Performance of Hypercube Routing Schemes With or Without Buffering

Emmanouel A. Varvarigos and Dimitri P. Bertsekas

Abstract— **We consider two different hypercube routing schemes, which we call the *simple* and the *priority* schemes. We evaluate the throughput of both the unbuffered and the buffered version of these schemes for random multiple node-to-node communications. The results obtained are approximate, but very accurate as simulations indicate, and are given in particularly interesting forms. We find that little buffer space (between one and three packets per link) is necessary to achieve throughput close to that of the infinite buffer case. We also consider two deflection routing schemes, called the *simple nonwasting deflection* and the *priority nonwasting deflection* schemes. We evaluate their throughput using simulations, and compare it to that of the priority scheme.**

I. Introduction

We consider several routing schemes for node-to-node communication in hypercube networks, and we analyze their performance under a stochastic traffic environment. In particular, we assume that packets having a single destination are generated at each node of a hypercube according to some probabilistic rule over an infinite time horizon. The destinations of the new packets are uniformly distributed over all hypercube nodes. Packets have equal length and require one unit of time to be transmitted over a link. We are interested in the average throughput of the network when it reaches steady state.

One-to-one routing has been extensively analyzed in the literature for a variety of network topologies. One line of research that has been pursued is related to the so-called *randomized algorithms*. In randomized algorithms a packet is sent to some random intermediate node before being routed to its destination. Randomized routing algorithms were first proposed by Valiant [25], and Valiant and Brebner [24] for the hypercube network. They were extended to networks of constant degree by Upfal [23], and they were improved to achieve constant queue size per node by Pippenger [21]. The results on randomized routing are usually of the following nature: it is proved that for sufficiently large size of the network, a permutation task can be executed in time less than some multiple of the network diameter with high probability. A disadvantage of this line of research is that the results obtained are of an asymptotic nature, and apply only to permutation tasks.

A second line of research, which is closer to our work, is the evaluation of the throughput of a network in steady state. A routing scheme, called *deflection routing*, has been investigated by Greenberg and Goodman [8] for mesh networks (numerically), Greenberg and Hajek [9] for hypercubes (analytically), Maxemchuk [18] for the Manhattan network and the shuffle-exchange (numerically and through simulations), Krishna and Hajek [13] for shuffle-like networks (numerically), and Varvarigos [27] for hypercubes (numerically, for a priority deflection scheme). The analysis in these works was either approximate, or it involved simulations. Since deflection routing is a way to circumvent the need for buffers, buffer size played a minor role in these analyses. Dias and Jump [6] analyzed approximately the performance of buffered Delta networks. Stamoulis [22] considered a greedy scheme in hypercubes and butterflies for infinite buffer space per node. He analyzed a closely related Markovian network with partial servers, and used the results obtained to bound the performance of the hypercube scheme, without using approximations.

In this paper we propose two new routing schemes for hypercube networks, which we call the simple and the priority routing schemes. In both schemes, packets follow shortest paths to their destinations. However, they may remain at some node even if there is a free link that they want to use (this phenomenon is called *idling*). The switch used at the nodes is simple, fast, and inexpensive, and uses $\Theta(d)$ wires, where d is the dimension of a hypercube, as opposed to the $\Theta(d^2)$ wires of a crossbar switch. In both schemes, packets may be dropped due to the unavailability of buffer space. In the simple scheme, packets competing for the same link have equal priority, except for the new packets, which have the lowest priority. In the priority scheme, however, packets that have travelled more hops have priority. As the results of Section IV will indicate, the priority scheme has significantly larger throughput than the simple scheme, especially when the buffer space is small and the load is heavy.

The time axis is slotted and each packet requires one unit of time (or slot) in order to be transmitted over a link. The destinations of new packets are assumed to be uniformly distributed over all nodes. The throughput results that we derive in Sections III and IV for the simple and the priority schemes are given in particularly interesting forms. The throughput of the simple scheme (with or without buffers) is given in parametric form as a function of the generation

Manuscript received December 15, 1992; revised September 1, 1993; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor I. Cidon. This work was supported by the NSF under Grants NSF-DDM-8903385 and NSF-RIA-08930554, and by the ARO under Grants DAAL03-86-K-0171 and DAAL03-92-G-0309.

E. A. Varvarigos is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA.

D. P. Bertsekas is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

IEEE Log Number 9403687.

rate of new packets. The parametric solution involves a single parameter, and is close to a closed-form solution. The throughput of the priority routing scheme is obtained from a recursion of only d steps, where d is the hypercube dimension. Our results are approximate but, as simulations given in Section V suggest, very close to being accurate.

Our results will indicate that buffer space for one to three packets per link is adequate to achieve throughput close to that of the infinite buffer case, even for rather heavy load. Increasing the buffer space further increases the throughput only marginally, and may be characterized as an ineffective use of the hardware resources.

If we are using packet switching, then the only way to avoid packets being dropped is deflection routing (other methods pose restrictions on the transmission rates of the sources, as in [7], or they do not use a packet switching format, as in the CSR protocol proposed in [26]). The Connection Machine model 2, uses deflection routing and has space only for the packets being transmitted. In Section VI we focus on two deflection schemes, which we call the *simple nonwasting deflection* and the *priority nonwasting deflection* schemes, and use simulations to evaluate their throughput. Both deflection schemes outperform the unbuffered priority scheme for hypercubes of large dimension. In fact, for uniform traffic, the throughput of the priority nonwasting deflection scheme seems to tend to the maximum possible when the dimension of the hypercube increases. However, if we can afford some buffer space for the priority scheme then its performance can be superior to that of the deflection schemes. Deflection routing has several disadvantages (see [19]), and requires the use of crossbar switches at the nodes. The simple and the priority schemes do not exhibit these disadvantages, and they use switches that are much simpler than crossbar switches.

II. Description Of The Hypercube Node Model And The Routing Schemes

Each node of an $N = 2^d$-node hypercube is represented by a unique d-bit binary string $s_{d-1}s_{d-2}\cdots s_0$. There are links between nodes whose representations differ in one bit. The *routing tag* $s \oplus t$ of a packet that is currently located at node s and has as destination node t is the result of the bitwise exclusive OR operation between the binary representations of s and t. We also denote by e_i the binary string whose ith bit is equal to one, and all other bits are equal to zero. A link connecting node s to node $s \oplus e_i$, $i = 0, 1, \ldots, d-1$, is called a *link of the* ith *dimension*. Note that if the ith bit of the routing tag of a packet is equal to one, then the packet must cross a link of dimension i in order to arrive at its destination.

Our node model is as follows: Each node has a queue for each of its outgoing links. The queue of node s that corresponds to link i is called the ith *link queue*, and is denoted by $\mathcal{Q}_i(s)$. A link queue is composed of two *buffers* that can hold $k+1$ packets each. The first buffer, denoted by $\mathcal{Q}_i^1(s)$, is called the *forward buffer*, and can be used only by packets that have to cross the ith dimension. The second buffer, denoted by $\mathcal{Q}_i^0(s)$, is called *internal buffer*, and can be used only by packets that do not have to cross the ith dimension. The case $k = 0$ is called the *unbuffered* case, and corresponds to the situation where a packet received during a slot is transmitted at the next slot, or is dropped, otherwise.

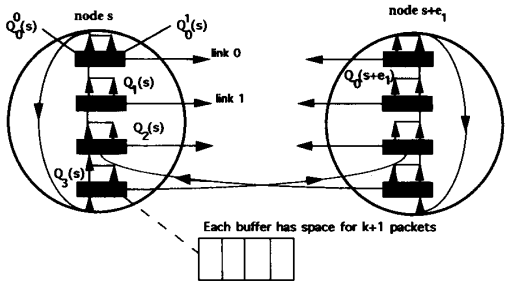


Fig. 1. A node of the hypercube. For a comparison with other node (router) models, see also Fig. 13.

The queues of the nodes are linked in the following way. The internal buffer $\mathcal{Q}_i^0(s)$ is connected to queue $\mathcal{Q}_{(i-1)\mathrm{modd}}(s)$ of the same node, while the forward buffer $\mathcal{Q}_i^1(s)$ is connected to queue $\mathcal{Q}_{(i-1)\mathrm{modd}}(s \oplus e_i)$ of the neighbor node $s \oplus e_i$ (see Fig. 1). This router, which we call *descending dimensions switch*, restricts the class of switching assignments that are feasible. It is, however, simpler, faster, and less expensive than a crossbar switch (see also Fig. 13).

