Distributed Resolution of Network Congestion and Potential Deadlock Using Reservation-based Scheduling

Yong Ho Song and Timothy Mark Pinkston*
Hanyang University University of Southern California

Abstract

Efficient and reliable communication is essential for achieving high performance in a networked computing environment. Finite network resources bring about unavoidable competition among in-flight network packets, resulting in network congestion and, possibly, deadlock. Many techniques have been proposed to improve network performance by efficiently handling network congestion and potential deadlock. However, none of them provide an efficient way of accelerating the movement of network packets in congestion toward their destinations. In this paper, we propose a new mechanism for detecting and resolving network congestion and potential deadlocks. The proposed mechanism is based on efficiently tracking paths of congestion and increasing the scheduling priority of packets along those paths. This acts to throttle other packets trying to enter those congested regions—in effect, locking out packets from congested regions until congestion has had the opportunity to disperse. Simulation results show that the proposed technique effectively disperses network congestion and is also applicable in helping to resolve potential deadlock.

Keywords: Interconnection Networks, Congestion, Deadlock, Router Scheduling, Router Architecture

1 Introduction

The interconnection network is the communication backbone for both tightly coupled multiprocessor servers and loosely coupled (and, oftentimes, heterogeneous) distributed network-based multicomputer clusters. The performance of the interconnection network—measured, in part, by packet delivery time from source to destination (i.e., latency) and by the number of packets delivered per unit time (i.e., throughput)—has a substantial impact on overall system performance. The finiteness of network resources inevitably brings about contention on network resources that may delay or prevent packet transmission in the network. Such contention causes packets to block which, eventually, can lead to network congestion and, possibly, deadlock. Deadlocks reduce communication efficiency and reliability, consequently degrading network and system performance consider-

*This work is supported in part by the research fund of Hanyang University HY-2003-T and in part by NSF grant CCR-0311742.

1Deadlocks occur as a result of cyclical hold-and-wait dependencies on network resources by in-flight messages (or packets) that block packets from making progress indefinitely.
ably. Thus, it is vitally important to guard against congestion and potential deadlock in such a way as not to impose overly restrictive measures that under-utilize network resources.

Much research has been conducted in the past toward increasing the performance of interconnection networks by addressing the issues of network congestion and deadlock. A common technique for handling network congestion is to limit packet population in the network by throttling the injection of new packets locally when the network is suspected of being congested [3, 15, 26, 32]. Another common technique is to allow packets to route adaptively around congested areas of a network [5, 36] and/or to send special control packets in advance of data packets to reserve available (non-congested) network resources [24]. However, none of these techniques incorporate measures to disperse congestion directly—that is, to aid packets already within congested areas. Only a few techniques are designed to react to congestion more directly. They facilitate the movement of congested packets by assigning scheduling priorities based on packet lifetime [18] or force congested packets at the head of channel queues to bypass occupied resources via misrouting, giving packets blocked behind them the possibility of making forward progress [12, 34]. These techniques can be effective in handling certain forms of network congestion, but they have no explicit mechanism for handling cyclic congestion configurations, which would be useful in the context of deadlock-recovery routing.

Deadlocks are commonly handled either by avoiding them via the use of a deadlock-free routing algorithm or by recovering from them via detecting and resolving potential deadlock configurations. Avoidance-based deadlock handling techniques prevent cyclic dependencies on all the network resources from occurring by forcing packets to use a subset of the resources or even all the resources in a partially or totally ordered fashion [5, 28, 29, 10]. These techniques trade off routing flexibility for deadlock freedom. On the other hand, recovery-based techniques allow potential deadlock configurations to form but be resolved once detected [6, 11, 22, 27, 36, 37]. They should have an efficient mechanism to detect the existence of deadlocks in order to avoid unnecessary performance degradation [17, 16]. Considering that deadlocks rarely occur for typical traffic loads and network parameters [38, 40], recovery-based techniques allow for better utilization of network resources. For either deadlock handling approach, however, no way of accelerating the movement of packets already involved in congestion or deadlock is provided. Avoidance-based techniques such as Duato’s adaptive routing [5] only provide a way for packets to evade congestion already formed along some links toward their destinations. Recovery-based techniques such as Disha [36, 37] resolve deadlocks by rescuing one of the packets in a detected potential deadlock, but the remaining packets are still susceptible to deadlocking again before making any progress. In both schemes, escape resources ultimately have the potential to become the main performance bottleneck.

What is desirable is the development of mechanisms that address the problem directly and more comprehensively, starting at its source: namely, blocked packets. Under certain circumstances of network loading, traffic bursts, and pathological communication behavior, blockage on a single resource can easily grow into paths of blockage dependencies, cyclic blocking, and even knotted cyclic blocking across many network resources by packets [38, 39, 40]. To increase performance, it is important to detect such blocking behavior and quickly disperse it, allowing packets involved in congestion and deadlock-precipitating situations to accelerate their
movement out of those regions, particularly when deadlock is handled based on recovering from it. Dispersing congestion is possible because real traffic typically is not uniform; some areas in the network are more heavily loaded (i.e., regions where hot-spots form) while other neighboring areas have available resources to route packets.

In this paper, we propose a new mechanism for detecting and dispersing acyclic as well as cyclic blocking dependencies on network resources. It can be used as a means of more rapidly advancing packets directly involved in network congestion as well as a means of helping to detect and recover from potential deadlock situations. The proposed mechanism is generally applicable to all store-and-forward and virtual cut-through networks, independent of network topology. Furthermore, it can be used in combination with other packet scheduling and/or routing techniques. In essence, the mechanism detects the configuration of network congestion by propagating a special “pinging” control packet over suspected resources. Congested resources over which such pinging control packets traverse are marked as reserved, resulting in chained reservations of congested network resources. Once a reserved resource becomes available to route a congested packet, all packets along the associated resource dependency chain (or cycle, if one exists) are shifted by at least one buffer position toward their destinations. Also, other packets are prevented from entering the congested region during the time this shift operation occurs. This acts to throttle other packets trying to enter congestion paths—in effect, locking out packets until congestion disperses (similar to the Rotary Rule of the Alpha 21364 [20]). The way special control packets are used in this work is different from that in Peh’s flit reservation technique [24]. Here, pinging control packets traverse congested resources, not available resources; and they reserve resources along paths intended for dispersing network congestion, not for routing around it.

After the shift, an empty buffer space or bubble that could hold at least one packet is guaranteed to be available at the terminus of an acyclic congestion chain or at the node which started the shift within a cyclic congestion path. That bubble can then be used again in subsequent shifting operations, propagating backward along the path with preference given to congested packets, allowing them to make forward progress. Even though shifting operations alone do not immediately remove packets out of congested areas, they effectively decrease the coupling among packets involved in dependency relations on congested resources. In effect, they allow packets the opportunity to disperse out of congested areas after each “shift” operation. This reduces the probability of recurrence of the dependency relation that causes congestion to persist since new routing candidates could arise as a result of the shift. That is, some congested packets being routed could be presented with the choice of alternative (non-congested) resources or even arrive at their destination after the shift.

The remainder of this paper is structured as follows. Section 2 describes previous work on techniques that deal with network congestion and deadlock. Section 3 presents our proposed mechanism for the detection and resolution of chained and cyclic congestion, and shows how it aids potential deadlock resolution. Section 4 evaluates the performance of the mechanism, and, finally, Section 5 draws some conclusions.
2 Problem Background, Motivation, and Related Work

A high-performance network allows the maximum number of packets to make forward progress toward their destinations in minimal time usually along shortest paths. Consider for the moment the forward movement of packets in the network. Each time a packet moves forward, an empty buffer in a queue associated with a router port within the network or a network interface port at the network endpoint is consumed by the head of the packet. Likewise, an occupied buffer is released by the tail of the packet. Assuming that the unit of an empty buffer space for a packet is defined as a bubble, each forward movement of a packet in one direction is equivalent to the backward propagation of a bubble in the opposite direction. Thus, the movement of packets in a network can be characterized simply by considering the availability of bubbles in resources needed by packets and how those bubbles flow through those network resources: the more bubbles flowing into congested network resources, the greater the number of packets that make forward progress.

