Congestion Control Techniques in Asynchronous Wormhole Routing Networks*

Emilio Leonardi, Fabio Neri
Dipartimento di Elettronica – Politecnico di Torino
{leonardi,neri}@polito.it

Prasasth Palnati, Mario Gerla
Computer Science Department – UCLA,
405 Hilgard Ave, Los Angeles, CA 90024
{palnati,gerla}@cs.ucla.edu
Phone: (310) 825 1888 – Fax: (310) UCLACSD

Abstract

High-speed networks use lightweight protocols and a simple switch architecture for achieving higher speeds. A popular lightweight protocol for local and campus high speed data networks is wormhole routing. Namely, the head of a packet (worm), upon arriving at an intermediate node, is immediately forwarded to the next node on the path. Thus, the packet, like a worm, may stretch across several intermediate nodes and links. Wormhole routing networks provide low switch processing, as well as low latency. However, they are prone to congestion, thus requiring careful flow control. In this paper, we consider high-speed, asynchronous, unslotted wormhole routing networks. For such networks, we compare and contrast two different flow control mechanisms, namely, backpressure flow control and deflection routing (with local input control). With backpressure, in order to maintain deadlock-free routing, we assume either Up/Down routing, or shortest path routing with virtual channels. With deflection routing, to avoid livelocks, we perform worm alignment (delayed deflection) at switches. We show via simulation that the throughput performance of the two schemes is comparable (except for Up/Down routing). We then discuss the tradeoffs with respect to complexity of hardware, routing protocols and buffer requirements.

Keywords: wormhole routing, deflection routing, backpressure flow control, congestion control.

*This work was supported by the USDOD ARPA/CSTO under Contract DABT63-93-C-0055. The Distributed Supercomputer Supernet – A Multi Service Optical Intelligent Network.

1 Introduction

Wormhole routing has been traditionally the scheme of choice in low-latency, high-speed networks [6, 10]. In wormhole routing, the unit of information transfer is a worm. A worm can range in length from a few bytes to several thousand bytes. Each worm has a head, a tail and a data portion. Each intermediate node[1] forwards the worm along the desired output port (if available) as soon as the head of the worm is received, without waiting for the rest of the worm (as is done in conventional store-and-forward networks). Thus, the worm can stretch across several nodes and links at any one time.

When the desired output port is unavailable (blocking), a portion of the worm is stored in a buffer associated with the input port called slack buffer. Using backpressure flow control, the rest of the worm is stored in the slack buffers of the upstream nodes. In a variation called cut-through routing, upon blocking the entire worm is buffered at that node itself. Thus, for cut-through routing the slack buffer must be at least equal to the maximum worm size, while, for wormhole routing, the slack buffer can be much smaller.

The key feature of wormhole routing high-speed networks is that they can be operated in an asynchronous and unslotted fashion. An example of wormhole routing network is Myrinet [9]. Myrinet LANs use source routing and backpressure flow control to achieve data rates up to 640 Mbit/s.

Typically, wormhole routing networks are based on very simple network protocols and therefore are prone to congestion. To prevent congestion and deadlocks, some form of flow control must be introduced.

In this paper, we wish to compare the throughput/delay performance of two different flow control mechanisms for such wormhole routing high-speed networks, namely, backpressure flow control and deflection. Backpressure flow control is an explicit link-by-link flow control mechanism requiring bidirectional links. Deflection works as a flow control strategy by "diffusing" worms around a congested area. The schemes have different requirements with regard to buffering and complexity of the protocols and hardware. This makes it imperative to compare and contrast them in the context of high-speed networks where lightweight protocols and simple switching mechanisms along with low-latency are design goals.

The first deflection routing studies [7, 3, 8] were based on synchronous, slotted case. Later, slotted and unslotted deflection-routing for packets of fixed size were studied in [4]. Recently, deflection in asynchronous networks was addressed in [11, 12]. However, deflection routing in both unslotted and asynchronous networks with wormhole routing (variable size worms) has not been studied.

This paper is organized as follows. Section 2 presents an overview of backpressure flow control

[1]We use node and switch interchangeably throughout the paper.

and deflection routing and the requirements of each. The complexity of protocols and hardware as well as buffer requirements are discussed. Section 3 explains the simulation model. The results are presented in section 4. We conclude the paper with section 5.

2 Routing and Flow Control Mechanisms

With backpressure flow control, when a worm arrives at a node, it has a set of output ports through which it can reach the destination in the fewest hops. These output ports are called preferred output ports. The worm is never routed through an output port not in this set. If none of the preferred output ports is available, the worm is blocked.

By contrast, in deflection routing, when the worm arrives at an intermediate node and none of the preferred output ports is available, the worm is deflected to another port. To provide an output to each incoming worm, in deflection routing each node must have the same number of input ports as output ports. In this paper we consider only bidirectional topologies, thus we automatically meet this requirement. Note however that bidirectionality is not required for a network to support deflection routing.

In this section, we describe in some detail the requirements for backpressure flow control and deflection routing with special emphasis on buffer requirements, complexity of protocols and flow control procedures.

2.1 Backpressure Flow Control

Wormhole routing presents two problems. First, since the slack buffers associated with each input port are of fixed size (and typically less than the maximum worm size), worm loss due to buffer overflow is possible. Secondly, deadlocks are possible since worms are blocked when the preferred output ports are not available potentially creating circular waits.

The problem of buffer overflow is solved using the backpressure link-by-link flow control mechanism. For this purpose, two thresholds are defined for the slack buffer: the stop threshold (K_s) and the go threshold (K_g). When the buffer is filled up to K_s, a STOP signal is sent to the upstream node (using the bidirectionality of the link). Upon receiving the STOP signal, the upstream node stops transmitting the remainder of the worm. It is easy to see that the node that sent the STOP signal must have enough buffer space left beyond K_s to store bytes transmitted by the upstream node in two one-way propagation delay times on the link. When the preferred output port becomes available, the slack buffer is drained until K_g is reached at which time a GO signal is sent upstream. The upstream node resumes transmission of the worm upon receiving the GO signal. Here, again, the buffer capacity from the start of the buffer to K_g must be at least two one-way propagation delays (in bytes) in order to prevent gaps in the transmission of worms. Therefore, the slack buffer size must be at least 4 times the one-way propagation delay on the link (in bytes). The effectiveness of this flow control scheme has been demonstrated in [15]. We point out that backpressure flow control is an explicit global flow control mechanism.

It should also be noted from the preceding description that, for backpressure flow control, the size of the input slack buffers is independent of the worm length. However it is dependent upon the round-trip propagation delay associated with the links of the network (i.e., the slack buffer size is a function of the link length, or alternatively, of the size of the network).

Deadlocks in wormhole routing networks have been investigated earlier [5, 13]. Two solutions exist: Up/Down routing on a spanning tree as in Autonet [16, 17] and the use of Virtual Channels [5, 13]. We now discuss the advantages and disadvantages of each method.

