Flow Control Techniques for Multicasting in Gigabit Networks *

X. Chen, L. E. Moser and P. M. Melliar-Smith
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106

Abstract

Communication protocol designers are facing new challenges due to the development of high-speed networks and to the increasing demands of network applications. Gigabit networks pose a challenge to flow control techniques developed for traditional acknowledgment-based, and typically packet-switched, lower-speed networks. Network applications such as distributed databases, groupware and distributed multimedia, all of which require multicasting, have rendered insufficient existing point-to-point flow control methodologies. We examine the problem of flow control for multicasting in gigabit networks. Instead of feedback or rate adaptation techniques, we employ more active flow control mechanisms that use traffic information in protocol control frames. These techniques are very effective for the totally ordered multicast protocol we have developed for the QuickRing gigabit local-area network.

1 Introduction

Emerging communication networks have increasingly higher speeds and greater reliability. Local-area networks (LANs) operating at 100 Mbit/s are now being used daily. Gigabit networks operating at one to multiple Gbit/s are being developed. The change to network architectures based on gigabit LANs urgently challenges us to develop new protocols to realize the great potential of these media. In particular, existing flow control methodologies [2] for ensuring high throughput and low latency in Ethernet and token ring LANS are inadequate for handling traffic in gigabit LANs.

Network applications also drive the development of high-speed networks by demanding increasingly higher bandwidth, real-time latency and latency jitter requirements, and higher Quality of Service (QoS). In addition, there are increasing demands to support multicast group communication in distributed applications. Thus, the expectation of gigabit networks is not only that they push bits to the destinations at higher rates, but that they also meet these demanding new requirements.

Group communication takes the form of one-to-many and many-to-many multicast communication. Providing a multicast service in a gigabit network is the logical way to meet the needs of the applications. However, including a multicast service in a gigabit network design further increases the difficulty and complexity of flow control.

Several gigabit networks have adopted a slotted ring architecture in which the network nodes are connected in a ring and provide separation between ring segments (slots). Traffic can flow simultaneously in non-overlapping segments. This spatial reuse has the potential of providing very high throughput. One slotted ring architecture is the QuickRing gigabit network [13], which provides hardware multicasting.

In [4] we presented a reliable totally ordered multicast protocol for the QuickRing network. Compared to point-to-point unicast communication, multicasts require the use of many more network resources, in particular buffers and links. Reliable totally ordered multicasting, on the other hand, requires that a multicast packet be received at the destinations in an "all or none" atomic fashion. Since flow control in QuickRing must be exercised at the hardware level, this again renders inadequate traditional flow control mechanisms based on point-to-point TCP/IP communication.

In this paper we address flow control issues in gigabit networks, with QuickRing as an example. We analyze the characteristics of multicast traffic and propose various flow control mechanisms, such as packet-level admission control, delayed retry, and dynamic regulation. The main contribution lies in the simplicity of these techniques, which make them suitable for hardware implementation. Using the simulator that we have built, we investigate the relative merits and limitations of the different policies, with the aim of finding the most effective scheme.

*This research was supported by a contract from DARPA N00174-95-K-0083 and by a grant from National Semiconductor Corporation through the State of California MICRO Program 94-094. X. Chen is now with FORE Systems, 5800 Corporate Drive, Pittsburgh, PA 15237.

0-8186-7453-9/96 $05.00 © 1996 IEEE

2 Related Work

Several other existing gigabit networks use a slotted ring architecture, namely ATMR [6, 11], CRMA-II [9, 12] and MetaRing [5]. All of these networks employ a dual counter-rotating ring topology.

Flow control issues in these networks are implicitly addressed as

- Fairness control to prevent giving preference to nodes positioned upstream in the circulation direction and, more importantly, to prevent deadlock, livelock and starvation.
- Admission control to the medium, while providing immediate access under low load, and bounded access otherwise.

ATMR implements flow control using a cycle reset and distributed window mechanism. Each sender on a ring can only send ATM cells up to the number allowed by a window counter value. A sender becomes inactive when its window counter has reached zero or it has no more cells to send. If a sender detects that it is the last active sender on the ring, it initiates a cycle reset by broadcasting a control frame which resets the window counter of every node to a value that is determined safe for flow control.

CRMA-II uses an explicit reservation strategy in successive transmission rounds. A designated scheduler is responsible for collecting buffer reservation requests from, and allocating buffers for, each of the senders. A scheduling algorithm determines a fair allocation based on the number of available buffers, the number each requester has used so far, and the number requested. Fairness is provided by giving preference to sender stations that have smaller transmission counter values, showing fewer buffers used earlier. Access to the medium is regulated by a threshold value calculated by the scheduler at each round. Packets are sent as determined by the scheduler. Those senders having transmission counter values larger than the threshold enter a defer mode and do not send a packet or attempt a reservation in the next round.

MetaRing adopts a quota-based approach, similar to the credit-based ATM flow control mechanism [8]. Each sender periodically receives a transmission quota updated by a central scheduler. Each sender is satisfied with a min-quota representing the minimum number of bytes a node can send, and is regulated by a max-quota representing the maximum allowable. The nodes on a ring transmit data as they successively enter and leave the sending state. A sender leaves the sending state when there are no more packets to send or when it has reached the max-quota. The immediate downstream node then enters the sending state.

In all of these networks, active flow control is used by anticipating congestion and by throttling the transmission at a threshold. Conservative flow control that relies on feedback or rate-based adaptation techniques is regarded as ineffective and in some cases a complete failure for high-speed networks [7].

