February 1990/Revised August 1990 LIDS-P-1948

# EFFICIENT ROUTING SCHEMES FOR MULTIPLE BROADCASTS IN HYPERCUBES †

by

**George D. Stamoulis ‡ and John N. Tsitsiklis ‡**


## Abstract

We analyze the following problem: Each node of the $d$-dimensional hypercube independently generates packets according to a Poisson process with rate $\lambda$; each of the packets is to be broadcast to all other nodes. Assuming unit packet length and no other communications taking place, we show that the system can be stable in steady-state only if the load factor $\rho \stackrel{\text{def}}{=} \lambda \frac{2^d - 1}{d}$ satisfies $\rho \leq 1$; moreover, we establish some lower bounds for the steady-state average delay $D$ per packet. We devise and analyze several distributed routing schemes that prove to be efficient, in the following sense: Stability is maintained for all $\rho < \rho^*$, where $\rho^*$ does not depend on the dimensionality $d$ of the network, while the delay $D$ is $\Theta(d) + O(d)\rho$ for small values of $\rho$. Performance evaluation of the various schemes is exact to a considerable extent; in some cases, exact analysis is intractable and we resort to approximations and/or simulations.


† Research 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.

‡ Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Mass. 02139, USA.

## 1. INTRODUCTION

During the execution of parallel algorithms in a network of processors, it is often necessary that one of the processors broadcasts a piece of information to all others; subsequent broadcasts (possibly by different processors) may also take place, until the algorithm terminates. In this paper, we consider a situation 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.

We consider the $d$-dimensional hypercube (or $d$-cube); e.g. see [BeT89]. This network consists of $2^d$ nodes, numbered from 0 to $2^d - 1$. Associated with each node $z$ is a binary identity $(z_d, \ldots, z_1)$, 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 $m \in \{1, \ldots, d\}$. Note that $(z, y)$ stands for a unidirectional arc pointing from $z$ to $y$; of course, if arc $(z, y)$ exists, so does arc $(y, z)$. It is easily seen that 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 $d$.

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 piece of information 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.

Whenever some node wishes to broadcast a packet, it just has to transmit it along a spanning tree emanating from this node. (That is, a spanning tree with all of its arcs pointing away from this node.) If no other packet transmissions take place at this time, then all nodes will have received the packet under broadcast after some time equal to the depth of the selected tree. This simple communication task (scenario) is called the single node broadcast. Clearly, the optimal time for performing this task in the $d$-cube is $d$ and it can be attained by broadcasting the packet along any tree with depth $d$. (There are several such trees from which to choose.) Another interesting communication task is the multinode broadcast, where all nodes wish to perform a broadcast at the same time (see [BeT89]). This situation arises in the distributed execution of any iterative algorithm of the form $x := f(x)$, where $f : \Re^n \to \Re^n$ and $n$ is the number of nodes; typically, the $i$th node knows the function $f_i$ and updates $x_i$. Assume that the problem is dense, i.e. each entry of the function $f(x)$ depends explictly 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. The minimum possible time for the multinode broadcast (in the $d$-cube) is $\lceil \frac{2^d-1}{d} \rceil$ and it can be attained by an algorithm by Bertsekas et al. [BOSTT89]. However, there are cases where the new values of the $x_i$'s are not all computed at the same time. Such asynchronous algorithms have received considerable attention in the literature, but have mainly been analyzed from the point of view of convergence. This kind of analysis often involves some a priori assumptions on the delay suffered in broadcasting some piece of information. In this paper we make a first attempt to analyze this communication delay in the context of the hypercube network, by considering a simple situation involving asynchronous single node broadcasts.

In particular, we assume that each node of the $d$-cube generates packets according to a Poisson process with rate $\lambda$; different nodes generate their packets independently of each other. All packets generated are to be broadcast; it is assumed that no other packet transmissions are taking place in the network. As will be proved in §3.1, the inequality

$$\rho \stackrel{\text{def}}{=} \lambda \frac{2^d - 1}{d} \leq 1$$

is a necessary condition for stability; $\rho$ will be called the load factor of the system. 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, the performance of such a routing scheme can be rather poor (see §4.1). Another straightforward approach to the problem of interest is to perform multinode broadcasts periodically. In §4.2, we propose three such schemes that are stable even when the traffic is high (namely, for $\rho \approx 1$). Unfortunately, these schemes will be seen to introduce unacceptably high delay even in light traffic (namely, for $\rho \approx 0$). Thus, we are mainly concerned with devising schemes that are stable even when the load factor is non-trivial, while for small values of $\rho$ the average delay $D$ be $\Theta(d) + O(d)\rho$. (Note that $D$ is defined as the steady-state average time spent by a packet in the system until its broadcast is completed.) The intuition for this requirement for the delay is basically the fact that it takes $d$ time units to perform a single node broadcast in the $d$-cube in the absence of other transmissions; more discussion on this point is given in §3.3, following the derivation (in §3.2) of two lower bounds for $D$. In §§4 and 5, we present several distributed routing schemes that meet the aforementioned performance criteria. The schemes discussed in §4 are characterized as direct ones, because each packet is broadcast along a spanning tree rooted at the node where it was generated. In particular, in §4.3, we present the non-idling versions of the two periodic schemes of §4.2. These schemes are stable even for $\rho \approx 1$, while they seem to satisfy the desirable delay properties; we provide some strong evidence for this, based on an approximate model and simulation. In §5, 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 boradcasts. This scheme is based on a construction of $d$ disjoint spanning trees by Johnsson and Ho [JoH89]; it will be seen to be stable for all $\rho < \frac{2}{3}(1 - \frac{1}{2^d})$, while it satisfies $D \approx 3d + 7 + \frac{9}{4}\rho$ for small $\rho$. In evaluating the performance analysis of the various schemes, we also consider the steady-state average queue-size

$Q$ per node. In fact, our schemes appear to be efficient also with respect to queue-sizes. Study of the behavior of the measure $Q$ aims at estimating the buffer capacity required for applying the schemes in practice. In particular, it is argued that the assumed infinite buffer capacity can be replaced by quite small buffers while keeping the probability of buffer overflow negligible.

To the best of our knowledge, the problem formulated in this paper as well as the results derived are new. Of course, there exists considerable literature on algorithms for communication tasks in various interconnection networks. However, several 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. [BOSTT89] have devised optimal algorithms for a variety of communication tasks. Previously, Saad and Schultz [SaS85], as well as Johnsson and Ho [JoH89], 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 [BeT89].

The communication tasks considered in the aforementioned papers as well as the respective algorithms do not employ any randomization. In his famous paper [Val82], 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 $\Theta(d)$ with high probability, by using a randomized two-phase algorithm. In a later paper, Valiant and Brebner [VaB81] modified this algorithm, thus simplifying considerably the analysis. Some related articles (e.g., [Ale82] and [Upf84]) deal with more general tasks or refer to other network topologies.

There is only one problem involving repetitive packet transmissions that has appeared in the literature; this is as follows: The nodes of the hypercube network generate packets according to some random process, with each packet having a single destination; each packet's destination is uniformly distributed among the nodes of the network. Abraham and Padmanabhan [AbP86] have constructed an approximate model for this problem, under various assumptions on the buffer capacity of the nodes. Greenberg and Hajek [GrH89] have analyzed this problem under the assumption that deflection routing is used instead of shortest path routing; that is, packets may be temporarily misrouted, rather than stored or dropped which was the case in [AbP86]. The analysis in [GrH89] is approximate too. Varvarigos [Var90] has formulated a Markov chain model for this problem, and has investigated its stationary statistics numerically. The same problem has been analyzed in the context of the Manhattan network (square mesh) by Greenberg and Goodman [GrG86], with their analysis being again approximate. Finally, similar problems were analyzed by Mitra and Cieslak [MiC86], as well as by Hajek and Cruz [HaC87], in the context of the extended Omega network (which is a crossbar switch). In these two papers, randomized schemes were used for routing packets from one end of the switch to the other, with each packet having one destination.

Motivation for studying the problem introduced in this paper is as follows: As already mentioned,

there are cases where different nodes wish to broadcast their packets in different and unpredictable times, rather than performing a multinode broadcast which would require a high level of synchronization. Our analysis may be viewed as a first step towards treating such chaotic situations. Our interest is towards the direction of general purpose computation, where lots of tasks appear unpredictably and their arrival rate $\lambda$ is unknown (or not well-defined). 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.

## 2. BACKGROUND MATERIAL

### 2.1 Background on Discrete-Time Queues

#### 2.1.1 The Discrete-Time $G/D/1$ Queue With Unit Service Time

In this subsection, we briefly present some properties of the discrete-time $G/D/1$ queue with unit service time; this is a single server queueing system operating as follows:

(a) The time axis is divided into slots of unit duration.

(b) The number of arrivals $A_t$ during slot $t$ is distributed as the random variable $A$, for $t = 0, \ldots$; thus, we have $E[z^{A_t}] = E[z^A] \stackrel{\text{def}}{=} G_A(z)$. Arrivals within different slots are mutually independent. Customers arrive in the system just before the end of each slot.

(c) Each customer's service time equals one slot.

The queue under consideration is stable if and only if either $E[A] < 1$ or $\Pr[A = 1] = 1$; henceforth, we assume $E[A] < 1$. We denote by $Y_t$ the number of customers present in the system at the beginning of slot $t$, including the customer in service (if any). It follows from the above discussion that

$$Y_{t+1} = [Y_t - 1]^+ + A_t \text{ , for } t = 0, \ldots , \tag{1}$$

where $[\alpha]^+ \stackrel{\text{def}}{=} \max\{0, \alpha\}$. The steady-state distribution of $Y_t$ can be derived by using standard tools of queueing theory. In particular, we have [KoK77]

$$G_Y(z) \stackrel{\text{def}}{=} \lim_{t \to \infty} E[z^{Y_t}] = (1 - E[A]) \frac{z - 1}{z - G_A(z)} G_A(z) \text{ , for } |z| \leq 1 . \tag{2}$$

Using the fact $\Pr[Y = 0] = \lim_{z \to 0} G_Y(z)$, it follows that

$$\lim_{t \to \infty} \Pr[Y_t \neq 0] = E[A] . \tag{3}$$

The expected number $N$ of customers in the system (in steady-state) can be calculated by evaluating $\frac{dG_Y(z)}{dz}|_{z=1}$; thus, it follows from (2) that

$$N \stackrel{\text{def}}{=} \lim_{t \to \infty} E[Y_t] = E[A] + \frac{E[A(A-1)]}{2(1 - E[A])} .$$

Using Little's formula, we have

$$D = \frac{N}{E[A]} = 1 + \frac{E[A(A-1)]}{2E[A](1-E[A])}, \tag{4}$$

where $D$ is the steady-state average delay of each customer, the service time included.

### 2.1.2 The Discrete-Time $M/D/1$ Queue With Synchronization

We consider a discrete-time $G/D/1$ queue with unit service time and with the number of arrivals per slot assuming the Poisson distribution. In this case, we have $E[A(A-1)] = E^2[A]$; thus, (4) gives

$$D = 1 + \frac{E[A]}{2(1-E[A])}. \tag{5}$$

So far, we have assumed that arrivals may occur only at the end of each slot. If we assume instead that the arrival instants are points of a continuous-time Poisson process with rate $\lambda < 1$, then the analysis leading to (2) still holds. (Again, service may only start at the beginning of a slot.) However, a correction in the expression for the average delay $D$ is necessary. In particular, we have to add to the right-hand quantity in (5) the expected duration of the time elapsing between the arrival instant of some customer and the beginning of the next slot. This quantity is called the average synchronization time and it equals $\frac{1}{2}$. Thus, we obtain

$$D = \frac{3}{2} + \frac{E[A]}{2(1-E[A])} = \frac{3}{2} + \frac{\lambda}{2(1-\lambda)}. \tag{6}$$

Finally, consider the queueing system described above, but modified as follows: Service may only start every $\Delta$ slots, say in the beginning of the slots numbered $0, \Delta, \ldots$; moreover, the service time duration is now set to $\Delta$. We will be referring to this system as the discrete-time $M/D/1$ queue with synchronization. The number of arrivals during $\Delta$ successive slots is Poisson, with rate $\lambda\Delta$. Moreover, the arrivals in different intervals of duration $\Delta$ are mutually independent. Thus, in order to derive the expression for the steady-state average delay $D$, we simply have to apply (6) with $\lambda\Delta$ instead of $\lambda$ and rescale time by $\Delta$; thus, we obtain

$$D = \Delta\Big(\frac{3}{2} + \frac{\lambda\Delta}{2(1-\lambda\Delta)}\Big). \tag{7}$$

Clearly, the discrete-time $M/D/1$ queue with synchronization is stable if and only if $\lambda\Delta < 1$.

### 2.2 Background on the Hypercube Network

The fundamental properties of the hypercube network were briefly mentioned in §1. In the present section, we present some additional background material.

#### 2.2.1 Definitions

Let there be two nodes $z$ and $y$ of the $d$-cube. We denote by $z \oplus y$ the vector $(z_d \oplus y_d, \ldots, z_1 \oplus y_1)$, where $\oplus$ is the symbol for the XOR operation. The $i$th (from the right) entry of $z \oplus y$ equals 1, if and only if $z_i \neq y_i$.

For $j \in \{1, \ldots, d\}$, 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 $j$th one (from the right), which equals 1. Nodes $e_1, \ldots, e_d$ are the only neighbors of node $(0, \ldots, 0)$. In general, each node $z$ has exactly $d$ neighbors, namely nodes $z \oplus e_1, \ldots, z \oplus e_d$.

Clearly, arc $(z, y)$ exists if and only if $z \oplus y = e_m$ for some $m \in \{1, \ldots, d\}$. Such an arc is said to be of the $m$th type; the set of arcs of the $m$th type is called the $m$th dimension.

Furthermore, we have $(v \oplus y) \oplus (v \oplus w) = y \oplus w$; this implies that if arc $(y, w)$ exists, then so does arc $(v \oplus y, v \oplus w)$ and vice versa. Next, consider a tree $T$ rooted at some node $z$. The aforementioned property implies that the set of arcs $\{((x \oplus z) \oplus y, (x \oplus z) \oplus w) : (y, w) \in T\}$ comprise a tree rooted at node $x$. This new tree is said to be a translation of the tree $T$; clearly, the two trees are isomorphic.

#### 2.2.2 The Completely Unbalanced Spanning Tree

For two nodes $z$ and $y$, let the $i$th entry of $z \oplus y$ be equal to 1, i.e. $z_i \neq y_i$; in this case, the shortest path from $z$ to $y$ contains one arc of the $i$th type. In general, let $i_1 < \cdots < i_k$ 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 $i_1$th type, one of them being of the $i_2$th 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 the various 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 $i_1 < \cdots < i_k$ 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 $i_1$th dimension, the second arc to the $i_2$th 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 §1, 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 $T_1, \ldots, T_d$. Each of them is hanging from one of the neighbors of $z$. Subtree $T_i$ consists of all nodes $y$ with the following property: $y_1 = z_1, \ldots, y_{i-1} = z_{i-1}$ and $y_i \neq z_i$ (see Fig. 1). Therefore, $T_i$ contains $2^{d-i}$ nodes which explains the terminology "completely unbalanced". 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 trees, isomorphic to the tree $T$ defined earlier. Henceforth, we call all of these trees completely unbalanced, as well.

Completely unbalanced trees have been used extensively in algorithms for hypercube communications (see [SaS85], [BOSTT89] and [JoH89]). Johnsson and Ho [JoH89] use the terminology "spanning binomial tree".

## 3. PRELIMINARY RESULTS

### 3.1 The 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 $\lambda 2^d(2^d - 1) \leq d2^d$. Thus, we have the following necessary condition for stability:

$$\rho \stackrel{\text{def}}{=} \lambda \frac{2^d - 1}{d} \leq 1 , \tag{8}$$

where $\rho$ is 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.

### 3.2 Lowers Bounds on the Average Delay per Packet

In the present section, we establish a universal lower bound on the 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.) We also establish a tighter lower bound applying to a certain class of routing schemes; whether the universal bound is tight or not is an open question.

First, we present the universal bound.

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

$$D = \Omega\Big(d + \frac{\rho}{d(1-\rho)}\Big) . \qquad \blacksquare$$

**Proof:** We denote by $\mathcal{L}(x)$ the set of arcs incoming at node $x$; i.e., $\mathcal{L}(x) \stackrel{\text{def}}{=} \{(x \oplus e_j, x) | j = 1, \ldots, d\}$. Any legitimate routing scheme for performing broadcasts must conform to the following two constraints:

(a) A packet generated at node $y$ must traverse at least one arc in the set $\mathcal{L}(x)$ for every node $x \neq y$.

(b) If a packet generated at node $y$ traverses arc $(x \oplus e_j, x)$, then either $x \oplus e_j = y$ or the packet has previously traversed an arc in the set $\mathcal{L}(x \oplus e_j)$.

