SHORTEST-PATH ROUTING IN HOMOGENEOUS POINT-TO-POINT NETWORKS WITH VIRTUAL CUT-THROUGH SWITCHING

Jennifer Rexford and Kang G. Shin

Real–Time Computing Laboratory
Computer Science and Engineering Division
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
email: {jrexford,kgshin}@eecs.umich.edu
Fax: (313) 763-4617

ABSTRACT

Much research in recent years has focused on various routing strategies for supporting efficient communication in point-to-point networks. Other research has emphasized novel switching techniques, such as *virtual cut-through* and *wormhole* switching. However, little attention has been paid to the subtle interplay between routing and switching schemes. It is desirable to select routing strategies that best exploit the chosen switching scheme. In particular, this paper presents an analysis of various shortest-path routing strategies in wrapped meshes, in the context of virtual cut-through switching. Routing strategies are compared, based on the cut-through probability and the packet delivery-time distribution, using both analytical modeling and simulation. Adaptive routing schemes that can dynamically select from multiple shortest-path links are shown to best capitalize on cut-through switching, particularly in the presence of "hot-spot" network traffic. Simulation results illustrate how inter-node dependencies impact the cut-through probability and packet delivery time. Implementation details are also discussed.

Key Words — Multicomputers, message-passing, routing, shortest-path routing, virtual cut-through switching

The work reported in this paper was supported in part by the ONR under a Graduate Fellowship and Grant N00014-92-J-1080, and by the NSF under Grant MIP-9203895. The opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

1 Introduction

The performance of multicomputers hinges heavily on the availability of fast, scalable interprocessor communication. In point-to-point networks, message exchange is complicated by the lack of a shared communication medium. A packet must travel through one or more nodes en route to its destination, incurring some delay at each hop. Long message latency limits a multicomputer to only those applications with a high degree of locality, since communication with nodes even just a few hops away can become prohibitively time-consuming.

Until recently, message-passing systems generally employed a store-and-forward approach for packet-switching. Sending a packet in the store-and-forward approach requires the entire packet to be buffered at each node in the journey before transmission to the subsequent node can begin. Even in a network free of other traffic, the packet delivery time depends on the *product* of the packet length and the number of hops between the source and destination. This severely limits the scalability of applications on multicomputers.

Virtual cut-through switching, first proposed by Kermani and Kleinrock in [1], can significantly improve delivery time in a message-passing system. This is accomplished by avoiding unnecessary packet buffering at nodes between the source and destination. Several hardware routing controllers designed in recent years have supported some sort of cut-through switching [2, 3, 4].

In a cut-through scheme, if an appropriate link is free at an intermediate node, then the incoming packet can proceed to the next node with only a few machine cycles of delay (four cycles, for example, in the scheme discussed in [5]). Upon reception of the packet header, the router can determine which direction the packet should travel and attempt to reserve the appropriate link. Once the link is reserved, the routing controller sends the updated packet header, followed by the remainder of the packet.

If a packet is unable to cut through an intermediate node and the node lacks the necessary buffer space to temporarily store the packet, the packet can be dropped (and later retransmitted by the source node) to avoid the possibility of deadlock.

Each cut-through in a packet's journey saves the time required to buffer the packet. Thus, achieving a high probability of cut-through can significantly improve delivery time. This paper considers routing strategies for increasing the likelihood of cut-throughs in homogeneous point-to-point networks, such as square meshes and other k-ary n-cubes.

Such wrapped meshes generally have efficient routing algorithms, since any node can be viewed as center of the network without loss of generality. A shortest path between any two nodes can be expressed as the difference between their addresses, where a unique n-tuple is associated with each node. For example, if the source node is $(6,5)$ and the destination node is $(2,2)$ in the square mesh, any shortest path involves four hops in the x-direction and three hops in the y-direction. Thus, addressing information for the packet need only indicate the offset in each direction from the current node to the destination node. In the example above, the initial packet header would include the offsets $(4,3)$. These offsets would decrease as the packet moves closer to its destination. A route between the source and destination involves bringing the offset in each dimension to zero.

Since each node in the network is connected to several links, there are often multiple shortest paths between a source and destination. At many intermediate nodes in a shortest path, there can be more than one suitable direction to send the packet. Taking advantage of these multiple possibilities can increase the probability of establishing cut-throughs at intermediate nodes, thereby improving performance.

This paper investigates routing schemes that can capitalize on virtual cut-through switching. The paper is structured as follows. Section 2 discusses several routing strategies that utilize cut-through switching, with varying performance and hardware complexity. The cut-through probability and delivery-time distribution for these strategies are derived in Sections 3 and 4, respectively. Section 5 investigates hardware implementation issues. A simulator and numerical results with it are presented in Section 6. Section 7 concludes the paper, suggesting possible areas for future work.

2 Routing Schemes

A wide variety of routing schemes for multicomputer networks have been considered in recent years [6, 7]. These schemes can be classified by the type of network information they consider, the type of routes they allow, and how these routes are chosen.

Routing can be *local* or *global*. In local routing, nodes do not exchange explicit routing and traffic information. Although a global scheme is theoretically more likely to generate "optimal" routes (if traffic information can be transmitted and utilized efficiently), it does so at the expense of additional complexity and transmission overhead. Most existing hardware routers in mesh networks employ local schemes for this reason.

The routing schemes can also be characterized as either *minimal* or *non-minimal*. In minimal (or *shortest-path*) routing, only routes of minimum length are considered. In non-minimal routing, packets can be *misrouted* (or *deflected*) in the hope of finding a less congested path. This increased flexibility can improve performance, but can result in more complex routers and the possibility of livelock, where a packet may never arrive at its destination [7, 8].

Routing strategies are also classified as either *deterministic* or *adaptive*. In deterministic (or *static*) routing, there is a single path between each source and destination node. The most common static scheme in point-to-point networks is e-cube routing (also called *dimension-order routing*), in which packets are routed completely in one dimension before proceeding to the next dimension. Static schemes are attractive since they are simple to implement and also preserve packet order. However, such deterministic algorithms often result in unnecessarily long packet latency, since a packet may have to follow a heavily congested path even when another, lightly-loaded route is available. Adaptive schemes can often avoid this problem, at the expense of added costs in hardware and/or time.

Routing schemes should also be categorized based on how they exploit the chosen switching method. In this paper we investigate shortest-path routing schemes that capitalize on virtual cut-through switching. The routing schemes are compared in the context of a wrapped square mesh, although the strategies are equally applicable in other topologies. When there are multiple shortest paths between the current node and the destination, the router can have additional flexibility in selecting the next node in the route. Whenever there is only one direction along a shortest path, a cut-through should be attempted in that direction. If two choices exist, there are three possible schemes:

- *One-choice:* Still attempt a cut-through in a single direction only.
- *One-choice with alternative:* Try one direction first, as in the *one-choice* scheme. If no cut-through can be made in that direction, try the second direction.
- *Diag:* Assign preference to the direction that keeps the packet closer to the "diagonal" between the source and destination. Follow the secondary direction only if a cut-through *can* be established in the secondary direction and *not* in the preferred direction.

In the *one-choice* scheme, the outgoing link can be chosen by a static algorithm, such as e-cube, or randomly. This routing scheme behaves passively, ignoring the possibility of utilizing the second link. If the chosen link is busy, the packet must be buffered, even if the other outgoing link is free. The *one-choice with alternative* scheme adds additional flexibility, by considering a second direction if the first link is busy.

The third scheme goes one step further, aggressively trying to increase the likelihood of future cut-throughs. When a packet must travel x hops in the x-direction and y hops in the y-direction, a cut-through is first attempted in the x-direction if $x > y$; likewise, the y-direction is given preference when $y > x$. The scheme attempts to keep packets close to the diagonal to allow more of the later nodes in the route to have two choices for routing decisions. This increases the probability of cut-throughs in the remainder of the path. This scheme is similar to a technique suggested in [9] for avoiding failures.

