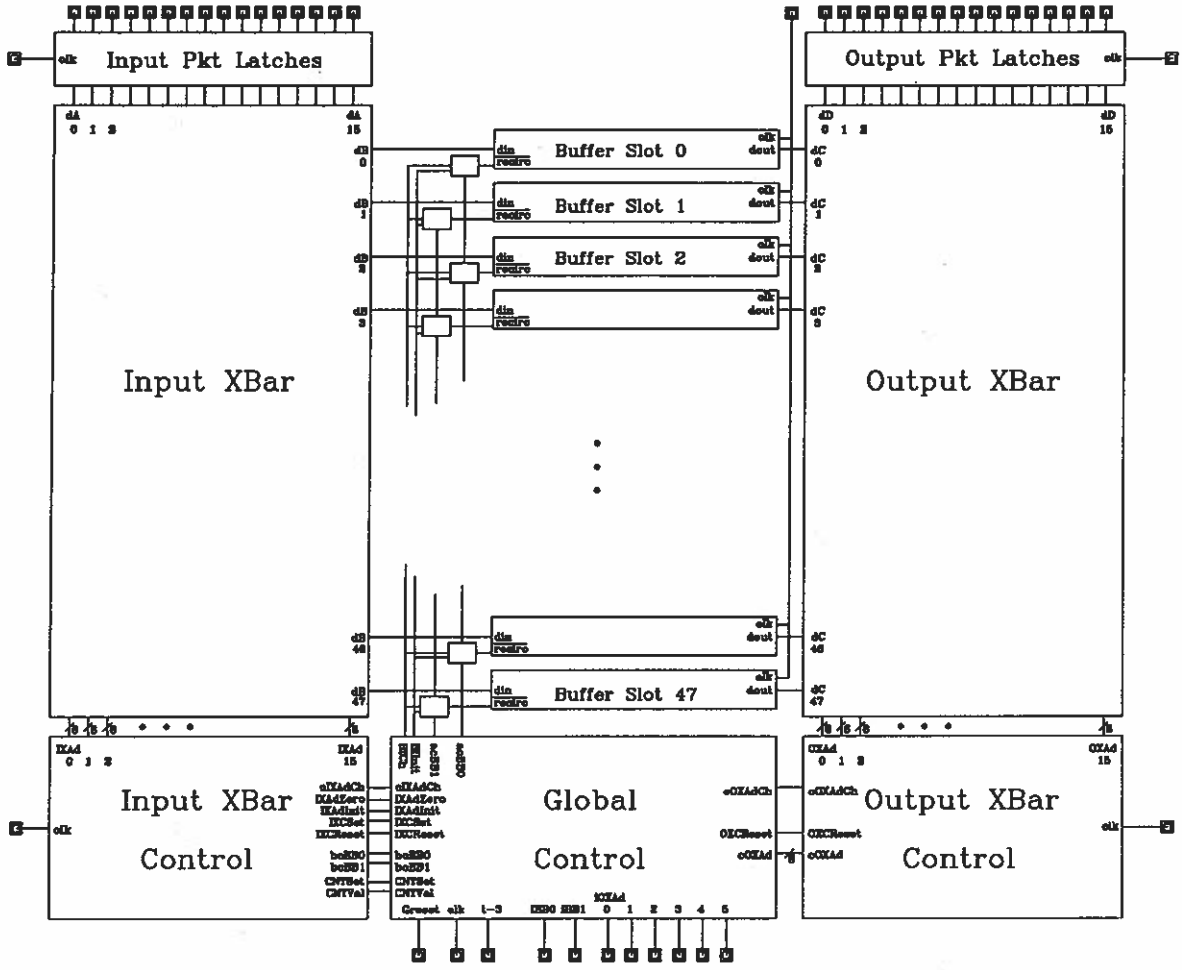
Advanced Networking Research

December 1991



Computer and Communications Research Center
Washington University, St. Louis

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. 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 are developing 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, image and video compression and design of specialized computer-aided design tools. Our research program includes a strong experimental component, which currently centers on the development of a prototype fast packet switching system supporting link speeds of 100 Mb/s and multicast. We have developed five integrated circuits to be used in this prototype system and plan to assemble a network of four switches to demonstrate applications of fast packet switching. 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
Guru Parulkar

Doctoral Students

Akira Arutaki
Millind Buddhikot
Andy Fingerhut
Larry Gong
Saied Hosseini
Seyed Mahdavian
Nader Mirfakhraei
Gopal Raman
James Sterbenz
Einir Valdimarsson
Ellen Witte

Masters Students

James Anderson
Charles Cranor
Apostolos Dailianas
Zubin Dittia
Rex Hill
Sanjay Kapoor
Lakshmana Kumar
Christos Papadopoulos

Our research program enjoys support from the following sponsors.

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

We thank all our sponsors for their collaboration and support.

Executive Summary

This is the sixth progress report for the Advanced Networks Group and covers calendar year 1991. It has been another busy year with good progress in several different areas. During the past year, the Advanced Networks Group has produced three journal articles, six conference papers, eleven technical reports, three MS theses, two doctoral theses and has applied for four patents.

The new students joining the group this year inlcude Saied Hosseini, Seyed Mahdavian, Gopal Raman and Apostolos Dailianas. All four are working with Andreas Bovopoulos. Jim Anderson, Sanjay Kapoor and Lakshmana Kumar all completed their masters degrees and left the group as have Victor Griswold and James Sterbenz who completed their doctoral degrees.

The Applied Research Laboratory, now headed by Dr. Jerome Cox, has continued in the past year to develop an experimental ATM network, which will shortly include four nodes deployed around the St. Louis area to demonstrate a wide range of applications including video distribution, video teleconferencing and medical imaging. This project, which grew out of the Advanced Network's Group's research program continues to be an important part of the overall networks research activity. Washington University is now seeking sponsors to initiate a major new effort, called Project Zeus, which seeks to develop and deploy ATM technology within a campus network. The plans for Project Zeus are summarized briefly below. ARL has been working with a major communications equipment manufacturer to help them develop a commercial version of the broadcast packet switch that can be used in Project Zeus and is working to develop similar relationships with other vendors.

The Advanced Networks Group's research agenda continues to focus on three primary areas: (1) design and analysis of switching systems, (2) internetworking and end-to-end protocol issues and (3) network performance and traffic engineering. The short articles that follow cover a variety of specific topics; Project Zeus, our planned campus ATM network; a quantitative evaluation of switching networks that compares different networks based on performance and cost; a more accurate method of analyzing the performance of buffered switching networks with shared buffer switch elements and flow control between stages; the design of a general purpose simulation system for evaluating switching system performance; a method for congestion control in ATM networks using a technique called fast buffer reservation; algorithms for designing low cost networks, given a set of traffic requirements and constraints on equipment location and link routing; design and implementation of a kernel-based implementation of the McHip internet protocol; design and evaluation of a pipelined televisualization of biological organisms; simulation and evaluation of the Axon host-network interface architecture; a model for analyzing the performance of general acyclic asynchronous processing networks; exact analysis of a traffic control module for ATM networks; traffic engineering of the primary statistical multiplexing stage in ATM networks; design and analysis of a protocol that solves the unfairness problem present in DQDB networks. More detailed accounts of these activities appear in the appendix.

We have just completed the first full year of our *Industrial Partnership Program* which replaced the prior ACS consortium. We currently have seven members of the IPP, providing $320,000 per year. Funding from the National Science Foundation has brought the total to approximately $470,000. While this is a reasonably healthy funding level, the search for additional funding continues apace. Guru Parulkar's current NSF grant expires this summer and he is seeking support for a new research initiative, together with Dave

Richards, a faculty member in electrical engineering. The membership of several of our IPP sponsors is up for renewal this year and we will be talking with each of you about continuing your support. We also continue to look for new sponsors in the program. We are hoping in the coming year to expand the IPP membership to ten.

Project Zeus

Jerome Cox, Jr. and Jonathan Turner

During the last several years there has been a growing recognition that fast packet switching technology (also known as *Asynchronous Transfer Mode* or ATM) will form the basis of next generation communication networks. One attractive aspect of ATM technology is its inherent scalability, both in the total throughput a network can support and the port data rates. While much of the focus in ATM has been on public network applications, most people now agree that the demand for these new networks will come from computer-based applications needing higher bandwidth than current shared-access LANs are able to deliver. LAN and workstation vendors are recognizing the need to introduce switching within campus networks to expand their capacity and range of applications, and are now moving aggressively to develop products to fill this need.

Washington University has been deeply involved in the development of ATM switching technology and its application to medical imaging. We now propose to work with a variety of industrial partners to create commercial implementations of the technology, to apply that technology throughout the university community for the benefit of users and to answer several pressing system questions that can only be addressed in an operational network environment.

Figure 1 illustrates the concept behind the proposed ATM network. The system would consist of several switches on each of the university's two campuses. The switches would be connected by transmission links operating at speeds of 155 Mb/s and 620 Mb/s. Each switch would support potentially several hundred interfaces, with a variety of port speeds. We expect the majority of ports to be 155 Mb/s but will support higher speed ports as the need for them arises. These interfaces could be connected directly to multimedia workstations and central compute servers or could be connected indirectly through shared access LANs such as Ethernet or FDDI. Video would play a central role in the network, allowing access to centrally stored video information through the network, two-way or multipoint video conferencing and remote classroom instruction using video.

The network will include connections to remote sites using either dedicated or switched channels provided by the local exchange carrier or interexchange carriers. In particular, connection to new broadband services planned by Southwestern Bell would make possible classrooms, medical offices and hospitals, all at locations more convenient to their clientele, and all linked to the university or the medical center by video, high resolution image transmission and shared databases. Connection to interexchange carriers would allow scientists to interact with their colleagues at other institutions and with distant supercomputers via the emerging National Research and Education Network.

A goal of Project Zeus is to explore possibilities which may transform daily practice in a number of application areas and, at the same time to conduct experiments useful in understanding the future bandwidth requirements of these application areas. Those of us familiar with network technology wish to work with scientists and scholars from a broad range of disciplines to increase our ability to generalize the experimental results to new areas.

Many possible applications of the proposed ATM network have been discussed with our colleagues at Washington University. Four seem particularly appropriate for the initial experiments with Zeus network technology. These applications are described briefly in the following paragraphs.

Medical Imaging and Electronic Radiology. Applications of broadband network technology in medicine are particularly

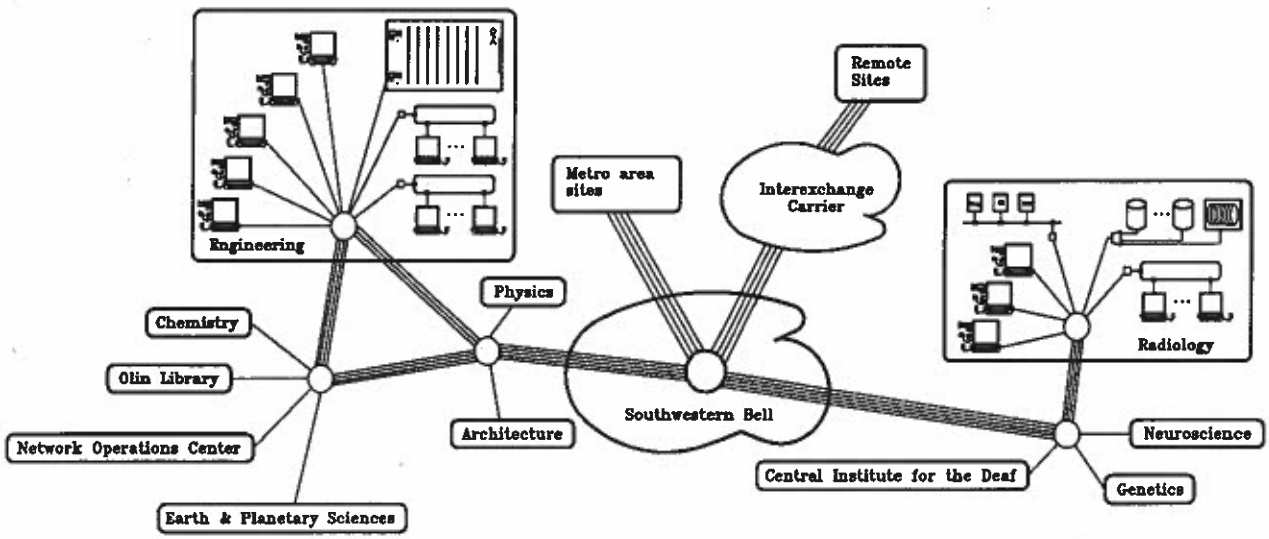


Figure 1: A Fast Packet Campus Network

attractive because of the extensive use of medical images in diagnosis and treatment, because of the need for prompt decision making and because of the promising results of early experiments, particularly in the field of radiology.

Optical Sectioning Microscopy. The optical sectioning microscope analyzes a series of 2D images obtained at different focal planes of a live organism and displays the organism's 3D structure. This new instrument promises to allow biologists to investigate a variety of fundamental, and previously insoluble, problems.

Earth and Planetary Sciences The Remote Sensing Laboratory in the Department of Earth and Planetary Sciences at Washington University houses the lead Geosciences Node of the NASA Planetary Data System. The node is responsible for working with the Magellan and Mars Observer missions to ensure that the data acquired from spacecraft exploring the solar system are properly documented and archived. These data are primarily in the form of that, in most cases, arrive over the Internet.

Visualization in Art and Architecture. The Urban Research and Design Center in the School of Architecture has identified a research agenda which addresses issues regarding the development of a designer's workstation. The design of buildings, urban places and cities is augmented through multimedia and 3D models that are linked to image collections, graphics and art.