Flow control mechanisms for high-speed networks, including the ATM rate-based [10] and credit-based flow control mechanisms [8], are implemented in communication hardware, such as the data-link controller, ATM switch, or ATM adapter card. In contrast, traditional flow control functions reside in software. The complexity of any flow control mechanism developed for gigabit local-area networks must be feasible for hardware implementation.

3 Network Model

In our investigations, we have considered gigabit networks in general and have focused on the QuickRing network in particular. The QuickRing data-link controller chip, being manufactured by National Semiconductor Corporation, currently uses copper interconnects and is being extended to fiber optics. The data rate for each ring segment is 1.6 Gbit/s. With spatial reuse of the link bandwidth, the aggregate bandwidth can be increased to many times the data rate of one ring segment.

The QuickRing network goes beyond the other three networks mentioned above in that multiple-ring topologies are supported. Multiple rings are interconnected by bridges, which consist of two controllers connected back-to-back. QuickRing uses unidirectional message passing on a ring, while bridge nodes can pass messages in both directions.

Like other gigabit networks, the physical medium of QuickRing is very reliable, especially compared to traditional shared-medium networks, such as the Ethernet and token ring. With a single bit error rate of less than 10^{-11}, and the hardware capability to detect and correct a single bit error, even with Gbit/s transmission, the mean time to one corrupted packet payload or control frame is one million years. Thus, we assume that the only cause of packet loss is buffer overflow at the destinations.

The high bandwidth of gigabit LANs and the intrinsic connectivity provided by a ring architecture suffices for all-to-all multicasting in these networks. We consider both one-to-all and all-to-all multicasts.

4 The QuickRing Gigabit Multicast Protocol

For the QuickRing network architecture, we have developed a multicast protocol that provides a reliable totally ordered multicast service. The main features of the protocol are highlighted below. The format and examples of the packet and control frames used in the protocol are shown in Figure 1. Further details of the protocol can be found in [3, 4], including proofs of correctness.

Based on the premise that the only cause of packet loss is buffer overflow, a buffer reservation strategy is used to ensure packet delivery. This strategy requires a sender to multicast a ticket first to reserve buffers at all of the intended destinations and intermediate bridges in the multicast tree before it multicasts the packet. Unlike the CRMA-II ring, which uses a centralized scheduler to reserve buffers on behalf of each requester in each round, buffer reservations in QuickRing proceed in a decentralized or symmetric fashion.

When a node has a packet to send, the node first transmits a voucher followed by a ticket around the local ring. The voucher is targeted to bridges only and is quickly forwarded down the multicast tree to each of the bridges, while the ticket is for both bridges and local destination nodes. Upon receiving the ticket, each group member on the ring reserves a buffer for the ticket depending on availability of a buffer. When the voucher arrives at a bridge, the bridge takes responsibility for reserving buffers for the corresponding ticket on the ring across the bridge by sending a ticket around that ring. Buffer reservations for the member nodes in the multicast group thus proceed hierarchically, using a breadth-first-search traversal of the multicast tree.

The ticket collects buffer information on its way back to the source. Each group member at which a buffer is reserved unsets a designated map bit in the corresponding position of the ticket. The bridge holds onto a ticket it received from the local ring until the ticket that was pushed down the multicast tree has been pushed back up. If the map bits in the pushed up ticket are all unset, the bridge then unsets its map bit in the ticket and forwards the ticket on its local ring. In this manner, the buffer reservation information is convergecast back to the source.

If the buffer reservation is successful, a buffer is reserved at each of the destinations and intermediate bridges and the source multicasts the packet. If the buffer reservation fails, the source retries until it finally succeeds.

Such a naive reservation strategy can lead to deadlock, livelock and starvation when buffers are reserved. Thus, a reservation retract mechanism is employed. Individual reservations are prioritized in a total order, which is established using voucher timestamps and unique source identifiers (UIDs). The voucher timestamp (VTS) is taken from the source's Lamport (logical) clock when the higher layer at the source supplies the packet. When a ticket finds that no buffers are available at a destination, it requests a retract of the buffer reservation for the ticket with the lowest priority (highest VTS/UID) among all of the tickets currently reserving a buffer at this node. The retract request is then sent back to the source. If the source has not multicast the packet, it then multicasts the retract to cancel the reservations previously made for this packet at the nodes in

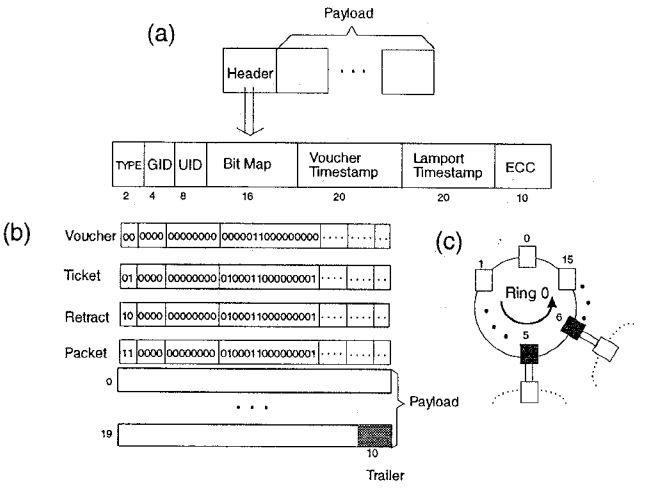

Figure 1: The format and examples of the packet and control frames used in the protocol. (a) A packet formed by a header (10 bytes) and payload (200 bytes), and the expanded view showing the various fields and field lengths (in bits) in a header (GID: group identifier, UID: unique source identifier, ECC: error correcting code). A voucher, ticket, or retract contains only the header, in the same format as the packet header. (b) Examples of a voucher, ticket, retract, and packet sent from source node 0 to its multicast group members in the network shown in (c). (c) The group spanning tree with group identifier 0 is rooted at node 0. Nodes 1 and 15 on ring 0 are also in the group. The two bridge nodes 5 and 6 on ring 0 are contained in the spanning tree.

the multicast tree. This guarantees that deadlock, livelock and starvation do not occur.