The above classification indicates how the router selects among the shortest path options at each node. This does not preclude the use of *misrouting* [4, 7], if all outgoing links along shortest paths are busy, although we will not specifically address this issue.

3 Cut-through Probability

To compare the performance of the three routing schemes, it is necessary to determine p_c, the probability of cut-through, for each scheme. Before deriving p_c for each scheme, we must establish some notation and conventions.

When the source and destination nodes of a packet are separated by h hops, as many as $h-1$ cut-throughs can occur, through the $h-1$ intermediate nodes. Consider a wrapped square mesh with balanced traffic intensity ρ (the utilization factor of the network). Parameter ρ is the probability of a link being busy, and thus unavailable for cut-through. The probability p_c of a packet cutting through a given node depends on ρ and P_2, the likelihood of an intermediate node having two outgoing links along shortest paths.

When there are two outgoing links at a node that run through shortest paths, the probability of finding at least one free shortest-path link is $1-\rho^2$. With average probability $(1-P_2)$, there is only a single link along a minimal route; this one link is free with probability $1-\rho$. Thus, $p_c = (1-\rho^2)P_2 + (1-\rho)(1-P_2) = (1-\rho)(1+\rho P_2)$, where p_c and P_2 are functions of h and ρ. The expected number of cut-throughs for an h-hop packet is then $(h-1)p_c$.

The likelihood of having two routing choices at intermediate nodes depends on the relative placement of the source and destination. We assume that the generation of packets is independent from node to node. The destination of packets is hop-uniform. That is, the destination is selected uniformly from all nodes h-hops away from the source, where the source is labeled (x,y) and the destination is $(0,0)$, with $|x|+|y|=h$. P_2 is found by averaging the probability of two routing choices over source-destination pairs that are h-hops apart. Since the topology is regular, it suffices to consider a single "quadrant" in (x,y), such as $(0,h),(1,h-1),\ldots,(h-1,1)$.

3.1 One-choice scheme

In the one-choice scheme, only one link is considered at each node. Even if the packet could conceivably travel in more than one direction, a single link is chosen statically (e.g., e-cube) or randomly. If the selected link is busy, then the packet must be buffered, since no alternate link is ever considered. Thus, $P_2 = 0$ and $p_c = 1-\rho$.

3.2 One-choice with alternative scheme

The one-choice scheme can be adapted to take advantage of multiple shortest paths between the source and destination. When two choices exist at a node, the first outgoing link is selected as in the one-choice scheme. However, if the chosen link is busy, a cut-through is attempted with the alternate link.

Determining p_c is more complicated in this scheme than in the one-choice case. Consider a packet that must travel x hops in the x-direction and y hops in the y-direction. The source node can be labeled (x,y), with $(0,0)$ as the destination node (see Figure 1), without loss of generality. Any intermediate nodes (i,j) with $0 \le i \le x$ and $0 \le j \le y$ can be on a shortest path from the source to the destination. Any node (i,j) with non-zero i and j has *two* outgoing links along shortest paths; these nodes are called *internal* nodes. At the *border* nodes, with $i=0$ or $j=0$, only *one* outgoing link lies along a shortest path [9]. Thus, P_2 depends on the probability of being at internal nodes during the journey.

Let $S_{(x,y),n}$ be the probability that a packet starting at node (x,y) encounters n two-choice nodes in its route, including the source node. The source node should not be included in the two-choice probability when determining the average cut-through probability, since the packet cannot actually "cut through" the source node. However, the source node is included in $S_{(x,y),n}$ to simplify the derivation; it is subtracted back out before P_2 is determined.

If a node has two possible routing directions, assume the packet travels in the x-direction with probability α, while the y-direction is traveled with probability $1-\alpha$. When the first-

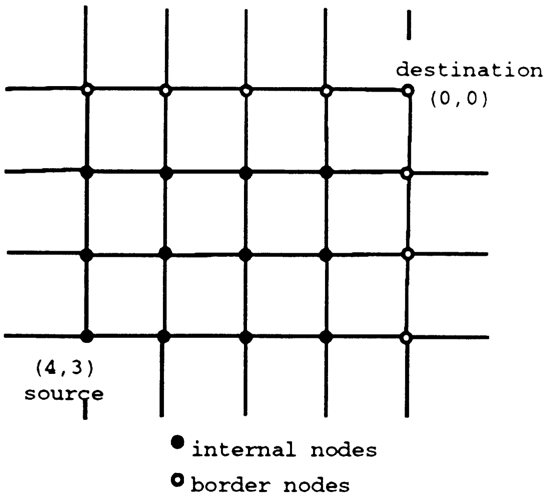


Figure 1: Message in square mesh, with source (4,3)

choice direction is chosen randomly, we have $\alpha = \frac{1}{2}$, independent of the system load ρ, since neither direction has preference over the other. In the e-cube routing strategy one direction (say the x-direction) is tried first; in this case α is not generally $\frac{1}{2}$ and depends on ρ.

When either x or y is zero, there are no internal nodes, since the packet can move in only one direction at each step; thus $S_{(x,y),0} = 1$ and $S_{(x,y),n} = 0$ for $n > 0$. When x and y are non-zero, the source node is an internal node. The remaining internal nodes appear in the routes for $(x-1, y)$ and $(x, y-1)$. Thus, for source node (x, y), with $y = h - x$ and $x, y \neq 0$, we have

$$S_{(x,y),n} = \begin{cases} 0 & \text{if } n = 0 \\ \alpha S_{(x-1,y),n-1} + (1-\alpha) S_{(x,y-1),n-1} & \text{otherwise.} \end{cases}$$

Let $S_{h,n}$ be the sum of $S_{(x,y),n}$ for $x = 0, 1, \ldots, h-1$, where $y = h - x$. Dividing $S_{h,n}$ by h gives the average probability of encountering n two-choice nodes in an h-hop route. This can be used to determine P_2. For $n > 0$ this recurrence yields

$$S_{h,n} = \sum_{\substack{x=1 \\ y=h-x}}^{h-1} S_{(x,y),n} = \alpha \sum_{x=1}^{h-1} S_{(x-1,y),n-1} + (1-\alpha) \sum_{x=1}^{h-1} S_{(x,y-1),n-1}$$

which simplifies to

$$S_{h,n} = \alpha \sum_{\substack{x=0 \\ y=(h-1)-x}}^{(h-1)-1} S_{(x,y),n-1} + (1-\alpha) \sum_{\substack{x=1 \\ y=(h-1)-x}}^{h-1} S_{(x,y),n-1}.$$

If $n - 1 = 0$, both summations evaluate to 1 because $S_{(0,y),0} = S_{(x,0),0} = 1$, while all other terms are zero, thus resulting in $S_{h,1} = 1$. If $n - 1$ is positive, $S_{(0,y),n-1} = S_{(x,0),n-1} = 0$, so both sums are on $x = 1, \ldots, (h-1) - 1$. Thus, for $n > 1$,

$$S_{h,n} = \alpha S_{h-1,n-1} + (1-\alpha) S_{h-1,n-1} = S_{h-1,n-1}.$$

Hence, $S_{h,n} = S_{h-1,n-1} = \ldots = S_{h-n,0}$. Since $S_{h,0} = 1$ for all $h > 0$, this implies $S_{h,n} = 1$, for $n = 1, \ldots, h-1$. So, $S_{h,n} = 1$ for $0 \leq n < h$. Thus, for h-hop packets, those routes with $0, 1, \ldots, h-1$ two-choice nodes are encountered with equal probability $\frac{1}{h}$.