Another goal of Project Zeus is to provide a realistic testbed suitable for communications research on pressing questions of network design and operation. Laboratory experiments, simulations and demonstrations all provide answers to some communications research questions, but until thousands of users with hundreds of applications test a network, questions that depend on real traffic and service patterns remain unanswered.

Network Congestion. Congestion control is an essential feature of networks with stochastically variable traffic. Evaluation of congestion control methods requires testing in a realistic testbed. Network instrumentation for this and other related purposes will be a key component of the Project Zeus network design.

Efficient Routing. The Project Zeus network will support general multicast

communication, in addition to point-to-point communication. Efficient algorithms for multicast connection routing have been studied by our group, and final evaluation of these algorithms will be undertaken in the context of the Zeus network.

Network Planning and Configuration. Switched networks require more attention to network capacity planning than the shared access LANs currently used in campus networks. We are working to create a general software tool that will support network planning and configuration, allowing detailed consideration of network expansion alternatives, taking into account traffic requirements, physical restrictions on cable and equipment placement as well as installation and maintenance costs.

Interoperability. The fast packet switching technology based on the ATM standard will be deployed both in campus networks and in public broadband networks. The trade offs for these two scenarios are different, and thus, can lead to differences in ATM signaling protocols. The success of ATM critically depends on the interoperability of different switches and associated signaling protocols.

Internetworking. The existing communication environment is best characterized as an internet consisting of a number of low speed networks interconnected by gateways. Extending the internet model to accommodate ATM subnetworks is essential to the successful incorporation of ATM into existing campus networks. We are developing a connection-oriented internet protocol that will fully exploit the capabilities of new high speed networks in the internet environment.

Project Zeus is organized in three phases. Phase 0, now underway, seeks to demonstrate feasibility of the core technology, provide a basis for a more complete design and provide a testbed for application development. The network created in this phase will be primarily an experimental vehicle, rather than an operational network supporting real users. This phase of the project began in 1988 and will continue through early 1992.

Phase 1, scheduled to begin at the start of 1992 and run through 1994, will create all the key components needed to establish an ATM campus network and provide extensive support for application development. When complete, the phase 1 network will be an operational system supporting a variety of users in key departments within the university.

During phase 2, which will run from 1994 through 1996, we plan to expand the range of interfaces that can be used to access the network, construct components for larger scale networks and reduce the cost of key network components. The phase 2 network will support users in all departments of the university.

More details on Project Zeus can be found in [9].

Quantitative Evaluation of Switching Networks

Ellen Witte

There have been a number of architectures for ATM switching systems proposed in the literature with extensive performance data, but little in the way of comparison to indicate which architectures are preferable from a cost standpoint given specific performance requirements. Any reasonable architecture can be configured to provide a given level of performance, but the associated costs can be quite different. In this study, we compare networks on the basis of both cost and performance, to determine which architecture provides a specified level of performance for the lowest cost. To measure cost we count the number of chips needed to realize the architecture, taking into account both pin constraints and device density. We have chosen chip count because it is a dominant component in the cost of a switching system.

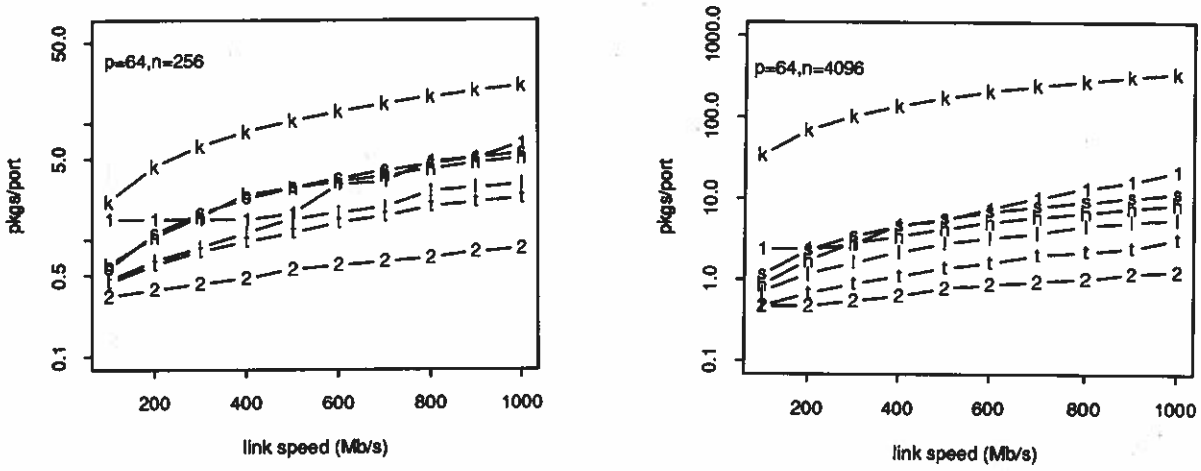
We use the term *switching system* to refer to the functional unit that interconnects the external data links. The switching system is responsible for receiving packets from external links, routing them as appropriate and transmitting the packets on external links. Within the switching system there is a *network* or *switching fabric* that performs the actual routing function. Many of the networks we consider are constructed by interconnecting multiple copies of some smaller building block. We use the term *switch element* to refer to the smaller building block.

We are interested in differences between switching systems based on architectural choices as opposed to details of implementation. Thus, we consider several broad categories of systems based on high level architecture choices. Within each category we consider one or more alternatives and develop an equation for the chip count for each alternative. These equations are used to make plots of chip count for each network over a range of parametric values. We then compare the chip counts of the various networks. Clearly the chip count depends on the strategy used to assign components of the system to chips. For each architecture, we develop a packaging strategy using as few chips as possible for that architecture, within the constraints on package size and transistor count for the chip. In considering the chip count, we focus on the switching network of each switching system and ignore the input and output circuits which interconnect the external data links. This is done on the grounds that the input and output circuit complexity is comparable for each of the networks.

In this study, we have considered only point-to-point networks. The following networks were examined:

- *Crossbar networks.* In particular, we consider the Knockout architecture in some detail.
- *Sorter Based Networks.* In particular, we study the Sunshine network and Lee's hybrid network, which includes a number of sorter based modules which feed into Knockout type output concentrators.
- *Unbuffered Networks with Deflection Routing.* In this category, we consider Tobagi's Tandem Banyan network and the Shuffleout network of Dècina, et. al. In both cases, we consider configurations with recirculation, which provide the least cost for a given performance level.
- *Buffered Beneš Networks.* Here there are many possible variations. We studied in particular, a network with fixed path routing and output buffering, and a second network with per cell routing and shared buffering.

For each of the networks, a configuration was chosen that allows the network to be essentially nonblocking and have acceptably low cell loss



k: Knockout
s: Sunshine
l: Lee's network
t: Tandem Banyan
h: Shuffleout
1: buffered Beneš – fixed path/output
2: buffered Beneš – per cell/shared

Figure 2: Comparison of Network Architectures

rates. In some cases, this requires a speed advantage for the network's internal data paths that is typically realized through added parallelism within the network.

Figure 2 shows a comparison of the networks studied for configurations with 256 inputs and outputs (left side) and networks with 4096 inputs and outputs (right side). In both plots, each integrated circuit was constrained to have no more than 64 inputs and 64 outputs and the transistor count per package was limited to 500,000. The plots show how the chip counts grow as a function of the speed of the switching system's external data links. Notice that the y-axis is logarithmic and gives the number of chips per input and output. Note that in the larger system, the various systems have costs that differ by more than two orders of magnitude. Even excluding the Knockout, which is very poor in large configurations, there is a striking difference among the various alternatives.

More details on this work can be found in [47].

Improved Queueing Analysis of Buffered Switching Networks

Jonathan Turner

Reference [39], analyzes the queueing performance of switching networks comprising switches with shared buffering and flow control. This analysis leads to a fast computational procedure for determining the delay and throughput of such networks.

We model each switch in the network as a $B+1$ state Markov chain. We let $\pi_i(s)$ be the steady state probability that a stage i switch contains exactly s packets and we let $\lambda(s_1, s_2)$ be the probability that a switch with s_1 packets during a given cycle contains s_2 packets in the subsequent cycle. Let $p_i(j,s)$ be the probability that j packets enter a stage i switch that has s packets in its buffer and let $q_i(j,s)$ be the probability that j packets leave a stage i switch that has s packets in its buffer. Then

$$\lambda_i(s_1, s_2) = \sum_h p_i(h, s_1) q_i(h - (s_2 - s_1), s_1)$$

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

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

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

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

Y is easily calculated, assuming all distributions of s packets to the d outputs are equally likely. We compute performance parameters by assuming a set of initial values for $\pi_i(j)$, then use the equations given above to compute $\lambda_i(s_1, s_2)$. These, together with the balance equations for the Markov chain are used to obtain new values of $\pi_i(j)$ and we iterate until we obtain convergence.

In this analysis, we represent the state of a switch by the number of packets it contains and *assume* that the stored packets are equally likely to be destined for any of the switch's outputs. This assumption is used in the equation for a_i and again in the equation for $Y_d(r,s)$. This assumption ignores the correlations between packet destinations that develop as packets contend with one another. Comparing the results of analysis with simulation, we have identified conditions under which the analysis overestimates a network's maximum throughput by as much as 30%.

Pattavina and Monterosso [30] call the above model the *scalar model* and have proposed instead, a *vector model* in which the state of a switch is represented not by the number of stored packets, but by a vector containing the number of packets for each destination. The vector model is exact for a single stage network and is reasonably accurate for multistage networks as well. On the other hand, the state space grows exponentially with the size of the switches, making it applicable only to networks with up to four ports per switch.

We have developed an alternative scalar model that seeks to match the accuracy of the vector model while avoiding its computational complexity. This model is based on the observation that when a switch is in the steady state, the average number of arriving packets destined for a particular switch output port equals the average number of packets departing via that output port. When the original scalar model is compared to the vector model, it's easy

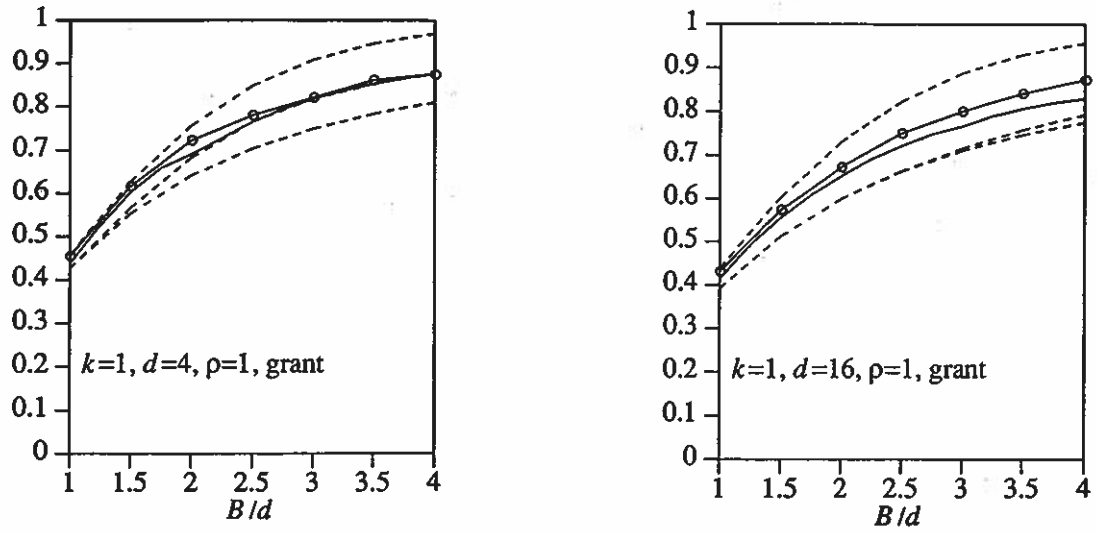


Figure 3: Comparison of Alternative Queueing Analyses

to see that the scalar model in effect, implicitly assigns probabilities to the vector model states according to a multinomial distribution (when $d = 2$, it is a binomial distribution). However, the observation concerning the balance of arriving and departing packets suggests that the probabilities of these states should be approximately equal. This leads to an alternative scalar model in which uniform probabilities are assigned to these states.

The *uniform scalar model* yields good results for networks with large buffers ($B > d^2$), relative to d, precisely the case where the original scalar model was least accurate. For these cases, the predicted throughput is generally within 5% of that predicted by simulation. However, substantial inaccuracies remain in the practically important case of large d and $B/d \leq 4$. The problem here is that for such switches, boundary states (states in which there are some outputs for which there are no packets in the buffer) are very common, but occur with lower probability than is assigned to them by the uniform scalar model.

To compensate for this, we have augmented the uniform scalar model, by the addition of one additional bit of information (roughly doubling the number of states needed to represent a switch). This bit is used to distinguish between boundary and nonboundary states; within each group, probabilities are assigned uniformly, as before. This *boundary method* yields dramatic improvements for switches with 2 or 4 ports, but only a slight improvement for 16 port switches. This is because, in large switches the probability of being in a boundary state is so overwhelmingly large that we gain little information by distinguishing boundary from nonboundary states. Consequently, we have developed a *threshold method*, which uses the extra bit of information to distinguish states in which the number of outputs for which there are packets is above or below a given threshold. With this approach, we can predict throughputs of networks with large switches accurately (within a few percent of simulation results) and quickly.