The first of the above constraints guarantees that each packet is received by all nodes, while the second constraint guarantees that a packet traverses an arc only after being received by the arc's starting node. In fact, any routing scheme for performing multiple broadcasts, may be visualized as a plan for scheduling packet transmissions subject to the two above constraints and to the underlying assumptions on communications (see §1).

Next, we fix some node $x$ and we focus on the transmissions to be performed over the arcs of the set $\mathcal{L}(x)$. At each slot there are a number of such transmissions pending, including those transmissions currently in progress. Suppose that we relax constraint (b); that is, we assume that any packet may traverse any arc of $\mathcal{L}(x)$, provided that it has already been generated. (This is equivalent to assuming that all neighbors of $x$ receive instantaneously any packet generated elsewhere in the network.) Transmissions over the arcs of $\mathcal{L}(x)$ may now be done sooner, because the permissible schedules are less constrained than previously. Therefore, by relaxing constraint (b), node $x$ receives each of the packets no later than previously; thus, the average delay $D$ per packet may only decrease. The smallest possible value for $D$ would be attained if the following were true: For every node $x$, none of the packets generated at $x$ traverses any arc of $\mathcal{L}(x)$, while any packet generated elsewhere only traverses the first available arc of $\mathcal{L}(x)$. Clearly, this situation refers to a discrete-time $M/D/d$ queue, with unit service time and arrival rate $\lambda(2^d - 1) = \rho d$. [Recall also the definition of $\rho$ in (8).] Thus, we have

$$D \geq \frac{2^d - 1}{2^d} \mathcal{D}(d; \rho)\,, \tag{9}$$

where $\mathcal{D}(d; \rho)$ is the average delay for a discrete-time $M/D/d$ queue with unit service time and arrival rate $d\rho$; the factor $\frac{2^d-1}{2^d}$ accounts for the fact that packets generated by node $x$ do not join this queue. Furthermore, it is known [Bru71] that $\mathcal{D}(d; \rho)$ satisfies

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

Combining this with (9) and the fact $\frac{2^d-1}{2^d} \geq \frac{1}{2}$, we obtain

$$D \geq \frac{\rho}{4d(1-\rho)}\,.$$

This together with the obvious bound $D \geq d$ implies that

$$D \geq \max\left\{d, \frac{\rho}{4d(1-\rho)}\right\}\,.$$

Combining this with the inequality $\max\{\alpha,\beta\} \geq \frac{\alpha+\beta}{2}$, we obtain the result in question. **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 in a dynamic fashion. This claim is further supported by Proposition 2, which establishes a lower bound on $D$ applying to a certain class of routing schemes, to be referred to as static schemes. Under such a scheme, each packet generated by some node $z$ decides (upon arrival) which paths to follow independently of all other packets in the network; all packet generated by the same node $z$ follow the same rules. Clearly, this class comprises all schemes where each packet independently selects which tree to be broadcast along by using a randomized rule that depends only on the identity of its origin node. The routing schemes discussed in §§4.1 and 4.3 are of this type.

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

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

$$D = \Omega(d + \frac{\rho}{1-\rho}) \,. \qquad \blacksquare$$

**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 some node $z$ will attempt to cross some of the arcs of $\mathcal{L}(x)$; which arcs will be crossed is determined by a randomized rule depending only on $z$. Similarly as in the proof of Proposition 1, we relax constraint (b); this may only result in node $x$ receiving packets earlier than previously. Recall now that, under a static scheme, packets select their respective paths independently. Therefore, after relaxing constraint (b), each arc $(x \oplus e_j, x)$ is fed by a Poisson stream with rate $r_j$; by constraint (a), we have $\sum_{j=1}^{d} r_j \geq \lambda(2^d - 1)$. [Notice that the arrival stream pertaining to each arc $(x \oplus e_j, x)$ has constant rate, because packets generated by the same node follow the same rule.] Again, the average delay $D$ is minimized if packets generated by $x$ do not cross any arcs of $\mathcal{L}(x)$ while packets generated elsewhere cross exactly one such arc; in this case, we have

$$\sum_{j=1}^{d} r_j = \lambda(2^d - 1) = \rho d \,, \tag{10}$$

where we have also used (8). It is well-known that the average delay induced by a stable $M/D/1$ queue with arrival rate $r$ and unit service time equals $1 + \frac{r}{2(1-r)}$; see [Kle75]. Thus, it follows that

$$D \geq \sum_{j=1}^{d} \frac{r_j}{\lambda 2^d} \left[1 + \frac{r_j}{2(1-r_j)}\right] \,, \tag{11}$$

where we have also taken into account the fact that packets generated by node $x$ do not join the queue for any of the arcs of $\mathcal{L}(x)$. Notice now that $r[1 + \frac{r}{2(1-r)}]$ is a convex function of $r$; therefore, in light of (10), the expression in the right-hand quantity in (11) is minimized for

$r_1 = \cdots = r_d = \lambda \frac{2^d - 1}{d} = \rho$. Thus, we obtain

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

Since $\frac{2^d - 1}{2^d} \geq \frac{1}{2}$, we have

$$D \geq \frac{\rho}{4(1-\rho)} .$$

This together with the fact $D \geq d$ proves the result. **Q.E.D.**

Proposition 2 implies that, for a rather broad class of schemes, the universal lower bound on the delay $D$ (see Proposition 1) is loose. 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 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 involve either centralized coordination or some form of adaptive routing.

### 3.3 Setting the Performance Criteria

The bounds provided by (8) and Proposition 1 demonstrate the fundamental limitations for any routing scheme that may be applied to our problem; also the bound of Proposition 2 gives an additional limitation applying to a certain class of schemes. Given these bounds, we are interested in devising schemes that come fairly close to meeting both of them; in particular, such efficient schemes should satisfy the following performance criteria:

(a) Stability should be maintained (for all $d$) even when the load factor $\rho$ is $\Theta(1)$.

(d) Under very light traffic, i.e. for $\rho \approx 0$, the average delay $D$ should equal $Kd$, with $K$ being some constant; that is, $\lim_{\rho \to 0} D = Kd$. This requirement is suggested by the fact that it takes $d$ slots to perform a broadcast in the absence of other transmissions. Furthermore, the first order approximation of $D$ (as a function of $\rho$) should be of the form $D \approx Kd + O(d)\rho$. This requirement basically implies that, for moderately small values of $\rho$, the additional delay due to queueing should not be larger than a multiple of the network diameter.

Motivation for considering the first order approximation of $D$ is as follows: As implied by Propositions 1 and 2, the delay $D$ will definitely be large when the system is loaded close to capacity (i.e., for $\rho \approx 1$). Thus, it is expected that in practical applications (as well as in situations modelling asynchronous algorithms), the load factor $\rho$ has some moderately small value, say 0.2; moreover, a negligible value of $\rho$ would make very inefficient use of the network and the routing problem would be trivial.

Based on these performance criteria, we have devised and analyzed several efficient schemes. In particular, in §4, we are considering schemes where each packet is broadcast along some spanning

tree rooted at the node where it was generated; we shall be referring to these schemes as direct. On the other hand, in §5, we are considering indirect routing schemes; under such schemes, packets are sent to special nodes, which are in charge of broadcasting them.

## 4. DIRECT ROUTING SCHEMES

### 4.1 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 possibly 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 §2.2.2). Every node $z$ receives through its adjacent arc of the $j$th 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 $\lambda 2^{d-j}$ packets that will eventually have to traverse arc $(z \oplus e_j, z)$. Therefore, the simple scheme under analysis may be stable only if $\lambda 2^{d-j} \leq 1$ for $j = 1, \ldots, d$, or equivalently

$$\rho \leq \frac{2}{d}(1 - \frac{1}{2^d}) .$$

Hence, the load factor that can be sustained by the above simple scheme deteriorates to zero as the dimensionality $d$ of the hypercube increases; furthermore, the first of the performance criteria set in §3.3 is not met, because there exists no constant $\rho^*$ such that stability be guaranteed (for all $d$) if $\rho < \rho^*$. 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 [BOSTT89]. Since this algorithm lasts for $\lceil \frac{2^d-1}{d} \rceil$ slots, it follows that each arc is shared by at most $\lceil \frac{2^d-1}{d} \rceil$ of the trees. Broadcasting the packets along these trees will create no additional bottlenecks in any of the arcs. As is discussed in §4.3, this scheme performs very satisfactorily.

An alternative way of balancing traffic over the hypercube arcs is to allow for multiple trees per node and distribute among them the packets to be broadcast; an efficient routing scheme of this spirit is presented in §4.3. Both this scheme and the one mentioned above are closely related to the periodic schemes presented in §4.2.

### 4.2 Performing Multinode Broadcasts Periodically

In this section, we discuss schemes where an efficient algorithm for performing multinode broadcasts is run periodically. These schemes induce rather large delays; for this reason, they will be modified in §4.3.

### 4.2.1 A Routing Scheme Based on the Optimal Algorithm for the Multinode Broadcast

Bertsekas et al. [BOSTT89] have constructed an optimal algorithm for the multinode broadcast task. The optimal completion time for this communication task is $\lceil \frac{2^d-1}{d} \rceil$ slots.

Using this algorithm, we may construct the following routing scheme for our problem: Perform a multinode broadcast every $\lceil \frac{2^d-1}{d} \rceil$ time units; for every node that has no packet to broadcast, introduce a null packet.

We consider the packets generated by some node $z$ that have not started being broadcast yet. Clearly, under the above scheme, these packets form (in the buffer of $z$) a discrete-time $M/D/1$ queue with synchronization; the arrival rate is $\lambda$ and the service time duration equals $\lceil \frac{2^d-1}{d} \rceil$. Therefore, the scheme is stable if and only if $\lambda \lceil \frac{2^d-1}{d} \rceil < 1$, or equivalently

$$\rho < \frac{(2^d-1)/d}{\lceil (2^d-1)/d \rceil} . \tag{12}$$

It is known that $\frac{2^d-1}{d}$ is never an integer for $d \geq 2$; see [BOSTT89] for a simple proof of this fact. This implies that the scheme under analysis becomes unstable if $\rho$ is sufficiently close to 1. Of course, if $d$ is not very small, then the right-hand quantity in (12) is very close to 1; e.g., for $d = 10$ (which is a realistic value), this quantity equals 0.9932. Thus, for all practical purposes, the load factor that can be accommodated under this scheme is very high.

The average delay $D$ may be derived by applying (7) with $\Delta = \lceil \frac{2^d-1}{d} \rceil$; thus, we obtain

$$D = \left\lceil \frac{2^d-1}{d} \right\rceil \left( \frac{3}{2} + \frac{\lambda \lceil \frac{2^d-1}{d} \rceil}{2(1 - \lambda \lceil \frac{2^d-1}{d} \rceil)} \right) = \left\lceil \frac{2^d-1}{d} \right\rceil \left( \frac{3}{2} + \frac{\rho}{2\left( \frac{(2^d-1)/d}{\lceil (2^d-1)/d \rceil} - \rho \right)} \right) .$$

For $\rho \approx 0$, we have $D \approx \frac{3}{2} \lceil \frac{2^d-1}{d} \rceil$; therefore, this routing scheme introduces unacceptably high delay when the network is lightly loaded. [Recall that we wish to have $D \approx \Theta(d)$ for $\rho \approx 0$.]

### 4.2.2 Two Routing Schemes Based on the Optimal Algorithm for the $d$ Simultaneous Multinode Broadcasts

The routing scheme presented in this subsection is of the same spirit as that of §4.2.1, except for the fact that the optimal algorithm by Saad and Schultz [SaS85] for the $d$ simultaneous broadcasts is now used. 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 $z$. In particular, the first packet is routed along that tree where the hypercube dimensions are crossed in the order $1, 2, \ldots, d$; the second packet is routed along that tree where the hypercube dimensions are crossed in the order $2, 3, \ldots, d, 1$ etc; the $d$th packet is routed along that tree where the hypercube dimensions are crossed in the order $d, 1, 2, \ldots, d-1$. Each packet that traverses the hypercube dimensions under the order $j, (j \bmod d) + 1, \ldots, [(j + d - 2) \bmod d] + 1$ will be called a <u>$j$-packet</u>.

The optimal algorithm of [SaS85] takes time $2^d - 1$. Using this algorithm, we may construct the following routing scheme for our problem: Perform $d$ simultaneous multinode broadcasts every $2^d - 1$ time units; for every node that has less than $d$ packets to broadcast, introduce null packets accordingly.

Unlike the scheme presented in §4.2.1, a discrete-time $M/D/d$ queue with synchronization is formed in each node's buffer; the arrival rate is $\lambda$, but the service time duration now equals $2^d - 1$. The system is stable if and only if $\lambda(2^d - 1) < d$, or equivalently

$$\rho < 1 .$$

Thus, stability is maintained even when the value of the load factor $\rho$ is arbitrarily close to 1. Since the average delay for a multi-server queue is rather hard to derive, we will not provide an exact expression for $D$. However, it is easy to see that, for $\rho \approx 0$, we have $D \approx \frac{3}{2}(2^d - 1)$, which is even worse than for the scheme of §4.2.1.

Next, we introduce a modification to the scheme presented above, in order to make it more tractable analytically. In particular, we assume that, upon arrival, each packet selects randomly the spanning tree along which it will be broadcast. As already mentioned, every packet has to choose from $d$ permissible trees; each of these trees is assigned an a priori probability $\frac{1}{d}$; different packets make their selections independently. Under the terminology introduced above, each packet selects an index $j$ randomly and declares itself as a $j$-packet. It is easily seen that all $j$-packets generated at the same node $z$ now join a discrete-time $M/D/1$ queue with synchronization (prior to undertaking any transmisssions). The queues corresponding to the various values of $j$ are independent of each other. The arrival rate for each such queue is $\frac{\lambda}{d}$ and the service time duration equals $2^d - 1$. The necessary and sufficient condition for stability is $\frac{\lambda}{d}(2^d - 1) < 1$, or equivalently $\rho < 1$. The average delay $D$ may be calculated by applying (7) with $\Delta = 2^d - 1$ and with $\frac{\lambda}{d}$ instead of $\lambda$. Thus, it follows that

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

Again, for $\rho \approx 0$, we have $D \approx \frac{3}{2}(2^d - 1)$, which is unacceptably large.

### 4.3 The Non-Idling Versions of the Periodic Schemes

The periodic schemes presented in §4.2 may accommodate very high load factor values (without any stability problems). However, their delay performance under light traffic is not satisfactory. This is primarily owed to the extensive occurrence of idling, due to 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. In the present section, we consider the non-idling versions of the periodic schemes of §4.2; that is, we assume, in addition, that if a packet can traverse an arc earlier than supposed then it is allowed to do so. (Also, no null packets are introduced in the network.) The

schemes to be presented appear to perform very satisfactorily. Unfortunately, these schemes seem to be analytically intractable; thus, their performance is studied by means of approximations and simulations.

### 4.3.1 The Non-Idling Scheme Based on the Optimal Algorithm for the Multinode Broadcast

Under the present scheme, each packet generated by node $x$ is broadcast along that tree used by $x$ in the optimal multinode broadcast algorithm of [BOSTT89]. Thus, the scheme under consideration is basically of the simple form considered in §4.1; moreover, it belongs to the class of static schemes defined in §3.2. As already mentioned in §4.1, each arc of the hypercube is shared by at most $\lceil \frac{2^d-1}{d} \rceil$ of the trees used; therefore, during each slot, there are generated an average of at most $\lambda \lceil \frac{2^d-1}{d} \rceil$ packets that will eventually have to traverse any particular arc $(z, z \oplus e_j)$. Thus, the standard argument of calculating the average demand for transmissions over each arc does not impose any necessary condition for stability other than (12), which is a rather favorable one. Below, we show that (12) is also the sufficient condition for stability, when a simple distributed priority scheme is used.

We fix a node $x$; let $\tau_n$ be the time instant when the $n$th packet was generated by $x$, for $n = 1, \ldots$ We assume that the $n$th such packet is attached with a label, denoted by $l_n$; we have $l_n = l$ if the following is true: In the periodic version of the algorthim, the $n$th packet would have been broadcast by node $x$ in the $l$th period, that is during the multinode broadcast starting at time $(l-1)\lceil \frac{2^d-1}{d} \rceil + 1$. It is easily seen that these labels may be updated in a recursive way, according to the following simple rule:

$$l_{n+1} = \max\left\{ l_n + 1, \left\lceil \frac{\tau_n}{\lceil (2^d - 1)/d \rceil} \right\rceil + 1 \right\},$$

and

$$l_1 = \left\lceil \frac{\tau_1}{\lceil (2^d - 1)/d \rceil} \right\rceil + 1 .$$

Having defined these labels, we consider the following rule for contention resolution: If two or more packets wish to traverse the same arc at the same time, then priority is given to the packet having the smallest label; if there are more than one such packets, then priority is given to the one (with minimum label) that would had crossed this arc the first during the multinode broadcast algorithm of [BOSTT89]. Using this priority discipline, the order of the various packet transmissions is preserved when the periodic routing scheme is converted to a non-idling 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; therefore, inequality (12) is a sufficient condition for the stability of the non-idling scheme as well.