Up/Down Routing

In the Up/Down routing strategy, one of the nodes is chosen at random as the root and all links of the topology are designated as 'up' or 'down' links with respect to this root. The 'up'/'down' state of a link is relative to a spanning tree computed in the background by a distributed algorithm. A link is 'up' if it points from a lower to a higher level node in the tree. Otherwise, it is 'down'. For same level nodes, node IDs break the tie. The routing from a source to a destination is done in such a fashion that zero or more 'up' links (towards the root) are traversed before zero or more 'down' links are traversed (away from the root) in order to reach the destination. This prevents circular waits and thus the routing is deadlock-free.

The advantage of this approach is that node hardware and software are simple and do not need to be changed. The drawbacks are that the selected paths are not always shortest paths (i.e., higher average hop distance) and links near the root get congested and become bottlenecks leading to lower throughput.

Virtual Channel Routing

In this approach, each link of the topology is split into a number of virtual channels each with its own slack buffer and STOP/GO backpressure protocol. The transmitter and the link capacity is shared among the virtual channels. The idea is to create layers of acyclic "virtual" networks with well-defined transitions from one layer to the next (from a lower numbered virtual channel to a higher numbered virtual channel, in such a manner that as the worm gets closer to the destination, it is using higher numbered virtual channel) such that circular waits can be avoided [5].

The advantage of the virtual channel approach is that it allows shortest path routing to be used for all source-destination pairs (leading to a lower average hop distance). Also, assuming uniform traffic distribution[2], links are uniformly loaded, and no bottlenecks are created because of routing restrictions. The drawback is that the complexity of the node is increased – each buffer must be replicated once per virtual channel, and also the flow control becomes more complex as the backpressure mechanism has to be used for each virtual channel. Another requirement, adding to the complexity of the node, is a protocol for serving virtual channels. One further disadvantage of this approach is that the virtual channel mappings are topology dependent (i.e., the number of virtual channels required to allow deadlock-free shortest path routing depends upon the topology and the routing function defined).

Several different protocols have been evaluated for serving virtual channels [13]. These protocols are required since the virtual channels associated with each link (or physical channel) share the same transmitter. The following protocols can be used to serve virtual channels.

a. **Fixed Assignment (FA)**: In this protocol, the virtual channels are each served in round-robin manner for a fixed duration of time called a slot (like TDM).

b. **Preemptive Priority (PP)**: In this protocol, priority is given to worms using higher numbered virtual channels since they are closer to the destination. The priority is given in a preemptive resume fashion. A worm in service is served until either the worm has been completely transmitted or it is blocked. In either case the next lower virtual channel with a worm is served. While the worm on a virtual channel is being served, if another worm arrives on a higher numbered virtual channel, then the newly arrived worm preempts the one being served.

c. **Non-preemptive Priority (NP)**: This protocol is similar to the Preemptive Priority protocol described above. The difference is that if a worm on a virtual channel is being served and a worm arrives at a higher numbered virtual channel, the new worm does not preempt the worm being served. When the worm being served has been completely transmitted or when it gets blocked, we pick the highest numbered virtual channel which has a worm waiting to be transmitted for service.

Of these three protocols, Non-preemptive Priority (NP) can be shown to provide the best results [13], thus this protocol will be assumed for virtual channel operation in this paper.

2.2 Deflection Routing

Deflection routing is possible on any regular topology (i.e., number of input ports equal to number of output ports). Deflection routing does not have the problem of worm loss due to buffer overflow, since there is always an output port available (not necessarily the preferred one) for the worm to be

[2] For this paper, we consider only the uniform traffic distribution model.

transmitted out from. Also, flow control in deflection routing is not by backpressure. Instead, it is achieved jointly by deflecting worms away from the congested area and by controlling the input of traffic from the hosts. We call the latter technique input control. Flow control in deflection routing happens by the implicit propagation of information about congestion in the network following a large number of deflections: deflections spread the congestion to the other nodes in the network until the hosts are throttled via input control.

Though deadlocks are not possible when deflection routing is used, livelocks are. A livelock occurs when worms are continuously deflected and will never reach the destination. Our simulations show this behavior (see Fig. 3 in Sect. 4). In order to overcome this problem, slotted networks use random choice of inputs at the nodes to probabilistically break livelocks. In the asynchronous, unslotted case, pure deflection is prone to livelocks and is also inefficient. The inefficiency is due to the fact that in pure deflection the worm is deflected even if the preferred output port would become available very shortly. In overload conditions, when a worm is routed, only one output is free, and most likely it is not a preferred output. Thus, worm alignment (or delayed deflection), as defined next, must be implemented.

Worm alignment (or delayed deflection) is carried out as follows. When the head of a worm arrives at a node and the preferred output port is not available, a calculation is made to see if the worm can be buffered in the input buffer for the period of time it takes for the preferred output port to become available (i.e., till the remainder of the worm using the output port is transmitted). If the buffer would overflow, then the worm is deflected through another output port. However, if the worm can be buffered without causing overflow, it is buffered (i.e, deflection is delayed). By the time the output port becomes available, other worms waiting for the same output port may have arrived. One of these worms is randomly chosen and the output port assigned to it. This worm alignment provides a mechanism by which the deterministic choice of purely asynchronous deflection is avoided. This eliminates livelocks and addresses the inefficiency of purely asynchronous deflection.

The buffer requirements for deflection routing are determined by the procedure for injection of worms into the network by the host connected to the node. When the host injected worm is being transmitted, it is possible that an incoming worm's preferred output ports are all in use. In order to prevent worm loss, this incoming worm can be either extracted by the host or it must be buffered. Extraction by the host is not always feasible and it adds complexity in the host. Thus, the preferred solution is to buffer the incoming worm. The simplest approach is to have each input buffer be the size of the largest possible worm. An alternative approach is to have a single buffer per node that can hold the largest possible worm. The incoming worm blocked due to host transmission is buffered in this special buffer. However, this approach makes buffer management by the node more complex. Regardless of the alternative, it is easy to see that the buffer associated with each input port is only a function of the worm size, while it is independent of the link length (or network size). This is in contrast with the buffer size requirements for backpressure flow control, which depend on link delay.

As far as complexity of the node architecture is concerned, deflection routing requires that the node be able to decide the routing on a per worm basis, and be able to run the different protocols required to achieve input control on a local basis.

3 Simulation Model

An evaluation of the routing and flow control schemes described in the previous section was carried out by simulation. The simulation program was written in Maisie [2], a C based, message passing environment.

A node in the network has a variable number of bidirectional ports. One or more ports (called host ports) are connected to hosts acting as sources and sinks of information. The remaining ports (transit ports) are connected to other nodes according to the topology specification. For the results presented in this paper, all links are assumed to be 1 Km in length, and the signal propagation velocity on the links is assumed to be equal to 2×10^8 m/s. The data rate on the links is equal to 640 Mbit/s; this means that the propagation delay on one link equals 400 byte times.