Previously proposed deadlock handling techniques in some way affect the availability and/or flow of bubbles in a network to increase performance. Store-and-forward and virtual cut-through packet switching defines the granularity of bubbles to be the size of a packet. Virtual channel flow control confines bubbles to flow within logical networks so that bubble movement within different logical networks can be independent of one another. Likewise, virtual output queuing confines bubbles to flow within separate network dimensions. Deadlock-free routing techniques based on avoiding deadlock in the strict sense or in the wide sense restrict bubble flow such that all bubbles supplied by network endpoints always flow through the entire set or some defined subset of network resources, respectively, in some total or partial order [5, 6, 29]. Techniques based on deflective re-routing of packets ensure that there is at least one packet in each cycle that can be supplied a bubble, even if non-minimal paths must be used in reaching the destination [7, 8, 12]. Alternatively, recovery-based deadlock handling techniques always make some bubble(s) available within any set of network resources on which a path of potential cyclic dependencies is detected as forming [11, 22, 27, 36, 37].

Increasing the routing freedom relaxes restrictions on packet movement, which allows bubbles to flow more freely among network resources, but some restrictions must still be enforced to ensure deadlock freedom, at least in the weak sense, e.g., as in progressive deadlock recovery. The challenging problem is how to apply the fewest restrictions on bubble flow to maintain deadlock freedom while selectively controlling the orientation of bubbles such that they migrate to specific areas of the network when such action is needed. Previous research has mostly addressed only the first part of this problem—increasing adaptivity. Only recently has emphasis on the second part come to the forefront, and that mainly on the issue of more effective load-balancing techniques [1, 31, 34]. Beyond these, a more comprehensive solution would support, when the need arises, accelerated dispersal of packets out of randomly located congested areas and restricted entry into those areas by packets not involved in congestion. This acts to quickly dispel congestion as well as deadlock-precipitating resource dependencies while still allowing packets to route adaptively along more profitable (less congested) paths. Addressing this challenge essentially comes down to solving a scheduling problem: How should freed resources (bubbles) be locally allocated to contending packets which may have multiple routing options such
that resource contention is minimized not only locally, but also globally throughout the network? Doing this requires exploiting some amount of global knowledge intelligently.

Many previous techniques have been proposed for preserving enough bubbles inside a network by detecting potential network congestion and, if needed, limiting the injection of new packets [3, 28, 33]. With these techniques, detected congestion on network resources is notified to source nodes of the congested packets or to adjacent nodes neighboring the congestion. Once notified, these nodes halt packet injection into congested resources. Although new packets are prevented from consuming bubbles—thus preserving some number of bubbles within the network—no mechanism is provided to ensure that packets most in need of the preserved bubbles (i.e., congested packets) ever get them in a timely manner. That is, the dispersion of network congestion already formed still remains unaddressed. In most cases, these techniques rely primarily on the routing and selection functions to determine bubble movement. What this work does is to make bubble movement a primary function of the scheduling algorithm during the time that congestion is being dispersed, which adds an extra degree of control over bubble movement.

When network routers forward packets over their internal crossbars, it is possible to prioritize the scheduling of candidate packets from the head of input ports based on one or more packet properties such as packet size, estimated hop count to destination, waiting time in the router, etc [21]. In addition, quality of service requirements can be taken into consideration, where packets may be prioritized based on bandwidth usage of their communication streams [19, 23, 41, 42]. However, to the best of our knowledge, no router scheduling techniques have provisions for directing incoming bubbles to blocked packets at input ports for the intended purpose of globally dispersing congestion, as is done here.

3 The Proposed Technique

3.1 The Basic Idea

The gist of the proposed technique is to precisely identify resources involved in paths of network congestion and temporarily increase the scheduling priorities for packets along them by allocating acceleration bubbles to those packets. This allows packets already occupying congested resources to make forward progress toward their destinations and release resources before needed resources become further congested with packets coming from outside the congested region—which would only serve to further expand the region. The proposed technique consists of a congestion detection mechanism and a concomitant resolution mechanism.

Congestion detection can be initiated by any router in the network which fails to forward packets from an input port to any of the routing candidate output ports supplied by the routing function under certain conditions. More than one router may start detection simultaneously, but only one router along a cyclic path of congestion (if it exists) needs to succeed in detecting it. The router generates a small control packet, called a ping, to probe the configuration of network congestion and sends it through the congested link to the attached neighboring router for further investigation. At the same time, a reservation is made in the router’s ping table for the next bubble supplied to the pinged output port to be allocated to the input port that sent the ping. If the neighboring
router similarly detects local congestion from its ping-receiving input port to its routing candidate output port(s), it propagates the ping to router(s) along those output ports and updates its ping table under certain conditions (see Section 3.2.2). This process is repeated by downstream routers until either no more congestion exists or the ping returns to the point where it was generated, at which point the ping terminates. In the latter case, the network has a cyclic congestion configuration as opposed to acyclic (chained) congestion in the former case.

Figure 1 illustrates a network consisting of six routers each holding in-transit packets in one of their input port’s buffers which comprise paths of congestion. The output ports consist of a single flit buffer that can temporarily hold a packet-sized flit being transmitted over the link. Router 1 is unable to transmit packets from input port Buffer A for some designated time period. After detecting this local congestion, the router then begins the process of detecting network congestion more globally by propagating the ping it generates (represented by a filled dot in the center of the router) along the congested candidate output link to its neighboring router (Router 2). This process is repeated at successive neighboring routers until the ping terminates.

Figure 1. Illustration of resource dependencies traced by ping propagation.

When propagating a ping, routers exclusively reserve the internal path between the input and output ports of the router through which the ping proceeds, called the ping path, for later backward propagation of a bubble. This is done by updating the reservation status for the input/output port pair in the router’s ping table. Hence, a total sequence of ping paths that a ping follows, referred to as the ping trace, is maintained by the associated routers’ ping tables in a distributed fashion. One of the challenges of ping propagation is in determining how pings should reserve network resources exclusively in such a way as not to incur ping-induced deadlocks arising from possible conflicts in resource reservations. The problem is that each router uses only locally available information in making decisions on further ping propagation without having a global view on resource reservations made by other pings. This issue is further addressed in Section 3.2. Nevertheless, once obtained, the ping path information is used by the router scheduler to ensure that the first bubble in the associated output port is allocated to the packet at the head of the paired input port in an attempt to resolve the detected congestion. The ping path information is removed after the head packet makes forward progress through any routing candidate output port, regardless of whether that output port is part of the reserved ping path or not. This could be the case, for instance, if only one of possibly several routing candidate output ports were reserved in the ping table for the next bubble through that port, but a bubble enters the router through a different candidate output port.
Alternatively, the information on ping paths could be cleared upon receiving cancellation requests, as described in Section 3.2.

After the ping terminates by detecting either a chained or cyclic congestion configuration, at least one bubble should be supplied to the ping trace to resolve the detected congestion via the advancement of all the associated packets along the trace by at least one buffer position. For chained (acyclic) configurations, the ping-terminating router is responsible for drawing a bubble into the ping trace from outside the configuration. To accomplish this, the router gives the highest scheduling priority to the head packet in the input port where the ping trace ends. The routed packet transfers an external bubble (that is, a bubble that originated from outside the ping trace routers) to its input port. This bubble propagates backward along the ping trace until it reaches the ping-initiating router. For cyclic congestion configurations, such external bubbles may not become available. For instance, in networks which use recovery-based deadlock handling, detected cyclic resource dependencies could be a part of knotted cycles [38, 39] forming potential deadlocks in the network. Considering this pathological case, the ping-terminating router must generate an internal bubble for the detected ping trace, i.e., one coming from a router within the ping trace.

