Efficient Routing Schemes for Multiple Broadcasts in Hypercubes

George D. Stamoulis and John N. Tsitsiklis, *Member, IEEE*

Abstract—**We analyze the following problem: Each node of the binary hypercube independently generates packets according to a Poisson process with rate λ; each of the packets is to be broadcast to all other nodes. Assuming unit packet length and no other communications taking place, we observe that the system can be stable in steady-state only if the load factor $\rho \overset{\text{def}}{=} \lambda(2^d - 1)/d$ satisfies $\rho < 1$, where d is the dimensionality (diameter) of the hypercube; moreover, we establish some lower bounds for the steady-state average delay D per packet. We devise and analyze two distributed routing schemes which are efficient, in the following sense: Stability is maintained for all $\rho < \rho^*$, where ρ^* does not depend on the dimensionality d of the network, while the average delay D per packet satisfies $D \leq Kd(1 + \rho)$ for small values of ρ (with constant K). The performance evaluation is rigorous for one scheme, while for the other we resort to approximations and/or simulations.**

Index Terms—**Communications, dynamic, hypercubes, multinode broadcasts, parallel computation, queueing theory, stochastic.**

I. Introduction

During the execution of parallel algorithms in a network of processors, it is often necessary that one of the processors broadcasts a message to all others; subsequent broadcasts (possibly by different processors) may also take place, until the algorithm terminates. In this paper, we consider a problem where the nodes (i.e., processors) of the hypercube network generate packets to be broadcast at random time instants. We propose several routing schemes for performing these broadcasts and we analyze their throughput and delay properties in steady-state.

The context of our analysis is the *d-dimensional binary hypercube* (or *d*-cube); the definition and main properties of this network are presented in Section II. The underlying assumptions for communications are as follows: The time axis is divided into slots of unit length; all nodes are following the same clock. Each message is transmitted as a packet of unit length. Only one packet can traverse an arc per slot; all transmissions are error-free. Each node may transmit packets through all of its output ports and at the same time receive packets through all of its input ports. Moreover, each node has infinite buffer capacity.

Manuscript received July 25, 1990; revised May 24, 1991 and April 12, 1992. This work was supported by the NSF under Grant ECS-8552419, with matching funds from Bellcore Inc. and Du Pont, by the ARO under Grant DAAL03-86-K-0171, and by a Fellowship from the Vinton Hayes Fund.

The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139.

IEEE Log Number 9209549.

The problem to be analyzed is as follows: Each node of the d-cube generates packets according to a continuous time Poisson process with rate λ; different nodes generate their packets independently of each other. All packets generated are to be broadcast to all nodes; it is assumed that no other packet transmissions are taking place in the network.

As will be proved in Section III-A, the inequality

$$\rho \overset{\text{def}}{=} \lambda \frac{2^d - 1}{d} < 1$$

is a *necessary* condition for *stability*; ρ will be called the *load factor* of the system. Thus, for $\rho \geq 1$, the total number of packets whose broadcast is not completed grows to infinity as time elapses. Henceforth, it is assumed that $\rho < 1$. Clearly, the simplest approach to our problem is for each node to choose a spanning tree rooted at itself and broadcast all of its packets along that tree. However, depending on the choice of the trees used, the stability region may vanish as d increases, which constitutes a very poor performance. Thus, we are interested in devising routing schemes maintaining stability for all $\rho < \rho^*$, with ρ^* being a constant independent of d (also, $\rho^* < 1$). In Section IV-B, we propose schemes that are stable even when the traffic is high (namely, for $\rho \approx 1$); the underlying idea is to perform multinode broadcasts *periodically*. Unfortunately, these periodic schemes introduce very high delay even in light traffic (namely, for $\rho \approx 0$). Thus, in addition to a nonvanishing (for $d \to \infty$) stability region, we also require that the average *delay* D induced per packet satisfy $D \leq Kd(1 + \rho)$ for small values of ρ (with K being constant). Note that D is defined as the steady-state average time spent by a packet in the system until its broadcast is completed. This requirement for the delay is motivated by the fact that it takes d time units to perform a single node broadcast in the d-cube in the absence of other transmissions. (Recall that the diameter of the d-cube equals d; see also Section II-A.) Thus, it is desirable that contention does not increase this delay by more than a factor depending on the load of the network. We are mainly interested in the case of light traffic, because two *lower* bounds for D (derived in Section III-B) imply that the delay is necessarily large under heavier traffic. In Sections IV-C and V, we present two *distributed* routing schemes that meet the aforementioned performance objectives. The scheme discussed in Section IV-C is characterized as *direct*, because each packet is broadcast along a spanning tree rooted at the node where it was generated. It is stable for all $\rho < 1$, while it seems to satisfy the required delay properties; we provide some strong evidence for this, based on an approximate model and simulation. In Section V, we present an *indirect* routing scheme; that is, all packets are sent to one of a set of special nodes, which are in charge of performing the various broadcasts. This scheme is based on a construction of d disjoint spanning trees by Johnsson and Ho [7]. It will be proved to meet both of the performance objectives set above; in particular, the scheme is stable for all $\rho < (2/3)(1 - (1/2^d)) \approx 2/3$, while it satisfies $D \approx 3d + 1 + (9/4)\rho$ for small ρ. In evaluating the performance analysis of the various schemes, we also consider the steady-state average *queue-size* Q per node; our schemes appear to be efficient also with this respect. Study of the behavior of the measure Q aims at estimating the buffer capacity required for applying the schemes in practice. The indirect scheme of Section V will be seen to be *deadlock-free* when implemented with finite buffers; for the direct scheme, deadlock prevention can be achieved by using standard techniques.

Motivation for studying the problem introduced arises from the context of *asynchronous* computation. To illustrate this, let us consider the distributed execution of an iterative algorithm of the form $x := f(x)$, where $f : \Re^n \rightarrow \Re^n$ and n is the number of nodes; typically, the ith node knows the function f_i and updates x_i. Assume that the problem is dense, i.e., each entry of the function f depends explicitly on almost all entries of x; then, once x_i is updated, its new value must be broadcast to all other nodes, in order to be used in their subsequent calculations. If all nodes are perfectly synchronized, then all entries of the vector x are to be broadcast at the same time. This prototype communication task is called the *multinode broadcast*. The minimum possible time for performing this task (in the d-cube) is $\lceil (2^d - 1)/d \rceil$ and it can be attained by an algorithm by Bertsekas *et al.* [3]. However, there are cases where the new values of the x_i's are not all computed at the same time, due to variability in communication delays or to variability in the difficulty of evaluating $f_i(x)$ for different choices of i and x. In such a case, broadcasts by different processors are initiated at rather unpredictable times, best modeled by stochastic processes. For analytical tractability, we have assumed in this paper that packets are generated by the various nodes according to independent Poisson processes; we hope that our analysis will be suggestive of the results holding under more general packet-generating processes.

There exists considerable literature on algorithms for communication tasks in various interconnection networks. However, most of the related articles analyze "one shot" communications, where each task has to be performed only *once*, and no other packet transmissions are taking place at the same time. In particular, for the hypercube network, Bertsekas *et al.* [3] have devised optimal algorithms for a variety of communication tasks. Previously, Saad and Schultz [11], as well as Johnsson and Ho [7], had constructed optimal or nearly optimal algorithms for hypercubes, under somewhat different assumptions on packet transmissions. The interested reader may find more references in these three papers and in [2]. The communication tasks considered in the aforementioned papers as well as the respective algorithms do not employ any randomization in path selection. In his famous paper [15], Valiant has demonstrated how to use randomization in order to perform a deterministic task. In particular, in the context of the d-cube, he considered the permutation task and showed that it may be accomplished in time $\Theta(d)$ (namely, of the order of magnitude of the diameter d) with high probability, by using a randomized algorithm. In a later paper, Valiant and Brebner [14] modified this algorithm, thus simplifying considerably the analysis.

To the best of our knowledge, there is only one problem involving repetitive packet transmissions in the hypercube that has been studied before, the following: Every node generates packets according to some random process; each packet has a *single* destination, which is uniformly distributed among the nodes of the hypercube. Approximate and/or numerical studies of certain communication algorithms for this problem can be found in Abraham and Padmanabhan [1], Greenberg and Hajek [6], and Varvarigos [16]; related also is the work by Chang and Simon [5]. Rigorous results for a "greedy" routing scheme have been obtained by Stamoulis and Tsitsiklis in [13].

II. Background Material on the Hypercube Network

A. Definitions

We consider the *d-dimensional binary hypercube* (or d-cube); e.g., see [2]. This network consists of 2^d nodes, numbered from 0 to $2^d - 1$. Associated with each node z is a binary identity $(z_d, \cdots, z_1)$, which coincides with the binary representation of the number z. There exist arcs only between nodes whose binary identities differ in a single bit. That is, arc (z, y) exists if and only if $z_i = y_i$ for $i \neq m$ and $z_m \neq y_m$ (or equivalently $|z - y| = 2^{m-1}$) for some $m \in \{1, \cdots, d\}$. Note that (z, y) stands for a *unidirectional* arc pointing from z to y; of course, if arc (z, y) exists, so does arc (y, z). Clearly, the d-cube has $d2^d$ arcs.

The *Hamming distance* between two nodes z and y is defined as the number of bits in which their binary identities differ. Any path from z to y contains at least as many arcs as the Hamming distance between z and y. Moreover, there always exist paths that contain exactly that many arcs; these paths are characterized as *shortest*. It is easily seen that the *diameter* of the d-cube equals d.

For two nodes z and y, we denote by $z \oplus y$ the vector $(z_d \oplus y_d, \cdots, z_1 \oplus y_1)$, where $\oplus$ is the symbol for the XOR operation. The ith (from the right) entry of $z \oplus y$ equals 1, if and only if $z_i \neq y_i$. For $j \in \{1, \cdots, d\}$, we denote by e_j the node numbered 2^{j-1}; that is, all entries of the binary identity of e_j equal 0 except for the jth one (from the right), which equals 1. Nodes $e_1, \cdots, e_d$ are the only neighbors of node $(0, \cdots, 0)$. In general, each node z has exactly d neighbors, namely nodes $z \oplus e_1, \cdots, z \oplus e_d$. Thus, arc (z, y) exists if and only if $z \oplus y = e_m$ for some $m \in \{1, \cdots, d\}$. Such an arc is said to be of the mth *type*; the set of arcs of the mth type is called the mth *dimension*.

