Peformance of Shuffle-Like Switching Networks with Deflection

Arvind Krishna Bruce Hajek
Coordinated Science Laboratory and the
Department of Electrical and Computer Engineering
University Of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

Four packet-switched networks using shuffle-exchange interconnections and deflection routing are analyzed. The first two are well-known networks based solely on shuffle interconnections, and the other two are variations in which the negative effects of deflection is reduced. Approximate state equations are given under a uniform traffic assumption. The equations predict the distribution of packet delay, and can be used in situations where packets are assigned priorities. The four networks are briefly compared to each other and to Batcher-Banyan sorting networks and hypercube deflection networks.

1 Introduction

Deflection routing, originally called *hot-potato routing* [1], is a technique for maintaining bounded buffers in a packet-switched communication network. If, due to congestion at a switch, not all packets can be sent out along shortest paths to their destinations, some packets are sent out on other links. The penalty is an increase in the distance traveled by packets, and the reward is the simplicity of switch design resulting from the absence of large buffers and routing tables. Traditional store-and-forward networks use extensive computation at the nodes to determine packet routes in order to use transmission bandwidth sparingly. In contrast, deflection leads to simple switches by making liberal use of transmission bandwidth. Since the penalty for longer routes increases as propagation delay becomes more of a factor, we consider deflection routing primarily for networks with a small physical diameter, such as those in a multiprocessor computer system or a packet switch for telecommunications.

We consider deflection routing in this paper for several well-known networks based on shuffle exchanges. In order to unify the description of the networks, in Subsection 2.1 we define the class of *shuffle-ring* networks. Informally, a shuffle-ring network with parameters n and k consists of k columns of switching nodes, with 2^n 2×2 nodes in each stage. The columns are arranged in a ring, with each column connected to the next by a perfect shuffle interconnection. We do not consider the network to be a "multi-stage" network in the strict sense of the word, because packets can be injected by, or destined to, any of the $k2^n$ nodes in the network. The important special cases of a shuffle-exchange network (k=1) and ShuffleNet (k=n) [2] are examined in Subsections 2.2 and 2.3, respectively.

This research was supported by the National Science Foundation under contract NSF ECS 83 52030

One reason for our interest in shuffle-ring networks is that they use 2×2 switching nodes, an important consideration for implementation using optical components. Another reason is that these networks have a diameter that grows typically logarithmically with the number of nodes, which is the smallest rate possible for 2×2 nodes. The networks admit simple routing rules based on packet destinations.

An apparent drawback of deflection routing in shuffle-ring networks is that a single deflection can cause the distance of a packet from its destination to increase significantly. The detrimental effect can be largely reduced by giving priority to packets closer to finishing [3]. We show for shuffle-exchange networks (k=1) that (for approximate evolution equations) giving priority to packets closest to their destinations is the optimal priority rule, in a certain sense. We also consider ShuffleNet when priority is based primarily on the number of deflections a packet has undergone, and in the case of ties also on the distance to the respective destinations. This priority rule is proposed for the network under construction at UC Boulder [4]. Our results indicate that this priority rule has the desired effect of reducing the tail of the delay distribution, at the expense of slightly increasing the mean delay.

Another approach for reducing the negative effects of deflection is to consider variations of shuffle networks for which a deflection causes the distance of a packet to its destination to increase only by a small amount. Two such variations are considered in Section 3. One of these (the *cross-back switch*) uses shuffles augmented by inverse shuffles, while the other uses a shuffle augmented by the identity interconnection. The cross-back switch is a slight generalization of the shuffle-exchange and exchange-unshuffle network of Tan et al [5].

We assume in this paper that the destinations of the packets generated at each node are distributed uniformly among the other nodes in the network. Our analysis begins with certain evolution equations, which are derived, roughly speaking, by ignoring the dependence between arrivals on different links of a switch. Similar evolution equations were given by [3,6,7]. We consistently found the solutions of the evolution equations to closely match simulations.

The bulk of the end results that we report in this paper concern the time it takes for the network to *evacuate* when operating synchronously with no packets injected other than those of the initial load. However, the evolution equation method works well in predicting delay and throughput for the networks in steady state operation as well. We illustrate this fact in our consideration of the UC Boulder ShuffleNet design, and we intend to report other delay/throughput results in the near future. We feel that the evacuation time, emphasized here, is an important performance parameter for several reasons. First, some networks support synchronous computation, and the normal operation of such networks consists of repeated evacuations. Secondly, as noted before, all

the networks we consider here are recirculating. However, the networks can be expanded in space in such a way that the evolution that we describe from-slot-to-slot in time applies as well to packets moving from-copy-to-copy in space, through copies of the original network. The evacuation time of the original network dictates how many copies the space expanded network must have. A final reason for our interest in evacuation time is that it requires *transient* analysis of the network, so that the agreement with simulation better confirms the accuracy of our method. The ability to predict transient response is important even if average delay is the main item of interest, for if the offered traffic is bursty the network may continually be in a transient mode.

A comparison of the complexity of several networks using deflection and a network based on the Batcher-Banyan sort [8] is given in Section 4. Complexity is measured in units of switches per packet per slot, and is adjusted by being multiplied by the product of the number of inputs and number of outputs of the component switches. The complexities reported are based on the analysis of Sections 2 and 3.

