unit capacity* for installing a link from u to v. For each vertex u, we are also given a *source capacity*, $\alpha(u)$ and a *sink capacity*, $\omega(u)$ These give upper bounds on the total traffic that can originate or terminate at each vertex. Our objective is to assign a capacity $\gamma(u, v)$ to every vertex pair (u, v) that minimizes the cost of the network (defined as $\sum_{(u,v)} c(u, v)\gamma(u, v)$), while supporting any traffic configuration permitted by the source and sink capacities. A set of capacities that yields a network supporting all possible traffic patterns is termed a *nonblocking network*.

The source and sink capacities of a switch can be used to represent the total raw bandwidth associated with the terminal interfaces that the switch supports. Alternatively, we might specify lower values for the source and sink capacities, based on estimated upper bounds on the amount of traffic that might originate or terminate at a switch. While detailed statistics for ATM networks are hard to obtain, we believe that such gross statistics as the total traffic entering or leaving the network from a given switch can be estimated with reasonable accuracy.

Given this formulation of the network design problem, we can find a lower bound on the cost of an optimal nonblocking network by solving a minimum cost maximum flow problem. The solution of this problem is a worst-case traffic configuration which must be supported by any nonblocking network. To determine how close we can come to the lower bound, we have carried out a series of random experiments in which the vertices of the network graph $G = (V, E)$ are placed at random in a unit square and the pairwise Euclidean distances are used as costs. For random graphs with up to 100 vertices, we computed the lower bound on the optimal nonblocking network cost and then constructed a nonblocking network consisting of a single central vertex and direct links to all the others. For this configuration, the required link capacities are easy to compute. We iterated over all possible choices of the central vertex and selected the lowest cost alternative. This simple approach gives remarkably good results, typically yielding networks with costs that are within a few percent of the lower bound. We have also proved analytically, that for problem instances created randomly on a unit disk, that as the number of vertices gets large, the ratio of the star network cost to the lower bound approaches one with high probability.

We can extend this simple formulation in two important ways, in order to allow a more flexible specification of the traffic. First, we introduce pairwise traffic constraints, $\mu(u,v)$ for switches u and v, allowing a network designer to specify upper bounds on the amount of traffic originating at one switch and terminating at another. This allows the designer to express the fact that certain pairs of switches have only limited need to communicate. We also allow the network designer to specify disjoint subsets $V_1, \ldots, V_r$ of the switches and specify source and sink capacities for the subsets. Such a source capacity limits the amount of traffic that can originate within the subset and terminate outside it. Similarly for the sink capacities. This allows designers to express the fact that clusters of switches may have lots of internal traffic but limited external traffic, reflecting natural "communities of interest." This idea can be extended to aribtrary cluster hierarchies and we can again compute a lower bound on the cost of an optimal nonblocking network satisfying the specified constraints using minimum cost maximum flow techniques. Again, we have found that simple network designs that directly reflect these hierarchies and are easy to construct can closely approximate the lower bound. Also, because these networks are tree-structured, they admit only one path between any pair of vertices, making them nonblocking for arbitrary multipoint traffic, as well as point-to-point traffic.

Our current formulations of the network design problem leave out two important considerations. First, they treat the collected link capacity between two switches as a single large link, rather than as multiple small links, which is a more realistic point of view. To capture this in our models, we need to introduce the notion of a single link capacity and the notion of a maximum data rate for a single virtual circuit. The cost of a nonblocking network under these conditions depends critically on the ratio of the link rate to the maximum connection rate. If this ratio is one, the nonblocking network cost can be infinite, since a link carrying an infinitesimal amount of traffic can block a new maximum rate connection. If the ratio is two (the links are twice as fast as the fastest connection), a link must be at least half full in order to block a connection. In this case the worst-case bandwidth fragmentation requires at most a doubling of the network cost over that determined in our current problem formulation.

Our problem formulations also fail to account for constraints on the amount of transit traffic a switch can support. Switches typically have a limited number of ports and the simple tree-structured networks which we have shown can closely approximate the optimal nonblocking network, can require more total throughput than real switches may be able to provide. One approach to addressing this problem is to devise general methods for decomposing large switches in a network design into multiple smaller switches, with only a small increase in cost. If this can be done in a general way, it could yield a general method for converting an idealized network design into one that can realized from real switches, while retaining the low cost of the original.

Performance Evaluation and Visualization System

Einir Valdimarsson

The evaluation of switching systems is a complex task, because there are many different components which interact in subtle and unexpected ways. Simulation is an essential tool for deriving insight into the way systems perform, as it allows the designer to reproduce the precise conditions under which a system will be used, while allowing him or her to observe the system's behavior at either a macroscopic or microscopic level.

Unfortunately, the design of effective simulators for switching systems is a time-consuming chore, since each switching system has its own set of characteristics and idiosyncrasies that must be captured, and since careful programming is necessary to achieve acceptable performance for system configurations of practical interest. To address this difficulty, we have designed and implemented a general purpose switching system performance evaluation tool. The tool includes the following features.

- A graphical user interface that can be used to create switching systems with arbitrary interconnection topologies.
- A variety of switching elements, including buffered switching elements with or without flow control, various buffer organizations and queueing disciplines. Sorting elements and other unbuffered elements are also supported.
- Various other components, including multi-channel bursty traffic sources, buffers and lookup tables.
- The ability to run the simulation forward by single-stepping or many steps at a time.
- Visualization of the simulation, both through the animation of packets moving through the system and real-time display of various user-specified performance curves.

While the system continues to undergo refinements to increase it's generality and improve both the performance and user-interface, it has become sufficiently stable that it has been used to support a graduate course in switching system design and analysis (CS 577) and has been exported to a few users outside the university. We invite others to use it as well and can make it available via ftp for those who wish to try it out.

The challenge in designing a tool like this is to find the right balance between the competing objectives of generality and performance. For example, because the system allows users to specify networks with arbitrary topology, the question of how to route packets to the proper outputs becomes an issue. Real switching systems can use simple routing methods because they depend on a knowledge of the topology, but this approach does not really work in a general simulation environment. We have addressed this so far, by supplying a variety of different switching element types and allowing the default routing to be modified by direct manipulation of a set of routing tables. A more general approach would allow users to specify the routing algorithm directly, making it possible for them to use their knowledge of the context in which a switching element is used to permit a simple description. The user's algorithmic description could be compiled and dynamically linked into a running simulation to yield an implementation that is easy to use, completely general and fast in execution. The same kind of issue arises in switching elements that replicate packets belonging to multicast connections.

Another example of the trade-off between generality and performance arises in the collection of performance statistics. We want to allow users to plot a variety of different performance parameters. So far, we are addressing this need by collecting many different basic statistics all the time and then using these statistics to generate a variety of different kinds of plots, at the user's request. Ideally, the system would collect only those statistics needed to generate the plots that a user has specified. However, making run-time decisions about what statistics to collect leads to poor performance, Again, the best alternative seems to be to generate code that will collect the desired statistics, in response to the user's plot specification, then compile and link that code into the running program. This approach may be incorporated into future versions of the system.

Another direction we are pursuing is the incorporation of analytical methods into the simulation tool. There are a variety of methods that have been developed for the steady-state analysis of multistage switching networks. These can be used to obtain steady-state distributions of various performance parameters, including buffer occupancies, delay, packet loss and so forth, under both uniform and non-uniform traffic conditions. To facilitate the use of such methods, we have generalized methods designed for uniform traffic to the non-uniform traffic situation and have devised the first queueing analysis of networks that replicate packets for multicast communication. While these methods have not been used for this purpose in the past, they can also provide insight into transient traffic phenomena, particularly in the context of a tool supporting real-time visualization. It may also be possible to switch back and forth between simulation and analytical modes. Analysis can be used to generate a steady state distribution, from which an intial state can be selected for simulation. Or, steady state analysis can be initiated from a state obtained via simulation, in order to obtain an understanding of the possible ways in which the system might evolve from that point. Such a combination of simulation and analysis promises to open up new ways of understanding the performance of complex systems.

Cost-Based State Dependent Routing

M.A. Rayes and P.S. Min

The routing of traffic is one of the key functions in the management of telecommunication networks. It consists of the decision rules the network uses upon the arrival of a call. The essential features of these decision rules include how to decide when an arriving call should be rejected, how to determine the set of available paths for an incoming call, and finally how to select one out of this set to transmit the call over. Addressing these components in a routing policy requires consideration of several performance factors such as network blocking, switching complexity, robustness to network element failures (e.g, switch or link failure), robustness to design errors (e.g., under-engineered capacity), performance of multi-class traffic, information exchange protocols, and ease of dimensioning.

Currently, a significant majority of the routing policies in circuit switched networks are Fixed Alternate Routing (FAR) schemes. These routing schemes use the available set of routes in some predetermined sequence. These schemes have been in place for some time in public telephone networks and were implemented to take into account the constraints imposed by the processing capacity of switching entities (e.g., eletromechanical switches or old analog switches), as well as the limited sophistication in information gathering techniques of network information such as occupancy, or exogenous traffic intensity. However, the widespread incorporation of intelligent switches (e.g., 5ESS or DMS100) and digital transmission combined with common channel signaling has made possible more sophisticated routing schemes which are real-time adaptive and based on network state information.

Dynamically controlled routing (DCR), a feature of Northern Telecom's dynamic network controller system is planned for the Trans-Canadian network, Dynamic Alternate Routing (DAR) was developed by British Telecom which plans to adopt it, and Real-Time Network Routing (RTNR) is a state-dependent routing scheme with quantized network state information which has been implemented by AT&T. Bellcore has developed State-Dependent Routing (SDR) and is currently conducting field-trials for the purpose of demonstrating its efficacy for Local Area Transport Areas (LATA's).

The nature of the routing decisions in state-dependent routing schemes is considerably more complicated than that in FAR. In FAR, there is a predetermined sequence of paths. The next path in the sequence is chosen only if the current path is determined to be in a blocked state. Thus, this decision is based only upon the state of the *single*, currently considered path. On the other hand, the decision to route a call in state-dependent routing schemes depends on the state of *multiple* paths. This added complexity along with the random nature of the demand and the multiple interactions that occur in the topologically rich structures of networks makes analysis particularly difficult even for moderately sized networks.

On the other hand, state-dependent routing schemes seem to offer much improvement in terms of performance. They improve network blocking in many of the scenarios we studied, and such improvement is maintained even as the size of the network increases. They provide the ability to withstand mismatches between the network capacity and the level of offered traffic, and provide robustness in the network performance to changing traffic. The survivability and self-healing properties of the network in the presence of failures are seen to be natural advantages of the state-dependent routing schemes. We also observed that multi-class traffic with differing grade of service requirements could be easily accommodated. These perceived advantages of state-dependent routing schemes have spurred the increased interest.

However, despite the clear trend of technology towards state-dependent routing, the level of knowledge about the complex dynamic behaviors that these schemes generate in the network appears very limited. Much of the published literature on the subject of state-dependent routing consists of discussions of the feasibility of implementation and the development of reasonable (but not necessarily accurate) performance evaluation algorithms to compute network blocking. We note that while the reduced network blocking is an important benefit, it does not characterize the entire benefit of the state-dependent routing schemes. For example, other features of the schemes such as robustness to component failures, robustness to under-engineered network capacity, additional processing complexity required at switching nodes, complex network management function such as gathering traffic information and state occupancy, survivability, tandem switch loading due to increased multi-link calls, coping with (possibly) coexisting FAR schemes, just to mention a few, are equally important and need comprehensive study in order to reflect the overall benefits of these state-dependent schemes.

The main thrust behind our work is to provide a thorough understanding of many issues associated with a certain class of state-dependent routing schemes in circuit-switched networks, which is illustrated below.