Figure 3 contains plots of maximum throughput as a function of the size of the shared buffer, for two networks with 256 ports ports each. The plot on the left is for a network constructed from four port switch elements and the one on the right is for a network with 16 port switch elements. The curve identified by the plotted circules was obtained from simulation and the other solid curve was obtained using the threshold analysis. The dotted curves are for the earlier analyses, with the top curve being the original scalar method, the bottom curve, the uniform scalar method, and the middle dotted curve being the boundary method.

There are two primary directions in which this

work can be developed. One is to develop still more accurate models of single stage switches, while maintaining the computational efficiency of the scalar models. The second is to model the correlations between packet destinations for packets in adjacent switches. For large networks of small switches, this is the largest remaining source of inaccuracy in the current models.

Design of a General Purpose Switching System 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. We have initiated work on a simulation tool that will make it possible to simulate a wide variety of different systems with little or no programming on the part of the performance analyst. The performance analyst will be able to specify the components of the system and the way in which they are interconnected, by way of a graphical user interface, with menus from which components can be selected and powerful network construction operators, which provide common interconnection patterns. Once the system is specified, the analyst will then be able to simulate the it using appropriate traffic models, also selected and modified through menus, while monitoring the traffic parameters of interest. The graphical user interface comes into play during simulation as well, allowing the user to observe the operation of the system through a continuous animation and/or through continuous plotting of the desired data. The system will provide several advantages over traditional approaches.

- It will allow the analyst to construct a specific network and traffic configuration with an absolute minimum of effort and verify that the network operates as expected using the animation features.
- It will make it much easier to compare different configurations. Because the different networks are constructed within the same environment, they can be subjected to identical traffic and compared with far greater precision than when simulations are done independently.
- The visualization features are an excellent vehicle for illustrating a system's operation. They are also an excellent way to obtain a detailed understanding of transient behavior.

This work has been inspired in part, by an earlier animation of the broadcast packet switch simulator. This tool, while relatively crude, has proven to be extremely useful for explaining the operation of the system to visitors and for developing an understanding of certain unexpected situations that arose when the prototype system was tested. The new tool provides similar animation features, but provides far more flexibility in how networks are configured, simulated and measured. Figure 4 shows an example of a simulation window containing a simple network with input-buffered switches preceded by a set of traffic sources and input buffers, and followed by a set of output buffers and traffic sinks. The tools menu shown in the figure provides primitives for selecting and instantiating basic components, repositioning them and connecting them together.

A number of pulldown menus provide additional capabilities. The *File menu* allows a given network to be saved or restored from a file and allows the contents of a simulation window to be printed. The *Specify menu* provides a means for changing the parameters

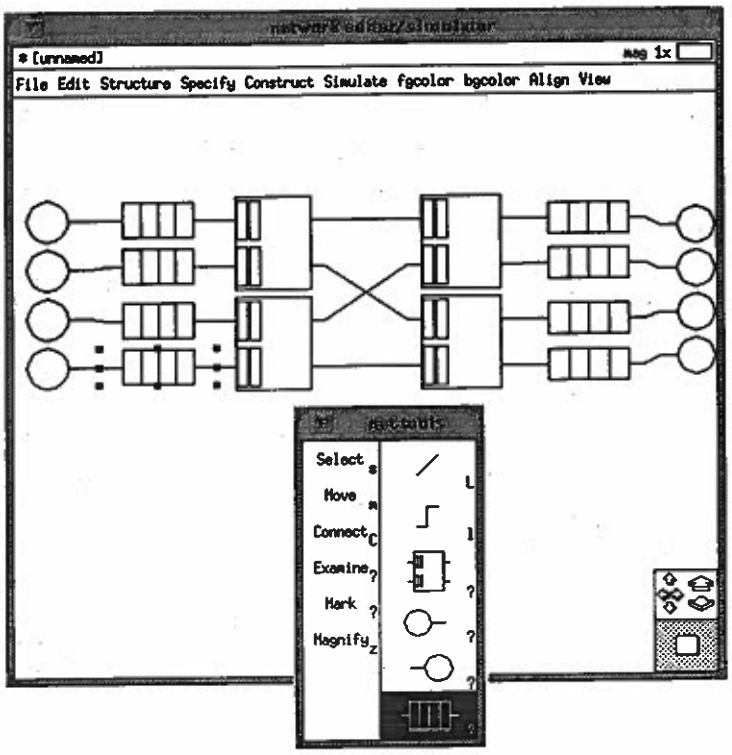


Figure 4: Example Simulation Window

of various components. For example, the number of inputs or outputs of a switch element can be varied, as can the size and placement of buffers (input, output or shared), the queueing discipline (fifo, age, priority, lifo), the type of flow control (none, grant or acknowledgement) and the function (route, distribute, copy). For traffic sources, the peak and average loads and burst length can be varied. For lookup tables, the table size and initial contents can be specified. The *Construct menu* includes options for series or parallel construction of networks, allowing large networks to be specified with just a few steps. The *View menu* controls the visual appearance of the simulation and allows multiple views of the same simulation to be shown. It also provides access to a graph editor, which is used to specify plots which can be attached to variables within the simulator, allowing the user to observe various traffic parameters as the simulation proceeds. The *Simulate menu* provides commands for controlling the simulation and includes both single-step and multi-step commands. It also includes commands for suppressing screen updates during multi-step commands, to speed up system operation.

Using the *Examine* command in the net tools window, information about each of the objects can be examined and modified. For example, if one selects a packet, one gets a dialog box containing information about the packet contents; the contents of any of the packet's fields can be modified through the dialog box, as can its color, making it possible to observe the progress of a particular packet as it passes through the network. Similarly, one can change the load offered by a source or the mapping provided by a lookup table.

Most of the key design issues for the system center on the competing objectives of generality and performance. For example, one issue arises from the question of how to route packets in networks that can be constructed with arbitrary topology. To handle this problem in full generality, each switch might require a

different routing table specifying the switch output to use to reach any given network output. In most common situations, a single table would suffice for all switches in a given stage, but users can certainly construct networks where this would not be the case.

The system is being written in C++ and is based on the InterViews user interface toolkit, which in turn is based on the X-windows system. Each of the graphical objects is implemented as a C++ class and includes member functions for accessing internal state, executing a simulation step and updating its on-screen representation. At this writing, the system is still in a preliminary stage of development. While many of the desired capabilities have been implemented, others are still being developed.

Congestion Control Using Fast Buffer Reservation

Jonathan Turner

A central objective in ATM networks is to provide virtual circuits that offer consistent performance in the presence of stochastically varying loads on the network. This objective can be achieved in principle, by requiring that users specify traffic characteristics when a virtual circuit is established, so that the network can select a route that is compatible with the specified traffic and allocate resources as needed. While this does introduce the possibility that a particular virtual circuit will be blocked or delayed, it allows established virtual circuits to receive consistent performance as long as they remain active.

Ideally, a bandwidth management and congestion control mechanism should satisfy several competing objectives. First, it should provide consistent performance to those applications that require it, regardless of the other virtual circuits with which a given virtual circuit may be multiplexed. Second, it should allow high network throughputs even in the presence of bursty traffic streams. Third, the specification of traffic characteristics should be simple enough that users can develop an intuitive understanding of the specifications and flexible enough that inaccurate specifications don't have seriously negative effects on the user. Fourth, it should not artificially constrain the characteristics of user traffic streams; the need for flexibility in ATM networks makes it highly desirable that traffic streams be characterized parametrically, rather than by attempting to fit them into a pre-defined set of traffic classes. Fifth, it must admit a simple realization for reasons of economy and reliability. Less crucial, but in our view, also important, is the requirement that the bandwidth management mechanism accommodate multicast virtual circuits with multiple transmitters. All proposals we have seen for connection management in ATM networks have serious deficiencies with respect to at least one of these objectives. We describe here an approach using fast buffer reservation which appears to satisfy them all.

To preserve the integrity of user information bursts, the network must detect and track activity on different virtual circuits. This is accomplished by associating a state machine with two states with each virtual circuit passing through a given link buffer. The two states are *idle* and *active*. When a given virtual circuit is active, it is allocated a prespecified number of buffer slots in the link buffer and it is guaranteed access to those buffer slots until it becomes inactive, which is signaled by a transition to the idle state. Transitions between the active and idle states occur upon reception of user cells marked as either *start-of-burst* or *end-of-burst*. Other cell types include *middle-of-burst* and *loner*, the latter is used to designate a low priority cell that is to be passed if there are unused buffer slots available, but which can be discarded if necessary. A forced transition from active to idle is also made if no cell is received on the virtual circuit within a fixed timeout period.

Figure 5 illustrates the buffer reservation mechanism. For virtual circuit i, the mechanism stores the number of buffer slots needed when the virtual circuit is active (B_i), the number of buffer slots used by unmarked cells (b_i) and a state variable (s_i: idle, active). The mechanism also keeps track of the number of unallocated slots in the buffer (B). The detailed operation of the state machine for virtual circuit i is outlined below.

When a start cell is received:

- If the virtual circuit is in the idle state and $B - B_i < 0$, the cell is discarded.
- If the virtual circuit is in the idle state and $B - B_i \geq 0$, s_i is changed to active, a timer for that virtual circuit is set and B_i is subtracted from B. If $b_i < B_i$, b_i is incremented and the cell

is placed (unmarked) in the buffer. If $b_i = B_i$, the cell is marked and placed in the buffer.

If a start or middle cell is received while the virtual circuit is in the active state, it is queued and the timer is reset. The cell is marked, if upon reception, $b_i = B_i$, otherwise it is left unmarked and b_i is incremented.

If a middle or end cell is received while the virtual circuit is in the idle state, it is discarded.

If an end cell is received while the virtual circuit is active or if the timer expires, s_i is changed from active to idle and B_i is subtracted from B.

If a loner is received, it is marked and placed in the buffer.

Whenever a cell is sent from the buffer, the appropriate b_i is decremented (assuming the transmitted cell was unmarked).

When a virtual circuit is routed, the software that makes the routing decisions attempts to ensure that there is only a small probability that the instantaneous demand for buffer slots exceeds the buffer's capacity. This probability is called the *excess buffer demand probability* and might typically be limited to say 1%.

To make this precise, let λ_i denote the peak data rate of a given virtual circuit and let μ_i denote the average rate. If the link rate is R and the buffer has L buffer slots, the number of slots needed by an *active source* with peak rate λ_i is defined to be $B_i = \lceil L\lambda_i/R \rceil$.

Since B_i buffers are allocated to a virtual circuit when it is active, the virtual circuit's instantaneous buffer requirement is either 0 or B_i. If we let x_i be a random variable representing the number of buffer slots needed by virtual circuit i at a random instant then $\Pr(x_i = B_i) = \mu_i/\lambda_i$ and $\Pr(x_i = 0) = 1 - \mu_i/\lambda_i$.

Consider then, a link carrying n virtual circuits with instantaneous buffer demands $x_1, \ldots, x_n$.

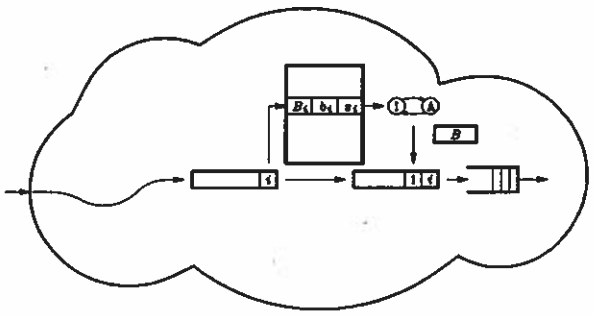


Figure 5: Fast Buffer Reservation

Define $X = \sum_{i=1}^{n} x_i$. Note that X represents the total buffer demand by all the virtual circuits. Suppose we have a new virtual circuit with buffer demand x_{n+1} and we want to decide if it can be safely added to the link. We first must compute, the probability distribution of the random variable $X' = X + x_{n+1}$. This can be obtained by numerical convolution of the distribution of X with the the distribution of x_{n+1}, assuming that the idle, active behavior of the new virtual circuit is independent of the the existing virtual circuits. To decide if the virtual circuit can be accepted we then simply verify that $\Pr\{X' > L\}$ is small. For a link with a 256 slot buffer, we estimate that about 2000 multiplications and 1250 additions are required to compute the distribution of X' and $\Pr\{X' > L\}$. Using fixed point arithmetic, this can be done in less than half a millisecond on a 10 MIPS processor.

To complete the bandwidth management scheme, a traffic monitoring mechanism is required at the user-network interface to ensure that the virtual circuit's long term average data rate does not exceed the value specified at virtual circuit establishment. We have designed an appropriate mechanism of this sort, and generalized it to support multicast virtual circuits with multiple sources. We have studied the implementation complexity of this approach and estimate that the incremental cost of adding bandwidth management hardware to a port controller of an ATM switch to be no more than 10%.

See [40] for further details.

Algorithms for Network Design

Andy Fingerhut

This work concerns the problem of how to design the least cost communication network that meets a given set of traffic requirements and satisfies practical constraints on placement of equipment and cable routing. While this is not a new problem, the ATM environment raises some new considerations. First, statistical methods are of relatively little use, as the traffic characteristics of ATM networks are unknown and in any case are expected to be highly variable and dynamic. This makes design methods based on worst-case analysis more approriate than statistical methods. Second, ATM networks are expected to support multicast virtual circuits, which magnifies the impact of virtual circuit routing algorithms on overall network capacity requirements.

The work has two parts. The first is concerned primarily with finding effective algorithms for network design and analysis. The second concerns design of a practical software tool that can be used for planning an ATM campus network and which uses the network design and analysis algorithms to assist the network planner in creating and evaluating network designs. We have concentrated so far on the first part, creating a general problem formulation and then developing and analyzing algorithms for several special cases.