The recovery technique proposed in this paper creates the internal bubble by use of a router-local acceleration buffer in a way such that a leading packet from the input port of the ping path is sheltered in it (a central bypass buffer in the router). Section 3.4 describes this in more detail. The bubble is then made to traverse the cyclic ping trace in the opposite direction, and, upon its return to the ping-terminating router, it is used by the sheltered packet from the acceleration buffer to free this resource for the next recovery operation. Irrespective of the source of acceleration bubbles, the packets in the chained or cyclic ping trace can eventually advance toward their destination at least by one buffer position. Such a packet-advancing operation via the scheduled use of bubbles is referred to as a shift operation. This may or may not provide immediate relief for those packets already involved in congestion. Nevertheless, new leading packets that emerge after the shift operation are likely to require a different set of candidate output ports or sink at the new router due to arriving at their destination, possibly breaking the chained dependencies. In any event, the repeated detection of congestion and possible re-use of acceleration buffers disperses the congestion configuration eventually.

To illustrate the usefulness of this idea, consider the simplistic example shown in Figure 2. After detecting congestion as in Figure 2(a) (alternatively, potential deadlock as in Figure 2(c)), the schedulers in all of the routers (not shown) determine that the preferred path through which bubbles should flow is counter to the cycle shown in Figure 2(a) or (c). Instead of randomly choosing between A1 and E1 (alternatively, E2) or naively selecting based solely on packet lifetime, the scheduler more intelligently grants an acceleration bubble to A1—one of the culprit packets acting to sustain congestion (alternatively, potential deadlock) along the cyclic resource dependency relation. If an external bubble is unavailable due to potential deadlock as in Figure 2(c), the proposed cyclic congestion resolution mechanism creates an internal bubble available to A1. Once all packets in the cycle have made use of the bubble that traverses the queues of the routers in the cycle, new packets are in the head-of-queue position and ready to be routed, as shown by the configuration in Figure 2(b). (Note that for cyclic congestion resolution, the bubble in Queue 4 is replaced by E1 in Figure 2(b).)
routing function may supply different or additional routing options to these packets, allowing congestion to be dissipated with the additional bubbles introduced to those packets resulting from the shift operation, i.e., the bubbles in Queue 7 and the Sink Queue.

3.2 Detecting Paths of Congestion

To be applicable in a realistic environment, any technique used to detect congestion should be correct (i.e., detect it when it occurs), efficient (i.e., detect it fast with minimal resources), and distributive (i.e., detect it at any node). Congestion can occur along acyclic paths, cyclic paths, or trees within the network; potential deadlock can occur only along cyclic paths of congestion. The detection of cyclic congestion does not imply the detection of actual deadlock; however, it is well known that cyclic congestion is necessary for deadlock to form and often precipitates it [38, 39, 40]. Therefore, it is an important phenomenon to detect nonetheless. Packets occupying congested resources can be identified and tracked by the dependency relation of resources along the congestion path. The detection technique we propose does this tracking using pings that probe resources suspected of such degradative behavior within the network. These congestion-detecting control packets are transmitted virtually [9] or physically [13] on the same control channels used to send other control signals.

3.2.1 Ping Initiation

Pings can be generated by a router (the ping initiator) for a given input port when the following conditions are satisfied locally: (1) the occupancy of the queue(s) associated with the port is above a certain threshold, i.e., $O_{\text{threshold}}$; (2) the packet at the head of the queue(s) cannot advance for a threshold period of time, i.e., $T_{\text{threshold}}$; (3) the input port is not reserved; (4) at least one of the routing-candidate output ports is not reserved; and (5) the input port is not an injection port. With the fifth condition, throttling occurs on injection channels. An input port meeting the first two conditions is determined to be locally congested, but all five conditions should be satisfied before a ping can be generated by the router to detect congestion more globally.
In addition to these conditions, a few others are imposed on ping generation and propagation, mainly for design simplicity and scalability. First, a router is allowed to generate a new ping only after it completely finishes all the congestion detection and resolution activities related to pings that were previously generated by the router. So, the maximum number of outstanding pings that a router can generate is limited to one. Second, an output port can be reserved by at most only one input port; it cannot be associated with another input port before it is released. Third, even if a router has multiple routing-candidate output ports, it is allowed to send the ping to at most only one of those output ports (preferably the most congested one) as acquiring a bubble from at least one of them can help to disperse congestion upstream.

With these conditions being satisfied, the process of detecting the configuration of global congestion is initiated. A ping is generated at this router with the corresponding input port used as the starting point of a ping trace. Each router’s ping path is recorded in the router’s ping table. This is indexed by an output port number, as illustrated in Figure 3. The ping table is referenced not only during global congestion detection but also during the resolution process to ensure deterministic control (i.e., reservation) of bubble movement.

Figure 3. Possible organization of a router ping table which records ping path information.

3.2.2 Ping Propagation

On receiving a ping from its neighbor, a router performs a ping propagation eligibility test to determine whether the ping should be further propagated to one of its neighbors or terminated at this router, thus completing the detection process. If the ping arrives at the router where it was generated, it terminates as a cyclic traversal over congested network resources has completed. Otherwise, it may propagate to a neighboring router if local congestion is detected at this router and other conditions are met as follows. Just as with ping generation, ping propagation can be determined based only on locally available information: the buffer occupancy of input ports, length of time the head packet has not advanced, the ping reservation status of the input port and routing candidate output ports, and whether the router has generated a ping that is still outstanding.

Reservation status has a major impact on ping propagation. Consider the case in which a ping arrives at an input port for which all the candidate output ports are already reserved by other pings. The incoming ping is said to collide with the other pings. Figure 4 illustrates this. Ping B whose trace is shown as a dotted line has detected chained congestion at Routers 1 and 2, and it now tries to proceed through the only candidate output port supplied to the head packet in input port 2 by the routing function. However, the output port was reserved prior to this by Ping A which arrived through input port 3. Both pings are not permitted to reserve the same output port since only one bubble is guaranteed to be supplied to that output port. If, for example, Ping B were permitted to reserve output port 2 without cancelling Ping A’s remaining trace, an internal bubble created at
Router 6 would flow erroneously to Router 1 and not flow back to Router 6, preventing the acceleration buffer at Router 6 from guaranteed release. On the other hand, if Ping B terminates its detection at Router 2 on account of Ping A, its ping trace should be cancelled since no bubble is guaranteed to be supplied to it.

![Figure 4. Possible collision situation of pings at a router (Router 2).](image)

When a ping is cancelled due to collision, the ping initiator may re-initiate the detection operation by generating another ping after some time-out interval. In a pathological case, a ping interlocking problem can occur in which two or more pings repeatedly collide and continue to get cancelled without detecting the congestion. This situation is illustrated in Figure 5, where Pings A and B collide with each other before either completes their detection of the cyclic congestion. One solution is to make ping initiators retry after an arbitrary backoff period has elapsed since the last cancellation, similar to the CSMA/CD protocol [35]. In addition to this, a ping trace should be cancelled only if it collides with a ping which has a higher priority. The higher priority ping waits until the lower priority ping’s trace has cleared up to that point (once the lower priority ping cancels itself), and then it continues to propagate. The priority can be represented by a ping identification (Ping ID) randomly assigned to pings which can be included as part of the ping control packet.

![Figure 5. Possible interlocking situation involving two ping traces.](image)

Figure 6 illustrates how priority-based ping cancellation works when more than one router tries to detect the same congestion configuration by each generating independent pings. In this example, two pings are generated (Routers 2 and 5), each propagating up to the router which generated the other ping. If the incoming ping is of higher priority, it outranks the other ping and, thus, waits for the lower-ranking ping to clear its cancellation before continuing its propagation. The higher-ranking ping proceeds along the former ping trace of the lower-
ranking ping, updating the Ping ID field in the ping tables with its own. If the incoming ping is of lower priority, it terminates its propagation and immediately cancels its trace as described in Section 3.2.3. This ensures that cyclic congestion is detected by at most one ping, the highest ranking one. As shown in Figure 6, Ping A returns to Router 2 to complete a cyclic traversal after reservations made by Ping B at Routers 5, 4, and 1, successively, have been cancelled.

