November 1992 LIDS-P-2149

# Performance of Hypercube Routing Schemes With or Without Buffering[1]

by

**Emmanouel A. Varvarigos[2] and Dimitri P. Bertsekas[3]**

**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 simulation results indicate, and are given in particularly interesting forms. We find that little buffer space (between one and three packets per link) is necessary and adequate to achieve throughput close to that of the infinite buffer case. We also consider two deflection routing schemes, called the *simple deflection* and the *priority deflection* schemes. We evaluate their throughput using simulations, and compare it to that of the priority scheme.

[1] Research supported by NSF under Grant NSF-DDM-8903385 and by the ARO under Grant DAAL03-92-G-0115.

[2] Department of Electrical and Computer Engineering, UCSB, Santa Barbara, CA 93106.
[3] Laboratory for Information and Decision Systems, M.I.T, Cambridge, Mass. 02139.

# 1. INTRODUCTION

The traffic environment under which the routing schemes of this paper are analyzed is stochastic. 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 [Val82], and Valiant and Brebner [VaB81] for the hypercube network. They were extended to networks of constant degree by Upfal [Upf84], and they were improved to achieve constant queue size per node by Pippenger [Pip84]. 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 (or partial permutations, and concatenation of permutations). Another disadvantage is that the communication task is proved to be executed within the specified time with "high probability" instead of "always".

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 [GrG86] for mesh networks (numerically), Greenberg and Hajek [GrH90] for hypercubes (analytically), Maxemchuk [Max87], [Max89] for the Manhattan network and the Shuffle Exchange (numerically and through simulations), and Varvarigos [Var90] for hypercubes (for a priority deflection scheme, through the formulation of a Markov chain). 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 [DiJ81] analyzed approximately the performance of buffered Delta networks. Stamoulis [Sta91] analyzed 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, described in Section 2 and called the simple and the priority routing schemes. In both of these schemes, packets follow shortest paths to their destinations. However, they may remain at some node without being transmitted even if there is a free link which 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 as opposed to the $\Theta(d^2)$ wires of a cross-bar switch. Packets can 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 been in the network longer have priority. The priority scheme has significantly larger throughput than the simple scheme, especially when the buffer space is small and the load is heavy.

The throughput results that we derive in Sections 3 and 4 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 rate of new packets. The parametric solution involves a single parameter, and is very close to a closed form solution. The throughput of the priority routing scheme is obtained from a (backward) recursion of only $d$ steps, where $d$ is the hypercube dimension. Our results are approximate but as simulations given in Section 5 indicate, 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 [Gol91], or they do not use a packet switching format, as in the CSR protocol proposed in [VaB92]). The Connection Machine model 2 of Thinking Machines Corp., which with its 65,000 bit processors is the biggest parallel computer currently available, uses deflection routing and has space only for the packets being transmitted. In Section 6 we focus on two deflection schemes, which we call the *simple non-wasting deflection* and the *priority non-wasting 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 non-wasting 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 [Max90]), and requires the use of cross-bar switches at the nodes. The simple and the priority schemes do not exhibit these disadvantages, and they use switches much simpler than cross-bar switches.

## 2. 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. Given two nodes $s$ and $t$, $s \oplus t$ denotes the result of their bitwise exclusive OR operation and is called the *routing tag* between the two nodes. A link connecting node $s$ to node $s \oplus e_i$, $i = 0, 1, \ldots, d-1$, is called a *link of the $i^{\text{th}}$ dimension*. Note that if the $i^{\text{th}}$ 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.

We now introduce the node model, and describe the simple and the priority routing schemes.

Each node has a queue for each of its outgoing links. The queue of node $s$ that corresponds to link $i$ is called the *i-th link queue*, and is denoted by $Q_i(s)$. A link queue is composed of two *buffers*, which can hold $k+1$ packets each. The first buffer is called the *forward buffer* and is denoted by $Q_i^1(s)$. The forward buffer can be used only by packets that have to cross the $i^{\text{th}}$ dimension. The second buffer, denoted by $Q_i^0(s)$, is called *internal buffer*, and can be used only by packets that do not have to cross the $i^{\text{th}}$ 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.

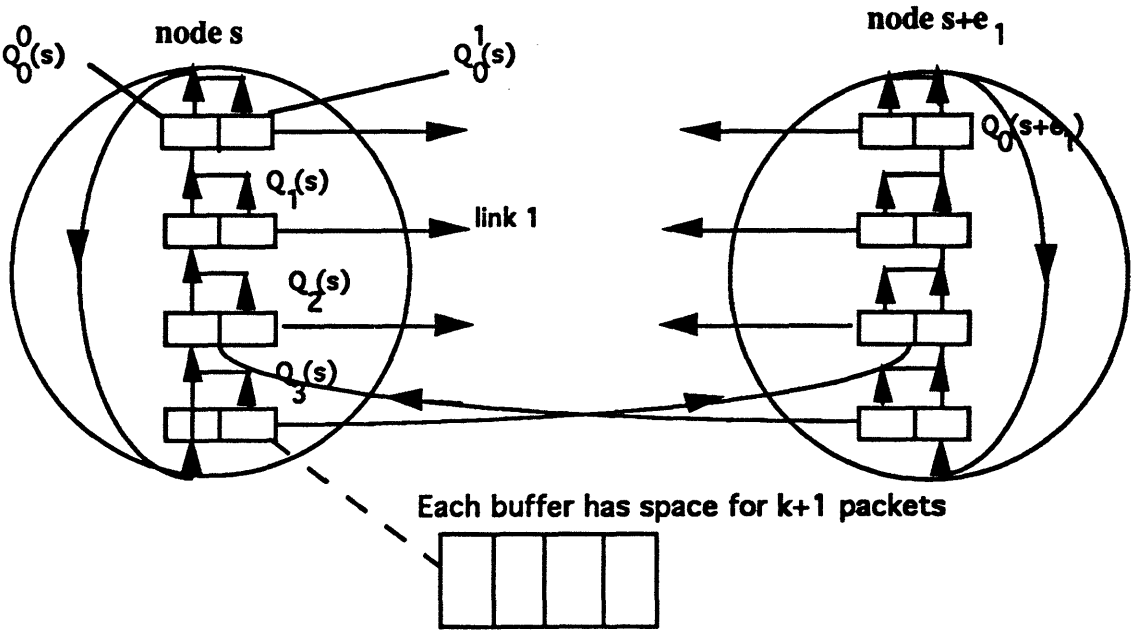


**Figure 1:** A node of the hypercube. For a comparison with other node (router) models, see also Fig. 13 of Subsection 6.2.

The queues of the nodes are linked in the following way: the internal buffer $Q_i^0(s)$ is connected to queue $Q_{(i-1) \bmod d}(s)$ of the same node, while the forward buffer $Q_i^1(s)$ is connected to queue $Q_{(i-1) \bmod d}(s \oplus e_i)$ 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 cross-bar 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 $Q_i(s)$ [either from buffer $Q^0_{(i+1) \bmod d}(s)$ of the same node, or from buffer $Q^1_{(i+1) \bmod d}(s \oplus e_{i+1})$ of the neighbor node $s \oplus e_{i+1}$, or a new packet]. Then the $i^{\text{th}}$ bit of its routing tag is checked. Depending on whether this bit is a one or a zero, the packet claims buffer $Q_i^1(s)$ in order to be transmitted during the next slot to queue $Q_{(i-1) \bmod d}(s \oplus e_i)$ of the neighbor node $s \oplus e_i$, or it claims buffer $Q_i^0(s)$ in order to be internally passed to the next queue $Q_{(i-1) \bmod d}(s)$ 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 been in the system longer 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 are removed from the network exactly $d$ time units (slots) after entering the network. 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 $d^{\text{th}}$ 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 [BeG87]). 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 $Q_i^j(s)$, $j \in \{0,1\}$, $i \in \{0,1,\ldots,d-1\}$. At most one new packet can claim a link buffer during a slot. New packets denied acceptance to the network, or packets that are dropped are not retransmitted (*memoryless property*). 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 mathematically equivalent. In order to see this, consider a packet located at the $i^{\text{th}}$ link queue of a node $s$ that does not have to cross dimension $i$, and is therefore passed to the $(i - 1 \bmod \text{d})^{\text{th}}$ queue of the same node. The probability of this event is equal to the probability of the event that the packet has to cross dimension $i$ and is sent to the $(i - 1 \bmod \text{d})^{\text{th}}$ queue of node $s \oplus e_i$.