In practice, most state-dependent routing schemes specify the routing policies in terms of *cost functions* which are, in turn, functions of the state of the network; at state $\underline{s}(t)$ under a certain routing policy, a routing decision is made by evaluating the cost function for each of the feasible actions in $A_{\underline{s}(t)}$. A network state $\underline{s}(t)$ is defined as $([i,j](t), x_1(t), x_2(t), x_3(t), ..x_K(t))$ where $[i,j](t)$ denotes the origin-destination pair corresponding to the last call arrival before time t and where $x_k(t)$ denotes the number of busy trunks in link k at time t.

The cost functions chosen for this purpose are usually based on some routing principles. This practice conforms to a suggestion made by [1] that given the complexity of state-dependent routing it is more reasonable to formulate sound routing principles rather than aim for optimality in routing. Such routing schemes using cost functions are referred to as *cost-based routing* (CBR). Almost all of the currently considered state-dependent routing schemes belong to the class of CBR. For the last two years, we have worked closely with Bellcore in investigating SDR, the Bellcore's version of CBR to be implemented in the regional telephone networks in the USA.

Below, we summarize some results attained during our work in 1992. The description here is purposely concise to present the overall picture of the project. A more detailed description can be found in [47].

- **Accurate Performance Evaluation Algorithm.** Blocking probability for the network is often the quintessential measure of the grade of service provided by a routing scheme. The blocking probability for FAR is usually calculated by a fixed-point method known as the Erlang approximation method. This method has been found to be computationally efficient even for large networks.

  Calculation of blocking probability in the case of CBR is considerably more complicated since it depends not on the state of a single, currently considered path but rather on the states of multiple paths. The main difficulty introduced by CBR is the complex patterns in the traffic intensity incident upon various portion of the network, which are dependent upon the entire state of the network. During 1992, we devised a way to determine the state-dependent traffic patterns of any arbitrary CBR scheme in a computationally efficient way, resulting in development of fixed point algorithms for CBR with highly accurate estimation of blocking probability. Initial implementation of this fixed point algorithm (for arbitrary CBR schemes with general network connectivity) has computational complexity of $O(C^2N^4)$ where C is an upper bound on the capacity of a link and N is the number of nodes. This complexity would be too high to be useful for large networks.

In order to be able to carry out analyses of various performance measures in the network, it is essential to have available a performance evaluation algorithm that is accurate and has reasonable complexity for large networks. We were able to improve upon the initial complexity to $O(C^2N^2)$ under very realistic assumptions. As a result, we can now calculate the blocking probability of networks with more than one hundred nodes which correspond to realistically sized LATA's in a populated metropolitan area in the USA. We are currently investigating the possibility of further reduction to $O(CN)$ using some form of diffusion approximation.

- **Traffic Patterns, Occupancy, and Link Cost.** An important characteristic of a routing scheme is the pattern of traffic it generates. For example, the efficiency of the routing scheme can be visualized by the fraction of carried calls that uses multi-link paths. Obviously, the most efficient use of network capacity is to allocate as many calls as possible to one-link paths as possible.

  We observed in simulation studies that the rate of traffic using one-link paths is constant for small values of state cost and drops rather rapidly above a certain threshold. To explain this phenomenon, we developed an asymptotic model (as the network size gets large) for the calls using one-link paths. Our model predicted accurately the behavior of such calls with respect to the values observed in the network via numerical studies. Similarly, we also developed an asymptotic model for the calls using multi-link paths. These two models together gave us an excellent tool for studying the dynamic behavior of traffic under varying capacity of the network. More detail can be found in [47].

  Another important characteristic of a CBR scheme is the shape of its cost function, and its effect on the routing decision. As a specific instance, we studied the cost function of SDR. It has a knee-like shape for one-link paths and the cost for two-link paths is derived from the convolution of the two one-link paths that it consists of. We developed an analytic model for such cost functions.

- **Dynamics Due to Network Size.** With increasing N, it is possible to increase the sizes of the set of paths for each origin-destination pair. As reported in [47], however, for FAR there is not necessarily a gain in network performance, such as blocking probability, when the size of the network in terms of N increases. We studied the impact of increasing the number of paths in CBR. The results reported in [47] show that this ability to use the entire network as a single resource is one of the most important benefits of CBR.

- **Network Survivability.** Another important potential advantage of state-dependent routing is its anticipated ability to recover reasonable performance in the face of link or switch failures. Intuitively, since all paths for an origin-destination pair are potential candidates for a call, a failure in a link or a switch ought not to result in a substantial degradation of performance. Moreover, one may expect that this degradation does not all devolve upon the origin-destination pair directly linked by the failed link but is shared by all the origin-destination pairs which utilize this link in their routing set. In our study, we observed that CBR is quite resilient to link or switch failures. The survivability of CBR may be the strongest motivation behind CBR.

Multi-channel Switching in Broadband ATM Networks

H. Saidi and P.S. Min

While strong consensus exists on ATM within the broadband community, some difficult technical challenges lie ahead. One such problem is the development of a switching methodology that can support the very high bandwidth transmission associated with lightwave technology. The transmission rate over optical fibers has reached tens of gigabits per second and is likely to increase in the coming years. On the other hand, the processing speed of electronic switches remains substantially slower. Such a mismatch between the two quantities creates a natural bottleneck at the switching nodes in the network.

Multi-channel switches have been proposed and studied in the literature [15] [29] [35] [36] [45] [49], as a means of alleviating the processing speed constraint of electronic switches in broadband networks. Multi-channel switches can provide higher performance (e.g., throughput, blocking probability, delay) by exploiting the concept of *trunk grouping.* Instead of being routed to a specific output channel, a packet is routed to any channel belonging to an appropriate trunk group. In ATM, a session is established by the assignment of a virtual circuit. The virtual circuit is defined in terms of a specific channel between the two end points of the session. However, many of the benefits associated with connection-oriented services do not require that the connection be specified at the channel level; it is sufficient to specify the path of the connection, not the specific channel within the path. This implies, among other things, that a packet can be routed to any output channel of a switch within a group of outputs, provided that it eventually leads to the same end point.

As the demand for new applications soars, greater variability in bandwidth and traffic characteristics (e.g., session duration, burstiness) is expected. The advantages of statistically sharing a higher channel capacity under such conditions are well known. For example, it increases the link efficiency by reducing the burstiness in the incoming traffic. Bit pipes of higher rates are formed which allow a number of applications to share bandwidth by dynamically allocating bandwidth. Assuming that the exogenous traffic intensity per channel is fixed, a larger trunk group size is less likely to incur blocking for a single ATM cell, for a burst of cells, or for a request for setting up a new session. Similarly, other performance measures such as cell delay, probability of buffer overflow, and congestion would improve when multiple channels are grouped together as a single resource.

Another important benefit of multi-channel switching is the ability to provide *super-rate switching.* Applications requiring peak rates greater than a single ATM channel capacity would suffer high cell loss probability unless these applications can be assigned over multiple channels. Trunk grouping would be a natural way to deal with such a problem. In reality, a trunk group may correspond to a single fiber operating at an extremely high rate, and multi-channel switches can provide a means of transporting this bit pipe across broadband networks without requiring complex multiplexing/demultiplexing functions at each switching node.

Cross-point complexity in multi-channel switches can be low. We illustrate such a possibility based on the following consideration: Given N inputs and N outputs, let us construct a switch that provides at least one path between every pair of input and output channels. A switch based on the banyan network would require $log_2 N$ interconnected stages, each consisting of $(N/2)$ 2×2 switching elements. On the other hand, if the requirement is changed such that each input is connected to any of the outputs in the same group, then the number of stages of the same switch can be reduced.

Assume, for example, that two outputs of the individual switching elements at the last stage of the Banyan network are bundled into a group such that there are $(N/2)$ trunk groups of two channels each. Then the switching function at the last stage is unnecessary since ATM cells arriving at these switching elements can be switched to any of the two outputs. In this case, the banyan network with $log_2 N - 1$ stages would suffice. We can continue this argument to show that for $2^{-r}N$ trunk groups ($r \leq log_2 N$), the resulting multi-channel switch requires r fewer stages than the original point-to-point switch.

One interesting result obtained during 1992 as part of our multi-channel switching work concerns a specific switching architecture that is internally non-blocking. The switching architecture proposed by [45] is an example of a non-blocking multi-channel switch employing the Batcher-banyan network. Strictly speaking, [45] is a non-blocking point-to-point switch; this switch can be used as a point-to-point switch, but when it is used in a multi-channel mode, the performance (viz., throughput and output contention probability) can be improved. There is no reduction in the hardware complexity.

In fact, any non-blocking, point-to-point switch is a non-blocking multi-channel switch. Using a point-to-point switch as a multi-channel switch does not capitalize on some significant benefits of multi-channel switching, such as reduction in the hardware and control complexities.

Lemma. Let $S_1' = [0, 1, 2, \ldots, K-2, K-1]$ and $S_2' = [N-1, N-2, N-3, \ldots, K+1, K]$. (For $K \geq N$, $S_1' = [0, 1, 2, \ldots, N-2, N-1]$ and $S_2' = \phi$. Similarly, for $K \leq 0$, $S_2' = [N-1, N-2, N-3, \ldots, 1, 0]$ and $S_1' = \phi$.) Now let S_1 be the sequence $S_1' \oplus v$ and let S_2 be the sequence $S_2' \oplus v$ where v is any number $0 \leq v \leq N-1$ and $\oplus$ is addition modulo N.

Define $S = (s_0, s_1, s_2, \ldots, s_{N-1})$ to be a sequence of N integers such that S_1 and S_2 are subsequences of S. Then, the N-input inverse omega network can simultaneously realize the set of N connection requests in the form of $r(i, s_i)$ for $i = 0, 1, 2, ..., N-1$ where $r(a, b)$ is the connection request between input a and output b.

We illustrate the application of the above lemma to broadband switching using the following example. Assume that the outputs of a switch are partitioned into four trunk groups which are labeled as X_{00}, X_{10}, X_{11}, and X_{01}. X_{00} has assigned to it output channels $0, 1, 2, \ldots, k_0 - 1$, and for X_{10}, the output channels are between k_0 and $k_0 + k_1 - 1$, for X_{11}, the output channels are between $k_0 + k_1$ and $k_0 + k_1 + k_2 - 1$, and finally for X_{01}, the output channels are between $k_0 + k_1 + k_2$ and $k_0 + k_1 + k_2 + k_3 - 1$. Assume $k_0 + k_1 + k_2 + k_3 = N$. Suppose that during a particular cell slot, input channels belonging to set Y_{00} need to routed to X_{00}. Similarly Y_{10}, Y_{11}, and Y_{01} are to be connected to X_{10}, X_{11}, and X_{01}, respectively.

If a cardinality constraint is not violated by any group (i.e., $card(Y_{00}) = card(X_{00})$, and so on), such a connection request can be satisfied by the following scheme: Consider the two staged binary sorting network shown in Figure 5, where each stage corresponds to the non-blocking binary sorter defined in the lemma.