General Problem Formulation

We have created an abstract formulation of the network design problem in a fairly general way, so as to capture the essential features of the problem while suppressing extraneous detail. The abstract formulation consists of several components. A *physical graph* describes physical constraints on where equipment may be placed. *Equipment descriptions* specify the types and properties of the components which may be used in constructing the network. A *logical graph* describes a way of connecting equipment together to form the network, and an *embedding* specifies where to place equipment and how to route links. *Traffic requirements* tell us how we expect the network to be used.

In detail, a physical graph is an undirected multigraph in which each vertex represents a *place* where switching equipment could be installed (e.g. a wiring closet or computer room) and each place has several parameters such as amount of available space, power availability, and presence of air conditioning. Each edge in the physical graph represents a *link path* where one or more transmission links may be placed, (e.g., a duct or chase in a building, a tunnel between buildings). Each link path has parameters such as length, cross-sectional area and cost of laying cable on this path.

A network is constructed from *switches*, of possibly different types. Each type is characterized by its physical volume, total switching capacity, number and types of links it can terminate, initial cost (both startup and per port), possibly a maintenance cost and power and environmental requirements. A *link* is some type of cable (e.g., twisted pair wire, coaxial or fiber optic cable) which can transmit information. Each type is characterized by cross-sectional area, maximum length over which signals can be carried with amplification or regeneration, data carrying capacity, cost per unit length and termination cost.

The purpose of the network is to connect *terminals* together. Terminals can be sources and destinations of messages in the network. They are characterized by certain traffic requirements which can take a variety of different forms. In the simplest case, traffic requirements are *fixed*; that is for each pair of terminals, we specify the amount of traffic that must be carried between that pair. This can be extended to multicast traffic requirements as well. More generally, we can give a *termination capacity* for a terminal which specifies the

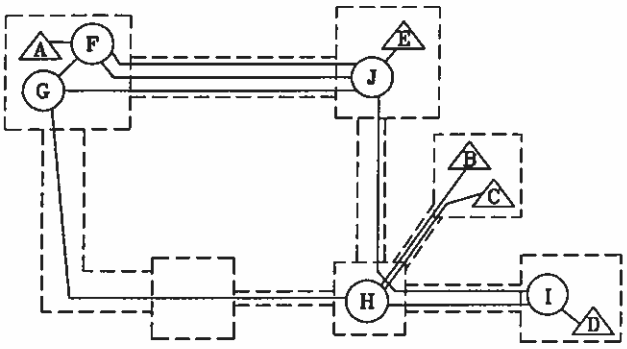


Figure 6: An example embedding

maximum traffic allowed to originate or terminate at that terminal. We then seek to construct a network that will support any fixed traffic requirement that is consistent with the specified termination capacities.

A *logical graph* represents a particular way of connecting switches, links, and terminals together. It does not specify anything about where such items are to be placed. To be *feasible*, a logical graph must satisfy constraints given by the equipment descriptions, e.g., the number and type of links which may be terminated at a particular switch cannot be violated.

An *embedding* is a mapping from a logical graph to a physical graph. It specifies where each piece of equipment is located. Vertices in the logical graph (switches and terminals) are mapped to vertices in physical graph (places). Edges in the logical graph (links) are mapped to *paths* in the physical graph (sequences of link paths).

An embedding is *feasible* if it satisfies constraints such as volume available in places, cross-sectional area available in link paths, and environmental constraints. Taken together, a physical graph, equipment descriptions, a logical graph, and an embedding determine the cost of the network. An example embedding is given in Figure 6. The physical graph is drawn with dashed lines while the logical graph is drawn in solid lines. The labeled circles represent switches and the triangles represent terminals.

The general network design problem can be described as follows: given a physical graph, a set of equipment descriptions, a set of terminals with assigned locations in the physical graph and a set of traffic requirements, create a logical graph and an embedding of that graph that realizes the traffic requirements, satisfies all the given constraints and has the lowest overall cost.

Algorithms and Complexity Results

We have concentrated initially on a number of special cases of the general network design problem formulated above, in order to obtain insight into the general problem and help to identify which parts of the general problem are difficult and which are easy. We also expect that some of the algorithms developed in this study will be useful as "subroutines" in the solution to the general problem.

One important subproblem of the network design problem is to determine if a given set of fixed traffic requirements is routable over a specific logical graph. We have shown that this problem is NP-complete, even for point-to-point traffic. On the other hand, we have devised an efficient algorithm for optimally selecting link capacities in the logical graph for the case where link costs are proportional to the capacity. In the multipoint case, the design problem is NP-complete. However, we have an approximation algorithm for multipoint connections, that produces networks whose cost exceeds that of an optimal network by no more than a factor of two. The design problem is also NP-complete for the case where link costs are independent of capacity, but in this case as well, there is an approximation algorithm that produces solutions with cost no more than twice optimal. No good approximation algorithms are known for the case in which links have both startup and capacity-dependent costs.

We have also studied problems in which a termination capacity is specified for each vertex in a logical graph and we are asked to

determine if when connection requests are received sequentially, over time, it is possible to avoid getting into a situation where a request is denied. Design of such *nonblocking networks* depends in general on the routing algorithm used to respond to connection requests. We have so far considered two types of algorithms. In *fixed path routing*, the route for any given pair of endpoints is fixed and independent of the traffic in the network; while this is too restrictive for general use, it does have some potential applicability and leads to some useful insights. In *constrained distance routing*, we select a path from among those having distance at most k; one specific algorithm of this type selects the shortest available path between a pair of endpoints and blocks if there is no available path with length $\leq k$.

We have shown that the problem of designing a nonblocking network for point-to-point connections and fixed path routing can be reduced to a generalized matching problem, for which efficient algorithms are known. A special case of fixed path routing in which the routes are always *shortest paths* can be solved using a somewhat faster method. For constrained-distance routing, we can say only that determining if a given network is blocking is in NP. While we suspect that this problem is NP-complete, we have not yet been able to prove that, nor do we have any positive algorithmic results to report.

More details on this work can be found in [10].

High Speed Internetworking

Guru Parulkar, James Anderson, Milind Buddhikot, Chuck Cranor, Zubin Dittia, Sanjay Kapoor, Tony Mazraani

We have proposed a very high speed internet abstraction (called VHSI) which can efficiently support guaranteed levels of performance for a variety of applications, and can cope with the diversity of underlying networks [23, 28, 29]. Important components of this abstraction are shown in Figure 7. We continue to make good progress on the research and development of various components of the VHSI abstraction. Progress is summarized for each component of the VHSI abstraction in the following paragraphs:

MCHIP. MCHIP is a novel multipoint congram-oriented high performance internet protocol, equal in status to IP in terms of the protocol hierarchy. The congram service primitive aims at combining the strengths of both classical connection and datagram approaches.

MCHIP includes two types of congrams: user congram (UCon) and persistent internet congram (PICon). UCons provide support for the connection-oriented applications, and PICons for datagram applications. Specification of PICon management was completed during this past year. PICon management involves specification of PICon setup, usage, and termination. We specify PICon management using specification of appropriate data and control packet formats, exchange of control messages, creation and management of various data structures to store state information, and interaction with resource managers to allocate resources for PICons. Details of PICon and UCon management can be found in [1, 2, 23, 24].

We have also made good progress with MCHIP implementation in the kernel of SunOS Unix 4.0 which is summarized in the next section.

Resource Server. The VHSI abstraction provides performance guarantees to applications by preallocating resources to congrams, based on the application needs. However, a number of networks do not do resource management on a per connection basis, and therefore the VHSI abstraction includes resource servers to provide this functionality.

We have completed a simulation study of a resource server for Ethernet with a variety of traffic sources, including Poisson and bursty sources [24]. This model uses a central resource manager in a directly connected gateway. The role of the resource manager is to keep track of all active congrams and their resource usage, and accept or block new congrams depending on resource availability. The resource manager is always consulted before a congram can be established.

The goal of the resource manager is to guarantee performance to established congrams, and to manage the network resources efficiently. Efficient resource management in this environment means maximum utilization of the network and minimum congram blocking, packet loss, and packet delay. Note, however, that minimizing packet delay may result in lower channel utilization. This study however shows that a simple resource management model can be devised to provide bounded packet loss and delay to various applications with reasonable channel utilization and blocking.

We have also made considerable progress on the simulation study of PICon multiplexing. Note that the PICons are provided to multiplex datagram traffic from a number of sources. A datagram source is considered to be a bursty source and its traffic output is characterized by three parameters: burst size, delay between bursts, and a time after which a burst is to be discarded. The simulation study is aimed at understanding how many and what kind of datagram sources can be successfully multiplexed on a given PICon. A number of

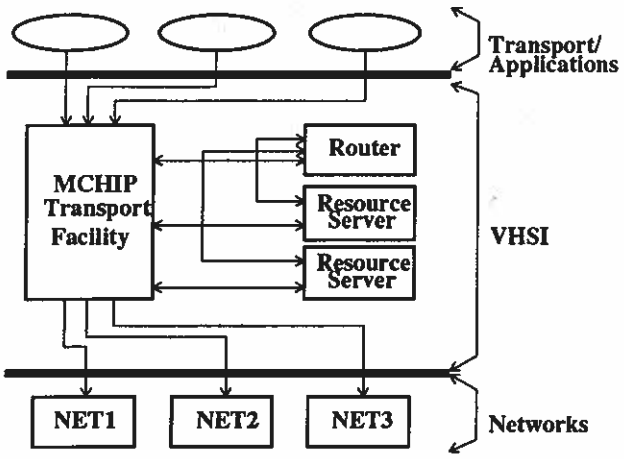


Figure 7: VHSI Abstraction

scheduling schemes have been studied, and a deadline based scheduling algorithm is found to provide the best results under the given conditions. For details of this study, refer to [1].

Gateway Architecture. The VHSI abstraction requires that the gateway architectures be able to support data rates of at least a few hundred Mbps, to interface with diverse networks, and to implement MCHIP without becoming a performance bottleneck. An important part of the gateway design philosophy is to partition the functionality into critical and non-critical paths. The critical path consists of per packet processing, and should be implemented in hardware for speed and performance. The non-critical path consists of congram management and resource and route management and is best implemented in software due to complexity, need to do fine tuning with time, and flexibility. It is inefficient to mix these paths as is generally done in current gateways.

To research high speed gateway architectures, a paper design and a simulation study of a two port ATM-FDDI gateway were undertaken [17, 18]. We believe that ATM and FDDI are excellent target networks to explore the VHSI gateway architecture because they pose the necessary challenges to the gateway designer due to their high data rates and significant diversity.

Design of the ATM-FDDI gateway requires appropriate separation of control and data paths, and design of protocol processors to implement the Segmentation and Reassembly (SAR) Protocol and MCHIP critical path in hardware. The SAR protocol processor implements the SAR protocol that is used in the gateway to reassemble incoming cells from the ATM network into FDDI frames, and also fragment FDDI frames into ATM cells. The MCHIP protocol processor implements some of the MCHIP protocol primitives. It contains channel translation tables and logic to append FDDI specific headers on reassembled frames, and ATM headers on incoming FDDI frames before they are forwarded for fragmentation. Protocol processor designs are sufficient in detail to allow a prototype implementation. A simulation study of the gateway has also been completed which has served two purposes: first to do functional verification of the design, and second, to characterize the performance of the gateway for different traffic patterns.

MCHIP Implementation

Charles Cranor, Guru Parulkar

Our MCHIP implementation essentially consists of two parts: COIP-K and MCHIP modules. COIP-K is the connection oriented internet protocol kernel which includes the common functionality necessary for most connection-oriented internet protocols. The MCHIP modules are protocol specific modules that use COIP-K as a toolkit to create an instantiation of MCHIP. We decided to partition the MCHIP implementation as described above because of the following two observations.

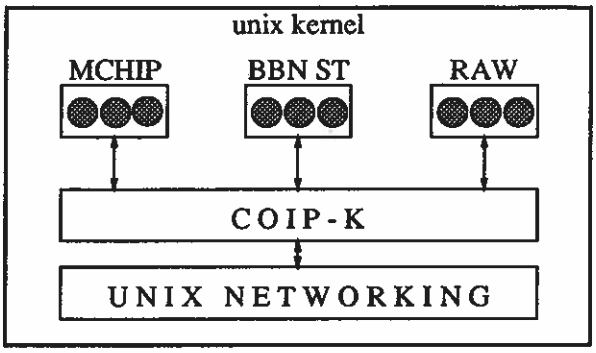


Figure 8: COIP-K structure

- A number of recently proposed COIP protocols have many parts in common. For example, COIP protocols, by definition, have a connection state machine. A connection starts out as "closed", then after a set up phase, it is considered "established." During the established state, data is exchanged using the connection, and once the data transfer is over, the connection is closed again. While the actual details of this state machine may vary from protocol to protocol, the basic model is the same for all COIP protocols. Other functionality that COIP protocols tend to share include resource reservation and enforcement and support for multipoint communication.
- Protocol development in an operating system kernel is a difficult task because the kernel is a rather large and complex program with many parts that must work together to keep the system running. A bug in the kernel can lead to system crashes which is very disruptive to users on the system.

The concept of COIP-K was also considered very useful by members of the COIP working group of the Internet Engineering Task Force (IETF) [27]. It is believed that connection-oriented protocols, such as BBN's ST and Xerox's flow protocol can also use the COIP-K as a toolkit for their implementation. Figure 8 shows how COIP-K is structured, with the COIP-K core code in the middle, and different protocol modules plugged in on top of it.