As far as the average delay $D$ is concerned, it is conjectured that, under light traffic, we have $D \approx d + \frac{1}{2} + \Theta(d)\rho$. The claim that $D \approx d + \frac{1}{2}$ for $\rho \approx 0$ is very plausible, because the trees used for

the various broadcasts have depth $d$. (Recall that the term $\frac{1}{2}$ equals the average synchronization time.) As far as the first order term is concerned, we believe that it is $\Theta(d)\rho$, because each packet faces additional contention for each arc it attempts to cross. Given the complex structure of the trees used in the scheme under consideration, we believe that a good estimate for $D$ is very hard to derive formally. For this reason, simulation was our best recourse; the experimental results obtained are presented in §4.3.3. In fact, as is discussed therein, the aforementioned conjectures for the delay are confirmed by the simulation outcomes to a satisfactory extent.

### 4.3.2 The Non-Idling Scheme Based on the Optimal Algorithm for the $d$ Simultaneous Multinode Broadcasts — An Approximate Delay Analysis

In the present subsection, we consider the non-idling version of the routing scheme presented in §4.2.2; this scheme is as follows: Each packet chooses randomly one of the $d$ completely unbalanced trees permissible (see §4.2.2) and traverses the corresponding arcs as soon as possible. Clearly, this scheme belongs to the class of static schemes defined in §3.2; notice, however, that this would have not been the case if, for some node $z$, packets generated by $z$ were assigned one of the $d$ permissible trees in a round-robin mode. By introducing a priority scheme similar to that described in §4.3.1, it is easily seen that the present scheme is stable for all $\rho < 1$; recall that this is the broadest possible range for stability.

As far as the delay $D$ is concerned, the same conjectures apply as for the scheme of §4.3.1. Again, we believe that a good estimate for $D$ is very hard to derive. Thus, we have performed several simulation runs for the present scheme as well; these we discuss in §4.3.3. In the rest of this subsection, we present an approximate model for estimating $D$; as will be seen in §4.3.3, this model is in excellent agreement with the simulation outcomes, provided that $\rho$ 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 $k$th slot, including the packet to be transmitted. Moreover, let $B_k$ be the number of packets generated by node $x$ during the $k$th slot; since arc $(x, x \oplus e_i)$ belongs to all of the $d$ trees that may 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 $k$th slot) through arc $(x \oplus e_m, x)$ and wishing to traverse arc $(x, x \oplus e_i)$. Notice that $P_k^{(m,i)} \in \{0, 1\}$, for $m = 1, \ldots, d$. Clearly, we have

$$Y_{k+1}^{(i)} = [Y_k^{(i)} - 1]^+ + B_k + \sum_{m=1}^{d} P_k^{(m,i)}, \quad \text{for } k = 1, \ldots ; \tag{13}$$

recall that $[\alpha]^+ = \max\{\alpha, 0\}$.

Next we introduce the following approximating assumptions:

**Assumption A:** The sequence $P_1^{(m,i)}, P_2^{(m,i)}, \ldots$ of random variables is taken to be a renewal Bernoulli process. Moreover, it is assumed that $E[P_k^{(m,i)}] = \rho_m g_{m,i}$, where $\rho_m$ is the probability

that a packet crosses arc $(x \oplus e_m, x)$ and $g_{m,i}$ is the probability that a packet has to cross arc $(x, x \oplus e_i)$ given that it has crossed arc $(x \oplus e_m, x)$. Notice that the parameters $(\rho_m)_{m=1,\ldots,d}$ are taken to be independent of the node $x$; this makes perfect intuitive sense, because of symmetry among the nodes. For the same reason, the parameters $(g_{m,i})_{(m,i)\in\{1,\ldots,d\}^2}$ are indeed independent of the node $x$; see also Appendix A.

**Assumption B:** The processes $(P_k^{(m,i)})_{k=1,\ldots}$ and $(P_k^{(l,i)})_{k=1,\ldots}$ are taken to be mutually independent for $l \neq m$.

These two approximating assumptions are of similar spirit as in other approximate models that have been used in the literature (e.g., see [GrH89]); note, however, that those models have been used in analyzing considerably different problems (see also §1).

In the analysis to follow, all the "equalities" to be derived are approximate (unless otherwise specified), since they are based on the two assumptions above.

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

$$A_k^{(i)} \stackrel{\text{def}}{=} B_k + \sum_{m=1}^{d} P_k^{(m,i)} \,. \tag{14}$$

Notice that $(B_k)_{k=1,\ldots}$ is (actually) a renewal process and is independent of the Bernoulli processes $(P_k^{(1,i)})_{k=1,\ldots}, \ldots, (P_k^{(d,i)})_{k=1,\ldots}$. Thus, under our approximation, $(A_k^{(i)})_{k=1,\ldots}$ is taken as a renewal process that assumes the distribution of a random variable $A^{(i)}$. Recalling also that $B_k$ is a Poisson random variable with mean $\lambda$, we have

$$E[A^{(i)}] = \lambda + \sum_{m=1}^{d} \rho_m g_{m,i} \,, \tag{15a}$$

and

$$\mathrm{var}[A^{(i)}] = \lambda + \sum_{m=1}^{d} \rho_m g_{m,i}(1 - \rho_m g_{m,i}) \,. \tag{15b}$$

Combining (14) with (13), we obtain

$$Y_{k+1}^{(i)} = [Y_k^{(i)} - 1]^+ + A_k^{(i)} \,, \quad \text{for } k = 1, \ldots \,. \tag{16}$$

Using the aforementioned approximations for the arrival process $(A_k^{(i)})_{k=1,\ldots}$, it follows from (16) that $(Y_k^{(i)})_{k=1,\ldots}$ may be approximated by the process of the number of customers in a discrete-time $G/D/1$ queue with unit service time. [Recall the definition of this queueing system in §2.1.1, and compare (1) with (16).] Using (3) in §2.1.1, it follows that

$$\Pr[Y_k^{(i)} \neq 0] = E[A^{(i)}] \,. \tag{17}$$

By definition of the parameter $\rho_i$, we have

$$\Pr[Y_k^{(i)} \neq 0] = \rho_i \,;$$

this together with (15a) and (17) imply that

$$E[A^{(i)}] = \rho_i = \lambda + \sum_{m=1}^{d} \rho_m g_{m,i} \,, \quad \text{for } i = 1, \ldots, d \,. \tag{18}$$

Recall now that $g_{m,i}$ denotes the probability that a packet has to cross arc $(x, x \oplus e_i)$ given that it has crossed arc $(x \oplus e_m, x)$. It is established in Appendix A that

$$g_{m,i} = \frac{2^{[(m-i) \bmod d]} - 1}{2^d - 1} \,, \; \forall (m,i) \in \{1, \ldots, d\}^2 \,; \tag{19}$$

note that this is an actual equality, not an approximation. Using this, we have

$$\begin{aligned} \sum_{m=1}^{d} g_{m,i} &= \sum_{m=1}^{d} \frac{2^{[(m-i) \bmod d]} - 1}{2^d - 1} \\ &= \frac{1}{2^d - 1} \sum_{m=1}^{d} 2^{[(m-i) \bmod d]} - \frac{d}{2^d - 1} \\ &= \frac{1}{2^d - 1} \sum_{m=1}^{d} 2^{m-1} - \frac{d}{2^d - 1} \\ &= 1 - \frac{d}{2^d - 1} \,. \end{aligned} \tag{20}$$

This together with (18) implies that

$$\rho_1 = \cdots = \rho_d = \lambda \frac{2^d - 1}{d} = \rho \,, \tag{21}$$

where we have also used (8). The above solution for the parameters $\rho_1, \ldots, \rho_d$ makes perfect intuitive sense; indeed, since all $d$ dimensions are treated symmetrically by the algorithm, the average number of transmissions per arc and slot is actually equal to the load factor $\rho$. What does not actually hold true is that the departure process of each arc is of the renewal type.

Next, we derive an expression for the average delay $D^{(i)}$ corresponding to the queue formed by the packets waiting to cross arc $(x, x \oplus e_i)$; as usual, $D^{(i)}$ includes the transmission time over that arc. Using (4) in §2.1.1, we have

$$\begin{aligned} D^{(i)} &= 1 + \frac{E[A^{(i)}(A^{(i)} - 1)]}{2E[A^{(i)}](1 - E[A^{(i)}])} \\ &= 1 + \frac{\mathrm{var}[A^{(i)}] + (E[A^{(i)}])^2 - E[A^{(i)}]}{2E[A^{(i)}](1 - E[A^{(i)}])} \\ &= \frac{1}{2} + \frac{\mathrm{var}[A^{(i)}]}{2E[A^{(i)}](1 - E[A^{(i)}])} \,. \end{aligned} \tag{22}$$

Furthermore, combining (18) with (21), we obtain

$$E[A^{(i)}] = \rho \,.$$

This together with (15b), (21) and (22) gives

$$D^{(i)} = \frac{1}{2} + \frac{1}{2\rho(1-\rho)}\Big[\lambda + \sum_{m=1}^{d} \rho g_{m,i}(1-\rho g_{m,i})\Big]$$
$$= \frac{1}{2} + \frac{1}{2(1-\rho)}\Big[\frac{\lambda}{\rho} + \sum_{m=1}^{d} g_{m,i}(1-\rho g_{m,i})\Big]$$
$$= \frac{1}{2} + \frac{1}{2(1-\rho)}\Big(\frac{d}{2^d-1} + \sum_{m=1}^{d} g_{m,i} - \rho\sum_{m=1}^{d} g_{m,i}^2\Big)\,, \qquad (23)$$

where we have also used (8). Using (19) and reasoning similarly as in proving (20), it follows (after some algebra) that

$$\sum_{m=1}^{d} g_{m,i}^2 = \frac{1}{(2^d-1)^2}\big[d + \frac{1}{3}(4^d-1) - 2(2^d-1)\big]\,.$$

Combining this with (20) and (23), we obtain (after cancelling two terms)

$$D^{(i)} = \frac{1}{2} + \frac{1}{2(1-\rho)}\Big(1 - \frac{\rho}{(2^d-1)^2}\big[d + \frac{1}{3}(4^d-1) - 2(2^d-1)\big]\Big)\,, \quad \text{for } i = 1,\ldots,d\,. \qquad (24)$$

Note that the above expression is independent of $i$, which is due to complete symmetry among the $d$ hypercube dimensions.

So far, we have derived an approximate expression for the average delay suffered by a packet while waiting to cross an arc of the $i$th 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 (24) and taking also the average synchronization time into account, we obtain the following approximate formula for the delay:

$$D \approx \frac{d}{2} + \frac{d}{2(1-\rho)}\Big(1 - \frac{\rho}{(2^d-1)^2}\big[d + \frac{1}{3}(4^d-1) - 2(2^d-1)\big]\Big) + \frac{1}{2}\,. \qquad (25)$$

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 $\rho$, we have

$$D \approx d + \frac{1}{2} + d\frac{\rho}{3}\,. \qquad (26)$$

As will be seen in the §4.3.3, the approximate formula (25) is in excellent agreement with the simulation outcomes, for $\rho \leq 0.3$. This together with (26) supports the conjecture that, for small $\rho$, there holds $D \approx d + \frac{1}{2} + \Theta(d)\rho$.

### 4.3.3 Numerical Performance Evaluation of the Schemes

In this subsection, we present both numerical and experimental results concerning the delay induced by the two schemes discussed in §§4.3.1 and 4.3.2.

First, we investigate the accuracy of the approximate model of §4.3.2 for estimating the delay $D$ induced by the routing scheme considered therein (namely, the non-idling version of the scheme based on the optimal algorithm of [SaS85] for the $d$ simultaneous multinode broadcasts). In Table 1, we compare the simulation outcomes for the 8-dimensional hypercube with the estimate given by (25) for $d = 8$; each of the simulation outcomes has been obtained over a period of 5,000 slots. Clearly, there is excellent agreement between the experimental results and the corresponding approximate estimate for $D$, for values of $\rho$ ranging from 0.05 to 0.3; the corresponding relative error is less than 2% (in absolute value) for all such values of $\rho$. In particular, 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 (25) deteriorates gradually for $\rho > 0.3$; however, as was explained in §3.3, it is expected that $\rho$ does not exceed this value in practical applications. In Table 2, we investigate the accuracy of the approximate formula (25) for different values of the hypercube dimension $d$; again, excellent agreement is observed for all $d = 5, \ldots, 10$ under moderately light traffic (namely, for $\rho = 0.1$, $\rho = 0.15$ and $\rho = 0.2$). Note that the experimental results of Table 2 as well as those shown in the figures to follow have been obtained over periods of 1,000 slots.

As already mentioned in §§4.3.1 and 4.3.2, it is expected (for both schemes) that the first order term in $D$ is $\Theta(d)\rho$. In order to examine the validity of this conjecture, we have plotted the measure $D - d - \frac{1}{2}$ as a function of $d$, for various values of $\rho$. (Recall that $d + \frac{1}{2}$ is the conjectured value for the zero order term for $D$, under both schemes; this has not been proved here, but it is certainly true.) In particular, in Fig. 2, we present these plots for the scheme of §4.3.1. Apparently, for all three values of $\rho$, $D - d - \frac{1}{2}$ is an increasing function of $d$; in fact, it seems that, for fixed $\rho$, $D - d - \frac{1}{2}$ grows linearly in $d$; note however, that the points corresponding to the smaller values of $d$ do not match very well to a linear pattern. Similar conclusions are drawn from observing Fig. 3, which corresponds to the scheme of §4.3.2. (Note that the vertical axis of this figure is in a different scale than that of Fig. 2.) In fact, the conjecture that $D - d - \frac{1}{2}$ grows linearly in $d$ (for fixed $\rho$) is supported rather strongly by these plots. This conjecture is further supported by the approximate model developed in §4.3.2; recall the simplified approximate formula (26), and the fact that this model has proved to be very accurate for $\rho \leq 0.25$.

Next, we study the behavior of the steady-state average queue-size $Q$ per node, defined as $\frac{1}{2^d}$ of the stationary average total number of packets stored within the entire network (per slot) in steady-state (*). Each 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

(*) Under both of the schemes discussed in this section, all nodes behave symmetrically; thus, this

| $d = 8$ | | | |
|---|---|---|---|
| $\rho$ | **Simulation** | **Approximation** | **Relative Error** |
| 0.025 | 8.5581 | 8.5689 | 0.13% |
| 0.050 | 8.6084 | 8.6414 | 0.38% |
| 0.075 | 8.6937 | 8.7179 | 0.28% |
| 0.100 | 8.7554 | 8.7986 | 0.49% |
| 0.125 | 8.8544 | 8.8839 | 0.33% |
| 0.150 | 8.9556 | 8.9742 | 0.21% |
| 0.175 | 9.0642 | 9.0699 | 0.06% |
| 0.200 | 9.1945 | 9.1718 | -0.25% |
| 0.225 | 9.3045 | 9.2801 | -0.26% |
| 0.250 | 9.4417 | 9.3957 | -0.49% |
| 0.275 | 9.6211 | 9.5192 | -1.06% |
| 0.300 | 9.7944 | 9.6515 | -1.46% |
| 0.325 | 10.0516 | 9.7938 | -2.56% |
| 0.350 | 10.2045 | 9.9469 | -2.52% |
| 0.375 | 10.4875 | 10.1123 | -3.58% |
| 0.400 | 10.7547 | 10.2914 | -4.31% |

Table 1

that a packet to be forwarded to several neighbors of $x$ is assumed to occupy one unit of buffer capacity, for obvious reasons. Fig. 4 corresponds to the scheme of §4.3.1, while Fig. 5 corresponds to the scheme of §4.3.2. These figures imply that, for fixed $\rho$, $Q$ is increasing (seemingly in a linear fashion) with $d$; this is rather reasonable, since (for fixed $\rho$) the delay $D$ was also seen to increase with $d$.

Next, we compare the performance of the two direct schemes. In particular, in Figs. 6 and 7, we compare the delay induced by the two schemes under consideration. Fig. 6 refers to the 10-dimensional hypercube, and illustrates that, for fixed $d$, the scheme of §4.3.1 induces the larger delay. Fig. 7 shows that, for fixed $\rho$, the delay $D$ grows more steeply in $d$ under the scheme of §4.3.1 rather than under the scheme of §4.3.2. Moreover, in Figs. 8 and 9, we compare the average queue-sizes of the two schemes. Fig. 8 refers to the 10-dimensional hypercube, and shows that, for fixed $d$, larger queues are built under the scheme of §4.3.1; a similar conclusion is drawn from Fig. 9, for the case where $\rho$ is fixed.