At the first stage, separation into two groups is done based on the first bit of the group index (e.g., 0 for Y_{00} and Y_{01}, and 1 for Y_{10} and Y_{11}). This implies that Y_{00} and Y_{01} are aggregated together with its i^{th} member from the top routed to output i for $i = 0, 1, 2, ..., k_0 + k_3 - 1$, and Y_{11} and Y_{10} are aggregated with the j^{th} element routed to output $N - 1 - j$ for $j = 0, 1, 2, ..., k_1 + k_2 - 1$. Then the lemma guarantees a non-blocking connection at the output of the first stage as shown in Figure 5. In Figure 5, a sample instance of a two digit group index is shown on each input link. To show the propagation through the network, we also show the identity of the connection by indicating the input channel number within the

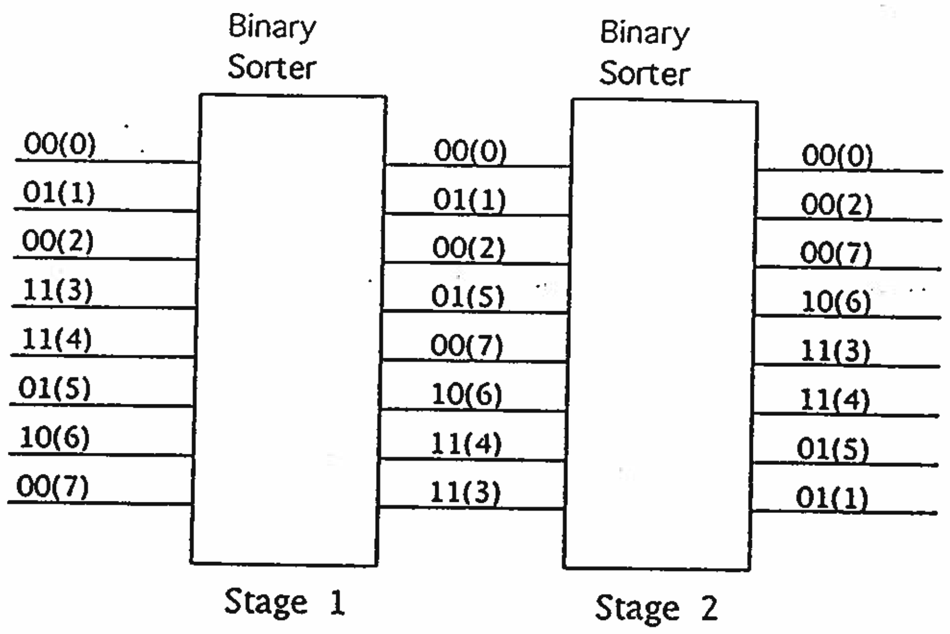

Figure 5: Non-Blocking Multi-Channel Switch for Four Trunk Groups

parenthesis next to the group indices.

Separation into four trunk groups is completed by applying the binary sorter again at the second stage. At the second stage, however, we group the connection requests based on the second bit of the group index (e.g., 0 for Y_{00} and Y_{10}, and 1 for Y_{01} and Y_{11}). This now implies that Y_{00} and Y_{10} are aggregated and Y_{01} and Y_{11} are aggregated together. Connections to proper outputs are shown at the output of the second stage. The resulting connection accomplishes the connection request for the four trunk groups.

So at least in theory, non-blocking connections to any multiple groups in the form of 2^r ($r \leq log_2 N$) can be achieved by cascading r binary sorting networks as shown in Figure 6.

In fact, the above argument suggest an alternative way to construct a non-blocking point-to-point switch by setting $r = log_2 N$. To resolve possible output contention, the switch requires concentration network followed by another banyan network each with $log_2 N$ stages.

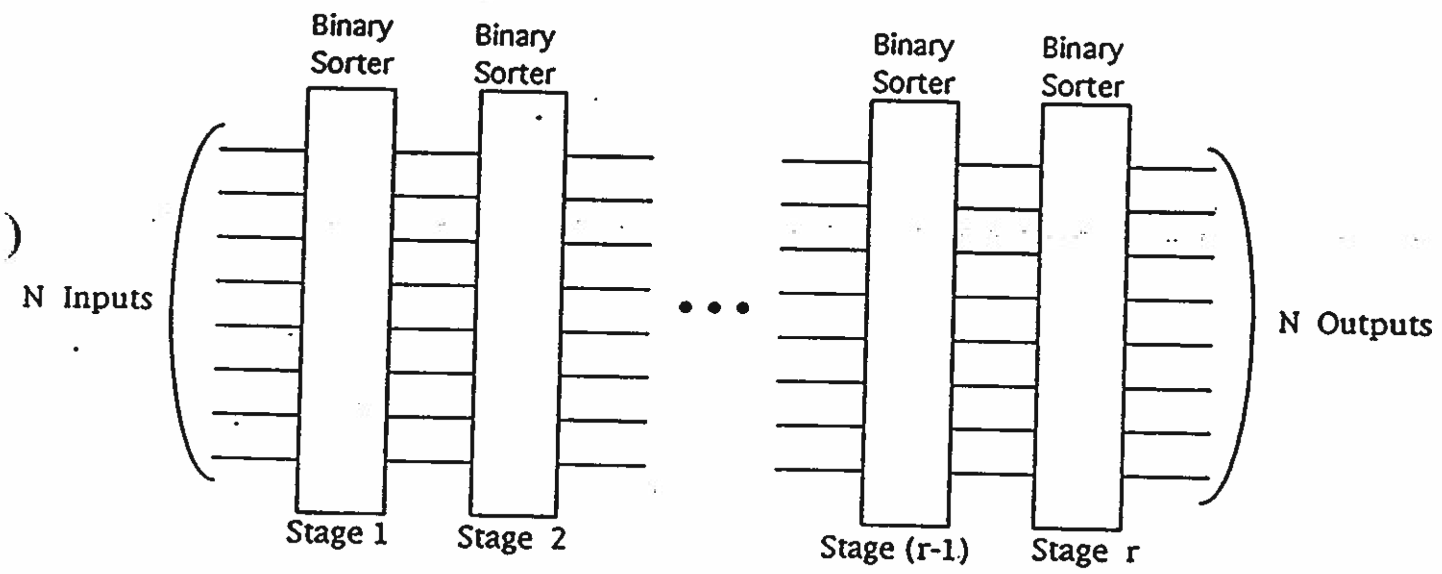

Figure 6: Non-Blocking Multi-Channel Switch for 2^r Trunk Groups

Congestion Control in Multimedia ATM Networks

Andreas D. Bovopoulos, Seyyed M-R Mahdavian and Apostolos Dailianas

High speed networking is promising to transform the nature of communications in the context of both public and private networks. In part this advancement is made possible by the development of the Asynchronous Transfer Mode (ATM) technology, which promises a far more flexible infrastructure than is currently available. The introduction of the ATM technology together with the deployment of fiber optic networks will result in an infrastructure capable of handling gigabits of multimedia (for example, voice, video, data) traffic in a very efficient way.

We are currently focusing in the development of traffic engineering rules for the efficient support of multimedia traffic. Typical media differ not only in terms of their time behavior, but also in their quality of service (QOS) requirements from the network. Current research has focused on the study of problems related to the multiplexing of traffic. The broad approaches taken focus on the cell loss incurred on a particular multiplexed cell stream.

For services which result in the transmission of information in blocks (bursts), such as in file transfers, the emphasis has been in trying to support multiplexing schemes which would result in preservation of bursts. Whereas these approaches aim at addressing the intuitive requirements for integrity of the burst, none so far has been able to correlate the performance requirements to the intrinsic information content of the media stream in a general and direct way. One of our research efforts aims at addressing this problem. We recognize the fact that cell loss cannot describe the information loss for a particular cell stream. To cope with this we introduce a new criterion called the virtual (information) loss, which measures the effect of lost cells on the integrity of the information in the whole burst.

In the sequel a random variable is represented by a symbol with a tilde "~" on top, while a particular value of a random variable is represented by the symbol alone. For example, $\tilde{m}$ represents a random variable, while m represents a particular value of this random variable.

The transmission of an arbitrary block of $\tilde{m}$ cells may result in a loss of $\tilde{\nu}$ cells at a cost $f_i(\tilde{m}, \tilde{\nu})$. As a first approximation it is assumed that f depends on the magnitude of loss, regardless of the precise location of the lost cells within the block. If different blocks are assigned different cost functions, the cost function will also be stochastic. However, for simplicity, it is assumed that $f_i = f$ for every i, such that f is a fixed function.

The virtual loss of a transmission is defined as follows [51]:

$$V \stackrel{\text{def}}{=} \frac{\mathrm{E}f(\tilde{m}, \tilde{\nu})}{\mathrm{E}f(\tilde{m}, \tilde{m})}$$

where E refers to the expectation over the probability space of $\tilde{m}$ and $\tilde{\nu}$. If $\tilde{m}$ and $\tilde{\nu}$ are discrete random variables, the virtual loss becomes:

$$V = \frac{\sum_{m,\nu} f(m,\nu)\mathrm{P}(m,\nu)}{\sum_{m,\nu} f(m,m)\mathrm{P}(m,\nu)} = \frac{\sum_{m,\nu} f(m,\nu)\mathrm{P}(m,\nu)}{\sum_{m} f(m,m)\mathrm{P}(m)}$$

where $\mathrm{P}(m,\nu)$ is the joint distribution of $\tilde{m}$ and $\tilde{\nu}$ and $\mathrm{P}(m)$ is the marginal distribution of $\tilde{m}$.

Virtual loss has the following properties:

1. If f is a non-decreasing function of ν (which is natural because more loss implies more cost), then $0 \leq V \leq 1$, because $0 \leq f(m,\nu) \leq f(m,m)$ for all m, ν.

2. If one defines the cost function to be $f(m,\nu) = \nu$, i.e. the cost of ν losses is just ν regardless of the size of the block, then $V = \mathrm{E}\tilde{\nu}/\mathrm{E}\tilde{m}$ which is simply the loss rate. This property is important because it shows that under the worst case condition where no natural blocks of information can

be identified, an arbitrary block division together with the above cost function gives the usual loss rate measure.

3. If f represents the amount of information which is *effectively lost* in a block, e.g. the size of the block in file transfers when part of the block is lost due to channel impairments, then the virtual loss represents the average rate of the information flow which is *effectively lost.*

Virtual loss has been used to compare the performance of reservation and non-reservation techniques [51]. Numerical results have shown that under certain circumstances, reservation may be a good protocol to protect the transmission channel against massive cell loss. These circumstances include:

- The traffic (and therefore the loss) at the multiplexer being high.
- The frequency of variation of the background traffic being comparable to that of the channel under consideration.
- The rate threshold being properly chosen.

If these conditions do not hold, however, the use of reservation may result in greater quality degradation.

We also attempt to develop comprehensive end-to-end admission control and resource allocation algorithms. Finally a problem we are currently studying is the effect of successive multiplexing stages on the traffic characteristics of a particular traffic stream.

SYMPHONY: An Architecture for Distributed Multimedia Systems

Andreas D. Bovopoulos, R. Gopalakrishnan, Saied Hosseini-Khayat

With the computer becoming more of a tool for interactive communication, designing the communication and computing components in isolation of one another is inadequate and inappropriate for offering multimedia services. The SYMPHONY design described herein, is based on the proposition that in order to provide multimedia services with performance guarantees effectively, an architectural design must be integrated with respect to these three aspects:

- Hardware design (Architecture Integration)
- Protocol support (Service Integration
- QOS specification (Performance Integration)

Given the above requirements, the design goals for the SYMPHONY architecture are formulated as :

1. Translate the high transmission rates provided by the network into application level throughput
2. Develop an architectural framework for integrating subsystems such as processors, storage, and digital I/O devices for different media and design uniform and efficient hardware and software interfaces between subsystems.
3. Allow negotiation of performance parameters and provide guaranteed performance of various subsystems used by an application.
4. Provide protocol support for communicating and processing digital streams of different media types with media and application specific performance requirements.

Hardware Organization

To preserve the high data rates provided by the network transport within the computer, the SYMPHONY architecture uses a separate network backplane, implemented as a pair of unidirectional buses. To be able to handle the high processing requirements for multimedia data, SYMPHONY is organized as a collection of autonomous units that perform network I/O independent of each other over the network backplane. Each unit negotiates usage of a bus, according to the bandwidth and delay requirements of the network traffic it sources or sinks. The salient features are retaining the *ATM cell* as the unit of data transfer on the network backplane, and to provide guaranteed access to the backplane for periodic data streams. This is similar to the approaches described in [48] and [32]. Apart from the guarantees provided by the network backplane, end-to-end QOS preservation requires that predictable performance is provided by each device for local processing, as well as the I/O interconnect (I-Bus) for inter-device transfers. Therefore the architecture requires an application to negotiate service requirements with each device, as well as the I/O interconnect. The hardware architecture requires appropriate choices for the operating system (OS) as well as the protocol processing architecture in order to achieve the design goals.