The BSD Unix networking model has three layers: the socket layer, the protocol layer, and the network interface layer. The protocol layer consists of a few communication domains with a number of protocols under each domain. For example, INET is a domain for the Internet suite of protocols, and protocols such as TCP, UDP, IP are part of this domain. For the purpose of connection-oriented protocols, we have created a new domain called COIP, and COIP-K is a protocol (actually a subset of a protocol) under this domain.

The socket layer is the layer the applications programmers use to interface to various protocol suites. Since COIP-K is built into the normal BSD Unix networking software system, the application programmer interface is very similar to any other protocol that runs under BSD Unix (e.g. TCP/IP). A typical COIP client and a typical COIP server are shown in figures 9 and 10 respectively. Note that the `setsockopt` is used to specify the performance requirements of the connection.

COIP-K maintains a per-socket protocol control block (PCB). The COIP-K PCB contains state

```
s = socket(PF_COIP, SOCK_RAW, 0)

err = setsockopt(s, level, CIN_SETPREQ, &preq,
sizeof(preq))


err = connect(s, addr, addrlen)


err = read(s, buf, buflen)
err = write(s, buf, buflen)


close(s)
```

Figure 9: COIP-K client

```
s = socket(family, type, protocol)

err = bind(s, addr, addrlen)

err = listen(s, 5)

s_new = accept(s, addr, addrlen)


err = read(s, buf, buflen)
err = write(s, buf, buflen)

close(s)
```

Figure 10: COIP-K server

information such as connection IDs (CIDs), addresses, port numbers, routing information, timer values, logical channel numbers (LCN), the state of the connection, and a pointer to a per-protocol PCB which allows protocols built with COIP-K to have their own protocol specific control blocks.

COIP-K also maintains the interface between itself and the socket and network interface layers. The interface with the socket layer is maintained using a COIP-K user request function which acts as a dispatcher for all calls received from the socket layer. COIP-K calls the standard socket functions to pass data and control information to the socket layer in the reverse direction. For the interface with the network layer, COIP-K provides an interrupt function which the network layer calls for received packets. COIP-K uses an output function to send and route packets to an appropriate network interface. There are also other functions for PCB manipulations and running the connection state machine and forwarding data in the case of a gateway. On the other hand, the protocol specific modules are expected to support protocol specific functions such as packet creation, data extraction from a packet, control packet processing, and PCB lookup and setup functions.

The first version of COIP-K is nearing completion and ready for release to interested parties. The first phase implementation of COIP-K and a test protocol modules (based on MCHIP) have been implemented and thoroughly debugged. A number of point-to-point and multipoint applications have been demonstrated over COIP-K. Measurements of COIP-K performance indicate that the per packet processing is indeed efficient with the connection-oriented approach. However, the mbuf manipulation and data copying overhead of the Unix kernel is the same for COIP-K and other protocol implementations. It is important to note that COIP-K is a carefully selected subset of MCHIP and can be used to create implementations of other connection-oriented internet protocols.

Our plan for the near future includes performance measurements of COIP-K and accordingly fine tuning the code and development of MCHIP specific modules to create a complete operational MCHIP for the Unix kernel. We also plan to help other research groups to use COIP-K to create implementations of their favorite COIP.

Simulation of the Axon Host-Network Interface Architecture

James Sterbenz, Milind Buddhikot, Guru Parulkar

A new communication architecture called **Axon** [31] has been proposed for distributed systems. The primary goal of the Axon architecture is to support a high performance data path delivering high network bandwidth directly to applications. The significant features of Axon are: (1) an integrated design of host and network interface architecture, operating systems, and communication protocols; (2) a network virtual storage facility which includes support for virtual shared memory on loosely coupled systems [32]; (3) a high performance, lightweight object transport facility which can be used by both message passing and shared memory mechanisms [33]; (4) a pipelined network interface which can provide a high bandwidth low latency path directly between the network and host memory [34].

Axon Simulation Model

This section describes the Axon architecture in the context of its simulation which was done using the BONeS™[1] simulation package [3]. The simulation of the Axon architecture serves two purposes. First, it is a verification of the design and the ability to implement key mechanisms in hardware. In particular, the simulation model of the CMP (communications processor) uses modules that correspond to functions available in a VLSI cell library. Secondly, the simulation provides a platform to evaluate design options and tradeoffs before a prototype system is built.

The Axon system level simulation consists of two hosts connected by the VHSI, as shown in Figure 11. The entire system model consists of 7 levels of hierarchy, which when fully flattened contains 1018 blocks. The upper levels of the Axon model will now be described.

[1] ™BONeS (Block Oriented Network Simulator) is a trademark of Comdisco Systems, Inc.

Figure 11: Axon System Level Simulation

VHSI. The VHSI is modeled at a high level of abstraction for the purpose of simulating the Axon host–network interface. The model includes end-to-end latency (and its variance), packet loss (including burst errors), missequencing, duplication, and bit errors. These are used as parameters into the behavioral simulation of paths connecting hosts through the VHSI. **Hosts.** In a real Axon implementation, there would be a number of symmetric hosts scattered throughout the internet. The simulation model consists of two hosts, which will be referred to as the local and remote hosts depending on where a request has been initiated. The Axon host model is presented in Figure 12. Each host contains a CPU, CAP (CMP assist processor), CMP (communications processor – the network interface chip), and CMM (communications memory module). The local host is responsible for generating requests based on the address reference trace model. The remote host receives requests and returns segments based on the requests.

This organization allows a direct connection through the CMP between the CMM and VHSI. Thus, packets can be transmitted and received without any host interaction, with the CMP performing all critical per packet processing. The CAP serves to perform functions that do not need to be implemented in CMP hardware, but involve protocol processing that may be offloaded from the CPU.

CPU. The CPU simulation model generates requests from a process model consisting of a dispatching queue served by the CPU

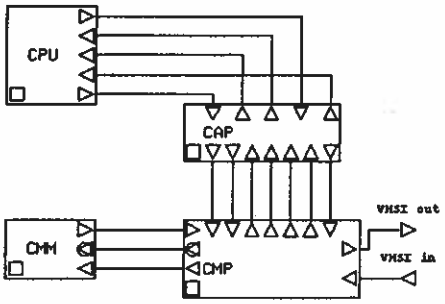


Figure 12: Axon Host

instruction processor. Processes receive service for a specified burst length (in number of instructions), and are returned to the dispatching queue. The program execution model assumes that process execution can be divided into phases during which a set of segments is in the locality set. This phase behavior is a common way of modeling program locality. In the Axon environment, segments may be located on other hosts across the VHSI. Thus, with a given probability, a remote segment fault will take place. This results in transfer of control to ALTP-OT which issues a **get-segment** request, which is then passed to the CAP.

When pages within the segment return, the CPU is notified of their arrival by the CAP. The CPU can then mark them present in the corresponding page table and recover from a page fault if necessary.

CAP. A high performance microprocessor, the *CMP assist processor* (CAP), performs functions that are not part of the critical path, but require higher performance than could be provided by the host CPU without adversely affecting the performance of other host processes. The CAP is responsible for building control packets and passing them to the CMP for transmission and checksumming. Similarly, control packets received by the CMP are passed to the CAP for full decoding and subsequent action, which may involve interaction with the host CPU. The CAP is involved in the timer management for request retransmission, and notifies the host when pages have been completely received.

Since the CAP is a microprocessor running software processes to perform its function, operations are modeled as processing delays with parameters indicating the number of instruction cycles. The simulator insures the serialization of processing delays. Additionally, there is some overhead modeled in task switching between operations.

The CPU sends requests to the CAP, such as for a remote segment fault. The CAP then builds the appropriate control packet and passes this to the CMP. The CAP also decodes incoming control packets from the CMP and takes the appropriate action. This may involve interrupting the CPU, for example when a link fault is required in response to a **get-segment** control packet received.

CMM. The method for providing direct access between host memory and the network interface without any store-and-forward hops is through the use of a special multi-ported *communications memory module* (CMM), similar in concept to VRAM (video-RAM) design. The CMM has a conventional random access port which appears like any other memory bank to the processor–memory interconnect, out of which the CPU may execute code and access data. The other ports are high speed sequential access interfaces to the CMP (transmit and receive), and must operate at a rate of the VHSI optical links scaled by the CMP datapath width.

A delay models the access time to start up the read from the sequential output port. The data packet length field is inserted in the packet for use by simulator probes in computing throughput. Individual read operations take place for a sequence of packets in a given page, given the page base address. A page transmission request from the CMP `Rate Control` results in a page burst of packets at full VHSI link rate.

CMP. The goals for the design of the CMP include the ability to perform critical path

functions in real time with no packet buffering and to incorporate the necessary function in VLSI. This may be realized by organizing the CMP as a dynamically reconfigurable pipeline, based on the ALTP type and options for a particular connection. The pipeline organization allows packets to be processed at the VHSI data rate.

It is important to note that the simulation model is constructed to reflect the actual hardware design of the Axon CMP. Since a major thrust of this research is to investigate the implementation of critical path function in hardware, simulation blocks are chosen carefully to represent function easily implementable in VLSI. Thus, high levels of simulation abstraction are not used within the CMP model. For the sake of brevity, details of the CMP simulation are not presented here but can be found in [37].

Simulation Results

Axon simulation results are divided into four groups: overall system behavior, rate control, error control, and functional partitioning. Two example results are presented in this section from rate and error control. A complete set of simulation results can be found in [37]. A metric of primary concern is the time that a process is blocked waiting for the return of a data segment. The time interval between the process referencing the segment and its complete return to the local host is defined as T_S. More important is the time a process is *blocked*. This is measured by the interval between reference and the return of the *first* page in the segment, since execution can begin when this page is marked present; this is defined as T_s.

A number of simulation experiments have been run to indicate the performance of the Axon system with respect to errors. An example is shown in Figure 13. In this case 3 connections on each host share 90% of a 1Gbps link. The packet size is 32 bytes of data with 16 bytes of

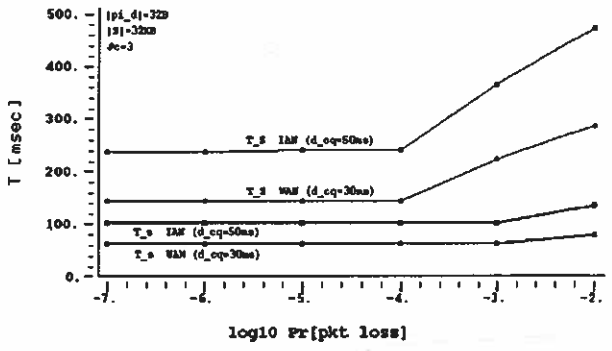


Figure 13: Latency vs. Packet Loss

header. The page size is 1 Kbyte, with an average segment size of 8 pages. Results are shown for both a WAN latency of 30ms and an internetwork (IAN) latency of 50ms.

Packets are lost with a probability varying from 10^{-8} to 10^{-2}. Note that artificially high error rates with very small packet sizes are simulated to push the error control mechanisms to extreme limits. The VHSI environment is expected to be much more reliable than this.

The latency performance is quite flat, even up to extremely high error rates, which can be attributed to the use of selective retransmission. The T_S (upper) curves indicate the blocking that would occur if a process would have to wait for an entire segment to arrive (as is the case with common general purpose protocols). The T_s (lower) curves indicate the advantage of knowledge by ALTP-OT of the page structure of segments, allowing processes to resume execution more quickly (the height of T_s related to the page size). Note that while the segments simulated are rather small, T_s is independent of segment size. This is the case since T_s is only dependent on the correct arrival of the first page in the segment, and retransmitted pages preempt the primary segment transmission in progress. As segment sizes increase, so will the difference between T_s and T_S, and thus the performance benefits of ALTP-OT.

For a rate control scheme to be fair, all processes should receive the specified share of bandwidth, and in the case where rate

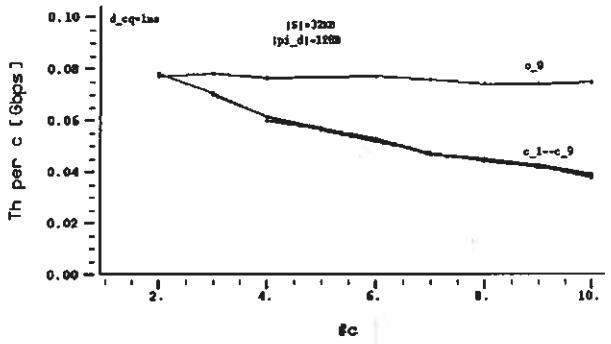


Figure 14: Rate Fairness

Table 1: Network Throughput by Connection [Mbps]

$\|s\|$	1		32		1K
$\|\pi\|$	1040B		144B		48B
$\#c$	10	100	10	100	10
min	3.92	3.93	61.5	9.99	94.8
max	3.93	3.93	73.0	10.38	99.5
mean	3.93	3.93	70.5	10.23	96.8
ideal	3.93	3.93	70.8	9.96	96.8
total	39.30	392.00	708.0	996.00	968.0

specifications are identical across processes none should receive significantly different service. Figure 14 compares the throughput across connections as they are added. A single connection c_0 was given a peak bandwidth of $\lambda_p = 0.5$Gbps with additional connections evenly sharing an additional 0.5Gbps as they were added. The curve for c_0 is flat as desired, and the curves for the other connections decrease and track one another closely.

Table 1 indicates the network throughput per connection for various combinations of packet size $|\pi|$ and segment size $|s|$ (with a 1KB page size), for 10 and 100 active connections per processor, indicating close correspondence across connections. Note that since the application is subjected to the rate control mechanism and has no direct access to flow control once the requested rate has been granted, a firewall is established preventing applications from harming one another in terms of bandwidth utilized. Simulations were also run verifying the insensitivity to individual connections as the error rates on other connections were driven extremely high.

