A General Packet Replication Scheme for Multicasting in Interconnection Networks

Soung C. Liew
Department of Information Engineering, Chinese University of Hong Kong

***Abstract*— Multicasting in broadband packet switches and metropolitan networks can be achieved by first replicating the packets and then routing them to their destinations. This paper studies a very simple but general replication scheme that can be applied to arbitrary interconnection-network topologies. The replication process of a packet adapts itself according to the network topology and the traffic condition. Hot spots of replication activities are diffused by this scheme which automatically moves part of the replication efforts to less active network regions. The scheme can potentially be used in networks (e.g., the Manhattan-street network) in which multicasting were thought to be inherently difficult. Fundamental issues and critical problem areas are laid out, and solutions addressing them are proposed. The performance of the replication algorithm and its implementation (logic diagram level) in the shuffle-exchange copy network are investigated in detail. It is found that the performance of the algorithm improves with the increase of network dimensions.**

I. Introduction

Multicasting, the delivery of information from one source to a several destinations in a network, is expected to be an important feature in future broadband networks [1]. Video teleconferencing, video-program vending, distributed computing and control, and many other advanced network services and applications involve multicast communication.

The topologies of interconnection networks [2] are suitable for both packet switches as well as local or metropolitan networks. One way to achieve multicasting in interconnection networks is to divide the multicasting process into two separate processes: replication and routing [1, 3, 4]. During replication, copies of made of a master packet without regard to which outputs (or destinations) the packet copies would end up. After replication, packet copies are then routed to their specific targeted outputs (destinations). This paper considers only the replication process.

The replication algorithms studied previously [1, 3, 4] are for specific copy-network architectures. This paper's approach, on the other hand, is applicable to arbitrary network topologies. This generality is made possible by a fundamental feature: in all previous approaches, the paths taken by a packet, referred to as the *replication tree*, is determined before the actual replication begins. Our approach, in contrast, determines the tree structure on the fly as the packet travels through the network. In other words, our approach adapts the replication process according to the network topology and the contention from other packets.

The remainder of this paper is organized as follows. In Section II, we review the important concepts put forth in previous work and show how they can be generalized for arbitrary network topologies. Section III considers the application of our approach to the elongated shuffle-exchange network. Section IV presents the performance results and their implications. Finally, Section V concludes this work and discusses areas for further work.

II. General Packet Replication Algorithm

Let us first focus on copy networks based on the shuffle-exchange topology [2, 5] before relating the algorithm to general network topologies. As shown in Fig. 1, a shuffle-exchange network consists of stages of 2×2 nodes interconnected in a regular fashion. We shall assume fixed-length packets and time-slotted operation of the copy network. In each time slot, a set of packets arrive at the inputs. The copy network produces the desired packet copies at the outputs.

A. Static Replication Tree

This subsection describes the packet replication trees associated with previous work. The approach depicted in Fig. 1 was proposed in [4]. It defines the replication tree in terms of a contiguous output-address interval as follows. Let the address interval be represented by a tuple (MIN, MAX) where MIN and MAX are the minimum and the maximum of the interval. The example in the figure assumes the assignment of interval $(0, F-1)$ to the packet, where F, the fanout, is the desired number of copies. In general, other intervals (i.e., $(i, F+i-1)$ for some i) can also be used. Reference [4] shows how output intervals can be assigned to the input packets in such a way that there is no internal conflict inside the shuffle-exchange network, and therefore internal buffering is not needed.

One nice feature about the algorithm of the approach is that only two bits need to be examined in each stage.