Operating System Architecture

The shift towards a multiprocessor architecture and the modifications to the I/O subsystem, necessitate changes to the OS structure. The OS must hide the functional asymmetry of the hardware from the application programs. A distributed operating system is proposed for SYMPHONY, in which each unit manages its own resources and provides its services through an interface. Each local OS is based on a microkernel that provides mechanisms for processes and interprocess communication (IPC). Depending on the functionality required, other facilities such as filesystem support may be provided. Two important features of the operating system is support for real time scheduling of periodic streams [33], and resource management mechanisms that provide QOS guarantees. A real-time microkernel [28] can be used to meet the former requirement. The IPC mechanism is similar to the remote procedure call, and is based on object method invocation. The reason for this choice is that it has high level language support, and can be implemented efficiently because of the proximity of the endpoints.

Protocol Processing Architecture

Multimedia applications have diverse and demanding transport requirements such as high throughput, low delay, and low delay jitter. In addition, they require communication primitives that model their complex interactions, such as multipoint *sends and receives*, temporal synchronization mechanisms, and QOS control mechanisms. We adopt a compositional approach to organizing protocols [26, 41, 50]. This allows fine tuning of protocol mechanisms, and offers structured mechanisms based on the object paradigm to develop and use protocols. The protocol processing architecture is designed to take advantage of the hardware and OS architectures [27]. The availability of sufficient processing power on each unit, and the ability of the network backplane to stream network data directly to each device, favors the strategy of moving protocol modules into the data stream path, rather than copy data into the address space of the protocol process. Thus the "in-band" processing is optimized, and the "out-of-band" processing is implemented by a decentralized control mechanism implemented using the IPC mechanism provided by the OS. This approach also exploits the inherent coarse grain parallelism in applications that allows media streams to be processed concurrently. The architecture is implemented using server and client objects that perform control functions using the IPC mechanism described earlier.

Experimentation with TCP/IP Protocols

Christos Papadopoulos, Guru Parulkar

There has been considerable debate in the research community regarding the suitability of existing protocols such as TCP/IP for future applications over high speed networks. One group of researchers believes that the TCP/IP protocols are suitable and can be adopted for use in high speed environments. Another group claims that the TCP/IP protocols are complex and their control mechanisms are not suitable for high speed networks and applications. This research project was motivated by the above controversy. The objectives are the following:

- Characterization of the performance of TCP/IP protocols when used for communication intensive applications.
- An attempt to investigate the scalability of TCP/IP protocols to future networks and applications.

The IPC under study is the implementation in SunOS 4.0.3, which is heavily based on 4.3BSD. The Ethernet controller is the AMD AM7990 LANCE. In this environment IPC resides in the kernel and is organized in 3 layers: (1) the socket layer, which is a buffered interface to the applications; (2) the protocol layer, where the Internet protocols reside grouped in domains; and (3) the network interface layer, which contains the device drivers for the network hardware interfaces. In the current implementation, as data travels through these layers, it may be queued at four separate points: (1) at the send socket buffer awaiting transmission by the protocol; (2) at the send network interface awaiting transmission to the network; (3) at the receive protocol queue, awaiting reception by the protocol; and (4) at the receive socket buffer, awaiting delivery to the application. Figure 7 depicts the four queueing points with respect to the IPC layers. During transmission, data is copied from the application address space to the kernel address space and from the kernel to the network. During reception, data is copied from network to kernel and then back to the application. Therefore, IPC requires two intermediate data copies during communication. Moreover, the CPU will scan the data once during transmission and once during reception to compute CRC.

We feel that the study of the behavior of the queues may provide insight into the operation of IPC. Therefore to monitor queue activity probes were inserted in the kernel. The probing mechanism developed is comprised of a number of small probes monitoring data entry and data exit from each queue. Each probe produces a record with minimal information every time the state of a queue changes (i.e. data is added or deleted). The location of each probe is depicted in Figure 7. The records are stored temporarily in the kernel address space to minimize overhead. At the end of each record collection, a user process reads the data out of the kernel into a file. A variety of filters are then applied to the file to extract the required information.

Results

Experiment 1. Several experiments were performed in order to assess the performance of the IPC mechanism. Some results were obtained by measuring performance from the application's point of view. In the first experiment the IPC throughput between processes on separate machines was measured when large segments of data were transferred from one machine to the other. During the experiment the size of the socket buffers (both send and receive) was varied in order to observe the effect on throughput. The results showed that throughput increases as the buffers are increased from the 4K default size to the maximum of 51K. The biggest increase comes with the buffers set at 16K, with slight improvements thereafter. Throughput was measured at 5.8 Mbps with 4K buffers, 7.2 Mbps with 16K buffers, with a maximum of 7.5 Mbps with maximum size buffers, which is quite good on a 10 Mbps Ethernet. It is

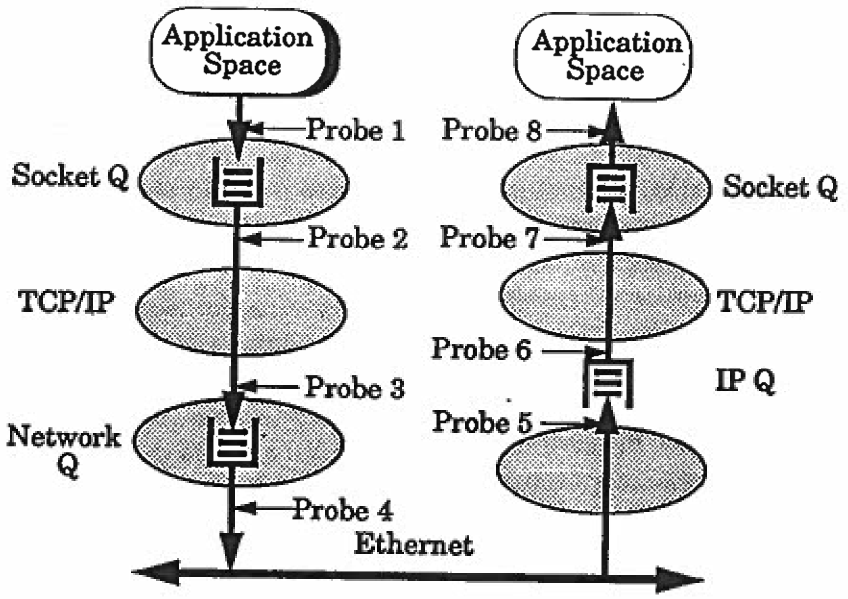

Figure 7: TCP/IP Queueing Network in SUN OS

interesting to note that the bottleneck which prevented higher throughput was traced to the network driver rather than the protocol, as one might suspect.

Experiment 2. In the second experiment the probing mechanism was used to get a better understanding of the behavior of IPC. The experiment had two parts. In the first part, a large chunk of data was sent through a unidirectional connection and traces were produced showing the flow of packets across the various IPC layers. The traces indicated that the protocol layer can process packets faster than the network layer can pass them on to the Ethernet. In addition to the traces, queue length and queue delay plots were also produced from the same data. The queue length plots showed clear queueing at the network interface (during sending) confirming the network layer' s inability to keep up with the protocol layer. On the receiving end, the plots showed that the IP queue never built up (during reception) and introduced a very low delay. The receive socket queue however, showed significant queueing. Therefore, the measurements show the bottleneck on the transmit side to be the network, and on the receive side the data delivery to the application from the socket layer. The protocol processing for the sending and receiving side was measured to be about 370 microseconds for the sending side, and about 260 microseconds for the receiving side.

In the second part of the experiment, a unidirectional connection was set up again, but only enough data to fit in a single packet was sent in order to isolate and measure the packet delay at each layer. A number of isolated packets were sent over a period of time, allowing enough time in between for each packet to reach its destination and the acknowledgment to come back. This allowed the measurement of processing for each individual packet at each layer. The data in each packet was 1024 bytes. The results show that on the sending side the socket processing including data copy is about 280 microseconds, the protocol processing increases to about 443 microseconds, and the network interface delay is about 40 microseconds. On the receiving side the IP queue delay is about 113 microseconds, the protocol processing about 253, and the receive socket processing about 412 microseconds. Memory to memory copy for 1024 bytes was measured to be about 130 microseconds.

Experiment 3. The third experiment investigated the effect of the receive socket queueing on the protocol operation, and especially on acknowledgment generation. It was determined that TCP acknowledges data in one of two cases: (1) whenever enough data is removed from the receive socket buffer as a result of the application performing a read; and (2) when the delayed acknowledgment timer process runs. For bulk data transfer however, the latter is much less significant. It was observed that data arrived in fast bursts from the sender and could accumulate in the socket buffer of the receiving machine. In extreme cases the whole window burst would accumulate in the buffer. The effect is more prominent with a slow, or otherwise loaded receiver. It was also observed that most acknowledgments were generated when the receive buffer became empty, since the application was trying to receive as much data as possible. This leads to the situation where a burst does not get acknowledged until all the data is passed to the application. Until this happens, the protocol is idle awaiting the reception of new data, which comes only after the receive buffer is emptied and the acknowledgment goes back to the sender. Therefore, idle periods are introduced during bulk data transfer which are equal to the time to copy out the buffer plus one round trip delay for the acknowledgment.

Experiment 4. The fourth experiment investigated the behavior of the various queues when the host machines were loaded with some artificial load. The experiment was aimed at determining which parts of the IPC mechanism are affected when additional load is present on the machine. The experiment was performed by setting up a unidirectional connection, running the workload, and blasting data from one machine to another. Throughput measurements were performed, and showed that throughput dropped dramatically as the background load on either the server or the client was increased. Figure 8 shows the behavior of the IPC queues during data transfer with background load. It clearly shows that the receiving application is not scheduled often enough to read data out of the socket buffer, with the result that after the window is fully open, the bursts fill up the buffer completely, which then empties and an acknowledgment is generated. This behavior exacerbates the acknowledgment starvation discussed in the previous experiment.

Conclusions

The above experiments suggest that some further tuning may be required before TCP is used successfully in high-speed networks. The reduction in acknowledgments could be especially harmful during congestion window opening on high-bandwidth, high-delay networks. The situation will be even worse if the TCP large windows extension is implemented, resulting in the slow start mechanism becoming too slow. Moreover, TCP could degrade to a form of stop-and-wait protocol, with the sender sending bursts at an unnecessarily high rate, since the receiver cannot keep up. Communication in this environment is also shown to require a large portion of the CPU to sustain high throughput. The arrival of new faster processors means that more CPU power could be devoted to communication, but future network speed is expected to increase faster than processor speed. Moreover, it is questionable whether a significant portion of CPU power should be allocated for protocol processing instead of computation. Details of this study are presented in [43, 44].