In both the simple and the priority scheme, the packets traverse the hypercube dimensions in descending (modulo d) order, starting with a randomly chosen dimension. In particular, consider a packet that arrives at queue $\mathcal{Q}_i(s)$ [either from buffer $\mathcal{Q}_{(i+1)\mathrm{modd}}^0(s)$ of the same node, or from buffer $\mathcal{Q}_{(i+1)\mathrm{modd}}^1(s \oplus e_{i+1})$ of the neighbor node $s \oplus e_{i+1}$, or a new packet]. Then the ith bit of its routing tag is checked. Depending on whether this bit is a one or a zero, the packet claims buffer $\mathcal{Q}_i^1(s)$ in order to be transmitted during the next slot to queue $\mathcal{Q}_{(i-1)\mathrm{modd}}(s \oplus e_i)$ of the neighbor node $s \oplus e_i$, or it claims buffer $\mathcal{Q}_i^0(s)$ in order to be internally passed to the next queue $\mathcal{Q}_{(i-1)\mathrm{modd}}(s)$ of the same node. In both cases conflicts may arise because more than one old and new packets may claim the same buffer of a link. When conflicts occur, packets may be dropped if there is not enough space at the buffer.

In the simple scheme, conflicts over buffer space are resolved at random, with the exception of the newly generated packets, which are always dropped in case of conflict. In the priority scheme the packets that have travelled more have priority when they compete for buffer space. Since the packets that are dropped are those that have travelled less, the waste of bandwidth resulting from dropping packets is expected to be smaller in the priority scheme than in the simple scheme. If two packets have travelled an equal number of links then one of them is dropped with equal probability.

In the unbuffered case, packets stay in the network exactly d time units (slots). Since the bits of the routing tag of a packet are cyclically made equal to zero, packets are delivered to their destination with delay d, unless dropped on the way. A packet may arrive at its destination earlier, but it is not removed before the d^{th} slot; during the last slots it may travel from

link-queue to link-queue of the same node until the time of its removal. The predictability of the packet delay facilitates the choice of an appropriate window size and time-out interval if a retransmission protocol is to be used (see [3]). In the buffered case, packets may wait in some buffers, thereby potentially increasing their total travel to more than d time units.

We denote by p_0 the probability that a new packet is generated during a slot and claims a buffer $\mathcal{Q}_i^j(s)$, $j \in \{0,1\}, i \in \{0,1,\ldots,d-1\}$. At most one new packet can claim a link buffer during a slot, and it will be rejected if there is a continuing packet already in the buffer. New packets denied acceptance to the network, or packets that are dropped are not retransmitted. One can visualize this model for the arrival process of new packets in the following way. Whenever a link is empty, a packet that wishes to use the link is requested, and with probability p_0 such a packet exists. If, for example $p_0 = 1/2$, and a link is empty 2/3 of the time, then the source inserts a packet 1/3 of the time. We will refer to p_0 as the *probability of access*. Other models for the arrival process can be incorporated in our model, as long as the new arrivals at each link are independent from the arrivals at other links, and p_0 can be calculated. Note that for both schemes under investigation, the internal and the external links are equivalent, in the sense that the probability of a packet being transmitted on an internal link is equal to the probability of a packet being transmitted on an external one.

III. The Simple Scheme

In this section we evaluate the throughput of the simple scheme. The unbuffered and the buffered cases are analyzed in Sections III-A and -B, respectively. Recall that in the simple scheme all packets have equal priority, except for the new packets, which have the lowest priority. Our analysis is approximate but very close to being accurate, as Section V will indicate. We will find a parametric expression that gives the throughput as a function of the probability of access p_0. The parametric solution involves a single parameter, and is almost as simple as a closed-form solution. This is significant because most of the results found in the literature for the steady-state throughput of various multiprocessor networks and routing schemes are given in a less direct way (typically they are numerical and simulation results, or they have an asymptotic character).

A packet will be referred to as a packet of *type* i when it has been transmitted (on forward or internal links) i times, including the current transmission. A packet received at a node with type different from d is a *continuing* packet. This definition does not include packets that have been received during previous slots, which are called *buffered* packets. Packets generated at a node are called *new* packets.

A. Analysis of the Simple Scheme Without Buffers

We first deal with the unbuffered case where each link buffer can hold only the packet under transmission (internally or to a neighbor node). We denote by $p_i, i = 1,2,\ldots,d$, the steady-state probability that a packet of type i is transmitted on a given link during a slot, and by e the steady-state probability

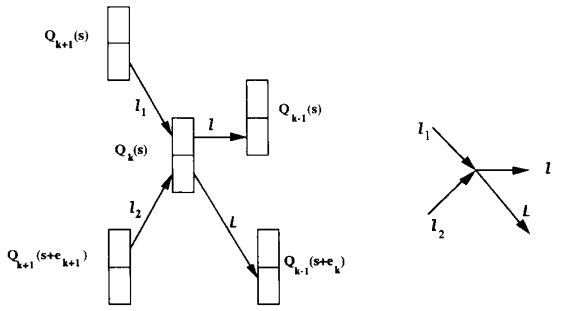


Fig. 2. A link l, and the two links leading to it.

that a link is idle during a slot. By symmetry, both the p_i's and e are the same for all links. Clearly, we have

$$e + \sum_{i=1}^{d} p_i = 1. \tag{1}$$

Note that since a type d packet is ready to exit the network, we may view p_d as the throughput per link. We will derive the relationship between p_d and the probability of access p_0.

Consider a particular link l (for example, the one connecting $Q_i^1(s)$ with $Q_{(i-1)\mathrm{modd}}(s+e_i)$). Call l_1 and l_2 the internal and the forward links, respectively, that lead to l (see Fig. 2). We make the following assumption.

Approximating Assumption A.1: Events in l_1 during a slot are independent of events in l_2 during the same slot.

In our schemes, a packet transmitted on l_1 and a packet transmitted on l_2 have used in the past links belonging to different subcubes of the hypercube. However, events in l_1 are not necessarily independent of events in l_2, as will be explained in Section V. Nonetheless, we believe that the approximation A.1 is quite accurate, as the discussion in Section V will suggest, and simulation results will support.

Let E_j, $j = 1,2$, be the event that a packet $\mathcal{P}$ of type $i-1$ arrives on link l_j, requests link l, and gets it. Then, for $i = 2,3,\ldots,d$, the probability p_i that a packet of type i is transmitted on link l is

$$p_i = \Pr(E_1) + \Pr(E_2) = 2\Pr(E_1).$$

Since a packet $\mathcal{P}$ of type $i-1$ arrives on l_1 with probability p_{i-1}, and requests link l with probability 1/2, the preceding equation gives

$$p_i = p_{i-1}\Pr(\mathcal{P} \text{ not dropped}), \qquad i = 2,3,\ldots,d.$$

Packet $\mathcal{P}$ may be dropped due to a conflict with a packet $\mathcal{Q}$ of type different from d coming on l_2 (packets of type d are removed from the network since they have arrived at their destination). The probability that a packet of type different from d arrives on link l_2 and claims link l is equal to

$$\frac{\sum_{j=1}^{d-1} p_j}{2}.$$

Since conflicts are resolved at random, such a packet will cause packet $\mathcal{P}$ to be dropped with probability 1/2. Thus, under the

approximating assumption A.1, we have

$$p_i = p_{i-1}\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{4}\right), \qquad i = 2,3,\ldots,d. \quad (2)$$

The probability p_1 that a packet of type 1 (that is, a new packet) is transmitted on link l is the product of two probabilities:

1) The probability that no packet arrives on l_1 or l_2 requesting link l; this probability is equal to

$$\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2.$$

2) The probability that a new packet is generated at l; this probability is equal to p_0.

Therefore, p_1 is given by

$$p_1 = p_0\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2. \quad (3)$$

Similarly, the probability e that a link is idle is equal to

$$e = (1 - p_0)\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2. \quad (4)$$

It is useful to define

$$\theta = p_d + e = 1 - \sum_{j=1}^{d-1} p_j, \quad (5)$$

where θ will be treated as a free parameter. Then (2)-(5) give

$$p_d = p_0 \frac{(1+\theta)^2}{4}\left(\frac{3+\theta}{4}\right)^{d-1} \quad (6)$$

and

$$e = \frac{1}{4}(1 - p_0)(1+\theta)^2, \quad (7)$$

Adding (6) and (7), and taking into account that $\theta = p_d + e$ we get

$$\theta = p_0 \frac{1}{4^d}(1+\theta)^2(3+\theta)^{d-1} + \frac{1}{4}(1-p_0)(1+\theta)^2,$$

or

$$p_0 = \left(\frac{1-\theta}{1+\theta}\right)^2 \frac{1}{1 - \left(\frac{3+\theta}{4}\right)^{d-1}}. \quad (8)$$

Equations (6) and (8) give the relationship between p_d and p_0 in parametric form.

Since there are $2d$ (internal or forward) links per node, the throughput R per node is

$$R = 2dp_d.$$

It can be proven that for uniformly distributed destinations, R is always less than or equal to two.