We close this section by briefly commenting on some other work. Maxemchuk [9,10,11] has extensively studied the performance of deflection routing in a Manhattan street network. Like the shuffle-ring networks, these networks are constructed from 2×2 switching nodes. They have the advantage that a single deflection increases the distance to the destination by only a small amount, and the disadvantage that the network diameter grows as the square root, rather than as the logarithm, of the number of nodes. Finished products using deflection routing include the Hep multiprocessor computer system [12] and the Connection Machine [13]. An interesting treatement of flow control and cut-through for deflection routing is given in the paper of Ngai and Seitz [14].

2 Shuffle Ring Networks

2.1 Network model and operation

An (n,k) shuffle ring network[1] has $k2^n$ nodes and can be conceptually visualized as having k columns of 2^n nodes each. The columns are connected in a unidirectional ring with shuffle interconnections between consecutive columns. Each node is labeled by a pair (c,r), where $0 \le c < k$ and $0 \le r < 2^n$. We often express the row identifier in base 2 notation, so $(c,r) = (c, r_{n-1}, r_{n-2}, \ldots, r_0)$ where $r_i \in \{0,1\}, 0 \le i < n$. Each node has two outgoing links and two incoming links. The two links going out of node $(c, r_{n-1}, r_{n-2}, \ldots, r_0)$ lead to nodes $(c \oplus 1, r_{n-2}, \ldots, r_0, 0)$ and $(c \oplus 1, r_{n-2}, \ldots, r_0, 1)$, where $c \oplus 1 = (c+1 \bmod k)$.

We will consider packet switching, with routes chosen as follows. Consider a packet starting at node (c,r) and destined for for node (c',r'). Let us first consider the special case when $c' = c \oplus n$. In this case, the packet wishes to pass through the sequence of n nodes (c^i, r^i), $1 \le i \le n$, where $c^i = c \oplus i$ and r^i is given by placing a window of length n over the sequence $r_{n-1}, r_{n-2}, \ldots, r_0, r'_{n-1}, \ldots, r'_1, r'_0$, with the window positioned i clicks from the left-most position. The construction guarantees that there is a link from (c^i, r^i) to (c^{i+1}, r^{i+1}) for $0 \le i < n$, where $(c,r) = (c^0, r^0)$.

In general, let j be the column number such that $c' = c \oplus n \oplus j$. For the first j hops, the packet does not care which output links it takes. After proceeding through j links, the packet wishes to follow the unique route of n more links described in the special case above. The diameter of this network is $n + k - 1$.

We have described the route a packet nominally takes. However, due to congestion, a packet may be forced to take an undesired link at some node. This packet is said to be *deflected*. After a deflection, the packet begins to follow a new desired route as if it is just starting. If the packet is at distance i, $(i \le n)$ from its destination and is deflected, the distance to destination increases by one less than a multiple of k – typically[2] the smallest multiple of k such that the new distance is at least n.

The network operates synchronously: the time axis is divided into slots corresponding to packet transmission times and each link can relay one packet per time slot. Consider a fixed node at the beginning of a time slot. Since the node has two input links, as many as two packets were received during the previous time slot. Any such packets destined for the node are removed from the network. New packets may be injected into the network at the node, bringing the total to at most two continuing packets. Under deflection routing, the continuing packets have to be assigned to the two outgoing links. Since two packets might desire the same output link, the deflection routing scheme requires a rule for resolving this conflict.

The rule for resolving routing conflicts is based on the states of the packets. The state of a packet is comprised of information that it carries in the form of control bits. As an example, the state of a packet may consist of the destination address, the source address and the number of times the packet has been deflected. The network designer has to decide what information is relevant for resolving conflicts, and include it in the state of a packet. One rule that suggests itself is based on priorities. A priority is computed for each packet based on its state. Packets are then ordered with respect to their priorities. All packets with the same priority can be ordered randomly amongst themselves, all orders being equally likely. The packets are considered one at a time, in order of decreasing priorities. When a given packet is considered, the node examines whether the packet has a preferred link. If that link is free (no packet has been assigned to that link), the packet is assigned to the link. Otherwise, the packet is assigned to a free link, all choices being equally likely.

Note that our conflict resolution rule is a *two-pass* strategy. On the first pass, the node computes the priority of each packet. The node uses the priority to order the packets and in the second pass routes the packets on the output links.

The arrivals of packets on different links of a node are not necessarily independent for the models described in this paper. Also, the choice of output links made by different packets at the same switch are not necessarily independent. However, under the uniform traffic assumption, on the average packets choose each output link of a switch equally often. We will derive an approximate performance analysis by pretending that arrivals on different links are independent, and packets choose the output links of a switch equiprobably and independently of each other. Specifically, we make the following approximations for each node and time slot:

Approximation 2.1 *A switch receives packets on an incoming link independently of whether a packet is received on other links. Also, the state of these packets is drawn independently from a single distribution.*

[1] All of the networks described in this paper can be easily generalized to have p-ary shuffle interconnections, with $p \times p$ switches at each node.

[2] It is possible to do a little better in some cases, but on the average the routes we have described are only one link longer than the shortest routes between two nodes in the network.

Approximation 2.2 *The choice of output link of a packet which is yet to reach its destination is randomly, uniformly distributed between the two output links, and this choice is made independently of other packets.*

2.2 Shuffle Exchange Network

We study deflection on the $(n,1)$ shuffle ring network in this section. Suppose each node has two packets for delivery to other nodes in the network. The destination of each packet is chosen uniformly from amongst the other $N-1$ nodes, where $N=2^n$, and each packet chooses its destination independently of other packets. We determine, under some simplifying approximations, how long the network takes to evacuate.

We use *closest-to-destination* priority routing, which implies that a packet which has fewer links to travel towards its destination gets preference in case there are conflicts. The intuition behind this rule is to try and reduce the number of packets in the network as soon as possible, hopefully decreasing the number of conflicts later.