Visualization of Cells in Biological Organisms

Christos Papadopoulos, Guru Parulkar

In collaboration with scientists in the department of biology, we are developing a distributed visualization scheme for the display of cells of biological organisms in 3D, in particular, the cellular slime mold which is an amoeba-like organism. This project has been motivated by the following two observations:

- Very little is understood about the real communication requirements for distributed computations on wide area networks, despite the fact that such understanding is necessary for design of efficient protocols to support these computations.
- Although pipelining is a simple special case of general distributed computation, it is readily applicable to many televisualization applications; and televisualization is a very important and challenging class of network applications because it is intensive both in computation and communication.

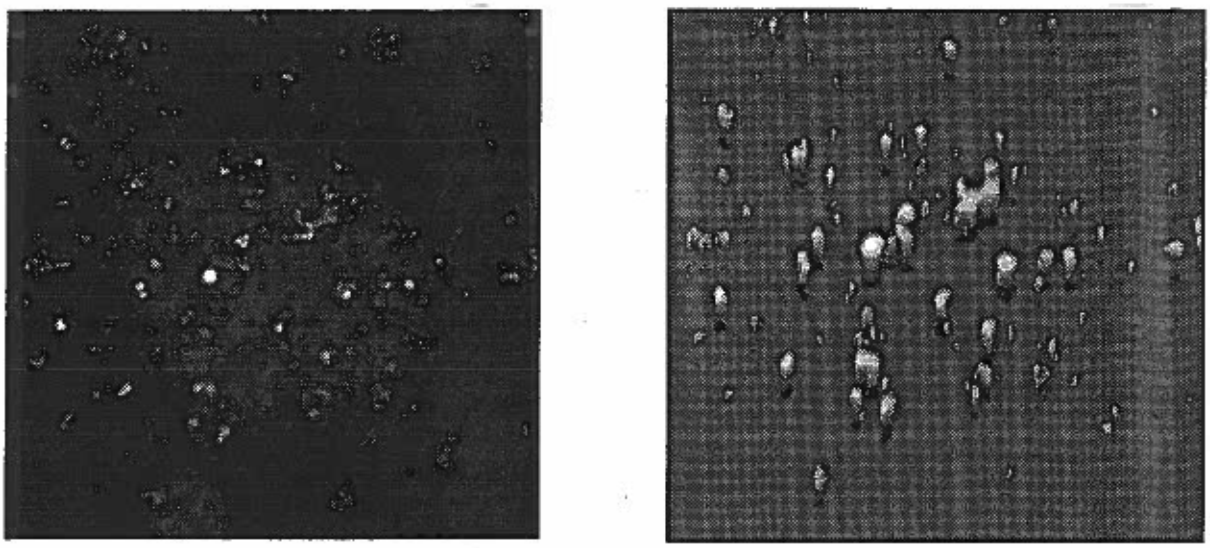
  Thus, the goal of this project is to better understand the remote visualization applications and to characterize the performance bottlenecks of the protocols, if any, for this class of applications.

The organisms being studied undergo a very interesting transformation as they develop: first, they aggregate in a hemispherical mount of about 10^5 cells; the mount is soon transformed into an upright finger called a slug; then, the slug topples over and moves as a unit before eventually standing upright again to differentiate into two parts, the stalk and the spores. This last process of differentiation is the focal point of the biological research.

The significance of this study lies in the promise that insight can be gained on how global shape changes are produced by coordinated movements of cells in a developing organism. An important question is how cells in different parts of the mount are determined to become a part of the stalk or the spores. Answers to this question can lead to better understanding of the development of any embryo. One of the visions of this project is that understanding of embryo development may lead to the ability to control regeneration of damaged organs in the body, by simulating the conditions of growth. For such biological applications, we have found volume visualization techniques to be very useful. In our current volume visualization scheme there are four major steps: data collection, thresholding, rendering and display. They are described in the following paragraphs.

Data is collected using a computer-controlled microscope used for time lapse 3D imaging. 3D images are collected in the form of 2D slices obtained by rapidly stepping through focal planes spanning the specimen. The whole data collection process takes about 2 hours, collecting a 3D image every 2 minutes, for a total of 60 images. Typical image sizes are $256 \times 256 \times 20$ at 2 bytes per voxel, giving a total data size of about 157 Mbytes. Each 2D slice collected by this method is blurred by out of focus light from the surrounding slices. Removal of the blurring is accomplished by thresholding the data at an appropriate value obtained from the data histogram. Rendering of the cleaned up data is accomplished using a Simulated Fluorescence Process (SFP) algorithm. This algorithm closely resembles a fluorescence process in that it simulates the flow of excitation light through the 3D data and then creates a projection by gathering the fluorescence from each individual voxel. Finally, the projected images are displayed using the X-window system. Each image is saved at the display stage, the researcher can choose to view them individually or as an animation sequence.

The four steps described above are naturally implemented as a pipeline of separate modules. While the raw data is collected and stored on a lab computer, the data viewing needs to be on



Raw Image

Processed Image

Figure 15: Unprocessed and Processed Images

the biologist's workstation remote from the lab. Instead of either doing all the computation on the lab computer and shipping images to the workstation for display, or shipping the raw data to the workstation first and doing all computation there, we assign each stage to a different machine so that pipelined parallelism can be achieved. Each of the modules is designed such that it can run on different computers to better utilize the computing resources. Communication between modules is through Unix IPC socket interface on top of TCP/IP over our campus network (10 Mbps Ethernet). Each module consists of three processes: reading, computing and writing. These processes communicate locally using shared memory. This allows communication to proceed independent of computation, better utilizing the local computing resources.

The left side of figure 15 shows the 3D organism as seen by a user from the microscope. Clearly, it is very difficult to trace various cells and their trajectories using such images. The right side of figure 15 shows the processed image which is quite good for identifying cells, and an animation of a sequence of such images is also found useful for studying the cell trajectories.

Initial experience with the implementation does show that there is speedup when using the pipeline. The speedup can be attributed to two factors: parallelism resulting from pipeline operation and the reduction in disk swapping due to availability of increased memory at different stages and reduced memory requirement at any given stage of the pipeline.

If the work load is partitioned to balance the computational requirements of the different parts, communication processing becomes the bottleneck and it will get worse as the data resolution increases. The communication overhead includes moving the data between user and system spaces, protocol processing, and data transmission (data transmitting time + propagation delay). The transmitting time is expected to decrease as faster networks become available, but the propagation delay will remain the same and become a more significant overhead. The protocol processing delay will also increase since transmitted data size is expected to increase. Locating bottlenecks in the current IPC paradigm, (i.e., the UNIX

socket interface on top of TCP/IP) may provide insight into what problems need to be addressed in the design of future protocols to support pipelining. To identify those problem areas, we plan to run the example application for different data sizes and pipeline configurations with a minimal number of probes strategically placed in the networking code. This will help us acquire a better understanding of the different processing delays in the individual protocol layers and characterize them as either protocol or operating system dependent. Currently such experimentation is in progress.

A Recurrence Model for Asynchronous Pipeline Analysis

Fengmin Gong, Zubin Dittia, Guru Parulkar

Motivation. We have proposed pipelining across high-speed networks as an efficient televisualization model, called pipelined televisualization (PTV) [11]. This has led us to develop tools for the analysis of asynchronous pipelines. In general, a pipeline can either operate in lock-step fashion by a global clock or operate by having each stage processor synchronize only with its neighbors using a handshaking protocol. In the first case, the pipeline is called a *synchronous* pipeline; the second case corresponds to an *asynchronous* pipeline. Issues surrounding synchronous pipeline design are well understood. Synchronous pipelines are generally simpler to design, and they can be very efficient if the computational task can be partitioned into stages of nearly equal processing time. However, asynchronous pipelines are absolutely necessary when pipeline stages are physically distributed or have varying processing delays, for example, in the case of a pipelined televisualization.

There are still two main obstacles to the wide use of asynchronous pipelines. The complexity of asynchronous circuit design is still high; and characterization of the asynchronous behavior has been a difficult task leading to limited understanding of asynchronous pipelines. We have developed a simple yet accurate method for analysis of asynchronous pipelines using recurrence relations. This development was done in two steps: (1) derivation and experimentation for pipelines of linear topology, and (2) derivation and verification for pipelines with general DAG (directed acyclic graph) topology. This note will present the recurrence relations for only the DAG topology and example verification results. For details refer to [12].

Notation. We view the set of dependencies among processors of the DAG as a set of dependency chains. We use a two dimensional indexing scheme for the identification of processors. The processors are partitioned into stages as follows. Let m be the total number of stages, which is defined as the number of processors on the longest dependency chain. Each source processor is labeled as a stage 1 processor and a sink processor as a stage m processor. For any internal processor (i.e., not a source or sink processor), if i is the maximum distance from the processor to any sink processor, this processor is labeled as a stage $(m-i)$ processor. If w_i is the number of processors at stage i, we label these processors by tuples $(i,1),(i,2),\cdots,(i,w_i)$. Also, we use $pred(i,j)$ to denote the set of predecessors for processor (i,j) and $succ(i,j)$ to denote the set of successors to processor (i,j).

Now we introduce recurrence variables. Let s_{ijk} be the time at which processing of data item k begins at processor (i,j). Let d_{ijk} denote the processing delay for data item k at processor (i,j), and let f_{ijk} denote the finish time of data item k at processor (i,j). Finally, l_{ijk} represents the time at which data item k leaves processor (i,j) and arrives at all processors in $succ(i,j)$. We assume that a processor begins execution when all input data items have arrived and can forward its output data item to its successors only when they are ready to receive the data item. A set of recurrence relations in these variables can be derived and organized as initial conditions, boundary conditions and recurrence body.

Initial Conditions. Since the pipeline is empty initially, the first data item $(k=1)$ can be processed at a given processor as soon as all the corresponding input data items are received from predecessors; and the resulting output data item can move to successors immediately after its processing is finished. The first data item is available to the source processors at

time 0. Therefore, the following initial conditions hold for the pipeline:

$$\begin{aligned} s_{1j1} &= 0 \\ f_{1j1} &= d_{1j1} \\ l_{1j1} &= f_{1j1} \\ s_{ij1} &= \max_{(i',j') \in pred(i,j)} (l_{i'j'1}) \\ f_{ij1} &= s_{ij1} + d_{ij1} \\ l_{ij1} &= f_{ij1} \end{aligned}$$

where $2 \leq i \leq m$ and $1 \leq j \leq w_i$.

Boundary Conditions. Operation of source and sink processors is also special. First, source processors never have to wait for input data items because they are assumed to have all data items to begin with. Sink processors never have to wait to forward an output data item because they consume their own output. These correspond to the boundary conditions of the pipeline:

$$\begin{aligned} s_{1jk} &= l_{1jk-1} \\ f_{1jk} &= s_{1jk} + d_{1jk} \\ l_{1jk} &= \max_{(i',j') \in succ(1,j)} (l_{i'j'k-1}) \\ s_{mjk} &= \max_{(i',j') \in pred(m,j)} (l_{i'j'k}) \\ f_{mjk} &= s_{mjk} + d_{mjk} \\ l_{mjk} &= f_{mjk} \end{aligned}$$

where $1 \leq j \leq w_i$ and $2 \leq k \leq n$.

Recurrence Body. Processors of internal stages ($2 \leq i \leq m-1$) have to wait for input data items from all predecessors to arrive before the processing can start; once the processing is complete, the output data item has to be held until all successors are ready to receive it. This logic is formulated into the recurrence body:

$$\begin{aligned} s_{ijk} &= \max_{(i',j') \in pred(i,j)} (l_{i'j'k}) \\ f_{ijk} &= s_{ijk} + d_{ijk} \\ l_{ijk} &= \max[\max_{(i^{prime},j^{prime}) \in succ(i,j)} (l_{i'j'k-1}), f_{ijk}] \end{aligned}$$

where $2 \leq i \leq m-1$, $1 \leq j \leq w_i$ and $2 \leq k \leq n$.

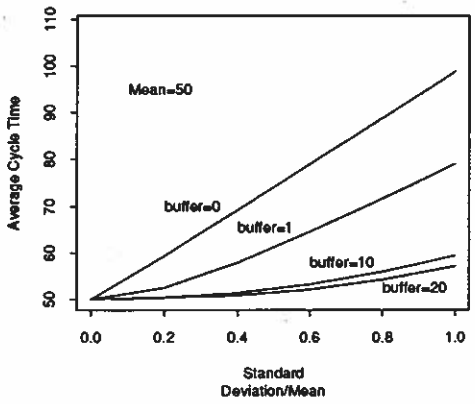


Figure 16: Linear Pipeline

Given a real-life asynchronous pipeline, we first apply two simple transformations (see [12] for details) that models the effect of interstage communication overhead and data buffers. We then apply the recurrence relations to the resulting pipeline. The evaluation of the recurrence relations provides a complete trace of the start time, finish time, and departure time for every data item at each processor.