Packets are totally ordered using Lamport timestamps and unique source identifiers. The Lamport timestamp (LTS) of a packet is distinct from the voucher timestamp (VTS). Like the VTS, the LTS is obtained from the source's Lamport (logical) clock but, unlike the VTS, the LTS of a packet is not assigned until the packet is ready to be multicast (*i.e.*, the buffer reservations for that packet have been completed). This scheme maintains Lamport's causality relationships.

5 Performance Metrics

We have developed a discrete-event simulator to measure the performance of the QuickRing totally ordered multicast protocol, and have investigated the effectiveness of the various flow control mechanisms described below. The simulator is written in the C programming language and is interfaced to Tcl/Tk for visualization.

The effectiveness of the flow control mechanisms is measured by several performance metrics:

- *packet delivery throughput:* the total number of packets sent by all sources

- *latency:* the *reservation latency* (the interval between the time a packet arrives from the sender's host and the time the packet is finally authorized for multicast) and the *queueing latency* (the aggregate queueing time due to network link contention between the time a packet is authorized to be multicast until it arrives at all of the destinations)
- *control frame complexity:* the mean number of buffer reservation attempts per packet and the mean number of retracts generated per packet
- *network link utilization:* the mean percentage of time that a network segment has traffic (packets or control frames) in flight.

With effective flow control, the packet delivery throughput should be stabilized at high offered loads, while at low offered loads the throughput should not be limited due to flow control. In addition, the stabilized throughput should be close to the physical saturation limit corresponding to 100% link utilization. The latency is at least as great as the ticket propagation delay required to complete all buffer reservations. The mean latency related to reservation retry and retract, if not regulated by flow control, can be rather significant. An effective method of flow control should keep the mean control frame complexity of the protocol at a mininum without compromising throughput. Although necessary for the protocol, control frames consume network bandwidth. Therefore, reducing the control frame complexity is another way of stabilizing throughput.

6 Flow Control Mechanisms

Our flow control mechanisms aim to achieve throughput close to the saturation limit at high offered loads, while preventing thrashing at high offered loads and accepting all traffic at low offered loads. They also aim to limit the network latency for message delivery even at high offered loads, so that real-time service requirements can be met.

We have developed the following flow control mechanisms (the indices below will be used hereafter to reference each flow control policy):

1. Admission control on packets from the higher layers
   (a) Limit the size of the sender packet queue
   (b) Dynamically regulate the size of the sender packet queue
2. Reservation attempt rate control at the senders
   (a) Defer the initial buffer reservation attempt and retry
   (b) Exponentially back off the ticket/voucher retransmissions
3. Bridge flow control
   (a) Defer pushing down vouchers for retry
   (b) Exponentially back-off the voucher retransmissions at the bridges
4. Bandwidth optimization and bottleneck avoidance
   (a) Circular shifting
   (b) Providing more buffers at the bridges
   (c) Merging retract requests.

The task of policing the behavior of an individual sender must be provided at the medium access control layer, and must not be relegated to a higher layer. We have developed a packet-level admission control mechanism and delayed retry mechanism for this purpose.

6.1 Packet-Level Admission Control

Admission control is employed to deny access to the network, depending on the underlying network traffic load. In QuickRing, a FIFO sender packet queue holds the packets for which a buffer reservation is currently being made. Admission control sets the size of this queue according to the following heuristics.

Suppose a multicast group consists of m members, where member node i has B_i buffers and the size of its sender packet queue is q. The maximum total number of outstanding (unfullfilled) reservation requests that each member can receive is $q \times (m-1)$ for all-to-all multicast, and is q for one-to-all multicast, during any given period. To avoid retry and retract, which wastes network bandwidth, the total number of such requests that node i can handle should not exceed the number B_i of buffers. Letting $B = \min\{B_1, B_2, ..., B_m\}$, this requires

$$q \times (m-1) \leq B$$

for all-to-all multicast or

$$q \leq B$$

for one-to-all multicast.

The above two inequalities are the conditions that q must satisfy upon initialization. Note that these two conditions do not guarantee that there are no retries or retracts. The above inequalities assume that a buffer at a receiver is freed when a packet is received but, in fact, the packet may have to be buffered at a receiver until it can be delivered.