For the case in which a higher-ranking ping completes the detection of cyclic congestion at a router which did not generate the ping, a shift operation is allowed to start at that router. In this case, a ping trace cancellation operation needs to be initiated by that router to clear the partial ping trace from it back to the ping-initiating router; otherwise, the partial ping trace remains reserved with no guaranteed supply of an internal or external bubble. This would be the case for the congestion configuration shown in Figure 5 when Ping A outranks Ping B. The tails of both ping traces would be cancelled: Ping B’s tail is cancelled since Ping B is found to be lower ranking, and Ping A’s tail is cancelled since a cycle of congestion is detected at Router 6.

![Figure 6. An example of two routers detecting the same cycle; Ping B’s trace is cancelled.](image)

### 3.2.3 Ping Trace Cancellation

Ping trace cancellation plays a crucial role in preventing network resources from remaining reserved due to unsuccessful detection of network congestion. Cancellation occurs by setting a cancellation flag of the corresponding ping control packet and propagating the packet backward along the ping trace. Figure 7 illustrates the cancellation operation done within a router. When receiving a cancellation packet, a router first references its ping table to locate the proper entry with the same Ping ID, clears the entry, and then propagates the cancellation control packet back through the input port recorded in the ping table entry using the bidirectional control channel. This continues until the cancellation packet reaches the point where the corresponding ping was created or where no entry is found with the same Ping ID.

### 3.2.4 Description of the Ping-based Congestion Detection Mechanism

As previously stated, the occupancy status of input port queues, the ping reservation status of input/output ports as given by the router’s ping table, and the router’s ping generation status is used to perform the ping propagation eligibility test. The input port status is checked to determine if the associated buffer is locally
Propagate the cancellation packet through the input port read from the ping table

Router

Port 0

Port 1

Port 2

Port 3

Ping Table

PID Origin Valid Input Port Num Output Port Num
0 no no
1 no no
2 1234 ext yes 0
3 no no

(1) Cancellation packet arrives (PID = 1234)
(2) Look up the ping table for Port 2
(3) Read out the entry and delete it
(4) Propagate the cancellation packet through the input port read from the ping table

Figure 7. Cancellation operation occurring within a router which receives a cancellation packet.

Congested (Locally Congested). The ping table is examined to determine whether the input port of the incoming ping has been previously reserved (Previously Reserved), and whether that port is reserved by the same ping (By Same Ping). Similarly, the ping table’s reservation status for the candidate output ports is inspected to determine whether any candidate output ports are reserved (Reserved Port(s)), whether there exists any candidate output port reserved by the same ping before (By Same Ping), and whether there exists at least one candidate output port available for further reservation (Any Port Not Pinged). The test also checks whether this router has generated a ping that remains outstanding in the network (Generated a Ping), whether the incoming ping was generated at this router (Same Ping), and whether the incoming ping’s priority level is greater than the pings which have previously reserved candidate output ports, if any (Ping Priority).

Table 1 summarizes the various cases that can arise and possible outcomes of the ping-based congestion detection mechanism. Entries with hyphens represent don’t cares or are not applicable for the respective cases. Action taken once a ping terminates or a cycle of congestion is detected is for the router to invoke the congestion resolution technique presented in the next section. Further description of the possible cases is given below.

Table 1. Reservation status checked at routers upon the arrival of a ping.

<table>
<thead>
<tr>
<th>Case</th>
<th>Locally Congested</th>
<th>Previously Reserved</th>
<th>By Same Ping</th>
<th>Reserved Port(s)</th>
<th>By Same Ping</th>
<th>Any Port Not Pinged</th>
<th>Generated a Ping</th>
<th>Same Ping</th>
<th>Ping Priority</th>
<th>Action to take on ping</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>no</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Terminate Ping</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>no</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Propagate Ping</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>-</td>
<td>low</td>
<td>-</td>
<td>Cancel Ping Trace</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>-</td>
<td>high</td>
<td>-</td>
<td>Outranking Propagate</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Propagate Ping</td>
</tr>
<tr>
<td>6</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
<td>-</td>
<td>no</td>
<td>-</td>
<td>-</td>
<td>Detect Cycle, Cancel Tail</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
<td>-</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>Cancel Ping Trace</td>
</tr>
<tr>
<td>8</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>yes</td>
<td>yes</td>
<td>-</td>
<td>yes</td>
<td>-</td>
<td>-</td>
<td>Detect Cycle</td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>high</td>
<td>-</td>
<td>Outranking Propagate</td>
</tr>
<tr>
<td>10</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>low</td>
<td>-</td>
<td>Cancel Ping Trace</td>
</tr>
<tr>
<td>11</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Detect Cycle</td>
</tr>
</tbody>
</table>

Case 1: Because the input port is not congested, the router terminates the incoming ping. The bubble inside the input port will be propagated backward along the ping trace. No further action is required.
Case 2: The input port is congested, so the incoming ping needs to be propagated further. As no reservations exist on routing candidate output ports, the router randomly selects one of the supplied output ports, propagates the ping through the port, and records (reserves) the ping path in the ping table.

Cases 3, 4 and 5: The incoming ping needs to be propagated, but some or all of the candidate output ports are reserved by other pings. If there exists at least one candidate output port available for reservation (Case 5), the ping can be propagated through one of them randomly or based on additional flow control information normally implemented in routers. Figure 8 illustrates this case with Ping B at Router 3. Otherwise (Cases 3 and 4), the ping collides with other pings. Now the ping priorities are compared to resolve potential ping interlocking. If the detection priority of the incoming ping is lower than the previous ping that reserved the candidate output port through a different input port, the incoming ping terminates and its trace is cancelled (Case 3). This is illustrated with Ping D at Router 1 in Figure 8, which has lower priority than Ping A. Otherwise, the incoming ping outranks the previous ping, so it continues to propagate once the reservation by the lower-ranking ping has been cancelled (Case 4).2

Cases 6, 7 and 8: The ping needs to be propagated, and some of the candidate output ports are reserved by the same ping. In this case, the ping can complete a cycle at this router if the output port is reserved for the input port through which the ping just arrived, which is different from the previous input port used initially by the same ping. This is the case of Ping A arriving at Router 2 through input port 3 in Figure 8. Even though Ping A was generated at Router 1, it discovers cyclic congestion that spans over the buffers of Routers 2, 3, 6 and 5. To distinguish between these three cases, the router checks whether it has generated a ping that remains outstanding in the network and if it is different from the incoming ping. If so (Case 7), the router’s generated ping is given priority over the use of the acceleration buffer at that router in case it completes a cycle, so the incoming ping is not allowed to complete its cycle (which would cause it to use the router’s acceleration buffer).3 An example scenario of Case 7 in Figure 8 would be if Router 2 had generated a ping from Port 1 to Port 3 (shown in the figure as an untraced resource dependency) and Ping A arrived at Router 2 through input port 3 to complete a cycle. For the other two cases (Case 6 and 8), the incoming ping is allowed to modify the ping table entry so that it completes a cyclic traversal. In Case 6, the tail of the ping trace from this router back toward the ping-initiator must be cancelled (e.g., from Router 1 to Router 2 for tail of Ping A’s trace in the figure); otherwise, the reserved resources do not have a guarantee of a bubble ever arriving. In Case 8, such cancellation is not necessary as the ping was initiated by this router. Figure 8 illustrates this with Ping A tracing Routers 1, 2, 3, 6, 5, 4 having been generated and terminated at Router 1 from input ports 0 and 3, respectively.

Cases 9, 10 and 11: The input port is congested, but is already reserved. If it is reserved by the same ping (Case 11), the incoming ping completes a cyclic traversal and terminates. This is the case in Figure 8 of Ping A arriving at Router 1 through input port 3, having been initiated at that router through the same input port. Otherwise, it is the case where more than one ping is detecting the same path of congestion. Since the router has

---

2The lower ranking ping cancels itself when it encounters a ping interlocking situation with a higher-ranking ping (Case 3 or 10).