Note that the eligible values of the parameter θ range continuously from 1 (when $p_0 = 0$, which gives $e = 1$ and $p_d = 0$) to some number close to 0 (when $p_0 = 1$, which gives $e = 0$ and p_d less than $1/d$). By giving values to θ we can find the corresponding values of p_0 and R. The fact that the range of θ is not the entire interval [0,1] does not create any difficulty, since the values of θ that are not feasible give $p_0 > 1$ [values of θ that give $p_0 < 1$ are feasible, since we can then find values for p_i, $i = 0,1,\ldots,d$, and e that satisfy (2)-(5)]. Fig. 3 illustrates the results obtained for $d = 11$.

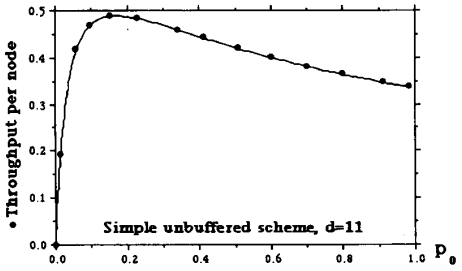


Fig. 3. Throughput per node of the simple scheme without buffers for $d = 11$.

B. Analysis of the Simple Scheme With Buffers

In this section we analyze the buffered version of the simple scheme. In particular, we assume that each link buffer can hold up to k packets, in addition to the packet under transmission. The scheme is the same with that analyzed in Section III-A with the difference that when packets want to use the same link, one of them is transmitted and the other is stored, if there is enough space in the buffer, or dropped, otherwise. The analysis of Section III-A corresponds to the case $k = 0$ of this section. We treat separately the unbuffered and the buffered cases because an additional approximation is needed for the buffered case. We assume that continuing packets have priority over new or buffered packets when claiming a link. New packets are dropped if there are continuing or buffered packets that want to use the link. The reason we do not allow new packets to be buffered is that we want to keep the buffer space for packets already in transit, and not to fill it with new packets. A new packet is available to enter at an otherwise idle link with probability p_0 during a slot.

We will find a relationship between the throughput p_d per link and the probability of access p_0. The relationship will be given in parametric form, involving a single parameter. Note that corresponding results in the literature are typically obtained through the use of numerical methods and/or simulations ([6], [9], [5], [18], [4], [27], [1]), or they are of an asymptotic nature in the number of processors or in the buffer size ([9], [22]).

In order to analyze the buffered version of the simple scheme we need, in addition to the approximating assumption A.1, the following assumption.

Approximating Assumption A.2: The arrivals of packets at a buffer during a slot are independent of the arrivals at the buffer during previous slots.

Independence approximating assumptions are found in most throughput analyses of buffered routing schemes of multiprocessor systems.

We denote by b_i, $i = 0,1,\ldots,k$, the probability that there are i packets at a link buffer at the beginning of a slot. The

remainder of the notation used in this subsection is the same with that used in Section III-A. Since there are two links (one forward and one internal) leading to a buffer, at most two continuing packets may arrive at a buffer during a slot. Thus, we have

$$b_i = b_i \Pr(\text{one arrival}) + b_{i-1} \Pr(\text{two arrivals}) + b_{i+1} \Pr(\text{no arrivals}), \qquad \text{for } i \neq 0, k. \tag{9}$$

In getting (9) we have used the approximating assumption A.2 in the following way: we have assumed that the events of one, two, or no arrivals at a buffer are independent of the arrival process at previous slots, and therefore of the number of packets already in the buffer. Calculating the probability of one, two, or no arrivals of continuing packets at a link buffer, and substituting in (9) we obtain

$$b_i = 2b_i\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)\frac{\sum_{j=1}^{d-1} p_j}{2} + b_{i-1}\left(\frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2 + b_{i+1}\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2, \; i = 1, \ldots, k-1. \tag{10}$$

The equations that give b_0 and b_k are slightly different:

$$b_0 = 2b_0\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)\frac{\sum_{j=1}^{d-1} p_j}{2} + (b_0 + b_1)\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2, \tag{11}$$

and

$$b_k = 2b_k\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)\frac{\sum_{j=1}^{d-1} p_j}{2} + (b_k + b_{k-1})\left(\frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2. \tag{12}$$

Letting

$$\theta = p_d + e = 1 - \sum_{j=1}^{d-1} p_j,$$

equations (10)-(12) can be rewritten as

$$b_i = b_i\frac{1-\theta^2}{2} + b_{i-1}\left(\frac{1-\theta}{2}\right)^2 + b_{i+1}\left(\frac{1+\theta}{2}\right)^2, \quad i = 1, 2, \ldots, k-1, \tag{13}$$

$$b_0 = b_0\frac{1-\theta^2}{2} + (b_0 + b_1)\left(\frac{1+\theta}{2}\right)^2, \tag{14}$$

and

$$b_k = b_k\frac{1-\theta^2}{2} + (b_k + b_{k-1})\left(\frac{1-\theta}{2}\right)^2. \tag{15}$$

The quadratic equation corresponding to the second-order recursion of (13) is

$$\rho^2\left(\frac{1+\theta}{2}\right)^2 - \rho\left(\frac{1+\theta^2}{2}\right) + \left(\frac{1-\theta}{2}\right)^2 = 0,$$

which has roots

$$\rho_1 = 1$$

and

$$\rho_2 = \left(\frac{1-\theta}{1+\theta}\right)^2.$$

The solution to the recursion is of the form

$$b_i = \alpha + \beta\left(\frac{1-\theta}{1+\theta}\right)^{2i}, \qquad i = 0, 1, \ldots k-1,$$

for some constants α and β. Taking into account (14) and (15) we obtain after some algebraic manipulation that

$$b_i = b_0\left(\frac{1-\theta}{1+\theta}\right)^{2i}, \qquad i = 0, 1, \ldots, k. \tag{16}$$

To verify this equation, use it to express b_i in terms of b_0 in (13)-(15), and see that these equations hold identically for all θ. Since

$$\sum_{i=0}^{k} b_i = 1$$

we finally get that

$$b_0 = \frac{1 - \left(\frac{1-\theta}{1+\theta}\right)^2}{1 - \left(\frac{1-\theta}{1+\theta}\right)^{2k+2}}. \tag{17}$$

For $k = 0$ (unbuffered case) (17) gives $b_0 = 1$ as expected. For infinite buffer space we have

$$b_0 = 1 - \left(\frac{1-\theta}{1+\theta}\right)^2, \quad \text{for } k = \infty. \tag{18}$$

The probability that a buffered packet is of type $i-1$, for $2 \leq i \leq d$, is proportional to p_{i-1}, because all the packets have the same priority during conflicts. Therefore,

$$\Pr(\text{buffered packet is of type } i-1) = \frac{p_{i-1}}{\sum_{i=1}^{d-1} p_i}.$$

A packet of type $i = 2, 3, \ldots, d$ transmitted over a link is either a continuing packet or a packet that was buffered. Recall that continuing packets have priority over buffered packets. The probability that a continuing packet is transmitted over a link is given by (2). The probability that a buffered packet is transmitted as an i-type packet is the product of three probabilities: i) the probability that no continuing packet requests the link, ii) the probability that the buffer is nonempty, and iii) the probability that the packet at the head of the buffer

is of type $i-1$. Using the approximating assumptions A.1 and A.2, we get

$$p_i = p_{i-1}\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{4}\right) + \left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2 \cdot (1-b_0)\frac{p_{i-1}}{\sum_{i=1}^{d-1} p_i}$$
$$= p_{i-1}\left(\frac{3+\theta}{4}\right) + \left(\frac{1+\theta}{2}\right)^2 (1-b_0)\frac{p_{i-1}}{1-\theta}, \qquad i > 1, \tag{19}$$

where the first term accounts for packets that are received and transmitted at the immediately following slot, and the second term accounts for packets that were buffered. Since new packets are accepted in the network only when there are no continuing or buffered packets, we have

$$p_1 = p_0 b_0 \left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2 = p_0 b_0 \left(\frac{1+\theta}{2}\right)^2. \tag{20}$$

A link remains idle if there are no continuing, buffered or new packets that want to use it. Thus,

$$e = (1-p_0)b_0\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2 = (1-p_0)b_0\left(\frac{1+\theta}{2}\right)^2.$$

Equations (19) and (20) give

$$p_d = \frac{p_0 b_0 (1+\theta)^2}{4^d}\left(3+\theta+(1-b_0)\frac{(1+\theta)^2}{1-\theta}\right)^{d-1}. \tag{21}$$

Adding the last two equations, substituting $\theta = p_d + e$, and solving for p_0 we finally obtain

$$p_0 = \frac{b_0(1+\theta)^2 - 4\theta}{b_0(1+\theta)^2 - \frac{b_0(\theta+1)^2}{4^{d-1}}\left(3+\theta+(1-b_0)\frac{(1+\theta)^2}{1-\theta}\right)^{d-1}}, \tag{22}$$

where b_0 is given by (17). Equations (21) and (22) give the relationship between p_d and p_0 in parametric form with parameter θ. The throughput per node is