---

definition of $Q$ really gives the average queue-size per node. However, under schemes where nodes behave asymmetrically (such as the scheme of §5.1), the queue-size statistics may vary for different nodes; for such schemes, the definition of $Q$ provides us with an overall estimate of the various queue-sizes.

| $\rho = 0.10$ | | | |
|---|---|---|---|
| $d$ | **Simulation** | **Approximation** | **Relative Error** |
| 5 | 5.6589 | 5.6957 | 0.65% |
| 6 | 6.7045 | 6.7288 | 0.36% |
| 7 | 7.7289 | 7.7632 | 0.44% |
| 8 | 8.7245 | 8.7986 | 0.85% |
| 9 | 9.8063 | 9.8346 | 0.29% |
| 10 | 10.8190 | 10.8711 | 0.48% |
| $\rho = 0.15$ | | | |
| $d$ | **Simulation** | **Approximation** | **Relative Error** |
| 5 | 5.8003 | 5.8108 | 0.18% |
| 6 | 6.8436 | 6.8633 | 0.29% |
| 7 | 7.8807 | 7.9180 | 0.47% |
| 8 | 8.9326 | 8.9742 | 0.47% |
| 9 | 10.0432 | 10.0315 | -0.11% |
| 10 | 11.0907 | 11.0894 | -0.01% |
| $\rho = 0.20$ | | | |
| $d$ | **Simulation** | **Approximation** | **Relative Error** |
| 5 | 5.8936 | 5.9403 | 0.79% |
| 6 | 7.0012 | 7.0015 | 0.19% |
| 7 | 8.1025 | 8.0921 | -0.13% |
| 8 | 9.1771 | 9.1717 | -0.06% |
| 9 | 10.2267 | 10.2529 | 0.26% |
| 10 | 11.3788 | 11.3350 | -0.39% |

Table 2

The above discussion implies that the scheme of §4.3.2 outperforms that of §4.3.1. The former scheme is preferable for an additional reason, namely for its simplicity of implementation; indeed, under the scheme of §4.3.2, each packet may be forwarded correctly by the various intermediate nodes, provided that it only carries the identity of its origin and a tag denoting which order of crossing the hypercube dimensions it has adopted. On the contrary, the scheme based on the optimal multinode broadcast algorithm of [BOSTT89] is considerably more complicated to implement, because the trees used by this algorithm cannot be described in a concise way. It is an interesting open problem to generate a set of $2^d$ simple spanning trees (one rooted at each node) so that no arc is contained in more than $\Theta(\frac{2^d}{d})$ arcs; such a set of trees may possibly qualify for efficient direct routing of multiple broadcasts (see also §4.1).

It is also worth noting that an approximate model similar to that of §4.3.2 was developed for the scheme of §4.3.1. Again, there is complete symmetry among different nodes and among arcs

of the same dimension, because the trees used are all obtained by translating a prototype tree (see §2.2.1 and [BOSTT89]). Thus, the parameters $g_{m,i}$ are again independent of the particular node considered, and the model gives rise to a system of equations invoving the $d$ unknown parameters $\rho_1, \ldots, \rho_d$. Unfortunately, this model was not a successful one; its accuracy was satisfactory only for very small values of $\rho$. A potential explanation for this fact is as follows: By the construction of the trees used in the multinode broadcast algorithm of [BOSTT89], for fixed $i$, most of the parameters $g_{m,i}$ are 0, one of them is close to 1, and the remaining have small positive values; thus, packets do not "mix" very well, and the two approximating assumptions used by the model are far from reality.

## 5. AN INDIRECT ROUTING SCHEME BASED ON THE $d$ 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, this scheme can be stable only if $\lambda 2^d \leq 1$, or equivalently $\rho \leq \frac{1}{d}(1 - \frac{1}{2^d})$. Therefore, the maximum attainable value for the load factor is $\Theta(\frac{1}{d})$. The reason for this poor performance is that only a fraction $\frac{1}{d}$ 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 $T^{(1)}, \ldots, T^{(d)}$, with each tree receiving the same amount of traffic; then, the maximum load factor might possibly be $\Theta(1)$. A routing scheme of this spirit is presented in this section; the scheme is based on the set of $d$ disjoint spanning trees introduced by Johnsson and Ho [JoH89] (see also §5.1). 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, contrary to the schemes of §4.

It will be established rigorously that the routing scheme analyzed in this section is stable for all $\rho \leq \frac{2}{3}(1 - \frac{1}{2^d}) \approx \frac{2}{3}$, while it satisfies $D \approx 3d + 7 + \frac{9}{4}\rho$ for small $\rho$. Thus, this scheme meets the performance criteria set in §3.3. In addition to the analysis, we discuss potential methods for devising even better indirect routing schemes. Finally, in §5.6, we compare all routing schemes considered in the paper.

### 5.1 The $d$ Disjoint Spanning Trees

Johnsson and Ho [JoH89] have constructed an imbedding of $d$ disjoint spanning trees in the $d$-cube; they call them "$d$ Edge-Disjoint Spanning Binomial Trees" ($d$ESBT). This imbedding consists of $d$ completely unbalanced trees $T^{(1)}, \ldots, T^{(d)}$. 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: $(j \bmod d) + 1, [(j+1) \bmod d] + 1, \ldots, [(j+d-1) \bmod d] + 1$. In Fig. 10, we present this construction for $d = 3$; this figure is taken from

[JoH89].

### 5.2 The Rules of the Routing Scheme

The set of rules for routing the packets is as follows:

**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 $T^{(1)}, \ldots, T^{(d)}$; each of them is assigned an a priori probability $\frac{1}{d}$. 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 $e_1, \ldots, e_d$ also follow Rules A and B, as well as the rules 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 some 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 $C_0 \stackrel{\text{def}}{=} \{t \geq 0 : t \bmod 3 = 0\}$, $C_1 \stackrel{\text{def}}{=} \{t \geq 0 : t \bmod 3 = 1\}$ and $C_2 \stackrel{\text{def}}{=} \{t \geq 0 : t \bmod 3 = 2\}$. 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 $C_1 \cup C_2$. 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 $C_0$. Thus, every three slots, each of the $d$ trees is reversed, and all its arcs point towards its root. Slots in the set $C_0$ are actually used for sending packets to the respective root nodes, while slots in the set $C_1 \cup C_2$ are used for the broadcasts. Note that for $z^* = (0, \ldots, 0)$ the arcs of the form $(z^*, e_i)$ do not belong to any of the $d$ disjoint trees (see [JoH89]); these arcs are only used during slots in $C_0$.

**Rule D:** Consider some node $y$ at Hamming distance $k$ from root $e_j$. Each packet originating at $y$ that selects to be routed along $T^{(j)}$ must traverse $k$ hypercube arcs in order to reach $e_j$. However, we assume that, before such a packet considers to cross the first arc of its path to $e_j$, it has to traverse a tandem of $d-k$ virtual arcs, all located at node $y$ (see Fig. 11, for $d = 3$). Such arcs may be traversed only during slots in $C_0$. Thus, all "paths" leading to the root $e_j$ contain exactly $d$ arcs, both actual and virtual ones. It is assumed that packets to be routed along different trees have to cross different "paths" of virtual arcs, even if they have been generated at the same node. It may be easily seen that 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 $B_2$; each buffer has unit capacity.

Consider a packet that has selected tree $T^{(j)}$; let node $y$ be the origin of this packet. If $y$ belongs to the subtree $T_1^{(j)}$, then it will be placed in buffer $B_1$; however, this occurs after the packet traverses virtual arc $l_1$ (see Fig. 11). If $y$ belongs to any other subtree except for $T_1^{(j)}$ (or if $y = e_j$), then it will be placed in buffer $B_2$, after traversing virtual arc $l_2$ (see Fig. 11). Similarly as in Rule D, $l_1$ and $l_2$ may be traversed only during slots in $C_0$. (The reader should compare Fig. 11 with Fig. 10.)

**Rule F:** During slots $t+1, t+2$, with $t \in C_0$, root node $e_j$ broadcasts the packets that arrived in its buffers $B_1$ and $B_2$ at the end of slot $t$. 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 has to be 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.

It is straightforward that the indirect scheme under analysis belongs to the class of static schemes defined in §3.2.

Some of the above rules may seem somewhat unmotivated. Instead of commenting on them at this point, we proceed with the analysis of the routing scheme, which will reveal the raison d' être for these rules. More discussion on the proposed scheme is presented in §5.5. First, we present a result to be used in the analysis to follow.

### 5.3 A Useful Lemma

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 $s_1, \ldots, s_n$ of the paths and exit only at the common end $f$ (see Fig. 12a); 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. 12b). 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 $t = 0, \ldots$; both systems start operating at time $t = 0$. The lemma to be proved is as follows:

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

**Proof:** First, we consider the case where all paths have length 2 (see Fig. 13). 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

$$\tilde{F}(t) = F(t) \quad \text{and} \quad \tilde{M}(t) = M(t)\,, \quad \text{for } t = 0, \ldots \tag{27}$$

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 (27) for $t = 0$. Next, we assume that the induction hypothesis holds for all $t \in \{0, \ldots, t^*\}$; 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 (27) 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 $t = 0, \ldots,$ by assumption. Moreover, we have $\tilde{F}(t) = F(t)$ for $t = 0, \ldots, t^*$ (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. 13, the result is easily extended for the general case in Fig. 12. It suffices to progressively collapse the paths of the tree $T$, starting from the lowest level. **Q.E.D.**

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 holds if packet transmissions can only start at the beginning of slots numbered $0, \Delta, \ldots$ and each transmission lasts for $\Delta$ slots. On the other hand, Lemma 3 does not hold if some of the paths have different length than the others.

### 5.4 Performance Evaluation of the Scheme

First, we notice that packets that are 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. (Note that 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 topologically isomorphic and are treated by the routing rules in a symmetric way. For the rest of the analysis, we fix some $j \in \{1, \ldots, d\}$ and we analyze the queues formed by the packets routed along $T^{(j)}$. First, we derive the condition for stability of the scheme:

**Proposition 4:** The routing scheme introduced in §5.2 is stable if and only if

$$\rho < \frac{2}{3}(1 - \frac{1}{2^d}) . \tag{28}$$

■

**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 set of paths leading from the hypercube nodes to $e_j$. Consider now the set of all paths leading to buffer $B_1$. Because of Rule D, all them consist of $d+1$ arcs; moreover, because of Rule C, all transmissions in these paths take place every 3 slots. Hence, we may apply Lemma 3. (Recall the comments following the proof in §5.3.) Therefore, all paths leading to $B_1$, may be collapsed into a single path $P_1$ of length $d+1$. Subtree $T_1^{(j)}$ has $2^{d-1}$ nodes, and the arrival process in each of them is Poisson with rate $\frac{\lambda}{d}$. Thus, the combined arrival process for the path $P_1$ is Poisson with rate $\frac{\lambda}{d}2^{d-1}$. It follows from the discussion above that the path $P_1$ behaves like a discrete-time $M/D/1$ queue with synchronization, followed by a tandem of $d$ deterministic servers. Since all $d+1$ servers involved are identical, the entire tandem is stable if and only if the queue formed in the first server is so. Since transmissions within $P_1$ may start every 3 slots and "last" for that long, the queue formed in the aforementioned queue is stable if and only if $3\frac{\lambda}{d}2^{d-1} < 1$; by the definition of $\rho$ in (8), this condition is equivalent to (28).

The set of all paths leading to buffer $B_2$ may be collapsed in a similar way. The resulting single path is identical to that derived for $B_1$; again the process feeding this path is Poisson with rate $\frac{\lambda}{d}2^{d-1}$. (Recall that all nodes that do not belong to $T_1^{(j)}$ send their packets to buffer $B_2$; there are $2^d - 2^{d-1} = 2^{d-1}$ such nodes, including $e_j$.) Therefore, the set of paths leading to $B_2$ is stable if and only if (28) holds. Furthermore, by Rule F, each packet arriving at $B_1$ or $B_2$ is immediately broadcast; thus, it follows that (28) is the necessary and sufficient condition for the entire scheme to be stable. **Q.E.D.**

Next, we derive the expression for the average delay $D$.

**Proposition 5:** The average delay $D$ for the routing scheme introduced in §5.2 is given as follows:

$$D = \frac{9}{2}d + 4 + \frac{3\rho}{2[\frac{2}{3}(1 - \frac{1}{2^d}) - \rho]} .$$

■

**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 $e_j$ and be placed in the corresponding buffer, and $V$ is the average time

from the moment a packet enters this buffer until its broadcast is completed. As already mentioned, the process feeding each of the buffers $B_1$ and $B_2$ is equivalent to the output process of a tandem of a discrete-time $M/D/1$ queue with synchonization followed by $d$ deterministic servers. Moreover, it is an immediate consequeunce of Lemma 3 that the steady-state average time $R$ for a packet to reach buffer $B_1$ (and resp. $B_2$) remains the same after merging the paths leading to this buffer. Thus, the expression for $R$ can be computed by analyzing the delay induced by the tandem of a discrete-time $M/D/1$ queue with synchonization followed by $d$ deterministic servers. In particular, the average delay $W$ induced by the first queue of this tandem can be derived by applying (7) with $\Delta = 3$ and with $\frac{\lambda}{d}2^{d-1}$ instead of $\lambda$. Thus, we obtain

$$W = 3\Big(\frac{3}{2} + \frac{3\frac{\lambda}{d}2^{d-1}}{2(1-3\frac{\lambda}{d}2^{d-1})}\Big) = 3\Big(\frac{3}{2} + \frac{\rho}{2[\frac{2}{3}(1-\frac{1}{2^d})-\rho]}\Big)\,. \tag{29}$$

The additional delay induced by the rest of the deterministic servers equals $3d$, because no queueing takes place in these servers. Thus, using (29), we obtain

$$R = W + 3d = 3d + \frac{9}{2} + \frac{3\rho}{2[\frac{2}{3}(1-\frac{1}{2^d})-\rho]}\,. \tag{30}$$

Once a packet has arrived at $e_j$ 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 $\lceil\frac{d}{2}\rceil + d - 1$. If the broadcast starts at a slot in $C_2$, then it is completed in time $\lfloor\frac{d}{2}\rfloor + d$. Since both these events occur with probability $\frac{1}{2}$ (because of Rule F), it follows that

$$V = \frac{1}{2}\Big(\Big\lceil\frac{d}{2}\Big\rceil + d - 1 + \Big\lfloor\frac{d}{2}\Big\rfloor + d\Big) = \frac{3}{2}d - \frac{1}{2}\,.$$

This together with (30) and with the fact $D = R + V$ proves the result. **Q.E.D.**

It follows from Proposition 5 that, in the case of light traffic, we have $D \approx \frac{9}{2}d + 4 + \frac{9}{4}\rho$. Therefore, the delay induced by the scheme under analysis meets the criterion set in §3.3. In fact, by slightly modifying the scheme, the delay $D$ may be improved; see Corollary 8.

Next, we derive the expression for the average queue-size $Q$; this measure was defined in §4.3.3 as $\frac{1}{2^d}$ of the average total number of packets stored in the entire network. As already mentioned therein, this definition is basically tailored to the indirect scheme under analysis.

**Proposition 6:** There holds

$$Q = \frac{3d}{4}\rho\frac{2^d}{2^d-1} + 3\rho\frac{d}{2^d-1}\left(3d + \frac{9}{2} + \frac{3\rho}{2[\frac{2}{3}(1-\frac{1}{2^d})-\rho]}\right)\,. \qquad \blacksquare$$

**Proof:** This proof is similar to that of Proposition 5.

First, we consider only packets that have chosen a particular tree $T^{(j)}$. Let $N$ be the average total number of such packets stored in the network. By symmetry, $N$ does not depend on $j$; therefore, we have

$$Q = \frac{dN}{2^d} . \tag{31}$$

Clearly, we have

$$N = N_r + N_b , \tag{32}$$

where $N_r$ is the average total number of packets heading towards root $e_j$ and $N_b$ is the average total number of stored packets undergoing broadcast. Since successive broadcasts are pipelined, each non-leaf node of $T^{(j)}$ stores at most one of the packets undergoing broadcast along $T^{(j)}$. Each of the two buffers $B_1$ and $B_2$ is fed at a rate of $3\frac{\lambda}{d}2^{d-1}$ (see the proof of Proposition 4); since these two buffers are served alternately (because of Rule F), the stationary probability that a non-leaf node (including $e_j$) stores a packet undergoing broadcast equals $3\frac{\lambda}{d}2^{d-1}$. Since $T^{(j)}$ has $2^d - 2^{d-1} = 2^{d-1}$ non-leaf nodes (see §5.1), it follows that

$$N_b = 2^{d-1}\left(3\frac{\lambda}{d}2^{d-1}\right) = 3\rho\frac{2^{2d-2}}{2^d-1} . \tag{33}$$

Next, notice that, by Lemma 3, the total number of packets present in a tree of paths of equal length remains unchanged after merging the paths. Thus, we may derive the expression for $N_r$ by merging the paths leading to each of the buffers $B_1$ and $B_2$ in the way described in the proof of Proposition 4. In particular, applying Little's formula, it follows from (30) that

$$N_r = 2\left(3\frac{\lambda}{d}2^{d-1}\right)R = 3\rho\frac{2^d}{2^d-1}\left(3d+\frac{9}{2}+\frac{3\rho}{2[\frac{2}{3}(1-\frac{1}{2^d})-\rho]}\right) . \tag{34}$$

Combining (32) with (33) and (34), we obtain

$$N = 3\rho\frac{2^{2d-2}}{2^d-1} + 3\rho\frac{2^d}{2^d-1}\left(3d+\frac{9}{2}+\frac{3\rho}{2[\frac{2}{3}(1-\frac{1}{2^d})-\rho]}\right) .$$

This together with (31) proves the result. **Q.E.D.**

Proposition 6 implies that, for small $\rho$, we have $Q \approx \frac{3d}{4}\rho\frac{2^d}{2^d-1} + 3\rho\frac{d}{2^d-1}(3d+\frac{9}{2})$; when $d$ is not very small, this simplifies to $Q \approx \frac{3d}{4}\rho$.

