June 1990 UILU-ENG-90-2221

COORDINATED SCIENCE LABORATORY
College of Engineering

PROCEEDINGS SPRING 1990 NETWORK TOPICS COURSE

Bruce Hajek, Instructor

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-90-2221	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Ave. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

PROCEEDINGS SPRING 1990 NETWORK TOPICS COURSE

12. PERSONAL AUTHOR(S)

BRUCE HAJEK, INSTRUCTOR

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	June 1990	147

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Computer communication networks

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Collected here are papers prepared by the students of EE497: High Speed Computer Communication Networks in the 1990 Spring Semester at the University of Illinois at Urbana-Champaign. Some minor revisions to the papers were made by the students after the end of the semester. The course was a special topics course at the graduate level. The first three papers deal with some aspects of routing in packet switched networks in which routing decisions are made in a distributed fashion. The next paper considers dynamic routing in a circuit switched network. The fifth and sixth papers address two different queueing problems that arise in high speed networks. The final three papers cover optical codes, network resource allocation from the perspective of economic theory, and an existing telecommunication switch.

Bruce Hajek, Instructor

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

TABLE OF CONTENTS

Packet Routing on a Butterfly in Time O(log N) with Small Constants

Arvind Krishna

Submitted in partial fulfillment of the requirements for EE497BH, Spring 1990

Abstract

Ranade, in his 1987 FOCS paper, presented an algorithm for a probabilistic PRAM emulation on the Butterfly. He mentioned that this algorithm could be modified in a straight-forward manner to route permutations. We simplify the scheme to handle packet routing and modify the algorithm in a number of ways. Our results are for permutation routing and more importantly, for the general case of uniform traffic. Uniform traffic is the case where each node has 1 packet to be delivered and its destination is chosen randomly amongst all nodes in the network. Our main results are related to finding upper bounds on the probability that the routing time exceeds t for a fixed queue size. We show that if $t = \Omega(\log N)$, then the probability is less than $c\alpha^t$, where $c, \alpha < 1$.

1 Introduction

We study the problem of packet routing on the Butterfly network. Let N be the number of nodes in the network, and let us consider routing a permutation on a bounded degree network. The minimum possible time to evacuate the network for a typical permutation is $\log_d N$, where d is the degree of the switches at each node. We propose schemes that can route a permutation with high probability on a Butterfly in time $k \log N$ and maintain queues of size less than q at each node. The novelty of these schemes lie in the fact that k can be as low as 12, and q can also be as low as 12 to finish routing in time t with probability greater than $1 - \alpha^t$, where $\alpha < 1$. We believe these results to be the best to date for routing permutations on a network whose nodes have small degree. The schemes to be presented are modifications of the memory sharing scheme of Ranade[15].

A lot of effort has been expended on this problem over the past few decades. Consider routing a permutation from N sources to N destinations on a bounded degree network. Borodin and Hopcroft [4] showed that any deterministic oblivious scheme, in the worst case, takes $T = \Omega(\sqrt{N})$. On removing the constraint of oblivious routes, sorting networks can be used to route messages. This was first demonstrated by Batcher [3], whose scheme used $T = O(\log^2 N)$. Using the sorting network of Ajtai, Komlos, and Szemeredi [1] the routing time reduces to $O(\log N)$. We should point out that while Batcher's algorithm is easily implemented, the AKS algorithm is impractical because of the large constants involved. The best constants to date for the AKS sorting network are still in the 1000's, for an example see Cole et al.[5]. Another deterministic routing scheme that finishes in $T = O(\log N)$ was given by Upfal[17] utilizing a *Multi-Butterfly* network with $O(1)$ queues at each node. Unfortunately, the constants involved are still in the 1000's, at least if the degree of the nodes is constrained to be smaller than 10.

There has been much greater success in the development of efficient, *randomized* packet routing algorithms. One of the earliest works in this area was by Valiant and Brebner [18], who developed an algorithm for routing permutations in $T = O(\log N)$ on a $N-$node hypercube network. Their algorithm used queues of size $O(\log N)$ at each node and finished the routing in the given time with high probability (where high probability means with probability exceeding $1 - N^{-c}$, and c is a given positive constant). Similar results were obtained by Aleliunas [2] and Upfal [16] for Butterfly and Shuffle-Exchange networks. Pippenger [14] and Ranade [15] improved the results for the Butterfly by using only constant-size queues at each node. However, Pippenger's algorithm allows the possibility of deadlock, while Ranade's algorithm does not. Further, the analysis of Ranade's algorithm is simpler than that of Pippenger's algorithm. The constants involved in Ranade's algorithm are perhaps the best to date, requiring queue sizes to be in the 100's and the routing time to be $k \log N$, where k is in the 100's. Leighton et al. [12] generalized Ranade's ideas without restriction to any particular network.

In this paper, we examine Ranade's work in detail. Ranade, in his original paper [15], presented an algorithm for a probabilistic PRAM emulation on the Butterfly. He mentioned that this algorithm could be modified in a straight-forward manner to route permutations. Our main contribution is to modify the algorithm in a number of ways, so that we can make do with queue sizes in the 10's and routing times to be $k \log N$, where k can be as low as 12. This is achieved by simplifying the original algorithm and making extensive use of randomization. The simplification of the algorithm results mainly from the fact that packet routing does not need message combining and messages do not need to retrace their paths with the data requested from

memory locations.

In brief, the modifications we make to the algorithm are as follows. Firstly, only unidirectional links between nodes are required as opposed to bidirectional links required in the original algorithm. Secondly, a *universal hash function* is not needed. Instead, that is replaced by randomization, where the routing of each packet depends on two parameters, chosen independently of other packets. Thirdly, the priority assigned to packets is divorced from their random intermediate destinations. Fourthly, on fixing queue sizes, we show that the network evacuates with probability greater than $1 - \alpha^t$ in time less than $t = O(\log N)$, where $\alpha < 1$, whereas in [15] the queue sizes needed to increase to improve the probability. Finally, we also consider the case of *uniform traffic* and not just permutation routing. Uniform traffic is the case where packet destinations are chosen randomly amongst all nodes in the network.

A more detailed statement of the results will be given in Sections 4 and 5. The rest of this paper is organized as follows. Sections 2–4 deal with the problem of permutation routing. In Section 2, we describe the network model and the routing algorithm. The key idea behind showing that the routing cannot take a long time is that of *delay paths*, and this is discussed in Section 3, following which the analysis is done in Section 4. In section 5 we take up the problem of uniform traffic. Here, each node has a packet, the destination of each packet being picked equiprobably from all nodes in the network. The ideas of Sections 2–4 generalize to this case, and we shall show that the time required for routing is $O(\log N)$. We end with a brief discussion in Section 6.

2 Network Model and Operation

An N-node Butterfly, where $N = n2^n$, can be conceptually visualized as having n columns of 2^n nodes each. A node is identified with a pair (c, r), where $0 \leq c < n$ and $0 \leq r < 2^n$. For node $i = (c, r)$, we define $row(i) = r$ and $column(i) = c$. Node (c, r) is connected by links to nodes $(c+1, r)$ and $(c+1, r')$, where $c+1$ is taken modulo n, and the binary representation of r' differs from that of r in the c-th most significant bit.

We wish to route a set of packets where no two packets have a common source or destination. The network operates synchronously and time is slotted. Nodes initiate transmissions at the beginning of time slots and one time slot corresponds to the packet transmission time over one link. Each node has two input queues, one associated with each input link. These queues have a fixed size, independent of n. A node can start to transmit a packet over a link only if the input queue to which the packet is destined has room for the packet at the beginning of the time slot.

Let Π denote the permutation to be routed, where $\Pi(i)$ is the destination of the packet at node i. The path taken by the packet starting from source i leading to its destination $\Pi(i)$ is described by the following four phases:

Phase 1: The packet makes its way from the starting node $i = (c, r)$ to node $(0, r)$.

Phase 2: The packet makes its way from node $(0, r)$ to node $(0,\ d(i)\ )$. Here, $d(i)$ is a random number chosen by each packet independently of all other packets, and $0 \leq d(i) < 2^n$. The purpose of the random destination is to make heavy utilization of any link in the network very unlikely.

Phase 3: The packet makes its way from node $(0, d(i))$, to node $(0, row(\Pi(i)))$.

Phase 4: The packet makes its way from node $(0,\ row(\Pi(i)))$ to $(\ column(\Pi(i)),\ row(\Pi(i))\ )$. Node $\Pi(i) = (\ column(\Pi(i)),\ row(\Pi(i))\ )$ is the destination of the packet.

In each phase, the path taken by the packet is unique. The uniqueness of paths between nodes is a property of the Butterfly topology. As seen in Figure 1, each phase is a transversal of the Butterfly. We describe the algorithm in the logical network of Figure 1, where columns are numbered from 1 to $4n$, and column j corresponds to column (j mod n) in the real network. The four sections of the logical network correspond to the four phases of the algorithm. Columns $(i-1)n, \ldots, in-1$ correspond to phase i of the algorithm. This implies that in the real network each node has the function of two 2×2 switches, two 1×1 switches, and six input queues.

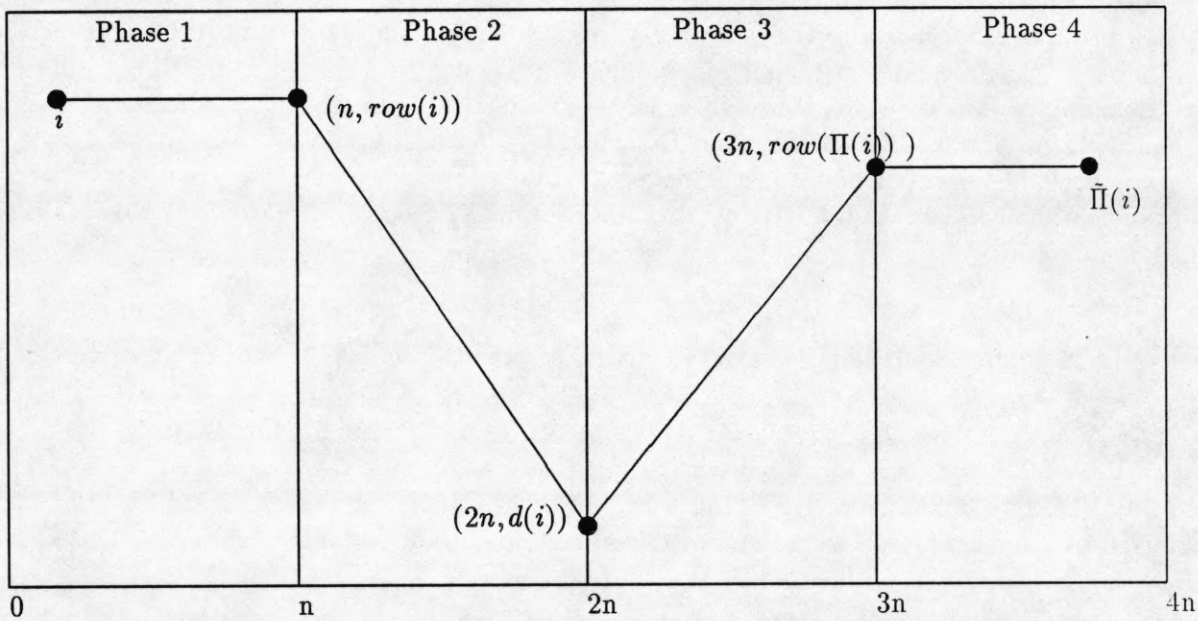


Figure 1: Logical Network, showing the path taken by the packet starting at node i to its destination $\tilde{\Pi}(i)$ (the node corresponding to $\Pi(i)$, namely $(column(\Pi(i)) + 3n,\ row(\Pi(i)))$).

Each node in the logical network has queues associated with each input link. We assume that each queue has size $q \geq 2$ for ease of later analysis. Once the random destinations, $d(i)$, have been chosen the path that each packet takes through the logical network is completely determined. The order in which packets use a given link is yet to be determined. This order is determined by priority tags, where $p(i)$ denotes the priority tag of the packet beginning at node i. Each packet chooses its priority tag independently of other packets and the random destinations, and each priority tag is chosen equiprobably from $0, 1, \ldots, M-1$, where M is a parameter to be specified later.

Packets are kept ordered in each queue by means of their priority tags. It will be shown later that packets arrive on each link in order of their priority tags. Thus, all queues can

be implemented as simple FIFO queues. Since priority tags were chosen independently from $0, 1, \ldots, M-1$ it is possible that two packets have the same priority tag. This problem is avoided by augmenting the priority tag of a packet starting at node i by its address, when necessary. Thus, if $p(i) = p(j)$ and $i < j$, then the priority tag of the packet starting at node j is larger than the priority tag of the packet starting at node i.

Definition 1 *The priority tag of a packet beginning at node i is specified by a lexicographic ordering on the pair $(p(i), i)$.*

There are three types of packets— EOS, MESSAGE, and GHOST. The difference between these three types will be made clear in the next few paragraphs. EOS is an acronym for End-Of-Stream, a packet of this type conveying the information that no more packets are to be received on that link. Packets of type MESSAGE are the real packets, the packets which have to be routed to their destinations. A GHOST serves the function of informing a node that all following packets on an input link will have larger priority tags than the GHOST.

We now give a detailed description of the algorithm at each node. The algorithm is synchronous, in the sense that all nodes begin the algorithm together at time $t = 0$.

Consider phase 1. At time $t = 0$, each node in the first column forwards its packet, if any, to the next column, and in the next time step forwards a packet of type EOS. A packet of type EOS has a priority tag of infinity. A node in the first column with no packet, forwards only a packet of type EOS. An EOS packet conveys the information that a node has no more packets to send.

Consider a node in column c , $0 < c \leq n-1$. It starts receiving packets at time $t = c - 1$. Suppose this node receives packets in order of their priority tags and then forwards the packets in that order, inserting its own packet in the correct position. Then, the next column, $c+1$, also receives packets in order of their priority tags. It follows, by induction, that all nodes receive packets in order of their priority tags. Note that a node in column c , $c > 0$, waits until it receives packets from the previous column and then starts its own operation. Moreover, in phase 1 each node uses only one input link and one output link.

In phases 2 and 3, each node examines the packets at the head of each input queue. The packet with the lower priority tag is selected for transmission on the appropriate link. An EOS or GHOST packet is transmitted on both links. If the selected packet is of type MESSAGE, then at the same time the node creates a GHOST packet (with the same priority tag as the selected packet) which is selected for transmission on the other link. A GHOST packet informs the receiving node that all further receptions on that link will have larger priority tags. For any packet selected for transmission we have two possibilities.

1. Receiving queue is not full. In this case the packet is transmitted.
2. Receiving queue is full. Then, the packet cannot be transmitted. If the packet has type EOS or MESSAGE, then it waits in the queue to be transmitted at some future time. If the packet has type GHOST it is removed from the network, because the next packet to be transmitted over that link will have a larger priority tag.

The only case when a packet is selected for transmission and still has to remain in the queue is when it is of type EOS or MESSAGE and the receiving queue on its preferred link(s) is full.

Moreover, when a node receives a GHOST packet which cannot be transmitted onward at the next time step, it removes the GHOST because it will receive some other packet with a larger priority tag at the next time step. As a result, the GHOST's never wait in queues. Also, GHOST's are not used after phase 3, so they can be removed from the network at column $3n$.

Lemma 1 *Every input queue in column c ,$n < c \leq 3n$, holds at least one packet at time c and at every subsequent time step until an EOS packet leaves the queue.*

Proof It is clear from the description of phase 1 that, from time n, each node in column n has packets in only one input queue. The packets in that input queue are in order of their priority tags. Starting at time n, each node in column $n+1$ receives a packet on each input queue unless the queue is full, or an EOS has been received. Hence the lemma is true for column $n+1$. An easy induction on column number completes the proof. □

It follows from the description of the algorithm that in phases 2 and 3, packets traverse each link in order of their priority tags. This implies that in column $3n$, the packets in each queue are ordered. In phase 4, each node in column c ,$c \geq 3n$, forwards packets on only one output link, in order of their priority tags. Thus, the creation of GHOST's is no longer necessary in this phase. Packets traverse a link in order of their priority tags in all four phases of the algorithm.

3 Delay paths

We begin this section by giving a rough introduction to the concept of *delay paths*. Roughly speaking, a delay path is a sequence of related nodes and packets. The positions of the nodes, and the priority tags of the packets have to obey certain relations. The idea behind these concepts is to construct an unlikely event, which is guaranteed to happen if the routing takes a long time to happen. An upper bound on the probability of this event will then provide an upper bound of the probability of routing taking a long time. These concepts will now be made more precise.

Definition 2 *A path S in the logical network is a sequence $\{S(i)\}$ of nodes with the property that there is a link between nodes $S(i)$ and $S(i+1)$, and the link can be oriented in either direction. A path starting in column $4n$ and ending in column 0 is called an input-output path.*

Notice that in the definition of a path, there only has to be a link in one of the two directions between nodes $S(i)$ and $S(i+1)$. Thus, input-output paths can be much longer than $4n$.

Definition 3 *Let P be a sequence of packets, $P = p_1, \ldots, p_\alpha$, such that the priority tag of p_i is larger than that of p_{i+1} for $1 \leq i < \alpha$. Let S be an input-output path such that p_i passes through $S(j_i)$ and $j_i \leq j_{i+1}$ for $1 \leq i < \alpha$. All the packets in P are required to be of type MESSAGE. Then, S is a delay path for P.*

We now define the *lag* of a packet, which denotes the amount of time a packet has spent waiting in queues.

Definition 4 *The lag of a packet p in node s, at time t, is defined to be $lag(p,t,s) = t - column(s)$. The lag is not defined if p is not in node s at time t.*

Note that the *lag* of a packet starting at node i is defined only from time $column(i)$ onwards. It is clear that the lag of every packet in the network is non-negative. Consider a packet p that has

to wait at a node s. The packet enters node s at some time t_E and leaves at time $t_L > t_E + 1$. There are only two possibilities which are enumerated below.

1. At time $t_L - 1$, another packet p' from the same node was transmitted. It is clear that $lag(p', t_L - 1, s) = lag(p, t_L, s) - 1$ and the priority tag of packet p' is smaller than the priority tag of packet p.

2. At time $t_L - 1$, p was selected for transmission to some node s', but could not be transmitted because the input queue corresponding to the link from node s to node s' was full. Since p was transmitted at time t_L, a packet p', at time $t_L - 1$, had to leave node s'. It follows that $lag(p', t_L - 1, s') = lag(p, t_L, s) - 2$ and the priority tag of packet p' is smaller than the priority tag of packet p. Packet p' is said to be a *q-delayer* for packet p.

The rest of this section is used to prove the following theorem.

Theorem 1 *Suppose that a given permutation Π requires time $4n + \delta$ to be routed. Let q be the size of each input queue in the logical network, and let Φ be any positive integer. Then, there exists a sequence of packets P with $|P| \geq min(\delta, q\Phi)$, and an input-output path S of length $4n + 2\Phi$, such that S is a delay path for P,*

Proof We begin by constructing a path S through the logical network and a sequence of packets P such that S is a delay path for P. Then, we modify S and P to have the required sizes.

Consider an EOS packet, p_0, which arrives in node σ_0 in column $4n$ at time $t_0 = 4n + \delta$. This implies that $lag(p_0, t_0, \sigma_0) = \delta$. Follow p_0 back in time until the last time it waited in a queue, say in node σ'_0 at time t'_0, that is $lag(p_0, t'_0, \sigma'_0) = \delta - 1$. Then, Lemma 1 implies that there was a packet p'_0 which delayed p_0 at time t'_0 with the priority tag of packet p'_0 is smaller than the priority tag of packet p_0.

- If p'_0 is not a GHOST (i.e. p'_0 is a MESSAGE) then set $p_1 = p'_0, t_1 = t'_0$ and σ_1 to be the switch in which p_1 is at time t_1. If p'_0 is a q-delayer, then $column(\sigma_1) = column(\sigma'_0) + 1$, else $\sigma_1 = \sigma'_0$. If p_1 was a q-delayer then $lag(p_1, t_1, \sigma_1) = lag(p_0, t_0, \sigma_0) - 2$, else $lag(p_1, t_1, \sigma_1) = lag(p_0, t_0, \sigma_0) - 1$.

- If p'_0 is a GHOST (in this case p'_0 cannot be a q-delayer), follow it backwards starting at time t'_0 until we reach a node where it was created. If its creator was a GHOST, follow this GHOST backwards until we reach a MESSAGE p_1 in switch σ_1 at time t_1 with the same priority tag as p'_0. Since GHOST's never wait, $lag(p_1, t_1, \sigma_1) = lag(p'_0, t'_0, \sigma'_0) = lag(p_0, t_0, \sigma_0) - 1$.

In a similar manner, given (p_i, t_i, σ_i) with $lag(p_i, t_i, \sigma_i) > 0$ we can construct another MESSAGE $(p_{i+1}, t_{i+1}, \sigma_{i+1})$ with the priority tag of packet p_{i+1} smaller than the priority tag of packet p_i and $0 \leq lag(p_{i+1}, t_{i+1}, \sigma_{i+1}) < lag(p_i, t_i, \sigma_i)$. Eventually, we reach the packet p_L at time t_L and switch σ_L such that $lag(p_L, t_L, \sigma_L) = 0$. By construction, the sequence of packets $P = p_1, \ldots, p_L$ are MESSAGES. $\sigma_1, \ldots, \sigma_L$ can be made into a path S by including the nodes through which the packets creating the sequence P passed through.

Suppose that in this path S, $column(s_i) = column(s_{i-1}) + 1$. Then, there is a packet $p_j \in P$ such that p_j passed through s_i and was a q-delayer for p_{j-1}. It is easy to see that the path S

constructed above has length $\leq 4n + 2\kappa_L$, where κ_L are the number of q-delayers in the sequence P. Let κ_i be the number of q-delayers in the sequence $p_0, p_1, \ldots, p_i \subset P$. From the construction above it follows that $lag(p_{i+1}, t_{i+1}, \sigma_{i+1}) = lag(p_i, t_i, \sigma_i) - 1 - (\kappa_{i+1} - \kappa_i)$. Then, summing both sides of the previous equation for $i = 0, \ldots, L-1$, we get $lag(p_L, t_L, \sigma_L) = lag(p_0, t_0, \sigma_0) - L - \kappa_L$. Hence,

$$0 = \delta - L - \kappa_L \Rightarrow L = \delta - \kappa_L. \tag{1}$$

Suppose packet p_{i+1} was a q-delayer for p_i. Then p_{i+1} leaves σ_{i+1} at time t_{i+1}, and the queue of p_{i+1} was full at time $t_{i+1} - 1$. Since p_i is a MESSAGE, there are $q - 1$ other MESSAGES, $m_1, \ldots, m_{q-1}$, in the same queue as p_{i+1}. Let the priority tag of MESSAGE m_i be denoted by μ_i. Then,

$$priority\ tag(p_i) > \mu_1 > \mu_2 > \ldots > \mu_{q-1} > priority\ tag(p_{i+1}).$$

Thus, on adding $m_1, \ldots, m_{q-1}$ to P, S is still a delay path for the *augmented* P.

- If $\kappa_L \leq \Phi$, we can add $(q-1)\kappa_L$ packets to P (as shown above), and we can easily extend S to an input-output path of length $4n + 2\Phi$. In this case, using equation (1), the number of packets in the sequence is

$$L + (q-1)\kappa_L = \delta - \kappa_L + (q-1)\kappa_L = \delta + (q-2)\kappa_L \geq \delta.$$

- If $\kappa_L > \Phi$, consider $p_1, \ldots, p_k$ where $\kappa_k = \Phi$. Note that $k \geq \Phi$. Then, adding packets for the q-delayers in $p_1, \ldots, p_k$ (as shown above) to $p_1, \ldots, p_k$ we get a packet sequence of length $(q-1)\Phi + k \geq q\Phi$. Let $S(j_i)$ be the switch where p_i meets S. Then, it is clear that $S(j_1), \ldots, S(j_k)$ can be extended into an input-output path of length $4n + 2\Phi$ (because there are only Φ links corresponding to the Φ q-delayers). □

4 Analysis

The randomization in the algorithm consists of the choice of intermediate destinations $(0, d(i))$ and the priority tags chosen by the packet starting at node i. The permutation Π that is being routed is fixed. Letting $\delta = q\Phi$, Theorem 1 implies that the probability of the routing taking longer than $q\Phi$ time units is less than the probability that there is an input-output path, S, of length $4n + 2\Phi$ which is a delay path for a sequence of $q\Phi$ packets. From this sequence of $q\Phi$ packets, there are $\binom{q\Phi}{\chi}$ ways to choose a subsequence, P, of χ packets such that the path S is a delay path for P.

Definition 5 *(a) Let Υ_1 be the probability that the routing takes longer than $4n + q\Phi$ time units.*

(b) Let Υ_2 be the number of distinct choices of intermediate destinations and priority tags such that there is an input-output path of length $4n + 2\Phi$ which is a delay path for some sequence of $q\Phi$ packets.

(c) Let Υ_3 be the number of distinct choices of sequences of χ packets, input-output paths of length $4n + 2\Phi$, intermediate destinations, and priority tags such that the path is a delay path for the sequence of χ packets.

Then, the following inequalities will provide an upper bound on Υ_1, the probability of interest.

$$\Upsilon_1 (M2^n)^{n2^n} \leq \Upsilon_2 \text{ and } \Upsilon_2 \binom{q\Phi}{\chi} \leq \Upsilon_3 \tag{2}$$

We begin by computing bounds on the number of ways to choose bn objects out of an objects. Stirling's formula [6] states that $n! = (\frac{n}{e})^n \sqrt{2\pi n}\alpha_n$, where $e^{\frac{1}{12n+1}} \leq \alpha_n \leq e^{\frac{1}{12n}}$. It follows that

$$e^{-\frac{1}{6bn}} \sqrt{\frac{a}{2\pi b(a-b)n}} \left(\frac{a}{b}\right)^{bn} \left(\frac{a}{a-b}\right)^{(a-b)n} \leq \binom{an}{bn} \leq \sqrt{\frac{a}{2\pi b(a-b)n}} \left(\frac{a}{b}\right)^{bn} \left(\frac{a}{a-b}\right)^{(a-b)n} \tag{3}$$

We now compute, for a given permutation Π, an upper bound for Υ_3. This is done by straightforward enumeration, upper bounding quantities which are hard to estimate. This enumeration is divided into 4 steps.

(1) Count the number of input-output paths of length $4n + 2\Phi$. Each path has to end in one of the 2^n nodes in column $4n$. There are exactly Φ backwards links which cannot be in the last n links of the path. Each link, backward or forward, can be chosen in two ways. There are at most $2^n 2^{2n+2\Phi} \binom{3n+2\Phi}{\Phi}$ input-output paths of length $4n + 2\Phi$.

(2) Once the path is fixed, the nodes where the χ packets meet the path can be chosen in at most $\binom{4n+2\Phi+\chi}{\chi}$ ways.

(3) Count the ways in which the MESSAGE packets, intermediate destinations, and priority tags associated with the above path can be chosen.

- If a packet passes through a given node in phase 1 or 2, there are 2^n ways in which to choose together, the originating node's $row(i)$ and $d(i)$, of a packet. There are at most n ways in which $column(i)$ can be chosen.
- If a packet passes through a given node in phase 3 or 4, there are 2^n ways in which to choose together the destination node's $row(\pi(i))$ and $d(i)$, of a packet. There are at most n ways in which the column, $column(\pi(i))$ can be chosen.

Note that if i is fixed, then $\pi(i)$ is fixed, and vice-versa. Since the priority tags of the packets in the sequence are ordered, the priority tags $p(i)$ can be chosen in at most $\binom{M+\chi}{\chi}$ ways. Thus, the total number of ways to choose the sequence of χ packets, and values of $d(i)$ and $p(i)$ for such packets, if the nodes where the packets meet the path are fixed, is given by

$$(2^n)^\chi n^\chi \binom{M+\chi}{\chi}$$

(4) So far, the intermediate destinations and priority tags of only χ packets have been fixed. These can be chosen for the remaining packets in $(2^n M)^{n2^n - \chi}$ ways.

Hence, an upper bound on Υ_3 is given by

$$\Upsilon_3 \leq 2^{3n+2\Phi} \binom{3n+2\Phi}{\Phi} \binom{4n+2\Phi+\chi}{\chi} n^\chi \binom{M+\chi}{\chi} (2^n M)^{n2^n} M^{-\chi} \tag{4}$$

Using equation (2), it follows that $\Upsilon_1 (M2^n)^{n2^n} \leq \Upsilon_3 \binom{q\Phi}{\chi}^{-1}$. Let $\Phi = \phi n$, $M = mn$, and $\chi = xn$. Then, substituting equation (3) in the equations above

$$\Upsilon_1 \leq \frac{e^{\frac{1}{6qn}}}{2\pi n}\sqrt{\frac{(3+2\phi)}{\phi(3+\phi)}\frac{(4+2\phi+x)}{x(4+2\phi)}\frac{(m+x)}{mx}\frac{x(q\phi-x)}{q\phi}}\left[2^{\frac{3+2\phi}{x}}\left(\frac{3+2\phi}{\phi}\right)^{\phi/x}\left(\frac{3+2\phi}{3+\phi}\right)^{(3+\phi)/x}\right.$$
$$\left.\frac{4+2\phi+x}{x}\left(\frac{4+2\phi+x}{4+2\phi}\right)^{(4+2\phi)/x}\frac{m+x}{x}\left(\frac{m+x}{m}\right)^{m/x}\frac{1}{m}\frac{x}{q\phi}\left(\frac{q\phi-x}{q\phi}\right)^{(q\phi-x)/x}\right]^{xn} \quad (5)$$

Before proceeding with an asymptotic analysis, we give some examples of how equation (5) provides a bound on Υ_1, the probability of the routing time exceeding a time t.

Example 1 *Suppose that all queue sizes in the logical network are 4, $M = 256n$, and we want the probability of the routing finishing in time $24n$. Set $m = 256, \phi = 5, x = 18$, and $q = 4$. Then, numerical substitution in equation (5) gives $\Upsilon_1 \leq \frac{0.0411}{n}2^{-0.4433n}$.*

Example 2 *Suppose that all queue sizes in the logical network are 12, $M = 256n$, and we want the probability of the routing finishing in time $16n$. Set $m = 256, \phi = 1, x = 10$, and $q = 12$. Then, numerical substitution in equation (5) gives $\Upsilon_1 \leq \frac{0.0383}{n}2^{-2.43383n}$.*

Example 3 *As a final example, suppose that all queue sizes in the logical network are 6, $M = 256n$, and we want the probability of the routing finishing in time $64n$. Set $m = 256, \phi = 10, x = 50$, and $q = 6$. Then, numerical substitution in equation (5) gives $\Upsilon_1 \leq \frac{0.00742}{n}2^{-129n}$.*

Asymptotic Analysis

We begin with a technical fact, that is used heavily in the proof of the following theorem.

Fact 1 *$(1+1/x)^x$ is an increasing function of x, for $x \geq 0$.*

Theorem 2 *Let the number of priority tags be $256n$. Then, given any queue size, $q \geq 8$, the probability that the time to route any permutation exceeds t, $t \geq \max(28n, 2qn)$, is less than $(2\Pi n)^{-1.5}\, 2^{-(t-4n)(1.46-3.02/q)}$.*

Proof Let $\chi = q\Phi = q\phi n$. The conditions of the theorem correspond to $m = 256$, $q \geq 8$, $\phi \geq 2$, and $q\phi \geq 24$. It follows from equations (2),(3), and (4) that a bound on Υ_1 is given by

$$\Upsilon_1 \leq (2\pi n)^{-1.5}\sqrt{\frac{(3+2\phi)}{\phi(3+\phi)}\frac{(4+2\phi+q\phi)}{q\phi(4+2\phi)}\frac{(m+q\phi)}{mq\phi}}\left[2^{\frac{3+2\phi}{q\phi}}\left(\frac{3+2\phi}{\phi}\right)^{1/q}\left(\frac{3+2\phi}{3+\phi}\right)^{(3+\phi)/q\phi}\right.$$
$$\left.\frac{4+2\phi+q\phi}{q\phi}\left(1+\frac{q\phi}{4+2\phi}\right)^{(4+2\phi)/q\phi}\left(1+\frac{q\phi}{m}\right)^{m/q\phi}\left(\frac{1}{m}+\frac{1}{q\phi}\right)\right]^{q\phi n} \quad (6)$$

We shall first bound terms inside the square brackets.

Using the fact that $q \geq 8$, and $q\phi \geq 24$, it follows that $2^{(3+2\phi)/q\phi} \leq 2^{3/24+2/8} \leq 1.297$. Next, $((3+2\phi)/\phi)^{1/q} \leq 3.5^{1/q}$ and $(4+2\phi+q\phi)/q\phi \leq 1.417$.

Since $(3+\phi)/\phi \leq 2.5$, it follows from Fact 1 that $(1+\phi/(3+\phi))^{(3+\phi)/q\phi} \leq 2.32^{1/q}$. Also, since $(4+2\phi)/(q\phi) \leq 4/24 + 2/q \leq .417$, Fact 1 implies that $(1+q\phi/(4+2\phi))^{(4+2\phi)/q\phi} \leq 1.666$. Since $m/q\phi \leq 256/24$, Fact 1 implies that $(1+q\phi/m)^{m/q\phi} \leq 2.601$.

Since $m = 256$ and $q\phi \geq 24$, it follows that $1/m + 1/q\phi \leq 0.0456$. Finally, it is easy to verify that the term inside the square root sign is less than 1. Thus, putting all of the above together

$$\Upsilon_1 \leq (2\Pi n)^{-1.5} \left[(1.297)3.5^{1/q}2.32^{1/q}(1.417)1.666(2.601)0.0456\right]^{q\phi n} = (2\Pi n)^{-1.5}\, 2^{-q\phi n(1.46-3.02/q)}$$

Setting the routing time to be $4n + q\phi n$, the theorem follows. □

The choice of the parameters in the statement of Theorem 2 are arbitrary and were chosen to give an exponentially decaying probability with time. If there is a specific time or queue size in mind, the parameters can be chosen to give a much better bound on the probability as shown in examples 1 — 3.

5 Uniform Traffic

5.1 Many-One Routing

We consider the problem of routing on a N-node Butterfly, $N = n2^n$, where each node has 1 packet to send. The destination of each packet is picked equiprobably from amongst the N nodes and independently of all the other packet destinations. The minimum expected time for routing such a configuration is n, because a typical packet is more than n links from its destination. We shall show that the routing finishes with high probability in time $O(n)$. Unlike the case of permutation routing, we shall not derive the tight bounds of examples 1–3, though a similar procedure can be carried out for this case as well. The bounds of this section are restricted to an asymptotic analysis for the purposes of brevity.

The ideal result to prove would be that if no node has to receive more than $O(n)$ packets, then the routing would finish in time $O(n)$ with high probability, where the probability refers to the randomization in the algorithm and not to the packet destinations. Permutation routing takes care of the case when each node has to receive exactly 1 packet. However, our result is slightly weaker than the ideal case. The network is partitioned into into 2^n sets of n nodes. We show that if none of the sets of n nodes has to receive more than $O(n)$ packets, then the routing finishes in time $O(n)$ with high probability. This allows $\Omega(n)$ packets to be received by 1 node, but not by more than $O(1)$ nodes in any of the sets of n nodes.

The algorithm and delay path described in Sections 2 and 3 apply to this case. The only place that the permutation Π came into use was in step (3) of the enumeration of Υ_3 in Section 4. Further, it is clear that Theorem 1 holds with the appropriate modification that the permutation Π is replaced by the set of packets to be routed.

Lemma 2 *Suppose there are $n2^n$ packets, each packet choosing one of the $n2^n$ destinations on a Butterfly equiprobably. For a fixed row r, let Y_r be the number of packets that choose any one of the destinations $(0,r),(1,r),\ldots,(n-1,r)$. Then, for $k \geq 2e$*

$$\Pr(\max_r Y_r < kn) \geq 1 - 2^{-(k-1)n}$$

Proof The union bound implies that $\Pr(\max_r Y_r \geq kn) \leq 2^n \Pr(Y_r \geq kn)$. Also,

$$\left(\frac{a}{b}\right)^{bn} \leq \binom{an}{bn} \leq \left(\frac{ae}{b}\right)^{bn}. \tag{7}$$

Since Y_r is the sum of $n2^n$ Bernoulli random variables, the union bound and equation (7) together imply that

$$\Pr(Y_r \geq kn) \leq \binom{n2^n}{kn} 2^{-kn^2} \leq \left(\frac{en2^n}{kn}\right)^{kn} 2^{-kn^2} = 2^{-nk\log_2 \frac{k}{e}} \leq 2^{-nk}$$

This proves the lemma □

The main implication of this lemma is that it is sufficient to find a bound on the probability for routing in time t for the case when each node has 1 packet to send, and no *row* of nodes has to receive more than kn packets. The case when any row of nodes has to receive more than kn packets has probability less than $2^{-(k-1)n}$.

Υ_1, Υ_2, and Υ_3 are defined as before. Then, equation (2) still holds.

Theorem 3 *Let the number of priority tags be $256kn$, $k \geq 2e$. For $\left(1 - 2^{-(k-1)n}\right) N^N$ out of the possible N^N configurations to be routed the probability that the time to route exceeds t, $t \geq \max(24kn + 4n, 2qn)$, is less than $(2\Pi n)^{-1.5}\, 2^{-(t-4n)(1.46-3.02/q)}$. given any queue size $q \geq 8$ in the logical network.*

Proof Consider the case when no row of nodes has more than kn packets destined for it. By Lemma 2, this happens in $\left(1 - 2^{-(k-1)n}\right) N^N$ out of the possible N^N configurations. All of the analysis done in the enumeration for the bound on Υ_3 carries over to this case, except for one exception. This one exception lies in part (3) of the enumeration. There, a further term of k has to be factored in for each of the χ packets, because

- knowing the destination row means that the packet could have come from any one of kn sources.
- knowing the source of a packet uniquely specifies the packet, and $k \geq 1$.

The bound on Υ_3 of equation (4) gets modified by multiplying the right hand side by k^{χ}.

Let $\Phi = \phi n$, $M = mkn$, and $\chi = xn = q\phi n$. Then, a slightly modified proof of Theorem 2 supplies the result.

□

5.2 Many-Many Routing

We consider the problem of routing on a N-node Butterfly, $N = n2^n$, where each node has as many as cn packets. The destination of each packet is picked equiprobably from amongst the N nodes and independently of all the other packet destinations. The minimum possible time for routing such a configuration is cn^2, because a typical packet is more than n links from its destination. We shall show that the routing finishes with high probability in time $O(n^2)$.

The purpose of considering this case is to make a start at establishing a steady state theory for packet routing on bounded degree networks. The known results for steady state delay on

self-routing bounded degree networks are not rigorous, at least for fixed length packets. Some progress has been made under some special traffic conditions [13,11], but the problem is still open. The results to be presented in this section are very similar in spirit to the h-relations described in [13]. However, unlike the extended Omega network, the routing network required here is of the same size as the number of inputs and outputs. The extended Omega network requires $\Omega(N \log N)$ switching nodes for N inputs and outputs.

The algorithm described in Section 2 generalizes to this case quite easily. Let intermediate destinations and priority tags be picked at random, like before. The only modification required is that in phase 1 of the algorithm more than one packet will be inserted by a node into the network. Further, it is clear that Theorem 1 holds with the appropriate modification that the permutation Π is replaced by the set of packets to be routed.

As a first step in the analysis, we establish the following lemma.

Lemma 3 *Suppose there are $cn^2 2^n$ packets, each packet choosing one of $n2^n$ destinations equiprobably. Let Y_i be the number of packets that choose destination i. Then, for $k \geq 2ce$*

$$\Pr(\max_i Y_i < kn) \geq 1 - 2^{-(k-2)n}.$$

Proof The union bound implies that $\Pr(\max_i Y_i \geq kn) \leq n2^n \Pr(Y_i \geq kn)$. Since Y_i is the sum of $cn^2 2^n$ Bernoulli random variables, the union bound and equation (7) together imply that

$$\Pr(Y_r \geq kn) \leq \binom{cn^2 2^n}{kn} (n2^n)^{-kn} \leq \left(\frac{ecn^2 2^n}{kn}\right)^{kn} 2^{-kn^2} n^{-kn} = 2^{-nk \log_2 \frac{k}{ce}} \leq 2^{-nk}$$

This proves the lemma □

The main implication of this lemma is that it is sufficient to find a bound on the probability for routing in time t for the case when each node has cn packets to send, and no node has to receive more than kn packets. The case when any node has to receive more than kn packets has probability less than $2^{-(k-2)n}$. We shall now concentrate on this case only.