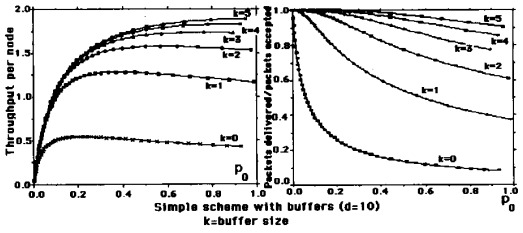
$$R = 2dp_d.$$

In the case of infinite buffer space, b_0 is given by (18), and (21) is simplified to

$$p_d = \theta p_0, \qquad \text{for } k = \infty.$$

As $k \to \infty$, (22) takes the indeterminate form $0/0$. By using L' Hospital's rule (alternatively, we can use (19) to prove that $p_i = \theta p_0$, $i = 1, 2, \ldots, d$, etc) we get after some calculations that

$$p_0 = \frac{1-\theta}{\theta(d-1)}, \qquad \text{for } k = \infty.$$



(a) (b)

Fig. 4. Results obtained for the simple scheme for $d = 10$. (a) illustrates the throughput per node as a function of p_0, for various buffer sizes k. (b) illustrates the average ratio of the number of packets delivered to their destination over the number of packets accepted in the network as a function of p_0, for various buffer sizes.

Combining the last two equations and using the fact $R = 2dp_d$ we obtain

$$R = \frac{2dp_0}{1+p_0(d-1)}, \qquad \text{for } k = \infty.$$

For $p_0 = 1$ and infinite buffer space, R is equal to two packets per node as expected.

In Fig. 4(a) we have plotted the throughput R per node as a function of p_0 for several buffer sizes k. We can see from this figure that buffer space for two or three packets per link is adequate to achieve throughput close to that of the infinite buffer case. Note also from Fig. 4(a) that for $k \geq 2$ the throughput of the (10-dimensional) hypercube for $p_0 = 1$ is not significantly smaller than the maximum throughput, and, therefore, no mechanism for controlling the rate at which the nodes transmit is necessary. This was found to be true for moderate values of d; in the following subsection we show that the value of p_0 that maximizes throughput is $\Theta(1/d^{1+\frac{1}{2k+1}})$, and therefore the value of k after which no control mechanism is needed depends on d. In Fig. 4(b), we have plotted the average ratio p_d/p_1 of number of packets delivered to their destination over the number of packets accepted in the network. We have chosen to illustrate the ratio p_d/p_1 instead of the ratio p_d/p_0, because the loss of packets inside the network has a worse effect than the blocking of packets at the input.

C. Asymptotic Behavior of the Thoughput

In this section we will examine the asymptotic behavior of the throughput of the hypercube for a fixed value of the buffer size k, as the number of nodes increases.

Combining (20), (21), and (17), we obtain after some calculations that

$$\frac{p_d}{p_1} = \left(\frac{3+\theta+\frac{(1+\theta)^2}{1-\theta}\cdot\frac{\left(\frac{1-\theta}{1+\theta}\right)^2 - \left(\frac{1-\theta}{1+\theta}\right)^{2k+2}}{1-\left(\frac{1-\theta}{1+\theta}\right)^{2k+2}}}{4}\right)^{d-1}. \tag{23}$$

We define

$$y = \frac{1-\theta}{1+\theta}. \tag{24}$$

Then (23) can be rewritten in terms of the new parameter as

$$\frac{p_d}{p_1} = \left(1 - \frac{y^{2k+1} - y^{2k+3}}{2(1+y)(1-y^{2k+2})}\right)^{d-1}. \quad (25)$$

As the free parameter θ ranges continuously from some small number to one, y takes all the values from zero to some number close to one. We choose

$$y = d^{-\frac{1}{2k+1}}, \quad (26)$$

which is clearly a legitimate value for y. This value of y corresponds to fixed values for p_0 and p_d; the corresponding value of the throughput can serve as a lower bound on the maximum achievable throughput. As $d \to \infty$, the right-hand side of (25) tends to a positive constant $c = e^{-1/2}$, that is

$$\lim_{d\to\infty} \frac{p_d}{p_1} = c > 0, \qquad \text{for } y \text{ chosen according to (26).} \quad (27)$$

For y given by (26) we have

$$\sum_{i=1}^{d-1} p_i = 1 - \theta = \frac{2y}{1+y} = \Theta\left(d^{-\frac{1}{2k+1}}\right).$$

Since

$$\frac{p_d}{p_1} \le \frac{p_i}{p_1} \le 1, \qquad \text{for } i = 1, 2, \ldots, d-1,$$

we have in view of (27) that $p_i = \Theta(p_d)$, for $i = 1, 2, \ldots, d$, and

$$p_d = \Theta\left(\frac{1}{d^{1+\frac{1}{2k+1}}}\right), \qquad \text{for} y \text{ given by (26).}$$

Therefore, the maximum total throughput H of the hypercube is

$$H = 2dNp_d = \Omega\left(\frac{N}{d^{\frac{1}{2k+1}}}\right). \quad (28)$$

In the case where $y = o(d^{-\frac{1}{2k+1}})$, we can similarly find that $p_d = o\left(d^{-1-\frac{1}{2k+1}}\right)$, and $H = o\left(N/d^{\frac{1}{2k+1}}\right)$. In the case where $y = \tilde{\Omega}(d^{-\frac{1}{2k+1}})$, with $\tilde{\Omega}$ standing for strictly larger order of magnitude, (25) gives $p_d/p_1 \to 0$ as $d \to \infty$. Based on these observations and (28) we conclude that

$$H = \Theta\left(\frac{N}{d^{\frac{1}{2k+1}}}\right). \quad (29)$$

For $y = o(d^{-\frac{1}{2k+1}})$ (or else, $p_0 = o\left(1/d^{1+\frac{1}{2k+1}}\right)$, i.e., small load), we expect almost all of the packets to succesfully reach their destination. For $y = \Theta(d^{-\frac{1}{2k+1}})$ [or else, $p_0 = \Theta(1/d^{1+\frac{1}{2k+1}})$] we expect a constant fraction of the packets to reach their destination. For $y = \tilde{\Omega}(d^{-\frac{1}{2k+1}})$ (or else, $p_0 = \tilde{\Omega}(1/d^{1+\frac{1}{2k+1}})$), almost all the packets are dropped (for large d).

The dependence of the total throughput on the buffer size per link k is an interesting one. In the case of no buffers, (29) gives $H = \Theta(N/d)$. Having buffer space for $k = 1$ packet (in addition to the one being transmitted) increases the throughput by a factor of $d^{\frac{2}{3}}$ over the unbuffered case, which is a significant gain. Increasing k further gives diminishing returns. For infinite buffer space ($k = \infty$) we have $H = \Theta(N)$ as expected (in fact we then have $H = 2N$).

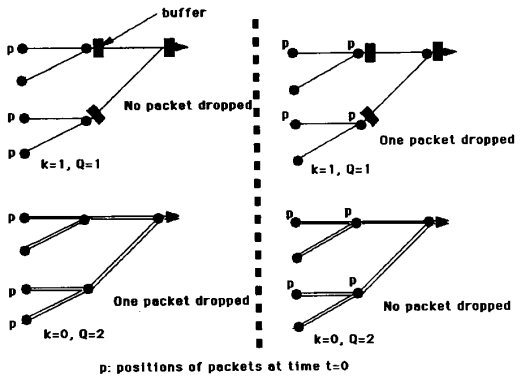
It is interesting to compare (29) with the throughput of another routing scheme, which we will call *greedy scheme*, in a *Q-dilated hypercube*. A Q-dilated hypercube is a hypercube whose links have capacity Q, that is, each link can be used for up to Q packets. In the greedy scheme, packets traverse the hypercube dimensions in descending order, *always* starting with dimension d. The node model assumed for this scheme is the same as in Fig. 1 (but the greedy scheme is different from the simple or the priority scheme where each packet starts transmission from a random dimension). The greedy scheme has been analyzed by Koch in [11] and [12] (the case of unit capacity was earlier analyzed in [20] and [15]; see also [16, pp. 612–620]). The maximum total throughput of the greedy scheme in a Q-dilated hypercube is given by

$$H = \Theta\left(\frac{N}{d^{\frac{1}{Q}}}\right). \quad (30)$$

Comparing (29) and (30) it seems (modulo our approximating assumptions, since Koch's result is rigorously obtained) that the dependence of the throughput on the buffer size k is stronger than the dependence on the capacity Q. One might think of attributing this to the fact that the simple scheme achieves uniform utilization of the hypercube links, while the greedy scheme does not: packets in the greedy scheme start transmission from a fixed dimension and some of them have already been dropped by the time they reach the links of the other dimensions. This argument, although correct, does not seem to explain the difference in the dependencies of the throughput on k and Q, because the unbuffered and uncapacitated hypercube ($k = 0$ and $Q = 1$) achieves throughput of the same order of magnitude ($\Theta(N/d)$) for both schemes. Therefore, the most plausible explanation is that increasing the buffer space by a constant factor results in substantially greater improvement of the throughput than increasing the capacity of the links by the same factor. This means that less packets are dropped when we have k buffer spaces per link than when we have k wires per link. This result should hold on the average, since one can devise scenaria where buffer space helps most, and scenaria where capacity helps most [17]. For example, Fig. 5(a) shows a scenario where having more buffer space per link results in fewer packet losses than having more capacity per link, while Fig. 5(b) shows a scenario where the opposite is true. One should also note here, that although buffer space may be more important than link capacity when the objective is the throughput, the situation will probably be different when the objective is the average packet delay.

