Chapter Eight
Flow Control

This dissertation addresses the design of high performance communication switches for scalable multicomputers. Previous chapters focused on the buffer architecture as being a critical functional unit for such a design. This chapter focuses on *flow control*. Flow control is the method used to regulate traffic in the network. It prevents packets from running over each other and controls how fast each advances through the network [Dall90a].

The primary task of a flow control mechanism is to prevent *uncontrolled* packet loss or corruption due to contention for network resources. Such contention may be handled by blocking, rerouting, or discarding and retransmitting packets. This chapter focuses on the potential of flow control mechanisms to optimize network performance.

Previous chapters of this dissertation examined the behavior of multistage interconnection networks transmitting uniformly-distributed traffic. However, a uniform traffic distribution cannot be guaranteed in the communication network of a multicomputer. The scalable computer system that we envision consists of hundreds or thousands of computing nodes linked together by a high throughput, low latency communication network. Multiple applications will execute on this system in a dynamic fashion; applications and processes will be spawned independently of one another, requiring system resources to be dynamically re-allocated to accommodate them. Due to the applications which will execute on this system exploiting fine-grained parallelism, and the computing nodes in the system

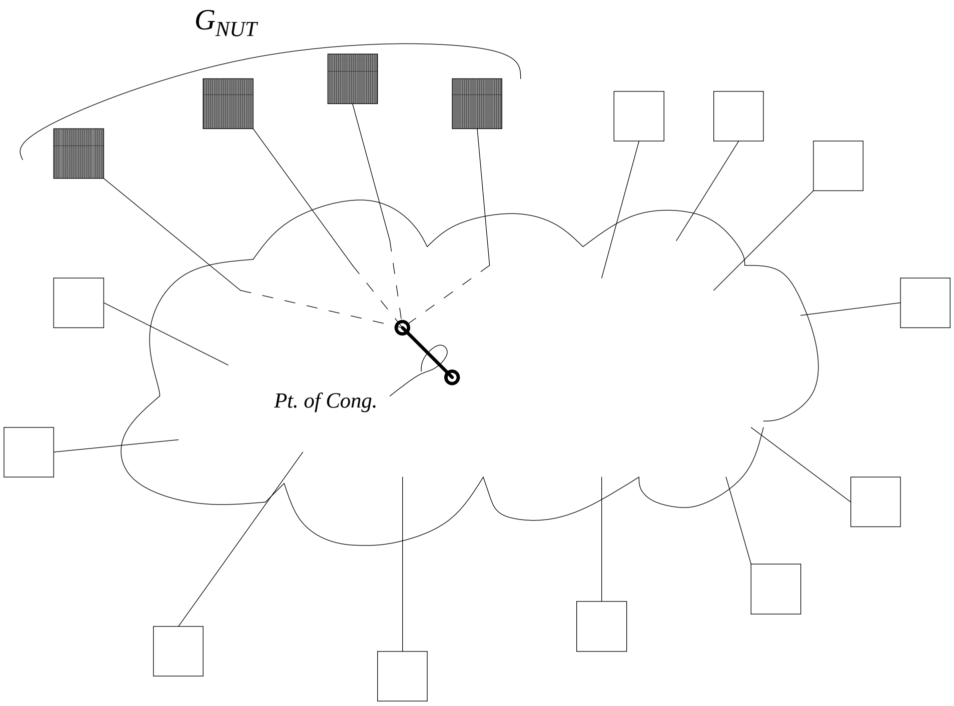


Figure 8.1: *Network Congestion.* A group of one or more senders (G_{NUT}) overload link(s) in the network (the *point(s) of congestion*) for some period of time.

being multitasking and / or multithreading, there will be a large number of packets traversing the network at any given point in time. In a system such as this, one cannot guarantee that communication traffic will remain evenly distributed or that there will be no attempt to over-utilize a network resource. Even if a single application can be analyzed to the extent that it is known to not overload any links within the network, contention between it and other applications for system resources cannot be predicted.

Figure 8.1 depicts such a situation. A group of nodes labeled G_{NUT} (the *non-uniform traffic* group) are transmitting packets with a non-uniform traffic pattern and at a throughput such that a *point of congestion* is created within the communication network (a point of congestion is a communication link whose applied load is higher than its bandwidth).

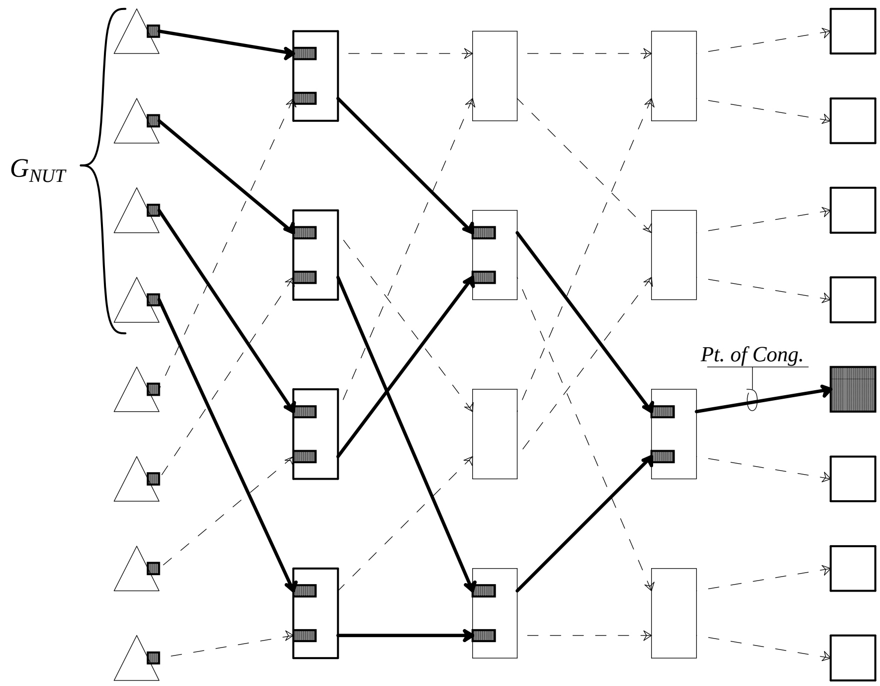


Figure 8.2: *Tree Saturation.* The senders in G_{NUT} overload a link in the network. The congestion propagates back to all senders (*tree saturation*) — including those senders not in G_{NUT}.

When a link is congested, the buffers of the switch which feeds the link become filled with packets waiting to traverse the link. When the buffers become full, the links which supply those buffers become congested, and the process may recur. We term this phenomena *congestion propagation*. When this occurs in a multistage interconnection network (MIN), and is due to the existence of a *hot spot* (a destination address which receives a higher percentage of packets than the other destinations in the network), then it is called *tree saturation* [Pfis85a]. In Fig. 8.2, the group of senders G_{NUT} has created a hot spot. As congestion propagates back from the point of congestion (the terminal link which connects the network to the hot destination), it forms a tree whose leaves include all of the senders attached to the network. This includes those senders which are not members of G_{NUT}. While these senders may be transmitting no packets to the hot destination, the congestion has propagated to links which are on the path from these senders to other destinations.

Our focus in this chapter is on the ability of flow control to increase the performance of communication networks in the presence of congestion by reducing congestion propagation. In the example in Fig. 8.1, the point of congestion is the bottleneck for the communication performance of the members of G_{NUT}. Flow control cannot increase the link's bandwidth; without some mechanism for eliminating the congested link as a heavily-used resource for the members of G_{NUT}, the bandwidth of the point of congestion is a ceiling for G_{NUT}'s performance[†]. The key to improving network performance via flow control is to identify the senders which are ''causing'' congestion, and to limit the rate at which those senders can introduce packets to the network. This prevents the remaining senders from having their performance impacted by the congestion propagation.

For a flow control mechanism to prevent congestion from propagating, it must be able to react quickly to the creation of congestion. When a link becomes congested, packets contesting for the link are blocked and must be stored in the

† There are mechanisms which do address contention prevention or resolution. Examples of this are dynamic routing (packet mis-routing in MINs), hot spot detection and elimination at compile time, tree-structured barriers, etc. Some of these mechanisms may utilize flow control in some way — dynamic routing, for example, may use flow control to determine when to use an alternate route for a packet. This interaction is beyond the scope of this dissertation, but is possible future work.

switch's buffers. Since the packet buffers of multicomputer communication network switches will be small (Ch. 4), they will quickly become filled with packets contesting for the congested link. Kumar and Pfister, examining hot spots and tree saturation in MINs [Kuma86], determined not only that tree saturation occurs quickly after the onset of a hot spot (i.e. congestion will propagate quickly), but also that the tree saturation dissipates slowly once the hot spot has ended. Their results indicate that a quick reaction by the flow control mechanism is necessary to prevent congestion propagation and that failure to prevent the propagation will result in network performance degradation that outlives the cause of the congestion.

This chapter details our search for a flow control mechanism which (1) is scalable, (2) reacts quickly to the onset of congestion, (3) protects the performance of senders which are not ‘‘causing’’ congestion and (4) does not ‘‘starve’’ the point of congestion. The rest of this chapter is structured as follows. In the next section, a taxonomy for flow control mechanisms is established. Sec. 8.2 reviews some previously published flow control mechanisms. In Sec. 8.3 we discuss the characteristics of flow control mechanisms that are suitable for the interconnection network of a scalable multicomputer. In Sec. 8.4, we present the specific flow control mechanisms that are evaluated in this chapter. The evaluation is presented in Sec. 8.5. The simulation methodology and metrics for measuring the performance of the flow control mechanisms are described, and the simulation results presented and evaluated. This section concludes with a summary of the performance of the flow control mechanisms under consideration. Sec. 8.6 describes the hardware necessary to implement those flow control mechanisms which promote high performance communication. This is all drawn together in Sec. 8.7, in which the costs and benefits of the flow control mechanisms are summarized and evaluated.

8.1. Flow Control Mechanism Taxonomy

In this section a taxonomy for flow control mechanisms is established. Three classifications for flow control mechanisms are defined. Secs. 8.2 and 8.3 use this taxonomy in their evaluation of flow control mechanisms.

One can classify flow control according to the action taken when a buffer becomes full. Under *discarding* flow control mechanisms, packets are allowed to be transmitted to full buffers, but these packets are then dropped from the network. Discarding flow control mechanisms have the ability to determine when a packet has not reached its destination and to retransmit the packet. Under *blocking* flow control mechanisms, on the other hand, packets are prevented from being transmitted to a full buffer. Unless an error occurs, every packet injected into a network with blocking flow control will eventually reach its destination.

A second characteristic by which flow control mechanisms are classified is by where in the network the flow control decisions are made. In [Zhan91], this characteristic as used to divide flow control mechanisms into three categories: *end-to-end*, *global* and *local*. End-to-end flow control schemes specify guidelines regarding individual traffic flows from the senders to the final destinations. With some end-to-end schemes, the destination nodes transmit flow control information directly to nodes which are sending them packets. Other end-to-end schemes restrict the senders without using any dynamic information from the destinations.

Global flow control mechanisms are those in which either flow control information is globally distributed or a centralized server makes flow control decisions for the entire network. Local flow control mechanisms are those in which each switch in the network makes flow control decisions independently, using locally available information. A similar categorization is made in [Gerl80]. In this work, end-to-end flow control is called *entry-to-exit* flow control. Global flow control is called *network access level* flow control, and defined as flow control in which some or all of the network state is used to determine whether packets can enter the network. Local flow control is referred to as *hop-level*.

The third characteristic by which flow control mechanisms are classified is the trigger used to initiate flow control actions. *Predictive* flow control mechanisms attempt to prevent congestion from occurring. They enforce guidelines that attempt to eliminate traffic patterns that are likely to lead to congestion. Flow control actions are triggered when such patterns are detected (e.g., a particular sender has transmitted its ‘‘quota’’ of traffic for a given time period). On the other hand, *reactive* flow control mechanisms are triggered by the onset of congestion. Their goal is to minimize the impact of congestion once it has been detected.

8.2. Selected Existing Flow Control Mechanisms

The previous section established a flow control taxonomy based upon three classifications. This section examines a cross-section of existing flow control mechanisms. The mechanisms are classified using the taxonomy of the previous section, and their applicability to scalable multicomputers is evaluated. The selection of mechanisms presented is not exhaustive for any category. Rather, these mechanisms are just samples of a large, dense design space. They were chosen to facilitate the rest of the discussion in this chapter.

8.2.1. Hop-Level, Blocking Flow Control

The most basic flow control mechanism is *hop-level blocking flow control* (also called *blocking flow control*). Under this flow control, an input port will not accept packets when there is no room to store them. This flow control mechanism is widely used in scalable multicomputers: the Intel Paragon [Lill91], the TMC CM-5 [Leis92], the IBM SP-1 [Stun94], Dally's J-Machine [Dall87b], Seitz's Mosaic [Lutz84], the DASH multicomputer at Stanford [Leno92], the Cray T3D [Oed], all use blocking flow control. This flow control mechanism's sole function is to prevent packet loss due to overflowing packet buffers; it does not have a positive effect on performance. Indeed, due to the undirected nature of its back-pressure, blocking flow control is a fundamental cause of congestion propagation. However, this reactive, hop-level flow control can respond quickly to the congestion, allowing small buffers to be used without fear of packet loss.

Hysteresis has been explored by several researchers [Reis83, Yum83] as a hop-level flow control mechanism to improve network performance. In [Reis83], hysteresis is applied to a buffer with a single class of packets (i.e. a FIFO buffer) by implementing blocking flow control with a lower threshold value for restarting traffic flow than for halting traffic flow. Reiser found that the only benefit derived from using hysteresis in this context is a reduction in the number of times the flow control state for a particular buffer changes. In a communication network with dedicated flow control lines, the amount the flow control state changes is not an issue (assuming that it changes at most twice per packet).

Yum and Yen [Yum83] implement hysteresis in a buffer with two classes of packets: high-priority and low-priority. In a multi-queue buffer, hysteresis is attached to a queue-oriented blocking flow control scheme similar to maximum usage flow control (discussed in Sec. 8.4). Both the Reiser and the Yum and Yen flow control mechanisms are hop-level blocking flow control mechanisms. Since they use a hysteresis mechanism, it is not clear whether they are predictive or reactive — they may throttle flow at times when there is no congestion, and not throttle at times when there is congestion.

8.2.2. End-to-End Flow Control

Another commonly used flow control mechanism is *windowing* [Klei80, Kerm80, Cerf74]. This scheme works by limiting the number of packets which can be traversing the network between each source / destination pair at any given point in time. It requires that a sender be notified when packets it has transmitted have reached their destination. A sender may transmit packets to a destination at any throughput until the number of unacknowledged packets equals the size of the window. At that point, the sender may not transmit further packets to the destination until an acknowledgment for previously transmitted packets is received. The window limits the amount of congestion propagation which can occur in the network by limiting the number of outstanding packets. The size of the window can be set in a number of ways: a system-wide static window size, prior agreement between source and destination (bandwidth reservation) and dynamic window sizing based upon feedback from the destination [Kerm80].

Windowing is an end-to-end flow control mechanism. All flow control decisions are based upon the interaction between source and destination nodes — the network itself is treated as a black box. It is also a blocking, predictive flow control mechanism. When the number of outstanding packets reaches a threshold, further packets are blocked from entering the network. While the number of outstanding packets may be indicative of the existence of congestion, there is no direct detection of congestion. Windowing is used in conjunction with a hop-level blocking or discarding flow control mechanism which prevents packet loss when congestion occurs.

