ADVANCED NETWORKING AND DISTRIBUTED SYSTEMS DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

ANNUAL TECHNICAL REPORT

May 31, 1992

Principal Investigator: Leonard Kleinrock

Computer Science Department
School of Engineering and Applied Science
University of California
Los Angeles

ADVANCED NETWORKING AND DISTRIBUTED SYSTEMS DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

ANNUAL TECHNICAL REPORT

May 31, 1992

Principal Investigator: Leonard Kleinrock

Computer Science Department
School of Engineering and Applied Science
University of California
Los Angeles

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* ADVANCED NETWORKING AND DISTRIBUTED SYSTEMS Annual Technical Report		5. TYPE OF REPORT & PERIOD COVERED Annual Technical Report June 1, 1991 - June 1, 1992
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Leonard Kleinrock		8. CONTRACT OR GRANT NUMBER(s) MDA 972-91-J-1011
9. PERFORMING ORGANIZATION NAME AND ADDRESS School of Engineering & Applied Science University of California, Los Angeles Los Angeles, CA 90024-1596		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DARPA Order No. 7728
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA 3701 N. Fairfax Drive Arlington, VA 22203-1714		12. REPORT DATE
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)*
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release: Distribution Unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ADVANCED SYSTEMS LABORATORY

ANNUAL TECHNICAL REPORT

June 1, 1992

Contract Number: MDA 972-91-J-1011
DARPA Order Number: 7728
Contract Period: June 1, 1991 to May 31, 1994
Report Period: June 1, 1991 to May 31, 1992

Principal Investigator: Leonard Kleinrock

(310) 825-2543

Computer Science Department
School of Engineering and Applied Science
University of California, Los Angeles

Sponsored by

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

ADVANCED NETWORKING AND DISTRIBUTED SYSTEMS

Defense Advanced Research Projects Agency

Annual Technical Report

May 30, 1992

This Yearly Technical Report covers research carried out on the Advanced Networking and Distributed Systems Contract at UCLA under DARPA Contract Number MDA 972-91-J-1011 covering the period from June 1, 1991 through May 30, 1992. Under this contract we have the following statement of work comprising five tasks:

STATEMENT OF WORK

Topic A: High Speed Networking

Task A1: Fast Packet Switching Using Multistage Interconnection Networks

We propose to investigate the performance of a variety of Multistage Interconnection Networks such as the Starlite network. We will develop analytical models to evaluate the throughput and response time of the overall traffic in the case of uniform traffic as well as certain forms of hot spot traffic. We will also evaluate the behavior of Message Combining to eliminate the effects of hot spots. A transformation and superposition method is being developed to be used with the analytical model to evaluate any given general traffic pattern (e.g., multiple hot spots). A delay model analysis comparing the discarding switch and the blocking switch will also be developed. We also propose to study a structured buffered pool scheme to prevent normal traffic from being blocked by the saturated tree caused by hot spot traffic.

Task A2: Analysis Of Competing Lightwave Networks

The use of Wavelength Division Multiple Access (WDMA) optical switching for high-speed packet networks is a predictable development in the evolution of fast packet switching. We propose to evaluate the behavior of single-hop WDMA optical switching, using agile receiver filters. Whereas our main thrust will be on these single-hop structures, we will also look at multi-hop access using fixed filters. We will compare the response time, blocking and throughput for each.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

TOPIC B: ARCHITECTURE AND PARALLEL PROCESSING

Task B1: Performance Of Boolean n-Cube Interconnection Networks

We propose to evaluate the performance of Boolean n-cube interconnection networks for parallel processing systems. The focus will be on data communication issues rather than on processing issues. By exploiting the homogeneity property of Boolean n-cube interconnection networks, we can design non-blocking routing algorithms with limited size buffers. A technique called referral is used to guarantee that every node accepts all the messages transmitted from its neighbors. This type of routing algorithm is critical in any implementation. Store-and-forward is one such routing algorithm. In this scheme, time is divided into cycles to which the network is synchronized. In each cycle every node simultaneously transmits some of its stored messages to its neighbors. An analytical model will be developed to predict the network performance under different traffic patterns. We also intend to design an intelligent routing algorithm to improve the performance. Another routing scheme to consider is a modified version of virtual cut-through. Virtual cut-through is a scheme such that when a message arrives at an intermediate node and its selected outgoing channel is free, then the message is sent to the adjacent node before it is completely received at this intermediate node. Therefore, the delay due to unnecessary buffering in front of an idle channel is avoided. Modified virtual cut-through is also a non-blocking algorithm. We will investigate the (positive or negative) effect of adding additional buffers to a node in this case. We are further interested in non-uniform traffic problems in Boolean n-cube networks.

We also propose to study the performance of these networks in a hostile and/or unreliable environment. In this environment, nodes and links may disappear and also unreliable (i.e., noisy) transmissions may occur.

Task B2: Distributed Simulation

Parallel asynchronous simulation methods (such as Time Warp) offer an optimistic alternative to synchronous conservative approaches to distributed simulation. We propose to evaluate the speedup of P processors conducting a parallel asynchronous simulation using analytic and simulation tools. We already have an exact solution for the case of two processors (P=2). Also, we have upper bounds on the best one can do by letting the P processors run ahead of each other as compared to forcing them to synchronize at every step. We are interested in extending the results to P processors and to include the effect of queued messages. Furthermore, we propose to investigate the use of the linear Poisson process as a model for these systems.

Task B3: A New Model Of Load Sharing

We are interested in studying the behavior of interacting processes which gobble up processing resources in their neighborhood. In particular, if we begin with a one-dimensional world, we can place processes on a ring, where there is a quantity of processing power distributed uniformly around the ring. A process requires a changing amount of processing capacity. As its needs increase, the process attempts to grow in both directions along the ring until it either has enough capacity, or it bumps into another process moving in its direction, in which case they both stop moving toward each other. As time progresses, a process may or may not have all the capacity it needs. The object is to study the response time of jobs represented by such processes in a limited resource, competitive environment. Clearly, this model extends to higher dimensions, and we propose to study the case where processors are distributed over a multi-dimensional hypersphere. The effect of distributed load sharing in this environment will be evaluated.

Accomplishments for this Period

During we have graduated 2 Ph.D. students, have had 7 papers published, 7 papers accepted and 5 papers submitted to the professional literature. Progress in this research effort is moving along very nicely.

We have encountered no obstacles to our research and have in fact made excellent progress toward our stated goals. In particular with respect to fast-packet switching using multistage interconnection networks, we have indeed carried out the analysis of finite and infinite buffered networks with arbitrary traffic patterns and a variety of internal operating procedures. An example of this work is contained in the attached report by Lin and Kleinrock entitled "Performance Analysis of Finite Buffered Multistage Interconnection Networks with a General Traffic Pattern".

In the area of lightwave networks, we have carried out the evaluation of wavelength division multiplexing for local area networks and for passive star topologies. An example of this work is contained in the attached publication by Lu and Kleinrock entitled "A Wavelength Division Multiple Access Protocol for High Speed Local Area Networks with a Passive Star Topology".

In the area of Boolean n-cube interconnection networks, we have studied routing in such networks under stable conditions with finite buffers (the typical case) and then further extended these studies to deal with the case of unreliable nodes, thereby providing a fault tolerant routing procedure, the paper by Horng and Kleinrock entitled "Fault Tolerant Routing with Regularity Restoration in Boolean n-Cube Interconnection Networks" is included in the appendix of this report as an example of that research.

In the area of distributed simulation we have carried out extensive analyses of the behavior of time warp systems for the two processor case; these are exact results and have led the way for considerable research in this area. The enclosedf paper by Felderman and Kleinrock "Two Processor Time Warp Analysis: Some Results on a Unifying Approach" summarizes some of our results to date.

We have begun to look at some of the new issues involved in gigabit networks since they are such an important part of the advanced networking arena these days. The paper by Kleinrock entitled "The Latency/Bandwidth Tradeoff In Gigabit Networks" is a summary of our preliminary thinking in this area.

Below we list our publications for this period. Progress continues in all these areas and we are very much encouraged with the prospects for additional insight and results in a variety of these systems. In addition we have launched a related study on the collective behavior of a large number of mobile automata whose behavior is characteristic of advanced networking and advanced distributed systems. Progress in this area will be reported in future technical reports.

ANDS PUBLICATIONS
DARPA CONTRACT - MDA 972-91-J-1011

Computer Science Department
University of California, Los Angeles
Professor Leonard Kleinrock

June 1, 1991 to May 31, 1992

Ph.D DISSERTATIONS COMPLETED

1. Shen, Shioupyn, "The Virtual-Time Data-Parallel Machine", December 1991.

2. Horng, Ming-yun, "Analysis of Boolean n-Cube Interconnection Networks For Multiprocessor Systems", March 1992

PUBLISHED PAPERS

1. Lin, T.I. and L. Kleinrock, "Performance Analysis of Finite-Buffered Multistage Interconnection Networks with a General Traffic Pattern," *1991 ACM Sigmetrics,* Conference on Measurement and Modeling of Computer Systems, May 21-24, 1991, San Diego, CA.

2. Horng, Ming-yun and L. Kleinrock, "On the Performance of a Deadlock-Free Routing Algorithm for Boolean n-Cube Interconnection Networks with Finite Buffers," *1991 International Conference on Parallel Processing,* pp. III-228-III-235, August 12-16, 1991, Pennsylvania State University.

3. Horng, Ming-yun and L. Kleinrock, "Fault Tolerant Routing With Regularity Restoration in Boolean n-Cube Interconnection Networks" *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing* Dallas, Texas, December 2-5, 1991, pp. 458-465.

4. Felderman, R. and L. Kleinrock, "Two Processor Conservative Simulation Analysis," published *1992 PADS Workshop,* January 1992.

5. Lu, J. and L. Kleinrock, "An Access Protocol for High-Speed Optical LANs." *ACM CSC '92* Kansas City, MO, March 3-5, 1992, pp. 287-293.

6. Lu, J. and L. Kleinrock, "On The Performance Of Wavelength Division Multiple Access Networks", *ICC'92.*

7. Kleinrock, L., "The Latency/Bandwidth Tradeoff In Gigabit Networks", *IEEE Communications Magazine,* April 1992, Vol. 30, No.4, pp.36-40.

PAPERS ACCEPTED FOR PUBLICATION

1. Lu, J and L. Kleinrock "Performance Analysis of Single-Hop Wavelength Division Multiple Access Networks" accepted to *Journal of High-Speed Networks* 1992.

2. Kleinrock, L. and R. Felderman, "Two Processor Time Warp Analysis: A Unifying Approach," accepted to *International Journal in Computer Simulation.*

3. Huang, J.H. and L. Kleinrock, "Performance Evaluation of Dynamic Sharing of Processors in Two-Stage Parallel Processing Systems," accepted for publication in *IEEE Transactions on Parallel and Distributed Systems.*

4. Lu, J. and L. Kleinrock, "A Wavelength Division Multiple Access Protocol for High-speed Local Area Networks with a Passive Star Topology," accepted *Performance Evaluation,*

5. Shen, S. and L. Kleinrock, "The Virtual Time Data-Parallel Machine" accepted *IEEE*

6. Felderman, R. and L. Kleinrock, "Two Processor Time Warp Analysis: Capturing the Effects of Message Queueing and Rollback/State Saving Costs," accepted for publication in the *ACM Transactions on Modeling and Computer Simulation,*

7. Felderman, R. and L. Kleinrock, "Bounds and Approximations for Self-Initiating Distributed Simulation Without Lookahead," accepted to *ACM Transactions on Modelling and Computer Simulation, special issue on Distributed and Parallel Simulation Performance,*

PAPERS SUBMITTED FOR PUBLICATION

1. Green, J. and L. Kleinrock, "Static Load Sharing of Processors in Broadcast Networks", submitted for publications in the *IEEE Transactions on Parallel and Distributed Systems,*

2. Kleinrock, L. and W. Korfhage, "Collecting Unused Processing Capacity: An Analysis of Transient Distributed Systems," submitted for publication in the *IEEE Transactions on Parallel and Distributed Systems,*

3. Mehovic, Farid and L. Kleinrock, "An Approach to Modeling Optimistic Concurrency Control," submitted for publication in the *Communications of the ACM,*

4. Powley, C., C. Ferguson and R. Korf, "Depth-First Heuristic Search on a SIMD Machine," submitted to *AI Journal,*

5. Shen, S. and L. Kleinrock, "The Effect of Network Topology On the Performance of Load Balancing in Distributed Systems" submitted to *Performance Evaluation.*

Fault-Tolerant Routing with Regularity Restoration in Boolean n-Cube Interconnection Networks*

Ming-yun Horng and Leonard Kleinrock
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024

Abstract

This paper proposes a set of techniques to restore the regularity of a Boolean n-cube network in the presence of node failures, and algorithms to effectively route messages among the surviving nodes. An analytical model to evaluate the degradation of a damaged network is also presented.

One way to restore the regularity of a damaged Boolean n-cube network is by simply disabling the nodes with more than one bad neighbor. The remaining network is called a "1-degraded subnet." A very simple optimal-path routing algorithm, which requires each node to know only its neighbor's status, is developed for such a subnet. Since many nonfaulty nodes may have to be disabled in constructing a 1-degraded subnet, we further develop a heuristic algorithm to restore the network's regularity by constructing a "subnet connected with optimal paths (SCOP)," where only a few nodes must be disabled. The routing algorithm used in 1-degraded subnets also works for SCOPs. To preserve the processing power of the network, we also propose a two-level hierarchical fault-tolerant routing scheme without disabling any nodes.

1 Introduction

A major problem in designing a multiprocessor system is to construct a reliable interconnection network which provides efficient routing of messages among processors. Recently, the Boolean n-cube network (also known as the hypercube network) has become a widely accepted interconnection architecture due to its topological properties as discussed, for example, in [1]. Several research and commercial systems built on this type of interconnection are now available [2].

The success of the simple routing algorithms [3] used in Boolean n-cube networks is based on the networks' regularity properties. Although an interconnection network is usually operated in a well-protected environment, faults may occur. When some nodes or communication links fail, the regularity of this "damaged" network is destroyed and the routing algorithm may no longer be applicable. To build a reliable multiprocessor system, the presence of fault-tolerant routing to ensure successful communications between any pair of nonfaulty nodes is essential. Moreover, since the channel speed of an interconnection network is very high, the amount of time that a node can afford to spend in making routing decisions is severely constrained. It is important to have the routing algorithm as simple as possible and hardware realizable.

To successfully route messages in a damaged Boolean n-cube network, either the surviving nodes or the messages must be equipped with information about the locations of the faults. Several algorithms requiring each node of the network to know only the status of its local components (links and nodes) have been presented in [4, 5]. However, the limitation of these approaches is that either the total number of faulty components is very restricted (e.g. less than n) or the number of hops traversed by a message may grow without bound. These problems can be solved by providing each node with more information and having it compute a "safe" route for each message. Algorithms that require each node to know the global status of the network have been reported in [6]. One can even assume the surviving part of the network has an arbitrary topology, in which case each node maintains a routing table as used in networks such as the ARPANET [7, 8]. However, as the network grows in size, the amount of storage space and time needed to maintain and update these routing tables become prohibitive [9].

Since a number of faults in a richly-connected Boolean n-cube network may not destroy its entire regularity, the routing algorithm may take advantage of

*This work was supported by the Defense Advanced Research Projects Agency under Contract MDA 903-87-C0663, Parallel Systems Laboratory.

the remaining topological regularity. Chen and Shin [10] developed and analyzed a set of fault-tolerant routing algorithms based on the depth-first search principle. In their algorithms, each message contains a tag to keep track of the path traveled so far to avoid visiting a node more than once. To tolerate more than $n-1$ faults, a more complicated procedure is required to guide backtracking whenever a message reaches a dead end. Thus, the length of the packets is variable and the computation overhead is not trivial. To further guarantee that every message is routed to its destination via a shortest path, every node must be equipped with nonlocal status [11].

In this paper, we begin with a queueing model to evaluate the degradation of a damaged Boolean n-cube network with node failures. We then develop a set of techniques to restore the regularity of the network, and algorithms to effectively route messages among the surviving nodes. Our algorithms work under any number of faults as long as the network remains connected.

We restore the regularity of a damaged Boolean n-cube network by disabling the nodes with more than one bad neighbor. The remaining network is called a "1-degraded subnet." We then develop a very simple routing algorithm for such a 1-degarded subnet. With this algorithm, each node only needs to know the status of its neighbors, and every message is routed to its destination via an "optimal path" (to be defined).

Though the 1-degraded subnet can easily be constructed in a distributed manner, many nonfaulty nodes may have to be disabled. We develop a heuristic algorithm to construct a subnet in which every pair of surviving nodes are connected with at least an "optimal" path. In this paper, such a subnet is called a SCOP (Subnet Connected with Optimal Paths). We show that only a small number of nonfaulty nodes will be disabled. The optimal-path routing algorithm for a 1-degraded subnet also works in a SCOP. Some simulation results are presented and compared with a lower bound we develop.

To fully preserve the processing power of the network, we further develop a two-level hierarchical fault-tolerant routing scheme without disabling any nonfaulty nodes. With this approach, a non-optimally connected network is decomposed into a set of clusters such that every cluster forms a subcube with the same property as a SCOP. Each node maintains a small routing table in which every entry of the table corresponds to a destination cluster. Messages are first routed to their destination clusters by use of these routing tables. After a message has arrived at its destination cluster, it is then routed to its destination via an optimal path.

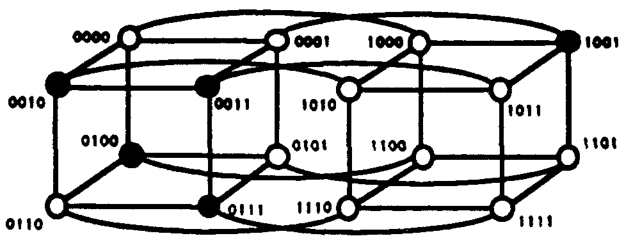


Figure 1: A Boolean 4-cube network with node faults.

2 Preliminaries

A Boolean n-cube network consists of 2^n nodes, each addressed by an n-bit binary number from 0 to 2^n-1. (See Figure 1, where faulty nodes are drawn as black dots.) Nodes are interconnected in such a way that there is a bidirectional link between two nodes, say i and j, if and only if $|i-j| = 2^k$ for some integer k from 0 to $n-1$; in this case, we say that these two nodes are linked together in dimension k. For example, in a Boolean 4-cube network, nodes 1000 and 1010 are linked together in dimension 1. It can be seen that by removing all the links in any particular dimension, a Boolean n-cube network is separated into two $(n-1)$-cube networks.

Every "subcube" in a Boolean n-cube network can be uniquely addressed by a string of n symbols drawn from the set {0, 1, X}, where X is a don't care symbol [10]. For example, in a Boolean 4-cube network, nodes 0001, 0011, 0101 and 0111 form a subcube addressed by 0XX1. A node is itself a subcube.

The Hamming distance between any two nodes is defined as the number of bits which differ between their addresses. The length of a path from one node to another is defined as the number of links on the path. An "optimal path" between two nodes is a path whose length is equal to their Hamming distance. A node might not be able to communicate with another via an optimal path in a damaged network. For example, in Figure 1, node 0110 cannot communicate with node 0101 via an optimal path. However, they are able to communicate with each other via the path through nodes 1110, 1100 and 1101. This path is called a shortest path since, among all the possible remaining paths between these two nodes, its length is minimal.

Routing algorithms for Boolean n-cube networks can be found, for example, in [3]. Let the *header* (address portion) of a newly generated message be the exclusive-OR of the message's source and destination addresses. Every one-bit in the header corresponds to a *valid* dimension over which the message can be sent one hop closer to its destination. When a message is sent over a valid dimension, the corresponding one-bit is changed to zero. Here, the selection of a possible valid dimension for transmission can be adaptive to traffic. A message reaches its destination when its header contains only zeroes. It is clear that, with this algorithm, all messages are routed to their destinations via their optimal paths. However, this routing algorithm cannot work for such a damaged network as shown in Figure 1 since all the optimal paths between nodes 0110 and 0001 are blocked.

In this paper, we make the following assumptions:

- The remaining network is connected.
- Since nodes (or processors) are more complex than links and therefore have higher failure rates, we assume only node failures. To consider a link failure between two nonfaulty nodes, one may disable one of these two nodes. In [12], our algorithms are extended to consider link failures.
- Each node knows the status of its neighboring nodes.
- A node cannot transmit a message to a faulty neighboring node.
- Messages are only destined for nonfaulty nodes.

3 Degradation of Networks with Node Faults

In this section, we present a simple queueing model to evaluate how much a network is degraded by node faults. In many cases it is likely that the failure rate of multiprocessor systems is very small. Let each node fail independently with probability p. We also assume that the arrival of input messages to each node follows a Poisson process with a rate of λ messages per unit time; message lengths are random and drawn independently from an exponential distribution. Since a link survives if and only if both nodes at its ends survive, we have

$$\text{Prob}[\textit{A link survives.}] = (1-p)^2.$$

Figure 2 shows a Boolean n-cube network with a "cut" in its third dimension. The expected number of surviving links crossing any dimension is given by

$$2^{n-1}(1-p)^2. \tag{1}$$

We further assume that messages are uniformly destined to all other surviving nodes in the network. Thus,

$$\begin{aligned} \alpha &\triangleq \textit{traffic intensity from a given source} \\ &\quad\ \textit{to a particular destination} \\ &= \frac{\lambda}{2^n(1-p)-1}. \end{aligned}$$

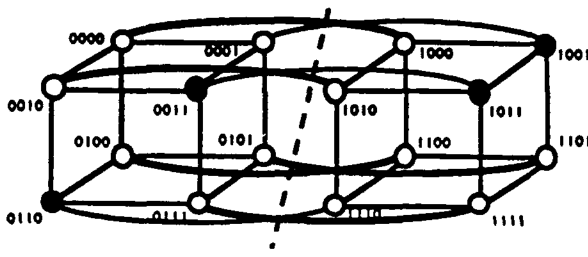


Figure 2: A cut in dimension 3. (Surviving links crossing the cut are shown as heavy lines).

From Figure 2 we note that every message which is generated from a node in the left subcube which is destined to a node in the right subcube must travel over one of the surviving links in order to cross the "cut". If a message travels along an optimal path between these two nodes, the message must travel across the "cut" exactly once. We further assume that the traffic crossing this "cut" is perfectly balanced. The average number of surviving nodes in each subcube is $2^{n-1}(1-p)$. Each will send α units of traffic to every other surviving node in the network. Thus each node will send $\alpha[2^{n-1}(1-p)]$ units of traffic across each cut. Since there are $2^{n-1}(1-p)$ nodes in each subcube doing this, and traffic is balanced on each link, we have

$$\begin{aligned} \rho &= \textit{Traffic load per channel} && (2) \\ &= \frac{[2^{n-1}(1-p)]^2\alpha}{2^{n-1}(1-p)^2} && (3) \\ &= \frac{\lambda}{2(1-p)-2^{1-n}}. && (4) \end{aligned}$$

Here, the channel is unidirectional. Obviously, this model yields an optimistic bound.

We apply Kleinrock's *Independence Assumption* [8] which is often used in the delay analysis of communication networks. This assumption states that each time a message is received at a node within the network, its transmission time is chosen independently from an exponential distribution. We assume the mean transmission time of a message equals one unit of time. Thus, each channel is modeled as an M/M/1 system with Poisson arrivals at a rate $\lambda/[2(1-p)-2^{1-n}]$ and with an exponential service time whose mean is one unit of time. In order for this system to be stable, we require that $\rho < 1$, that is,

$$\lambda < 2(1-p) - 2^{1-n}. \tag{5}$$

We define

$$\gamma = \textit{Throughput of the network,}$$

then we have

$$\begin{aligned}\gamma &= \lambda 2^n(1-p) & (6)\\ &< 2[2^n(1-p)-1](1-p), & (7)\end{aligned}$$

where $2[2^n(1-p)-1](1-p)$ is clearly the mean network communication capacity. The mean message delay is then given by [8]

$$\begin{aligned}T &= \sum_{i=1}^{M}\frac{1}{\gamma}\frac{\rho}{1-\rho}\\ &= M\left\{\frac{1}{\lambda 2^n(1-p)}\right\}\left\{\frac{\frac{\lambda}{2(1-p)-2^{1-n}}}{1-\frac{\lambda}{2(1-p)-2^{1-n}}}\right\}\end{aligned}$$

where M is the number of surviving channels in the network ($M = 2^{n-1}(1-p)^2 2n$). Thus,

$$T = \frac{n(1-p)}{2(1-p)-2^{1-n}-\lambda}. \qquad (8)$$

We further obtain the following approximations for $n >> 1$.

$$\rho \approx \frac{\lambda}{2(1-p)}, \qquad (9)$$

$$\gamma < 2^{n+1}(1-p)^2, \qquad (10)$$

and

$$T \approx \frac{n(1-p)}{2(1-p)-\lambda}. \qquad (11)$$

In Figure 3 we show the mean message delay obtained from our optimistic model for a Boolean 4-cube network with two different failure rates. We also ran a flow deviation program [14] to find the minimal achievable delay. We find that our assumptions are appropriate if failure rates are small.

Moreover, in most queueing systems, two performance measures, response time and throughput, compete with each other. Typically, by raising the throughput of the system, which is desirable, the mean response time is also raised, which is undesirable. Here, we combine the throughput and the mean message delay of the network into a single measure, *power*, which is defined as follows [13].

$$\textit{Power} = \frac{\textit{Throughput of the network}}{\textit{Mean message delay}}.$$

A system is said to be operating at an optimal point if the power at that point is maximized. For $n >> 1$, we find that power is maximized when $\lambda = 1-p$ which is equal to half the maximum allowed throughput per node, as found in [13].

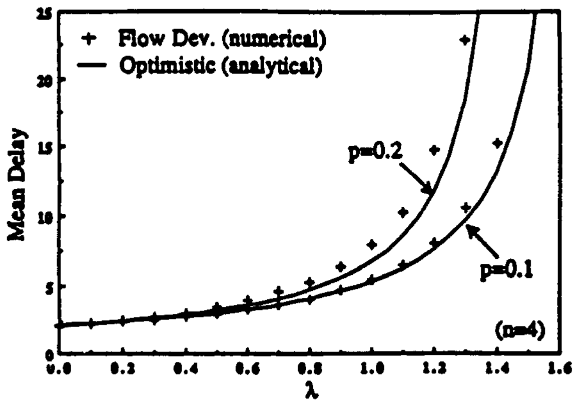


Figure 3: Minimal achievable delay of a Boolean 4-cube network with two different failure rates.

4 Routing in 1-Degraded Subnets

A network is said to be k-degraded if every surviving node in the network has at most k "bad" (to be discussed) neighbors. A damaged network can easily be made k-degraded in a distributed manner as follows: Every surviving node (or nonfaulty node initially) has a list which gives the status of its neighboring nodes. Every surviving node checks its list and disables itself if it has more than k "bad" neighbors; in this case, it must inform all its surviving neighbors of the change in its status. Every surviving node keeps updating its list until it disables itself or the disabling process stops. Here, during each step of the iteration, the "bad" nodes include all faulty nodes and all nodes which have been disabled in previous iterations. We call the remaining network a "k-degraded subnet."