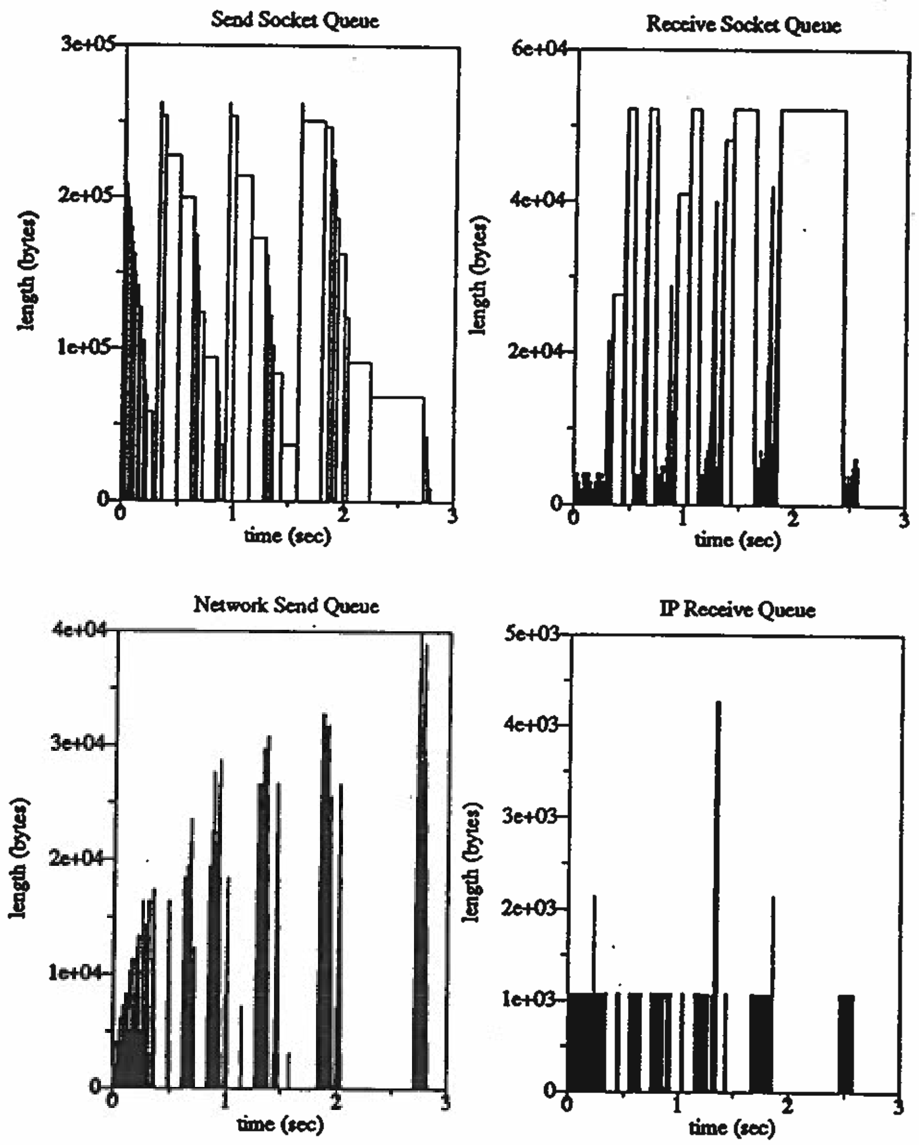

Figure 8: TCP/IP Performance

Segment Streaming IPC for Network Pipelines

Fengmin Gong, Guru Parulkar

There are two main motivations for using a network to distribute a computational task across more than one machine. The first is to make use of advanced data and computing resources that are not available on the local machine. Secondly, by distributing a computation among multiple machines a speedup in performance (*e.g.*, throughput) can be achieved due to concurrency. There are two well-recognized techniques for achieving concurrency by simultaneous operation on multiple computing elements: *parallelism* and *pipelining*. In the parallelism approach, a hardware structure is replicated many times and each replica executes in parallel different parts of the large computation; pipelining technique splits the hardware structure into a sequence of substructures (called stages) corresponding to phases of the computation task and allows all stages to operate simultaneously on different parts of the computation. Thus, a network pipeline has the potential to allow both speed up in performance due to concurrency and access to remote resources. In fact, pipelining is the most effective approach to achieving concurrency when the computation itself consists of sequential steps.

Pipelining has been successfully used at different levels within a machine, but use of a pipeline as a model for network computing is new. This paradigm is worth exploring because it very well characterizes an important class of network computing and visualization applications, as explained in the following paragraphs.

Scientific visualization, as defined in the 1987 NSF report *Visualization in Scientific Computing*, is a method of computing which transforms symbolic information into geometric information, enabling researchers to gain better insight into their simulations and computations. The NSF report has identified a large number of scientific and engineering applications that can benefit from visualization. A significant class of these applications have a computation model that contains the following general steps:

Data computing. This is the step in which the discipline-specific computation takes place. Numerical simulation and image scanning are two examples. This step generates the original scientific data that needs to be visualized for human interpretation.

Data preparation. This stage derives visualization data from observational, experimental, or simulation data. For example, for visualization of a simulation, additional data points may be interpolated to transform an irregular grid into a regular one so that certain visualization algorithms can be applied.

Visualization mapping. This stage constructs abstract visualization objects from the data derived at the previous stage. Specifically, it maps the derived data (*e.g.*, pressure, temperature, and intensity) to the attributes of the visual representation (*e.g.*, geometry, color, and opacity) for graphics rendering.

Graphics rendering. Abstract visualization objects generated in the last step are rendered into images for display at this step. The actual operations may include image processing, surface rendering, and volume rendering.

User interaction. The user interacts with the entire computation through the image display and input devices such as a keyboard, a mouse, and a set of dials. Interpreting user input and interfacing with the rest of the computational stages are functions of this stage. The user interaction functions are often

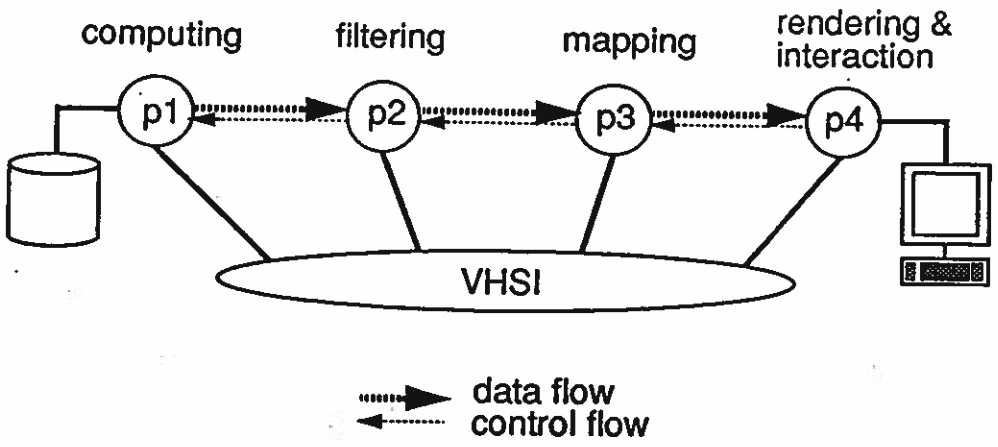

Figure 9: A Pipelined Network Computing Application

implemented on the same machine where the graphics rendering takes place.

Clearly, the first four major tasks have to be sequential. Therefore, an efficient model for implementing such applications is to distribute different stages of the computation onto separate machines interconnected through a pipeline on a very high speed internetwork (VHSI), as shown in Figure 9. The figure shows four major pipeline stages on four separate machines, although in practice a change in the configuration of these stages may be needed due to constraints of compute power or communication bandwidth.

In a network pipeline such as the one described above, interstage communication incurs significant delay due to the physical distribution of pipeline stages. The slowest stage becomes the bottleneck and limits pipeline speed. In order to achieve efficient pipelining, the interstage communication has to satisfy a number of conditions:

- A pipelined network computing application typically involves a large number of data segments of considerable size. These segments have to be streamed through the pipeline with minimum delay to allow overlapped processing.
- Computation and communication speed should be well balanced for optimal utilization of both resources.
- An application should be able to specify its error tolerance requirements to the IPC mechanism and the IPC mechanism should enforce it only to the degree that is necessary to satisfy the requirements, thus avoiding any unnecessary error control overhead.
- Buffer overflow in the pipeline should be avoided because loss of partially processed data due to overflow may require restarting the pipeline from an earlier stage, thus wasting computing cycles and introducing unnecessary (and unacceptable) delay.

Existing IPC mechanisms and underlying communication protocols do not have the functionality to meet these demands. Significant progress is being made in the development of high-speed packet switching networks, internetworks, and host-network interfaces. This development sets the stage for the development of IPC facilities that are necessary for converting the high bandwidth of networks into high performance for distributed applications. To provide efficient interstage communication for a network pipeline, we have proposed an IPC solution which consists of three important parts: a segment streaming IPC paradigm, a two-level flow control method, and an application-oriented error control method. We summarize segment steaming in this section.

Segment streaming

We have proposed a new IPC primitive called segment streaming, especially for efficient pipeline communication. It allows exchange with high bandwidth and low latency of a large number of data segments among processes. Segment streaming works as follows: An application process starts a segment stream by making a system call (`send_stream`); the transport protocol mechanism then establishes the stream with the remote system through a single request-response procedure; from then on each segment will be transmitted when ready without the latency of further request or setup. Moreover, the `send_stream` call supports a set of options which allow applications to specify their particular error and flow control requirements. The key idea of segment streaming is to maintain an efficient and continuous flow of data items throughout the pipeline by (1) reducing the overhead associated with local interactions between the application process and the communication process, (2) minimizing the overhead for stage-to-stage (i.e. end-to-end) communication setup, and (3) providing error and flow control functions that can be easily tailored to the application needs. Segment streaming is invoked through a *send-stream* system call. Logically speaking, the call takes the following form:

```
send-stream(group, host, access, options)
```

where group is the name of the group of segments to be transferred as a stream; `host` is the host where the segment group is sent to; the `access` structure contains information such as process name and authorization code that will be used to determine the remote communicating process and verify access permission. The options are used to specify a number of modes for streaming operation:

Repeated/Sequential Transfer: This option allows the application to specify if the stream consists of repeated transfers of a single segment or sequential transfer of segments in a group.

Loss Tolerance: An application specifies its loss tolerance which will be satisfied by the error control mechanism with minimal overhead. The exact tolerance specification and details of the error control mechanism are presented in a subsequent section on error control.

Flow Control Mode: This option allows the application to specify the flow control methods to be used with the stream. Flow control determines when to initiate the transmission of each segment in the stream. This transmission can be initiated at fixed time intervals (*interval synchronized*), triggered by the execution of a specific program call (*program synchronized*), or determined by a two-level flow control mechanism which will be described in a subsequent section on flow control.

Send-stream works as follows. Upon invocation by an application process, a local control block for the stream is created and a *request* packet is sent to the remote host through a connection which is established by calling the network service. The packet contains the name of the segment group and destination user and process information. After an *acceptance* is received from the remote host, a *confirmation* packet is sent, followed by a series of data packets which correspond to segment(s) as they are streamed. There is no further end-to-end request or setup overhead for the whole segment group. Figure 10 is a high level view of *send-stream* operation. The actual time elapsed t between two successive segment transmissions is determined by the particular flow control method used.

In segment streaming, the synchronization between the sender process and the communication process takes on the form of a producer and consumer. A tag associated with each segment indicates if the segment is ready

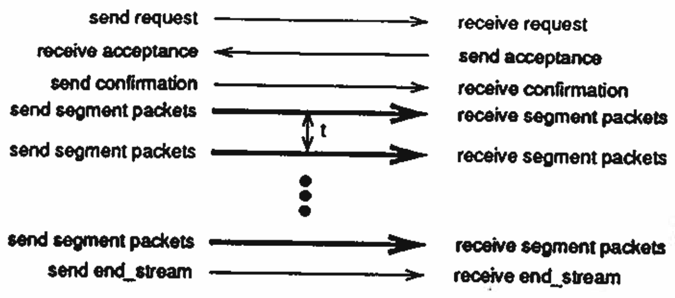

Figure 10: Send-Stream Flow

for transmission by the communication process and if it can be released by the sender process. A segment buffer space for both computation and communication is reserved for each segment streaming application at an initial setup time. Access to the shared buffer space as well as the tags are protected through semaphores. On the receiving side also, the application process and underlying communication system (kernel and transport protocol) have a producer-consumer relationship using shared preallocated segment buffers with tags.

It is clear that segment streaming allows overlapping between intrastage computation and interstage communication. This provides a necessary condition for efficient operation of the pipeline. Applications can specify their error and flow control requirements through the IPC primitive that allows the underlying protocol to avoid unnecessary overhead. For a typical distributed pipeline application, the sequential transfer mode will be used. The error tolerance can be determined from the amount of redundancy in the data, the delay requirement of the application, and how efficiently the application can cope with loss of data by itself. The flow control option should be set to activate the two-level control mechanism in order to avoid overflow in the presence of user interactions as well as speed fluctuations in pipeline stages. A segment stream will be established between a pair of neighboring stages. A segment group is defined as the set of data items to be processed, with each data item corresponding to one segment. Detailed specification of segment streaming can be found in [20, 22].