## 3. THE SIMPLE SCHEME

In this section we evaluate the throughput of the simple scheme. The unbuffered and the buffered cases are analyzed in Subsections 3.1 and 3.2, respectively. Recall that in the simple scheme all packets have equal priority, except for the new packets, which have the lowest priority. The analysis to be given is approximate, but very close to being accurate as Section 5 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 elegant as a closed form solution. We consider this important, since 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 are of an asymptotic nature).

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 than $d$ 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.

### 3.1 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 particular link during a slot. We also denote by $e$ the steady-state probability that a link is idle during a slot. By symmetry, both the $p_i$'s and $e$ are independent of the particular link. 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 $p_0$.

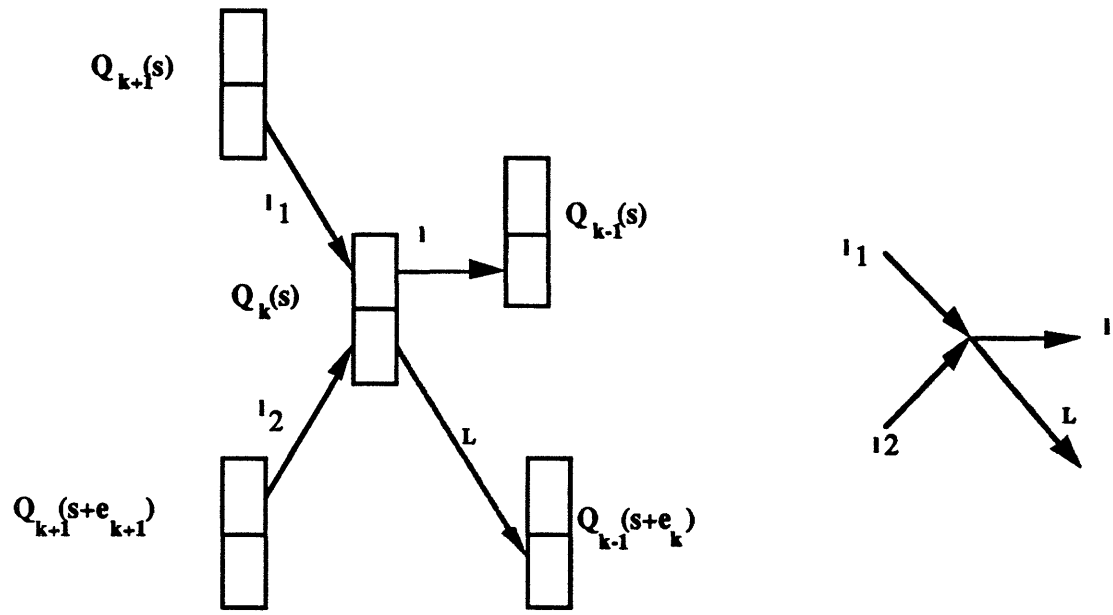


**Figure 2:** A link $l$, and the two links leading to it.

Consider a particular link $l$ (for example, the one connecting $Q_i^1(s)$ with $Q_{(i-1) \bmod d}(s + e_i)$). Call $l_1$ and $l_2$ the internal and the forward links, respectively, that lead to $l$ (see Fig. 2). We make the following approximating 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 5. Nonetheless, we believe that the approximation A.1 is quite accurate, as the discussion in Section 5 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 $\mathcal{Q}$ coming on $l_2$. Of course, packet $\mathcal{Q}$ should then be of type different than $d$ (packets of type $d$ are removed from the network since they have arrived at their destination). Under the approximating assumption A.1, the probability that a packet of type different than $d$ 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

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

The probability $p_1$ that a packet of type 1 (i.e., 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

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

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

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

It is useful to define

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

where $\theta$ will be treated as a free parameter. Then Eqs. (2)-(5) give

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

and

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

Adding Eqs. (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

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

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, using arguments similar to those used later in Subsection 6.2, that for uniformly distributed destinations, $R$ is always less than two.

Note that the eligible values of the parameter $\theta$ 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 $\theta$ we can find the corresponding values of $p_0$ and $R$. The fact that the range of $\theta$ is not the entire interval [0,1] does not create any difficulty, since the values of $\theta$ which are not feasible give $p_0 > 1$.

Figure 3 illustrates the results obtained for $d = 11$. To evaluate the results it is useful to have in mind typical values of the traffic load that appears in real systems. Measurements reported in [HsB90] for numerical and simulation algorithms have given that in almost all cases links were idle for more than 95% of the time. In our model such loads correspond to values of $p_0$ much less than 0.05[1]. Note that the maximum throughput is achieved for $p_0$ less than one. Therefore, when the simple scheme without buffers is used, some mechanism to control the rate at which nodes transmit would be necessary.

[1] This value is probably an overestimate. If the measurements reported in [HsB90] corresponded to uniform traffic, then the corresponding $p_0$ would be considerably less; however, the communication patterns in the measurements of [HsB90] may have had locality properties. Note also that even with infinite buffers the case $2dp_0 > 2$ would correspond to an unstable system, since 2 is the maximum throughput that can be sustained.

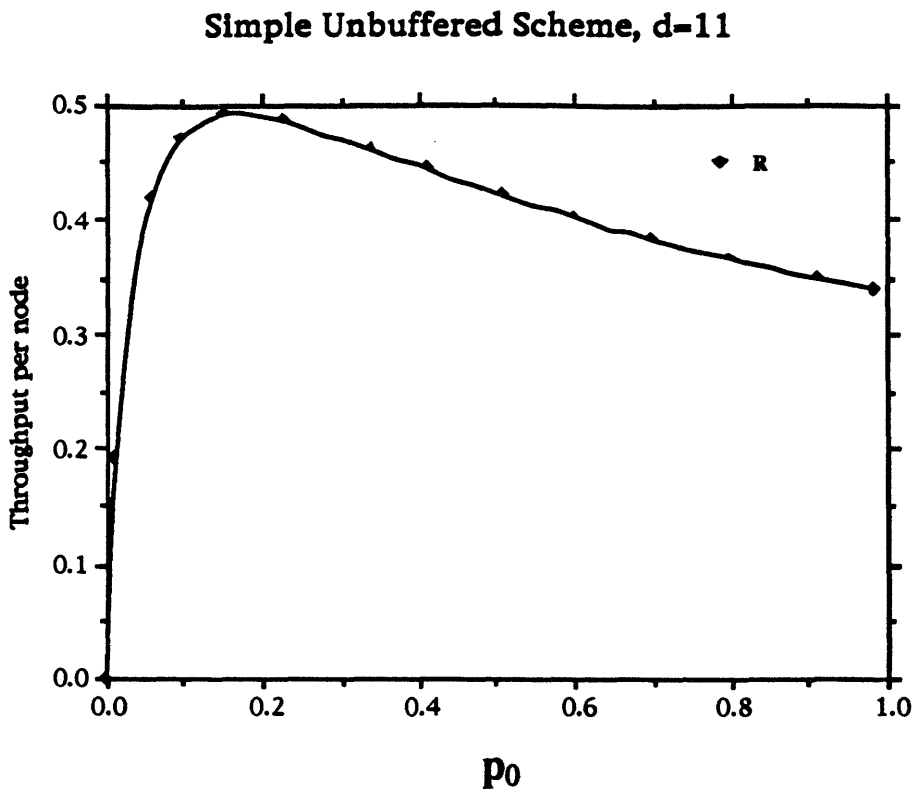


Figure 3: Simple scheme without buffers (d=11).