Leaky bucket [Turn86, Rath89] is an end-to-end flow control scheme based upon bandwidth reservation, rather than network feedback. When a sender reserves a path to a destination, it is assigned a maximum throughput. It is guaranteed that, if the sender does not exceed its maximum throughput to any destination, then the links in the network will not be over-utilized. If, however, this limit were to be strictly enforced, then the network would be under-utilized due to the bursty nature of computer communication. So, to increase the network utilization, the instantaneous throughput of the senders' communication is not restricted. Rather, when the sender is not using all of the bandwidth it has reserved it accumulates *credits* which allow it to use more than its reserved bandwidth at a later time. When the sender sends data, it can transmit at saturation throughput until it has used up all of its credit, at which point it becomes limited to the allocated bandwidth. To prevent a sender from accumulating too much credit (which would enable it to flood the network with packets), there is a maximum number of credits which can be accumulated; credits accumulated beyond this are discarded (the leak in the bucket).

Leaky-bucket flow control, like windowing, is a predictive, blocking, end-to-end flow control mechanism. There are many such flow control mechanisms described in the literature: [Rama91, Mukh86, Kalm90, Clar88, Cher89]. End-to-end flow control mechanisms limit congestion by restricting the rate at which individual senders can inject packets into the network or the total number of packets from each sender which may be in the network at one time. The extent to which such schemes reduce congestion depends on the severity of restrictions on the senders. However, tight restrictions on the senders are likely to result in under-utilization of the network. Under most practical end-to-end policies, congestion internal to the network is still possible. Hence some other flow control mechanism is needed in addition to the end-to-end scheme.

An additional problem with end-to-end schemes is that each sender must maintain state information regarding its traffic to each of its destinations. This can limit scalability. In practice, with multicomputers, end-to-end flow control is often used at the application level. As stated earlier, this does not eliminate the need for flow control at the interconnection network level, especially (but not exclusively) if there are multiple independent applications running on the system.

VirtualClock [Zhan91] is an alternative to ‘‘straight’’ end-to-end flow control. It is a hybrid flow control mechanism, operating at both the end-to-end and the hop levels. This gives it the functionality of an end-to-end flow control mechanism with the ability to detect and prevent internal network congestion.

VirtualClock is implemented at the switch level; it determines whether or not packets can proceed toward their destinations on a hop-by-hop basis. In order to perform end-to-end flow control, VirtualClock is built on top of virtual circuits. Each circuit has associated with it the maximum bandwidth that it will support. Every switch in the network maintains a separate virtual clock for every virtual circuit crossing the switch. The circuit's virtual clock is incremented each time a packet arrives at the switch on that circuit. The amount the clock is incremented depends upon the bandwidth assigned to the circuit — packets traversing circuits which have been allocated large bandwidths cause smaller increments than packets on low-bandwidth circuits. Flow is throttled if a circuit's clock advances ahead of a real-time clock. While flow control decisions are being made on a hop-by-hop basis, based upon information locally available, this is a blocking, predictive flow control mechanism operating on packet flows not unlike the end-to-end flow control mechanisms discussed earlier.

While VirtualClock has demonstrable strengths as a flow control mechanism for loosely-coupled systems [Zhan91], it has two drawbacks which make it inappropriate for a scalable multicomputer. First, it requires that significant computing resources exist at each switch, in order to maintain the virtual clock for each circuit which traverses the switch. Second, VirtualClock is a predictive flow control mechanism, and, as is described in Sec. 8.3, will potentially under-utilize a communication network. Thus, while VirtualClock is very effective in loosely-coupled distributed systems, there is reason to believe that it would be both unreasonably expensive and significantly less effective for scalable multicomputers.

8.2.3. Global Flow Control

In [Scot90], a global feedback flow control scheme is proposed for multistage interconnection networks (MINs) operating with a synchronous communication protocol (Ch. 4). In their scheme, the buffers which feed destination nodes (i.e., the packet buffers in the last switching stage) are monitored. Since the authors assume that the buffers are located at the output ports of the switches, each buffer in the last switching stage is associated with a single destination. When the number of packets in a monitored buffer reaches a *hot* threshold, the destination associated with that buffer is ‘‘hot’’. Similarly, when the queue length falls below a *not hot* threshold the destination becomes ‘‘cold’’. These changes in destination state are transmitted to all senders over a dedicated flow control feedback network. Congestion is controlled by preventing senders from transmitting packets to hot modules.

This is a blocking, global, reactive flow control mechanism. This combination of characteristics imposes requirements on the system. First, because reactive mechanisms take action after congestion has occurred, and because there is a significant latency between detecting congestion and reacting to it, this mechanism is implemented in conjunction with hop-level blocking flow control to prevent packet loss. In addition, the last switching stage is equipped with large buffers to prevent congestion from propagating during the period between detection and reaction.

The global flow control mechanism proposed by Scott and Sohi has several strengths. First, it is simple to understand and to implement. Second, the scheme is effective for the size and type of system the authors simulated — multistage interconnection networks with inputs and outputs numbering in the hundreds. The scheme does have weaknesses, though, which make it less appropriate for scalable multicomputer systems.

First, the feedback network itself is expensive. For a MIN, the feedback network will have approximately half the number of links that the communication network itself has. In addition, the scheme is oriented towards MINs. The problem with implementing their scheme over commonly used topologies such as hypercubes, fat trees, tori, and meshes is that these topologies have non-uniform distances between source / destination pairs. Implementing the flow control feedback network with these topologies presents significant technical difficulties. The long feedback latency of the flow control mechanism would be tolerable for nodes communicating over a great distance, but it would not work at all for nodes whose packets traverse a small number of hops.

A third issue which makes this scheme untenable is that this flow control mechanism is incapable of detecting NUT spots internal to the communication network. It is similar to the end-to-end protocols discussed previously, in that only the endpoints of the network are considered by the flow control mechanism. However, unlike the end-to-end flow control mechanisms, if a point of congestion was created on an internal link, the congestion would propagate indefinitely.

The crucial problem with MIN feedback flow control is that it is global, and thus not scalable (Sec. 8.3).

A global (network access) flow control scheme which does not attempt to broadcast data is *isarithmetic flow control* [Davi72]. The idea behind this scheme is that there is a pool of tokens available to network. For a node to transmit a packet, it must first grab a token. The token traverses the network with the packet and is freed upon packet reception. Thus, the token pool limits the total number of packets in the network in a distributed fashion; this is a blocking, predictive, global flow control mechanism.

Kleinrock and Gerla [Gerl80] comment that isarithmetic flow control works well when the traffic is uniformly distributed, but that it ‘‘may lead to unnecessary throughput restrictions, and therefore, to poor performance in the case of nonuniform, time-varying traffic patters.’’ The problem is partly attributed to the fact that, when a source transmits a significant number of packets to a particular destination, that destination will accumulate a large fraction of the token pool which must then be redistributed. If the tokens are sent back to the source of the flow, then that source will end up capturing a large percentage of the total available network bandwidth. If they are not, then the sender will run out of tokens and will be starved.

There are a number of additional problems with isarithmetic flow control. First, if there are enough tokens in the pool to fully load the network under uniform traffic conditions, then it will still be possible to congest the network under non-uniform traffic conditions. Further, with a significant percentage of the tokens associated with packets that are involved in congestion, the non-congested areas of the communication network will be under-utilized (i.e. the negative feedback is not directed to the nodes involved in the congestion). Finally, there is the increased communication latency incurred by having to obtain a token for each packet transmitted. All of these problems are exacerbated as the communication network scales, making isarithmetic flow control inappropriate for a scalable multicomputer.

8.2.4. Discarding, Hop-Level Flow Control

Dias and Kumar [Dias89] have proposed a packet switching flow control mechanism that displays some of the characteristics of a virtual circuit flow control mechanism (e.g. VirtualClock[Zhan91]). Their scheme, which we call *destination-based* flow control (Dias and Kumar did not name it) allows only a single packet to any single destination within a buffer at any point in time. If a packet arrives at a buffer that is already holding a packet destined for the same address, the packet is discarded, and the sending switch re-queues the packet at the tail of the FIFO buffer from which it came (Fig. 8.3). Thus, destination-based flow control is a discarding, hop-level, reactive flow control mechanism. It provides an elegant two-pronged solution to tree saturation. By allowing only a single packet per destination per buffer, a hot spot cannot cause a buffer to become filled, and by requeueing rejected packets at the tail of the list, packets not addressed to the hot spot can bypass those that are.

Destination-based flow control has two key positive features. First, it is *directed* explicitly only towards packets addressed to hot destinations. Second, it is *selective* and exerts back-pressure only on packets addressed to the hot destination. Other packets are not restricted.

The ability to direct the back-pressure comes from the basis for flow control decisions. By using a comparison of destination addresses to make flow control determinations, this scheme assures that no matter how far from the point of congestion the back-pressure is felt, it is only packets which will traverse the point

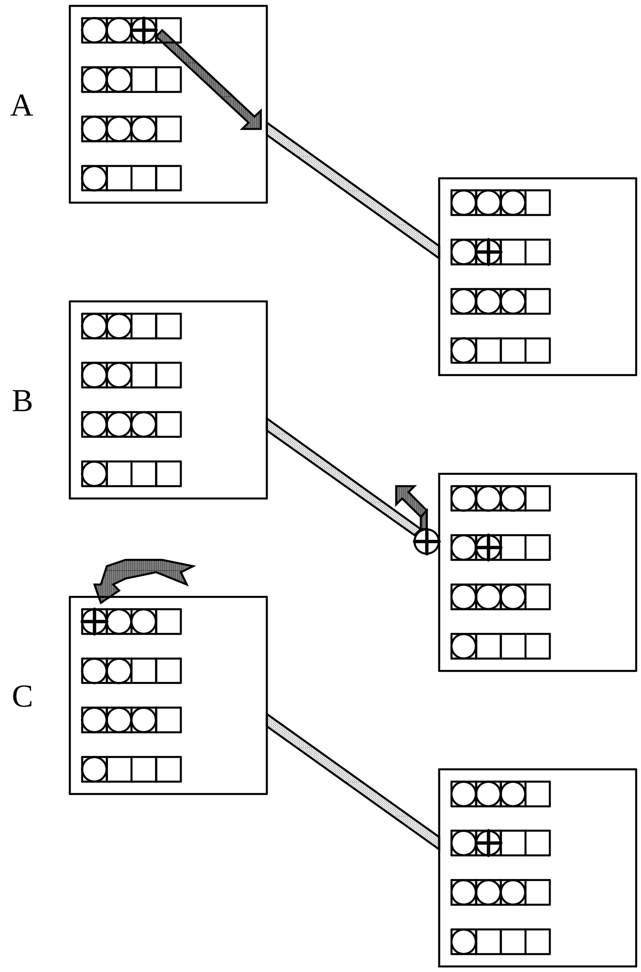


Figure 8.3: *Operation of Destination-Based Flow Control.* Only one packet per destination per buffer is allowed. If the destination of the incoming packet matches that of a packet already in buffer, the incoming packet is discarded (B) and requeued at the tail of the previous buffer (C).

of congestion which are throttled. The ability to avoid throttling those packets which are not involved in the congestion comes from requeueing packets at the tail of the buffer when they are rejected. Destination-based flow control can do this and still maintain the FIFO ordering of packets between each source / destination pair, because there will be at most one packet to a destination in a queue at any point in time.

Due to its ability to direct back pressure, the behavior of destination-based flow control is similar to the behavior of schemes based on virtual circuits, such as VirtualClock. Functionally, however, destination-based flow control is entirely different from VirtualClock; it is hop-level (not end-to-end), packet switching (not based upon virtual circuits) and reactive (not predictive).

8.3. Flow Control Schemes for Tightly-Coupled Interconnection Networks

A large number of flow control mechanisms have been proposed in the literature; Sec. 8.2 presents only a small subset. The choice of a ‘‘good’’ flow control mechanism depends on the characteristics of the network. The focus of our investigation of flow control is on schemes which are appropriate for the interconnection networks of tightly-coupled multicomputers and multiprocessors. In this section we describe the key characteristics of such tightly-coupled networks. We then discuss the flow control mechanism design space (as described in Sec 8.1) and identify the class of flow control schemes that are most appropriate for tightly-coupled networks.

The property of being *tightly-coupled* indicates both support for and reliance upon low communication latencies. Fine-grained distributed applications require high bandwidth communication paths and communication latencies on the same order as access latencies to local primary memory. These goals must be met in conjunction with efficient support for small messages (tens of bytes). For example, in [Dall87b], it is suggested that the processes of a fine-grained distributed application will transmit a ten-byte packet every one hundred clock cycles. For the system to be *scalable*, no single object or resource in the system can be a performance bottleneck. A general purpose scalable multicomputer should support multiple applications executing simultaneously, with new applications being initiated on the system at any given point in time. With multiple applications competing for shared resources, predicting resource utilization (including network resources) is not feasible.

Very few large-scale multicomputers implement discarding flow control [Crow85, Rett90]. The first problem with discarding flow control is that, when a packet is discarded, it must be retransmitted by the sender and then re-traverse all of the communication links that it had traveled prior to being discarded. This is contrary to the goal of minimizing transmission latency. It also creates a positive feedback for network congestion — when congestion occurs, packets are discarded more frequently and thus require *more* network bandwidth than they did at lower communication throughputs.

The second problem deals with returning acknowledgments to packet senders. Since the regular packets are often small, if each packet is acknowledged separately, the acknowledgement traffic will consume a significant fraction of network bandwidth. The acknowledgement traffic can be reduced by only acknowledging groups of packets (i.e., windowing). However, this increases the average packet latency since there will be longer delays for the retransmission of discarded packets. A third problem with discarding flow control is that it may fail to maintain FIFO packet ordering between sender and destination. This requires packet reordering by higher levels of the communication protocol.

Destination-based flow control is a discarding flow control scheme which behaves similarly to blocking flow control — FIFO packet ordering is maintained and communication latencies are kept low by performing discarding / retransmission on a hop-by-hop basis. For these reasons, an evaluation of destination-based flow control is included in this chapter, even though it is, technically, a discarding flow control mechanism. Enhancements to the destination-based flow control described in Sec. 8.4 minimize link re-traversals, making it even more similar to blocking flow control.

Predictive flow control mechanisms attempt to prevent congestion by throttling traffic which has the potential to cause congestion. In a general-purpose multicomputer, traffic patterns may be highly irregular. If the predictive flow control severely restricts traffic, it may unnecessarily reduce the available network bandwidth. However, if the predictive flow control is less restrictive, it may fail to significantly reduce the probability of congestion. Thus, in this case, the utility of the predictive flow control is in doubt and a different ‘‘good’’ flow control mechanism is needed in addition to the predictive scheme.

Section 8.2.2 discusses some of the problems of implementing end-to-end flow control in a scalable multicomputer. The problems generally stem from the conflict between low latency and scalability. As a system scales, destinations move further away from sources, increasing the latency of end-to-end feedback. Furthermore, most end-to-end flow control mechanisms are not capable of explicitly detecting and reacting to congestion that occurs in the middle of the network. There are end-to-end flow control schemes that operate without any feedback of dynamic network activity. However, as discussed earlier, such predictive schemes are not likely to perform well in the dynamic environment of a general-purpose multicomputer.

Since they must either process all flow control information centrally or broadcast changes in flow control state, global flow control mechanisms scale poorly, as well.

Hop-level flow control displays the characteristics desired in a scalable multicomputer. Since flow control decisions are made independently by each switch, there is no increase in complexity as the system scales. Similarly, because all decisions are based upon locally available information, scaling does not impact the latency of the mechanism. A distributed flow control mechanism is also more robust in the face of faults; the failure of a single link or switch will only impact those nodes adjacent to the failure. Finally, there are the benefits of having the same flow control hardware and algorithm repeated throughout the network — the system is easier to design, build and maintain (e.g. MIT's J-Machine[Dall90a]).