B. The Completely Unbalanced Spanning Tree

For two nodes z and y, suppose that the ith entry of $z \oplus y$ equals 1, i.e., $z_i \neq y_i$; in this case, a shortest path from z to y contains one arc of the ith type. In general, let

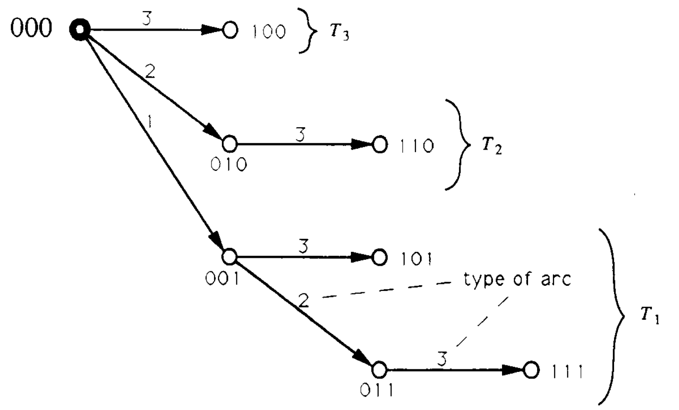


Fig. 1. A completely unbalanced spanning tree, for $d = 3$.

$i_1 < i_2 < \cdots < i_k$ be the only entries of $z \oplus y$ that equal 1. Then, any shortest path from z to y consists of k arcs, with one of them being of the i_1th type, one of them being of the i_2th type, etc. Thus, any packet originating at z will reach node y if it traverses exactly one arc of each of the aforementioned types. Such a packet will reach node y *regardless of the order* in which it crosses these hypercube dimensions.

The completely unbalanced spanning tree rooted at some node z is defined as the spanning tree with the following property: Every node y is reached from the root z through the unique shortest path in which the hypercube dimensions are crossed in *increasing order*. That is, if $i_1 < i_2 < \cdots < i_k$ are the dimensions to be crossed in any shortest path from z to y, then the tree under consideration contains that shortest path where the first arc belongs to the i_1th dimension, the second arc to the i_2th dimension, etc. One can easily see that this collection of paths constitutes a tree. A completely unbalanced spanning tree of the 3-cube is presented in Fig. 1; the root of that tree is node $(0,0,0)$. As already mentioned in Section II-A, the hypercube arcs are unidirectional; thus, all spanning trees considered throughout the paper are *directed*.

A completely unbalanced spanning tree T rooted at node z has d subtrees $T_1, \cdots, T_d$. Subtree T_i is hanging from node $z \oplus e_i$, and consists of all nodes y with the following property: $y_j = z_j$ for all $j < i$ and $y_i \neq z_i$ (see Fig. 1). Therefore, T_i contains 2^{d-i} nodes. Another interesting property of a completely unbalanced spanning tree is that it has 2^{d-1} leaves.

By considering different orders for crossing the hypercube dimensions, we can obtain other spanning trees, isomorphic to the tree T defined earlier. Henceforth, we call *all* of these trees completely unbalanced, as well.

Completely unbalanced spanning trees have been used extensively in algorithms for hypercube communications (see [11], [7], and [3]). Johnsson and Ho [7] use the terminology "spanning binomial tree."

III. Limits on the Achievable Performance

A. A Necessary Condition for Stability

The average total number of packets generated in the network during one slot equals $\lambda 2^d$. Broadcasting a packet (using any routing scheme) requires at least $2^d - 1$ transmissions. Therefore, during each slot, an average total demand for at least $\lambda 2^d(2^d - 1)$ packet transmissions is generated in the system. Since at most $d2^d$ packet transmissions may take place during each slot, it follows that the system can be stable only if $\lambda 2^d(2^d - 1) < d2^d$. Thus, we have the following necessary condition for stability:

$$\rho \overset{\text{def}}{=} \lambda \frac{2^d - 1}{d} < 1 \qquad (1)$$

where ρ, as defined above, will be called the *load factor* of the system. This terminology is appropriate, because when $\rho \approx 1$ all hypercube arcs are almost always busy, even if no redundant packet transmissions take place.

B. Lower Bounds on the Average Delay per Packet

In the present subsection, we establish a *universal* lower bound on he steady-state average delay D; that is, a bound that applies to any routing scheme. (Recall that D is defined as the stationary average of the time elapsing between the moment a packet is generated until the completion of its broadcast.) Since we are not sure whether this bound is tight, we also establish a *sharper* lower bound applying to a certain *class* of routing schemes. First, we present the universal bound.

Proposition 1: The average delay D per packet induced by any routing scheme satisfies

$$D \geq \max\left\{d, \frac{2^d - 1}{2^d}\mathcal{D}(d;\rho)\right\} = \Omega\left(d + \frac{\rho}{d(1-\rho)}\right)$$

where $\mathcal{D}(d;\rho)$ is the average delay for an $M/D/d$ queue with unit service time and arrival rate $d\rho$. (The notation $\Omega(A)$ means "of larger order of magnitude than A.") □

Outline of the Proof: We fix a node x; let $\mathcal{L}(x)$ be the set of arcs incoming at node x; i.e., $\mathcal{L}(x) \overset{\text{def}}{=} \{(x \oplus e_j, x) \mid j = 1, \cdots, d\}$. Under any legitimate routing scheme, all packets generated by nodes $y \neq x$ have to traverse at least one arc of $\mathcal{L}(x)$. It can be seen that the average delay until a packet reaches x will not increase if we change the original system by one in which the following conditions hold: Packets generated by x never traverse the arcs of $\mathcal{L}(x)$; each packet generated by a node $y \neq x$ is present at all neighbors of x *immediately* upon generation, and it *only* traverses the *first available* arc of $\mathcal{L}(x)$. In this ideal case, the d arcs of $\mathcal{L}(x)$ would operate as an $M/D/d$ queue with unit service time and arrival rate $\lambda(2^d - 1) = \rho d$ [recall the definition of ρ in (1)], and this provides a lower bound for the average delays in the actual system. Thus, we obtain

$$D \geq \frac{2^d - 1}{2^d}\mathcal{D}(d;\rho) \tag{2}$$

where the factor $(2^d - 1)/2^d$ accounts for the fact that packets generated by node x do not cross the arcs of $\mathcal{L}(x)$. Furthermore, it is known [4] that $\mathcal{D}(d;\rho)$ satisfies

$$\mathcal{D}(d;\rho) \geq \frac{1}{2} + \frac{\rho}{2d(1-\rho)}.$$

The proof is easily completed by using this inequality, (2), and the obvious fact $D \geq d$. Q.E.D.

As suggested by the proof of Proposition 1, a scheme attaining the universal lower bound on the delay D (if there exists such a scheme) would probably schedule transmissions by making use of *global* information. This claim is further supported by Proposition 2, which establishes a lower bound on D applying to the class of *oblivious* routing schemes. Under such a scheme, each packet decides (upon generation) which paths to follow, *independently* of all other packets in the network; also, each packet insists on traversing the selected paths, regardless of the contention encountered. Note that packets generated by the same node z follow the *same* rules. Clearly, the oblivious class comprises all schemes where each packet independently selects which tree to be broadcast along by using a randomized rule depending only on the identity of its origin node. The routing schemes discussed in Sections IV-A and IV-C are of this type.

We now present the lower bound on the delay induced by oblivious schemes.

Proposition 2: The average delay D per packet induced by any *oblivious* routing scheme satisfies

$$D \geq \max\left\{d, \frac{2^d - 1}{2^d}\left[1 + \frac{\rho}{2(1-\rho)}\right]\right\} = \Omega\left(d + \frac{\rho}{1-\rho}\right). \quad \square$$

Outline of the Proof: This proof is similar to that of Proposition 1. Again, we fix a node x and we consider the set $\mathcal{L}(x)$ of arcs incoming at x. Each packet generated at a node z will attempt to traverse some of the arcs of $\mathcal{L}(x)$; which arcs will be traversed is determined by a randomized rule depending only on node z. As in the proof of Proposition 1, we assume that each packet generated at a node $z \neq x$ crosses only one arc of $\mathcal{L}(x)$, while the ones generated at x cross no such arcs; also, a packet to traverse arc $(x \oplus e_j, x)$ is taken as present at node $x \oplus e_j$ immediately upon generation. Under these conditions, arc $(x \oplus e_j, x)$ is fed by a Poisson stream with rate r_j, because packets choose their respective paths independently; thus, each arc $(x \oplus e_j, x)$ operates as an $M/D/1$ queue with rate r_j, which implies

$$D \geq \sum_{j=1}^{d} \frac{r_j}{\lambda 2^d}\left[1 + \frac{r_j}{2(1-r_j)}\right] \tag{3}$$

where we have used the expression for the delay of an $M/D/1$ queue [8]. Notice now that $r[1 + (r/2(1-r))]$ is a convex function of r; it may also be seen that $\sum_{j=1}^{d} r_j = \lambda(2^d - 1) = d\rho$. Therefore, the right-hand quantity in (3) is minimized for $r_1 = \cdots = r_d = \lambda(2^d - 1)/d = \rho$. This implies that

$$D \geq \frac{2^d - 1}{2^d}\left[1 + \frac{\rho}{2(1-\rho)}\right];$$

the proof is easily completed by using this inequality and the obvious fact $D \geq d$. Q.E.D.

Proposition 2 implies that, for a rather broad class of schemes, the universal lower bound on the delay D is loose; notice that the factor $1/d$ is present only in the bound of Proposition 1. Suppose now that we allow packets generated by each node z to look at the routing decisions taken by packets previously generated by the *same* node. It is an interesting open question to investigate whether Proposition 2 still holds. We believe that this may possibly be true because each packet has a very limited knowledge of the routing decisions taken within the entire network. If this is indeed the case, then a scheme attaining the universal lower bound on D should either involve *centralized coordination* or some form of *adaptive* routing.

IV. Direct Routing Schemes

A. A Simple Approach to the Problem

The simplest approach to our problem is as follows: Each of the nodes broadcasts its packets along a certain spanning tree emanating from itself. Such a scheme can have rather *poor* performance. In fact, its performance depends heavily on the selection of the trees. For example, consider the case where each node routes its packets along the corresponding unbalanced spanning tree in which the hypercube dimensions are crossed in increasing order (see Section II-B). Every node z receives through its adjacent arc of the jth type all packets originating at all nodes x satisfying $x_m = z_m$ for $m > j$ and $x_j \neq z_j$. Thus, during each slot, there are generated an average of $\lambda 2^{j-1}$ packets that will eventually have to traverse arc $(z \oplus e_j, z)$. Therefore, the simple scheme under analysis may be stable only if $\lambda 2^{j-1} < 1$ for $j = 1, \cdots, d$, or equivalently

$$\rho < \frac{2}{d}\left(1 - \frac{1}{2^d}\right).$$

Hence, the maximum load factor that can be sustained by the above simple scheme *vanishes* as the dimensionality d of the hypercube increases. The reason for this undesirable behavior is that some of the arcs are shared by far more trees than the others.