A fixed size for the sender packet queue may be impractical, since multicast traffic patterns can change from all-to-all to one-to-all and vice versa, and the traffic rate can fluctuate. An admission control algorithm must adapt to these changes. How then does a sender detect and adapt to these changes?

```
FLOW CONTROL PARAMETERS:
    q_i: sender packet queue size at node i;
    p_i: total number of packets to be sent at node i
        (including those pending at the higher layer);
    p_qi: number of packets in the sender packet queue at node i;
    O_i: buffer occupancy at node i;
    U_i: number of different UIDs in the tickets received by node i;
INITIALIZATION:
    q_i := B/(m - 1);
    p_i := 0;
    p_qi := 0;
    O_i := 0;
    U_i := 0;
WHILE true DO
    Update p_i, p_qi, O_i and U_i;
    IF p_i > q_i THEN
        SWITCH (U_i, O_i)
            CASE U_i = 0          // one-to-all multicast
                q_i := min{p_i, B};
            CASE U_i > 1          // all-to-all multicast
                IF q_i > B/(m - 1) THEN q_i = q_i - 1;
            CASE U_i = 1 AND O_i > B/2  // two active senders
                IF q_i > B/2 THEN q_i = q_i - 1;
            CASE U_i = 1 AND O_i < B/2  // other sender is not active
                q_i := min{p_i, B/2};
        ENDSWITCH
    ENDIF
    // Adaptation:
    IF failed a reservation attempt THEN
        mark the packet that fails as retrying;
        q_i := q_i - 1;
            // if the sender packet queue is full, it will be reduced to
            // the new q_i value when a packet is dispatched
    ENDIF
    IF received a retract request THEN
        mark the packet that is requested as retracting;
        q_i := q_i - 1;
            // if the sender packet queue is full, it will be reduced to
            // the new q_i value when a packet is dispatched
    ENDIF
    IF the packet marked retrying or retracting is sent THEN
        q_i := q_i + 1;
    ENDIF
ENDWHILE
```

Figure 2: Adaptive admission control heuristics.

- The occupancy of the receiver buffers can provide information about the activity level and the nodes that are multicasting. The total number of reserved buffers indicates the activity level of the group multicast. Each reserved buffer for a multicast contains a ticket with a UID indicating the source. A single UID differentiates the one-to-all multicast from the all-to-all multicast.

- The first occurrence of a retry or retract after a long period of time signals network congestion. At the same time, it implies that the sender should slow down in accepting new packets from its host. As network congestion is resolved, the sender can resume its normal rate of accepting new packets.

In our admission control heuristics, a sender adapts to these changes, as shown in Figure 2.

Setting the upper bound and dynamically regulating the size of the sender packet queue is somewhat similar to the rate-based flow control of ATM networks [10]. It forces each sender to restrain itself at a rate that does not cause network congestion by frequent retries/retracts, while allowing high utilization of an available resource.

By observing and reacting to the retry and retract events, as well as the buffer utilization (multicast) patterns, the flow control can be dynamically adjusted to respond and adapt to the network traffic.

6.2 Delayed Initial Reservation Attempt and Delayed Retry

This is similar to delaying data retransmission in the presence of collision in CSMA/CD networks. Buffer reservation retry is delayed after a failed attempt or after retracting all of the previous reservations for a packet. While the delay in CSMA/CD is necessary to avoid further collisions, the delay for retry is necessary for preserving network bandwidth and for preventing the holding of buffers at some node when the probability of overall success of buffer reservation is small. On the other hand, excessive delay should be avoided because it might underload the network and introduce unnecessary latency.

The delayed retry algorithm involves two aspects:

- The delay Δ in the initial buffer reservation attempt for a packet that has just entered the sender packet queue when higher priority packets are already being delayed, or the sender's receiver packet queue is fully used or reserved by other senders, indicating that an immediate attempt would most likely fail. The delay Δ is a flow control parameter.

- An exponential back-off strategy in which delays in the second and subsequent retries are increased by backing off the retransmissions exponentially. The retransmission delay after n failures is

$$\Delta_n = \Delta_0 e^{\delta n}$$

where Δ_0 and δ are critical parameters.

Our delayed retry algorithm is similar to the leaky bucket flow control scheme [1]. All of the delayed tickets and vouchers enter a delay priority queue (priority is determined by the VTS/UID pairs). A permit is generated periodically to allow the highest priority ticket/voucher to enter the transmission queue for retransmission. The period of permit generation is set to a value approximately equal to the packet interarrival time under the current network offered load, since the packet interarrival time closely represents the duration over which some destinations continue to have full receiver packet queues.

6.3 Refining the Flow Control Mechanisms

We now consider several second order effects, beyond the primary characteristics of gigabit network multicast traffic,

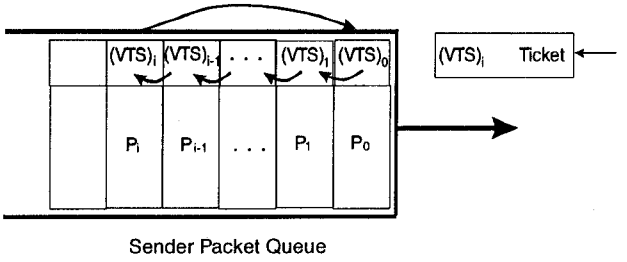

Figure 3: Circular shifting of voucher timestamps occurs when a ticket for packet P_i, rather than for P_0, returns with all of the map bits unset. The shifting of the VTS proceeds in the direction of the arrows.

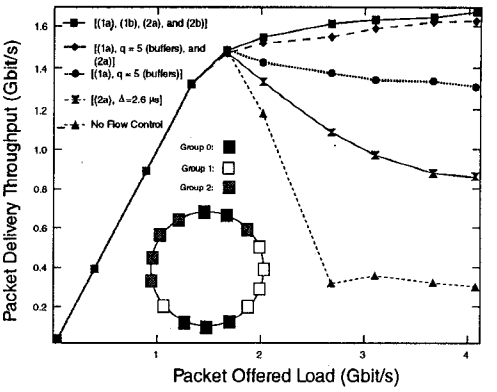

Figure 4: Effective flow control is demonstrated in this example with a single ring, consisting of three multicast groups. Each group contains five members as shown in an all-to-all multicast scenario. There are 20 receiver buffers at each node. The performance curves show the packet delivery throughput vs. packet offered load for different flow control policies. The indices in the legend indicate the policies at the beginning of Section 6 that are being used. In all cases, circular shifting (4a) is used.

which mandate flow control via admission control and delayed retry policies. These second order effects do not occur very frequently, but their handling optimizes flow control and streamlines the protocol.