Results. The recurrence model has been implemented as a single C program. Results of two experiments have been reported that help verify the correctness of the recurrence relations. These experiments involved a 3-stage linear pipeline and a 3-stage pipeline with two processors at the second stage, referred to as a parallel pipeline. These pipelines are evaluated using the recurrence model as well as a discrete event simulation using BONeS, a network simulator from COMDISCO Systems Inc. In this study, all stages (computation/communication) were assumed to have a normally distributed processing latency with mean equal to 50. The standard deviation was varied from 0 to 50 with step size 10 and four buffer sizes (0,1,10,20) were examined. Buffer sizes were always the same across stages. In all cases, both simulation and the recurrence model were run with more than 9000 data items to ensure statistical significance for the results. Figures 17 and 18 presents respectively

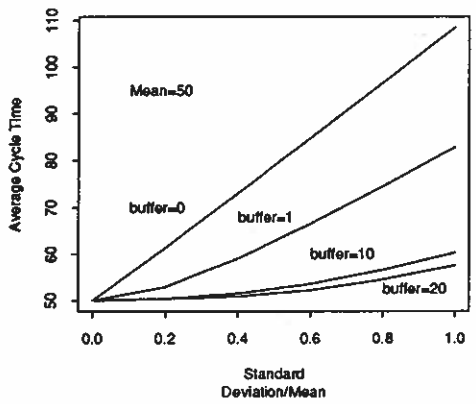


Figure 17: Parallel Pipeline

the results for the linear and the parallel pipelines from the recurrence model. The same set of results were also obtained with the BONeS simulation model, and they match almost exactly with results of the recurrence model. Hence, the simulation results are omitted from the plot. It should be noted that the metric used in these plots is the average cycle time $AC^+ = \frac{f_{mn} - f_{m1}}{n-1}$ $(n > 1)$.

First we examine the linear pipeline results shown in Figure 17. When processing delay for all stations is constant at 50 (i.e., standard deviation = 0), the average cycle time is 50 regardless of buffer size. As the standard deviation is increased, and there is no buffering, the average cycle time shows a linear increase; with buffer size 20, the increase in average cycle time is very little. Thus, the family of curves (for buffer sizes 0,1,10,20) shows that the increase of average cycle time with the increase in standard deviation is slower with larger buffers. This result fully conforms to our expectation of the effects of buffering.

In the case of a parallel pipeline, Figure 18 shows a similar trend, that is, the average cycle time increases linearly with increase in standard deviation when buffer size is 0. As larger buffers are used, average cycle increases slowly with increasing standard deviation. Comparison of Figures 17 and 18 show that for the same processing delay and buffer size, a parallel pipeline has a larger average cycle time than that of a linear pipeline except when standard deviation is zero. This is correct, because in the case of parallel pipelines, the waiting resulting from splitting and merging of stages contributes to a longer average cycle time. This kind of waiting is due to synchronization among parallel data streams, and it cannot always be eliminated by using extra buffers.

Integrated Traffic Characterization and Control

Andreas D. Bovopoulos, Apostolos Dailianas, Seyyed Mahdavian

Research concerning ATM networks is currently focused on the study of teletraffic problems appearing in an ATM environment. The fundamental problem of the ATM technology stems from the fact that the performance of an ATM statistical multiplexer depends on the time behavior of the incoming traffic. As a result, without sufficient traffic control provisions, an ATM network may fail to provide grade of service (GOS) guarantees to end users.

Each connection in the ATM layer results in a cell sequence (CS) that can be both analyzed and controlled at one or more of the following time scales: call, burst, and cell. At each of the time scales for which a CS control is provided, a resource allocation scheme can be introduced and classified as corresponding to the call, burst, or cell time scale.

In order to design an integrated traffic characterization and control infrastructure capable of guaranteeing a GOS and providing that GOS at the minimum cost, the following are required: (i) the design of a traffic control and resource allocation infrastructure capable of efficiently handling a wide variety of traffic behaviors, (ii) given a particular traffic control and resource allocation scheme, the determination of the GOS that can be provided to an incoming CS, (iii) given a particular GOS requirement, the determination of the traffic control and resource allocation scheme required to provide to an incoming CS the desired GOS at the minimum cost. The work presented in [7] is part of an ongoing effort directed towards satisfying these requirements.

In [7], a traffic generator capable of modeling a broad spectrum of incoming traffic behaviors is introduced. A traffic control mechanism (TCM) capable of supporting a number of different cell time scale control and resource allocation schemes is described. For each of a variety of incoming CS time behaviors and a given control and resource allocation scheme, the GOS that can be provided by the TCM is determined.

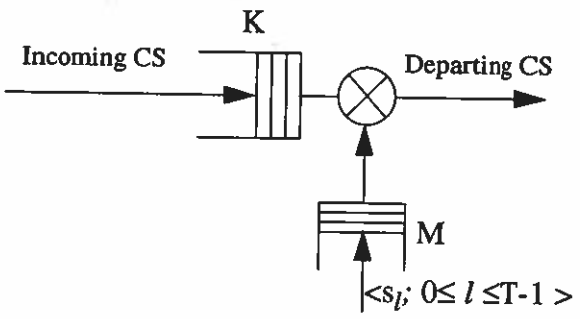


Figure 18: Traffic Control Module

The TCM (Figure 18) has a cell buffer of size K and a token pool of size M. Time is divided into units called slots, where the time interval corresponding to a slot equals the service time of a cell. Incoming cells are served by the TCM in the order of their arrival. Tokens act like permits for cell service. The cell at the front of the buffer is served at the beginning of a slot if a token exists, in which case a token is removed from the token buffer. If no token is available at the beginning of a slot, no cell is serviced during that time slot. Incoming cells are stored in the cell buffer as long as the buffer is not full; if the buffer is full, cells are discarded.

With the control and resource allocation just described, the TCM provides cell level control and resource allocation; i.e. control and bandwidth allocation decisions are made at the cell time scale. In addition to this cell level control, burst or call level controls and resource allocation schemes can be utilized. The focus of [7], however, is on cell level control and resource allocation. Issues relating to burst level control and its interaction with cell level control are currently under investigation.

For reasons explained below, tokens are generated *with a deterministic and periodic pattern*. Specifically tokens are generated in accordance with a T-state, cyclic, deterministic Markov chain. The number of tokens generated

in state l of the Markov chain is equal to s_l for $0 \leq l \leq T-1$. Once generated, a token is either (i) immediately utilized by a cell requiring service, (ii) stored in the token pool if there is no cell requiring service and the token pool is not full, or (iii) immediately discarded if the token cannot be immediately utilized by a cell and if the token pool is full. The time between two transitions of the Markov chain is D slots, and therefore the period of the deterministic and periodic pattern is $D \times T$ slots. The token patterns that can be generated in this fashion are deterministic and thus predictable and yet at the same time rich enough to implement a wide variety of control policies. The TCM control scheme can be easily implemented using two counters, one which records the state of the token pool and one which keeps track of the state of the token Markov chain.

Generating tokens in the manner just described is desirable for the following reasons. By appropriately selecting D and $T \times D$, the size of the cell buffer K, the size of the token pool M, and the number of tokens generated at each transition of the Markov chain, s_l, for $0 \leq l \leq T-1$, a wide variety of GOSs can be provided to an incoming CS. Because tokens are generated according to a deterministic and periodic pattern, a TCM can optimally support deterministic and periodic incoming CSs; a deterministic and periodic incoming CS controlled by a TCM results in a departing CS which is also deterministic and periodic. Further by using a deterministic control scheme, an open loop control scheme can be implemented when desired.

Recently, attention has been given to a hierarchical family of protocols [11] based on the periodic exchange of information related to the transmission of bursts. Note that a burst is a group of packets, and a packet is the fundamental information block at the transport layer. The structure of the TCM makes the TCM the ideal mechanism through which hierarchical type protocols can be implemented at the cell level. In such an environment the TCM parameters remain fixed for the duration of a burst and may be reset from burst to burst.

The TCM is a new traffic control mechanism. It differs from the variations of the leaky bucket mechanism that have appeared in the literature in the fact that it is integrated into the traffic characterization and control infrastructure of the network and in the fact that it can monitor and control a much richer family of traffic behaviors. The analysis presented in [7] is complete and informative for performance parameters including loss distribution, waiting time distribution, token loss distribution, cell buffer occupancy and token buffer occupancy.

The TCM performance analysis presented in [7] is developed in the following manner. First, a traffic generator capable of modeling a broad spectrum of traffic behaviors is introduced and extensively studied. Next the TCM performance is evaluated under the assumptions that the service time of a cell is zero and that a slot is not defined with respect to the cell service time, i.e. a slot is simply a time interval of unit length. These assumptions are widely used in the literature to simplify analysis. Later, the simplifying assumptions are removed, and the TCM performance is evaluated assuming a constant positive cell service time equal to a unit of time referred to as a slot.

Current experience suggests that the computational procedures introduced in [7] are stable and efficient. Examples with 60,000 states have been evaluated without difficulty. Nevertheless one aspect of our research focuses on the development of continuous time flow algorithms with even higher computation efficiency.

One of the limitations of the current version of the TCM is its inability to handle traffic sources with long active and idle periods. For such sources, enhancement of the TCM control capability is necessary in order to do resource reservation not only at the cell level but at the burst level as well. A desirable TCM enhancement currently being developed is the capability of deallocating buffer space and bandwidth (i.e. tokens) from idle sources and

reallocating this buffer space and bandwidth to sources currently in need of resources. Such an enhancement will facilitate the multiplexing of highly bursty sources and may prove useful for the development of efficient burst level resource allocation schemes.

Engineering ATM Statistical Multiplexors

Andreas Bovopoulos, Apostolos Dailianas, Saied Hosseini-Khayat

An integrated broadband network should be able to support a variety of existing and yet to be defined services. It is widely accepted that the transport technology that provides the most flexibility for the realization of such an ambitious undertaking is ATM.

In an ATM network, the access segment is considered the most crucial segment because it includes systems that cannot be tuned more than once every Operationan Planning Period and because, owing to its proximity to traffic sources, the access segment design must take into consideration the dependency of ATM traffic handler performance on traffic mix. For these reasons, the most crucial component of a broadband network is the First Statistical Multiplexing Stage (FSMS), which is the network segment connecting NT2s to Inlets of Remote Switching Units. The FSMS should be able to absorb time and local variations of demand using a few standardized, modular, scalable network components arranged in a flexible, fault and overload tolerant network architecture. More detailed discussion of the importance of the FSMS can be found in [3]. The central design problem of the FSMS is the problem of developing multiplexing rules for practically important and representative classes of the ATM traffic.

In order to fully understand the requirements imposed on the FSMS, let us assume that I multiservice terminals (MSTs) are connected to the FSMS. Each MST supports one or more services, and at the cell level, each terminal issues calls that generate cell sequences (CSs) with identical or distinct statistical characteristics. For example a digital voice conversation results in one CS, whereas a data terminal generates a family of statistically distinct CSs.

The simplest possible traffic multiplexing problem that the FSMS must handle is the case in which each of the terminals generates the same CS. This is a classical multiplexing problem that has been extensively studied and is referred to here as Multiplexing Problem A. A more complicated traffic multiplexing problem arises when each terminal generates one CS, while different terminals generate different CSs. This problem is referred to here as Multiplexing Problem B. If each terminal is a multiservice terminal, then each terminal can generate a family of CSs. If all terminals generate the same family of CSs, then the resulting multiplexing problem is referred to as Multiplexing Problem C_1. If the different multiservice terminals generate different families of CSs, then the resulting multiplexing problem is referred to as Multiplexing Problem C_2.

Our current efforts are focused in the solution of type A and B multiplexing problems. We are currently studying the behavior of a multiplexer when the incoming CS is composed of a number of multiplexed, independent, and identically distributed bursty sources. Our engineering objective is to completely and efficiently characterizate the behavior of a large number (at least 100) of multiplexed, independent, and identically distributed bursty sources. Such a result would be necessary in order to solve the access problem of how to effectively multiplex low rate bursty sources (say up to 2Mbps) over a 140Mbps access multiplexer. In [7] a number of necessary tools have been developed that are currently applied for the solution of this multiplexing problem.

We are also attempting to identify the macrodynamic behavior of a multiplexer, by identifying invariant parameters and simple traffic engineering rules that adequately describe the relationship between the source behavior, a multiplexer's bandwidth and buffer parameters, and the provided GOS, without having to resort to the mathematical equations that describe the microdynamic behavior of the multiplexer. Such results would be necessary in

order to make a real time evaluation of the performance that could be guaranteed to a particular incoming CS by a particular multiplexer and to make a real time calculation of the multiplexer's buffer and bandwidth requirements needed in order to achieve a particular GOS for a particular incoming CS.

We anticipate that the satistactory solution of the above problems will facilitate the solution of the multiplexing problem B, whose solution we anticipate will lead to the solution of the multiplexing problems of type C.

The Access Unfairness Problem in DQDB MANs

Andreas D. Bovopoulos and Lakshmana N. Kumar

The Distributed Queue Dual Bus (DQDB) MAN is composed of two unidirectional buses, to which the individual nodes (access units) are connected across the buses. In addition a head-end station (also known as a frame generator) is provided for each bus. The head-end generates the DQDB frames (or segments which carry the data) and transmits them along a single bus. The two unidirectional buses operate in opposite directions. The nodes are connected to both buses, using a read and write connection to each bus. The two buses are named bus-A and bus-B. The head-end that controls bus-A, is known as the *Head-end Of Bus-A* (HOB-A). The portion of the bus between node-i and the HOB-A on bus-A is known as the *upstream* of node-i, with respect to bus-A. Similarly the portion between node-i and HOB-B on the bus-A is known as the *downstream* of node-i with respect to bus-A. Every node has to file a *REQUEST* by setting the *REQ* bit of a bus-B slot, in order to grab an empty slot on bus-A for transmission of its packet. The requests are filed on the bus opposite the one that the node wants to gain access to. The detailed operation of the protocol can be found in [12], [18] and [25].

One of the most important problems appearing in DQDB MANs is the access unfairness problem. Access unfairness can be defined as the inability of a subset of the nodes to gain access to a bus as quickly as the other nodes under the given access protocol mechanism. As a result some nodes experience longer delays than the other nodes before gaining access.