A potential remedy to the above problem is to select 2^d trees (one rooted at each node) such that all arcs are shared by approximately the same number of trees. There does exist such a set of trees, namely the ones used in the optimal multinode broadcast algorithm of [3]. Since this algorithm lasts for $\lceil(2^d-1)/d\rceil$ slots, it follows that each arc is shared by at most $\lceil(2^d-1)/d\rceil$ of the trees; thus, broadcasting the packets along these trees will create no bottleneck in any of the arcs. This scheme is mainly of theoretical interest, because it is rather hard to implement; this is due to the fact that the trees used are complicated to describe. An alternative way of balancing traffic over the hypercube arcs is to use multiple trees per node and randomly distribute among them the packets to be broadcast. An efficient routing scheme that is based on this idea is presented in Section IV-C. This scheme is closely related to a periodic scheme presented in Section IV-B.

B. Performing Multinode Broadcasts Periodically

Another simple approach to our routing problem is to run *periodically* an efficient algorithm for multinode broadcast(s), such as the optimal algorithm of [3] and the nearly optimal ones of [7]. These algorithms utilize the hypercube arcs almost fully (during their respective running times); thus, the resulting periodic schemes would be stable even for $\rho \approx 1$ (see also below). On the other hand, any algorithm for multinode broadcast(s) takes at least $\lceil(2^d-1)/d\rceil$ slots. (To see that this quantity is a lower bound on the time required for the multinode broadcast, just notice that each node receives 2^d-1 packets during this task and it may only receive at most d of them during each slot.) Hence, a periodic routing scheme would induce an average delay of $\Omega(2^d/d)$ even for $\rho \approx 0$, which is too high.

We now present a periodic scheme that is based on the optimal algorithm by Saad and Schultz [11] for the d simultaneous multinode broadcasts. For this communication task, every node z has d packets to broadcast; each of these packets is routed along a completely unbalanced spanning tree rooted at node z (see Section II-B). In particular, the first packet is routed along that tree where the hypercube dimensions are crossed in the order $1, 2, \cdots, d$; the second packet is routed along that tree where the hypercube dimensions are crossed in the order $2, 3, \cdots, d, 1$, etc; the dth packet is routed along that tree where the hypercube dimensions are crossed in the order $d, 1, 2, \cdots, d-1$.

The above-described algorithm takes time 2^d-1 [11]. By pipelining successive instants thereof, each node may broadcast d packets every 2^d-1 time units; this applies if the tree to be assigned to each packet is determined either on the basis of availability, or *randomly* with all d permissible trees being equiprobable. Henceforth, we focus on this randomized scheme, which is seen to be stable if $(\lambda/d)(2^d-1) < 1$, or equivalently $\rho < 1$. This is the broadest possible stability region, due to the necessary stability condition in (1). Since each period of the scheme lasts for 2^d-1 slots, we have $D = \Omega(2^d)$ even for $\rho \approx 0$, which is too high.

The unsatisfactory delay performance of the periodic schemes is primarily due to the extensive occurrence of *idling*, caused by the periodic feature of these schemes; that is, it often occurs that arcs are idle while packets have to wait for the next period in order to cross them. As will be seen in the next subsection, avoidance of this idling phenomenon improves performance dramatically.

C. A Nonidling Direct Scheme—An Approximate Delay Analysis

In the present subsection, we consider the *nonidling* version of the periodic scheme described in Section IV-B (namely, of that using the algorithm of [11]). The scheme to be presented performs very satisfactorily. Unfortunately, exact analysis seems to be impossible; thus, we proceed by means of an approximate model, which we then validate using simulation.

The nonidling scheme to be discussed is as follows: Each packet chooses *randomly* one of the d completely unbalanced trees permissible (see Section IV-B) and traverses the corresponding arcs as soon as possible. This scheme belongs to the oblivious class defined in Section III-B.

By introducing an appropriate priority discipline for contention resolution, we can achieve the following: the *order* of the various packet transmissions is preserved when the periodic routing scheme is converted to a nonidling one. This implies that, under this priority discipline, each packet arrives at its destination *no later* than under the periodic version of the routing scheme. Since the periodic scheme is stable for all $\rho < 1$, it is seen that the nonidling scheme is also stable for all $\rho < 1$, which is the broadest possible stability region. Henceforth, we assume that contention is resolved on a FIFO basis, which is a more natural priority discipline.

Regarding the steady-state average delay D per packet, it is easily proved that $\lim_{\rho\to 0} D = d+(1/2)$; the term d accounts for the average of the maximum *propagation* time per packet (which equals the depth of the trees used); the term $1/2$ is the average *synchronization* time, namely the average of the time elapsing from the moment a packet is generated until the beginning of the next slot. (Recall that packets are assumed to be generated in continuous time, while transmissions may only start at the beginning of each slot.) Below, we derive an approximate expression for D, which will be seen to be in excellent agreement with simulation outcomes when ρ is not large.

In the analysis to follow, it is assumed that the various random processes involved are in steady-state. We fix a node x. Let $Y_k^{(i)}$ be the number of packets waiting to cross arc $(x, x\oplus e_i)$ at the beginning of the kth slot, including the packet to be transmitted. Moreover, let B_k be the number of packets generated by node x during the kth slot; since arc $(x, x\oplus e_i)$ belongs to all of the d trees that may possibly be selected by a packet generated by x, all B_k newly generated packets will join the queue for this arc. Also, let $P_k^{(m,i)}$ be the number of packets received by node x (during the kth slot) through arc $(x\oplus e_m, x)$ and wishing to traverse arc $(x, x\oplus e_i)$. Notice that $P_k^{(m,i)} \in \{0,1\}$; that is, all of these random variables are of the Bernoulli type. Clearly, we have

$$Y_{k+1}^{(i)} = [Y_k^{(i)} - 1]^+ + B_k + \sum_{m=1}^{d} P_k^{(m,i)}, \qquad \text{for } k = 1, \cdots, \tag{4}$$

where $[\alpha]^+$ stands for $\max\{\alpha, 0\}$.

By symmetry, the traffic is split *evenly* (on the average) among the various arcs; thus, each of them is busy at a particular slot with probability ρ. Using also the definition of $P_k^{(m,i)}$, we obtain

$$E[P_k^{(m,i)}] = \rho g_{m,i} \tag{5}$$

where $g_{m,i}$ is the probability that a packet has to cross arc $(x, x \oplus e_i)$ given that it has crossed arc $(x \oplus e_m, x)$. Taking into account the d possible orders of crossing the hypercube dimensions, it may be proved [13] that

$$g_{m,i} = \frac{2^{[(m-i) \bmod d]_{-1}}}{2^d - 1}, \qquad \forall (m,i) \in \{1, \cdots, d\}^2; \tag{6}$$

notice that, by symmetry among the various nodes, the parameters $g_{m,i}$ are *independent* of x. Also, there holds $g_{i,i} = 0$, which is due to the fact that no packet having crossed arc $(x \oplus e_i, x)$ attempts to cross arc $(x, x \oplus e_i)$.

Next we introduce two *approximating assumptions*. These assumptions will only be in effect in the present subsection.

Assumption A: For any pair (m, i), the random variables $(P_k^{(m,i)})_{k=1,\cdots}$ are taken to be *independent* and identically distributed.

Assumption B: The processes $(P_k^{(1,i)})_{k=1,\cdots}, \cdots, (P_k^{(d,i)})_{k=1,\cdots}$ are taken mutually *independent*.

These assumptions are of similar spirit as those used in [1] and [6] for a considerably different routing problem. In the analysis to follow, all equalities to be derived are approximate (unless otherwise specified), since they are based on the two assumptions above.

We define the random process $(A_k^{(i)})_{k=1,\cdots}$ as follows:

$$A_k^{(i)} \overset{\text{def}}{=} B_k + \sum_{m=1}^{d} P_k^{(m,i)}. \tag{7}$$

$(B_k)_{k=1,\cdots}$ is (actually) a renewal process, and is independent of $(P_k^{(1,i)})_{k=1,\cdots}, \cdots, (P_k^{(d,i)})_{k=1,\cdots}$. Thus, under our approximation, $(A_k^{(i)})_{k=1,\cdots}$ is taken as a renewal process that assumes the distribution of a random variable $A^{(i)}$. Since B_k is a Poisson random variable with mean λ, it follows from (5) and (7) that

$$E[A^{(i)}] = \lambda + \sum_{m=1}^{d} \rho g_{m,i},$$

and

$$\begin{aligned}\text{var}[A^{(i)}] &= \lambda + \sum_{m=1}^{d} \rho g_{m,i}(1 - \rho g_{m,i}) \\ &= \lambda + \rho \sum_{m=1}^{d} g_{m,i} - \rho^2 \sum_{m=1}^{d} g_{m,i}^2\end{aligned}$$

where $\text{var}[A^{(i)}]$ denotes the variance of $A^{(i)}$. Using (6), it follows (after some algebra) that

$$E[A^{(i)}] = \rho \quad \text{and}$$

$$\text{var}[A^{(i)}] = \rho - \rho^2 \frac{1}{(2^d - 1)^2}\left[d + \frac{1}{3}(4^d - 1) - 2(2^d - 1)\right]. \tag{8}$$

The leftmost result is actually true and could have been taken for granted; indeed, it may be seen from (9) that $E[A^{(i)}]$ is the average input traffic rate for arc $(x, x \oplus e_i)$, which equals ρ.

Furthermore, combining (7) with (4), we obtain

$$Y_{k+1}^{(i)} = [Y_k^{(i)} - 1]^+ + A_k^{(i)}, \qquad \text{for } k = 1, \cdots. \tag{9}$$

Since the arrival process $(A_k^{(i)})_{k=1,\cdots}$ was taken as a renewal process, it follows from (9) that $(Y_k^{(i)})_{k=1,\cdots}$ may be approximated by the process of the number of customers in a discrete-time $G/D/1$ queue with unit service time. Let $D^{(i)}$ be the average delay associated with this queue, including the transmission time over that arc. The expression for $D^{(i)}$ is well-known (e.g., see [9]); in particular, we have

$$\begin{aligned}D^{(i)} &= 1 + \frac{E[A^{(i)}(A^{(i)} - 1)]}{2E[A^{(i)}](1 - E[A^{(i)}])} \\ &= 1 + \frac{\text{var}[A^{(i)}] + (E[A^{(i)}])^2 - E[A^{(i)}]}{2E[A^{(i)}](1 - E[A^{(i)}])} \\ &= \frac{1}{2} + \frac{\text{var}[A^{(i)}]}{2E[A^{(i)}](1 - E[A^{(i)}])}.\end{aligned}$$

Combining this with (8), we obtain

$$\begin{aligned}D^{(i)} &= \frac{1}{2} + \frac{1}{2(1-\rho)} \\ &\quad \times \left(1 - \frac{\rho}{(2^d - 1)^2}\left[d + \frac{1}{3}(4^d - 1) - 2(2^d - 1)\right]\right), \\ &\qquad \text{for } i = 1, \cdots, d.\end{aligned} \tag{10}$$

Due to complete symmetry, the above expression is independent of i and of the fixed node x.