An Application-Oriented Error Control Scheme

Fengmin Gong, Guru Parulkar

The new generation of high-speed networks will interconnect machines with very high bandwidth over long distances. The high data rates and long propagation delays result in very large bandwidth-delay products. This large bandwidth-delay product has a number of implications on strategies for error control: (1) Time for end-to-end control actions (minimum of one round trip delay) becomes very significant with respect to the high data rate; and therefore, frequent end-to-end control actions should be avoided. (2) It becomes very difficult to achieve efficient error control using only timer-based loss detection, because high data rates make it more difficult to set timers accurately and also make it costly to have an inaccurate timer. (3) It becomes very expensive to recover erroneous packets through retransmission.

On the other hand, new applications, such as pipelined network computing and visualization, require high bandwidth and low latency communication with performance guarantees. They also deal with different types of data streams (*e.g.*, voice and video streams, image sequence, and data set) which have very different error tolerances. It is thus highly desirable to have a flexible service interface provided by the transport level that allows the application to accurately specify its error tolerance and thus avoid unnecessary and long recovery delays due to retransmissions.

Existing transport protocols provide only two classes of error control to applications. In one class, no error control function is provided, and it is up to applications to decide if error control is needed and do it by themselves as necessary. The second class of error control provides 100% reliability, *i.e.*, the transport protocol requests retransmission for every lost or corrupted packet regardless of the error tolerance of the application or the "cost" involved. Furthermore, existing error control schemes rely heavily on timers for loss detection and have to perform frequent end-to-end control due to their use of small data granularity. Clearly, such error control strategies are not appropriate for the new applications.

A new application-oriented error control scheme has been developed that provides efficient error control for a class of pipelined network computing applications. This scheme has been evaluated using analysis and simulation, and it has been implemented in software inside SunOS 4.0.3 kernel. The following paragraphs summarize the key features of the error control scheme and present some selected results. Details of this work are available in [20, 24].

Characterization of application error tolerance

The first requirement for designing an efficient error control scheme is a suitable characterization of the application's error tolerance. Specifically, it should be powerful enough to describe detailed distribution of tolerable errors, *e.g.*, within a data segment. It should also be expressive enough to describe a wide range of error tolerances (*e.g.*, from tolerating no error to tolerating all errors). In the proposed scheme, the application specifies its error tolerance as a triplet (W, E, B), which is interpreted by the error control mechanism as follows:

1. For every W packets transmitted, there can be no more than E packet lost when delivered to the application process at the receiver.

2. Among the lost packets there cannot be more than B packets with consecutive sequence numbers.

The error control mechanism will perform retransmissions as necessary to satisfy the application error tolerance.

Error detection and recovery

Packet losses must be detected before their recovery can be done. Since the receiver has to make the final decision for accepting a segment according to the application error tolerance, loss detection is also performed at the receiver to avoid extra information exchange between the sender and the receiver.

Two mechanisms are used to ensure early detection of losses. First, each packet from the sender to the receiver contains a field called the `shipment sequence number`, which uniquely identifies the sequence in which a packet is sent out at the sender. At the receiver, a gap detection mechanism checks the `shipment sequence number` of all incoming packets and detects a loss if there is a gap (*i.e.*, a hole) in the sequence number. In addition to the gap detection mechanism, the receiver also uses a loss timer. The purpose of the timer is to detect very long bursts of losses, in which there may be no subsequent packets to enable detection of gaps. The timer is set when a connection is established and reset upon reception of any packet. The timer value should be relatively long (e.g. equivalent to the time for transmitting several segments).

Once loss detection is made, recovery from the loss involves three steps: (1) The receiver still has to determine which lost packets need to be requested for retransmission. (2) The request information needs to be passed to the sender. (3) The sender needs to verify the request and retransmit the necessary packets.

The packets that have been successfully received are marked with a "1" in the corresponding position of a packet-presence bitmap. At the time of a loss detection, retransmission request list needs to be generated from the packet-presence bitmap and the application error tolerance. The retransmission list is created by scanning through the packet-presence bitmap and artificially setting "1" to the positions where loss of the packet does not violate the application error tolerance.

Acknowledgement scheme

With any retransmission-based recovery scheme, there are two pieces of information concerning the state of the receiver that must be available to the sender: (1) which packets have been correctly received so that the buffer at the sender can be released, and (2) which packets need to be retransmitted. Since loss detection is made at the receiver in the proposed scheme, the receiver has to send explicit information about what packets need to be retransmitted as well as which segments have been correctly received.

The proposed scheme uses two types of acknowledgments. The first type is positive acknowledgment (PACK) and it is used to inform the sender that certain segments have been accepted and can be released; A PACK can acknowledge a single isolated segment or it can acknowledge a contiguous block of segments if the block is the next expected in sequence. Selective negative acknowledgment (SNAK) is the second type of acknowledgment. It is mainly used to request packet retransmissions from the sender. However, it also carries a segment sequence number that informs the sender of the acceptance of all segments with sequence numbers below this number. The proposed scheme also transmits SNAK messages to the sender periodically so that long delays can be avoided should a SNAK get lost. Periodic transmission of a SNAK stops when the corresponding segment is successfully accepted.

Performance results

The performance of the error control mechanisms has been evaluated using both analysis and simulation. Simple analyses are performed in order to gain insight into the mechanisms and to provide verification for the simulation model. Discrete event simulations are used for more detailed study with a wide range of operational parameters. Example results of the simulation study are presented here. For details see [20, 24].

Maximum Segment Delay with Bursty Loss. Figure 11 shows the trade-off between

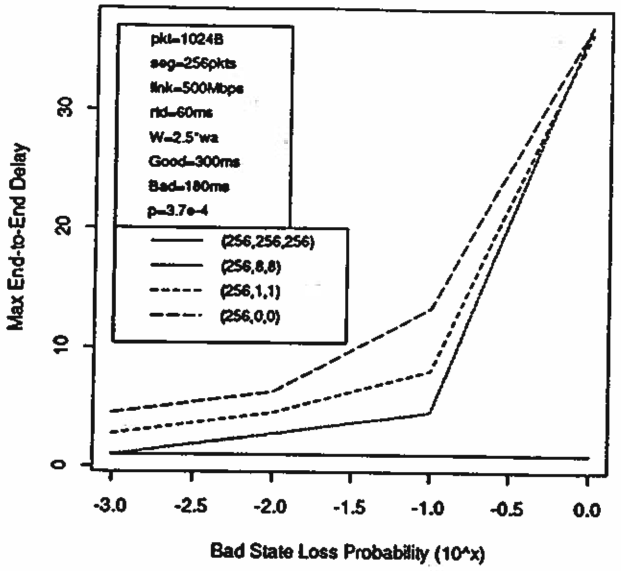

Figure 11: Maximum End-to-End Delay with Variable Tolerance

the loss tolerance and the maximum segment delay achievable under bursty losses. In the simulation, bursty loss is modeled using two alternating states: a good state with fixed duration of 300 ms and a bad state with an exponentially distributed length. To maintain the readability of the plot, only the result for a mean bad state duration of 180 ms is included. The set of four curves represent the maximum delay vs. loss probability relationships for four different loss tolerances. The following can be observed from the plot:

- Similar to the results under random losses, as the application tolerance increases from (256, 0, 0) to (256, 256, 256) the maximum segment delay shows significant decrease accordingly, despite the fact that segment size is larger (256 packets). With larger segment sizes, more packet losses will fall within one segment and thus reduce the impact of loss on maximum segment delay.
- As the bad state loss probability approaches 1 the differences among the three lowest tolerance settings diminish. This is not surprising for the following reasons:
  - All three tolerances will be violated under the same condition because every packet will be lost once in a bad state.
  - Recall that the retransmission strategy will request retransmission for all the packets lost in a burst from the point the tolerance is violated, which are almost all the packets ever lost for all the three tolerance settings.
  - Once the three tolerance settings lead to the same number of segment retransmissions, waiting time due the round trip delay affects them the same way.

Additional results with different bad state durations have shown that: (1) As the bad state duration increases, the tolerance vs. delay trade-off becomes much more significant as an indirect result of increased impact of bursty loss on the maximum delay; (2) When the bad state duration is decreased, the reduction in maximum delay with increasing tolerance becomes less. However, with bad state duration as short as 15 ms the trade-offs with the same set of tolerances as used above are still appreciable.

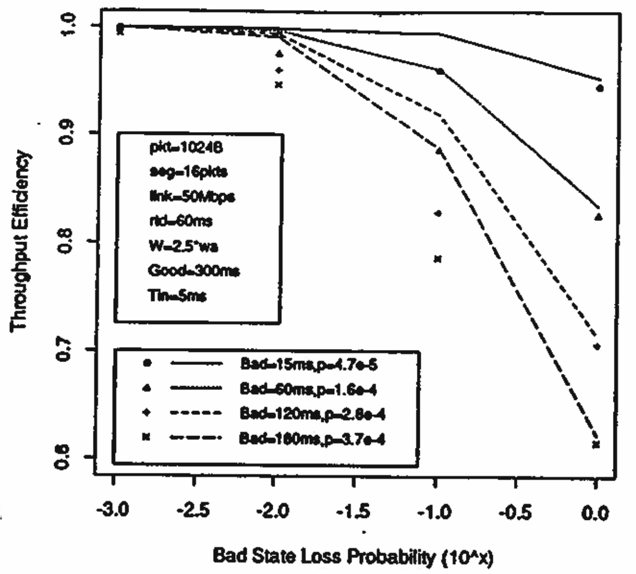

Figure 12: Throughput Efficiency Comparison

Throughput Comparison with SNR Scheme. SNR is a transport protocol developed in AT&T Bell Laboratories [40]. It has been shown that the SNR error control scheme out performs those used in existing protocols [17]. Figure 12 shows a comparison of throughput efficiency between the proposed scheme (in line style) and SNR scheme (by discrete points). There are four sets of data corresponding to four different mean bad state durations. As expected, there is little difference between the two schemes when bad state loss probability is very low ($< 10^{-3}$); but as bad state loss probability increases, the proposed scheme performs better and better than the SNR scheme; the difference will reach a maximum before it starts to diminish as the bad state loss probability approaches 1. The diminishing difference is also expected, because when almost every packet is lost in the bad state, selective retransmission at the packet level has very little advantage over retransmission at the block (segment) level. It should also be noted that, the performance advantage of the proposed scheme increases with longer bad state duration.

Simulations have also been done for comparison of the SNR error control and the proposed scheme under random loss conditions. Throughput efficiency, average delay, and maximum delay are all collected for each of the simulations. All the comparison results showed superior performance by the proposed scheme over the SNR scheme.

Two-Level Flow Control for High-Speed Networks

Fengmin Gong, Guru Parulkar

There are many proposed end-to-end flow control schemes and some of them have been implemented in transport protocols [16, 40]. For example, a sliding window scheme is used in TCP and a rate control scheme is used in NETBLT [11]. However, the emerging high speed networks (*e.g.*, ATM) and new applications (*e.g.* pipelined network computing and visualization) have changed many assumptions on which the design of existing schemes have been based. Examples of such assumptions are: the networks provide only unreliable datagram-oriented services, they have relatively small bandwidth-delay product, and the network applications require 100 % reliability. It is unrealistic to expect that the existing schemes would support efficient communication in the new environment.

Flow control has two priciple objectives. First of all, flow control ensures that the data traffic from the application will not exceed the negotiated rate with the underlying network; otherwise, the network may not guarantee the level of service needed for the application. Secondly, the pipeline flow should be maintained close to the rate of the bottleneck stage in order to avoid any buffer overflow. A two-level control mechanism designed to achieve these objectives has the following features (Figure 13):

The two objectives of flow control address different levels of data granularity, namely, the rate agreement at the packet level, and the interstage (end-to-end) flow at the segment level. A two-level control mechanism is introduced to achieve the two objectives.