### 3.2. Analysis of the Simple Scheme With Buffers

In this subsection 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 Subsection 3.1 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 Subsection 3.1 corresponds to the case $k = 0$ of this subsection. The reason the unbuffered and the buffered cases were treated separately is that 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 have it filled 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 ([DiJ81], [GrG86], [Dal90a], [Max89], [Var90], [Bra91]), or they are of an asymptotic nature in the number of processors or in the buffer size ([GrH90], [Sta91]).

In order to analyze the buffered version of the simple scheme we need, in addition to the approximating assumption A.1, the following approximating 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 direct or indirect 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 Subsection 3.1. 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 Eq. (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 Eq. (9) we obtain

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

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

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

Eqs. (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 Eqs. (13)-(15) 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 $\alpha$ and $\beta$. Taking into account Eqs. (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 Eqs. (13)-(15), and see that these equations hold identically for all $\theta$. Since

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

we finally get that

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

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

For $k = 0$ (unbuffered case) Eq. (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 Eq. (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 non-empty, and (iii) the probability that the packet at the head of the buffer is of type $i-1$. Using the approximating assumption A.1, we get

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

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

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

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 Eq. (17). Equations (21) and (22) give the relationship between $p_d$ and $p_0$ in parametric form with parameter $\theta$. The throughput per node is

$$R = 2dp_d.$$

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

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

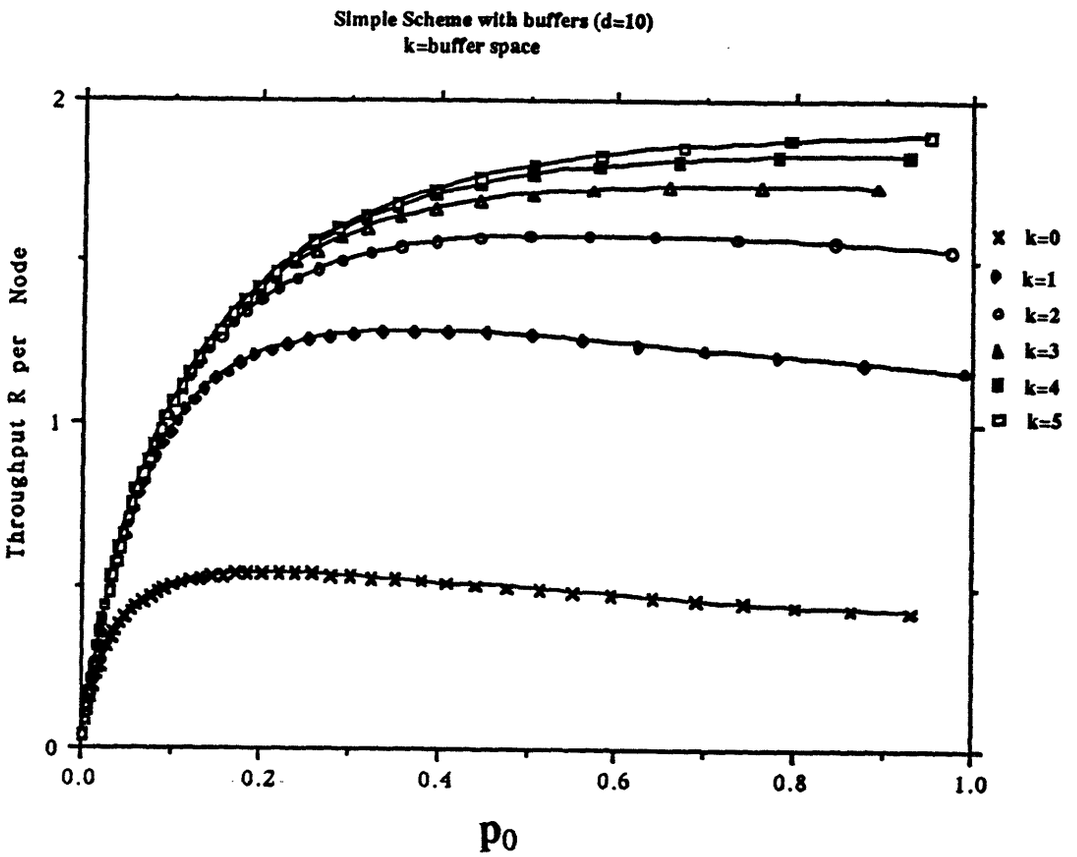


**Figure 4:** Throughput per node of the simple scheme for various buffer sizes.

As $k \rightarrow \infty$, Eq. (22) takes the indeterminate form $0/0$. By using L' Hospital's rule we get after some calculations that

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

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

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

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

In Fig. 4 we have plotted the throughput $R$ per node as a function of $p_0$ for several buffer sizes $k$. It can be seen 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 that for $k \geq 2$ the throughput 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.

### 3.3. Asymptotic Behavior of the Thoughput

In this subsection 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 Eqs. (20), (21), and (17), we obtain after some calculations that

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

Define

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

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

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

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

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

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 Eq. (25) tends to a positive constant $c = e^{-2}$, that is

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

For $y$ given by Eq. (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 Eq. (27) that $p_i = \Theta(p_d)$, for $i = 1, 2, \ldots, d$, and

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

Therefore, the maximum total throughput $H$ of the hypercube is

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

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, Eq. (25) gives $p_d/p_1 \to 0$ as $d \to \infty$. Based on these observations and Eq. (28) we conclude that

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

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, $p_0 = o(1/d^{1+\frac{1}{2k+1}})$] 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, Eq. (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 Eq. (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 with that of Fig. 1 (but the greedy scheme is different than the simple or the priority scheme where each packet starts transmission from a random dimension). The greedy scheme has been analyzed by Koch in [Koc88] and [Koc89] (the case where the capacity is equal to one was previously analyzed in [Pat81] and [KrS83]; see also [Lei92a], pp. 612-620). The maximum total throughput of the greedy scheme in a $Q$-dilated hypercube is given by

