On the Stability of Shuffle-Exchange and Bidirectional Shuffle-Exchange Deflection Networks

Soung C. Liew

***Abstract*— In a stable packet-switched network, throughput equals offered load and packet backlogs do not build up in an unbounded manner. A network with an unstable operating region poses the problem that it may evolve eventually to a stable but saturated operating point with a low throughput. This paper considers the shuffle-exchange and bidirectional shuffle networks when operated with deflection routing. It is shown that both networks exhibit instability when packet contention is resolved in a random manner. However, instability can be avoided if contention is resolved in a manner that favors packets closest to their destinations. This obviates the need for complicated network access control to prevent instability.**

***Index Terms*—Deflection routing, hot-potato routing, network congestion, network stability, packet switching, shuffle-exchange network.**

I. Introduction

IN INTERCONNECTION networks [1], contention arises when two or more packets arrive at a common node and want to go out on the same output. There are two ways to deal with contention. One of the packets could be declared the winner and routed to the output while the rest are buffered in the node so that they can try to access the output after the transmission of the first packet. The other method, often referred to as *deflection routing* [2]–[14], is to route the losing packets to the "wrong" output. Attempts will be made later to route these packets to their destinations via a possibly different path. Apart from its simplicity, deflection routing has the advantage that packets are automatically diffused away from a congested link.

It is common to characterize the performance of a network by a delay-offered load curve. The delay-offered load curve, however, does not generally reveal potential network instability. The same offered load could correspond to two different operating points, one of which is unstable and which may evolve quickly to a stable but saturated operating point, where the throughput cannot sustain the offered load (i.e., input rate higher than output rate). This leads to a breakdown in the performance. Networks with unstable operating regions are undesirable because access control needs to be exercised to control the overall load in order to prevent network saturation.

In this paper, we show that shuffle-exchange and bidirectional shuffle networks experience throughput instability when packet contention is resolved in a random manner. However, the networks become stable if we favor packets closer to their destinations when resolving contention. Before deriving and discussing these results, the next section first describes the network model and assumptions adopted.

Manuscript received May 17, 1995; revised May 3, 1996; approved by IEEE/ACM Transactions on Networking Editor R. Ramaswami. This work is supported by a Chinese University of Hong Kong Direct Grant and a UGC Earmarked Grant: CUHK 276/94E.

The author is with the Department of Information Engineering, The Chinese University of Hong Kong.

Publisher Item Identifier S 1063-6692(97)01707-X.

II. Network Model and Assumptions

Interconnection networks are commonly classified as *direct* (or *static* [1]) networks or indirect (or *dynamic* [1]) networks. In the indirect networks, which are basically switches, packets enter the network via a set of input links and exit the network via a set of output links. Examples of indirect networks and results relevant to them can be found in [1], [5], [6], [8]. In a direct networks, the sources and destinations are nodes rather than links. This papers concerns direct networks.

A. Network Connectivity and Routing

The basic structure of the shuffle-exchange network that this paper studies is a single-stage network with the output links connected back to the input links. Fig. 1 shows an example with four 2×2 nodes. Each 2×2 node in an N-node network can be labeled by a distinct binary number $x_n \cdots x_1$, where $x_i = 0$ or 1 and $n = \log_2 N$. The upper output of the node is labeled 0 and the lower output is labeled 1. Node $x_n \cdots x_1$ is connected to node $x_{n-1} \cdots x_1 0$ via its upper output and to node $x_{n-1} \cdots x_1 1$ via its lower output. This means that the next node that will be visited by a packet at the current node $x_n \cdots x_1$ is found by removing bit x_n from the label and stuffing on the right the single-bit label of the output on which the packet exits the current node.

Consider the routing of a packet from source $s_n \cdots s_1$ to destination $d_n \cdots d_1$. The bits of the destination address are put into a header of the packet and used successively for routing. At the source node, d_n is used. If $d_n = 0$, the packet is routed to output 0 and it reaches node $s_{n-1} \cdots s_1 0$. If $d_n = 1$, the packet is routed to output 1 and it reaches node $s_{n-1} \cdots s_1 1$. It can be seen that without contention from other packets, the sequence of $n = \log_2 N$ nodes traversed by the packet is

$$s_n \cdots s_1 \to s_{n-1} \cdots s_1 \, d_n \to s_{n-1} \cdots s_1 \, d_n \, d_{n-1} \to \cdots \to d_n \cdots d_1.$$

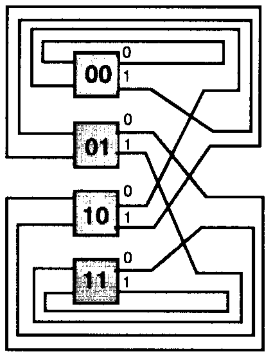

Fig. 1. A four-node shuffle network.

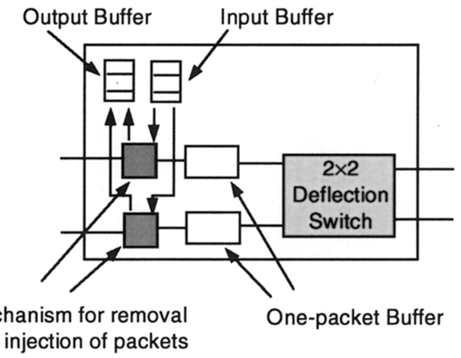

Fig. 2. Internal structure of a node in the shuffle network.