The performance of the system described above can be determined by a Markov chain analysis. However, the number of states is N^N, making an exact analysis prohibitively expensive (in terms of computational resources). Similarly, simulation also requires large computational resources. Under approximations 2.1 and 2.2, we will derive evolution equations which determine the behaviour of a typical packet in the system. The number of states reduces to $n+1$, where $n=\log_2 N$. These evolution equations will then be used to derive bounds on the evacuation time, suitably defined, for the approximate network. These bounds are tight in the sense that the upper and lower bounds are both $O(n^2)$.

Consider a fixed packet, which has yet to reach its destination, at the beginning of a time slot. The node makes a decision about which output link the packet will traverse during this time slot. This implies that the distance to the packet's destination may decrease by one, or if the packet is deflected the distance to its destination will increase to n. No packet has a distance greater than n from its destination, because there is a route of n links from a node to any other. Thus, we view the packets as performing a random walk on the integers $0,1,2,\ldots,n$, where state i corresponds to the distance of a packet from its destination. The packets start at state n at time 0, and their destination is state 0. If a packet is not deflected in a time slot its state decreases by one, and if deflected, the packet goes to state n.

Consider a fixed link and define $p_t(i)$, $0 \le i \le n$ to be the probability that a packet i links from its destination (at the end of the slot) traverses the link during time slot t, and set $\mathbf{p_t} = (p_t(0), p_t(1), \ldots, p_t(n))$. Also, let $p_t(0)$ include the probability that there is no packet on the link, so that $\mathbf{p_t}$ is a probability distribution.Let us determine the deflection probability for a packet. Suppose a packet at distance i $(i>0)$ from its destination arrives on a link. This packet can be deflected by a packet from the other link only if the second packet is in a state $1,\ldots,i$, and the second packet prefers the same output link as the first packet. The second packet prefers the same output link with probability 1/2. If the second packet is at state i, it will win the conflict with probability 1/2, and if its state is less than i, it will always win the routing conflict. Thus, the deflection probability for a packet at state i is $\frac{1}{2}\sum_{j=1}^{j=i-1} p_t(j) + \frac{1}{4}p_t(i)$. All deflected packets go to state n. The evolution of $\mathbf{p_t}$ is given by the following *update equation*:

$$\begin{aligned} p_{t+1}(0) &= p_t(0) + p_t(1)\left(1 - \tfrac{1}{4}p_t(1)\right) \\ p_{t+1}(i) &= p_t(i+1)\left(1 - \tfrac{1}{2}\sum_{j=1}^{j=i} p_t(j) - \tfrac{1}{4}p_t(i+1)\right) \\ p_{t+1}(n) &= \tfrac{1}{4}\left(\sum_{j=1}^{j=n} p_t(j)\right)^2 \end{aligned} \tag{2.1}$$

where $1 \le i < n$. These equations are similar to the evolution equations numerically evaluated in [3]. The probability mass $p_{t+1}(n)$ is the mass that is deflected at time $t+1$. The expression for $p_{t+1}(n)$ in equation (2.1) follows from the equation below.

$$p_{t+1}(n) = \frac{1}{4}p_t^2(1) + \sum_{j=2}^{j=n} p_t(j)\left(\frac{1}{2}\sum_{k=1}^{k=j-1} p_t(k) + \frac{1}{4}p_t(j)\right)$$

We simulated a shuffle exchange graph using deflection routing to compare the behavior of the network with predictions derived from the update equations. Figures 1 and 2 show the averaged results of twenty simulation runs for a 2^9 node shuffle exchange graph, together with the corresponding predictions. The nodes of the network were all filled with two packets at the beginning of the first slot, and then no new packets were added to the network. The destination of each packet in a node was chosen equiprobably from amongst the other 2^9-1 nodes, and independently of all other packets. The close agreement between simulation and predictions apparent in the data presented in figures 1 and 2 is representative of all the data we have observed.

We define the evacuation time, T_{evac}, as the first time, t, at which the expected link utilization in the system, $\sum_{i=1}^{i=n} p_t(i)$, is less than $2^{-(n+1)}$. The intuition behind this definition is that it corresponds to the first time at which there is less than one packet in all the nodes put together. A numerical value for T_{evac} can be computed by iterating the update equation and observing the time, t, at which the required condition on $\mathbf{p_t}$ is met. On the other hand, obtaining an expression for T_{evac} in terms of n requires an explicit evaluation of $p_t(0)$ in terms of $\mathbf{p_0}$.

We introduce a partial order $\prec$ on the link occupation probabilities $\mathbf{p_t}$. This partial order serves two main functions in this paper. Firstly, it is used for a comparison of priority rules, which is summarized in Theorem 2.1. Secondly, it is used to provide bounds on the evacuation time, and these bounds are summarized in Theorem 2.2. The partial order is defined by $\mathbf{p_t} \prec \mathbf{q_t}$ if and only if

$$\sum_{j=i}^{j=n} p_t(j) \le \sum_{j=i}^{j=n} q_t(j),\ 1 \le i \le n. \tag{2.2}$$

Roughly speaking, $\mathbf{p} \prec \mathbf{q}$ corresponds to there being more packets further away from their destinations under distribution $\mathbf{q}$ than under distribution $\mathbf{p}$. This order relation, $\prec$, has the usual properties of a partial order, i.e. it is a transitive and reflexive relation. The next lemma states that the relation $\prec$ is preserved under equation (2.1).