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

Comparing Eqs. (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

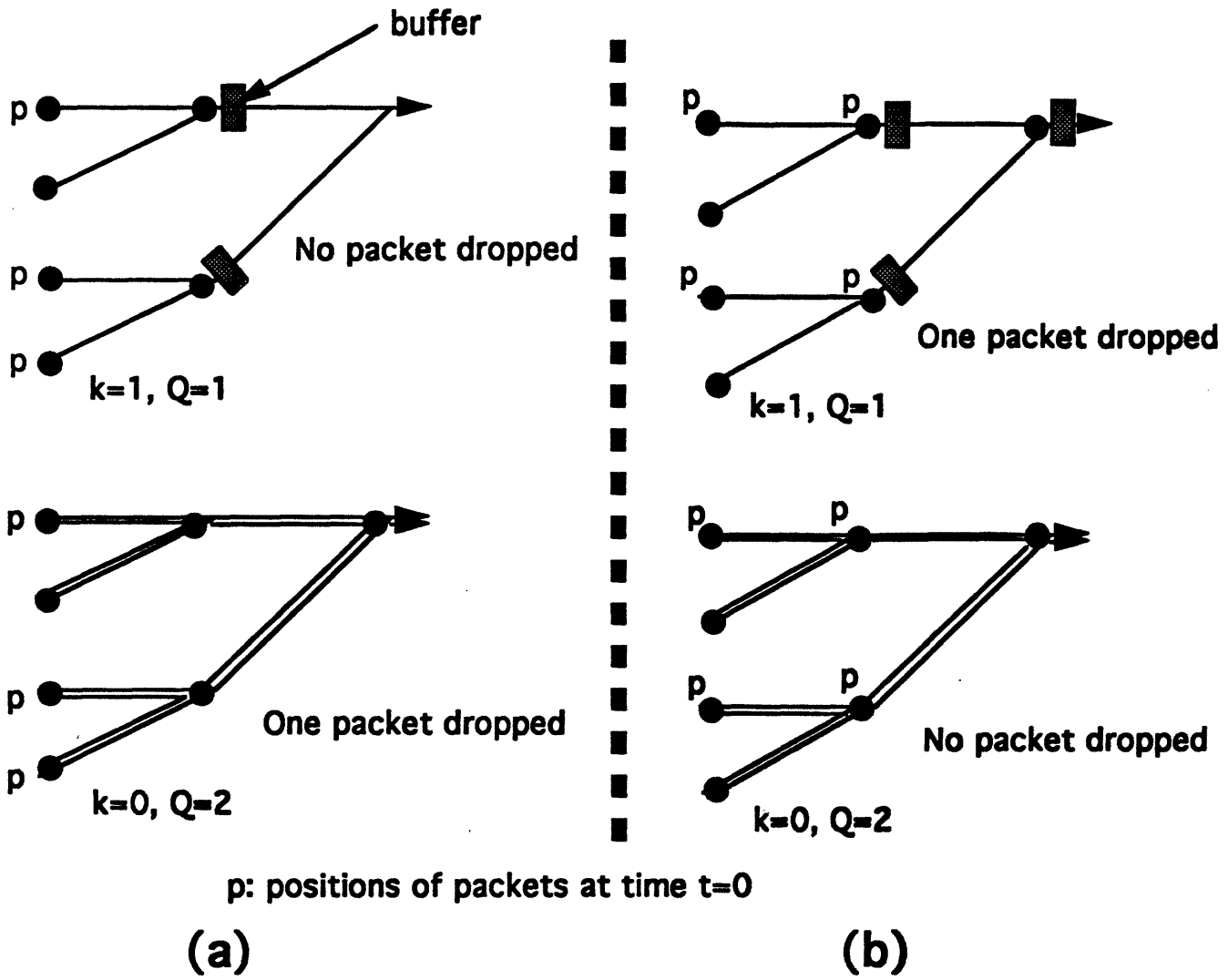


**Figure 5:** In scenario (a) more packets are dropped when links have capacity two and no buffer space than when they have capacity one and buffer space for one packet in addition to the packet under transmission. In scenario (b) the opposite is true.

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 impoves the throughput significantly more 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 should hold on the average, since one can devise scenaria where buffer space helps most, and scenaria where capacity

helps most ([Lei92b]). For example, Fig. 5a illustrates a scenario where having more buffer space per link results in less packets losses than having more capacity per link, while Fig. 5b illustrates a scenario where the opposite is true. One should also note here, that although buffer space may be more important than capacity when the objective is the throughput, the situation will probably be different when the objective is the average packet delay.

# 4. 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 been in the system longer 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 at a link buffer only if there are no continuing or buffered packets claiming the link. We analyze the unbuffered priority scheme in Subsection 4.1, and the buffered priority scheme in Subsection 4.2.

## 4.1. Analysis of the Priority Scheme Without Buffers

In this subsection we analyze the unbuffered case. Since packets which 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 Subsection 3.1. Consider a link $l$, and the two links $l_1$ and $l_2$ leading to it. Making again the approximating assumption A.1 and reasoning as in Subsection 3.1 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: A packet $Q$ of type $i-1$ was transmitted on link $l_2$ during the previous slot, its routing tag was such that $l$ was selected, and $Q$ 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, Eqs. (31)-(33) give

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

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 no 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 particulal value of $p_d$ we can use Eqs. (34)-(36) to find the corresponding values of $p_{d-1}, p_{d-2}, \ldots, p_0$. This is done by solving Eq. (34) with respect to $p_{i-1}$, and keeping the solution that gives a legitimate probability distribution (the other solution corresponds to $p_{i-1} > 1$):

$$p_{i-1} = 2 - \sum_{j=i}^{d-1} p_j - \sqrt{\left(2 - \sum_{j=1}^{d-1} p_j\right)^2 - 4p_i}, \quad i = 0, \ldots, d-2. \tag{37}$$

For $p_0$ we have from Eq. (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 Eq. (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. Figure 6 illustrates the results obtained for $d = 11$.

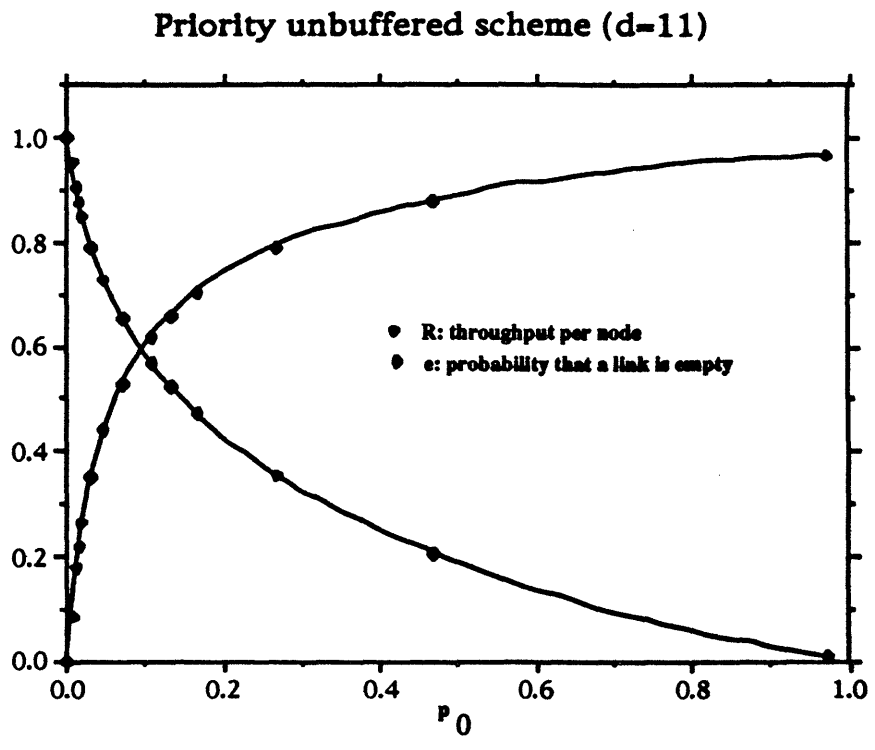


**Figure 6:** Priority scheme without buffers (d=11).

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 $p_d$. It can be proved 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 Eq. (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. 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.

### 4.2. 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 dropped. The packet

which is transmitted is the one that has crossed more forward and internal links with ties resolved at random. The analysis of Subsection 4.1 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 analyze.

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

Following the notation of Subsection 3.2, 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 Eqs. (16) and (17), which we repeat here for completeness:

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

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

$$p_i = p_{i-1}\left(1 - \frac{\sum_{j=i}^{d-1} p_j}{2} - \frac{p_{i-1}}{4}\right) + \left(1 - \frac{\sum_{j=1}^{d-1} p_j}{2}\right)^2 (1-b_0)\frac{2p_{i-1}\left(\frac{p_{i-1}}{2} + \sum_{j=i}^{d-1} p_j\right)}{(1-\theta)^2}$$

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

where the first term in the above summands is the same with the right side of Eq. (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