8.4. Hop-Level Flow Control

The previous section of this chapter discussed the reasons that end-to-end and global flow control mechanisms are unsuitable candidates for implementation in a scalable, tightly-coupled computer network. Hop-level flow control, on the other hand, displays a number of features which mark it as appropriate for such a system. With hop-level flow control, each sender and switch in the system makes flow control decisions based upon information that is locally available. Thus, the complexity, cost and/or requirements of each flow control calculation is independent of the network size. Second, flow control information is propagated through the network via *back pressure* [Gerl80]. This avoids the use of a broadcast or a multicast flow control feedback, which, as was discussed in the previous section, does not promote scalability. The remainder of this section describes the hop-level flow control mechanisms evaluated in this chapter.

8.4.1. Blocking Flow Control

Being the flow control mechanism used in so many of the current multicomputers, hop-level blocking flow control was an obvious choice for evaluation. One of the key choices in the design of blocking flow control is whether (1) to support blocking of transmissions after only part of a packet has been transmitted or (2) to restrict blocking to occur only between packets. In the former case, a buffer accepts packets right up to the point where the buffer is completely filled. If the buffer is in the middle of receiving a packet at that point, the reception / transmission is halted, and the packet is stored for some period of time in two (or more) switch buffers. *Wormhole routing* [Dall86] requires this form of blocking flow control since the buffers are too small to store an entire packet in a single node. With wormhole routing the nodes *must* support *virtual cut-through* [Kerm79], i.e., be able to begin forwarding a packet before all of it is received.

In the second case — restricting blocking to between packets — buffers must halt flow if there is not enough space in the buffer to store an entire packet. While this style of blocking flow control restricts the maximum packet size in a network, it also ensures that once a packet transmission begins, it will complete without interruption. This form of blocking flow control can be used with nodes that support virtual cut-through as well as with nodes that do not. We evaluated the second type of blocking flow control with nodes that do support virtual cut-through. Specifically, we assume a maximum packet size of thirty-two bytes so flow is halted when there are less than thirty-two bytes available in a buffer.

8.4.2. Maximum Usage Flow Control

In their survey of flow control schemes, Kleinrock and Gerla [Gerl80] discuss *channel queue limits*. This is a mechanism which limits the amount of buffer space a particular class of packets can occupy in a buffer pool. In the case of hop-level flow control, the packets are classified according to which output port they will be leaving the switch on. There are a number of ways in which the buffer memory usage can be limited: each output port can have a maximum memory allowance (sharing with maximum queues — SMXQ), each can be guaranteed a minimum availability (sharing with minimum availability — SMA), SMXQ and SMA can be combined, or the buffer space an be strictly partitioned (see the SAMQ switch, Ch. 3). In their discussion of channel queue limits, Kleinrock and Gerla refer primarily to [Irla78].

In [Irla78], Irland found that, under non-uniform traffic, complete partitioning (SAMQ buffers) performed slightly worse than the other channel queue limit mechanisms, while at high offered loads (saturation), all channel queue limit mechanisms supported approximately the same throughput. He also found that switches with unrestricted sharing performed significantly worse than those with some form of channel queue limits, *even under uniformly distributed traffic*.

This dissertation refers to channel queue limits as *maximum usage flow control* (the networks we evaluated do not have channels). Under maximum usage flow control, each queue of a buffer has a bit of flow control state associated with it. When the length of the queue (the number of *buffer blocks* occupied by the queue) exceeds a threshold (or fills the queue, in the case of static buffer allocation), then that queue's flow control signal is asserted. Similarly, when the length is equal to or less than the threshold, the signal is dropped. If there are fewer than thirty-two bytes (four blocks) available in the buffer, then all of the queues' flow control signals are asserted.

The use of maximum usage flow control implies a dynamically allocated buffer (there is sharing of storage space between queues of packets destined to different output ports). As discussed in Ch. 3, implementation issues direct one to implement switch buffers at the input ports. Thus, maximum usage flow control is evaluated using dynamically allocated, multi-queue buffers located at the input ports — the DAMQ buffer.

We also simulated *static buffer allocation*; this is a form of maximum usage flow control in which the buffer is strictly partitioned (statically allocated) among the queues. Chs. 4 and 6 examine static buffer allocation as an implementation option for non-FIFO buffers: the SAMQ and SAFC buffers. Their performance under non-uniform traffic conditions is presented as part of the evaluation of maximum usage flow control.

Maximum usage flow control requires that packets be pre-routed. *Pre-routing* is the determination of which output port *of the next switch* a packet will traverse. Thus, for each hop through a network using maximum usage flow control, each packet must be routed twice: once to determine which output port the packet is destined for, and once to determine which queue the packet will be appended to in the next switch. If the packet at the head of a queue will be append to a queue on the next switch which is currently blocked, then the crossbar arbiter will not connect the buffer to that output port; the packet is blocked until the destination queue unblocks. In our evaluation of maximum usage flow control, we looked at blocking thresholds of six blocks and ten blocks and static buffer allocation (both SAMQ and SAFC buffers), all with 128 byte buffers.

8.4.3. Two-Counter Flow Control

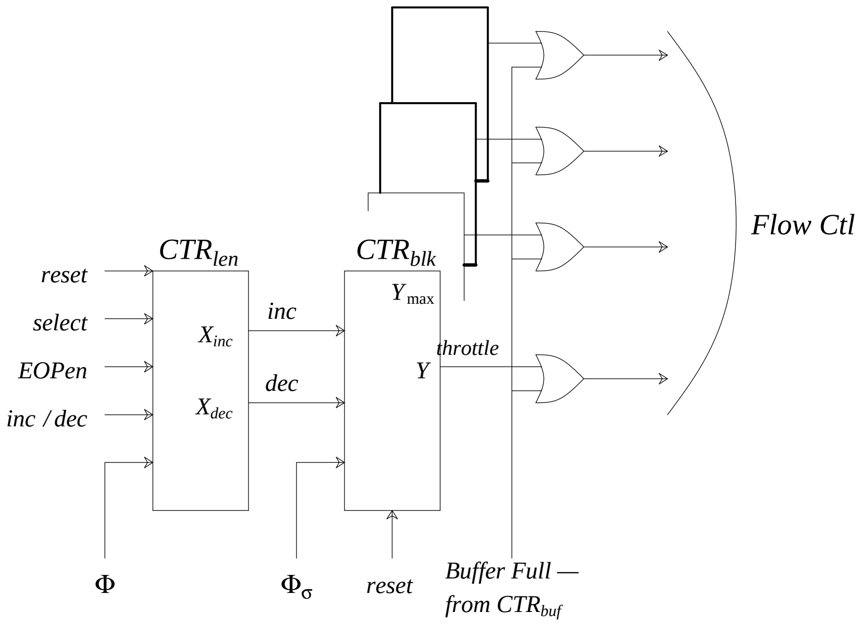


Figure 8.4: *Block Diagram, Two-Counter Flow Control.* CTR_{len} indicates how many blocks of the buffer are occupied by the queue. CTR_{blk} is incremented / decremented once every σ clock cycles, depending on whether CTR_{len} is above X_{inc} or below X_{dec}. Flow to the queue is throttled when the value of CTR_{blk} is above Y, and is throttled to all queues when the buffer is full. The maximum value of CTR_{blk} is Y_{max}.

Two-counter flow control is a general implementation of hysteresis flow control (it subsumes the hysteresis flow control mechanisms discussed in Sec. 8.2). It is similar to maximum usage flow control (channel queue limits), in that it limits the amount of buffer space a single packet queue can occupy. Two-counter flow control is more complicated than the hysteresis schemes mentioned above, in that it provides three separately configurable mechanisms for hysteresis. A block diagram of the flow control mechanism for a single queue is shown in Fig. 8.4. Each queue has two counters associated with it. CTR_{len} keeps track of the current length of the queue. CTR_{blk} maintains the current flow-control state for the queue. When the value of CTR_{blk} reaches the threshold Y, the queue blocks (i.e. does not accept packets). When its value drops below Y, the queue unblocks. CTR_{blk} is updated every σ clock cycles (ϕ_σ, shown in Fig. 8.4, is the system clock divided by σ). Whether CTR_{blk} is incremented, decremented or left unchanged on an update cycle depends upon the number of buffer blocks occupied by the queue. When the flow control state is updated, CTR_{blk} is incremented if more than X_{inc} buffer blocks are occupied by the queue, and decremented if less than X_{dec} are occupied.

σ	Interval between CTR_{blk} updates.
X_{inc}	CTR_{len} threshold, beyond which CTR_{blk} is incremented.
X_{dec}	CTR_{len} threshold, below which CTR_{blk} is decremented.
Y	CTR_{blk} threshold, at which packets to the queue are blocked.
$Y_{\max}$	Maximum value of CTR_{blk}; further increments have no effect.

Table 8.1: *Two-Counter Flow Control Parameters.* See Fig. 8.4 for a block diagram of the two-counter flow control hardware.

The behavior of the two-counter flow control mechanism is controlled by the values of the full and not full thresholds (X_{inc} and X_{dec}), the CTR_{blk} threshold at which the queue blocks (Y), the maximum CTR_{blk} value and rate of ϕ_σ (see

Tab. 8.1). Being a flexible and fully parameterized flow control mechanism, two-counter flow control is an ideal test-bed for hysteresis flow control. For example, if the blocking threshold and maximum value of CTR_{blk} are both set to one ($Y = Y_{MAX} = 1$), σ is set to one ($\phi_\sigma = \phi$ — the system clock), and $X_{inc} = X_{dec} - 1$, the resulting flow control behaves like the maximum usage flow control described above. If the above parameters are changed such that $X_{inc} \geq X_{dec}$, then then the flow control is identical to previously published hysteresis flow control schemes, which depend strictly upon separate blocking and unblocking thresholds [Reis83, Yum83].

Table 8.2 displays the two-counter flow control parameter configurations which we evaluated, with an explanation of the intended behavior of each of the parameter settings. The goal in choosing these five versions (H, T, tH, A and aH) of two-counter flow control is to broadly explore the behavior-space of this flow control mechanism. While this does not represent an exhaustive search of the two-counter parameter space, it is expected that evaluating these five parameter settings will broadly indicate which aspects of two-counter flow control improve network performance. We found that assigning more ‘‘extreme’’ values to the parameters reduced the performance of the resulting network under a variety of traffic loads.

name	settings	explanation
H	$\sigma = 1$ $X_{inc} = 8$ $X_{dec} = 4$ $Y = 1$ $Y_{\max} = 1$	Hysteresis flow control, as presented in [Reis83, Yum83]. The queue is throttled when there are more than eight buffer blocks in the queue; flow is restored when there are less than four blocks in the queue.
T	$\sigma = 20$ $X_{inc} = 6$ $X_{dec} = 6$ $Y = 1$ $Y_{\max} = 1$	Infrequent sampling. The queue is throttled when there are more than six buffer blocks in the queue; flow is restored when there are less than six buffer blocks in the queue. CTR_{len} is sampled once every twenty clock cycles.
tH	$\sigma = 20$ $X_{inc} = 8$ $X_{dec} = 4$ $Y = 1$ $Y_{\max} = 1$	Hysteresis with infrequent sampling.
A	$\sigma = 20$ $X_{inc} = 8$ $X_{dec} = 8$ $Y = 2$ $Y_{\max} = 4$	Accumulator hysteresis. CTR_{blk} increments every 20 clock cycles that $CTR_{len} > 8$, decrements every 20 cycles that $CTR_{len} < 8$. CTR_{len} will be > 8 for at most 40 cycles before flow is halted, and < 8 for at most 40 cycles before flow is resumed.
aH	$\sigma = 20$ $X_{inc} = 8$ $X_{dec} = 4$ $Y = 2$ $Y_{\max} = 4$	‘‘Straight’’ and accumulator hysteresis combined.

Table 8.2: *Parameter Settings for Two-Counter Flow Control.* See Tab. 8.1 and Fig. 8.4 for explanations of the function of each parameter.

8.4.4. Destination-Based Flow Control

The destination-based flow control described by Dias and Kumar in [Dias89] was implemented with FIFO buffers. Since the superiority of DAMQ buffering has already been demonstrated (Chs. 4 and 6), we simulated destination-based flow control with DAMQ buffers. One could also evaluate destination-based flow control with SAMQ or SAFC buffers; in the context of this chapter, this represents a combining of multiple flow control mechanisms (static buffer allocation and destination-based flow control). Sec. 8.5.10 examines the performance of combined flow control mechanisms.

In our simulator, every time a buffer using destination-based flow control receives a packet, it responds to the sending switch with either a positive or a negative acknowledgment (ACK / NACK). The sending buffer does not de-allocate the buffer memory occupied by the packet until the ACK is received. When a packet is rejected, the sending switch receives the NACK within six clock cycles of the initiation of transmission, which is considerably less than the time required to send the maximum-length packet (thirty-two bytes, in our simulator). The packet is re-linked to the tail of the queue, the sending buffer is considered to be ‘‘busy’’ for the time required to transmit the entire packet, and the communication link is ‘‘busy’’ for the time required to transmit the entire packet plus the inter-packet idle period (Sec. 6.1.1).

We enhanced our implementation of destination-based flow control by aborting transmissions as soon as a NACK is received (since Dias and Kumar worked with a synchronous network, this optimization was not applicable to their work). This enhancement has the potential to reduce the amount of time spent by buffers and output ports transmitting already-rejected packets. In the case of simulating SAMQ and SAFC buffers using destination-based flow control with this extension, we do not account for the overhead of moving the packet from the head of the queue to the tail, nor do we consider the possible conflict of a packet arriving while this movement is taking place. For the case of implementing the scheme on a DAMQ buffer, the buffer cannot transmit for a period of time equal to two clock cycles for each buffer block occupied by the packet, to account for the time required to manipulate the linked lists. We refer to the enhancement of aborting transmissions and requeueing as soon as a NACK is received as *abortive transmissions.*

We also designed a second extension to destination-based flow control. This extension operates on packets which have been rejected from a succeeding switch, and requeued in a preceding switch. When the packet comes back to the head of its queue, whether or not it is transmitted depends upon whether the succeeding switch has indicated that the packet which blocked this packet on its previous transmission attempt has been transmitted from the succeeding switch. If the *BLOCKER* packet has not been transmitted, then the *BLOCKEE* packet is re-linked at the tail of its queue without attempting to transmit it. The idea behind this second enhancement is to not transmit packets which are known will be discarded by the next switch, with the goal of reducing the number of times a packet is transmitted and rejected. This extension is referred to as *restricted retry.*

8.4.5. Path-Based Flow Control

Path-based flow control is identical to destination-based flow control, except for the basis it uses for throttling traffic. Instead of rejecting packets whose destination matches the destination of a packet residing in the buffer, path-based flow control rejects packets whose next n hops are identical to the next n hops of a packet residing in the buffer. The idea behind path-based flow control is to achieve the functionality of destination-based flow control, with the additional ability to detect congestion which is not associated with a particular destination.

Path-based flow control can be implemented in an identical fashion to destination-based flow control, including the enhancements. For the four stage omega network of our simulator, setting $n = 4$ causes path-based flow control to behave identically to destination-based flow control. Setting $n = 1$, on the other hand, causes the buffers to behave like SAMQ buffers which can store one packet per queue. The simulation results of path-based flow control presented in this dissertation use $n = 2$.