0743-166X/95 $04.00

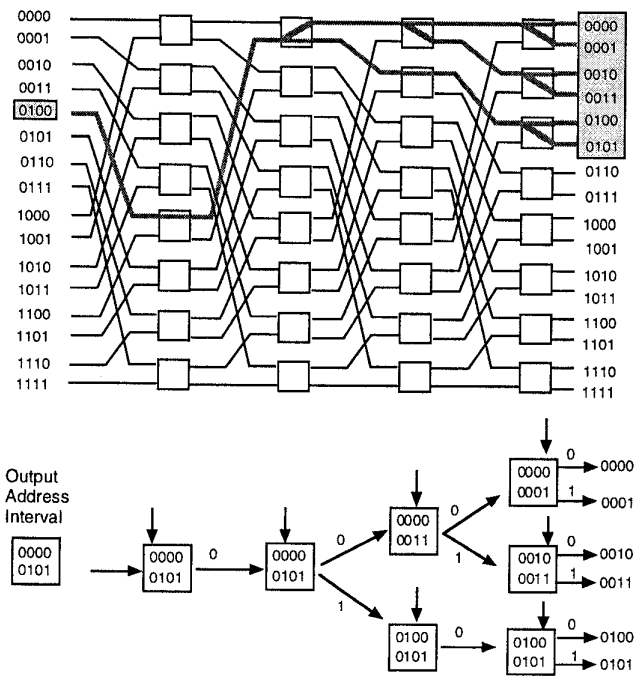

Fig. 1. Interval-splitting algorithm applied on a shuffle-exchange network.

Specifically, let the binary representation of the address interval be $(m_1 m_2 \ldots m_n, M_1 M_2 \ldots M_n)$. At stage k bits m_k and M_k are examined and routing decisions are made as follows:

Interval-splitting Scheme

1. If $m_k = M_k = 0$ or $m_k = M_k = 1$, then send the packet out on link 0 (upper link) or 1 (lower link), respectively.
2. If $m_k = 0$ and $M_k = 1$, then duplicate the packet and modify the header (according to the scheme described below) and send both packets out on both links.
   (a) For the packet sent out on link 0, MIN remains unchanged, and MAX is set to $M_1 \ldots M_{k-1}01 \ldots 1$.
   (b) For the packet sent out on link 1, MAX remains unchanged, and MIN is set to $M_1 \ldots M_{k-1}10 \ldots 0$.

Physically, the above interval splitting means that the upper branch is responsible for replicating and routing packets to output interval $(m_1 \ldots m_n, M_1 \ldots M_{k-1}01 \ldots 1)$, and the lower branch to output interval $(M_1 \ldots M_{k-1}10 \ldots 0, M_1 \ldots M_n)$. These two intervals are contiguous to each other, and together they cover the original interval. Note from the above rules that $m_i = M_i, i = 1, \ldots, k-1$ holds for every packet that arrives at stage k (this can be easily argued by induction on k). Therefore, the event $m_k = 1$ and $M_k = 0$ is impossible due to the min-max representation of the address intervals.

To guarantee nonblocking operation of the shuffle-exchange network, the input packets must be assigned disjoint output address intervals and be concentrated before being forwarded to the shuffle-exchange network [4]. The preprocessing hardware is more complex than the shuffle-exchange network itself.

B. Dynamic Replication Tree

Doing away with the preprocessing will result in a blocking copy network. However, this can be dealt with by delaying the replication process whenever conflict occurs. This strategy is elaborated as follows. Instead of a $\log_2 N$-stage network, we have an L-stage network in which $L \geq \log_2 N$. And instead of assigning a static replication tree to a packet, the replication tree is dynamically modified based on contention from other packets.

Figure 2 illustrates the idea with an example in which the interval $(0, F-1)$ is assigned to all packets. In [4], the address interval specifies the copy network output addresses for the packet copies. Here, we only use the interval for replication purposes and the packet copies do not necessarily end up at outputs 0 to $F-1$. In the example, there are two packets, A and B, requesting for two and six copies, respectively. Let us track the replication of packet B.

Initially, its address interval is $(0000, 0101)$. Unlike the original interval-splitting algorithm, bits k are not necessarily used at stage k here. In our example, the first bits of 0000 and 0101 are both 0, and so we progress directly to the second bits – in general, we skip to the first bit position where MIN and MAX differ so that splitting can be performed immediately. After splitting, both copies of packet B find a copy of packet A at the other input of the same node at the second stage.

Let us first look at the upper copy of packet B. Since we do not want to buffer packets, duplication at this stage is not possible because each input packet must exit on one of the outputs. The duplication is therefore delayed until the third stage. This means that the third stage should examine the same bit position as the second stage to decide whether duplication is required. In the example, replication is possible at the third stage because the other input does not have a packet. The