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

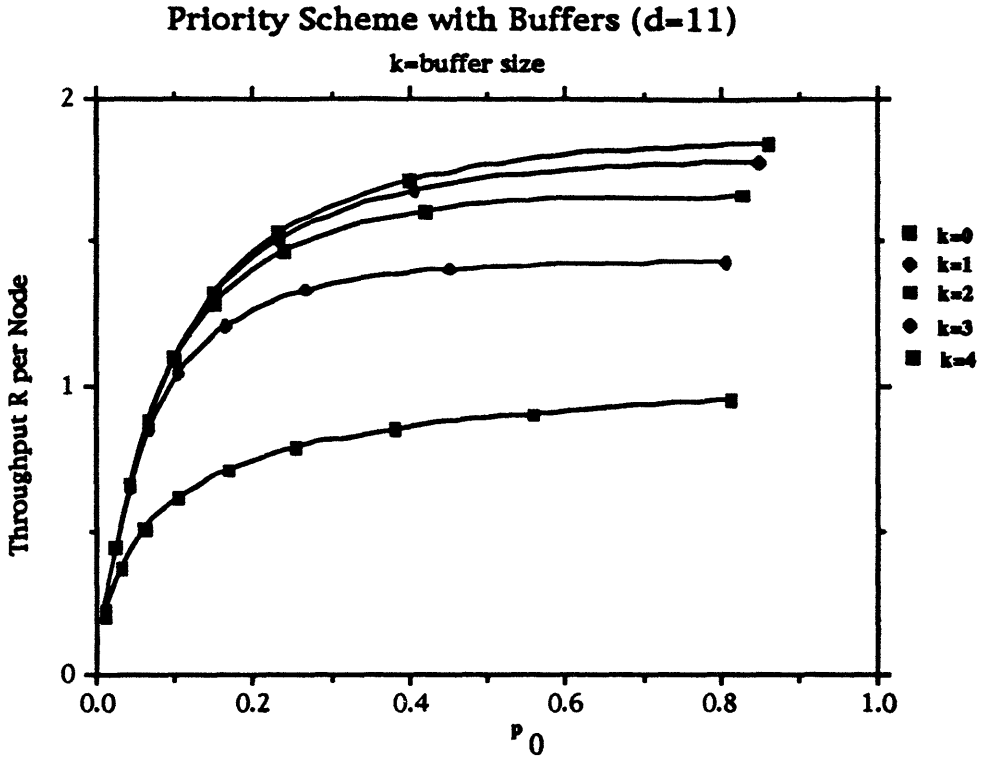


**Figure 7:** The throughput of the priority scheme for various buffer sizes.

We used a Gauss-Seidel type of algorithm to find numerically $p_i$'s that satisfy Eqs. (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 Figs. 7 and 8. Figure 7 illustrates the throughput $R = 2dp_d$ per node as a function of $p_0$ for several buffer sizes $k$. Figure 8 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 $k$. Figures 7 and 8 suggest that little buffer space is adequate in practice to achieve

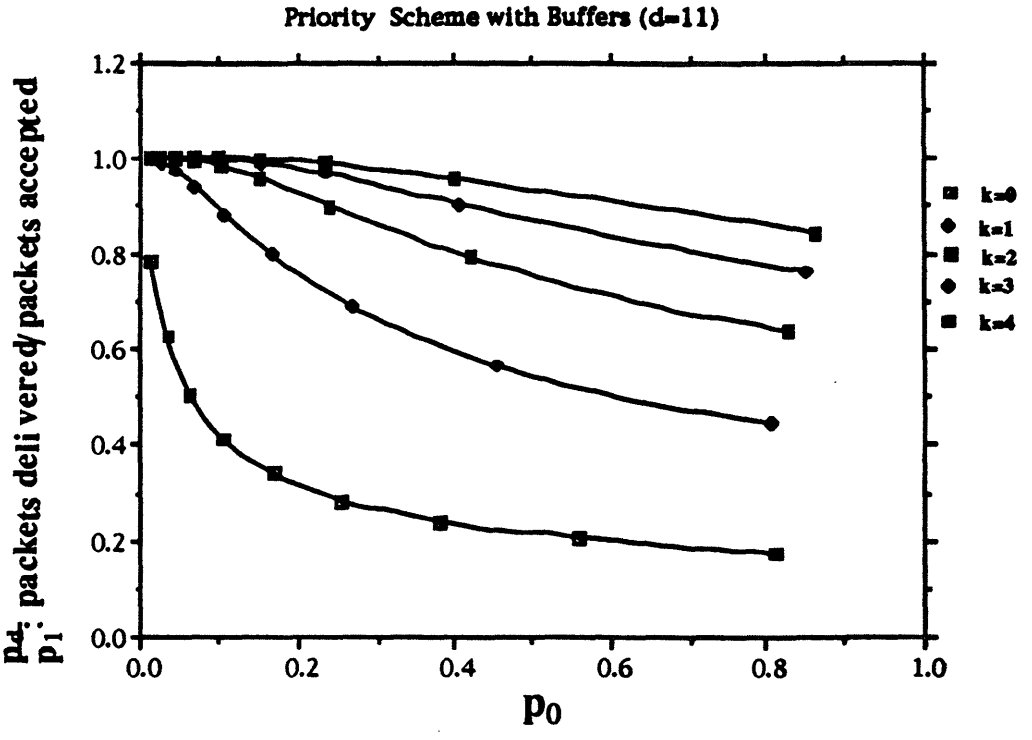


**Figure 8:** Packets delivered to their destination as a fraction of those accepted in the network for the priority scheme and various buffer sizes.

satisfactory throughput, and low probability of packet losses (recall that the load region where we are primarily interested is $p_0 < 1/d$).

Comparing Figs. 4 and 7 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 scheme (especially the unbuffered version) is so simple that it can be implemented entirely in hardware. A similar priority rule for deflection routing will be examined in Section 6.

## 5. 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. 9; the direction of the links, which is from left to right, is not shown for simplicity). 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.

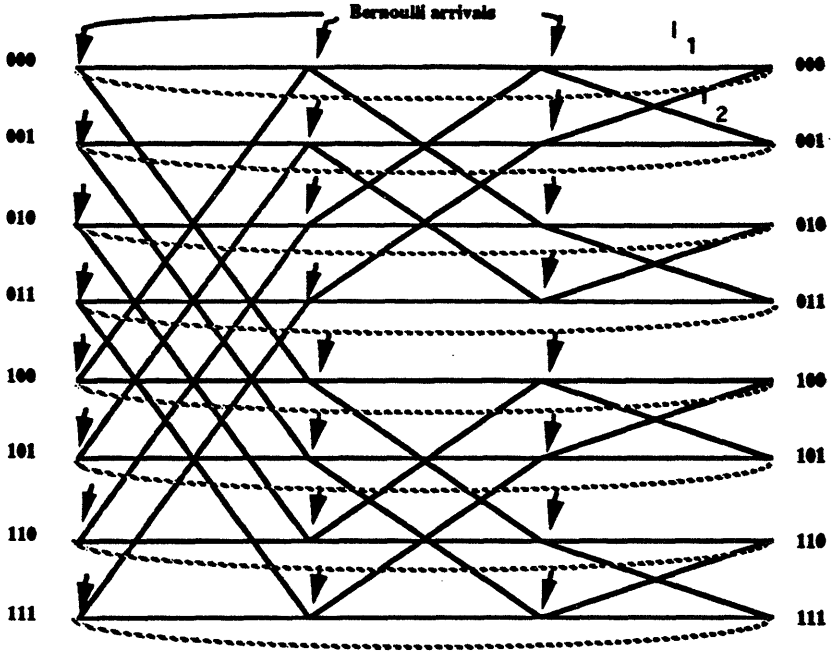


**Figure 9:** A wrapped butterfly. The nodes of the last stage are the same with the nodes of the first stage. The destination of a packet has to be at the same stage with its source.

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:

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 gives a partial answer to these questions.

**Lemma 1:** Events on links $l_1$ and $l_2$ at time $T$, are dependent only through events that happened at time $T-d$ and before.

**Proof:** By the symmetry of the system we can assume without loss of generality that the links $l_1$ and $l_2$ are of dimension $d$ (see Fig. 9).

The sequence of links traversed by a packet, together with the corresponding times will be referred to as the *time-path* of the packet. The *length* of a time-path is the number of links it traverses; we are only interested in time-paths of length less than or equal to $d$. Let $P_1 = \{(x_1,t_1),(x_2,t_1+1),\ldots,(l_1,T)\}$ be the time-path of a packet that passes from $l_1$ at time $T$. We use the same symbol $P_1$ to denote the event of a packet following that path. Let $P_2 = \{(y_1,t_2),(y_2,t_2+1),\ldots,(l_2,T)\}$ be a time-path leading to link $l_2$ at time $T$. We are interested in the dependence between events $P_1$ and $P_2$.