We now present a very simple adaptive routing algorithm for 1-degraded Boolean n-cube subnets. This algorithm requires each node to know only its neighbors' status. We let *neighbor_status* be an n-bit binary number in which a bit is set to one if its corresponding neighbor is surviving. Otherwise, the bit is reset to zero. This routing algorithm is shown in Figure 4, where "&"" is a bit-wise AND function. We note that, with this algorithm, every message is routed to its destination along an optimal path.

The proof that this routing algorithm works for 1-degraded Boolean n-cube subnets is as follows: If a node receives a message with more than one one-bit in its header, the node surely can find a valid dimension (or channel) to transmit the message. If the message has only a single one-bit in its header, then the corresponding neighbor must be surviving. Otherwise, the assumption that messages are only destined for surviving nodes is violated.

```
When a message is received,
if ( header = 0 )
  Send the message to the local processor.
else
  valid_channels <- header & neighbor_status.
  Randomly select a 1-bit from valid_channels.
  Change the selected 1-bit in the header to 0.
  Send the message over the selected channel.
```

Figure 4: The optimal-path routing algorithm for 1-degraded subnets

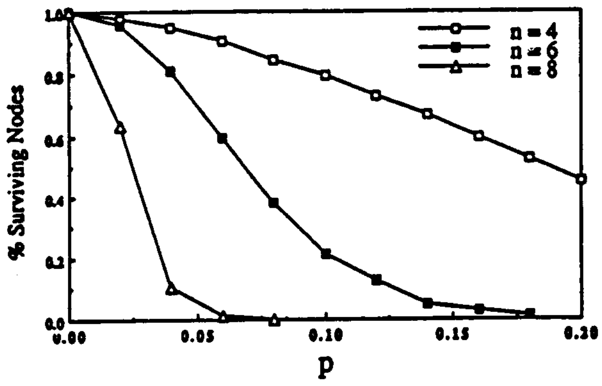


Figure 5: Percentage of surviving nodes in the 1-degraded subnets.

This approach works well in the situation where a whole cluster of nodes has been "bombed out." As an example of such spatially correlated faults, one may consider a power-supply failure which disables the entire cluster of nodes supported by it.

The disadvantage of this approach for 1-degraded subnets is that, in a large Boolean n-cube network with moderate failure rates, since every node has a large number of neighbors, the probability that a node has more than one bad neighbor can be large. As a result, many nodes may have to be disabled; hence, the computational power of the system is significantly reduced. Figure 5 shows, for networks of different sizes, the percentage of nodes that remain after the disabling iteration settles down, given that nodes initially fail independently with a given probability.

5 Construction of a SCOP

In this section, we develop a heuristic algorithm to construct a subnet where every pair of surviving nodes is connected with at least one optimal path (or we say every pair of nodes is *optimally connected*). We call such a subnet a SCOP (Subnet Connected with Optimal Paths). We show that our routing algorithm for 1-degraded subnets works in such a subnet.

In a Boolean n-cube network, a node and its k neighbors can uniquely identify a Boolean k-subcube, where $0 \leq k \leq n$. We note that such a subcube is the smallest subcube containing the node and its k neighbors. For example, in a Boolean 4-cube network, the node 0110 and nodes 0100 and 0111 can identify the subcube 01XX. Moreover, any two nodes which are k hops away in distance can also identify a Boolean k-subcube.

We assume there is a central control unit that collects information from every surviving node of the network and makes decisions about how to disable a node. Here is a heuristic algorithm for constructing a SCOP: We let *List[i]* be a check-list which contains all surviving nodes with i bad neighbors. Again, the "bad" nodes include all faulty nodes and all nodes having been previously disabled. Nodes on *List[i]* have higher priority for disablement than any other nodes on the list with smaller i. That is, the node with most bad neighbors (worst connection) has the highest priority of being disabled.

We choose a node, say node j, from the highest priority non-empty check-list, and simply find the smallest subcube containing node j and all its bad neighbors (e.g. in Figure 1, node 0110 and subcube 0XXX). It is clear that without routing through the links outside the subcube, node j cannot communicate with any other surviving nodes of the subcube. If the total number of surviving nodes in the subcube is more than 2, node j is disabled. If the number of surviving nodes in the subcube is exactly 2, we choose to disable either one of them (See [12].). Otherwise, node j is safe at this moment and is removed from the lists. A safe node may be brought back to the check-lists if any of its neighbors is disabled. The algorithm stops when every surviving node is safe. Figure 6 illustrates a resulting SCOP for the sample Boolean 4-cube network as shown in Figure 1. In [12], we show the number of surviving nodes of the SCOPs obtained from our heuristic algorithm is very close to the number of surviving nodes achievable by an exhaustive search.

We now prove the optimal-path routing algorithm we developed for 1-degraded subnet works for a SCOP. If the destination node of a message is k hops away from the node where the message is currently residing, the current node should have k valid dimensions to choose from. Since the k-subcube identified by the current node and the destination node is optimally connected, the current node must be able to find at least one valid dimension to transmit the message.

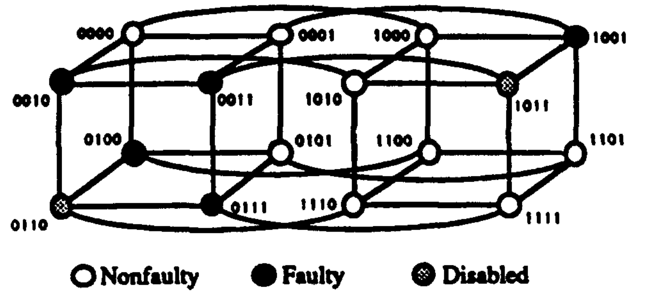


Figure 6: A SCOP of the Boolean 4-cube as shown in Figure 1.

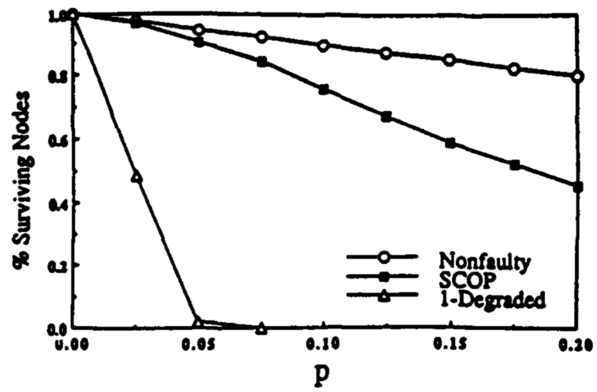


Figure 8: Comparison of percentage of surviving nodes, n=8.

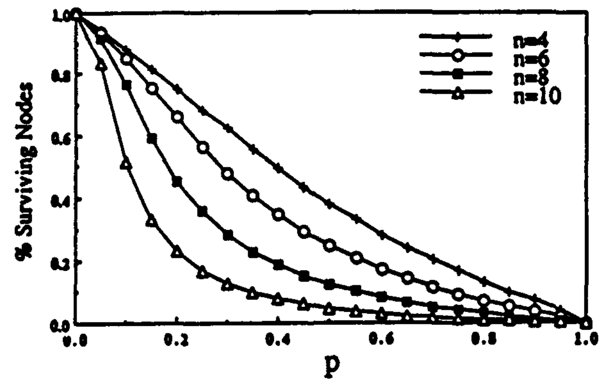


Figure 7: Percentage of nodes remaining in the SCOPs.

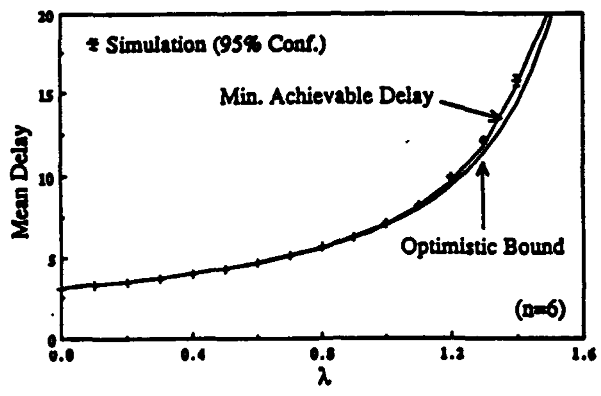


Figure 9: Mean delay of the SCOPs of Boolean 6-cube netowrks. Each network has 6 faulty nodes.

Thus, if every subcube is optimally connected, then the optimal-path routing algorithm works.

Figure 7 shows the percentage of nodes which remain in the SCOP, given that nodes initially fail independently with a given probability. Comparing this with the percentage of nodes which remain in the 1-degraded subnet, we find the number of surviving nodes is dramatically increased. In Figure 8, we compare these two disabling schemes by showing the percentage of surviving nodes in a Boolean 8-cube network.

It is very difficult to analytically evaluate the performance of arbitrary networks in a dynamic traffic environment. In this paper, routing in the SCOPs is extensively simulated. To verify the effectiveness of our routing algorithm, for each failure pattern, we also ran a flow deviation program to find the minimal achievable delay. These results are also compared with the optimistic bound obtained in Section 3. Figure 9 shows, for different input rates, the mean message delay in the SCOPs of a Boolean 6-cube network. The results with 95% confidence shown here were from 100 randomly generated patterns, each containing 6 faulty nodes. We find that the mean message delay is very close to the minimal achievable bound, which is also very close to the optimistic bound.

6 Two-level Hierarchical Routing

In this section, without disabling any nonfaulty nodes, we restore the regularity of a damaged Boolean n-cube network by decomposing the network into a set of clusters such that every cluster forms a subcube with the same property of a SCOP (i.e. every pair of the surviving nodes of a cluster is connected with at least an optimal path). A *two-level hierarchical* routing algorithm, which requires every node to maintain a small routing table, is then developed.

node	bad dimensions
0000	1 2
0001	1 3
0101	0 1
0110	0 1 2
1011	1 3

Table 1: Bad dimensions for each surviving node with more than one bad neighbor.

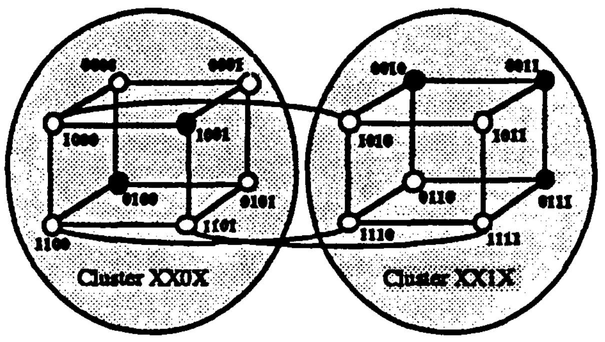


Figure 10: A two-level hierarchical structure of a 4-cube network with 5 node faults as shown in Figure 1.

6.1 The Two-level Network Decomposition

A non-optimally connected Boolean n-cube network is decomposed into a set of clusters as follows. For all the surviving nodes which have more than one bad neighbor, the central control unit counts the number of bad links in each dimension. The central control unit then chooses the dimension having the most bad links and cuts the network into two clusters along this dimension. Each of these two clusters is an $(n-1)$-subcube. A subcube must be further decomposed if not every pair of surviving nodes of the subcube is optimally connected.

Again, as an example, let us examine the Boolean 4-cube network with 5 faulty nodes as shown in Figure 1. Clearly the network is not optimally connected. In Table 1 we show, for each surviving node with more than one bad neighbor, the dimensions along which its neighbors are bad. In this example, dimension 1 has the the maximum number of bad links (i.e. 5). We decompose the network along dimension 1. As a result, the network is separated into the following subcubes: XX0X and XX1X. In this case, both of these two subcubes are optimally connected. The two-level hierarchical structure is shown in Figure 10.

6.2 Routing in the Two-level Hierarchical Network

Messages must first be routed to their destination clusters. Every surviving node maintains a cluster routing table with one entry for each destination cluster. Each entry gives the address of a destination cluster, the best outgoing channel for that cluster, and a relative weight (usually delay is used as the weight). Any algorithm (e.g. the ARPANET-like algorithm) can be used to maintain the cluster routing table.

When a node receives a transit message, if the destination node of the message does not belong to the cluster in which the node resides, the message is sent to a neighbor based on the node's cluster routing table. Otherwise, the message is routed to its destination node using the routing algorithm for 1-degraded subnets. To exploit the possible multiple paths from one node to another in a cluster and balance the network's traffic, the message is sent along a most lightly loaded channel in its destination cluster. We may further improve the performance by providing multipath routing, where each entry of the routing table gives multiple choices of outgoing channels.

6.3 Discussion

This hierarchical routing approach has the following advantages:

- No good links or nodes are eliminated. The processing power of the nonfaulty part of the network is fully maintained.

- The size of the routing table is significantly reduced from the ARPANET-like routing table. In [12], we show that the number of clusters generated by decomposition cannot exceed the number of faulty nodes in the network.

- The optimal-path routing algorithm for 1-degraded subnets works for each cluster.

- The number of hops traversed by a message is bounded.

- The increase in the mean path length caused by hierarchical routing is typically very small [12].

7 Conclusions

In this paper, we first developed a queueing model to evaluate the degradation of a damaged Boolean n-cube network with node faults. We next developed an adaptive fault-tolerant routing algorithm for 1-degraded subnets. This algorithm is very simple; it makes routing decisions based only on the node's local status. This algorithm routes every message to its destination via an optimal path.

We further exploited the remaining regularity of a damaged Boolean n-cube network and developed a heuristic algorithm to construct a subnet (i.e. SCOP) in which every pair of surviving nodes is connected with at least one optimal path. We showed the algorithm used in a 1-degraded subnet also works for a SCOP. Only a small number of nonfaulty nodes must be disabled. The performance of the optimal-path routing algorithm in SCOPs was studied. We found the mean message delay is very close to the minimal achievable bound.

To preserve all the nonfaulty nodes in the network, we also developed a two-level hierarchical routing scheme. A damaged Boolean n-cube network is decomposed into a set of clusters; each of them is a SCOP. Every surviving node in the network is required to maintain a small routing table. A two-level hierarchical routing algorithm has also been developed. This approach maintains the network's rich connection without disabling a single nonfaulty node. In [12], we show that the probability of routing a message to its destination via a shortest path is very high and that the increase in the mean path length caused by hierarchical routing is very small. More simulation results are being collected and some other performance measures such as the mean delay, the throughput of the network and hot spot problems are also being evaluated.

References

[1] Y. Saad and M. H. Schultz, "Topological Properties of Hypercubes," *IEEE Trans. Comput.*, vol. 37, pp. 867-872, July 1988.

[2] J. P. Hayes and T. Mudge "Hypercube Supercomputers," *Proc. of the IEEE*, vol. 27, pp. 1829-1841, Dec. 1989.

[3] H. Sullivan and T. R. Bashkow, "A Large Scale, Homogeneous, Fully Distributed Parallel Machine I," in *Proc. 4th Symp. Comput. Architecture*, pp. 105-117, Mar. 1977.

[4] T.-C. Lee and J. P. Hayes, "Routing and Broadcasting in Faulty Hypercube Computers," in *Proc. 3rd Conf. Hypercube Concurrent Comput. Appl.*, pp. 346-354, Jan. 1988.

[5] J. M. Gordon and Q. F. Stout, "Hypercube Message Routing in the Presence of Faults," in *Proc. 3rd Conf. Hypercube Concurrent Comput. Appl.*, pp. 318-327, Jan. 1988.

[6] M.-S. Chen and K. G. Shin, "On Hypercube Fault-Tolerant Routing Using Global Information," in *Proc. 4th Conf. Hypercube Concurrent Comput. Appl.*, pp. 83-86, Mar. 1989.

[7] J. M. McQuillan and D. C. Walden, "The ARPA Network Design Decisions," *Comput. Networks*, vol. 1, pp. 243-289, Aug. 1977.

[8] L. Kleinrock, *Queueing Systems Volume II: Computer Applications*, John Wiley and Sons, New York, 1976.

[9] L. Kleinrock and F. Kamoun, "Hierarchical Routing for Large Networks, Performance Evaluation and Optimization," *Comput. Networks*, vol. 1, pp. 155-174, Jan. 1977.

[10] M.-S. Chen and K. G. Shin, "Depth-First Search Approach for Fault-Tolerant Routing in Hypercube Multicomputers," *IEEE Trans. Parallel and Distrib. Syst.*, vol. 1, pp.152-159, Apr. 1990.

[11] M.-S. Chen and K. G. Shin, "Adaptive Fault-Tolerant Routing in Hypercube Multicomputers," *IEEE Trans. Comput.*, vol. 39, pp. 1406-1416, Dec. 1990.

[12] M.-Y. Horng, *Performance Analysis of the Boolean n-Cube Interconnection Network for Multiprocessors*, Ph.D. Dissertation, Comput. Sci. Dep., Univ. California, Los Angeles, 1991.

[13] L. Kleinrock, "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications," *Int. Conf. on Commun.*, pp. 43.1.1-43.1.10, June 1979.

[14] L. Fratta, M. Gerla, and L. Kleinrock, "The Flow Deviation Method - An Approach to Store-and-forward Communication Network Design," *Networks*, vol. 3, pp. 97-133, 1973.

Two Processor Time Warp Analysis: Some Results on a Unifying Approach*

Robert E. Felderman and Leonard Kleinrock
UCLA Computer Science Dept.
3732L Boelter Hall
Los Angeles, CA 90024-1596

Abstract

We present some results from an exact analysis of a new model for the problem of two processors running the Time Warp distributed simulation protocol. The model creates a unifying framework for previous work in this area and additionally provides some clear insight into the operation of systems synchronized by rollback.

1 Introduction

Distributed simulation has proven to be an important application for parallel processing. Accordingly, sever¬¹ algorithms have been developed to perform simula..on with multiple processors. The most well known techniques are generally classified into two types; conservative [8] and optimistic [2]. Generally speakinᵹ, optimistic simulation on multiple processors is a technique which allows each processor to proceed with its portion of a simulation independent of the other processors (optimistically assuming that the others will not interact with that processor); if, at a later time, it ᷉ıds that some other processor caused its earlier assumption to be false, it will roll back and proceed forward again. Our research focuses on the analysis of the average case behavior of Time Warp, the most well known optimistic technique.

Very little work has appeared in the literature which discusses average case behavior of Time Warp (TW). Lavenberg et al. [5] and Mitra and Mitrani [9] have examined models similar to ours, and we will address their relationship to this work in Section 5. Recently, Lin and Lasowska [6] have examined Time Warp and conservative methods by appealing to critical path analysis. Though their work provides important insights, it generates different types of results than ours. Finally, Madisetti [7] provides bounds on the performance of a two processor system where the processors have different speeds of processing and move at constant rates. Madisetti extends his model to multiple processors, something we do not address in this work.

The next section introduces our model for Time Warp. Section 3 provides its exact solution while in Section 4 we derive some performance measures. In Section 5 we examine the model as we take limits on various parameters and discuss the relationship of this work to that of Lavenberg et al. and Mitra and Mitrani. Section 6 discusses what we can learn from the model. Finally, in Section 7 we provide some concluding remarks and notes on future research directions.

2 A Model for Two Time Warp Processors

Assume we have a job which is partitioned into two processes, each of which is executed on a separate processor. As these processes are executed, we consider that they advance along the integers on the x-axis in discrete steps, each beginning at $x = 0$ at time $t = 0$. Each process independently makes jumps forward on the axis where the size of the jump is geometrically distributed with mean $1/\beta_i$ ($i = 1,2$) The amount of real time between jumps is a geometrically distributed number of time slots with parameter α_i ($i = 1,2$). After process i makes an advance along the axis, it will send a message to the other process with probability q_i ($i = 1,2$). Upon receiving a message from the other (sending) process, this (receiving) process will do the following:

Case 1: If its position along the x-axis is equal to or behind the sending process, it will ignore the message.

Case 2: If it is ahead of the sending process, it will immediately move back (i.e., "rollback") along the x-axis to the current position of the sending process.

Let $F(t)$= the position of the First process (process one) at time t and let $S(t)$= the position of the Second process (process two) at time t. Further, let

$$D(t) = F(t) - S(t).$$

*This work was supported by the Defense Advanced Research Projects Agency under Contract MDA 903-87-C0663, Parallel Systems Laboratory.

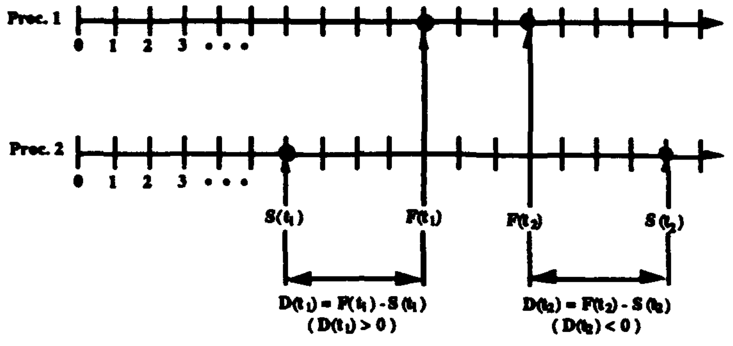


Figure 1: States of two processors at times t_1 and t_2.

$D(t) = 0$ whenever Case 2 occurs (*i.e.*, a rollback). We are interested in studying the Markov process $D(t)$. From our assumptions that $F(0) = S(0) = 0$, we have $D(0) = 0$. Clearly, $D(t)$ can take on any integer value (i.e., it certainly can go negative, see Figure 1). We will solve for

$$\begin{aligned} p_k &= \lim_{t\to\infty} P[D(t) = k] \quad k = 0, 1, 2, \ldots \\ n_k &= \lim_{t\to\infty} P[D(t) = -k] \quad k = 1, 2, 3, \ldots \end{aligned}$$

namely, the equilibrium probability for the Markov chain $D(t)$. Moreover, we will find the speedup with which the computation proceeds when using two processors relative to the use of a single processor.

This is a simple model of the Time Warp distributed simulation algorithm where two processors are both working on a simulation job in an effort to speed it up. They both proceed independently until such time as one (behind) process transmits a message in the "past" of the other (ahead) process. This causes the faster process to "rollback" to the point where the slower process is located, after which they advance independently again until the next rollback, etc. The interpretation of the model is that the position of each process on the axis is the value of the local clock (or virtual time of the message being processed) of each process. The amount of real time to execute a particular event is modelled by the geometric distribution of time slots between jumps. The jumps in virtual time indicate the increase in the virtual timestamp from one event to the next. Messages passed between processors (with probability q_i) have virtual time stamps equal to the virtual time of the sending process. Our model assumes that states are stored after every event, otherwise a rollback would not necessarily send the processor back to the time of the tardy message; rather it might have to roll back to a much earlier time, namely, that of the last saved state. Another implicit assumption is that each process always schedules events for itself. Finally, the interaction between the processes is probabilistic.

3 Discrete Time, Discrete State Analysis

In this section we provide the exact solution for the discrete time, discrete state model introduced in Section 2. Although, as we proceed, the equations may look formidable, the analysis is quite straightforward. First, we provide some definitions.

$$\begin{aligned} \alpha_i &= \text{P[}i^{th}\text{ processor advances in a time slot]} \\ \overline{\alpha}_i &= 1 - \alpha_i \\ A_1 &= \alpha_1\overline{\alpha}_2 \text{ (Only proc. 1 advances)} \\ A_2 &= \alpha_2\overline{\alpha}_1 \text{ (Only proc. 2 advances)} \\ A_3 &= \alpha_1\alpha_2 \text{ (Both advance)} \\ A_4 &= \overline{\alpha}_1\overline{\alpha}_2 \text{ (Neither advance)} \\ g_j &= \text{P[processor 1 advances } j \text{ units]} \\ &= \beta_1\overline{\beta}_1^{\,j-1} (j > 0) \quad (\overline{\beta}_1 = 1 - \beta_1) \\ f_j &= \text{P[processor 2 advances } j \text{ units]} \\ &= \beta_2\overline{\beta}_2^{\,j-1} (j > 0) \quad (\overline{\beta}_2 = 1 - \beta_2) \\ \gamma &= \text{P[procs. 1 and 2 advance the same dist.]} \\ &= \frac{\beta_1\beta_2}{1 - \overline{\beta}_1\overline{\beta}_2} \\ q_i &= \text{P[} i^{th}\text{ proc. sends a message]} \\ \overline{q}_i &= 1 - q_i \end{aligned}$$

Since the transitions in our system are quite complex (there are an infinite number of transitions into and out of each state) we choose to show the state diagram only for a simplified version of our system where $\beta_1 = \beta_2 = 1$ in Figure 2. This is the case where

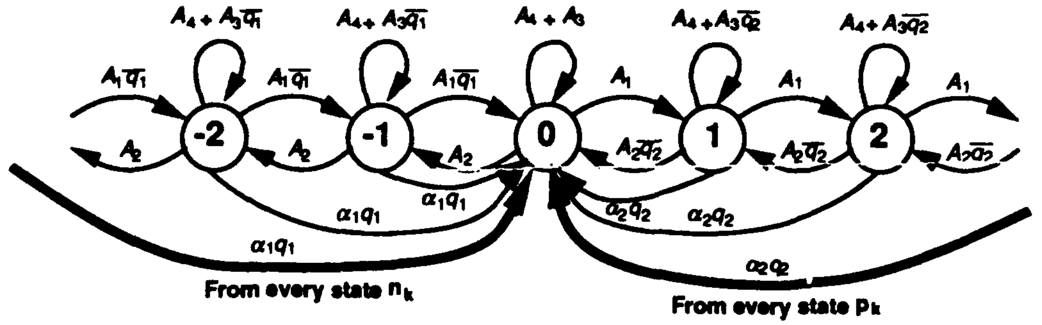


Figure 2: State Diagram for $\beta_1 = \beta_2 = 1$.

the processors only make jumps of a single step (from k to $k+1$).

The balance equations for our completely general system (no restrictions on β_i) are:

$$(A_1 + A_2 + A_3((1-\gamma) + q_2\gamma))\,p_k =$$

$$\begin{aligned}
= \quad & A_1\left(\sum_{i=0}^{k-1} p_i g_{k-i} + \sum_{i=1}^{\infty} n_i g_{k+i}\right) \\
+ \quad & A_2\bar{q}_2 \sum_{i=k+1}^{\infty} p_i f_{i-k} \\
+ \quad & A_3\bar{q}_2 \sum_{i=k+1}^{\infty} p_i \sum_{j=1}^{\infty} g_j f_{j+i-k} \\
+ \quad & A_3\bar{q}_2 \sum_{i=0}^{k-1} p_i \sum_{j=1}^{\infty} g_{j+k-i} f_j \\
+ \quad & A_3\bar{q}_2 \sum_{i=1}^{\infty} n_i \sum_{j=1}^{\infty} g_{j+k+i} f_j \qquad k \geq 1
\end{aligned}$$

$$(A_1 + A_2 + A_3((1-\gamma) + q_1\gamma))\,n_k =$$

$$\begin{aligned}
= \quad & A_2\left(\sum_{i=0}^{k-1} n_i f_{k-i} + \sum_{i=1}^{\infty} p_i f_{k+i}\right) \\
+ \quad & A_1\bar{q}_1 \sum_{i=k+1}^{\infty} n_i g_{i-k} \\
+ \quad & A_3\bar{q}_1 \sum_{i=k+1}^{\infty} n_i \sum_{j=1}^{\infty} f_j g_{j+i-k} \\
+ \quad & A_3\bar{q}_1 \sum_{i=0}^{k-1} n_i \sum_{j=1}^{\infty} f_{j+k-i} g_j \\
+ \quad & A_3\bar{q}_1 \sum_{i=1}^{\infty} p_i \sum_{j=1}^{\infty} f_{j+k+i} g_j \qquad k \geq 1
\end{aligned}$$

$$p_0 = 1 - \sum_{i=1}^{\infty} p_i - \sum_{i=1}^{\infty} n_i.$$

By using the technique of z-transforms we are able to solve explicitly for p_k and n_k. We only give the results here; the full analysis will be found in a forthcoming paper [4].