Inside a node, each bidirectional port can be viewed as two independent ports, one input and one output port. Some buffer space is associated with each input port. The size of the buffer for a host port is generally different from the buffer for a transit port. However, all host ports have the same buffer size, as do all transit ports.

For deflection routing, each transit port buffer is large enough to store one maximum-size worm. This prevents buffer overflow when a host worm is being transmitted; other, more efficient, ways to cope with this problem are possible, as previously discussed, but we took the most straightforward approach for simplicity.

For backpressure flow control, each transit buffer is large enough to store at least twice the link propagation delay (800 bytes in our case) to avoid overflow when a STOP signal is sent upstream. To avoid gaps inside worms after backpressure release, transit buffers should indeed be larger than four times the link propagation delay.

If virtual channels are used to avoid deadlocks, several identical transit buffers must be associated with each transit port. Transmission (reception) of transit worms can be active only from one output (input) buffer at a time.

Hosts are sources and destinations for worms transferred by the network. Worm generation follows a Poisson process of given rate. Worm sizes are random, and follow a truncated geometric distribution, with variable average and maximum value. Destinations are randomly chosen, according to source/destination probabilities. In this paper we only consider average worm sizes equal to 500 or 5000 bytes, and a maximum worm size of 10000 bytes. Destinations are uniformly distributed, i.e., a worm is equally likely to be sent from one node to all other nodes in the network. When a worm is generated by a host, it is queued to the host buffer for transmission. If the empty space in the host buffer is not enough to store the entire worm, the worm is lost. Host buffers are assumed to be large enough to store 100 average size worms. This model does not account for distances between hosts and nodes; if a link must be taken into account between the host and the switch, our model remains valid if a constant is added to worm delays, and backpressure flow control is implemented between the host and the switch.

Inside each node, the first bytes of an incoming worm are queued to either the host or the transit buffer. We assume the worm size and destination to be known to the node when the first byte of the worm arrives at the input port. If the incoming worm is the first in the input buffer, the node controller tries to route it to an output port, according to the routing policy, using routing tables provided at simulation initialization time. If a free output port is found, transmission starts. Otherwise transmission is delayed. Every time an output becomes free, all input ports are searched for worms awaiting transmission. Input ports are scanned in a random order, giving priority to transit ports over host ports (all transit ports are scanned first). If backpressure flow control is active, a STOP signal is sent from transit ports to the upstream node whenever the buffer occupancy exceeds K_s. Similarly, a GO signal is sent upstream whenever the buffer occupancy drops below K_g. In our simulations, K_s is set to twice the propagation delay (i.e., to 800 bytes) from the end of the transit buffer, while K_g is at most twice the propagation delay from the beginning of the transit buffer, but never farther than K_s.

If virtual channels are used, input buffers are scanned for selection of worms to transmit also when the transmission of a worm is interrupted by the reception of a STOP signal from the downstream node. In fact, the transmission of a worm belonging to a higher-order virtual channel can start at those time instants. This means that we adopted the non-preemptive protocol for serving virtual channels: a flowing worm is never interrupted by a worm arriving at the same input port.

Routing decisions at a node are based on the distance from the current node to the destination. For backpressure flow control, one of the preferred free output ports is selected at random (i.e., we have implemented an adaptive minimum distance routing, not pure source routing wherein the head of the worm has a fixed output port number for each intermediate node).

For deflection routing, a weight is associated with each output port according to the distance from the destination, minimum weight indicating a preferred port. For each incoming worm, all preferred output ports are scanned to find a free port. If none is found (and the worm should be deflected), deflection is delayed using the worm alignment protocol described earlier. Since the node knows the size of the worms, it also knows how much time is necessary for the transmission of the worm currently occupying the output link to terminate. If the free space in the input buffer is enough, the incoming worm is not deflected upon arrival, and the routing/deflection decision is delayed. Whenever an output becomes free, all input ports are randomly scanned (transit ports first) looking for worms awaiting transmission. If the worm at the head of an input buffer is not being transmitted, and the free output is at minimum weight, transmission starts. In any case, the input buffer overflow condition is re-checked whenever an output becomes free at all inputs, and those worms for which waiting is no longer possible are deflected. Deflection is done according to the weight associated with each output port (lower-weighted ports are considered first); the regularity of the topology guarantees that a free output can always be found.

Since we assume that hosts can receive only one worm at a time, deflection may occur also at the destination: if the output port associated with the destination host is busy, an arriving worm may be deflected away from the destination.

When a host starts transmission of a new worm, overflow of transit buffers must be avoided. Transmission of the host worm is thus enabled only when all transit worms have a free space at least equal to the size of the new worm. This means that the size of the input buffer is constrained by the maximum worm size. In addition to this constraint, some buffer space is needed at input ports when worm alignment is done. The amount of buffer used by transit worms (we call it deflection buffer) is in general less than the total transit buffer size. We therefore identify the "deflection buffer" (as an initial portion of the transit buffer), and decide if a worm must be deflected according to the free space in the deflection buffer. The deflection buffer size can vary from 0 (pure – immediate – deflection of incoming worms) to the transit buffer size.

Host worms are never deflected. According to the input control policy, they sit in the host buffer until a preferred output port is free and a transit worm is not waiting on it, before starting transmission. In addition to this, the transmission of a host worm does not start if the amount of free space in each transit buffer is less than the worm size (to avoid buffer overflow). Although this approach seems to be very restrictive with respect to the admittance of host worms into the network, it is not. Simulation results have shown that the host may still be able to overload the network, causing an increase of deflection and, as a consequence, degraded performance. We have therefore implemented a more restrictive input control policy, by enforcing the condition that the occupancy of all deflection buffers must be below a threshold (specified as a percentage) for the host worm to start transmission. We call this percentage Flow Control Coefficient (FCC); when FCC=1, this last restriction is not effective.

As already noted, the intersting behavior of deflection routing networks is that congestion situations are sooner or later propagated to all nodes, so that the admittance of new worms can be made adaptive to the load observed on the links. Thus, more sophisticated forms of input flow control based on purely local information can be implemented, albeit they will not be considered in this paper.

4 Simulation Results

Simulation results are presented in this section for two 64-node bidirectional topologies with connectivity equal to 4 (four bidirectional transit ports per node). The first topology is a perfect shuffle [1] with $k = 4$ columns and $p = 2$ bidirectional links from one node to nodes of the next column; the number of nodes is $kp^k = 64$. The average distance between nodes is 3.375 (note that we are considering the bidirectional version of the perfect shuffle [15]). The other topology is an 8×8 bidirectional Manhattan Street Network [3]. This is a square grid wrapped around the surface of a torus, where bidirectional circular paths (rings) connect nodes in the same column and in the same row. The average distance is 4.063.