So far, we have derived an approximate expression for the average delay suffered by a packet while waiting to cross an arc of the ith type. The overall delay of a packet will be approximated with the delay suffered in the *longest* path; this path consists of d arcs, one from each of the d hypercube dimensions. Thus, using (10) and taking also the average synchronization time into account, we obtain the following approximate formula for the average delay per packet:

$$\begin{aligned}D &\approx \frac{d}{2} + \frac{d}{2(1-\rho)} \\ &\quad \times \left(1 - \frac{\rho}{(2^d - 1)^2}\left[d + \frac{1}{3}(4^d - 1) - 2(2^d - 1)\right]\right) + \frac{1}{2}.\end{aligned} \tag{11}$$

For d not being very small (say $d \geq 10$), the above formula may be simplified to the following:

$$D \approx \frac{d}{2} + \frac{d}{2(1-\rho)}(1 - \frac{\rho}{3}) + \frac{1}{2} = d + \frac{1}{2} + d\frac{\rho}{3(1-\rho)}.$$

Furthermore, for small ρ, we have

$$D \approx d + \frac{1}{2} + \frac{d}{3}\rho. \tag{12}$$

As will be seen below, the approximate formula (11) is in excellent agreement with the simulation outcomes, for $\rho \leq 0.3$. This together with (12) supports the conjecture that, for small

TABLE I

$d = 8$			
ρ	Simulation	Approximation	Relative Error
0.050	8.620	8.641	0.24%
0.100	8.763	8.799	0.41%
0.150	8.950	8.974	0.27%
0.200	9.165	9.172	0.08%
0.250	9.480	9.396	−0.89%
0.300	9.808	9.652	−1.59%
0.350	10.246	9.947	−2.92%
0.400	10.733	10.291	−4.12%
0.450	11.295	10.699	−5.28%
0.500	12.201	11.187	−8.31%

ρ, there holds $D \approx d + (1/2) + Kd\rho$ with constant K; thus, it appears that the routing scheme using d trees per node meets the performance objectives set in Section I.

Next, we investigate the accuracy of (11). In Table I, we compare the simulation outcomes for the eight-dimensional hypercube with the estimate given by (11) for $d = 8$; each of the simulation outcomes was obtained over a period of 5000 slots. Clearly, there is *excellent agreement* between the experimental results and the corresponding approximate estimate for D, for values of ρ ranging from 0.05 to 0.3; for all such values of ρ, the magnitude of the relative error is less than 2%. In fact, the agreement for values of $\rho \leq 0.25$ is striking; in all simulations performed, the relative error did not exceed 1% for $\rho \leq 0.25$. Unfortunately, the accuracy of the approximate formula (11) deteriorates gradually for $\rho > 0.3$ and is quite bad for for $\rho > 0.5$. In Table II, we investigate the accuracy of (11) for different values of the hypercube dimension d; again, excellent agreement is observed for $d = 5, \cdots, 10$ under moderately light traffic (namely, for $\rho = 0.10$, $\rho = 0.15$, and $\rho = 0.20$). The experimental results of Table II were obtained over periods of 1000 slots.

TABLE II

$\rho = 0.10$			
d	Simulation	Approximation	Relative Error
5	5.659	5.696	0.65%
6	6.705	6.729	0.31%
7	7.729	7.763	0.44%
8	8.725	8.799	0.85%
9	9.806	9.835	0.30%
10	10.819	10.871	0.48%
$\rho = 0.15$			
d	Simulation	Approximation	Relative Error
5	5.800	5.811	0.19%
6	6.844	6.863	0.28%
7	7.881	7.918	0.47%
8	8.933	8.974	0.46%
9	10.043	10.031	−0.12%
10	11.091	11.089	−0.02%
$\rho = 0.20$			
d	Simulation	Approximation	Relative Error
5	5.894	5.940	0.78%
6	7.001	7.002	0.01%
7	8.102	8.092	−0.12%
8	9.177	9.172	−0.05%
9	10.227	10.253	0.25%
10	11.379	11.335	−0.39%

V. An Indirect Routing Scheme Based on d Disjoint Trees

Consider the following simple routing scheme: All packets are sent to a specific node, which broadcasts them along a spanning tree emanating from itself. By pipelining successive broadcasts, it is seen that this scheme can route one broadcast per slot. Thus, stability can be maintained only if $\lambda 2^d < 1$, or equivalently $\rho < (1/d)(1 - (1/2^d))$. Therefore, the maximum attainable value for the load factor is $\Theta(1/d)$. (The notation $\Theta(A)$ means "of the same order of magnitude as A.") The reason for this poor performance is that only a fraction $1/d$ of the available hypercube arcs are used for broadcasting the packets.

The above discussion leads to the following idea: Suppose that we could broadcast packets along d *disjoint* spanning trees $T^{(1)}, \cdots, T^{(d)}$, with each tree receiving the same amount of traffic; then, the maximum load factor might possibly be $\Theta(1)$, which coincides with one of our performance objectives. A routing scheme of this spirit is presented in this section; the scheme uses the set of d disjoint spanning trees introduced by Johnsson and Ho [7] (see Section V-A). Under this scheme, each packet is sent to the root of $T^{(j)}$, for some j, and then it is broadcast along this spanning tree. Since packets are not broadcast directly by their respective origins, the scheme to be presented is characterized as *indirect.*

It will be established *rigorously* that the routing scheme analyzed in this section is stable for all $\rho \leq (2/3)(1 - (1/2^d)) \approx 2/3$, while it satisfies $D \approx 3d + 1 + (9/4)\rho$ for small ρ and our performance objectives. The indirect scheme has one more interesting property, namely it is *deadlock-free* when implemented with finite buffers (see Section V-E). In Section V-G, we compare the indirect scheme with the direct scheme analyzed approximately in Section IV-C.

A. The d Disjoint Spanning Trees

Johnsson and Ho [7] have constructed an imbedding of d disjoint spanning trees in the d-cube; they call them "d Edge-Disjoint Spanning Binomial Trees" (dESBT). This imbedding consists of d completely unbalanced trees $T^{(1)}, \cdots, T^{(d)}$. Tree $T^{(j)}$ is rooted at node e_j. The order of crossing the hypercube dimensions in the paths of $T^{(j)}$ is as follows:

$$(j \bmod d) + 1, [(j + 1) \bmod d] + 1, \\ \cdots, [(j + d - 1) \bmod d] + 1.$$

Tree $T^{(j)}$ has d subtrees denoted as $T_1^{(j)}, \cdots, T_d^{(j)}$, with $T_i^{(j)}$ having 2^{d-i} nodes.

B. The Rules of the Routing Scheme

In what follows we present the set of rules for routing the packets. Rules A, B, and C are the main ones, and require that a packet is sent to the root of one of the trees $T^{(1)}, \cdots, T^{(d)}$, from where it is actually broadcast; transmissions towards the roots may only be performed every three slots, while the rest of the time is dedicated to transmissions away from the roots. Rules D and E are only introduced for analytical tractability.

Rule A: Each packet generated at some node selects the tree along which it will be broadcast. Selection is *randomized*, with the only permissible trees being $T^{(1)}, \cdots, T^{(d)}$; each of them is assigned an *a priori* probability $1/d$. Different packets make their selections independently.

Rule B: Consider a packet, originating at some node y, that has chosen tree $T^{(j)}$. This packet must be *sent to the root* e_j of this tree, which will actually perform the broadcast; the path to be followed is the reverse of the path from e_j to y that is contained in $T^{(j)}$. That is, this packet will traverse the reverse of those arcs of $T^{(j)}$ that lead from e_j to y. Note that packets generated by the root nodes $e_1, \cdots, e_d$ also follow Rules A and B, as well as the ones presented below. Thus, it may occur that a packet generated by node e_m is sent to some other root e_j, in order to be broadcast along $T^{(j)}$.

Rule C: Consider an arc $(z, z \oplus e_i)$ belonging to $T^{(j)}$, while its reverse arc $(z \oplus e_i, z)$ belongs to some other of the d disjoint trees, say to $T^{(m)}$. Because of Rule B, it is possible that some packet to be broadcast along $T^{(m)}$ has to traverse arc $(z, z \oplus e_i)$ while heading towards the root node e_m; we impose the restriction that such an arc traversal is permissible only every *three* slots. In order to make this rule more specific, we define $\mathcal{C}_0 \overset{\text{def}}{=} \{t \geq 0 : t \bmod 3 = 0\}$, $\mathcal{C}_1 \overset{\text{def}}{=} \{t \geq 0 : t \bmod 3 = 1\}$ and $\mathcal{C}_2 \overset{\text{def}}{=} \{t \geq 0 : t \bmod 3 = 2\}$. Arc $(z, z \oplus e_i)$ may be traversed by packets that have selected tree $T^{(j)}$ (where the arc belongs) only during time slots in $\mathcal{C}_1 \cup \mathcal{C}_2$. Moreover, this arc may be traversed by packets that have selected tree $T^{(m)}$ (where its reverse arc $(z \oplus e_i, z)$ belongs) only during time slots in $\mathcal{C}_0$. Thus, every three slots, each of the d trees is *reversed*, and all of its arcs point towards its root. Slots in the set $\mathcal{C}_0$ are used for sending packets to the respective root nodes, while slots in the set $\mathcal{C}_1 \cup \mathcal{C}_2$ are used for the broadcasts. Note that the d arcs $(0, e_i)_{i=1,\cdots,d}$ do not belong to any of the d disjoint trees (see [7]); these arcs are only used during slots in $\mathcal{C}_0$.

Rule D: Consider a packet generated at some node y that has selected tree $T^{(j)}$. If y is *not a leaf* of $T^{(j)}$, then before the packet considers to cross the first arc of its path to e_j it has to traverse one *virtual* arc located at node y (see Fig. 2 for the case $d = 3$, and compare it with Fig. 1). Such arcs may be traversed only during lots in $\mathcal{C}_0$. It is assumed that packets to be routed along different trees have to cross different virtual arcs, even if they have been generated at the same node. Obviously, virtual arcs can be realized by appropriately delaying packets in their respective origins.

Rule E: Every root node e_j has a pair of buffers B_1 and B_2. Consider a packet that has selected tree $T^{(j)}$; let node y be he origin of this packet. If y belongs to the largest subtree $T_1^{(j)}$, then it will be placed in buffer B_1; if y belongs to any subtree other than $T_1^{(j)}$ (or if $y = e_j$), then it will be placed in buffer B_2; see Fig. 2. During slots $t+1, t+2$, with $t \in \mathcal{C}_0$, root node e_j broadcasts one packet from each of buffers B_1 and B_2. Which of the two packets will be broadcast first is determined by tossing a fair coin. In the case where one of the two buffers is empty, *only one* packet is broadcast; again, the slot when the broadcast will start is determined by tossing a fair coin. Of course, if both B_1 and B_2 are empty, then no further action is taken.

Prior to evaluating the performance of the proposed scheme, we present some results to be used in the analysis to follow.