8.5. Flow Control Performance Evaluation

This section presents an evaluation of the performance of the flow control mechanisms which were discussed in the previous section. First, the network simulator that was used for the evaluation is characterized. Then, the methodology of the evaluation is described. Finally, the simulation results are presented and used to evaluate the flow control mechanisms' performance.

8.5.1. The Simulator

The flow control mechanisms were evaluated using the asynchronous network simulator presented in Ch. 6. Thus, the flow control mechanisms were simulated in a computer network supporting virtual cut-through, variable-length packets and multi-packet messages. The simulator was configured as a 256×256 multistage interconnection network in the omega topology. The network was composed of four stages of sixty-four 4×4 switches. For the simulations, each switch was implemented with a 128-byte buffer† at each input port (unless otherwise stated). The senders generated messages with geometric interarrival times, as described in Ch. 6. The maximum packet size allowed in the network was thirty-two bytes; when senders generated messages longer than thirty-two bytes, the messages were broken into multiple packets which are transmitted sequentially at maximum throughput.

The *applied load* of the system is characterized by two components: the distribution of message lengths and the distribution of message interarrival times. As stated above, the message interarrival times are geometrically distributed, so for each sender there is a fixed probability of a message being generated on each clock cycle. We specify this probability as *messages per sender per clock cycle* (m/sc). Throughput (thpt) is measured in *bytes per link per clock cycle* — since the links are one byte wide, and can transmit one byte per clock cycle, throughput is presented as the fraction of available bandwidth utilized (i.e. $0 \leq thpt \leq 1$). Average latency (lat) is the average number of clock cycles from packet creation to the first byte of the packet reaching its destination. See Ch. 4 for a more detailed discussion of the simulator.

† Flow control mechanism functionality improves with larger buffers. We restricted our buffers to 128 bytes (storage for four thirty-two byte packets) because, in Ch. 6, we established that larger buffers provided negligible performance improvement under conditions of uniformly distributed traffic, and the cost of additional buffer memory is linear. The diminishing benefits of additional buffer space is a well-known result [Dias81].

8.5.2. Evaluation Methodology

In the introduction to this chapter, it was mentioned that network congestion is caused by a group of senders (G_{NUT}) attempting to over-utilize one or more links within the network. Further, we asserted that, unless some mechanism such as dynamic routing reduces the dependency of G_{NUT} on the critical link(s), the communication throughput of G_{NUT} is determined by the bandwidth of the critical link(s), and no flow control mechanism is capable of improving it. Flow control can, however, improve the performance of the communication network as a whole by preventing the congestion caused by G_{NUT} from reducing the network bandwidth available to the other nodes in the system. Thus, we evaluated the flow control mechanisms by performing network simulations in which a set of senders was designated to be G_{NUT}. These senders were directed to transmit packets in a manner that would create congestion in the network. Various aspects of the performance of the network was measured under these conditions to determine the degree with which each flow control mechanism could preserve network performance.

Congestion within a network can display a range of characteristics which are determined by (a) the relative locations of the senders which comprise G_{NUT} and (b) with what non-uniform traffic pattern does G_{NUT} create congestion. We have developed a suite of network benchmarks in order to explore the performance of flow control mechanisms under a variety of congestion patterns. This suite is intended to stress the communication network in a variety of ways, with the idea that a flow control mechanism that can successfully handle a wide variety of stochastically-generated congestion patterns in the simulator will be effective against application-generated congestion when implemented in an actual multicomputer.

The benchmarks specify the senders which will comprise G_{NUT} and the pattern with which they will transmit data. For each benchmark, G_{NUT} creates a point or points of congestion within the network by transmitting packets at saturation throughput in the traffic pattern associated with that benchmark. The remaining senders in the system are partitioned into groups according to their interaction with G_{NUT}. The applied load of the senders in these groups is varied from 0.0 m/sc to saturation.

We used three performance metrics to evaluate the flow control mechanisms. The first is the maximum throughput of the network. This statistic reflects the ability of the flow control mechanisms to protect the performance of the ''innocent'' senders (the members of $G_0, G_1, \cdots$) from the network congestion caused by members of G_{NUT}. The second metric is the average latency of G_{NUT} and G_i per throughput. Finally, the utilization of the critical network resources was examined. Since the point(s) of congestion are the performance bottleneck for G_{NUT}, it is highly desirable that these links be fully utilized. For the benchmarks with a static point of congestion, we measured the throughput of data through this link. For the benchmarks with dynamic congestion patterns, we measured the total network throughput to evaluate the critical resource utilization.

Ideally, a flow control mechanism ensures that the nodes which are not members of G_{NUT} are not negatively impacted by the congestion created by G_{NUT}. Thus, the performance of a blocking flow control under uniform traffic conditions is the case against which the flow control mechanisms' performances under the

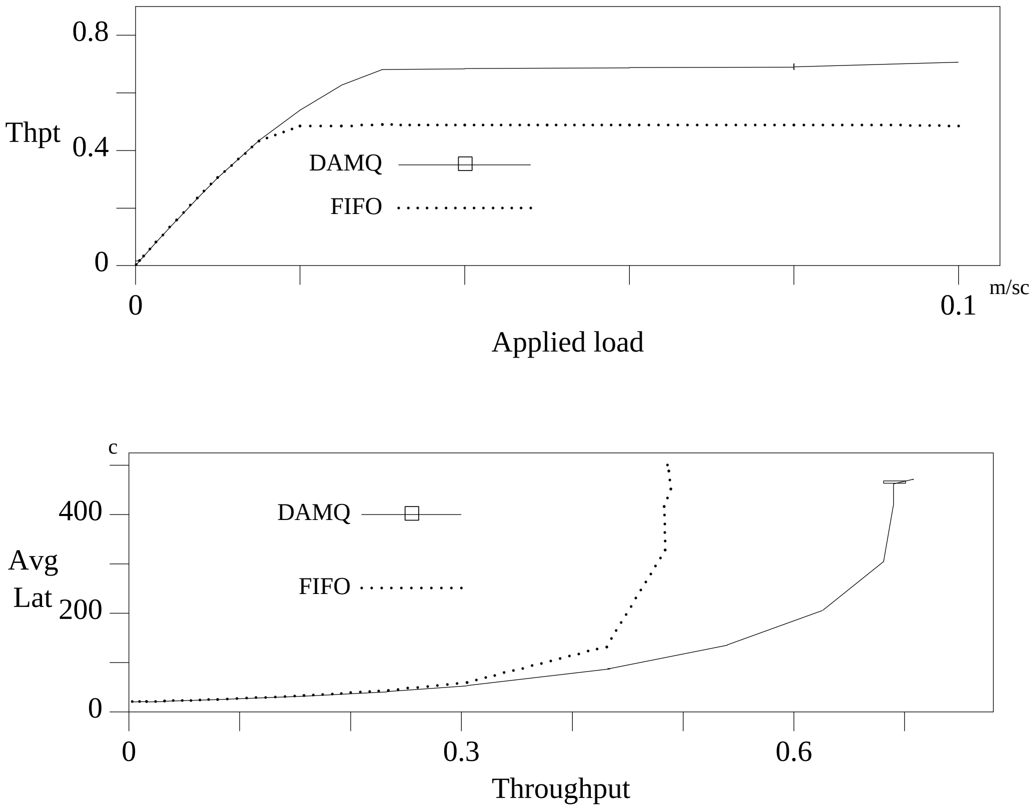


Figure 8.5: *Throughput and Latency, Uniform Traffic Distribution.* 256×256 omega network, composed of 4×4 switches. DAMQ and FIFO buffers. Thirty-two byte packets. Packet destinations randomly chosen, uniform distribution. The rectangles associated with the DAMQ buffer results are the confidence interval measures — see Sec. 8.5.3.

benchmarks is compared. Figure 8.5 shows both throughput vs. applied load and average latency vs. throughput for networks of FIFO and DAMQ switches. All senders in these simulations generate thirty-two byte packets whose destinations are randomly chosen from a uniform distribution. As can be seen from the throughput vs. applied load graph, at lower applied loads the throughput is directly proportional to the applied load, indicating that there is no congestion in the network. For both FIFO and DAMQ switch networks, as saturation throughput is approached, the lines quickly become horizontal (i.e. additional applied load results in no additional throughput). Similarly, the average latency vs. throughput graph shows only slight increase in the average latency until the saturation throughput is approached, at which point the lines become vertical (i.e., additional applied load results in additional latency, but no additional throughput).

These graphs demonstrate two desirable features which are present under uniform traffic conditions and which a flow control should preserve in networks with non-uniform traffic conditions. The first is that, until saturation throughput is approached, the network is transparent to the senders. That is to say, from the senders' point of view, it appears that each sender is directly connected to all of the destinations, with no network in between. The network is transparent when it imposes no back pressure upon the senders and imparts a negligible latency upon the packets traversing it. The second desirable trait is that, when the applied load approaches and moves beyond the point of maximum throughput, the throughput of the network does not drop. Networks which experience a reduction in throughput when an excessive load is applied are termed *reactive.* Non-uniform traffic conditions can cause a network to be reactive. Reactivity is an undesirable characteristic, in that network throughput is reduced at exactly that point in time when it is most needed — when communication bandwidth is the performance bottleneck.

8.5.3. Confidence Intervals

The simulation results presented in this chapter required extensive CPU cycles to generate. This prevented the generation of confidence intervals for every point on every line in every graph presented. Instead, confidence intervals were generated for three points in a representative subset of lines. The lines which have confidence intervals are indicated in the graph legends by a rectangle across the middle of the line in the legend, and the confidence intervals themselves are presented as rectangles in the graph. The vertical dimension of the rectangle represents the 95% confidence interval of the y-value of that point in the line, and the horizontal dimension reflects the x-value. Tab. 8.3 presents the lines for which confidence intervals exist.

The confidence figures were generated by running the simulator for a ‘‘warm-up period’’ and then executing it for five ‘‘intervals’’. Each interval was of the same length as the simulation runs which generated the other points in the graphs. After each interval, the performance statistics were recorded and then reset for the next interval. The rectangles in the graphs provide quantitative information as to the reliability of the results represented by that data point — there is a probability of 0.95 that, if the results of an infinite number of intervals were collected, the data point representing their average would fall within the rectangle. The rectangles also provide qualitative information as to the reliability of the remaining points in the graphs. All of our confidence intervals are ‘‘tight’’ — there is little variance in the simulation results. Thus, while a confidence interval cannot be generated for a single simulation result, we are confident in the accuracy of the data points which were generated from single simulation runs.

Figure	Line
8.5	DAMQ
8.8	DAMQ
8.9	Blocking
8.10	tH
8.13	DAMQ
8.14	Blocking
8.16	Path-Based
8.19	MU-6
8.20	aH
8.24	MU-6
8.27	FIFO
8.28	SAFC
8.29	A
8.33	Dst-Based, MU-22
8.34	Dst-Based, SAFC
8.36	Dst-Based
8.37	Dst-Based, MU-22
8.37	Dst-Based, SAFC

Table 8.3: *Confidence Interval Map.* The figures and results for which confidence intervals have been generated.

It should be emphasized that there are *three* points for which a confidence interval exists for each of the results specified in Tab. 8.3. For many of the points, particularly those generated at lower network throughputs (for which there is less variance in performance), the rectangles representing the confidence intervals are too small to be seen.

8.5.4. Benchmark I: Static Hot Spot, Serial Requests

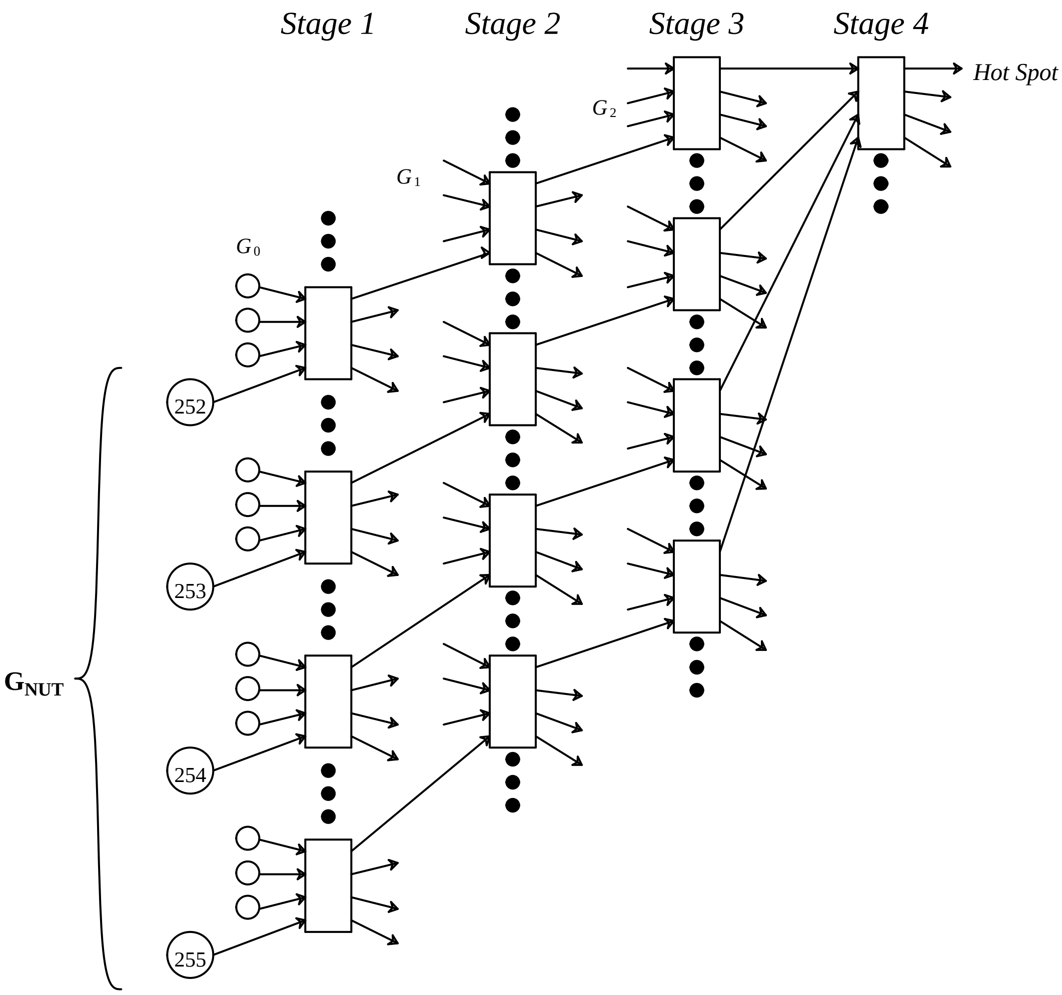


Figure 8.6: *Benchmark I: Hot Spot, Serial Requests.* G_{NUT} (<252>, <253>, <254>, <255>), transmitting all packets to the ‘‘hot’’ destination, creating a point of congestion at *output port* #0 of *switch* #0 of the fourth switching stage.

In this benchmark, the members of G_{NUT} transmit thirty-two byte packets to a single destination. There are four senders in G_{NUT} — their addresses (*senderIDs*) are <240>, <244>, <248> and <252>. These particular senders were chosen so that the path from each member of G_{NUT} to the hot spot would not intersect with that of any other other member of G_{NUT} until the final link to the hot destination; thus there is no contention between the members of G_{NUT} prior to the hot spot itself. The senders of G_{NUT} transmit 100% of their packets to the hot spot at saturation throughput.

Name	Function	Explanation
G_{NUT}	$senderID$ & 0xFC = 0xFC	Four senders whose paths intersect at final switching stage.
G_0	$senderID$ & 0x3C = 0x3C	Senders whose paths intersect G_{NUT} at first switching stage (i.e. they share a first stage switch with a member of G_{NUT}).
G_1	$senderID$ & 0x0C = 0x0C	Senders whose paths intersect G_{NUT} at the second switching stage.
G_2	TRUE	All remaining switches — their paths intersect with the paths of a member of G_{NUT} at the third switching stage.