### 5.5 Discussion on the Scheme and Further Related Results

In the present section, we further investigate the performance of the scheme under analysis. In particular, in §5.5.1, we examine which of the constituent rules are actually limiting its performance and to what extent. In §5.5.2, we introduce a slight modification to the scheme, thus attaining further improvement in the delay properties. In §§5.5.3 and 5.5.4, we discuss other potential methods for devising even better indirect schemes, thus proposing some directions for further research.

### 5.5.1 The Impact of the Various Rules

The routing scheme introduced in §5.2 has proved to be rather tractable analytically. Since the $d$ trees used for routing are disjoint, Rules A, B and C have resulted in the decoupling of packets that are routed along different trees. Thus, queueing analysis within each of the $d$ trees was done independently, which was rather convenient. The analysis was simplified even more, by introducing the virtual arcs in Rule D and applying Lemma 3. As will be seen later in this subsection, virtual arcs do not influence the stability properties of the scheme, because they do not create additional bottlenecks. On the other hand, virtual arcs result in additional delays; however, it will be argued that most of these delays may actually be avoided.

As far as Rule B is concerned, it is clear that it results in redundant transmissions over the (actual) hypercube arcs. 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 $e_j$. On the other hand, $2^d - 1$ transmissions would suffice, if the packet were to be broadcast by node $y$, using the arcs of $T^{(j)}$ and the reverse of them; again, only $k$ of the transmissions would be over the reverse of the arcs in $T^{(j)}$. Thus, Rule B induces an additional $\frac{d}{2}$ transmissions per packet, on the average. Moreover, because of Rule C, one third of the time slots are devoted to performing transmissions towards the roots (which transmissions are far fewer). The above arguments seem 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 7:** Any routing scheme that conforms to Rule A is stable only if

$$\rho \leq \frac{2}{3}\left(1 - \frac{1/3}{2^d - 2/3}\right) . \tag{35}$$

■

**Proof:** We fix some $j \in \{1, \ldots, d-1\}$. 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 $j$th dimension.

In the context of $T^{(j)}$, subtree $T_1^{(j)}$ is hanging from node $e_j \oplus e_{j+1}$; this follows easily from the discussion in §§2.2.2 and 5.1. 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 $\frac{\lambda}{d}2^{d-1}$ 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 $\frac{\lambda}{d}(2^d - 1)$ transmissions over arc $(e_j \oplus e_{j+1}, e_j)$ per slot.

Clearly, some routing scheme may be stable only if the average total demand per slot for transmissions over any fixed arc does not exceed 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 is stable only if

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

recalling the definition of $\rho$ in (8), it follows that the inequality above is equivalent to (35).

**Q.E.D.**

It may be seen that the right-hand quantity in (28) is very close to that in (35), even for moderately small values of $d$; e.g., for $d = 8$, their relative difference equals 0.26%, and for $d = 10$, it equals 0.065%.

The proof of Proposition 7 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. At this point, it may be seen that virtual arcs $l_1$ and $l_2$ do not influence the stability properties of the scheme. Indeed, the true bottleck of the set of paths leading to $B_1$ is arc $(e_j \oplus e_{j+1}, e_j)$ rather than virtual arc $l_1$, which is free of contention; moreover, the existence of virtual arc $l_2$ does not result in further restriction on the condition for stability. Rule E was basically introduced for illustrative purposes. In particular, this rule makes even more clear that, under heavy load, the scheme reduces to round-robin. Indeed, it may be seen that, for $\rho \uparrow \frac{2}{3}(1 - \frac{1}{2^d})$, root $e_j$ is alternately broadcasting one packet originating at some node in $T_1^{(j)}$ and one packet originating elsewhere. This is intuitively clear, because these two classes of packets represent equal average input rate.

Finally, it is obvious that the virtual arcs introduced in each of the nodes (by Rule D) do not influence the stability of the system at all.

### 5.5.2 Improving the Delay Properties of the Scheme

We have already seen that introduction of virtual arcs has considerably simplified the analysis, without creating any stability problems. However, virtual arcs result in additional delays. Indeed, consider some packet generated at a node $y$, whose Hamming distance from root $e_j$ equals $k$. If this packet selects to be routed along $T^{(j)}$, then, before it traverses any hypercube arc, it has to traverse $d - k$ virtual arcs located at node $y$. Since such arcs may be traversed only during slots in $C_0$, it follows that they induce an average delay of $\frac{3}{2}d$ time units per packet. Even though application of Lemma 3 would not have been possible without the existence of these virtual arcs, we can actually dispense with most of them, without any further complication in the delay analysis. To see how this may be done, notice the following: Consider a single path, such as the one presented

in Fig. 12b; assuming that the arrival process feeding this path is Poisson, it is clear that the steady-state statistics of the corresponding departure process do not depend on the length of the path (provided that the path is non-trivial). Therefore, the steady-state statistics of the processes feeding the actual arcs will not change if Rule D is modified as follows:

**Rule D′:** Consider some node $y$ at Hamming distance $k \neq d$ for root $e_j$. Each packet originating at $y$ that selects to be routed along $T^{(j)}$ has to traverse one virtual arc, prior to taking any steps in the hypercube; see Fig. 14 and compare it with Fig. 11. Packets to be routed along different trees have to cross different virtual arcs, even if they have been generated at the same node.

Of course, Lemma 3 is not applicable under Rule D′. However, as was argued above, this lemma is still applicable for the stationary analysis, in which we are actually interested. On the contrary, under Rule D, Lemma 3 was applicable for both transient and steady states.

Next, we present the new expression for the average delay $D$.

**Corollary 8:** Suppose that the scheme introduced in §5.2 is modified, by using Rule D′ instead of Rule D. Then the average delay $D$ is given as follows:

$$D = 3d + 7 - \frac{3}{2^d} + \frac{3\rho}{2[\frac{2}{3}(1 - \frac{1}{2^d}) - \rho]} \,. \qquad \blacksquare$$

**Proof:** We notice that a packet will only have to traverse one virtual arc prior to entering the hypercube, unless it is generated at the node that is at distance $d$ from the root $e_j$ selected by the packet. If this is the case, then the packet will not traverse any virtual arc at this point. Because of symmetry among the $2^d$ nodes, this event occurs with probability $\frac{1}{2^d}$. (Recall that for each node $z$ there exists a unique node $y$ at Hamming distance $d$ from $z$.) Thus, the average delay induced by the virtual arcs located at the various hypercube nodes is now $3(1 - \frac{1}{2^d})$; under Rule D, the corresponding value was $\frac{3}{2}d$. Therefore, the average delay $D$ is now equal to that derived in Proposition 5, reduced by $3(\frac{d}{2} - 1 + \frac{1}{2^d})$; this immediately proves the result. **Q.E.D.**

In the case of light traffic, the average delay $D$ is now reduced to $D \approx 3d + 7 + \frac{9}{4}\rho$.

Note that the average queue-size $Q$ per node is also reduced after modifying the scheme. We refrain from presenting the new expression for $Q$, because the corresponding dominant terms are the same as in the formula derived in Proposition 6.

### 5.5.3 Potential Methods for Further Improving the Scheme

We have already seen that the stability properties of the scheme under analysis are rather satisfactory; unfortunately, they are not the best possible, since the scheme becomes unstable for $\rho \uparrow \frac{2}{3}(1 - \frac{1}{2^d})$. It is rather unlikely that the load factor $\rho$ would ever approach this stability limit in a practical application (see §3.3); nevertheless, some further improvement of this upper limit may be desirable, since it would also result in even better delay properties. In this subsection, we discuss some modifications in the rules of §5.2 that could possibly result in some improvement of the throughput properties of the scheme.

We shall be using the terminology input load of a subtree $T_i^{(j)}$ to denote the average number of packets that are generated at nodes of $T_i^{(j)}$ and select to be routed along $T^{(j)}$ during one slot. Under Rule A, the input load of $T_i^{(j)}$ equals $\frac{\lambda}{d}2^{d-i}$, for $i = 1, \ldots, d$, because $T_i^{(j)}$ contains $2^{d-i}$ nodes. Thus, we notice that there are large discrepancies in the input load received by the various subtrees. This suggests the following potential improvement on the scheme: If it were possible for all subtrees $T_1^{(j)}, \ldots, T_d^{(j)}$ to receive the same (or approximately the same) input load (for $j = 1, \ldots, d$), then the throughput properties of the scheme would improve significantly. (Of course, this could occur only after modifying Rule A in such a way that each node $y$ is not confined to split the packets it generates evenly among the $d$ trees $T^{(1)}, \ldots, T^{(d)}$.) In particular, if such a balancing of the input load were possible, then by serving the $d$ subtrees of each tree and the root in a round-robin mode, the scheme would be stable for values of the load factor close to $\frac{d+1}{d+2}$. Unfortunately, this is not possible [Sta91]; the underlying idea for proving this is that there exist too many small subtrees.

Even though nearly perfect load balancing is not possible, it is conceivable that some improvement on the throughput properties of the scheme could be attained by a different kind of balancing. In particular, suppose that it is possible to partition the various subtrees of each tree (including the root) in $L$ classes and balance the total input load per class. Then, by serving these classes in a round-robin mode (similarly as in Rule E), the scheme would be stable for values of the load factor close to $\frac{L}{L+1}$. Note that perfect balancing corresponds to $L = d + 1$, but this has already been proved to be impossible. On the other hand, the scheme under analysis balances the load between $L = 2$ classes. (One class consists of subtree $T_1^{(j)}$ and the other consists of the remaining subtrees together with the root $e_j$.) It is an open problem whether or not such a load balancing is possible with a larger number $L$ of classes. Some negative results proved in [Sta91] suggest that this open problem is rather hard. Moreover, because of the inherent "unbalancedness" of the $d$ Disjoint Spanning Trees, we do not expect that an upper stability limit of $1 - o(1)$ is attainable. In the next subsection, we discuss other indirect routing schemes, which may possibly perform even better than the one analyzed so far.

### 5.5.4 Other Possible Indirect Routing Schemes

As already argued in §5.5.3, there is not much opportunity to improve the (rather satisfactory) perfomance of the scheme of §5.2. In the present subsection, we discuss some other indirect schemes, based on imbeddings of subgraphs (in the $d$-cube) other than the $d$ Disjoint Spanning Trees of [JoH89]; whether or not these imbeddings are feasible is still an open question to us. Note that all the trees to be considered are assumed to have depth $\Theta(d)$, for obvious reasons.

As already discussed in §5.5.3, the scheme of §5.2 may possibly be improved by balancing the input load received by the various subtrees. Another approach to this problem is to imbed $d$ balanced disjoint spanning trees. (A spanning tree is characterized as "balanced", if it has $d$ subtrees of approximately the same size.) If such an imbedding is possible, then Rule A would remain as is, and the input load received by the various subtrees would automatically be balanced.

In general, imbedding $d$ balanced disjoint trees would result in various other improvements, such as increased fault tolerance etc. Unfortunately, we expect this to be a rather hard problem, since the imbedding of balanced spanning trees in the hypercube has proved to be rather complicated (see [BOSTT89] and [JoH89]).

Next, we present another indirect scheme, based on an idea by Leighton [Lei90]. Consider an imbedding of $d$ spanning disjoint trees with the following properties: All $d$ trees are rooted at node $z^* = (0, \ldots, 0)$; $d-1$ of the trees are pointing away from $z^*$, while the last one is balanced and is pointing towards $z^*$; the latter tree will be referred to as the "reverse" tree. By using such an imbedding, multiple broadcasts may be routed as follows: Packets are sent to node $z^*$ through the reverse tree; node $z^*$ broadcasts each of the packets it receives, along one of the $d-1$ trees emanating from itself. Since the reverse tree is balanced, such a scheme is expected to be stable for all $\rho \leq 1 - o(1)$. To see why the performance of this scheme would be rather satisfactory, consider the extreme case where the reverse tree has only $d-1$ subtrees, of approximately equal size. Then, by assigning one of the $d-1$ trees emanating from $z^*$ to each of the reverse subtrees, packets would suffer no additional queueing delay in node $z^*$. Thus, a packet already received by $z^*$ would be broadcast in $\Theta(d)$ time units. Moreover, for small $\rho$, the average time for a packet to reach node $z^*$ would be $\Theta(d) + \Theta(1)\rho$. (This may be seen by reasoning similarly as in the proof of Proposition 5.) Thus, the delay $D$ induced by such a scheme would be $\Theta(d) + \Theta(1)\rho$; whether or not this improves on the delay the scheme of §5.2 would depend on the values of the constants involved.

The basic merit of the routing scheme described above is that it does not require a periodic alternation of the directions of the arcs (such as that imposed by Rule C of §5.2). However, the performance of such a scheme would depend heavily on the degree of "balancedness" of the reverse tree. In particular, for the upper limit of the stability region to be $1 - o(1)$, the subtrees of the reverse tree should contain no more than $\frac{2^d}{d}(1 + o(1))$ nodes. Of course, whether or not such an imbedding of $d$ disjoint trees is feasible is an open and seemingly challenging question.

Finally, it is worth mentioning that if any of the above schemes (or an improved version of the scheme of §5.2) is stable for $\rho = 1 - o(1)$, then the corresponding delay will asymptotically (for $d \to \infty$) be $\Theta(d + \frac{\rho}{1-\rho})$. (The reasoning for this is similar to the proof of Proposition 5.) Thus, such a scheme would be optimal (with respect to $D$) over the class of static schemes (see §3.2), because it would attain the lower bound of Proposition 2.

### 5.6 A Brief Comparison of the Various Routing Schemes

In the present section, we briefly compare the efficient direct schemes of §4 (namely, the two schemes of §§4.3.1 and 4.3.2) with the indirect scheme analyzed in §5. Since all three efficient schemes exhibit very satisfactory stability properties, we shall focus on their delay properties and on the sizes of the queues involved.

As argued in §4.3, the two schemes presented therein appear to satisfy $D \approx d + \frac{1}{2} + \Theta(d)\rho$ when $\rho$

is small. On the other hand, the indirect scheme of §5 satisfies, under light traffic, $D \approx 3d + 7 + \frac{9}{4}\rho$; this scheme is outperformed by the two direct schemes (as far as delay is concerned), because its respective delay for $\rho \approx 0$ is considerably larger. In fact, any efficient indirect scheme would satisfy $\lim_{\rho \to 0} D = Kd$ with $K > 1$ (and $K$ constant); this is owed to the fact that, under such schemes, packets do not travel through shortest paths. Notice now that the indirect scheme of §5 exhibits idling, due to Rule C of §5.2; this would also be true for any indirect scheme involving periodic alternation of the directions of the arcs. If this restriction is relaxed, then the zero order term in $D$ would certainly decrease; however, it would still be of the form $K'd$ with $K' > 1$. For example, after relaxing Rule C of §5.2, it is seen that the delay induced by the indirect scheme satisfies $\lim_{\rho \to 0} D = \frac{3d}{2}$. Thus, eliminating idling from an indirect scheme may be rather advantageous for sufficiently small values of $\rho$.

We have performed several simulation runs for the non-idling version of the scheme of §5; in fact, the simulated scheme did not involve any redundant transmissions (see also §5.5.1). Somewhat surprisingly, it appeared that, for fixed $\rho$, the first order term in the delay $D$ for this scheme increases with $d$; see Fig. 15. Despite the decrease in the delay $D$ attained by avoiding idling, the direct schemes of §4.3 are still preferable; see Fig. 16. As for the average queue-size $Q$ per node, under the indirect scheme of §5 there holds $Q \approx \Theta(d\rho)$, for small $\rho$; the same conclusion appears to hold for the non-idling version of the scheme, as shown in Fig. 17.

Next, we compare the values of the measure $Q$ for the various schemes. As revealed by Fig. 18 and 19, the non-idling version of the indirect scheme of §5.2 outperforms the efficient direct schemes with respect to this criterion. It is particularly important that, for fixed $\rho$, the queue-size $Q$ grows more slowly with $d$ under the non-idling version of the 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. 18 and 19. Again, the non-idling version of the indirect scheme is superior to the direct ones; see Fig. 20 and 21.