At low loads, the observed demand for service is less than the bandwidth available. The low load range may extend up to about 0.4 of the channel capacity, depending on the actual traffic pattern. Because the demand is less than the unused bandwidth, the unfairness is not an issue at low loads. At higher loads three types of persistent unfairness may arise.

1. **Latency-Related Unfairness:** The nodes that are closer to the frame generating head-end have no knowledge of the *REQUEST*s that are still in transition. As a result the distributed queue maintained by DQDB protocol is not perfect. Therefore, the latency results in a type of unfairness which is called latency-related unfairness. The latency-related unfairness is predominant when the load presented on a single bus is less than the bus capacity. When the DQDB network is overloaded, two different persistent types of unfairness may appear.

2. **Access-Related Unfairness Due to Request Flooding:** If the overload situation is caused by a surge of traffic demand from the downstream nodes, then the upstream nodes are at a disadvantage compared to the downstream nodes. The earlier nodes along the direction of a bus may end up heavily blocked for access — because under the DQDB access mechanism the overwhelmingly high number of *REQUEST*s need to be honored. Honoring all the *REQUEST*s also invites a fresh influx of *REQUEST*s, if the downstream nodes are sufficiently busy.

3. **Access-Related Unfairness Due to Message Flooding:** If the overload situation is caused by a surge of traffic from the nodes closer to the head-end, then the downstream nodes are at a disadvantage compared to upstream nodes. The ability of a node to have more than one pending request coupled with the latency creates such a situation. As a consequence, the busy downstream nodes may suffer unduly with their access blocked and with the upstream nodes claiming an unfair share of bandwidth.

At heavy loads the actual unfairness pattern is very dependent on the load distribution pattern

(state of the network) along the bus, at the time when the heavy load conditions set in. This is especially true with large inter-node distances. In overload situations the unfairness phenomenon is persistent. Unfairness due to flooding can be resolved only through the application of an *access protection* mechanism. Because of latency and flooding phenomena, the high load situation can further be subdivided into the following categories depending on the overall load presented to a bus.

- **Heavy Load:** When the offered load is heavy but still below the channel capacity unfairness is caused predominantly by the propagation latency of the downstream *REQUEST*s, rather than by the flooding phenomenon.
- **Overload:** When the offered load exceeds the channel capacity, unfairness is caused predominantly by the flooding phenomenon, with latency playing a secondary role.

A protocol aimed at addressing the access unfairness problem must satisfy a number of requirements: (i) It should work for both small and large networks. (ii) It should work for any number of attached nodes and for arbitrary internodal distances. (iii) It should resolve both the latency and the flooding related unfairness.

At low loads there is no unfairness, and the DQDB protocol is fair enough. Therefore any suggested protocol should preserve the behavior of the DQDB protocol at low loads. At heavy loads any proposed protocol should address the latency related unfairness by allowing some extra slots to meet the unseen (yet registered) demand from the downstream. The unfairness due to flooding should be resolved by appropriately reducing the amount of issued *REQUESTs*.

As is evident from the previous discussion, one of the factors that affect the unfairness problem is the actual topology of the network, defined as the inter-nodal distance between the *active* nodes (*i.e.* the nodes that are part of the network and are potential candidates to contest for access) attached on the bus. The DQDB protocol does not provide information about the topology of the network. In [25] a protocol called *the Dynamic Assessment of Network Topology (DANT),* designed to achieve dynamic updates of the network dependent parameters is presented. This protocol provides *in real time* information about the active node population in the network, the length of the bus segment in downstream of each node and the distance between successive nodes in the network. Such information is useful in a dynamic network environment, in which new nodes may join and some existing nodes may leave the network. In [25], an *Anticipatory Demand Scheme* (ADS) aimed at resolving the latency related unfairness and an *Access Protection Scheme* (APS) aimed at resolving the heavy load unfairness due to flooding are introduced. Elements of these three mechanisms are then combined in the *3-Tier Fairness Protocol*, which aims at retaining the performance of the DQDB protocol at low loads while improving its performance at normal and heavy loads by effectively addressing the various sources of unfairness.

The performance of all the introduced protocols is studied through a number of detailed simulations. Two different types of load behaviors have been studied: in the *symmetric load* type, packet destinations are chosen at random with equal probability; in the *equal probability* type, packets are sent on each bus with equal probability and their destinations selected at random from among those in the chosen direction. The effects of both Poisson and bursty arrival processes were studied. A detailed description of the simulation studies can be found in [25]. The overall performance improvements of the 3-Tier fairness protocol scheme can be summarized as follows:

- The 3-Tier fairness protocol divides the load activity in the network into three different domains and adopts different access policies to suit the needs.

- At low loads the 3-Tier fairness protocol performs exactly as if only the DQDB protocol were running. This domain extends up to load values of about 0.4.
- At heavy loads the 3-Tier fairness protocol produces performance enhancements, with the domain extending to about a load of 0.75. The performance is very similar to that achieved with the ADS protocol in this region. In the case of equal probability load, a few end-nodes along the bus come under the influence of and experience a small increase in their access delays.
- At loads of 0.9 and above, the characteristics of the APS mechanism come into play. The performance improvement is dramatic with symmetric load. With equal probability load, the end-nodes' access delays suffer sharp rises, and all other nodes have uniform access delays. While we cannot eliminate this unfairness, it appears tolerable given the extreme nature of the traffic conditions.

With multiple packet sized messages and different network configurations, the 3-Tier fairness protocol continues to provide consistently better performance than DQDB. Performance improves further with heterogeneous packet sizes. In conclusion, the 3-Tier fairness protocol presents a significant performance enhancement over DQDB over the entire range of traffic demand.

References

[1] Anderson, James A., "Persistent Connections in High Speed Internets," MS Thesis, Department of Computer Science, Washington University. 1991.

[2] Anderson, James, Zubin Dittia and Guru Parulkar. "Persistent Connections in High Speed Internets," Proceedings of the IEEE Globecom 91, December 1991.

[3] M. Bonatti, G. Gallassi, O. G. Soto, "Role of Ergodic Approximations and of Ergodic Samples in IBC Strategical Planning: Lessons from POTS Traffic Engineering," Proceedings of the Integrated Broadband Communications: Views from RACE, Bonatti, Casali and Popple (Editors), Elsevier Science Publishers, 1991.,

[4] *Block Oriented Network Simulator™ (BONeS™) User's Guide*, version 1.5, Comdisco Systems, Inc., Nov. 1990.

[5] Bovopoulos, Andreas. "Resource Allocation for Markovian Queueing Networks: the Partial Information Case," Washington University Computer Science Department, WUCS-89-22.

[6] Bovopoulos, Andreas and Aurel Lazar. "The Effect of Delayed Feedback Information on Network Performance," *Annals of Operations Research*, 1991.

[7] Bovopoulos, Andreas and Einir Valdimarrson. "Performance Evaluation of a Traffic Control Module for ATM Networks," Washington University Computer Science Department, WUCS-91-22.

[8] Bovopoulos, Andreas. "Optimal Resource Allocation for Markovian Queueing Networks: the Complete Information Case," *Stochastic Models*, 1991.

[9] Bubenik, Rick, John DeHart and Mike Gaddis. "Multipoint Connection Management in High Speed Networks," *Proceedings of Infocom*, 4/91.

[10] Cox, Jerome R. and Jonathan Turner. "Project Zeus: Design of a Broadband Network and its Application on a University Campus," Washington University Computer Science Department, WUCS-91-45.

[11] Doshi, B.T., Johri, P.K., Netravali, A.N. and Sabnani, K.K., "Error and Flow Control Performance of a High Speed Protocol," preprint, 1991.

[12] J. Filipiak, "Access Protection for Fairness in a Distributed Queue Dual Bus Metropolitan Area Network," *Proceedings of the International Conference on Communications*, 6/89.

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

[14] Gong, Larry. "Segment Streaming for Efficient Pipelined Televisualization," Washington University Computer Science Department, WUCS-90-39.

[15] Gong, Larry, Zubin Dittia and Guru Parulkar. "A Recurrence Model for Asynchronous Pipeline Analysis," Washington University Computer Science Department, WUCS-91-14.

[16] Gong, Larry, Zubin Dittia and Guru Parulkar. "An Novel Interprocess Communication Paradigm for Pipelined Televisualization," *Proceedings of SIGGraph*, 7/91.

[17] Griswold, Victor. "Core Algorithms for Autonomous Monitoring of Distributed Systems," Washington University Computer Science Department, DSc thesis, 2/91.

[18] E. L. Hane, A.K.Choudhury, N. F. Maxemchuk, "Improving the Fairness of the Distributed-Queue-Dual-Bus Networks," *Proceedings of IEEE INFOCOM*, 6/90.

[19] "Implementation of a Gigabit Packet Switch Element," Washington University Computer and Communications Research Center, WUCCRC-91-3.

[20] 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.

[21] Kapoor, Sanjay. "Design of an ATM-FDDI Gateway," Washington University Computer Science Department, MS thesis, 8/91.

[22] Kapoor, Sanjay and Guru Parulkar. "Design of an ATM-FDDI Gateway," *Proceedings of SIGComm*, 9/91.

[23] Kapoor, Sanjay and Guru Parulkar. "Design of an ATM-FDDI Gateway," Proceedings of the ACM SIGCOMM'91. Also, Technical Report WUCS-91-11, Department of Computer Science, Washington University.

[24] Kapoor, S., "Design and Simulation of an ATM-FDDI Gateway," MS Thesis, Department of Electrical Engineering, Washington University.

[25] Kumar, Lakshmana. "A 3-Tier Fairness Protocol: A Proposed Solution for the Unfairness Problem in DQDB MANs," Washington University Computer Science Department, MS thesis, 8/91.

[26] Kumar, Lakshmana and Andreas Bovopoulos. "An Access Protoection Solution for Heavy Load Unfairness in DQDB," Washington University Computer Science Department, WUCS-91-39. Bell Communications Research, 1987.

[27] Mazraani, Tony and Guru Parulkar. "Specification of a Multipoint Congram-Oriented High Performance Internet Protocol," *Proceedings of IEEE INFOCOM 90*, 6/90.

[28] Mazraani, Tony. "High Speed Internet Protocols and Resource Management in the Ethernet," MS Thesis, Department of Electrical Engineering, Washington University.

[29] Mazraani, Tony and Guru Parulkar. "Performance Analysis of the Ethernet Under Conditions of Bursty Traffic," Washington University Computer Science Department, WUCS-91-12.

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

[31] Parulkar, G.M. (editor), "The Connection-oriented Internetworking: Collaboration Plan," Technical Note, Department of Computer Science, Washington University in St. Louis, 1989.

[32] Parulkar, Guru and Jonathan Turner. "Towards a Framework for High Speed Communication in a Heterogeneous Networking Environment," *IEEE Network* 3/90.

[33] Parulkar, Guru. "The Next Generation of Internetworking," ACM SIGCOMM, *Computer Communication Review*, January 90.

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

[35] Sterbenz, James P.G. and G.M. Parulkar, "Axon: A High Speed Communication Architecture for Distributed Applications", *INFOCOM'90*, Vol.II, IEEE Comp.Soc., Wash., D.C., June 1990, pp. 415–425.

[36] Sterbenz, James P.G. and G.M. Parulkar, "Axon Network Virtual Storage for High Performance Distributed Applications", *10th ICDCS*, IEEE Comp.Soc., Wash., D.C., June 1990, pp. 484–492.

[37] Sterbenz, James P.G. and G.M. Parulkar, "Axon: Application-Oriented Lightweight Transport Protocol Design", *ICCC'90*, ICCC, Narosa Publishing House, New Dehli, India, Nov. 1990, pp. 379–387.

[38] Sterbenz, James P.G. and G.M. Parulkar, "Axon: Host–Network Interface Architecture for Gigabit Communications", in *Protocols for High Speed Networks, II*, Marjory J. Johnson ed., IFIP, Elsevier (North-Holland), 1991, pp. 211–236.

[39] Sterbenz, James and Guru Parulkar "Host-Network Interface Architecture for Gigabit Communications," in *Protocols for High Speed Networks, II*, Marjory Johnson (ed.), Elsevier Science Publishers, 1991.

[40] Sterbenz, James. "AXON: a High Speed Communication Architecture for Distributed Applications," Washington University Computer Science Department, DSc thesis, 10/91.

[41] Sterbenz, James, Anshul Kantawala and G.M. Parulkar, "Axon Network Host Network Interface Functional Simulation," Washington University Computer and Communications Research Center, WUCCRC-91-2.

[42] 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.

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

[44] Turner, Jonathan. "A Proposed Bandwidth Management and Congestion Control Scheme for Multicast ATM Networks," Washington University Computer and Communications Research Center, WUCCRC-91-1.

[45] Turner, Jonathan. "Packet Switch with Broadcasting Capability for ATM Networks," U.S patent application, 1/91.

[46] Turner, Jonathan. "Data Packet Resequencer for a High Speed Data Switch," U.S patent application, 3/91.

[47] Turner, Jonathan. "Nonblocking Multicast Switching System," U.S patent application, 4/91.

[48] Turner, Jonathan. "Resequencing Cells in an ATM Switch," Washington University Computer Science Department, WUCS-91-21.

[49] Turner, Jonathan. "A Practical Version of Lee's Multicast Switch Architecture," Washington University Computer Science Department, WUCS-91-46.

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

[51] Witte, Ellen. "A Quantitative Comparison of Architectures for ATM Switching Systems," Washington University Computer Science Department, WUCS-91-47.