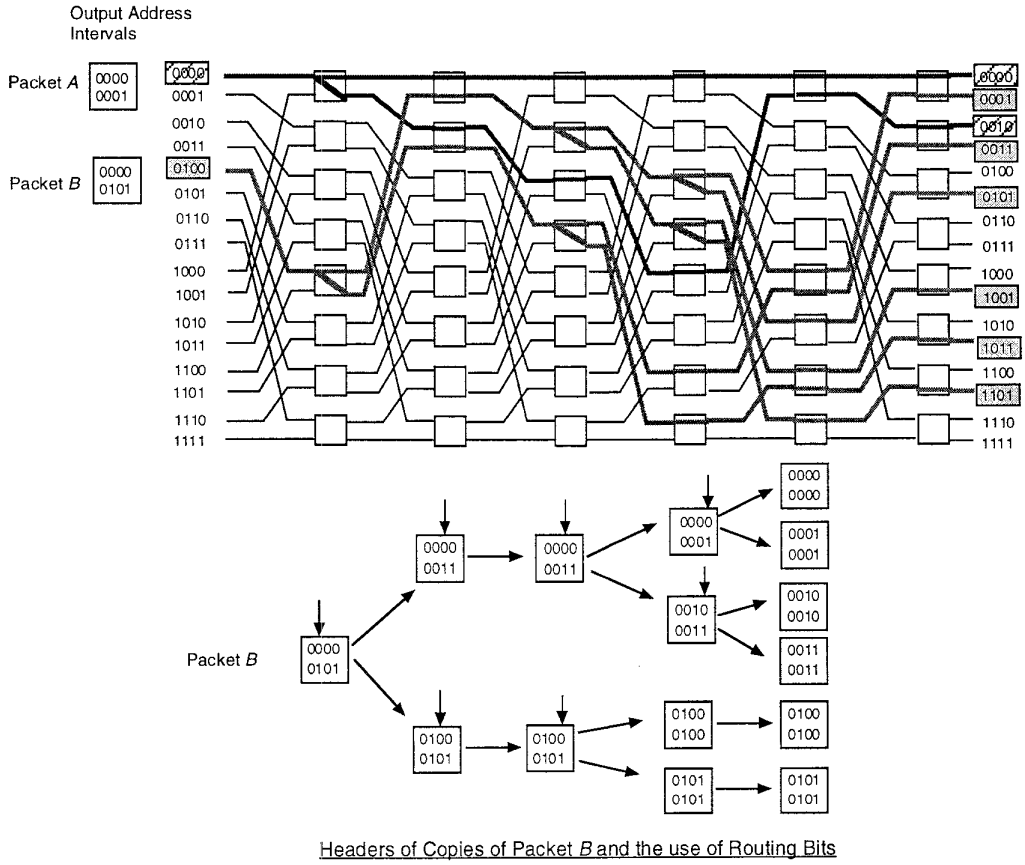

Fig. 2. Generalized interval-splitting algorithm applied on a shuffle-exchange network.

replication process is completed after five stages.

Now look at the lower copy of packet B at the second stage. Both bits examined are 0 and replication is not needed. Therefore, unlike the upper copy, we can progress to the next bit position at the third stage here. In general, only when replication is required and is denied will we examine the same bit position at the next stage.

The general algorithm can be described as follows:

Generalized Interval-splitting Algorithm

1. Initially, skip directly to the first bit position where MIN and MAX differ.
2. At any stage, examine the two routing bits.
   (a) If the bits are different and the other input of the switch node does not have a packet, perform splitting according to the original interval-splitting algorithm.
   (b) If the bits are different and the other input of ths switch node has a packet, do not perform splitting. At the next stage, the same two bits will still be examined.
   (c) If the bits are the same, progress to the next bits at the next stage.

Since the bits to be examined at stage k are not necessarily bits k, a way to indicate the effective routing bits is needed. This can be solved rather easily by positioning the effective routing bits at the front of the packet header at all times so that they are by default the first two bits. For conciseness, however, the details of implementation will not be discussed in this paper.

There is a tradeoff between the number of stage L and the packet-loss probability. If the MIN and MAX of a packet copy remains different after L stages, its replication is not completed yet, and some copies are said to be lost. Since there are more chances for replication attempts for larger L value, the loss probability will corre-

spondingly be smaller. On the other hand, the network is more complex.

C. Application to Arbitrary Topologies

Let us now illustrate the generality of the above strategy. The first observation is that the correctness of the algorithm does not depend on the shuffle-exchange pattern. The adjacent stages can be interconnected in any fashion, either regularly or irregularly. Butterfly, baseline and other patterns are all possible [2].