Lemma[3] 2.1 *If* $\mathbf{p_{t_0}} \prec \mathbf{q_{t_0}}$ *and* $\mathbf{p}$ *and* $\mathbf{q}$ *evolve according to equation (2.1), then* $\mathbf{p_t} \prec \mathbf{q_t}$, $t \ge t_0$.

[3]Due to space limitations, the proofs of all Lemmas and Theorems is omitted

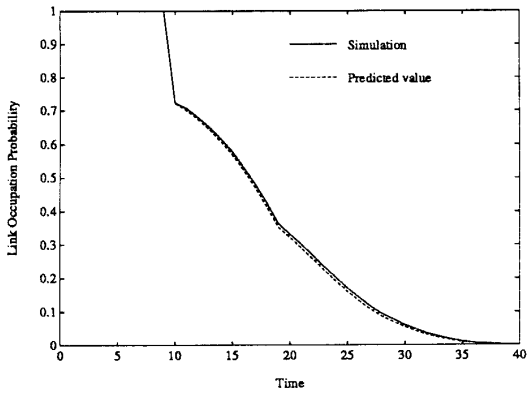

Figure 1: Link Occupation Probabilities in a 2^9 node shuffle exchange graph. The link occupation probability denotes the fraction of the 1024 links used in each time slot. Predicted values are obtained from the update equation.

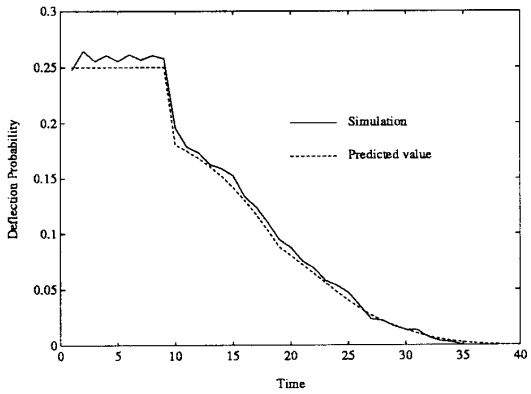

Figure 2: Fraction of packets deflected in a 2^9 node shuffle exchange graph. The deflection probability denotes the fraction of the packets deflected in each time slot. Predicted values are obtained from the update equation.

An important issue that we now address is whether the *closest-to-destination* priority rule is optimal for the evacuation time. Consider a priority rule, δ, which is used to resolve routing conflicts. Suppose two continuing packets are at a node at the beginning of a time slot. One packet is in state i, the other is in state j, and both packets desire the same output link. Then, let δ_{ij} be the probability that the packet in state i wins the routing conflict. It follows that $\delta_{ji} = 1 - \delta_{ij}$. Set $\mathbf{p}_t^\delta$ to be the probability distribution of packet states under routing rule δ. In general, δ may also be a function of the time slot t. The evolution equations are

$$\begin{aligned} p^\delta_{t+1}(0) &= p^\delta_t(0) + p^\delta_t(1)\left(1 - \tfrac{1}{2}\textstyle\sum_{j=1}^{j=n} p^\delta_t(j)\delta_{j1}\right) \\ p^\delta_{t+1}(i-1) &= p^\delta_t(i)\left(1 - \tfrac{1}{2}\textstyle\sum_{j=1}^{j=n} p^\delta_t(j)\delta_{ji}\right) \\ p^\delta_{t+1}(n) &= \tfrac{1}{4}\left(\textstyle\sum_{j=1}^{j=n} p^\delta_t(i)\right)^2 \end{aligned} \tag{2.3}$$

where $1 < i \leq n$. The next lemma states that the closest-to-destination priority rule is the best possible rule amongst the class of rules described above.

Lemma 2.2 *Let* $\mathbf{p_0} = \mathbf{p_0^\delta}$, *where* $\mathbf{p_0}$ *is a probability distribution on* $0, 1, \ldots, n$. *If* $\mathbf{p_0}$ *evolves according to equation (2.1) and* $\mathbf{p_0^\delta}$ *evolves according to equation (2.3), then* $\mathbf{p_1} \prec \mathbf{p_1^\delta}$.

Let δ be any local rule which is used to resolve routing conflicts. We define a symmetric rule as any local rule which ignores the link on which packets enter the node and also that packets desire each output link equally often under this rule. These are the intuitive conditions that correspond to approximations 2.1 & 2.2. We define the evacuation time for strategy δ, T^δ_{evac}, in a similar manner to T_{evac}, as the first time t at which the link utilization probability is less than $2^{-(n+1)}$. The following theorem summarizes the reason for choosing closest-to-destination priority routing. The proof of this theorem is an easy consequence of Lemmas 2.1 and 2.2.

Theorem[4] **2.1** *Let* T_{evac} *be the evacuation time of the network under closest-to-destination priority routing, and let* T^δ_{evac} *be the evacuation time for any other symmetric rule,* δ, *used to resolve routing conflicts. Then, under Approximations 2.1 & 2.2,* $T^\delta_{evac} \geq T_{evac}$.

We have found explicit bounds on the evacuation time, T_{evac}, which are summarized in the following theorem.