The conclusion drawn from the previous discussion is that efficient indirect schemes may be preferable to the direct ones in practice, provided that traffic is not very light. Moreover, as was argued in §5.5.4, there is a potential for devising static indirect schemes that would asymptotically (for $d \to \infty$) meet the lower bound $\Omega(d + \frac{\rho}{1-\rho})$ on the average delay $D$ . Finally, there is one more point that would allow one to advocate for the indirect schemes; that is, such schemes seem to result in a smaller variance of the delay. This claim is based on the fact that indirect schemes make more effective use of pipelining than direct schemes do; unfortunately, such a claim seems rather hard to establish.

## 6. 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 $\lambda$. 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 $\rho \stackrel{\text{def}}{=} \lambda \frac{2^d - 1}{d}$ 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 (without any stability problems), while only introducing a delay of $\Theta(d) + O(d)\rho$ slots per packet under light trafiic.

We have devised and analyzed several routing schemes that meet the aforementioned performance criteria. We considered two classes of schemes, namely direct and indirect ones. Under direct schemes, packets are broadcast directly by the respective origins; on 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 [JoH89], was shown to be stable for all $\rho < \frac{2}{3}(1 - \frac{1}{2^d})$; the corresponding average delay per packet was proved to be $D \approx 3d + 7 + \frac{9}{4}\rho$ for small values of $\rho$; both of these results were established rigorously. Furthermore, a direct routing scheme, using a non-idling version of the optimal algorithm for $d$ simultaneous multinode broadcasts by Saad and Schultz [SaS85], was shown to be stable for $\rho < 1$. For this scheme, it was conjectured that $D \approx d + \frac{1}{2} + \Theta(d)\rho$; 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. Of similar properties is a direct scheme using a non-idling version of the multinode broadcast algorithm by Bertsekas et al. [BOSTT89]; however, this scheme is more complicated to implement than the other ones. 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; in fact, indirect schemes appear to require smaller buffer capacity than the direct ones. Finally, we have presented several open problems that are related to methods of potential improvement of the schemes analyzed. Even though it was assumed that the randomly generated broadcasts were the only transmissions taking place in the network, all of the efficient schemes analyzed would also perform rather satisfactorily in the presence of a few other packet transmissions.

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

## REFERENCES


[AbP86] S. Abraham and K. Padmanabhan, "Performance of the Direct Binary $n$-Cube Network for Multiprocessors", *Proceedings of the 1986 International Conference on Parallel Processing.*

[Ale82] R. Aleliunas, "Randomized Parallel Communication", *Proceedings of the 1st ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pp. 60-72.

[BeT89] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall.

[BOSTT89] D.P. Bertsekas, C. Ozveren, G.D. Stamoulis, P. Tseng, and J.N. Tsitsiklis, "Optimal Communication Algorithms for Hypercubes", Report LIDS-P-1847, Laboratory for Information and Decision Systems, M.I.T.

[Bru71] S.L. Brumelle, "Some Inequalities for Parallel-Server Queues", *Operations Research*, vol. 19, pp. 402-413.

[GrG86] A.G. Greenberg and J. Goodman, "Sharp Approximate Models of Adaptive Routing in Mesh Networks", preprint.

[GrH89] A.G. Greenberg and B. Hajek, "Deflection Routing in Hypercube Networks", preprint.

[HaC87] B. Hajek and R.L. Cruz, "Delay and Routing in Interconnection Networks", In A.R. Odoni, L. Bianco, and G. Szago (Eds.), *Flow Control of Congested Networks*, Springer-Verlag.

[JoH89] S.L. Johnsson and C.-T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes", *IEEE Trans. Comput.*, vol. 38, pp. 1249-1267.

[Kle75] L. Kleinrock, *Queueing Systems, Vol. I: Theory*, John Wiley.

[KoK77] H. Kobayashi and A.G. Konheim, "Queueing Models for Computer Communications Systems Analysis", *IEEE Trans. Commun.*, vol. 25, pp. 2-29.

[Lei90] F.T. Leighton, private communication.

[MiC87] D. Mitra and R.A. Cieslak, "Randomized Parallel Communications on an Extension of the Omega Network", *J. ACM*, vol. 34, pp. 802-824.

[SaS85] Y. Saad and M.H. Schultz, "Data Communication in Hypercubes", Dept. of Computer Sciences, Research Report YALEU/DCS/RR-428, Yale University.

[Sta91] G.D. Stamoulis, "Routing and Performance Evaluation in Interconnection Networks", Ph. D. Thesis in preparation, Dept. of Electrical Engineering and Computer Science, M.I.T.

[Sto83] D. Stoyan, *Comparison Methods for Queues and Other Stochastic Models*, Edited by D.J. Daley, John Wiley.

[Upf84] E. Upfal, "Efficient Schemes for Parallel Communication", *J. ACM*, vol. 31, pp. 507-517.

[VaB81] L.G. Valiant and G.J. Brebner, "Universal Schemes for Parallel Communication", *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pp. 263-277.

[Val82] L.G. Valiant, "A Scheme for Fast Parallel Communication", *SIAM J. Comput.*, vol. 11, pp. 350-361.

[Var90] E.A. Varvarigos, "Optimal Communication Algorithms for Multiprocessor Computers", Report CICS-TH-192, Center for Intelligent Control Systems, M.I.T.

## APPENDIX A

In this appendix, we prove equation (19) in §4.3.2; this equation is as follows:

$$g_{m,i} = \frac{2^{[(m-i) \bmod d]} - 1}{2^d - 1} . \qquad (A.1)$$

Recall that $g_{m,i}$ denotes the probability that a packet has to cross arc $(x, x \oplus e_i)$ given that it has to cross arc $(x \oplus e_m, x)$ (*); note that, as implied by $(A.1)$, $g_{m,i}$ is independent of $x$.

Clearly, we have $g_{i,i} = 0$, which agrees with $(A.1)$ for $m = i$. Henceforth, we assume that $m \neq i$.

We consider a fixed packet, denoted by $\mathcal{P}$. Let $\Gamma_j$ be the event that $\mathcal{P}$ is a $j$-packet (see §4.2.2). Also, let $\Delta_l^z$ be the event that $\mathcal{P}$ has to cross arc $(z, z \oplus e_l)$. Using this notation, we have

$$g_{m,i} = \Pr[\Delta_i^x | \Delta_m^{x \oplus e_m}] = \sum_{j=1}^{d} \Pr[\Delta_i^x | \Gamma_j \text{ and } \Delta_m^{x \oplus e_m}] \cdot \Pr[\Gamma_j | \Delta_m^{x \oplus e_m}] . \qquad (A.2)$$

Furthermore, we have

$$\Pr[\Gamma_j | \Delta_m^{x \oplus e_m}] = \frac{\Pr[\Delta_m^{x \oplus e_m} | \Gamma_j] \cdot \Pr[\Gamma_j]}{\sum_{l=1}^{d} \Pr[\Delta_m^{x \oplus e_m} | \Gamma_l] \cdot \Pr[\Gamma_l]} . \qquad (A.3)$$

There holds $\Pr[\Gamma_l] = \frac{1}{d}$ for $l = 1, \ldots, d$, because each of the $d$ permissible trees has an a priori probability of $\frac{1}{d}$ to be selected by the fixed packet $\mathcal{P}$. Using this, it follows from $(A.3)$ that

$$\Pr[\Gamma_j | \Delta_m^{x \oplus e_m}] = \frac{\Pr[\Delta_m^{x \oplus e_m} | \Gamma_j]}{\sum_{l=1}^{d} \Pr[\Delta_m^{x \oplus e_m} | \Gamma_l]} . \qquad (A.4)$$

Notice now that an $l$-packet generated at node $y$ will have to traverse arc $(z, z \oplus e_m)$ if and only if it will reach node $z$ prior to crossing the $m$th dimension (possibly allowing for $z = y$); recalling the order under which $l$-packets cross the hypercube dimensions (see §4.2.2), it is seen that this occurs if and only if one of the following holds:

(a) $m < l$ and $z_m = y_m, \cdots, z_{l-1} = y_{l-1}$;

(b) $m \geq l$ and $z_m = y_m, \cdots, z_d = y_d, z_1 = y_1, \cdots, z_{l-1} = y_{l-1}$, with an obvious interpretation for $l = 1$.

(*) The expression "the packet has to cross arc $(z, z \oplus e_j)$" should be interpreted as follows: "arc $(z, z \oplus e_j)$ belongs to the tree selected by the packet".

For fixed values of $z, l$ and $m$, the above condition holds for $2^{[(m-l) \bmod d]}$ different nodes $y$. Since each node is a priori equiprobable to have generated the fixed packet $\mathcal{P}$, it follows that

$$\Pr[\Delta_m^z | \Gamma_l] = \frac{2^{[(m-l) \bmod d]}}{2^d} .$$

Applying this with $z = x \oplus e_m$ and using $(A.4)$, we obtain

$$\Pr[\Gamma_j | \Delta_m^{x \oplus e_m}] = \frac{2^{[(m-j) \bmod d]}/2^d}{\sum_{l=1}^{d} 2^{[(m-l) \bmod d]}/2^d} = \frac{2^{[(m-j) \bmod d]}}{2^d - 1} , \qquad (A.5)$$

where we have used the fact $\sum_{l=1}^{d} 2^{[(m-l) \bmod d]} = \sum_{l=1}^{d} 2^{l-1} = 2^d - 1$.

Combining $(A.2)$ with $(A.5)$, we obtain

$$g_{m,i} = \sum_{j=1}^{d} \Pr[\Delta_i^x | \Gamma_j \text{ and } \Delta_m^{x \oplus e_m}] \frac{2^{[(m-j) \bmod d]}}{2^d - 1} . \qquad (A.6)$$

Conditioning on the union of the events $\Gamma_j$ and $\Delta_m^{x \oplus e_m}$ is equivalent to conditioning on the fact "the $j$-packet $\mathcal{P}$ has to cross arc $(x \oplus e_m, x)$". Given this fact, $\mathcal{P}$ has to cross arc $(x, x \oplus e_i)$, with probability 1, if and only if it does not cross the $i$th dimension prior to crossing the $m$th dimension; otherwise, the probability that $\mathcal{P}$ has to cross $(x, x \oplus e_i)$ is zero. Again, recalling the order under which $j$-packets cross the hypercube dimensions (see §4.2.2), it is seen that $\mathcal{P}$ will cross arc $(x, x \oplus e_i)$ if and only if one of the following holds: $j \leq m < i$ or $i < j \leq m$ or $m < i < j$. Based on the above discussion, we consider three cases separately:

(a) Case $m < i \leq d$: Using $(A.6)$, we have

$$\begin{aligned} g_{m,i} &= \sum_{j=1}^{m} \frac{2^{[(m-j) \bmod d]}}{2^d - 1} + \sum_{j=i+1}^{d} \frac{2^{[(m-j) \bmod d]}}{2^d - 1} \\ &= \frac{1}{2^d - 1} \sum_{j=1}^{m} 2^{m-j} + \frac{1}{2^d - 1} \sum_{j=i+1}^{d} 2^{d+m-j} \\ &= \frac{1}{2^d - 1} \left(2^m - 1 + 2^{d+m-i} - 2^m\right) \\ &= \frac{1}{2^d - 1} \left(2^{d+m-i} - 1\right) , \end{aligned}$$

which proves $(A.1)$ for the present case, since $[(m-i) \bmod d] = d + m - i$.

(b) Case $m < i = d$: Using $(A.6)$, we have

$$g_{m,i} = \sum_{j=1}^{m} \frac{2^{[(m-j) \bmod d]}}{2^d - 1} = \frac{1}{2^d - 1} \sum_{j=1}^{m} 2^{m-j} = \frac{1}{2^d - 1} (2^m - 1) ,$$

which proves $(A.1)$ for the present case, since $[(m-i) \bmod d] = [(m-d) \bmod d] = m$.

(c) Case $m > i$: Using $(A.6)$, we have

$$g_{m,i} = \sum_{j=i+1}^{m} \frac{2^{[(m-j) \bmod d]}}{2^d - 1} = \frac{1}{2^d - 1} \sum_{j=i+1}^{m} 2^{m-j} = \frac{1}{2^d - 1}(2^{m-i} - 1)\,,$$

which obviously proves $(A.1)$ for the present case.

The above three cases establish $(A.1)$ for $m \neq i$; since $(A.1)$ is trivially true for $m = i$, the proof is completed. **Q.E.D.**

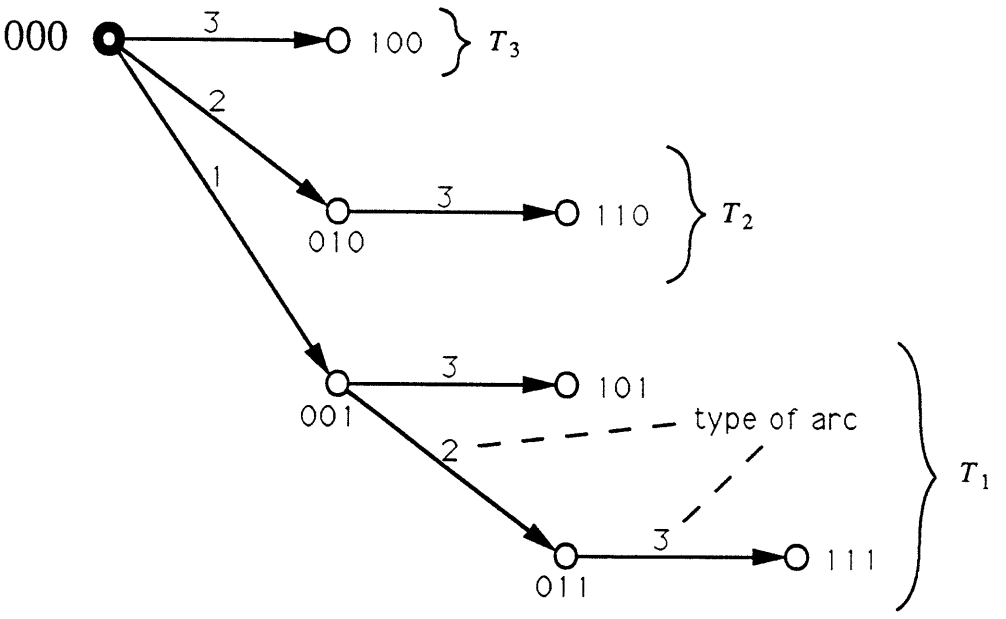



Figure 1: A completely unbalanced spanning tree, for $d = 3$.

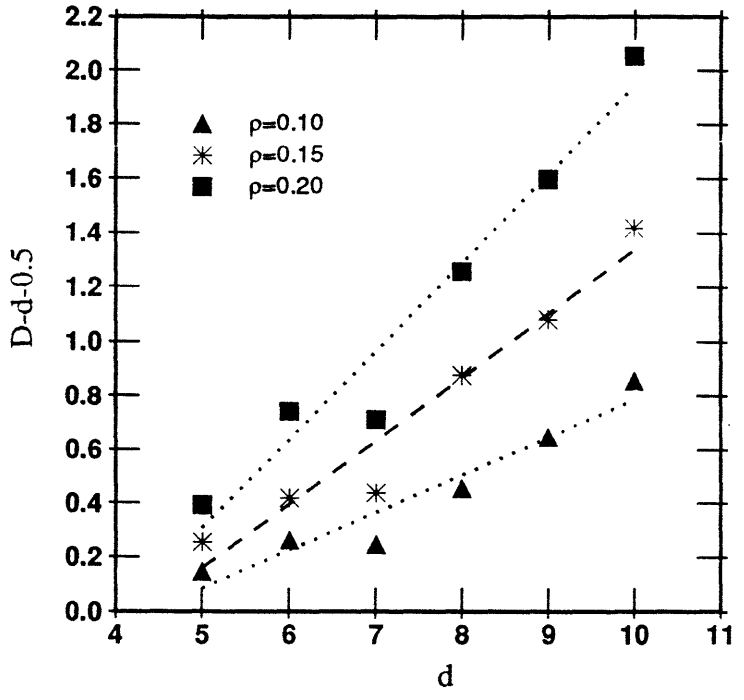



Figure 2: The first order term in the delay $D$ for the scheme of §4.3.1.