The same scheme also applies to "closed" networks. In these networks, the sources and destinations of packets are nodes rather than input and output links. An example is the shuffle-exchange network consisting of only one stage in which the output links of the stage are connected back to the inputs of the same stage via a shuffle pattern. For closed networks, the same network can be used for both replication and routing: the replication algorithm is performed, and a copy is routed to its destination when no further splitting is required of it. Since the copies of the same master packet may be produced at different times, some copies may embark on their journeys to their actual destination nodes while the others are still being replicated. Also, packets of different multicast connections may be replicated and routed simultaneously, and the network operation does not have to be divided into separate replication and routing phases.

Let us consider the application of the generalized interval-splitting algorithm in the Manhattan-street network. Point-to-point routing in this network has been well studied [6, 7]. However, support of multicasting is perceived to be a major challenge [7]. Our framework can be used to incorporate the multicasting capability into the network. Figure 3 shows the application of the generalized interval-splitting algorithm in a network with 16 nodes [6, 7]. In this network, the nodes are arranged as a 4×4 square grid. Each node has two inputs and two outputs. The data flows on the rows (columns) alternate between left-to-right (upward) and right-to-left (downward). The example adopts the interval-splitting algorithm. We have a packet at node $(1,1)$ requesting for six copies, and its splitting interval is set to $(0000, 0101)$. We end up with two copies each at nodes $(0,3)$ and $(1,2)$ and one copy each at nodes $(2,1)$ and $(3,0)$.

In general, when several packets are being replicated and routed, contention may occur and the replication process on some branches may be delayed. In addition, in closed networks, copies of the same master packet may interfere with each other. For example, this happens in Fig. 3 if the two copies at node $(0,3)$ is to be split further (i.e., more than six copies are requested). Another situation that must be taken care of in closed networks

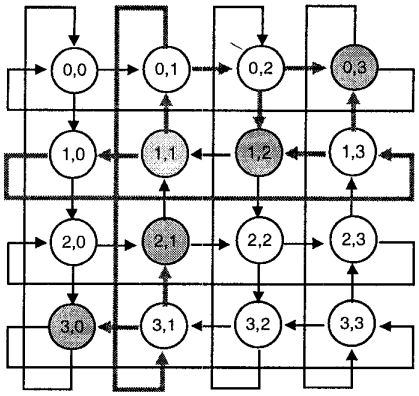

Fig. 3. Generalized interval-splitting algorithm applied on a Manhattan-street network: an example in which the source node is (1,1) and the packet requests six copies.

is when all the links have a packet and the splitting process of none of them is completed yet. In this situation, the copies circulate around the network indefinitely. A mechanism must be installed to either prevent this from happening or break the deadlock.

The packet replication algorithm can readily be generalized to networks in which the nodes are not 2×2 (e.g., the hypercube network). The basic idea remains the same, except that now *partial* splitting is possible. For instance, when two packets enter a 4×4 nodes both wanting to split three ways, we may split both packets two ways.

III. Shuffle-Exchange Copy Networks

This section discusses the details of the shuffle-exchange copy network. The problems identified and the solutions proposed also apply to many other network topologies with minor modifications.

A. Distinguishing Packet Copies

At the outputs of a copy network, packets of different multicast connections are distinguished using a *broadcast channel number* (BCN) [4]. In addition, the copies of the same master packet must also be distinguished so that they can be routed to their respective outputs. This is achieved using an *index reference* (IR) [4]. A distinct (BCN, IR) pair identifies a distinct output. This mapping is usually stored in a memory [4].

All copies of the same master packet have the same BCN, and so the BCN can be incorporated into the header of the packet and replicated during the splitting process. The IRs of different copies must be distinct.

For generation of distinct IRs, first consider the interval-splitting algorithm. Suppose that the interval $(0, F-1)$ is assigned to all master packets. After successful replication, there will be F copies, all with MIN equal to MAX in their interval fields. Furthermore, the MIN (MAX) of the different copies are distinct and range from 0 to $F-1$. Therefore the MIN (or MAX) value can be used as the IR and no explicit IR field is needed in the header. If the replication process is not completed yet for a copy at an output of the copy network, then its MIN $<$ MAX, and (MAX $-$ MIN) copies will be considered as lost. The copy chosen to be the successful one can have IR anywhere between MIN and MAX, inclusively.

If a general interval $(i, F+i-1)$ is assigned to the master copy such that the offset i of the associated multicast connection can be changed time slot to time slot in a way that depends on the traffic condition, then i must be written into a dedicated field in the header so that it taken into account when deriving the IR.