Theorem 2.2 *Let* $\mathbf{p_0}$ *be given by* $p_0(n) = 1$; $p_0(i) = 0$, $i < n$, *and* $\mathbf{p_t}$ *evolve according to equation (2.1). Let* T_{evac} *be the evacuation time for this system. Then,*

$$\frac{n^2}{16} \leq T_{evac} \leq \frac{4n^2}{9} + n + n\log_2(n)$$

Equation (2.1) is hard to solve explicitly in a closed form. However, observe that this equation has the property that if $p_t(j) = 0$, $1 \leq j \leq k$, then for $0 \leq i \leq (k-1)$,

$$p_{t+i+1}(k-i) = p_{t+i}(k-i+1)\left(1 - \frac{1}{4}p_{t+i}(k-i+1)\right). \tag{2.4}$$

In order to take advantage of this simplification, consider a system in which the mass from states $1, 2, \ldots, n$ is swept into state n every n time slots. Suppose a sweep is done at time t, setting $p_t(i) = 0$, $1 \leq i \leq n-1$. Then, this system will follow equation (2.4) with $k = n-1$. Also, the link occupation probabilities of this system dominate those of the original system, in the sense of $\prec$. The evacuation time of this system provides the upper bound in Theorem 2.2. In a similar fashion, we construct another simple system to provide the lower bound in Theorem 2.2.

[4]Lemmas 2.1, 2.2, and Theorem 2.1 do not extend to (n,k) shuffle ring networks with $k \geq 2$

2.3 (n,n) Shuffle Ring Networks

We now consider a (n, n) shuffle ring network, as defined in Subsection 2.1. This network has been proposed in [2] as the ShuffleNet and is being implemented as an optical interconnection network at the University of Colorado [4]. The state of a packet includes:

Age – the number of times a packet has been deflected so far.

Distance – the number of links a packet has to traverse before reaching its destination.

We include the *age* in the state to try and ensure that packets do not remain in the network for an extremely long time. If a packet has been deflected a number of times, its age is large. Hence, if we give priority to older packets, the hope is that they will not be deflected and will soon leave the network. The state is written as a two-tuple (age, distance). Packets which do not have a preferred output link (distance $> n$) get lower priorities than the remaining packets. Priority of packets which have a preferred output link is based on a lexicographic ordering of (Age, Distance) as follows:

- Older packets take precedence over younger packets (lower age packets).
- Within the same age, packets with lower distance take precedence over packets with higher distance.

In order to use only a few bits for the *age* information we set a maximum age, m. The minimum age is one. If a packet with age m gets deflected its age remains at m. Otherwise, if a packet gets deflected, its age increases by one. Every packet in the system makes a state transition in every time slot. Possible state transitions for a packet with state (i, j) are:

$j > n \Longrightarrow$ the only transition is to $(i, j-1)$.

$j \leq n$ and $i < m \Longrightarrow$ the transition could be to $(i, j-1)$ or $(i+1, j+n-1)$.

$j \leq n$ and $i = m \Longrightarrow$ the transition could be to $(m, j-1)$ or $(m, j+n-1)$.

Let $p_t(i, j)$ be the probability that a packet j hops from its destination (at the end of the slot), with age i, traverses a given link during time slot t. Set $\mathbf{p_t}$ to be the collection $\{p_t(i, j),\ 1 \leq i \leq m,\ 0 \leq j < 2n\}$. The transitions from distance j to distance $(j+n-1)$ correspond to deflections. A packet with higher priority chooses the same output link as a given packet with probability $\frac{1}{2}$, by approximation 2.2. Then, using approximation 2.1, it is clear that the deflection probability, $d_t(k, l)$, for a packet in state (k, l) at the end of time slot t is nonzero only when $l \leq n$, and is given by

$$d_t(k,l) = \frac{1}{4}p_t(k,l) + \frac{1}{2}\Big(\sum_{j=1}^{j=l-1} p_t(k,j) + \sum_{i=k+1}^{i=m}\sum_{j=1}^{j=n} p_t(i,j)\Big) \tag{2.5}$$

Let $\nu_t(i, j)$ represent the external (from the node, not an input link) arrivals to state (i, j) during time slot t. $I_{[.]}$ is the usual indicator function. Now, given $\mathbf{p_t}$ and $\nu_t(.,.)$, we can compute $\mathbf{p_{t+1}}$ by the following algorithm.

Procedure Update

1. $\hat{p}_t(i,j) = p_t(i,j) + \nu_t(i,j)$
2. Compute $d_t(i,j)$ according to equation (2.5), using $\hat{p}_t(i,j)$ instead of $p_t(i,j)$.
3. For $0 < j < n$, $p_{t+1}(i,j) = \hat{p}_t(i,j+1)(1 - d_t(i,j+1))$.
4. $p_{t+1}(i,j+n-1) = \hat{p}_t(i,j+n) + I_{[i>1]}\hat{p}_t(i-1,j)d_t(i-1,j) + I_{[i=m]}\hat{p}_t(i,j)d_t(i,j)$, $1 \leq j \leq n$.
5. $p_{t+1}(i,0) = \hat{p}_t(i,0) + \hat{p}_t(i,1)(1 - d_t(i,1))$.

Suppose the desired throughput of the network is τ, and traffic is balanced (all destinations are equally likely for newly generated packets). One way of achieving this is to set $\nu_t(1,j) = \tau/n$, $n \leq j < 2n$, and all other $\nu_t(i,j) = 0$, for $t > 0$. Then, a fixed point, $\mathbf{p^*}$, can be computed for procedure update, with $\mathbf{p_{t+1}} = \mathbf{p_t} = \mathbf{p^*}$. Once $\mathbf{p^*}$ has been determined, all the state transition probabilities are known and the delay distribution can be found for the approximated network. This can be of great help to the network designer to answer questions such as

- How many ages are desirable?
- What is the average delay versus age curve for a fixed throughput?
- What is the delay distribution?