In the QuickRing protocol, a node attempts to reserve buffers for all of the packets in its sender packet queue at about the same time. The FIFO order of the queue determines the priority of packets, and tickets are multicast in this FIFO order. However, it is possible that a packet P_i of lower priority completes its buffer reservation before the packet P_0 at the head of the sender packet queue. To comply with the FIFO order, the packet P_i with the completed buffer reservation cannot be multicast. Rather, the head packet P_0 is now multicast and stored in the buffer reserved for P_i.

Since the receiver of a packet locates the buffer by matching the VTS and UID for the packet and the ticket in the buffer, the VTS for P_0 must be circularly shifted with the VTS for P_i before P_0 is sent. This circular shifting scheme is shown in Figure 3. Compared to a simple interchange between P_i and P_0, circular shifting preserves the relative reservation priority between undelivered packets, *i.e.*, the VTS of P_j is still smaller than the VTS of P_i. Also, the priority of the first packet in the sender packet queue, among all of the undelivered packets in the network, is maintained.

This circular shifting scheme provides an optimization that reduces the withholding of network resources, while allowing the protocol to meet the total order requirement. Circularly shifting the VTS poses no harm to the total order, since the order of the packets is determined by the LTS, rather than the VTS.

The second refinement provides more buffers at the bridge nodes to avoid a possible bottleneck when multiple multicast groups use the same bridge node for routing along their multicast trees.

The third refinement merges multiple retract requests for the same packet. When a retract request is forwarded around the ring to the source, each node within the same multicast group determines if it has already sent the same retract request. If so, the retract request is discarded without being forwarded further. This effectively optimizes the bandwidth.

6.4 Bridge Flow Control

Bridge nodes are responsible for initiating buffer reservations in the lower portions of the multicast tree on behalf of the source node. If a buffer reservation at a lower-level node of the tree fails or is retracted, the bridge node is responsible for retrying at a later time. Thus, the bridge nodes exercise similar delayed retry and exponential back-off policies as the source node. However, the delay parameters Δ_B and δ_B with which they operate are different.

The admission control policies implemented in the bridge nodes are not so explicit, since a packet is admitted to initiate a buffer reservation beyond the bridge when a buffer can be reserved at the bridge.

7 Performance Comparison

We now compare the performance of the above flow control schemes. The comparison is organized as follows:

- Single-ring vs. multiple-ring topology
- All-to-all multicast vs. one-to-all multicast
- Poisson vs. uniform vs. bursty traffic
- Large number of small groups vs. small number of large groups.

7.1 Single-Ring Flow Control

We consider a typical QuickRing network consisting of 16 nodes. The 16 nodes are equally partitioned into 1, 2, 3, 5 and 8 groups. The packet delivery throughput curves for the optimal flow control setting, which stabilizes the performance at high offered loads, are shown in Figure 4. In this example, for both Poisson and uniform distributions of packet arrivals, we compare the effectiveness of the different flow control policies. The different arrival distributions do not show appreciable difference in results. Thus, we choose to show the results for a uniform distribution. Without flow control, the network thrashes at high offered loads. Simply delaying the buffer reservation retry alleviates the situation, but does not suffice to stabilize throughput. Delayed retry together with the exponential back-off strategy further improves the throughput at high offered loads. Limiting the sender packet queue size q to only five buffers (which satisfies $q \times (m-1) \leq B$ for $m = 5$ and $B = 20$) and delaying the retry works better, but combining all of the proposed strategies renders the best performance for a single ring.

The appropriate setting of the sender packet queue size q varies with the one-to-all and all-to-all multicast patterns, as well as with the offered load. By setting q to be greater than one at low offered loads, we achieve a pipeline effect in reserving buffers for multiple packets, which increases the throughput and decreases the latency. A low value of q also protects the network from thrashing during a burst of high offered load. The most appropriate value of q is thus a compromise. In Figure 5, for 1, 3 and 5 groups, the variations of performance for different sender packet queue sizes q are shown, while the receiver packet queue size is fixed at 20 buffers.

Figure 5a clearly indicates that $q = 2$ is the best choice, which also satisfies $q \times (m-1) \leq B$, while $q = 1$ underloads the network, and $q > 2$ starts to cause thrashing as the network becomes saturated. In Figures 5b and 5c, as the number of groups increases, and the number of multicast senders for each group decreases, $q \geq 4$ and $q \geq 6$, respectively, do not underload the network nor cause thrashing. From Figure 5, it appears that $B/(m-1)$ is a good choice for initializing q to avoid underloading the network at low offered loads, although the performance increase over $q = 2$ is quite small.