A form of equation (2) still holds. All of the analysis done in the enumeration for the bound on Υ_3 carries over to this case, except for two exceptions. One exception lies in part (3) of the enumeration. There, a further term of kn has to be factored in for each of the χ packets, because

- knowing the destination means that the packet could have come from any one of kn sources.
- knowing the source means that the packet could be any one of cn packets, and $k > c$.

The second exception is in part (4) of the enumeration, because the number of packets left over is larger. The bound on Υ_3 gets modified as follows.

$$\Upsilon_3 \leq 2^{3n+2\Phi} \binom{3n+2\Phi}{\Phi} \binom{4n+2\Phi+\chi}{\chi} \left(kn^2\right)^{\chi} \binom{M+\chi}{\chi} (2^n M)^{cn^2 2^n} M^{-\chi} \qquad (8)$$

Let $\Phi = \phi n^2$, $M = mkn^2$, and $\chi = xn^2$. With this normalization, the bound for Υ_1 reduces to a form very similar to equation (6), and leads to the following theorem.

Theorem 4 *Let the number of priority tags be* $256kn^2$, $k \geq 2ce$. *For* $\left(1 - 2^{-(k-2)n}\right)(cn^2N)^N$ *out of the possible* $(cn^2N)^N$ *configurations to be routed the probability that the time to route exceeds* t, $t \geq \max(24kn^2 + 4n, 2qn)$, *is less than* $(2\Pi n)^{-1.5}\, 2^{-(t-4n)(1.46-3.02/q)}$, *given any queue size* $q \geq 8$.

6 Conclusion

The analyses in Sections 4 and 5 showed that as the number of values, M, for priority tags increased, the lower bounds on the probability of routing finishing in a given time increased. This seems to indicate that the performance of the algorithm would actually improve if the number of priority tags increase. However, simulations do not support this hypothesis. We performed simulations for permutation routing on various network sizes, from 256 nodes to 10240 nodes. Our main observations were that the routing time was not significantly affected on varying M from 2 to 2000000. Another observation was that the variance of the routing time was extremely small. A simulation of the basic algorithm with no priority tags, but still using FCFS queues and random conflict resolution was done. It was found that this algorithm performed better than the algorithm proposed in this paper. All these observations bear further investigation.

The intuition behind the routing completing in time $O(\log N)$ for the uniform traffic case in Section 5.1 is that an averaging effect is taking place. The algorithm described in this paper routes packets to the correct row ignoring the particular node in the row to which the packet is destined. Thus, in some sense, the algorithm is behaving like running $\log N$ versions simultaneously. This is achieved by the packet paths having $3 \log N$ links on the average. It is still an open question whether there is an algorithm which can come closer to having optimal length paths, which have $1.5 \log N$ links on the average for the Butterfly network. An even bigger open question is whether there is a *local* routing algorithm for a Shuffle-Exchange network, or any other optimal diameter bounded degree network, that can complete routing with high probability in $O(\log N)$ time with bounded queue sizes.

One point which many readers may question is the assumption that packets take unit time to be transmitted over a link. After all, packet lengths are $\Omega(\log N)$. On letting the packet transmission time over one link be proportional to $\log N$, the routing time goes to $O((\log N)^2)$. However, for the uniform traffic case, we expect that some node has to receive $\Omega(\log N)$ packets. This implies that for uniform traffic, if all links are bit-serial, that the time has to be $\Omega((\log N)^2)$, since there are only 2 links entering any given node.

The previous paragraph raises another related question. One of the schemes that has been proposed for packet routing is based on the Batcher-Banyan [3,9]. This scheme can be pipelined so that the operation is actually bit-serial and that the total time is $O((\log N)^2)$, assuming that one bit takes unit time to be transmitted over one link. The question naturally arises whether a similar pipelining effect can be used for the scheme presented in this paper and in [15]. There does not seem to be any way to pipeline this scheme and still preserve the guaranteed bounds on the performance. This is because $\Omega((\log N)^2)$ bits have to be transmitted over some links in the logical network. It is worth investigating if some other schemes can offer better performance assuming that 1 bit takes unit time to be transmitted.

The next point to be made concerns steady state operation. Suppose that a version of the

algorithm is used to route packets in a continuous fashion, i.e. nodes inject packets at random intervals into the network and time stamps or some other device are used. A question which arises is how delays behave in such a scenario. Various authors have studied this problem in a variety of contexts, and have met with only partial success. Most of the analysis in the literature is based on some kind of independence assumptions [8,7,10,11,13]. However, simulation results match the approximate analysis extremely well, usually within a few percent. A rigorous theory for the performance of packet routing networks in steady state is sorely needed.

Acknowledgements
The author wishes to thank Bruce Hajek and Andrea Pietracaprina for many helpful discussions. They will be the co-authors once this course is over and the paper is improved. (We hope!)

References

[1] M. Ajtai, J. Komlos, and E. Szemeredi. An $O(N \log N)$ sorting network. In *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing*, pages 1–9. ACM, 1983.

[2] R. Aleliunas. Randomized parallel communication. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 60–72. ACM, August 1982.

[3] K. Batcher. Sorting networks and their applications. In *Proceedings AFIPS Spring Joint Comput. Conf.*, volume 32, pages 307–314, 1968.

[4] A. Borodin and J. E. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and Systems Sciences*, 30:130–145, 1985. An earlier version of this paper was presented at the 14th Annual ACM Symposium on the Theory of Computing.

[5] R. Cole and C. O. Dunlaing. Note on the AKS sorting network. Technical report, Computer Science Department, New York University, 1988. Ultracomputer Note #109.

[6] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume one. John Wiley & Sons, New York, 3rd edition, 1968.

[7] A. Greenberg and B. Hajek. Approximate analysis of deflection routing in hypercube networks. Submitted to IEEE Trans. on Communications. Also, presented at the TIMS meeting, Osaka, July 1989, August 1989.

[8] A. G. Greenberg and J. Goodman. Sharp approximate models of adaptive routing in mesh networks. In O.J. Boxma, J. W. Cohen, and H. C. Tijms, editors, *Teletraffic Analysis and Computer Performance Evaluation*, pages 255–270. Elsevier, Amsterdam, 1986. revised, 1988.

[9] A. Huang and S. Knauer. Starlite: a wideband digital switch. In *Proceedings of Globecom Conference*, pages 121–125. IEEE Press, 1984.

[10] A. Krishna and B. Hajek. Performance of shuffle-like switching networks with deflection. In *Proceedings of the IEEE Infocom'90*, San Francisco, CA, June 1990. IEEE Press.

[11] C. P. Kruskal, M. Snir, and A. Weiss. The distribution of waiting times in clocked multistage interconnection networks. *IEEE Transactions on Computers*, 37(11):1337–1352, November 1988.

[12] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 256–269. IEEE Press, 1988.

[13] D. Mitra and R. A. Cieslak. Randomized parallel communications on an extension of the omega network. *Journal of the Association for Computing Machinery*, 34(4):802–824, October 1987.

[14] N. Pippenger. Parallel communication with limited buffers. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, pages 127–136. IEEE Press, 1984.

[15] A. Ranade. How to emulate shared memory. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 185–194. IEEE Press, 1987.

[16] E. Upfal. Efficient schemes for parallel communication. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 55–59. ACM, August 1982.

[17] E. Upfal. An $O(\log N)$ deterministic packet routing scheme. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 241–250. ACM, 1989.

[18] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 263–277, May 1981.

A Preliminary Study of Deflection Routing on the Butterfly Network.

Term Paper EE497 Spring 90

Andrea Pietracaprina

Abstract

In this paper we consider deflection routing on the butterfly network and concentrate on the number of packets which are delivered to the destinations without being deflected. We study the mean and the distribution of this number. Previous work has been concerned only with the mean. We rederive the lower and upper bounds for the mean and develop an approximate generating function for the density. Simulation results show that the approximation is quite accurate. It turns out that the distrubution is concentrated around the mean value. Unfortunately, the generating function is still in a form which is not easy to manipulate and we are working to find a nicer form.

1 Introduction.

The *Packet Routing* is considered one of the most important issues for packet switching networks. In general, the problem consists of delivering a certain number of packets from the inputs to the outputs of a network. Each input may have $d \geq 0$ packets and each packet may be destined to one or more outputs. Several versions of this problem have been analysed in the past few decades for different kinds of networks. A switching network is called *non-blocking* for a particular set of input-output connection requests if the routes of the packets through the network do not interfere with each other, that is the packets can be delivered with no two packets contending for the same link at any point in the network.

Unfortunately, networks turn out to be non-blocking in very few cases. More frequently, collisions occur among the packets which travel toward their destinations. Different approaches can be adopted to deal with this problem. One approach is to provide the nodes of the network with buffers for queuing packets temporarily blocked. In this case, the routing strategy should try to minimize the packet delays and the queue sizes. Alternatively, queues can be avoided by eliminating all the packets contending for the same link, except one. If packets are eliminated dynamically during the routing process, they can be lost. In order to avoid that, a preprocessing phase may determine a subset of packets which can be delivered without collisions; the other packets could be stored in input buffers, so that they may try the connection successively. In both cases, the goal is to maximize the number of packets delivered at once and minimize the total time needed to deliver all packets.

Another approach consists of misrouting a packet which loses the contention for a link, which means that the packet is sent along another link and the distance to its destination possibly increases. This approach was originally proposed by [Ba64] who called it *hot potato routing*; later, the term *deflection routing* has been adopted. Most of the work on this subject has been done in recent years. Borodin and Hopcroft [BH85] propose deflection routing for a permutation problem on a N-node hypercube network. They don't give a formal analysis, but only report some promising simulation results. Greenberg and Hajek [GH89] present an approximate analysis of transient and

equilibrium behaviour of the hypecube under deflection routing. A similar analysis is given by Krishna and Hajek [KH89] for several networks based on Shuffle Exchanges. Shuffle Exchange is also considered in [TSH88], but the authors only present some simulation results. Deflection routing on two dimensional meshes is adopted by Maxemchuk [Ma87] and by Greenberg and Goodman [GG86]. Greenberg and Goodman give estimates of the steady state throughput and the packet delay. An actual application of deflection routing can be found in the HEP multiprocessor system [Sm81] where this strategy has been used in the switching network which connects processors, memory modules and I/O devices.

In this paper we present a preliminary study of deflection routing on the butterfly network. We assume the network synchronous and the time slotted. At the beginning, each input independently generates a packet, with probability p_0 for a random destination. Each packet is sent through the unique path from the input to the output and traverses a link at each time slot. The deflection strategy is applied in case of conflicts. A more detailed description of the model is given in section 2. Because the paths are unique, a deflected packet will necessarily reach a wrong output and, if wrap-around connections are provided, it can start, at that point, a new journey toward the right destination. Our study is preliminary in the sense that we only focus on the number of packets which reach their destinations without being deflected. This number can be represented by a random variable. Bounds on the mean value of this variable are known, but no work has been done, so far, about the distribution. In section 3.1, we briefly reconsider the mean and derive a different lower bound. In section 3.2, we develop an approximate generating function for the density. Although it is only an approximation, simulation results, reported in section 4, suggest that it is quite accurate. The generating function is still in a form which is not easy to manipulate and our present research is focused on deriving a nicer form. Finally, section 5 discusses some open problems which should be investigated.

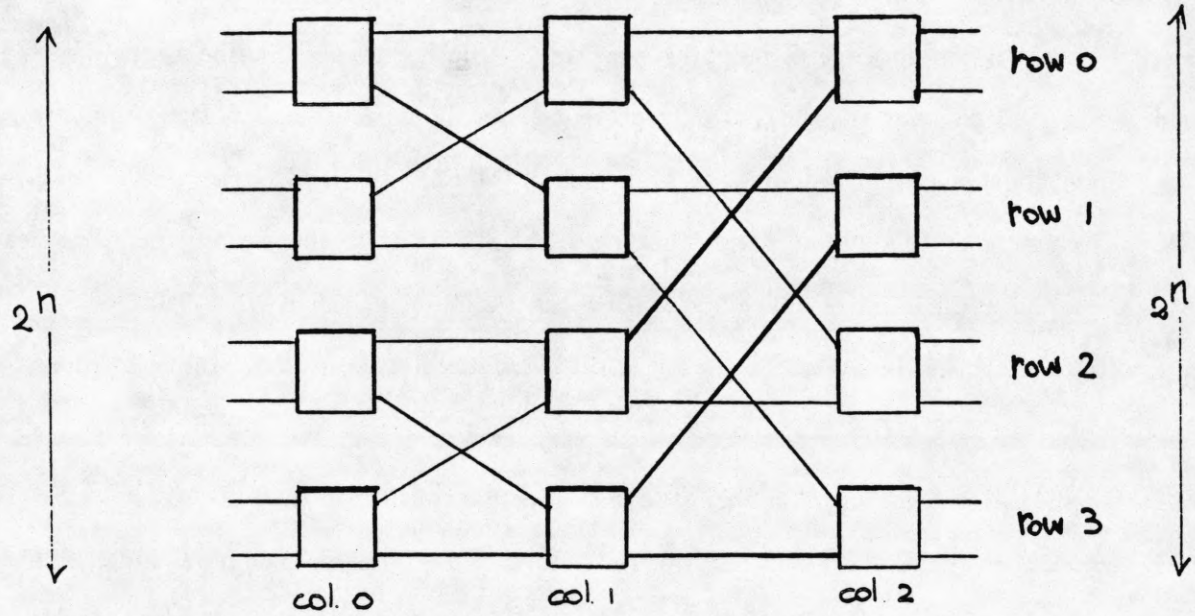


Figure 1: A butterfly of order 3.

2 The Model.

2.1 The network.

A *butterfly network of order* n, belongs to the wide class of banyan networks and consists of n stages of 2X2 switches which connect 2^n inputs to 2^n outputs. Each stage contains 2^{n-1} switches. Following a usual notation, we call the stages *columns* and subdivide the switches in each column into *rows*. A switch is identified by a pair $< c, r >$ where c, $0 \leq c \leq n-1$ represents the column, and r, $0 \leq r \leq 2^{n-1} - 1$ represents the row. A switch $v =< c, r >$ with $c < n - 1$ is directly connected to two switches $v_1 =< c+1, r >$ and $v_2 =< c+1, r' >$ in column $c+1$, where $r' = r \oplus 2^c$ and $\oplus$ denotes bitwise exclusive or. The inputs enter the switches in column 0 and the outputs leave the switches in column $n-1$. Usually, the network is wrapped-around by providing fixed connections between input i and output i, $0 \leq i \leq 2^n - 1$. A butterfly of order $n = 3$ is shown in fig.1. The network has a symmetric and recursive structure. It is easily seen that column $n-1$ is connected to the top columns of two disjoint butterflies of order $n-1$. In general, column c, $1 \leq c \leq n-1$ is connected to the top columns of 2^{n-c} disjoint butterflies of order c. As stated in the introduction, the network is synchronous and the time is slotted. The i-th time slot will be identified by t_i and

all operations begin at time t_0. At each time slot a switch in column c can recive a packet from one of the two links coming from column $c-1$ (if $c=0$ they are input links) and send the packet through one of the two links going to column $c+1$ (if $c=n-1$ they are output links).

2.2 The routing problem.

It can be easily shown that in the butterfly there exists a unique path connecting one input to one output, and this path is completely determined by the output address. In particular, each transition between two adjacent columns is determined by one bit of this address. We assume that at time t_0, each input link independently generates a packet for a random destination, with uniform probability p_0, $0 \leq p_0 \leq 1$. Therefore, the number of packets offered to the network at time t_0 has a binomial distribution with parameters 2^n and p_0. The destinations are independent and uniformly distributed in $[0, 2^n - 1]$ (*uniform traffic*). Each packet contains a header, with the destination address, and some data information. At each time slot a packet traverses a link between two nodes in adjacent columns. We call it a *transition*. Each packet tries to follow the unique path which leads to its destination. No packet can be buffered at a switch at any time. This implies that all the packets are forced to make a transition at each time slot, and then, at time t_c all packets are in column c. For this reason, in what follows we will avoid specifying the time, unless it is necessary. Since the paths are not necessarily disjoint, a *collision* may arise when two packets, in input to switch $< c, r >$, require a transition along the same output link, as shown in fig. 2. In this case we apply a deflection strategy. One of the two packets, chosen by tossing a fair coin, is deflected to the other link. By uniqueness of input-output paths, the deflected packet will travel up to a wrong destination from which it can try to reach the right destination in a successive traversal of the network. While the packet travels along the wrong path, any transition is equivalent, so that it can never be involved in a conflict. For our purposes, we don't care about deflected packets. We say that an input to a switch $< c, r >$ contains a *live packet* if there is a non deflected packet, and *no packet* if there is no packet or a deflected packet.

In the following sections we study the distribution of the number of live packets in each column and, in particular, the number of live packets which arrive at the right destinations without being

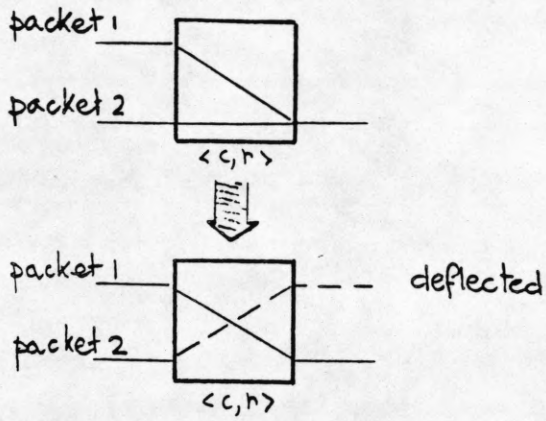


Figure 2: A collision.

deflected.

3 Statistics.

3.1 Average number of survivals.

In this section we determine the average number of live packets arriving to each column.

Definition 1 *For a butterfly of order n, the random variable $X_n(c, p_0)$, $0 \leq c \leq n-1$, denotes the number of live packets input to column c and $X_n(n, p_0)$ denotes the number of packets that arrive at the desired outputs without being deflected*

For notational simplicity, we will omit the subscript n when the context makes it clear.

Definition 2 *Let l be an input link to a switch in column c, $0 \leq c \leq n-1$. Then, $p_c(l)$ is the probability that the link contains a live packet. If l is an output link, then $p_n(l)$ is the probability that l receives a packet destined to it.*

We have $p_0(l) = p_0$ for any l, and a symmetry argument easily shows that, for fixed $c \geq 1$, also $p_c(l)$ does not depend upon l. Thus, we can omit the parameter l. As noticed before, $X(0, p_0)$ has a binomial distribution with parameters 2^n and p_0, since it can be expressed as sum of 2^n independent Bernoulli random variables with parameter p_0. For $c \geq 1$, $X(c, p_0)$ is still a sum of 2^n Bernoulli variables, but they are no longer independent because of the collisions among the packets. Thus,

we cannot claim that $X(c, p_0)$ has binomial distribution. However, the statistical dependence does not affect the mean, so that we have:

$$E(X(c, p_0)) = 2^n p_c, \ 0 \leq c \leq n \tag{1}$$

Given p_0, we can determine $p_c, c \geq 1$, by solving the recurrence relation given in the following lemma.

Lemma 1 *Given p_0, for $c = 0 \ldots n-2$,*

$$p_{c+1} = p_c - \frac{1}{4}(p_c)^2.$$

Proof: Consider a switch $< c, r >$ and let l_1 and l_2 be the two input links. Each link has a live packet with probability p_c and independently of the other link. This fact follows because, as shown in fig. 3, the two links come from two disjoint butteflies of order c (if $c = 0$ the probabilities p_0 are independent by definition). Therefore, an input link to column $c+1$, which is an output of a switch in column c, will have a packet with probability

$$p_{c+1} = 1 - (1 - \frac{p_c}{2})^2 = p_c - \frac{(p_c)^2}{4}$$

□

Kruskal and Snir [KS83] give an asymptotical bound for this recurrence relation. They show that for any fixed initial value p_0,

$$p_c = \frac{4}{c}\left[1 - \frac{\ln c}{c} + O\left(\frac{1}{c}\right)\right]$$

where $\ln x$ denotes the natural logarithm. We now prove a lower and an upper bound which are functions of p_0.

Theorem 1 *For $c = 0 \ldots n-1$,*

$$g(c) \leq p_c \leq f(c)$$

where $g(c) = \frac{4-p_0}{c+\frac{4-p_0}{p_0}}$ and $f(c) = \frac{4}{c+\frac{4}{p_0}}$.

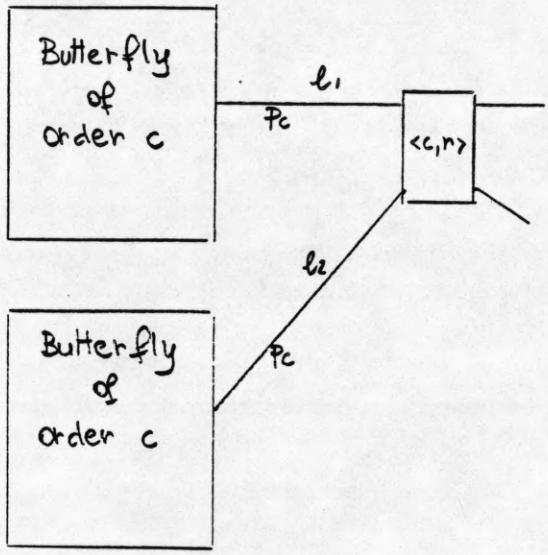


Figure 3: Transition between column $c-1$ and column c

Proof: It can be easily verified that for $c \geq 1$,

$$g(c) \leq g(c-1) - \frac{1}{4}[g(c-1)]^2$$

and

$$f(c) \geq f(c-1) - \frac{1}{4}[f(c-1)]^2.$$

We now show $g(c) \leq p_c \leq f(c)$ by induction on c. For the basis, we have $g(0) = p_0 = f(0)$. Inductively assume that $g(c-1) \leq p_{c-1} \leq f(c-1)$ for some $c \geq 1$. Since the function $\phi(x) = x - \frac{1}{4}x^2$ is strictly increasing for $0 \leq x \leq 2$ and $g(c-1)$, $f(c-1)$ and p_{c-1} are probabilities, we have

$$p_c = p_{c-1} - \frac{1}{4}(p_{c-1})^2 \geq g(c-1) - \frac{1}{4}[g(c-1)]^2 \geq g(c)$$

and

$$p_c = p_{c-1} - \frac{1}{4}(p_{c-1})^2 \leq f(c-1) - \frac{1}{4}[f(c-1)]^2 \leq f(c)$$

□

The same upper bound is also proved in [KH89]. If p_0 is a constant, then $p_c \in \Theta\left(\frac{1}{c}\right)$, and, by applying relation 1, the average number of packets which arrive at the right destination after n

time slots is $E(X(n,p_0)) \in \Theta\left(\frac{2^n}{n}\right)$ to be compared with the $\Theta(2^n)$ packets which start. On the other hand, for $p_0 = \frac{1}{n}$ we have $\simeq \frac{2^n}{n}\frac{4}{5}$ arrivals and $\frac{2^n}{n}$ departures on average, which means that a constant fraction of packets has been deflected.

3.2 The density function of $X(n,p_0)$

In this section we investigate the distribution of the random variable $X(n,p_0)$ previously defined. Let $\pi_{n,p_0}(j) = \mathrm{P}(X(n,p_0) = j)$, for $0 \leq j \leq 2^n$. In other words, $\pi_{n,p_0}(j)$ is the probability that, when deflection routing is performed on a butterfly of order n, with initial load p_0, exactly j packets are delivered to the outputs after n time slots.

Definition 3 *The generating function for $\pi_{n,p_0}(j)$ is defined by*

$$G_{n,p_0}(z) = E(z^{X(n,p_0)}) = \sum_{j=0}^{2^n} \pi_{n,p_0}(j) z^j$$

We want to derive $G_{n,p_0}(z)$ in terms of the probabilities of lower order, that is in terms of $\pi_{n',p_0}(j)$ for $n' < n$. For this purpose, we exploit the recursive structure of the butterfly. As noticed before, the last column (i.e. column $n-1$) of a butterfly of order n is connected to two disjoint butterflies of order $n-1$. Since these two smaller butterflies are disjoint, their histories are described by independent distributions π_{n-1,p_0}. The idea is to start from the number of packets which arrive to the outputs of these butterflies and consider the last transition up to column $n-1$. Note that $\pi_{n-1,p_0}(j)$ gives only the probability of having j live packets, but it doesn't say which outputs contain the live packets. In general, the $\binom{2^{n-1}}{j}$ configurations with exactly j live packets do not have the same probability. In order to determine the generating function we need to assume that each subset of j outputs contains the packets with the same probability $\frac{\pi_{n-1,p_0}(j)}{\binom{2^{n'}}{j}}$.

Consider a butterfly of order n and a switch $< n-1, r >$ in the last column. According to the presence of live packets on its inputs, this switch can be in one of the four states s_i, $i = 0 \ldots 3$, represented in fig. 4. Assume that in column $n-1$ there are k_i switches in state i, $i = 0 \ldots 3$, where $\sum_{i=0}^{3} k_i = 2^{n-1}$, and let $p_{n,p_0}(k_0, k_1, k_2, k_3)$ be the probability of such event. The subscripts n and p_0 denote, as usual, the order of the butterfly and the initial load, respectively. In the following

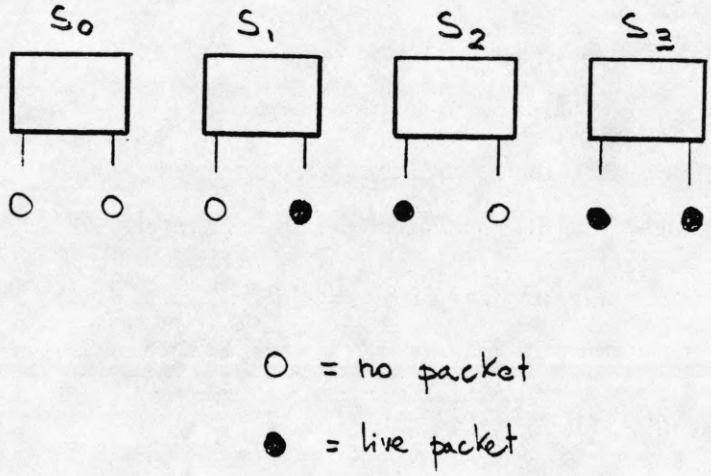


Figure 4: States of a switch

lemma we give an exact form of the generating function $G_{n,p_0}(z)$.

Lemma 2

$$G_{n,p_0}(z) = \sum_{k_3=0}^{2^{n-1}} \sum_{k_2=0}^{2^{n-1}-k_3} \sum_{k_1=0}^{2^{n-1}-k_3-k_2} p_{n,p_0}(k_0,k_1,k_2,k_3) z^{k_1+k_2+k_3} \left(\frac{1+z}{2}\right)^{k_3}$$

Proof: Note that k_0 is uniquely determined by k_1, k_2, k_3 that is $k_0 = 2^{n-1} - k_1 - k_2 - k_3$. The three summations fix the values for k_1, k_2, k_3 in all possible ways. Now, the switches in state s_1 or s_2 output exactly one packet with probability 1 and contribute a factor $z^{k_1+k_2}$ to the generating function. The switches in state s_3 output either one packet with probability $\frac{1}{2}$, or two packets with probability $\frac{1}{2}$. Then they contribute a factor $\left(\frac{z}{2} + \frac{z^2}{2}\right)^{k_3}$ to the generating function. The lemma follows by rearranging the terms in z. □

Now, we express $p_{n,p_0}(k_0,k_1,k_2,k_3)$ in terms of the distribution π_{n-1,p_0}. As stated before, we need to pretend that all configurations with the same number of live packets at the outputs of a butterfly of order n are equiprobable.

Lemma 3

$$p_{n,p_0}(k_0,k_1,k_2,k_3) = \frac{\pi_{n-1,p_0}(k_1+k_3)}{\binom{2^{n-1}}{k_1+k_3}} \frac{\pi_{n-1,p_0}(k_2+k_3)}{\binom{2^{n-1}}{k_2+k_3}} \frac{(2^{n-1})!}{k_3!k_2!k_1!k_0!}$$

Proof: Recall that the inputs to column $n-1$ are the outputs of two disjoint butterflies of order $n-1$. It is easily seen that a quadruplet (k_0, k_1, k_2, k_3) implies that $k_1 + k_3$ packets are output by one butterfly and $k_2 + k_3$ packets by the other. There are $\frac{(2^{n-1})!}{k_3!k_2!k_1!k_0!}$ ways of fixing the k_i switches in state i, for $i = 0 \ldots 3$. Fixed the state of each switch, we have a particular configuration of $k_1 + k_3$ packets at the outputs of one butterfly and a particular configuration of $k_2 + k_3$ packets at the outputs of the other. By the fact that the two butterflies are independent and by the assumption made above, the two configurations occur with probability

$$\frac{\pi_{n-1,p_0}(k_1+k_3)}{\binom{2^{n-1}}{k_1+k_3}} \frac{\pi_{n-1,p_0}(k_2+k_3)}{\binom{2^{n-1}}{k_2+k_3}}$$

and the lemma follows. □

We are now ready to write the approximate generating function that we denote by $\tilde{G}_{n,p_0}(z)$.

Theorem 2

$$\tilde{G}_{n,p_0}(z) = \sum_{k_3=0}^{2^{n-1}} \sum_{k_2=0}^{2^{n-1}-k_3} \sum_{k_1=0}^{2^{n-1}-k_3-k_2} \frac{\pi_{n-1,p_0}(k_1+k_3)}{\binom{2^{n-1}}{k_1+k_3}} \frac{\pi_{n-1,p_0}(k_2+k_3)}{\binom{2^{n-1}}{k_2+k_3}} \frac{(2^{n-1})!}{k_3!k_2!k_1!k_0!} z^{k_1+k_2+k_3} \left(\frac{1+z}{2}\right)^{k_3}$$

Proof: Immediate from lemma 2 and lemma 3. □

4 Simulation results.

The generating function $\tilde{G}_{n,p_0}(z)$, given in the previous section, has been derived by using the approximation that, in a butterfly of order n, if exactly k outputs contain live packets at time t_n, these packets may appear in any subset of k outputs, with the same probability. Unfortunately, this is not true. It could be shown that these subsets can be partitioned into classes with the same probability, according to a permutation group. The proof is pretty tedious and we prefer giving an intuitive argument by the following example. Consider a butterfly of order 2. The two configurations shown in fig. 5 do not have the same probability. For $p_0 = 0.5$, configuration A has

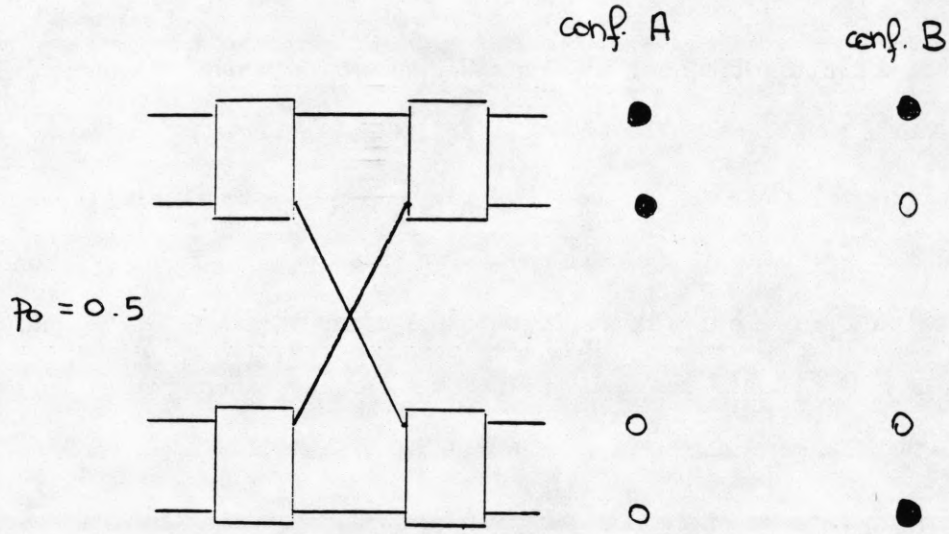


Figure 5: Configurations A and B.

probability $\simeq 0.049$ and configuration B has probability $\simeq 0.085$. This depends on the fact that the 'histories' which led to these configurations are different. Only one configuration of live packets in input to column 1 may generate A, while four configurations may generate B.

Nevertheless, the generating function turns out to be quite accurate. We conducted a set of experiments to evaluate confidence intervals for the mean and the variance of $X(n, p_0)$ by means standard simulation techniques [La83]. In particular, the confidence interval for the variance has been extimated by using a Jackknifing technique. We simulated deflection routing on butterflies with 16, 32, 64 and 128 inputs, and with initial probabilities $p_0 = 0.1, 0.3, 0.5, 0.7, 0.9$. For each dimension and each value p_0 we performed 10000 independent runs. The results are summarized in tables 1-4. Each table refers to a particular dimension of the network. The first column indicates the initial probability. Columns 2-4 indicate the exact mean, determined by the recurrence relation presented in section 3.1, the value given by the generating function and the interval estimated with 95% of confidence, respectively. Columns 5-7 compare the variance obtained by the generating function, the sample variance s^2 and the interval estimated with 95% of confidence. Note that, in all cases, there is a close agreement between the simulation estimates and the values given by the generating function.

	Mean			Variance		
p_0	exact	Gen. Fun.	interval 95%	Gen. Fun.	s^2	interval 95%
0.1	1.45	1.45	1.42 ÷ 1.46	1.09	1.09	1.06 ÷ 1.12
0.3	3.63	3.63	3.62 ÷ 3.67	1.65	1.71	1.64 ÷ 1.76
0.5	5.13	5.13	5.11 ÷ 5.16	1.64	1.67	1.62 ÷ 1.71
0.7	6.18	6.18	6.15 ÷ 6.20	1.60	1.61	1.57 ÷ 1.66
0.9	6.92	6.92	6.91 ÷ 6.96	1.63	1.64	1.60 ÷ 1.69

Table 1: Butterfly 16X16.

	Mean			Variance		
p_0	exact	Gen. Fun.	interval 95%	Gen. Fun.	s^2	interval 95%
0.1	2.84	2.84	2.82 ÷ 2.88	2.04	2.02	1.96 ÷ 2.07
0.3	6.85	6.85	6.83 ÷ 6.89	2.94	2.90	2.82 ÷ 2.98
0.5	9.44	9.44	9.43 ÷ 9.50	2.92	2.92	2.84 ÷ 3.00
0.7	11.17	11.17	11.13 ÷ 11.19	2.91	2.93	2.85 ÷ 3.01
0.9	12.34	12.34	12.31 ÷ 12.38	3.00	3.03	2.94 ÷ 3.11

Table 2: Butterfly 32X32.

	Mean			Variance		
p_0	exact	Gen. Fun.	interval 95%	Gen. Fun.	s^2	interval 95%
0.1	5.55	5.55	5.49 ÷ 5.57	3.84	3.76	3.66 ÷ 3.87
0.3	12.97	12.97	12.95 ÷ 13.04	5.28	5.17	5.02 ÷ 5.31
0.5	17.49	17.49	17.44 ÷ 17.53	5.29	5.32	5.17 ÷ 5.46
0.7	20.38	20.38	20.33 ÷ 20.42	5.36	5.40	5.25 ÷ 5.55
0.9	22.30	22.30	22.24 ÷ 22.34	5.55	5.47	5.32 ÷ 5.62

Table 3: Butterfly 64X64.

	Mean			Variance		
p_0	exact	Gen. Fun.	interval 95%	Gen. Fun.	s^2	interval 95%
0.1	10.86	10.86	10.80 ÷ 10.91	7.26	7.19	6.98 ÷ 7.39
0.3	24.62	24.62	24.58 ÷ 24.70	9.60	9.64	9.55 ÷ 10.12
0.5	32.59	32.59	32.56 ÷ 32.68	9.71	9.75	9.48 ÷ 10.02
0.7	37.52	37.52	37.45 ÷ 37.57	9.96	10.13	9.84 ÷ 10.41
0.9	40.71	40.71	40.61 ÷ 40.74	10.34	10.55	10.25 ÷ 10.84

Table 4: Butterfly 128X128.

By observing the above results, we notice that the distribution of $X(n, p_0)$ is quite concentrated around the mean value. To have a more tangible proof of this fact, the first graphic in fig. 6 reports the results of 10000 independent runs for a butterfly with 128 inputs and with initial probability 0.5. The horizontal axis shows the number of arrivals and the vertical axis the occurrences of each particular value. For comparison, the other graphic reports the values predicted by the generating function; once again we observe the accurancy of this function.

5 Conclusions and open problems.

In this paper we considered deflection routing on the butterfly network under the assumption of uniform and independent traffic offerd at the inputs. In particular, we focused on the first phase of the routing (i.e. the first traversal of the network) and we studied the distribution of the random variable $X(n, p_0)$ which represents the number of packets delivered to the right destinations within this phase. Previously, only the mean value has been studied. We slightly modified the lower bound for the mean and developed a generating function for the density. The generating function is based on the approximation that all configurations of exactly k live packets at the outputs have the same probability. Although this assumption is not generally true, simulation estimates for mean and variance closely match the values obtained by the generating function. Other extensive simulations, not shown in the paper, have confirmed its accurancy. Unfortunately, the function is still in a form which is not easy to manipulate and our present research is focused on finding a nicer or even a close formula.

Both simulations and generating function show that the distribution is concentrated around the mean. It is interesting to note that, when the initial probability p_0 is small, the values of mean and variance are pretty close. In this case, the poisson distribution seems to be a good approximation. This fact is not surprisingly because it is known that under certain hypothesises, the sum of dependent bernoulli variables converges to the poisson distribution [AGG89]. However, when the value p_0 increases, this approximation is no more applicable because the mean and the variance tend to differ each other, as indicated by the results in the previous section.

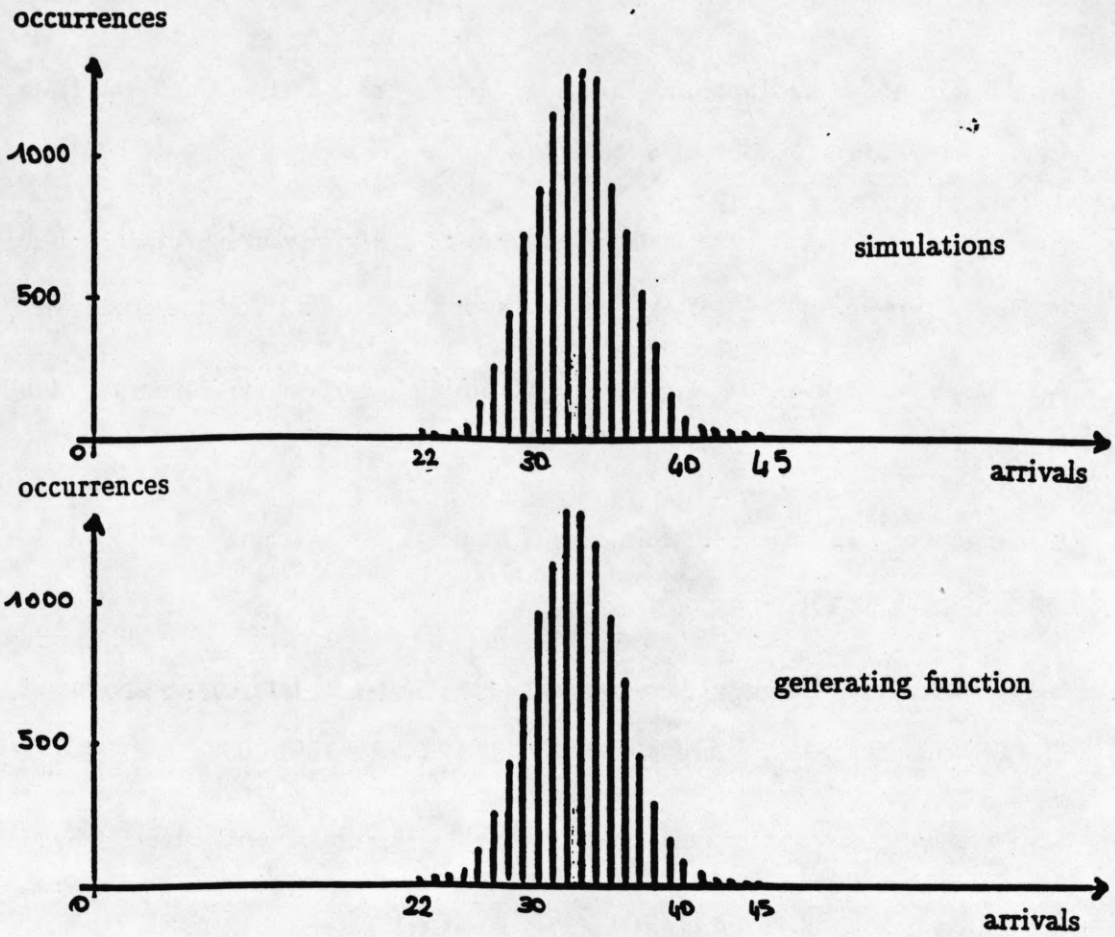


Figure 6: 10000 runs for a butterfly 128X128 with $p_0 = 0.5$.