C. Auxiliary Results

First, we consider a tree T of n paths of the same length, with all paths having their *final* arc in common. Packets arrive at the starting nodes $s_1, \cdots, s_n$ of the paths and exit only at the common end f [see Fig. 3(a)]; packets are stored in the intermediate nodes of the tree (if necessary) and are forwarded as soon as possible. All packet transmissions start at the beginning of slots and each of them lasts for one slot. We claim that if we *collapse* all paths into one (with the same length as before) and we combine the arrival processes, then the departure process at node f will remain the same [see Fig. 3(b)]. This is proved in Lemma 3. Note that this result is basically a consequence of synchronization and pipelining.

In the context of the tree T of paths, we denote by $A_i(t)$ the number of packets that arrive at node s_i just before the end of slot t. Moreover, we denote by $F(t)$ the number of packets that depart from node f at slot t; clearly, $F(t)$ equals either 0 or 1. In the context of the single path P, we define $\tilde{A}(t)$ and $\tilde{F}(t)$ in a similar way. All the above processes are defined for $t = 0, \cdots$; both systems start operating at time $t = 0$. The result to be presented is established in the Appendix.

Lemma 3: If $\tilde{A}(t) = \sum_{i=1}^{n} A_i(t)$ for $t = 0, \cdots$, then $\tilde{F}(t) = F(t)$ for $t = 0, \cdots$. □

Lemma 3 holds even if packets arrive according to some continuous time process, provided that packet transmissions start at the beginning of slots. Also, the lemma still applies if packet transmissions can only start at the beginning of slots numbered $0, \Delta, \cdots$, and each transmission lasts for Δ slots. Even though Lemma 3 does *not* hold if some of the paths have different length than the others, we are still able to prove an interesting result applying to such a case; this result is presented next.

We now consider a tree $\tilde{T}$ consisting of n paths with possibly *different* lengths $l_1, \cdots, l_n$; see Fig. 4. Again, new packets enter the tree at the leaves and exit at the common end f. At each of the starting nodes $s_1, \cdots, s_n$, new packets are now assumed to arrive according to a *Poisson* processes with rate λ; arrivals at different starting nodes are mutually independent. Transmissions may only start at the beginning of slots numbered $0, \Delta, \cdots$, and each of them lasts for Δ slots. Let $\tilde{D}$ denote the steady-state average delay per packet induced in the tree $\tilde{T}$. Below, we present the stability condition for $\tilde{T}$ and the expression for $\tilde{D}$; the proof of Lemma 4 is summarized in the Appendix.

Lemma 4: The tree $\tilde{T}$ of paths is a stable queueing system if and only if $\lambda n \Delta < 1$. Moreover, in the stable case, there

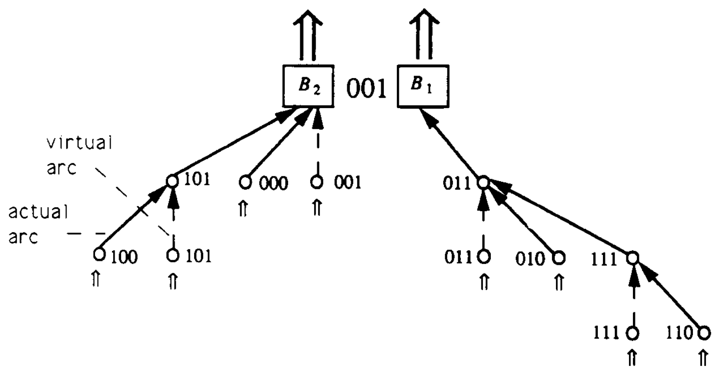


Fig. 2. Introducing the virtual arcs and buffers B_1 and B_2 in $T^{(1)}$, for $d = 3$.

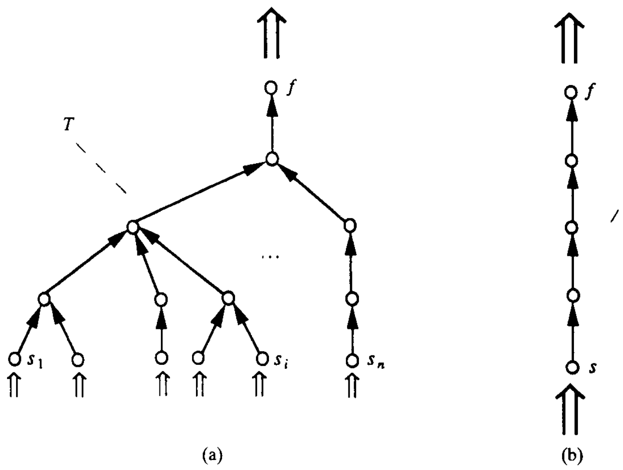


Fig. 3. (a) The tree T of paths. (b) The single path P.

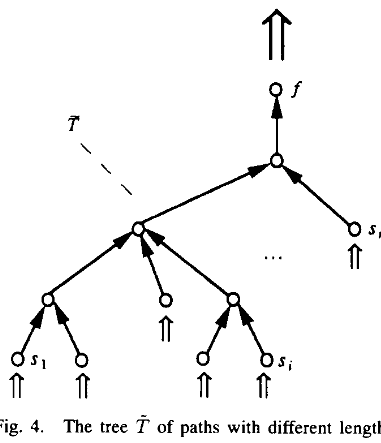


Fig. 4. The tree $\tilde{T}$ of paths with different lengths.

holds

$$\tilde{D} = \Delta\left[\frac{3}{2} + \frac{\lambda n \Delta}{2(1 - \lambda n \Delta)}\right] + \frac{\Delta}{n}\sum_{i=1}^{n} l_i - \Delta. \qquad \square$$

Lemma 4 will be seen to be very useful in the subsequent analysis; it is also applicable to other routing problems involving trees.

D. Performance Evaluation of the Scheme

First, we notice that packets routed along different trees do *not interfere* at all, in the following sense: Two such packets will never attempt to traverse the same arc at the same time. Indeed, if both packets are under broadcast, then they cannot collide, because they have to traverse disjoint sets of arcs. (Recall that the d trees used for routing are disjoint.) If both packets are heading towards the respective root nodes, again their paths are disjoint. (Because the d disjoint trees remain disjoint after reversing their arcs.) Moreover, even if these packets have been generated at the same node and are still traversing the corresponding virtual arcs, they cannot collide, as guaranteed by Rule D. Finally, if one of the packets is under broadcast and the other is heading towards the corresponding root, then they may not both take a step at the same time, because of Rule C.

Since packets that are routed along different trees do not interfere, we may analyze the performance of the scheme *separately* for each tree. In fact, we have to consider only one of them, because the d trees are isomorphic and are treated by the routing rules in a symmetric way. Thus, for the rest of the analysis, we focus on the queues formed by the packets routed along $T^{(1)}$. First, we derive the condition for stability of the scheme:

Proposition 5: The routing scheme introduced in Section V-B is stable if and only if

$$\rho < \frac{2}{3}\left(1 - \frac{1}{2^d}\right). \tag{13}$$

Proof: Clearly, when a packet starts being broadcast, it does not suffer any more delay due to other packets, because successive broadcasts are pipelined. Thus, the traffic load that can be accommodated by the scheme is determined exclusively by the processes of packets departing from buffers B_1 and B_2 of e_1. Due to Rule E, these two porcesses are independent; hence the stability region coincides with the intersection of the regions obtained by considering each of buffers B_1 and B_2.

First, we consider the set of all paths leading to buffer B_1. All of them have their final arc in common [namely, arc $(e_2 \oplus e_1, e_1)$]; see Fig. 2. Moreover, because of Rule C, all transmissions in these paths take place every 3 slots; finally, due to the *virtual* arcs added in the nonleaf nodes of $T^{(1)}$ (see Rule D), all new packets are now generated at "leaves," each fed by a Poisson process with rate λ/d. (Notice that, due to Rule D, each nonleaf node of $T^{(1)}$ is converted to a "leaf" hanging from a virtual arc.) These properties of the set of paths leading to buffer B_1 allow us to apply Lemma 4 with $\Delta = 3$, $n = 2^{d-1}$ and with λ/d instead of λ. (Recall that B_1 is in charge of all packets originating in the largest subtree $T_1^{(1)}$ of $T^{(1)}$, which contains 2^{d-1} nodes.) Thus, the set of these paths is stable if and only if $3(\lambda/d)2^{d-1} < 1$; by the definition of ρ in (1), this condition is equivalent to (13).

Reasoning as above, it may be seen that the set of all paths leading to buffer B_2 of e_1 is stable if and only if (13) applies, which completes the proof. The only subtle point in the new argument is that the paths leading to B_2 do *not* share their final arc (see Fig. 2). Nevertheless, Lemma 4 is still applicable, because packets depart from B_2 *one-by-one* and every three slots (due to Rule E); thus, the process of packets broadcast through buffer B_2 is the *same* as if the paths leading to B_2 had their final arc in common. Q.E.D.

Next, we derive the expression for the average delay D per packet under our indirect routing scheme.

Proposition 6: The average delay D for the routing scheme introduced in Section V-B is given as follows:

$$D = 3d + 1 + \frac{3\rho}{2[\frac{2}{3}(1 - \frac{1}{2^d}) - \rho]}. \qquad \square$$

Proof: The delay D may be expressed as the sum of two terms R and V, where R is the average time for a packet to reach e_1 and exit from the corresponding buffer, and V is the average time from the moment a packet exits this buffer until its broadcast is completed. As already argued in the proof of Proposition 5, the set of paths leading to buffers B_1 (resp., B_2) of e_1 satisfies the conditions of Lemma 4 with $\Delta = 3$, $n = 2^{d-1}$ and with λ/d instead of λ; thus, the average delay R_1 (resp., R_2) associated with the set of paths leading to B_1 (resp., B_2) is given as follows:

$$R_1 = 3\left[\frac{3}{2} + \frac{3\frac{\lambda}{d}2^{d-1}}{2(1 - 3\frac{\lambda}{d}2^{d-1})}\right] + \frac{3}{2^{d-1}}\sum_{i=1}^{2^d-1} l_i^{(1)} - 3, \tag{14}$$

and

$$R_2 = 3\left[\frac{3}{2} + \frac{3\frac{\lambda}{d}2^{d-1}}{2(1 - 3\frac{\lambda}{d}2^{d-1})}\right] + \frac{3}{2^{d-1}}\sum_{i=1}^{2^d-1} l_i^{(2)} - 3 \tag{15}$$

where $l_i^{(j)}$ is the length of the ith path leading to buffer B_j (for $j = 1, 2$), with paths numbered arbitrarily. Since each of buffers B_1 and B_2 is in charge of broadcasting the packets originating from 2^{d-1} hypercube nodes, an arbitrary packet routed along $T^{(1)}$ is equally likely to be traveling in either of the two sets of paths. Thus, using (14) and (15), we have

$$\begin{aligned} R = \frac{1}{2}(R_1 + R_2) &= 3\left[\frac{3}{2} + \frac{3\frac{\lambda}{d}2^{d-1}}{2(1 - 3\frac{\lambda}{d}2^{d-1})}\right] \\ &\quad + \frac{3}{2^d}\left[\sum_{i=1}^{2^d-1} l_i^{(1)} + \sum_{i=1}^{2^d-1} l_i^{(2)}\right] - 3. \end{aligned} \tag{16}$$

Clearly, $\sum_{i=1}^{2^d-1} l_i^{(1)} + \sum_{i=1}^{2^d-1} l_i^{(2)}$ equals the sum of the Hamming distances $H(y, e_1)$ of all nodes y from root e_1 plus a contribution of 1 per *nonleaf* node in $T^{(1)}$; the latter contribution accounts for the virtual arcs added by Rule D. We have $\sum_{y=0}^{2^d-1} H(y, e_1) = \sum_{k=0}^{d} k\binom{d}{k} = d2^{d-1}$, because there are $\binom{d}{k}$ nodes at Hamming distance k from each fixed node z; also note that $T^{(1)}$ has 2^{d-1} nonleaf nodes, because it is a completely unbalanced spanning tree (see Section II-B). Therefore, it follows from (16) that

$$\begin{aligned} R &= 3\left[\frac{3}{2} + \frac{3\frac{\lambda}{d}2^{d-1}}{2(1 - 3\frac{\lambda}{d}2^{d-1})}\right] + \frac{3}{2^d}(d2^{d-1} + 2^{d-1}) - 3 \\ &= \frac{3}{2}d + 3 + \frac{3\rho}{2[\frac{2}{3}(1 - \frac{1}{2^d}) - \rho]} \end{aligned} \tag{17}$$

where we have also used the definition of ρ in (1).