However, at high offered loads or during a burst in the offered load, too large a value of q causes thrashing. Figure 6 shows this effect by comparing the different values of q for a burst of high offered load. In all three cases, for a burst in offered load of 4 Gbit/s over a 5 ms period (equivalent to 2 Megabytes of multicast data), lower values of q work the best. Larger values of q result in too many reservation attempts with a consequent lower probability of success in reserving buffers in the

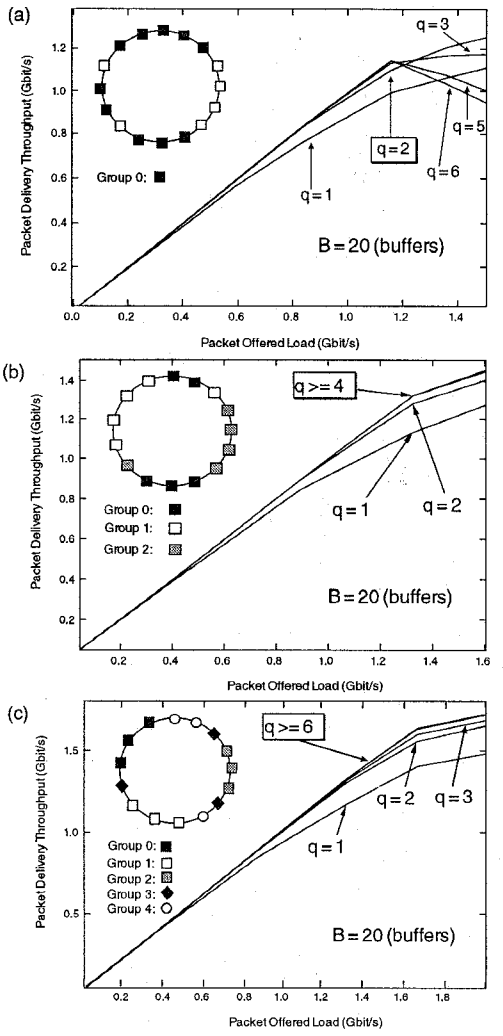

Figure 5: Determining the optimal size of the sender packet queue for different all-to-all multicast scenarios: (a) one group containing 10 members, (b) three groups containing five members each, and (c) five groups containing three members each. The q value attached to each curve is the sender packet queue size. The flow control methods include limiting the sender packet queue size and delaying reservation retries. In (b) the curve for $q = 3$ is very close to that for $q = 4$ and thus is omitted and, similarly, in (c) for $q = 4$ and $q = 5$. The boxed q values are the optimal choices that do not underload the network below saturation.

first attempt. The subsequent retries consume network bandwidth and decrease the effective throughput. Since a fixed value of q does not work well for all offered loads, adaptiveness in the flow control heuristics is needed.

For one-to-all multicast, flow control is easier to achieve and a simple strategy with a suitable size of the sender packet queue and delayed retry mechanism works quite well. In Figure 7a, setting the sender packet queue size to one clearly underloads the network and introduces unnessary latency. The gradual increase in the size of the sender

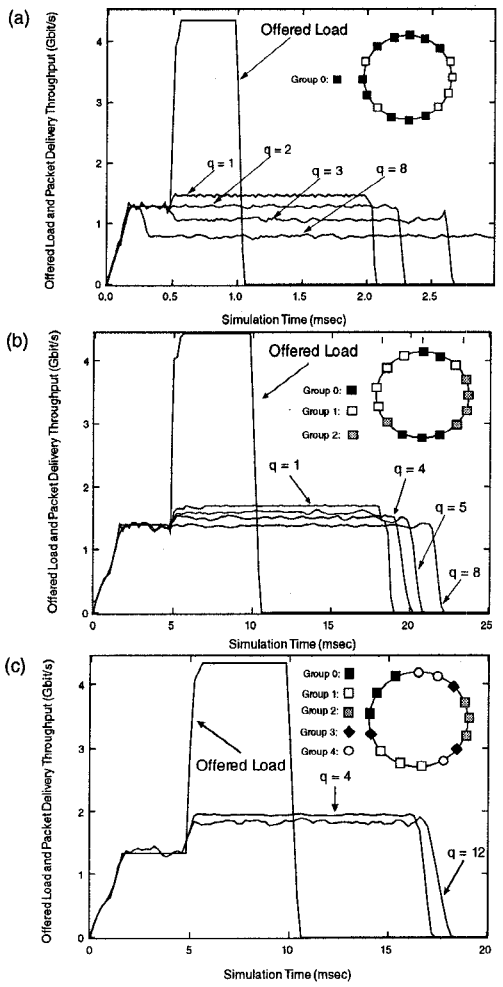

Figure 6: Comparison of the effectiveness of flow control with different sender packet queue sizes q. (a) One multicast group with ten members, (b) three multicast groups with five members each, and (c) five multicast groups with three members each. For clarity, only two throughput curves are shown in (c). The curve for $q = 1$ is very close to that for $q = 4$ and the curves for q between 4 and 12 lie between the two curves shown.

packet queue allows the sender to exploit the network bandwidth, thus achieving the saturation throughput. But once the sender packet queue size is large enough, as Figures 7a and 7b show, increasing q does not further increase the throughput nor reduce the latency. For all offered loads up to saturation, $q = 3$ suffices.

Since the costs of different flow control policies vary, we must also consider their relative significance. Comparing the flow control policies defined in Section 6, policies (1b) and (2b) are considered to be an enhancement of policies (1a) and (2a), and require a slightly more complicated implementation. Thus, the relative merit should be considered.

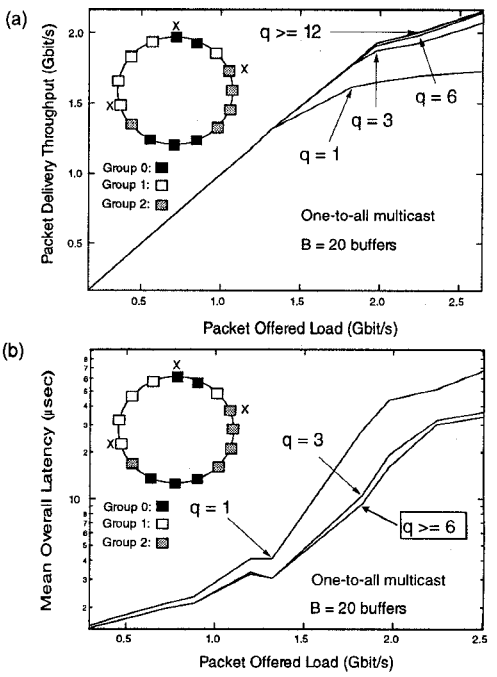

Figure 7: (a) Packet delivery throughput in one-to-all multicast with different sender packet queue sizes q. The nodes marked X are the senders. (b) Mean latency vs. packet offered load for different sender packet queue sizes q. In each of these scenarios, there are three multicast groups with five members each.