B. Deadlock Prevention

In general, the larger the number of stages L in the shuffle-exchange network, the more likely the copy requests of packets can be fulfilled, since larger L implies there are more chances for packet replication attempts. There are two phenomena, however, that may prevent the completion of the replication process, no matter how large L is.

This first is obvious, and it is request overflow, which occurs when the sum of the fanouts of input packets exceed N, the capacity of the network. Subsection C discusses ways for dealing with overflow. The second phenomenon is deadlock, which occurs even when the total copies requested is less than N if certain deterministic routing policies are used,

Figure 4 shows an example of deadlock with the following policy: at a 2×2 node, if there are two input packets, set the switch element to *bar* state (i.e., forward the upper input packet to the upper output and the lower input packet to the lower); also set the switch element to bar state under the "don't care" situation when one input has a packet that does not need to be replicated while the other input does not have an active packet. In the example, the top input has a packet with $F = 2$, the bottom input does not have an active packet, and each of the rest of the inputs has a packet with $F = 1$. It is easily seen that with the above policy, no matter how large L is, the top packet will stay at the top link while the bottom link remains idle throughout all the stages.

There are two ways to prevent deadlocks. One is to use random-routing policy. Whenever we have a "don't care" situation, set the switch element randomly either to bar or cross state (upper input connected to lower output and lower input to upper output). One disadvantage with this approach is the need for implementing a random-number generator at each switch node. The alternative is to devise deterministic routing policies that are deadlock free. We explain below a very simple deadlock-free policy.

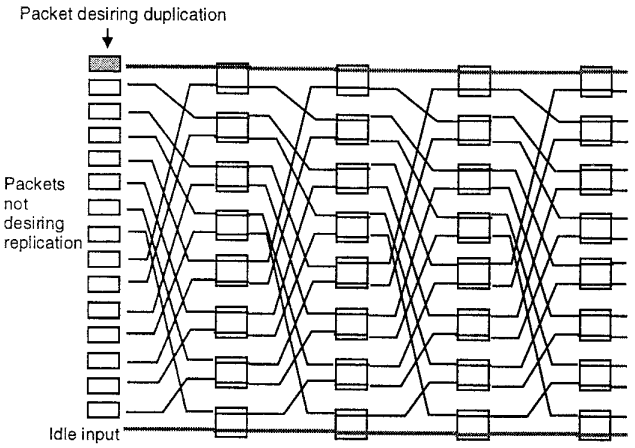

Fig. 4. A deadlock example with a deterministic routing policy: set switch elements into bar state under "don't care" situations. A packet fails to be duplicated regardless of the number of stages while the bottom link is idle at each stage.

For simple explanation, one can picture a token on each idle link. In order for a packet to be duplicated at a node, it must acquire a token on the other input. Thus, the problem becomes that of devising a routing policy that will ensure the meeting of a token and the packet desiring replication. There are two "don't care" situations that must be made into "care" conditions: 1) whenever there is a token at one input and the other input does not have a packet that desires duplication, we forward the token to the upper output; 2) whenever the two inputs are occupied and at least one packet desires duplication, we forward that packet to the upper output.

To see why this strategy works, suppose we label the nodes in each stage from top to bottom in a binary fashion. The construction of the shuffle-exchange network is such that a packet or a token residing at node $d_1 d_2 \ldots d_{n-1}$ will be forwarded via the upper outgoing link to node $0 d_1 d_2 \ldots d_{n-2}$ at the next stage [2, 5]. Therefore, the above policy can be viewed as attempting to route tokens and packets desiring replication to the top node $00\ldots0$. Thus, if there is a token and a packet that desires replication in the network, they will eventu-

ally meet at node $00\ldots0$, if not earlier. Consequently, deadlocks cannot occur.

In the worst case, a packet can be duplicated at node $00\ldots0$ every $\log_2 N$ stages. In actuality, packets desiring replication and tokens meet much more often. This becomes evident when we view the routing policy as attempting to concentrate tokens and packets desiring replications at nodes with many 0s (but not necessarily all 0s) in their labels. Packets whose replication has been completed are "pushed" to nodes with many 1s in their labels.