The perfect shuffle is a better topology than the square grid as far as average distance, but the latter has the advantage of a more uniform loading of the links in the network (it is easy to see that links in a shuffle in uniform and symmetric traffic conditions are not equally loaded).

Note that for bidirectional networks the cost of a deflection is at most two hops (it is always possible for a deflected worm to return to the same node in two hops).

One host is assumed to be connected to each node of the network. As already mentioned, the traffic is uniform: all destinations are equally likely, and all sources generate the same amount of information.

For the virtual channel approach, it is possible to show [13] that the bidirectional shuffle requires 4 virtual channels, while the square grid requires 3 virtual channels.

The following performance measures are reported as a function of the normalized host input load (a host load equal to 1 equals 640 Mbit/s).

Host throughput – This is the average traffic received by each host. The difference between input and output load is due to worms that are lost because of host buffer overflow. The host throughput is a clear measure of the network capacity.

Transit delay – This is the time from the transmission of the first byte of the worm until the last byte is received by the destination host. We report transit delays, disregarding delays in the host buffer, because the former are more representative of network behavior. The delays in the host buffer can be minimized by proper buffer management policies (e.g., by using priorities for delay sensitive traffic, etc.).

Hop count – For deflection routing and flow control, this quantity gives an idea of the amount of deflection faced by worms. We show average and maximum hop count for received worms.

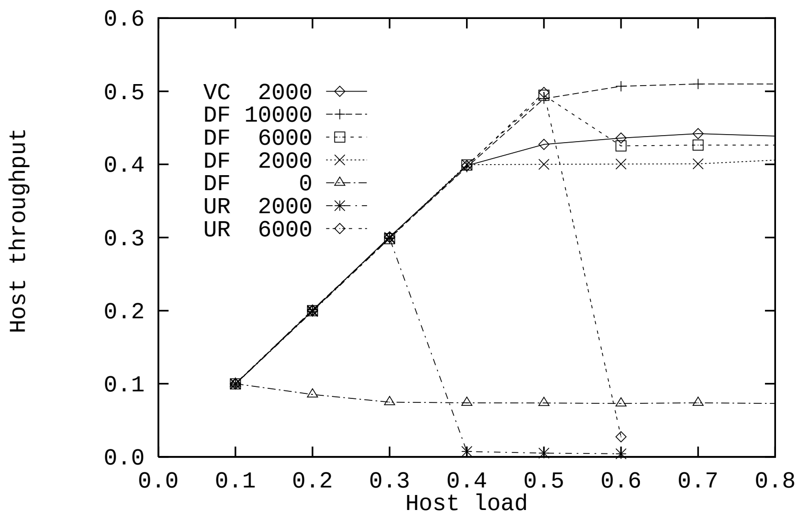

Figure 1: Host throughput for an 8×8 bidirectional grid. Average worm size is 500; maximum worm size is 10000.

Although the maximum value observed in simulation is strongly dependent on the length of the simulation, it gives an approximate idea of the worst case behavior.

The first network configuration is the 8×8 grid with an average worm size equal to 500. This means that worms are short with respect to the link size (which is 400 bytes: one average-size worm spans little more than one link), or, equivalently, that the network is large.

Fig. 1 shows host throughputs for this network configuration. Seven curves are shown, for different flow control techniques, and different transit buffer sizes. One curve (labeled VC) refers to the virtual channel approach, with transit buffers of 2000 bytes (slightly more than four times the link propagation delay) on each virtual channel. Three virtual channels are needed, thus the total buffer requirement on each transit port is 6000 bytes. Two other curves (labeled UR) refer again to backpressure flow control, but with unrestricted shortest path routing, i.e., without virtual channels or Up/Down routing. In this situation deadlocks occur, as shown by our results. We considered transit buffers of size 2000 bytes (equal to the size of one virtual channel buffer) and 6000 (equal to the overall buffering capacity on the port with virtual channels) bytes. The remaining 4 curves (labeled DF) refer to deflection routing, with deflection buffer size equal to 0 (pure asynchronous deflection), 2000 (equal to one virtual channel buffer), 6000 (equal to the total buffer per port with virtual channels), and 10000 (equal to the transit buffer size, that we took equal to the maximum worm size), respectively.

As expected, UR shows deadlock problems, since the throughput plummets to 0 if the input traffic becomes large and deadlocked worms block the network. We will not show any further result

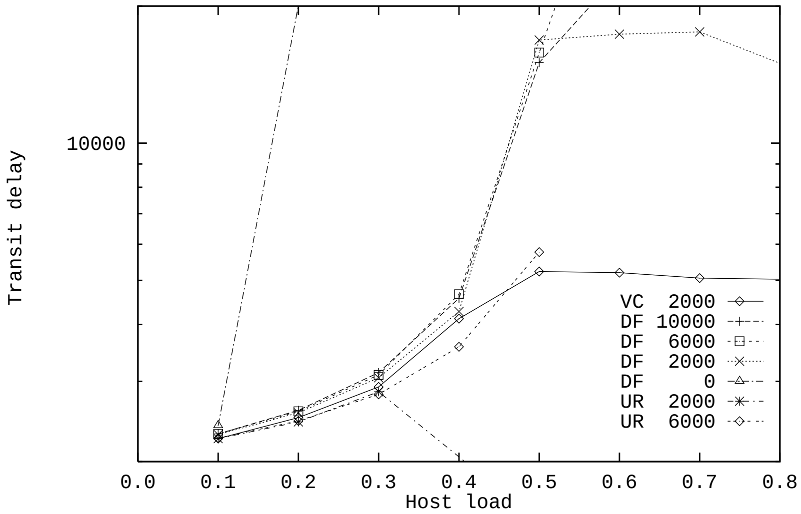

Figure 2: Average transit delay for an 8×8 bidirectional grid. Average worm size is 500; maximum worm size is 10000.

for the UR case.

Pure deflection (0 deflection buffer) also shows poor performances. The network capacity remains below 0.1.

Delayed deflection and virtual channels show comparable results, depending on the buffer size. Note the behavior of DF 6000, where the throughput decreases after reaching its maximum. This behavior was already observed by other researchers [11, 4], and is due to an increase in congestion (larger number of deflections, hence larger path lengths) in overload conditions because of a mild input control policy at the sources. By being more restrictive in the admittance of host worms, the throughput decrease would disappear. Simulation results not reported here have shown that, by using FCC=0.5[3], the DF 6000 curve has the horizontal asymptote at around 0.47, and the throughput peak disappears.

Fig. 2 shows transit delays for the same scenario. Note that, as expected, transit delays are much larger for deflection routing than for backpressure flow control with virtual channels. This is one of the major drawbacks of the deflection approach, as we shall discuss further below.