We show some typical results in figures 3 and 4, computed using procedure update. All the results are for a (6,6) shuffle ring network, which has 384 nodes. The *number of ages* represents the maximum number of ages allowed, m. Figure 3 represents an average load case, where each link is utilized approximately in 1 of every 3 time slots. Figure 4 is for a heavily loaded case, where links are utilized in almost every time slot. Similar results can be computed for different sizes of networks. We verified some of these results against simulations, and found the approximations to be accurate within 1%.

These results indicate that more ages result in a higher average delay, but reduce the spread of the delay distribution (since the ninety-ninth percentile is consistently lower for $m = 4$, compared to $m = 1$).

3 Variations on Shuffle Networks

3.1 Crossback Switching Network

The network we describe here is a variant of the (n, k) shuffle ring network obtained by adding a link in the opposite direction alongside each of the original links. Thus, every node has four outgoing and four incoming links. Recall the notation of Subsection 2.1. The four links going out of node $(c, r_{n-1}, r_{n-2}, \ldots, r_0)$ lead to nodes $(c \oplus 1, r_{n-2}, \ldots, r_0, 0)$, $(c \oplus 1, r_{n-2}, \ldots, r_0, 1)$, $(c \ominus 1, 0, r_{n-1}, \ldots, r_2, r_1)$, and $(c \ominus 1, 1, r_{n-1}, \ldots, r_2, r_1)$, where $c \ominus 1 = (c - 1 \bmod k)$.

We will consider packet switching, with routes chosen as follows. We classify packets as either *left-packets* or *right-packets*. A right-packet desires exactly the same route as it would in a shuffle ring network. The construction of desired routes for left-packets is similar to the construction of routes in Subsection 2.1, except that these routes use only edges that go from columns c to $c \ominus 1$. We describe the route of left-packets in detail. Consider a left-packet starting at node (c, r)

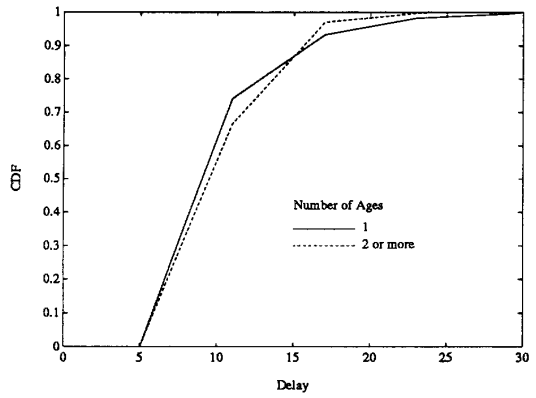

Figure 3: Distribution of delay in a (6,6) shuffle ring network for a throughput, $\tau = 0.03$. This corresponds to 23.04 packets for the network per time slot.

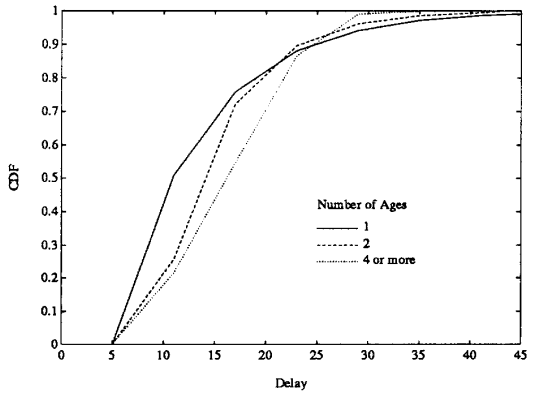

Figure 4: Distribution of delay in a (6,6) shuffle ring network for a throughput, $\tau = 0.05949$. This corresponds to 45.69 packets for the network per time slot.

and destined for node (c', r'). Let us first consider the special case when $c' = c \ominus n$. In this case, the packet wishes to pass through the sequence of n nodes (c^i, r^i), $1 \leq i \leq n$, where $c^i = c \ominus i$ and r^i is given by placing a window of length n over the sequence $r'_{n-1}, r'_{n-2}, \ldots, r'_0, r_{n-1}, \ldots, r_1, r_0$, with the window positioned i clicks from the right-most position. The construction guarantees that there is a link from (c^i, r^i) to (c^{i+1}, r^{i+1}) for $0 \leq i < n$, where $(c, r) = (c^0, r^0)$.

In general, for left-packets, let j be the column number such that $c' = c \ominus n \ominus j$. For the first j hops, the packet does not care which of the two output links it takes in the direction of decreasing column number. After these j links are taken, the packet wishes to follow the unique route of n more links described in the special case above. The longest route of this network is $n + k - 1$.

We have described the route a packet nominally takes. However, due to congestion, a packet may be forced to take an undesired link at some node. This packet is said to be *deflected*. As an example, consider a packet at node (c, r) which wishes to visit node node $(c \oplus 1, r^1)$ next. If the packet is deflected to node $(c \ominus 1, r^2)$, in the next time slot, this packet will attempt to take any link to column c. If the packet at node (c, r) is deflected to node $(c \oplus 1, r^3)$, in the next time slot, this packet will attempt to take any link to column c. In both the above cases, there is a link to node $(c \oplus 1, r^1)$ from the node reached in column c. In general, if a packet is at distance i from its destination and gets deflected, the distance to its destination increases by one to $i + 1$.

The network operation is largely the same as described in Subsection 2.1. Consider a fixed node at the beginning of a time slot. Since the node has four input links, as many as four packets were received during the previous time slot. Any such packets destined for the node are removed from the network. New packets may be injected into the network at the node, bringing the total to at most four continuing packets. A packet, upon entering the network, is classified as either a left-packet or a right-packet (equiprobably and independently of other packets).