Table 8.4: *Group Definition, Benchmark I.* The functions listed use the routing characteristics of the omega network to determine which senders are members of which sender groups. The parameter $senderID$ is the source address.

The remaining 252 sending nodes are divided into three groups, according to the degree to which they interact with G_{NUT} (Fig. 8.6). G_0 comprises the twelve senders which share a first-stage switch with a member of G_{NUT}. G_1 is the forty-eight senders which can reach the same second-stage switches as G_{NUT} (and G_0). Finally, G_2 is the 192 senders (all remaining) which intersect G_{NUT} (and $G_{0,1}$) at the third switching stage. Tab. 8.4 demonstrates how these groups were defined in the simulator using bit operations on the $senderIDs$. These three groups generated uniformly-distributed traffic with the same applied load as each other. This applied load is graduated from 0.0 m/sc to saturation.

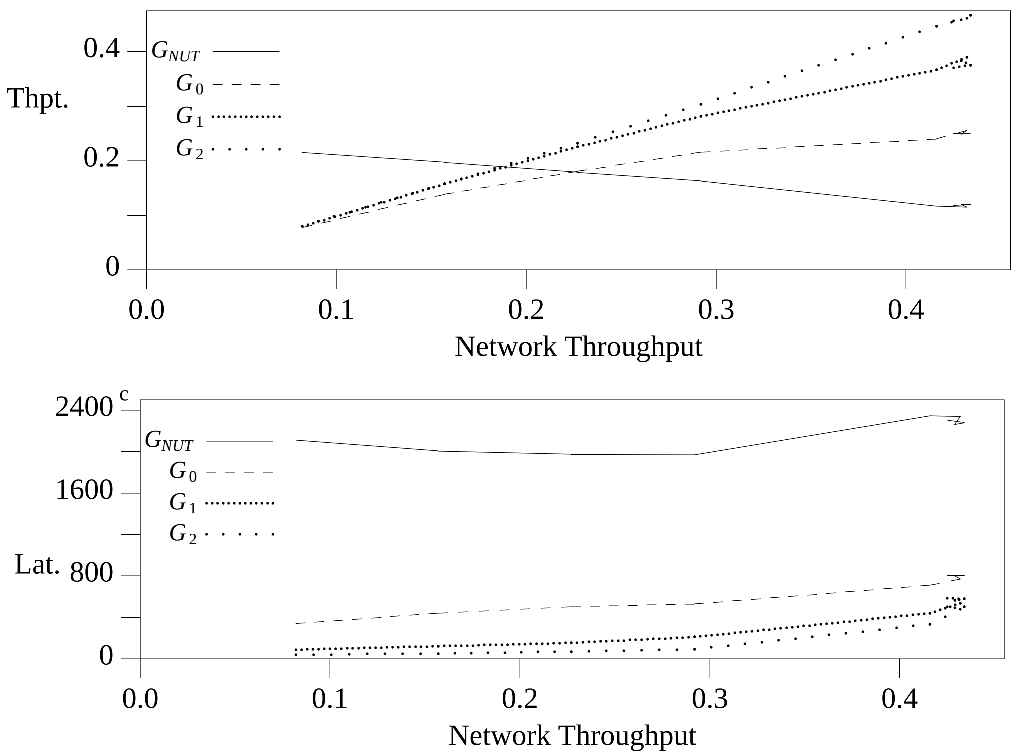


Figure 8.7: *Benchmark I: FIFO Buffers, Blocking Flow Control.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic ($G_{0\text{–}2}$). Switches with FIFO buffers, blocking flow control. Throughput and average latency vs. total network throughput, G_{NUT} and $G_{0\text{–}2}$.

Figure 8.7 shows the throughput of each sender group vs. the total network throughput and the average latency vs. total network throughput, for a network of FIFO switches using blocking flow control. The throughput of a sender group is its average per-sender throughput. The lowest network throughput occurs when the applied load for groups $G_{0\text{–}2}$ is 0.0 m/sc, the highest when all groups are at saturation throughput. The throughput of G_{NUT} drops as the applied load of the other three groups is increased because of the increasing contention in the network. At low applied loads, the throughput of G_{0-2} is proportional to the applied load. As the applied load (and, hence, throughput) is increased, G_0 is the first group impacted by the G_{NUT} traffic; at saturation, its throughput barely exceeds 0.20. The congestion caused by G_{NUT} has only minor impact upon G_2 — a network of FIFO switches under uniformly distributed traffic achieves 0.49, compared to 0.45 of G_2 in this benchmark.

This graph presents two of the performance objectives of interest. The first is the throughput degradation experienced by the sender groups G_{0-2}. In particular, G_0 achieves a saturation throughput which is a fraction of that achievable under uniform traffic (Fig. 8.5, FIFO data). We are looking for a flow control mechanism which will reduce the impact of congestion traffic on senders not participating in the congestion. The second performance objective is to maintain the throughput of data to the point of congestion (the hot spot, in this benchmark). When the applied load of $G_{0-2} = 0.0\,m/sc$, G_{NUT} operates a throughput of 0.235, which implies that the throughput of data to the hot spot is 0.94; since our transmission protocol requires that each link be idle for two clock cycles between each packet, with thirty-two byte packets the maximum throughput across a link is 0.941. When G_{0-2} are also operating at saturation throughput, the total throughput of data to the hot spot decreases to 0.893.

Figure 8.8 displays the same information as Fig. 8.7, but for DAMQ buffers with blocking flow control. A quick comparison of the two shows that the addition of non-FIFO buffering provides a significant boost in network performance. There is still some reduction of the performance of G_{0-2} due to the congestion caused by

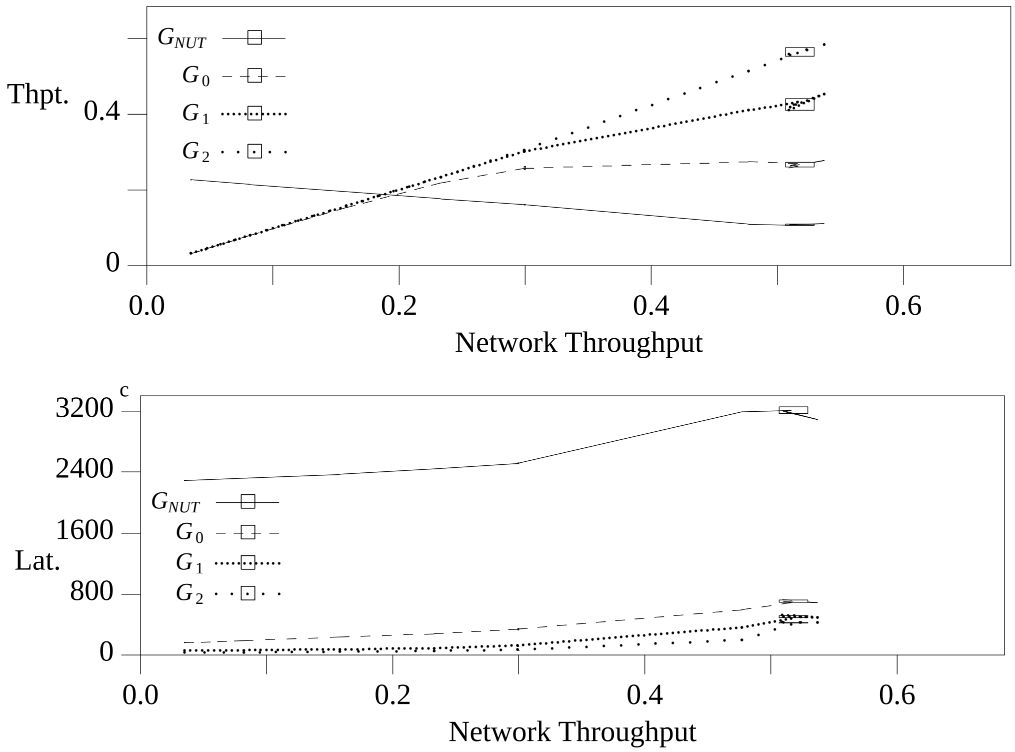


Figure 8.8: *Benchmark I: DAMQ Buffers, Blocking Flow Control.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic ($G_{0–2}$). Switches with DAMQ buffers, blocking flow control. Throughput and average latency vs. total network throughput, G_{NUT} and $G_{0–2}$.

G_{NUT}. Under uniform traffic conditions, a network of DAMQ switches with blocking flow control can achieve a throughput of 0.71; in this benchmark, G_2 is limited to 0.58, while G_0 only reaches 0.28. Further, the latency of G_0 is extraordinarily high. Even at low throughputs, where average latency of G_2 is 39 c (the minimum latency for four hops is 20 c), G_0 experiences an average latencies of ≈ 200 c for DAMQ buffered networks, 450 c with FIFO buffers. Transmission latencies of this order could significantly degrade the performance of fine-grained distributed programs.

While the saturation throughput of the network of DAMQ-buffered switches is higher than that of the network of FIFO-buffered switches, G_0 does not achieve significantly higher throughputs with DAMQ buffers than with FIFO. The throughput of the members of G_0 is determined by the available bandwidth of the output port of the switches in the first switching stage which leads to the hot spot, and the ability of packets from G_0 to compete with the hot spot packets for that bandwidth. DAMQ buffers do not and cannot help to resolve this — once the buffers of the first-stage switches connected to the members of G_0 become filled with packets destined for the congested output port, their throughput drops nearly to that of switches with FIFO buffers. The average latency of packets in the DAMQ buffer network is much lower, because packets routed to the other output ports are not queued behind the packets in the hot output port queue. The saturation throughput of the DAMQ buffer network is higher, because the back-pressure from the succeeding switch stages is lower than it is in the FIFO buffer network — the DAMQ buffer network maintains a throughput to the hot spot of 0.94.

Figure 8.9 shows the G_0 throughput vs. network throughput and G_0 latency vs. network throughput for switches with DAMQ buffers with blocking flow control, static partitioning (SAMQ and SAFC buffering), and maximum usage flow control with blocking thresholds of six and ten buffer blocks. This graph shows a significant improvement in G_0 throughput for both static allocation and maximum usage flow control mechanisms over blocking flow control (0.46 vs. 0.28). The throughput is still significantly lower than the throughput under uniform traffic,

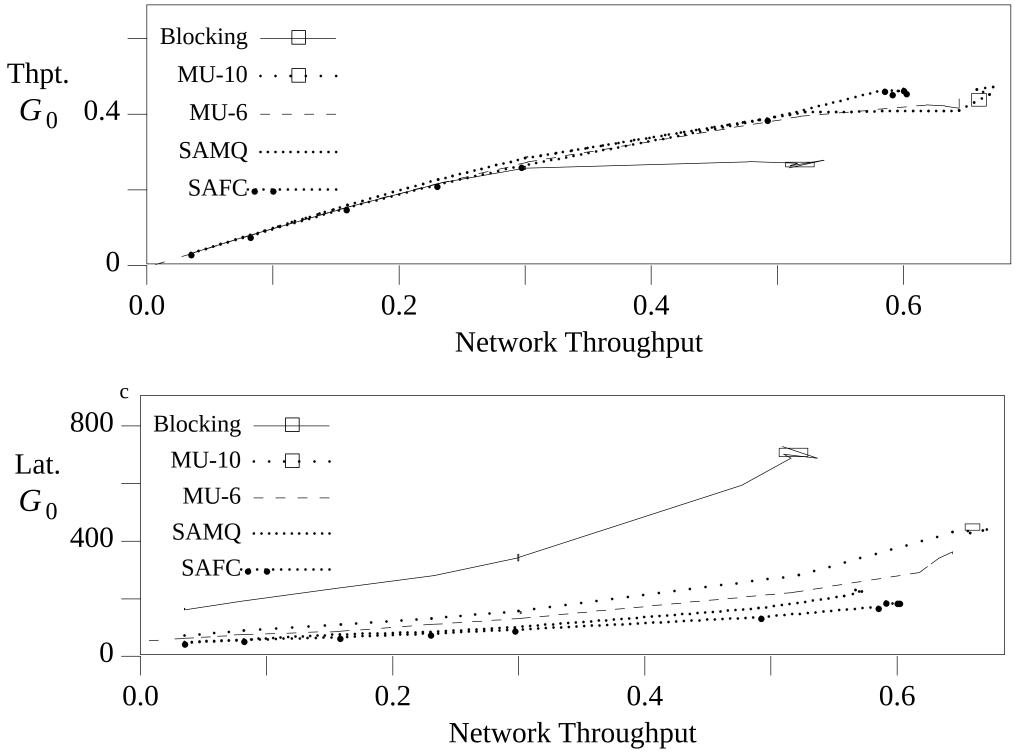


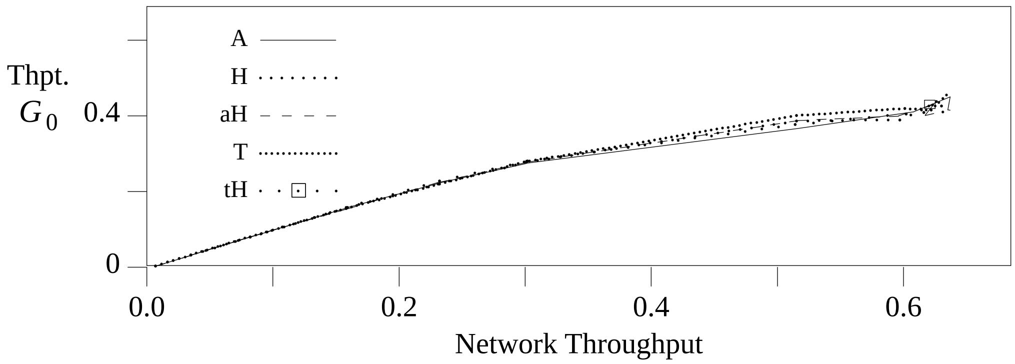
Figure 8.9: *Benchmark I: Maximum-Usage Flow Control.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_{0-2}). Shown is the G_0 throughput vs. network throughput and G_0 latency vs. network throughput for blocking flow control, static buffer allocation and maximum-usage flow control (blocking thresholds of six and ten buffer blocks).

however. The positive results achieved by maximum usage flow control under this benchmark are due to the fact the packets transmitted by the senders of G_0 conflicted with those from G_{NUT} at the output ports of the first switching stage. The flow control prevents the buffers in the second switching stage from becoming completely filled with packets destined to the hot spot, dramatically improving the performance of the G_0 senders. The heightened activity of the G_0 senders, in turn, increases the contention experienced by G_{NUT}, reducing its throughput and the network contention it causes. Thus, under this benchmark, the ‘‘hot’’ output port of the first-stage switches which are connected to G_0 and G_{NUT} senders are *the* critical points for flow control decisions.

Static buffer partitioning provides a lower G_0 average latency than does maximum usage flow control and a similar saturation throughput for G_0. However, it does not achieve as high a network saturation throughput. G_0 benefits not from the superior flow control properties of static buffer allocation, but from the fact that it causes the buffer space to be used less efficiently than does maximum usage flow control. The throughput of the G_1 and G_2 traffic is unduly throttled by this inefficient buffer utilization; while the throughput of G_0 increases due to the resulting reduction in network contention, the throughput of the network as a whole is reduced.