We obtain

$$p_k = C_p p_0 \left(\frac{1}{r_1}\right)^k \quad k \geq 1 \tag{1}$$

$$n_k = C_n p_0 \left(\frac{1}{s_1}\right)^k \quad k \geq 1 \tag{2}$$

$$p_0 = \frac{1}{1 + \left(\frac{C_p}{r_1 - 1}\right) + \left(\frac{C_n}{s_1 - 1}\right)}. \tag{3}$$

Where

$$C_p = \frac{\beta_1 D_p (1 + K_n)}{(1 - K_p K_n)(1 - \bar{\beta}_1 r_2)(A_1 + \bar{\beta}_1(A_2 + A_3))}$$

$$C_n = \frac{\beta_2 D_n (1 + K_p)}{(1 - K_p K_n)(1 - \bar{\beta}_2 s_2)(A_2 + \bar{\beta}_2(A_1 + A_3))}$$

$$K_p = \frac{\beta_1 \bar{\beta}_2 D_p}{(1 - \bar{\beta}_1 r_2)(A_1 + \bar{\beta}_1(A_2 + A_3))(r_1 - \bar{\beta}_2)}$$

$$K_n = \frac{\beta_2 \bar{\beta}_1 D_n}{(1 - \bar{\beta}_2 s_2)(A_2 + \bar{\beta}_2(A_1 + A_3))(s_1 - \bar{\beta}_1)}$$

$$\begin{aligned}
D_p &= (A_3\bar{\beta}_1\beta_2\bar{q}_2 + A_1(1 - \bar{\beta}_1\bar{\beta}_2)) \\
D_n &= (A_3\bar{\beta}_2\beta_1\bar{q}_1 + A_2(1 - \bar{\beta}_1\bar{\beta}_2))
\end{aligned}$$

$$(r_1, r_2) = \frac{a_p \pm \sqrt{b_p^2 - 4a_p c_p}}{2a_p}$$

$$\begin{aligned}
a_p &= A_1 + \overline{\beta}_1(A_2 + A_3) \\
b_p &= -\left((A_1 + A_2 + A_3)(1 + \overline{\beta}_1\overline{\beta}_2) + A_1\beta_1\overline{\beta}_2 \right. \\
&\quad \left. + \beta_2\overline{q}_2(A_2\overline{\beta}_1 - A_3\beta_1)\right) \\
c_p &= \overline{\beta}_2(A_1 + A_2 + A_3) + A_2\beta_2\overline{q}_2
\end{aligned}$$

$$(s_1, s_2) = \frac{a_n \pm \sqrt{b_n^2 - 4a_n c_n}}{2a_n}$$

$$\begin{aligned}
a_n &= A_2 + \overline{\beta}_2(A_1 + A_3) \\
b_n &= -\left((A_1 + A_2 + A_3)(1 + \overline{\beta}_1\overline{\beta}_2) + A_2\beta_2\overline{\beta}_1 \right. \\
&\quad \left. + \beta_1\overline{q}_1(A_1\overline{\beta}_2 - A_3\beta_2)\right) \\
c_n &= \overline{\beta}_1(A_1 + A_2 + A_3) + A_1\beta_1\overline{q}_1.
\end{aligned}$$

4 Performance Measures

With the complete solution to the Markov chain in hand, we calculate several interesting quantities. The first is $\overline{K}_i$ which is defined as the average distance that processor i is ahead of the other. This measure is useful in getting a fix on the number of states which will need to be saved on average.

$$\begin{aligned}
\overline{K}_1 &= \sum_{k=0}^{\infty} kp_k + 0 \cdot \sum_{k=1}^{\infty} n_k = \frac{C_p p_0 r_1}{(r_1 - 1)^2} \\
\overline{K}_2 &= \sum_{k=1}^{\infty} kn_k + 0 \cdot \sum_{k=0}^{\infty} p_k = \frac{C_n p_0 s_1}{(s_1 - 1)^2}
\end{aligned}$$

Since the average size of a state jump at processor i is $1/\beta_i$ then average number state buffers needed at processor i is $\overline{K}_i\beta_i$.

Another useful measure, Θ_b^1, is the probability that processor one is ahead of processor two by more than b units. This measure is exactly the probability that a fixed size state buffer of size b at processor one overflows if $\beta_1 = \beta_2 = 1$ (if only single steps forward are allowed).

$$\begin{aligned}
\Theta_b^1 &= \sum_{k=b+1}^{\infty} p_k \\
&= 1 - p_0 - \sum_{k=1}^{b} p_k - \sum_{k=1}^{\infty} n_k \\
&= 1 - p_0 - \frac{p_0 C_p \left(r_1 - \left(\frac{1}{r_1}\right)^b\right)}{r_1 - 1} - \frac{p_0 C_n}{s_1 - 1}
\end{aligned}$$

A similar (symmetric) value, Θ_b^2, can be found for processor two. The quantity of most interest though is speedup, and we calculate its value in the next section.

4.1 Speedup

Using the formulas for p_k and n_k we can calculate the speedup S when using two processors versus using only one. S is simply the rate of virtual time progress per real time step when using two processors (R_2) divided by the rate of progress when using only one processor (R_1). The rate of forward progress for one processor is defined simply as the average rate of progress of the two processes

$$R_1 = \frac{\frac{\alpha_1}{\beta_1} + \frac{\alpha_2}{\beta_2}}{2}.$$

The rate of forward progress for two processors is the expected "unfettered" progress (without rollbacks) per time step minus the expected rollback distance per time step for the two processors.

$$\begin{aligned}
R_2 = \; & \frac{A_1}{\beta_1} + \frac{A_2}{\beta_2} + A_3\left(\frac{1}{\beta_1} + \frac{1}{\beta_2}\right) \\
& - A_3 q_2 p_0 \sum_{i=2}^{\infty} g_i \sum_{j=1}^{i-1} f_j (i - j) \\
& - A_3 q_1 p_0 \sum_{i=2}^{\infty} f_i \sum_{j=1}^{i-1} g_j (i - j) \\
& - A_2 q_2 \sum_{k=1}^{\infty} p_k \sum_{i=1}^{k-1} i f_{k-i} \\
& - A_1 q_1 \sum_{k=1}^{\infty} n_k \sum_{i=1}^{k-1} i g_{k-i} \\
& - A_3 q_2 \sum_{k=1}^{\infty} p_k \sum_{i=1}^{\infty} g_i \sum_{j=1}^{k+i-1} j f_{k+i-j} \\
& - A_3 q_1 \sum_{k=1}^{\infty} n_k \sum_{i=1}^{\infty} f_i \sum_{j=1}^{k+i-1} j g_{k+i-j} \\
& - A_3 q_1 \sum_{k=1}^{\infty} p_k \sum_{i=1}^{\infty} g_i \sum_{j=1}^{\infty} j f_{k+i+j} \\
& - A_3 q_2 \sum_{k=1}^{\infty} n_k \sum_{i=1}^{\infty} f_i \sum_{j=1}^{\infty} j g_{k+i+j}
\end{aligned}$$

As with the p_k calculation we omit the derivation of the following result. Combining all the terms together we find the formula for speedup.

$$
\begin{aligned}
S = \left(\frac{2}{\frac{\alpha_1}{\beta_1}+\frac{\alpha_2}{\beta_2}}\right) \Bigg[& \frac{A_1}{\beta_1}+\frac{A_2}{\beta_2}+A_3\left(\frac{1}{\beta_1}+\frac{1}{\beta_2}\right) \\
& -\frac{A_3\overline{\beta}_1\beta_2 q_2 p_0}{\beta_1(1-\overline{\beta}_1\overline{\beta}_2)}-\frac{A_3\overline{\beta}_2\beta_1 q_1 p_0}{\beta_2(1-\overline{\beta}_1\overline{\beta}_2)} \\
& -\frac{A_2 q_2\beta_1 C_p p_0 r_1}{(r_1-\overline{\beta}_1)(r_1-1)^2}-\frac{A_1 q_1\beta_2 C_n p_0 s_1}{(s_1-\overline{\beta}_2)(s_1-1)^2} \\
& -\frac{A_3 q_2\beta_2 C_p p_0}{(1-\overline{\beta}_1\overline{\beta}_2)(r_1-1)^2}\left(\frac{(r_1-\overline{\beta}_1)}{\beta_1}+\frac{\beta_1\overline{\beta}_2 r_1}{(r_1-\overline{\beta}_2)}\right) \\
& -\frac{A_3 q_1\beta_1 C_n p_0}{(1-\overline{\beta}_1\overline{\beta}_2)(s_1-1)^2}\left(\frac{(s_1-\overline{\beta}_2)}{\beta_2}+\frac{\beta_2\overline{\beta}_1 s_1}{(s_1-\overline{\beta}_1)}\right) \\
& -\frac{A_3 p_0}{1-\overline{\beta}_1\overline{\beta}_2}\left(\frac{q_1\beta_1\overline{\beta}_2^2 C_p}{\beta_2(r_1-\overline{\beta}_2)}+\frac{q_2\beta_2\overline{\beta}_1^2 C_n}{\beta_1(s_1-\overline{\beta}_1)}\right)\Bigg]
\end{aligned}
$$

5 Limiting Behavior

The reason we chose to use a discrete time, discrete state (DD) model was to allow ourselves to take limits on the α and β parameters thus creating models which are continuous in time and state (whereby geometric distributions become exponential distributions). We omit the actual formulae here due to space considerations; see [4] for the full details.

We can transform our model into a continuous time, discrete state (CD) model by taking the limit as α_1 and $\alpha_2 \rightarrow 0$ while keeping the ratio $\frac{\alpha_1}{\alpha_2}$ constant and defining $\frac{A_1}{A_1+A_2} = a$. We can take the limit either on the p_k equations or on our formula for speedup.

Alternatively, we create a discrete time, continuous state (DC) model by taking the limit as β_1 and $\beta_2 \rightarrow 0$ while keeping $\frac{\beta_1}{\beta_2} = b$. We find the value for speedup by taking limits on the speedup formula calculated for the discrete time, discrete state model.

Finally, we can solve a continuous time, continuous state (CC) model by taking limits on α_i and β_i simultaneously. This can be done either by going first to the CD (α_i) or DC (β_i) model from DD, and then finishing by taking limits on the other variable.

5.1 Previous Work on 2-Processor Models

There has been some similar work on two-processor Time Warp models. Lavenberg, Muntz and Samadi [5] used a continuous time, continuous state model to solve for the speedup (S) of two processors over one processor. Their work resulted in an approximation for S which was valid only for $0 \leq q_i \leq 0.05$. Remember that q_i is the interaction parameter; the probability that processor i will send a message to the other processor. Their result is only valid for very weakly interacting processes. Our result for this CC case has no restrictions on any of the parameters and therefore subsumes their work. In fact, we can compare our results directly for a simplified case where $a = 1/2$ (same processing rate for both processors), $b = 1$ (same average jump in virtual time for both) and $q_1 = q_2 = q$ (same probability of sending a message), which is the completely symmetric case. Lavenberg et al. derive the following approximation for speedup:

$$S_L \approx 2 - \sqrt{2q}.$$

Our equation for speedup in this restricted case is:

$$S = \frac{4\left(\sqrt{q}\,(5+q)+(1+q)\sqrt{8+q}\right)}{\sqrt{q}\,(2+q)(7+q)+\sqrt{8+q}\,(2+5q+q^2)}.$$

If we expand this formula using a power series about the point $q = 0$ and list only the first few terms, we see the essential difference between our result and Lavenberg et al.

$$S \approx 2 - \sqrt{2q} + \frac{q}{2} + O(q^{\frac{3}{2}})$$

This clearly shows that our result matches Lavenberg et al. in the first two terms. We see that their result is only accurate for very small values of q as they mention in their paper. Figure 3 shows the Lavenberg et al. result and our result compared to simulation with 99% confidence intervals.

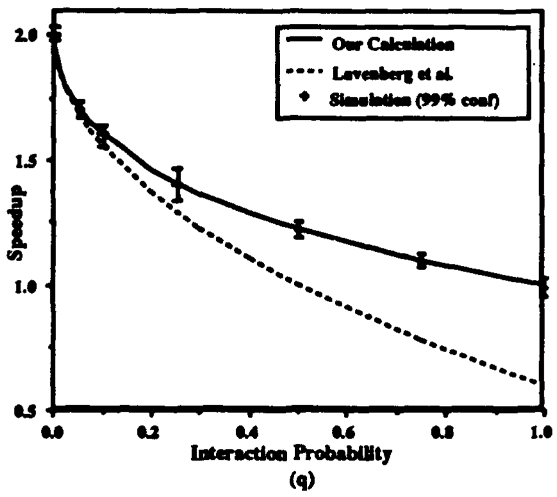


Figure 3: Comparison of speedup results for a simplified case.

Mitra and Mitrani [9] also solve a two-processor model but use a discrete time, continuous state approach to solve for the distribution of the separation between the two processors and the rate of progress of the two processors. In the definition of their model, a processor sends a message (with probability q_i) before advancing. Our model has a processor send a message after advancing. This difference between the two models disappears in the calculation of the average rate of progress. Their solution allows a general continuous distribution for the state jumps (virtual time), but requires (deterministic) single steps for the discrete time. In our model this is equivalent to setting $\alpha_1 = \alpha_2 = 1$. Since our analysis only supports an exponential distribution for state changes, but their analysis doesn't have a distribution on time, neither model subsumes the other.

Finally, the DD and CD models do not seem to have appeared in the literature, although a simplified version of the CD model where $\beta_1 = \beta_2 = 1$ (a preliminary version of this work which only allowed single-step state jumps) has been published by Kleinrock[3] . Figure 4 shows how all of this work fits together. The work discussed in this paper covers the shaded region.

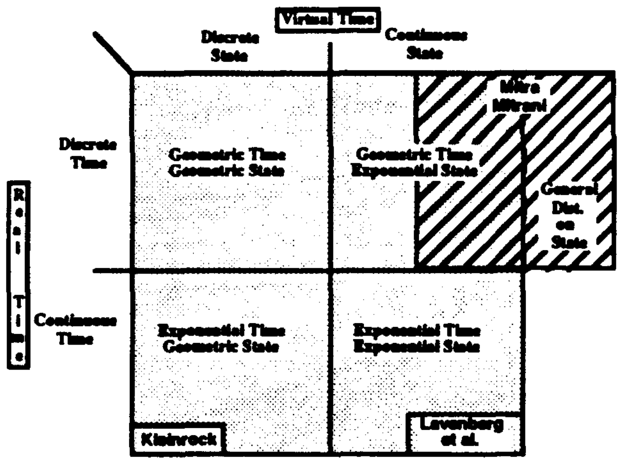


Figure 4: Previous work.

6 Results for a Restricted Model

In order to better understand our results, we examine a restricted version of the CD model (i.e. the model analyzed in [3]). In this less general model we eliminate two variables by forcing the processors to advance exactly one step each time they advance ($\beta_1 = \beta_2 = 1$). Again, we define q_i as the interaction parameter; the probability that processor i sends a message to the other processor. We also define a as the ratio $\frac{\lambda_1}{\lambda_1+\lambda_2}$ where λ_i is the rate for the continuous time distribution for processor i (rate at which messages are processed). Figure 5 shows the speedup for the Symmetric case where $q_1 = q_2 = q$. Figure 6 shows the speedup for the Balanced case

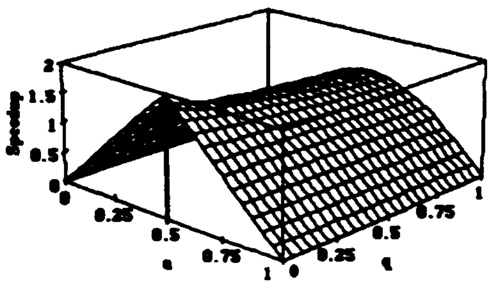


Figure 5: Speedup for the Symmetric case $q_1 = q_2 = q$

where $\lambda_1 = \lambda_2$. Figure 7 shows the speedup for

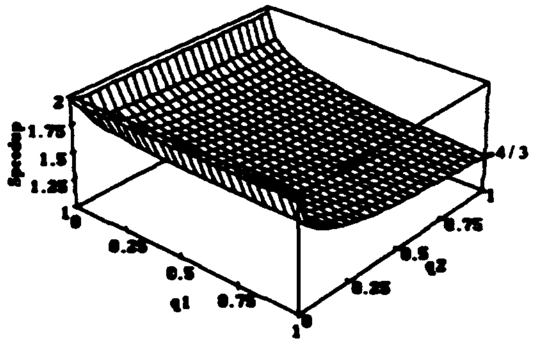


Figure 6: Speedup for the Balanced case $\lambda_1 = \lambda_2 = \lambda$.

the extremely simplified Symmetric, Balanced case where $q_1 = q_2 = q$ and $\lambda_1 = \lambda_2 = \lambda$. For this special case the formula for speedup is

$$S = \frac{4}{2+\sqrt{q}}.$$

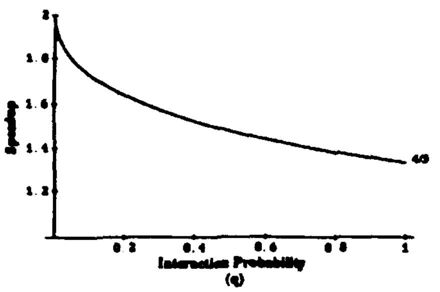


Figure 7: Speedup for the Symmetric, Balanced case $q_1 = q_2 = q$ and $\lambda_1 = \lambda_2 = \lambda$.

6.1 Optimality Proofs

Using the simple model described above, we can prove several results about optimality with respect to the parameters of the system. We first show that the speedup is monotonically decreasing in both q_1 and q_2, the interaction parameters. We do this by showing that $\frac{\partial S}{\partial q_1}$ is negative. First, a definition:

$$h(q_1) = (1 - 4a\bar{a}\bar{q}_1).$$

If we differentiate S with respect to q_1 we arrive at the following formula

$$\frac{\partial S}{\partial q_1} = \Phi \cdot \left(-(-1+2a)^2 - 2a\bar{a}q_1 + (1-2a)\sqrt{h(q_1)} \right),$$

where Φ is a non-negative function of q_1,q_2, and a.

In order to show that $\frac{\partial S}{\partial q_1}$ is negative, we must show that

$$-(-1+2a)^2 - 2a\bar{a}q_1 + (1-2a)\sqrt{h(q_1)} \leq 0. \quad (4)$$

When $a \geq \frac{1}{2}$ Equation 4 is trivially solved. Our concern is in the range $0 \leq a < \frac{1}{2}$, in which case our condition becomes:

$$-(-1+2a)^2 - 2a\bar{a}q_1 + (1-2a)\sqrt{h(q_1)} \leq 0$$

$$\begin{aligned} -(-1+2a)^2 - 2a\bar{a}q_1 &\leq -(1-2a)\sqrt{h(q_1)} < 0 \\ \left(-(-1+2a)^2 - 2a\bar{a}q_1\right)^2 &\geq \left(-(1-2a)\sqrt{h(q_1)}\right)^2. \end{aligned}$$

Expanding and simplifying, our condition reduces to

$$4a^2\bar{a}^2{q_1}^2 \geq 0.$$

Since a and $\bar{a}$ and q_1 are all non-negative, the inequality holds. A similar (symmetric) proof for q_2 is omitted here.

Optimization with respect to a is a little more difficult. When we differentiate S with respect to a we get such a complicated formula that it is prohibitive to solve for the optimum value of a. Fortunately, by plotting S versus a, q_1 and q_2 we see that S is unimodal and that the optimum value of a is 1/2 ($\lambda_1 = \lambda_2$). When we plug this value ($a = 1/2$) into $\frac{\partial S}{\partial a}$ we see that the result is 0.

$$\left.\frac{\partial S}{\partial a}\right|_{a=\frac{1}{2}} = \frac{2(-((1-\bar{q}_1)q_2) + q_1(1-\bar{q}_2))}{(1-\bar{q}_1)(1-\bar{q}_2)} = 0$$

To show that this is a maximum we must show that the second derivative is negative at $a = 1/2$. This is not difficult, though we omit the equations here for brevity. For the more general case, where the processors are not restricted to single step advances, the result that $a = 1/2$ ($\lambda_1 = \lambda_2$) for optimal performance generalizes to

$$\frac{\lambda_1}{\beta_1} = \frac{\lambda_2}{\beta_2} \text{ or } b = \frac{a}{1-a} \quad (5)$$

meaning that the average "unfettered" rate of progress in virtual time for each processor should be the same. For a fixed value of a the best performance can be found when Equation 5 is true, and overall best performance is found at $a = 1/2$ with Equation 5 holding true. Speedup is not constant for a/b constant.

6.2 Adding a Cost for State Saving

One simple way of examining how state saving overhead affects the performance of the system is to modify the value of R_1, the rate of progress on a single processor. For example, if we examine the CD model with the single step restriction (as above) we arrive at the following value for R_1.

$$R_1 = \frac{c(\lambda_1 + \lambda_2)}{2}$$

The parameter c ($c \geq 1$) indicates how much faster events are executed without state saving. If $c = 2$ state saving doubles the amount of time it takes to process an event. For the CD model we find that the new formula for speedup is simply $1/c$ times the old value. Let us examine a very simple case in detail. If we look at the Symmetric, Balanced case, the updated formula for speedup is

$$S = \frac{4}{c(2 + \sqrt{\bar{q}})}.$$

It is easy to see that as $c \to \infty$ speedup will go to zero. We are most concerned with the boundary where $S = 1$ which is the transition from areas where TW on two processors helps to where it hurts. Setting $S = 1$ and solving for q we find the necessary condition for two processors running TW to be faster than only one.

$$q \leq \frac{4(2-c)^2}{c^2}$$

For $c \geq 2$ TW with two processors is always worse than running on one processor without TW. Conversely, for $c \leq 4/3$ TW wins out. The interesting range is where $4/3 \leq c \leq 2$. In this range, certain values of q will yield speedup, while others won't. Figure 8 shows the regions in the $c - q$ plane where TW on two processors is effective and where it is not. Thus, if we know the values of both c and q for our Symmetric, Balanced system we can immediately tell whether we can speed up the application by running it under Time Warp on two processors.

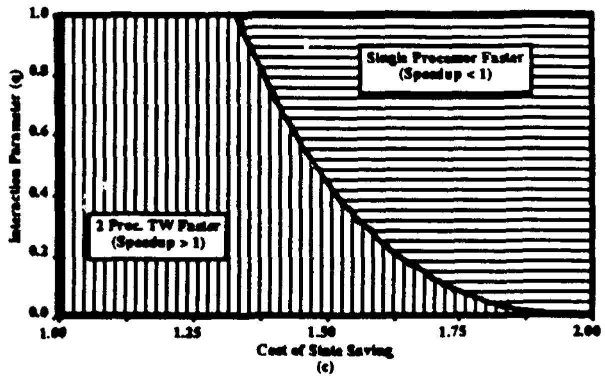


Figure 8: Cost for state saving and it's effect on performance.

7 Future Work and Conclusions

There are several avenues to follow for future work. One is to add message queueing to our model. Currently any message that arrives in the future is ignored. This is unrealistic since the messages in TW actually carry some work. Another addition would be to charge some cost for rollback. In the present model, rollbacks are free and therefore there is no penalty for speculative computing. We have exact solutions for models that address these concerns and they will appear in a future work [1]. We also would like to extend the model to accommodate more than two processors. Certainly, an exact Markov chain analysis will quickly become intractable. We are currently investigating extensions of our present model to many processors without using a complicated Markov chain approach.

In this paper we have presented a model for two processor Time Warp execution and provided the results of its exact solution. The model is general enough to subsume the work of Lavenberg, Muntz and Samadi [5] and to partially subsume the work of Mitra and Mitrani [9]. Further, we examined a simplified version of our model and showed for optimal performance that the processors should send as few messages as possible and that their independent rates of progress in virtual time should be the same. Finally, we addressed the cost of state saving and it's effect on performance. Large state saving costs or frequent message interactions indicate that TW is ineffective in gaining speedup. The detailed analysis of our model and logical generalizations to it will appear in future works [4][1].

References

[1] Robert E. Felderman and Leonard Kleinrock. Extensions to the time warp analysis. Submitted to ACM Transactions on Modelling and Computer Simulation, October 1990.

[2] David R. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3), July 1985.

[3] Leonard Kleinrock. On distributed systems performance. In *Proceedings of the 7th ITC Specialist Seminar, Adelaide, Australia.* ITC, September 1989.

[4] Leonard Kleinrock and Robert E. Felderman. Two processor time warp analysis: A unifying approach. Submitted to International Journal of Computer Simulation, August 1990.

[5] Steven Lavenberg, Richard Muntz, and Behrokh Samadi. Performance analysis of a rollback method for distributed simulation. In *Performance '83*. North-Holland, 1983.

[6] Yi-Bing Lin and Edward D. Lazowska. Optimality considerations for "time warp" parallel simulation. Technical Report 89-07-05, Department of Computer Science and Engineering, University of Washington, July 1989.

[7] Vijay Krishna Madisetti. Self synchronizing concurrent computing systems. Technical Report UCB/ERL M89/122, Electronics Research Laboratory, College of Engineering University of California Berkeley, CA 94720, October 1989.

[8] Jayadev Misra. Distributed discrete-event simulation. *Computing Surveys*, 18(1), March 1986.

[9] Debasis Mitra and I. Mitrani. Analysis and optimum performance of two message-passing parallel processors synchronized by rollback. In *Performance '84*. North-Holland, 1984.

Performance Analysis of Finite-Buffered Multistage Interconnection Networks with a General Traffic Pattern*

T. Lin
University of California, Los Angeles
Computer Science Department

L. Kleinrock
University of California, Los Angeles
Computer Science Department

Abstract

We present an analytical model for evaluating the performance of finite-buffered packet switching multistage interconnection networks using blocking switches under any general traffic pattern. Most of the previous research work has assumed unbuffered, single buffer or infinite buffer cases, and all of them assumed that every processing element had the same traffic pattern (either a uniform traffic pattern or a specific hot spot pattern). However, their models cannot be applied very generally. There is a need for an analytical model to evaluate the performance under more general conditions.

We first present a description of a decomposition & iteration model which we propose for a specific hot spot pattern. This model is then extended to handle more general traffic patterns using a transformation method. For an even more general traffic condition where each processing element can have its own traffic pattern, we propose a superposition method to be used with the iteration model and the transformation method. We can extend the model to account for processing elements having different input rates by adding weighting factors in the analytical model.