Figure 5 shows that properly sizing the sender packet queue and delaying the buffer reservation retry must be adopted to stabilize throughput and latency. Figures 5, 6 and 7 suggest that a fixed sender packet queue size does not work well at both high and low offered loads, for both one-to-all and all-to-all multicasts, and for different size group memberships. Consequently, we have investigated adaptive admission control heuristics.

For a single ring of 15 nodes partitioned into three multicast groups, the packet offered load was modulated from low to high, as shown in Figure 8a with alternate increasing and decreasing numbers of senders between intervals. Figures 8b, 8c and 8d also show the throughput, the mean number of packets in the sender packet queue, and the sender packet queue limit, as set by our heuristics. Figure 8d shows that our admission control heuristics quickly adjust the sender packet queue limit to changes in multicast type and offered load, with changes in the actual sender packet queue length, as shown in Figure 8c. Combining these admission control heuristics with the delayed retry and backoff mechanisms, the throughput is stabilized, as shown in Figure 8b. The throughput remains stable even though the network changes abruptly from one-to-all multicast to all-to-all multicast, while the offered load increases.

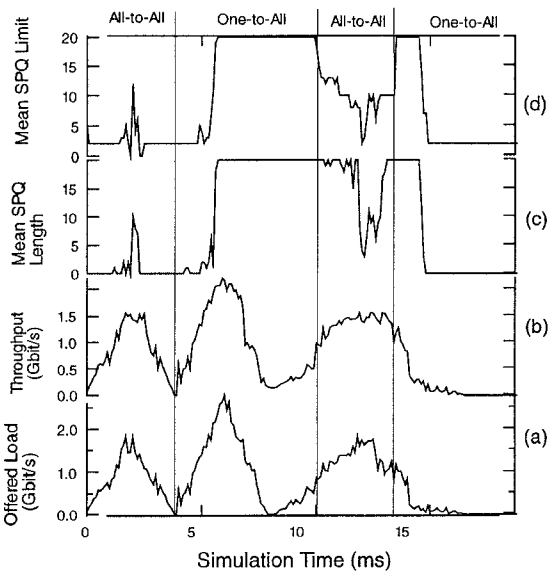

Figure 8: Adaptiveness of flow control to network offered load, one-to-all and all-to-all multicasts: (a) offered load, (b) throughput, (c) mean sender packet queue (SPQ) length (buffer occupancy), and (d) mean sender packet queue (SPQ) limit (corresponds to q_i in the admission control heuristics), traced for 16 ms of simulation time. Each data point on the trace is sampled in each of the successive 100 μs intervals. The jitter of the curves is attributed to statistical noise due to sampling. Within the 20 ms, the network traffic alternates between all-to-all multicasts and one-to-all multicasts, as indicated at the top, with on-ramp and off-ramp load changes as shown in (a).

On the other hand, the exponential back-off strategy of flow control policies (2b) and (3b) provides a remedy when the initial choice of retry delay does not work well. Even though we only further delay the retry for one ticket that fails to reserve buffers, or must retract its reservations, other tickets of lower priority from the same sender are also delayed. However, for the exponential back-off strategy, we see from Figure 4 only a marginal improvement in the throughput and latency, which are already stabilized or limited by admission control and delayed retry at high loads. Since admission control and delayed retry work so well, the probability of more than one attempt of retries is quite small and the exponential back-off strategy is less critical.

7.2 Multiple-Ring/Bridge Flow Control

A multiple-ring network has a larger network diameter and possibly encompasses a larger multicast group than a single-ring network. We have investigated the effectiveness of our flow control techniques for various multiple-ring topologies. We present the results for a few of these topologies in Figure 9.

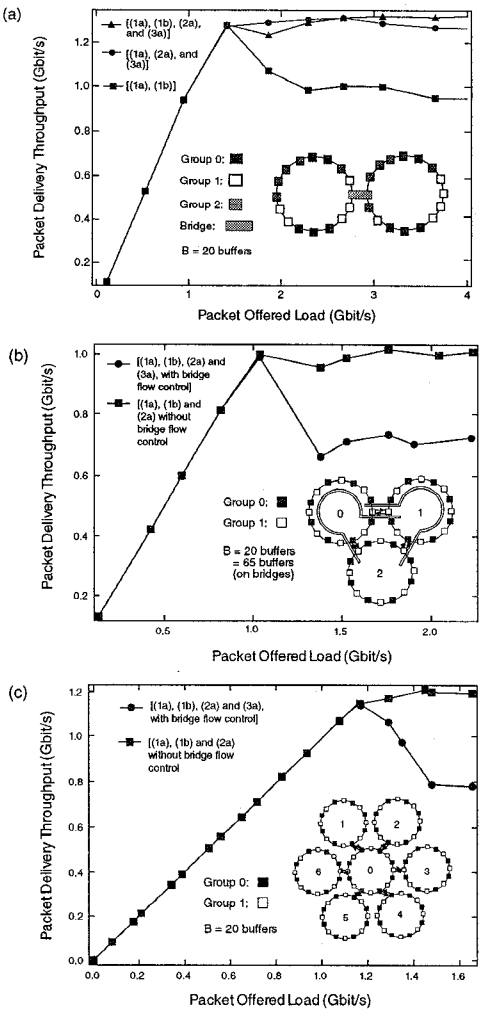

Figure 9: Effective flow control for multiple-ring networks. (a) A two-ring network with three multicast groups, where each group contains 10 members spanning the two rings shown in an all-to-all multicast, (b) a three-ring network with disjoint multicast spanning trees as shown by the two thicker curves, and (c) a seven-ring network with the multicast senders located on the center ring. The performance curves show the packet delivery throughput vs. packet offered load, indicating the effectiveness of bridge flow control.