IV. The Priority Scheme

In this section we will evaluate the throughput of the priority scheme. Recall that in the priority scheme the packets that have travelled more have priority when they compete for a forward or an internal link. If two packets that have travelled equal distances request a link at the same time, one of them is transmitted with equal probability. New packets are admitted



(a) (b)

Fig. 5. In scenario (a) more packets are dropped when links have capacity two and no buffer space than when they have capacity one and buffer space for one packet in addition to the packet under transmission. In scenario (b) the opposite is true.

at a link buffer only if there are no continuing or buffered packets claiming the link. We analyze the unbuffered priority scheme in Section IV-A, and the buffered priority scheme in Section IV-B.

A. Analysis of the Priority Scheme Without Buffers

In this section we analyze the unbuffered case. Since packets that have travelled more have priority when claiming the same link, the packets of type i are not affected by the existence of packets of type $1, 2, \ldots, i-1$ or new packets. Let p_i and e be as defined in Section III–A. Consider a link l, and the two links l_1 and l_2 leading to it. Using again the approximating assumption A.1 and reasoning as in Section III–A, we find that

$$p_i = p_{i-1} \Pr(\text{packet of type } i-1 \text{ not dropped}). \tag{31}$$

A packet $\mathcal{P}$ of type $i-1$ that arrives on link l_1 is dropped if and only if one of the following two events happens:

Event 1: A packet of type $i, i+1, \ldots, d-1$ was transmitted on link l_2 during the previous slot and its routing tag was such that l was chosen. This happens with probability

$$\frac{1}{2}\sum_{j=i}^{d-1} p_j, \tag{32}$$

or Event 2.

Event 2: A packet $\mathcal{Q}$ of type $i-1$ was transmitted on link l_2 during the previous slot, its routing tag was such that l was selected, and $\mathcal{Q}$ was chosen (with probability 0.5) instead of $\mathcal{P}$. The probability of this event is

$$\frac{p_{i-1}}{4}. \tag{33}$$

Since Events 1 and 2 are mutually exclusive, (31)-(33) give

$$p_i = p_{i-1}\left(1 - \frac{1}{2}\sum_{j=i}^{d-1} p_j - \frac{p_{i-1}}{4}\right), \qquad i = 2, 3, \ldots, d. \tag{34}$$

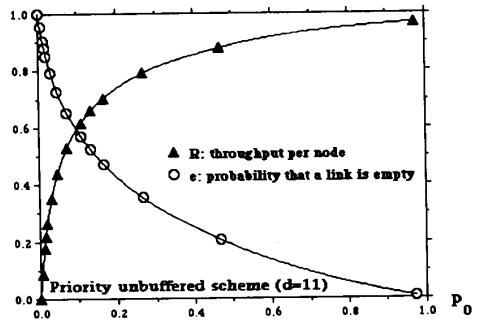


Fig. 6. Priority scheme without buffers ($d = 11$).

For p_1 we have a slightly different equation:

$$p_1 = p_0\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2, \tag{35}$$

where p_0 is the probability that a new packet is available and

$$\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2$$

is the probability that no packet that wishes to use link l arrives on l_1 or l_2. The probability that a link is idle can be seen to be

$$e = (1-p_0)\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2. \tag{36}$$

For a particular value of p_d we can use (34)-(36) to find the corresponding values of $p_{d-1}, p_{d-2}, \ldots, p_0$. This is done by solving (34) with respect to p_{i-1}, and keeping the solution that gives a legitimate probability distribution (the other solution corresponds to $p_{i-1} > 1$):

$$p_{i-1} = 2 - \sum_{j=i}^{d-1} p_j - \sqrt{\left(2 - \sum_{j=i}^{d-1} p_j\right)^2 - 4p_i}, \quad i = 2, 3, \ldots, d. \tag{37}$$

For p_0 we have from (35) that

$$p_0 = \frac{p_1}{\left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2}. \tag{38}$$

Giving a value to the throughput per link p_d we can obtain the corresponding value of the access probability p_0 by using the backward recursion of (37). The remaining performance parameters of interest (for example, the probability e that a link is idle, and the mean throughput per node $R = 2dp_d$) can then be computed easily. Fig. 6 illustrates the results obtained for $d = 11$.

Let $p_d = F(p_0)$, where the function F is not known in closed form. We already presented a simple recursion to compute $p_0 = F^{-1}(p_d)$ for each p_d. We can prove by induction that F^{-1} (and therefore F) is monotonically increasing. This shows that F is 1-1 [this is also evident from the fact that (34) has a unique solution in the interval $[0, 1]$], and the maximum of p_d and R occurs for $p_0 = 1$. The

monotonicity of the throughput with respect to the offered load is a desirable characteristic of the priority scheme. It indicates that if we superimpose on it a retransmission scheme, then the system will behave well when congestion arises without having to control the rate at which the sources inject packets. By contrast, for the simple scheme the relationship between p_d and p_0 was not 1-1, and the maximum throughput was attained for p_0 less than one.

The recursion of (34) turns out to be similar to a recursion that arises in the analysis of a priority deflection scheme for the shuffle-ring network by Krishna and Hajek [13]. A question that is motivated by the discussion in [13] is whether or not the collision resolution rule of the priority scheme is optimal under approximation A.1 among rules that use only local information. The following lemma shows that this is indeed the case. It can be proven by using the tools developed by Krishna (of [14, Theorem 2.4]). The proof is omitted.

Lemma 1: Under the node model of Fig. 1, and under approximation A.1, the collision resolution rule of the priority scheme achieves optimal throughput over all rules that are based on the distance of the packet to the destination.

B. Analysis of the Priority Scheme with Buffers

We now evaluate the throughput of the priority scheme when there is buffer space at each link. We assume that each buffer can hold up to k packets in addition to the packet under transmission. When two packets arrive at a node and request the same link, one of them is transmitted over the link, and the other is either stored (if there is enough space in the buffer), or is dropped. The packet that is transmitted is the one that has crossed more forward and internal links with ties resolved at random. The analysis of Section IV-A corresponds to the case $k = 0$ of this subsection. Continuing packets have priority over buffered packets or new packets when claiming a link. New packets are admitted in the network only if the buffer where they enter is completely empty. The buffers are FIFO, and packets in the buffer that have higher priority do *not* overtake packets of lower priority that are in front of them. If we were using a priority discipline within the buffer then we could probably obtain higher throughput, but the system would be more difficult to implement and analyze.

In order to analyze the buffered priority scheme we make again the approximating assumptions A.1 and A.2.

Following the notation of Section III-B, we denote by b_i, $i = 0, 1, \ldots, k$, the probability that a link buffer contains i packets just before the beginning of a slot, and we define the parameter $\theta = p_d + e$. The probability b_i is given by (16) and (17), which we repeat here for ease of reference:

$$b_i = b_0 \left(\frac{1-\theta}{1+\theta} \right)^{2i}, i = 0, 1, \ldots, k, \tag{39}$$

and

$$b_0 = \frac{1 - \left(\frac{1-\theta}{1+\theta} \right)^2}{1 - \left(\frac{1-\theta}{1+\theta} \right)^{2k+2}}, \tag{40}$$

with $\theta \in (0, 1)$.