3Note that if routers implement more than one acceleration buffer, it would be possible for the incoming ping to complete its cycle and use one of the available acceleration buffers, leaving one for the ping generated at that router.
a previous reservation on the input port, it compares the detection priorities of the two pings. If the incoming ping has a lower priority than the previous ping (Case 10), it is terminated and its trace is cancelled. Otherwise, if the incoming ping has a higher priority (Case 9), it outranks the previous ping and continues to propagate once its reservation has been cancelled. Note that the lower ranking ping cancels itself only if it encounters a ping interlocking situation as it propagates, which may not happen. For example, in Figure 8, if Ping A has a lower priority than incoming Ping D which arrives afterward (assuming both pings are associated with input port 0), Ping A completes a cycle at Router 1 without interlocking with a higher ranking ping, including Ping D. Nothing is lost by having Ping D delay its propagation until Ping A is cancelled or terminates after completing its cyclic traversal as, either way, the path of congestion involving input port 3 is detected and action is taken to resolve it, described in the next section.

![Diagram of ping trace configuration](image)

**Figure 8. Examples of reservation configuration.**

Figure 9 shows a flowchart describing the proposed ping-based congestion detection mechanism. The algorithm starts when an incoming ping arrives at an input port of a router.

### 3.3 Congestion and Potential Deadlock Resolution

The resolution technique proposed in this section resolves temporal and spatial resource dependencies by prescheduling bubbles to needed input ports before those bubbles actually arrive at routing candidate output ports. This is applicable both to networks based on deadlock avoidance as well as to those based on deadlock recovery. Hence, the proposed technique resolves congestion as well as potential deadlock. It works in conjunction with the ping-based congestion detection mechanism described in the previous section. For congestion to be resolved, precise control over bubble movement is required such that priority for bubble flow is along the links which most contribute to congestion. As resource reservation for bubbles is based on the ping-based congestion detection mechanism, we refer to this technique as the **Ping & Bubble** (or PB) technique.

In networks which always avoid deadlock, only congestion can occur which should be resolved quickly after detection. External bubbles are guaranteed to be supplied to network resources that need them eventually once a ping trace is formed. Reservations on the ping path made for each pinged output port indicate that the next bubble which sooner or later comes to that port will be allocated to the associated input port that is either (1) detected as closing a congestion cycle or (2) detected simply as being a part of chained congestion. When a
bubble arrives at an output port, the scheduler looks up the ping table to see if a reservation has been made for that port. If so and if the grant signal for the output port is accepted, the corresponding entry in the ping table is cleared. In this way, arriving bubbles are coerced to flow along a pre-detected path of congestion, allowing the opportunity for packets along that path to disperse out of the congestion by shifting forward, as illustrated in Figure 2(b).

Now consider the case of deadlock recovery networks. In these networks, both congestion and potential deadlock situations can occur which should be resolved quickly after being detected. Internal bubbles must be introduced into the network resources that need them by removing at least one of the packets involved [39], i.e., packet A1 in Figure 2(c). This can be done by sheltering into the acceleration buffer the culprit packet at the head of the router’s input port identified as closing the dependency cycle. This bypass buffer should be large enough to hold an entire packet without loss of information. When the internal bubble completes circulation along the cyclic ping trace, it returns to the router and evacuates the culprit packet from the acceleration buffer.

It is also possible for one or more external bubbles to be drawn into ping traces during the circulation of internal bubbles. This is because it is possible for ping-propagating routers to route packets from the input port of the ping path to one of the output ports outside of the ping path upon receiving external bubbles on those candidate output ports (which were not in the ping table). Once drawn into ping traces, external bubbles are treated the

---

**Figure 9. Flowchart describing the ping-based congestion detection mechanism.**

---

COP: Candidate Output Port
COPNP: Candidate Output Port that is Not Pinged by others
same way as internal ones and, therefore, are used to evacuate acceleration buffers faster without harm.

The bubble mechanism assumed in this work allows each router to generate at most one internal bubble, i.e., only one acceleration bypass buffer per router. This resource requirement is small compared to other techniques such as BLAM [34] which provides a bypass buffer\(^4\) for each virtual channel and generates bubbles for packets blocked on congestion through the use of misrouting. This alternative may be as effective as the proposed PB technique in quickly dispersing packets out of locally congested areas. However, it has the possible side-effect of out-of-order packet delivery, increased resource cost, and non-dispersal of paths and cycles of global congestion. In contrast, the PB technique supports in-order delivery of packets and uses nominal resources to achieve eventual dispersal of global congestion.

### 3.4 Modified Router Architecture

Figure 10 shows a basic pipelined router architecture [25] that supports our proposed PB technique. The additional hardware beyond a traditional router architecture includes two tables (Routing Failure Table and Ping Table), two registers, and a packet-sized acceleration bypass buffer used for generating an internal bubble in the case of deadlock recovery.

The Routing Failure Table is indexed by input port number. It caches stalled routing options due to congestion at the corresponding output ports. This makes it possible for the ping propagation eligibility test to acquire outstanding dependency relations between input and output ports quickly without having to look up the routing table. Depending on the actual implementation, each routing failure table entry contains 1 to \(n\) output ports, where \(n\) is the maximum number of output ports. The size complexity of the table is calculated as \(O(mn)\) in the maximal configuration, where \(m\) and \(n\) are the number of input and output ports, respectively. Each entry should be cleared upon progress made by a packet in the corresponding input port.

As mentioned earlier, the ping table (see Figures 3 and 10) records the reservation status of ping paths. Because an output port is associated with at most one ping path, the ping table can be organized such that each entry is indexed by an output port number. The ping table has one input port per entry, which limits the size complexity to \(O(n)\), where \(n\) is the number of output ports.

The Ping ID Register and the Ping Status Register are used only when the associated router generates a ping to initiate global detection of network congestion. The former records the Ping ID of a newly generated ping. Each router can have a maximum of one outstanding ping at a time. The information in this register is used to compare detection priorities with any incoming pings. The Ping ID consists of a random number portion and an identification portion for the ping initiator. For fair distribution of detection priorities over ping initiators, the random number portion is the most significant part of the Ping ID. The Ping Status Register indicates the current status in congestion detection and resolution. Possible states include *idle*, *detection* and *resolution* states. The detection state represents that the ping generated by the router is alive and is still exploring the congestion configuration. The resolution state represents that the recovery operation is on-going and that the

---

\(^4\)The Alpha 21364 router also implements a bypass mechanism in its channel queues.
acceleration buffer is occupied by a packet from the pinged input port. This register is mainly used to prevent other pings from completing a cyclic traversal and using the acceleration buffer that has higher priority for use by this router’s generated ping.

Overall, augmentation of the router architecture should impose nominal impact on the critical path. Most of the added hardware is invoked only when the router experiences enough local network congestion to initiate or assist more global congestion detection activities. In addition, the added hardware works in parallel with existing router hardware, e.g., routing table look-up and crossbar arbitration.

![Diagram of router architecture](image)

**Figure 10.** A basic pipelined router architecture which incorporates the additional hardware (shown in bold boxes) for detection and resolution of cyclic and acyclic network congestion.

4 Performance Evaluation

4.1 Simulation Methodology

The simulator used for evaluation is Flexsim 1.2p which is a direct successor of the flit-level network simulator, Flexsim 1.2, developed by the SMART Interconnects Group at USC. It incorporates the ping-based congestion detection mechanism as well as the bubble-based recovery technique. The simulator inherits from its predecessor many useful features, such as performing flit-level traffic flow within the interconnection network and maintaining data structures that represent resource allocations and dependencies (resource wait-for relationships) occurring within the network. Also, store-and-forward type of packet switching of the acceleration buffer is mimicked by the simulator as it supports virtual cut-through switching (which behaves like store-and-forward when packets block).

Pings are assumed to be transmitted across the network through physically separated links (i.e., out-of-band). They could, however, be implemented in-band, logically—over the same physical links as those used for data transmission. In either case, the actual bandwidth consumed by ping propagation has minimal impact on the transmission of data packets. Simulation results confirm that in-band ping transmission uses only a small
fraction (less than 2%) of data transmission bandwidth.