As mentioned earlier, before determining P_2, we must subtract the source node. Any route with one or more two-choice nodes has the source as one of these nodes. So, each $n > 0$ decreases by one two-choice node, thus leading to:

$$P[N = n \text{ two-choice intermediate nodes}] = \begin{cases} \frac{2}{h} & \text{if } n = 0 \\ \frac{1}{h} & \text{if } n = 1, 2, \ldots, h-2 \,. \end{cases}$$

So, P_2, the likelihood of an intermediate node having two shortest-path links, equals the expected number of two-choice nodes divided by the number of steps to intermediate nodes $(h-1)$. This gives

$$P_2 = \frac{1}{h-1} \sum_{n=0}^{h-2} n \cdot P[N = n] = \frac{1}{2} - \frac{1}{h}.$$

So for the *alternate* scheme the cut-through probability simplifies to

$$p_c = (1-\rho)\left(1 + \rho\left(\frac{1}{2} - \frac{1}{h}\right)\right) .$$

For large h this approaches $p_c = (1-\rho)(1+\rho/2)$. Thus, considering the second link, when possible, can increase the cut-through probability by a factor of $\rho/2$ over the standard *one-choice* schemes. This suggests that considering an alternate link when the first link is busy can improve packet delivery time over the one-choice scheme.

It is interesting to observe that P_2, and hence p_c, is independent of α. This implies that choosing the first-choice link randomly (instead of using a static strategy, such as e-cube) does not necessarily improve the likelihood of two-choice nodes. On average, both approaches lead the packet to a *border* node in the same number of hops.

3.3 Diag

As seen in the previous section, taking advantage of multiple shortest paths between a pair of nodes can improve the likelihood of cut-throughs. The *one-choice* scheme ignores this opportunity; the *one-choice with alternate* scheme capitalizes on it; the *diag* scheme tries to create such opportunities.

The *diag* scheme is a more aggressive approach. It is often possible to have two-choice opportunities up until the last hop of a packet, whereas the *one-choice with alternate* scheme has two choices less than half of the time, on average. In the *diag* scheme the packet moves toward the "diagonal" between the source and destination whenever it can cut through in that direction. The alternate direction, away from the diagonal, is utilized only when a cut-through can be established in this direction, and not in the preferred direction. If the packet must be buffered, it is buffered in the preferred direction. In other words, when no cut-through can be established in the preferred direction, take the "sure-thing" of a cut-through in the alternate direction, if possible.

Determining the cut-through probability for the *diag* scheme follows an analysis similar to that in the previous section. Let α be the likelihood of traveling in the preferred direction. Accordingly, $1 - \alpha$ is the probability of moving in the alternate direction. This only occurs when the preferred link is busy and the alternate one is free, so $1 - \alpha = \rho(1 - \rho)$, resulting in $\alpha = 1 - \rho + \rho^2$. Since $0 \leq \rho \leq 1$, α has a minimum value of $\frac{3}{4}$. So, no matter what the system load is, packets travel in the preferred direction at least three-fourths of the time.

At node (x, y), the x-direction is *preferred* whenever $x > y$; when $y > x$ the y-direction is preferred. When $x = y$, either direction can be considered the first choice; for this analysis, it is assumed that the x-direction is preferred, but it can be shown that this assumption does not affect the result of the derivation.

As in the previous section, if $x = 0$ or $y = 0$, $S_{(x,y),n} = 1$ for $n = 0$ and 0 for all $n > 0$. For $x, y \neq 0$,

$$S_{(x,y),n} = \begin{cases} 0 & \text{if } n = 0 \\ \alpha S_{(x,y-1),n-1} + (1-\alpha)S_{(x-1,y),n-1} & y > x, n > 0 \\ \alpha S_{(x-1,y),n-1} + (1-\alpha)S_{(x,y-1),n-1} & x \geq y, n > 0\,. \end{cases}$$

When $n = 0$ only the $(0, h)$ source contributes to $S_{h,n}$, so $S_{h,0} = 1$. For $n > 0$ and even h,

$$S_{h,n} = \sum_{\substack{x=1 \\ y=h-x}}^{h-1} S_{(x,y),n} = \sum_{\substack{x=1 \\ y=h-x}}^{\frac{h}{2}-1} \{\alpha S_{(x,y-1),n-1} + (1-\alpha)S_{(x-1,y),n-1}\} + \sum_{\substack{x=\frac{h}{2} \\ y=h-x}}^{h-1} \{\alpha S_{(x-1,y),n-1} + (1-\alpha)S_{(x,y-1),n-}$$

These summations can be simplified to

$$S_{h,n} = \alpha(S_{h-1,n-1} + S_{(\frac{h}{2}-1,\frac{h}{2}),n-1}) + (1-\alpha)(S_{h-1,n-1} - S_{(\frac{h}{2}-1,\frac{h}{2}),n-1})\,.$$

So, for even h we have $S_{h,n} = S_{h-1,n-1} + (2\alpha - 1)S_{(\frac{h}{2}-1,\frac{h}{2}),n-1}$, when $n > 1$. A similar expression can be found for odd h, resulting in

$$S_{h,n} = \begin{cases} S_{h-1,n-1} + (2\alpha-1)S_{(\frac{h}{2}-1,\frac{h}{2}),n-1} & \text{for even } h \\ S_{h-1,n-1} + (2\alpha-1)S_{(\frac{h-1}{2},\frac{h-1}{2}),n-1} & \text{for odd } h\,. \end{cases}$$

When $\alpha = \frac{1}{2}$ these two recurrences both reduce to the *one-choice with alternate* case. So, for the *diag* scheme, as n grows, so does $S_{h,n}$, since routes with a larger number of two-choice nodes are more likely. These recurrences can be used to generate P_2, as in the previous section.

The plots in Figure 2 show the expected number of cut-throughs for each scheme, for $h = 5$ and $h = 20$ as a function of system load ρ. The benefits of the *alternate* and *diag* schemes are more apparent for packets that travel a larger number of hops. For example, when $h = 20$, when the diag scheme is used in a system with 50% load, it can provide about the same number of cut-throughs as the one-choice scheme provides in a system operating at just 35% load.

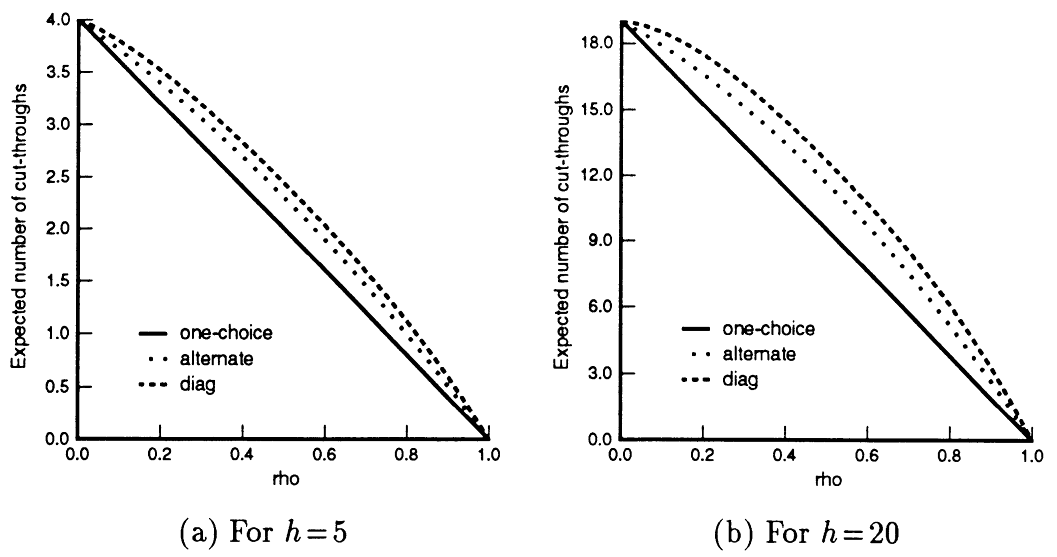


Figure 2: Expected number of cut-throughs

Since packet bufferings are so time-consuming, even a slight improvement in the cut-through probability can have a significant impact on packet delivery time. The effects of the routing schemes on delivery time are investigated in the next section.