The probability that a buffered packet is of type $i-1$, for $2 \leq i \leq d$, is proportional to

$$p_{i-1} \left(\frac{p_{i-1}}{2} + \sum_{j=i}^{d-1} p_j \right).$$

This is because only conflicts with packets of types $i-1, i, \ldots, d$ can cause a packet of type $i-1$ to be buffered (and conflicts with packets of type $i-1$ are resolved at random). Therefore, a buffered packet is of type $i-1$ with probability

$$\frac{p_{i-1}\left(\frac{p_{i-1}}{2} + \sum_{j=i}^{d-1} p_j \right)}{\sum_{l=1}^{d-1} p_l \left(\frac{p_l}{2} + \sum_{j=l+1}^{d-1} p_j \right)} = \frac{2p_{i-1}\left(\frac{p_{i-1}}{2} + \sum_{j=i}^{d-1} p_j \right)}{\left(\sum_{l=1}^{d-1} p_l \right)^2} = \frac{2p_{i-1}\left(\frac{p_{i-1}}{2} + \sum_{j=i}^{d-1} p_j \right)}{(1-\theta)^2}.$$

The probability that a packet of type $i > 1$ is transmitted over a link is

$$\begin{aligned} p_i &= p_{i-1}\left(1 - \frac{\sum_{j=i}^{d-1} p_j}{2} - \frac{p_{i-1}}{4} \right) + \left(1 - \frac{\sum_{j=i}^{d-1} p_j}{2} \right)^2 \\ &\quad \cdot (1-b_0) \frac{2p_{i-1}\left(\frac{p_{i-1}}{2} + \sum_{j=i}^{d-1} p_j \right)}{(1-\theta)^2} \\ &= p_{i-1}\left(1 - \frac{\sum_{j=i}^{d-1} p_j}{2} - \frac{p_{i-1}}{4} \right) + \frac{(1+\theta)^2}{2(1-\theta)^2} \\ &\quad \cdot (1-b_0) p_{i-1} \left(\frac{p_{i-1}}{2} + \sum_{j=i}^{d-1} p_j \right), \qquad i > 1 \end{aligned} \tag{41}$$

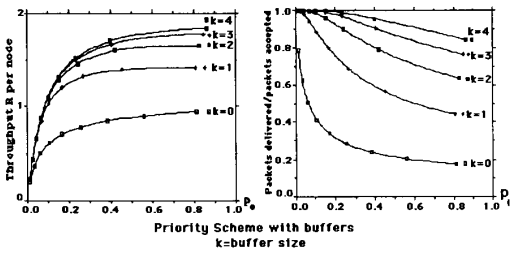
where the first term in the above summands is the same with the right side of (34) and accounts for packets that were received during the preceding slot, and the second term accounts for packets that were buffered. Since new packets are accepted in the network only when there are no continuing or buffered packets, we have

$$p_1 = p_0 b_0 \left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2} \right)^2 = p_0 b_0 \left(\frac{1+\theta}{2} \right)^2. \tag{42}$$

The probability that a link is idle is given by

$$e = (1-p_0) b_0 \left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2} \right)^2 = (1-p_0) b_0 \left(\frac{1+\theta}{2} \right)^2.$$

We used a Gauss-Seidel type of algorithm to find numerically p_i's that satisfy (41) and (42). We did not prove the uniqueness of the solutions obtained; however, we tried a number of different initial conditions always arriving at the same solution. The results obtained are shown in Fig. 7. Fig. 7(a) illustrates the throughput $R = 2dp_d$ per node as a function of p_0 for several buffer sizes k. Fig. 7(b) illustrates the ratio p_d/p_1, that is, the steady-state fraction of packets accepted in the network that arrive at their destination, as a function of the offered load for several values of the buffer size k. Fig. 7 suggests that little buffer space is adequate in practice to achieve satisfactory throughput, and low probability of packet



(a) (b)

Fig. 7. Results obtained for the priority scheme for $d = 11$. (a) illustrates the throughput per node as a function of p_0, for various buffer sizes k. (b) illustrates the average ratio of the number of packets delivered to their destination over the number of packets accepted in the network as a function of p_0, for various buffer sizes.

losses (we are primarily interested in the load region where $p_0 < 1/d$; higher values of p_0 correspond to more than two new packets per node attempting to enter the network on the average).

Comparing Figs. 4(a) and 7(a) we see that the priority rule increases the throughput significantly. The priority rule is designed to decrease the waste of resources caused from packets being transmitted several times and then being dropped. Note also that the priority rule is simple and it can be implemented entirely in hardware. A similar priority rule for deflection routing will be examined in Section VI.

V. Quality Of The Approximations, And Simulation Results

The unbuffered simple scheme is similar to the greedy routing scheme in a *wrapped butterfly*. A wrapped butterfly is obtained by merging the first and the last levels of an ordinary butterfly into a single level (see Fig. 8). Each link of the wrapped butterfly is assumed to have buffer space only for the packet under transmission. All the nodes of the wrapped butterfly are seen as potential sources or destinations. A new packet is available to enter at a link (including the links of the intermediate stages) with probability p_0, and has as destination a node at the same stage with its source. The internal links of the hypercube node model of Fig. 1 correspond to *straight links* of the wrapped butterfly, that is, links connecting a node of some stage with the node of the next stage that has the same binary representation. Similarly, the forward links of the hypercube node model correspond to the *cross links* of the wrapped butterfly, that is, links connecting a node of a stage with the node of the next stage whose binary representation differs in one bit. The simple scheme corresponds to the greedy scheme in a wrapped butterfly, where packets follow the unique shortest path to their destination, while the priority scheme corresponds in a natural way to a priority greedy scheme in a wrapped butterfly.

Consider a link l, and the two links l_1 and l_2 that lead to it. We want to investigate the quality of the approximation A.1 used in the analyses of the unbuffered simple and priority schemes. In particular, we are interested in the following questions.

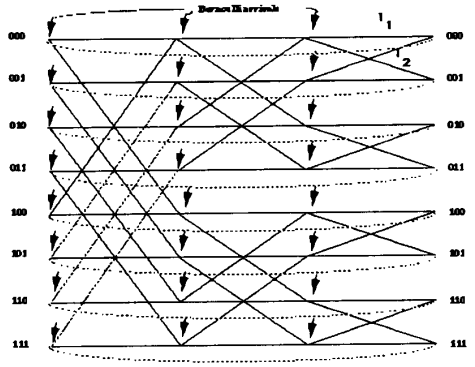


Fig. 8. A wrapped butterfly. The direction of the links, which is from left to right, is not shown for simplicity. The nodes of the last stage are the same with the nodes of the first stage. The destination of a packet has to be at the same stage with its source.

1) Are the packet arrivals during a *particular* slot T on link l_1 independent of the packet arrivals on link l_2 during the same slot?

2) If they are dependent, where does the dependence come from, and how strong is it?

The following lemma, proved in [28], gives a partial answer to these questions.

Lemma 2: Approximating assumption A.1 is weaker than the following assumption: "Events that take place at time T are independent from events that take place prior to time $T - d$."

Lemma 2 suggests that the dependence between an event on link l_1 and an event on link l_2 at a given time is weak, and thus assumption A.1 is a fairly accurate approximation. In fact, the larger d is the better the approximation is expected to be. The above lemma does not assume any particular way of resolving conflicts, and it therefore holds for both the simple and the priority unbuffered schemes.

Another way to see that the previous dependence is weak is the following. Given that a packet of a particular type is transmitted on the straight link l_1, the *a posteriori* probability that a packet is transmitted on a cross link l of dimension d is the same for all l. For any cross link l let

$$\Delta p(l) = \Pr(l \text{ has a packet} \mid l_1 \text{ has a packet of type } i) - \Pr(l \text{ has a packet})$$

be the difference between the *a priori* and the *a posteriori* probabilities. The smaller $\Delta p(l_2)$ is, the more accurate the approximation A.1 is. Observe that

$$\Delta p(l) = \Delta p(l_2) = \Delta p \text{ for all cross links } l \text{ of dimension } d.$$

The average total flow of packets through the links of dimension d conditioned on the presence of a packet on link l_1 differs by $N \cdot \Delta p$ units from its a priori value. It is reasonable to expect that the knowledge that a particular link l_1 has a packet will not significantly change the average flow through links of dimension d because this flow is a quantity of a global nature. Thus, Δp must be small (we believe that $\Delta p = O(1/N)$).

We simulated the unbuffered simple scheme for various network sizes, and several values of p_0. The difference between

p_0	0.9983	0.9288	0.8045	0.6972	0.6042	0.5234	0.4871	0.3642	0.3142	0.2915	0.2145	0.1982	0.1094	0.0030
$R^{(a)}$	0.6325	0.6401	0.6539	0.6657	0.6754	0.6827	0.6852	0.6886	0.6859	0.6831	0.6628	0.6552	0.5712	0.0448
$R^{(s)}$	0.6331	0.6401	0.6540	0.6650	0.6744	0.6824	0.6843	0.6883	0.6852	0.6828	0.6621	0.6557	0.5721	0.0446

Fig. 9. Analytical and simulation results obtained for the unbuffered simple scheme for hypercube dimension $d = 8$. The throughput per node found by the analysis (or the simulation) is denoted by $R^{(a)}$ (or $R^{(s)}$, respectively). The first row corresponds to the corresponding probabilities of access p_0.

p_0	0.931384	0.566517	0.302901	0.199937	0.169829	0.144199	0.103110	0.086444	0.052758
$R^{(a)}$	1.493738	1.477039	1.345483	1.189335	1.116160	1.038224	0.871355	0.783858	0.557855
$R^{(s)}$	1.451239	1.433139	1.354165	1.162777	1.092926	1.020776	0.861196	0.777389	0.554911

Fig. 10. Simulation and analytical results for the buffered simple scheme for hypercube dimension $d = 7$ and buffer size $k = 1$. The throughput per node found by the analysis (or the simulation) is denoted by $R^{(a)}$ (or $R^{(s)}$, respectively). The first row corresponds to the corresponding probabilities of access p_0.