C. Overflow Problem

Overflow occurs when the sum of the fanouts of input packets exceed N. There are two classes of approaches to this problem. The first is to incorporate a reservation (contention-resolution) mechanism so that the packets allowed to enter the copy network desire at most a total of N copies. The overflow requests are buffered at the inputs so that replication can be attempted in the next time slot. This approach has been well explored [8, 9, 10]. It has been shown that the delay and loss probability at the input buffers can be made rather small when the arrivals of overflow requests are not bursty. Situations in which the arrived requests are positively correlated in successive time slots may lead to worse performance and they remain to be investigated.

One of the goals of the copy network proposed in this paper is to eliminate preprocessing. For this, we examine the second approach, which is to increase the bandwidth (capacity) inside the network with respect to the external bandwidth.

One possibility is to employ expansion [4, 11]. An $M \times M$ ($M > N$) shuffle-exchange copy network could be used [4, 11] in such a way that only the upper N of its M input ports are connected to input links. Because only N of the input links are used, some switch elements at the initial stages are guaranteed not to be traversed by packets. An $N \times M$ expansion network results when these unused switch elements are removed. In this way, the capacity of the network is expanded to M and the internal load to the copy network is reduced, making the occurrence of overflow less likely.

Another possibility is to speed up the internal link with respect to the external link. With a speedup factor of two, for example, each external time slot corresponds to two internal time slots, and by dividing the copy requests into two batches, one for each internal time slot, the overflow probability can be reduced. With this approach, if the point-to-point switch that follows the copy network does not employ speedup, buffers are needed in between them, since multiple copies may be delivered to the same link interconnecting them in the same time slot. In this sense, this scheme is compatible with point-to-point switches that employ input buffering [12, 13, 14]. This scheme is also compatible with those point-to-point output-buffered switches that employs internal speedup with the same or higher speedup factor. The overall multicast switch will then be output-buffered.

IV. Performance Study and Implications

This section considers the loss probability of packet copies in the shuffle-exchange copy network. Although an approximate analysis is possible, to limit scope, we present only simulation results here. Only the results of interval-splitting algorithm that assigns interval $(0, F-1)$ to all packets are presented.

In our simulations, the fanout F is distributed in a truncated-geometric fashion:

$$P\{F=k\} = \frac{(1-q)q^{k-1}}{1-q^N}, \qquad 1 \le k \le N \qquad (1)$$

The expected fanout is

$$\overline{F} = \frac{1}{1-q} - \frac{Nq^N}{1-q^N} \qquad (2)$$

For simulation, we fix $\overline{F}$ and derive q from it.

Figure 5 shows the simulation results for shuffle-exchange copy networks with $n = \log_2 N = 128$ and 256, and $\overline{F} = 2$ and 3. The confidence interval for the data point at $P_{loss} = 10^{-6}$ for each curve is roughly plus or minus 10^{-7}. The load ρ refers to the offered load at each *output*, and it is related to the input offered load ρ_i by $\rho = \overline{F}\rho_i$

Graphs (a) and (b) plot P_{loss} versus L for offered loads of 0.4 and 0.6. Fnor each curve, P_{loss} decreases with L up to a certain point, at which increasing L is not effective in reducing P_{loss} anymore. This is due to the overflow effect: the network can make at most a total of N packet copies in each time slot, and beyond that, having larger L does not help. We also note that P_{loss} is smaller when $N = 256$ than when $N = 128$. Larger N reduces the overflow probability by allowing more sharing of replication resources in the network.

Graph (c) shows the case where overflow events are eliminated in the simulations. That is, whenever the simulation program finds that there is a total of more than N copies being requested, the event is filtered out. This is an approximation to the situation where a reservation device (see preceding section) is used to deal with the overflow situation. We see that P_{loss} decreases roughly exponentially (the y-axis is on logarithmic scale) with L. This agrees with an approximate-analytical result not presented in this paper.