So far we have only considered the first phase of deflection routing on the butterfly. The author's future research will be concentrated on the global routing, where several traversals of the network may be necessary to deliver all packets. Moreover, in a more realistic scenario, we can suppose that the traffic is offered dynamically. The knowledge of the distribution of the number of packets delivered in one traversal becomes helpful if we want to study the overall time and the optimal tradeoff between time and offered traffic.

References

[AGG89] R. Arratia, L. Goldstein and L. Gordon: 'Two Moments Suffice for Poisson Approximations: the Chen-Stein Method', in *The Annals of Prob.* 17,1 (1989) 9-25.

[Ba64] P. Baran: 'On Distributed Communication Networks', in *IEEE Trans. Comm. Systems* March (1964) 1-9.

[BH85] A. Borodin and J.E. Hopcroft: 'Routing, Merging and Sorting on Parallel Models of Computation', in *JCSS* 30 (1985) 130-145.

[GG86] A.G. Greenberg and J. Goodman: 'Sharp Approximate Models of Adaptive Routing in Mesh Networks', Manuscript (1986).

[GH89] A.G. Greenberg and B. Hajek: 'Deflection Routing in Hypercube Networks', Manuscript (1989).

[KH89] A. Krishna and B. Hajek: 'Performance of Shuffle-Like Switching Networks with Deflection', Manuscript (1989).

[KS83] C.P. Kruskal and M. Snir: 'The Performance of Multistage Interconnection Networks for Multiprocessors', in *IEEE Trans. Comp.* C-32,12 (1983) 1091-1098.

[La83] S.S Lavenberg: 'Computer Performance Modeling Handbook', Academic Press (1983).

[Ma87] N.F. Maxemchuk: 'Routing in the Manhattan Street Network', in *IEEE Trans. Communications* COM-35,5 (1987) 503-512.

[Sm81] B.J. Smith: 'Architecture and Applications of the HEP Multiprocessor Computer System', in *Real Time Signal Processing IV, Proc. of SPIE* (1981) 241-248.

[TSH88] X.N. Tan, K.C. Sevcik and J.W Hong: 'Optimal Routing in the Shuffle-Exchange Networks for Multiprocessors Systems. in *CompEuro 88 - System Design: Concepts, Methods and Tools* IEEE Euromicro (1988) 255-264.

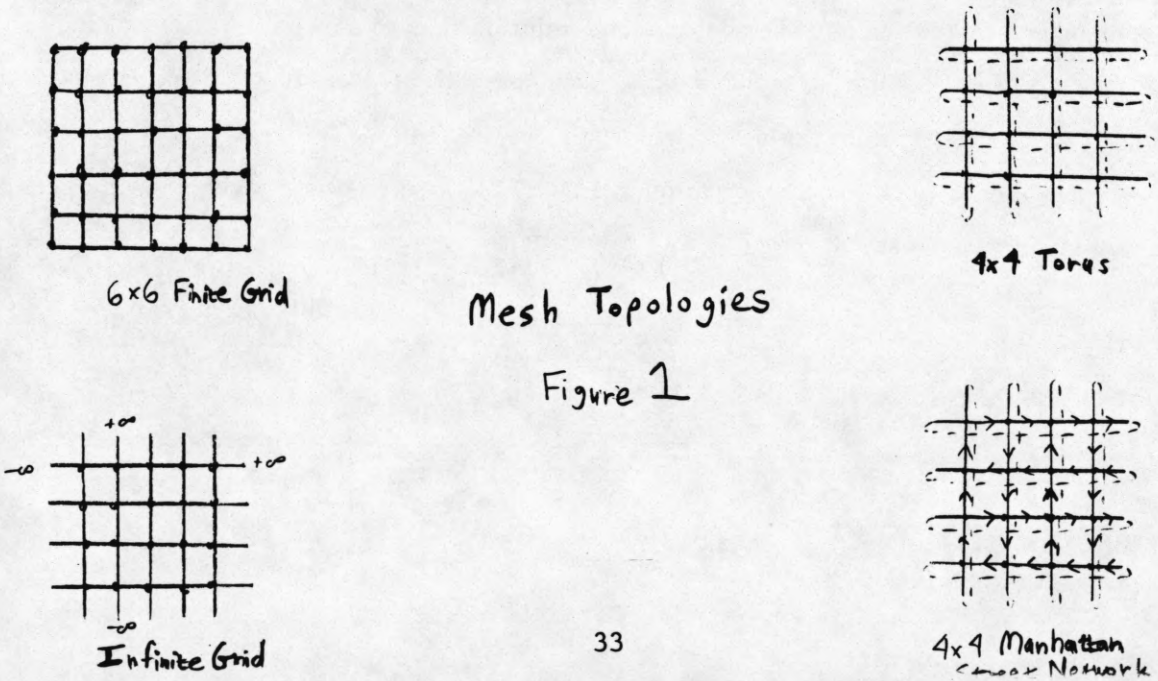
An Optimal Policy for Deflection Routing on a 2-Dimensional Grid

Timothy Weller

May 11, 1990

1 Introduction

The class of mesh networks is a broad class of *regularly connected* networks like hypercubes, grids, etc. In this paper we consider only 2-D rectangular grid topologies where each node is connected to 4 nearest neighbors. If the links are bidirectional and the boundary nodes have degree 3 (degree 2 for corner nodes) then we call the topology a *finite grid.* If the boundaries extend to infinity, it is an *infinite grid.* A finite grid with each row connected in a bidirectional ring and each column connected in a bidirectional ring is called a *torus.* Finally, a torus structure with an even number of columns and rows and every other row and every other column and unidirectional rings in opposite directions is called a *Manhattan Street Network (MSN)*, so named because the rows and columns are like one-way streets in downtown Manhattan. See Figure 1 for diagrams of these networks. We assume all finite networks to be square.



Mesh Topologies

Figure 1

Our main result is for an infinite grid. Consider a packet at a source node on this grid. The packet wishes to go to a destination node. Define the two *diagonals* with respect to this destination node as the set of nodes whose vertical distance from the destination is the same as its horizontal distance. In any time slot at any intermediate node the packet will decide which of the four outgoing links is preferred as a first choice. Due to several reasons such as a conflict with another packet, a link failure, or congestion at the corresponding receiving node, the packet may not be transmitted on the first choice link. If this transmission fails, the packet will try to get out on its second choice of outgoing links. This and subsequent attempts may also fail. Since the topology is a grid, at any node there can be at most two outgoing links which bring the packet closer to its destination (only one if the packet is aligned with the destination in one coordinate). These will be called *shortest path links.*

If a packet fails to get out on a shortest path link in the current time slot, there are three possible strategies. One strategy is to delete the packet from the network. A second way to handle the packet is to queue it at the intermediate node and transmit it in a later time slot. However, this is not always desirable because it requires memory at each node (perhaps a prohibitive amount if the network speed is very high) and may cause buffers to overflow. These two methods are consider by Badr and Podar in [1]. A third choice, which we consider here, is to "deflect" the packet by sending it out on a link which makes it farther from the destination, a *non-shortest path link.* Some analytical results and simulations have shown this to work well for at least some topologies under certain traffic models (see [5] for example).

Using deflection routing and the additional assumption that no new packets can be injected into a node in any time slot during which a node has 4 packets, it is easily seen that no packet will ever need to be deleted from the network. It may, however, be desirable to delete packets which have been in the network too long or to offer priority to these or other packets in an attempt to maintain stability, increase throughput, or preserve fairness.

2 Our Deflection Routing Model

In our model we consider the travel of a single packet from a source to a destination node. At any intermediate node this packet will decide on a first, second, third, and last choice of outgoing links which it prefers. It will get its first choice with highest probability, its second choice with the next highest probability, etc. Because we assume deflection routing and restrict all new traffic injected into the network to those nodes which have fewer than 4 packets in any time slot, it is clear that our packet will leave any intermediate node on some link during a single time slot (no queueing necessary). We will make special consideration of the case where a packet can't get out on any link later.

The four outgoing transmission probabilities are assumed to be non-increasing in order of the packet's preference, uniform across the network (a reasonable assumption under uniform traffic), and independent from slot to slot (time-invariant). In general these will be time-varying parameters which depend on the traffic in the network, but we require this simplifying stationarity assumption. A steady-state fixed point analysis may allow us to improve on this assumption at a later time. The only cost incurred by the packet will be unit cost to traverse each link. Boundary conditions are not present as the grid will be considered to extend to infinity. Our result is conjectured to hold for a finite grid, but this introduces some additional boundary modeling issues and also makes the DP solution more difficult.

In this paper, we present a control policy which a packet should use to choose its outgoing link preferences for the network model described above, and we prove the optimality of this policy. This can be considered as an extension of the result of [1] for deflection routing.

3 The Zig-Zag Routing Policy

In keeping with the terminology used in [1], we call our optimal policy the Z^2 (zig-zag deflection routing) policy. In words, this policy says that among the shortest path links, the one which moves the packet closer to the diagonal with respect to the destination is to be preferred. The same preference is true for non-shortest path links if a packet must be deflected. This is intuitively

satisfying because moving toward the diagonal leaves a packet with more alternate shortest paths to the destination. As a example, with the destination $(0,0)$ and the packet at $(4,2)$, the link to $(3,2)$ is preferred over the link to $(4,1)$. The name comes from the fact that after reaching the diagonal, an undeflected packet will take a zig-zag path to the destination. See Figure 2.

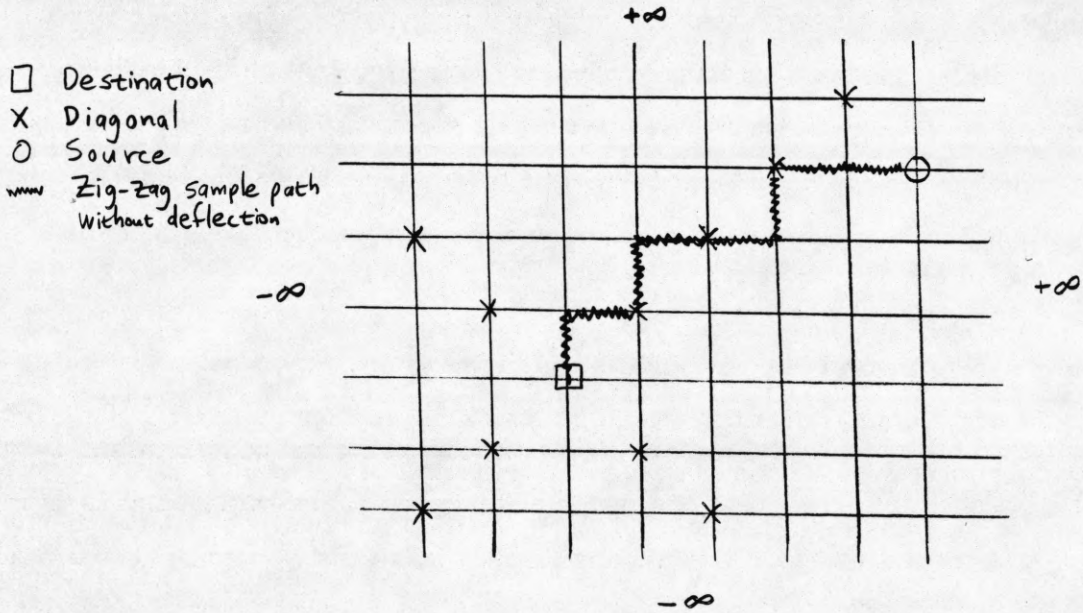


Figure 2

4 Summary of the Result of Badr and Podar

The result in our paper can be considered as an extension of a result by Badr and Podar in [1]. Their work will be summarized here, but not in detail because the analysis is similar, and our proof technique works for this problem also. Consider a packet at a given source node which wants to get to a given destination node on a finite grid. At any point along the way there will be either one or two outgoing shortest path links. A packet can be successfully on any given link with probability p, independently of all other links. If a packet can not get out on a shortest path link in a time slot, then the packet is dropped from the network (no deflection). This paper proves that to maximize the probability of delivery, a packet should first try to go out the shortest path link in the direction of the diagonal if such a link exists. If that is not possible then it tries to go out the other shortest path link if another exists. This is called Z^2 routing, as described above. Suppose instead of being

dropped when it can not leave on a shortest path link, the packet is instead delayed by W slots, at which point it can try again from the same node (the packet is never dropped; it just may be delayed a long time). To minimize expected time of arrival at the source, the Z^2 routing is again shown to be optimal. It turns out that the Dynamic Programming Equations (DPE) are the same as in the first problem, which will also be true in our case. These are the two results in this paper. The second follows from the first, and the first is proved by writing the DPE (by conditioning on the first move a packet makes) and then showing that a Z^2 policy achieves the minimum (using induction on the state space). It is interesting to note that the value function here represents probability.

5 Other Topologies and Suboptimal Routing

As mentioned earlier we have proved the optimality of Z^2 routing for an infinite grid, and DP value iteration for the finite grid suggests it to be true there as well. However, DP value iteration for the torus and the MSN shows that Z^2 routing is NOT optimal for these topologies (see attached program and output for example on a torus). This is because near the boundary a packet maintains more shortest path options at each step by staying on the boundary (due to wraparound) than by going toward the diagonal, which can result in a lower expected delivery time. In [1, page 1365], the Z^2 policy is stated as optimal for a torus as a representative example of all mesh topologies, which is clearly in error. Their proof as given holds up for finite and infinite grid problems. Unfortunately they have failed to distinguish the torus and the finite and infinite grid.

As a practical matter, when analyzing the value functions obtained from value iteration, we see that Z^2 routing will perform very well (within a few percent of optimal) for both the torus and the MSN, and due to its simplicity is probably desirable for implementation in most cases. It is not known how well this policy performs when the network topology is altered due to links which are known to have failed with probability 1. Due to the fact that the number of shortest paths is maximized at every step with Z^2, we conjecture that it performs as well as any policy which is uniform at all nodes (spatially invariant).

6 Our Problem Statement

We will use a dynamic programming approach to rigorously formulate and prove the optimality of the Z^2 control policy for our problem. See [2] for more on standard formulations of dynamic programming problems. We adopt the notation used there. Our method is simliar to that of Hajek in [4]. Our state space is the two dimensional integer plane $\mathbf{X}=\mathbf{Z}^2$. We consider the movement of a single packet through the network. Labeling the destination node $(0,0)$, we will define $V^*(x)$ as the minimum expected time for the packet at x to reach $(0,0)$. This minimum is taken over all control policies. The state is x_k, the location of the packet at time k . Let p_i be the probability that this packet is transmitted on its ith preferred outgoing link in any time slot from any intermediate node x_k. We require $p_1 \geq p_2 \geq p_3 \geq p_4$ and $p_1 > p_4$. Let w_k be a random variable which takes value i with probability p_i, $i = 1,2,3,4$. This represents the choice (from first to last) link that the packet actually gets transmitted out on in slot k. Let an admissible control be $u_k = (u_k^1, u_k^2, u_k^3, u_k^4) \in U = \{\text{permutations of } [(-1,0),(1,0),(0,-1),(0,1)]\}$. Our state equation is then

$$x_{k+1} = \begin{cases} x_k + u_k^{w_k} & \text{if } x_k \neq (0,0) \\ (0,0) & \text{if } x_k = (0,0) \end{cases}$$

So we have a semi-Markov decision process. Our underlying probability space will be the set of all random walks on $\mathbf{X}$ which are absorbed at $(0,0)$.

$$\begin{aligned} \Omega &\equiv \{\text{all infinite sequences of integer pairs ending in} \\ &\quad (0,0),(0,0),...\} \\ \tau(\omega) &\equiv \min\{i > 0 : \omega_i = (0,0)\},\ \forall \omega = (\omega_0, \omega_1, \ldots) \in \Omega \end{aligned}$$

We have defined the random variable $\tau(\omega)$ to be the hitting time of $(0,0)$ for the sample path ω. Our packet will be assessed unit cost to traverse a link and no other costs, so that the cost per stage is $g(x_k) = I_{\{x_k \neq (0,0)\}}$.

<u>Model 1:No packet rejections</u>

Suppose that $p_1 + p_2 + p_3 + p_4 = 1$, i.e. a packet always gets out on some link in each time slot. We define the cost for a policy $\mu = (u_0, u_1, ..)$ and the optimal policy as follows for an infinite

horizon problem.

$$\begin{aligned} V^{\mu}(x) &\equiv E_x^{\mu}[\tau] \\ &= lim_{N\to\infty} E_x^{\mu} \sum_{k=0}^{N-1} g(x_k) \\ V^*(x) &\equiv \inf_{\mu} V^{\mu}(x) \\ &= \text{minimum expected time to reach } (0,0) \text{ from } x \end{aligned}$$

Our objective is to minimize the expected delivery time by proper choice of the optimal control policy.

Model 2:Rejected packets are dropped

Suppose that $p_1 + p_2 + p_3 + p_4 < 1$, i.e. a packet is rejected and dropped from the network in any time slot with probability $p_r = 1 - (p_1 + p_2 + p_3 + p_4)$. For $i = 1,2,3,4$ we replace p_i by $p_i/(1-p_r)$. Now $p_1 + p_2 + p_3 + p_4 = 1$. We can think of allowing a virtual packet to travel as in Model 1 until it hits $(0,0)$, and then we decide if the actual packet reaches $(0,0)$, which will happen with probability p_r^{τ} since at every step it was rejected with probability p_r. Now our objective is to maximize our probability of arrival, P_A.

$$\begin{aligned} P_A &\equiv E[(1-p_r)^{\tau}] \\ &= \sum_{k=1}^{\infty} (1-p_r)^k P\{\tau = k\} \\ &= 1 - p_r \sum_{k=0}^{\infty} (1-p_r)^k P\{\tau > k\} \\ &= 1 - p_r \sum_{k=0}^{\infty} (1-p_r)^k E[I_{\{x_k \neq (0,0)\}}] \end{aligned}$$

To maximize P_A, we can clearly minimize the last summation without the constant. By defining the discount factor $\beta = (1-p_r)$ we get the standard definition of the cost functions for an infinite horizon discounted cost problem. Note that the Monotone Convergence Theorem allows us to interchange the lim and $E[\cdot]$.

$$\begin{aligned} V^{\mu}(x) &\equiv lim_{N\to\infty} E_x^{\mu} \sum_{k=0}^{N-1} \beta^k g(x_k) \\ V^*(x) &\equiv \inf_{\mu} V^{\mu}(x) \end{aligned}$$

So the two models are equivalent if we treat the problem with a discount factor $\beta \leq 1$. Model 1 has $\beta = 1$ and Model 2 has $\beta < 1$. The rejection probability simply discounts our cost.

We can get the corresponding DPE by conditioning on the first move from $x = (i, j)$.

$$\begin{aligned} V^*(x) &= 1 + \beta \, min_u E_w V^*(x + u^w) && (6.1) \\ &= 1 + \beta \, min_u [p_{u_1} V^*(i+1, j) + p_{u_2} V^*(i-1, j) && (6.2) \\ &\quad + p_{u_3} V^*(i, j-1) + p_{u_4} V^*(i, j+1)] && (6.3) \\ V^*(0) &= 0 && (6.4) \end{aligned}$$

This is the celebrated Bellman Equation, whose solution unfortunately is not guaranteed to be unique in the undiscounted case. However we do know that it has at least one solution since our cost per stage is bounded below [2]. Later we will show it has unique solution for our problem. To prove some properties about V^* and to find the minimizing u^* we consider iterates of the value function in an infinite horizon dynamic programming problem defined for $x = (i, j)$ as

$$\begin{aligned} V^{n+1}(x) &\equiv \inf_{\mu} E_x^{\mu}[\tau \wedge (n+1)] \\ &= 1 + min_u [p_{u_1} V^n(i+1, j) + p_{u_2} V^n(i-1, j) + p_{u_3} V^n(i, j-1) + p_{u_4} V^n(i, j+1)] \\ V^0(x) &\equiv 0 \end{aligned}$$

This defines a mapping $V^{n+1} = T(V^n)$. We will show that these iterates have two special properties.

Property S We say that a function $V(\cdot)$ defined on $\mathbf{X}$ has Property S (8-way symmetry) if for every $(i, j) \in \mathbf{X}$ we have $V(i, j) = V(i, -j) = V(-i, j) = V(j, i)$.

Property O We say that a function $V(\cdot)$ defined on $\mathbf{X}$ has Property O (Ordering of Neighbors) if it has Property S and for every $(i, j) \in \mathbf{X}$ such that $i, j \geq 0, i \geq j$ we have $V(i-1, j) \leq V(i, j-1) \leq V(i, j+1) \leq V(i+1, j)$.

Proposition 6.1 *For every $n = 0, 1, 2, ...,$ V^n has Property S.*

Proof. This follows easily by induction because the mapping T preserves symmetry and V^0 is trivially symmetric. □

Proposition 6.2 *For every $n = 0, 1, 2, ...$, V^n has Property O.*

Proof. Again we proceed by induction. Trivially, V^0 has Property O. Suppose $V^n, n \geq 0$ has Property O. Fix any $(i,j) \in \mathbf{X}$ such that $i,j \geq 0, i \geq j$. There are four separate cases to be considered. We use the symbol $\equiv$ to mean by definition. To simplify the notation, we refer to coordinates by the letter designations as $a = (i, j-2)$, $b = (i-1, j+1)$, $c = (i, j+1)$, $d = (i+1, j+1)$, $e = (i-2, j)$, $f = (i-1, j)$, $g = (i, j)$, $h = (i+1, j)$, $k = (i+2, j)$, $l = (i-1, j-1)$, $m = (i, j-1$, $p = (i+1, j-1)$, and $q = (i-2, j)$. See Figure 3 for these relationships.

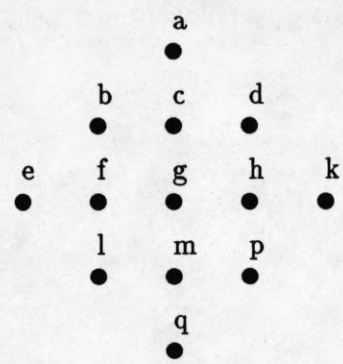


Figure 3

<u>*Case 1 (Interior) $i > j, j > 0$*</u>

$$
\begin{aligned}
V_f^{n+1} &\equiv 1 + min_u[p_{u_1} V_e^n + p_{u_2} V_l^n + p_{u_3} V_b^n + p_{u_4} V_g^n] \\
&\leq 1 + min_u[p_{u_1} V_l^n + p_{u_2} V_q^n + p_{u_3} V_g^n + p_{u_4} V_p^n] \\
&\equiv V_m^{n+1} \\
&\leq 1 + min_u[p_{u_1} V_b^n + p_{u_2} V_g^n + p_{u_3} V_a^n + p_{u_4} V_d^n] \\
&\equiv V_c^{n+1} \\
&\leq 1 + min_u[p_{u_1} V_g^n + p_{u_2} V_p^n + p_{u_3} V_d^n + p_{u_4} V_k^n] \\
&\equiv V_h^{n+1}
\end{aligned}
$$

where each inequality follows from the induction hypothesis for V^n applied at the four points *f,m,c,h.*

Case 2 (Diagonal) $i = j, i > 0$

$$\begin{aligned}
V_f^{n+1} &= V_m^{n+1} \\
&\equiv 1 + min_u[p_{u_1} V_l^n + p_{u_2} V_q^n + p_{u_3} V_g^n + p_{u_4} V_p^n] \\
&\leq 1 + min_u[p_{u_1} V_b^n + p_{u_2} V_g^n + p_{u_3} V_a^n + p_{u_4} V_d^n] \\
&\equiv V_c^{n+1} \\
&= V_h^{n+1}
\end{aligned}$$

where the inequality follows from the induction hypothesis for V^n applied at the four points f,m,c,h and Property S, and the two equalities follow from Property S directly.

Case 3 (Horizontal Axis) $i > 0, j = 0$

$$\begin{aligned}
V_f^{n+1} &\equiv 1 + min_u[p_{u_1} V_e^n + p_{u_2} V_l^n + p_{u_3} V_b^n + p_{u_4} V_g^n] \\
&\leq 1 + min_u[p_{u_1} V_l^n + p_{u_2} V_q^n + p_{u_3} V_g^n + p_{u_4} V_p^n] \\
&\equiv V_m^{n+1} \\
&= V_c^{n+1} \\
&\equiv 1 + min_u[p_{u_1} V_b^n + p_{u_2} V_g^n + p_{u_3} V_a^n + p_{u_4} V_d^n] \\
&\leq 1 + min_u[p_{u_1} V_g^n + p_{u_2} V_p^n + p_{u_3} V_d^n + p_{u_4} V_k^n] \\
&\equiv V_h^{n+1}
\end{aligned}$$

where the inequalities follows from the induction hypothesis for V^n at the four points f,m,c,h and Property S, and the equality follows from Property S directly.

Case 4 (Origin) $i = j = 0$

$$V_f^{n+1} = V_m^{n+1} = V_c^{n+1} = V_h^{n+1}$$

which follows directly from Property S.

This completes the proof of Proposition 6.2 since the point $g = (i, j)$ is in one of these four regions.

□

Proposition 6.3 *If for every $n = 0, 1, 2, ..., V^n$ has Properties S and O and the $lim_{n\to\infty} V^n = V^\infty$ exists, then V^∞ has Properties S and O.*

This is true since the inequalities in Property O are not strict.

Proposition 6.4 *For every n and for $x \in \mathbf{X}$, $V^{n+1}(x) \geq V^n(x)$.*

This is true since the cost per stage is nonnegative.

Since $V^n(\cdot)$ increases pointwise, $\lim_{n\to\infty} V^n = V^\infty$ exists, although $V^\infty(x)$ may be infinite for any particular x. However, this is not the case, as we will show in Proposition 6.6. Note that Proposition 6.3 shows that V^∞ has Properties O and S.

Proposition 6.5 *$V^\infty(x) = V^*(x)$ for all $x \in \mathbf{X}$.*

Proof. $V^n(x) \leq V^*(x)\, \forall x, n$ so $V^\infty(x) \leq V^*(x)$. But since V^∞ has Property O we have a minimizing u^* to solve $V^\infty = T(V^\infty)$, and so $\forall x$, $V^\infty(x) = E_x^{\mu^*}[\tau] \geq \inf_\mu E_x^\mu[\tau] = V^*(x)$. □

Next we will show that $V^*(x)$ is finite for all x so that we know our control policy is optimal (otherwise starting from x we would get a random walk which had infinite expected τ for any policy, and all policies would be equally bad).

Proposition 6.6 *For every $x \in \mathbf{X}$, $V^*(x) = V^\infty(x) < +\infty$.*

Proof. The proof is by drift analysis of the stochastic process $\{x_k\}$. Let $x_k = (i_k, j_k)$ be the location of the packet at time k. Using the Euclidean norm of the location as a Lyapunov function, $f(x) = \| x \|$, and doing a Taylor series bound on $f(x)$ to degree 2 using ∇f and $\nabla^2 f$, we can upper bound the drift. Without loss of generality (Property S) we fix x_k such that $i_k, j_k \geq 0, i_k \geq j_k$. Let $\theta_k = tan^{-1}(\frac{j_k}{i_k})$. Note that $0 \leq \theta_k \leq 45°$.

$$drift \;=\; E[\| x_{k+1} \| - \| x_k \| \mid x_k]$$

$$
\begin{aligned}
&\leq \; cos\theta_k(Ei_{k+1} - i_k) + sin\theta_k(Ej_{k+1} - j_k) + \frac{4}{\| x_k \|} \\
&= \; cos\theta_k(p_4 - p_1) + sin\theta_k(p_3 - p_2) + \frac{4}{\| x_k \|} \\
&\leq \; -\epsilon \text{ for large enough } \| x_k \|, for some \epsilon > 0
\end{aligned}
$$

since $cos\theta_k > 0$, $sin\theta_k \geq 0$, $p_4 < p_1$, and $p_3 \leq p_2$. This drift condition is called $(C1)$. We also note that $| \| x_{k+1} \| - \| x_k \| | \leq 1$ which gives us the necessary boundedness condition $(C2)$ along with condition $(C1)$ above to apply a theorem of Hajek[3], to infer

$$V(x_k) \equiv E_{x_k}^{\mu^*}[\tau] < +\infty$$

Actually [4] implies that we have a finite set containing $(0,0)$ which is positive recurrent, but since our chain is irreducible, the above fact is also a direct consequence. So the random walk starting at x_k with transition probabilities given by the Z^2 policy is expected to hit the destination $(0,0)$ in finite time. This gives $V(x) < +\infty, \forall x \in \mathbf{X}$ as desired. □

Since $V^* = V^\infty$ we know that V^* has Properties S and O, and hence we can see that the minimizing control policy is indeed the Z^2 policy, achieving the minimum u^* in equation (6.3). It is seen here that the primary goal is to move toward the destination and the secondary goal is to move toward the diagonal. The primary goal takes precedence when these two goals conflict. The four combinations of success/failure of these two goals represent the four directions the packet can request. In closing, we mention that a Model 3 can be considered. Suppose as in [1] that rejected packets are not deleted, but instead they retry after failing to leave a node in any time slot, waiting a delay W slots at the node before each retry. The DPE for this problem have exactly the same form as the DPE for Model 1 and 2 and the same analysis holds. So the Z^2 routing policy is optimal with respect to several possible measures of cost for deflection routing on an infinite grid.

7 Conclusion and Future Research

We have reviewed an optimal routing policy [1] for mesh topology networks and then extended this result for deflection routing. Some questions remain. First, it seems clear that the new result will

extend to 3-D networks (more dimensions seems unnecessary) and to finite grids. It would also be nice to generalize the three models enough to contain the result in [1] as a special case. While one would like to obtain the optimal policy for the torus or MSN, this seems unlikely to be of a simple closed form because the policy will obviously depend on (p_1, p_2, p_3, p_4). Finally, there is the question of "closing the loop". We solved the problem for a single packet with fixed link utilization probabilities. By fixing the network throughput, we can derive the link utilization probabilities, which will lead to some value of delay, and hence we can derive a non-linear relationship between delay, throughput, and the average number of customers in the system. We can get a throughput-delay curve for the network by solving (by iteration to a fixed point) this equation for fixed throughput. We can also find the throughput corresponding to a particular policy for a given reasonable traffic model and then try to show that the Z^2 policy has the maximum throughput. Subsequent work will attempt to show this is true.

References

[1] S. Badr and P. Podar. An optimal shortest-path routing policy for network computers with regular mesh-connected topologies. *IEEE Transactions on Computers*, 38(10):1362–1371, October 1989.

[2] D. Bertsekas. *Dynamic Programming.* Prentice-Hall, 1987.

[3] B. Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, 14:502–525, 1982.

[4] B. Hajek. Optimal control of two interacting service stations. *IEEE Transactions on Automatic Control*, AC-29:491–499, June 1984.

[5] N. Maxemchuk. Routing in the manhattan street network. *IEEE Transactions on Communications*, COM-35(5):503–512, October 1987.

Value Iterations for N x N Torus

N=18 (p1,p2,p3,p4)=(0.50, 0.50, 0.00, 0.00)
No Deflections
The positive quadrant of V*(i,j) is shown below with V*(0,0)
at the lower left corner

18.500	18.500	18.521	18.601	18.794	19.150	19.709	20.488	21.488	22.488
17.500	17.500	17.521	17.601	17.794	18.150	18.709	19.488	20.488	21.488
15.499	15.501	15.541	15.681	15.987	16.507	17.267	18.267	19.488	20.488
13.498	13.502	13.581	13.822	14.292	15.027	16.027	17.267	18.709	19.709
11.494	11.506	11.660	12.062	12.762	13.762	15.027	16.507	18.150	19.150
9.481	9.519	9.815	10.463	11.463	12.762	14.292	15.987	17.794	18.794
7.444	7.556	8.111	9.111	10.463	12.062	13.822	15.681	17.601	18.601
5.333	5.667	6.667	8.111	9.815	11.660	13.581	15.541	17.521	18.521
3.000	4.000	5.667	7.556	9.519	11.506	13.502	15.501	17.500	18.500
0.000	3.000	5.333	7.444	9.481	11.494	13.498	15.499	17.500	18.500

Property O holds.

N=18 (p1,p2,p3,p4)=(0.40, 0.30, 0.20, 0.10)
Deflection routing
The positive quadrant of V*(i,j) is shown below with V*(0,0)
at the lower left corner

37.442	37.594	38.180	39.216	40.610	42.261	44.114	46.040	47.795	48.795
35.948	36.126	36.771	37.885	39.394	41.211	43.211	45.230	46.999	47.795
32.852	33.069	33.846	35.177	36.963	39.050	41.270	43.458	45.230	46.040
29.306	29.580	30.531	32.107	34.151	36.480	38.917	41.270	43.211	44.114
25.578	25.938	27.127	28.999	31.316	33.865	36.480	39.050	41.211	42.261
21.696	22.202	23.733	25.966	28.571	31.316	34.151	36.963	39.394	40.610
17.592	18.383	20.416	23.081	25.966	28.999	32.107	35.177	37.885	39.216
13.078	14.505	17.278	20.416	23.733	27.127	30.531	33.846	36.771	38.180
7.667	10.718	14.505	18.383	22.202	25.938	29.580	33.069	36.126	37.594
0.000	7.667	13.078	17.592	21.696	25.578	29.306	32.852	35.948	37.442

WARNING: Property O FAILS to hold

N=18 (p1,p2,p3,p4)=(0.26, 0.25, 0.25, 0.24)
Deflection routing
The positive quadrant of V*(i,j) is shown below with V*(0,0)
at the lower left corner

321.544	322.207	324.109	326.972	330.408	333.986	337.292	339.966	341.715	342.334
319.642	320.371	322.440	325.533	329.210	333.000	336.478	339.274	341.090	341.715
313.987	314.909	317.488	321.265	325.654	330.094	334.092	337.257	339.274	339.966
304.477	305.755	309.264	314.257	319.894	325.436	330.312	334.092	336.478	337.292
290.673	292.603	297.703	304.652	312.178	319.323	325.436	330.094	333.000	333.986
271.591	274.794	282.658	292.665	302.895	312.178	319.894	325.654	329.210	330.408
245.100	251.174	264.045	278.767	292.665	304.652	314.257	321.265	325.533	326.972
206.135	220.029	242.337	264.045	282.658	297.703	309.264	317.488	322.440	324.109
140.449	180.252	220.029	251.174	274.794	292.603	305.755	314.909	320.371	322.207
0.000	140.449	206.135	245.100	271.591	290.673	304.477	313.987	319.642	321.544

WARNING: Property O FAILS to hold

Otmar S. Schlunk
EE 497
May 1990

An Analysis of Alternative Routing in Complete Networks with Unit Capacity Links

I. Introduction

Alternative routing is a routing scheme in circuit-switched networks wherein calls that would be blocked in a regular fixed routing scheme might be routed instead along alternative paths, enabling them to reach their destination. Such a scheme can under certain circumstances significantly reduce call blocking probabilities. When used injudiciously, however, it can seriously cripple a network. Therefore, a clear understanding of its use is essential. In this paper, various methods of analysis have been undertaken for the case of alternative routing in complete networks with unit capacity links in the hope of determining when and how to implement such a strategy.

II. The model:

The model used here for alternative routing on a complete network is as follows. First off, begin with a complete graph of N nodes. There is one link connecting every pair of nodes, i.e. N(N-1)/2 links in all. Calls arrive at some rate on the link between any two nodes (source and destination are irrelevant). The links have capacity one -- at most one call can be serviced on any link at any given time. In all cases if the link between these two nodes is free, the call will be accepted on this link. If this link is servicing a prior call, however, one of the following will happen. One might automatically drop the call. If this strategy is always employed, this is called routing without deflection or fixed routing. On the other hand one might try routing the call through some other 3rd party node or nodes (see fig. 1). Consider what is called a two-link deflected call. In such a call, one would route the call from one of the two original nodes through a link to a third node, and then through the link from that third node to the other original node. Such a call uses two links -- hence the name two-link deflected call (or deflected call of length two). Similarly, using more and more third party nodes, one could establish a connected path of maybe 3, 4, ... up to N-1 links. This is what is known as alternative routing -- routing that, given one's primary route is full, attempts to establish connection via some alternative route or routes.

The networks considered throughout this paper are completely symmetric. The arrival distribution per any given link is exponential with rate λ. All calls are serviced with exponentially distributed times of rate μ. Optimality throughout this paper is viewed in terms of maximizing the expected total

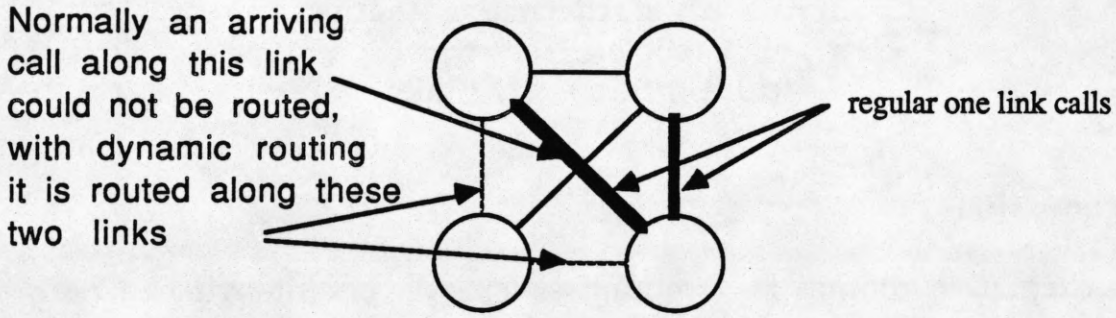


Figure 1. A pictorial explanation of dynamic routing.

longrun throughput achievable. This is equivalent to minimizing the probability of blocking (i.e. achieving a better probability of blocking).

A routing policy then is used to determine when to route an arriving call over k-links. There are two main categories of routing policies that will be examined. They are static and dynamic policies. When deciding whether or not to route a call, the former policies ignore the current state of the network, while the latter policies take some or all of the state information into account.

As a general rule, static policies should be simpler to implement than dynamic ones as they require less knowledge to make decisions. For similar reasons, they should in general be easier to analyze as well. On the other hand, one might pay a penalty if one uses a static policy as not all the available information is considered. In other words, static policies would seem unlikely to be optimal policies (it turns out that they might nevertheless be optimal, however). The main static policies considered in this paper are the following:

1) using fixed routing (i.e. never using an alternative route),

2) choosing one alternative path at random -- that is, if a call can not be directly connected, one two-link policy is chosen at random, and if available used to route the call,

3) finding in a random order the first available two-link alternative path -- this is really just policy two extended to looking at all N-2 possible two-link paths in random order and choosing the first available one encountered, and

4) using the shortest available path of up to k links in length -- this is really policy three extended -- after looking at all N-2 two-link calls in a random order, one looks through all three length calls, then four length calls, ... , up to k length calls.

All four policies were simulated with service rate of one on some small networks. The results are presented in figure 2. It is interesting to note that, for all the simulations run, alternative routing was only useful when the arrival rate was shorter than the service rate. For the cases where only two-link deflected calls were attempted, regardless of the number of such calls attempted, equality of blocking probability with the fixed routing case was reached at exactly where the arrival rate equaled the service rate. For the fixed routing case (mostly included to give an indication as to the error bounds of the simulations) the probability of blocking as expected (see section III) is independent of the number of nodes in the network. Under policy two, the probability of blocking begins to decrease somewhat with increasing N. This effect is even more marked under policy three. This leads us to question what happens asymptotically as N goes to infinity. Is there some curve it approaches or does it asymptotically go to zero for all arrival rates less than one?

Because of the large quantity of knowledge that can be conditioned upon, dynamic policies tend to be harder to categorize than static ones. Nevertheless, they will be looked at to some extent in section V.

The ultimate goal then in all this is to figure out the optimal policy -- when should a call be routed on an available k-link deflected path and when should it be thrown away. Unfortunately, the answer to the above remains an open problem (at least with regards to this paper). Some useful insights, though, have nevertheless been found.