This paper assumes fixed-length packets and time-slotted synchronous network operation. Each time slot corresponds to the time needed to transmit a packet from a node to another node. Thus, a packet needs at least n time slots (steps) before reaching its destination. If it is deflected, more steps are needed.

The above routing scheme, which is adopted in this paper, assumes a deflected or new packet is n steps away from its destination. This is not optimal: the shortest path between two nodes is not always n steps in a shuffle-exchange network. However, optimal routing of packets along their shortest paths are more complicated to implement—either a lookup table is needed at each node or some real-time computation needs to be done. The analysis of shortest path routing is also more difficult. Whether instability exists when shortest-path routing is used is an interesting subject for further study.

B. Network Access

Once a packet enters the network, it is a matter of time before it reaches its destination. There will be no packet loss inside the network. However, packets may be dropped outside the network if the offered load exceeds the maximum throughput.

With reference to Fig. 2, which shows the internal structure of a node, there is a mechanism at each input for the removal of packets destined for the node and the injection of packets originating from the node. A packet from the "outside" enters the network via an input queue. It must wait for an empty slot on the input links (i.e., when one of the inputs does not carry an active arriving packet from another node) before it can enter the network. If packets enter the input queue faster than the rate at which empty slots arrive (which could happen when the offered load is higher than the network throughput), the input queue fills up and packets are dropped when the buffer overflows.

Various access policies for the network are possible. The simplest is the *greedy policy,* in which the packet at the head of the input queue immediately enters the network upon seeing an empty slot on one of the inputs. The analytical throughput results of this paper are independent of access schemes. They point to the possibility of instability when contention in the network is resolved randomly and the greedy network-access scheme is used.

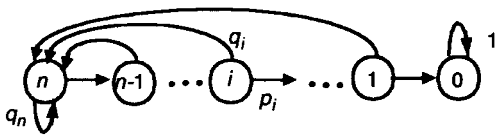

Fig. 3. The state-transition diagram of a packet in the shuffle-exchange network.

C. Analysis Assumptions

This paper, like many others [5]–[8], adopts the balanced-traffic and independence assumption that an input packet to a node that is not its final destination is equally likely to desire any of the node outputs, independently of the situation at the other input. This is a good approximation when the network is large and when the packets originating from each node in the network is equally likely to be destined for any other nodes in the network.

We further assume that links in the network have the same loading. Again, this is a good approximation when the traffic is uniform. Whether instability exists when the traffic in nonuniform and the links are unevenly loaded is a topic for further study.

For random contention resolution, packets desiring the same output at a node are equally likely to be chosen as the winner. For the shortest-distance priority scheme, the packet closest to its destination will be the winner; in case of a tie, contention among winners is resolved in a random manner.

Before leaving this section, we note that the delay performance of the basic shuffle-exchange network has been studied in [7]. In fact, contention resolution that favors packet closest to its destination was originally suggested there. The focus in [7] was on the mean delay as a function of link loading. Network instability is, however, not revealed readily from the delay-link load curve. Our work focuses on the throughput instead. Apart from the basic shuffle-exchange network, this paper also studies the performance of the bidirectional shuffle network and shows that despite the improved throughput, instability still occurs so long as random contention resolution is used.

III. Shuffle-Exchange Network

Fig. 3 shows the state-transition diagram of a packet in which the state is the *distance* (or number of undeflected steps) to destination. The distance decreases each time the packet is

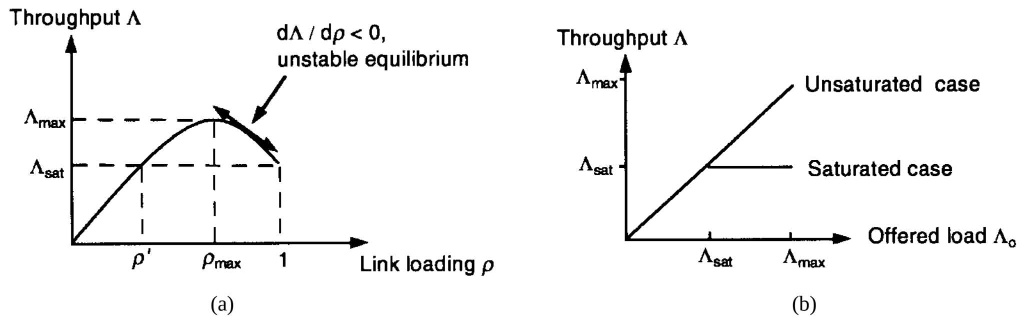

Fig. 4. (a) Throughput as a function of link loading and (b) throughput as a function of offered load; for shuffle network with $n \geq 5$.

successfully routed at a node and the distance is set back to n each time the packet is deflected: routing starts anew after each deflection. The absorbing state, state 0, in the state-transition diagram, corresponds to the packet reaching its destination.