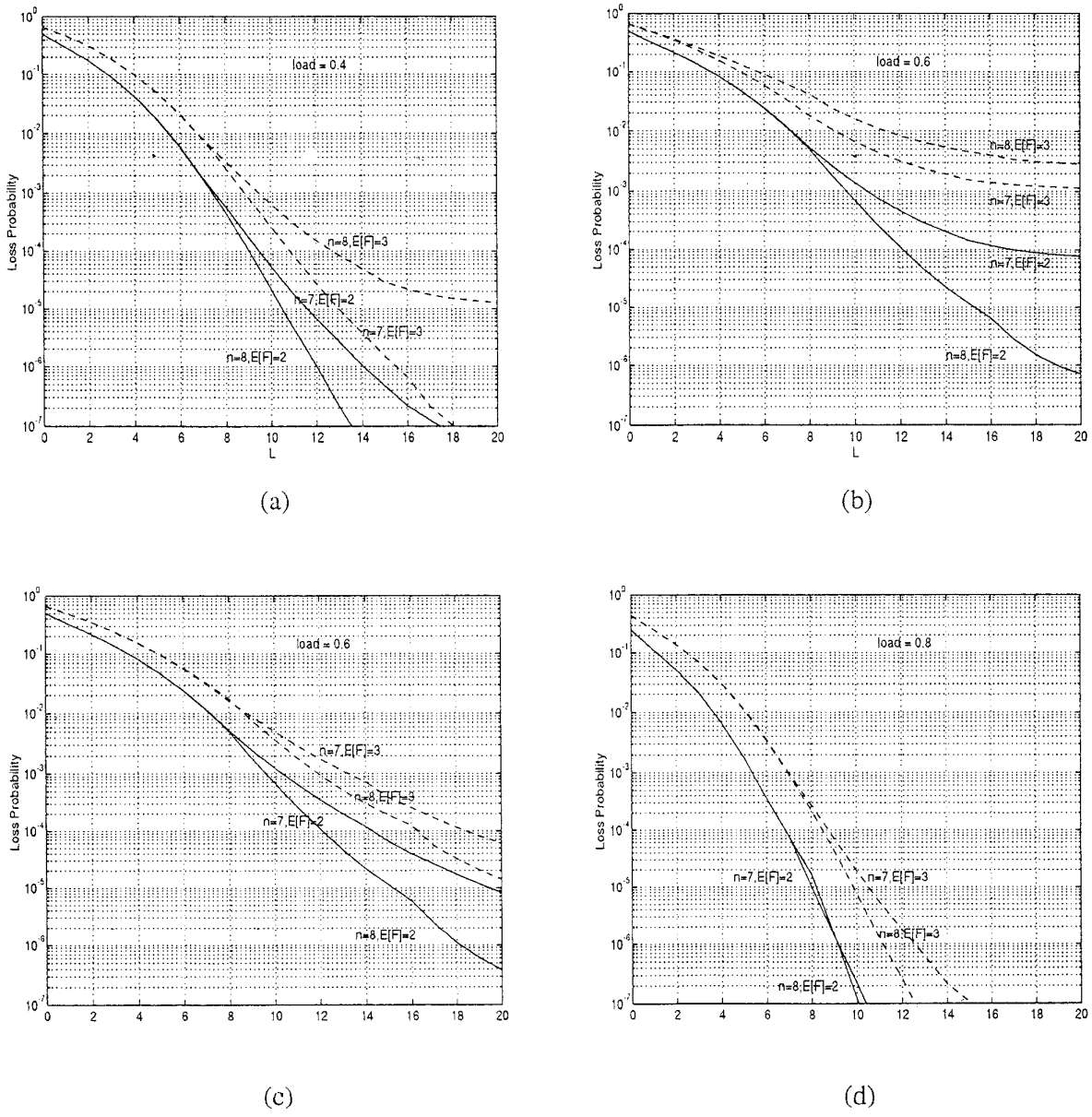

Fig. 5. Simulation results for shuffle-exchange copy networks: P_{loss} versus L with (a) $\rho = 0.4$; (b) $\rho = 0.6$; (c) overflow events eliminated and $\rho = 0.6$; (d) speedup factor of two and $\rho = 0.8$.

We also explored using speedup to solve the overflow problem. Graph (d) plots the results when the offered load is 0.8 and the speedup factor is 2. At the beginning of each time slot, each input with an active packet divides its requested copies into two batches, one for each internal time slot in the speedup network. One batch is of size $\lceil F/2 \rceil$ while the other if of size $\lfloor F/2 \rfloor$. Overflow is in principle still possible but is unlikely, as indicated by the curves. We note that for the 256×256 network, $L = 16 = 2n$ is sufficient. So, multiplying the number of switch nodes by the speedup factor, we get a network complexity of $2N \log_2 N$.