Rate Control: A rate control mechanism is adopted to regulate the traffic at the interface according to the specification agreed upon at the underlying network connection setup. In exchange, the underlying network provides a guaranteed level of service in terms of bandwidth, packet loss probability, and delay. This is critical for the target applications.

Window Control: A simple window mechanism is used for stage-to-stage flow control. A window of size W at instance i defines a segment sequence number space $[i, i+1, \ldots, i+W-1]$. The sender can transmit a segment only if its number lies within this space. Only the receiver can advance the window by increasing either i or W. This credit-based scheme ensures quick control activation and deactivation in response to load fluctuation or user interactions.

Avoiding Congestion Complication: While rate control does help the congestion control in the underlying network, the windowing mechanism performs end-to-end flow control. This avoids adverse interactions that may take place when an end-to-end window is overloaded with flow, error, and congestion control functions (as in the case of the Internet protocol TCP).

A detailed specification of the proposed two level flow control scheme is presented in [20].

Performance results

Given the design of the two-level flow control scheme, a few important performance questions need to be examined. (1) How can the importance of rate control be justified quantitatively? (2) How does the window control perform given that the rate is guaranteed? (3) Is it necessary for the size of the receiver's buffer to be as large as the window? These questions have been explored, but we present results for only the last one here. See [20, 25] for a complete set of performance results on the proposed flow control scheme.

Buffer Requirements. It should be pointed out that the sending buffer size always has to

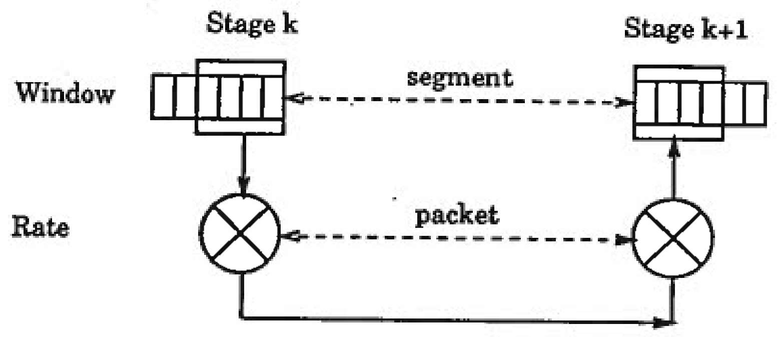

Figure 13: Two-Level Flow Control Structure

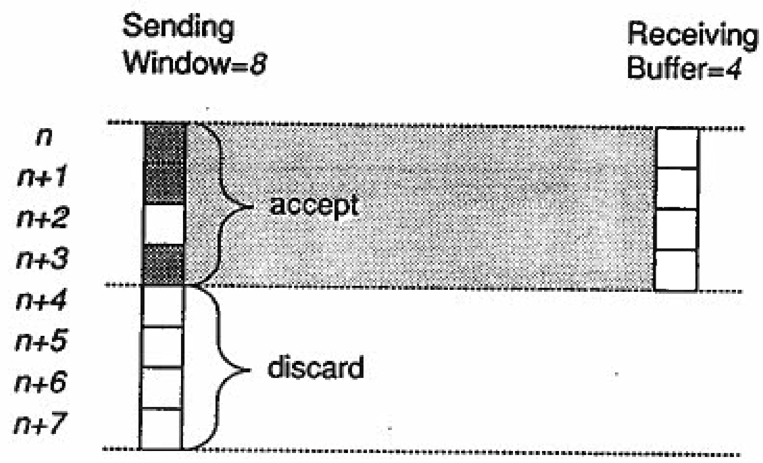

Figure 14: Hard ACK Strategy

be at least as large as the window size. The receiving buffer, on the other hand, can have a different size than the end-to-end window size. In particular, we would like to use a smaller receiving buffer if possible due to the constraint of physical memory sizes. For example, if we have a transcontinental connection with a bandwidth of 1 Gbps, with a round trip delay of 100 ms the bandwidth-delay product is 1 Gbps ×100 ms = 12.5 megabyte. According to the result above, the window requirement is 2.5 × 12.5 = 31.25 megabyte under practical network conditions. Even by today's standards this is a significant memory requirement for a workstation.

Indeed, under error-free conditions, the receiving buffer requirement is usually much smaller than the window size. However, when there are errors or when the speeds of the protocol and the application do not match perfectly, data segments may have to be dropped or overwritten if no additional buffer slots are provided. An interesting question is: how effective it is to advertise larger windows for achieving higher performance? Hard ACK and soft ACK are two types of acknowledgment strategies that can be used when the receiver is advertising a window larger than the actual receiving buffer available. To this date, there has been no performance study done for these strategies, to the best of the our knowledge. The hard ACK and soft ACK strategies will be defined first and all results should be interpreted with respect to these definitions.

Let $(n, n+1, \ldots n+W-1)$ be the current window of size W that the receiver has advertised to the sender. Let B be the actual buffer size and by assumption, $B < W$. Figure 14 illustrates how the *hard* ACK strategy works with $W = 8$ and $B = 4$. We can see that the receiver simply discards any segments with segment numbers outside the range $[n..n+3]$.

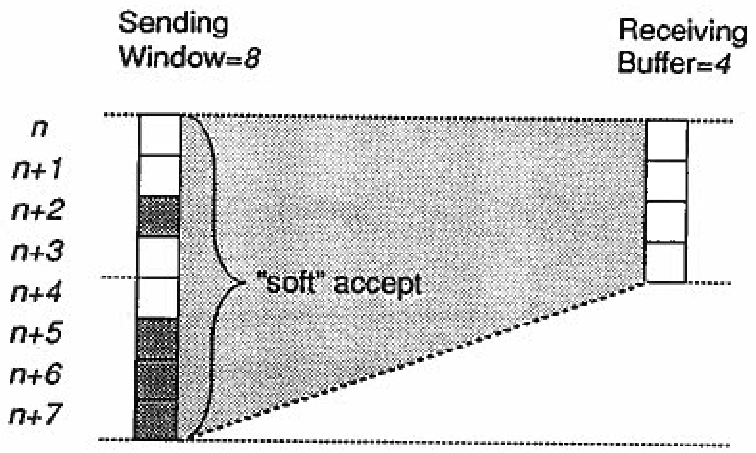

Figure 15: Soft ACK Strategy

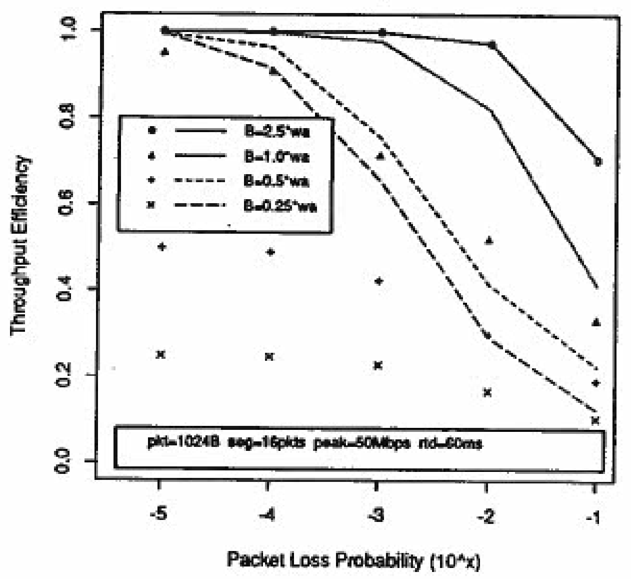

Figure 16: Advertising Larger Windows

All segments accepted will be acknowledged with certainty.

The *soft* ACK strategy, however, will save a segment into the receiving buffer as long as it is inside the current window $[n, n+1, \dots n+7]$ and there is space available for the segment, as shown in Figure 15. Because of retransmission and resequencing inside networks, a scenario like the one shown in the figure can occur. In this case, segments $n+2$, $n+5$, $n+6$, and $n+7$ arrived at the receiver before segments n, $n+1$, and $n+3$, and filled up the buffer. In order to deliver the next contiguous block of data to the application, some of the saved segments (*e.g.*, $n+5$, $n+6$, and $n+7$) will have to be discarded to make room for segments n, $n+1$, and $n+3$. This means that when those out-of-order segments are received, they could only be acknowledged with a special message indicating their successful arrival at the receiver but the sender should not release those segments until further acknowledgment is received. The name soft ACK reflects exactly this special requirement.

Three simulation experiments have been conducted to (1) quantify the advantage of

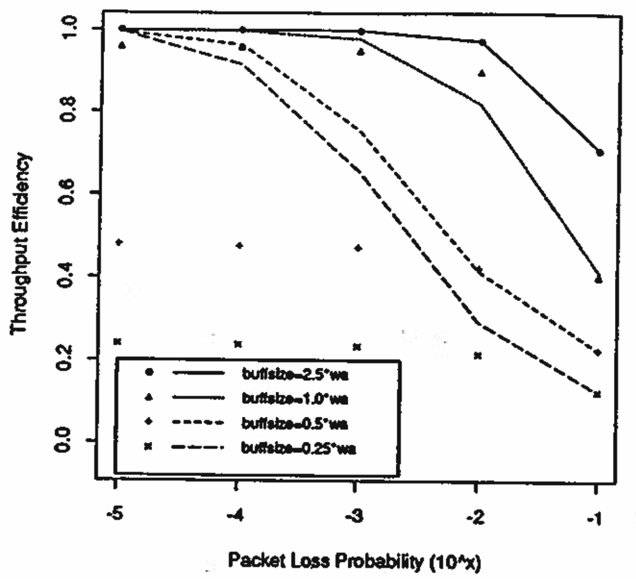

Figure 17: Hard ACK versus Soft ACK - Throughput

advertising larger windows, (2) compare quantitatively the hard ACK and soft ACK strategies.

Larger Window Advantage.

Figure 16 shows simulation results that compare the throughput efficiencies achieved when the end-to-end window size is the same as the receiving buffer size, and when larger windows are used with a hard ACK strategy. The window size is fixed at $2.5 \times w_a$ (normalized end-to-end propagation delay). The vertical axis represents the throughput efficiency and the horizontal axis is the logarithmic value of random packet loss probability. Clearly, by advertising larger windows than the actual buffer size, significantly higher throughput is achieved. This holds for the whole range of loss probability studied and for buffer size as small as 1/4 of the bandwidth-delay product. In particular, over 70% higher throughput is achieved by advertising the larger window for a buffer size of $0.25 \times w_a$ when loss probability is 10^{-5}.

The key advantage of advertising larger windows is that it allows the sender to continue transmitting data to fully utilize the connection. Although a segment may be dropped when it finds no buffer space upon arrival at the receiver, such an event is not expected to occur very often because data will normally arrive successfully in sequence at the receiver and get consumed immediately, thus freeing the buffer space.

Hard ACK versus Soft ACK.

Figure 17 compares the throughput performance between the hard ACK and a soft ACK strategy. At any time, the end-to-end window size is fixed at $2.5 \times w_a$ and only buffer sizes are varied. The results from the hard ACK strategy are plotted using lines while the soft ACK results are shown as discrete symbols.

From Figure 17 we see that when buffer size is equal to the window size, that is $2.5 \times w_a$, the two strategies perform exactly the same (the solid line and the circles match) because segments are not discarded at the receiver due to lack of buffer space. But as the actual buffer size decreases, performance of the soft ACK strategy deteriorates dramatically. The most significant drop in throughput occurs when the buffer size is reduced to $0.5 \times w_a$. The performance difference between the two strategies remains significant until the loss probability approaches 10^{-1}, by that time the hard ACK performs as bad as the soft ACK due to the tremendous loss. The simulation results also show that the maximum segment delay for the hard ACK scheme is much lower than that for the soft ACK. The soft ACK strategy was found to be susceptible to an interesting abnormal behavior that periodically saves and then discards sequences of segments, which explains the poor performance observed.