A. Random Contention Resolution

For random contention resolution, the deflection probability q_i is independent of the state and can be written as q. Similarly, the success probability $p_i = 1 - q_i$ can be written as p. Let $T(i)$ denote the expected additional number of steps (deflected or undeflected) needed by a packet in state i to reach its destination node. We have

$$\begin{aligned} T(i) &= 1 + pT(i-1) + qT(n), \qquad 1 \leq i \leq n \\ T(0) &= 0. \end{aligned} \tag{1}$$

Equation (1) is a linear difference equation in i. The homogeneous solution to $T(i) - pT(i-1) = 0$ is $T(i) = p^i$ and a particular solution to $T(i) - pT(i-1) = 1 + qT(n)$ is $T(i) = [1 + qT(n)]/(1-p) = [1 + qT(n)]/q$. Therefore, the general solution is given by

$$T(i) = cp^i + \frac{1 + qT(n)}{q} \tag{2}$$

where c is a constant to be found by matching the boundary condition. The boundary condition $T(0) = 0$ yields $c = -[1 + qT(n)]/q$. Substituting this into (2) and setting $i = n$, we get

$$T(n) = \frac{1 - p^n}{Pp^n q}. \tag{3}$$

Let ρ be the probability of finding a packet at an input link at the beginning of a time slot; this is the link loading. Then, with probability $\rho/2$ an input packet will encounter a packet at the other input of the same node that desires the same output. Thus, with the random contention-resolution scheme, the packet will be deflected with probability $q = \rho/4$. The total throughput of the network Λ is given by Little's Law (i.e., the expected number of packets in the whole network divided by the expected packet delay)

$$\begin{aligned} \Lambda &= \frac{2N\rho}{T(n)} \\ &= \frac{8Nq^2p^n}{1 - p^n} \\ &= \frac{8N\left(\frac{\rho}{4}\right)^2\left(1 - \frac{\rho}{4}\right)^n}{1 - \left(1 - \frac{\rho}{4}\right)^n}. \end{aligned} \tag{4}$$

It can be shown from the above that for $n \leq 4$, $d\Lambda/d\rho \geq 0$ for $0 \leq \rho \leq 1$ and Λ reaches its maximum at $\rho = 1$. This can be achieved using the greedy access policy when the input queues at the nodes are saturated. Note that we assume that a node can inject/remove a number of packets up to the number of input links to it. In this case, the maximum throughput of the system $\Lambda_{\max}$ is equal to its saturation throughput Λ_{sat}. As long as the offered load is lower than $\Lambda_{\max}$, the input queues will not saturate.

Network Congestion and Instability: Things become more complicated when $n \geq 5$ because then the maximum throughput is not obtained when the input queues are saturated. For each $n \geq 5$, we can find a link loading $\rho^* < 1$ such that $d\Lambda/d\rho < 0$ for all $\rho > \rho^*$. This implies $\Lambda(\rho = \rho^*) > \Lambda(\rho = 1) = \Lambda_{\text{sat}}$. Note that when the traffic pattern is nonuniform, the boundary case may well be a value of n other than five.

Physically, what happens is that there are two factors that influence Λ in opposing directions. On one hand, a higher value of ρ indicates more packets are being routed, and therefore throughput should be higher. On the other hand, when there are more packets in the system, packet deflections due to contention are more likely. When ρ is large enough it may take so many steps for packets to reach their destinations that the throughput actually decreases. The reason why this does not happen for smaller n is that the distance to destination is bounded by n, and the penalty associated with deflections is not so high when n is small.

The fact that $\Lambda_{\max} > \Lambda_{\text{sat}}$ also reveals that the system is unstable. Let Λ_o denote the total offered load (incoming traffic to the input queues). With reference to Fig. 4(a) and (b), there are three regions of operation: $\Lambda_o > \Lambda_{\max}$, $\Lambda_{\text{sat}} \leq \Lambda_o \leq \Lambda_{\max}$, and $\Lambda_o < \Lambda_{\text{sat}}$.

When $\Lambda_o > \Lambda_{\max}$, the input queues saturate because the input rate of packets is greater than the sustainable output rate, the throughput. If the greedy access policy is used, a new packet will enter the system as soon as a packet reaches its destination and leaves the system. Therefore, the achieved throughput $\Lambda = \Lambda_{\text{sat}}$, corresponding to the full link-loading situation when $\rho = 1$.

When $\Lambda_o < \Lambda_{\text{sat}}$, the queues will not saturate, and throughput equals offered load: $\Lambda = \Lambda_o$. When $\Lambda_{\text{sat}} \leq \Lambda_o \leq \Lambda_{\max}$, the system can either be in the stable or unstable regions. In the stable region, throughput equals offered load: $\Lambda = \Lambda_o$. In the unstable region, the system quickly evolves away from the region, ending up either at $\rho = 1$ and $\Lambda = \Lambda_{\text{sat}}$, the saturated case, or at $\rho \in (\rho', \rho_{\max})$ and $\Lambda = \Lambda_o$, the unsaturated case.