Fig. 3 shows throughput results similar to those presented in Fig. 1, but for an average worm size equal to 5000 bytes. In this case worms are longer (or, equivalently, the network is smaller), and can stretch through several nodes. This is the typical scenario where wormhole routing is currently used [13]. The two curves for backpressure with unrestricted routing are replaced by two curves

[3]All the results presented in the paper were obtained with FCC=1.

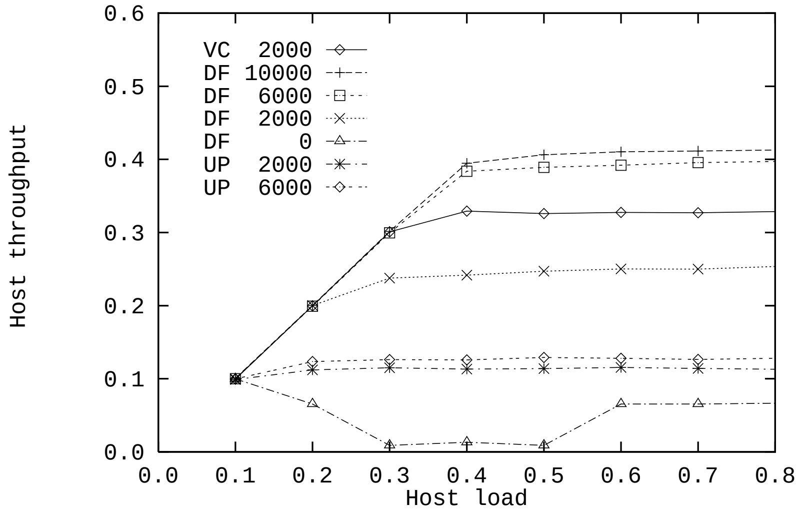

Figure 3: Host throughput for an 8×8 bidirectional grid. Average worm size is 5000; maximum worm size is 10000.

(labeled UP) for backpressure with Up/Down routing.

Note that Up/Down routing provides poor results: the network capacity is slightly above 0.1. Although this poor performance was not observed for smaller networks (see [15]), it becomes evident for the 64-node topologies considered in this paper, and it may be even more pronounced in larger topologies.

The purely asynchronous deflection (DF 0) exhibits livelocks in this network configuration: for input loads equal to 0.4 and 0.5, the throughput drops to zero due to worms spinning in paths not containing the destination (values of the hop count as large as several tens of thousands were observed for transit worms). In contrast, virtual channels and deflection perform very well, and-provide comparable results. If the same overall buffer space is associated with input ports (6000 bytes), deflection provides slightly higher throughputs than virtual channels. Note that, as already remarked, the virtual channel approach entails a higher complexity with respect to deflection (whose complexity is comparable to the one required by the Up/Down approach).

With respect to an average worm size equal to 500, the 5000 case provides lower network capacities (the best throughputs are around 0.4 as against 0.5). This is mainly due to the wormhole approach: long worms use a large amount of network resources (links and switches), and prevent the transmission of other worms. Note that one might have expected a large network (or large propagation delays) to degrade performances, specially for deflection routing, since the cost of deflections in terms of delays is increased.

Figs. 4 and 5 show average transit delays, and average and maximum hop counts for deflection routing, in the scenario of Fig. 3. Transit delays are again very large for Up/Down routing, due to

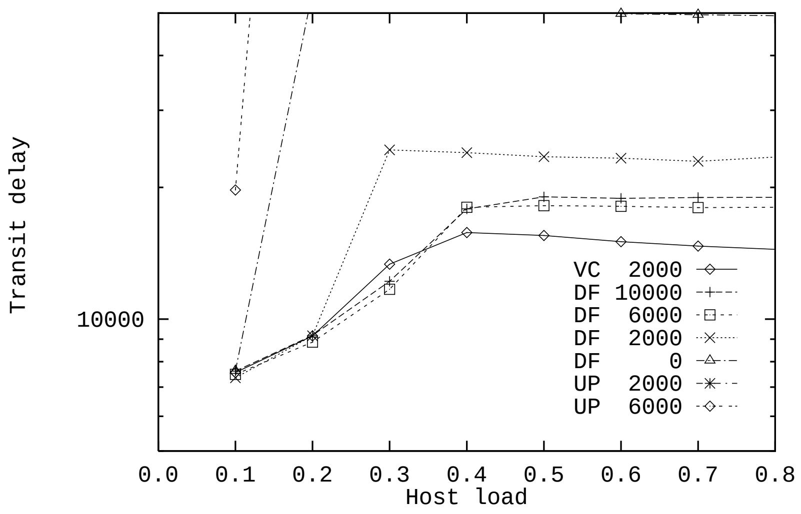

Figure 4: Average transit delay for an 8×8 bidirectional grid. Average worm size is 5000; maximum worm size is 10000.

congestion around the root of the spanning tree. Deflection and virtual channels show comparable average delays, although they are slightly smaller for backpressure in overload, as they are for backpressure in underload. More differences between these two flow control techniques would be observed for maximum delays, or for the delay distribution. The reason for the largest delays is that deflection can cause some unlucky worm to take a much longer trip than needed.

The curves in Fig. 5 show this behavior. The average distance for shortest path routing is slightly above 4, while the average hop count in overload is around 6 or 7, and the maximum hop count ranges from 30 to 100, depending on buffer size. Some considerations are in order here. First, the pure minimum distance routing can lead to an uneven load over the network links, causing throughput degradation; a limited amount of deflections may alleviate this problem. Second, the cost of deflections depends on the link propagation delay: if links are short, a large hop count does not mean large delays. Finally, with backpressure, a small hop count does not mean small delays, because worms can be frozen several times along their shortest route.

Fig. 6 shows throughputs for a 64-node perfect shuffle. Comments similar to those for Fig. 3 apply here. Backpressure with Up/Down routing yields low throughput because of congestion around the root of the spanning tree. Little improvement is gained by increasing the buffer size. Backpressure flow control can compete with deflection only by using virtual channels (thus requiring higher complexity). The curve for DF 2000 shows remarkable congestion problems in overload: again, these can be avoided by using a more restrictive input control policy. The asynchronous deflection DF 0 exhibits the lowest performance, as previously observed.