Deflection routing requires a rule for resolving conflicts. We propose a very simple rule for resolving these, based on a *two-pass* strategy by the node at the beginning of each time slot. On the first pass, the node sequentially considers the continuing packets in random order. When a given packet is considered, the node determines if there are any links that are both unassigned and desired by the packet. If so, the packet is assigned to one such link. Otherwise, the packet is not allocated to any link. On the second pass, the controller assigns the remaining packets to the unoccupied links, all choices being equally likely. All of the packets allocated links in the second pass have been deflected.

An analysis of this network is made in the same spirit as the analysis of Section 2.2. and under very similar approximations. We compute an upper bound an upper bound on the deflection probability and this is used to derive an upper bound on the evacuation time. These results are summarized in the following theorem.

Theorem 3.1 *Consider an (n, k) crossback network operating as described above. The evacuation time, T_{evac}, is defined as the time at which all, except a fraction 2^{-n}, of the packets have reached their destinations. Suppose the network began with 4 packets at each node, with the destinations of each packet chosen independently and at random from amongst the other $2^n - 1$ nodes. Then, under the approximations described above, $T_{evac} \leq 6(n + k)$.*

This is in contrast to the shuffle network, where T_{evac} was proportional to n^2. However, the price for this reduction in evacuation time is paid for by increased switch complexity. We will further compare these networks in Section 4 and Table 2.

The shuffle-exchange and exchange-unshuffle networks considered by Tan et al [5] are precisely the crossback networks with $k = 1$. They showed how to route packets along shortest length paths. Such paths are, on average, two to four hops shorter than the nominal paths we use, though the savings come at the expense of increased complexity needed for string matching. Their strategy applies to crossback switches in general.

3.2 Stay-or-Shuffle network

The network we describe here is a variation on the shuffle exchange network ($(n, 1)$ shuffle ring). The stay-or-shuffle network is constructed from the $(n, 1)$ shuffle ring by adding links from every node to itself. Every node has three outgo-

ing and three incoming edges. Since there is only one column, we drop the column index from the node labels. Two of the outgoing links from node $(r_{n-1},\ldots,r_1,r_0)$ lead to nodes $(r_{n-2},\ldots,r_1,r_0,0)$, and $(r_{n-2},\ldots,r_1,r_0,1)$. We call these links the shuffle links. The third link out of $(r_{n-1},\ldots,r_1,r_0)$ leads to node $(r_{n-1},\ldots,r_1,r_0)$, and all such links are referred to as non-shuffle links. The non-shuffle links are equivalent to having one buffer in each node.

Routes are chosen as described in Subsection 2.1 for shuffle ring networks. The difference between these networks lies in what happens to a packet upon being deflected. Deflection occurs if a packet is routed on a link distinct from its preference. A packet which is routed onto a non-shuffle link does not change its distance to destination during the time slot. As in the case of the shuffle exchange graph, a continuing packet always prefers a particular outgoing edge. Let the distance to destination of a packet be k $(k \leq n)$. Then, if a packet is routed according to its preference, this distance decreases to $k-1$. If a packet is deflected onto a non-shuffle link, the packet remains at distance k from its destination. If deflected onto a shuffle link, the distance of the packet from its destination increases to n. Hence, if we decide that the state space of the packets is the distance to destination, then as before, the packets are performing a random walk on the integers $0,1,\ldots,n$.

A rule is required to resolve conflicts. These conflicts arise when two or three packets at a node prefer the same output link. We use a priority rule to break conflicts. The packets are routed in order of their priority, which is the distance to destination. Packets closer to their destinations get higher priorities, and ties are broken randomly. When it is a packet's turn to be routed, the node examines the preferred link. If that link is free, the packet is assigned that link. Otherwise, the node examines the non-shuffle link from the node to itself. If this link is free, the packet is assigned that link. If neither of these two links is free, the packet is assigned the third link. This rule is a *two-pass* strategy by the node.

An analysis of this network is made in the same spirit as before. We derive approximate evolution equations based on approximations similar to those in Section 2. Set T_{evac} to be the first time that the link occupation probability falls below 2^{-n}. Then, the evolution equations above can be used to evaluate T_{evac}. Some typical values are shown in Table 1. Some of these values were compared to evacuation times for a simulation of the actual network, and they were found to be accurate within 5%.

n	6	7	8	9	10	12	15	20
T_{evac}	15	18	21	24	28	35	47	67

Table 1: Evacuation time for stay-or-shuffle networks: the first row represents n, where the network has 2^n nodes, and the second row contains the evacuation time.

4 Comparison

In this section we compare the complexity of $N \times N$ switching networks, built using the various networks discussed earlier. There are two approaches to building these switching networks from a given network.

1. Let the switching network be the given network. Then, if the network is filled up with packets, it delivers them to their destinations in T_{evac} time units.

2. As described in the introduction, the networks can be expanded in space in such a way that the evolution that we described from-slot-to-slot in time applies as well to packets moving from-copy-to-copy in space, through copies of the original network. The evacuation time of the original network dictates how many copies the space expanded network must have. This approach gives a pipelined version of the original network. The cost of this network is T_{evac} times the original cost, but the throughput is T_{evac} times as much as the original throughput.

A comparison of various networks is shown in Table 2. An example of a Batcher-Banyan[5] network is Starlite [8]. The second column contains estimates on T_{evac} for the various networks. These estimates are first order estimates that fall between the bounds we have proven and agree well with numerical computations for n upto 100. The third column gives the individual node complexity, where a node with n_1 input edges and n_2 output links is considered to have $n_1 n_2$ complexity. Also, if a node in the graph has k input(output) links, it actually has $k+1$ input(output) links, since one link is needed for the external host to inject (remove) traffic. The fourth column represents the number of packets each node injects into the network. The fifth column shows a measure of the overall complexity, which is given by the product of T_{evac} and node complexity divided by the packets per node per time slot. The comment on *multiple packets* refers to whether multiple packets for the same destination can be delivered in T_{evac} time steps in the network.