An approximation method is also proposed to refine the analytical model to account for the memory characteristic of a blocking switch which causes persistent blocking of packets contending for the same output ports. The analytical model is used to evaluate the uniform traffic pattern and a very general traffic pattern "EFOS". Comparison with simulation indicates that the analytical model is very accurate.

*Supported by a grant from DARPA MDA 903-87-C-0663

1 Introduction

Packet-switching Multistage Interconnection Networks (MIN) have been proposed for applications in multiprocessor systems for interconnecting a large number of processing elements (PE) and memory modules (MM), and also for fast packet switching various packets to their destination ports [21]. The performance analysis of MIN's thus becomes an important issue.

A considerable amount of performance analysis has been reported on clocked, packet-switched multistage interconnection networks. Example networks are the Banyan [5] and Omega [13] networks. Most of the previous research was limited to only unbuffered or infinite buffered cases with a uniform traffic pattern or a particular hot spot traffic pattern. Dias and Jump [4] and Jenq [7] analyzed the single buffer case under a uniform traffic pattern. Kim and Garcia [8] analyzed a single buffered network with nonuniform traffic patterns. Kruskal and Snir [9], [10], [11] analyzed the performance of banyan networks with infinite buffer sizes under uniform and a particular hot spot traffic pattern. Szymanski and Shaikh [19], Yoon et al. [23] all presented an analytical model to analyze a finite buffered MIN under a uniform traffic pattern only. Willick and Eager [22] also analyzed the infinite buffered case; their model is able to analyze any given general traffic pattern except that they need a special purpose analysis for each non-uniform pattern. Theimer et al. [20] proposed a technique for modelling the persistent blocking behavior of a single buffered network with a uniform traffic pattern.

In this paper, we present an analytical model for clocked, packet-switched multistage interconnection networks built with 2x2 blocking switches. The model is based on a decomposition and iteration analysis of the Markov chain representing the output queue in each switch. The goal is to provide a simple analytical model which can be applied to evaluate the performance of multistage interconnection networks with

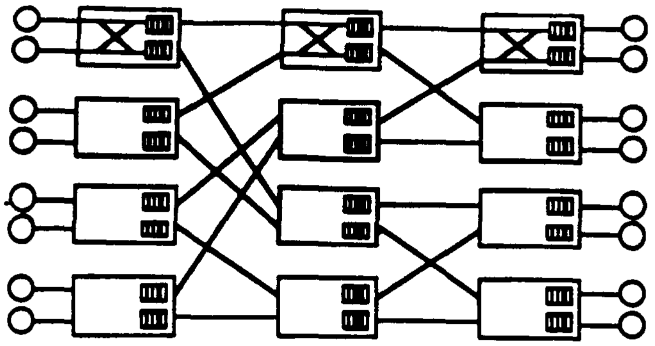


Figure 1: The 3 stage Banyan network with buffers at output ports of each switch

arbitrary switch sizes, arbitrary buffer sizes, arbitrary input rates for each processing element and any general traffic pattern.

This paper is organized as follows. Section 2 describes the model assumptions and the approach. The decomposition and iteration method is introduced with a unique routing probability matrix to represent the steady state traffic flow. A uniform traffic pattern and a special hot spot pattern are discussed using this matrix. A transformation method is introduced in Section 3 to allow any general traffic pattern to be mapped onto the routing matrix to represent the steady state flow. A superposition method is also introduced to allow each processing element to have its own traffic pattern. The resulting memory referencing pattern is even more general; as an example we analyze the Non-Uniform Traffic Spots (NUTS) pattern [12]. A weighting factor is used with the mapping process to handle the case when processing elements are allowed to have different input rates. An approximation method is introduced in section 4 to account for the "memory" characteristics inherent in the blocking scheme. Results are verified through simulation in section 5. Conclusions are given in section 6.

2 Analytical Model for Uniform Traffic and Hot Spot Traffic Patterns

2.1 Architecture Description and Assumptions

In this paper, the interconnection network we consider is a clocked, packet-switched finite-buffered Banyan network made up of 2x2 switches, each of which has buffers of finite size K at their output ports (see Figure 1). There are N processing elements and N memory modules interconnected by the n-stage (i.e. $N = 2^n$) interconnection network. All the operations of input and output take place at the end of each cycle. The interconnection network accepts requests from the input nodes (processing elements), then routes them to the output nodes (memory modules). Responses to these requests are returned from the output nodes through the interconnection network in the reverse direction to the original requesting nodes. The "forward" network and "backward" network are distinct, but are identical in topology. It is sufficient to discuss the delay and throughput performance of the forward network only.

Each packet generated at the processing elements carries an address tag with a number of bits equal to the number of stages of the interconnection network. The address tag is a binary representation of the destination address. It is then fed into the first stage of the network. The first stage switch examines the first (i.e. most significant) bit of the address tag; if it is a 0, the packet is routed to the queue at the upper output port. If the first bit is a 1, the packet is routed to the queue at the lower output port (see Figure 1). The packet then waits in the queue until its turn to be served. The routing process repeats in each stage to choose a proper output port. A blocking switch is assumed in which if a head-of-queue packet cannot go to the next stage due to a full buffer or a "contention failure" for a single available position, it stays at the current queue and waits for the next cycle to try again. The blocking phenomenon has an implied memory characteristic in that a blocked packet will attempt to reach the same output port again. This memory characteristic makes the analytical modelling difficult. We shall discuss an approximation method to be incorporated into the analytical model to model this memory characteristic in later sections.

When two packets from different queues in the same stage contend for the same output queue in the next stage, a contention occurs. If there are more than two spaces available at the output queue, the switch is assumed to be fast enough to accept both packets in one cycle. If there is only one space available, a packet is randomly chosen to fill up this space; the other packet is then "blocked" and stays at the original queue. However, if no space is available in the next stage (i.e. the queue at the output port is full), then both packets are blocked.

Packets are assumed to be of the same length (i.e. fixed size packets). A packet is generated by each processing element independently with probability q in each cycle. All processing elements are assumed to have this identical bernoulli input process. This assumption is later relaxed by using weighting factors to allow each processing element to have its own input rate q_j, $1 \leq j \leq N$. We assume that there is no buffer space at the processing elements. After being generated, a packet is discarded if it cannot be delivered to the first stage of the interconnection network either due to a full buffer or a contention failure. Discarded packets are not re-submitted. A packet, once accepted by the network, is never discarded inside the network. The input process is independent of the discarding process. (An extension of the current model to allow blocked packets to be stored in a finite-sized queue or an infinite queue is underway.) An important performance measure is the total time a packet spends in the network. Time delay is meaningful only for those packets accepted into the network. The probability of acceptance, another performance measure, is the probability that a packet is accepted into the network after it is generated. The normalized throughput is simply the probability of acceptance multiplied by the input rate. Current work also includes an extension to the case of multiple packet generation.

Each processing element has a memory module referencing pattern. A referencing pattern is the set of probabilities with which a packet accesses the various memory modules. All previous work assumes that processing elements have the same referencing pattern. We shall allow PE's to have their own traffic pattern in Section 3. The memory module is assumed to be fast enough to accept 1 packet per cycle from switches at the last stage. This fast memory module assumption implies that there is no blocking at the last stage since a dedicated link connects 1 memory module to the output queue (see Figure 1). A slower memory module (e.g. 2 cycles to accept a packet) will have a severe effect on the performance of the network. Extension to slower memory models is underway.

2.2 Routing Model

In the real world, the packets are routed according to their destination address. However, in order to analyze the network analytically, an abstract flow model that can be used in an analytical model must be established that at least faithfully reflects the steady state flow situation in the network. We propose a routing matrix $r_{i,j}$, $1 \leq i \leq n, 1 \leq j \leq N$ where $r_{i,j}$ is the routing probability of the jth input port in stage i. A packet entering a switch will be routed either to the upper output queue with probability $r_{i,j}$ or to the lower output queue with probability $1 - r_{i,j}$. To simulate a uniform traffic pattern, we simply let all $r_{i,j}$ be 0.5. With equal probability of choosing output queues, no memory module is preferred. A special hot spot pattern can be created by letting all $r_{i,j}$ be an identical value greater than 0.5. For instance, by letting all $r_{i,j}$ be 0.8 in a 10 stage network, 10.7% ($= .8^{10}$) of the total traffic will go to memory module 0 in a 1024-node network with 2.7% of the traffic going to the second highest referenced memory modules (all memory modules with a single 1-digit in their address tag) and other fractions of traffic to the other memory modules. The advantage of this routing model is that by changing the value of $r_{i,j}$ with proper mappings from real traffic patterns, we can evaluate any general traffic pattern. We leave the general $r_{i,j}$ to be discussed in Section 3. Throughout this section, all $r_{i,j}$ are assumed to have the same value, $r_{i,j} = r$.

2.3 Analysis

The proposed approximate analytical approach employs a decomposition and iteration strategy. The real interconnection network is in fact a network of finite-buffered queues with blocking. The dependency among queues, caused by the blocking from stage to stage, makes the exact analysis intractable. We shall use a similar approximation technique as that applied in tandem queues with blocking [2], [3] and [17] where approximate analyses are used. The approximation method is to decompose each queue in the tandem configuration with assumed input rates and blocking conditions. The exact Markov chain is then solved to find the corresponding input rates and blocking conditions. Each decomposed queue is analyzed, then the whole process is repeated until it converges, if it is to have steady state. The concept of using this decomposition and iteration approximation method in analyzing the finite-buffered Banyan network is very similar to that of tandem queues except that instead of a single input source and a single output queue for each queue in the tandem configuration, the interconnection network has 2 input sources and 2 output queues for each queue in the network (except the last stage queue where only 1 output sink is presented, namely, the memory module). Therefore, when we solve for the equivalent input rates and blocking conditions for a decomposed queue, we consider the combined input from 2 input sources and the combined probability of blocking from the 2 output queues. The approach is as follows :

Let $Q_{i,j}$ represent the jth queue in stage i and $P_{i,j}(k)$ be the steady state probability that there are k packets in the queue $Q_{i,j}$. Let $Q_{i-1,j1}$ and $Q_{i-1,j2}$ be the two input sources from stage i-1 that feed $Q_{i,j}$. Let $X[i]$ be the probability that there are i packets destined to $Q_{i,j}$ from its two input sources. In the following, we solve for the equivalent input rates for a queue $Q_{i,j}$ which is located at output port 0 :

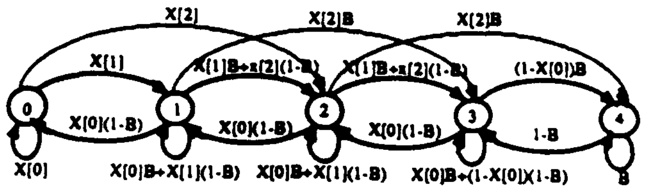


Figure 2: Markov chain of a queue $Q_{i,j}$ extracted from the network where the state variable represents the number of packets in that queue

$$X[1] = 2r(1-r)(1-P_{i-1,j1}(0))(1-P_{i-1,j2}(0))+$$
$$r[P_{i-1,j1}(0)(1-P_{i-1,j2}(0))+P_{i-1,j2}(0)(1-P_{i-1,j1}(0))]$$
$$X[2] = [r(1-P_{i-1,j1}(0))]\cdot[r(1-P_{i-1,j2}(0))]$$
$$X[0] = 1 - X[1] - X[2] \qquad (1)$$

The first term in the $X[1]$ equation corresponds to the case that both queues in stage i-1 are not empty, and one chooses output port 0 with probability r and the other chooses another output port with probability $1 - r$. The second term corresponds to the case that one queue in the previous stage is empty, and the other is not. The non-empty one chooses the output port 0 with probability r. The summation of probabilities in both cases represents the probability that only one input packet feeds the queue. The $X[2]$ equation represents the case when both queues in the previous stage are not empty and they both choose output port 0 with probability r.

Regarding the equivalent blocking condition, let $B_{i,j}$ be the probability that a packet in the jth queue in stage i is blocked at the end of the cycle. Let $C_{i,j}$ be the probability that the jth queue in stage i is blocking a packet in stage $i-1$. Let $Q_{i+1,j1}$ and $Q_{i+1,j2}$ be the two output queues of $Q_{i,j}$ and let $Q_{i,l}$ be the queue that feeds both $Q_{i+1,j1}$ and $Q_{i+1,j2}$. Then the equivalent blocking condition for queue $Q_{i,j}$ is as follows :

$$B_{i,j} = r\cdot C_{i+1,j1} + (1-r)\cdot C_{i+1,j2}$$
$$C_{i+1,j1} = P_{i+1,j1}(K) + \frac{r}{2}\cdot(1-P_{i,l}(0))\cdot P_{i+1,j1}(K-1) \qquad (2)$$

The first term in the $B_{i,j}$ equation represents the case when the packet at the head of queue $Q_{i,j}$ chooses $Q_{i+1,j1}$ with probability r and is blocked by $Q_{i+1,j1}$. The second term represents the other case when the packet chooses $B_{i+1,j2}$ and is blocked. There are two situations in which a queue blocks a packet in the preceding stage : firstly, when the queue is full, and secondly, when the queue has only one more space and a contention from $Q_{i,l}$ wins the arbitration.

Given a set of initial values for the variables of the network, we "extract" queue $Q_{1,1}$ from the network (with the equivalent input rates and blocking conditions as exist in the network) as an independent queue. The Markov chain for this queue is then solved to get new values for the state probabilities. A sample Markov chain for the queue $Q_{i,j}$ with buffer size 4 is shown in Figure 2 where B represents the blocking probability $B_{i,j}$. We repeat this process for other queues in the first stage, in the order $Q_{1,2}, Q_{1,3}, ...Q_{1,N}$. Using these new state probabilities as the new input rates, we repeat the same process for all queues in the second stage in the order $Q_{2,1}, Q_{2,2}, ...Q_{2,N}$. This process is repeated for all stages. Now we have a new set of values for the network variables which can be used to compute the new input rates and blocking probabilities. This new set of values is used in the next iteration to compute another set of new values, etc.. The iteration process is repeated until the difference between two consecutive iterations is below 10^{-6}.

The performance measures that are of interest are the probability of acceptance, the normalized throughput and the average time delay. There are two ways to calculate the probability of acceptance. If we sum the output rate over all output ports and divide it by the total input rate, we get the probability of acceptance :

$$PA_{out} = \frac{\sum_{i=1}^{N}[1-P_{n,i}(0)]}{N \times q} \qquad (3)$$

The total output rate over the input rate is the probability of acceptance at the output port. From the input port, we solve for the probability that a packet generated at the PE's is discarded due to a full buffer or a contention failure at the first stage. This discarding probability is $B_{0,j}$, which can be solved for using equation (2). Hence,

$$PA_{in} = 1 - B_{0,j} \qquad (4)$$

Both values, although solved in different ways, should be equal when the MIN reaches steady state. (This can be used to test for the correctness of the model.) The normalized throughput is found by multiplying the probability of acceptance by the input rate. We apply Little's result to calculate the average time delay of a packet. When the network reaches steady state, we take the sum of the mean queue size for the whole network using the steady state probabilities of queue size of each queue. Given the throughput and the average number of customers in the system, the average time delay can be solved for by applying Little's result.

2.4 Results

The analytical results of a finite-buffered multistage interconnection network under a uniform traffic pattern and a hot spot pattern were shown in [14]. For various traffic loads and various network sizes (from a single stage to a 9-stage Banyan), the improvement of the probability of acceptance by adding buffers (from unbuffered to buffer size 8) were shown for both the uniform traffic pattern and various hot spot traffic patterns. Another experiment was shown for a 9-stage, 8-buffered Banyan where the average busy buffer size was calculated in each stage. By varying the offered load and the degree of hot spots ($0.5 \leq r \leq 1.0$), we can see how the average busy buffer size grows according to different situations. The average time delay for a 9-stage, 8-buffered Banyan network was shown with various offered loads and various degrees of hot spots. A tree build-up time, defined as the time for the saturated tree [18] to build up, was discussed and a method for calculating the upper bound was shown using the analytical model. The analytical model can also be extended [14] to analyze the performance of the combining network suggested in the NYU Ultra computer project [6] and [18]. The effectiveness of a combining switch was analyzed against various offered loads and queue sizes for a 9-stage combining network. It was shown that the combining switch works only when there is a "hot" memory cell inside the hot spot memory module.

3 Analytical Model for General Traffic Conditions

In this section, we consider very general traffic conditions where not only a general traffic pattern is allowed, but also each processing element can have its own traffic pattern and its own input rate. The basic decomposition and iteration model remains as the main modeling approach. The additional analysis which is needed for general traffic conditions is reduced to finding the proper representation of the steady state traffic flows in terms of the routing probabilities $r_{i,j}$.

3.1 Model Assumptions

We still consider a clocked, finite-buffered multistage interconnection network as discussed in the previous section. All assumptions made in section 2 remain the same except :

- Any general memory referencing pattern is allowed.

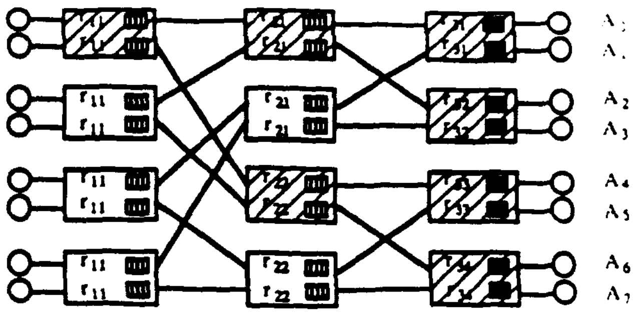


Figure 3: A General memory referencing pattern shown in terms of accessing probabilities A_j

- Each processing element can have its own referencing pattern.
- Each processing element can have its own input rate.

These three different assumptions represent different levels of general traffic patterns. We shall discuss the modeling approaches for these three different assumptions in the next subsections.

3.2 Identical General Traffic Patterns for the Processing Elements

A traffic pattern that can be in any form implies that the $r_{i,j}$'s in the routing matrix no longer have the same value r as discussed in the previous section. The approach to model this general traffic pattern is to find a mapping scheme that transforms the given referencing pattern into a set of $r_{i,j}$'s which reflects the steady state traffic flow in the network.

Let us take a 3 stage Banyan network as an example, as shown in Figure 3. Since we assume that all processing elements have the identical general traffic pattern, we only discuss the transformation method for one processing element. If there exists a steady state referencing pattern, we can represent it in terms of destination accessing probabilities A_j, the probability that a new packet generated by a processing element chooses memory module j as its destination. Consider a packet generated by processing element 0 and observe the path it takes as it travels through the network to access the memory modules. A packet chooses memory module 0 with probability A_0 which equals $r_{11} \cdot r_{21} \cdot r_{31}$. Similarly, a packet chooses memory module 1 with probability $A_1 = r_{11} \cdot r_{21} \cdot (1 - r_{31})$. Using these two equations, we find r_{31} in terms of A_0 and A_1.

$$r_{31} = \frac{A_0}{A_0 + A_1}$$

The other routing Probabilities can be found in a similar way :

$$r_{32} = \frac{A_2}{A_2 + A_3}$$

$$r_{33} = \frac{A_4}{A_4 + A_5}$$

$$r_{34} = \frac{A_6}{A_6 + A_7}$$

$$r_{21} = \frac{A_0 + A_1}{A_0 + A_1 + A_2 + A_3}$$

$$r_{22} = \frac{A_4 + A_5}{A_4 + A_5 + A_6 + A_7}$$

$$r_{11} = \frac{A_0 + A_1 + A_2 + A_3}{A_0 + A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7}$$

Since there is only one traffic pattern for all processing elements, this routing probability set is valid for all other processing elements.

This transformation method is simply a mapping of the memory referencing pattern onto a set of routing probabilities. We incorporate the transformation method into our analytical model approach in section 2. When we calculate the equivalent input rates and blocking probabilities, we replace the value r in equations (1)-(2) with the proper $r_{i,j}$ values. Thus, the decomposition and iteration model we proposed in section 2 can be extended to analyze a multistage interconnection network with any general traffic pattern.

3.3 Different General Traffic Patterns for Processing Elements

For more generality, we allow each processing element to have its own general traffic pattern. In practice, it is quite possible that the traffic requirement for each processing element is different from the others. Let us assume that they all have the same input rate. The modelling approach for this case is to find a proper set of routing probabilities $r_{i,j}$ which reflects the combined traffic flow in steady state. A superposition method is proposed, in addition to the use of the transformation method, to find this proper set of routing probabilities. We first apply the transformation method to all processing elements to transform their general memory referencing patterns into N sets of routing matrices. Since all processing elements have the same input rate, we simply take the mean value of these N sets of routing probabilities to find a routing matrix that reflects the combined traffic flows in steady state. This routing matrix is used in the decompostion and iteration model. Thus, the analytical model can be extended to analyze the case where each processing element has its own traffic pattern.

3.4 Different Input Rate and Different Traffic Patterns for the Processing Elements

In many cases, the processing elements might have different packet input rates. This presents the most general traffic condition. The modelling approach, again, is reduced to finding the proper representation of the routing matrix.

A slower source contributes less to the steady state flows. Hence we must determine a weighting factor for each source to reflect its contribution to the steady state traffic. The ideal weighting factor is the input rate of each source. When we take the mean of the N routing matrices, we then weight their contribution according to their input rates. If we incorporate this weighting factor, the model can be extended to analyze the case where each processing element has its own traffic pattern and its own input rate.

3.5 Results

Since the proposed analytical model employs several approximate methods, it is important to study how these approximations affect the model accuracy. There are two approximations in the modelling approach :

- decomposing a queue from a network of queues with blocking into an independent queue.
- using a general routing matrix to model the steady state flows.

The first approximation is obvious since dependent queues are decomposed into equivalent independent queues and solved individually. Some accuracy is lost because our model neglects the dependency and coupling among the queues. The second approximation allows packets to choose their output ports every cycle independently according to the routing probabilities, instead of the real-world address tag. This renewal routing choice allows a blocked packet to choose a different output port in the next cycle. This renewal assumption renders the analytical model optimistic since it "allows" blocked packets to be routed around a congested queue. In the real world, blocked packets repeatedly access the same destination, and most likely, these blocked packets will be blocked again (especially when the traffic is not uniform).

The EFOS (Even-First-Odd-Second) pattern was proposed in [12] using an Omega network where even addressed processing elements send all their traffic to the first half of memory modules uniformly while the odd addressed ones send their traffic to the second half of memory modules uniformly. The destination traffic

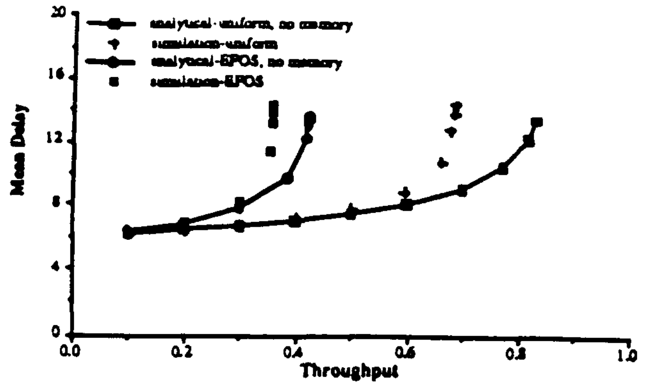


Figure 4: Comparison of results for a 6 stage, 4-buffered Omega network

distribution looks uniform, but there are severe contentions for the common paths inside the network. It is an example of our general traffic pattern where there are two traffic patterns for the processing elements. A 6 stage Omega network with 4 output buffers at each switching element was evaluated under both a uniform traffic pattern and an EFOS traffic pattern. The analytical results we obtained are plotted against simulation results in Figure 4. As predicted , the analytical model is very optimistic due to the independent routing choices it allows. When severe blocking is present due to contention, the blocked packets will choose the same output queues repeatedly in the real world while the renewal choice in the analytical model allows the blocked packets to choose other queues. This inherited "memory" structure in blocking switches severely degrades the performance since it is likely to have persistent contention for a queue once contention occurs. The discrepancy between analytical and simulation data is caused mainly by this memory characteristic of the blocking switch. We propose an improvement in the next section to model this "memory" behavior of a blocking switch.

4 Analytical Model for a Blocking Switch with Persistent Blocking

4.1 Model Approach

Since the basic model is a renewal process, we continue to model the memory behavior as a renewal process. However, the behavior of a blocked packet, after its first blocking, is such that the routing choice no longer uses the renewal probability $r_{i,j}$. Biasing the routing probabilities to account for this does not help since it

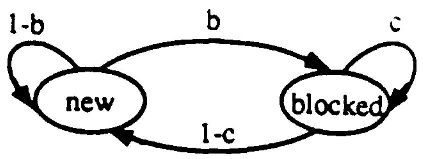


Figure 5: The states of a server during its busy period.

changes the memory referencing pattern. The routing probabilities were created to reflect the steady state memory referencing pattern; therefore, it is necessary to keep the values unchanged.

Although an exact model of this persistent blocking behavior would require that we keep track of how many times a packet has been blocked at a given node, we choose an approximation which captures the "first order" effect of this persistence using the following two state model. When the queue is not empty : we model the server as being either in the "new" state or the "blocked" state. When a packet first comes into the server, the server is in the new state. The server enters the blocked state when the packet is blocked, and it remains in the blocked state until the blocked packet finally goes through to the next stage. This cycle repeats until the server empties the queue and becomes idle. Observe that the server is inactive when it is in the blocked state. While in the new state, the server obeys the renewal behavior choosing an output port according to the routing probability $r_{i,j}$. Hence, we can approximate a blocking switch with "memory" characteristics by a finite buffer queue with a reduced service rate. The reduced portion is the probability that the server is in the blocked state.

The diagram in Figure 5 shows how the server alternates between the new state and the blocked state during its busy period. Let b be the probability that a new packet is blocked when it tries to go to the next stage. Let c be the probability that a blocked packet is blocked again when it tries to go to the same destination. Then for our approximation, the steady state probability that the server is in the blocked state, $P_{blocked}$, can be solved in terms of b and c :

$$P_{blocked} = \frac{b}{1 - c + b}$$

where b is the blocking probability (for which we used the notation $B_{i,j}$ in section 2.3). Once blocked, it is more likely that a blocked packet gets blocked again; therefore, the value of c is selected to be larger than the value of b. In fact, when a packet is in the blocked state, the length of the destination queue in the next cycle will be either K (full) or K-1 (only one space available). If we disregard how many times it has been blocked previously, there will be only two cases : either the blocked packet faces a full queue or a queue with one space left. In the first case, with probability $\frac{P[K]}{P[K]+P[K-1]}$, the packet will be blocked again. In the second case, with probability $\frac{P[K-1]}{P[K]+P[K-1]}$, the packet will face possible contention from the other queue in the same stage which feeds this destination queue. Incorporating these two probabilities in equation (2), c can be found in a similar way :

$$c = r \cdot C_{i+1,j1} + (1-r) \cdot C_{i+1,j2}$$

The probability that the j1-th queue in stage i+1 is blocking a packet in stage i, $C_{i+1,j1}$, is :

$$C_{i+1,j1} = \frac{r}{2} \cdot (1 - P_{i,l}(0)) \cdot \frac{P_{i+1,j1}(K-1)}{P_{i+1,j1}(K) + P_{i+1,j1}(K-1)}$$

$$+ \frac{P_{i+1,j1}(K)}{P_{i+1,j1}(K) + P_{i+1,j1}(K-1)}$$

$P_{blocked}$ is the probability that the server is in the blocked state. During this period, the server is inactive. Therefore, we may use this probability to approximate the blocking switch with "memory" characteristic. At the beginning of each cycle, the server tosses a coin which comes up heads with probability $P_{blocked}$, in which case the server will be blocked (inactive). If there is a packet at the server, it stays idle until the next cycle when the coin will be tossed again. With probability $1 - P_{blocked}$, the server will be active. The queue length then determines whether the server will send a packet or not. If there are packets in the queue, the server takes the first packet and routes it according to the routing probability.