References

[1] Benes, V., "Programming and Control Problems Arising from Optimal Routing in Telephone Networks," *Bell System Technical Journal*, vol. 45, pp. 1373 - 1439, 1966.

[2] Bhargava, Deepak, "Synchronization and Collaboration in Multimedia Applications," MS Thesis, Department of Computer Science, Washington University. Expected December 1992.

[3] Bianchi, Giusseppe and Jonathan Turner. *Improved Queueing Analysis of Shared Buffer Switching Networks*, Washington University Computer Science Department, WUCS-92-19.

[4] A.D. Bovopoulos, R. Gopalakrishnan and S. Hosseini, "SYMPHONY: A Hardware, Operating System and Protocol Processing Architecture For Distributed Multimedia Applications," Technical Report WUCS-93-06, Department of Computer Science, Washington University, 1993.

[5] Bovopoulos, A. D. and Lazar, A. A., "The Effect of Delayed Feedback Information on Network Performance," *Annals of Operations Research*, Vol. 36, 1992, pp. 101-124. 1992,

[6] Kumar, L. and Bovopoulos, A. D., "An Access Protection Scheme for Heavy Load Unfairness in DQDB," *Proceedings of the IEEE INFOCOM'92 Conference*, May 6-8, 1992, Florence, Italy, pp. 190-199.

[7] Bovopoulos, A. D., "Performance Evaluation of a Traffic Control Module for ATM Networks," *Proceedings of the IEEE INFOCOM'92 Conference*, May 6-8, 1992, Florence, Italy, pp. 469-478.

[8] Bovopoulos, A. D., "Optimal Burst Level Admission Control in a Broadband Network" *STOCHASTIC MODELS*, Vol. 8, No 2, 1992, pp. 289- 305, and presented at *The First International Conference of Computer Communications*, San Diego, California, June 1992.

[9] Kumar, L. and Bovopoulos, A. D., "A Protocol for Dynamic Assessment of Network Topology in DQDB MANs," *Proceedings of the First International Conference of Computer Communications*, San Diego, California, June 1992.

[10] Buddhokot, M.M., Kapoor, S., Parulkar, G.M., "Simulation Analysis of a Two Port ATM-FDDI Gateway," Technical Report WUCS-92-36, Department of Computer Science, Washington University, 1993.

[11] Clark, David D., Mark L. Lambert, and LiXia Zhang, "NETBLT: A High Throughput Transport Protocol", *SIGCOMM '87 Symposium: Frontiers in Computer Communications Technology (Computer Communication Review)*, Vol. 17, No. 5, ACM, New York, 1987, pp. 353–359.

[12] Cox,Jr., J.R., Gaddis, M.E., Turner, J.S., Project Zeus: Design of a Broadband Network and its Application on a University Campus," Technical Report WUCS-92-52, Department of Computer Science, Washington University, 1993.

[13] Cranor, C., "An Implementation Model for Connection-oriented Internet Protocols," MS thesis, Department of Computer Science, Washington University in St. Louis, May 1992.

[14] Cranor, C., Parulkar, G.M., "An Implementation Model for Connection-oriented Internet Protocols," *Proceedings of the IEEE Conference on Computer and Communications, INFOCOM'93*. Also, Technical Report WUCS-92-16.

[15] Cruz, R.L., "The Statistical Data Fork: A Class of Broadband Multichannel Switches," *IEEE Transactions on Communications*, to appear.

[16] Doeringer, Willibald A., et al., "A Survey of Light-Weight Transport Protocols for High-Speed Networks", *IEEE Trans. Communications*, Vol. 38, No. 11, November 1990, pp. 2025–2039.

[17] Doshi, B. T., et al., "Error and Flow Control Performance of a High Speed Protocol", to be published, 1991.

[18] Fingerhut, A., "Algorithms for Designing Nonblocking Communication Networks with General Topologies" Technical Report WUCS-TM-92-05 Washington University, St. Louis, Missouri, 1992

[19] Fingerhut, Andy. "Designing Communication Networks with Fixed or Nonblocking Traffic Requirements," Washington University Computer Science Department, WUCS-91-55.

[20] Gong, Fengmin "A Transport Solution for Pipelined Network Computing," Washington University Computer Science Department, D.Sc. dissertation, December 1992.

[21] Gong, F., C. Papadopoulos and G. M. Parulkar, "A Novel Interprocess Communication Paradigm for Pipelined Televisualization," Workshop on Computer Graphics in the Network Environment, SIGGraph'91, July 28 - 29, 1991, Las Vegas, Nevada.

[22] Gong, F. and G. M. Parulkar, "Segment Streaming for Efficient Pipelined Televisualization," *Proceedings of the IEEE Military Communications Conference, MILCOMM'92.*

[23] Gong, Fengmin, Z. Dittia and G. M. Parulkar, " A Recurrence Model for Asynchronous Pipeline Performance Analysis", Technical Report WUCS-91-14, Department of Computer Science, Washington University (also submitted for publication).

[24] Gong, F. and G. M. Parulkar, "An Application-oriented Error Control Scheme for A High Speed Transport Protocol," Technical Report WUCS92-37, Department of Computer Science, Washington University in St. Louis. with F. Gong. Submitted for publication.

[25] Gong, F. and G. M. Parulkar, "A Two Level Flow Control Scheme for A High Speed Transport Protocol," Technical Report WUCS92-38, Department of Computer Science, Washington University in St. Louis. Submitted for publication.

[26] R. Gopalakrishnan and A.D. Bovopoulos, "Design of a Multimedia Applications Development System," Technical Report WUCS-92-27, Department of Computer Science, Washington University, 1992.

[27] R. Gopalakrishnan, and Andreas D. Bovopoulos "A Protocol Processing Architecture for Networked Multimedia Computers" Technical Report WUCS-93-07, Department of Computer Science, Washington University, 1993.

[28] M. Guillemont, "Microkernel Design Yields Real Time in a Distributed Environment," Technical Report CS/TR-90-65.1, Chorus Systems, 1991.

[29] Jaffe, J.M., Moss, F.H., and Weingarten, R.A., "NA Routing: Past, Present, and Possible Future," *IBM Sys. J.*,, Volume 22, No. 4, Dec, 1983, pp. 417-434.

[30] 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.

[31] Jordan, B. W., Jr. and G. M. Masson. "Generalized Multistage Connection Networks," *Networks 2*, 1972, 191–209.

[32] M. Hayter and D. McAuley, "The Desk Area Network," *ACM Operating Sytems Review*, October 1991.

[33] R.G. Herrtwich, "An Introduction to Real-Time Scheduling," Technical Report TR-90-035, International Computer Science Institute, Berkeley, California, July 1990.

[34] Lee, C. Y. "Analysis of Switching Networks," *Bell Systems Technical Journal*, 1955.

[35] Liew, S., and Lu, K., "Performance Analysis of Asymmetric Packet Switch Modules with Channel Grouping," *Proceedings of IEEE INFOCOM '90*, June, 1990, pp. 668-676.

[36] Lin, A., and Silvester, J.A., "Fixed-Node Routing and Its Performance in ATM Networks," *Proceedings of IEEE INFOCOM '90*, June, 1990, pp. 803-810.

[37] Mazraani, T.Y., Parulkar, G.M., "Performance Analysis of the Ethernet under Conditions of Bursty Traffic," *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOMM'92.*

[38] Melen, Riccardo and Jonathan Turner. "Distributed Protocols for Access Arbitration in Tree-Structured Communication Channels," *IEEE Transactions on Communications*, 3/91.

[39] 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.

[40] Netravali, Arun N., W. D. Roome, and K. Sabnani, "Design and Implementation of a High-Speed Transport Protocol", *IEEE Trans. on Communications*, Vol. 38, No. 11, November 1990, pp. 2010–2024.

[41] S.W. O'Malley, L.L. Peterson, "A Dynamic Network Architecture," Technical Report, Departement of Computer Science, University of Arizona, Tucson, 1992.

[42] Pattavina, Achille and Roberto Monterosso. "A Vector Model for Analysis of Buffered Switching Networks," CEFRIEL technical report, 1991.

[43] Papadopoulos, C., "Remote Visualization on a Campus Network," MS thesis, Department of Computer Science, Washington University in St. Louis, May 1992.

[44] Papadopoulos, C. and G. M. Parulkar, "Experimental evaluation of SUNOS IPC and TCP/IP Protocol," *Proceedings of IEEE INFOCOM 93*, and IEEE/ACM Transactions on Networking (to appear).

[45] Pattavina, A., "Multichannel Bandwidth Allocation in a Broadband Packet Switch," *IEEE Journal of Selected Areas in Communications*, Volume 6, No. 9, Dec, 1988, pp. 1489-1499.

[46] Pippenger, Nick. "On Crossbar Switching Networks." *IEEE Transactions on Communications*, 6/75.

[47] Rayes, M.A., Min, P.S. and Hegde, M. V., "Analysis of State Dependent Routing (SDR) and Modified SDR," *Technical Report*, Department of Electrical Engineering, Washington University. Also, submitted *IEEE Transactions on Communications*, October 1992.

[48] A. Sah, D.C. Verma and V.G. Oklobdjiza, "A Study of I/O Architecture for High Performance Next Generation Computers," Technical Report TR-91-008, ICSI, Berkeley, January 1991.

[49] Sasaki, Galen H. and Bellur, Bhargav R. "Buffered Statistical Data Fork," *Conference on Information Sciences and Systems, Johns Hopkins University*, Dec, 1988, pp. 33-38.

[50] D.C. Schmidt, D.F. Box and T. Suda, "ADAPTIVE A Flexible and Adaptive Transport System Architecture to Support Multimedia Applications on High-Speed Networks," Technical Report 92-46, Department of Information and Computer Science, University of California, Irvine, 1992.

[51] Seyyed M-R Mahdavian and Andreas D. Bovopoulos, "Virtual Loss of Multiplexed ATM Cell Streams," Technical Report WUCS-93-08, Department of Computer Science, Washington University, 1993

[52] Sterbenz, J.P.G., Kantawala, A., Buddhikot, M., Parulkar, G.M., "Hardware Based Error and Flow Control in the Axon Gigabit Host-Network Interface," *Proceedings of the IEEE Conference on Computer and Communications, INFOCOM'92.*

[53] 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.

[54] Turner, Jonathan, "Managing Bandwidth in ATM Networks with Bursty Traffic," *IEEE Network Magazine*, vol. 6, no. 5, September 1992, 50–58.

[55] Turner, Jonathan, "Bandwidth Management in ATM Networks Using Fast Buffer Reservation," *Proceedings of the Australian Broadband Switching and Services Symposium*, 7/92.

[56] Turner, Jonathan. "Queueing Analysis of Buffered Switching Networks," *Proceedings of the International Teletraffic Congress*, 6/91.

[57] Valdimarsson, Einir, "Blocking in Multirate Networks," *Proceedings of Infocom*, 4/91.

[58] Witte, E., "The Clos Network as a Multirate Distributor with a Greedy Routing Algorithm," Washington University Computer Science Department, WUCS-92-13.

[59] Yang, Yuanyuan and Gerald Masson. "Nonblocking Broadcast Switching Networks," *IEEE Transactions on Computers*, 1991.

[60] Zegura, E., "A Quantitative Comparison of Architecutres for ATM Switching Systems." Presented at the 3rd Workshop on Very High Speed Networks. Greenbelt, Maryland, March 1992.

[61] Zegura, E., "The Clos Network as a Multirate Distributor with a Greedy Routing Algorithm." Technical report WUCS-92-13, Department of Computer Science, Washington University.

[62] Zegura, E., "Research Proposal: Design and Analysis of Practical Switching Networks." Technical memo WUCS-TM-92-2, Department of Computer Science, Washington University.