In this study, regular \((k\text{-ary } n\text{-cube toroidal})\) networks with the following default parameters are simulated: 8-ary 2-cube, bidirectional channels, virtual cut-through flow control, 4 virtual channels per physical link and random traffic patterns. Each virtual channel is assumed to hold one packet at a time. The default threshold occupancy of the queue \((O_{threshold})\) is set to 100%, and the default threshold timeout for initiating ping navigation \((T_{threshold})\) is set to 25 cycles. All of these parameters and traffic patterns will be varied through a number of simulation experiments. The ping propagation latency through a router is assumed to be one cycle. Performance results are given in terms of separate latency and throughput plots over a wide range of applied network loading conditions, from no-load to deep saturation. The proposed Ping & Bubble (PB) technique is implemented on the network assuming true fully adaptive routing (TFAR) as default, but it is also used in conjunction with other well-known avoidance-based and recovery-based routing techniques: Duato’s Protocol [5] and Disha [36]. The PB technique is also compared against a recently proposed congestion limitation technique, ALO [3]. The ALO technique requires that at least one virtual channel queue along the path towards the destination of the packets at a router be empty in order for injection into the network not to be limited or halted. Comparisons are made over a range of traffic patterns, network sizes, queue sizes, bristling factors, and occupancy thresholds.

4.2 Performance Results

4.2.1 Efficiency of Congestion Handling

The PB technique implemented on networks with deadlock avoidance-based approaches is used to help disperse chains of network congestion. It is enabled by the deadlock-avoidance routing function which guarantees delivery of packets in the network and creation of external bubbles for congestion dispersion. Therefore, the main role of the PB technique is to efficiently deliver those external bubbles to places where they are needed most. For this experiment, we use torus networks with Duato’s adaptive routing protocol with 4 virtual channels per physical link: two channels for escape paths and two for adaptive paths. For simplicity, ping generation and propagation are all based on the congestion status of adaptive paths only.

Figure 11 shows the performance of the two networks: one using Duato’s Protocol only (i.e., without the congestion relieving capabilities of the PB technique), and one using Duato’s Protocol in conjunction with the PB technique. As shown in the figure, peak throughput can be improved by 5% by incorporating the PB technique, and for saturated network loads of 0.75 and greater, sustained network throughput can be improved by up to 100%. These results show that having bubbles in the network is not sufficient to efficiently resolve network congestion. Instead, bubble movement in the network needs to be properly controlled to achieve better performance. These results also confirm that the PB technique is useful even in deadlock-free networks—with the benefit being that paths of network congestion can be efficiently dispersed once it starts to form.

Figure 12 shows the effectiveness of two congestion-aware scheduling techniques: the PB technique and an age-based scheduling technique, each combined with Duato’s Protocol. In age-based scheduling, each packet
Figure 11. Network throughput and latency for networks with and without the proposed PB congestion sensitive scheduling technique in an $8 \times 8$ bidirectional torus with 4 virtual channels. (a) Network performance in latency vs. applied network load, and (b) peak and sustained network throughput vs. applied network load.

has a lifetime in its header and is given a scheduling priority based on it. As can be seen, age-based scheduling is also effective in dispersing network congestion by assigning higher scheduling priorities to packets with longer lifetimes within the network. However, it is also observed that in deep saturation the PB technique yields slightly higher throughput than age-based scheduling. Packet age is a good indicator of congestion along the path experienced thus far by the packet, but it is not as good as the pinging mechanism at drawing bubbles into the root of congestion and keeping them within the congestion path. From the perspective of implementation complexity, however, age-based scheduling pays much less for slight loss of performance, which could be considered good trade-off between cost and performance. When these two techniques are used together, network performance increases slightly as both forward and backward congestion path information is used in making scheduling decisions.

The average number of hops each ping takes before it is terminated or cancelled is seen to be relatively small: 1.28 hops. Results show that 81.1%, 13.2%, 3.3% and 1.3% of the pings are terminated or cancelled after going 1, 2, 3 and 4 hops away from their initiating routers, respectively. From this observation, it is conceivable that a reservation-based scheduling algorithm can be developed for chained congestion that does not propagate pings outside routers but, instead, only reserves local resources. However, the propagation of pings are still necessary to handle the case of cyclic congestion, as discussed in the next section.

4.2.2 Efficiency of Deadlock Handling

The PB technique can also be used for handling potential deadlock anomalies in networks by detecting and resolving network congestion—which precipitates deadlock. For this experiment, three deadlock handling techniques are compared in terms of network throughput and packet delivery latency: true fully adaptive routing with the PB technique, true full adaptive routing with the Disha technique, and avoidance-based routing using Duato’s Protocol.

Figure 13 shows the simulation results for the three techniques with varying network loads. As can be seen, PB and Disha techniques yield similar performance in terms of peak network throughput, outperforming
Figure 12. Network throughput for networks with and without the proposed PB congestion sensitive scheduling, each with and without an age-based scheduling in an $8 \times 8$ bidirectional torus with 4 virtual channels.

the avoidance-based technique. The difference in network throughput increases as the network load increases beyond the saturation point, as shown in Figure 13(b). Both Disha and Duato significantly suffer from degradation in network throughput, while PB sustains its throughput beyond network saturation. Among these techniques, PB best disperses network congestion by drawing available bubbles to the congested region through its congestion-aware scheduling. Once accelerated toward their destinations, many packets are able to make forward progress or get delivered to their destinations instead of remaining congested in the network. This creates new bubbles that flow into the network which can be used by other packets to make forward progress to their destinations.

Figure 13. Network throughput and latency for networks with deadlock avoidance and recovery routing. Network performance is plotted in terms of (a) latency and (b) measured network throughput vs. applied network load. Duato’s adaptive routing, true fully adaptive routing with Disha progressive recovery, and true fully adaptive routing with Ping & Bubble technique in an $8 \times 8$ bidirectional torus with 4 virtual channels are shown.

Further analysis indicates the major contribution to congestion dispersion comes from the detection of chained (acyclic) resource dependencies. Results shown that 85% of newly generated pings terminate after detecting acyclic dependencies. Only 14.95% of them are cancelled due to ping collisions, and less than 0.1% of cancelled pings are due to interlocking. The remaining 0.05% of the pings complete cyclic traversal on network resources, thus tracking cyclic congestion. This is a rare situation indeed, which confirms prior observations on the infrequency of this deadlock-precipitating phenomenon [38, 39, 40]. This also highlights the PB
technique’s feature of handling potential deadlock anomalies at very early stages by dispersing cyclic network congestion well before knotted resource dependencies take hold. During the simulations, no deadlocks were observed for any of the networks.

The impact of the shift operation on network performance could be affected by routing flexibility, average distance to destination, and many other network parameters. In particular, the first affects how shifted packets differ from their predecessors in terms of resource dependency, while the second impacts how many packets could sink at their destinations after shift operations. Simulation results show that 10% of the shifted packets sink after the shift operation, another 10% have routing dependencies on non-pinged output ports, about 2-3% of them become free from congestion by having bubbles at needed output ports, and the rest still remain blocked by the previous packets.

A further advantage of the PB technique over Disha (in addition to achieving better performance) is that it does not cause out-of-order delivery of packets during the recovery operation. Even though Disha is capable of progressively recovering from potential deadlocks, the packets recovered via the resources for deadlock-recovery may bypass preceding packets which are in the same communication stream. These packets, in the worst case, may cause re-transmission of a large chunk of packets due to the execution consistency enforced by the application or a higher level network protocol. Our simulation results show that in Disha-based networks, around 10% of the packets are delivered out-of-order through deadlock recovery resources when network load is beyond 80% of the maximum network capacity. In networks using the PB technique to recover from potential deadlocks, packets in cyclic congestion configurations are shifted toward their destinations by one buffer position while maintaining their relative order within the network.

4.2.3 Effect of Injection Limitation Capability

The performance improvement from the proposed techniques is mostly due to efficient dispersion of network congestion. But, it is also partially due to the capability of the PB mechanism to limit injection of new packets by end nodes. In fact, this capability is provided as a by-product of the ping generation and propagation policy. First, packet congestion at an injection port is not allowed to trigger ping generation, which prohibits active dispersion of the congestion formed at injection ports. Second, resource reservations via ping operations give priority to the associated input ports for the potential bubbles at the outputs, thus throttling packet injection toward those output ports from end nodes.