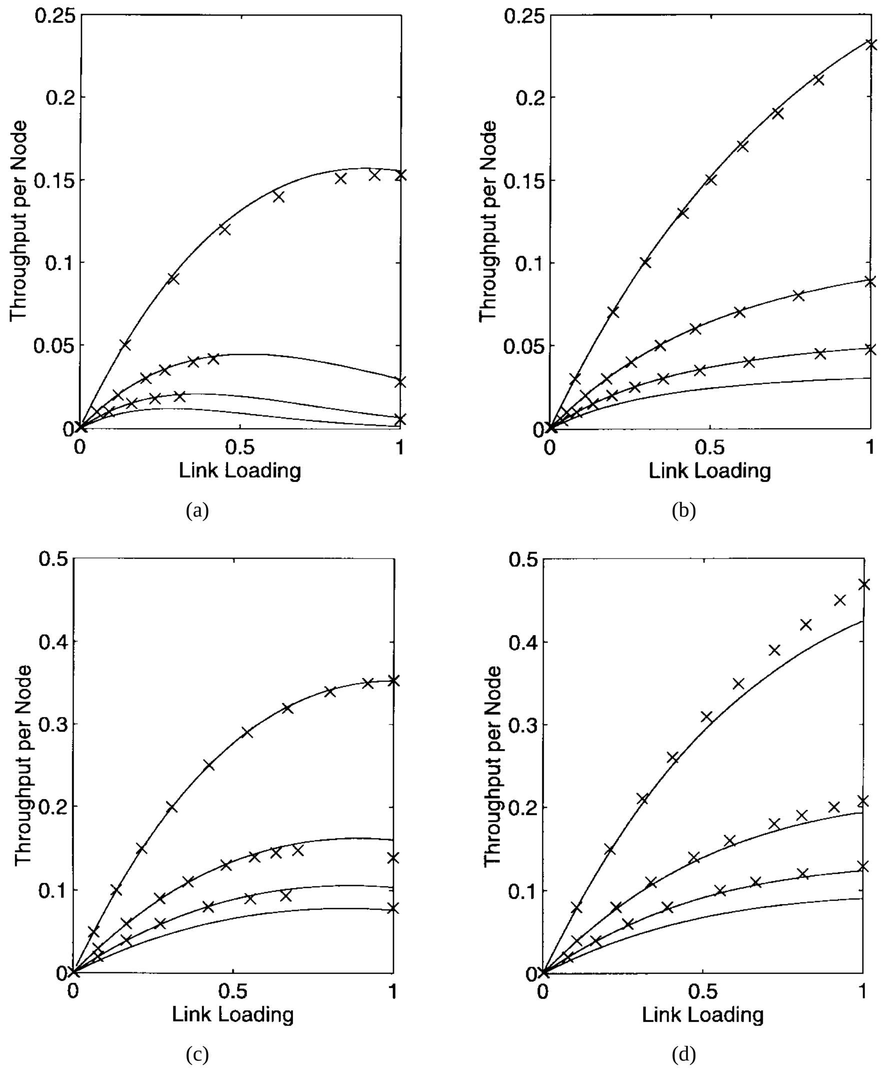

Fig. 5. Throughput per node versus link loading for $n = 5,\ 10,\ 15,\ 20$ (top to bottom curves in each graph) for: (a) shuffle network with random contention resolution (RCR), (b) shuffle network with shortest-distance priority contention resolution (SDPCR), (c) bidirectional shuffle network with RCR, and (d) bidirectional shuffle network with SDPCR. Solid lines are analytical results and plotted points are simulation results.

This kind of instability is akin to that in the Aloha multiaccess network, of which a discussion can be found in [15]. A more detailed explanation of the instability in the shuffle-exchange network can be found in [16].

A general stability condition can be derived for arbitrary symmetric networks (not just the shuffle-exchange network) as follows: Let there be M links in the system. Then, Little's Law gives $\Lambda = M\rho/T$, where T is the average delay. Differentiating Λ with respect to ρ, we have $d\Lambda/d\rho = [TM - M\rho(dT/d\rho)]/T^2$. Thus, $d\Lambda/d\rho \le 0$ if and only if $dT/d\rho \le T/\rho$. Thus, the general condition for stability is

$$\frac{dT}{d\rho} \le \frac{T}{\rho} \tag{5}$$

for all $0 \le \rho \le 1$, where $T'(\rho) = dT/d\rho$.

Fig. 5(a) plots throughput per node, Λ/N, versus ρ for $n = 5$, 10, 15, and 20 ($n = 5$ for the top curve and $n = 20$ for the bottom curve). The solid lines are the analytical results and the points are simulation results. Only the simulation data for $n = 5$, 10 are presented because the program for networks of $n = 15$ or larger takes considerable time to run.

From the analytical curves, it can be seen that each curve has an unstable region. This is further confirmed by in our simulation experiments in which no data points can be collected at the unstable region because the system equilibrium is not possible there.

When the network is large (large n) throughput per node degrades quite substantially. Furthermore, the instability problem becomes more severe: 1) ρ' and $\rho_{\max}$ become smaller so that the system can be "jolted" into the unstable region more easily and 2) Λ_{sat} becomes a smaller fraction of $\Lambda_{\max}$.

Table I presents with more precision the simulation results of a network with $n = 10$, assuming greedy access algorithm. As can be seen, the maximum throughput that can be reached is about 0.42. Also, when the offered load to the input queues is larger than can be sustained, the network becomes unstable, and the link loading becomes saturated. The throughput degrades to a very low value when instability occurs. The results in Table I also show that it is impossible to obtain throughput-versus-link-loading data at the unstable region in simulation (thus, confirming instability), although the region can be derived analytically.