Consider a packet $p$ that entered the wrapped butterfly at time $t_0$ at dimension $i_0$. The link traversed by the packet during slot $t \geq t_0$ has dimension $i$, with $i+t$ mod d=$i_0+t_0$ mod d. This is because packets travel dimensions in descending order and there is no buffering. Therefore, the sum $i+t$ mod d of the time slot and the dimension traversed at that slot is a constant of the packet (or of the corresponding time-path), and is called *class* of the packet. We denote the class of packet $p$ by $c(p) = i_0+t_0$ mod d.

We will denote the dependence between two events A and B by $A \sim B$. It is easy to see that $\sim$ is an equivalence relation. Two time-paths intersect (or equivalently, two packets that follow them collide) only if they pass through the same link during a slot. Only time-paths of the same class may intersect, and only packets of the same class may collide. Dependencies are created and spread only through the intersection of time-paths. For example, if time-path A intersects time-path B, and B intersects time path C, then events A and C may be dependent. Events corresponding to time-paths of different classes are independent.

Let $H_0$ (or $H_1$) be the sub-wrapped-butterfly that consists of the nodes whose least significant bit is equal to zero (or one, respectively), and the links that connect them. A time-path belongs to $H_0$ (or $H_1$) if all the links that it traverses belong to $H_0$ (or $H_1$, respectively). The time-paths $P_1$ and $P_2$ that lead to links $l_1$ and $l_2$ at time $T$ satisfy

$$P_1 \in H_0 \tag{43}$$

and

$$P_2 \in H_1. \tag{44}$$

Events $P_1$ and $P_2$ can be dependent only in the following two cases:

Case A: The time-paths $P_1$ and $P_2$ intersect before time $T$, or

Case B: There is an integer $k$ and time-paths $X_1, X_2, \ldots, X_k$ such that

$$P_1 \text{ intersects } X_1 \text{ at time } T_1 < T$$

$$X_1 \text{ intersects } X_2 \text{ at time } T_2 < T$$

$$\vdots$$

$$X_{k-1} \text{ intersects } X_k \text{ at time } T_k < T$$

$$X_k \text{ intersects } P_2 \text{ at time } T_{k+1} < T.$$

Case A cannot happen because of Eqs. (43)-(44), and the fact that $H_0$ and $H_1$ are disjoint. In view of Eqs. (43) and (44), case B can happen, only if there is an $i \in \{1, 2, \ldots, k\}$ such that time-path $X_i$ traverses a link of dimension $d$ (passing from $H_0$ to $H_1$ or vice versa) at some time prior to $T$. But

$$c(P_1) = c(X_1) = \cdots = c(X_k) = c(P_2) = d + T \text{ mod d},$$

because any two intersecting time-paths have the same class. Thus $c(X_i) = d + T$ mod d, which means that packet $X_i$ crosses dimension $d$ at or prior to time $T - d$. Therefore, $X_i$ intersects with either $X_{i+1}$ or $X_{i-1}$ prior to time $T - d$. This proves that events on links $l_1$ and $l_2$ are dependent only through events (conflicts or non-conflicts) that have taken place before time $T - d$. **Q.E.D.**

Lemma 1 says that the approximating assumption A.1 is weaker than the following assumption: "Events that take place at time $T$ are independent from events that take place at time prior to $T - d$". This 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 assumption A.1 is a quite accurate approximation. In fact, the larger $d$ is the better the approximation. The above arguments do not assume any way to resolve conflicts, and therefore hold 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, $\Delta p$ 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 the analytical and the simulation results has been consistently negligible for all network sizes and values of $p_0$ (see Table 1 for $d = 8$). This is a further indication that the parametric equations obtained in Subsection 3.1 are very accurate.

| $p_0$ | Throughput/node (analytical) | Throughput/node (simulations) |
|---|---|---|
| **0.9983** | 0.6325 | 0.6331 |
| **0.9288** | 0.6401 | 0.6401 |
| **0.8045** | 0.6539 | 0.6540 |
| **0.6972** | 0.6657 | 0.6650 |
| **0.6042** | 0.6754 | 0.6744 |
| **0.5234** | 0.6827 | 0.6824 |
| **0.4871** | 0.6853 | 0.6843 |
| **0.3642** | 0.6888 | 0.6883 |
| **0.3142** | 0.6859 | 0.6852 |
| **0.2915** | 0.6831 | 0.6828 |
| **0.2145** | 0.6628 | 0.6621 |
| **0.1982** | 0.6552 | 0.6557 |
| **0.1094** | 0.5712 | 0.5721 |
| **0.0030** | 0.0448 | 0.0446 |

**Table 1:** Simulation and analytical results for the unbuffered simple scheme for $d = 8$.

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 unbyffered schemes, and were, therefore, expected to be less accurate. Table 2 illustrates the results obtained for $d = 7$ and $k = 1$.

| $p_0$ | Throughput/node (analytical) | Throughput/node (simulations) |
|---|---|---|

| | | |
|---|---|---|
| **0.931384** | 1.493738 | 1.451239 |
| **0.566517** | 1.477039 | 1.433139 |
| **0.302901** | 1.345433 | 1.354165 |
| **0.199937** | 1.189335 | 1.162777 |
| **0.169829** | 1.116160 | 1.092926 |
| **0.144199** | 1.038224 | 1.020776 |
| **0.103110** | 0.871355 | 0.861196 |
| **0.086444** | 0.783858 | 0.777389 |
| **0.052758** | 0.557855 | 0.554911 |

**Table 2:** Simulation and analytical results for the buffered simple scheme for $d = 7$ and $k = 1$.

## 6. COMPARISON WITH DEFLECTION ROUTING

In this section we describe two deflection schemes, called the ***simple non-wasting deflection*** and the ***priority non-wasting deflection*** schemes. We use simulations to find the throughput of these schemes and compare it with that of the priority scheme of Section 4. In Subsection 6.1 we describe the two deflection schemes, and the stochastic model under which they are examined. In Subsection 6.2 we present and discuss the results obtained.

### 6.1. The Deflection Schemes, and the Stochastic Model

Each node has a queue which 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 a free 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. At every slot, exactly $d$ packets are received by each node. Some of these packets exit the system because they have arrived at their destination, and are replaced by an equal number of new packets. Under this model the hypercube is a closed network and every node always has exactly $d$ packets. The destinations of the new packets are uniformly distributed over all nodes, except for their origin.

Before describing the deflection schemes, we give some definitions.

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 *non-wasting* switching assignment such a situation is not allowed. In Fig. 11, both a and b are non-wasting switching assignments, while c is not.

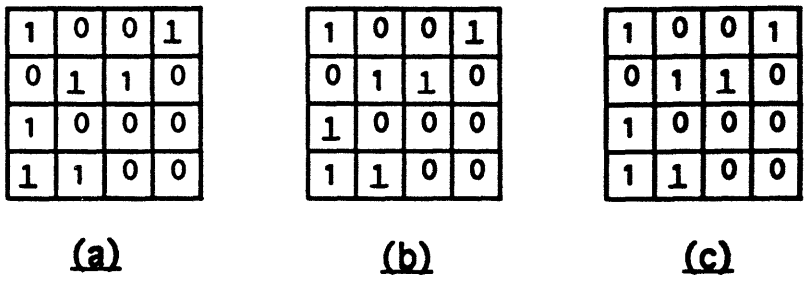


**Figure 11:** Cases a and b correspond to non-wasting assignments, while c is a wasting assignment.

There are many ways to obtain a non-wasting assignment. A simple procedure 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 previously considered 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 *non-wasting phase*, a non-wasting 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 non-wasting deflection scheme the processing order is random with all orders (permutations) being equally probable. In the priority non-wasting deflection scheme the processing order is found as follows. The packets are partitioned in priority classes, so that the $i^{th}$ priority class consists of the packets that are currently located at a distance $i$ from their destination. The order of the packets within the same class is random; however, packets that are closer to their destination precede in order packets that are farther from their destination.

In general, the partial assignment found in the non-wasting 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. This extension is achieved by arbitrarily mapping the remaining $d-z$ 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.