Incorporating the probability $P_{blocked}$ into our previous model, the approach is then similar except that the equivalent input rates and blocking probabilities are different. In the original model, when a queue is not empty (with probability $1 - P_{i,j}(0)$), it tries to transmit a packet to the destination in stage i+1. However, for the persistent blocking model, a queue tries to transmit a packet to the next stage with probability $(1 - P_{i,j}(0)) \cdot (1 - P_{blocked})$, the former is the probability that the server is not empty and the latter is the probability that the server is in the "active" state. When the server is not empty and it is active, it transmits a packet to the next stage.

Let us define $P^{eff}_{i,j}(0)$ to be the effective probability that $Q_{i,j}$ will not send a packet (either the server is empty or the server is not empty and is blocked). Let $P_{i,j,blocked}$ be the probability that $Q_{i,j}$ is not empty and is in the blocked state. Then the effective input rates of a queue, $Q_{i,j}$ in this persistent blocking model are similar to the ones in section 2.3 :

$$X[1] = 2r(1-r)(1 - P^{eff}_{i-1,j1}(0))(1 - P^{eff}_{i-1,j2}(0))$$

$$+ r[P^{eff}_{i-1,j2}(0)(1 - P^{eff}_{i-1,j1}(0)) + P^{eff}_{i-1,j1}(0)(1 - P^{eff}_{i-1,j2}(0))]$$

$$X[2] = [r(1 - P^{eff}_{i-1,j1}(0))] \cdot [r(1 - P^{eff}_{i-1,j2}(0))]$$

$$X[0] = 1 - X[1] - X[2]$$

$$P^{eff}_{i-1,j1}(0) = 1 - (1 - P_{i-1,j1,blocked}) \cdot (1 - P_{i-1,j1}(0))$$

$$P^{eff}_{i-1,j2}(0) = 1 - (1 - P_{i-1,j2,blocked}) \cdot (1 - P_{i-1,j2}(0))$$

The equivalent blocking probability $B_{i,j}$ can be found as follows :

$$B_{i,j} = r \cdot C_{i+1,j1} + (1-r) \cdot C_{i+1,j2}$$

$$C_{i+1,j1} = P_{i+1,j1}(K) + \frac{r}{2} \cdot (1 - P^{eff}_{i,l}(0)) \cdot P_{i+1,j1}(K-1)$$

Incorporating these equivalent input rates and blocking probabilities into the previous model, we can evaluate a multistage interconnection network with persistent blocking behavior. We first calculate the equivalent input rates and blocking probabilities for each queue in the first stage. Then we decompose the first queue in the first stage and solve its Markov chain with the equivalent input rates and blocking probabilities. The remaining queues in the first stage are decomposed and their Markov chains are solved one by one. The steady state probabilities for these queues are used to calculate the equivalent input rates to the queues in the second stage. The queues in the second stage are then decomposed and solved. This process is repeated for all stages. The analytical model iterates this decomposition process until the throughput converges. Then we calculate the time delay using Little's result. Other performance measures can be computed using the steady state parameters of the system.

4.2 Results

We ran our model incorporating this new technique to handle the memory behavior for the same 6-stage Omega network (as in Section 3.5) with buffer size 4 under both the uniform traffic and the EFOS traffic pattern. The result is shown in comparison with the former model in Figure 6. The improved model greatly reduces the discrepancy between the simulation and analytical model results.

For a detailed study, we compare the improved analytical results with simulation in Figures 7-14. The confidence range of these simulations is 95%. The first case shown in Figures 7-8, is for a 4-buffered, 6 stage

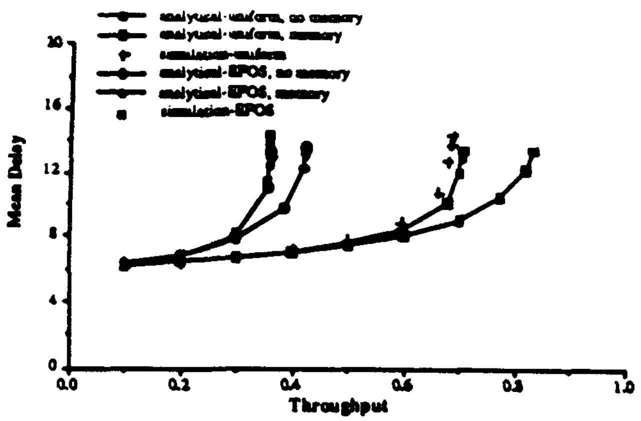


Figure 6: Comparison of results for a 6-stage, 4-buffered Banyan network with and without the "memory" behavior improvement

Omega network with a uniform traffic pattern. In Figure 7, the throughput is compared with various offered loads and in Figure 8, the average time delay is compared. The offered load is varied from 0.1 pkt/cycle to 1.0 pkt/cycle for each processing element. For offered loads within the range of small loss probability ($q \leq 0.7$ pkt/cycle), the simulations verify the accuracy of the analytical results. Beyond this load (when packets begin to be discarded), the analytical results are slightly optimistic.

Figures 9-10 show the case of a 4-buffered, 6 stage Omega network with an EFOS traffic pattern. Throughput and average time delay are plotted against offered load. The non-uniformity of this pattern severely degrades the performance. The throughput graph shows very good correspondence between analytical results and simulations. For offered load within the low-loss range ($q \leq 0.4$ pkt/cycle), delay performance of the analytical result is very accurate. However, delay performance of analytical results are still slightly optimistic in heavy load cases.

The third case is included to determine whether the analytical model performs well with a larger buffer. The results for an 8-buffered, 6 stage Omega network with a uniform traffic pattern are shown in Figures 11-12. Except with heavy load (q=0.9 and 1.0), analytical results measure well when compared to simulations. The throughput and delay performance are optimistic when the total input enters the range of heavy load. This simulation indicates that the analytical model performs well for networks with other buffer sizes.

We show the analytical results of a large sized network in Figures 13-14, namely a 4-buffered, 10 stage (1024x1024) Omega network with a uniform traffic

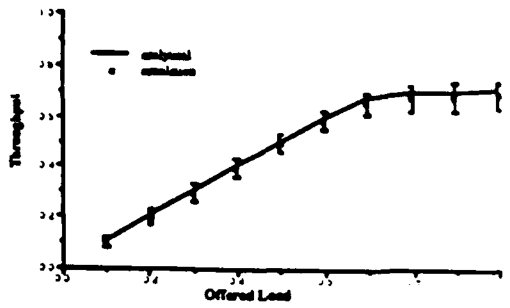


Figure 7: Throughput comparison for a 4-buffered, 6 stage Omega MIN with a uniform traffic pattern

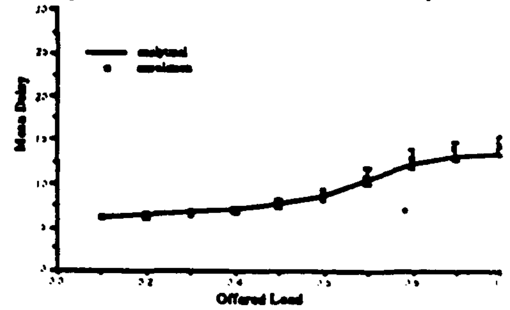


Figure 8: Mean delay comparison for a 4-buffered, 6 stage Omega MIN with a uniform traffic pattern

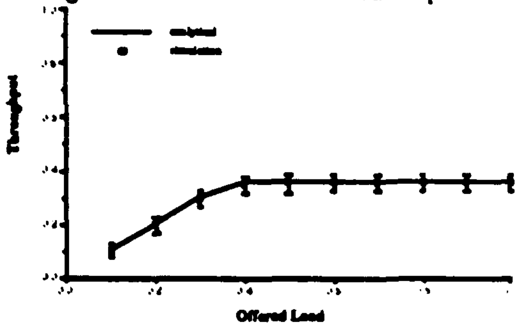


Figure 9: Throughput comparison for a 4-buffered, 6 stage Omega MIN with EFOS traffic pattern

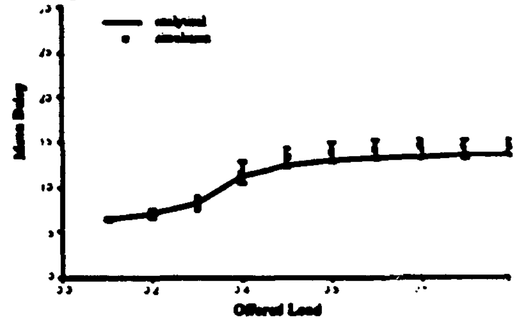


Figure 10: Mean delay comparison for a 4-buffered, 6 stage Omega MIN with EFOS traffic pattern

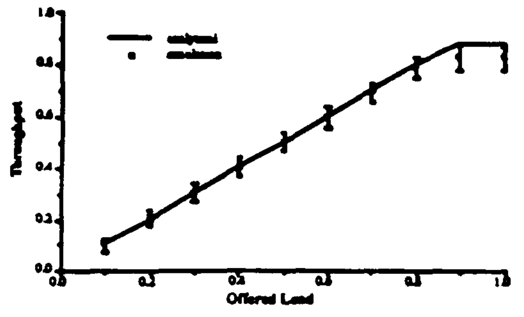


Figure 11: Throughput comparison for an 8-buffered, 6 stage Omega MIN with a uniform traffic pattern

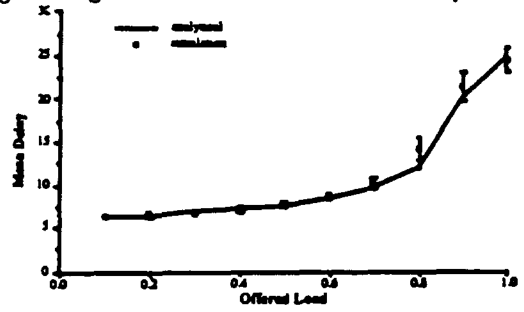


Figure 12: Mean delay comparison for an 8-buffered, 6 stage Omega MIN with a uniform traffic pattern

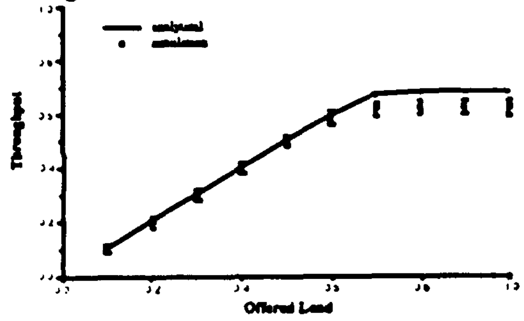


Figure 13: Throughput comparison for a 4-buffered, 10 stage Omega MIN with uniform traffic pattern

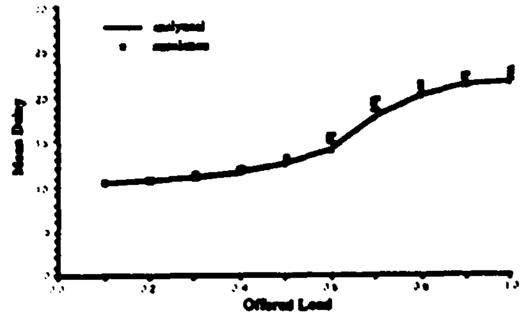


Figure 14: Mean delay comparison for a 4-buffered, 10 stage Omega MIN with uniform traffic pattern

pattern. Again, the analytical model is slightly optimistic when the offered load exceeds the maximal attainable output. This indicates that the model is suitable for large sized networks as well.

4.3 Conclusion

An analytical model was developed to evaluate the performance of finite-buffered Multistage Interconnection Networks with any general traffic pattern. The Interconnection Network was modelled as a network of finite-buffered queues with blocking. We proposed a decomposition and iteration method to analyze this network of queues with blocking. A transformation and superposition method was then proposed to analyze these networks for any general traffic pattern. However, the renewal routing choice assumption caused the modeling results to be optimistic. Therefore, we proposed approximate techniques to model the persistent blocking situation. The analytical results were compared to the simulation in different buffer sizes and different network sizes under both a uniform traffic and a general traffic pattern. The simulations show that the analytical results are very accurate when the offered load does not exceed the maximal attainable total output. For cases where the offered load is beyond the maximal attainable output, the analytical results are slightly optimistic. This verifies the accuracy and the flexibility of the analytical model.

References

[1] L.N. Bhuyan, "An Analysis of Processor-Memory Interconnection Networks ", *IEEE Transaction on Computers*, Vol. c-34, No.3, March 1985, pp.279-283

[2] A. Brandwajn, Y.L. Jow, "An Approximation Method for Tandem Queues with Blocking", *Operations Research*, Vol.36, No.1, January-February 1988, pp.73-83

[3] P. Caseau, G. Pujolle, "Throughput Capacity of a Sequence of Queues with Blocking Due to Finite Waiting Room", *IEEE Transaction on Software Engineering* Vol. SE-5, No. 6, November 1979, pp.631-642

[4] D.M.Dias, J.R. Jump, "Analysis and Simulation of Buffered Delta Networks", *IEEE Transaction on Computers*, Vol.C.30, No.4, April 1981, pp.273-282

[5] L.R. Goke, G.J. Lipovski, "Banyan Networks for Partitioning Multiprocessor Systems", *The Proceedings of the First Annual Symposium on Computer Architecture*, 1973, pp21-28

[6] A. Gottlieb, et al, "The NYU Ultracomputer – Designing an MIMD Shared Memory Parallel Computer", *IEEE Transaction on Computers*, Vol. C.32, No.2, February 1983, pp. 175-189.

[7] Y. Jenq, "Performance Analysis of a Packet Switch Based on Single-Buffered Banyan Network", *IEEE Journal on Selected Areas in Communications*, Vol. SAC-1, No. 6, December 1983, pp.1014-1021

[8] H.S. Kim, A. Leon-Garcia, "Performance of Buffered Banyan Networks under Nonuniform Traffic Patterns", *IEEE Transactions on Computers*, Vol. 38, No. 5, May 1990, pp.648-658

[9] C.P. Kruskal, M. Snir, "The Performance of Multistage Interconnection Networks for Multiprocessors", *IEEE Transaction on Computers*, Vol. C.32, No. 12, December 1983, pp.1091-1098

[10] C.P. Kruskal, M. Snir, A. Weiss, "The Distribution of Waiting Times in Clocked Multistage Interconnection Networks", *Conference of Parallel Processing*, 1986, pp.12-19

[11] C.P. Kruskal, M. Snir and A. Weiss, "The distribution of Waiting Times in Clocked Multistage Interconnection Network", *IEEE Transaction on Computers*, vol. 37, No. 11, November 1988, pp1337-1352

[12] T. Lang, L. Kurisaki, "Nonuniform Traffic Spots (NUTS) in Multistage Interconnection Networks", *UCLA Computer Science Department Technical Report* CSD-880001, January 1988

[13] D.H. Lawrie, "Access and Alignment of Data in an Array Processor", *IEEE Transaction on Computers*, Vol.C-24, Dec. 1975, pp. 1145-1155

[14] T. Lin and A. Tantawi,"Performance Evaluation of Packet-Switched Multistage Interconnection Networks under a Model of Hot-Spot Traffic", *ORSA/TIMS Joint National Meeting*, Philadelphia, PA, October 29-31, 1990.

[15] J.H. Patel, "Processor-Memory Interconnections", *IEEE Transaction on Computers*, Vol. c-30, No. 10, October 1981, pp.306-310

[16] N.M. Patel, P.G. Harrison, "On Hot-Spot Contention in Multistage Interconnection Networks", *ACM SIGMETRICS*, May 1988, pp. 114-123.

[17] H.G. Perros, T. Altiok, "Approximate Analysis of Open Networks of Queues with Blocking : Tandem Configurations", *IEEE Transaction on Software Engineering*, Vol. SE-12, No. 3, March 1986, pp.450-461

[18] G.F. Pfister, V. A. Norton, "Hot-Spot Contention and Combining in Multistage Interconnection Networks", *IEEE Transaction on Computers*, Vol. c.34, No. 10, October 1985, pp.943-948

[19] T. Szymanski, S. Shaikh, "Markov Chain Analysis of Packet-Switched Banyans with Arbitrary Switch Sizes, Queue Sizes, Link Multiplicities and Speedups", *1989 IEEE INFOCOM*, pp.960-971

[20] T.H. Theimer, E.P. Rathgeb, M.N. Huber, "Performance Analysis of Buffered Banyan Networks", *Performance of Distributed and Parallel Systems*, Kyoto University 1988, pp. 57-72

[21] J.S. Turner, "Design of an Integrated Services Packet Network", *IEEE J. Select. Areas of Communication*, vol. SAC-4, no. 8, Nov. 1986, pp.1373-1380

[22] D.L. Willick, D.L. Eager, "An Analytical Model of Multistage Interconnection Networks", *ACM SIGMETRICS* 1990, pp. 192-199

[23] H. Yoon, K.Y. Lee, M.T. Liu, "Performance Analysis of Multibuffered Packet-Switching Networks in Multiprocessor Systems", *IEEE Transaction on Computers*, Vol. 39, No. 3, March 1990, pp.319-327

This work was supported by the Defense Advanced Projects Agency under contract MDA-972-91-J-1011, Advanced Networking & Distributed Systems

The Latency/Bandwidth Tradeoff in Gigabit Networks

Gigabit networks really are different!

Leonard Kleinrock

The bandwidth for data communications has been growing steadily and dramatically over the last twenty years. Some of us remember the early days of data modems which provided access speeds of 10 characters per second (cps) in the late 1960s. When 300 baud speeds became available (providing 30 cps), we thought of it as a major improvement (and it was).

In the mid-70s, as packet-switched networks [1] began to proliferate, we saw the standard set at 64 kb/s trunk speeds;[1] of course, by the time one paid for the software and protocol overhead, we were happy to end up with about 10 kb/s file transfer speeds (by now, the dial-up data modem speeds had reached 2400 bits per second). The killer application which drove the penetration of these X.25 networks was that of transaction processing.

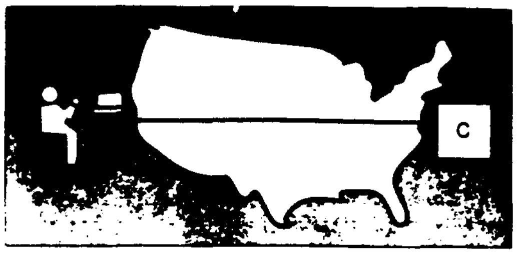
In the 1980s we witnessed the proliferation of T1 channel speeds, providing 1.533 megabit per second (Mb/s) trunk speeds. Private T1 networks exploded in the 1980s because of the cost savings they provided by allowing corporations to integrate their voice and data networks into a single network. This was the killer application for corporate T1 networks. In the scientific community, T1 was introduced toward the end of the 1980s due to the killer applications of e-mail and file transfers; the load from these applications arose very quickly due to the enormous growth in the number of connected users. However, the packet-switched networks still had 64 kb/s backbone speeds due largely to the complex operations the switches were required to carry out; specifically, each switch had to process every packet up to the third layer (the network layer) of the seven-layer OSI architecture.[2]

As we entered the 1990s, we saw a grass roots development in the form of Frame Relay networks [3,4]. These nets offer packet switching at T1 speeds, a significant step above the 64 kb/s packet switching nets of the 1980s. Both hardware and software developments led to these higher speed packet switched networks. On the hardware side, the widespread deployment of fiber optic communication channels by the long-haul carriers was critical. Besides having enormous bandwidths, these fiber optic channels are extremely noise-free, thereby greatly relieving the network of extensive error control. Faster switches have also been developed due to the progress in VLSI technology. However, the communication bandwidth has grown much more rapidly due to fiber optics than has the speed of the switch due to VLSI. Indeed, prior to the fiber optic revolution, the communication link had represented the performance bottleneck, and so one was prepared to waste switch capacity in order to save communication capacity. This took the form of packet switching in which intelligent switches were introduced into our data communication networks in order to dynamically assign the channel bandwidth on a demand basis. However, now that fiber optics has appeared, the communication bandwidth is no longer a constraint; in fact, a reversal in the relative cost of switching and transmission has taken place and has led us to architectures in which the switch has now become the economic as well as the performance bottleneck. Considerable research and development effort is currently under way to produce high-speed packet switches [5].

New protocols which take advantage of these hardware improvements have also been developed. In particular, the ISDN signalling channel (the D channel) uses a streamlined protocol for routing signalling packets (known as the Link Access Protocol for the D channel - LAPD) [6]; indeed, it only processes these packets up to the second layer (the data link layer) of the seven layer model, extracting a minimal amount of network layer information. Frame Relay uses the LAPD protocol for the data channel (rather than just for the signalling channel), thereby achieving much higher transfer speeds than were possible with X.25 packet networks. Thus, by relegating as much function to hardware as possible, by moving function out of the network when possible (e.g., error control on the data packets), and by taking advantage of streamlined packet protocols, Frame Relay is able to achieve packet switching at T1 speeds. The killer application which has been the driving force behind Frame Relay is that of local area network (LAN) interconnection.

Dr. LEONARD KLEINROCK is professor and chairman of computer science, University of California, Los Angeles.

[1] In fact, in 1969, we already had the early ARPANET operating at 50 kb/s out of UCLA.



■ **Figure 1.** *Sending a 1-Mb file across the U.S.*

In addition, we have seen some multi-megabit data network plans, announcements and offerings. Among these are the Fiber Distributed Data Interface (FDDI) at 100 Mb/s [7], Switched Multimegabit Data Services (SMDS) at 45 Mb/s [8], the Distributed Queue Dual Bus access protocol at 45 and 150 Mb/s [9], ATM switches and Broadband ISDN [10] at 155 Mb/s up to 2.4 gigabits per second (Gb/s), the High Performance Parallel Interface (HIPPI) at 800 Mb/s, etc. Indeed, the Synchronous Optical Network (SONET) [11] standard has defined speeds for optical systems well into the multigigabit range.

It is clear we are moving headlong into an era of gigabit per second speeds and networks.

The Major Issue: Latency vs. Bandwidth

As we move into the gigabit world, we must ask ourselves if gigabits represent just another step in an evolutionary process of greater bandwidth systems, or, if gigabits are really different? In the opinion of this author, gigabits are indeed different, and the reason for this difference has to do with the effect of the latency due to the speed of light.

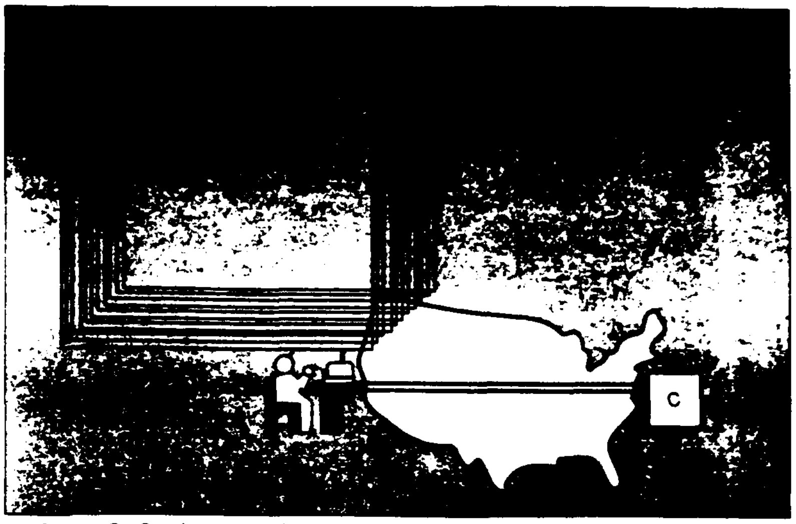
Let us begin by examining data communication systems of various types. It turns out that there are a few key parameters of interest in any data network system. These are:

C = Capacity of the network (Mb/s)
b = Number of bits in a data packet
L = Length of the network (miles)

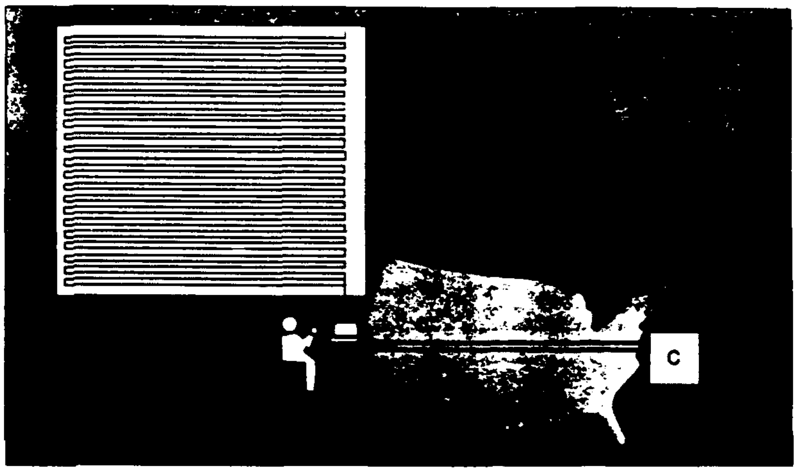
It is simplest to understand these quantities if one thinks of the network simply as a communication link. One can combine these three parameters to form a single critical system parameter, commonly denoted as a, which is defined as:

$$a = 5LC/b \qquad (1)$$

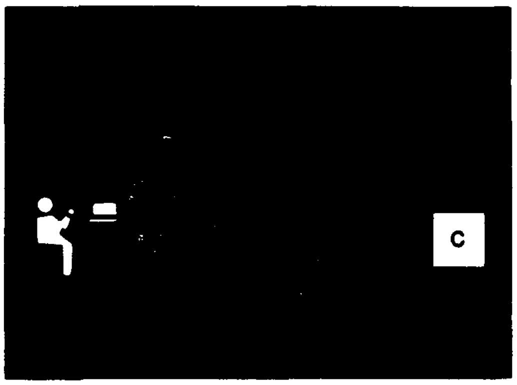
This parameter is the ratio of the latency of the channel (i.e., the time it takes energy to move from one end of the link to the other) to the time it takes to pump one packet into the link. It measures how many packets can be pumped into one end of the link before the first bit appears at the other end [12]. The factor 5 appearing in the equation is simply the approximate number of microseconds it takes light to move one mile.[2] Now, if we calculate this ratio for some common data networks, we find the values shown in Table 1: Note the enormous range for the parameter a. At one extreme, namely, local area networks, it is as small as 0.05, while at the other extreme, namely, a cross-country gigabit fiber optic link, it is as large as 15,000. This is a range of nearly six orders of magnitude for this single parameter!