Fig. 8.10 shows latency vs. throughput for networks using two-counter (hysteresis) flow control, with a variety of parameter settings (these settings are presented in Sec. 8.4.3 and Tab. 8.2). Comparing these results to each other and to those in Fig. 8.9, it can be seen that (a) the parameter settings for two-counter flow control make almost no difference in the performance of the network in this benchmark and (b) the hysteresis in two-counter flow control provides no improvement in performance over maximum usage flow control. This may be due to the static nature of the congestion in this benchmark.

Figure 8.11 shows the throughput vs. network throughput and average latency vs. network throughput results of G_0 for destination- and path-based flow control. Destination- and path-based flow control both achieve a higher network throughput



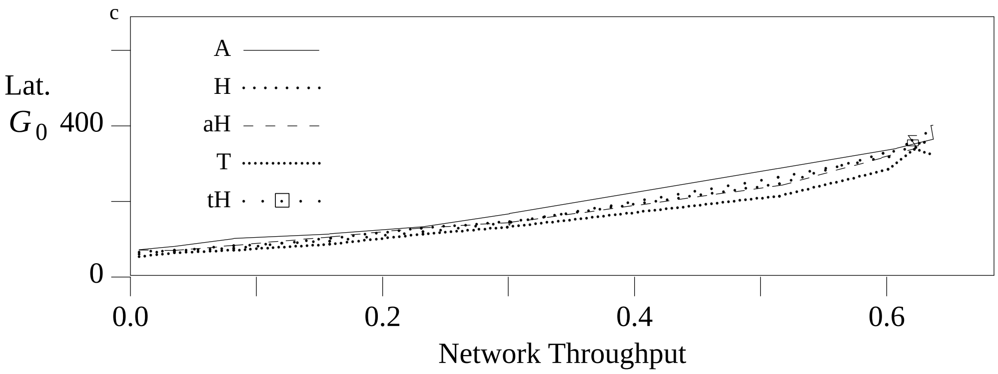


Figure 8.10: *Benchmark I: Two-Counter Flow Control.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_{0-2}). Shown is the throughput vs. network throughput and latency vs. network throughput of G_0 for two-counter flow control with a variety of parameters (see Tab. 8.2 for details of the parameters).

and a higher G_0 throughput than do the other flow control mechanisms. Destination-based flow control achieves the best results for G_0, saturating at a throughput of 0.80 for G_0. This throughput is higher than the saturation throughput of a blocking DAMQ buffer network under uniform traffic conditions. Destination-based flow control achieves this because the back-pressure from congestion in this flow control scheme is strongly directed — the *only* packets from G_0 which conflict with the G_{NUT} traffic are those addressed to the hot

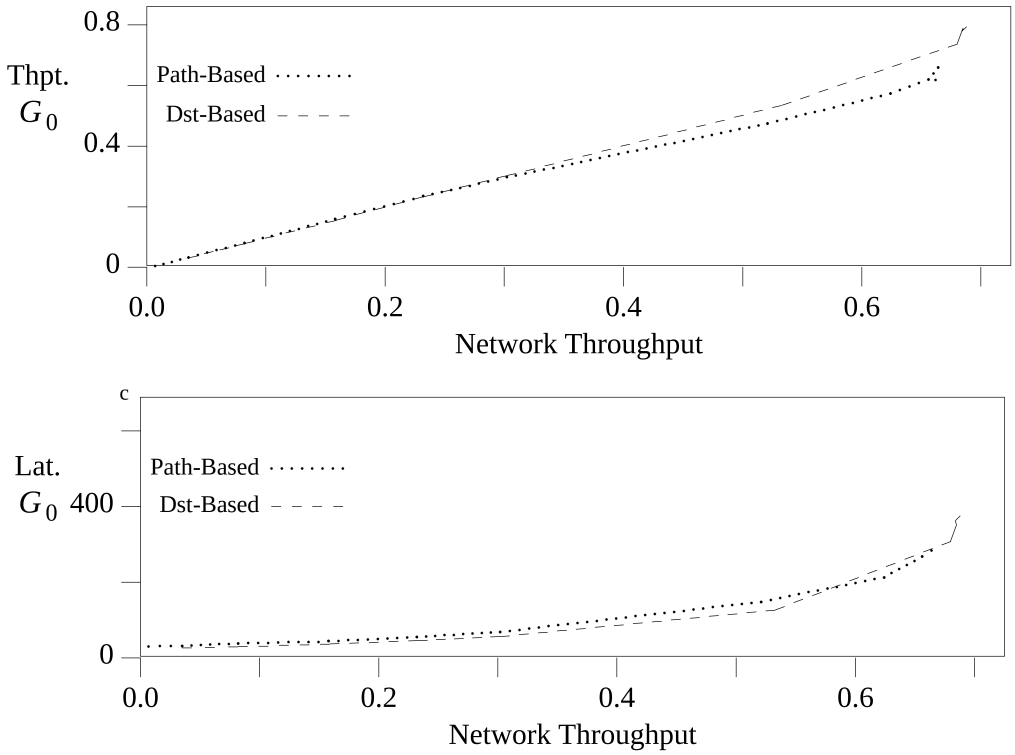


Figure 8.11: *Benchmark I: Link-Based Flow Control.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_{0-2}). Shown is the G_0 throughput vs. network throughput and G_0 latency vs. network throughput for destination-based and path-based.

destination. Since destination-based flow control severely restricts the throughput of G_{NUT}, G_0 experiences a situation where the first stage switches have four output ports but only three active input ports (the senders of G_{NUT} do not contend in a meaningful way for the output ports of the first stage switches). Thus, G_0 experiences less contention in this network than do the senders of a network transmitting uniformly-distributed traffic.

Under destination-based flow control, when all senders are transmitting at saturation throughput, G_{NUT} is reduced to a throughput of only 0.06. This is as compared to 0.10 for blocking flow control and 0.08 for maximum usage flow control, threshold of ten blocks. Despite the tight reigns on G_{NUT}, however, the throughput to the hot spot from all senders, when all senders are operating at saturation, is 0.93 for destination-based flow control, due to the high throughput of the other sender groups.

Under this benchmark, the four senders which comprise G_{NUT} create a point of congestion associated with a terminal link in the network (a hot spot). This congestion can be characterized as being static and associated with a destination node. These are ideal conditions for destination-based flow control, which is explicitly designed to handle hot spot conditions. Path-based flow control does well, but by comparing the next two hops of the packets to determine collisions, it mistakenly categorizes some ‘‘innocent’’ packets as contributing to the congestion, and thus throttles G_{0-3} traffic unnecessarily. The static nature of the congestion completely negates the effects of hysteresis — its performance is similar to maximum usage flow control. Maximum usage and static allocation flow controls are both improvements over blocking flow control, due to their ability to moderate the congestion in the first two switching stages of the network.

8.5.5. Benchmark II: Static Hot Spot, Many Participants

The previous benchmark examined network performance in the presence of a static hot spot caused by a small number of senders (four of the 256) sending 100% of their traffic to a single ‘‘hot’’ destination at saturation throughput. In this benchmark, a static hot spot is caused by directing half of the senders (G_{NUT}) to

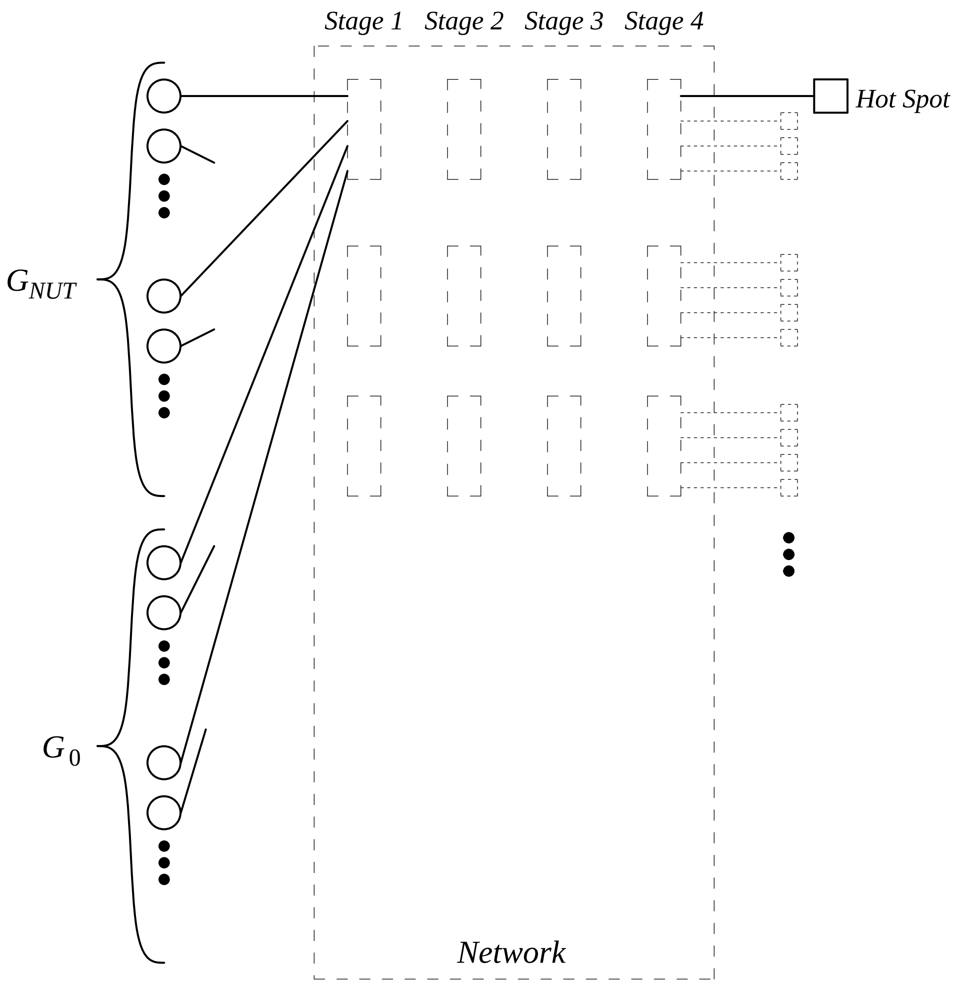


Figure 8.12: *Benchmark II: Static Hot Spot, Barrier Traffic.* Half of senders (G_{NUT}) operating at saturation, assigning uniformly-distributed destinations to 97% of packets generated, with the other 3% being sent to the hot spot. Other half of senders (G_0) transmit uniformly distributed traffic with graduated applied load. G_{NUT} and G_0 ‘‘mix’’ in first stage of omega (perfect shuffle) network.

transmit 3% of their packets to the hot destination, with the destinations of the other 97% being chosen from a uniform distribution of all possible destinations. Senders <0-127> comprise G_{NUT}, and senders <128-255> comprise G_0 (Fig. 8.12). Due to the perfect shuffle topology of the omega network, these two groups of senders ‘‘mix’’ at the first switching stage — every switch in the first switching stage is connected to two members of G_{NUT} and two members of G_0. All messages are thirty-two bytes long, and G_{NUT} transmits at saturation throughput. Since there are 128 senders in G_{NUT}, each sending 3% of their packets to the hot spot, and the bandwidth to the hot spot is 0.94, an upper bound for the throughput for G_{NUT} (T_{NUT}) can be found by the equation

$$0.94 \geq 0.97 \cdot (128/256) \cdot T_{NUT} + 0.03 \cdot 128 \cdot T_{NUT}.$$

The saturation throughput of G_{NUT} is less than or equal to 0.23.

There are three significant differences between the setup of this and the previous benchmark. The first difference is that contention for the hot spot occurs in all stages of the network in this benchmark. In the previous benchmark, G_0 absorbed the brunt of the performance penalty in the first switching stage. Second, all senders which are not members of G_{NUT} experience the same amount of contention, hence the single group of senders transmitting uniformly-distributed traffic (G_0). Third, G_{NUT} transmits packets to destinations other than the hot spot, and thus competes for some of the bandwidth of links not on the path to the hot spot.

Figure 8.13 shows the G_{NUT} and G_0 results for networks composed of FIFO and DAMQ switches with blocking flow control. The fact that throughput results for FIFO and DAMQ buffers are nearly identical highlights the fact that non-FIFO buffering does not prevent congestion propagation. The G_0 average latency is significantly lower for the network composed of DAMQ buffered switches than it is for the FIFO network. Multi-queue buffers prevent packets which are not

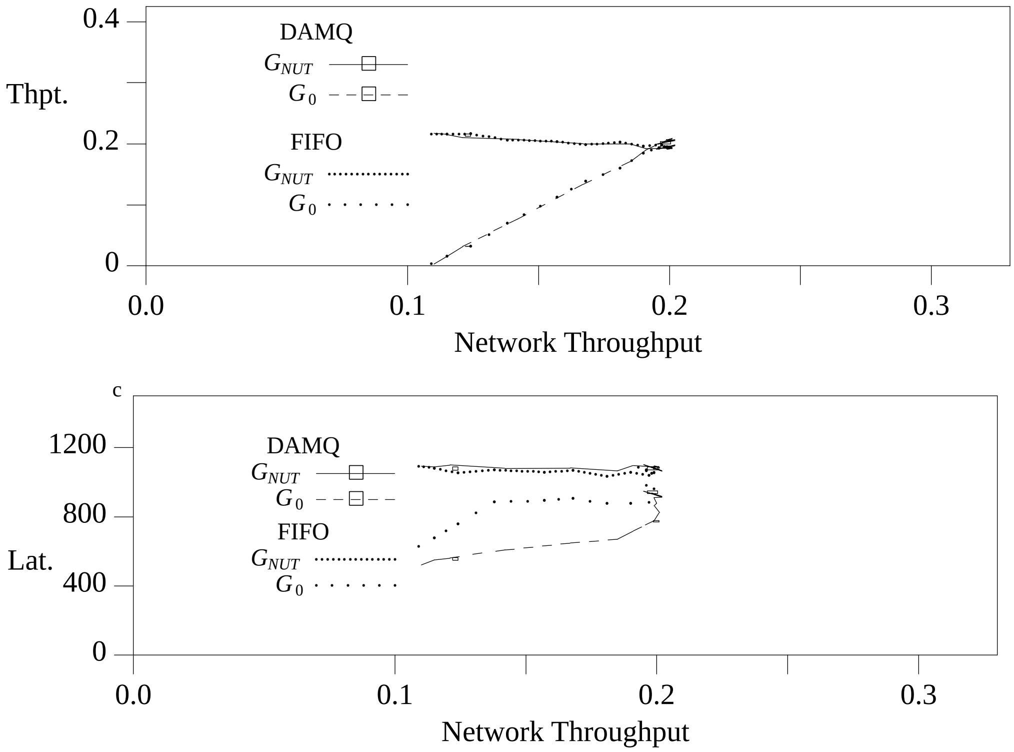


Figure 8.13: *Benchmark II: FIFO and DAMQ Buffers, Blocking Flow Control.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). FIFO and DAMQ Switches, blocking flow control. Throughput and average latency vs. total network throughput, G_{NUT} and G_0.

contending for the hot output port of a switch from being queued behind packets which are contending for the hot output port.

These results suggest three areas in which network performance could be improved under the congestion pattern of this benchmark: (a) the maximum throughput of G_0 can be increased (ideally to 0.71), (b) total network throughput can be increased and (c) average latency for G_0 can be reduced. One area which will not be significantly improved is hot spot utilization — if $T_{NUT} = T_0$, then the point of congestion is fully utilized when the throughput for G_{NUT} and G_0 is 0.20, which is achieved by blocking flow control.