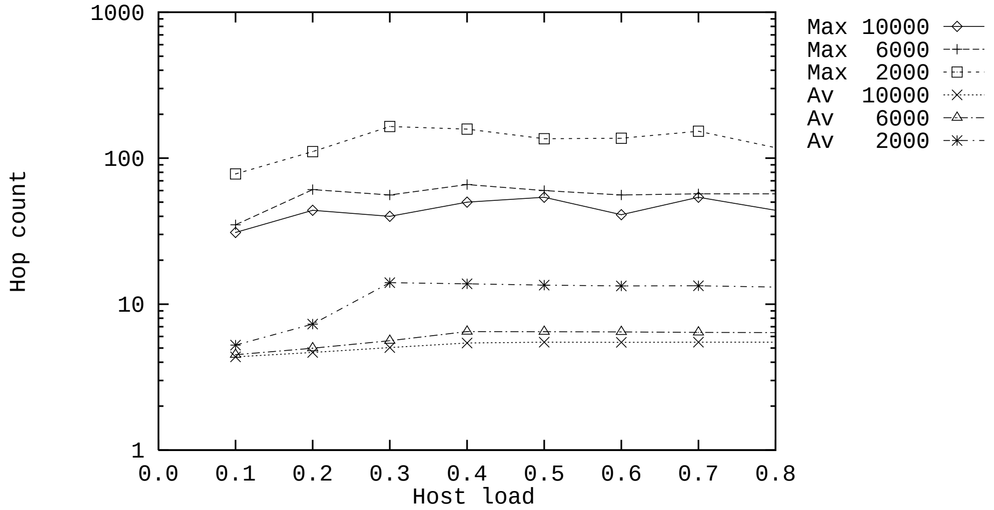

Figure 5: Average and maximum hop count for an 8×8 bidirectional grid with deflection routing. Average worm size is 5000; maximum worm size is 10000.

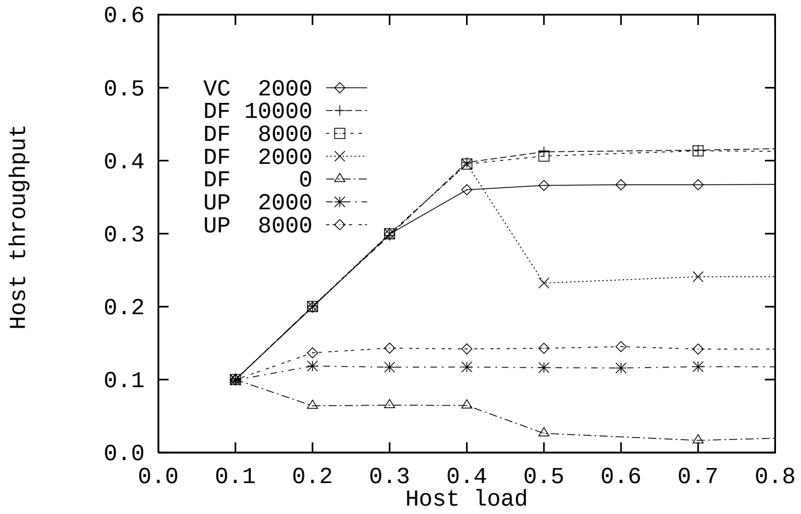

Figure 6: Host throughput for a 64-node bidirectional perfect shuffle. Average worm size is 5000; maximum worm size is 10000.

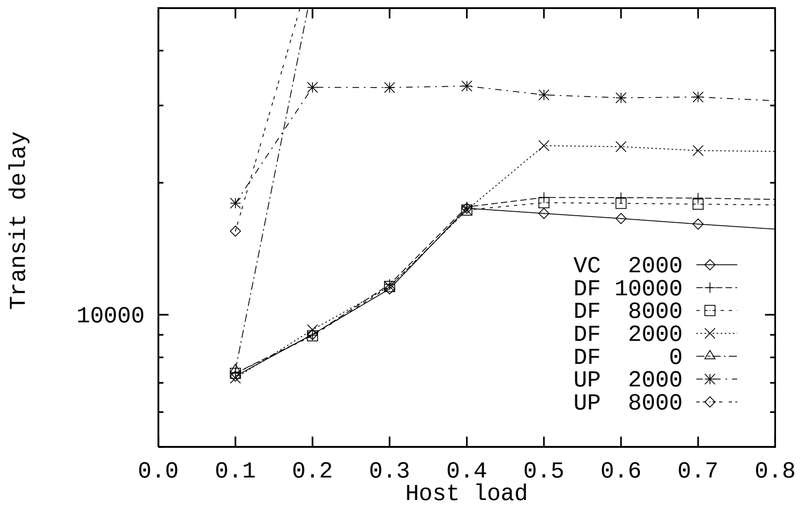

Figure 7: Average transit delay for a 64-node bidirectional perfect shuffle. Average worm size is 5000; maximum worm size is 10000.

By comparing Fig. 3 and Fig. 6, note that throughput for the shuffle and the grid is almost the same. The difference between the two topologies is not evident for this network size.

Figs. 7 and 8 show average transit delays, and average and maximum hop counts, for the perfect shuffle. The behavior is very similar to that observed in the square grid. The maximum hop count remains very large. A significant reduction of this parameter can be obtained by giving a priority in deflection to worms with a large hop count.

The results presented so far have shown that backpressure flow control can provide throughput results similar to deflection only when virtual channels are used. Throughputs (and delays) were also shown to be rather sensitive to the size of transit buffers. The next set of results evaluates the effect of different transit buffer sizes for deflection and backpressure with virtual channels. We only consider the 64-node perfect shuffle in this study.

Fig. 9 shows host throughputs in the virtual channel case. As expected, throughputs consistently grow as transit buffers become larger. Fig. 10 shows transit delays. We could expect to observe an undesirable increase of delays to compensate for larger throughputs, but this is not the case. For VC 1000, the slack buffer is smaller than four times the link propagation delay; gaps are present inside worms, and performance is degraded. Performance degradation is also observed when the buffer size is 2000. This is due to the fact that when backpressure flow control is used, wormhole routing performs better with cut-through than with the wormhole form of switching: blocked worms spanning several links and switches lock network resources that cannot be allocated to other worms. This problem disappears if the blocked worm is stored in a buffer inside the blocking switch. Indeed,

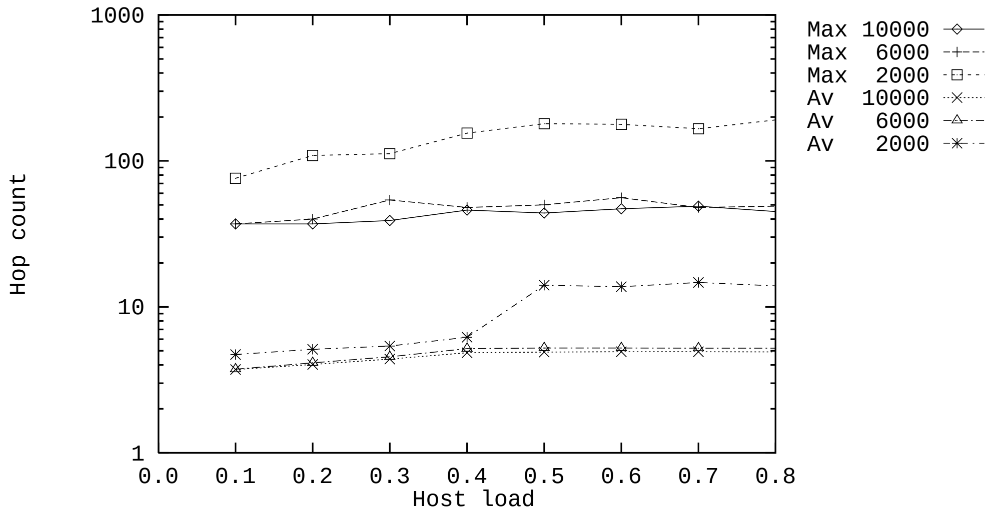

Figure 8: Average and maximum hop count for a 64-node bidirectional perfect shuffle with deflection routing. Average worm size is 5000; maximum worm size is 10000.