To observe the effect of the injection-limiting capability of our PB technique, two networks with true fully adaptive routing are simulated, only one of which implements relaxed injection limitation by allowing injection ports to trigger ping operations. Note that the ping mechanism, as it is, has an added advantage of limiting injection by disallowing network end nodes from injecting new packets into pinged output ports.

Figure 14 shows the effect of injection throttling inherently provided by the proposed PB technique. As shown in the figure, networks yield better performance when injection ports are prohibited from triggering ping operations, although the effect is not significant (see the curves with the labels PB with No InjLim and PB with...
When injection ports are allowed to trigger pinging, the network bubbles leave the network which, in turn, increases the congestion level in the network. One of the reasons for having such a small difference in the performance is that such inherent limitation itself is not sufficient enough to prevent networks from becoming deeply saturated. The networks with fully adaptive routing can provide more than one routing path for new packets waiting for injection, which enables those packets to enter the network via non-reserved router ports.

The figure also shows the impact of deadlock handling techniques on the performance of networks using a previously proposed injection limitation technique, called ALO (At Least One) [3]. As indicated in the figure, the Disha-based networks benefit less from the ALO technique. ALO is reported to be effective in wormhole networks [3], but it is less efficient for throttling packet injection in virtual cut-through networks as shown here. This is because by buffering entire packets at the routers, virtual cut-through networks slowly propagate congestion to end nodes which could mislead end nodes into believing that the network is not congested yet. In other words, when ALO starts to limit packet injection, the network is saturated to a certain extent, but not fully. In this case, while the Disha-based networks suffer from overhead due to deadlock handling, the PB technique can efficiently utilize the remaining bubbles to make progress in packet delivery.

![Network latency for the networks with injection-limiting techniques in an 8x8 bidirectional torus with 4 virtual channels, and with PB-based true fully adaptive routing protocol.](image)

**Figure 14.** Network latency for the networks with injection-limiting techniques in an 8x8 bidirectional torus with 4 virtual channels, and with PB-based true fully adaptive routing protocol.

### 4.2.4 Effect of Traffic Patterns

In this section, various traffic patterns are used to measure the efficiency of the proposed technique. Non-uniform traffic such as bit-reversal and perfect shuffle are used for the evaluation and can be compared against results for uniform random traffic presented earlier. This experiment is later augmented by using the self-similar traffic model presented in [2, 14, 30]. Self-similar traffic has been observed in many LANs by well-known network applications such as telnet, ftp and WWW [4]. A body of research has performed the evaluation of networks under this more realistic traffic pattern to confirm previous performance results under Poisson-generated traffic loads.

Figure 15 shows network throughput for networks with the three deadlock avoidance and recovery techniques. As can be seen, recovery-based deadlock handling techniques continue to achieve better performance in terms of network throughput than the avoidance-based technique. However, there is an insignificant difference between Disha and PB. This result is a consequence of the unbalanced distribution of network traffic.
that incurs high congestion only on a subset of network resources. The traffic patterns used in this experiment allow one end node to exchange packets with only one specific counterpart as determined by the mapping function used in destination selection. This causes a subset of network resources to be used more frequently than other parts of the network, and congestion occurs mainly on the more frequently used resources. Therefore, the throughput achieved under high traffic loads mostly comes from packet delivery through less congested resources.

![Network throughput for the networks with deadlock avoidance and recovery routing assuming (a) bit-reversal traffic, and (b) perfect shuffle traffic. Duato’s adaptive routing, true fully adaptive routing with Disha progressive recovery, and true fully adaptive routing with PB technique in an 8 × 8 bidirectional torus with 4 virtual channels are assumed.](image)

To observe the performance of the deadlock handling techniques using more realistic traffic loads, self-similar traffic is used. Previous work by Leland et al., [14] shows that the traffic in Ethernet is bursty over a wide range of aggregation scales, which is modelled by the accumulation of a large number of ON/OFF packet train sources. For this type of traffic, packets are generated during ON states. Figure 16 shows network throughput for the networks with three deadlock avoidance and recovery techniques under self-similar traffic. As can be seen, the overall performance of these techniques under self-similar traffic is very similar to that under random traffic. It follows intuitively because self-similar traffic is based on random traffic with ON/OFF states for generating artificial traffic bursts. The PB technique outperforms the Disha technique by up to 200% in sustained network throughput when saturated network loads are used. The peak throughput each technique achieves under self-similar traffic is slightly lower than that under random traffic.

To observe the effectiveness of the congestion/deadlock handling capabilities of the PB and Disha techniques under even more bursty traffic patterns, the periodic traffic pattern shown in Figure 17 is used for simulations. During the evaluation, maximum load is fixed to 100% while the normal load varies from 30% to 70%. In addition, the period of normal load ($t_{normal}$) used is fixed to 500 cycles, and the period of maximum load ($t_{maximum}$) is varied from 500 to 2000 cycles.

Table 2 summarizes network performance in terms of network throughput, packet delivery latency and congestion/deadlock resolution activities. For PB, the third item is the number of pings generated for congestion resolution; for Disha, it is the occurrence of token possessions for deadlock resolution. As shown in the table,...
Figure 16. Network throughput for networks with deadlock avoidance and recovery routing under self-similar traffic. Duato’s adaptive routing, true fully adaptive routing with Disha progressive recovery, and true fully adaptive routing with PB technique in an $8 \times 8$ bidirectional torus with 4 virtual channels are assumed.

Figure 17. Periodic-varying network load to emulate temporal non-uniformity.

ble, both techniques perform almost equally well when traffic is less bursty (for maximum load period of 500 cycles). However, the effectiveness of the congestion handling capability of the PB technique can be vividly observed when traffic becomes more bursty and when normal network load increases. When the maximum load lasts for 500 cycles and the minimum load is set to 30%, both PB and Disha achieve almost the same performance in terms of throughput and latency. However, Disha executes the recovery operation through deadlock buffers 1058 times, once per 20.69 packets delivered. When the maximum load period is 2000 cycles and the minimum load is 70%, 40.76% of the delivered packets are through deadlock buffers. This implies that network performance is limited by the deadlock recovery bandwidth and that Disha performs inefficiently when the network is deeply saturated. PB continues to perform efficiently in deep saturation.

4.2.5 Performance in Large Networks

We experimented with a $16 \times 16$ torus network to observe the performance of the proposed congestion control technique in larger scaled networks. Figure 18 shows the performance of the two deadlock handling techniques: PB and Disha. As can be seen, the performance of the two techniques is similar, i.e., both suffer from performance degradation when the network is deeply saturated. The reasons for this are the following. In Disha, once the network is fully saturated, more routers experience timeouts on packet delivery and compete for the use of recovery resources which are not abundant, resulting in recovery resources becoming the performance bottleneck. However, in PB, as the size of the network increases, the length of chained congestion tends to increase which causes pings to travel longer paths (more routers) before discovering cyclic dependencies or
Table 2. Performance of the two congestion/deadlock handling techniques in resolving temporal network congestion.