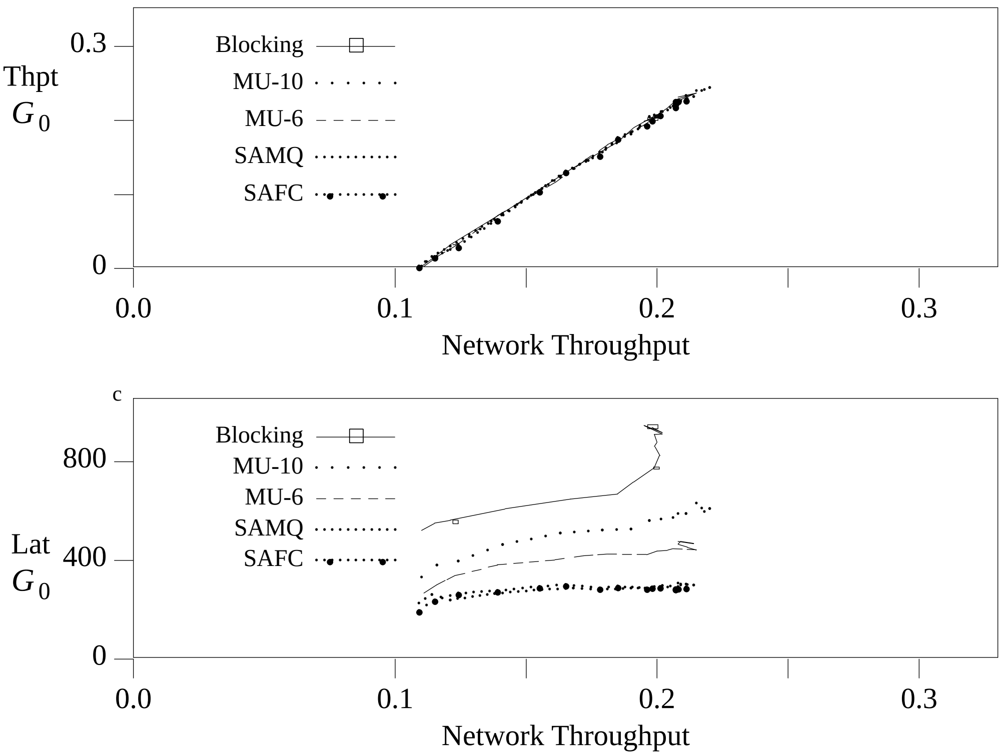


Figure 8.14: *Benchmark II: Maximum-Usage Flow Control, Group 0.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). DAMQ switches with blocking flow control, maximum usage flow control, and statically allocated buffers. Throughput and average latency vs. total network throughput, G_0.

Figure 8.14 shows the performance of the G_0 traffic for a network of DAMQ switches with maximum usage flow control with thresholds of ten and six buffer blocks, and for a network of multiqueue buffers with a static buffer allocation (SAMQ and SAFC buffers). In the first benchmark, we noted that these flow control mechanisms caused a measurable increase in the saturation throughput of

G_0 and significantly reduced its average latency. That is not the case for the current traffic pattern; while these flow control mechanisms do still reduce the latency, there is little improvement to the saturation throughput of G_0 or of the network as a whole.

In Benchmark I, the packets transmitted by the senders of G_0 conflicted with those from G_{NUT} at the first switching stage. At this point, because there were only four senders in G_{NUT}, the congestion traffic was ‘‘diluted’’, making the first switching stage *the* critical point for flow control decisions. In this benchmark, on the other hand, half of the senders are members of G_{NUT}, and there are points of congestion throughout the network, including at the final switching stage. Since the congestion is occurring at a distance from the senders and maximum usage flow control directs its back-pressure for only two hops, this traffic pattern results in tree saturation.

Figure 8.15 presents G_0 performance figures for two-counter (hysteresis) flow control. Results are given for a variety of two-counter parameter values. The behavior of each of the parameter assignments is presented in Tab. 8.2. As can be seen from Fig. 8.15, these behaviors all result in the same network performance. This stems from the fact that Benchmark II is a static congestion pattern. Hysteresis is a mechanism for differentiating between short-term variance and long-term changes in the value of a random function. Since this benchmark, like Benchmark I, creates a static congestion pattern, there are few dynamics upon which hysteresis can operate.

Figure 8.16† shows the G_0 throughput and latency vs. total network

† Note the change in scale between the graphs in Fig. 8.16 and the graphs in the previous figures in this section.

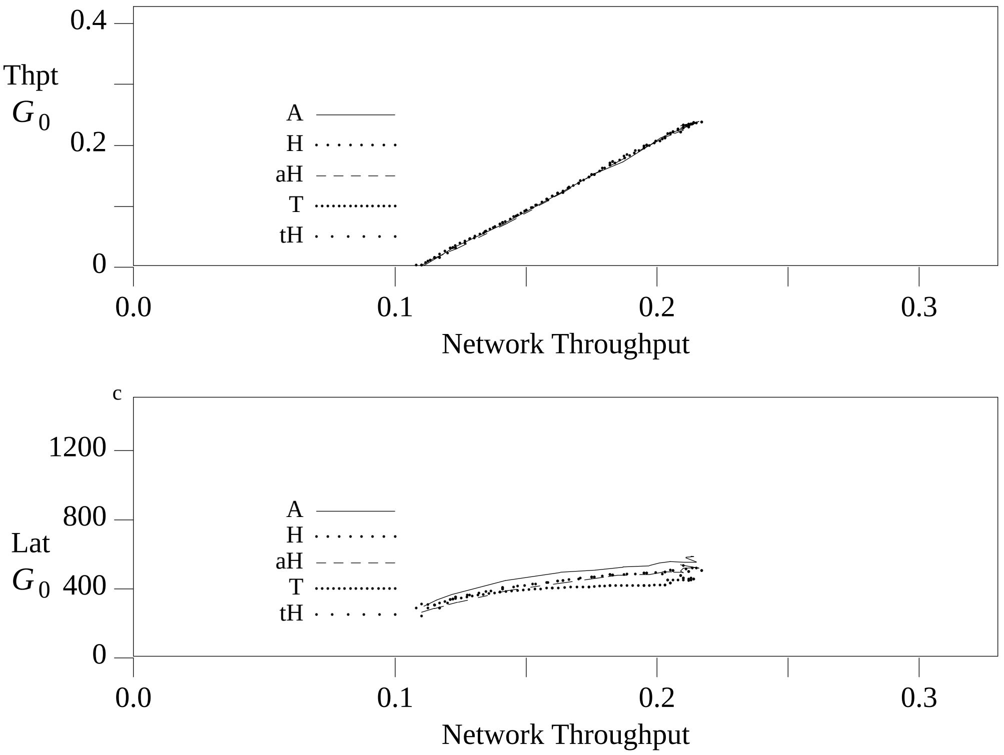


Figure 8.15: *Benchmark II: Two-Counter Flow Control, Group 0.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). Two-counter flow control, variety of parameters (Tab. 8.2). Throughput and average latency vs. total network throughput, G_0.

throughput for networks using destination-based flow control and path-based flow control. In the group throughput vs. network throughput graph, the destination-based and path-based flow control throughputs are identical up to a network throughput of 0.26. At that point, the network using path-based flow control saturates.

A result which surprised us is the degree to which destination-based flow control out-performed path-based flow control. Not only did destination-based

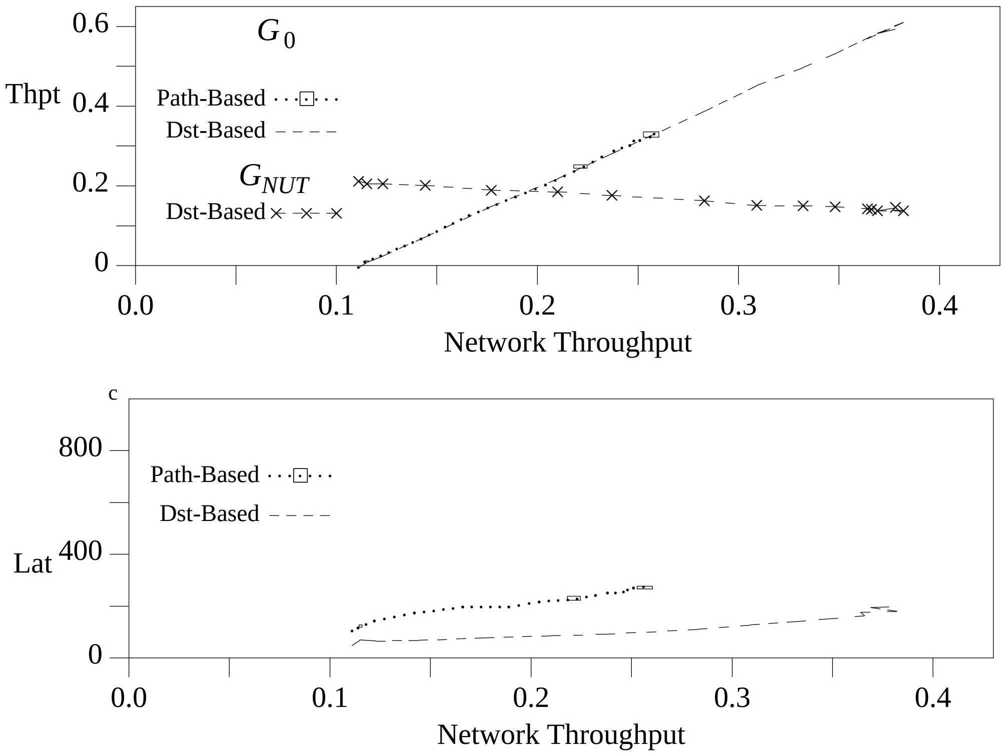


Figure 8.16: *Benchmark II: Link-Based Flow Control, Group 0.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). Shown is the G_0 throughput vs. network throughput and G_0 latency vs. network throughput for destination-based and path-based flow control. In the top graph, G_0 throughputs for destination-based and path-based flow controls are equal up to a network throughput of 0.26.

flow control support a G_0 saturation throughput *two times greater* than path-based flow control, but the average latency of G_0 traffic under destination-based flow control *at saturation throughput* (0.3 +) was lower than the average G_0 latency under path-based flow control *at throughputs approaching 0.2.* Given such a tremendous difference between the performance of G_0 traffic under destination-based flow control and any other flow control mechanism, one must ascertain

(a) that the total network throughput was not sacrificed to provide this level of G_0 performance and (b) that the point of congestion was not under-utilized.

Figure 8.16 shows that the total network throughput under destination-based flow control was also significantly higher than under the other flow control mechanisms. The throughput through the point of congestion (T_{PC}) when the network is operating at its maximum throughput can be calculated from the throughputs of G_0 and G_{NUT} at that point (0.60 and 0.15, respectively).

$$T_{PC} = \frac{128}{256} \cdot 0.60 + \frac{128}{256} \cdot 0.97 \cdot 0.15 + 128 \cdot 0.03 \cdot 0.15$$

$$= 0.94.$$

Thus, for this benchmark, the hot link is fully utilized under destination-based flow control.

Benchmark II presented markedly different results than did Benchmark I. The most significant difference was the ineffectiveness of all of the flow control mechanisms except destination-based flow control in this benchmark. Even path-based flow control, which performed comparably to destination-based flow control under Benchmark I, and which outperformed all other flow control mechanisms under this benchmark, fell significantly short of destination-based flow control. The reason for this was two-fold. First, the congestion was destination-oriented (a hot spot). Since destination-based flow control keys on packets' addresses to detect congestion, it has a clear advantage in terms of being able to detect the congestion in this benchmark. Second, the point of congestion was four hops away from the senders. For flow control mechanisms such as maximum usage, hysteresis and static buffer allocation, which only provide directed flow control for a single hop, having the point of congestion be this far from the senders allows tree saturation to occur.

The difference between the performances of destination-based and path-based flow control stems from the fact that path-based flow control suffers from ‘‘false blocking’’. Since it considers multiple packets traversing the same next n hops as being congested (i.e. it does not allow two such packets in a single buffer memory), and, for our simulations, n was set to two, all packets whose paths shared two or more links with the hot spot traffic were blocked as through they *were* hot spot traffic. This significantly reduced path-based flow control’s performance relative to destination-based flow control.

8.5.6. Benchmark III: Static NUT Spot

The first two benchmarks explored the utility of flow control mechanisms in situations where congestion was caused by contention for a terminal link of the network (i.e. a hot spot). Under those conditions, packets which are ‘‘involved’’ in the congestion can be identified by their destination address. Destination-based flow control does exactly that, and thus achieves higher performance for senders not participating in the hot spot (members of G_i). In this benchmark, G_{NUT} creates a non-uniform traffic spot internal to the network. This is done by directing senders <207>, <223>, <239> and <255> (G_{NUT}, for this benchmark) to transmit thirty-two byte messages to destinations randomly chosen from the range $(0 \cdots 15)$. The members of G_{NUT} were chosen such that their paths to any destination intersect at the second switching stage. By restricting the destinations

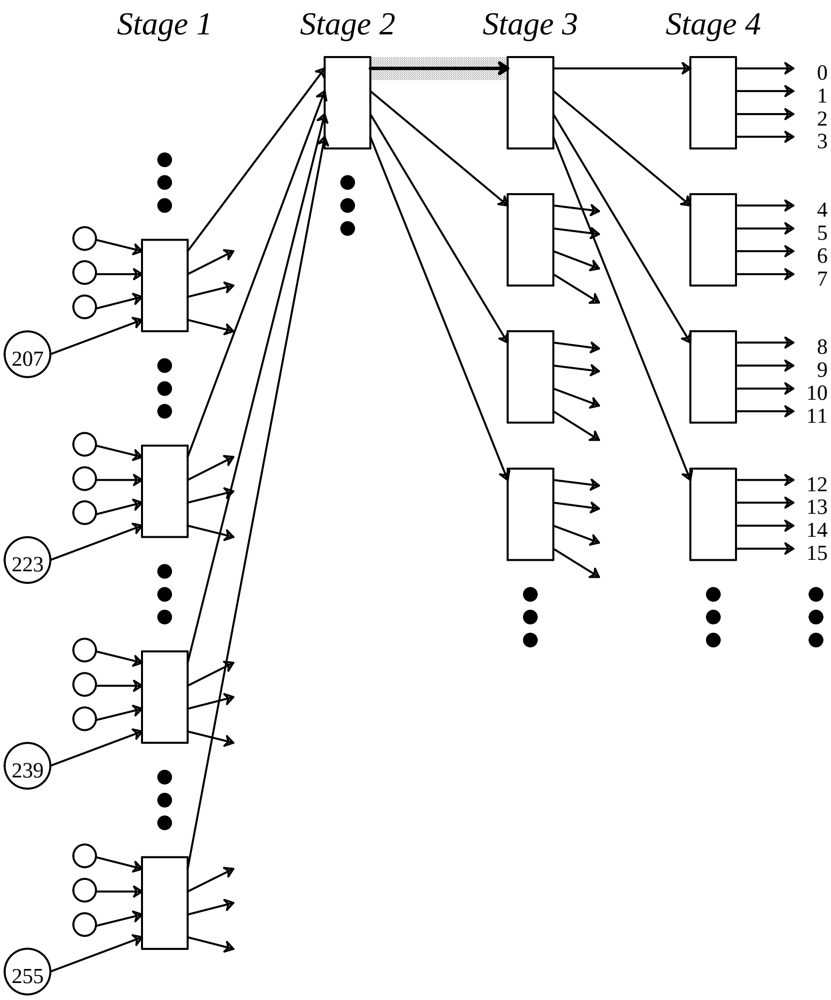


Figure 8.17: *Benchmark III: Non-Uniform Traffic Spot.* G_{NUT} (<207>, <223>, <239>, <255>), transmitting all packets to destinations $(0 \cdots 15)$, create a non-uniform traffic spot at *output port* #0 of *switch* #0 of the second switching stage.

of packets transmitted by G_{NUT} to the range $(0 \cdots 15)$, a non-uniform traffic spot is created at *output port* #0 of *switch* #0 of the second switching stage (Fig. 8.17). The NUT spot is traversed by packets addressed to destinations $(0 \cdots 15)$ from twelve other senders; these senders comprise G_0. The remaining senders (G_1) cannot reach the congested link, and are thus not negatively impacted by the congestion (in fact, the performance of the remaining senders is improved, since the existence of the congestion will limit the throughput of the senders in G_{NUT} and G_0).