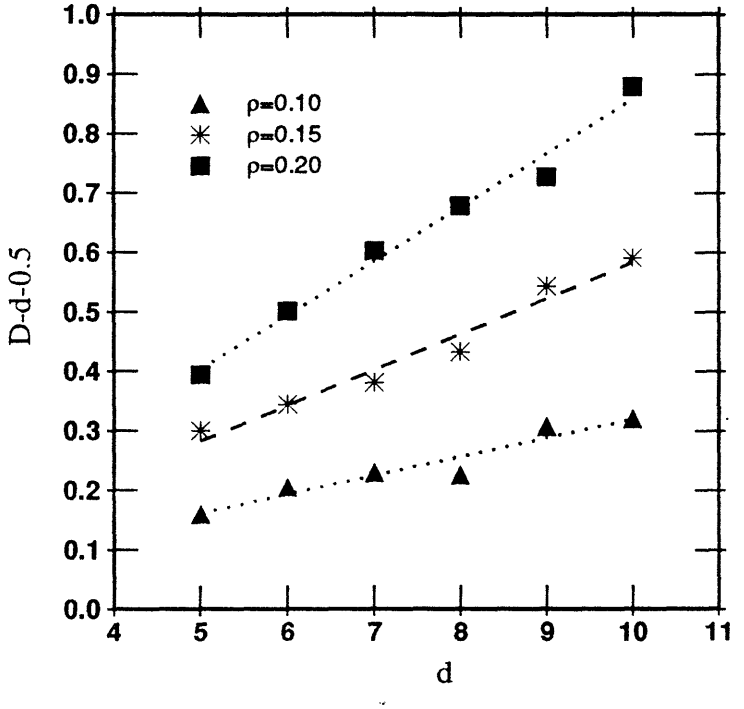



Figure 3: The first order term in the delay $D$ for the scheme of §4.3.2.

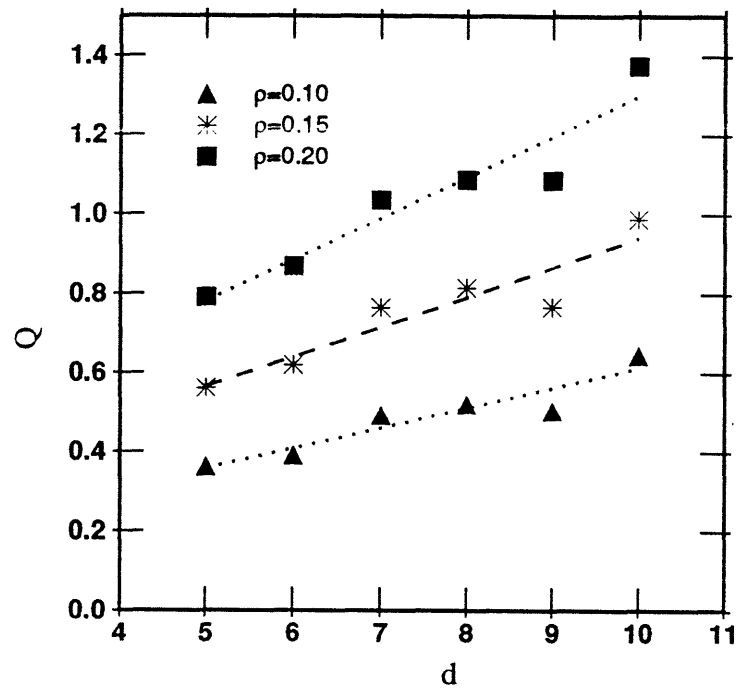



Figure 4: The average queue-size $Q$ per node for the scheme of §4.3.1.

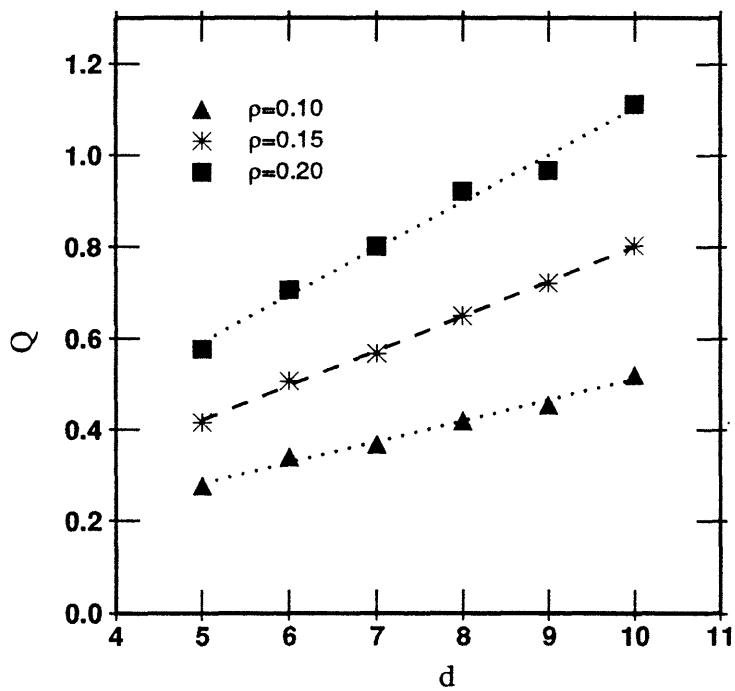



Figure 5: The average queue-size $Q$ per node for the scheme of §4.3.2.

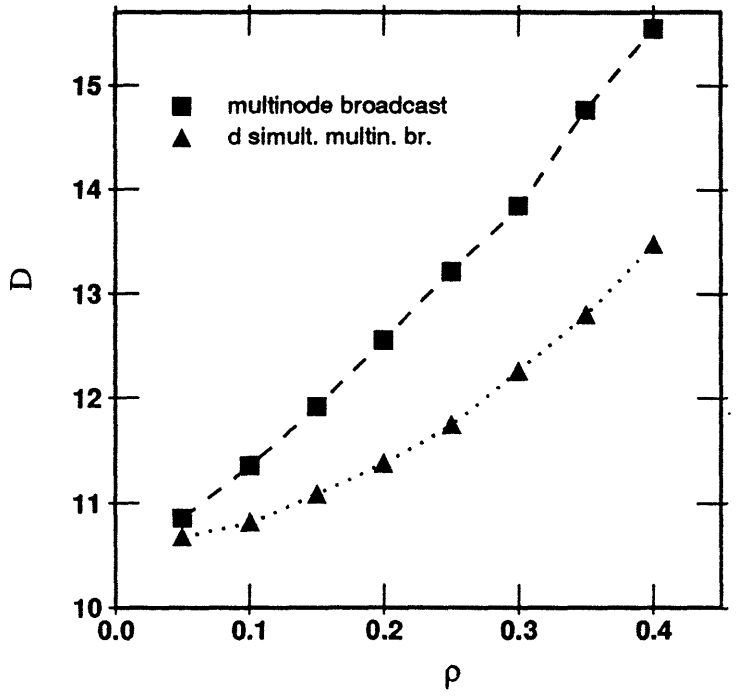



Figure 6: Comparing the delay induced by the schemes of §§4.3.1 and 4.3.2, for $d = 10$.

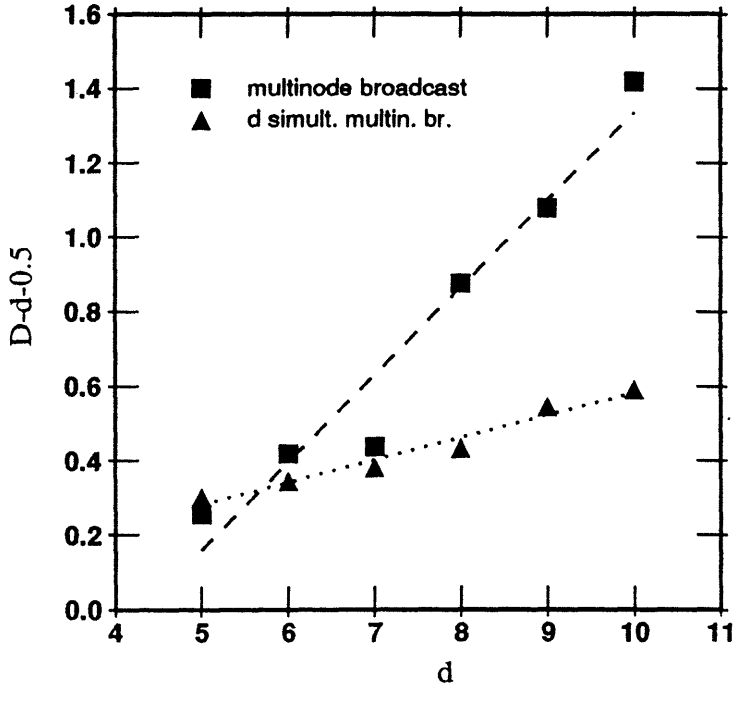



Figure 7: Comparing the delay induced by the schemes of §§4.3.1 and 4.3.2, for $\rho = 0.15$.

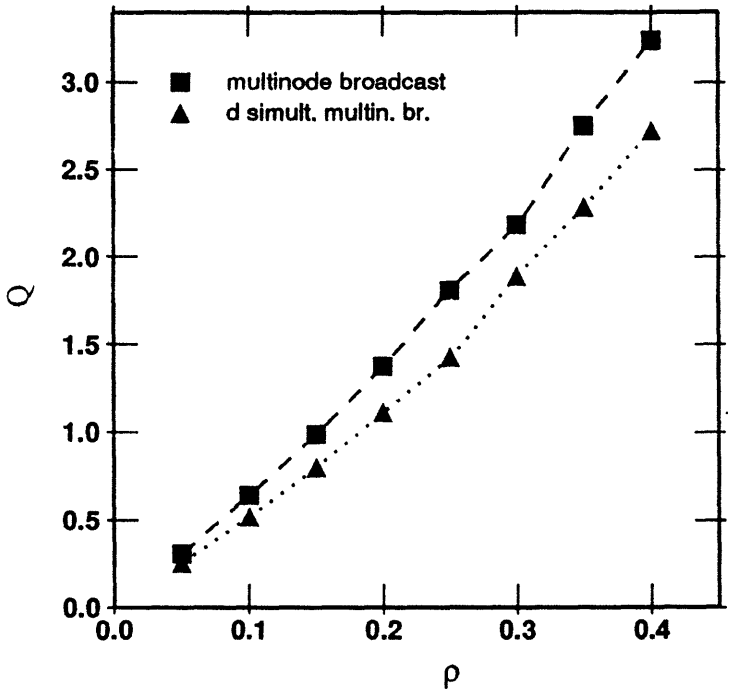



Figure 8: Comparing the queue-sizes for the schemes of §§4.3.1 and 4.3.2, for $d = 10$.

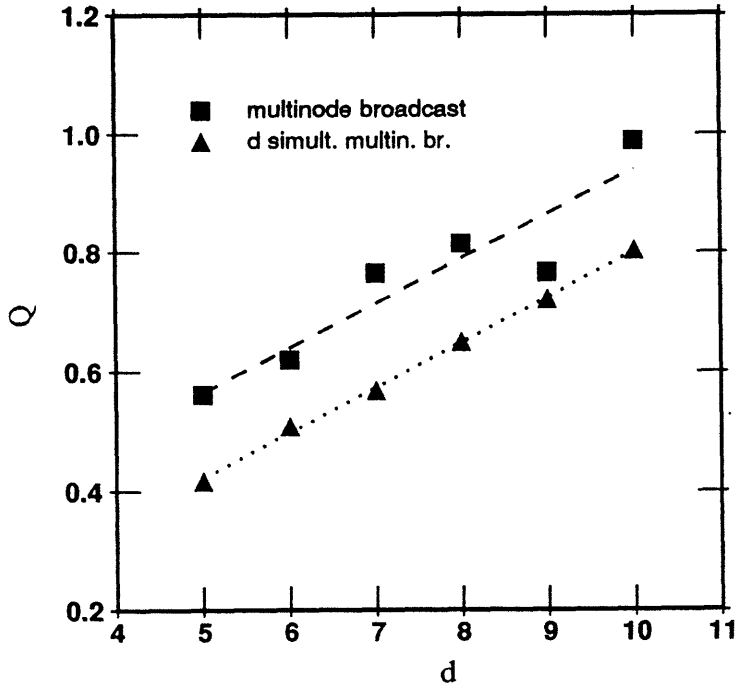



Figure 9: Comparing the queue-sizes for the schemes of §§4.3.1 and 4.3.2, for $\rho = 0.15$.

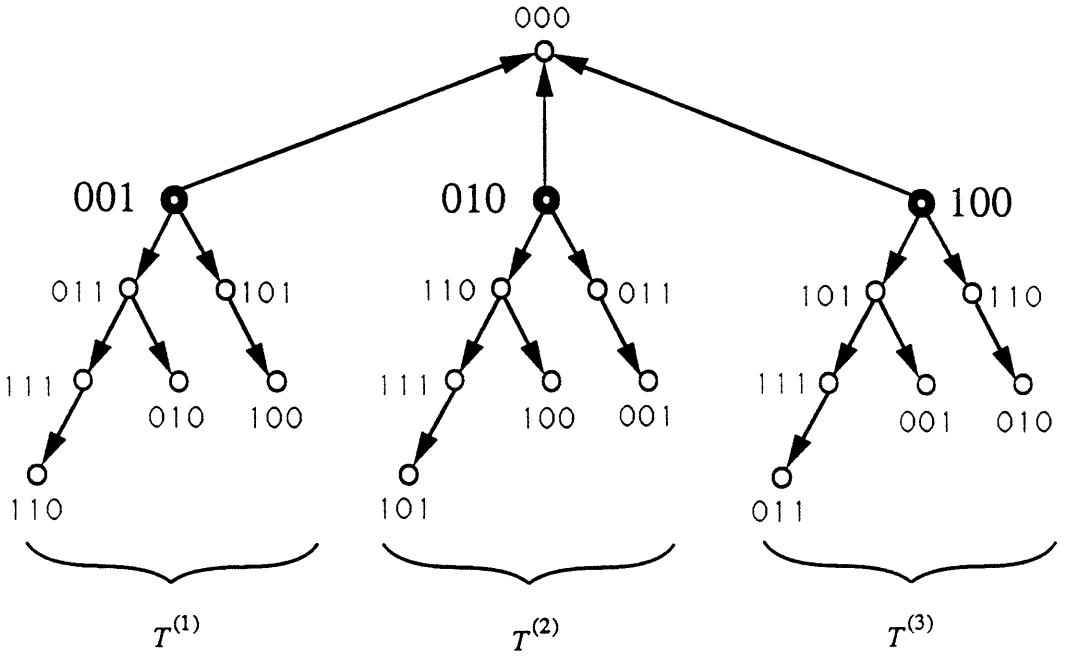



Figure 10: The $d$ disjoint spanning trees, for $d = 3$.

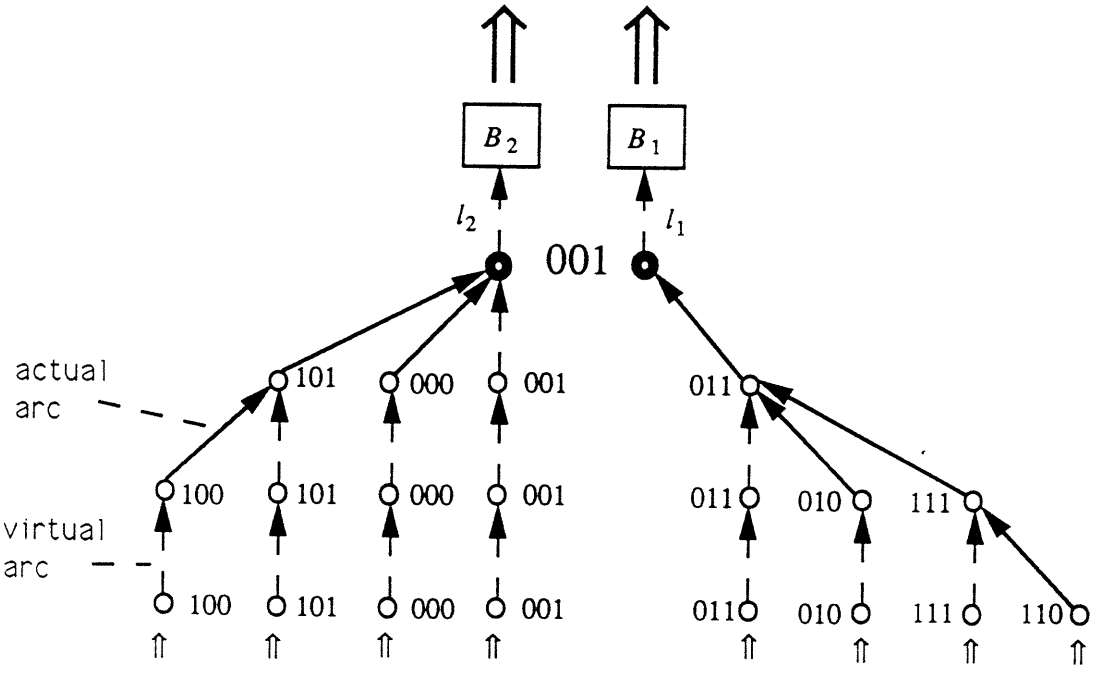



Figure 11: Introducing the virtual arcs in $T^{(1)}$, for $d = 3$.