<table>
<thead>
<tr>
<th>Loads</th>
<th>Technique</th>
<th>PB</th>
<th>Disha</th>
<th>PB</th>
<th>Disha</th>
<th>PB</th>
<th>Disha</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal/Max</td>
<td>Max Load Period</td>
<td>500</td>
<td></td>
<td>1000</td>
<td></td>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>0.3/1.0</td>
<td>Throughput</td>
<td>0.553</td>
<td>0.547</td>
<td>0.614</td>
<td>0.616</td>
<td>0.628</td>
<td>0.575</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
<td>166.9</td>
<td>162.0</td>
<td>208.0</td>
<td>208.5</td>
<td>316.4</td>
<td>373.5</td>
</tr>
<tr>
<td></td>
<td>Ping/Recovery</td>
<td>1</td>
<td>1058</td>
<td>103</td>
<td>1567</td>
<td>1373</td>
<td>2217</td>
</tr>
<tr>
<td>0.5/1.0</td>
<td>Throughput</td>
<td>0.632</td>
<td>0.637</td>
<td>0.649</td>
<td>0.616</td>
<td>0.622</td>
<td>0.302</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
<td>186.6</td>
<td>191.9</td>
<td>268.2</td>
<td>314.6</td>
<td>317.2</td>
<td>1178.0</td>
</tr>
<tr>
<td></td>
<td>Ping/Recovery</td>
<td>34</td>
<td>1419</td>
<td>544</td>
<td>1903</td>
<td>1663</td>
<td>3754</td>
</tr>
<tr>
<td>0.7/1.0</td>
<td>Throughput</td>
<td>0.620</td>
<td>0.616</td>
<td>0.613</td>
<td>0.239</td>
<td>0.611</td>
<td>0.238</td>
</tr>
<tr>
<td></td>
<td>Latency</td>
<td>302.7</td>
<td>320.8</td>
<td>337.8</td>
<td>1471.1</td>
<td>368.0</td>
<td>1370.5</td>
</tr>
<tr>
<td></td>
<td>Ping/Recovery</td>
<td>1398</td>
<td>2126</td>
<td>2015</td>
<td>4114</td>
<td>2371</td>
<td>4462</td>
</tr>
</tbody>
</table>

terminating at the end of acyclic dependencies. The longer pings travel, the higher the probability of collisions and interlocking with other pings, which increases pings cancellation overhead. This implies that in order for networks to gracefully degrade their performance under high network load, they should have an efficient way of reducing the average length of ping traces, i.e., by having more available bubbles.

Figure 18. Network throughput and latency for PB and Disha deadlock handling techniques in an $16 \times 16$ bidirectional torus with 4 virtual channels. (a) Network performance plotted in terms of latency and (b) measured network throughput vs. applied network load.

In order to improve the throughput in larger networks, it is necessary to prevent networks from being too saturated so that more bubbles could be available for resolving congestion. In Section 4.2.3, it is shown that resource reservation through pings has an advantage of limiting injections from the end nodes. This could be extended further in the following way based on ping history. If a router has an output port which has been pinged over the last $C_{threshold}$ cycles, it disallows attached end nodes from injecting new packets. This is useful since even though the router may not currently have any pinged output ports, its neighboring routers may still be congested due to relatively slow propagation of bubbles. Figure 19 shows the performance of the...
modified PB technique with such an explicit injection limiting feature enabled. As can be seen in the figure, larger thresholds contribute to preventing the network from suffering a severe performance drop when the network load increases beyond the saturation point. One of the advantages of this limiting technique is its low implementation cost: it requires only a timeout counter at each router. The counter is reset whenever an output port is pinged; otherwise, it counts up every cycle. When the counter value is less than $C_{\text{threshold}}$, packets from injection channels are not routed.

![Figure 19. Effect of limiting injection based on ping history. Network throughput for PB and Disha deadlock handling techniques with varying $C_{\text{threshold}}$ (shown in parenthesis) in a 16 × 16 bidirectional torus with 4 virtual channels is plotted.](image19)

4.2.6 Effect of Ping Eligibility Parameters

The PB technique includes two threshold values associated with ping generation and propagation. The first is the threshold for queue occupancy, $O_{\text{threshold}}$, where the queue occupancy should be equal to or greater than $O_{\text{threshold}}$ to initiate or propagate congestion detection. The second is the threshold in the number of clock cycles for timeout, $T_{\text{threshold}}$, where congestion should last longer than $T_{\text{threshold}}$ cycles to generate a new ping. By reducing the occupancy threshold, network congestion can be detected at an earlier stage. If the threshold is set to a low value, pings could be generated even with slight network backup. In this case, there is a

![Figure 20. Effect of threshold parameters of ping mechanism on network performance in an 8 × 8 bidirectional torus with 4 virtual channels, and with PB-based true fully adaptive routing protocol: (a) effect of $O_{\text{threshold}}$ and (b) effect of $T_{\text{threshold}}$.](image20)
possibility that valuable bandwidth for data transmission could be wasted by excessive ping packets. Likewise, when the timeout threshold is set to a low value, the pinging mechanism could be activated upon short-term congestion in packet buffers. Otherwise, only mid- or long-term congestion would trigger pinging—in which case, network congestion might be widely spread over many network routers.

Figure 20 shows the effect of threshold parameters of the ping mechanism on network performance. $O_{\text{threshold}}$ varies from 5% to 100% of the total queue capacity and $T_{\text{threshold}}$ increases from 5 to 80 cycles. Both parameters have little impact on performance at lower network loads. When decreasing the values for both $O_{\text{threshold}}$ and $T_{\text{threshold}}$, there exists a slight performance improvement, but not significant. For the presented network loads, setting both $O_{\text{threshold}}$ and $T_{\text{threshold}}$ to low values such as 5% and 5 clocks, respectively, improves the performance a little more, but not significantly.

4.2.7 Effect of Input Buffer Size

As advances in VLSI technology continues, routers tend to provide larger and larger input buffers (channel queues). Large buffer resources enable queues to be shared by more than one packet in a non-atomic fashion [6]. To see the effect of large input buffers, simulations were run assuming each router has an input buffer of four 32-flit packets per virtual channel. The affluence in buffer resources effectively increases network capacity such that traffic bursts can be accommodated by network resources. Figure 21 (a) shows the performance of the PB and Disha techniques. As can be seen, no difference can be observed between the two techniques. This is because the simulation model used in this experiment assumes that packets delivered to the destination sink immediately, and, thus, the performance bottleneck is placed at the injection of new packets.

![Figure 21. Performance of congestion/deadlock handling techniques in networks with the input buffers of 4 packets. An 8 × 8 bidirectional torus with 4 virtual channels is simulated with the PB and Disha techniques: (a) bristling factor of 1 and (b) bristling factor of 2.](image)

To observe the performance of the proposed technique on routers with large input buffers in a network that is deeply congested, the bristling factor has been increased to 2 by doubling the number of network end nodes connected to routers. Figure 21 (b) shows the performance of the two techniques in the bristled network. As shown, PB has graceful performance degradation as network load increases while Disha sustains only half the maximum throughput under high network loads. This observation confirms that resolving congestion prior to
forming deadlocks is an efficient approach to achieve high network performance and throughput.

5 Conclusion

In this paper, we propose a new technique for efficiently handling network congestion and potential deadlock anomalies. The proposed technique not only precisely identifies paths of congested resources but also actively disperses detected congestion by providing bubbles for blocked packets due to such congestion. This technique is applicable to networks with deadlock recovery-based routing algorithms such as true fully adaptive routing as well as to networks with deadlock avoidance-based routing. In the former case, acceleration buffers implemented in routers are used to create necessary bubbles for the resolution of potential deadlock.

The technique proposed in this paper provides several advantages for system and network designers. First, the technique makes true fully adaptive routing approaches more practical by providing an efficient way to handle network congestion and potential deadlock. Unlike other recovery-based deadlock handling techniques, the proposed Ping & Bubble technique does not cause out-of-order delivery of packets during recovery even when networks are deeply saturated. Second, no single point-of-failure exists as the need for a token mechanism is obviated. Third, the proposed PB technique is able to actively disperse network congestion by precisely controlling the movement of bubbles in the network as opposed to relying on unpredictable stochastic bubble movement as in other schemes. Fourth, congestion dispersion may take place in more than one place simultaneously (i.e., concurrent dispersion), which enables packets to experience less blockage during transmission, thus increasing network performance. Fifth, the PB technique implicitly provides injection-limiting capability to throttle networks such that when the network becomes saturated, new packets are automatically prevented from being injected into the network. If used in conjunction with other congestion handling techniques such as explicit injection-limitation and age-based scheduling, the PB technique can yield even better performance.

In the future, it could be interesting to investigate ways of efficiently reducing the average length of ping traces in large networks. Also, design synthesis of a router hardware prototype that implements the proposed technique can be pursued to better understand the cost/performance tradeoffs in hardware complexity, critical path operations, and improved sustained throughput.

References