4 Delivery-Time Distribution

In the previous section we derived the expected number of cut-throughs for each of the three schemes, in terms of h and ρ. This provides a good model of *average* performance for each scheme, but mean behavior is not always a sufficient measure of system performance. The delivery-time distribution provides more complete information about system performance.

4.1 Components of packet delay

The delivery time for a packet has three largely independent components: inspection of header bytes, transmission, and queueing delay (when the packet is buffered). For our purposes, time is normalized to the time required to send one byte across a link. The distributions for the three sources of delay are determined separately, then combined together to form the delivery-time distribution.

Whether or not a cut-through occurs, the packet incurs a delay of τ units at each node for the inspection and modification of the header bytes. So, let τ be the time between the time the first byte of a packet is sent from the preceding node to when this byte is sent across the outgoing link. The *one-choice with alternate* and *diag* schemes can require additional time, if a second outgoing transmitter must be considered. For now we assume the delay is always τ; this assumption is relaxed in the simulator in Section 6. For an h-hop packet, this consumes $h\tau$ time units.

If no buffering occurs along the packet's route, the only other source of delay is the time to transmit a packet. Time $h\tau$ is required to get the packet to the switch at the destination node. If the packet is ℓ bytes long, then ℓ time units are necessary to complete packet delivery.

Any remaining packet delay is caused by bufferings at the source and intermediate nodes. This can be modeled by determining the queueing delay that a buffered packet must incur at a node.

To simplify the derivation, the following assumptions are made:

1. Generation of packets is Poisson with rate λ at each node.
2. Packet lengths are exponentially distributed with mean $\bar{\ell}$ bytes.
3. Packet length is regenerated at each node in the packet's route.
4. Packets are delayed by time τ at each node before a routing/buffering decision is made

The first three assumptions are necessary for tractability and are compatible with the analytical work in [1]. Assumption 3 provides independence between the nodes and is shown in [10] to be realistic. Assumptions 3 and 4 are relaxed in the simulator. Since time is normalized to the time to transmit a byte across a link, the service rate μ of the links is $1/\bar{\ell}$.

Since packet lengths are exponentially distributed, the cdf F_L of transmission time is $P(L \leq t) = 1 - e^{-\mu t}$. Header byte inspection/modification contributes a constant delay, which simply shifts the distribution. Queueing delay, with cdf F_Q, contributes the remainder of the delivery time. The components of delay can be combined by determining $P(L + Q + h\tau \leq t)$, but first the distribution of Q must be found.

4.2 Queueing delay

The queueing delay is what differentiates the three routing schemes. The *one-choice* scheme results in fewer cut-throughs, so packets spend more time in queues. The *one-choice with alternate* and *diag* schemes alleviate this. The derivation of the queueing delay distribution follows the approach in [11], which presents the distribution for the one-choice scheme. This must be generalized to incorporate the other two schemes.

When a packet is unable to cut through a node, it enters a FIFO queue at the node and awaits service. Jackson's theorem [10] indicates that the collection of queues has a product form solution. An h-hop packet incurs queueing delay at the source node and it may also be buffered at any of the $h - 1$ intermediate nodes. If a packet gets buffered, it spends time T_i at the node, where T_i is a random variable. When an h-hop packet is buffered at d intermediate nodes, the total time spent buffered at nodes is $Q = \sum_{i=0}^{d} T_i$. Thus,

$$P[Q \leq t] = \sum_{d=0}^{h-1} P\left[\sum_{i=0}^{d} T_i \leq t\right] \cdot P[\text{packet buffered at } d \text{ intermediate nodes}] .$$

The term $P[\sum_{i=0}^{d} T_i \leq t]$ corresponds to an Erlang distribution [11]. The family of Erlang distributions is described by two parameters (λ, k); the density function

$$f_X(x) = \frac{\lambda(\lambda x)^{k-1}}{(k-1)!} e^{-\lambda x}$$

measures the time needed for k events to occur from a Poisson process [10].

The d-node queueing delay can be described by $ERL(\mu(1-\rho), d+1)$, where μ is the service rate for the links. This results in the density function

$$f_Q(t) = \sum_{d=0}^{h-1} P[\text{packet buffered at } d \text{ intermediate nodes}] \cdot [\mu(1-\rho)]^{d+1} \frac{t^d e^{-\mu(1-\rho)t}}{d!} .$$

The three routing schemes have different queue-delay distributions because they each have a distinct P[packet buffered at d intermediate nodes].

It remains to determine P[packet buffered at d intermediate nodes] for the three routing schemes, so the queueing-delay distributions can be generated and compared. For nodes with only a single shortest-path link, packets are buffered with probability ρ; if two links can be considered, buffering occurs with probability ρ^2. The d nodes that block the progress of the packet can be chosen from both the one-choice and two-choice nodes. Consider an h-hop route with n two-choice intermediate nodes and $h-1-n$ one-choice intermediates. The packet is buffered at d intermediate nodes with probability P[buffered at d intermediate nodes$|n$] =

$$\sum_{k} \binom{n}{k} (\rho^2)^k (1-\rho^2)^{n-k} \cdot \binom{h-1-n}{d-k} \rho^{d-k} (1-\rho)^{(h-1-n)-(d-k)} .$$

Thus,

$$P[\text{buffered at } d \text{ nodes}] = \sum_{n} P[\text{buffered at } d \text{ nodes}|n] \cdot P[n \text{ two-choice intermediates}] .$$

Each of the three routing schemes has a different expression for P[n two-choice intermediates], as derived in the previous section.

For the one-choice scheme, the expression for P[buffered at d nodes] simplifies to

$$\binom{h-1}{d} \rho^d (1-\rho)^{h-1-d}$$

since all intermediate nodes are, in effect, one-choice nodes. The one-choice with alternate scheme improves delivery time because alternative links are considered. A route with a large number of two-choice nodes (i.e., n large) has a better delivery-time distribution than a route with fewer two-choice nodes. The diag scheme further enhances packet delivery time by increasing $P[n$ two-choice intermediate nodes] for larger n.

4.3 Combining the delays

Since packet length is independent from node to node (by Assumption 3), the transmission and queueing delays are independent and can be combined by convolving the distributions [1]:

$$F_{L+Q}(t) = P(L + Q \leq t) = \int_0^t F_L(t - y) f_Q(y)\, dy$$

with the integration over $[0, t]$, since L and Q take on only positive values. The integration can be performed using the expressions for F_L and f_Q, and simplified. This derivation is deferred to the Appendix for the sake of brevity. The delivery-time distribution, then, is $F_{L+Q}(t)$, shifted right by $h\tau$.

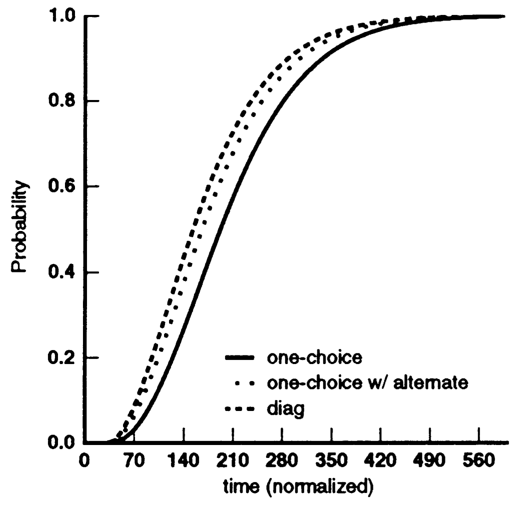
It remains to determine the relationship between the system load ρ (traffic intensity) and the service rate μ. When the packet generation rate is λ and each packet travels h hops between the source and destination, the link throughput is $\lambda h/4$ (in the case of a square mesh). Thus, $\rho = \lambda h/4\mu = \lambda h\bar{\ell}/4$.