TABLE I
SIMULATION RESULTS OF A SHUFFLE-EXCHANGE NETWORK WITH $n = 10$

Offered Load	*Link Loading*	*Throughput*	*Mean Delay*
0.010000	0.054383	0.010005	10.871390
0.020000	0.120500	0.020010	12.044059
0.030000	0.207470	0.030016	13.823862
0.040000	0.355659	0.040036	17.767030
0.041000	0.382201	0.041031	18.629557
0.042000	0.417855	0.042020	19.888816
0.042500	1.000000	0.028216	70.881493
0.045000	1.000000	0.028201	70.919990

The analysis is valid regardless of the specific access-control policy used, whereas the simulation assumes the greedy-access scheme. If an appropriate congestion-control access mechanism other than the greedy-access scheme is exercised, it is possible to stabilize the system and it will then be possible to operate in the previously unstable but now stabilized region (this has been confirmed in simulation experiments to be reported elsewhere). However, it is not desirable to operate in this region because there is a better operating point with the same throughput but lower link loading (hence, lower delay).

With appropriate access control, offered load of up to $\Lambda_{\max}$ can be sustained without instability. As an example of an access control scheme, we could introduce $2N\rho_{\max}$ tokens in the network. A packet is not allowed to enter the network until it has acquired a token from one of the input links, and the token will be released back to the network only when the packet has reached its destination. In this way, ρ is kept to be no more than $\rho_{\max}$.

B. Shortest-Distance Priority Contention Resolution

One might also explore contention-resolution policies that are more clever and hope that instability does not occur. In this section, we show that an arbitration policy that favors packets closer to their destinations will stabilize the system [6]–[8]. When a packet in state i is deflected back to state n, $n-i+1$ previous routing steps are wasted. The idea of favoring packets closer to their destinations is to minimize the amount of routing effort that is wasted given that deflection is inevitable.

The analysis of this strategy is more complicated because the deflection probability is state-dependent. Instead of the uniform deflection probability q, we have q_i for the deflection probability at state i, $1 \le i \le n$. Although tokens are not used in reality, we can still imagine for analytical purposes that there are $2N$ fictitious tokens, one on each link, circulating around the network. A packet at an input queue must acquire an unused token before it can enter the network. When the packet reaches its destination, the token is released and can be acquired by another packet.

A token is said to be active and in state i, $1 \le i \le n$, if the packet it carries is in state i. A token is inactive or in state 0 if it is unused. Note that as far as the evolution of the state of the token is concerned, there is no absorbing state, since it can get out of state 0 if it is acquired by another packet.

We trace the evolution of the state of a particular token. Let $P(i)$ be the probability that the token is in state i at the beginning of a time slot. We have

$$P(i-1) = P(i)(1-q_i), \qquad 2 \le i \le n. \tag{6}$$

Define

$$\pi(i) = P(1) + \cdots + P(i) \qquad 1 \le i \le n \tag{7}$$

to be the probability of the token in either state 1, 2, $\cdots$, or i. By definition, the probability of the token being active is $\pi(n)$. An active token implies the link it occupies is active. Since every link has a token on it at all time, the probability of an active token must also be the probability of an active link. Thus

$$\pi(n) = \rho. \tag{8}$$

The probability of a packet in state i being deflected is given by the sum of the probability of being deflected by a packet in state i and the probability of being deflected by a packet in state below i

$$\begin{aligned} q_i &= \frac{P(i)}{4} + \frac{P(1)+\cdots+P(i-1)}{2} \\ &= \frac{P(i)}{4} + \frac{\pi(i-1)}{2}, \qquad 1 \le i \le n. \end{aligned} \tag{9}$$

Substituting the above into (6) for $i = 1, 2, \cdots$, summing equations up to index i, and after some manipulation, we obtain

$$\pi(i-1) = [\pi(i) - \pi(1)]\left\{1 - \frac{\pi(1)}{2} - \left[\frac{\pi(i)-\pi(1)}{4}\right]\right\}, \quad 2 \le i \le n. \tag{10}$$

Equation (10) is subject to this interpretation: the probability that the token is active and in state $i-1$ or lower is equal to the probability that it was in the previous time slot an active token in state between two and i and that the packet it carries was not deflected.

The probability of a token in state 1 not being deflected is $1 - \pi(1)/4$, and not being deflected means the packet carried by it reaches the final destination. Therefore the throughput of the network is given by

$$\Lambda = 2N\pi(1)\left[1 - \frac{\pi(1)}{4}\right]. \tag{11}$$

Unfortunately, $\pi(1)$ cannot be solved in closed form in terms of ρ from (8) and (10).

Equations (8) and (10), however, do let us show that the kind of instability associated with random contention-resolution policy does not occur here. The key is to show that $d\Lambda/d\rho > 0$ for all $0 \le \rho \le 1$. From (11)

$$\frac{d\Lambda}{d\pi(1)} = 2N\left[1 - \frac{\pi(1)}{2}\right] > 0. \tag{12}$$

From (10)

$$\frac{d\pi(i)}{d\pi(i-1)} = \frac{\left\{1 + \left[1 - \frac{\pi(1)}{2}\right]\frac{d\pi(1)}{d\pi(i-1)}\right\}}{\left[1 - \frac{\pi(i)}{2}\right]}, \quad 2 \le i \le n. \tag{13}$$

Substituting $i = 2$, we find that $d\pi(2)/d\pi(1) > 0$. It can be easily shown by induction using (13) that $d\pi(i)/d\pi(1) > 0$ for all $1 \le i \le n$. Thus

$$\begin{aligned}\frac{d\Lambda}{d\rho} &= \frac{d\Lambda}{d\pi(n)} \\ &= \frac{d\Lambda}{d\pi(1)}\frac{d\pi(1)}{d\pi(n)} \\ &> 0. \end{aligned} \tag{14}$$

Fig. 5(b) shows throughput per node versus link loading for $n = 5$, 10, 15, and 20. As shown, there is a marked improvement in the maximum achievable throughput compared with random contention resolution. It can also be verified that there is no instability. Although instability does not occur, the deflection penalty is still rather significant for large n. The next section considers a version of shuffle network in which the shuffle links are bidirectional so as to reduce the deflection penalty.