Once a packet has arrived at e_1 and has entered the corresponding buffer, the time required for its broadcast to be completed depends on the slot when the broadcast starts. Thus, if the broadcast starts at a slot in C_1, then it is completed in time $\lceil d/2 \rceil + d - 1$. If the broadcast starts at a slot in C_2, then it is completed in time $\lfloor d/2 \rfloor + d$. Since both these events occur with probability $1/2$ (because of Rule E), it follows that

$$V = \frac{1}{2}\left(\left\lceil \frac{d}{2} \right\rceil + d - 1 + \left\lfloor \frac{d}{2} \right\rfloor + d\right) = \frac{3}{2}d - \frac{1}{2}. \tag{18}$$

Notice now that

$$D = R + V - \frac{3}{2}. \tag{19}$$

The correction term $-3/2$ is due to the following fact: In estimating R, each packet is considered as departing from B_1

(or from B_2) at some time $(t + 3) \in \mathcal{C}_0$, while the packet actually starts being broadcast either at time $t + 1$ or at $t + 2$; thus, an average of $3/2$ slots per packet is counted in *both* R and V. The proof is completed by using (17), (18), and (19). Q.E.D.

It follows from Proposition 6 that, in the case of light traffic, we have $D \approx 3d + 1 + (9/4)\rho$. Also, by Proposition 5, our indirect scheme is stable for all $\rho < \rho^* \approx 2/3$. Hence, the scheme meets the performance objectives set in Section I.

E. Buffer Sizes and Deadlock Prevention

Next, we study the behavior of the steady-state average queue-size Q per node, defined as $1/2^d$ of the average total number of packets stored within the entire network (per slot) in steady-state. Note that, under the present scheme, the queue-size statistics may vary for different nodes; thus, the above definition of Q provides us with an "overall" estimate of the various queue-sizes. In evaluating Q, a packet is considered as stored at some node x only if x has yet to *forward* the packet towards one of its neighbors; packets to start transmission(s) immediately are also included. Note that a packet to be forwarded to several neighbors of x is assumed to occupy *one* unit of buffer capacity, for obvious reasons.

Next, we present the expression for the average queue-size Q per node; its derivation is similar to that of the delay D, and can be found in [12].

Proposition 7: There holds

$$Q = \frac{3d}{4}\rho\frac{2^d-2}{2^d-1} + 3\rho\frac{d}{2^d-1} \times \left(\frac{3}{2}d + \frac{17}{6} + \frac{3\rho}{2[\frac{2}{3}(1-\frac{1}{2^d})-\rho]}\right). \qquad \square$$

Proposition 7 implies that, for small ρ, we have $Q \approx (3d/4)\rho(2^d-2)/(2^d-1) + 3\rho(d/(2^d-1)((3/2)d + (17/6))$; when d is not very small, this simplifies to $Q \approx (3d/4)\rho$.

Even though our analysis was based on the assumption of infinite buffer capacity, in practical applications all nodes have *finite* buffer capacity. Study of the average-queue size Q per node aims at estimating the buffer capacity required for applying the indirect scheme in practice. As will be seen in Section V-G, our indirect routing scheme is very efficient with this respect. Another important feature of this scheme is that it is *deadlock-free* when implemented with finite buffers; this fact is established below.

Clearly, when some node x has finite buffer capacity, then a buffer overflow may occur; there are basically two approaches for dealing with such an overflow:

1) Packets waiting to cross an arc incoming to x are *blocked*, until there is empty space available in the buffer of x.
2) Packets continue to arrive at x, and they are *dropped* if there is no empty space in the buffer.

Under the second approach, it is possible that packets are *partly* broadcast; that is, it may occur that a packet is not received by all nodes. On the other hand, the first approach runs the risk of a *deadlock*. Such a deadlock *never* occurs under the indirect scheme analyzed. Indeed, recall that packets routed along different trees do not interfere; also, notice that all simultaneous transmissions along the same tree are pointing at the *same* "direction." (That is, at each time, all packets are heading either towards or away from the corresponding root.) Thus, there never arises a group of packets blocking one another in a "cyclic" pattern. Therefore, even a buffer capacity of two units per node and per *tree* $T^{(j)}$ is sufficient to guarantee that no deadlock will ever occur; one unit of buffer capacity is dedicated to packets heading towards root e_j, while the other unit is dedicated to packets already undergoing broadcast along $T^{(j)}$. It follows from the above discussion that when implementing the indirect scheme with buffer capacity of $\Theta(d)$ per node, all packets *admitted* in the network are guaranteed to be broadcast in finite time. In fact, the same statement applies even in the presence of other packet transmissions (not necessarily broadcasts), provided that each of these packets also is routed along one of the d disjoint trees and conforms to Rules B and C.

F. Discussion on the Scheme

Under our indirect scheme, one third of the time (namely, all slots in $\mathcal{C}_0$) is dedicated to transmissions towards the roots $e_1, \cdots, e_d$, due to Rule C. However, these transmissions constitute a very small portion of those perfomed overall. Indeed, consider a packet that is generated at some node y, which is at Hamming distance k from root e_j. If this packet selects to be routed along $T^{(j)}$, then it will undertake $k + 2^d - 1$ transmissions; k of them are required for the packet to reach e_j and the rest $2^d - 1$ are undertaken during the broadcast performed by the root node e_j. In fact, k of the transmissions are *redundant*, because they bring the packet to nodes that have already received it.

The above discussion seems to suggest that Rule B and especially Rule C result in a considerable decrease of the maximum load factor that can be accommodated by the routing scheme. However, this claim is *not* correct, as proved below:

Proposition 8: Any routing scheme that conforms to Rule A is stable only if

$$\rho < \frac{2}{3}\left(1 - \frac{1/3}{2^d - 2/3}\right). \tag{20}$$

Proof: We fix some $j \in \{1, \cdots, d-1\}$. We consider node $e_j \oplus e_{j+1}$, which is the neighbor of e_j through the $(j+1)$st dimension; similarly, node $e_j \oplus e_{j+1}$ is the neighbor of e_{j+1} through the jth dimension.

In the context of $T^{(j)}$, the largest subtree $T_1^{(j)}$ is hanging from node $e_j \oplus e_{j+1}$. (To see this, just recall the order of crossing hypercube dimensions in the paths of $T^{(j)}$; see also Sections V-A and II-B.) Clearly, all packets originating at any node of $T_1^{(j)}$ and routed along $T^{(j)}$ have to traverse arc $(e_j \oplus e_{j+1}, e_j)$, in order to be received by e_j. Since subtree $T_1^{(j)}$ contains 2^{d-1} nodes, these packets represent an average total demand for $(\lambda/d)2^{d-1}$ transmissions over arc $(e_j \oplus e_{j+1}, e_j)$ per slot.

In the context of $T^{(j+1)}$, node e_j is a leaf and its parent is node $e_j \oplus e_{j+1}$. Thus, all packets generated at any node other that e_j and routed along $T^{(j+1)}$ have to traverse arc $(e_j \oplus e_{j+1}, e_j)$, in order to be received by e_j. These packets represent an average total demand for $(\lambda/d)(2^d - 1)$ transmissions over arc $(e_j \oplus e_{j+1}, e_j)$ per slot.

Clearly, a routing scheme may be stable only if the average total demand per slot for transmissions over any fixed arc is less than unity. Considering only some arc of the form $(e_j \oplus e_{j+1}, e_j)$, it follows from the previous discussion that any routing scheme conforming to Rule A may be stable only if

$$\frac{\lambda}{d}2^{d-1} + \frac{\lambda}{d}(2^d - 1) < 1;$$

using the definition of ρ in (1), it is seen that the inequality above is equivalent to (20). Q.E.D.

The right-hand quantity in (13) is very close to that in (20), even for moderately small values of d; e.g., for $d = 8$ they differ only by 0.26%, and for $d = 10$ only by 0.065%.

The proof of Proposition 8 suggests that the basic limitation of the routing scheme under analysis lies on the fact that each of the trees used for routing is unbalanced; this creates *bottlenecks* in the arcs from which the largest subtrees are hanging. The previous argument is further supported by looking at the proof of Proposition 5, as well; it was proved therein that the nodes of the largest subtree $T_1^{(j)}$ introduce as much input traffic to tree $T^{(j)}$ as all other nodes together. This is also demonstrated by Rule E, which makes even more clear that, under heavy load, the scheme reduces to *round-robin*. Indeed, it may be seen that, for $\rho \to (2/3)(1 - (1/2^d))$, root e_j is alternately broadcasting one packet originating at some node in $T_1^{(j)}$ and one packet originating elsewhere.

The stability properties of the scheme under analysis are rather satisfactory; unfortunately, they are not the best possible, since the scheme becomes unstable for $\rho \approx \frac{2}{3}$. Suppose now that we could imbed d *balanced* disjoint spanning trees in the d-cube. (A spanning tree of the d-cube is characterized as "balanced," if it has d subtrees of approximately the same size.) If such an imbedding is possible, then our scheme could be modified so as to remain stable up to $\rho \approx d/(d+1)$. The main idea would be to use one slot for transmissions towards the roots and d slots for broadcasting away from the roots. However, we do not know at present whether such an embedding is possible.

G. A Brief Comparison of the Various Routing Schemes

In the present subsection, we briefly compare the indirect scheme analyzed in Section V with the direct scheme of Section IV-C, namely the one involving d completely unbalanced trees per node. Both schemes exhibit satisfactory stability properties. Henceforth, we focus on the delay properties of the schemes and on the sizes of the queues involved.