Figure 3 shows delivery-time distributions for the three schemes for an 8-hop packet in a system with an average packet length of 24 bytes and $\tau = 4$ time units for handling packet header bytes. As the plot shows, taking advantage of two-choice nodes increases the likelihood of quick packet delivery. For example, in Figure 3(a), the one-choice with alternate scheme delivers over 95% of packets in less than 355 time units, whereas nearly 40 more time units are required in the one-choice scheme for the same likelihood of delivery.

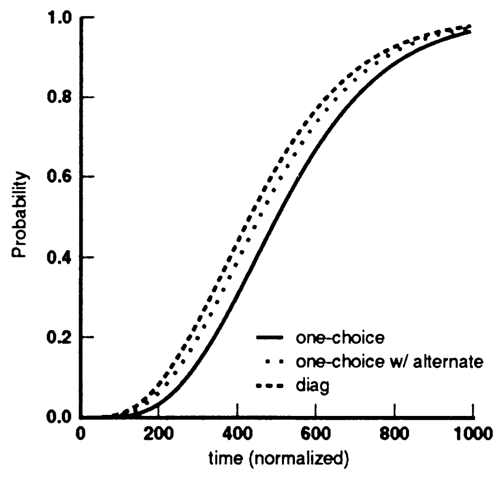
5 Implementation Issues

The delivery-time distribution illustrates how the performance of a switching method, such as virtual cut-through, can be quite dependent on the routing scheme. As the routing schemes get more complex, the benefits of virtual cut-through switching are better exploited. However, the more complicated schemes also require more elaborate (thus complex) routing controllers.

Two general approaches currently prevail in hardware routing controllers. Many controllers employ some type of crossbar switch, with buffering, between the input and output ports of each node [2, 4, 5]. Such a scheme is shown in Figure 4. To support cut-through switching, each input port has hardware to inspect the header byte(s) of an incoming packet and make a routing decision. Since the next node cannot be determined instantly, a small buffer is necessary to store the first few bytes of the packet.



(a) For $\rho = 0.40$



(b) For $\rho = 0.70$

Figure 3: Delivery-time distribution for $h = 8$, $\bar{\ell} = 24$ bytes

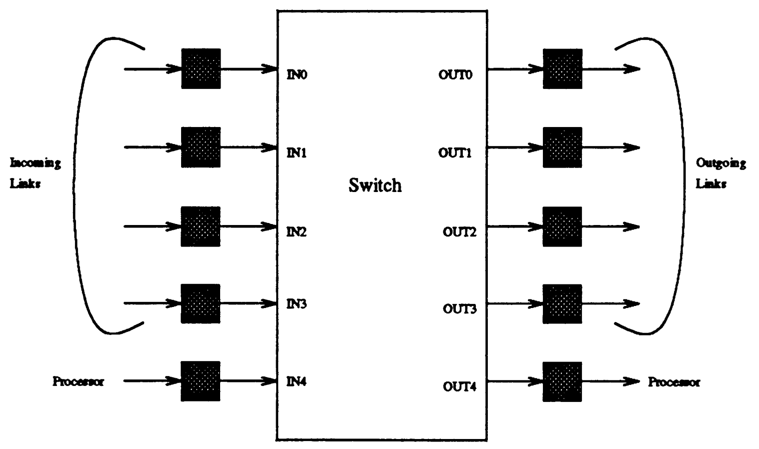


Figure 4: Switch for router in a square mesh

If the chosen link is available, a circuit is established through the switch. The updated header (and the rest of the packet) can then proceed to the next node. If the link is unavailable, the packet is directed to a queue. Several buffering schemes (such as output buffering, input buffering, and other variations) are possible [5].

Another approach separates the buffering from the packet transmission/reception by providing a separate, shared memory for buffered packets [3, 12]. This allows for larger packet sizes and greater flexibility, at the expense of providing an efficient, shared interface between the transmitters/receivers and the memory.

The various routing schemes analyzed thus far are applicable to both approaches. To provide a single framework for comparing the routing schemes, we consider the switch architecture in greater detail in the simulator. The *one-choice* scheme is attractive for its simplicity. Particularly with a static scheme (such as *e*-cube), the router quickly recognizes which outgoing link to consider. If that link is busy, the packet is buffered.

The *two-choice* scheme (e.g., one-choice with alternate and diag) requires more hardware support. As in the one-choice scheme, the first few bytes of the packet must be received and analyzed before the router can try to gain control of an outgoing link. When a packet can cut through to the first link, the delay for the *two-choice* scheme is the same as with the one-choice scheme.

Considering a second link requires more flexibility, as well as some additional time. In the meantime, a few more bytes of the packet can arrive from the previous node, so a slightly larger on-line buffer may be necessary to hold these incoming bytes until the final routing decision is made. In existing routing controllers [3, 5], around four bytes are sufficient for a one-choice scheme. An eight byte on-line buffer, then, should suffice for the schemes that consider an alternate transmitter.

If the cut-through is established through the alternate link, additional cycles are also required to make the second routing decision. Still, this is significantly less time-consuming than buffering the entire packet, which would have to occur in the one-choice scheme if the first (and only) link is busy. Instead of just four to eight cycles for header inspection, buffering the packet incurs a delay proportional to the packet length, in addition to any queueing delay.

Supporting the diag scheme also requires logic to compare the relative address offsets in each direction and determine the preferred link. Selecting the preferred direction requires a comparator or subtracter. Even in the one-choice scheme, though, some sort of counter/decrementer is required to update the addresses in the packet header.

Schemes that can consider a second link at each node necessitate more complicated routing control. However, with the additional complexity comes the potential for improved system performance. As systems grow in size, providing fast communication becomes progressively more difficult. More complex routing strategies will be necessary to support packet delivery in these larger systems.

Although we have used square meshes to describe the routing schemes, these schemes are also applicable to higher dimensional networks, such as k-ary n-cubes. In other topologies the packet router may be able to select from more than two outgoing links. In this situation, the controller can be extended to consider additional links. Or, to minimize complexity and overhead, the router could still consider at most two links, as in planar-adaptive routing [8].

6 Simulation Results

The analytical expressions in Sections 3 and 4 provide an estimate for the performance of the routing schemes in a cut-through switching environment. However, some lower-level design decisions and inter-node dependencies are difficult to model mathematically. To gain deeper insight into the interplay of routing and cut-through switching, a discrete-event simulator was developed to model the switch architecture and routing protocols in greater detail.

6.1 Model

The torus simulator models the generation, transmission, and routing of packets at each node, including the overhead of packet header inspection and switch arbitration in the routers. The simulator models each node as a 5×5 switch with input-queueing with fully-connected buffers [5]. That is, each input port has four queues associated with it, one for each of the three outgoing links (not including the link that is paired with the incoming link) and one for the node itself (for packets reaching their destination node).

When a packet cannot cut through a node, it is queued at its incoming link in one of the four buffers. Since each input port has separate lines to each possible output port, an arriving packet destined for one output need not wait in a single queue behind an earlier packet waiting for a different outgoing link to become available.