Otmar S. Schlunk
EE 497
May 1990

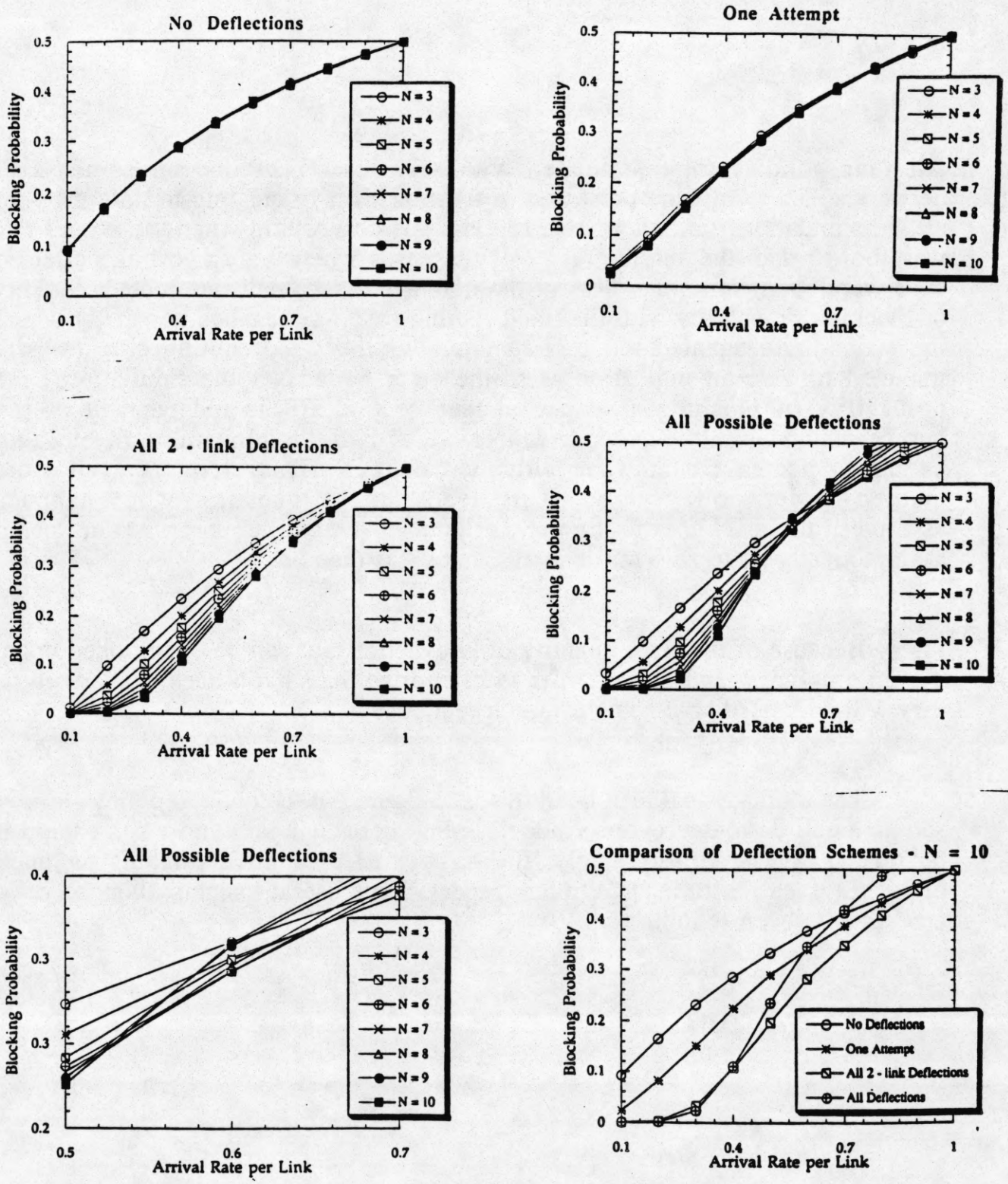


Figure 2. Simulations and comparisons of static alternative routing policies.

Otmar S. Schlunk
EE 497
May 1990

III. Analysis:

Various approaches were attempted to gain insight into this problem. The following are the the results of these attempts. For a fixed network, the blocking probability is readily determined.

$$P_{blocking} = \lambda / (\lambda+\mu). \qquad (1)$$

This is easily attained from standard Markov chain theory:

each link has two possible states, therefore,

$$P_{link\ is\ free} + P_{link\ is\ full} = 1,$$

and the flow between states must be conserved,

$$\lambda P_{link\ is\ free} = \mu P_{link\ is\ full}. \qquad (2)$$

Similarly, Markov theory can be employed to analyze dynamic routing schemes in a three node network (see fig. 3). In the case where all calls that can not be directly connected are connected along the alternative two-link path whenever available, one gets that the probability of blocking is

$$P_{blocking} = \frac{x^2(9+7x+x^2)}{2+8x+15x^2+8x^3+x^4} \qquad (3)$$

where $x = \lambda/\mu$. Figure 4 compares this against fixed routing, and it can be seen that dynamic routing is preferable, whenever $\lambda < \mu$.

Otmar S. Schlunk
EE 497
May 1990

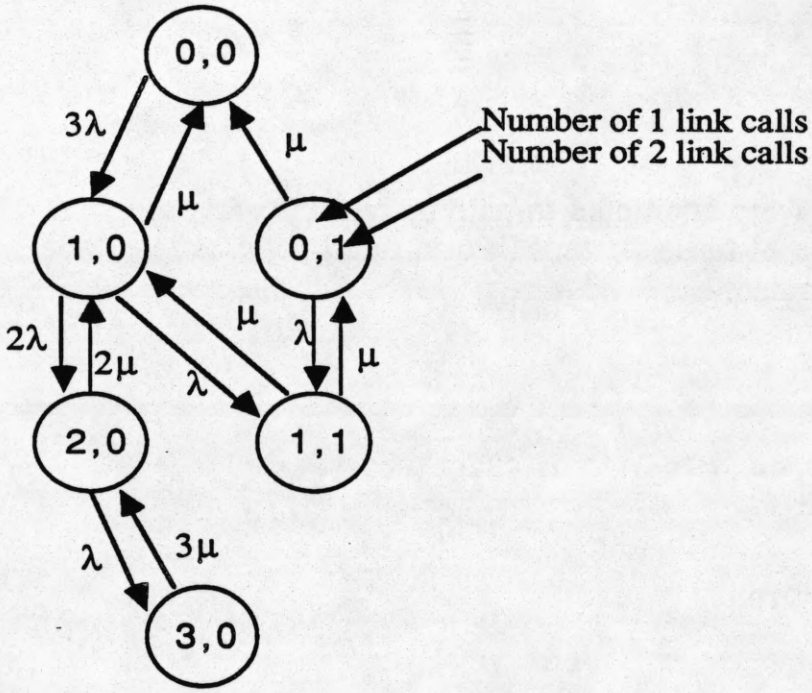


Figure 3. A Markov chain model for dynamic routing in a three node network.

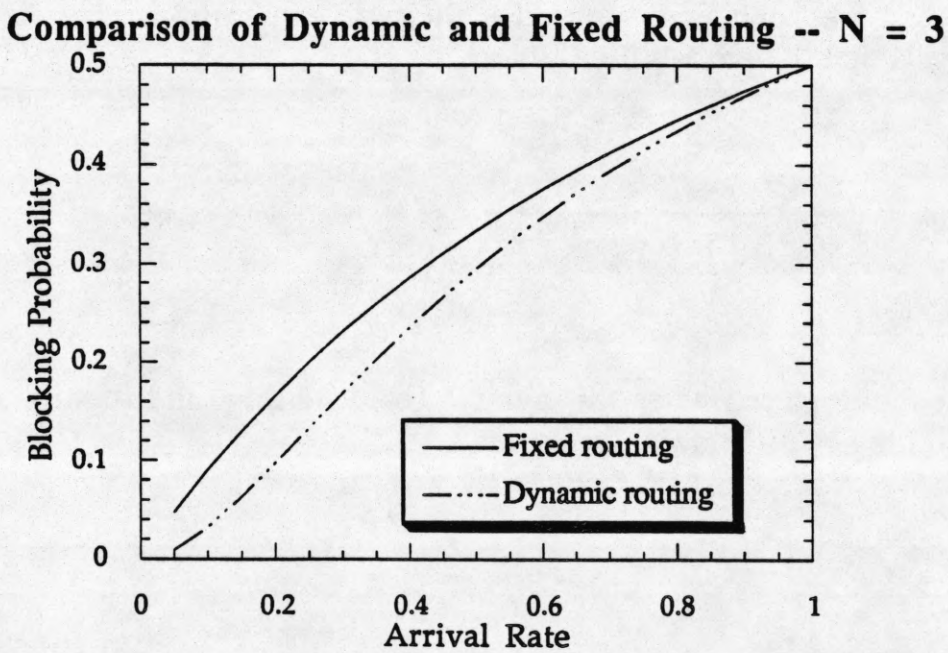


Figure 4. Dynamic versus fixed routing for a three node network.

Otmar S. Schlunk
EE 497
May 1990

Although the next result might seem intuitively obvious, it nevertheless needs to be firmly established. The following trick will come in handy. Consider a sequence $\{b\} = b_1, b_2, \ldots$ of exponentially distributed independent random service times. As calls arrive in the network they can take any one of the service times b_i not already taken by a previous call provided that no information regarding these times is used. If one knows that a service time is at least x time units long, but the residual time after the x units is unknown, this residual time is exponentially distributed with the same mean as the original service time and can therefore be used as a service time as well (provided one never uses the original service time). The same sequence of service times can then be used when comparing two different systems.

<u>Proposition 1</u>: Accepting direct one-link calls is always optimal.

Consider an arbitrary state of the network with at least one link free, at time t_0. Let $\{b\} = b_1, b_2, \ldots$ be a sequence of independent exponentially distributed times with mean $1/\mu$. Given a call is offered upon an open link in this state (call it link AB), the call may either be accepted or rejected. Assume for the sake of argument by contradiction that the optimal strategy in this case is not to accept the call. Call this system 1. Let us compare this case in terms of throughput (the criterion of optimality) to the case where one does accept the call (assign the call a service time b_j). Call this latter case system 2.

In both system, all previously chosen service times are equivalent. Also, all future arrivals will occur at the same exact points in time for both systems. While time is less than $t_0 + b_j$, assign every arrival in system 1 which is routed along a path which does not contain AB an unused service time in $\{b\}$ which is not b_j. Clearly any such call can be made in system 2 as well with the exact same service time -- as all these links have equivalent status in both systems. This guarantees these links remain equivalent in both systems. Now consider the first call that system 1 chooses to route through link AB. Say this call occurs at time t_1. If $b_j < t_1 - t_0$, then the accepted call has already finished in system 2. Both systems therefore have completely equivalent states at this point in time, and one can mimic the optimal policy of system 1 for the rest of time with system 2 by never choosing service time b_j for system 1 and matching service times in both systems. If the call in system 2 is still in progress, however, one can assign the residual service time of b_j to the system 1's new call. Thus,at time t_1, system 2 will no longer have any links full which system 1 has empty. Furthermore, by only accepting calls in system 2 which are accepted in system 1 and assigning them the exact same service time in the sequence of service

times as picked by system 1, one can ensure this always remains the case. By counting the throughput over both systems, one immediately sees that the throughput of system 2 will always be better or equal that of the optimal system -- system 1. This implies system 2 is optimal which further implies that accepting direct one-link calls is always optimal. Q.E.D.

This now firmly established leads to the question of when should a k-link call be made, and when shouldn't one. If one believes that if one can route a k-link call, it is always preferable to routing a call of length k', where k' > k; and if one further believes that if at a given moment it is optimal to throw away a k-link call, then at any other moment and place one should never consider routing calls of k or more links, then the following proposition implies that one need never consider routing a k-link call when $k > (\lambda+\mu)/\lambda$. The first assumption isn't a very large leap of faith. The latter, however, is far from clear. Nevertheless, through several simulations that were run, most of which are discussed in section V, no evidence to indicate the contrary was ever found.

<u>Proposition 2.</u> An otherwise nondeflecting policy which at one given instant in time is allowed to make a k-link deflected call is inferior in terms of average longrun throughput to a strictly nondeflecting policy whenever $k > (\lambda+\mu)/\lambda$.

Let system 1 and system 2 be identical, except that at some time t_0 system 1 is makes a k-link deflection. At all other times, however, only one-link calls are permitted in both systems. Define the net future gain in expected throughput of not routing this call versus routing it to be G_k. Then

$$G_k = E\{\text{future throughput system 2 - future throughput system 1}\}. \quad (4)$$

If this is greater than one, then routing the k-link call leads on average to a lower longrun throughput. If this is the case, not routing the call would be optimal.

The first thing to notice here, is that all links not along the k-link path can be considered identical in both systems, as having routed a k-link call elsewhere does not hinder or help any link in accepting calls in the future. Next, due to the symmetric nature of the network,

$$G_k = k\, E\{\text{future throughput on a single link among the k of system 2 - future throughput of the same link in system 1}\}. \quad (5)$$

Now define $H_k = G_k/k$.

Let us focus in on one of the k links along the path. If the k-link call manages to leave before any arrival on the link, $H_k = 0$. With probability $\lambda/(\lambda+\mu)$, however, a call will arrive on this link before the k-link call is removed from system 1. If this happens, the new call can be routed in system 2 for a throughput of one call, whereas in system 1 it must be thrown away. After accepting the call in system 2 then, both systems will have a call on the link and equal service rates. From this point on, the expected throughput on the link will be equal in both systems, since they have equal probabilities of finishing first and equal equivalent expected gains given their job finishes first. Therefore, in this case, $H_k = 1$. Thus one gets

$$H_k = \mu/(\lambda+\mu) \cdot 0 + \lambda/(\lambda+\mu) \cdot 1 = \lambda/(\lambda+\mu) \tag{6}$$

which gives us

$$G_k = k\,\lambda/(\lambda+\mu). \tag{7}$$

As mentioned above then, if $G_k > 1$, not routing the k-link deflected call is the optimal choice. Q.E.D.

IV. Fixed-point approximations

Fixed-point approximations are used throughout networking to rapidly determine estimates of otherwise virtually impossible numbers to calculate. Unfortunately, fixed-point approximations themselves tend to be hard to analyze. For example it tends to be quite difficult if not impossible to prove the uniqueness of a solution. And even if a unique solution does exist, proving convergence to it or even getting the approximation to converge in practice, is often far from easy.

A. History

Many fixed point expressions exist for dynamic alternate routing schemes. All seem to have the assumption that the probability of one link in the network being full is independent of the probability of other links in the network being full. Furthermore, all are geared towards networks with large capacity links (and often different capacity links), such as phone trunk lines. Below is a small sampling of what is currently available. Only [1] analyzes complete networks and mentions the idea of finding alternative paths in random orders.

In Lin *et al* [4], analysis is performed on the alternate routing scheme of originating-office control with spill-forward used in the European network AUTOVON. Originating-office control simply means that the originator of a call knows what his order of alternate path choices is. If his first choice is unavailable the second is tried. This continues till there are no more choices left. The spill-forward part means that if a node has very few neighbors and not a lot of different paths, it can allow a couple adjacent nodes to take over the originating-office function of trying to find an available path. The fixed point model for this is essentially straightforward. Assume all links are independent of one another. The fixed point works by trying to find the probabilities of various links being available, then the probability that paths using them are attempted/accepted. Iterations are then performed over all these factors from all possible source-destination pairs and their routing tables till everything converges.

Another model is presented in Kuczura and Bajaj[3]. They employ what they call a three-moment method. The general method involves determining overflow from the Erlang loss function, and redistributing this overflow over the alternate routes to minimize the total network overflow. Then the probability of blocking is estimated on a point to point basis by assuming independent blocking in the links. Their results supposedly are sufficiently accurate for engineering purposes (whatever that means).

In [1], Gibbens discusses various aspects of dynamic routing in circuit-switched networks. Among his contributions is a fixed point analysis on a trunk reservation scheme. In such a scheme alternative calls are blocked if a certain fraction of the capacity of a link is full. He also provides some insight into what this fraction should be to best guarantee traffic under worst case conditions and performs some asymptotic analysis on the result. Simple bounds on loss networks are also discussed in the paper, and numerous simulation results are given.

Finally, in [2], Kelly looks at these networks from a somewhat different perspective. His main analysis is focused on fixed networks, though some extensions to alternative routing are presented. He looks at the network from the perspective of how the net gains are affected by incremental changes in the flow along the routes. Using this idea, he suggests nodes can, in a distributed fashion, follow an adaptive routing scheme, changing with the changes in network demands.

B. Fixed point for attempting all possible two-link deflections

All the above fixed points assumed independence between the status of links. A fixed point using some knowledge of dependence can however be developed for the case where link capacities are one.

Let us consider the static policy three as defined in section II. Define A1 to be the equilibrium probability of accepting an offered call as a one-link call and P1 to be the equilibrium probability of a link having a one-link call on it. Similarly, let A2 be the probability of accepting an offered call as a two-link call, and P2 be the probability of a link being full with a two-link call on it. Finally, for convenience, let

$$P = P1 + P2, \tag{8}$$

which is simply the probability a link is full. Then iterating the following four equations produces a fixed point:

$$\begin{aligned} A1 &= 1 - P \\ A2 &= P1\left(\sum_{k=0}^{k=N-2}\binom{N-2}{k}(1-P)^k P^{N-2-k}(1-P^k)\right) \\ &\quad + P2\left(\sum_{k=0}^{k=N-3}\binom{N-3}{k}(1-P)^k P^{N-3-k}(1-P^k)\right) \\ P1 &= \lambda\, A1 \\ P2 &= 2\,\lambda\, A2. \end{aligned} \tag{9}$$

The A1 equation is obvious as it stems from the probability an arrival meets with an open link. The A2 equation separates whether an arrival that couldn't be routed met with a one-link call or a two-link call. If the former, it checks for the probability of k free links to third parties that exist from one of the two original nodes. By assuming that the blocking probability of each individual link is still P, this can be determined via a binomial distribution. Then by assuming that the probability of the link from a reachable third party node back to the other original node is full with probability P as well, one gets for k free links, one or more two-link paths will be available with probability $1 - P^k$. If the new call was blocked by a two-link call, one does virtually the same thing. In this case though one starts at the center node of the call that is blocking the new one. At most N-3 possible connections (instead of N-2) can be made from this side. Otherwise the concept is the same. The probability a link is full with a one link call is simply the arrival rate multiplied by the probability of accepting a call as a

one-link call. Similarly the probability of a link being full with a two-link call is simply twice the probability of accepting a call as a two-link call multiplied by the arrival rate, where the factor of two comes in since every accepted call takes up two links.

When these values have converged, the probability of blocking is simply the probability that you are not accepted as a one or a two link call. That is

$$P_{blocking} = 1 - A1 - A2 \qquad (10)$$

Fixed points were calculated for some of the networks simulated in section II (service rate μ was one) and were compared with the simulations (see fig. 5). On the whole they faired quite well. The predicted results seemed to slightly lower bound the simulations (though by never much more than about 3% probability of blocking). As arrival rates neared one, the predicted results converged to the simulations and met at the point 1/2 probability of blocking for an arrival rate of one. It should also be noted that this fixed point was closer than one which employed no information at all.

A similar fixed-point expression can be obtained for static policy two of section II. The only difference is in the probability of accepting a two-link call. In this case, for simplicity one can set

$$A2 = P\,(1 - P)^2. \qquad (11)$$

The results for this case are given in figure 6. One could also have chosen

$$A2 = (1 - P)^2(P - P2/(N-2)), \qquad (12)$$

making use of the fact that one knows the probability the call was blocked initially due to two-link call, and under those circumstances, 1 out of N-2 random attempts will automatically be blocked.

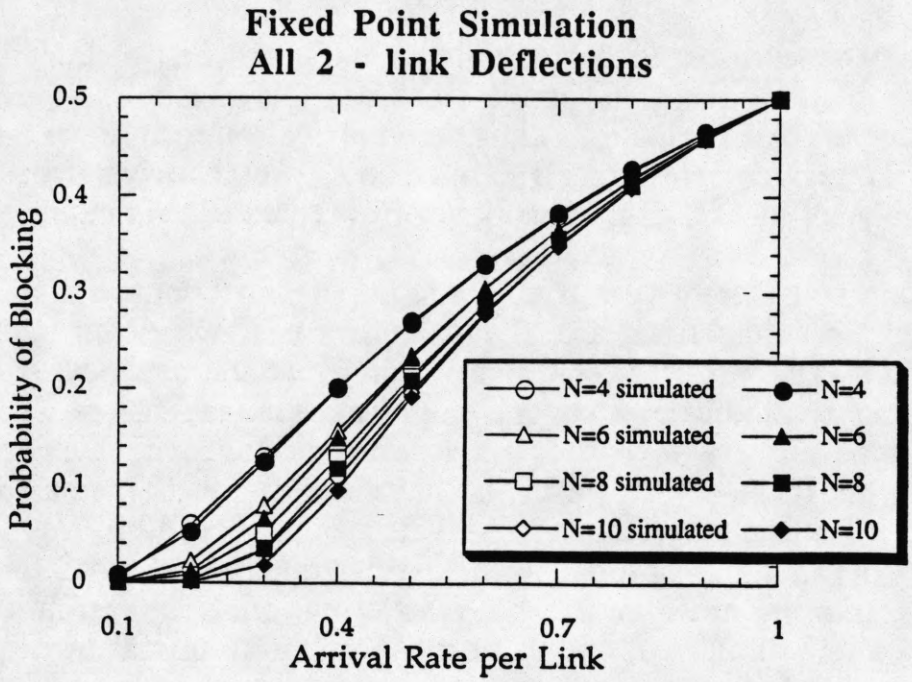


Figure 5. Fixed-point simulation for policy three.

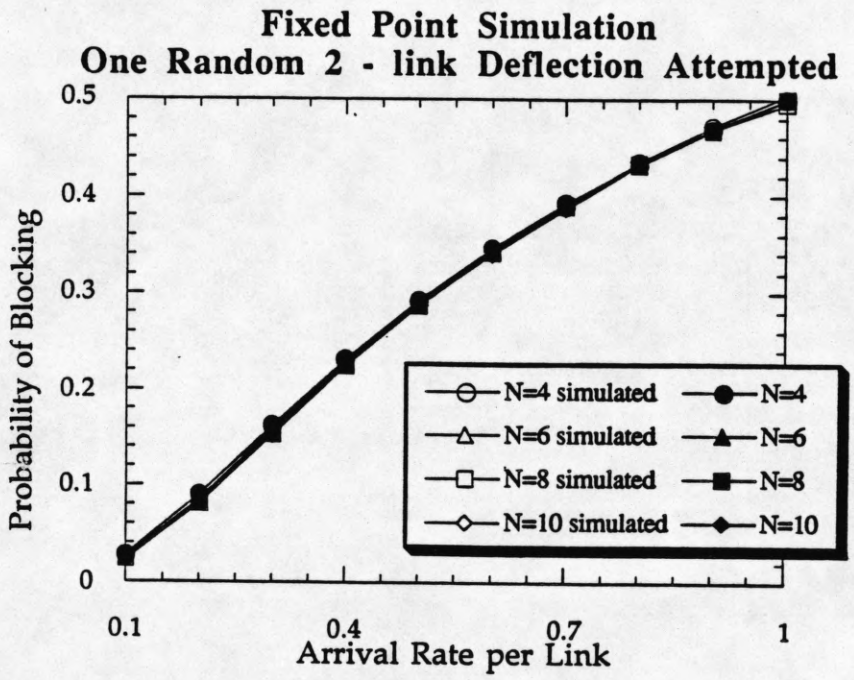


Figure 6. Fixed-point simulation for static policy two.

Otmar S. Schlunk
EE 497
May 1990

V. Optimizing strategies

One of our main concerns is to decide when a deflected call of k links should be accepted. In an attempt to gain some insight into this question, several simulations were conducted. All simulations considered arrival rates in the range from .1 to 1, with unit service rates. In all cases 100000 total arrivals were tested for their probability of blocking on networks of from three to ten nodes.

The first question considered was, of all static policies that route calls on k or fewer links (shorter connections are always preferred), whenever it is possible to do so, which k is optimal (i.e. which k optimizes static policy 4 in section II). The results here indicate what one might expect (see figure 7) -- longer length deflections at very low rates provide better long run average throughput. Somewhat unexpectedly, however, is that virtually all the gains are gotten in the two-link deflection stage. And, as arrival rates increased towards 1, the two-link strategy became the optimal such static strategy. So what this seems to indicate is that in most cases, by restricting one's attention to optimizing over all two-link strategies, one can probably attain optimal or near optimal results.

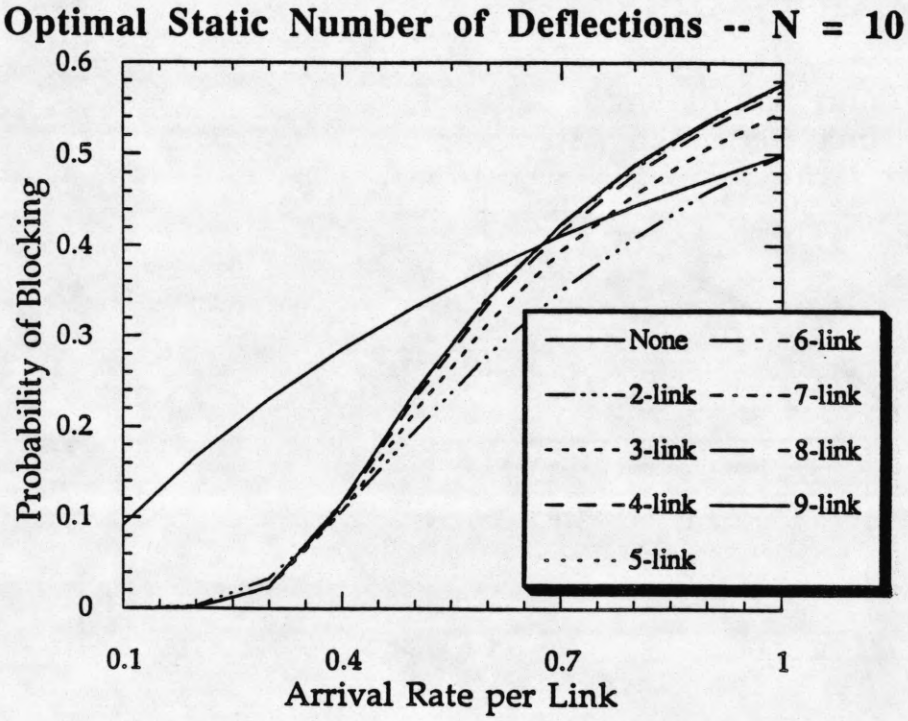


Figure 7. Finding the optimal value of k for static policy 4.

Given these results, the main concern seemed to be optimizing over all possible two-link strategies. Several possible strategies were tested. The first was to only route two-link deflected calls when x or fewer links in the system were full (see figure 8). In these tests, gains were had as expected at low values of x (only systems which had shown improvements under two-link static policies were considered). I was quite frankly expecting a point at which the system might be so full though that allowing more deflected calls into the system might decrease performance. This never seemed to happen. One possible reason is that perhaps the system rarely achieves a state where this would be the case, so that any losses that might be occurring are virtually unnoticeable. Another possibility is that such losses occur when N becomes large. Or maybe, such losses just never occur. An interesting study might be to artificially start the system at a really full level, and measure total throughput over some short period of time thereafter for each of two systems -- one where one does accept a two-link call as opposed to one where one does not. Then over several thousand such tests, perhaps a more accurate conclusion could be drawn.

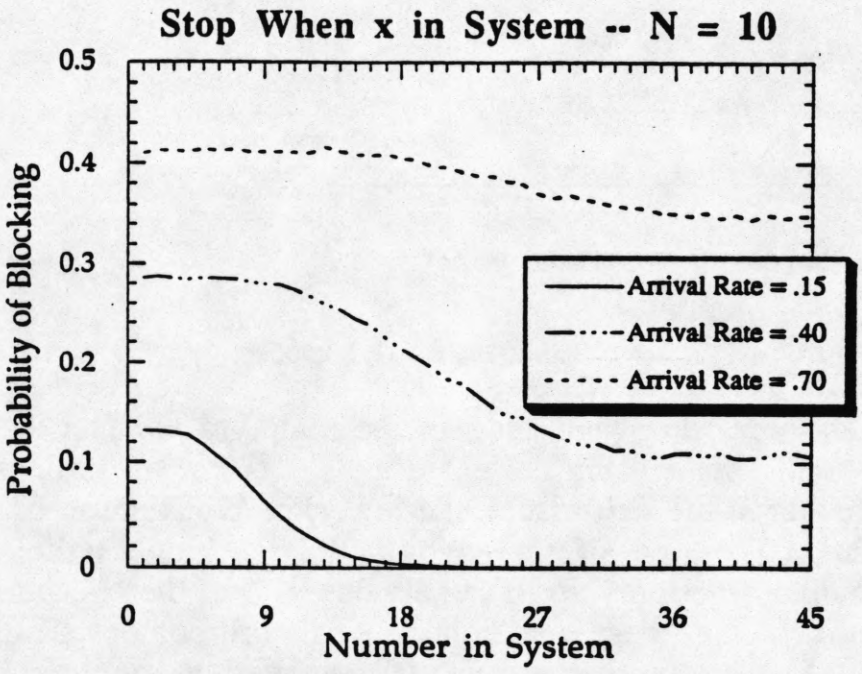


Figure 8. Dynamic policy of only routing two-link deflected calls when x or fewer links are full in the system.

Next, a test was conducted to limit the number of two-link deflected calls permitted into the network at any given moment. The results were similar to those of the previous experiment (see figure 9). As more two-link calls were allowed to enter the network, the probability of blocking steadily decreased, eventually reaching a saturation point. Beyond this point, admitting more deflected calls did not seem to affect the network much one way or the other. Once again, it is not clear if this plateau in the blocking probability is due more to the fact that a nearly full network of deflected calls virtually never arises, or that the probability of blocking due to accepting calls when the network is nearly saturated with two-link calls remains virtually unchanged.

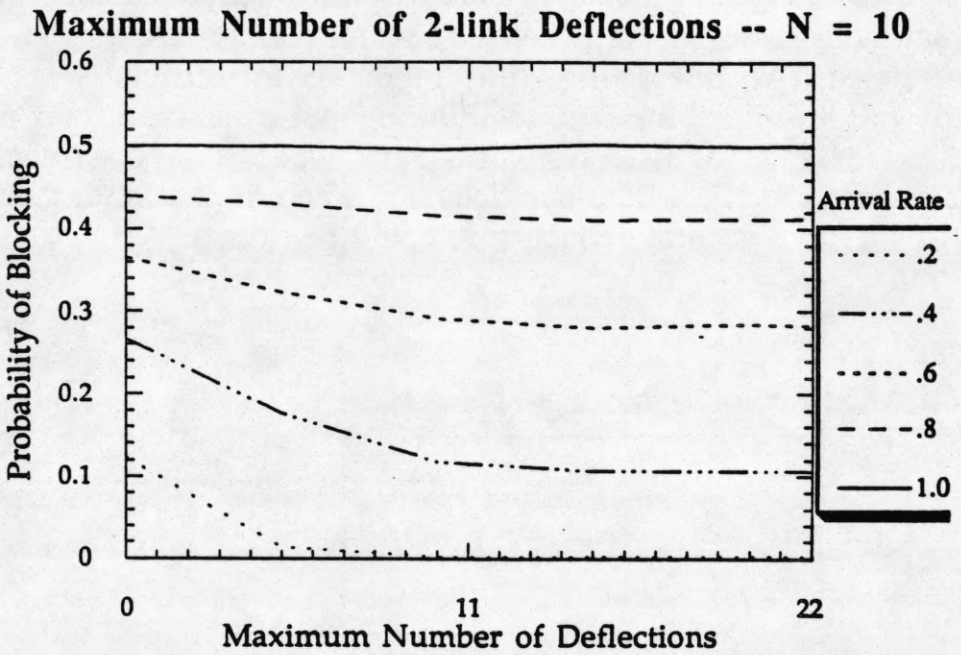


Figure 9. Permit a maximum of x deflected calls into the network at once.

Two more experiments that were conducted, seemed to indicate that the more deflected calls tried (when using deflected calls was known to decrease the probability of blocking in the first place), the lower the blocking probability that could be achieved. The first of these compared the static policies of attempting calls in random order only up to some value of k of the N-2 possible two-link deflected paths. If after these k attempts the call still couldn't be made, it was thrown out. The results (see figure 10) showed a monotonically decreasing probability of blocking with increasing values of k. The final experiment involved only routing two-link deflected calls if x or fewer neighboring links were full on both the source's and the destination's sides. As in the last experiment, the more calls routed, the better the probability of blocking achieved (see figure 11).

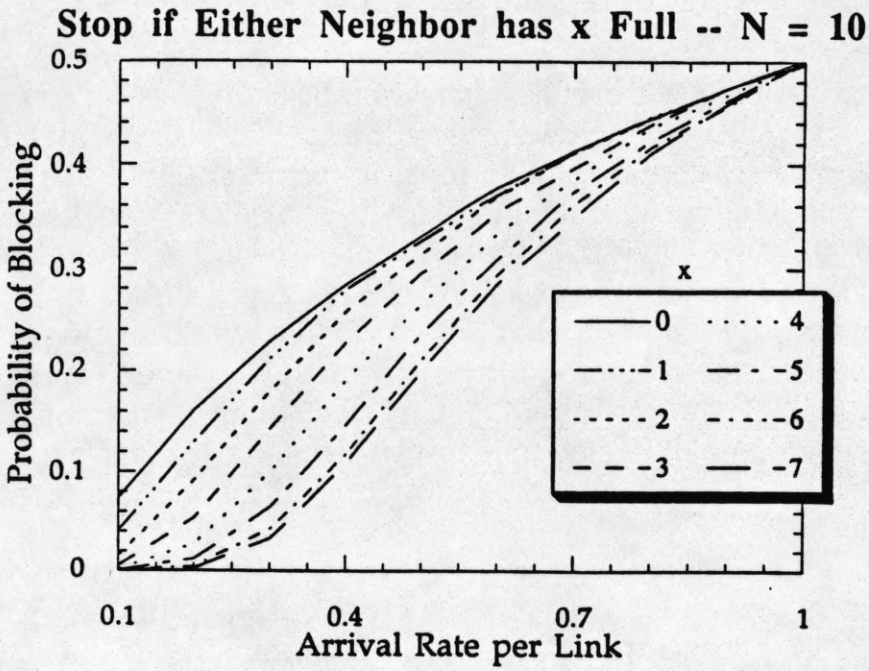


Figure 10. Look at up to k two-link alternate paths.

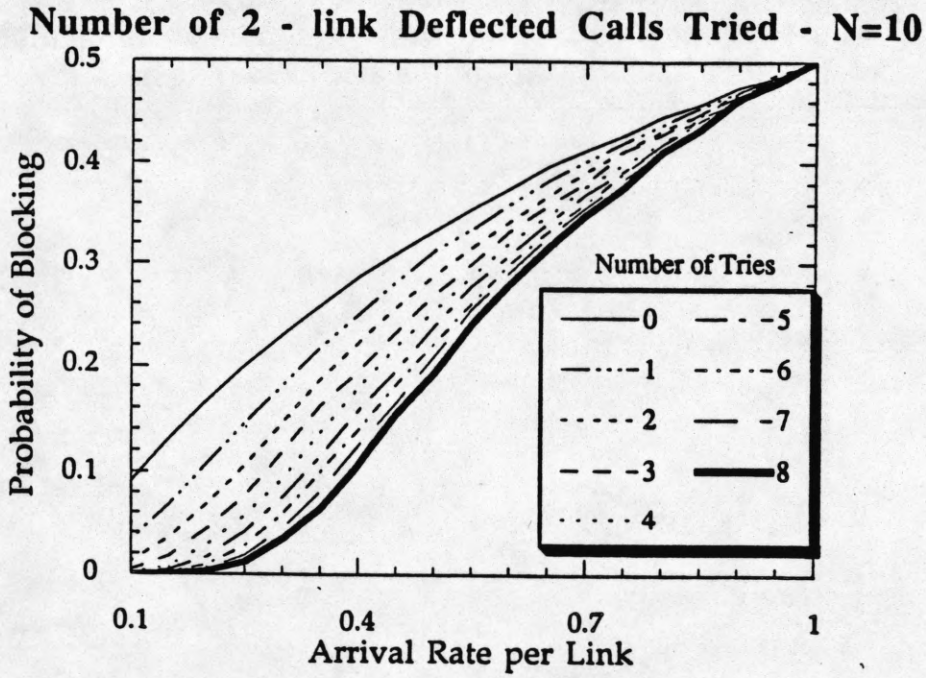


Figure 11. Accept only if fewer than x calls on both neighbors side links.

VI. Conclusion

Unfortunately solving problems when dealing with alternative routing is quite difficult, even when dealing with such drastic simplifications as symmetric networks and unit capacity links. However, some basic ideas seem to have come forth. From the previous sections simulations, it seems possible that if a k-link call is optimal at some point in time and place in a network, a k-link call will always be optimal whenever it is the shortest possible call that could be routed. This would imply that a static approach of always accepting calls up to some value k of links is actually the optimal policy for this network. In any case, it seems reasonably probable that a static policy is almost always very close to an optimal strategy. Another interesting discovery was that restricting the maximum length of an allowable deflected call to two does not, except perhaps under some very strict blocking requirements at low arrival rates, seem to hurt performance very much.

Although some progress has been made, a great deal is still left to be discovered and proven. Can, for example, the conjectured optimality of the static policies be proven or disproven? Can an arrival rate threshold be found for when to route calls of k-links? What happens if the symmetry of the network is removed? Can closer performance bounds or approximations be found? etc ... The answers to these questions are probably not simple, but they may very well be important. They should not only increase our understanding of this simple network, but that of perhaps other much more complicated networks as well.

Otmar S. Schlunk
EE 497
May 1990

References

[1] R.J. Gibbens, "Dynamic Routing in Circuit-Switched Telecommunications Networks", Rayleigh Prize Essay, University of Cambridge.

[2] F.P. Kelly, "Routing in Circuit-Switched Networks: Optimization, Shadow Prices and Decentralization", *Adv. Appl. Prob.*, Vol. 20, No. 2, pp.112-144, 1988.

[3] A. Kuczura and D. Bajaj, "A Method of Moments for the Analysis of a Switched Communication Network's Performance", *IEEE Trans. Comm.*, Vol. 25, pp. 185-193, Feb. 1977.

[4] P.M. Lin, B.J. Leon, and C.R. Stewart, "Analysis of Circuit-Switched Networks Employing Originating-Office Control with Spill-Forward", *IEEE Trans. Comm.*,Vol. 37, No. 16, pp. 754-765, June 1978.

On Optical Orthogonal Codes

Harlan Russell

For EE497
Professor B. Hajek
Spring 1990

INTRODUCTION

The rate of optical-to-electrical and electrical-to-optical conversions is limited by the speed of the electronic components, so much of the bandwidth available in a fiber optic link will not be utilized if the signaling rates are limited to the rates that can be achieved by current electronic devices. One solution is to perform many of the signaling functions with optical components. For example, the receiver can consist of optical components rather than first converting the optical signals to electrical signals and using a conventional receiver.

One method proposed by Salehi et. al. [1, 2, 3] to increase the signaling rates for fiber optic based networks is to use fiber-optic code division multiple-access (FO CDMA). This allows low information rate electrical signals to be converted to high rate optical pulse sequences. Figure 1 (taken from [1]) shows the system for a source and a destination using an optical encoder and decoder. One possible implementation of a network using FO CDMA is shown in Figure 2 (taken from [1]). Each user's input optical signal is combined in a optical star coupler so that all the input signals are received by each user. One signaling strategy (often referred to as receiver-directed signaling) is to use the optical sequences as addresses or signature sequences. Each user has its own sequence and to send data to that user, the information is coded on that user's specific address sequence (or signature sequence). This type of signaling allows the network to achieve random, asynchronous communication access which is free of network control. Good sequences must satisfy two conditions: each sequence can be easily distinguished from a shifted version of itself and each sequence can be easily distinguished from (a possibly shifted version of) every other sequence in the set.

OPTICAL ORTHOGONAL CODES (OOC)

A detailed development of the signature sequences is presented in References 1 and 2. An optical orthogonal code C is a family of (0, 1) sequences and is described by parameters $(F,K,\lambda_a,\lambda_c)$ (or (F, K, λ) if $\lambda_a = \lambda_c$). Each sequence has length F (or has F chips) and has weight K (or has K pulses). The sequences must satisfy the above two conditions which are equivalent to the following two conditions. Assume that the sequences are extended periodically.