As argued in Section IV-C, the direct scheme appears to satisfy $D \approx d + (1/2) + (d/3)\rho$ when ρ is small. On the other hand, the indirect scheme of Section V satisfies, under light traffic, $D \approx 3d + 1 + (9/4)\rho$; thus, it is outperformed by the direct scheme with this respect. This is due to the fact that the maximum propagation time per packet [which equals $\lim_{\rho\to 0}(D - (1/2))$] is considerably larger under the indirect scheme. Notice now that the indirect scheme exhibits

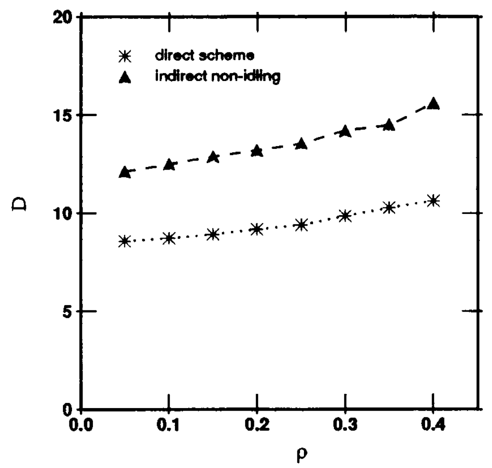


Fig. 5. Comparing the delay induced by two schemes, for $d = 8$.

idling, because of the periodic alternation of the directions of the arcs, due to Rule C. If this rule is relaxed and redundant transmissions are avoided (see Section V-E), then there holds $\lim_{\rho\to 0} D = (3d/2)$; see [12]. (Notice that $\lim_{\rho\to 0} D$ exceeds the optimal value $d + (1/2)$, because packets do *not* travel through *shortest* paths.) Despite the improvement attained by eliminating idling from the indirect scheme, the direct scheme is still preferable with respect to delay. This is observed in Fig. 5, where we compare experimental results regarding the average delay per packet under the two schemes.

Next, we compare the values of the average queue-size Q for the various schemes; the values for the indirect scheme of Section V-B were computed by using Proposition 7, while the ones for the other two schemes were obtained experimentally. (Notice the peculiar behavior of Q for the indirect scheme of Section V-B; this is due to the nondominant term in the expression of Proposition 7, which is nonnegligible for small values of d.) As revealed by Fig. 6, the nonidling version of the indirect scheme is the most efficient one with respect to queue-sizes. It is particularly important that, for fixed ρ, the queue-size Q grows more slowly with d under the nonidling indirect scheme. In order to make the comparison even more clear, we have also plotted the *maximum queue-sizes M* observed in the various simulation runs corresponding to Fig. 6; each simulation lasted for 1000 slots. Again, the nonidling indirect scheme is superior to the direct one; see Fig. 7.

Finally, we discuss the issue of deadlock prevention, when implementing the routing schemes with finite buffers. As argued in Section V-E, the indirect scheme (in its original version) is deadlock-free when implemented with $\Theta(d)$ buffer capacity per node. Regarding the nonidling indirect scheme, again deadlocks can be prevented by dedicating (at each individual node) constant buffer capacity to packets routed along each tree $T^{(j)}$; packets heading towards root e_j should have access to *different* buffers from those used by packets traveling away from e_j. Despite the contention among packets routed along different trees, there never arises a group of packets blocking the *buffer* of one another in a "cyclic"

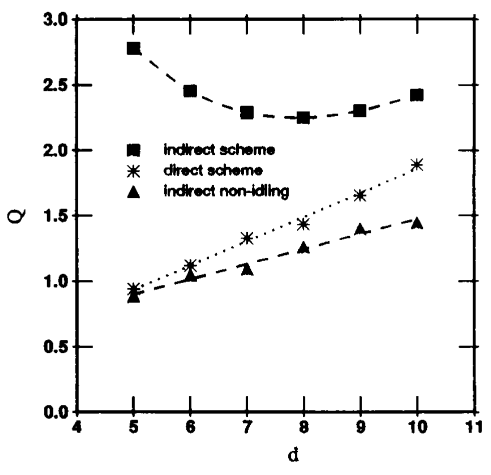


Fig. 6. Comparing the average queue-size Q per node for the various schemes, for $\rho = 0.3$.

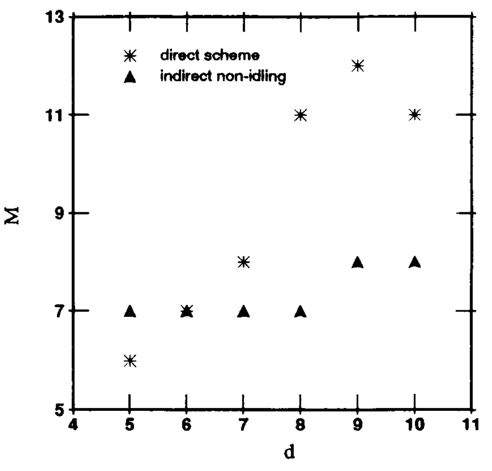


Fig. 7. Comparing the maximum queue-size M per node for two schemes, for $\rho = 0.3$.

pattern. As for the direct scheme, deadlocks can be prevented by using one of the standard techniques, such as that of "structured buffer pool" introduced by Raubold and Haenle [10]. According to this well-known technique, the buffer of each node of the d-cube should be partitioned in several segments, with the ith segment being accessible *only* to packets already having traversed $i-1$ arcs. Therefore, a buffer capacity of $\Theta(d)$ per node is required to prevent deadlocks, both for the direct and the nonidling indirect schemes. Thus, in principle, *all* of our schemes can be made deadlock-free, even in the presence of other packet transmissions. Since deadlock prevention techniques often result in degradation of the throughput, it is not clear which technique is the most appropriate for each routing scheme. However, we conjecture that, under deadlock prevention, both the original and the nonidling indirect schemes perform considerably better than the direct one, because they are "inherently" deadlock-free. Further investigation of these issues exceeds the scope of the present paper.

The conclusion drawn from the previous discussion is that the direct scheme is preferable under light traffic (because it induces smaller delays) and under very heavy traffic (because it maintains stability, unlike the indirect schemes). On the other hand, the nonidling indirect scheme may be preferable under moderate traffic because it involves smaller queues. Finally, deadlock prevention is much easier for the two indirect schemes.

VI. Concluding Remarks

In this paper, we have formulated a problem where packets to be broadcast are generated by the nodes of the d-dimensional hypercube at random instants, according to Poisson processes with rate λ. All packets were taken to have unit length; also, it was assumed that no other packet transmissions are taking place in the network. We showed that any routing scheme used for performing these broadcasts can be stable only if $\rho < 1$, where $\rho \overset{\text{def}}{=} \lambda(2^d - 1)/d$ is the load factor of the system. Moreover, we derived two lower bounds on the steady-state average delay D per packet. Given these limitations, our goal was to devise distributed schemes that can accommodate considerably high traffic (while maintaining stability), while satisfying the following delay requirement under light traffic: $D \leq Kd(1+\rho)$, with constant K.

We considered two classes of schemes, namely direct and indirect ones. Under direct schemes, packets are broadcast directly by the respective origins; to the contrary, under indirect schemes, packets are sent to special nodes, which perform the broadcasts. In particular, an indirect scheme based on a construction of d disjoint spanning trees by Johnsson and Ho [7], was shown to be stable for all $\rho < (2/3)(1-(1/2^d)) \approx 2/3$; the corresponding average delay per packet was proved to satisfy $D \approx 3d + 1 + (9/4)\rho$ for small values of ρ; both of these results were established rigorously. Furthermore, a direct routing scheme, using a nonidling version of the optimal algorithm for d simultaneous multinode broadcasts by Saad and Schultz [11], was shown to be stable for $\rho < 1$. For this scheme, it was conjectured that $D \approx d + (1/2) + Kd\rho$; this claim has been verified by means of an approximate model as well as by simulation. The aforementioned scheme is also rather easy to implement. For all of the schemes considered, we also studied the behavior of the average queue-size Q per node, in order to estimate the buffer capacity required in practice. The schemes proved to be efficient with respect to queue-sizes, as well. We also discussed the issue of deadlock prevention when implementing the schemes with finite buffers.

A considerable part of the analysis would also hold under a more general distribution of the packet-generating random process; in particular, the various conditions for stability are rather general. Some of the techniques used may also be applied in analyzing the same problem in the context of other network topologies. Of course, in a more general version of the problem, it may be assumed that each packet is destined for a different subset of the nodes; it may also be assumed that the packets received by a node influence the packet-generating process of this node as well as the length of the new packets. This situation arises in the distributed execution

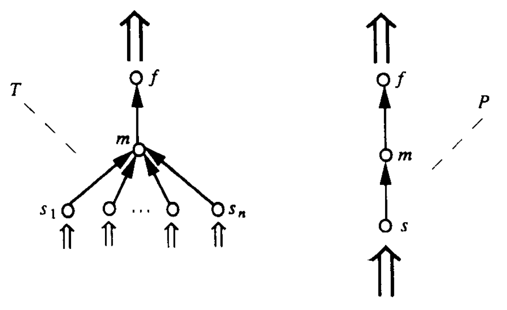


Fig. 8. The simple case for the proof of Lemma 3.

of iterative algorithms. Analyzing this general problem seems to be a rather challenging and interesting direction for further research.

APPENDIX

In this Appendix, we prove Lemmas 3 and 4 of Section V-C.

Proof of Lemma 3: First, we consider the case where all paths have length 2 (see Fig. 8). We denote by $M(t)$ a binary variable that equals 1 if and only if there is some packet buffered at node m of the tree T just before the end of slot t. The variable $\tilde{M}(t)$ refers to the single path P, and is defined in a similar way. Below, we prove that

$$\tilde{F}(t) = F(t) \quad \text{and} \quad \tilde{M}(t) = M(t), \qquad \text{for } t = 0, \cdots \tag{A.1}$$

(recall the notation introduced in Section V-C). The proof will be done by induction on t.