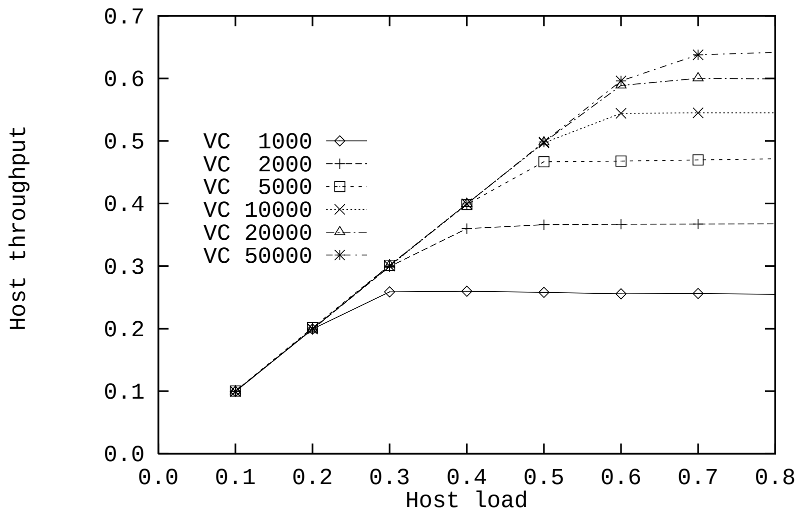

Figure 9: Host throughput for a 64-node bidirectional perfect shuffle with virtual channels and different buffer sizes. Average worm size is 5000; maximum worm size is 10000.

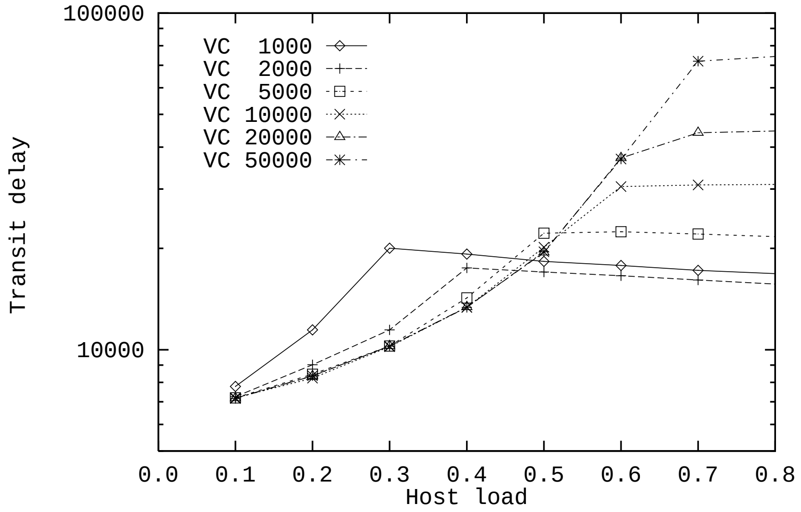

Figure 10: Average transit delay for a 64-node bidirectional perfect shuffle with virtual channels and different buffer sizes. Average worm size is 5000; maximum worm size is 10000.

we observe the best results when the transit buffer is more or less as large as an average size worm. With larger buffers, we have some throughput gain, but transit delays grow very large (note the logarithmic scale for delays), and this is against the goals of the networks we are considering in this paper.

Note that network links can be viewed as "extensions" of input buffers. This means that performance is best when the propagation delay on the link plus the input buffer size are in the order of the average worm size. This brings additional arguments in the explanation of the behaviors observed for 500-byte worms with respect to 5000-byte worms.

Similar considerations apply, for different reasons, to deflection routing. Larger deflection buffers permit more synchronization among worms, thus leading to performance that is closer to slotted deflection. When the deflection buffer becomes larger than the average worm size, transit delays grow unacceptably large. Figs. 11, 12, and 13, show, respectively, host throughputs, transit delays, and average hop counts. Note that, while the deflection buffer can be increased up to 10000 bytes without changing the transit buffer, the latter must be also changed when a deflection buffer equal to 20000 and 50000 bytes is used.

The best throughput (for very large buffers) is achieved by the virtual channel scheme. This is most likely due to the partitioning of the input buffer space, which leads to a reduction of the head-of-the-line blocking phenomenon typical of the input buffering approach. Note also that the hop count (hence the deflection probability) is not reduced by much even by increasing buffer size.

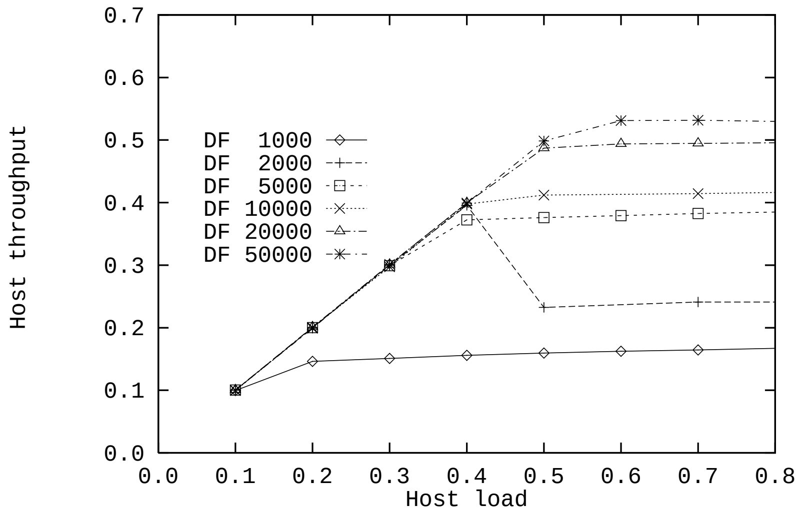

Figure 11: Host throughput for a 64-node bidirectional square grid with deflection routing and different buffer sizes. Average worm size is 5000; maximum worm size is 10000.