IV. Bidirectional Shuffle Network

Suppose that the links of the shuffle-exchange network is bidirectional so that in each time slot, a packet can travel in the backward direction while another one can travel in the forward direction. In actual implementation, each node is 4 × 4 in dimensions, and there is a set of "unshuffle" links laying side-by-side with the set of shuffle links. When a packet is deflected from node i to node j, thanks to the link in the reverse direction connecting node j to node i, the packet can travel back to node j from node i in the next time slot, correcting the deflection in one step.

In general, a packet can be deflected a number of times in succession. In the above scenario, for example, the packet deflected from node i to node j can be deflected again from node j to node k. So, one can easily come up with scenarios in which the packets are deflected a number times, correctly routed a number of times, but before returning back to node i, are deflected again. But in each case, there is always a route back to node i, and ways can be found to keep track of the deflections and therefore the correction steps for them can be devised systematically.

A packet that enters the network can choose to travel to its destination via two possible routes: the shuffle route is the same as that in the original basic shuffle network, and the unshuffle route consists of a set of unshuffle links. In the latter, the order in which the destination address bits are used for routing is reversed (i.e., starting from bit d_1 up to bit d_n). We assume here that the shuffle and unshuffle routes are chosen with equal probability. A packet at a node generally wants to go to one output but could be deflected to one of the other three outputs. Deflections to a shuffle link are corrected by traversing in a reverse-direction unshuffle link and deflections to an unshuffle link are corrected by traversing in the reverse-direction shuffle link. A routing scheme that manipulates the packet headers in a systematic fashion to convey the routing actions needed to correct deflections can be found in [8]. (Note: although [8] concerns an indirect network rather than direct network, the routing mechanism can be extended to the undirected network in a straightforward manner.)

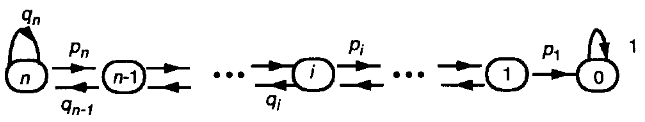

Fig. 6. The state-transition diagram of a packet in the bidirectional shuffle network.

Fig. 6 shows the state-transition diagram in which the state is distance to destination. Each deflection causes the distance to increase by one. There is a "reflection" boundary at state $n = \log_2 N$, where the distance remains the same even after a deflection. Strictly speaking, the distance alone does not fully capture all the essential components for a precise performance study. The deflection probability of a packet, say on a shuffle input (unshuffle) link, also depends on whether it wants to go to a shuffle (unshuffle) or unshuffle (shuffle) output link. The complete analysis is rather involved, and we will make the simplifying assumption that the deflection probability only depends on the distance of the packet. In addition, we will also assume that an arriving packet to a node that is not its final destination is equally likely to be destined for any of the four outputs.

A. Random-Contention Resolution

For the random-contention resolution scheme, $p_i = p$ and $q_i = q$. As before, let $T(i)$ be the expected number of additional hops a packet in state i will experience before reaching the destination node. Then

$$T(0) = 0 \tag{15}$$

$$T(i) = 1 + pT(i-1) + qT(i+1), \quad 1 \le i \le n-1 \tag{16}$$

$$T(n) = 1 + pT(n-1) + qT(n). \tag{17}$$

It is routine to solve the above difference equation with the two boundary equations to yield [17]

$$\begin{aligned} T(i) &= T(i) - T(0) \\ &= \sum_{j=1}^{i} M(j) \\ &= \frac{i}{p-q} - \frac{\left(\frac{1}{p-q} - \frac{1}{p}\right)\left[\left(\frac{p}{q}\right)^i - 1\right]}{\left(\frac{p}{q}\right)^{n-1}\left(\frac{p}{q} - 1\right)}. \end{aligned} \tag{18}$$

To find p in terms of link loading ρ, focus on a particular output and consider the probability of at least one input packet desiring it. This is $1 - (1 - \rho/4)^4$. If there is at least one input packet targeted for the output, there is an undeflected packet on it after the routing at the node. Thus, this is also the probability of finding an undeflected packet on an output link. By conservation, the probability must equal the probability of finding a packet on an input link and that it will not be deflected, ρp. Hence

$$p = \frac{1-\left(1-\frac{\rho}{4}\right)^4}{\rho}. \tag{19}$$

Let us examine $T(n)$ for large n. Substituting $i = n$ into (18) and after some minor simplification, we obtain

$$\begin{aligned} T(n) &= \frac{n}{p-q} - \left(\frac{1}{p-q} - \frac{1}{p}\right)\left[\frac{1-\left(\frac{q}{p}\right)^n}{1-\frac{q}{p}}\right] \\ &\le \frac{n}{p-q}. \end{aligned} \tag{20}$$

In particular, $T(n)$ is dominated by the term $n/(p-q)$ for large n, and we can write

$$\begin{aligned} T(n) &\approx \frac{n}{p-q} \\ &\equiv \frac{n}{2p-1}. \end{aligned} \tag{21}$$

Comparing the above with (3), the expected delay in the basic shuffle network, we notice a very significant improvement: whereas the expected delay in (3) grows exponentially with n, the expected delay here grows only linearly with n. We should expect a marked improvement in throughput also.