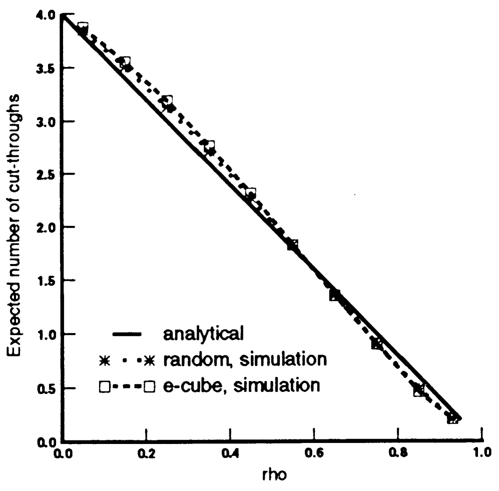
The three routing schemes are applicable to other switch architectures and buffering schemes. A single model is chosen in order to have a consistent framework for studying the routing algorithms.

As in the analytical modeling, packets are generated independently at each node at rate λ with packet lengths exponentially distributed with mean $\bar{\ell}$. However, Assumptions 3 and 4 in Section 4 are not made in the simulator.

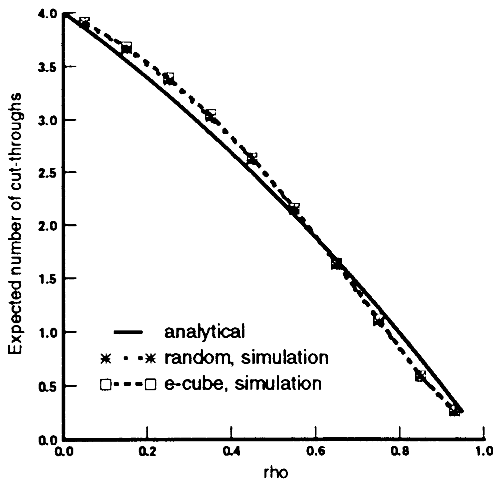
6.2 Cut-through probability

Figure 5 shows simulation results for the average number of cut-throughs for the one-choice and one-choice with alternate schemes. The curve for the diag scheme is similar. The simulation data match the analytical results (from Section 3) fairly well, but the graphs have an interesting symmetry. At low load, the simulated packets consistently experience slightly more cut-throughs than the analytical model predicts; at higher loads, the reverse occurs.

Looking at the behavior of the packets in greater detail provides an explanation for this result. At 25% load ($\rho = 0.25$) in the one-choice scheme in Figure 5, we expect a cut-through probability of 0.75 ($= 1 - \rho$). The observed probability, though, is 0.778. Looking at the



(a) One-choice scheme



(b) One-choice with alternate scheme

Figure 5: Expected number of cut-throughs for $h = 5$ in 16×16 Torus

packet behavior one hop at a time, on average a packet establishes a cut-through on a hop with probability 0.80 when the *previous* hop was also a cut-through. This probability drops to 0.68 when the *previous* hop required buffering. In turn, the probability of buffering when the previous hop resulted in a cut-through is 0.20; if the previous hop also caused buffering, this probability increases to 0.32.

In the one-choice scheme only one outgoing link is considered at each node, irrespective of the behavior of previous hops, so this type of correlation is seemingly unusual. Figure 6 illustrates this phenomenon for the random, one-choice scheme across the range of system loads. The solid line shows the (simulated) cut-through probability. The probability of a cut-through when a cut-through occurred on the previous hop and the probability of a cut-through when the previous hop resulted in a buffering are also shown.

The impact of the previous hop holds throughout the ρ values. A previous cut-through encourages an additional cut-through, while buffering a packet is more likely to lead to future packet bufferings. At low loads, where p_c is already high, this works to *increase* the number of cut-throughs, since the frequent cut-throughs perpetuate themselves. However, at higher loads, buffering is more likely, so the number of cut-throughs is lower than anticipated. Effectively, then, in Figure 6 the solid curve is a weighted average of the other two curves.

This correlation occurs because in the simulator, as in an actual system, the nodes are not truly independent. If a packet must buffer on one hop, then it eventually gets sent to the next node, right behind at least one other packet. It is more likely, then, that the next outgoing link will be busy. Likewise, if a packet can cut through a node, then the link it uses has been free for some time. Thus, the next node is less likely to be loaded, since it has not had to deal with

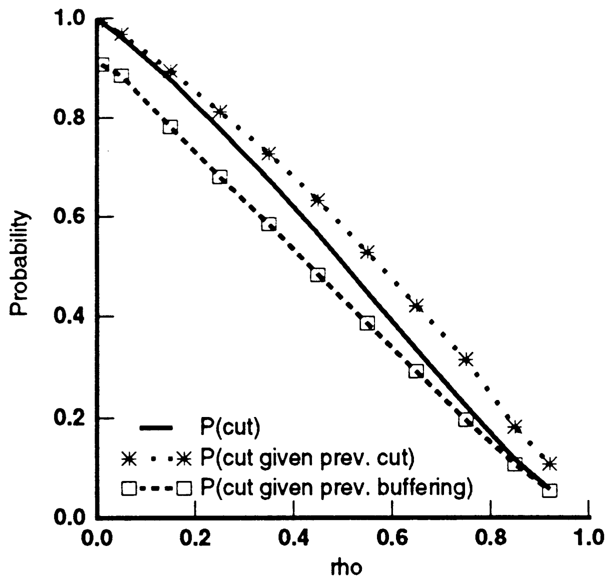


Figure 6: p_c in simulation for random one-choice scheme

that incoming link for some length of time. Instead, the node could deal with traffic on the other incoming links.

Independence is still shown to be a fairly accurate assumption. The violation of independence in "real life" is not entirely arbitrary, though, and produces interesting symmetries as those seen in Figures 5 and 6. Since cut-throughs have a tendency to perpetuate themselves, routing schemes that increase the cut-through probability are further enhanced. Buffering at a node not only costs the packet buffering and queueing time, but may decrease the likelihood of future cut-throughs.

6.3 Delivery-time distribution

To analyze the performance of the routing schemes in greater detail, the simulator captures the delivery-time distributions for the various routing schemes. Figure 8 shows the distributions in a 24×24 torus, where packets travel 10 hops to reach their destinations. The graphs illustrate a number of interesting phenomena.

Simulation vs. analytical

For all three routing schemes in Figure 8, the likelihood of fast delivery (low communication latency) is *higher* in the simulation than in the analytical results. Interestingly, though, the simulation curves cross the analytical results at larger delivery time.

At small delivery times, the higher cut-through probabilities (as described in Section 6.2) result in faster delivery. This explains why the simulation curves are *above* the analytical ones on the left part of the graphs. In the right part of the graph, the analytical results predict slightly faster delivery than the simulation. Packets that incur a larger delay generally experience few cut-throughs and are subject to queueing delays at several intermediate nodes.

The simulation captures some delays not accounted for in the analytical expressions. In the simulator, an outgoing link must wait at least one time unit between transmitting the last byte of one packet and the first byte of the next packet. This is necessary for the switch to set up a new connection to the outgoing link. The link is effectively unavailable during this time.

When the routing scheme considers a second link, additional delay occurs. This affects the diag and one-choice with alternate schemes. In the analytical model, making a routing decision requires a constant time τ. In actuality, if a second routing choice is considered, this delay is doubled. If a packet is buffered many times in its route, this extra header delay is incurred on several hops.