V. Conclusions

This paper has described a general packet replication scheme that can be applied to arbitrary network topologies. Philosophically, it is very similar to deflection routing [5, 6]. In deflection routing, when it is not possible to route a packet to the desired output at a node, it is routed (deflected) to a wrong output. Attempts are made later to correct for the deflection. The path traversed by a packet is therefore probabilistic. The essence of the general packet replication scheme is similar: when it is not possible to perform replication because of contention, the packet is forwarded to only one output link, and replication is deferred. It is this simplicity that makes the scheme implementable in arbitrary interconnection network topologies, including networks in which multicasting were thought to be inherently difficult. Replication hot spots are diffused by this scheme which automatically moves part of the replication efforts to less active network regions.

The use of the replication scheme in the shuffle-exchange network has been discussed in detail. It is found that the performance of the algorithm improves with the increase of network dimensions and that a modest speedup of two on the network operation is enough to make the packet-loss probability sufficiently small.

There is an issue that must be investigated further. This paper has mainly focused on the replication process. At the end of the process, the final destination of each packet copy must be derived. If the fanout is small, the destinations of all packet copies can be embedded in the header of the master packet [4]. Otherwise, the destinations can be stored in memories at the outputs of the copy network. Since packet copies of a master packet can emerge at arbitrary outputs, the destination information must be accessible at all outputs. Reference [11] describes several strategies for reducing the memory requirements with respect to the copy network in [4]. Some of the methods are applicable to the shuffle-exchange copy network in this paper.

Closed networks (e.g., the Manhattan-street network [6]) in which the sources and destinations of packets are nodes also presents a challenging problem with respect to the retrieval of destination information. To save memory in a large network, one may store the destination information of packet copies of a multicast connection in only some of the nodes. Upon the completion of its replication process, the packet is first routed to the nearest node containing its destination information before being routed to its actual destination. Two questions are how to distribute the storage of the destination information and what is the tradeoff between memory and routing requirements.

References

[1] J. Turner, "Design of a Broadband Packet Switching Network," *IEEE Trans. Commun.*, Vol. 36, No. 6, June 1988.

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

[3] A. Huang and S. Knauer, "Starlite: A Wideband Digital Switch," *Proc. IEEE Globecom '84* , pp. 121-125.

[4] T. T. Lee, "Non-blocking Copy Networks for Multicast Packet Switching," *IEEE J. Select. Areas Commun.*, Vol. 6, No. 9, Dec. 1988, pp. 1455–1467.

[5] S. C. Liew and T. T. Lee, "$N \log N$ Dual Shuffle-exchange Network with Error-correcting routing," *IEEE Trans. Commun.*, Vol. 42, No. 2/3/4, Feb./Mar./Apr. 1994, Part I of three parts, pp. 754–766.

[6] N. F. Maxemchuk, "Routing in the Manhattan Street Network," *IEEE Trans. Commun.*, Vol. 35, No. 5, May 1987, pp. 503–512.

[7] C. Partridge, *Gigabit Networking*, Addison Wesley, 1994, pp. 143–147.

[8] C. J. Chang and C. J. Ling, "Overflow Controller in Copy Network of Broadband Packet Switch," *Elect. Lett.*, Vol. 27, No. 11, May 1991, pp. 927–939.

[9] W. De Zhong, Y. Onozato, and J. Kaniyil, "A Copy Network with Shared Buffereds for Large-Scale Multicast ATM Switching," *IEEE/ACM Trans. Networking*, Vol. 1, No. 2, Apr. 1994, pp. 157–165.

[10] J. W. Byun and T. T. Lee, "The Design and Analysis of an ATM Multicast Switch with Adaptive Traffic Controller," to appear in IEEE/ACM Trans. Networking.

[11] J. S. Turner, "A Practical Version of Lee's Multicast Switch Architecture," *IEEE Trans. Commun.*, Vol. 41, No. 8, Aug 1993, pp. 1166–1169.

[12] M. J. Karol and M. G. Hluchyj, "Input versus Output Queueing on Space-Division Packet Switch" *IEEE Trans. Commun.*, Vol. 35, Dec. 1987, pp. 1587–1597.

[13] Y. N. J. Hui and E. Arthurs, "A Broadband Packet Switch for Integrated Transport," *IEEE J. Select. Areas Commun.*, Vol. 5, No. 8, Oct. 1987, pp. 1264–1273.

[14] T. T. Lee and S. C. Liew, "Broadband Packet Switches based on Dilated Interconnected Networks," *IEEE Trans. Commun.*, Vol. 42, No. 2/3/4, Feb./Mar./Apr. 1994, Part I of three parts, pp. 732–744.