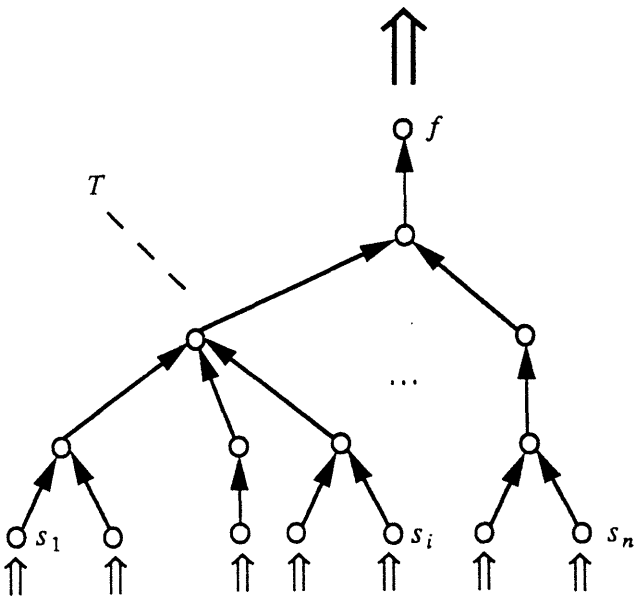



Figure 12.a: The tree $T$ of paths.

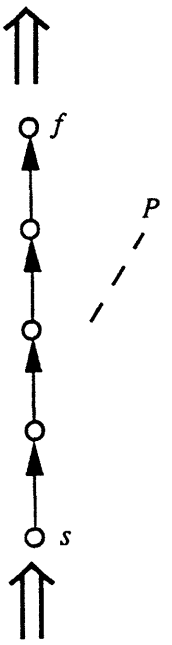
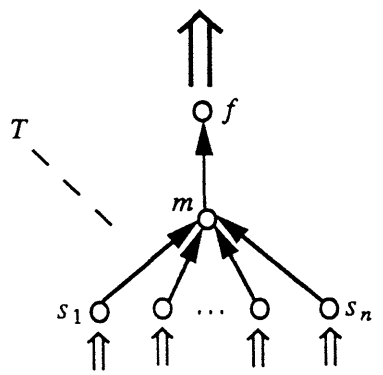



Figure 12.b: The single path $P$.



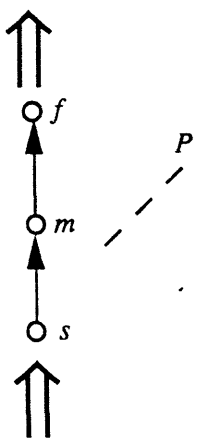




Figure 13: The simple case for Lemma 3.

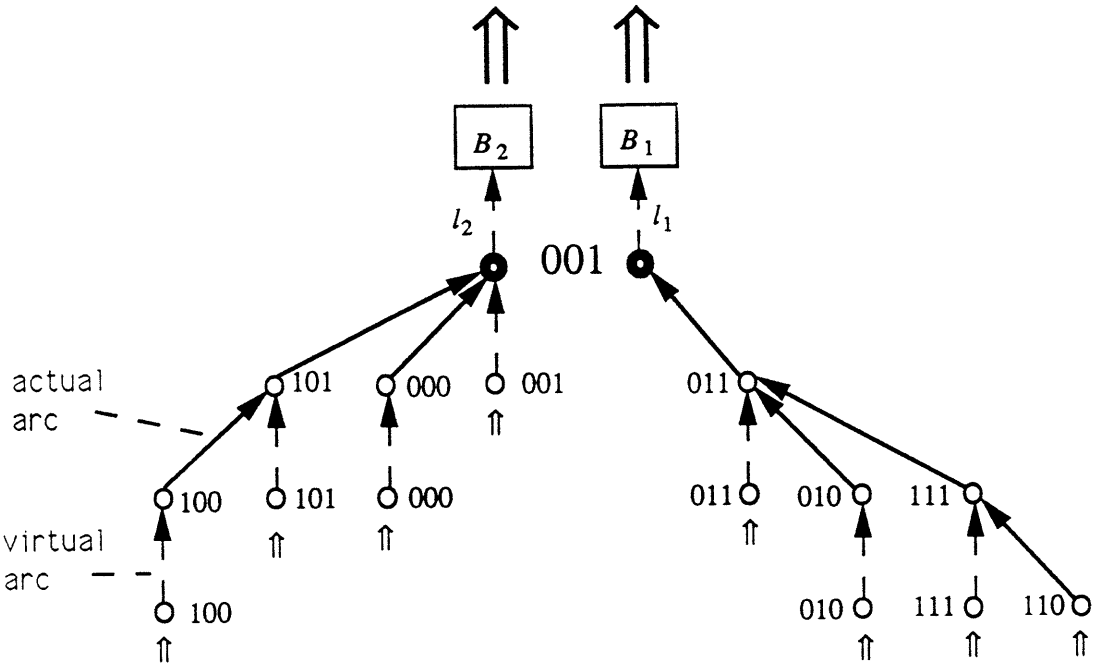



Figure 14: Eliminating most of the virtual arcs from $T^{(1)}$, for $d = 3$.

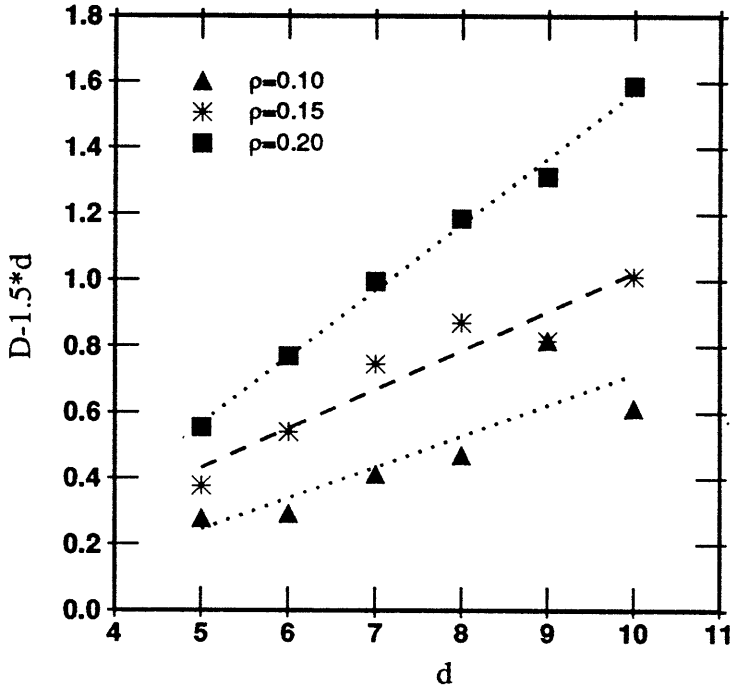



Figure 15: The first order term in the delay for the non-idling version of the indirect scheme of §5.1.

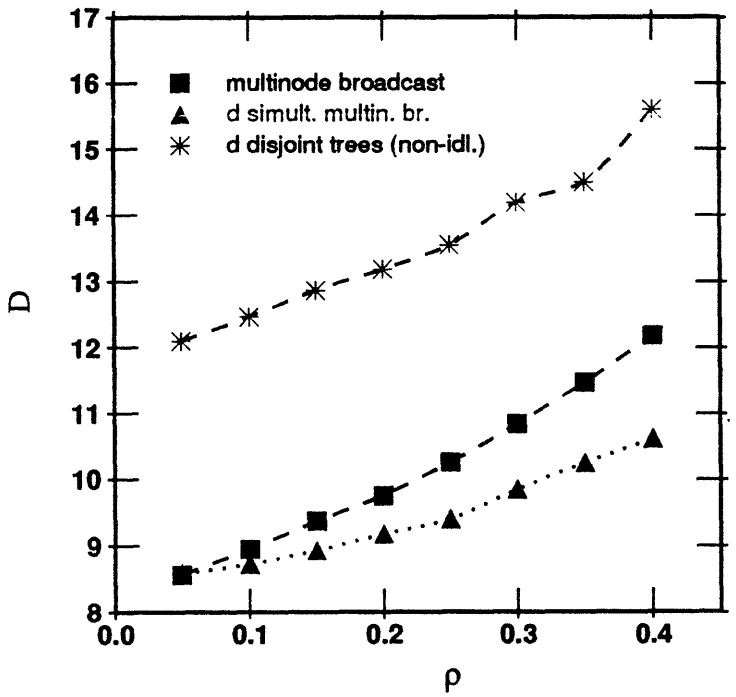



Figure 16: Comparing the delay induced by the various schemes, for $d = 8$.

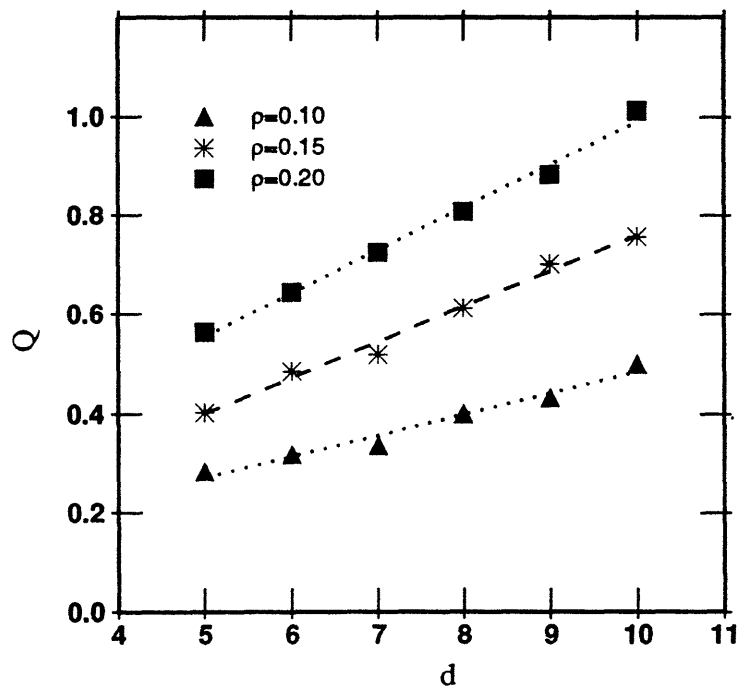



Figure 17: The average queue-size $Q$ per node for the non-idling version of the indirect scheme of §5.1.

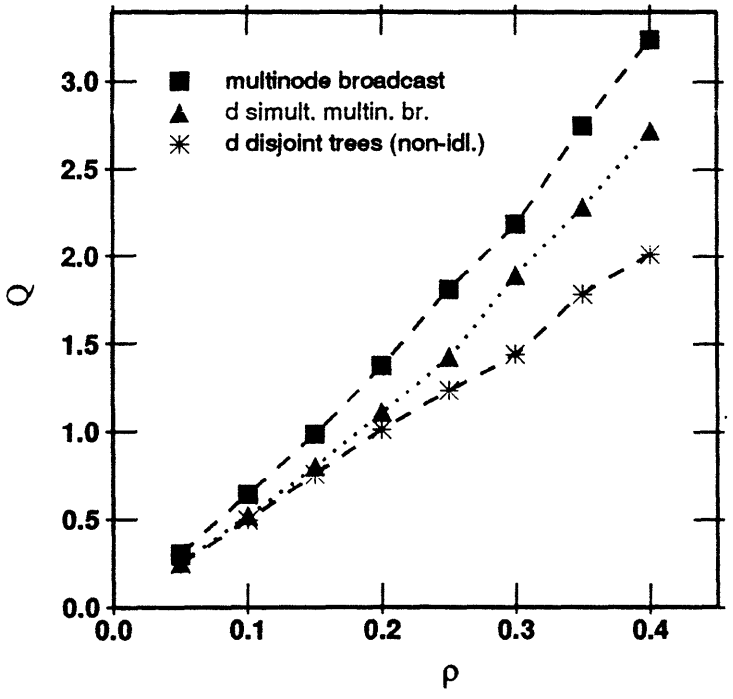



Figure 18: Comparing the average queue-size $Q$ per node for the various schemes, for $d = 10$.

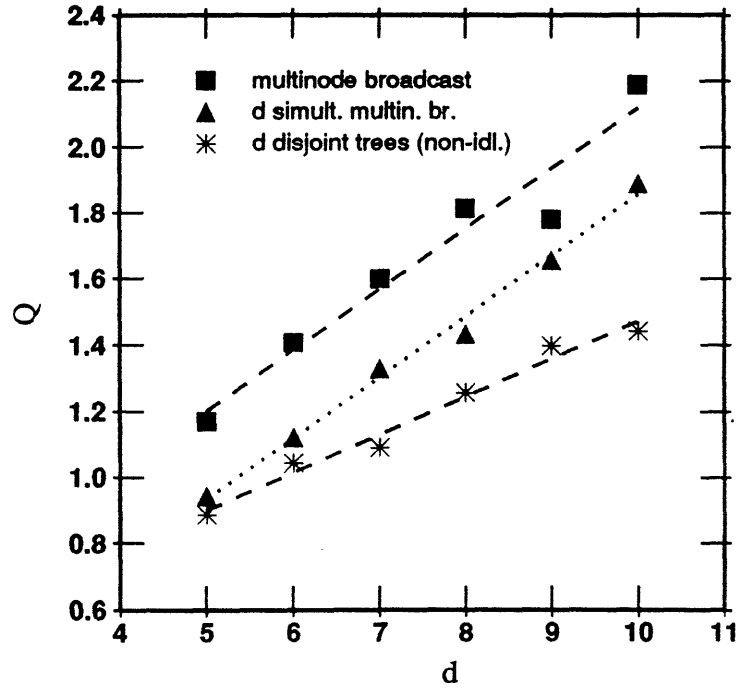



Figure 19: Comparing the average queue-size $Q$ per node for the various schemes, for $\rho = 0.3$.

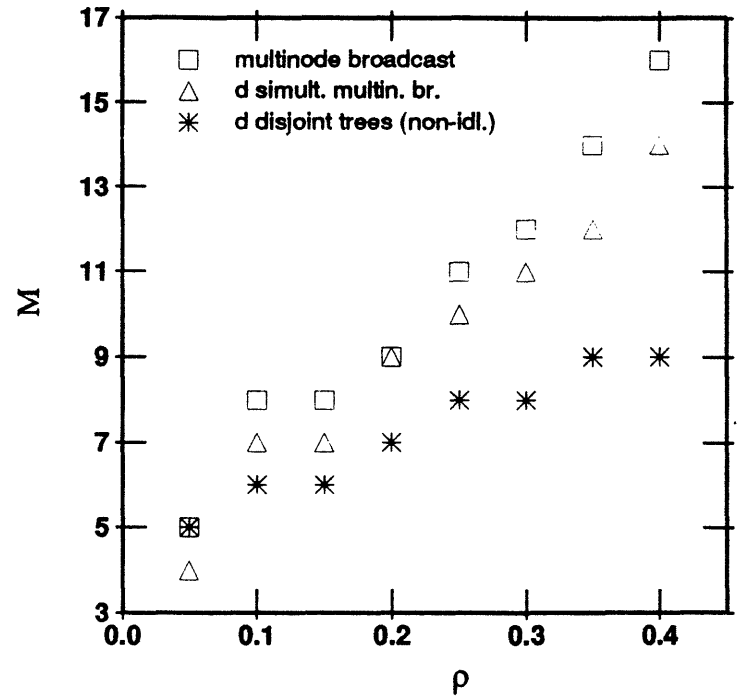



Figure 20: Comparing the maximum queue-size $M$ per node for the various schemes, for $d = 10$.

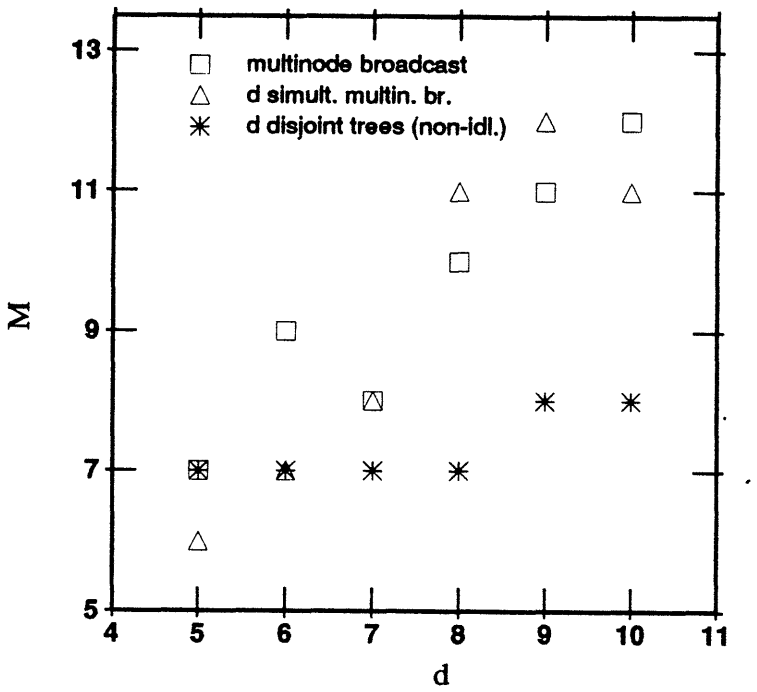



Figure 21: Comparing the maximum queue-size $M$ per node for the various schemes, for $\rho = 0.3$.