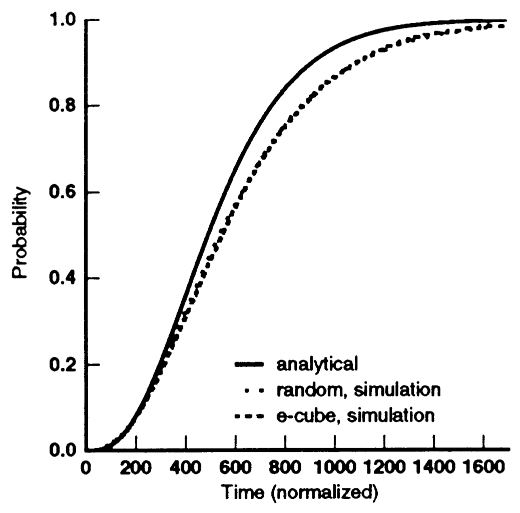
In addition, the analytical model includes queueing delay, but this does not always include buffering time. In the analytical model, a packet that cannot cut-through an intermediate node is forwarded to the subsequent node when the desired outgoing link becomes free. However, it is possible for the outgoing link to become available *before* the packet completes buffering. In this situation, the packet must complete buffering before it can be sent to the outgoing link (in traditional virtual cut-through switching). Thus, the outgoing link may be idle for some time, waiting for the packet to complete buffering. The simulator can capture this type of delay.

Figure 7 shows the delivery-time distributions for the three schemes in a heavily-loaded network (at 75% load). As described in Section 6.2, at higher loads the analytical expressions estimate a slightly *higher* cut-through probability than is actually realized; thus, the analytical results predict faster packet delivery. So, the simulation delivery-time curves tend to stay *below* the analytical curves at higher load. The simulated diag scheme, in Figure 7(b), performs nearly as well as the analytical results would suggest. This occurs because the diag scheme has a higher cut-through probability than the other two schemes, and thus can continue to propagate cut-throughs even at higher loads.

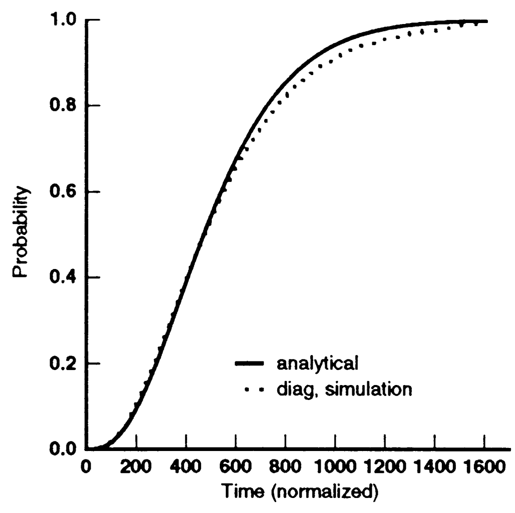
E-cube vs. random link selection

Another interesting feature of the graphs is the difference in performance between the random one-choice and *e*-cube one-choice scheme in Figure 8(a); the *e*-cube scheme results in faster packet delivery. This phenomenon has also been observed in [13, 14]. In *e*-cube routing, a packet entering a node in one direction tends to depart that node traveling in the same dimension. So, packets arriving on different incoming links are less likely to go through the same outgoing link, thus decreasing the likelihood of contention. This effect is more dramatic in higher-dimensional networks and in packets traveling a large number of hops, but is still observable in a 2D network.

The above phenomenon was not noticeable in Figure 5 because packets travel only 5 hops. The difference between random and *e*-cube routing is not significant in the *one-choice with*



(a) One-choice w/ alternate scheme



(b) Diag scheme

Figure 7: Delivery-time for $h = 5$, $\rho = 0.75$ in 16×16 torus

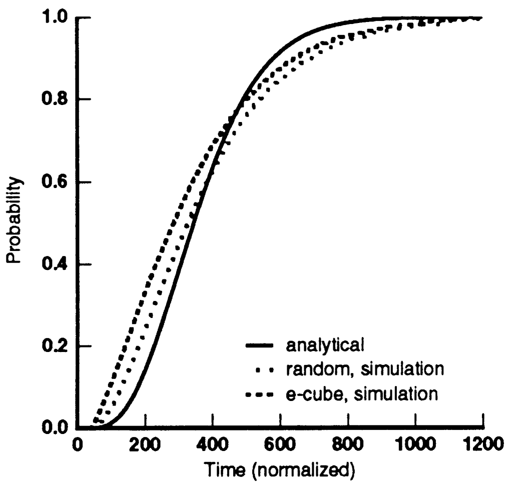
alternate scheme in Figure 8(b). When alternate links are considered, conflicts between packets arriving at the same node can be resolved by dynamically selecting a second-choice link for one of the packets.

The three schemes

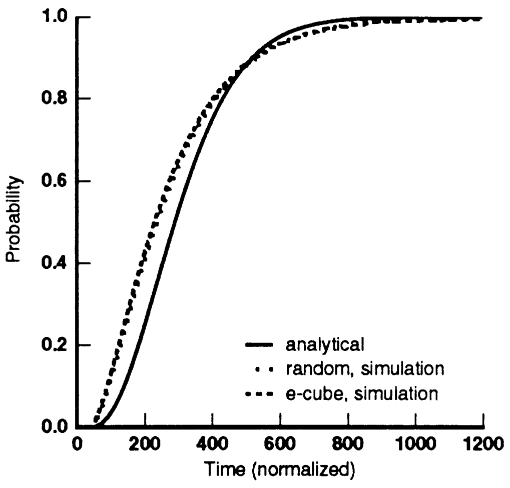
The *one-choice with alternate* and *diag* schemes consistently outperform the *one-choice* scheme at varying loads, number of hops, and packet lengths. Considering a second transmitter has a significant impact on delivery time. Moving from the *alternate* scheme to the *diag* scheme has a less dramatic effect. However, the diag scheme further improves upon the other two schemes at higher load and a larger number of packet hops.

Further simulation runs show that the diag scheme also excels with larger packet sizes. As packet lengths grow, the cost of buffering escalates as well. The more complex schemes invest more effort in avoiding unnecessary packet bufferings, so they deliver large packets more quickly than the one-choice scheme.

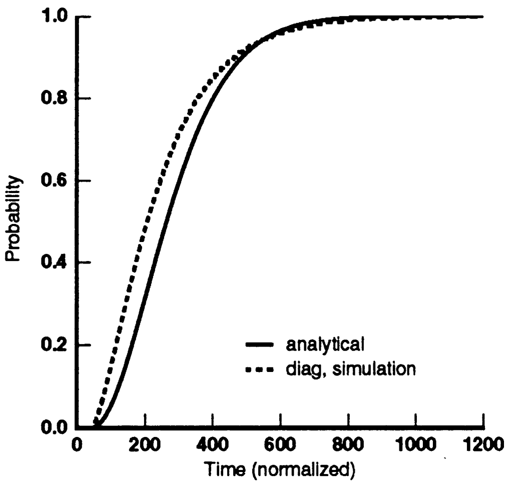
The improved performance from the diag scheme may not outweigh its hardware costs for small-scale systems; a one-choice with alternate scheme would suffice. However, as networks and packet sizes continue to grow, more aggressive schemes such as diag become more attractive.