Let us consider the system throughput Λ as a function of link loading ρ. There are four links to a node. By Little's Law,

$$\Lambda = \frac{4N\rho}{T(n)}. \tag{22}$$

With $\rho = 1$ in (19), (21), and (22), we get

$$\Lambda_{\text{sat}} \approx \frac{1.469N}{n}. \tag{23}$$

Network Congestion and Instability: We now show that the instability problem exists in this network. Differentiating (19) with respect to ρ, we have

$$\rho\frac{dp}{d\rho} = \left(1-\frac{\rho}{4}\right)^3 - p. \tag{24}$$

Using the above in differentiating the approximate relationship in (21) with respect to ρ, we can get

$$\frac{\rho}{T(n)}\frac{dT(n)}{d\rho} = 1 + \frac{1-2\left(1-\frac{\rho}{4}\right)^3}{2p-1}. \tag{25}$$

From the above and the general stability condition of (5), the system is stable if

$$\frac{1-2\left(1-\frac{\rho}{4}\right)^3}{2p-1} \le 0$$

for all $0 \le \rho \le 1$. It can be verified from (19) that $(2p-1) > 0$ for $0 \le \rho \le 1$. Thus, stability would require that $1 - 2(1-\rho/4)^3 \le 0$, from which we can get

$$\rho \le 4\left(1-\frac{1}{2^{1/3}}\right) = 0.825.$$

Thus, the system becomes unstable once ρ exceeds 0.825. However, the instability is less severe than that in the basic shuffle network for two reasons: 1) in the basic shuffle network, the critical ρ is much smaller and 2) the critical ρ decreases to zero as n increases.

Fig. 5(c) shows throughput per node versus link loading for $n = 5$, 10, 15, and 20. Compared with Fig. 5(a), where the results of the basic shuffle network are plotted, we see a substantial throughput improvement. Furthermore, instability is minor and that $\Lambda_{\max}$ is close to Λ_{sat}. Although instability is still caused by packets with large distance values winning over packets with small distance values, the smaller deflection penalty accounts for the less severe form of instability. That Λ_{sat}/N is inversely proportional to n is also indicated by these curves.

B. Shortest-Distance Priority Contention Resolution

We now investigate the stabilizing effect brought about by the shortest-distance priority contention resolution scheme. The notation used will be the same as that in Section III-B. For a packet in state i, the probability of being deflected is

$$q_i = u_i + v_i$$

where u_i is the probability of being deflected by a packet of the same state and v_i is the probability of being deflected by a packet of a lower state. For each of the other inputs, $x = P(i)/4$ is the probability of finding a packet of the same state destined for the same output. The arbitration among packets of the same state is random: i.e., if there are j packets of the same state for the same output, each packet is the winner with probability $1/j$, we have

$$\begin{aligned} u_i &= \tfrac{1}{2}\binom{3}{1}x(1-x)^2 + \tfrac{2}{3}\binom{3}{2}x^2(1-x) + \tfrac{3}{4}\binom{3}{3}x^3 \\ &= \tfrac{1}{4}x^3 - x^2 + \tfrac{3}{2}x \\ &= \tfrac{1}{256}P^3(i) - \tfrac{1}{16}P^2(i) + \tfrac{3}{8}P(i). \end{aligned} \tag{26}$$

Let $y = \pi(i-1)/4$, which is the probability of finding a packet of a lower state destined for the same output. Then,

$$\begin{aligned} v_i &= \binom{3}{1}y(1-y)^2 + \binom{3}{2}y^2(1-y) + \binom{3}{3}y^3 \\ &= 1-(1-y)^3 \\ &= 1-\left[1-\tfrac{1}{4}\pi(i-1)\right]^3. \end{aligned} \tag{27}$$

For $i = 1$, $v_1 = 0$, and $q_1 = u_1$ since packets cannot have a lower state than one.

For $2 \le i \le n-1$, $P(i) = P(i+1)p_{i+1} + P(i-1)q_{i-1}$, from which we can get

$$\begin{aligned} \pi(i) &= P(i-1) + \cdots + P(2) + P(1)q_1 \\ &\quad + P(i)p_i + P(i+1)p_{i+1} \\ &= \pi(i-1) - \pi(1)p_1 + [\pi(i) - \pi(i-1)]p_i \\ &\quad + [\pi(i+1) - \pi(i)]p_{i+1} \end{aligned} \tag{28}$$

$$\begin{aligned} &= \pi(i-1)q_i + \pi(i)(p_i - p_{i+1}) \\ &\quad + \pi(i+1)p_{i+1} - \pi(1)p_i. \end{aligned} \tag{29}$$