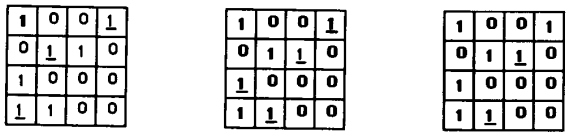
the analytical and the simulation results has been consistently negligible for all network sizes and values of p_0 (see Fig. 9 for $d = 8$). This is a further indication that the parametric equations obtained in Section III-A are very accurate.

We have also performed simulations for the buffered simple scheme. The results obtained from the simulations were found to be within 3% from the analytical results. Recall that the analyses of the buffered schemes use approximating assumption A.2 in addition to the approximating assumption A.1 used by the analyses of the unbuffered schemes, and were, therefore, expected to be less accurate. Fig. 10 illustrates the results obtained for $d = 7$ and $k = 1$.

VI. Comparison with Deflection Routing

In this section we describe two deflection schemes, called the *simple nonwasting deflection* and the *priority nonwasting deflection* schemes. The simple deflection scheme has been analyzed by Greenberg and Hajek [9] using approximations. Hajek [10] has also examined the evacuation time of the priority scheme. The throughput of both schemes has also been analyzed independently by Varvarigos [27] through the formulation of an approximate Markov chain. We will present simulation results on the throughput of the deflection schemes, in order to compare the throughput of the two deflection schemes to that of the priority scheme of Section IV. We first describe the deflection schemes, and the stochastic model used in their analysis, and then present and discuss the results obtained.

Each node has a queue that can hold up to d packets. During a slot each node transmits all the packets of its queue, either by transmitting them on their *preferred links*, that is, links that take the packets closer to their destination, or by simply transmitting them on any available link. We assume that new packets are always available, and for every packet that exits the network at some node a new packet enters the network at the same node. The destinations of the new packets are uniformly distributed over all nodes, except for their origin. Under this model the hypercube is a closed network and every node always has exactly d packets.



(a) (b) (c)

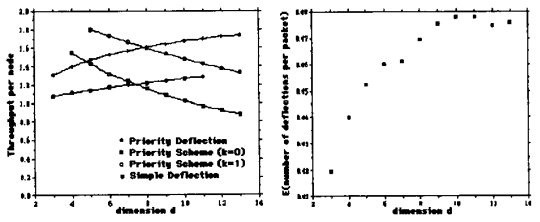
Fig. 11. The figure indicates some possible assigments of packets residing at a node to outgoing linksof the node. Each row corresponds to the routing tag of a packet, while each column corresponds to an outgoing link. We indicate the assignment of a packet to a particular link is indicated by underlining the corresponding unit entry of its routing tag. Cases (a) and (b) correspond to nonwasting assignments, while case (c) corresponds to a wasting assignment.

To describe the deflection schemes some definitions are useful. A *partial* switching assignment is a 1-1 match between packets and preferred links, where each packet (or link) is matched to at most one link (or packet, respectively). A *full* switching assignment is a match between all the d packets residing at the node and the d outgoing links of the node. A partial switching assignment is *wasting* if there exists a packet that has not been assigned to a preferred link, although one of its preferred links is free. By transmitting the packet on this link the number of packets that are sent towards their destinations is increased by one and the assignment remains feasible. In a *nonwasting* switching assignment such a situation is not allowed. In Fig. 11, both a and b are nonwasting switching assignments, while c is not.

A simple procedure to obtain a nonwasting assignment is the following. At each slot, an order (called *processing order*) of the packets stored at a node is found. The packets are then picked in that order, and each of them is assigned to one of its preferred links, provided that this link has not been assigned to any of the preceding packets. If more than one unassigned preferred links exist, one of them is chosen at random.

A deflection scheme consists of two phases. During the first phase, called the *nonwasting phase*, a nonwasting partial switching assignment is found. This assignment matches some of the packets with an equal number of links. The assignments made depend on the order in which the packets are processed. In the simple nonwasting deflection scheme the processing order is random with all orders (permutations) being equally probable. This scheme is similar to the one analyzed (approximately) by Greenberg and Hajek in [9]. In the priority nonwasting deflection scheme the processing order is found as follows. The packets are partitioned in priority classes, so that the ith priority class consists of the packets that are currently located at a distance i from their destination. Packets that are closer to their destination precede in order packets that are farther from their destination; the order of the packets within the same class is random.

The partial assignment found in the nonwasting phase will cover only z of the d packets with $z \leq d$. In the second phase, called *deflection phase*, the partial assignment is extended to a full assignment by arbitrarily mapping the $d - z$ unassigned packets to the $d - z$ unassigned outgoing links. The $d - z$ packets that are not transmitted over preferred links increase their distance to the destination. We will refer to such events as *packet deflections*. Every time a packet is deflected, the number of links it has to traverse increases by two.



(a) (b)

Fig. 12. (a) illustrates the throughput per node of (1) the simple nonwasting deflection scheme, (2) the priority nonwasting deflection scheme, (3) the unbuffered priority scheme, and (4) the buffered ($k = 1$) priority scheme. The results assume that new packets are always available to enter the network when there is an idle link. (b) illustrates the average number of deflections suffered by a packet in the priority nonwasting deflection scheme.

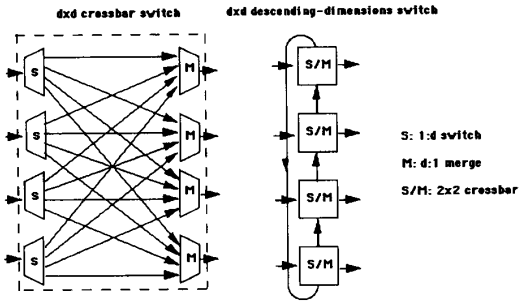


Fig. 13. A $d \times d$ crossbar switch (required by deflection schemes), a $d \times d$ descending-dimensions switch (required by the simple and the priority schemes), and the modules out of which they are composed. The number of wires of a descending-dimensions switch is only $\Theta(d)$. A crossbar router is larger and slower, and results in a slower network (the processing time at a node and the clock cycle is larger). The switching assignments possible with the descending-dimensions switch are of course more restricted, and suffer from internal message collisions (the collisions on the internal links of the node model of Fig. 1). Since the descending-dimensions switch uses simple 2:2 switch/merge switches, it can be made to operate very quickly, which may offset the degradation in the performance due to the restrictions in the routing algorithm (see [5]).

The rationale behind the priority deflection scheme is the following. If the processing order is random the packets that are at distance one from their destination have a higher probability of being deflected than packets at distance $i > 1$ from their destination. For example, a packet that is one hop away from its destination has only one preferred link, and there is a large probability that this link will have already been assigned when the packet is processed. In contrast, a packet at distance $i > 1$ from its destination has i preferred links, and will probably not be hurt if some of its preferred links have been assigned to other packets. A packet at distance d from its destination is never deflected, and it is logical that it should be assigned last.

Fig. 12(a) illustrates the simulation results obtained for the simple and the priority nonwasting deflection schemes, together with the analytical results obtained for the priority scheme (with buffer size $k = 0$ and $k = 1$) of Section IV. Note that as the dimension of the hypercube increases the throughput per node of the deflection schemes increases. However, for small dimensions the priority scheme with $k = 1$ outperforms deflection routing. Thus, the priority scheme may be preferable for small hypercube dimensions (the crossover point is $d = 8$). If we take into account that the switch used at each node by the priority scheme is simpler, faster, and less expensive than the crossbar switch used by the deflection schemes (see Fig. 13), then the priority scheme may be preferable for large hypercube dimensions as well. By increasing the buffer size of the priority scheme it becomes even more appealing (at the expense of additional hardware).

Let λ be the average throughput per node at steady-state (therefore, λN is the total throughput), and T be the average delay of a packet from the time it is accepted in the network until the time it arrives at its destination. Since the number of packets in the hypercube is constant and equal to Nd, Little's theorem gives $Nd = (\lambda N)T$, or

$$\lambda = \frac{d}{T}. \tag{43}$$

The average delay of a deflection scheme satisfies

$$T = \frac{d}{2}\frac{N}{N-1} + 2E(\text{Number of Deflections}), \tag{44}$$

where E(Number of Deflections) is the average number of deflections suffered by a packet. The first term of the right-hand side of (46) is the average delay of a packet when it is not deflected. The second term is due to the fact that every time a packet is deflected, its delay increases by two steps. Fig. 12(b) illustrates the average number of deflections suffered by a packet in the priority nonwasting deflection scheme, for various dimensions d of the hypercube.