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. To see that consider a packet which is one hop away from its destination. Such a packet has only one preferred link, and the probability that this link will have already been assigned when the packet is processed is large. 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 to assign it to a link only after all other packets have been processed.

### 6.2. Steady State Throughput of the Deflection Schemes

Before presenting the simulation results for the two deflection schemes, we give an upper bound on their throughput.

Let $\lambda$ be the average total throughput of the hypercube at steady-state. Since the number of packets in the hypercube is constant and equal to $Nd$, Little's theorem gives

$$Nd = \lambda T,$$

where $T$ is the mean delay of a packet from the time it is accepted in the network until the time it arrives at its destination. For uniformly distributed packet destinations we have

$$T \geq \frac{d}{2}\frac{N}{N-1}.$$

Combining the last two equations we get

$$\lambda \leq 2(N-1). \tag{45}$$

Figure 12 illustrates the simulation results obtained for the simple and the priority non-wasting deflection schemes, together with the analytical results obtained for the priority scheme (with buffer size $k = 0$ and $k = 1$) of Section 4. Note that as the dimension of the hypercube increases the throughput of the deflection schemes increases. However, for small dimensions the priority scheme

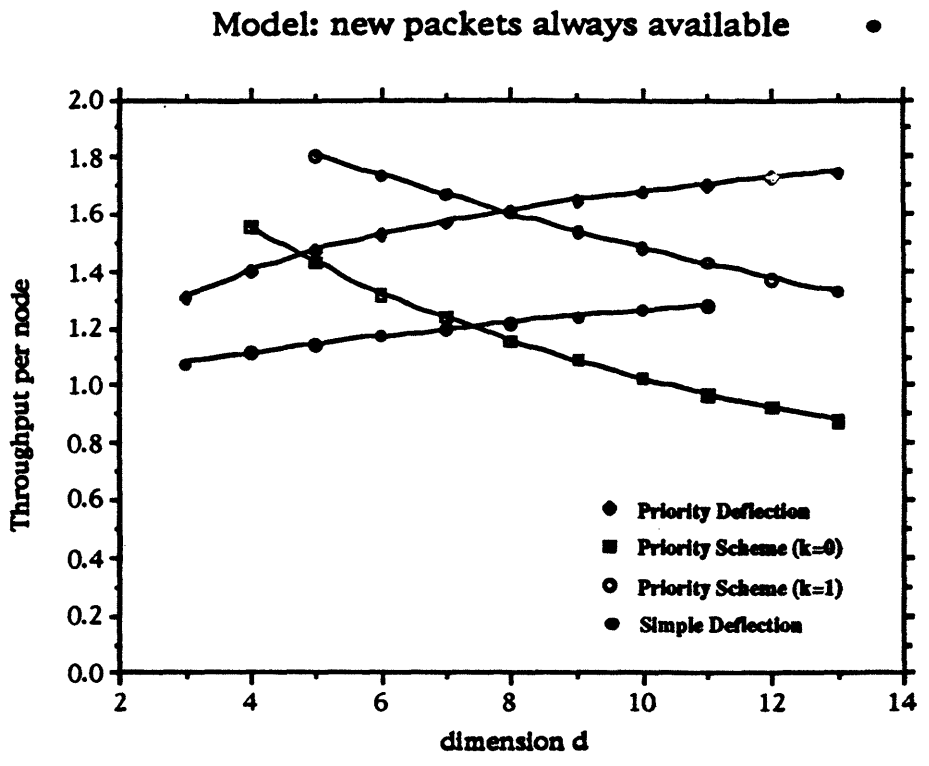


**Figure 12:** Throughput per node of (1) the simple non-wasting deflection scheme, (2) the priority non-wasting deflection scheme, (3) the unbuffered priority scheme, and (4) the buffered ($k = 1$) priority scheme.

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 cross-bar 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 (provided that we can afford the additional hardware).

The average delay of the deflection schemes satisfies

$$T = \frac{d}{2}\frac{N}{N-1} + 2E(\text{Number of Deflections}), \tag{46}$$

where $E$(Number of Deflections) is the average number of deflections suffered by a packet. The first term of the right hand side of the preceding equation comes from the fact that the mean delay of a packet when it is not deflected is equal to $\frac{d}{2}\frac{N}{N-1}$. For the second term note that every time a packet is deflected, its delay increases by two steps. Figure 14 illustrates the average number of deflections suffered by a packet in the priority non-wasting deflection scheme, for various dimensions $d$ of the

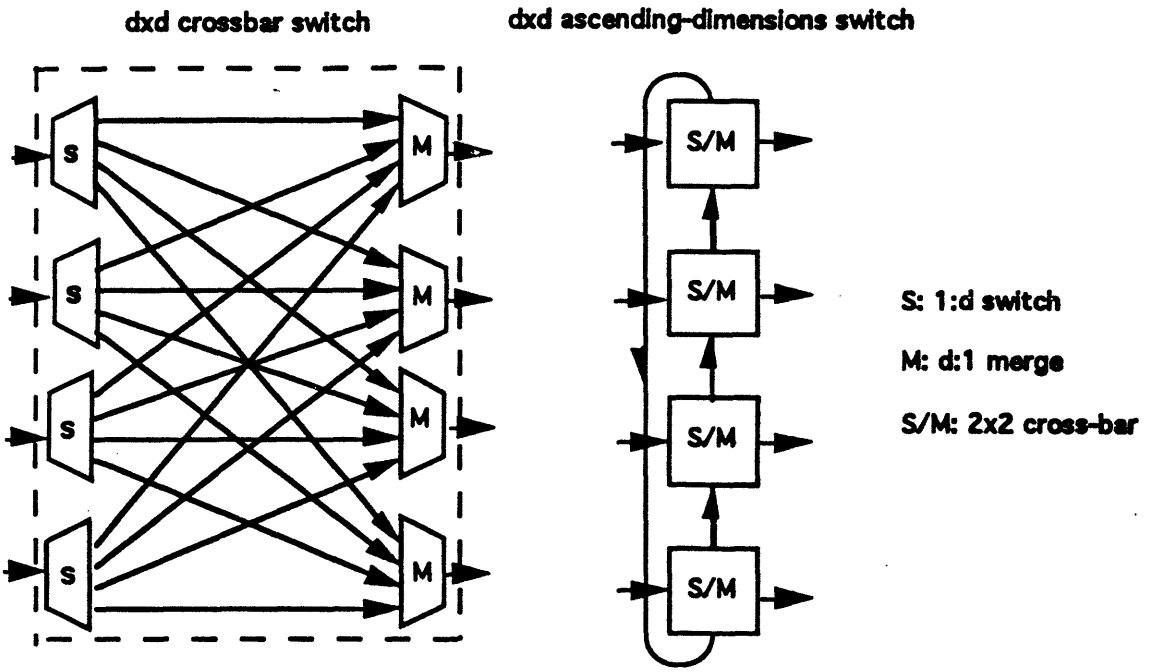


**Figure 13:** A $d \times d$ cross-bar switch (required by deflection schemes), a $d \times d$ descending-dimensions switch (required by the simple and the priority schemes), and the modules out of which they are composed. The number of wires of a descending-dimensions switch is only $\Theta(d)$. A cross-bar 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 [Dal90]).

hypercube (the average number of deflections is obtained by Eq. (46) and Little's theorem by using the simulation results for the throughput $\lambda$).

An interesting observation concerning Fig. 14 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 actually 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 of the priority deflection scheme will tend to the upper bound of two packets per node, which is the maximum possible for uniformly distributed destinations [for this to be true it would be enough to prove that the average number of deflections is $o(d)$]. We could not prove this by a rigorous analysis, so we leave it as a conjecture.