Let λ be the probability an active packet (fictitious token) reaching its destination in a given time slot. Then

$$\pi(1)p_1 = \lambda \tag{30}$$

where $p_1 = 1 - q_1$ is a function of $\pi(1)$ only. Equation (30) becomes a fourth-order polynomial in $\pi(1)$ after the substitution of p_1 in terms of $\pi(1)$. Given a throughput Λ, $\lambda = \Lambda/4N$, from which we can then solve for $\pi(1)$ by the finding the root of the polynomial (30) which is between zero and one. From $\pi(1)$, we can get $\pi(2)$ by substituting it into

$$\pi(1) = [\pi(2) - \pi(1)]p_2 \tag{31}$$

where $p_2 = 1 - q_2$ is a function of $\pi(2)$ and $\pi(1)$. Again, $\pi(2)$ can be found by solving for the root of a polynomial. After that, $\pi(i)$ for $3 \leq i \leq n$ can then be found using the dynamic equation (29). This numerical method is not iterative and one pass will let us find $\pi(i)$ for all i. The link loading is then found by $\pi(n) = \rho$. Note that this method finds ρ given a Λ rather than the other way round.

Fig. 5(d) plots the results of the numerical calculations. Compared with Fig. 5(c), in which random contention resolution is assumed, we see only slight improvements in throughput performance (relative to those seen in the basic shuffle-exchange network). This is because the deflection penalty in the bidirectional shuffle network is not as high as in the basic shuffle network. Therefore, even though the number of deflections has been reduced with the shortest-distance priority scheme, the effect is not large. As can also be seen, the improvements are only obvious when ρ is close to one, where the random contention resolution scheme is unstable.

V. Conclusion

This paper has investigated the issue of stability in shuffle-exchange and bidirectional shuffle networks when operated with deflection routing. In a stable packet-switched network, throughput equals offered load and packet backlogs do not build up in an unbounded fashion. We have shown analytically the possibility of the existence of instability when packet contention in the networks is resolved randomly. The existence of instability has been confirmed by simulation. Resolving contention in a manner that favors packets closest to their destinations is an obvious way to improve the throughput performance in the shuffle-like networks. Perhaps more importantly, the shortest-distance priority scheme removes instability. This implies that elaborate network access control for instability prevention is not needed and the simple greedy access policy can be used. Although not shown in this paper, the above observations are also true for the stay-or-shuffle network [6], as detailed in [16].

Acknowledgment

The author thanks C. Chan for helping with generating the simulation data.

References

[1] C. L. Wu and T. Y. Feng, *Tutorial: Interconnection Networks for Parallel and Distributed Processing.* Los Alimitos, CA: IEEE Computer Society Press, 1984.

[2] P. Baran, "On distributed communications networks," *IEEE Trans. Commun.,* vol. COM-12, pp. 1–9, 1964.

[3] N. F. Maxemchuk, "Routing in the Manhattan street network," *IEEE Trans. Commun.,* vol. COM-35, pp. 503–512, May 1987.

[4] N. F. Maxemchuk, "Comparison of deflection and store and forward techniques in the Manhattan street network and shuffle exchange networks," in *Proc. IEEE INFOCOM '89,* pp. 800–809.

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

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

[7] D. H. Lawrie and D. A. Padua, "Analysis of message switching with shuffle-exchanges in multiprocessors," in *Proc. Workshop on Interconnection Networks for Parallel and Distributed Processing,* 1980, pp. 116–123. Also reprinted in [1].

[8] S. C. Liew and T. T. Lee, "$N \log N$ dual shuffle-exchange network with error-correcting routing," *IEEE Trans. Commun.,* vol. 42, pp. 754–766, Feb./Mar./Apr. 1994.

[9] A. K. Choudhury and V. O. K. Li, "An approximate analysis of the performance of deflection routing in regular networks," *IEEE J. Select. Areas Commun.,* vol. 11, pp. 1302–1316, Oct. 1993.

[10] S.-H. Chan and H. Kobayashi, "Performance analysis of shufflenet with deflection routing," in *Proc. GLOBECOM '93,* pp. 854–859.

[11] A. G. Greenberg and J. Goodman, "Sharp approximation models of deflection routing in mesh networks," *IEEE Trans. Commun.,* vol. 41, pp. 210–223, Jan. 1992.

[12] J. Brassil, A. K. Choudhury, and N. F. Maxemchuck, "The Manhattan street network: A high performance, highly reliable metropolitan area network," *Computer Networks ISDN Syst.,* no. 26, pp. 841–858, 1994.

[13] K. Khanabish, "A new method for evaluating packet routing policies in supra-high-speed metropolitan (or wide) area networks," *Computer Networks ISDN Syst.,* vol. 26, pp. 195–216, 1993.

[14] Z. Zhang and A. Acampora, "Performance analysis of multihop lightwave networks with hop-potato routing and distance-age priorities," in *Proc. INFOCOM '91,* pp. 1012–1021.

[15] D. Bertsekas and R. Gallager, *Data Networks,* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[16] S. C. Liew, "On the stability of shuffle-like interconnection network employing reflection routing," Dept. of Information Engineering, The Chinese Univ. of Hong Kong, Tech. Rep. TRR001.SCL.95.

[17] W. Feller, *An Introduction to Probability Theory and Its Application.* New York: Wiley, 1957.

Soung C. Liew, photograph and biography not available at the time of publication.