The Auto-correlation Property: For any sequence $x = (x_n) \in C$

$$\left|\sum_{n=0}^{F-1} x_n x_{n+l}\right| = \begin{cases} K & l = 0 \\ \leq \lambda_a & 1 \leq l \leq F-1 \end{cases}$$

Cross-Correlation Property: For any pair of sequences $x = (x_n) \in C$ and $y = (y_n) \in C$

$$\left|\sum_{n=0}^{F-1} x_n y_{n+l}\right| \leq \lambda_c \qquad 0 \leq l \leq F-1$$

The fiber-optic communications system is modeled as a positive system; hence, the sequences consist of 0's and 1's only. This corresponds to the receivers using noncoherent detection techniques (i.e., power measurements). The signals cannot be manipulated to add to zero. Therefore, codes based on +1/-1 sequences (such as for direct-sequence spread spectrum) would not be well suited for this model. In [1, 2] the authors assume $\lambda_a = \lambda_c = \lambda = 1$, that is, two sequences will overlap in at most one position. We will also consider codes for $\lambda = 2$. More general codes are considered in [3]. Two typical OOC with $\lambda = 1$ are shown in Figure 3 (taken from [1]).

Properties for OOC are developed in [1, 3], including methods for constructing codes given $(F, K, \lambda_a, \lambda_c)$. Of particular interest are bounds on the number of codewords, Φ, that can be found for a given family. The bounds are

$$\Phi(F,K,\lambda) \leq \frac{(F-1)\cdots(F-\lambda)}{K(K-1)\cdots(K-\lambda)}$$

$$\Phi(F,K,\lambda_a,\lambda_c) \geq \frac{\binom{F}{K} - \frac{F-1}{2}\binom{K}{\lambda_a+1}\binom{F}{K-\lambda_a-1}}{n \cdot \sum_{i=\lambda_c+1}^{\min\{F-K,K\}} \binom{F-K}{K-i}\binom{K}{i}}$$

$$\Phi(F,K,\lambda_a,\lambda_c) \geq \frac{\lambda_c(F-K+1) - (\lambda_c/\lambda_a)(K-1)^2(K-2)}{K(K-1)^2}$$

Graphs for $F = 1000$ and $1 \leq \lambda \leq 3$, as K varies are shown in Figures 4-6. Notice that, except for when $\lambda = 1$, both lower bounds are essentially 0 for the parameters considered here. In [3] the authors claim that the upper bound is particularly strong for small values of λ. In fact, they prove that when $\lambda = 1$ the upper bound is achieved for many values of K. For the system considered here, each receiver is assigned a unique codeword, so N, the number of users, can be at most Φ, the number of codewords. Derivations of these bounds can be found in [3]. We present the bounds here for the purpose of determining valid ranges for N, F, K, and λ.

PROBABILITY DENSITY FUNCTION FOR TWO INTERFERING OOC

The probability density function for two interfering OOC (with $\lambda = 1$ only) is developed in [1]. A density function is developed for three cases: chip synchronous, strong chip asynchronous, and weak chip asynchronous. For the chip synchronous model, at each receiver the chips are aligned, but, the frames are not assumed to be synchronized. The chip asynchronous models do not assume that the chips will be aligned; the strong and weak models concern the patterns of overlaps between codewords that are allowed and are precisely defined in [1]. It is shown that the chip synchronous model is an upper bound for the actual performance and that the weak chip asynchronous model is a lower bound for the actual performance (the performance is measured in terms of the bit error probability, which will be discussed in the following sections). For the remainder of this paper we will consider the chip synchronous model only. The chip synchronous model is more tractable, whereas the chip asynchronous models are not as well developed.

Consider the probability that two codewords from the same OOC overlap. Since each codeword has K pulses, there are K^2 ways to overlap two pulses. Because $\lambda = 1$, only two pulses can overlap for any particular offset of the two codewords. Hence, there are K^2 offsets that result in an overlap of one pulse. Each codeword has F chips and we assume the offset between two codewords is distributed with equal probability for each offset so that the probability for each overlap is $1/F$. Thus, the probability that two codewords overlap in one position is K^2/F. The probability that the two codewords do not overlap is $1 - K^2/F$. Furthermore, the two codewords will overlap in more than one position with probability zero. We say a *hit* occurs if two pulses overlap.

ON–OFF FO–CDMA

The network is assumed to consist of N transmitter-receiver pairs, and it is assumed that all transmitters are in use. Figure 2 shows one possible implementation. The performance is evaluated by the bit error probability where the only source of performance degradation is from the presence of other users. The effects of quantum and thermal noise are neglected. Also, the optical energy level from all users that occur at the same time are assumed to be additive.

Each transmitter uses on-off keying. When the data bit is 1 the desired codeword is transmitted, and when the data bit is 0 nothing is transmitted. Each user supplies a continuous stream of data bits with the value of each bit being equally likely. Each receiver has an ideal optical correlator (or matched filter) that is matched to its assigned codeword. Figure 7 shows a typical receiver. Notice that the receiver integrates the correlated sequence over the entire sequence length, not on a chip by chip basis. For user 1, the output of the correlator is $Z_1 = b_1K + I_1$ where b_1 is the desired data bit and I_1 is the interference experienced by receiver 1 from the N - 1 interfering transmitters.

For this paper we assume that if Z_1 is less than K then the output data bit is a 0, otherwise the output is a 1. If the desired data bit is a 1 then an error cannot be made (in [2] it is implicitly assumed that the receiver achieves and maintains perfect synchronization). An error can only be

made when the desired data bit is a 0 and there are K or more hits from the N - 1 interfering transmitters. Notice that for this receiver and channel model the distribution of hits over the codeword duration is not important, only the total number of hits. An example of an error is shown in Figure 8. The desired data bit is a zero, the weight of the code is 4, and there are 4 hits. Thus, the receiver incorrectly outputs a 1.

User 1's codeword	1000000001001000000000000000010000
Received sequence	3000000000001000000000000000000000

Figure 8. Example of an error when user 1's received bit should be 0. The received sequence is after the correlator.

The conditional probability of i hits given N - 1 interfering transmissions is binomial and is given by

$$P_h(i\,\text{hits}\,|N-1\,\text{trans})=\binom{N-1}{i}\left(\frac{K^2}{2F}\right)^i\left(1-\frac{K^2}{2F}\right)^{N-1-i}$$

Notice that the probability that a particular interfering transmission hits the desired sequence is now $K^2/2F$ since on-off keying is used (the interfering user is present with probability 1/2[1]). The probability of a bit error for the desired user is

$$P_E=\frac{1}{2}\sum_{i=K}^{N-1}P_h(i|N-1)$$

The factor of 1/2 is included because an error can occur only if the desired data bit is a 0.

OPTICAL HARD-LIMITER

A second model for the fiber optic channel is considered in [2] by placing an optical hard-limiter before the receiver. The hard-limiter is defined as

[1] In [2] the implicit assumption is that, if present, interference from a particular user will be present during the desired user's entire frame, and the interference is independent between frames. But, because the frames are not synchronized, part of two frames of an interfering user will overlap the desired frame. The two interfering frames are independent, so, the interference may be present during only part of the desired frame. This also introduces dependence in the bit error probability between frames of the desired user.

$$g(x)=\begin{cases}1, & x \geq 1\\ 0, & x = 0\end{cases}$$

An example of a receiver using the hard limiter is shown in Figure 9. In effect the hard-limiter reduces the effect of the interference by limiting the interfering power during any particular chip. Consider Figure 10 with the same desired sequence and received signal as in Figure 8. If the data bit for user 1 is 0 then the hard-limiter prevents an error because the total interference power is now only 2 instead of 4. Notice that for the receiver with the hard-limiter, the distribution of the number of hits is important. An error is made only if there is a hit in all of the pulses used by the desired sequence. The number of hits in each chip is not needed.

User 1's codeword	1000000001001000000000000000010000
Received sequence	3000000000001000000000000000000000
Limited sequence	1000000000001000000000000000000000

Figure 10. The hard-limiter prevents an error event by limiting the received power.

In [2] an approximation for the bit error probability is presented and it is claimed to be an upper bound. We show it is not an upper bound, and we will present an upper bound. The probability that a bit error occurs is the probability that all of the pulses of the desired sequence are hit and that the desired data bit is a 0. The probability that all pulses are hit can be expressed as

$$P_E = P(A)P(B|A)P(C|A,B)\cdots$$

were $P(A)$ is the probability that one of the pulses is hit by 1 or more of the N - 1 interfering users, $P(B \mid A)$ is the probability that a second pulse is hit by 1 or more interfering users given that pulse A is hit, $P(C \mid A, B)$ is the probability that a third pulse is hit by 1 or more interfering users given that both pulses A and B are hit, etc., so that there is a term for each of the K pulses. Notice that $P(B \mid A)$ depends on the number of interfering users that hit pulse A. A user that hits pulse A cannot hit pulse B because $\lambda = 1$. The bounds use the fact that $P(B \mid A) \leq P(B \mid A_1)$ where A_1 denotes the event that pulse A was hit exactly once, and a similar approximation is used for the remaining terms. When the probability that A is hit by more than one user is very small (i.e., when $P(A) \approx P(A_1)$), $P(B \mid A) \approx P(B \mid A_1)$ and the approximation is very good. The bit error probability is

$$P_E \le P(A)P(B|A_1)P(C|A_1,B_1)\cdots$$

Define $P(A_i)$ as the probability that A is hit by exactly i users. The probability that an interfering user does not hit a particular pulse is given by

$$q = 1 - \frac{K}{2F}$$

So, the probability that a pulse is hit 1 or more times is $P(A) = 1 - q^{N-1}$. Given that A is hit once, the probability that the next pulse is hit is

$$P(B|A_1) = 1 - \left(1 - \frac{K}{2(F-K)}\right)^{N-2}$$

This follows because $\lambda = 1$ implies a codeword can hit the desired sequence in one place only. Also, since exactly one codeword can hit the desired pulse no other codeword can hit that desired pulse. An interfering codeword can hit the desired pulse with any one of its K pulses. So of the F possible offsets, K of them cannot occur because of the conditioning on A_1. The upper bound is

$$P_E \le \frac{1}{2}\prod_{m=0}^{K-1} 1 - \left(1 - \frac{K}{2(F-mK)}\right)^{N-1-m}$$

The bound in [2] incorrectly assumes that given A is hit once, the probability that the next pulse is hit is $P(B \mid A_1) = 1 - q^{N-2}$. Thus, the bit error probability is approximated by

$$P_E \approx \frac{1}{2}\prod_{m=0}^{K-1}\left(1 - q^{N-1-m}\right)$$

In the next section, we calculate the exact bit error probability, and calculations on Mathematica show that the exact calculation is in fact greater than the upper bound given in [2], though the two values are very close. For example, for a (1000,5,1) code with 50 users the claimed upper bound is 8.4 10^{-6}, while the exact bit error probability is 8.5 10^{-6}. The upper bound presented here is 8.8 10^{-6}.

CALCULATION OF PROBABILITY OF BIT ERROR

We found that it is possible to make an exact calculation for the bit error probability when the hard-limiter is used. Because $\lambda = 1$, two sequences can overlap in one position only, making the hits independent. The probability of a bit error is

$$P_E = \frac{1}{2}\sum_{i=K}^{N-1} P(i \text{ hits occupy all } K \text{ pulses})$$

$$= \frac{1}{2}\sum_{i=K}^{N-1} P(\text{all } K \text{ pulses hit} \mid i \text{ hits in } K \text{ pulses}) \cdot P(i \text{ users hit the } K \text{ pulses})$$

The probability $P(i$ users hit the K pulses) is simply the probability of i hits given N - 1 interfering users, $P_h(i \mid N\text{-}1)$, calculated above for the receiver without the limiter. The probability that all K pulses are hit given i hits is equivalent to the probability that K bins are occupied by i balls. The solution to the occupancy problem can be found in [4], for example. The probability that all bins are occupied given i balls and K bins is

$$p_0(i,K) = \sum_{v=o}^{K} (-1)^v \binom{K}{v} \left(1 - \frac{v}{K}\right)^i$$

Thus, for the hard-limiter, the bit error probability is

$$P_E = \frac{1}{2}\sum_{i=K}^{N-1} p_0(i,K) P_h(i \mid N-1)$$

The bit error probabilities for both the receiver without the limiter and the receiver with the hard-limiter are evaluated for some specific parameter values and are shown in Figure 11. Notice that the receiver with the hard-limiter performs significantly better. Figure 12 compares the exact bit error probability, the upper bound, and the approximate bit error probability from [2]. There is very little difference between the probabilities. The exact bit error probability is slightly greater than the approximate bit error probability.

LARGER WEIGHT CODES

For a fixed number of users and a fixed sequence length, lower bit error probabilities can be achieved as K, the weight of the code, is increased. For example, Table 1 shows the bit error probabilities for the two receivers considered in this paper with 10 users and a sequence length of 1000. But, the value of K is limited because $\lambda = 1$. Referring back to Figures 4-6, to increase the number of codewords (and correspondingly the number of users) while holding F fixed, λ will need to be increased. For example, to support 50 users, K is limited to 5 when $\lambda = 1$. But, if $\lambda = 2$, K can be as large as 28. Unfortunately, the bit error probabilities developed here all assume that $\lambda = 1$, that is that two sequences overlap in at most one position. When $\lambda = 2$, two sequences can overlap in 1 or 2 positions.

Table 1. Bit error probability with $N = 10$ and $F = 1000$.

K	P_E(hard-limiter)	P_E(no limiter)
1	$2.25\ 10^{-3}$	$2.25\ 10^{-3}$
3	$8.39\ 10^{-7}$	$3.75\ 10^{-6}$
5	$7.20\ 10^{-10}$	$1.84\ 10^{-8}$
7	$5.69\ 10^{-13}$	$9.13\ 10^{-11}$
9	$1.37\ 10^{-16}$	$1.47 10^{-13}$

Using the previously developed equations, some approximations can be easily made for the bit error probability when $\lambda = 2$. A simple lower bound for both systems is to assume that, even though $\lambda = 2$, each codeword will overlap in one position only. That is, the equations from above are used for larger values for K. Of course, this bound becomes worse as K increases since more of the sequences will overlap in 2 places. A pessimistic approximation that can be used for both systems is to double the number of interfering users and assume each user is independent and can hit the desired sequence in 1 position only. In the above equations N is replaced by $2N - 1$. For the system with the hard-limiter, a pessimistic assumption can be made by assuming each of the N users overlaps with the desired sequence in 2 positions, and the overlaps are independent. This can be thought of as having each user hit the desired sequence with the same probability as in the λ

= 1 case, but each hit results in two independent balls. In actuality the balls are not independent since they cannot both occur in the same position. For this case, the bit error probability is approximated by

$$P_E \approx \frac{1}{2}\sum_{i=K}^{N-1} p_0(2i,K)P_h(i|N-1)$$

An upper bound can be formulated for the system without a limiter. Assume that all hits result in an overlap in 2 positions. Since an error event can occur only if the number of hits is greater than or equal to K, and the distribution of the hits is not important, an error event can occur if half as many transmissions hit the desired signal as in the previous case. The upper bound is given by

$$P_E \leq \frac{1}{2}\sum_{i=\lceil K/2 \rceil}^{N-1} P_h(i|N-1)$$

Figures 13-14 show these bounds for both the system without a hard-limiter and the system with a hard-limiter. The number of users is taken to be 50, and the sequence length is 1000. The solid black square in both graphs is the exact bit error probability for the (1000,5,1) OOC with 50 codewords. This is the largest value of K that can be used with 50 codewords. For the larger values of K, λ must be 2. For the system without a limiter, the lower bound indicates that not much improvement is possible in the bit error probability when K is increased. However, for the system with the hard-limiter, the graph indicates that the bit error probability may indeed be lower for some values of K. The graph indicates that further investigation into the bit error probability for codes with $\lambda = 2$ is warranted.

MODEL FOR BIT ERROR PROBABILITY WHEN $\lambda = 2$

Next, we consider the bit error probability when $\lambda = 2$. Two sequences can overlap in 1 or 2 positions. The total number of ways for an overlap to occur is K^2. Let the number of offsets that result in only one overlap between the two codewords be k_1 and the number of offsets that result in two overlaps be k_2. Because each overlap between two particular pulses occurs only

once, $K^2 = k_1 + 2k_2$. When using on-off keying, the probability that a codeword overlaps exactly once with the desired sequence is $p_1 = k_1/2F$ and the probability that they overlap in two positions is $p_2 = k_2/2F$. For the system without the limiter, the probability of a bit error is the probability that the number of hits is greater than K, and this is simply the multinomial distribution given by

$$P_E = \frac{1}{2} \sum_{i+2j \geq K}^{i+j \leq N-1} \binom{N-1}{i,j} p_1^i p_2^j (1 - p_1 - p_2)^{N-1-i-j}$$

Unlike in the case when $\lambda = 1$, the bit error probability for the system with the hard-limiter is not an easy generalization from the system without the limiter. When two hits occur, they are not independent. The second hit cannot occur in the same location as the first hit. An approximation is to assume that the hits are independent, but, this may not be good when K is small. The approximation is

$$P_E = \frac{1}{2} \sum_{i+2j \geq K}^{i+j \leq N-1} p_0(i+2j,K) \binom{N-1}{i,j} p_1^i p_2^j (1 - p_1 - p_2)^{N-1-i-j}$$

The values for k_1 and k_2 are, in general, not easy to determine. They depend on the two particular sequences that are overlapping. One possibility is to determine average values for k_1 and k_2 for an OOC. A best case estimate can be made by taking $k_1 = K^2$ and $k_2 = 0$ and a worst case estimate by taking $k_1 = 0$ and $k_2 = K^2/2$. We believe that a better estimate of the system performance can be made with these approximations than is available with the bounds in the previous section.

CONCLUSIONS

We have examined a signaling technique for a fiber optic based network. A user's low rate data stream is encoded onto a high rate optical sequence. Each sequence has good auto-correlation and cross-correlation properties so that random, asynchronous communication can take place free of network control. A special class of sequences, called optical orthogonal codes, are examined and shown to be suitable. The sequences examined in [1, 2] are limited to the class where only one overlap occurs between two sequences (i.e., $\lambda = 1$). Two different types of receivers are

considered: a optical correlation receiver and the same receiver with a hard-limiter placed at its input. To determine the performance of these receivers using these sequences and on-off keying, the bit error probability is calculated. The bit error probability for the receiver without the limiter is developed and an approximation for the receiver with the hard-limiter is given in [2]. We have shown that the exact bit error probability for the receiver with the hard-limiter can be calculated. We have also investigated codes that allow the overlap between two sequences to be 2 instead of 1 (i.e., $\lambda = 2$). It is shown that for the receiver without the hard-limiter, the performance is not likely to improve by considering sequences with $\lambda = 2$. But, for the receiver with the hard-limiter it may be possible to improve the bit error probability by using larger weight codes. We have proposed a model for calculating the bit error probability for codes with $\lambda = 2$, but use of this model is limited by the fact that probabilities are dependent on the particular codewords used. Some approximation techniques are suggested that may provide further insight.

ACKNOWLEDGEMENT

The author would like to thank Colin Frank, Arvind Krishna, and Upamanyu Madhow for their many constructive suggestions.

REFERENCES

[1] J. A. Salehi, "Code Division Multiple-Access Techniques in Optical Fiber Networks—Part I: Fundamental Principles," *IEEE Trans. Commun.*, vol. 37, pp. 824-833, Aug. 1989.

[2] J. A. Salehi, C. A. Brackett, "Code Division Multiple-Access Techniques in Optical Fiber Networks—Part II: System Performance Analysis," *IEEE Trans. Commun.*, vol. 37, pp. 834-842, Aug. 1989.

[3] F. R. K. Chung, J. A. Salehi, V. K. Wei, "Optical Orthogonal Codes: Design, Analysis, and Applications," *IEEE Trans. Inform. Theory*, vol. 35, pp. 595-604, May 1989.

[4] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 1, edition 3, New York: John Wiley & Sons, 1968.

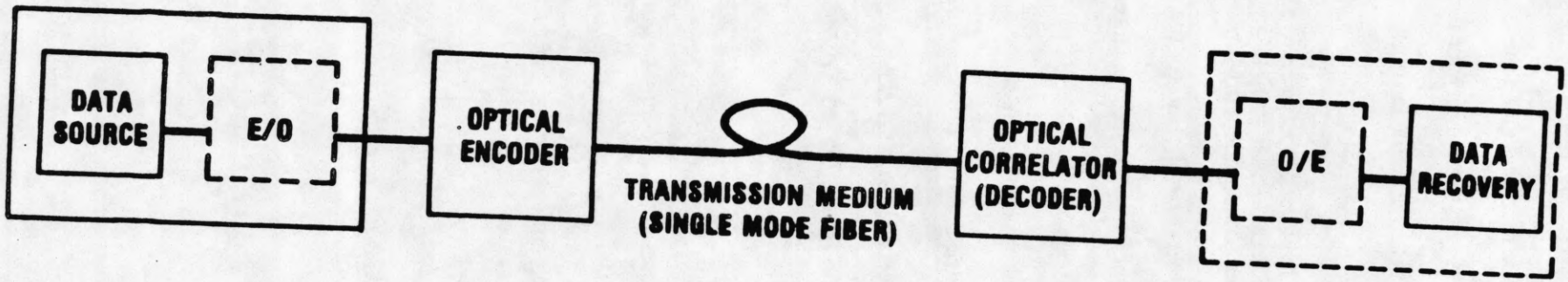


Figure 1: Fiber Optics Communication System Using Optical Encoder and Decoder (Correlator)

taken from Reference 1

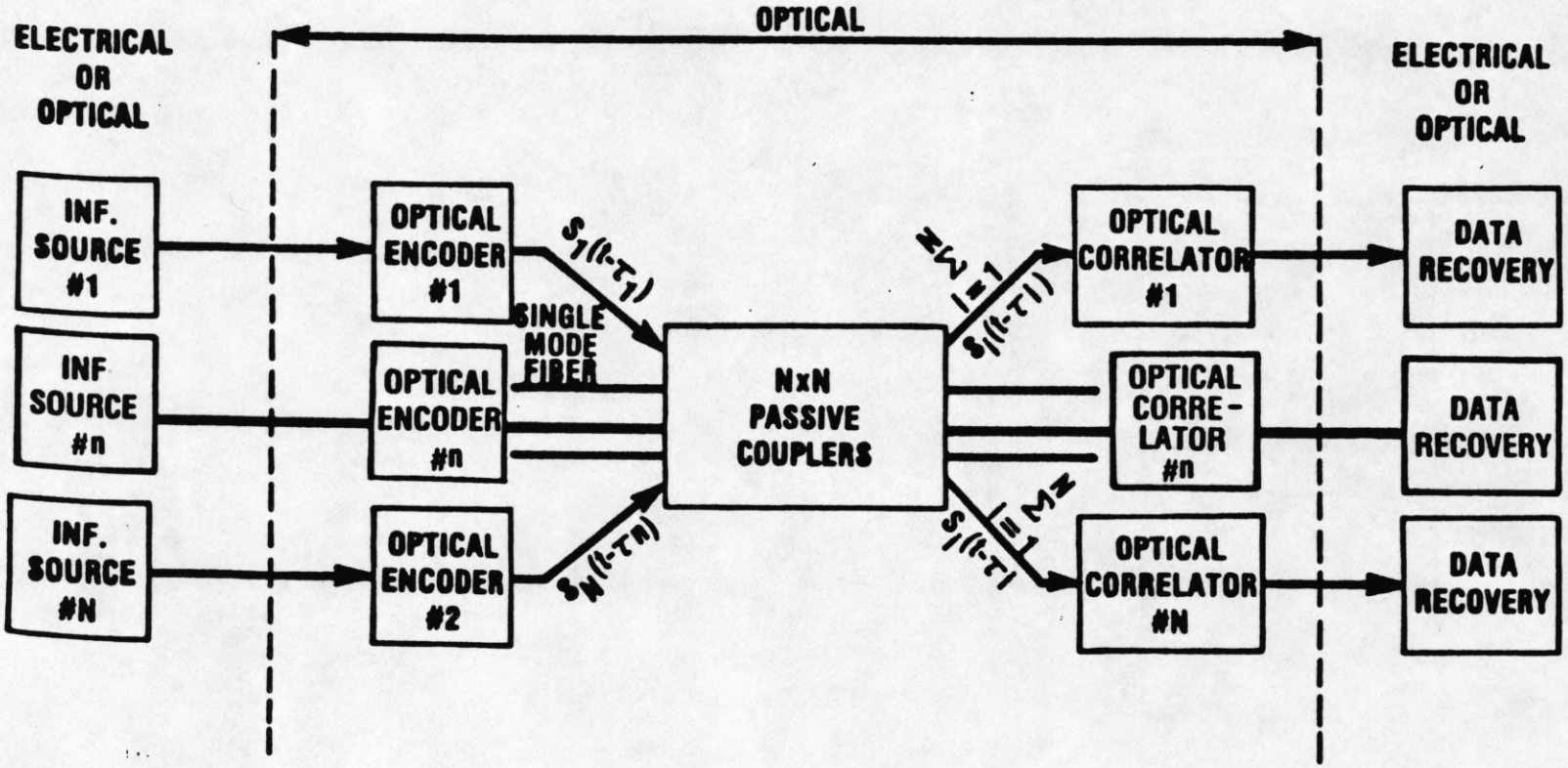


Figure 2: A Typical Fiber Optics Code Division Multiple Access (FO-CDMA) Network in a Star Configuration

taken from Reference 1

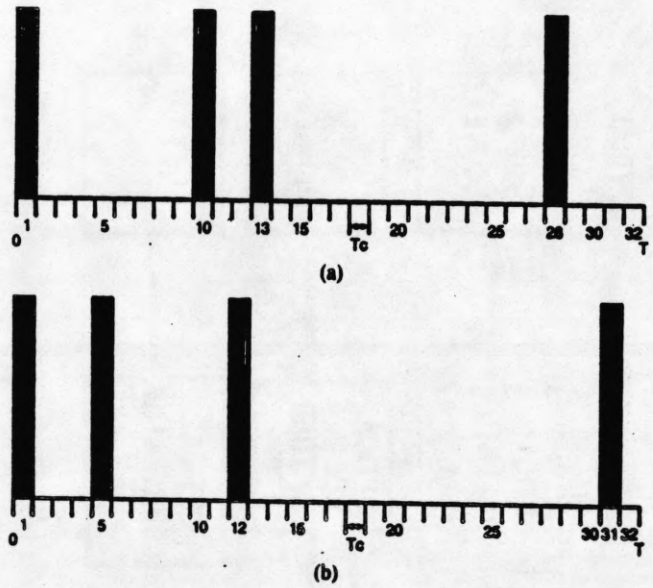


Fig. 3. Two "optical orthogonal codes" with length $F = 32$, $K = 4$, and $\lambda_a = \lambda_c = 1$. (a) First code is represented by placing a pulse at the 1st, 10th, 13th, and 28th chip positions. Here, we designate this code as 9, 3, 15, 5. (b) Second code is represented by placing a pulse at the 1st, 5th, 12th, and 31st chip positions; the code is designated as 4, 7, 19, 2.

taken from ~~Figure~~ Reference 1

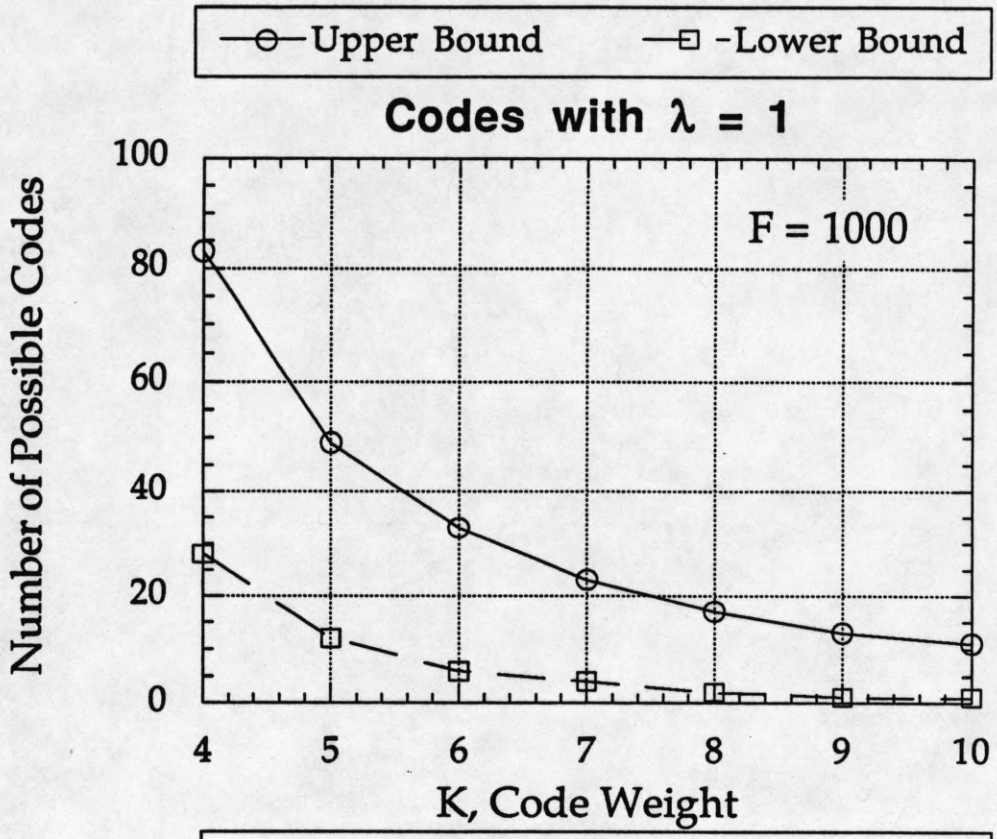


Figure 4

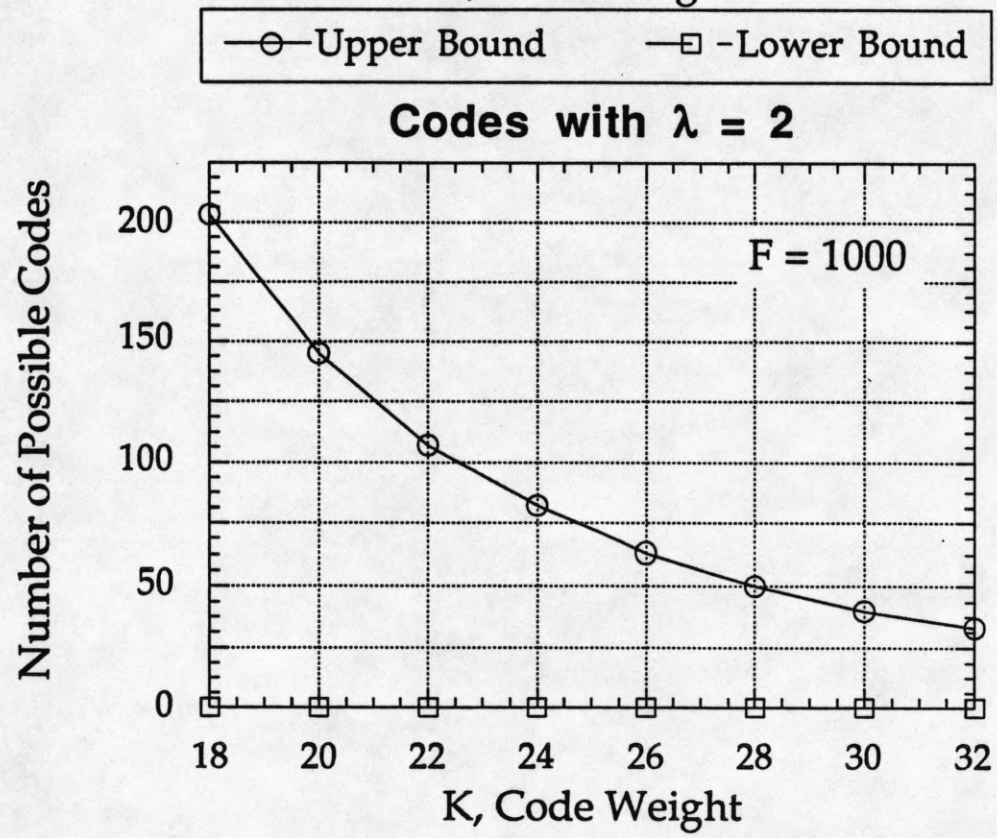


Figure 5

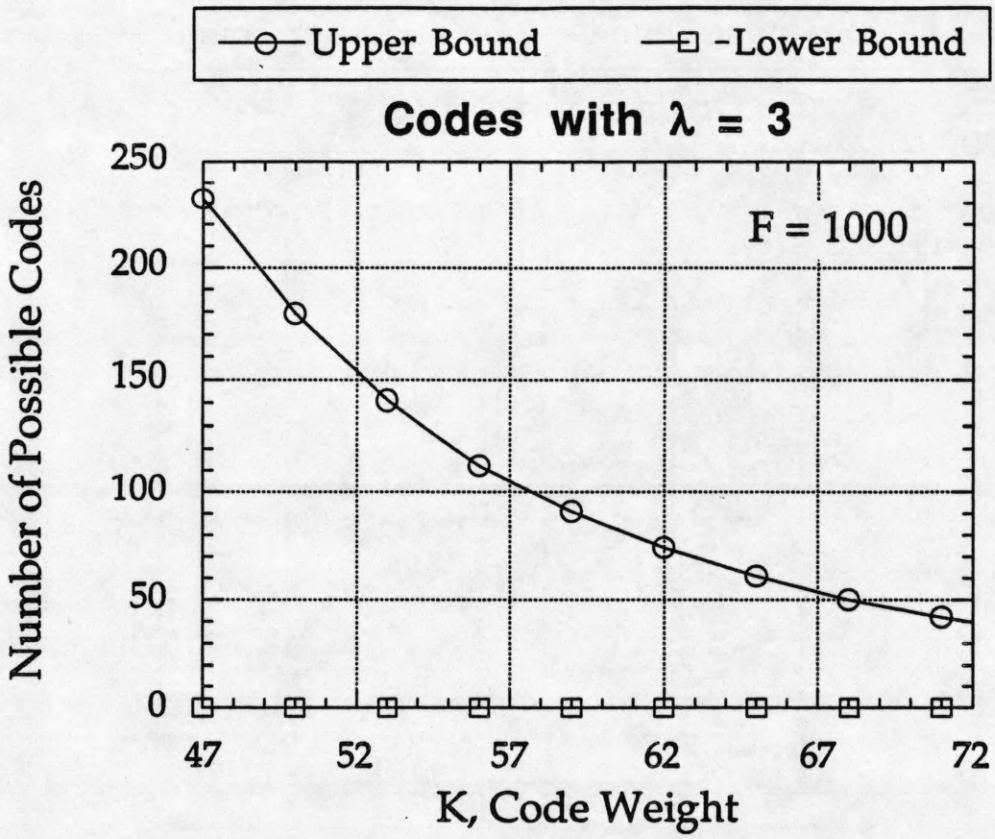

Upper Bound
Lower Bound
Codes with λ = 3
F = 1000
Number of Possible Codes
250
200
150
100
50
0
47
52
57
62
67
72
K, Code Weight

Figure 6

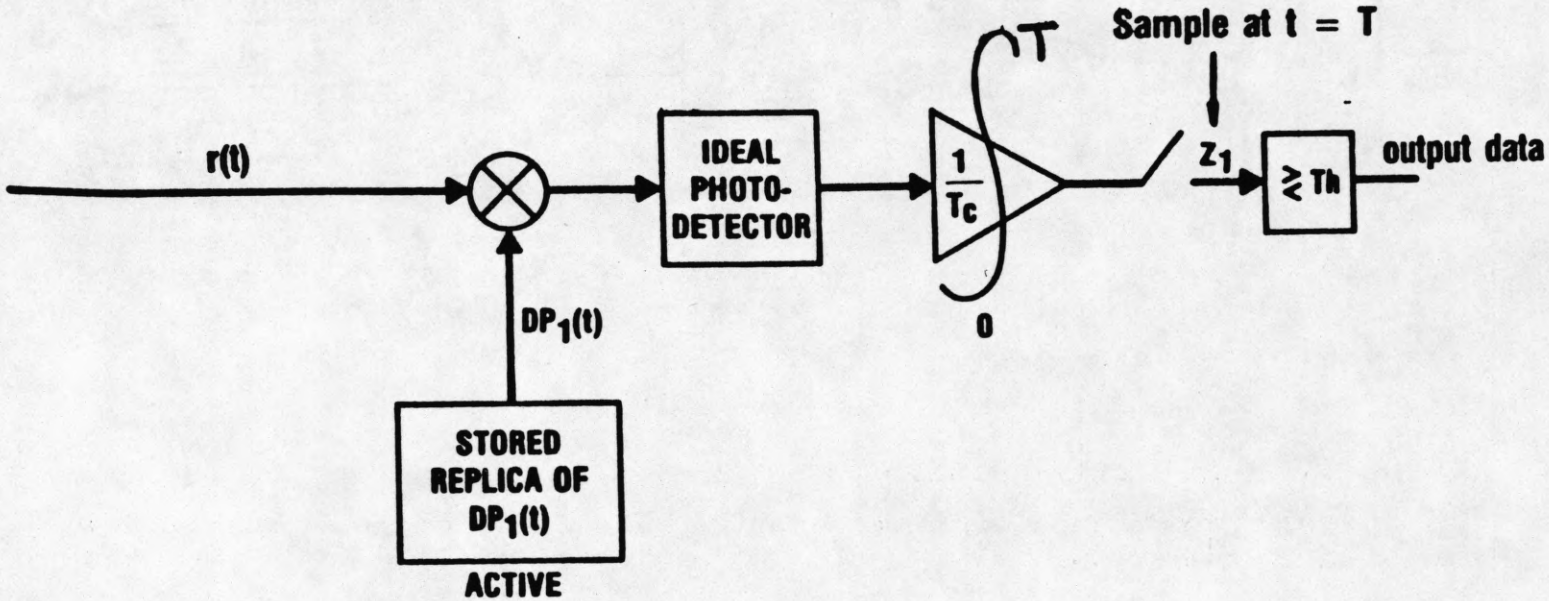
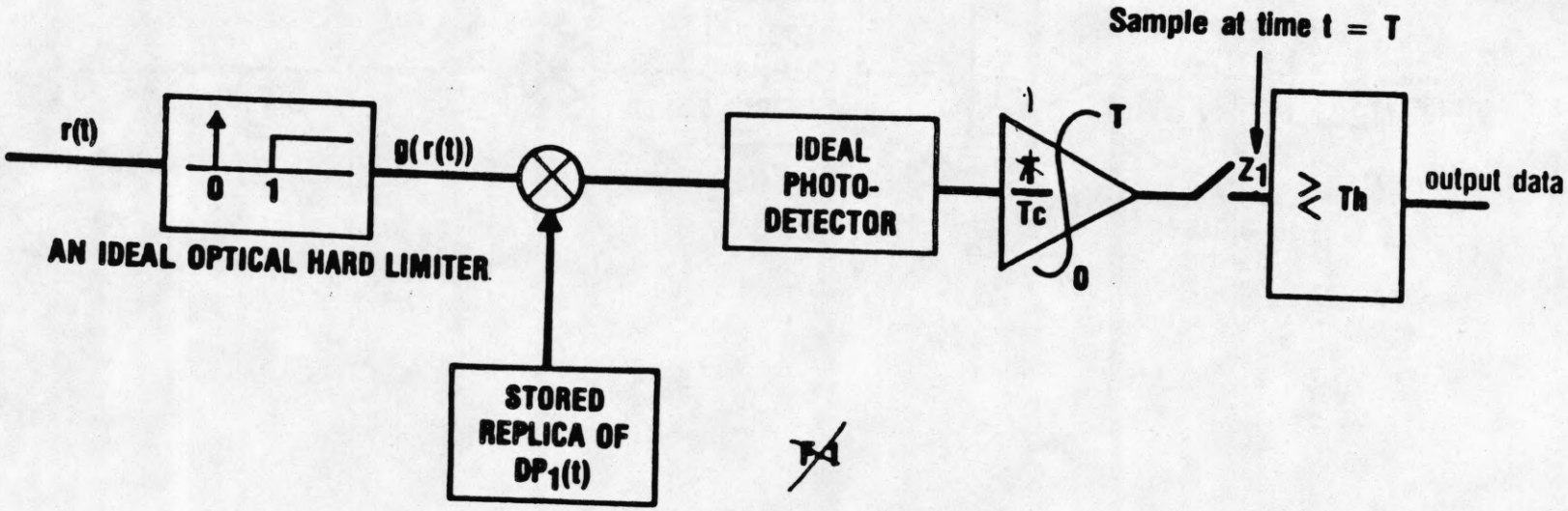