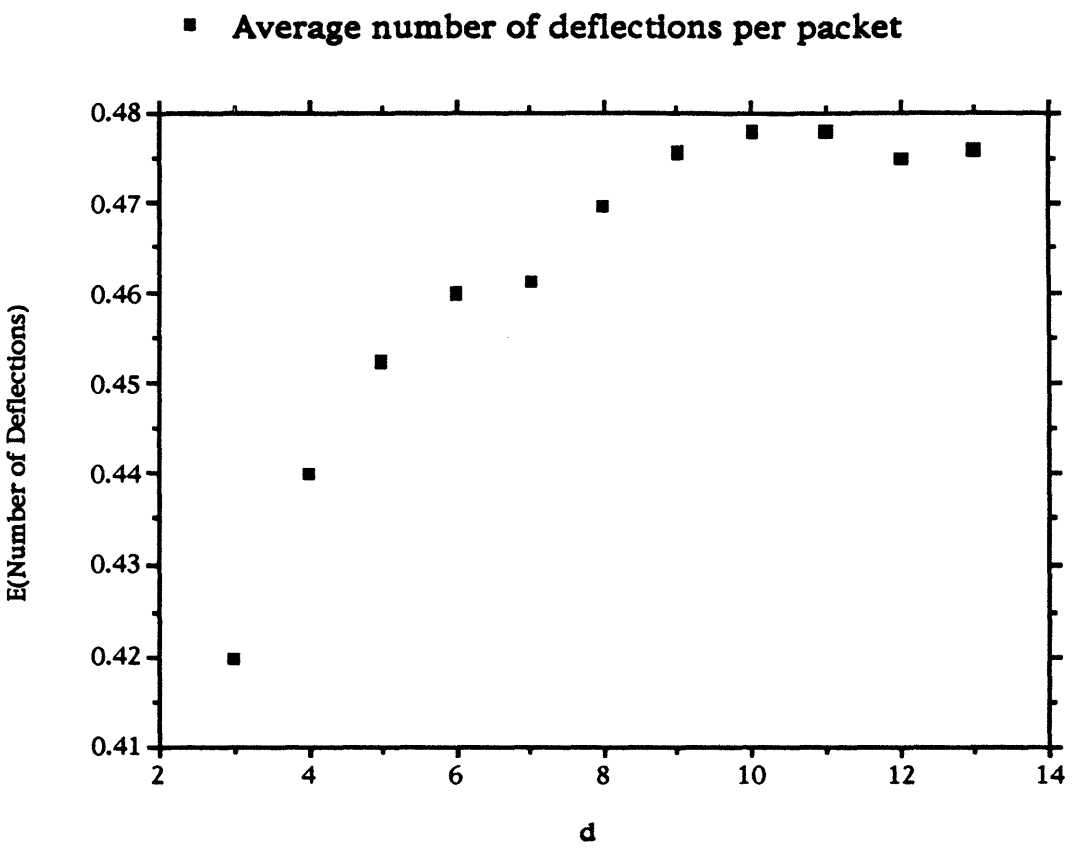


Figure 14: Average number of deflections per packet for the priority non-wasting deflection scheme.

## 7. 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 simulation results indicate, and they were given in particularly interesting forms. The 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 necessary and adequate 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

[Bra91] Brassil, J. T., *Deflection Routing in Certain Regular Networks*, Ph.D. Thesis, UCSD, 1991.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

[BeG87] Bertsekas, D. P., and Gallager R., *Data Networks*, Prentice-Hall, 1987.

[ChL89] Choudhury A., and Li, V. O. K., "Performance Analysis of Deflection Routing in the Manhattan Street and Minimum-Distance Networks," preprint.

[Dal90] Dally, W. J., "Network and Processor Architecture for Message-Driven Computers," in R. Suaya, and G. Birtwistle (Eds.), *VLSI and Parallel Computation*, Morgan Kaufmann Publishers, San Mateo, CA, pp. 140-222, 1990.

[DiJ81] Dias, D. M., and Jump, J. R., "Analysis and Simulation of Buffered Delta Networks," *IEEE Trans. on Computers*, Vol. C-30, pp. 273-282, August 1981.

[GrG86] Greenberg, A. G., and Goodman, J., "Sharp Approximate Models of Adaptive Routing in Mesh Networks," in J. W. Cohen, O. J. Boxma and H. C. Tijms (Eds.), *Teletraffic Analysis and Computer Performance Evaluation*, pp. 255-270, Elsevier, Amsterdam, 1986, revised 1988.

[GrH90] Greenberg, A. G., and Hajek, B., "Deflection Routing in Hypercube Networks," to appear it IEEE Trans. on Communications, June 1989 (revised December 1990).

[HaC90] Hajek, B., and Cruz, R. L., "On the Average Delay for Routing Subject to Independent Deflections," submitted to *IEEE Trans. on Information Theory*, June 1990.

[Haj91] Hajek, B., "Bounds on Evacuation Time for Deflection Routing," Distrib. Comput., 5:1-6, 1991.

[HsB90] Hsu, J., and Banerjee, P., "Performance Measurements and Trace-Driven Simulation of Parallel CAD and Numeric Applications on Hypercube Multicomputers," in Proc. 17th Intl. Sypm. Computer Architecture, Seattle, WA, May 1990.

[Gol91] Golestani, S. J., "Congestion-Free Communication in High-Speed Packet Networks," *IEEE Trans. on Communications*, Vol. 39, No. 12, December 1991.

[Koc88] Koch, R., "Increasing the Size of the Network by a Constant Factor Can Increase Performance by More than a Constant Factor," in 29$^{th}$ Annual Symposium on Foundations of Computer Science, IEEE, pp. 221-230, October 1988.

[Koc89] Koch, R., *An Analysis of the Performance of Interconnection Networks for Multiprocessor Systems*, Ph.D. Thesis, MIT, May 1989.

[KrS83] Kruskal, C., and Snir, M., "The Performance of Multistage Interconnection Networks for Multiprocessors," *IEEE Trans. on Computers*, C-32(12), pp. 1091-1098, December 1983.

[Lei92a] Leighton, F. T., *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*, Morgan Kaufmann, San Mateo, CA, 1992.

[Lei92b] Leighton, F. T., personal communication, 1992.

[Max87] Maxemchuk, N. F., "Routing in the Manhattan Street Network," *IEEE Trans. on Communications*, COM-35(5), pp. 503-512, May 1987.

[Max89] Maxemchuk, N. F., "Comparison of Deflection and Store-and-Forward Techniques in the Manhattan Street and Shuffle-Exchange Networks," in INFOCOM '89, Vol. 3, pp. 800-809, April 1989.

[Max90] Maxemchuk, N. F., "Problems Arising from Deflection Routing: Live-lock, Lock-out, Congestion and Message Reassembly," Proceedings of NATO Workshop on Architecture and High Performance Issues of High Capacity Local and Metropolitan Area Networks, France, June 1990.

[Pat81] Patel, J. H., "Performance of Processor-Memory Interconnection for Multiprocessors," *IEEE Trans. on Computers*, Vol. C-30, pp. 545-556, April 1981.

[Pip84] Pippenger, P., "Parallel Communication with Limited Buffers," Proc. of the 25th Annual IEEE Symposium on Foundations of Computer Science, pp. 127-136, 1984.

[Sta91] Stamoulis, G., *Routing and Performance Evaluation in Interconnection Networks,* Ph.D. Thesis, MIT, Report LIDS-TH-2035, May 1991.

[Upf84] Upfal, E., " Efficient Schemes for Parallel Communication," J. ACM, Vol. 31, pp. 507-517, 1984.

[VaB81] Valiant, L. G., and Brebner, G. J., "Universal Schemes for Parallel Communication," in Proc. of the 13th Annual symposium on Theory of Computing, pp. 263-277, 1981.

[Val82] Valiant L. G., " A Scheme for Fast Parallel Communication," SIAM J. Comput., Vol. 11, pp. 350-361, 1982.

[VaB92] Varvarigos, E. A., and Bertsekas, D. P., "A Conflict Sense Routing Protocol and its Performance for Hypercubes," September 1992, submitted *IEEE Trans. on Computers.*

[Var90] Varvarigos, E. A., *Optimal Communication Algorithms for Multiprocessor Computers,* MS. Thesis, Report CICS-TH-192, Center of Intelligent Control Systems, MIT, 1990.