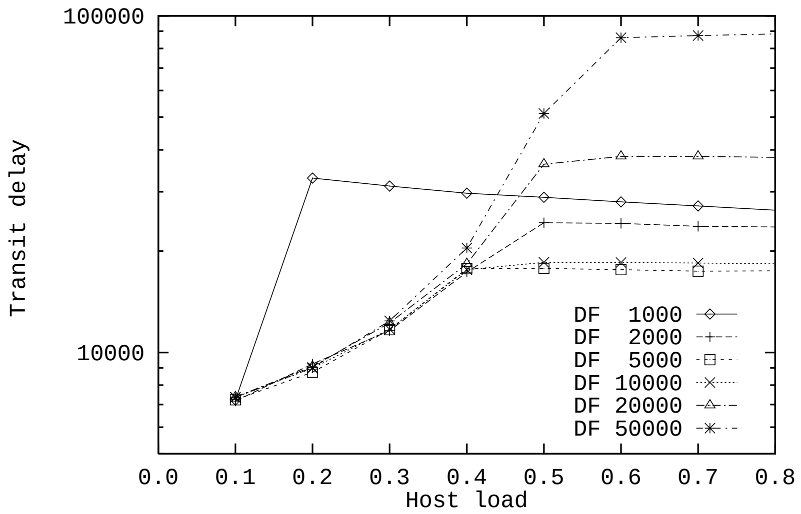

Figure 12: Average transit delay for a 64-node bidirectional square grid with deflection routing and different buffer sizes. Average worm size is 5000; maximum worm size is 10000.

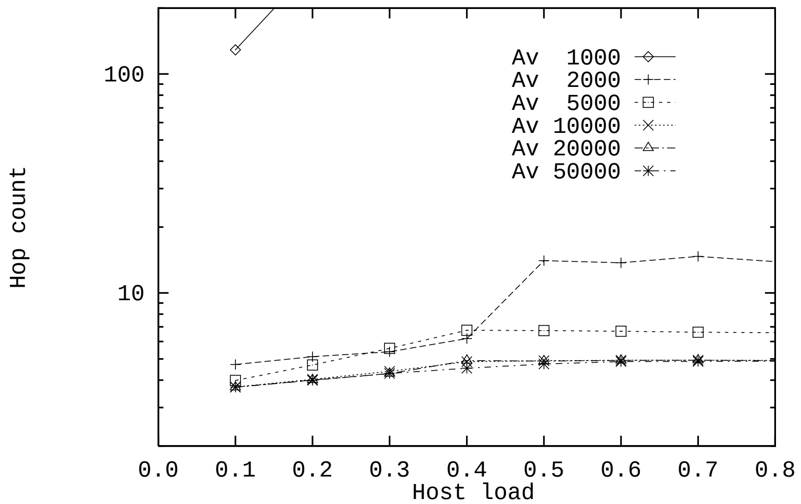

Figure 13: Average hop count for a 64-node bidirectional square grid with deflection routing and different buffer sizes. Average worm size is 5000; maximum worm size is 10000.

5 Conclusions

We have considered asynchronous, unslotted networks for high-speed, low-latency transfer of variable size data units called worms. Backpressure flow control and deflection routing were considered for congestion control, and evaluated and compared by simulation. The implementation complexity of the two schemes was shown to be comparable.

Backpressure has deadlock problems, which can be tackled by restricting the routing, or by using the virtual channel technique (at the cost of higher complexity). The Up/Down form of restricted routing has shown poor performances, mainly due to congestion on the most used network links near the root of the spanning tree. All backpressure schemes have memory (and also flow control for virtual channels) requirements depend on network size.

Deflection routing suffers of the livelock problem and exhibits poor performance if the deflection process is completely asynchronous. A simple form of synchronization (i.e., worm alignment) using worm input buffering was proposed. The synchronization solves both problems: livelocks are avoided because several worms can be routed together and random choices avoid deterministic circular paths, and performance is greately improved due to the reduced deflection probability.

The memory requirements of deflection routing are related to the size of the data units, but are independent of network size. The main drawbacks of the deflection approach are out-of-sequence delivery of data units and unbounded delay. A compression of the tail of the delay distribution can be obtained by making deflection adaptive to the age (i.e., hop count) of the worms. This extension was not explored in this paper.

The major advantages of deflection routing are implementation simplicity, robustness against failures, less restrictions on topology (bidirectionality not required), and ability to implement more sophisticated flow control schemes based upon observed link load.

References

[1] A. S. Acampora. A Multichannel Multihop Local Lightwave Network. In *Proceedings of GLOBECOM 87*, pages 1459–1467, November 1987.

[2] R. Bagrodia, K.M. Chandy, and J. Misra. A Message-Based Approach to Discrete-Event Simulation. *IEEE Transactions on Software Engineering*, 13(6), June 1987.

[3] F. Borgonovo and E. Cadorin. Locally-Optimal Deflection Routing in the Bidirectional Manhattan Network. In *Proceedings of INFOCOM 90*, pages 458–464, June 1990.

[4] F. Borgonovo, L. Fratta, and J. Bannister. On the Design of Optical Deflection-Routing Networks. In *Proceedings of INFOCOM 94*, pages 120–129, June 1994.

[5] W. J. Dally and C. L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[6] P. Kermani and L. Kleinrock. Virtual Cut-through: A New Computer Communication Switching Technique. *Computer Networks*, 3(3):267–286, September 1979.

[7] N. F. Maxemchuk. Routing in the Manhattan Street Network. *IEEE Transactions on Comunications*, COM-35(5):503–512, May 1987.

[8] N. F. Maxemchuk. Comparison of Deflection and Store-and-Forward Techniques in the Manhattan Street and Shuffle-Exchange Networks. In *Proceedings of INFOCOM 89*, volume 3, pages 800–809, April 1989.

[9] Myricom, Inc. *Myrinet in Brief*, 1993.

[10] L. M. Ni and P. K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, 26(2):62–76, February 1993.

[11] C. Ongaro. Ottimizzazione dell'instradamento a deflessione per reti M-MAN in modalita' unslotted. Technical Report RI 93017, CEFRIEL, March 1993.

[12] C. Ongaro. Problematiche di ottimizzazione dell'instradamento in reti M-MAN. Technical Report RI 93050, CEFRIEL, June 1993.

[13] P. Palnati, M. Gerla, and E. Leonardi. Deadlock-free Routing in a Hierarchical Supercomputer Interconnection Network. Submitted for publication.

[14] P. Palnati, E. Leonardi, and M. Gerla. Performance Analysis of Bidirectional Shufflenet: A Multihop Topology for Backpressure Flow Control. Submitted for publication.

[15] P. Palnati, E. Leonardi, and M. Gerla. Bidirectional Shufflenet: A Multihop Topology for Backpressure Flow Control. September 1995. To appear in *Proceedings of* 4^{th} *International Conference on Computer Communications and Networks.*

[16] T. L. Rodeheffer. Experience with Autonet. *Computer Networks and ISDN Systems*, 25(6):623–629, January 1993.

[17] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, T. L. Rodeheffer, E. H. Satterthwaite, and C. P. Thacker. Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links. *IEEE Journal on Selected Areas in Communications*, 9(8):1318–1335, October 1991.