Clearly, we have $\tilde{M}(0) = M(0) = 0$ and $\tilde{F}(0) = F(0) = 0$, which establishes $(A.1)$ for $t = 0$. Next, we assume that the induction hypothesis holds for all $t \in \{0, \cdots, t^*\}$; based on this, we show that it holds for $t = t^* + 1$, as well. Indeed, we have $F(t^* + 1) = M(t^*)$, because a packet may depart from the tree T at the end of slot $t^* + 1$ only if it were present at node m at the end of slot t^*. Similarly, we have $\tilde{F}(t^* + 1) = \tilde{M}(t^*)$. By the induction hypothesis, we have $\tilde{M}(t^*) = M(t^*)$; thus, it follows that $\tilde{F}(t^* + 1) = F(t^* + 1)$, which establishes the first part of $(A.1)$ for $t = t^* + 1$. It remains to show that $\tilde{M}(t^* + 1) = M(t^* + 1)$. We have $M(t^* + 1) = 1$ if and only if some packets that arrived at the tree by the end of slot t^* have not departed by the end of slot $t^* + 1$. That is, we have $M(t^* + 1) = 1$ if and only if $\sum_{t=0}^{t^*} \sum_{i=1}^{n} A_i(t) > \sum_{t=0}^{t^*+1} F(t)$. Similarly, we have $\tilde{M}(t^*+1) = 1$ if and only if $\sum_{t=0}^{t^*} \tilde{A}(t) > \sum_{t=0}^{t^*+1} \tilde{F}(t)$. Recall now that $\tilde{A}(t) = \sum_{i=1}^{n} A_i(t)$ for $t = 0, \cdots$, by assumption. Moreover, we have $\tilde{F}(t) = F(t)$ for $t = 0, \cdots, t^*$ (by the induction hypothesis) and we have established that $\tilde{F}(t^*+1) = F(t^*+1)$. Thus, it follows that whenever $M(t^*+1) = 1$ holds, then $\tilde{M}(t^* + 1) = 1$ also holds, and vice versa. Clearly, this proves that $\tilde{M}(t^* + 1) = M(t^* + 1)$.

Now that the lemma has been established for the simple case in Fig. 8, the result is easily extended for the general case in Fig. 3. It suffices to progressively collapse the paths of the tree T, starting from the lowest level. Q.E.D.

Outline of the Proof of Lemma 4: We consider the tree T obtained from $\tilde{T}$ by adding a tandem of $l^* - l_i$ incoming arcs to each leaf s_i, where $l^* \overset{\text{def}}{=} \max\{l_1, \cdots, l_n\}$; thus, all paths of the new tree T have the *same* length l^*. (For the tree $\tilde{T}$ of Fig. 4, tree T coincides with the one of Fig. 3(a).) In the context of T, transmissions also start at beginning of slots $0, \Delta, \cdots$, and each of them lasts for Δ slots. We *couple* the arrivals in the two trees T and $\tilde{T}$. A straightforward inductive argument shows that the departure process from T is a *delayed* version of that from $\tilde{T}$; that is, the jth departure time in T is greater than (or equal to) the jth departure time in $\tilde{T}$, for $j = 1, \cdots$. (Notice that, for a single path such as the one of Fig. 3(b), the departure process is delayed when the path is augmented.) Therefore, on a sample-path basis, tree T contains *at least* as many packets as tree $\tilde{T}$; this implies that if T is stable, then $\tilde{T}$ is stable as well.

Notice now that we may apply Lemma 3 and collapse the paths of tree T, because all of them have the same length. (Recall the comments in Section V-C on the validity of Lemma 3 in more general cases.) Thus, regarding its departure process, T is equivalent to a tandem P of l^* deterministic servers (each with service time Δ) fed by a Poisson process with rate λn. Since all servers of this tandem P are identical, *no queueing* takes place in servers other than the first. Thus, the tandem P is stable if and only if the first server is stable, namely if and only if $\lambda n \Delta < 1$. Of course, this stability condition also applies to the tree T. By the conclusion of the previous paragraph, the original tree $\tilde{T}$ is also stable if $\lambda n \Delta < 1$. It is straightforward that this condition is also necessary for $\tilde{T}$ to be stable.

Furthermore, the steady-state average delay D' per packet for both tandem P and tree T is given as follows:

$$D' = D_1 + \Delta(l^* - 1) \tag{A.2}$$

where D_1 is the average delay induced per packet by the first server of the tandem P. Recall now that this server is fed by a Poisson process with rate λn; also, service may only start at times $0, \Delta, \cdots$ and lasts for Δ slots. It is easily seen that the departure process from this server would have been the *same* if new packets were arriving in batches only at times $0-, \Delta-, \cdots$, with the batch-size distributed as Poisson with expected value $\lambda n \Delta$. In such a case we would have

$$D_1 = \Delta\left[1 + \frac{\lambda n \Delta}{2(1 - \lambda n \Delta)}\right];$$

see [9]. Now that arrivals at the server occur in continuous time, each packet waits for an additional $\Delta/2$ slots on the average; therefore, we actually have

$$D_1 = \Delta\left[1 + \frac{\lambda n \Delta}{2(1 - \lambda n \Delta)}\right] + \frac{\Delta}{2};$$

this together with (A.2) implies that the average delay D' per packet induced by the tree T is given as follows:

$$D' = \Delta\left[\frac{3}{2} + \frac{\lambda n \Delta}{2(1 - \lambda n \Delta)}\right] + \Delta(l^* - 1). \tag{A.3}$$

Next, consider a single path, such as the one presented in Fig. 3(b); assuming that the arrival process feeding this path is Poisson, it is obvious that the *steady-state* statistics of the corresponding *departure* process *do not depend on the length* of the path (provided that the path has nonzero length). Using this property, and a straightforward inductive argument, the following result may be proved: At each *nonleaf* node where streams of packets *merge*, the processes feeding the node have the *same* steady-state statistics in both trees $\tilde{T}$ and T. Thus, an arc *shared* by packets originating from multiple leaves induces the *same average delay* per packet in both trees. Recall now how T was constructed from the original tree $\tilde{T}$. Observing also Figs. 4 and 3(a), it is apparent that $\tilde{T}$ can be obtained from T by eliminating a tandem of $l^* - l_i$ *contention-free* arcs for each leaf s_i; the arcs eliminated should be among the ones traversed by the packets originating at s_i *prior to meeting* with packets generated elsewhere. We have already proved that each arc shared by packets originating from multiple leaves induces the same average delay in both trees. Hence, the average delay $\tilde{D}$ per packet in the context of $\tilde{T}$ equals that induced in T (namely, D') *reduced* by the average time spent per packet in the contention-free arcs that must be eliminated from T (to yield $\tilde{T}$). A packet originating at leaf s_i would spend $\Delta(l^* - l_i)$ time units in these arcs; since a typical packet is equally likely to originate at any of the leaves $s_1, \cdots, s_n$ of T, it follows that

$$\tilde{D} = D' - \frac{\Delta}{n}\sum_{i=1}^{n}(l^* - l_n).$$

Combining this with (A.3), we obtain the expression for $\tilde{D}$.
Q.E.D.

References

[1] S. Abraham and K. Padmanabhan, "Performance of the direct binary n-cube network for multiprocessors," in *Proc. 1986 Int. Conf. Parallel Processing.*

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[3] D. P. Bertsekas, C. Ozveren, G. D. Stamoulis, P. Tseng, and J. N. Tsitsiklis, "Optimal communication algorithms for hypercubes," *J. Parallel Distrib. Comput.*, vol. 11, pp. 263–275, 1991.

[4] S. L. Brumelle, "Some inequalities for parallel-server queues," *Oper. Res.*, vol. 19, pp. 402–413, 1971.

[5] Y. Chang and J. Simon, "Continuous routing and batch routing on the hypercube," in *Proc. 5th ACM Symp. Principles Distrib. Comput.*, 1988, pp. 272–281.

[6] A. G. Greenberg and B. Hajek, "Deflection routing in hypercube networks," preprint, 1989.

[7] S. L. Johnsson and C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.*, vol. 38, pp. 1249–1267, 1989.

[8] L. Kleinrock, *Queueing Systems, Vol. I: Theory.* New York: Wiley, 1975.

[9] H. Kobayashi and A. G. Konheim, "Queueing models for computer communications systems analysis," *IEEE Trans. Commun.*, vol. 25, pp. 2–29, 1977.

[10] E. Raubold and J. Haenle, "A method of deadlock-free resource allocation and flow control in packet networks," in *Proc. 3rd Internat. Conf. Comput. Commun.*, 1976, pp. 483–487.

[11] Y. Saad and M. H. Schultz, "Data communication in hypercubes," Dep. Comput. Sci., Res. Rep. YALEU/DCS/RR-428, Yale Univ., 1985.

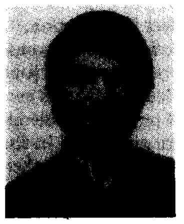
[12] G. D. Stamoulis, "Routing and performance evaluation in interconnection networks," Ph.D. dissertation, Dep. Elec. Eng. and Comput. Sci., M.I.T, 1991.

[13] G. D. Stamoulis and J. N. Tsitsiklis, "The efficiency of greedy routing in hypercubes and butterflies," in *Proc. 3rd ACM Symp. Parallel Algorithms and Architectures,* Hilton Head, NC, July 1991.

[14] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *Proc. 13th Annu. ACM Symp. Theory of Comput.*, 1981, pp. 263–277.

[15] L. G. Valiant, "A scheme for fast parallel communication," *SIAM J. Comput.*, vol. 11, pp. 350–361, 1982.

[16] E. A. Varvarigos, "Optimal communication algorithms for multiprocessor computers," Rep. CICS-TH-192, Center for Intelligent Control Systems, M.I.T., 1990.



George D. Stamoulis was born in Athens, Greece, in 1964. He received the diploma in electrical engineering in 1987 (with highest honors) from the National Technical University of Athens, Athens, Greece, and the M.S. degree in 1988 and the Ph.D. degree in 1991 in electrical engineering from the Massachusetts Institute of Technology, Cambridge.

He is currently serving in the Hellenic Navy, as a lecturer in the Hellenic Naval Academy. He is also a Research Associate with the communication networks group of the Department of Electrical and Computer Engineering, National Technical University of Athens; he is participating in RACE projects. His research interests are in the areas of routing and performance evaluation of multiprocessing systems and communication networks, and queueing theory.

Dr. Stamoulis was among the winners of the Greek Mathematic Olympiad in both 1981 and 1982. He also participated in the 23rd International Mathematic Olympiad, in Budapest, in July 1982. He is a member of the Technical Chamber of Greece and Sigma Xi.

John N. Tsitsiklis (S'80–M'83) was born in Thessaloniki, Greece, in 1958. He received the B.S. degree in mathematics in 1980, the B.S. degree in 1980, the M.S. degree in 1981, and the Ph.D. degree in 1984 in electrical engineering, all from the Massachusetts Institute of Technology, Cambridge.

During the academic year 1983–1984 he was an Acting Assistant Professor of Electrical Engineering at Stanford University, Stanford, CA. Since 1984, he has been with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, where he is currently Professor. His research interests are in the areas of parallel and distributed computation, systems and control theory, and operations research. He is co-author of *Parallel and Distributed Computation: Numerical Methods* (1989).

Dr. Tsitsiklis has been a recipient of an IBM Faculty Development Award in 1983, an NSF Presidential Young Investigator Award in 1986, an Outstanding Paper Award by the IEEE Control Systems Society, and of the Edgerton Faculty Achievement Award by M.I.T. in 1989. He is an associate editor of *Applied Mathematics Letters* and *Automatica* and has been an associate editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.