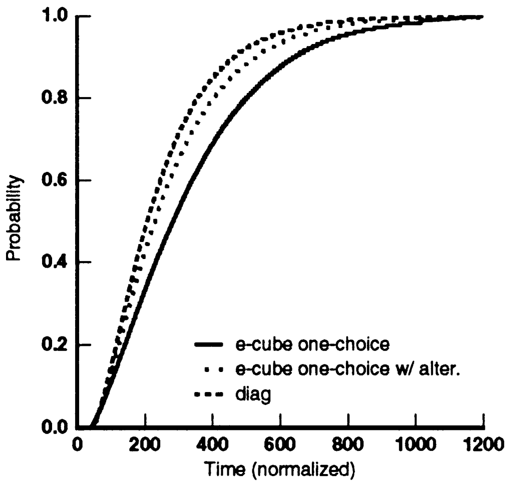
(a) One-choice scheme



(b) One-choice with alternate scheme



(c) Diag scheme



(d) All three schemes (simulation only)

Figure 8: Delivery-time for h=10, $\rho = 0.45$, in a 24×24 torus

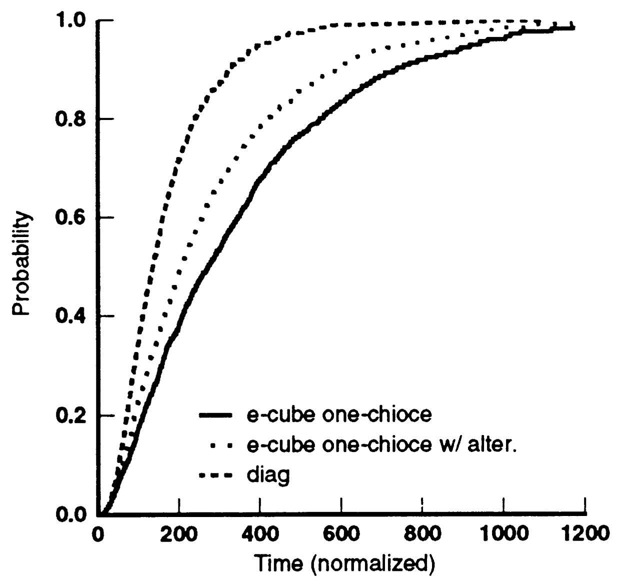


Figure 9: Delivery-time of hot packets in 16 × 16 torus ($\rho = 0.25$), 3% hot

7 Conclusions and Future Work

Greater flexibility in selecting routes gives packets more opportunities to cut through intermediate nodes. The *one-choice* scheme does not take advantage of these opportunities, since only a single outgoing link is considered at each node. The *one-choice with alternate* scheme capitalizes on multiple shortest paths between the source and destination. The *diag* scheme takes a more aggressive approach; it is biased toward shortest-path routes with a larger number of two-choice nodes.

Several other issues can be investigated in this context. In particular, the impact of non-uniform communication patterns merits attention. Although the routing schemes considered here are *local*, the *alternate* and *diag* schemes do make use of some basic traffic information. We believe that these schemes will offer further performance advantages in the presence of non-uniform traffic.

For example, Figure 9 shows the delivery-time distribution (from simulation) in a torus where "normal" packets account for 97% of the traffic. The remaining 3% is "hot traffic," all directed at one node (a *many-to-one* communication, as might be used for synchronization or access to a single server). The more flexible routing schemes allow packets to route around heavily-loaded links. The diag scheme is particularly useful here, since it maximally allows two routing choices, even as the packet gets close to the hot node. These "hot spot" issues warrant further investigation, and the corresponding results will be reported in a forthcoming paper.

Also, to improve system performance, alternate links can be considered even *after* a packet has been buffered. The final routing decision can be postponed until a suitable outgoing link is free, so the packet can be sent out as early as possible. In this paper, it is assumed that multiple links are considered only for cut-through and that, after buffering, the packet is assigned to a queue for a single link. Considering alternate links after packet buffering effectively allows a "shortest queue first" strategy when choosing the outgoing link. Trying alternate links, before *and* after packet buffering, can allow a packet to avoid a congested region (i.e., buffering at a heavily-loaded link) if another suitable link is free. Flexibility in selecting outgoing links for packets can significantly improve delivery time.

Scalable multicomputing demands fast communication. Traditional store-and-forward packet transmission will be unacceptable for large distributed systems. Instead, some type of cut-through switching will be necessary to avoid unnecessary packet delay. Each cut-through in a packet's journey significantly reduces delivery time, since buffering the packet is quite time-consuming. The power of cut-through switching is complemented by the need to support routing schemes that can exploit this power. We have shown in this paper that an emphasis on the interplay between routing and switching schemes can significantly improve communication performance.

Acknowledgments

We would like to thank Jim Dolter for his constructive suggestions for the simulator. Thanks also to Ashish Mehra for several helpful discussions.

References

[1] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, vol. 3, no. 4, pp. 267–286, September 1979.

[2] W. J. Dally and C. L. Seitz, "The torus routing chip," *J. Distributed Systems*, vol. 1, no. 3, pp. 187–196, 1986.

[3] J. W. Dolter, P. Ramanathan, and K. G. Shin, "A microprogrammable VLSI routing controller for HARTS," in *International Conference on Computer Design: VLSI in Computers*, pp. 160–163, October 1989.

[4] S. Konstantinidou and L. Snyder, "Chaos router: Architecture and performance," in *Proc. Int'l Symposium on Computer Architecture*, May 1991.

[5] Y. Tamir and G. Frazier, "High-performance multi-queue buffers for VLSI communication switches," in *Proc. Int'l Symposium on Computer Architecture*, pp. 343–354, June 1988.

[6] A. Tanenbaum, *Computer Networks*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

[7] S. Felperin, L. Gravano, G. Pirarre, and J. Sanz, "Routing techniques for massively parallel communication," *Proceedings of the IEEE*, vol. 79, no. 4, pp. 488–503, April 1991.

[8] A. A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," in *Proc. Int'l Symposium on Computer Architecture*, pp. 268–277, Australia, May 1992.

[9] H. G. Badr and S. Podar, "An optimal shortest-path routing policy for network computers with regular mesh-connected topologies," *IEEE Trans. Computers*, vol. C-38, no. 10, pp. 1362–1370, October 1989.

[10] L. Kleinrock, *Queueing systems*, volume I: Theory, John Wiley & Sons, 1975.

[11] J. W. Dolter, P. Ramanathan, and K. G. Shin, "Performance analysis of virtual cut-through switching in HARTS: A hexagonal mesh multicomputer," *IEEE Trans. Computers*, vol. 40, no. 6, pp. 669–680, June 1991.

[12] O. Menzilcioglu and S. Schlick, "Nectar CAB: A high-speed network processor," in *Proc. Int'l Conf. on Distributed Computing Systems*, pp. 508–515, May 1991.

[13] A. Agarwal, "Limits on interconnection network performance," *IEEE Trans. Parallel and Distributed Systems*, vol. 2, no. 4, pp. 398–412, October 1991.

[14] S. Abraham and K. Padmanabhan, "Performace of the direct binary n-cube network for multiprocessors," *IEEE Trans. Computers*, 1987.

Appendix A Derivation of Delivery-time Distribution

We want to simplify

$$F_{L+Q}(t) = P(L+Q \le t) = \int_0^t F_L(t-y) f_Q(y)\, dy$$

where $f_Q(y) = \sum_{d=0}^{h-1} c_d y^d e^{-\mu(1-\rho)y}$, with $c_d = P[\text{buffered at } d] \cdot [\mu(1-\rho)]^{d+1}/d!$. So,

$$F_{L+Q}(t) = \sum_{d=0}^{h-1} c_d \cdot \left(\int_0^t y^d e^{-\mu(1-\rho)y} (1 - e^{-\mu(t-y)})\, dy \right).$$

The integral can be simplified by using the following relation [11]:

$$\int x^m e^{ax} dx = \frac{e^{ax}}{a} \sum_{k=0}^{m} \binom{m}{k} (-1)^k k! \frac{x^{m-k}}{a^k}.$$

After simplification, the distribution is

$$F_{L+Q}(t) = \sum_{d=0}^{h-1} P[\text{buffered at } d] \left\{ 1 - e^{-\mu(1-\rho)t} \sum_{k=0}^{d} \frac{(\mu(1-\rho)t)^k}{k!} - (\frac{\rho-1}{\rho})^{d+1} e^{-\mu t} \left(1 - e^{\mu\rho t} \sum_{k=0}^{d} \frac{(-\mu\rho t)^k}{k!} \right) \right\}.$$

The header-inspection delay simply shifts the distribution to the right by $h\tau$.