Figure ~~12a~~ 3a: An ideal optical correlation using active optical components

Figure 7 taken from Reference 2



~~Figure 24:~~ A typical optical receiver with an ideal optical hard limiter.
Figure 9

taken from Reference 1

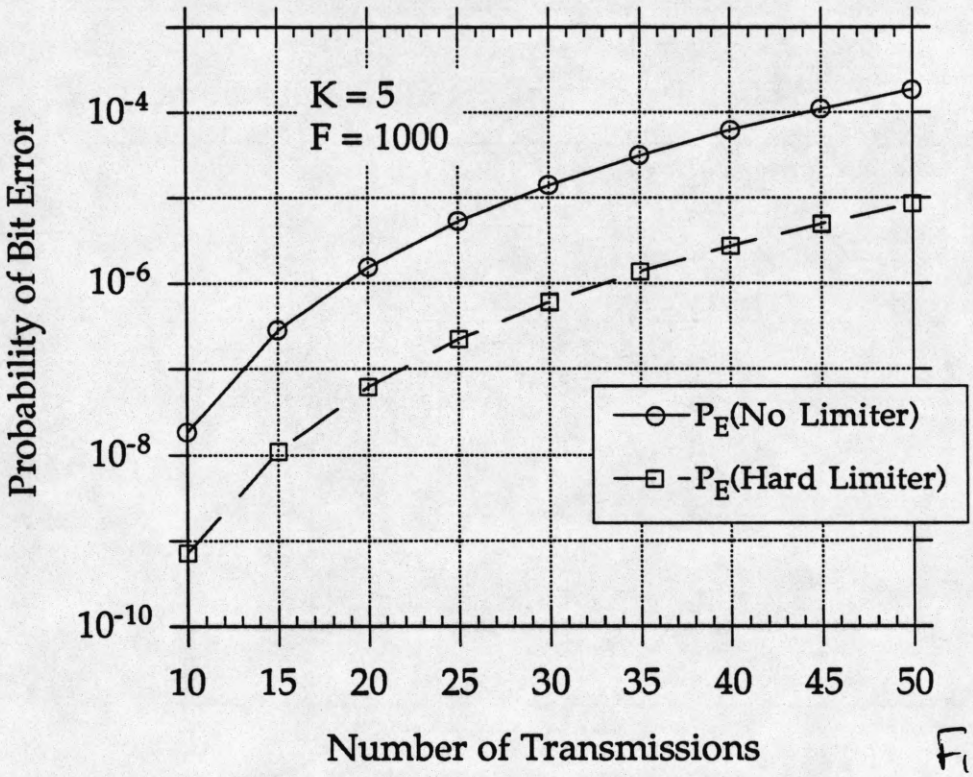


Figure 11

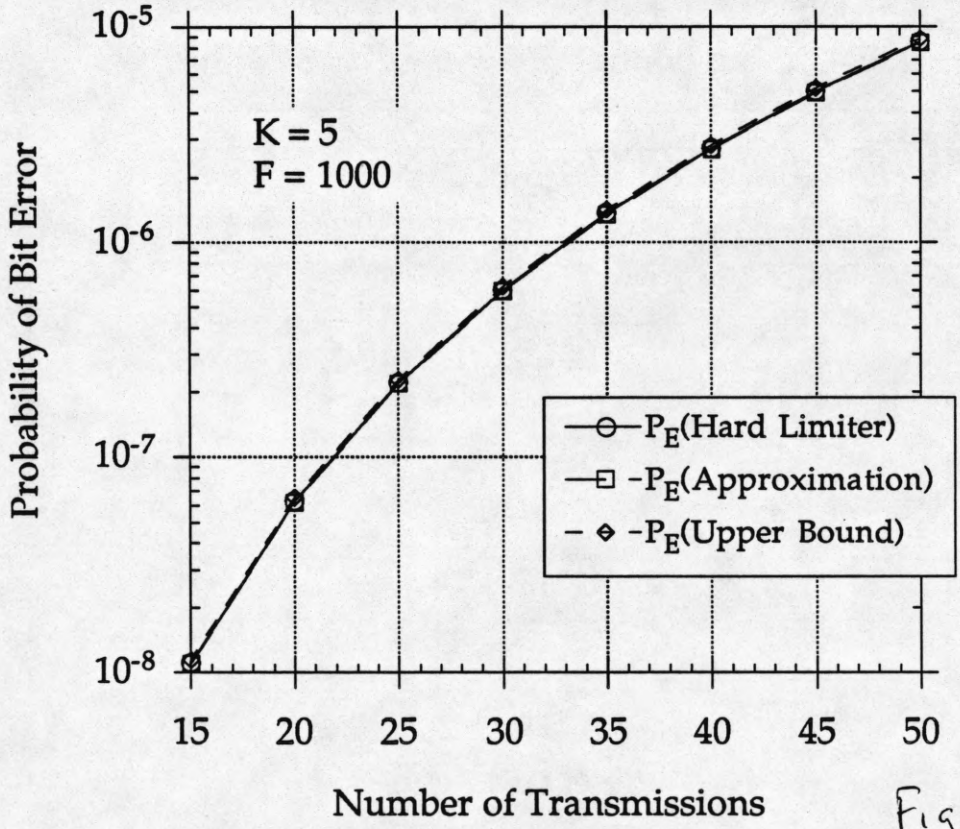


Figure 12

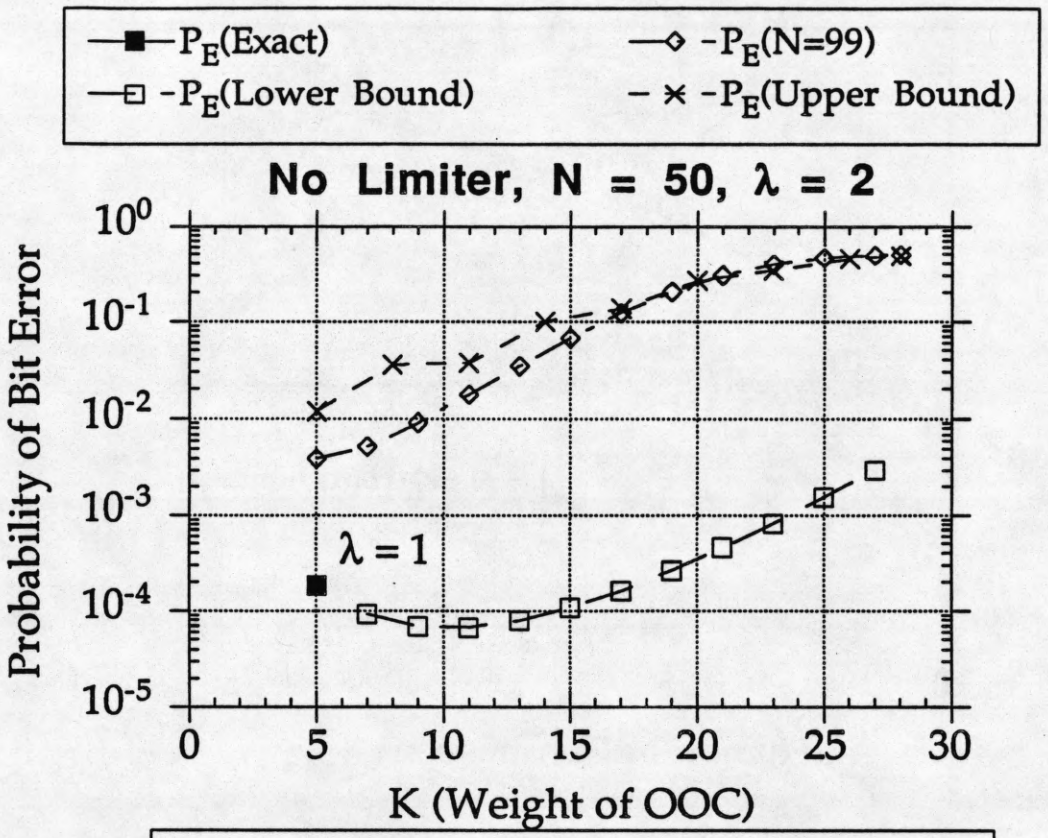


Figure 13

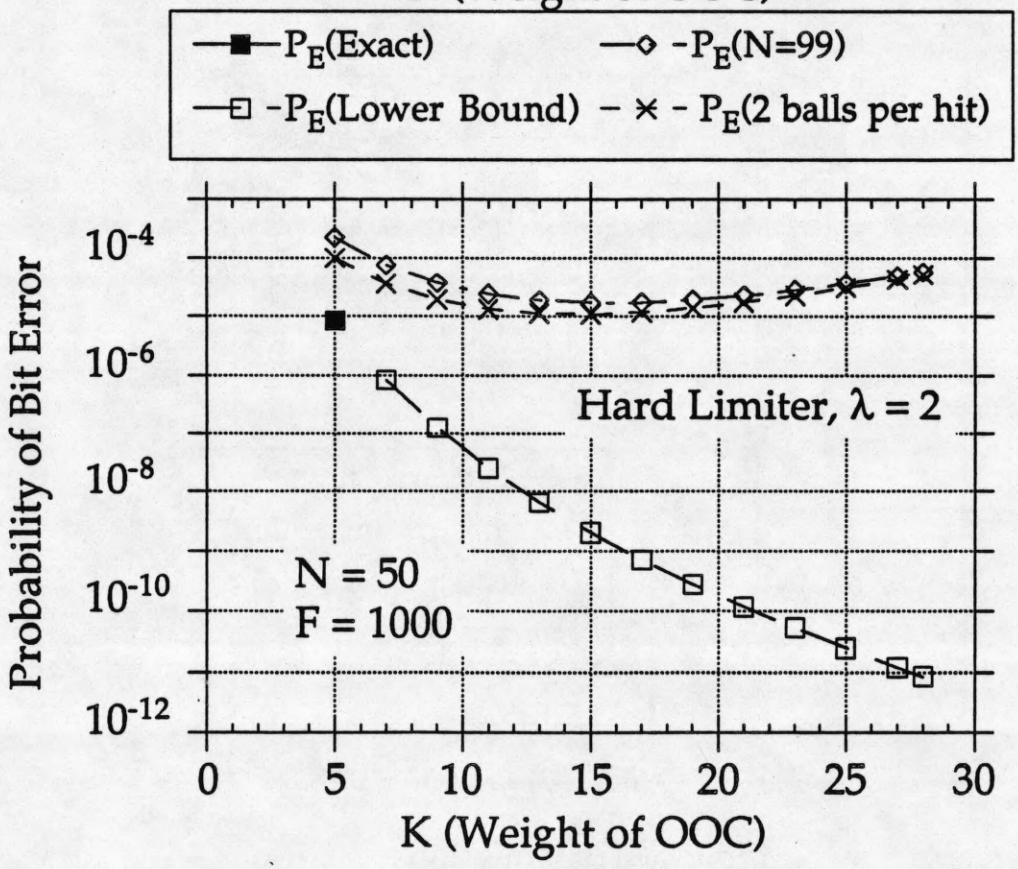


Figure 14

APPLYING ECONOMICS TO NETWORK RESOURCE ALLOCATION

Michael Peercy

Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801
(217) 333-6564
FAX: (217) 244-1764
peercy@lucky.csg.uiuc.edu

ABSTRACT

In this paper we examine the application of economics to the network resource allocation problem. We begin with the presentation of work by Ferguson in creating and analyzing a flow control economy through economic theory. Then we propose a different approach to the application of economics to flow control. We create a mechanism whereby network resources become real world resources subject to real world economic laws. We present simulation results of our strategy. We also propose extensions of the basic strategy.

The rush hour problem is a good example. The size of city expressways is determined almost entirely by the peak traffic that they have to bear. The extra cost to the city of an additional driver at 3 a.m. is essentially zero—the roads are there anyway and nobody is using them. The extra cost of an extra driver at rush hour averages out, I am told, to about five dollars per trip. Presently, both drivers are charged essentially the same price, in the form of higher gas costs due to gas taxes. If the roads were privately run, it would pay their owners to encourage off-hour traffic by charging a low price, and to discourage people from driving at rush hour by charging them the full cost of their trip.

David Friedman
The Machinery of Freedom

1. INTRODUCTION

There are many problems in distributed systems which deal with diverse users. While most previous solutions have tried to neglect the diversity as much as possible, some new solutions actually use diversity in their favor. These solutions model problems to resemble some other distributed system of diverse agents: for instance, ecology or economics [1]. In this paper we are interested in applying economics to the network resource allocation problem. A recent paper by Ferguson [2] uses economic theory to propose a flow control economy to allocate network resources.

We begin in Section 2 by presenting Ferguson's flow control economy. We then continue with an approach different from Ferguson's. While he applies economic theory to networks, he does not actually bring the economic laws of the real world to bear on the problem of resource allocation in the network. In Section 3 we propose a pricing mechanism which does just that. By bringing the laws of supply and demand of the outside world into the network, we elevate the links of the network from mere network resources to real world resources. In section 4 we present simulations and discussion of our strategy, and in section 5 we present extensions of our ideas. Section 6 concludes the paper.

2. A FLOW CONTROL ECONOMY

2.1. Flow Control Economy Model

Donald Ferguson's PhD dissertation at Columbia [2] gives one use of economic theory in the design of a network resource allocation strategy. He proposes a flow control economy with features of a real economy. Ferguson's model begins with a network of M links. Each link i has a supply S_i which it may allocate to users of the link. The supply S_i is defined to be strictly less than the actual link capacity C_i so that misfortunes in queuing are avoided. The other components of the network model are the N virtual circuits vying for link resources. Virtual circuit (VC) a is allocated a wealth W_a. By spending this wealth among the links in the path P_a, the virtual circuit receives link resources. In economic terms the links act as supplier agents, and the VCs act as consumer agents.

An allocation is a vector $\vec{x} = <x_1, x_2, ..., x_M>$, where x_i is amount of the resources of link i used by a particular VC. Each virtual circuit has its own allocation vector. A price scheme $\vec{p} = <p_1, p_2, ..., p_M>$, with $p_i \geq \varepsilon > 0$,

defines the *price per unit* of the resources of each link i. The goal of Ferguson's flow control economy is to achieve a utilization-maximizing equilibrium among the allocations demanded by the VCs using the mechanism of the prices of the links.

2.2. Demands of Virtual Circuits

In an economy each agent has a preference relation describing the relative value to the agent of the many different allocations possible to it. The agent uses this preference relation to select the best allocation it can afford given the price vector $\vec{p}$ and the agent's wealth. The preference relation in Ferguson's flow control economy is based on throughput and delay. Each VC a is characterized by a (constant) path P_a and a *throughput goal* γ_a. VC a adapts its allocation $\vec{x}$ to bring the throughput of the allocation, $T(\vec{x})$, to the throughput goal γ_a. If the VC is successful in attaining the throughput goal at the current prices, its surplus wealth is spent in order to buy excess space from the links to minimize the delay $D(\vec{x})$.

This preference relation permits a wide diversity of virtual circuits to specify accurate measures of goodness. A throughput-oriented session, such as file transfer, could set $\gamma_a = \infty$. A real time session, such as voice communication, could set γ_a equal to the minimum requirement of transmission and have a large surplus with which to minimize delay. However, this preference relation offers no means for a virtual circuit to specify a lower-bound average rate (LAR) [3] to indicate the minimum throughput the VC will tolerate.

A VC's demand, $\vec{\phi}^a(\vec{p})$, is its most preferred allocation at prices $\vec{p}$. The above preference relation defines a continuous function of demand, permitting a mathematical algorithm to determine $\vec{\phi}^a(\vec{p})$ from $\vec{p}$ and P_a. Let the total price per unit of transmission be Q_a. Set

$$Q_a = \sum_{i \in P_a} p_i$$

If $W_a \le Q_a \gamma_a$, then, for all $i \in P_a$,

$$\phi_i^a = \frac{W_a}{Q_a}$$

That is, if VC a can not afford the price per unit to send at its throughput goal, it sends at the throughput it can afford, demanding the same affordable throughput from each link in the path. (Of course, if link i is not in path P_a, $\phi_i^a = 0$.)

If $W_a > Q_a \gamma_a$, then VC a *can* afford its throughput goal. VC a then sets

$$\phi_i^a = \gamma_a + u_i^*$$

where u_i^* is the excess resources purchased from link i in order to minimize total delay. The delay expression which is minimized to determine $\vec{u}^*$ is

$$D(\vec{x}) = \sum_{i \in P_a} \frac{1}{C_i - (S_i - x_i) - \gamma_a}$$

Assuming independence between the links, this determines the total average delay experienced by VC a as the sum of the delay experienced at each link in P_a. The delay experienced by VC a with allocation $\vec{x}$ due to link i is

$$D(x_i) = \frac{1}{C_i - (S_i - x_i) - \gamma_a}$$

S_i is the allocatable supply on link i. x_i is the supply being consumed by VC a. Therefore, assuming that all the remaining supply $S_i - x_i$ is being consumed by other VCs, $C_i - (S_i - x_i)$ is the available service rate of link i. Since the throughput goal γ_a is the planned arrival rate, the above expression for link delay follows for an M/M/1 queue. Ferguson minimizes the aggregate delay, arriving at an expression for the Lagrange multiplier. His proposed algorithm to determine $\vec{u}^*$ performs a binary search on the possible values of the Lagrange multiplier.

2.3. Equilibrium Through Pricing

We have described the VCs' responses to the link prices $\vec{p}$: each VC a calculates a demand vector $\vec{\phi}^a(\vec{p})$. If an equilibrium is to be reached among allocations, the links must push the VCs' demands closer to equilibrium by changing their prices. Each link sees a total demand

$$d_i(\vec{p}) = \sum_a \phi_i^a(\vec{p})$$

and an excess demand

$$Z_i(\vec{p}) = d_i(\vec{p}) - S_i$$

Ferguson initially defines equilibrium as having the following properties: (1) no resources are unused, and (2) every agent's demand is satisfied. However, if a link's supply, even at the lowest possible price ε, exceeds its total demand, then resources must remain unused. Thus equilibrium in the flow control economy allows surplus supply. The flow control economy is in equilibrium at prices $\vec{p}^*$ if, for all links i, $Z_i(\vec{p}^*)=0$, or, $Z_i(\vec{p}^*)<0$ and $p_i^*=\varepsilon$.

The following algorithm from [2] is used to attain equilibrium of prices and demands in the network.

ALGORITHM 1:

1. Choose an initial prices $\vec{p}$.

2. For all VCs a, compute $\vec{\phi}^a(\vec{p})$ as described above.

3. If for all links i, ($Z_i(\vec{p})=0$) or ($Z_i(\vec{p})<0$ and $p_i=\varepsilon$), stop, for an equilibrium has been reached.

4. Otherwise, for all links i,

$$p_i = \max\left[p_i + p_i \cdot \frac{Z_i(\vec{p})}{S_i}, \varepsilon\right] = \max\left[p_i \cdot \frac{d_i(\vec{p})}{S_i}, \varepsilon\right]$$

5. Go to 2.

Step 2 of this algorithm can be performed by agents acting in the VCs' behalfs, and thus can be distributed. Step 4 can be performed by agents acting for the links and is similarly distributable. Step 3, on the other hand, requires the global determination of equilibrium, and Ferguson gives no distributed form for it (although he conjectures its existence).

Because the flow control economy Ferguson develops is intentionally designed to parallel a real economy, the same techniques can be used to prove the existence of equilibrium. Ferguson proves that there is, in fact, an equilibrium price vector $\vec{p}^*$ for the flow control economy. However, there may be infinitely many equilibrium price schemes. The example Ferguson gives is shown in Figure 1. The single virtual circuit of wealth 2 wishes to send throughput 1 across both links of supply 1. Any price scheme in which $p_1+p_2=2$ is an equilibrium price scheme. But note that there is only one equilibrium *allocation* in this example: each link allots one unit of resource. Whether only one equilibrium allocation exists in a flow control economy is an open problem.

No proof for convergence of the flow economy described by ALGORITHM 1 is known. However, in "several hundred simulation experiments" [2] the flow control economy always converged. Demonstrated conver-

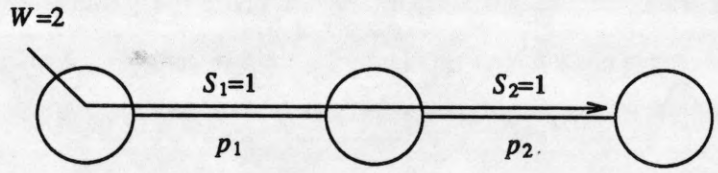


Figure 1. Any Pricing Scheme With $p_1+p_2=2$ Is Equilibrium.

gence was rather rapid, although recognition of convergence was sluggish. Heuristics were used to prevent overiteration, and adequate convergence was usually attained in about 25 iterations.

In determining the fairness of the equilibrium allocation of the flow control economy on the network links, economic theory provides another powerful concept. *Pareto-optimality* is a definition of fairness used in economics. To define pareto-optimality for the flow control economy, we begin by defining a *coalition* S as a subset of the VCs in the economy. S can *improve* on an solution $(\vec{x}^1, \vec{x}^2, ..., \vec{x}^N)$ if there is another solution $(\vec{y}^1, \vec{y}^2, ..., \vec{y}^N)$ such that:

1. $\vec{y}^a = \vec{x}^a, a \notin S$
2. $\vec{y}^a \geq_a \vec{x}^a, a \in S$, where $\geq_a$ means a prefers $\vec{y}^a$ to $\vec{x}^a$
3. There exists $a \in S$ for which $\vec{y}^a >_a \vec{x}^a$, where $>_a$ means a strictly prefers $\vec{y}^a$ to $\vec{x}^a$

A solution is pareto-optimal if no coalition can improve on their allocations. That is, a solution is fair by the measurement of pareto-optimality if no VC or set of VCs can improve their allocations without affecting the allocations of the VCs outside the set. A theorem proven by Ferguson asserts that, given an equilibrium price scheme $\vec{p}^*$, the VCs' demanded allocations $\phi^1(\vec{p}^*), \phi^2(\vec{p}^*), ..., \phi^N(\vec{p}^*)$ are pareto-optimal and unique.

2.4. Discussion

Ferguson has successfully brought economic theory into the design and analysis of network resource allocation. By proposing a simple yet versatile demand utility function for the virtual circuits, and then subjecting the VCs to the laws of supply and demand, he has attained a provably fair allocation of the network resources. His work is an excellent application of economics to the flow control problem.

However, there are some disadvantages in Ferguson's flow control economy. The first drawback is that the allocation and behavior of wealth is unspecified in the flow control economy. Another drawback, due to the requirement of attaining equilibrium, is that VCs are not allowed to choose paths; they must follow fixed paths. Still another disadvantage is the fact that only a centralized scheme is known for attaining the global equilibrium in ALGORITHM 1.

But perhaps the greatest disadvantage is the demand preference relation. The economic parallels exhibited by the flow control scheme require that this relation be continuous from 0. That is, a virtual circuit is assumed to be satisfied if it can obtain *any* resources, no matter how small. This may not be realistic. Many applications would

prefer to wait and try later than tolerate a smaller throughput.

In the next section, we discuss another approach to the application of economics to the network resource allocation problem. The strategies proposed there offer the following solutions to the disadvantages of Ferguson's flow control economy. First, this scheme does not require virtual circuits to *pre*specify wealth; they instead specify a willingness to pay. Also, this scheme allows noncontinuous allocation functions, permitting a greater range of users to accurately describe their demands. Most importantly, the approach of the next section does not seek a global equilibrium, thus avoiding the centralization problem of Ferguson's algorithm. Finally, because no global equilibrium is sought, the virtual circuits are allowed, in fact, encouraged, to choose their favorite paths.

3. ANOTHER APPROACH

3.1. Principles

While Ferguson's work is an excellent application of economic theory to the network allocation problem, it ignores the great influence economic laws in the real world could have on the load of the network. In the real world prices are the means for equating supply and demand. And when demand exceeds supply for some resource, say a particular network link, a raised price can bring demand back into equality with supply. This is the idea behind Ferguson's flow control economy, but he does not attach it to dollars in the real world. Because such an attachment would show up on the user's bottom line, real dollars should be a very effective means of regulating the demand for network resources. Our primary goal is now clear: *to create a mechanism whereby network resources become real world resources subject to real world economic laws.*

We can arrive at a mechanism which achieves this goal through the following argument. A network is not a natural resource: someone has to pay for the creation and maintenance of the network. Ideally, the people who use the network should be the ones who pay for it. Simple application of this principle would lead one to conclude that a user should pay for the network in proportion to the amount of resources he uses. However, there is an essential difference between resources granted to the only user on the network and resources granted to one user while another user's resource demand is left unmet. The former user induces no costs outside the network. The latter user brings a cost to the unsatisfied user: the cost of not receiving the desired resources. That induced cost can actually

be equated to dollars—dollars the unsatisfied user would be willing to pay the network owner to displace some other user from the network to get resources for himself. It must be stressed that this is a *real cost.* (Just ask the unsatisfied user.) Therefore, economically this should be the *real price* for the resource. It is what some user is willing to pay to receive the scarce resource; it is the price at which that resource could be sold. The network owner should make sure that every user currently occupying resources is willing to pay the price at which he could reallocate the resources, and then he should charge them that price.

We have introduced the basic idea behind the setting of prices:

1. The price of a scarce resource is the maximum someone who does not have it is willing to pay.

2. The price of a nonscarce resource is roughly the cost of its production.

The basic (nonscarce) price of resource i we call 1.00, and any price charged for i will be greater than or equal to 1.00. In an actual implementation, the basic price of 1.00 on one resource may be monetarily different from what we call 1.00 on another resource. Exchange rates easily allow conversion of the base price on any resource to be converted to real dollars.

3.2. The Link Manager

We now enter into the world of the network to investigate the application of the principles outlined above. The pricing scheme described above is, in fact, very simple to implement on a link level. Each link is managed by a *link manager* situated, for example, at the source end of the link. The link manager holds a *bid table* containing an entry for each virtual circuit desiring service from the link. In a link's bid table entry a, for virtual circuit a, contains the following fields:

1. R^a: the request of VC a,

2. O^a: options for service for VC a, and

3. P^a: the price that VC a is willing to pay per unit of service (VC a's *bid*) on this link.

The manager's task is very straightforward and is shown here in Algorithm MANAGER.

Algorithm MANAGER

1. Dereserve all link resources.

2. Let $A = \{a : R^a \neq \emptyset\}$.

3. Let $a = a'$, where $a' \in A$ and $P^{a'} = \max_{a''} P^{a''}$.

4. If R^a can be granted, then reserve requested resources.

   Otherwise, stop and reallocate (or continue prior allocation of) resources according to reservations, charging $X = P^a$ per unit.

5. Let $A = A - \{a\}$.

6. Go to 3.

Table 1 gives an example of the operation of Algorithm MANAGER. Let the link in the example have a supply of 10 units of resource, and let requests be a specification of units of resource desired. Assume that a VC would rather have no resources than have less than it requests. The link manager allocates the 10 units of supply in order of decreasing bid. Thus VC 2, which bid 2.00, has its request for 4 units granted. Similarly, VC 3's bid of 1.50 allows its request for 5 units to be considered and granted. VC 1 is considered next, but its request of 2 units of resource can not be granted; only one unit of supply remains. VC 1's demand is left unsatisfied, and, since VC 1 would pay 1.20 to satisfy that demand, each of the VCs must pay 1.20 for the resources they consume. This, of course, does not bother VCs 2 and 3 because they state in their bid that they are willing to pay more than that price in order to have their demands satisfied. Note that the charge for VCs 2 and 3 is *per unit* of resource. VC 2's service costs 4.80 in total, while VC 3's service costs 6.00.

To continue the example, suppose VC 3 leaves the link. (Perhaps it has found an alternate link which is not so expensive to use.) Then the link manager would grant VCs 1 and 4 the resources they request. Because all members in the bid table are satisfied, there is no excess demand to force the price up. No other user is willing to pay anything to displace the current users of the link. Therefore, the price at which all VCs on the link are charged

Table 1. Example of Algorithm MANAGER.

VC	Request	Price	Supply	Charge
2	4	2.00	4	1.20
3	5	1.50	5	1.20
1	2	1.20	0	.00
4	3	1.10	0	.00

is the base price of 1.00.

An immediate modification to the manager and pricing as we have described is obvious if we suppose that VC 5 enters the example of Table 1 with a bid of 1.15 and a request for 1 unit. After granting the requests of VCs 2 and 3, VC 1's request cannot be granted, but VC 5's request can. In the interests of efficient use of resources, VC 5 should be granted its demand of 1 unit; otherwise that unit would remain unused. However, VC 5 should not be charged the going rate of 1.20. It is not willing to pay 1.20, and it is not displacing any VC willing to pay 1.20. Whether VCs squeezed onto the link after some VC's unsatisfied request should be charged their bids or should be charged the bids of the VCs they are displacing (1.00 in our example) is debatable.

3.3. The Virtual Circuit Agent

The other participants in our network resource allocation scheme are VC agents. Each VC in the network is represented by one agent in the requesting and bidding for link resources. VC agents communicate with link managers through free (or low-cost) short messages given reserved bandwidth in the network. They send bids and receive allocation information from the link managers. While link managers in our scheme are very simple to specify, VC agents are quite difficult. Perhaps the best description of a VC agent is the following: an algorithm which minimizes the charge to a virtual circuit using the network.

Link managers are unintelligent and their behavior is easily understood. VC agents, on the other hand, are expected to be intelligent. They are expected to convey correctly the preferences of the VC and to represent accurately the price the VC is worth to the user. The only tools each agent has at its disposal are the bid tables at the various link managers. What agents write into these tables determines the services and the charges they receive. Each VC agent is in competition with other VC agents for the same resources. If those resources are scarce, the competition will incur costs that users must bear. Minimizing or limiting those charges while seeing to the success of the VC is the difficult job of the VC agent.

There are various differences which can exist among VC agents. VC agents can represent very diverse preference relations. Agents may update their entries in link managers' bid tables frequently or infrequently. Or an agent may begin with an allocated wealth, as in Ferguson's scheme. An agent may be given instructions to get the job done immediately regardless of cost. Or an agent may try to find the cheapest path or stick to one path through the network. In fact, the agent can be left as an exercise for the user.

3.4. Options

We have yet to discuss the options field of the bid table. This field contains options offered by the manager which might be desired by the agent. A number of options would be worth implementing due to their power of expressing the wishes of a VC agent and their ease of implementation in the link manager. Note that the line between requests and options can be somewhat fuzzy; options are, in fact, requested. But it is useful to separate options from requests because requests indicate service demanded from the resource while options indicate instructions to the link manager granting the request.

The first option we discuss is the lower-bound average rate (LAR) [3]. A VC would specify this option if there is some rate LAR, lower than the actual request, at which the VC can perform its job satisfactorily. If the link manager can not grant the full request of a VC specifying an LAR, the link manager gives as much as it can grant, provided this amount exceeds the LAR. If the manager can not grant the LAR, no resource is allocated to the VC, and the VC is unsatisfied.

Another worthwhile option is called STAY. A VC specifying STAY wishes to continue its current service at any cost. If a VC with a higher bid threatens to displace a staying virtual circuit, the manager automatically raises that circuit's bid to match the potentially displacing bid. This is useful for continuity critical jobs, such as teleconferences. If the raised charge far exceeds the desired charge, the agent may seek an alternate, less expensive link. But service will be uninterrupted.

A third option is DON'T. A VC agent bidding with the option DON'T does not actually want service; it only would like to know what its allocation would be if it did want service. This option has two primary uses. First, DON'T allows a VC agent to probe the network to find the best, least expensive path available to it. Second, if the virtual circuit requires supply from all links on the path simultaneously (as in TDM networks), DON'T allows the bid to sit in the bid tables of available links free of charge while service is sought at unavailable links.

4. SIMULATION

4.1. Simulator

We wrote a simulator to examine the behavior of this network allocation strategy. This 750 line program implements link managers and VC agents as processes created through CSIM routines [4]. The specific network simulated has a multislot TDM paradigm. The many slots per frame on a link can be allocated in any amount to the virtual circuits requesting resources. A virtual circuit is required to have a complete path before using the circuit. This is enforced to restrict the necessary queuing. The simulator implements the three options discussed in the previous section. However, in our implementation LAR is one of two values: the desired rate or zero; that is, either the agent will not settle for any allocation less than the requested allocation, or the agent will be happy with any rate that can be granted. For simplicity all links have a base charge of 1.00.

The difficulties in implementing the proposed network resource allocation scheme lie in the design of the VC agents. Here we do not make a detailed investigation of agent strategies. For instance, while agents send bids to managers through mailboxes (simulating free reserved bandwidth), agents are given perfect knowledge of the internal state of the system. In an actual implementation where the agents do not have perfect knowledge of the state of the links, decisions must be made regarding the frequency and contents of state information feedback from managers to agents. These factors would be specified by agent-selected options, and, again, we are not attempting the difficult problem of determining optimal agent strategy.

Instead we provide only two types of agents: passive and aggressive. A passive agent selects a price it is willing to pay for each link and then bids that price for each link on its fixed path. A passive agent uses the DON'T option to avoid buying resources on nonscarce links until it believes that it has a complete path. Other than using DON'T, a passive agent takes no action to attempt to increase its performance.

An aggressive agent, on the other hand, has all the characteristics of a passive agent along with two more. Aggressive agents specify the STAY option so that, once they occupy resources, they hold those resources until their job is finished regardless of the price they must pay. The second difference separating the aggressive agent from the passive agent is that the aggressive agent will raise its bid for each link to the current charge on that link if that charge exceeds its bid. We do not investigate the results of raising the bid to the current charge plus a small increment.

Virtual circuits are introduced into the network at an exponentially-distributed rate of mean λ and are directed between a source and destination chosen at random uniformly. The price willing to pay, or bid, of each VC is drawn from an exponential distribution shifted to 1.00 with a mean β. Of course, the actual mean does not matter in our simulator since all decisions made by the link managers are based on *relative* values of bids and our VC agents are not concerned with their absolute expenditures. The total traffic a virtual circuit needs to transmit is drawn from a uniform distribution. The virtual circuit finishes and releases its resources when it has transmitted an amount of data equal to or exceeding its chosen limit. The rate at which it transmits, its request, is drawn either from a uniform distribution or a truncated exponential distribution.

4.2. Experiments

We performed experiments on the network in Figure 2. The number beside each link indicates the units of resource, in slots per frame, in each direction of the link. Note that there is no choice of path in this network.

The experiments performed on the simulated network had one main feature in common: enormous variance. Very few trends are visible in the majority of the results. This is not entirely unexpected; a cornerstone of our scheme is the diversity and intelligence of VC agents. We simulate neither diversity nor intelligence. However, two experiments we performed are quite instructive, giving good results.

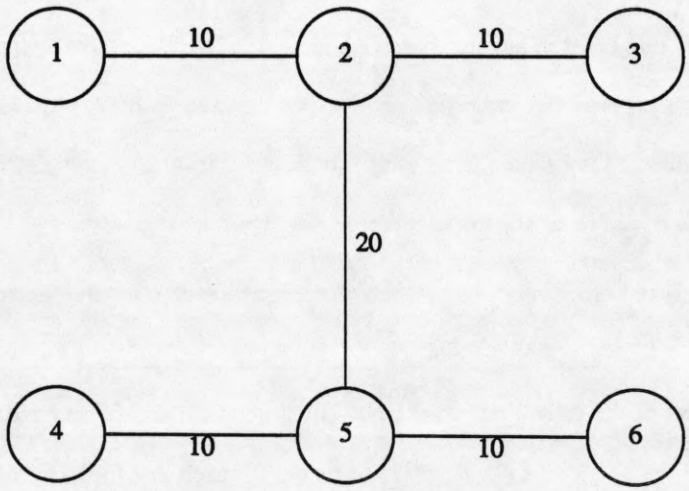


Figure 2. Simulated Network

4.2.1. Overtime as a function of bid

In our simple TDM network, an excellent measure of performance for a virtual circuit is a ratio called *overtime*. Overtime is defined as the time a VC takes to complete its job divided by the time it should have taken, where the time it should have taken is given by the total traffic of the job divided by the rate of the circuit. An overtime of one is ideal, and implies a net delay of zero in transmission.

Figure 3 shows overtime as a function of bid. The bids are blocked into increments of .20, and the overtime given is the average of all overtimes in each bid block. The solid bars are for an experiment with a rate request drawn from an exponential distribution shifted to 1, with mean 3, truncated at 5. The dotted bars are for an experiment with a constant rate request of 3. All agents in these experiments are passive.

We notice on the graph that there is a monotonic decrease in overtime as bid increases. This result is to be expected since, with passive agents, the link managers simply run a preemptive priority queue, where bid represents the priority.

4.2.2. Overtime and cost as a function of agent and load

A number of interesting results are found in another collection of experiments. These experiments vary network load and the mix of different agent types to investigate how effectively aggressive agents increase their own

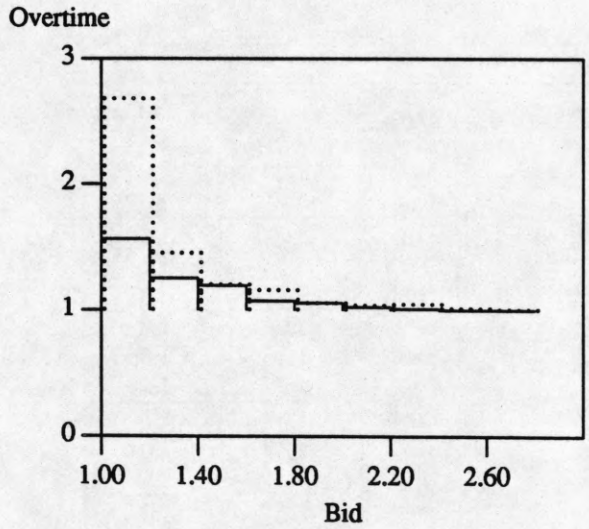


Figure 3. Overtime as a function of bid.

performance without excessively harming the performance of passive agents. The data in Table 2 are the average values from 10 simulations. The two columns are overtime and cost, the actual cost per unit for the average VC to complete its job. Three separate ratios of passive and aggressive agents are shown. In the first block, all agents in the network are passive. In the second block, all agents in the network are aggressive. The third, fourth, and fifth blocks give data for experiments in which half of the agents are passive and half are aggressive. This condition for each agent is chosen at random, irrespective of its bid. The third block is the overall results for this mix of agents. Blocks four and five separate the data for passive and aggressive agents in the mixed system.

The first item to which we should draw our attention is the effect aggressive agents have on overall overtime. Comparing the overtimes of the passive block and the mixed (overall) block, we see that average overtime is not significantly increased, and, in fact, is decreased for high load. Of course, the prices also are higher due to increased competition for the links.

Another result to notice is the superior overtimes of aggressive agents compared to passive agents. In the mixed experiments, the aggressive agents suffer less delay, while paying relatively little more in cost. Comparing the experiments of all aggressive agents to all passive agents, we find that, at low loads, the competition among all involved agents causes overtime to increase for the aggressive agents while they pay much more for service. At high load, however, the passive agents suffer high overtimes while the aggressive agents do not. However, the

Table 2. Overtime and Cost vs. Agent Type and Load.

Agents	Load	Overtime	Unit Cost
Passive	Low	1.04	1.01
	Med.	1.24	1.02
	High	6.60	1.22
Aggressive	Low	1.11	1.08
	Med.	1.34	1.27
	High	3.48	3.73
Mixed (Overall)	Low	1.06	1.02
	Med.	1.26	1.05
	High	5.77	1.64
Mixed (Passive)	Low	1.07	1.01
	Med.	1.36	1.04
	High	10.24	1.56
Mixed (Aggressive)	Low	1.05	1.02
	Med.	1.16	1.06
	High	1.67	1.72

aggressive agents do end up paying a high price for their low overtime.

Some of the results in Table 2 are explained by a hidden factor in the simulations. As mentioned previously, the DON'T option is used in both passive and aggressive agents to allow a bid to remain in the bid tables without receiving (or paying for) service. When VC agent *a* recognizes, through the complete state information available to it, that supply exists at every link along the path, it rescinds its DON'T specification, telling all managers in the path to consider it for allocation. If, however, any link in that path receives a request before the next auction which takes the supply agent *a* thought it had, VC *a* will not have a complete path. Furthermore, all links which *are* able to get resources must pay for them, at least for one frame, even though they are completely useless to the TDM virtual circuit. Thus occasional bursts of charges can be generated without any work being done.

The effect of this misallocation can be very bad. Since the VCs with the lowest bids are the ones waiting in tables the longest, they suffer this costly misallocation more often then other VCs. Thus for the all-passive, high-load experiment, an appreciable amount of the cost 1.22 is likely due to spurious allocations. In fact, examining unit cost, the actual charge per unit transmitted, as a function of bid can show a fairly level curve; the lower bidding agents charges are artificially increased by useless allocations. Investigations into the magnitude of this problem's effects and possible solutions is needed.