In multiple-ring networks, both the sender and the intermediate bridge nodes must exercise flow control. Since a bridge implemented in hardware is simple (a QuickRing bridge is nothing more that two controllers connected back-to-back), the flow control scheme is also required to be simple. Figure 9 shows the importance and effectiveness of flow control at the bridges. Combining bridge flow control with single-ring flow control at the sender is effective in stabilizing the throughput and reducing the latency under all types of loads for the three multiple-ring topologies shown.

8 Conclusion

We have presented simple and effective flow control techniques for multicasting in gigabit networks that achieve high throughput and low latency. Packet-level admission control effectively throttles the packet offered load and, together with a delayed retry mechanism, reduces the burstiness of the traffic and smoothes the traffic offered by the higher network layers. Moreover, admission control heuristics allow each sender to withhold sending when congestion occurs and to increase sending when the traffic load is low. These heuristics can adapt rapidly to various traffic loads and patterns.

Gigabit data rates render feedback-based flow control ineffective. The latency due to feedback, coupled with the gigabit data rate, creates a significant buffering requirement for the network as a whole. An approach based on monitoring the failure rate (the retract or retry rate) is also inadequate, since there is no appropriate sampling period for both bursty and regular traffic. The latency due to monitoring presents similar problems to those encountered with feedback. In contrast, our approach uses individual retry and retract requests to trigger flow control. The response is thus more immediate and effective.

Reliable totally ordered multicasting raises an additional challenge. Each multicast packet sent incurs substantial costs in the use of network resources (buffering, link bandwidth, and time). One unprudently sent multicast packet results in a higher penalty than one point-to-point packet when the network is congested. Our admission control and delayed retry policies work quite well in meeting this challenge.

References

[1] M. Butto, E. Cavallero and A. Tonietti, "Effectiveness of the 'leaky bucket' policing mechanism in ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. SAC-9, no. 3 (April 1991), pp. 335-342.

[2] B. Bux and D. Grillo, "Flow control in local-area networks of interconnected token rings," *IEEE Transactions on Communications*, vol. COM-33, no. 10 (October 1985), pp. 1058-1066.

[3] X. Chen, *Gigabit Network Multicast Protocols*, Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of California, Santa Barbara (June 1996).

[4] X. Chen, L. E. Moser and P. M. Melliar-Smith, "Reservation-based totally ordered multicasting," *Proceedings of 16th IEEE International Conference on Distributed Computing Systems*, Hong Kong, (May 1996), pp. 511-519.

[5] I. Cidon and Y. Ofek, "MetaRing - A full-duplex ring with fairness and spatial reuse," *Proceedings of IEEE INFOCOM*, San Francisco, CA (1990), pp. 969-981.

[6] H. Kasahara, K. Imai, N. Morita and T. Ito, "Distributed ATM ring-based switching architecture for MAN and B-ISDN access networks," *Proceedings of 1st IFIP Conference on Broadband Communication*, Estoril, Portugal (1992), pp. 89-99.

[7] H. Kim and D. J. Farber, "The failure of conservative congestion control in high-speed networks," in *Report and Discussion of IEEE Computer Society Technical Committee on Gigabit Network Gigabit Networking Workshop 1995*, *IEEE Network*, vol. 9, no. 4 (July/August 1995), pp. 9-21.

[8] H. T. Kung, "New flow control methods for high speed networks," *Proceedings of 2nd International Symposium on High Performance Distributed Computing*, Spokane, WA (July 1993), pp. 4.

[9] W. W. Lemppenau, H. R. van As and H. R. Schindler, "A 2.4 Gbit/s ATM implementation of the CRMA-II dual-ring LAN and MAN," *Proceedings of Eleventh Annual Conference on European Fibre Optic Communications and Networks*, The Hague, Netherlands (June/July 1993), pp. 274-281.

[10] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda and H. Miyahara, "Rate-based congestion control for ATM networks," *Proceedings of ACM SIGCOMM, Computer Communication Review*, vol. 25, no. 2 (April 1995), pp. 60-72.

[11] T. Okada, H. Ohnishi and N. Morita, "Traffic control in asynchronous transfer mode," *IEEE Communications Magazine*, vol. 29, no. 9 (September 1991), pp. 58-62.

[12] H. R. van As, W. W. Lemppenau, P. Zafiropulo and E. A. Zurfluh, "CRMA-II: A Gbit/s MAC protocol for ring and bus networks with immediate access capability," *Proceedings of Ninth Annual European Fibre Optic Communications and Local Area Network Conference*, London, UK (June 1991), pp. 262-277.

[13] M. Valerio, L. E. Moser, P. M. Melliar-Smith and P. Sweazey, "The QuickRing network," *Proceedings of 1994 ACM Computer Science Conference*, Phoenix, AZ (March 1994), pp. 233-240.