■ **Figure 2.** *Sending a 1-Mb file across the U.S. via an X.25 network.*



■ **Figure 3.** *Sending a 1-Mb file across the U.S. via a T1 channel.*

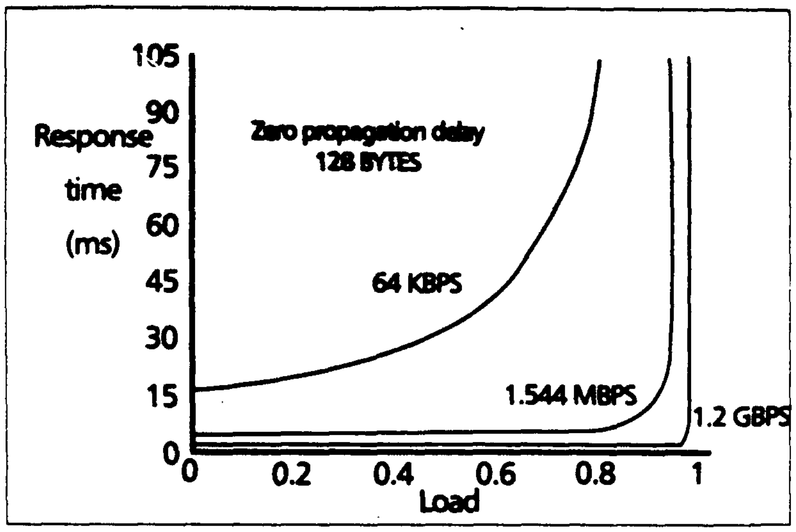


■ **Figure 4.** *Sending a 1-Mb file across the U.S. via a 1.2-Gb link.*

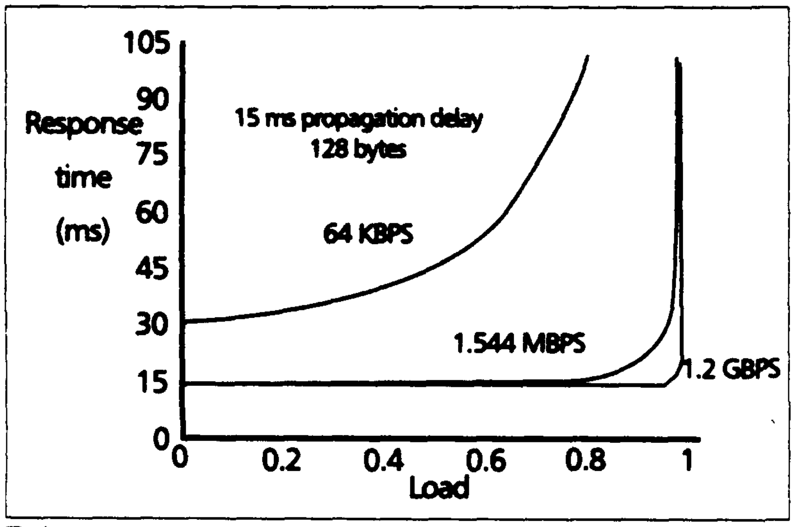
We see that a grows dramatically when we introduce gigabit links. So we naturally must ask ourselves if networks made out of gigabit links are different in some fundamental way from those made out of kilobit or megabit links. There are two cases of interest to consider. First, we have the case that a large number of users are each sharing a small piece of this large bandwidth. In this case it is fairly clear that to each of them, a gigabit network looks no different from today's networks.

However, if we have a few users each sending packets and files at gigabit speeds, then we do

[2] *Throughout this paper we make the simplifying assumption that light propagates in a fiber optic channel as quickly as it propagates in free space.*



■ **Figure 5.** *Response time vs. system load with no propagation delay.*



■ **Figure 6.** *Response time vs. system load with a 15 ms propagation delay.*

see a change in behavior and we do run into new problems. At these speeds, *a* gets very large. To see the effect of this change, let us consider the following scenario. Assume we are sitting at a terminal and wish to send one megabit across the United States to some remote computer as shown in Fig. 1.

Now, if the speed of the communication channel we have available is 64 kb/s (as in, say, an X.25 packet network), then, as shown in Fig. 2, the first bit of this transmission will arrive at the East Coast computer after approximately 1000 bits have been pumped into the channel. Thus we see that the channel is buffering roughly 0.001 of the message; that is, there is 1000 times as much data stored in the terminal's buffer as there is in the channel. Clearly, if we had a higher-speed channel, the time to transmit our 1 Mb file could be reduced. That is, we can benefit from more bandwidth.

Thus, let us now increase the speed of the channel and use a T1 channel (1.544 Mb/s). In Fig. 3 we show this new configuration. Now we find that the terminal is buffering roughly 40 times as much data as is the channel. Once again, we see that we can benefit from more bandwidth.

Let us now increase the channel speed to a gigabit channel; in particular, we will assume a 1.2 Gb/s link (the OC-24 SONET offering). This case is shown in Fig. 4 where we see the entire 1 Mb file as a small pulse moving down the channel. Indeed, the pulse occupies roughly only 0.05 of the channel "buffer." It is now clear that more bandwidth is of no use at all in speeding up the transmission of the file; it is the latency of the channel that dominates the time to deliver the file!

Therein lies the fundamental change that comes about with the introduction of gigabit links into nationwide networks. Specifically, we have passed from the regime (of pre-gigabit networking) in which we were capacity limited, to the new regime of being latency limited in the post-gigabit world. Things do indeed change (as we shall see below). The speed of light is the fundamental limitation for file transfer in this regime! And the speed of light is a constant of nature which we have not yet been able to change!

In the considerations above, we assumed that our file was the only traffic on the link. Let us now consider the case of competing traffic with smaller packets. Indeed, let us now assume that we have the classical queueing model of a Poisson stream of arriving messages requesting transmission over a communication link, where each message has a length which is exponentially distributed with a mean of 128 bytes (i.e., a classic M/M/1 queueing system) [13]. If, as usual, we let ρ denote the system utilization factor, then $\rho = \lambda(1024/C)$ where λ is the arrival rate (messages per microsecond) and C is the channel capacity (Mb/s). In this situation, we know that *T*, the mean response time (milliseconds) of the system (i.e., the mean time from when the message arrives at the tail of the transmit queue until the last bit of the message appears at the output of the channel, including any propagation delay), is given by

$$T = \frac{1.024 / C}{1 - \rho} + \tau \qquad (2)$$

where τ is the propagation delay (i.e., the channel latency) in milliseconds.

Let us ask ourselves if gigabit channels actually help in reducing the mean response time, *T*. In Fig. 5, we show the mean response time (in millisec) versus the system load ρ for three different channel speeds. In this figure, we assume that the speed of light is infinite, and so $\tau = 0$. The channel speeds we choose are the same as those considered above, namely 64 kb/s, 1.544 Mb/s and 1.2 Gb/s. We note a significant reduction in *T* when we increase the speed from 64 kb/s to 1.544 Mb/s; thus, the faster T1 channel helps. However, note that when we go from 1.544 Mb/s to 1.2 Gb/s, we see almost no improvement. (The only region in which there is an improvement with gigabits is at extremely high loads, a situation to be avoided for other reasons). As far as response time is concerned,

	Capacity (C) MBPS	Pkt Lngth (b) Bits	Prop Delay (τ) Microsec	*Ratio "a"
Local net	10.00	[illegible]	[illegible]	[illegible]
Wide Area net	0.05	1000	20,000	1.00
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Fiber link	1,000.00	1000	15,000	15,000.00

■ **Table 1.** *Effect of parameters: propagation delay/Packet Tx time — an enormous variation.*

gigabits do not help here!

One might argue that the assumption of zero propagation delay has biased our conclusions. Not so; in Fig. 6 we show the case with a 15-ms propagation delay, (i.e., the propagation delay across the USA) and we see again that gigabits do not help.

We can sharpen our treatment of this latency-versus-bandwidth discussion as follows. Let us assume that we have an M/M/1 model as above, where the messages have an average length equal to b bits. Assume we wish to transmit these files across the United States, as in the earlier figures. Now, as can be seen from Eq. (2), there are two components making up the response time, namely, the queueing-plus-transmission time delay (the first term in the equation) and the propagation delay (τ). In this paper, we have been discussing the relative size of each of these and we referred to regions of bandwidth- limited and latency-limited systems. Let us now make those concepts more precise. We choose to define a sharp boundary between these two regions. In particular, we define this boundary to be the place where the two terms in our equation are exactly equal, namely, where the propagation delay equals the queueing-plus-transmission time delay. From Eq. (2) we see that this occurs when the bandwidth of the channel takes on the following critical value,

$$C_{CRIT} = \frac{1000b}{(1-\rho)\tau} \qquad (3)$$

In Fig. 7, we plot this critical value of bandwidth (on a log scale) versus the system load ρ; we have drawn this plot for the case of $\tau = 15$ ms and a message length of one megabit. Above this boundary, the system is latency limited, which means that more bandwidth will have negligible effect in reducing the mean response time, T. Below this boundary, the system is bandwidth limited which means that it can take advantage of more bandwidth to reduce T. Note that for these parameters the system is latency limited over most of the load range when a gigabit channel is used; this means that for these parameters, a gigabit channel is overkill so far as reducing delay is concerned.

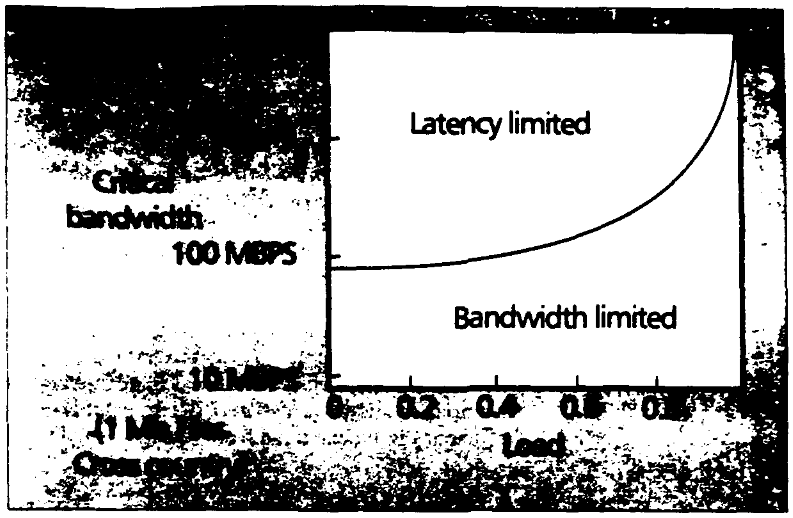
We repeat this plot in Fig. 8 for a number of different message sizes. Without labeling the regions, the same comments apply, namely, systems above the curve are latency limited, and below they are bandwidth limited. We note that gigabit channels begin to make sense for message sizes of size 10 megabits or more, but are not helpful for smaller file sizes. This comment about message size refers to the file size that the user application generates; the fact that ATM uses 53-byte cells has little to do with this comment.

Figures 7 and 8 apply to the case of a cross country link (i.e., with a propagation delay of roughly ms). For other than $\tau = 15$ ms, the critical bandwidth which defines the boundary is given from Eq. (3).

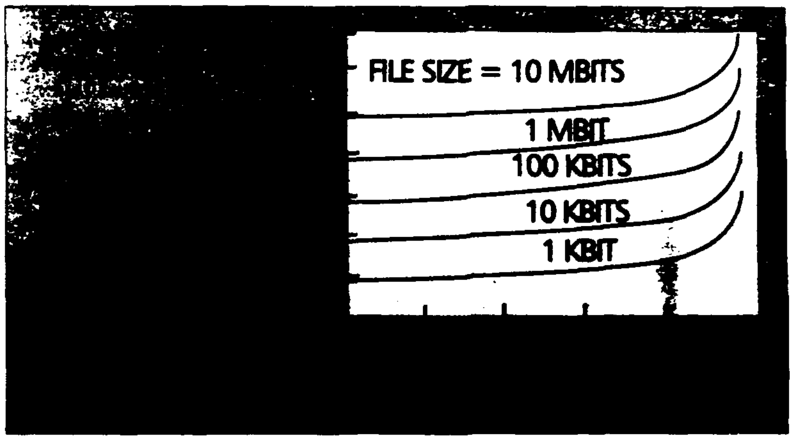
Other Issues

We have dealt with the latency-bandwidth tradeoff for gigabit networks in this paper. Of course there are a number of other issues to be addressed in gigabit nets, some of which we choose to mention in this section.

Consider the example from the previous section,



■ **Figure 7.** *Bandwidth vs. system load for a 1-Mb file sent across the U.S.*



■ **Figure 8.** *Bandwidth vs. system load for files sent across the U.S.*

namely, a gigabit link spanning the United States. Suppose we start transmitting a file a time t=0. Roughly 15 ms later, the first bit will appear across the country. Now suppose that the receiving process decides immediately that it cannot accept this new flow which has begun. By the time the first bit arrives, however, there are roughly 15 million bits already in the pipe heading toward this receiving process! And, by the time a stop signal reaches the source, another 15 million bits will have been launched! It does not take too much imagination to see that we have a problem here. It is basically a congestion control and flow control problem. Clearly, a closed control feedback method of flow control is too sluggish in this environment (due, once again, to latency). Some other forms of control must be incorporated. For example, one could use rate-based flow control in which the user is permitted to transmit at a maximum allowable rate.

Moreover, at the application level, it is important to find ways to hide this latency, in order to get full advantage of the gigabit links and of the high performance processors attached to a gigabit network. One way to hide latency is to use some form of parallelism (or pipelining) such that while one process is waiting for a response, another process, which does not depend upon this response, may proceed with its processing.

Another issue has to do with the maximum attain-

Gigabit networks have forced us to deal with the propagation delay due to the finite speed of light.

able efficiency that one can obtain by taking advantage of statistical multiplexing of bursty sources in a gigabit environment. If we have a large number of small bursty sources, then statistical multiplexing takes exquisite advantage of the Law of Large Numbers [13], and allows one to drive these channels at very high efficiencies. However, if we have a small number of large sources, then the multiplexing does not usually lead to very high efficiencies. This is because statistical smoothing of a small number of sources is not sufficient to bring about the advantages of statistical multiplexing. Furthermore, if we have a large number of nonhomogeneous sources, one must calculate the effective number of such sources in order to calculate the efficiency to be expected from multiplexing [14].

Conclusions

The major conclusion of this paper is to recognize that gigabit networks have forced us to deal with the propagation delay due to the finite speed of light. Fifteen milliseconds to cross the United States is an eternity when we are talking about gigabit links and microsecond transmission times. As we saw earlier, the propagation delay across the USA is forty times smaller than the time required to transmit a 1-Mb file into a T1 link. At a gigabit, the situation is completely reversed, and now the propagation delay is 15 times larger than the time to transmit into the link. We have moved into a new domain in which the considerations are completely reversed. We must rethink a number of issues. For example, the user must pay attention to his file sizes and how latency will affect his applications. The user must try to hide the latency with pipelining and parallelism. Moreover, the system designer must think about the problems of flow control, buffering, and congestion control. Some form of rate-based flow control will help the designer here. He must also design algorithms which make rapid decisions if enormous buffer requirements are to be avoided. The designer cannot depend on global state information being available in a timely fashion; this affects his choice of control algorithms. In many ways, the user will see gigabit networks as being different from megabit networks; the same is true for the designer/implementer.

Much more research must be done before we can claim to have solved many of the problems that this new environment has exposed. We must solve these problems in the near future if we are to enjoy the benefits that fiber optics has given us in the form of enormous bandwidths.

References

[1] Kleinrock, L., *Queuing Systems, Volume II: Computer Applications.* New York, Wiley, 1976.

[2] Tanenbaum, A., *Computer Networks, 2nd Edition.* New Jersey, Prentice Hall, 1988.

[3] Lowe, S.J. "Plugging into frame relay for speed and flexibility", *Data Communications Magazine,* pp. 54-62, April 1990.

[4] Mier, E.E., "New signs of life for packet switching", *Data Communications Magazine,* pp. 90-106, December 1989.

[5] Hui, J. Y. *Switching and Traffic Theory for Integrated Broadband Networks.* Boston, Kluwer, 1990.

[6] Stallings, W., *ISDN: An Introduction.* New York, Macmillan, 1989

[7] Thurber, K. J., "Getting a handle on FDDI", *Data Communications Magazine,* pp. 28-32, June 1989.

[8] "Generic system requirements in support of switched multimegabit data service," Bellcore Technical Advisory, TA-TSY-000772, Issue 3, October 1989.

[9] Kessley, G. C., "IEEE 802.6 MAN," *LAN Magazine,* pp. 102-16, April 1990.

[10] Handel, R., "Evolution of ISDN towards broadband ISDN," *IEEE Network Magazine,* pp. 7-13, January 1989.

[11] Ballart, R. and Y.C. Ching, "SONET: Now it's the standard optical network," *IEEE Communications Magazine,* pp. 8-15, March 1989.

[12] Kleinrock, L., "Channel efficiency for LANs," *Local Area and Multiple Access Networks,* Computer Science Press, Raymond L. Pickholtz, ed., pp.31-41, 1986.

[13] Kleinrock, L., *Queuing Systems, Volume I: Theory.* New York, Wiley, 1975.

[14] Kleinrock, L., "Performance of distributed multiaccess computer-communication systems", *Proceedings of IFIP Congress 77,* pp. 547-52, August 1977.

Biography

LEONARD KLEINROCK is professor and chairman of computer science, University of California, Los Angeles. He is the principal investigator for the DARPA Parallel Systems Laboratory project at UCLA. Dr. Kleinrock is the founder and CEO of Technology Transfer Institute, a computer/communications seminar, conference, and consulting organization located in Santa Monica, Calif. He is a fellow of the IEEE, member of the National Academy of Engineering, recipient of the Ericsson Prize for Outstanding contributions in packet switching technology, and recipient of the 12th Marconi International Fellowship Award for pioneering work in the field of computer networks.

IEEE Communications MAGAZINE

April 1992 VOL 30 NO 4

SPECIAL ISSUE

Gigabit Networks

A Publication of the IEEE Communications Society

A Wavelength Division Multiple Access Protocol for High-Speed Local Area Networks with a Passive Star Topology[1]

Jonathan C. Lu and Leonard Kleinrock
lu@cs.ucla.edu lk@cs.ucla.edu
Computer Science Department
University of California, Los Angeles
3732L Boelter Hall
Los Angeles, CA 90024-1596
(213) 825-3643

Abstract

In this paper a multiple access protocol is proposed for a system consisting of many high-speed bursty traffic stations via an optical passive star coupler. Each user has access over a range of wavelengths, thus resulting in a wavelength division multiplexed communication. The performance of both the infinite and finite population cases has been modeled and analyzed. Numerical results show that low delay and high throughput (larger than the electronic speed of a single user) can be achieved. The analysis also shows that the best performance is obtained when the capacities of the reservation channels and the data channels are balanced.

Keywords: Wavelength Division Multiplexing, Multiaccess Protocol, Local Area Networks, Performance Analysis, Fiber Optics.

[1]This work was supported by the Defense Advanced Research Projects Agency under Contract MDA 903-87-C0663, Parallel Systems Laboratory.

1 Introduction

The explosive advance of fiber optics technology in the past decade offers a combination of wide bandwidth and low attenuation unmatched by any other transmission medium. In high-capacity Local Area Networks (LANs), where distance is short and propagation loss is of little concern, it is the availability of high bandwidth that makes fiber so attractive. It is conceivable that by using the low-loss passband of optical fibers (1200–1600 nm), we could construct multiple access networks carrying a total traffic of around 50 terabits per second [1]. An obstacle to realizing such high-speed transmission of optical signals is the bottleneck at the electronic interface. A fundamental limitation of single-channel high-speed networks such as Expressnet [2], FDDI [3], and DQDB [4] is that in these networks, the maximum throughput of the entire network is limited to the rate that can be supported by the electronics of one of the end user stations. Wavelength Division Multiple Access (WDMA) eliminates this bottleneck by operating on multiple channels at different wavelengths, with each channel running at a moderate speed. However, the control of the WDMA system has proven to be a major obstacle to turning the vast link capacity into system-wide, user-accessible capacity; it is necessary to develop efficient medium access techniques for packet communications in this environment.

Today's electronically tunable semiconductor lasers and filters can tune from one wavelength to another in a few nanoseconds. However, the tuning range is limited. Therefore, each node can only operate on a small number of wavelengths [5]. One class of WDMA networks can be constructed by the use of fixed (wavelength) transmitters and fixed (wavelength) receivers. These networks employ multi-hop topologies in which a packet may be routed through several intermediate nodes before it is delivered to its destination. Examples of this type of networks can be found in [6, 7]. The second class of WDMA networks assumes single-hop communications [8–10] where multiple channels are created by employing tunable transmitters and/or receivers. In [8], a station is equipped with multiple fixed transmitters and multiple tunable receivers. Various architectural alternatives for

WDMA LANs are discussed. In [9], a single tunable transmitter with limited tunability and multiple fixed receivers are provided to each station. A random time division multiple access (TDMA) protocol is used to determine which wavelength a station is allowed to transmit on. In [10], tunable filters (i.e., receivers) capable of rapid tuning over a large number of channels are assumed in order to support packet switching. Also, large packets have to be used since the slot (and thus the packet) size is proportional to the number of stations in the network.

The WDMA local area network can be realized using a bus or a star topology. A star topology is preferred [1] because the station-to-station link attenuation grows linearly with the number of stations. In a bus topology, the excess loss is grows quadratically with the number of stations.

The system under consideration in this paper is a passive star network as shown in Figure 1. There are $(W + 1)$ wavelengths available, $\lambda_0, \lambda_1, \ldots, \lambda_W$ to serve N attached stations. The channel at wavelength λ_0 serves as the control channel for the exchange of the control traffic, while the other W channels are for actual data traffic. Each station is equipped with two lasers: one fixed laser tuned at λ_0 and the other laser tunable to any of the wavelengths $\lambda_1, \ldots, \lambda_W$. The output of the two lasers is coupled into a 2×1 combiner, the output of which is connected to one of the inputs of the $N \times N$ star coupler. Signals transmitted at all of the $(W + 1)$ wavelengths are combined at the star coupler and distributed to all of the stations. Each station also has two receivers: one fixed filter tuned at λ_0 and the other tunable to any of the wavelengths $\lambda_1, \ldots, \lambda_W$. At the receiver, the input optical signal is split into two parts by means of a 1×2 splitter. One part goes to the fixed optical filter which passes only the control wavelength λ_0, and the other output goes to the tunable filter which is tuned to pass the desired data wavelength.

In this paper, a multiaccess protocol based on reservation-ALOHA [11] is proposed and analyzed. In Section 2, we describe the details of the protocol. Section 3 presents the analysis of mathematical models of both the infinite and the finite population cases. In Section 4, numerical results from both analysis and simulation are given. Section 5 concludes the paper.

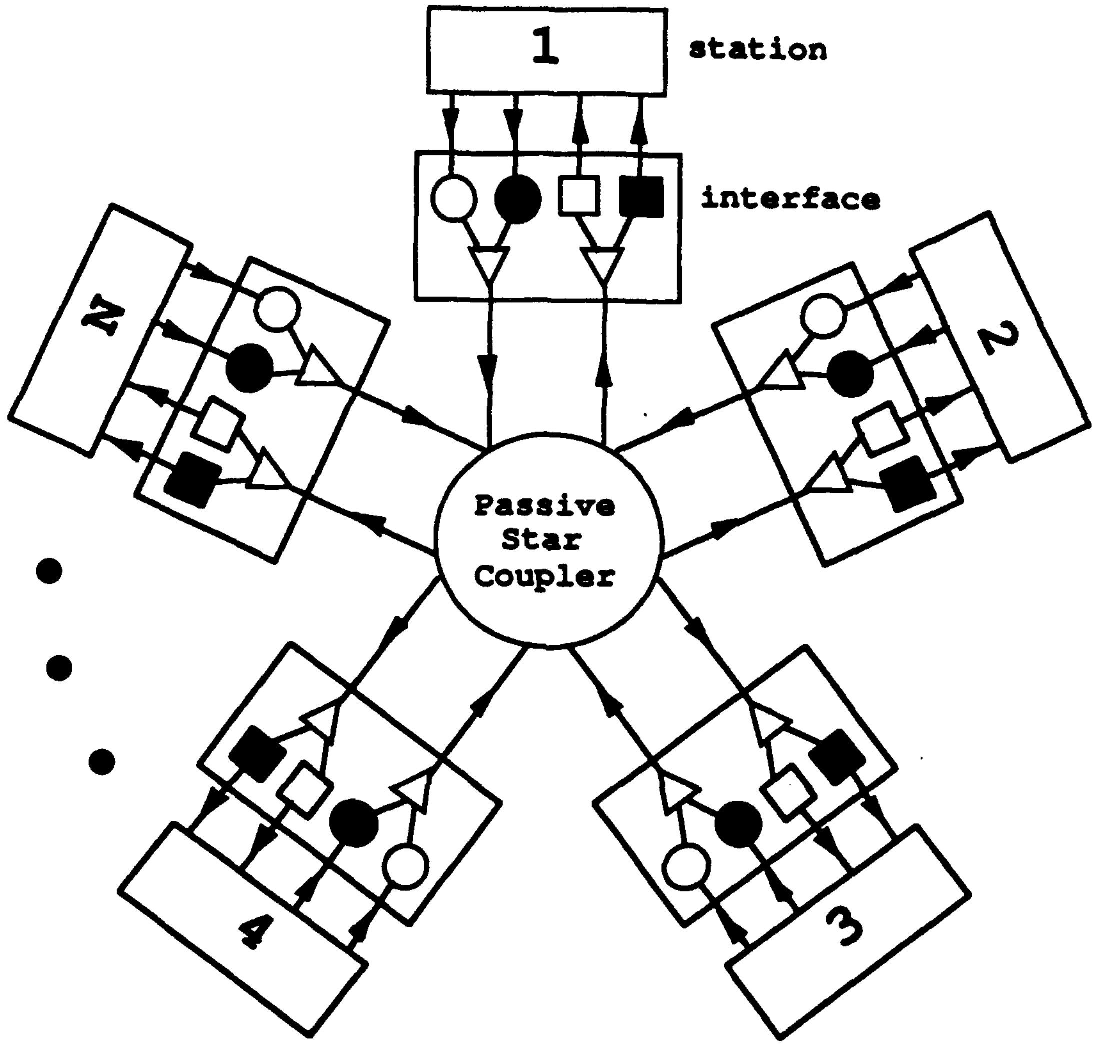
2 Description of Protocol

2.1 The Protocol

We assume the existence of a common clock, obtained either by distributing a clock to all the stations or by means of some self-clocking mechanism inherent in the data. The problem of generating the global clock is addressed and solved in [12]. Time is divided into slots. Packets are of fixed length, which is equal to one slot. The propagation delay from any station to the star coupler and then to any other station is assumed to be equal to R slots. Slots on the data channels are called *data slots* and contain the actual data packets. Slots on the control channel are called *control slots* because they carry only control information about the packets and the transmitters. Each control slot consists of a reservation subpart and a tuning subpart. The reservation subpart is divided into V minislots to be used on a contention basis with the slotted ALOHA protocol, and the tuning subpart is divided into W minislots to convey the wavelength tuning information. The structure of a control slot is shown in Figure 2.