5. EXTENSIONS

We have proposed a mechanism for allocating resources according to real world supply and demand laws through the real world medium of money. Extensions to this mechanism can also be drawn from economics. One aspect of our scheme is the uncertainty of receiving resources. If some other user is willing to spend an enormous amount of money, he could purchase entire links, and service expected by some other user would be gone. Thus some sort of reservation system would be valuable. This reservation system could be in one of two forms: futures trading or insurance.

In futures trading a user could purchase a resource for a period of time in the future at some price. The immediate difficulty of this proposal is that the price must be determined by the link manager. This requires the introduction of some type of intelligent decision making into the manager. Possible strategies for a manager to use in setting future prices are taking bids for the future and prediction from past use (such as the same time last week).

Insurance, on the other hand, is paid by the agent to insure that it will not be asked to pay *more* than a certain price for its future use of the resource. The pricing scheme for this insurance feature is even more difficult than that for futures trading.

To introduce another extension, note that we are not suggesting that a network owner necessarily fleece his customers if their demands drive up their own prices. The actual cost to the owner of this higher load is probably negligible. The owner could uniformly scale down the users' accounts so that the sum of all accounts equals his costs plus profits. Such a rebate should not seriously alter the actions of supply and demand on resource allocation, but it would make it more difficult for the VC agents to predict how much money they are actually spending. Frequent announcements from a network manager of how much above cost the system is as a whole would help the VC agents in their bidding.

Of course, the wise network owner would take excess profit and reinvest it into new or higher capacity links, thus relieving the overutilization of the links which, after all, produce the excess profit. Whether this sort of reinvestment is worthwhile to the owner is a very interesting economic question. If the charges brought by new users drawn to the improved link are greater than the charges brought by the original users competing for the original link, the move results in a net profit for the owner without any adjustment of rates. If there is a profit to the owner, then everybody is happier with the new link, and expansion of the network to decrease overutilization is economically natural.

6. CONCLUSIONS

We have presented two economically-inspired approaches to the network resource allocation problem. Ferguson's approach equates supply and demand in the network with supply and demand in economics, develops algorithms to implement the balancing of supply and demand through pricing, and uses economic theory to prove the optimality of his algorithms. The economic measure of fairness Ferguson brings to network allocation is *pareto-optimality*, in which no user or group of users can receive a better allocation without forcing a worse allocation on other users.

In this paper we have developed another approach to applying economics to resource allocation. We began with the goal of making conflicting demands for the scarce network resources subject to economic laws of the real

world. The mechanism proposed for this is charging high prices for the use of overutilized links. The charges are determined by the prices users displaced from the links are willing to pay to receive service. We described a simple link manager to take care of this charging, and proposed features intelligent and diverse virtual circuit agents could have. We also presented results of simulations of this strategy.

Future research in this area includes the simulation of more flexible networks than our TDM network. Also, studies into more intelligent and diverse agents are needed, as they are the foundation of the successful implementation of our network resource allocation strategy. The nature of our scheme warrants investigations into its behavior with respect to the discontinuities inherent in auctioning of resources and with respect to convergence to some equilibrium. Another area for future research is the comparison of our strategy with other allocation strategies, using an appropriate overall measure of goodness. And, finally, a true economic analysis should be performed to see if everyone does indeed profit if the charges paid by users overutilizing a link go to the expansion of that link. For, if this is the case, then our scheme is an ideal way to manage network resources, and it truly does bring economics in the real world to bear on the problem of resource allocation in the network world.

REFERENCES

[1] B. A. Huberman, ed., *The Ecology of Computation.* Amsterdam: North-Holland, 1988.

[2] D. F. Ferguson, "The application of microeconomics to the design of resource allocation and control algorithms.," PhD Thesis, Columbia University, 1989.

[3] L. Zhang, "A new architecture for packet switching network protocols," PhD Thesis, Massachusetts Institute of Technology, 1989.

[4] H. D. Schwetman, "CSIM: A C-based, process-oriented simulation language," Tech. Report PP-080-85, Austin, Texas, 1985.

QUEUE LENGTH DISTRIBUTIONS IN THE PACKETIZED TRANSMISSION OF REAL-TIME TRAFFIC

Upamanyu Madhow

Consider a slotted communication channel with a periodic arrival process. There are K independent periodic sources, each generating one fixed-length packet every M slots. One packet can be transmitted in every time slot. Packets are queued upon arrival, and no packets are lost, so we need $K \leq M$ for stability. In this paper, we consider the distribution of the typical and the maximum queue lengths for such a system. Several recent papers [1, 2, 7, 8] have considered various aspects of this problem. The queueing system here can be thought of as modeling a link in a network operating in asynchronous transfer mode (ATM), with the periodic arrivals modeling real-time traffic. An understanding of the buffering problems in this system, therefore, is a first step towards the analysis of ATM networks, which are becoming increasingly popular as a means of integrating a wide variety of traffic types.

An elegant $O(K^2)$ algorithm is given in [1] for computing the distribution of the typical queue length. An $O(M^3 K)$ algorithm for computing the distribution of the maximum queue length is given in [7]. It is shown in [8] that if K/M is held constant at λ, then the typical queue length is increasing in M with respect to a stochastic convex ordering, and the limiting distribution is that of an $M/D/1$ queue with arrival rate λ.

In this paper, we first present an $O(M K^4)$ algorithm for computing the distribution of the maximum queue length. The algorithm is based on the methods of [1]. Next, we consider, for the special case $K = M$, the asymptotics of the distribution of the typical queue length and the maximum queue length as $M \to \infty$, both of which are shown to grow as $M^{1/2}$.

II. Computation of the Maximum Queue Length Distribution

Let Q_t be the queue length at the end of the t-th slot, and let a_t be the number of arrivals in the t-th slot. A frame is defined to be M consecutive slots. The time at which any one of the K independent periodic sources produces a packet is assumed to be uniformly distributed over a given frame interval. We have that $a_t = a_{t \bmod M}$, and that $(a_t, \ldots, a_{t+M-1})$ are interchangeable, identically distributed random variables that sum to K. Starting from an empty system, we have

$$Q_t = (Q_{t-1}-1)^+ + a_t, \; t > 0,$$

$$Q_0 = 0 \ .$$

It is easy to see [1, 2] that $\{Q_t, t \geq M\}$ is periodic, with period M. Thus, we assume, for convenience, that the periodic pattern is established for all time, i.e., for $t \in (-\infty, +\infty)$. It follows [1] that

$$Q_t = \max_{0 \leq i < M} \{ \sum_{j=t-i}^{t} a_j - i \} \ . \tag{1}$$

Consider a typical slot, say slot 0. Since $\{Q_t\}$ is periodic, it suffices to consider this slot and the $(M-1)$ previous slots to compute the distribution of the maximum queue length Q^*. We can write

$$P[Q^* \geq L] = \sum_{l=0}^{K} P[Q^* \geq L, Q_0 = l] \ . \tag{2}$$

For $l \geq L$, we have

$$P[Q^* \geq L, Q_0 = l] = P[Q_0 = l] \ , l \geq L \ ,$$

so that

$$\sum_{l=L}^{K} P[Q^* \geq L, Q_0 = l] = P[Q_0 \geq L] \ .$$

The right hand side above can be computed with $O(K)$ complexity [1]. For $l < L$, we have, conditioning on the number of arrivals j in the 0-th slot, that

$$P[Q^* \geq L, Q_0 = l] = \{ \sum_{j=0}^{l-1} P[Q^* \geq L, Q_{-1} = l-j+1 | a_0 = j] p_{M,K}(j) \}$$
$$+ (P[Q^* \geq L, Q_{-1} = 1 | a_0 = l] + P[Q^* \geq L, Q_{-1} = 0 | a_0 = l]) p_{M,K}(l) \ , \tag{3}$$

where $p_{M,K}$ is the probability mass function of a binomial distribution with parameters K and $1/M$. We proceed similarly with each of the joint probabilities on the right hand side, conditioning on the number of arrivals in slot -1, and so on. As we repeat this, a typical term to be evaluated is of the form $P[Q^* \geq L, Q_{-n} = r | a_0 = j_0, a_{-1} = j_1, \ldots, a_{-(n-1)} = j_{n-1}], 1 \leq n \leq M-1$. Since $r = l < L$ for $n = 0$, and since the value of r can increase by at most one (whenever some $j_i = 0$) as n increases, this term need never be computed for values of r exceeding L. This is because, when $r = L$, we have

$$P[Q^* \geq L, Q_{-n} = L | a_0 = j_0, \ldots, a_{n-1} = j_{n-1}] = P[Q_{-n} = L | a_0 = j_0, \ldots, a_{-(n-1)} = j_{n-1}] \ . \tag{4}$$

To compute the right hand side of (4), it suffices to consider arrivals only in slot $-n$ and in the $(M-1)$ slots that precede it. By periodicity of the arrival process, the arrivals in n of those slots are already constrained. Specifically, we have $a_{-n-(M-1)}=j_{n-1}, \ldots, a_{-M}=j_0$. The backlog due to these at the end of slot $-(M-1)$ is given by

$$b_n = \max\{0, j_0-1, j_0+j_1-2, \ldots j_0+j_1+\ldots+j_{n-1}-n\} \quad . \tag{5}$$

We can therefore consider the following reduced problem: find the probability of a queue length of L at the end of the last slot (slot $-n$) in a frame of length $M' = M-n$, given that there are $K' = K-j_0-\ldots-j_{n-1}$ arrivals in the frame, and that the initial backlog is b_n. Using methods based on a ballot theorem, as in [1], this can be done in $O(K)$ computations, as will be shown later. Note that we need not have $K' \le M'$ here.

In order to do this terminal computation, however, we need to propagate efficiently the value of b_n until condition (4) is satisfied. To this end, define the running sum

$$s_n = \sum_{i=0}^{n-1} j_i - n \quad .$$

Note that

$$b_n = \max\{b_{n-1}, s_{n-1} + (j_{n-1}-1)\} \quad ,$$

and

$$s_n = s_{n-1} + (j_{n-1}-1) \quad .$$

Thus, it suffices to preserve the values of b_n and s_n to compute the initial backlog for the reduced problem which must be solved when the termination condition (4) is satisfied. To make the recursion more explicit, we establish a new notation for the intermediate probabilities $P[Q^* \ge L, Q_{-n}=r \mid a_0 = j_0, \ldots, a_{-(n-1)} = j_{n-1}]$. Renumber slot $-n$ to be slot 0 in a reduced problem with $K' = (K-j_0-\ldots-j_{n-1})$ arrivals in $M' = (M-n)$ slots, numbered from $-(M'-1),\ldots,0$, with a running sum $s' = s_n$ and a backlog $b' = b_n$ as defined above. Denote the probability of interest by $P_{M',K'}[L, r \mid s', b']$. The recursion for this function is then given by conditioning on the number of arrivals in the slot being considered, and then shifting the time axis one unit to the left. This yields

$$P_{M',K'}(L, r \mid s', b') = \sum_{j=0}^{r-1} P_{M'-1,K'-j}[L, r-j+1 \mid s'+j-1, \max(b', s'+j-1)]\, p_{M',K'}(j)$$

$$+ (P_{M'-1,K'-l}[L,1|s'+l-1,\max(b',s'+l-1)] + P_{M'-1,K'-l}[L,0|s'+l-1,\max(b',s'+l-1)])\, p_{M',K'}(l) \quad . \tag{7}$$

The quantities of interest are $P_{M,K}(L,l|0,0)$, $l=0,\ldots,L-1$. The termination conditions are as follows:

$$P_{1,K'}[L,r|b',s'] = \begin{cases} 1 & , r = L = K' + b' \ , \\ 0 & , else, \end{cases} \tag{8}$$

$$P_{M',K'}[L,r|b',s'] = 0 \quad , K' + b' < L \quad , \tag{9}$$

and

$$P_{M',K'}[L,L|b',s'] = P_{M',K'}[L|b'] \quad . \tag{10}$$

Condition (8) deals with the degenerate case of a one-slot frame and condition (9) specifies the probability to be zero when there are too few packets in the reduced problem to attain a maximum queue length of L. The last condition, (10), has already been discussed and is the only condition that requires non-trivial ($O(K)$) computation upon termination.

To compute the complexity of this algorithm for a given L, we first count the number of points to be computed. Note that s' in (7) is completely specified by M' and K', since

$$s' = (K-K') - (M-M') \quad .$$

We then have that $1 \le M' \le M$, $0 \le K' \le K$, $0 \le b' \le K-1$, and $0 \le r \le L$, where the last condition is by virtue of (10). Thus, there are $O(M K^2 L)$ points to be computed in the recursion. The computation for each of these points is trivial except for points that terminate in (10) due to $r = L$. There are $O(M K^2)$ of the latter, and the computation at each of them is of complexity $O(K)$. This results in a complexity of $O(M K^3)$. Thus, in our original notation, for any given L, the computation of $P[Q^* \ge L, Q_0 = l]$, $l = 0,\ldots,L-1$, can be done in $O(M K^3)$ computations. Thus, the complexity of computing $P[Q^* \ge L]$ using (2) and the recursion (7), is $O(M K^3)$ for each L. Since this needs to be done for $L = 1,\ldots,K$, we have an overall complexity of $O(M K^4)$ for the distribution of Q^*.

It remains to show that the computation of the right hand side of (10), which corresponds to solving a reduced problem as described earlier, has $O(K)$ complexity. We replace M', K' and b' with M, K and b, respectively, for notational convenience. These are not to be confused with the parameters of the original prob-

lem. We want to compute $P[Q_0>q]$, given that there are K uniformly distributed arrivals in the M slots numbered $-(M-1),\ldots,0$, and that the backlog at the end of slot $-(M-1)$ is b. Let a_j be the number of arrivals in slot j. Then

$$Q_0 = \max_{0\le i<M} \phi(i) \quad , \tag{11}$$

where

$$\phi(i) = \sum_{j=-i}^{0} a_j - i \quad , 0 \le i \le M-2 \ , \tag{12}$$

$$\phi(M-1) = b + \sum_{j=-(M-1)}^{0} a_j - (M-1) = b + K - M + 1 \ . \tag{13}$$

We have $P[Q_0 > q] = 1$ if $\phi(M-1) \ge q+1$, so we assume $\phi(M-1) = b + K - M + 1 < q + 1$ to avoid triviality. This implies, using (11), that there exists an $i < M-1$ such that $\phi(i) = q+1$, since $\phi(i)$ can increase by at most one as i decreases, and $Q_0 > q$. Let $(x-1)$ be the largest such i. Define the following events:

$$A_x = \{\phi(x-1) = q+1\} = \{\sum_{j=-(x-1)}^{0} a_j = q + x\} \ ,$$

$$B_x = \{\phi(i) < q+1 \ , -(M-1) \le i \le -x\} \ .$$

Then

$$P[Q_0>q] = \sum_x P[A_x \cap B_x] = \sum_x P[A_x]\, P[B_x|A_x] \ . \tag{14}$$

We have

$$P[A_x] = \begin{cases} \binom{K}{q+x} (x/M)^{q+x} (1-x/M)^{K-q-x} & , 0 \le x \le K-q \ , \\ 0 & , else \ . \end{cases} \tag{15}$$

Notice that $P[B_x|A_x] = P[B_x'|A_x]$, with

$$B_x' = \{\sum_{j=-i}^{-x} a_j < (i-x+1) \ , i = x,\ldots,(M-2)\} \ ,$$

where we have subtracted the $(q+x)$ arrivals in the last x slots, and have used the assumption that $\phi(M-1) < q+1$. Due to the independence of the sources, the event A_x influences the conditional probability of

B_x' only by specifying the number of arrivals in the $(M-x)$ slots numbered $-(M-1),\ldots,-x$ to be $(K-q-x)$. Conditioning further on the number of arrivals in the first slot $-(M-1)$, we have by a ballot theorem (see [1]) that

$$P[B_x \mid a_{-(M-1)}, A_x] = 1 - (K-q-x-a_{-(M-1)})/(M-x) \quad .$$

Conditioned on A_x, $a_{-(M-1)}$ is binomial with parameters $(K-q-x)$ and $1/(M-x)$, so we can remove the conditioning on $a_{-(M-1)}$ to obtain

$$P[B_x \mid A_x] = 1 - (K-q-x)/(M-x) + (K-q-x)/(M-x)^2 \quad ,1 \le x \le M-1. \tag{16}$$

Recall that it was assumed that $K+b-(M-1) < q+1$ to avoid triviality. This implies that $K-q \le M-1$. Thus, from (15) and (16), the range of x of interest is $1 \le x \le K-q$, so that, substituting (15) and (16) into (14), we get an $O(K)$ complexity for the desired computation as promised. It is interesting to note that the expression for the resulting probability does not depend on the initial backlog b. We have for $b<q+M-K$,

$$P_{M,K}[Q_0>q \mid b] = \sum_{x=1}^{K-q} \binom{K}{q+x} (x/M)^{q+x} (1-x/M)^{K-q-x} \; [1-(K-q-x)/(M-x) + (K-q-x)/(M-x)^2] \quad ,$$

and for $b \ge q+M-K$,

$$P_{M,K}[Q_0>q \mid b] = 1 \quad .$$

This completes the description of the algorithm for computing the maximum queue length distribution. The overall complexity is $O(M\,K^4)$ as shown earlier. The algorithm in [7] has complexity $O(M^3K)$, and is therefore better for typical applications, in which K is $O(M)$. Our algorithm will be competitive only for $K < M^{2/3}$.

III. Asymptotic Results for $K=M$

We are interested in the asymptotic behavior of Q_0 and Q^* as $M \to \infty$, with $K = M$. Assume, as in the previous section, that the periodic pattern has been established. Then the queue length at a typical slot is written as

$$Q_0 = \max_{0 \le i < M} \{ \sum_{j=-i}^{0} a_j - i \}$$

Define

$$\phi(i) = \sum_{0}^{i} a_j - i \quad , i \geq 0 \ .$$

Note that this definition is slightly different from that of the previous section. By periodicity and interchangeability, $(a_0, a_{-1}, \ldots, a_{-(M-1)})$ has the same distribution as $(a_0, a_1, \ldots, a_{M-1})$, so we can write $Q_0 = \max_{0 \leq i < M} \phi(i)$ without changing the distribution of Q_0. The maximum queue length Q^* can also be written as a function of the $\phi(i)$. It is easy to see that

$$Q^* = \max_{s} \max_{t > s} [\phi(t) - \phi(s)] \ .$$

Since $K = M$, we have $\phi(i + j\,M) = \phi(i)$, $0 \leq i < M$, $j \geq 1$. This implies that we can express Q^* as

$$Q^* = \max_{0 \leq i < M} \phi(i) - \min_{0 \leq i < M} \phi(i) \ .$$

Let $G_M(x)$ be the empirical distribution of M independent and identically distributed samples from the uniform distribution over the unit interval. Since

$$(1/M)\phi(i-1) = (\sum_{0}^{i-1} a_j)/M - i/M + 1/M \quad ,$$

so that

$$(1/M)\phi(i-1) = (G_M(i/M) - i/M) + O(1/M) \quad , M \to \infty \ .$$

Denote $M^{1/2}(G_M(x) - x)$ by $\beta_M(x)$. It is known [4] that β_M converges weakly to the Brownian bridge W_0, which is given as

$$W_0(x) = W(x) - x\,W(1) \quad , 0 \leq x \leq 1 \quad ,$$

where W is standard Brownian motion starting at the origin. Let H be a functional on the space of CORLOL functions on the unit interval. If H is continuous with respect to the sup metric, we have $H(\beta_M) \to H(W_0)$ in distribution as $M \to \infty$. The two functionals of interest here are

$$H_1(f) = \sup_{0 \leq x \leq 1} f(x) \quad ,$$

and

$$H_2(f) = \sup_{0 \leq x \leq 1} f(x) - \inf_{0 \leq x \leq 1} f(x) \ .$$

Both these functionals are easily seen to be continuous in the sup metric.

We now have

$$M^{-1/2} Q_0 = \max_{1 \le i \le M} M^{1/2}(G_M(i/M) - i/M) + O(M^{-1/2}) \quad , M \to \infty \ .$$

Similarly,

$$M^{-1/2} Q^* = \max_{1 \le i \le M} M^{1/2}(G_M(i/M) - i/M) - \min_{1 \le i \le M} M^{1/2}(G_M(i/M) - i/M) + O(M^{-1/2}) \quad , M \to \infty \ .$$

Note that for $0 \le x \le 1$, we have

$$G_M(\lfloor xM \rfloor/M) - \lfloor xM \rfloor/M - 1/M \le G_M(x) - x \le G_M(\lceil xM \rceil)/M - \lceil xM \rceil/M + 1/M \quad ,$$

so that

$$\max_{1 \le i \le M} (G_M(i/M) - i/M) = \sup_{0 \le x \le 1} (G_M(x) - x) + O(1/M) \quad , M \to \infty,$$

and

$$\min_{1 \le i \le M} (G_M(i/M) - i/M) = \inf_{0 \le x \le 1} (G_M(x) - x) + O(1/M) \quad , M \to \infty,$$

Thus, we can write

$$M^{-1/2} Q_0 = H_1(\beta_M) + O(M^{-1/2}) \quad , M \to \infty \ ,$$

and

$$M^{-1/2} Q^* = H_2(\beta_M) + O(M^{-1/2}) \quad , M \to \infty \ .$$

It now follows that $M^{-1/2}Q_0$ converges in distribution to $H_1(W_0)$, which we denote here by X_0. Similarly, $M^{-1/2}Q^*$ converges in distribution to $H_2(W_0)$, which we denote by X^*. The distribution of X_0 is given by [9]:

$$P[X_0 \le x] = \begin{cases} 1 - \exp(-2x^2) & , x \ge 0 \ , \\ 0 & , x < 0 \ , \end{cases}$$

which yields $E[X_0] = \frac{1}{2}(\pi/2)^{1/2}$. The distribution of X^* is the same as that of the maximum of a Brownian excursion [10], and is given by [3]

$$P[X^* \le x] = \begin{cases} 1 - 2\sum_{k=1}^{\infty} (4k^2x^2 - 1)\exp(-2k^2x^2) & , x \ge 0 \ , \\ 0 & , x < 0 \ . \end{cases}$$

It is shown in [3] that $E[X^*] = (\pi/2)^{1/2}$.

Thus, roughly speaking (convergence in distribution does not imply convergence of means), Q_0 grows as $\frac{1}{2}(\pi/2)^{1/2}M^{1/2}$, and Q^* as $(\pi/2)^{1/2}M^{1/2}$ for large M. It would be interesting to examine the asymptotic behavior of these quantities for $K = M - c\,M^{\alpha}$, for $0 < \alpha < 1$. For $K/M = \lambda$, the system behavior tends to that of an $M/D/1$ queue with unit service time and arrival rate λ (see [8]). The queue becomes unstable for $\lambda = 1$, so both Q_0 and Q^* will grow with M for $K = M - c\,M^{\alpha}$. We would like to find the growth rate as a function of α. The problem is to find an appropriate normalization for $\phi(i)$. We have to account for the fact that ϕ is no longer periodic; in fact, we have $\phi(i+M) = \phi(i) - c\,M^{\alpha}$.

IV. Conclusions

We have given an $O(M\,K^4)$ algorithm for computing the maximum queue length distribution for an ATM link with periodic arrivals. When the system is operating at maximum load, i.e., $K = M$, we have shown that both the typical queue length and the maximum queue length grow as $M^{1/2}$ by showing that the asymptotic distributions of $M^{-1/2}Q_0$ and $M^{-1/2}Q^*$ are equal to those of the maximum of a Brownian bridge and Brownian excursion, respectively. These are also the limiting distributions of certain Kolmogorov-Smirnov statistics, and this was implicit in the development in Section III.

It would be interesting to examine the system behavior for large M when $K = M - c\,M^{\alpha}$ for $0 < \alpha < 1$, as pointed out at the end of the previous section. Another open problem in ATM systems is the analysis of networks of links; it is necessary to understand the output process from a single link in order to carry out such an analysis. The assumption of independent sources breaks down if the input to a given link is the output from some other link(s). In a somewhat different vein, if low priority aperiodic traffic is also present at a given link, we would like to compute the delay it suffers. If we assume that the aperiodic traffic can only use the slots that are left over after in a frame after the periodic traffic is served, the expected delay can be computed as a function of the number and spacing of the available slots [2, 6]. It can be shown, using a result of Hajek [5], that the expected delay is minimized for equally spaced slots, if the number of available slots in a frame is given.

This provides a lower bound for the expected delay. A better lower bound can be obtained by accounting for the fact (see [5]) that the spacing between the available slots has to be an integer. We conjecture that, given the number of available slots in a frame, the expected delay is maximized if there is no spacing between the available slots in a frame.

ATM is becoming increasingly popular for networks that provide integrated services, and a great deal of research effort is currently being devoted to gaining a basic understanding of ATM systems. This paper considers some analytical problems that arise in this context: some partial solutions are given, and some open problems are pointed out.

References

[1] A. Bhargava, P. Humblet and M. G. Hluchyi, "Queuing analysis of continuous bit-stream transport in packet radio networks," *Proc. IEEE Globecom 1989*, pp. 25.6.1-25.6.5.

[2] I. Cidon and M. Sidi, "Performance analysis of asynchronous transfer mode (ATM) systems," preprint, November 1988.

[3] R. T. Durrett and D. L. Iglehart, "Functionals of Brownian meander and excursion," *Ann. Probability*, vol. 5, pp. 130-135, 1977.

[4] P. Gaenssler and W. Stute, "Empirical processes: a survey of results for independent and identically distributed random variables," *Ann. Probability*, vol. 7, pp. 193-243, 1979.

[5] B. Hajek, "Extremal splittings of point processes," *Math. Operations Research*, vol. 10, pp. 543-556, November 1985.

[6] M. Hofri and Z. Rosberg, "Packet delay under the golden ratio weighted TDM policy in a multiple-access channel," *IEEE Trans. Inf. Theory*, vol. IT-33, pp. 341-349, May 1987.

[7] T. Ott and J. G. Shanthikumar, "On a buffer problem for packetized voice with an N-periodic strongly interchangeable input process," preprint, July 1989.

[8] T. Ott and J. G. Shanthikumar, "Structural properties and stochastic bounds for a buffer problem in packetized voice transmission," preprint, October 1989.

[9] R. J. Serfling, *Approximation Theorems of Mathematical Statistics.* New York: Wiley, 1980.

[10] W. Vervaat, "A relation between Brownian bridge and Brownian excursion," *Ann. Probability,* vol. 7, pp. 143-149, 1979.

Branko Radosavljevic
High Speed Computer Communication Networks
EE 497 BH
Spring 1990
Prof. Hajek

Head of Line Blocking in Packet Switches

I. Introduction

This paper contains an analysis of Head of Line (HOL) blocking in internally nonblocking packet switches. HOL blocking occurs when too many packets simultaneously request the same output. Initially, only a portion of these packets can be delivered, due to the finite capacity of each output port. The other packets must wait in a queue. This effect limits the maximum throughput of a switch, and determining this throughput is the subject of this work.

The model is defined as follows. The packet switch has N inputs and N outputs. Each output consists of r trunks or servers. When a packet requests a given output, it considers the r associated servers to be equivalent destinations. The switch is internally nonblocking, which means that blocking occurs only due to output contention. Thus, if m HOL packets request the same output, then $m \wedge r$ of them will immediately start transmission through the switch. The rest of the packets must wait in queues; a queue is associated with each input port of the switch. The method for determining which HOL packets win an output contention is arbitrary, as long as it does not depend

on the destinations of the other packets in the input queues. This decision influences packet delay, but not throughput.

Each input of the packet switch receives packets according to a Poisson process with rate λ, and arrivals at different inputs are independent. The packet destinations are independent and uniformly distributed over the set of outputs. Concerning packet lengths, two different cases are considered. For the continuous time case, packet lengths are independent and exponentially distributed with mean $1/\mu$. For the discrete time case, time is slotted, and each packet requires one slot for transmission. In what follows, the HOL queue for output i is defined to be the virtual queue consisting of the HOL packets that request output i.

In this paper, results are derived concerning the saturation throughput of a switch, which is the expected throughput when each input queue (as opposed to HOL queue) is never empty. This quantity equals the maximum throughput, which can be shown by sample path comparison. Section II considers the saturation throughput for the continuous time case, and Section III studies the discrete time case. Section II also discusses the equivalence between packet switches and closed networks of queues. Concluding remarks are presented in Section IV.

II. The Continuous Time Case

A. One server per output

This section considers the continuous time model with r, the number of servers per output, equal to one. As discussed in Section I, I assume the switch is saturated. This system can be modeled as a continuous time Markov chain, with state space

$$S = \left\{ (x_1, \ldots, x_N) : x_i \geq 0, \sum_{i=1}^{N} x_i = N \right\}$$

where x_i is the size of the HOL queue for output i. The allowable transitions take the form

$$x'_i \leftarrow x_i - 1$$
$$x'_j \leftarrow x_j + 1$$

where i must satisfy $x_i \geq 1$. This corresponds to a packet with destination i completing service (i.e., transmission) and being replaced by a packet with destination j. The corresponding rate is μ/N, since a packet completes service at rate μ, and the next packet in the input queue has destination j with probability $1/N$.

Then $\pi(s) = K$ for some constant K is a stationary distribution. This follows from the fact that if s and s' are two states in S, then $s \rightarrow s'$ implies $s' \rightarrow s$, where the arrow notation denotes one-step reachability. In addition, the transition rate is the same in both directions. Thus, $\pi(s) = K$ satisfies the

local balance equations. This shows $\pi(s)$ is stationary, as well as that the process is reversible. To determine K, note that the number of states in S equals the number of ways to put N indistinguishable balls into N distinct bins. Thus,

$$K = \binom{2N-1}{N}^{-1}$$

To determine the saturation throughput, ν, let

$$p_0 = \mathrm{P}(x_i = 0)$$

where i is arbitrary and P is calculated according to the stationary distribution π. Then

$$\begin{aligned} p_0 &= K \binom{2N-2}{N} \\ &= \frac{N-1}{2N-1} \end{aligned}$$

since the number of states with $x_i = 0$ equals the number of ways to put N balls into $N-1$ bins. Then ν is given by

$$\begin{aligned} \nu &= (1-p_0)\mu \\ &= \frac{N}{2N-1}\mu \end{aligned}$$

Note that $\nu > 1/2$, and $\lim_{N\to\infty} \nu = 1/2$. This behavior agrees with the approximate analysis performed in class.

B. Many servers per output

Now we allow r, the number of servers per output, to be greater than one. Again we model the system by a Markov chain, with the same state space S defined in Section I.A. We have the same set of allowable transitions, but now the transition defined by

$$x'_i \leftarrow x_i - 1$$
$$x'_j \leftarrow x_j + 1$$

has rate $(x_i \wedge r)\mu/N$, since $x_i \wedge r$ is the number of busy servers for output i.

The corresponding stationary distribution is given by

$$\pi(s) = \frac{1}{G} \frac{1}{\prod_{i=1}^{N} \beta(x_i)}$$

where

$$\beta(x) = \begin{cases} x! & x \leq r \\ r^{x-r} r! & x \geq r \end{cases}$$

and G, a normalizing constant, is given by

$$G = \sum_{s \in S} \frac{1}{\prod_{i=1}^{N} \beta(x_i)}$$

It is easy to verify that $\pi(s)$ satisfies the local balance equations, which implies that $\pi(s)$ is stationary and the process is reversible.

Notice that $\pi(s)$ has the form of the joint pdf of a set of N iid random variables, conditioned on the event that their sum equals N. Therefore, one can use tilted probabilities to find the limit of the marginal distribution of x_i as N goes to infinity. (This is a well-known result; see, for example, Van Campenhout and Cover (1981).) Then

$$\begin{aligned} p_k &\equiv \lim_{N\to\infty} \mathrm{P}(x_i = k) \\ &= \frac{1}{G(\alpha)} \frac{\alpha^k}{\beta(k)} \end{aligned}$$

where

$$G(\alpha) = \sum_{k=0}^{\infty} \frac{\alpha^k}{\beta(k)}$$

and α satisfies

$$\mathrm{E}[x_i] = 1$$

Then the saturation throughput ν is given by

$$\begin{aligned} \nu &= \mathrm{E}[x_i \wedge r]\, \mu \\ &= \frac{\mu}{G(\alpha)} \sum_{k=1}^{\infty} (k \wedge r) \frac{\alpha^k}{\beta(k)} \\ &= \frac{\mu}{G(\alpha)} \sum_{k=1}^{\infty} \frac{\alpha^k}{\beta(k-1)} \\ &= \alpha\mu \end{aligned}$$

I wrote a computer program to compute α given r. It uses the bisection

method, together with the following formulas.

$$G(\alpha)=\sum_{k=0}^{r-1}\frac{\alpha^k}{k!}+\frac{\alpha^r}{r!(1-\alpha/r)}$$

$$\mathrm{E}[x_i]=\frac{1}{G(\alpha)}\left[\alpha\sum_{k=0}^{r-2}\frac{\alpha^k}{k!}+\frac{\alpha^r(\alpha/r+r-\alpha)}{(1-\alpha/r)^2}\right]$$

The results are shown in Figure 1.

r	α
1	.5000
2	.8284
3	.9611
4	.9934
5	.9990

Figure 1. Normalized throughput, α, as a function of the number of servers per output, r, for exponentially distributed packet lengths.

C. Equivalent systems

This section discusses the similarities between packet switches and closed networks of queues. First, we define a closed network of Markovian queues as follows. There are M nodes and K customers. Node i consists of a queue and r_i exponential servers, each with rate μ_i. Upon completion of service at node i, a customer moves to node j with probability r_{ij}, where $\sum_{j=1}^{M} r_{ij}=1$.

Then a saturated packet switch with exponential packet lengths corresponds to a closed network of Markovian queues with $M = K = N$, $r_i = r$, $\mu_i = \mu$, and $r_{ij} = 1/N$. Specifically, the two systems behave as continous time Markov chains with the same state space, the same set of allowable transitions, and the same transition rates. For the packet switch, the state is defined as in Section I.B, where the state equals $(x_1, \ldots, x_N)$ and x_i is the size of the HOL queue for output i. For the network of queues, x_i is simply the queue size at node i.

An unsaturated packet switch can be modeled as a closed network of Markovian queues as well. Here we assume exponential packet lengths and Poisson arrivals to the switch with rate λ. The corresponding network of queues is the same as that for a saturated switch, except that an extra node is added with an infinite number of rate λ servers to model the arrival process.

Networks of Markovian queues were studied by Gordon and Newell (1967). They determined the unique stationary distribution for the associated Markov chain, and also considered the limiting behavior of the marginal distribution of a subset of the state vector as M and K go to infinity. The stationary distribution is given by

$$P(x_1, \ldots, x_M) = G^{-1} \prod_{i=1}^{M} \frac{\alpha_i^{x_i}}{\beta_i(x_i)}$$

where G is a normalizing constant, α_i satisfies

$$\mu_i \alpha_i = \sum_{j=1}^{M} \mu_j x_j r_{ji} \qquad i = 1,2,\ldots,M$$

and β_i is the function β defined in Section I.B with $r = r_i$. This result agrees with those presented in Sections I.A and I.B. Jackson (1963) considered an open network of Markovian queues where the customer arrival process depends on the total number in the system, and the service rate at a node depends on the number on the number of customers at that node. This includes the closed network as a special case.

Packet switches with packet length distributions other than exponential can be modeled as networks of queues as well. The queues have service time distribution equal to the packet length distribution. However, it is difficult to analyze general networks of non-Markovian queues.

III. The Discrete Time Case

We now assume time is slotted, and each packet requires one time slot for transmission. For a given output i, the dynamics of its HOL queue are given by

$$q_{n+1} = q_n - q_n \wedge r + v_{n+1}$$

where q_n is the size of the HOL queue at time n, r is the number of servers per output, and v_{n+1} is the number of new arrivals at time $n+1$ (i.e., new HOL packets that request output i). To simplify the analysis, I assume v_{n+1} is Poisson with mean λ and independent of q_n. This follows the approach

taken by Hui and Arthurs (1987) for the case $r = 1$. This is a reasonable assumption for large N because then the number of empty input queues is approximately constant. If it were constant, then v_{n+1} is the sum of a large number of improbable events, which is approximately Poisson. That the mean of v_{n+1} equals λ follows from stability.

To determine ν, the saturation throughput, we will determine the minimum value of λ for which saturation occurs. Equivalently, this is the smallest value of λ for which $E[q]$, the steady state expected queue size, equals one (Section I). Then $\nu = \lambda$. For $r = 1$, the dynamic equation for q_{n+1} is the same as for an M/D/1 queue, which is well understood. However, when $r > 1$, the update equation corresponds to that of an M/D/r queue with gated service (i.e., service starts only at slot boundaries). As far as I know, there is no closed form solution for $E[q]$; no results in this direction are presented in the books by Kleinrock or Kelly. In general, the queue M/G/r is not well understood (as opposed to G/M/r).

The random process $(q_n)_{n \geq 0}$ is a discrete time Markov chain. A stationary distribution vector π must satisfy $\pi = \pi P$, where P is the transition matrix. One way to solve for π is through the method of successive approximations. Start with an initial distribution $\pi^{(0)}$, and successively compute

$$\pi^{(n+1)} = \pi^{(n)} P$$

Expanding the matrix P, we obtain

$$\pi^{(n+1)}(k) = \sum_{i=0}^{r-1} \alpha_k \pi^{(n)}(i) + \sum_{i=r}^{k+r} \alpha_{k+r-i} \pi^{(n)}(i)$$

where

$$\alpha_k = e^{-\lambda} \frac{\lambda^k}{k!}$$

is the Poisson distribution.

A computer program was written to implement this procedure. Given r and λ, it iterates to find the stationary distribution, and uses this to compute $E[q]$. Due to finite memory and processing time, only the first one hundred elements of $\pi^{(n)}$ were stored. The iterations stop when there is no change in $E[q]$ (with resolution 10^{-7}). The bisection method was then used to find ν, which is the value of λ that gives $E[q] = 1$. The results are shown in Figure 2. The result for $r = 1$ agrees with the value obtained by Hui and Arthurs.

r	ν
1	.5858
2	.8845
3	.9755
4	.9956
5	.9993

Figure 2. Saturation throughput, ν, as a function of the number of servers per output, r, for slotted time and fixed length packets.

The following formula was used to compute E[q].

$$\mathrm{E}[q] = \frac{rA_1 - A_2 + r\lambda - \lambda^2 + \lambda}{2(r-\lambda)}$$

where

$$A_i = \sum_{k=0}^{r-1} k^i \pi(k) \qquad i \geq 0$$

This expression has the advantage that it does not require an infinite sum to be evaluated. It is obtained by squaring both sides of the dynamic equation for q_{n+1}, taking expectations, and assuming steady state behavior. It also uses the relation

$$rA_0 - A_1 = r - \lambda$$

which is obtained by simply taking expectations of both sides of the dynamic equation for q_{n+1} and assuming stability. It would be desirable to obtain a closed form expression for $rA_1 - A_2$ in terms of r and λ, because then it would not be necessary to determine the distribution π in order to obtain E[q]. I have not obtained this. However, it is known that $rA_1 - A_2$ converges to $r\lambda - \lambda^2 - \lambda$ for $r \gg \lambda$.

IV. Conclusions

In conclusion, this paper presented an analysis of Head of Line (HOL) blocking in an internally nonblocking packet switch with r servers per output. Results were obtained concerning ν, the saturation throughput. For the

continuous time case with exponential packet lengths and $r=1$, ν was obtained as a function of N, the number of switch inputs. For $r>1$, the asymptotic value of ν was obtained as N goes to infinity. In addition, the packet switch was shown to behave equivalently in a sense to a closed network of queues. For the discrete time case with fixed length packets, an approximate analysis was performed to determine the asymptotic value of ν for large N.

For both the continuous time and discrete time models, ν quickly approached 1 with increasing r, and $\nu \geq .999$ for $r=5$ (both cases). The discrete time value was always higher, but the relative improvement decreased with increasing r. The discrete time case may perform better because of zero variance in the packet lengths, unlike the exponential packet length case.

As stated before, the analysis for the discrete time case was approximate. For future work, it would be desirable to obtain similar results through an exact analysis. The author conjectures that the results obtained through the approximate analysis are correct asymptotically in N.

References

Gordon, W. J. and G. F. Newell, "Closed Queueing Systems with Exponential Servers," *Operations Research*, 15, pp. 254-265 (1967).

Hui, Joseph Y. and Edward Arthurs, "A Broadband Packet Switch for Integrated Transport," *IEEE Journal on Selected Areas in Communications*, SAC-5, pp. 1264-1273 (1987).