An interesting observation concerning Fig. 12(b) is the following. As d increases, the average delay suffered by a packet also increases, but the $E($ Number of Deflections) seems to remain almost constant (between 0.42 and 0.48 for $d = 3, 4, \ldots, 13$). If the average number of deflections is bounded above by a constant for every d, then the average delay T of the priority deflection scheme will be $T = \frac{d}{2}\frac{N}{N-1} + O(1)$. If in addition, the higher moments of the average number of deflections are also $O(1)$, then the throughput λ of the priority deflection scheme, given from (45), will tend to two packets per node, which is the maximum possible for uniformly distributed destinations [for this to be true it would be enough for the average number of deflections to be $o(d)$]. We could not prove this by a rigorous analysis, so we leave it as a conjecture.

VII. Conclusions

We considered two different hypercube routing schemes, and evaluated their steady state throughput for various traffic loads. The schemes require simple, low-cost switches at the hypercube nodes, instead of crossbar switches. The results obtained were approximate, but very accurate as simulations indicate, and they were given in particularly interesting forms. One of the two schemes uses a priority rule to resolve conflicts over a link. The priority rule was found to increase the throughput significantly. For both routing schemes we examined the effect of the buffer size on the throughput, and

found that little buffer space is needed to achieve throughput close to that of the infinite buffer case. We also considered two deflection routing schemes, and evaluated their throughput using simulations. These schemes, which use crossbar switches at the nodes, have very satisfactory throughput; in fact, the priority deflection scheme is conjectured to have throughput asymptotically equal to the maximum possible.

REFERENCES

[1] J. T. Brassil, "Deflection routing in certain regular networks," Ph.D. dissertation, Univ. California, San Diego, 1991.

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

[3] D. P. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[4] A. Choudhury and V. O. K. Li, "Performance analysis of deflection routing in the manhattan street and minimum-distance networks," preprint.

[5] W. J. Dally, "Network and processor architecture for message-driven computers," in *VLSI and Parallel Computation*, R. Suaya and G. Birtwistle, Eds. San Mateo, CA: Morgan Kaufmann Publishers, 1990, pp. 140–222.

[6] D. M. Dias and J. R. Jump, "Analysis and simulation of buffered delta networks," *IEEE Trans. Comput.*, vol. C-30, pp. 273–282, Aug. 1981.

[7] S. J. Golestani, "Congestion-free communication in high-speed packet networks," *IEEE Trans. Commun.*, vol. 39, Dec. 1991.

[8] A. G. Greenberg and J. Goodman, "Sharp approximate models of adaptive routing in mesh networks, " in *Teletraffic Analysis and Computer Performance Evaluation*, O. J. Boxma, J. W. Cohen, and H. C. Tijms, Eds. Amsterdam, The Netherlands: Elsevier, 1986, pp. 255–270.

[9] A. G. Greenberg and B. Hajek, "Deflection routing in hypercube networks," *IEEE Trans. Commun.*, vol. 35, pp. 1070–1081, June 1992.

[10] B. Hajek, "Bounds on evacuation time for deflection routing, *Distrib. Comput.*, vol. 5, pp. 1–6, 1991.

[11] R. Koch, "Increasing the size of the network by a constant factor can increase performance by more than a constant factor," in *IEEE 29th Annu. Symp. Foundations of Computer Science*, Oct. 1988, pp. 221–230.

[12] ——, "An analysis of the performance of interconnection networks for multiprocessor systems," Ph.D. dissertation, Massachusetts Inst. of Technol., May 1989.

[13] A. Krishna and B. Hajek, "Performance of shuffle-like switching networks with deflection," in *Proc. IEEE INFOCOM'90*, June 1990, pp. 473–480.

[14] A. Krishna, "Communication with Few Buffers: Analysis and Design," Ph.D. dissertation, Dept. of ECE, Univ. of Illinois at Urbana-Champaign, Dec. 1990.

[15] C. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," *IEEE Trans. on Comput.*, vol. C-32, pp. 1091–1098, Dec. 1983.

[16] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*. San Mateo, CA: Morgan Kaufmann, 1992.

[17] ——, personal communication, 1992.

[18] N. F. Maxemchuk, "Comparison of deflection and store-and-forward techniques in the Manhattan street and shuffle-exchange networks," in *INFOCOM '89*, vol. 3, Apr. 1989, pp. 800–809.

[19] ——, "Problems arising from deflection routing: Live-lock, lock-out, congestion and message reassembly," in *Proc. NATO Workshop Architecture and High Perform. Issues of High Capacity Local and Metropolitan Area Networks*, France, June 1990.

[20] J. H. Patel, "Performance of processor-memory interconnection for multiprocessors," *IEEE Trans. on Comput.*, vol. C-30, pp. 545–556, Apr. 1981.

[21] P. Pippenger, "Parallel communication with limited buffers," in *Proc. 25th Annu. IEEE Symp. Foundations of Comput. Sci.*, 1984, pp. 127–136.

[22] G. Stamoulis, "Routing and performance evaluation in interconnection networks," Ph.D. dissertation, M.I.T., Report LIDS-TH-2035, May 1991.

[23] E. Upfal, "Efficient schemes for parallel communication," *J. ACM*, vol. 31, pp. 507–517, 1984.

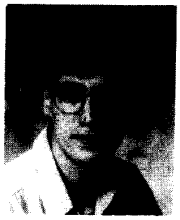
[24] L. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *Proc. 13th Annu. Symp. Theory of Comput.*, 1981, pp. 263–277.

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

[26] E. A. Varvarigos and D. P. Bertsekas, "A conflict sense routing protocol and its performance for hypercubes," MIT, Lab. for Inform. and Decision Syst. Rep. LIDS-P-2133, Sept. 1992, *IEEE Trans. Comput.*, to be published.

[27] E. A. Varvarigos, "Optimal communication algorithms for multiprocessor computers," M.S. thesis, Dep. of EECS, Rep. CICS-TH-192, Massachusetts Inst. Technol., 1990.

[28] ——, "Static and dynamic communication in parallel computing," Ph.D. dissertation, Dep. of EECS, Massachusetts Inst. Technol., Sept. 1992.



Emmanouel A. Varvarigos was born in Athens, Greece, in 1965. He received the Diploma in electrical engineering and computer science from the National Technical University of Athens, Greece, in 1988, and the M.S. degree in 1990, Electrical Engineer degree in 1991, and the Ph.D. degree in 1992 degrees in electrical engineering and computer science from the Massachusetts Institute of Technology.

In 1990 he conducted research at Bell Communications Research, Morristown, NJ. He is currently an Assistant Professor in the Electrical and Computer Engineering Department of the University of California, Santa Barbara. His research interests are in the areas of high-speed data networks, parallel and distributed computation, and wireless communications.

Dr. Varvarigos has received the first panhellenic prize in the Greek Mathematic Olympiad, and the Technical Chamber of Greece award .four times. He is a member of the Technical Chamber of Greece.

Dimitri P. Bertsekas received the combined B.S.E.E. and B.S.M.E. degrees from the National Technical University of Athens, Greece, in 1965, the M.S.E.E. degree from George Washington University in 1969, and the Ph.D. degree in system science from the Massachusetts Institute of Technology, Cambridge, in 1971.

He has held faculty positions with the Engineering-Economic Systems Department, Stanford University (1971–1974) and the Electrical Engineering Department of the University of Illinois, Urbana (1974–1979). He is currently Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology. He consults regularly with private industry and has held editorial positions in several journals. He was elected Fellow of the IEEE in 1983. He has done research in the areas of estimation and control of stochastic systems, linear, nonlinear and dynamic programming, data communication networks, and parallel and distributed computation, and has written numerous papers in each of these areas.

Dr. Bertsekas is the author of *Dynamic Programming and Stochastic Control* (New York: Academic, 1976), *Constrained Optimization and Lagrange Multiplier Methods* (New York: Academic, 1982), *Dynamic Programming: Deterministic and Stochastic Models* (Englewood Cliffs, NJ: Prentice-Hall, 1987), *Linear Network Optimization: Algorithms and Codes* (Cambridge, MA: M.I.T. Press, 1991), and co-author of *Stochastic Optimal Control: The Discrete-Time Case* (New York: Academic Press, 1978) *Data Networks* 1987, and *Parallel and Distributed Computation: Numerical Methods* (Englewood Cliffs, NJ: Prentice-Hall, 1989).