A station generating a packet will randomly select one of the V reservation minislots in the next control slot and transmit a reservation minipacket on the control channel. R slots later the station will hear the result of its reservation. If it is successful, it is received by all the stations because of the broadcast nature of the control channel. All successful reservations join a common distributed queue of stations waiting to transmit. If there is a collision, the station will transmit another reservation minipacket in the next slot with probability p, and with probability $(1 - p)$ it will defer the decision by one slot and transmit the reservation in this next slot with probability p, etc.

In the tuning subpart of each control slot, each of the first W stations in the distributed queue will transmit a tuning minipacket in an assigned tuning minislot; the minipacket contains the destination address and other relevant information



●	tunable transmitter	■	tunable receiver
○	fixed transmitter tuned at λ_0	□	fixed receiver tuned at λ_0

Figure 1: N stations connected by a passive star coupler

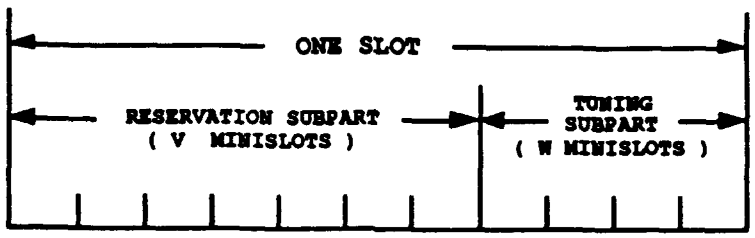

ONE SLOT
RESERVATION SUBPART (V MINISLOTS)
TUNING SUBPART (W MINISLOTS)

Figure 2: Structure of a control slot

for the data packet to be transmitted in the next slot. In particular, the first station in the queue transmits its minipacket in the first tuning minislot, the second station in the queue transmits in the second minislot, ..., and the Wth station transmits in the Wth minislot. The position of the tuning minislot uniquely determines the wavelength to use. At the beginning of the next slot, those W stations tune their tunable transmitters to their assigned wavelengths, with the ith station in the queue using wavelength λ_i to transmit its data packet. When there are fewer than W stations in the distributed queue, some data wavelengths will be unused. When the destination sees its address announced in a tuning minislot on the control channel, it tunes its tunable receiver to the corresponding wavelength and receives the data packet at the beginning of the next slot. If two or more packets are addressed to the same destination in a slot, we arbitrarily select the one transmitted on the lower wavelength number to win the competition (the arbitration can also be made by the use of relevant information carried in the tuning minipacket, such as the packet age or priority). The losing stations must start over with the reservation procedure again.

Thus, a station desiring to send a packet must first compete on the ALOHA reservation subchannel to gain access to a minislot on the tuning subchannel. The station then informs its intended receiver to listen (i.e., tune) to a particular wavelength on which the data packet will be transmitted. If a given receiver is informed by more than one station, only one station will be selected, and the others must repeat the entire procedure.

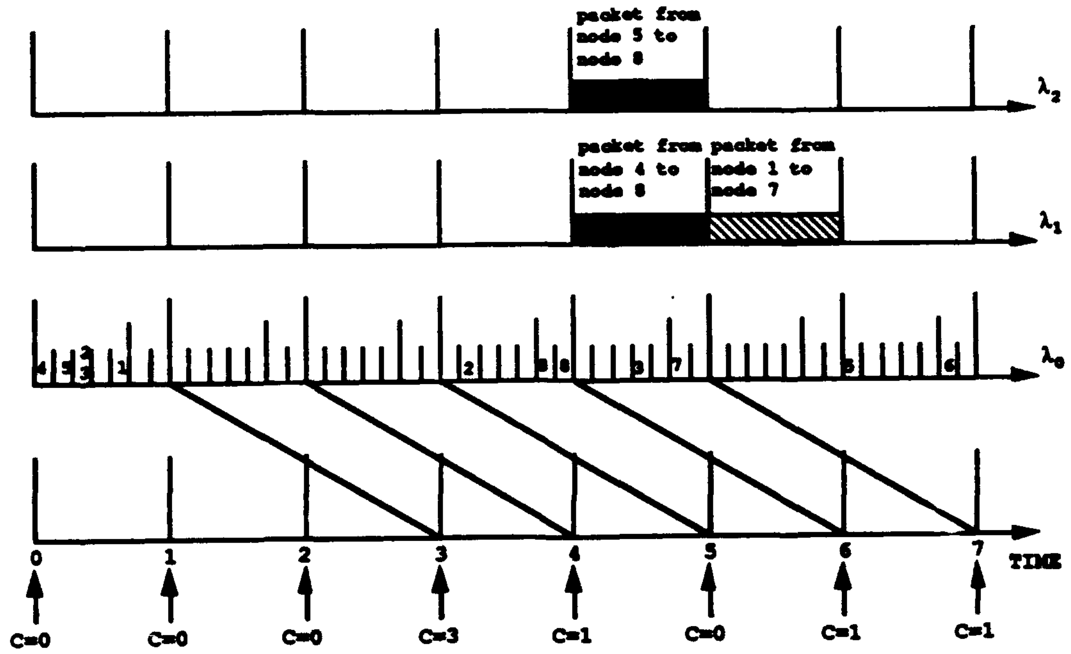
2.2 An Illustrative Example

Here we give a simple example where we assume N=10, V=5, W=2, and R=2. Let slot t denote the slot from time t to time $t + 1$. Define (s,d) as a packet, where s denotes the source node and d the destination. At time 0, five packets, (1,7), (2,6), (3,9), (4,8), (5,8), are generated. In slot 0, five reservation packets are transmitted with node 2 and 3 transmitting in the same minislot. At time 3, nodes 2 and 3 find out that they are involved in a collision, while nodes 4, 5, and 1 realize the success of their reservations and join the distributed queue (in the order 4, 5, 1). Define C_t as the length of the distributed queue at time t. At this time the queue length C_3 equals three. In slot 3, node 2 tosses a coin and decides to transmit a reservation minipacket again, while node 3 tosses a coin and decides to defer the decision by one slot. Meanwhile, in the tuning subpart of slot 3, nodes 4 and 5 write their destination addresses (both are 8) into tuning minislots 1 and 2, and, in slot 4, transmit their actual data packets on wavelengths λ_1 and λ_2, respectively. In slot 5, node 8 finds out from the control channel that two packets are coming for it and tunes its tunable receiver to λ_1 (the lower wavelength) to receive the data packet from node 4. At the same time node 5 realizes that it lost the competition (because of the broadcast nature of the control channel). It tosses a coin, then decides to transmit a reservation minipacket in slot 6 and restarts its reservation procedure again.

3 Performance Analysis

3.1 Model Assumptions

We assume that there are (W+1) wavelenghts available and that the number of stations in the network is N. Each station has a single buffer which is equal to the size of a packet. A new packet arrives at a station with an empty buffer with probability σ at the end of a slot. A packet generated by a station is addressed to any of the other (N-1) stations with equal probability. A source station with a full


packet from node 5 to node 8
λ2
packet from node 4 to node 8
packet from node 1 to node 7
λ1
4 5 2 3 1 2 8 8 3 7 5 6
λ0
0 1 2 3 4 5 6 7
TIME
C=0
C=0
C=0
C=3
C=1
C=0
C=1
C=1

C : the length of the distributed queue

Figure 3: A scenario of packet transmissions. (N=10, V=5, W=2, R=2.)

buffer (i.e., one packet) will not discard its packet until its successful reception at its destination is recognized.

3.2 The Infinite Population Case

In this subsection we consider the infinite population case where the number of stations, N, is infinitely large. We assume that the total reservation traffic offered to the V ALOHA channels forms a Poisson process with rate G requests per slot. Thus the traffic offered to one ALOHA channel is G/V requests per slot. Specifically, $G = \lim_{\substack{\sigma \to 0 \\ N \to \infty}} N\sigma$. The probability that exactly one reservation is transmitted in a single ALOHA channel is $\frac{G}{V}e^{-\frac{G}{V}}$ [13]. Define A_t as the number of successful requests in slot t. Let $A \triangleq \lim_{t\to\infty} A_t$. The probability mass function (pmf) and z-transform of A are

$$a_j \triangleq Prob[A=j] = \binom{V}{j}(\frac{G}{V}e^{-\frac{G}{V}})^j(1-\frac{G}{V}e^{-\frac{G}{V}})^{V-j} \qquad j=1,\ldots,V$$

$$A(z) \triangleq \sum_{j=0}^{\infty} a_j z^j = \left[1 - \frac{G}{V}e^{-\frac{G}{V}} + \frac{Gz}{V}e^{-\frac{G}{V}}\right]^V$$

The throughput of the V ALOHA reservation channels is

$$S_{V\text{-}ALOHA} = E[A] = \left.\frac{dA(z)}{dz}\right|_{z=1} = Ge^{-\frac{G}{V}} \tag{1}$$

with the maximum throughput, V/e, occurring at $G = V$. Therefore, the capacity of the system will be the minimum of the reservation channel throughput and the data channel throughput, namely, $\min(V/e, W)$.

The packet delay D, defined as the time interval from the packet arrival instant to the successful reception at the destination, can be computed as follows :

$$D = D_r + D_q + (R+1)$$

where D_r is the reservation delay defined as the interval between the packet arrival instant and the moment the success of the reservation is recognized. D_q is the queueing delay, which is the time period from the instant the success of reservation is recognized until the beginning of the successful transmission of the data packet. $(R+1)$ slots account for the transmission (1 slot) and propagation delay (R slots). Therefore, the average packet delay is

$$E[D] = E[D_r] + E[D_q] + (R+1) \tag{2}$$

We compute $E[D_r]$ first. Define

$$q_r \triangleq Prob\,[\text{a transmitted reservation request is successful}] = e^{-\frac{G}{V}}$$

Therefore, we have

$$\begin{aligned} r_n &\triangleq Prob\,[\text{reservation succeeds at the } n\text{th trial}] \\ &= q_r(1-q_r)^{n-1} \end{aligned}$$

and

$$\begin{aligned} r &\triangleq \text{the average number of reservation requests sent per packet} \\ &= 1/q_r = e^{\frac{G}{V}} \end{aligned}$$

The average time between two consecutive transmissions of the request is $(\frac{1}{p}+R)$; therefore,

$$E[D_r] = (R+1) + (r-1)(\frac{1}{p}+R) \tag{3}$$

To compute $E[D_q]$, let $C = \lim_{t\to\infty} C_t$ be the length of the distributed queue at the end of a slot in steady state, and X be the position of a typical (say tagged) successful reservation among those successful ones in the same slot. $E[D_q]$ can be computed as follows:

$$E[D_q] = \left\lceil \frac{E[C]+E[X]}{W} \right\rceil \tag{4}$$

To get $E[C]$, we first have

$$C_{t+1} = \max(0, C_t + A_{t+1} - W)$$

Assume that steady state exists. Solving this using the technique in [14], we get

$$C(z) = \sum_{j=0}^{\infty} c_j z^j = \frac{\sum_{i=0}^{W-1}\sum_{j=0}^{W-i-1} c_j a_j (z^{i+j} - z^W)}{(1 - \frac{G}{V}e^{-\frac{G}{V}} + \frac{Gz}{V}e^{-\frac{G}{V}})^V - z^W}$$

where $c_j \triangleq Prob(C = j)$ is the pmf of C. We denote the denominator of $C(z)$ by $D(z)$. Using Rouche's theorem [14], it can be shown that W roots of $D(z)$ are on or inside the unit circle $|z| = 1$. Those roots must cancel out with the roots of the numerator. Therefore, $C(z)$ becomes

$$C(z) = \frac{B}{(z - z_1)(z - z_2)\cdots(z - z_{(V-W)})}$$

where $z_1, z_2, \ldots, z_{(V-W)}$ are the $(V-W)$ roots of $D(z)$ outside the unit circle $|z| = 1$ and B is a constant. The condition $C(1) = 1$ gives us $B = (1 - z_1)\ldots(1 - z_{(V-W)})$. Therefore,

$$C(z) = \frac{(1 - z_1)(1 - z_2)\cdots(1 - z_{(V-W)})}{(z - z_1)(z - z_2)\cdots(z - z_{(V-W)})}$$

and

$$E[C] = \left.\frac{dC(z)}{dz}\right|_{z=1} = -\sum_{i=1}^{V-W} \frac{1}{1 - z_i} \tag{5}$$

Now we compute $E[X]$. Let $x_j \triangleq Prob(X = j)$ be the pmf of X. Define the random variable K to be the total number of successful reservations in the same slot where the tagged successful reservation resides. The pmf for K is

$$Prob(K = k) = \frac{ka_k}{E[A]} \qquad k = 1, \ldots, V.$$

Since the tagged reservation can be at any position in a group with equal probability, we have

$$Prob(X = j|K = k) = \frac{1}{k} \qquad j = 1, \ldots, k.$$

Unconditioning on k, we get

$$\begin{aligned} x_j &= \sum_{k=j}^{V} \frac{1}{k} \frac{ka_k}{E[A]} \\ &= \frac{1}{E[A]} \sum_{k=j}^{V} a_k \qquad j = 1, \ldots, V \end{aligned}$$

and

$$\begin{aligned} E[X] &= \sum_{j=1}^{V} j x_j = \frac{1}{2} + \frac{E[A^2]}{2E[A]} \\ &= 1 + \frac{1}{2}(V-1)\frac{G}{V}e^{-\frac{G}{V}} \end{aligned}$$

where we have evaluated $E[A^2] = Ge^{-\frac{G}{V}}\left[1 + (V-1)\frac{G}{V}e^{-\frac{G}{V}}\right]$ from the expression for $A(z)$ given earlier. $E[D_q]$ can now be computed according to Eq. (4). The average packet delay is finally obtained from Eq. (2).

In Figure 4, we plot the ALOHA channel capacity and the data channel capacity. For a fixed value of the reservation traffic, we see that the throughput is the minimum of the ALOHA channel capacity and the number of wavelengths. Figure 5 shows the throughput-delay curve. These curves give the performance for an infinite population of stations whose collective generation rate of new packets is S. From Figure 5 we see that the system is not stable (there are two different values of delay associated with a given throughput) and some dynamic control procedure (e.g., see [15]) will be required to stabilize the system.

3.3 The Finite Population Case

In this subsection we analyze the case when the number of stations in the system is finite. An approximate model of the system is shown in Figure 6. In this model, each station can be in one of the following $(3R+3)$ *modes* at any instant: TH, RT, Q, PQ_m, PR_m, and PS_m $(1 \leq m \leq R)$. Each station can move from one mode to another mode only at the beginning of each slot.

Stations in each mode act as follows. Stations in the TH (thinking) mode generate a packet with probability σ at the end of a slot. Stations in the Q (queued) mode are currently in the distributed queue. A station that had suffered a collision of its reservation packet and has realized it is said to be in the RT (retransmission) mode and will retransmit the reservation with probability p in the next slot. Stations in the PQ_m mode will move into the PQ_{m-1} mode at the next slot with probability 1. Thus, as can be seen in Figure 6, stations in the PQ_m mode will

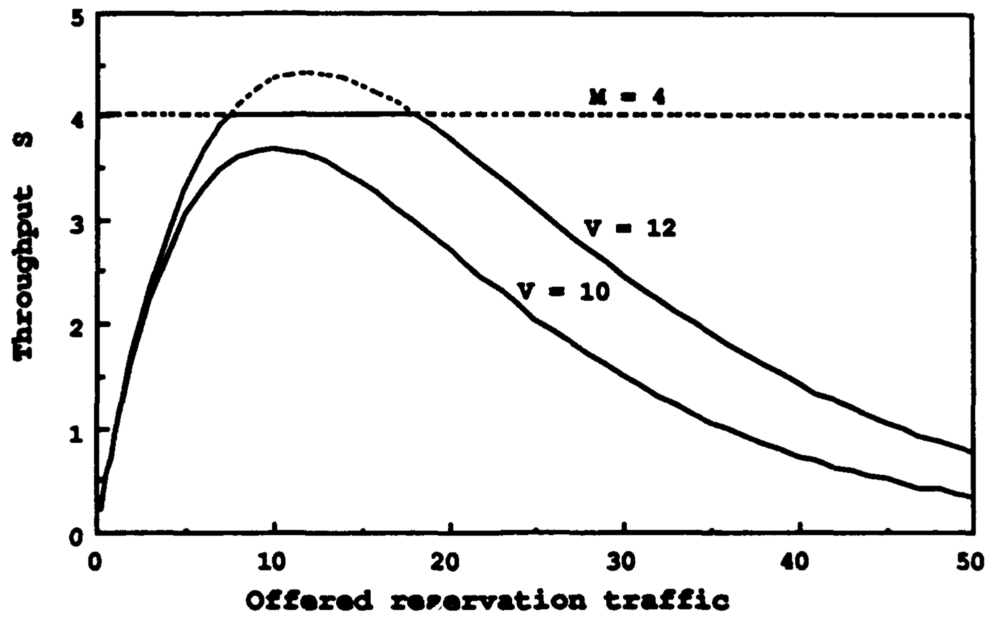


Figure 4: Throughput versus offered reservation traffic. $N = \infty, W=4$.

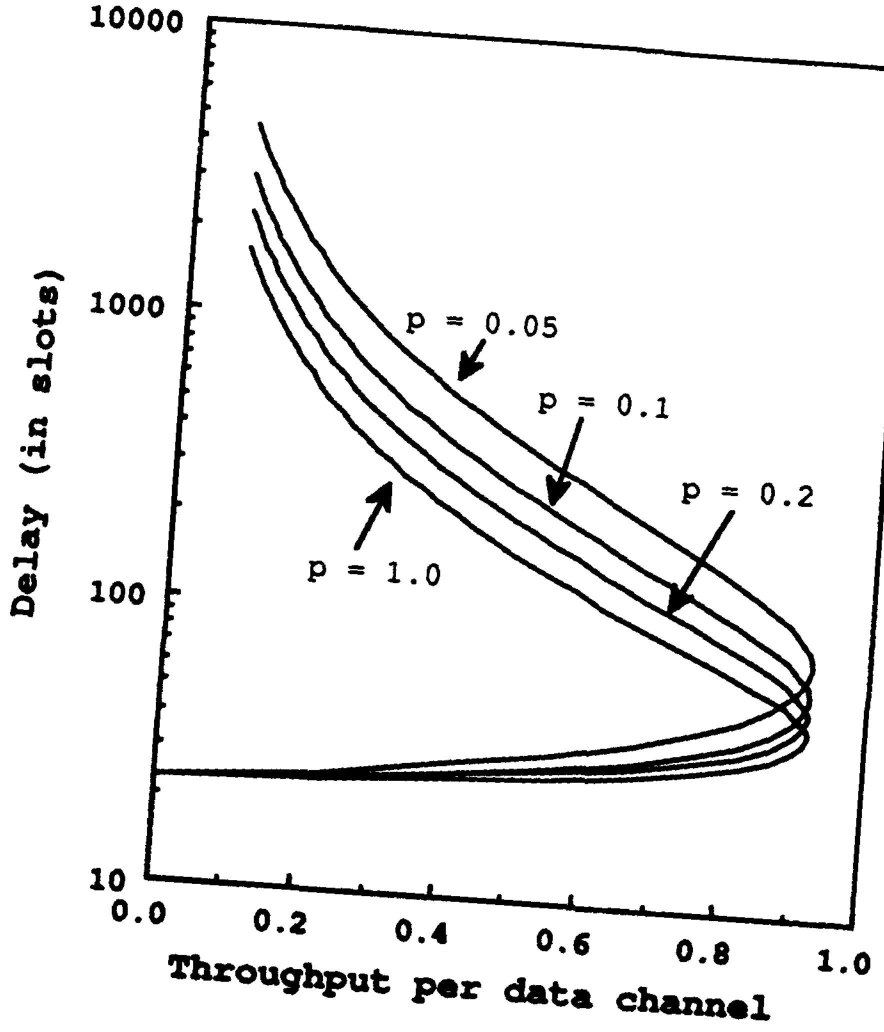


Figure 5: Throughput per data channel versus delay curve. $N = \infty$, V=10, W=4, R=10.

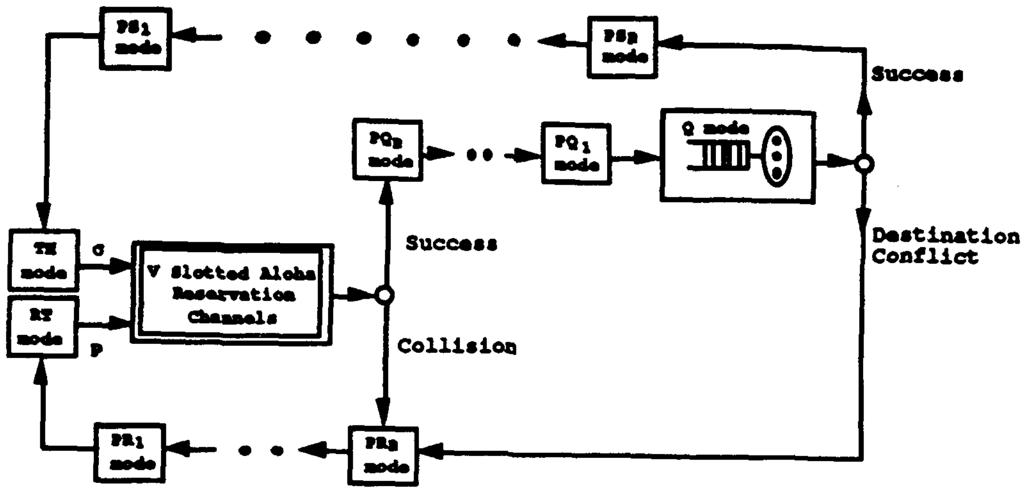


Figure 6: An approximate model of the system

enter the Q mode after m slots. The meaning of the PR_m and the PS_m modes is similar to that of the PQ_m mode; that is, these three kinds of modes are unit delay elements and express the influence of the channel propagation delay.

We now define a state vector of the system. Let n_{RT} be a random variable representing the number of stations in the RT mode, n_Q that in the Q mode, i_m that in the PR_m mode, j_m that in the PR_m mode, and k_m that in the PS_m mode, $m = 1, \ldots, R$. In the model we will further make a *nonpersistence assumption* : a station, upon entering the RT mode, will randomly reselect a destination for its packet (It is not the case in the real system, but later we will see that the model still predicts the performance very well under this assumption). Define the vector $\mathbf{n} \triangleq (n_{RT}, n_Q, i_1, \cdots, i_R, j_1, \cdots, j_R, k_1, \cdots, k_R)$ as the state vector of the system. Then we can see that the vector $\mathbf{n}$ forms a discrete-time Markov chain with a finite state space.

Unfortunately, since the state space is so large, it is difficult for us to solve this Markov chain. Therefore, we utilize the technique of equilibrium point analysis (EPA) [16] to analyze this chain.

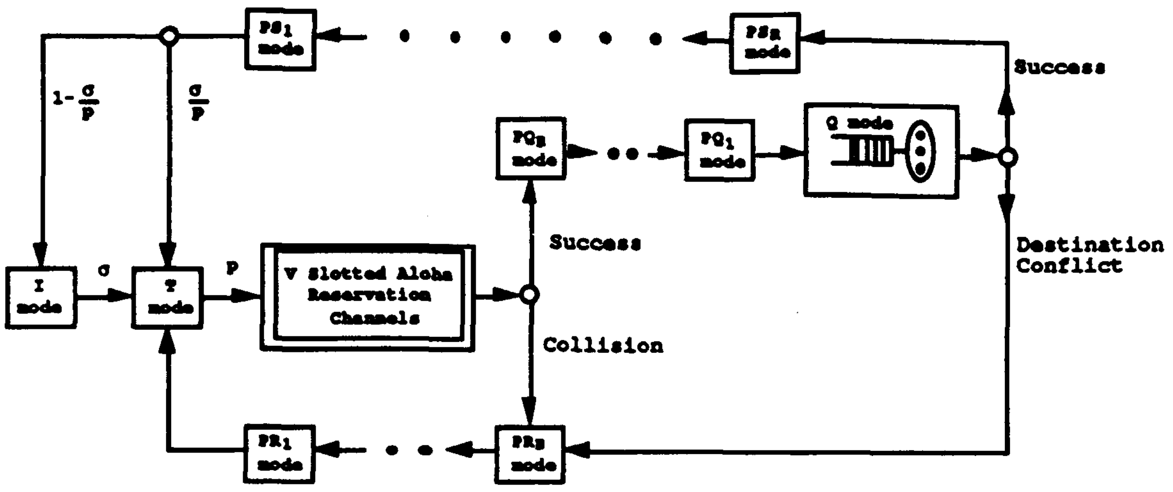


Figure 7: A modified model of Fig. in the case of $\sigma \leq p$

3.3.1 The Modified Model

For simplicity of analysis, we first consider a modification of the model in Figure 6 as suggested in [16], which combines the two inputs (from the *TH* mode and the *RT* mode) of the slotted ALOHA reservation channel into one. Since we have assumed bursty users, we shall confine ourselves to the case $\sigma \leq p$. The modified model is shown in Figure 7, where the *TH* mode in Figure 6 has been decomposed into two modes, *I* and *T*, and the *RT* mode in Figure 6 has become part of the *T* mode. A station that has just moved out from the PS_1 mode moves into the *I* and the *T* modes with probabilities $(1 - \sigma/p)$ and σ/p, respectively, instead of entering the *TH* mode with probability 1. A station in the *I* mode will move into the *T* mode at the next slot with probability σ, and a station in the *T* mode transmits a reservation minipacket (i.e., moves out from the *T* mode) with probability p. The model in Figure 7 is equivalent to that in Figure 6 from the viewpoint of the stochastic behavior to be explained below; consequently, we can derive any characteristic of the model in Figure 6 by using the model in Figure 7. Thus, there is no performance difference between these two models.

The equivalence of the two models can be interpreted as follows. Let X_1, Y_1, and Y_2 be random variables representing the time (number of slots) during which a station is in the *TH, RT,* and T modes, respectively. Also let X_2 be a random variable representing the time from the moment a user enters either the I mode or the T mode from the PS_1 mode until the instant the station moves out of the T mode. (Note that X_2 corresponds to X_1.) It can be shown [16] that X_2 and Y_2 have the same pmf's as X_1 and Y_1, respectively.

In the modified model in Figure 7, we now let n_T be a random variable representing the number of stations in the T mode. Then, it is apparent that the modified state vector $\mathbf{n} = (n_T, n_Q, i_1, \cdots, i_R, j_1, \cdots, j_R, k_1, \cdots, k_R)$ is also a Markov chain under the nonpersistence assumption.

3.3.2 The Equilibrium Point Equation

An equilibrium point is a point in state space such that at that point the expected increase in the number of stations in each mode per unit time is zero [17]. In the EPA method, we assume that the system is always at an equilibrium point. Applying the above condition to all the modes, we get a set of equations called equilibrium point equations, whose solution gives one or more equilibrium points.