Jackson, J. R., "Jobshop-Like Queueing Systems," *Management Science*, 10, pp. 131-142 (1963).

Van Campenhout, J. M. and T. M. Cover, "Maximum Entropy and Conditional Probability," *IEEE Trans. on Information Theory*, IT-27, pp. 483-489 (1981).

Matthew T. Busche
May 10, 1990
EE 497

SWITCHING ARCHITECTURE OF THE 5ESS™

INTRODUCTION

The 5ESS switching system is the first 'world class' switch, capable of operating in both American and European networks [1 and 2] despite differences in operations standards in these two environments. The 5ESS is also one of the first switches designed to support all functions required for ISDN data formats and protocols.

This paper examines the underlying Time-Space-Time architecture of the 5ESS. All major elements affecting end to end data flow are examined. The paper closes with a brief discussion of the 5ESS switching capacity.

The 5ESS has a distributed architecture based on three major components: an administrative module (AM), a communications module (CM) and a number of switching modules (SM). See Figure 1. Each SM contains a time slot interchanger (TSI) and the CM contains a time multiplexed switch (TMS). The CM and SMs are networked together to give the 5ESS its time-space-time (TST) switching function.

With special interface equipment a SM can be configured to operate remotely (up to 100 miles away). The remote switching module (RSM) interfaces with a host SM rather than the CM. The RSM provides an economical means to support customers in sparsely populated areas.

1. ADMINISTRATIVE MODULE

The AM provides interfaces to operate and maintain the switch. Processing is performed by the administrative processor (AP), an AT&T 3B20D computer. The AP is fully duplicated for reliability. If a fault occurs in the active AP control is transferred to the backup with no loss of data.

The call-processing functions performed by the AP are routing and time slot allocation. Routing is the determination of the destination SM from dialed digits. If a call is destined for a trunk group (as opposed to a customer line) the AP will also select an available trunk within the group as a part of the routing function.

The AP has a disk memory for storage of call data, routing data and programs used by the AP and SMs. The AP also interfaces with an I/O Processor which supports video display units, hard copy printers, magnetic tape drives and a Master Control Center (MCC).

The MCC provides the 5ESS with a man-machine interface. Functions performed by the MCC include system status display and manual control of system operations.

2. COMMUNICATIONS MODULE

The CM contains two 32x32 (i.e. 32 input and 32 output) single stage time multiplexed switches (TMSs). Each TMS is connected to each SM by two fiber optic links--one input link and one output link. Thus a SM is connected to the CM by four fiber optic links--two links for each TMS within the CM. See Figure 2. Both TMSs and all fiber optic links are duplicated for reliability. So there are actually four TMSs in the CM of which two are active. Similarly each SM is actually connected to the CM by eight fiber optic links of which four are active.[1]

Each link transmits data at a rate of 32.768 Mb/s. Data on the links is partitioned into frames. Frames are further partitioned into 256 time slots, with each time slot carrying 16 bits of data. 8000 frames per second are carried over the links. Each TMS changes its connection pattern 2.048 million times per second--one switch setting for each time slot.

One input and output on each TMS is reserved for testing. Another input and output on each TMS is used by the message switch (MSGS) as described below. The remaining 30 inputs and outputs may be used by SMs. Thus, the CM may support up to 30 SMs as depicted in figure 2.

Since there is no need for unidirectional channels in a telephone network, all data channels are bidirectional (i.e. full duplex). Unidirectional data paths between SMs are set up in pairs. Here, a unidirectional data path from SM A to SM B refers to a fixed time slot in each frame of one of the output links from A and the same fixed time slot on the corresponding input link to B. A channel between two SMs A and B consists of two unidirectional data paths, one from A to B and one from B to A. The two data paths use the same time slot through the TMS. With 256 time slots on the fiber optic links and one link to and from each TMS, a SM has the potential to support five hundred twelve 128 kb/s full duplex channels.

In addition to the two TMSs, the CM also contains a message switch (MSGS). Like a SM the MSGS has one input link and one output link for each TMS. The MSGS is a relay station for control messages sent from SM to SM and from SMs to the AM. Every SM is assigned a unique control time slot from the available 256 time slots. This time slot is fixed.

Messages carried over the control channels are sent in packet form by the CCITT X.25 level 2 protocol [1, pg. 1342]. Types of control messages carried over these control channels include call routing requests, time slot allocation directives, and synchronization signals.

Data originating from a SM on a control time slot is automatically switched through the TMS to the MSGS. To send control messages to a SM, the MSGS need merely send the message to the TMS in the control time slot associated with the destination SM. Every SM thus has two permanent channels (one through each TMS) with the MSGS.

The network clock is responsible for synchronization. It also resides in the CM and interfaces with the MSGS. Buffering between the CM and SMs allow for up to one full frame of delay so cable length matching is not required [1, pg. 1425].

3. SWITCHING MODULE

[1] High reliability requirements of telephone switches necessitate the redundancy of critical elements within the switch. It is highly improbable, for example, that without a good deal of knowledge about the construction of a switch, that one could bring it down with a single blow of a hammer.

The SM consists of three major sections: the control unit, the time slot interchanger unit (TSIU) and peripheral equipment. See figure 3. SMs will vary in the type and quantity of peripheral equipment depending upon the characteristics of the lines and trunks terminating on the SM. The TSIU and control unit are the same in all SMs however, and since these units are critical to the operation of the SM, components within them are fully duplicated for reliability.

3.1 CONTROL UNIT

The control unit is based on a 32-bit processor. 95% of call processing functions required by the lines and trunks terminating on a SM are performed by the control unit (with the remainder being performed by the AP). The control unit also provides maintenance functions and control over the peripheral equipment.

3.2 TIME SLOT INTERCHANGER UNIT

The TSIU contains a duplicated full duplex TSI. Functionally the TSI performs the job of two unidirectional TSIs with the input-output permutation of one just the inverse of the other. Figure 4 depicts the major function of the TSIU which is to provide switching between peripheral equipment and the CM. Note that the TSI has been separated into two unidirectional TSIs for clarity.

Each frame through the TSI has 512 time slots of 16 bits each for a data rate of 65.536 Mb/s. The dual link interface (DLI) connects the TSI to the four multimode fiber optic links which in turn are connected to the CM. The DLI uses LEDs for optical transmission on two of the links and a pin photo diode as a light transducer for reception on the other two links. Information is transmitted in non return to zero format at 32.768 Mb/s.

Serial output from the TSI directed to the CM is demultiplexed in the DLI onto the two data links by sending odd time slots on one data link and even time slots on the other. Frames on the data links then have 256 time slots. Input links to the TSI from the CM carry data in the same format, which the DLI multiplexes into a serial data path for use by the TSI.

The data interface (DI) connects 32 peripheral interface data buses (PIDBs) to the TSIU. Each PIDB has 32 channels. Peripheral units may use one or more PIDBs for connection to the TSIU. 32 PIDBs with 32 channels each introduce 1024 channels to the TSIU. The TSI can only operate on 512 of these channels. The DI multiplexes the 32 data busses and performs a 2:1 concentration function generating 512 serial channels for the TSI.

Often calls originating at a SM are destined for another customer homed on the same SM. Instead of routing these calls to the CM and back again, a special switching function is provided in the TSIU allowing these types of calls to loop back to the peripheral units. See figure 5. In the figure the two directions of data flow have again been separated for clarity.

The TSIU also provides special switched access for the Local Digital Service Unit (LDSU). The LDSU is equipped with a universal tone generator (UTG) that provides a large variety of call progress tones, including dial tone ringing tone, busy tone, congestion tone, call-waiting tone, as well as multi-frequency tones used on inter-exchange circuits. A universal tone detector (UTD) detects MF as well as dual tone multi-frequency (DTMF) signals. The LDSU can also be equipped with recorded announcement circuits.

Figure 5 also shows two channels used for control data entering the TSIU from the control unit. Note that these channels bypass the TSI and are connected directly to the DLI. The

DLI replaces two consecutive time slots from the TSI with the control data. The two control channels traverse different data links, but in the same time slot. In the other direction the DLI extracts data from the control time slot on each link and directs it to the control unit.

3.3 PERIPHERAL EQUIPMENT

Four types of line and trunk peripheral units are available. Line units (LUs) interface to analog subscriber lines. Analog trunk units (ATUs) interface to analog trunks. Integrated services line units (ISLUs) interface to both analog and digital subscriber lines. Digital trunk units (DTUs) interface with digital trunks, RSMs and subscriber multiplex systems.[2] It is important to note that much circuitry within peripheral equipment must be dedicated to the lines or trunks that they serve. Thus to serve a large number of customers, a large number of peripheral circuits are required. The dedicated circuitry required in peripheral equipment generates most of the hardware expense for the switch.

3.3.1 LINE UNITS

LUs terminate all circuits in the subscriber line category, including payphone lines and private automatic branch exchange (PABX) lines. The LU interfaces with the TSIU through two PIDBs. A LU provides all circuitry to provide subscriber lines with the BORSCHT [3] services. The BORSCHT services are:

Battery	The provision of power to a customer line.
Overvoltage protection	Protection of the LU and the rest of the switch from lighting hits on customer lines and other high voltages generated by the switch itself (e.g. ringing and test signals can exceed 100 volts).
Ringing	Ringing includes, 20 Hz 88 V_{RMS} ringing voltages and 130 Volt coin control pulses.
Supervision	Supervision determines when a customer has gone off hook.
Coding and decoding	A to D and D to A conversion.
Hybrid	Customer lines carry a two way data path over two wires. Hybrid circuits provide 2 wire to 4 wire conversion.
Testing	Customer lines are regularly tested to detect impairments and to measure impedance and losses.

Figure 6 shows a diagram of a LU. The LU consists of 64 channel circuits, an access network, high level service circuits (HLSCs) and a concentration network.

The 64 channel circuits permit up to 64 simultaneous customer connections. Each channel circuit provides the BOCH portion of the BORSCHT services for an active line. Channel

---

[2] A subscriber multiplex system multiplexes several customer lines onto one service circuit. [3, 341-342]

circuits also contain supervision circuitry which periodically pole inactive customer lines to detect off hook status.

HLSC's perform the ringing and test functions. The HLSC's interface with the analog circuits through a space division access network.

The concentration network is constructed of gated diode crosspoints (GDX). GDX is a solid state silicon based 590 volt technology. High voltage circuitry is needed to switch power and ringing and test voltages and also to protect from lightning. The GDX network provides concentration ratios from 2:1 to 10:1. With a 10:1 concentration ratio the LU can support a maximum of 640 subscriber lines.

3.3.2 ANALOG TRUNK UNIT

The ATU uses two PIDBs to provide terminations for 64 analog voice-frequency trunks. Each trunk has an associated trunk circuit which provides A to D and D to A conversion, dc signalling and test access functions. The ATU also has a set of common circuits that perform testing, alarming and multiplexing functions.

3.3.3 INTEGRATED SERVICES LINE UNIT

An ISLU allows for the connection of both analog and digital subscriber lines. The ISLU has the same components as a LU to provide analog service, but in addition it provides an interface for the ISDN 144 kb/s basic rate interface (BRI). [4] The ISLU can support a maximum of 512 lines analog or digital. Concentration ratios for digital lines can be varied from 2:1 to 16:1. Analog concentration may be varied from 1:1 to 8:1. Another difference between the ISLU and the LU is that maximum number of terminations is not affected by the concentration ratio. Rather the number of PIDBs is affected. The number of PIDBs may vary from 2 to 16.

3.3.4 DIGITAL TRUNK UNIT

The DTU interfaces with 1.544 MB/s T1 facilities (or 2 Mb/s facilities for European applications). The primary function of the DTU is to convert the bipolar T1 carrier format into a unipolar format and distribute the T1 carrier frame onto a 32 time PIDB. The 24 PCM time slots from the T1 line are evenly distributed over the 32 time slots available on a PIDB. Idle code is used to fill the remaining time slots.

The DTU provides the 1.544 Mb/s ISDN primary rate interface (PRI) and connection of the 5ESS to other exchanges. Another important function of the DTU is interface with a RSM. See below.

4. REMOTE SWITCHING MODULE

The RSM provides economical service to customers in sparsely populated areas up to 100 miles from the rest of the 5ESS. The RSM is simply a SM with special circuits to connect it to a host SM. The data links from the RSM's DLI are converted to T1 data format and transmitted across T1 facilities to the host SM. A maximum of 20 T1 lines with 24 channels each may connect the RSM to its host SM. The RSM can support a maximum of 4096 lines with a concentration ratio of 8:1.

5. 5ESS CAPACITY

5.1 TERMINATION CAPACITY

The number of lines and trunks that can be terminated on the 5ESS is limited by the PIDBs. A trunk uses a dedicated channel on a PIDB while a line can share a PIDB channel with several other lines. The total number of lines and trunks will depend on the concentration ratio and the ratio of lines to trunks. Figure 7 summarizes the termination capacity of a 5ESS using 30 SMs for various line concentration ratios.

The TMS in the CM can be expanded to support a maximum of 190 SMs. Figure 8 summarizes the termination capacities for the 190 SM exchange.

5.2 TRAFFIC CARRYING CAPACITY

With 512 channels available, a SM can handle approximately 450 erlangs of traffic. The TMS is the bottle neck for the 5ESS as a whole. With 190 SMs, the 5ESS can handle approximately 45,000 erlangs of traffic.

5.3 CALL-PROCESSING CAPACITY

The 5ESS can process approximately 300,000 busy-hour calls, although call processing capacity depends largely on signaling arrangements, call mix and feature options selected.

CLOSING COMMENTS

Although this paper examined only the 5ESS it should be noted that most telephone switches are similar in architecture. The time-space-time architecture for example is very popular (e.g. AT&T's System 85 PBX). All switches that are designed to serve customer lines must provide the BORSCHT services and interface with analog trunks requires A to D and D to A conversion. Thus, peripheral equipment among different telephone switches is similar. In short, the uniform functionality of a telephone switches to a large extent forces uniformity in architecture.

REFERENCES

1 W.S. Hayward, editor, Special issue on the 5ESS Switching System, *AT&T Technical Journal*, vol-64 no. 6 Part 2, July-August 1985.

2 AT&T and Phillips Telecommunications promotional document, "5ESS-PRX Digital Switching System, The Network Machine" AT&T en Philips Telecommunicatie, Bedrijven B.V., P.O. Box 1168, 1200 BD Hilversum, The Netherlands

3 R. F. Key, Editor, *Engineering and operations in the Bell System*, second edition, AT&T Bell Laboratories, 1986

4 A.S. Tanenbaum, *Computer Networks*, second edition, Prentice Hall, Englewood Cliffs, 1988

REFERENCES DESCRIBING OTHER TELEPHONE SWITCH ARCHITECTURES

5 A. E. Spencer *et. al.*, Special issue on thee 4ESS switch, *Bell System Technical Journal*, vol-61, no. 4, Sept. 1977.

6 C. D. Weiss, *et. al.*, Special issue on the System 75 Digital Communications System, Bell System Technical Journal, vol-64, no. 1 part 2, January 1985.

ACRONYMS AND ABBREVIATIONS

A to D	Analog to Digital
AM	Administrative Module
AP	Administrative Processor
ATU	Analog Trunk Unit
BRI	Basic Rate Interface
BORSCHT	Battery, Overvoltage, Ringing, Supervision, Coding and decoding
CCITT	International Telegraph and Telephone Consultative Committee
CM	Communications Module
D to A	Digital To Analog
DI	Data Interface
DLI	Dual Link Interface
DTMF	Dual Tone Multi-Frequency
ISDN	Integrated Services Digital Network
ISLU	Integrated Services Line Unit
LU	Line Unit
LDSU	Local Digital Service Unit
MCC	Master Control Center
MF	Multi-Frequency
MSGS	Message Switch
PABX	Private Automatic Branch Exchange
PIDB	Peripheral Interface Data Bus
PRI	Primary Rate Interface
RSM	Remote Switching Module
SM	Switching Module
TMS	Time Multiplexed Switch
TSI	Time Slot Interchanger Unit
TST	Time-Space-Time
TU	Trunk Unit
UTD	Universal Tone Detector
UTG	Universal Tone Generator

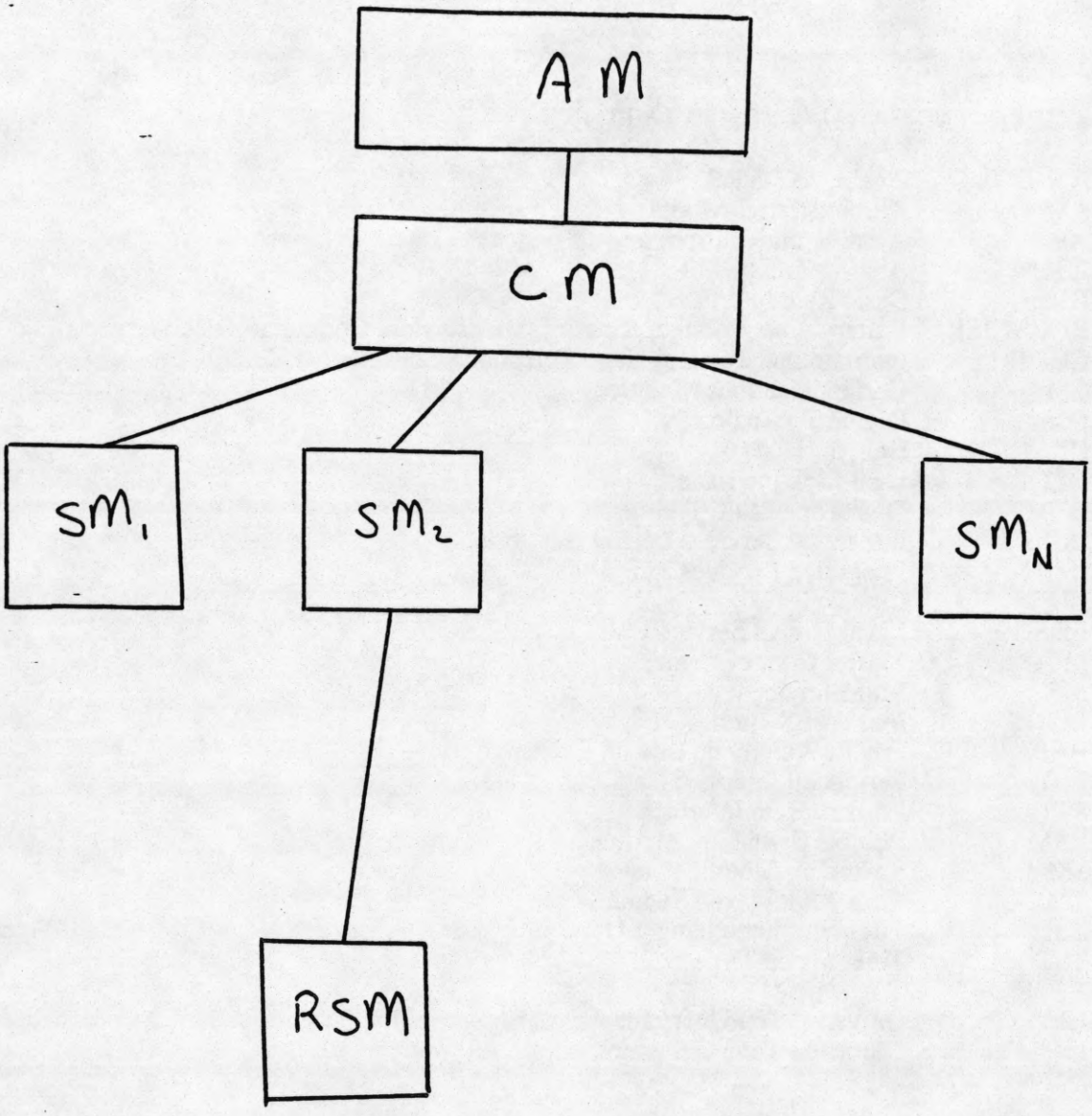


FIGURE 1. 5ESS HIGH LEVEL ARCHITECTURE

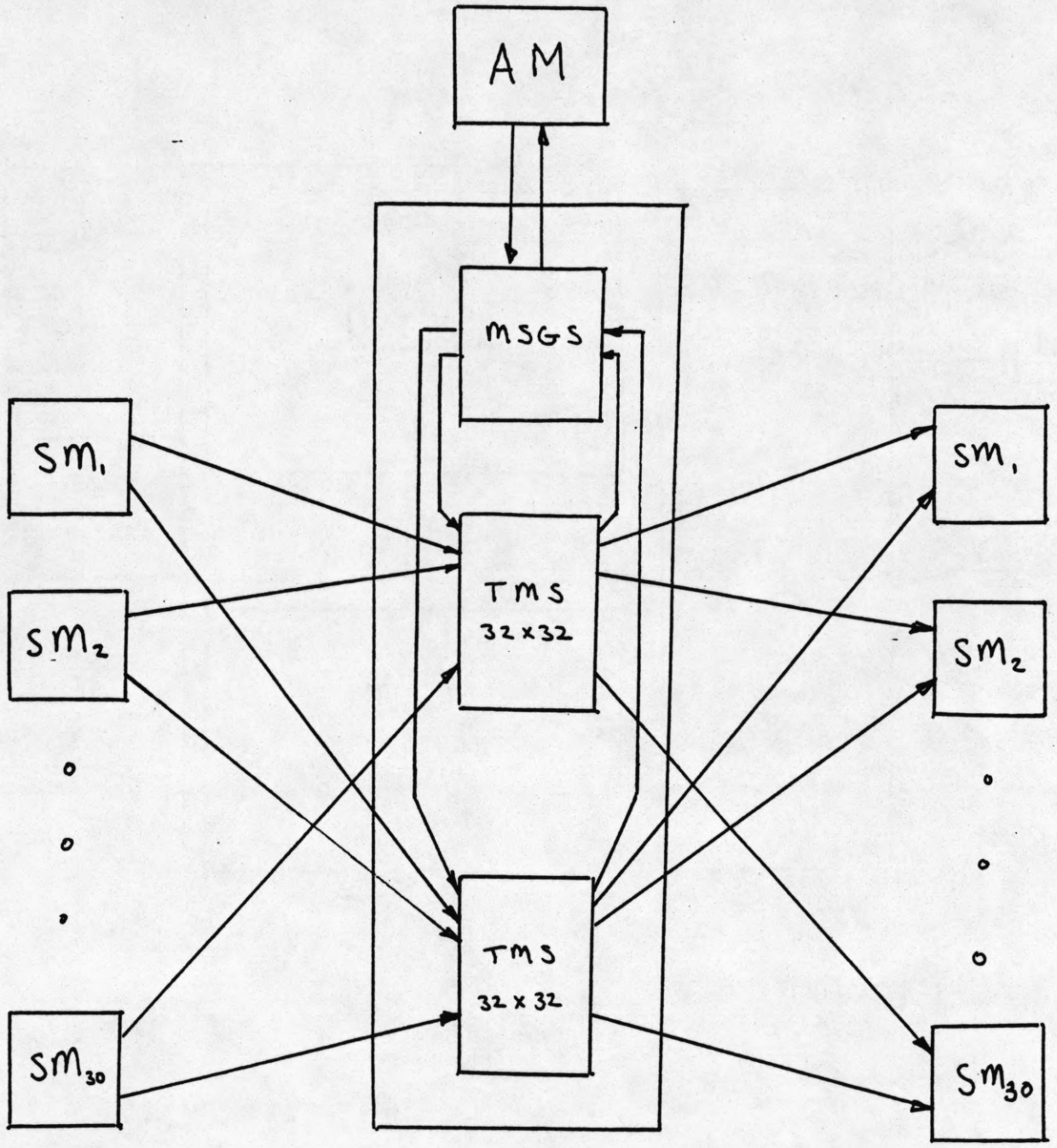


FIGURE 2. THE COMMUNICATIONS MODULE

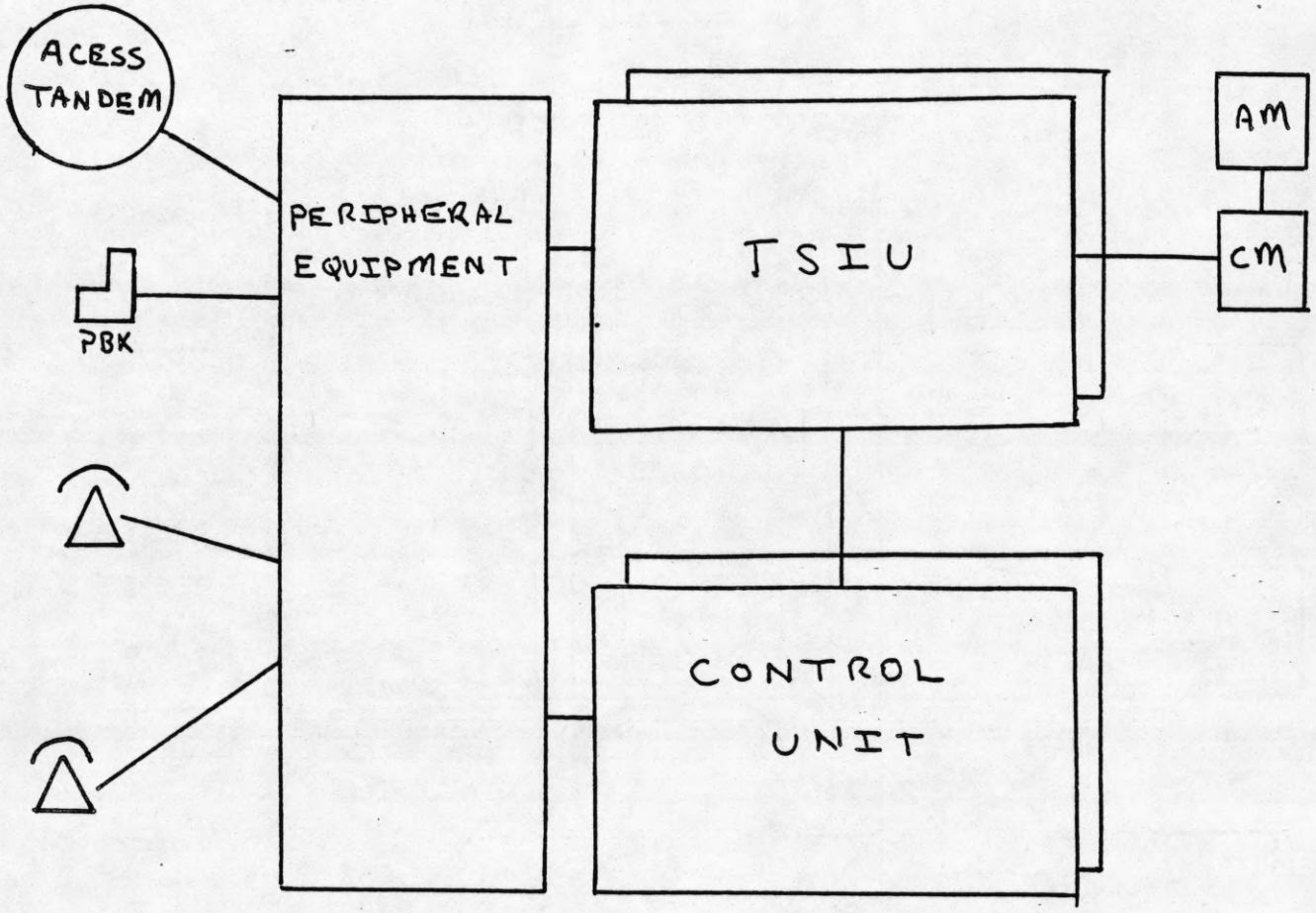


FIGURE 3
THE SWITCHING
MODULE

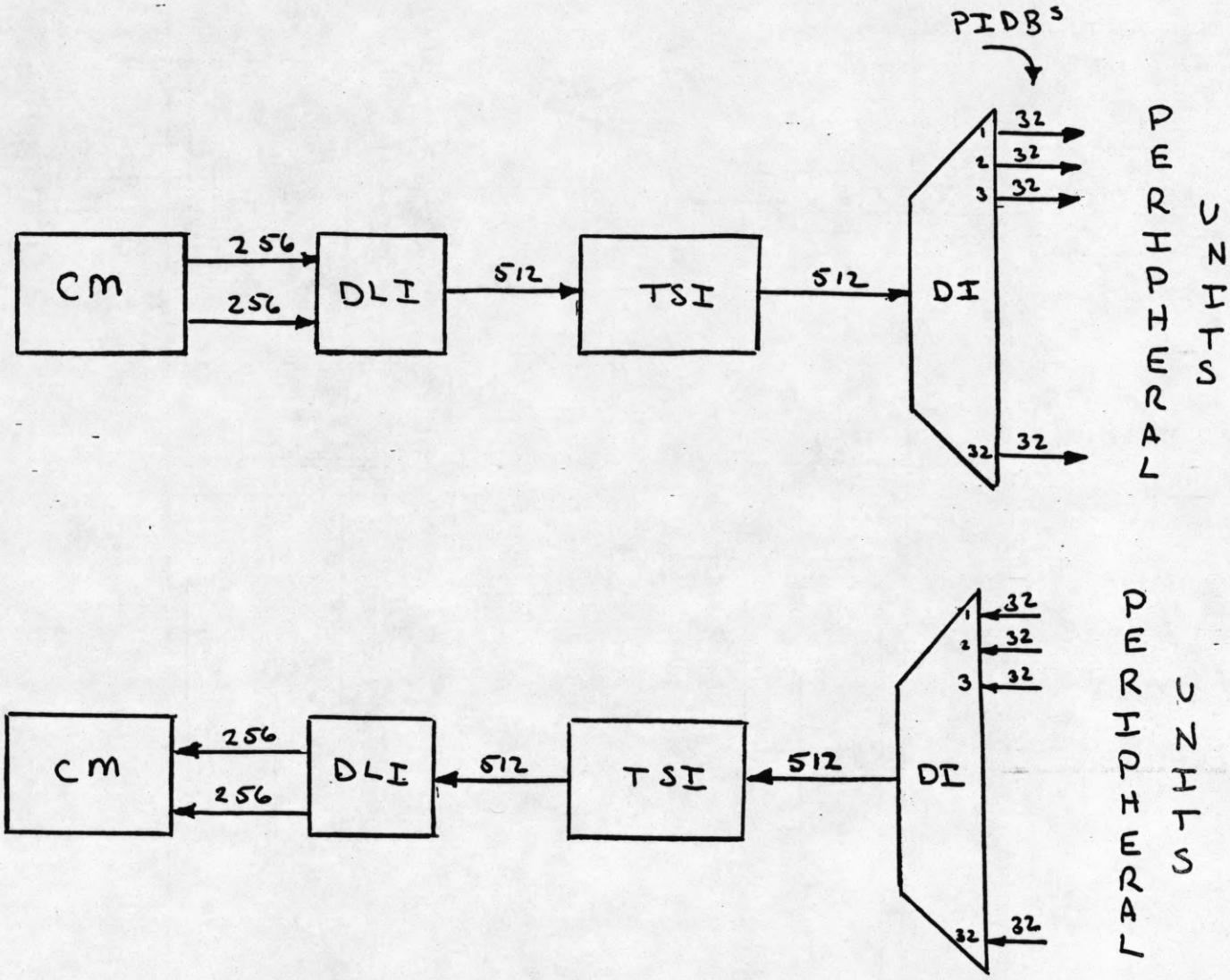


FIGURE 4

MAJOR SWITCHING FUNCTION OF THE TSIU

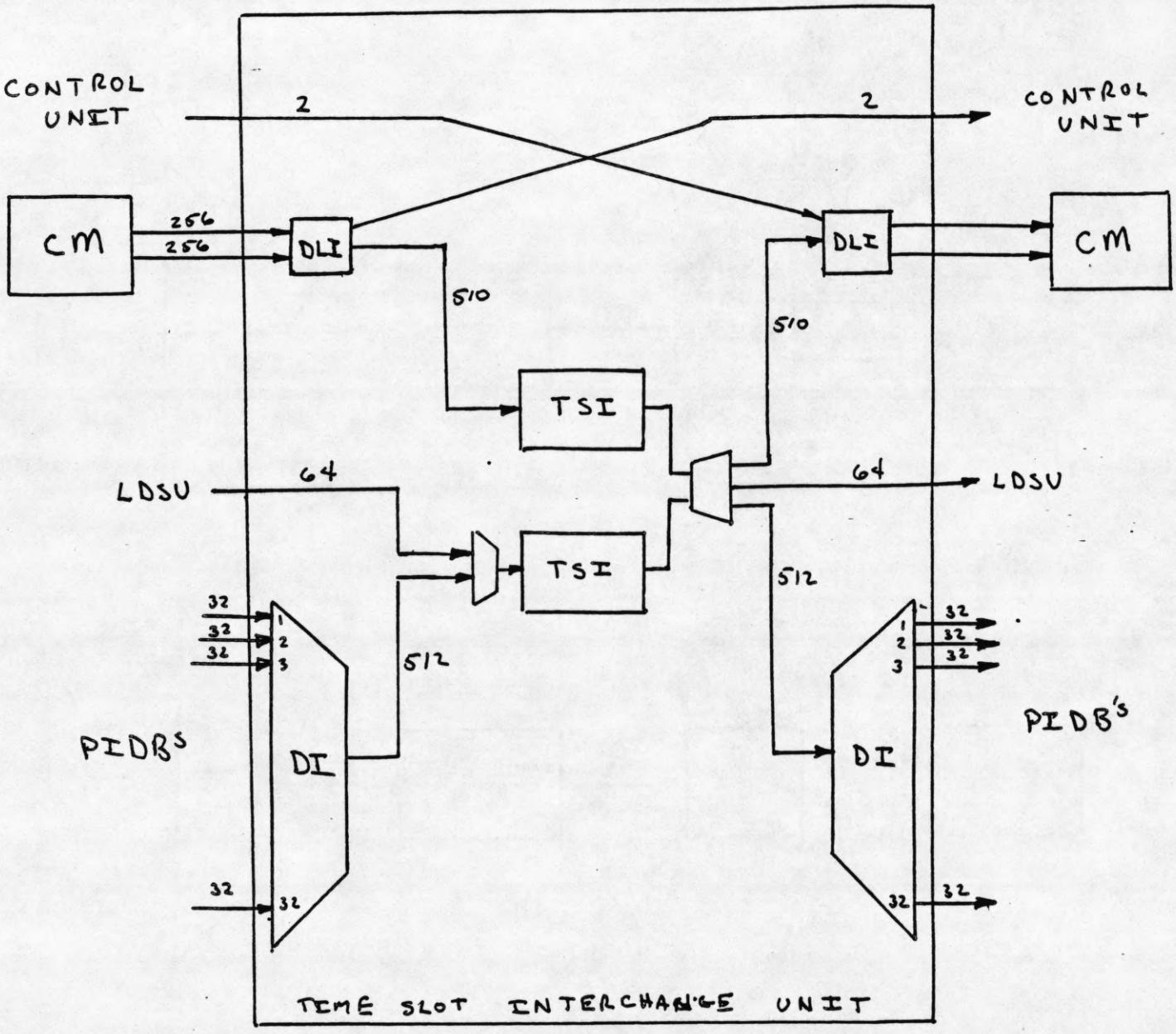


FIGURE 5

THE TSIU

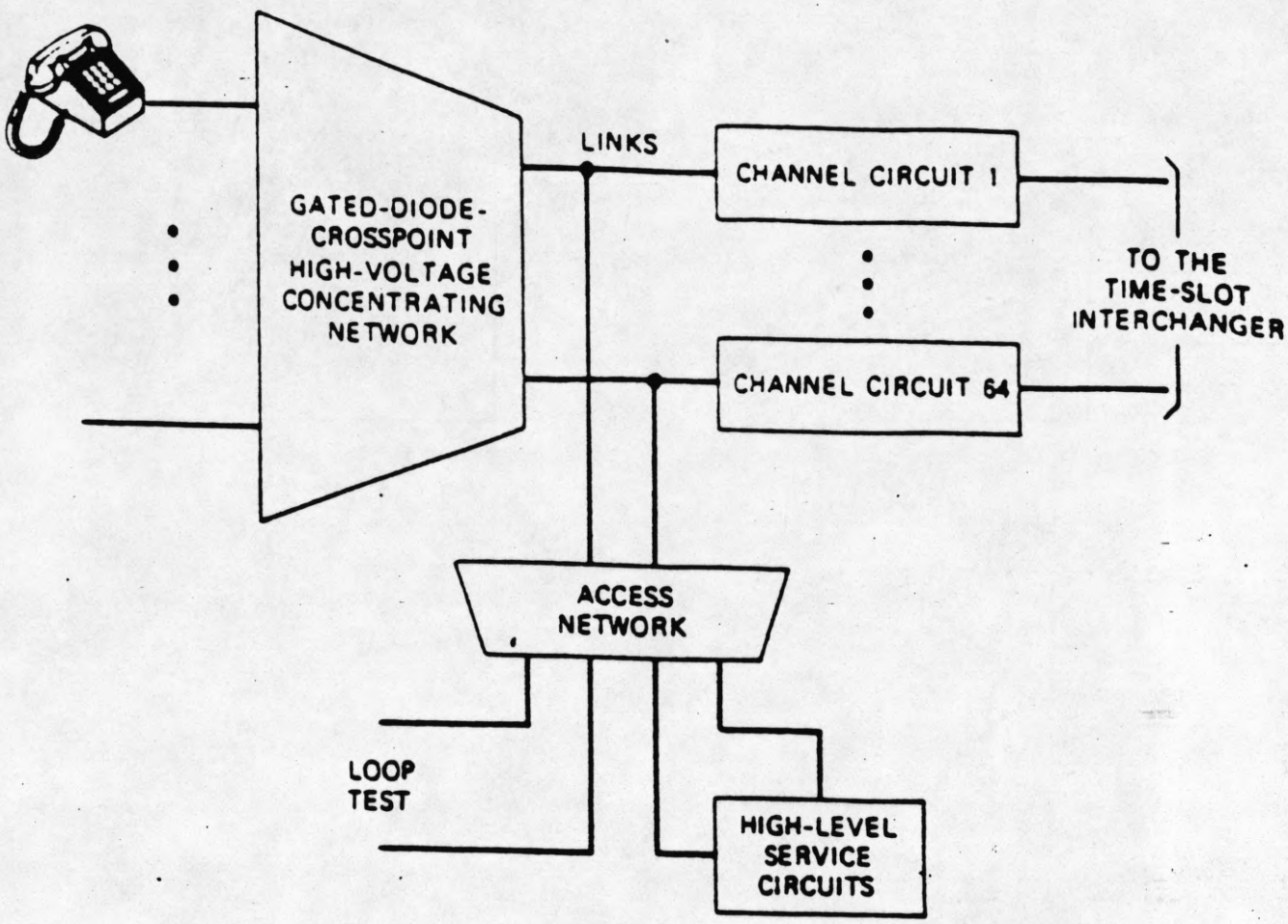


Fig. 6—Subscriber line unit.

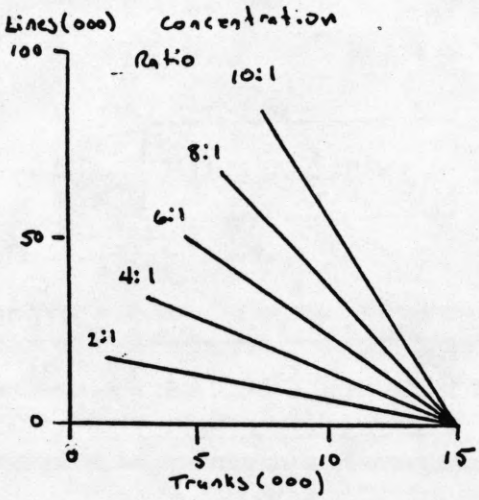


Figure 7
Line & Trunk Capacities for 30 SM exchange.

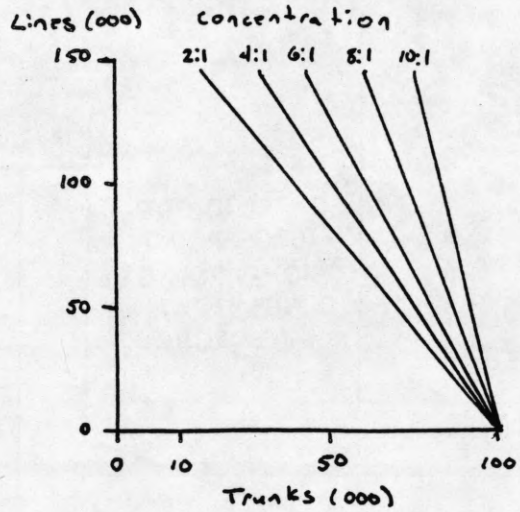


Figure 8
Line & Trunk capacities for 190 module exchange