5 Conclusion

The similarity of the complexities of the various networks listed in Table 2 is striking. Both the crossback switch and the stay-or-shuffle switches have relatively small values of evacuation time, as we had hoped. However, it appears that the improved performance there may barely, if at all, compensate for the increased complexity of the component switches. This suggests that the search for improved networks should be pursued within the family of binary switching networks. In particular, a network having the approximate evacuation time of the cross-back switch, but made from 2×2 switches, would be quite attractive. The network should admit simple deflection routing, or some other simple form of dynamic routing based on local information.

Our analysis has been based on the approximate evolution equations and simulation. Even in situations with priority classes, the evolution equations matched simulations quite well. The problem of analytically validating the evolution equations without any independence approximations has to date appeared intractable. Nevertheless, we have found the approximate evolution equations to be useful tools for quickly exploring a large class of networks, and networks of fairly large size (over 10^6 nodes) can be readily handled.

Acknowledgements

The authors wish to thank Albert Greenberg and Jon Sauer

[5]The $\frac{1}{2}$ in the column for node complexity refers to the fact that only $\frac{N}{2}$ switches are needed in each stage of Batcher's sorting network. The T_{evac} value refers to the number of stages needed for Batcher's sorting network.

Network type	T_{evac}	Node complexity	Packets per node per time slot	Overall complexity	Comments
Shuffle network	$\frac{1}{4}n^2$	$9 = 3 \times 3$	2	$\frac{9}{8}n^2$	Multiple packets Average performance
Crossback network	$4n$	$25 = 5 \times 5$	4	$25n$	Same as above
Stay-or-shuffle network	$\frac{1}{10}n^2$	$16 = 4 \times 4$	2	$\frac{4}{5}n^2$	Same as above
Batcher-Banyan Network	$\frac{1}{2}n^2$	$2 = 2 \times 2 \times \frac{1}{2}$	1	n^2	No multiple packets All permutations are allowed
Hypercube network	n	$n^2 = n \times n$	n	n^2	Multiple Packets Average performance

Table 2: Comparison of the complexity of $N \times N$ switching networks, $n = \log_2 N$

for many useful discussions.

References

[1] P. Baran, "On distributed communication networks," *IEEE Trans. Communication Systems*, vol. 12, pp. 1–9, 1964.

[2] A. S. Acampora, M. J. Karol, and M. G. Hluchyj, "Terabit lightwave networks: the multihop approach," *AT&T Technical Journal*, pp. 21–34, 1987.

[3] D. H. Lawrie and D. A. Padua, "Analysis of message switching with shuffle-exchanges in multiprocessors," in *The Proceedings of the Workshop on Interconnection Networks for Parallel and Distributed Processing*, pp. 116–123, 1980. reprinted in IEEE Press book, Interconnection Networks, Wu and Feng Eds., 1984, IEEE Computer Society Press.

[4] J. R. Sauer, "An optoelectronic multi-Gb/s packet switching network," February 1989. Preprint, Optoelectronic Systems Center, Univ. of Colorado.

[5] X. N. Tan, K. C. Sevcik, and J. W. Hong, "Optimal routing in the shuffle-exchange networks for multiprocessor systems," in *CompEuro 88 – System Design: Concepts, Methods and Tools*, pp. 255–264, IEEE, Euromicro, April 1988. published by IEEE Comput. Soc. Press, Washington, D.C.

[6] A. G. Greenberg and J. Goodman, "Sharp approximate models of adaptive routing in mesh networks," in *Teletraffic Analysis and Computer Performance Evaluation*, (O. Boxma, J. W. Cohen, and H. C. Tijms, eds.), pp. 255–270, Elsevier, Amsterdam, 1986. revised, 1988.

[7] A. Greenberg and B. Hajek, "Approximate analysis of deflection routing in hypercube networks," August 1989. Submitted to IEEE Trans. on Communications. Also, presented at the TIMS meeting, Osaka, July 1989.

[8] A. Huang and S. Knauer, "Starlite: a wideband digital switch," in *Proceedings of Globecom Conference*, pp. 121–125, IEEE Press, 1984.

[9] N. F. Maxemchuk, "Regular mesh topologies in local and metropolitan area networks," *AT&T Technical Journal*, vol. 65, pp. 1659–1685, September 1985.

[10] N. F. Maxemchuk, "Routing in the manhattan street network," *IEEE Transactions on Communications*, vol. COM-35, pp. 503–512, May 1987.

[11] N. F. Maxemchuk, "Comparison of deflection and store-and-forward techniques in the manhattan street and shuffle-exchange networks," in *Proceedings of IEEE Infocom'89*, pp. 800–809, 1989.

[12] B. Smith, "Architecture and applications of the HEP multiprocessor computer system," in *Real-time signal processing IV-Proc SPIE 298*, (T. F. Tao, ed.), pp. 241–248, Society Photo-Optical Eng, 1981.

[13] W. D. Hillis, *The Connection Machine*. Cambridge, Mass.: MIT Press, 1985.

[14] J. Y. Ngai and C. L. Seitz, "A framework for adaptive routing in multicomputer networks," in *Proc. ACM Symp. Parallel Alg. and Architech.*, June 1989.