Let $\overline{\mathbf{n}} = (\overline{n}_T, \overline{n}_Q, \overline{i}_1, \cdots, \overline{i}_R, \overline{j}_1, \cdots, \overline{j}_R, \overline{k}_1, \cdots, \overline{k}_R)$ be an equilibrium point. Let $\delta_I(\overline{\mathbf{n}})$ denote the conditional expectation of the increase in the number of stations in the I mode in a slot, given that the system is at $\overline{\mathbf{n}}$. Setting $\delta_I(\overline{\mathbf{n}}) = 0$, we then have

$$\delta_I(\overline{\mathbf{n}}) = \overline{k}_1(1 - \frac{\sigma}{p}) - \left[N - \overline{n}_T - \overline{n}_Q - \sum_{m=1}^{R}(\overline{i}_m + \overline{j}_m + \overline{k}_m)\right]\sigma = 0 \qquad (6)$$

Next, let $X(\overline{\mathbf{n}})$ denote the conditional expectation of the number of stations that move out from the Q mode in a slot, given that the system is in state $\overline{\mathbf{n}}$. Evaluating mode Q, we get

$$\delta_Q(\overline{\mathbf{n}}) = X(\overline{\mathbf{n}}) - \overline{j}_1 = 0 \qquad (7)$$

Next we define two other terms. Let $f_T(\overline{\mathbf{n}})$ denote the conditional expectation of the number of stations that successfully transmit reservation minipackets (thus move from mode T to mode PQ_R) in a slot, given that the system is in state

$\overline{\mathbf{n}}$. Let $g_Q(i)$ denote the average number of stations that successfully transmit a data packet (therefore moving from mode Q to mode PS_R) in a slot, given that i stations transmit (i.e. move out of mode Q). It can be derived (see the Appendix) that

$$f_T(\overline{\mathbf{n}}) = \overline{n}_T p(1 - \frac{p}{V})^{\overline{n}_T - 1}$$

and

$$g_Q(i) = N\left[1 - (1 - \frac{1}{N-1})^{i-1}(\frac{N^2 - 2N + i}{N(N-1)})\right]$$

Evaluating the conditional expectation of increase for the PR_m, PQ_m, and $PS_m (1 \leq m \leq R)$ modes, we obtain the corresponding equations as follows.

$$\overline{i}_1 = \overline{i}_2 = \cdots = \overline{i}_R = \overline{n}_T p - f_T(\overline{\mathbf{n}}) + X(\overline{\mathbf{n}}) - g_Q(X(\overline{\mathbf{n}})) \tag{8}$$

$$f_T(\overline{\mathbf{n}}) = \overline{j}_R = \cdots = \overline{j}_1 \tag{9}$$

$$\overline{k}_1 = \cdots = \overline{k}_R = g_Q(X(\overline{\mathbf{n}})) \tag{10}$$

We did not write down the equation for the T mode above since it is linearly dependent on the others. From the equations above, we get the following equations:

$$f_T(\overline{\mathbf{n}}) - X(\overline{\mathbf{n}}) = 0 \tag{11}$$

$$g_Q(X(\overline{\mathbf{n}}))(1 - \frac{\sigma}{p}) - [N - \overline{n}_T - \overline{n}_Q - R(\overline{n}_T p + f_T(\overline{\mathbf{n}}))]\,\sigma = 0 \tag{12}$$

We model the queueing system in mode Q as a W-server system with a binomial input with mean $f_T(\overline{\mathbf{n}})$ and a fixed one slot service time for each customer. Define

$$\rho = \frac{f_T(\overline{\mathbf{n}})}{W}$$

which is the utilization of the queueing system. The z-transform of the arrival process is

$$A(z) = (1 - \frac{\rho W}{V} + \frac{\rho z W}{V})^V$$

This system was solved in the previous section, and the average number of customers in the system (see Eq. (5)) should be equal to $\overline{n}_Q$, the average number of stations in mode Q. Therefore, we have

$$\overline{n}_Q = -\sum_{i=1}^{V-W} \frac{1}{1-z_i} \tag{13}$$

where $z_i, i = 1, \ldots, (V-W)$, are the roots of $A(z) - z^W = 0$ outside the unit circle $|z| = 1$. Also, since $X(\overline{\mathbf{n}}) = f_T(\overline{\mathbf{n}}) = \rho W$, equations (11) and (12) become

$$\overline{n}_T p(1 - \frac{p}{V})^{\overline{n}_T - 1} - \rho W = 0 \tag{14}$$

$$\frac{g_Q(\rho W)}{\sigma}(1 - \frac{\sigma}{p}) = N - (1 + pR)\overline{n}_T - \overline{n}_Q - pRW \tag{15}$$

It is apparent that the equations (13) (14) and (15) can be solved for ρ. The system is said to be *stable* if only one solution exists. Otherwise, if there is more than one solution, the system is said to be *unstable* [17].

3.3.3 Throughput and Delay

Let us now define the throughput $S(\mathbf{n})$ to be the conditional expectation of the number of correctly transmitted data packets in a slot, given that the system is in state $\mathbf{n}$. Then, it is clear that the throughput at an equilibrium point is expressed as

$$S(\overline{\mathbf{n}}) = g_Q(X(\overline{\mathbf{n}})) = g_Q(\rho W) \tag{16}$$

The average packet delay, which is the average time, in number of slots, from the moment the packet is generated until the instant the packet is correctly received by the destination, can be calculated from Little's result [14] to give

$$E[D] = \frac{N}{S(\overline{\mathbf{n}})} - \frac{1}{\sigma} \tag{17}$$

4 Numerical Results

In this section we use both the analytical model and simulation to investigate a specific system consisting of 500 stations interconnected through 5 wavelengths (W= 4 in this case). The propagation delay R is equal to 10 slots. In all figures, we note the excellent agreement between the analysis and simulation.

Figure 8 shows the throughput per data wavelength versus delay curve for $V = 10$, $p = 0.2$. and σ increasing from 0 to 0.2. Note that in the lower part of the curve the delay increases very slowly with the throughput; thus a high throughput per data channel and low delay can be achieved.

Figures 9 and 10 show the effect of varying V and W while keeping their sum constant by fixing the slot size. We can see that for different cases the maximum throughput always occurs at the point where $V \approx eW$. This is not surprising since the capacity of a slotted ALOHA channel is $1/e$. By making $V \approx eW$ the capacities of the V ALOHA channels and the W data channels are balanced. When $V \ll eW$, there is not enough throughput coming out of the ALOHA reservation channels to keep the W data channels busy. When $V \gg eW$, most stations are waiting (in the Q mode) for a wavelength on which to transmit a data packet. When $V \approx eW$, the throughput is roughly equal to the load offered by the stations.

Let D_T and D_Q denote the average time a station spends in modes T and Q, respectively, in a cycle. By Little's result, the throughput is equal to $N/(\frac{1}{\sigma} + D_T + R + D_Q + R)$. Setting both D_T and D_Q to 1, which is the minimum possible value, we obtain an upper bound for the throughput,

$$S_{UB} = \frac{N}{\frac{1}{\sigma} + 2(R+1)}$$

and a lower bound for the delay,

$$D_{MIN} = \frac{N}{S_{UB}} - \frac{1}{\sigma} = 2(R+1)$$

which are the "flat" region in Figures 9 and 10, respectively. Note that when $S_{UB} > \min(V/e, W)$ (the case of $V + W$=14), the flat region disappears since the throughput is not limited by the user-generated load.

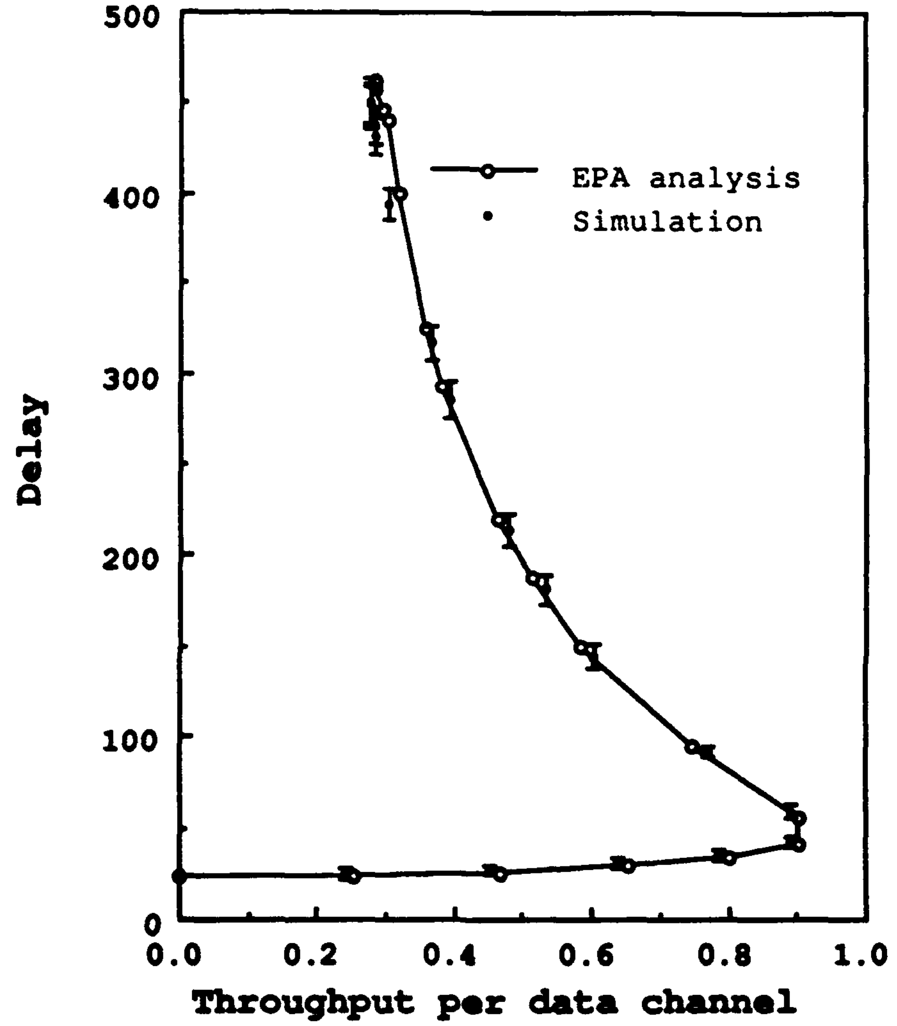


Figure 8: Throughput per data channel versus delay. N=500, W=4, V=10, R=10, p=0.2.

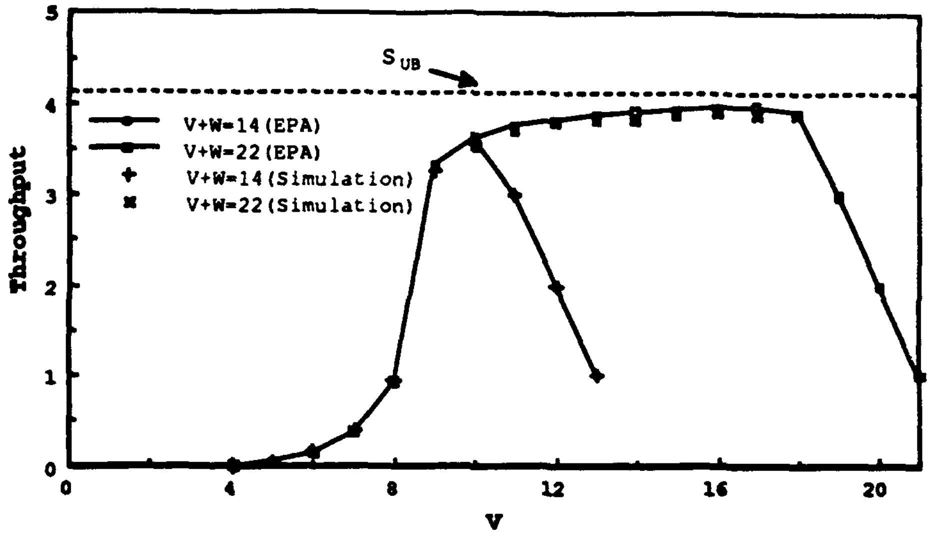


Figure 9: Throughput versus V for fixed slot sizes ($V + W$ constant). $N = 500$, R=10, σ=0.01, p=0.2.

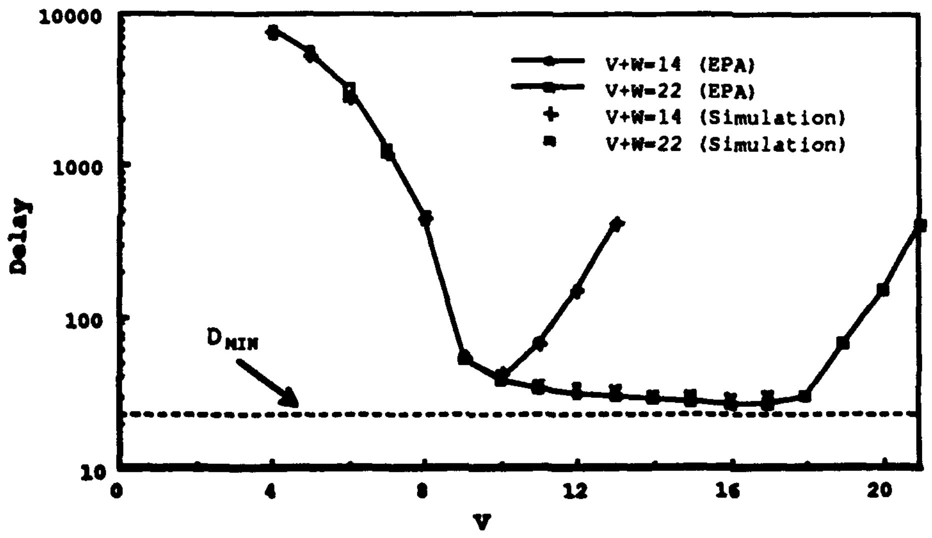


Figure 10: Delay versus V for fixed slot sizes ($V + W$ constant). $N = 500$, R=10, σ=0.01, p=0.2.

The effect of varying the propagation delay, R, is shown in Figures 11 and 12. Although the physical distance of the network is usually fixed, R can be varied by varying the packet (and thus the slot) size. We see that when R is small, too much traffic is offered to the V ALOHA reservation channel and the throughput is small. As we increase R, the throughput increases too. It reaches the maximum at $R = 7$, then falls off because there is not enough traffic in each slot when R becomes large. The tail of the throughput curve is bounded by the upper bound S_{UB}. The reason the increase is so sharp around $R = 6$ and 7 is that the system changes from an overloaded system (see Figure 7(d) in [17]) to a bistable one and then to a stable one as R increases from 5 to 7. This phenomenon is observed in both the EPA analysis and the simulation. (For the case $R = 6$, the average value of the two solutions obtained from EPA is plotted.)

Figure 13 shows the influence of the destination conflicts. Suppose i data packets are transmitted in a slot. The number of packets successfully received by their destinations is $g_Q(i)$. We plot the fraction of success, $g_Q(i)/i$, versus i assuming $N = 500$. Near-term technology limits W, the number of transmitter/receiver-tunable wavelengths available, to be fewer than about twenty; thus we see that destination conflicts are not a serious concern.

5 Conclusions

In this paper a wavelength division multiple access protocol (with W wavelength channels) was proposed to provide a high-capacity optical fiber local area network to a large population of N users. We assumed $N \geq W$. The users' traffic was assumed to be bursty as in the case of computer communications. The performance of the protocol was completely analyzed for both the infinite and finite population cases. The numerical results show that an aggregate throughput substantially larger than the electronic speed of a single station can be supported. The effects of various system parameters and their optimal selection were also investigated. By comparison with simulation, our analytical approximations were shown to be

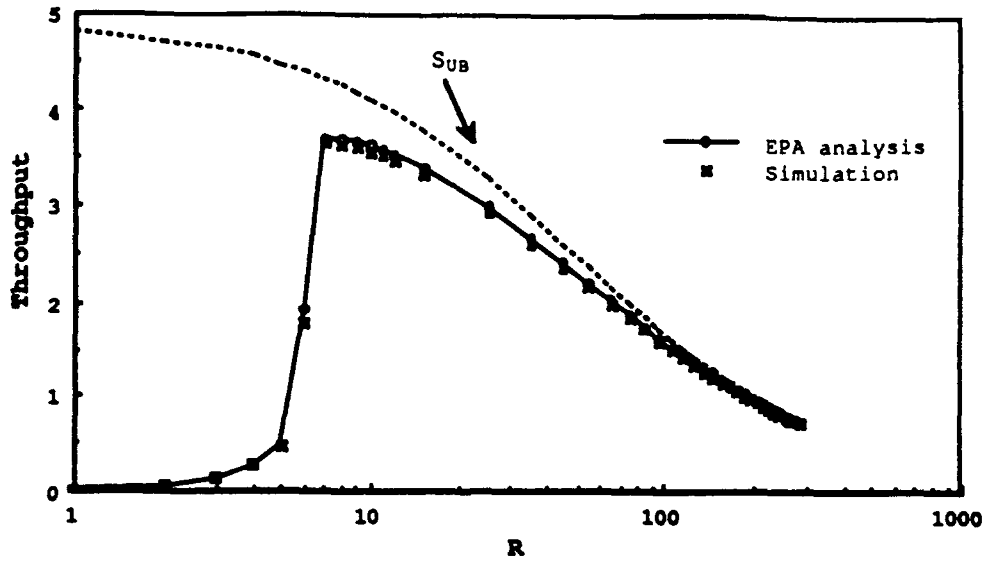


Figure 11: Throughput versus propagation delay. N = 500, W=4, V=10, σ=0.01, p=0.2.

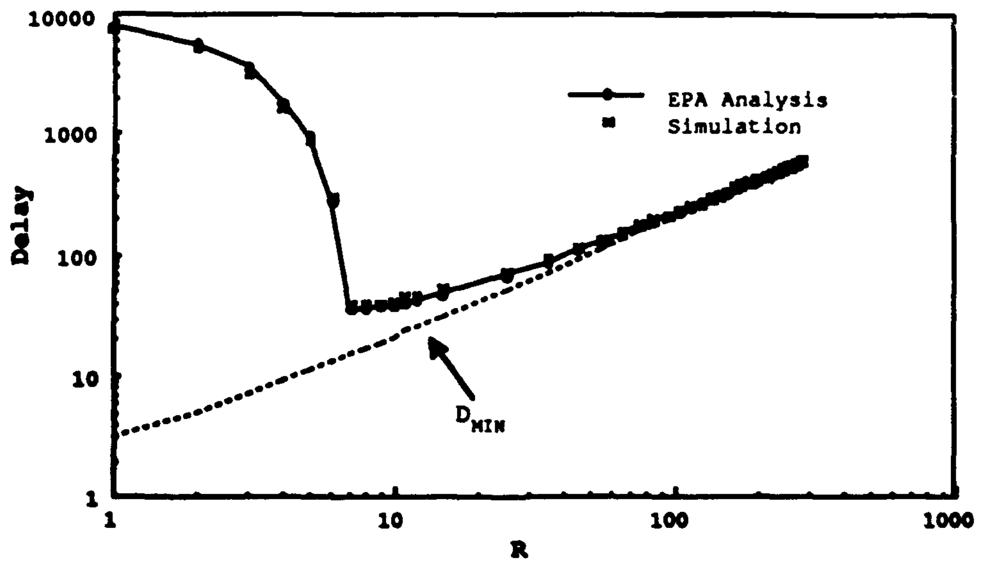


Figure 12: Delay versus propagation delay. N = 500, W=4, V=10, σ=0.01, p=0.2.

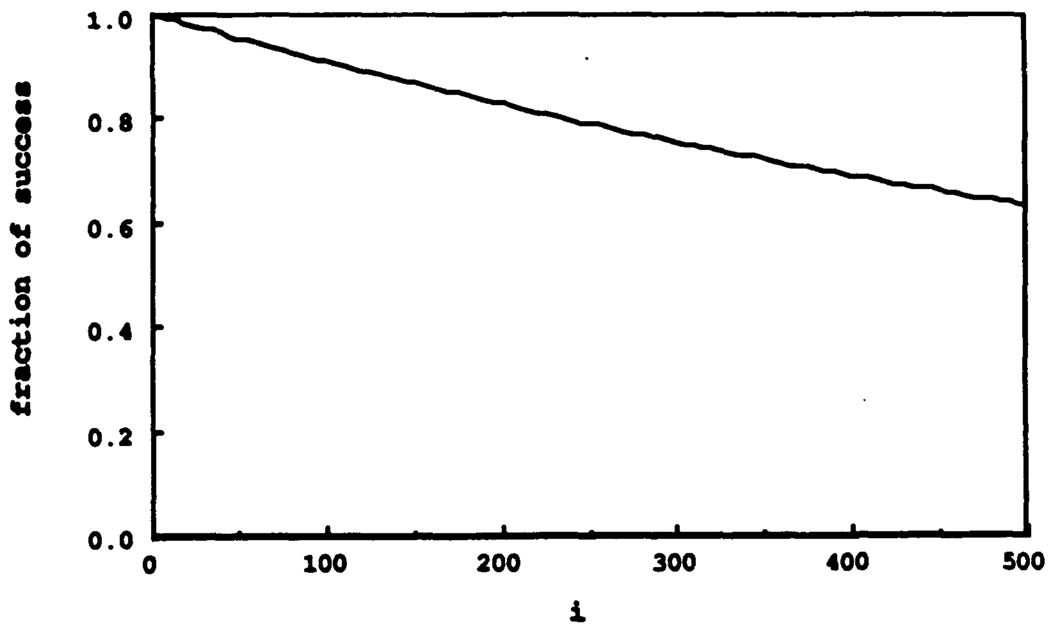


Figure 13: The fraction of success versus the number of data packets transmitted. $N = 500$.

excellent.

Appendix : Derivations of $f_T(\overline{\mathbf{n}})$ and $g_Q(i)$

Here we derive $f_T(\overline{\mathbf{n}})$, the average number of stations that successfully transmit a reservation minipacket in a slot, given that the system is in state $\overline{\mathbf{n}}$. Suppose that i reservation minipackets are transmitted in a slot. The probability that a particular one of them is successful is equal to $(1-1/V)^{i-1}$ and the average number of successful reservations is $i(1-1/V)^{i-1}$. Also, the conditional probability that i reservation minipackets are transmitted in a slot given that the system is in state $\overline{\mathbf{n}}$ is

$$\binom{\overline{n}_T}{i} p^i (1-p)^{\overline{n}_T - i}$$

Therefore, we have

$$\begin{aligned} f_T(\overline{\mathbf{n}}) &= \sum_{i=1}^{\overline{n}_T} i(1-\frac{1}{V})^{i-1} \binom{\overline{n}_T}{i} p^i (1-p)^{\overline{n}_T - i} \\ &= \overline{n}_T p (1-\frac{p}{V})^{\overline{n}_T - 1} \end{aligned}$$

Now we derive $g_Q(i)$, the average number of successfully transmitted data packets given that i data packets are transmitted, under the assumption that a station does not transmit to itself. Consider a destination station, say station k, for example. The probability that station k is the source of one of the i transmitted packets is i/N, and given this, the probability that none of the other $(i-1)$ packets are destined to station k is $(1-\frac{1}{N-1})^{i-1}$. The probability that none of the i packets is transmitted by station k is $(1-i/N)$, and given this the probability that none of the i packets are destined to station k is $(1-\frac{1}{N-1})^i$. Therefore, the probability that at least one among those i packets is going to station k (which is also equal to the average number of packets successfully received by station k) is equal to

$$1 - \left[\frac{i}{N}(1-\frac{1}{N-1})^{i-1} + (1-\frac{i}{N})(1-\frac{1}{N-1})^i\right]$$

Therefore, we have

$$\begin{aligned} g_Q(i) &= N\left[1-\left[\frac{i}{N}(1-\frac{1}{N-1})^{i-1}+(1-\frac{i}{N})(1-\frac{1}{N-1})^{i}\right]\right] \\ &= N\left[1-(1-\frac{1}{N-1})^{i-1}(\frac{N^2-2N+i}{N(N-1)})\right] \end{aligned}$$

References

[1] P. S. Henry, " High-capacity lightwave local area networks," *IEEE Commun. Mag.*, vol. 27, pp. 20 - 26, Oct. 1989.

[2] F. A. Tobagi, F. Borgonovo, and L. Fratta, " Expressnet: A high performance integrated-services local area network," *IEEE J. Select. Areas Commun.*, vol.1, no. 5, pp. 898 - 913, Nov. 1983.

[3] F. E. Ross, "FDDI - A tutorial," *IEEE Commun. Mag.*, vol. 24, May 1986.

[4] R. M. Newman, Z. L. Budrikis, and J. L. Hullet, "The QPSX MAN", *IEEE Commun. Mag.*, vol. 26, pp. 20 - 28, Apr. 1988.

[5] C. A. Brackett, "Dense wavelength division multiplexing networks: Principles and applications," *IEEE J. Select. Areas. Commun.*, vol. 8, no. 6, pp. 948 - 964, Aug. 1990.

[6] M. G. Hluchyj and M. J. Karol, "ShuffleNet : An application of generalized perfect shuffles to multihop lightwave networks," pp. 4B.4.1 - 4B.4.12, *Infocom '88*, 1988.

[7] I. Chlamtac and A. Ganz, "Toward alternative high-speed network concepts: The SWIFT architecture," *IEEE Trans. Commun.*, vol. COM-38, pp. 431 - 439, Apr. 1990.

[8] I. Chlamtac and A. Ganz, "Design alternatives of asynchronous WDM star networks," pp. 23.4.1 - 23.4.6, *ICC '89*, 1989.

[9] A. Ganz and Z. Koren, "WDM passive star - Protocols and performance analysis," pp. 9A.2.1 – 9A.2.10, *Infocom '91*, 1991.

[10] M. -S. Chen, N. R. Dono, and R, Ramaswami, "A media-access protocol for packet-switched wavelength division multiaccess metropolitan area networks," *IEEE J. Select. Areas. Commun.*, vol. 8, no. 6, pp. 1048 – 1057, Aug. 1990.

[11] L. G. Roberts, "Dynamic allocation of satellite capacity through packet reservation," in *AFIPS Conf. Proc.*, vol. 42, pp. 711 – 716, 1973.

[12] Y. Ofek and M. Sidi, "Design and analysis of a hybrid access control to an optical star using WDM," pp. 2A.3.1 – 2A.3.12, *Infocom '91*, 1991.

[13] L. Kleinrock, *Queueing Systems, Vol. II: Computer Applications,* John Wiley and Sons, New York, 1976.

[14] L. Kleinrock, *Queueing Systems, Vol. I: Theory,* John Wiley and Sons, New York, 1975.

[15] S. S. Lam, "Packet switching in a multi-access broadcast channel with applications to satellite communication in a computer network," Computer Science Department, School of Engineering and Applied Science, Engineering Report UCLA-ENG-7429, Mar. 1974.

[16] S. Tasaka and Y. Ishibashi, "A reservation protocol for satellite packet communication – A performance analysis and stability considerations," *IEEE Trans. Commun.*, vol. COM-32, pp. 920 – 927, Aug. 1984.

[17] L. Kleinrock and S. S. Lam, "Packet switching in a multiaccess broadcast channel: Performance evaluation," *IEEE Trans. Commun.*, vol. COM-23, pp. 410 – 423, Apr. 1975.