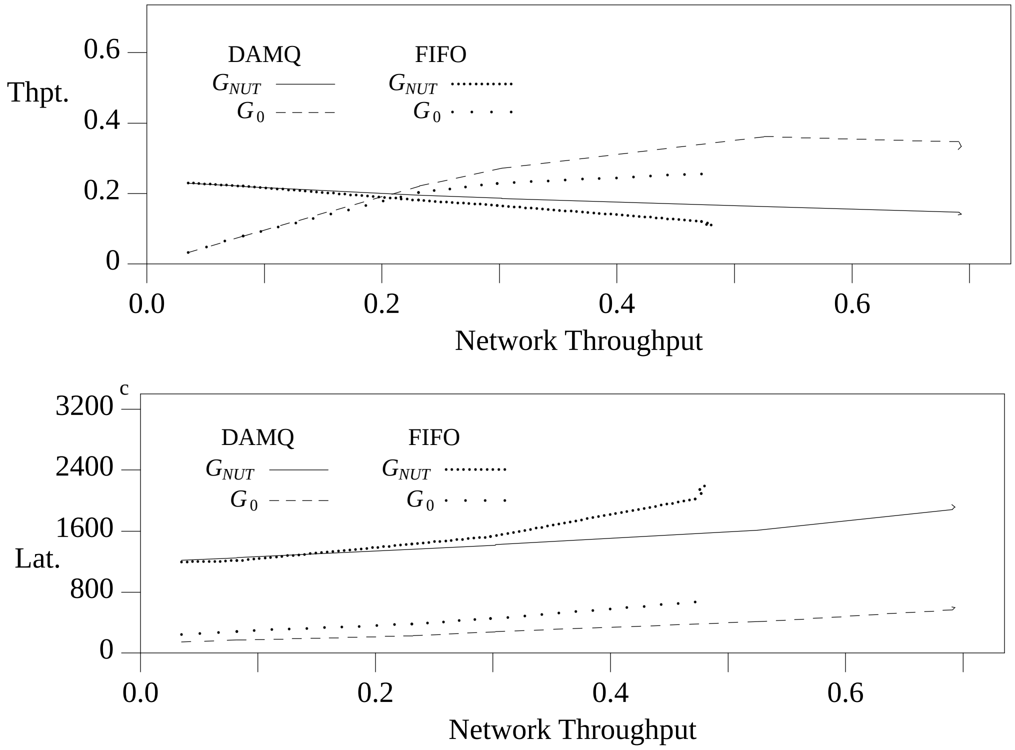


Figure 8.18: *Benchmark III: FIFO and DAMQ Buffers, Blocking Flow Control.* Congestion traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic ($G_{0,1}$). FIFO and DAMQ Switches, blocking flow control. Throughput and average latency vs. total network throughput, G_{NUT} and G_0.

Fig. 8.18 shows the results for networks of FIFO and DAMQ switches with blocking flow control (the results for G_1 are not shown, since G_1 does not interact with the congestion). As with Benchmark I, the point of highest throughput for G_{NUT} is when the applied load for $G_{0,1}$ is 0.0 m/sc — as the applied load for the other groups is increased, G_{NUT} loses network bandwidth and experiences higher latencies. This graph shows G_0 severely impacted by the NUT spot; G_0's throughput saturates at 0.36 and experiences significant degradation as the applied load is increased beyond that point (i.e. as the applied load is increased beyond the point of maximum throughput for G_0, the throughput of G_0 drops). The throughput degradation is due to contention between G_0 and G_1. Since G_1 is not impacted by the NUT spot, it achieves a higher packet throughput, as evidenced by the high total network throughputs achieved by both buffer types. Packets from G_0 do, however, interact with packets from G_1 at the third and fourth switching stages. Thus, at top throughputs, G_1 uses significant fractions of the available bandwidth at those stages, reducing the throughput of G_0.

Figure 8.19 shows the performance of networks using maximum usage flow control mechanisms in comparison to blocking flow control. The graphs present the throughput and latency of G_0 vs. total network throughput, as this provides the key performance metrics (total network throughput and degree of protection offered G_0). The figure shows maximum usage flow control significantly improving the throughput of G_0. This degree of success is due to the proximity of the point of congestion to the senders. The point of congestion is a link connecting the second and third switching stages; it is thus two hops from the senders. Maximum usage flow control, which determines whether or not to transmit packets based upon the output port of the *next switch* that the packet will traverse, provides *directed back pressure* (Sec. 8.4) for only a single hop. It is, thus, much more effective in situations where the senders which are causing the congestion are a small number of hops away from the point of congestion. Under this benchmark,

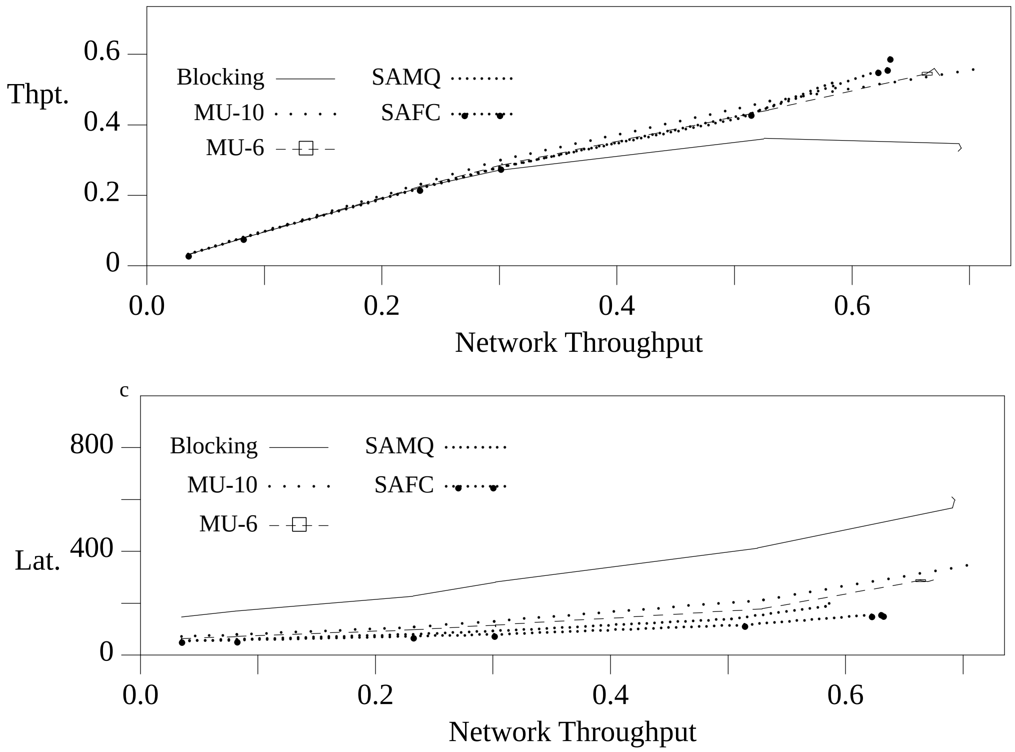


Figure 8.19: *Benchmark III: Maximum Usage Flow Control, Group 0.* Congestion traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic ($G_{0,1}$). Maximum-usage flow control with thresholds of ten and six buffer blocks and statically allocated buffers. DAMQ buffers with blocking flow control presented for comparison. Throughput and average latency vs. total network throughput, G_0.

switches in the first switching stage can distinguish between packets which are destined for the hot switch in the second stage and will traverse the congested link, and those destined for the hot switch but will not contest for the congested link.

Static buffer allocation and maximum usage flow control with a threshold of six buffer blocks, however, have significantly lower maximum network throughputs than does blocking flow control. This reflects blocking flow control's superior performance under uniform traffic conditions (i.e. G_1 performance), as will be seen in Sec. 8.5.9. Maximum usage flow control with a threshold of ten blocks demonstrates clear performance advantages over the other mechanisms in the figure; it achieves the highest network throughput, the highest G_0 throughput *and* an average G_0 latency as low as any of the four flow control mechanisms shown.

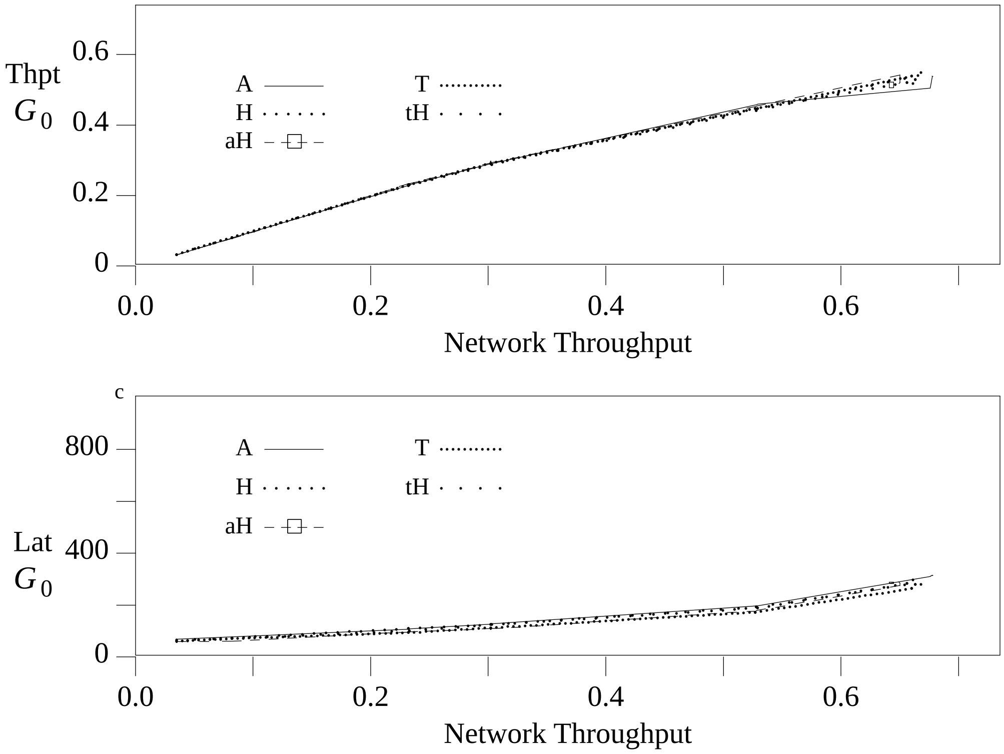


Figure 8.20: *Benchmark III: Two-Counter Flow Control, Group 0.* Congestion traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic ($G_{0,1}$). Two-counter flow control with a variety of parameters (Tab. 8.2) Throughput and average latency vs. total network throughput, G_0.

Figure 8.20 shows the performance for two-counter flow control with a variety of parameters (see Tab. 8.2). This being a static congestion pattern, we did not expect hysteresis to positively impact the performance of the flow control mechanisms. As the graphs show, the two-counter flow control mechanism behavior is similar to maximum usage flow control behavior, with the maximum network throughput falling short of that achieved by maximum usage, threshold of ten blocks.

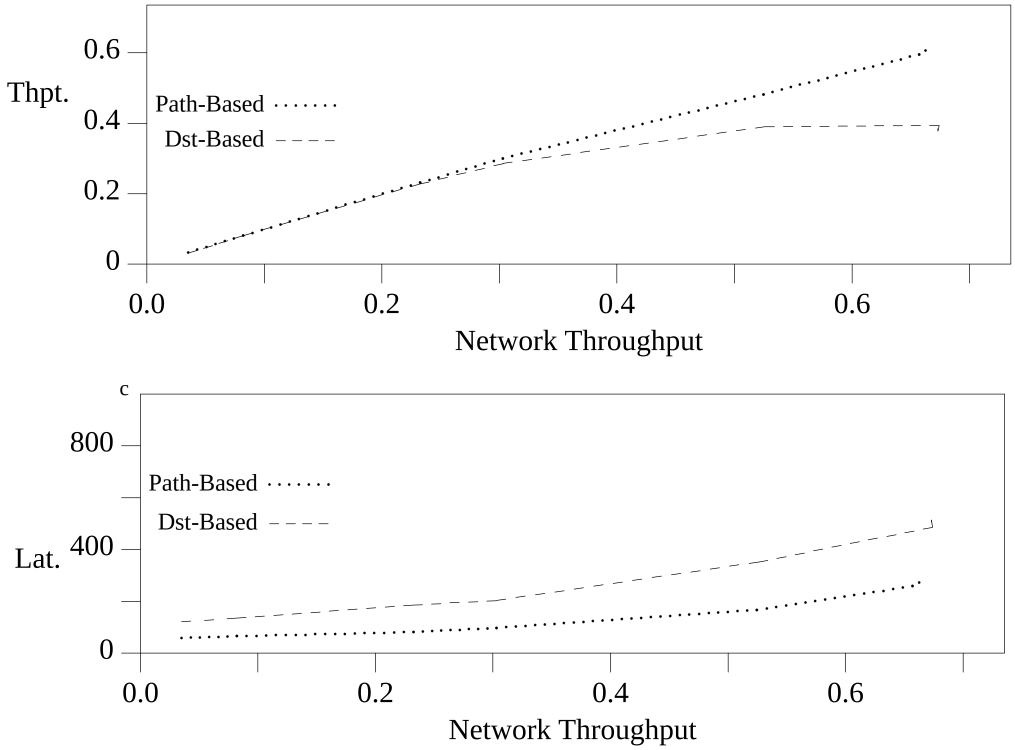


Figure 8.21: *Benchmark III: Link-Based Flow Control, Group 0.* Congestion traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic ($G_{0,1}$). Destination- and path-based flow control. Throughput and average latency vs. total network throughput, G_0.

Figure 8.21 shows the G_0 throughput and average latency vs. network throughput curves for destination-based and path-based flow control. Destination-based flow control performance is similar to that of blocking flow control; it achieves a slightly higher throughput for G_0 and a slightly lower maximum network throughput. This is due to the fact that destination-based flow control bases its flow control decisions on the destination addresses of packets. Since the congestion is not correlated to individual destinations, the flow control is ‘‘blind’’ to the congestion. Path-based flow control performs as well in this benchmark as in Benchmarks I and II — that is to say, it provides good protection for G_0, but does suffer some network throughput degradation relative to other flow control mechanisms. This follows from the fact that path-based flow control detects link contention, independently of where the link is located in the network. Thus, the congestion occurring in this benchmark is indistinguishable from the congestion experienced in the first two, from path-based flow control’s viewpoint.

This benchmark provided insights into the behavior of each of the flow control mechanisms under investigation. It demonstrated the reliance of destination-based flow control on there being a correlation between congestion and a particular destination address. If such a correlation does not exist, then destination-based flow control will be ineffective. Path-based flow control remains effective, due to its use of packets’ paths as a basis for flow control decisions. It suffers, however, from the same fault as static buffer allocation; it too often throttles flow when throttling is unnecessary, which negatively impacts the total network throughput. Buffer-based flow control schemes such as maximum usage and two-counter flow control are effective in situations where the point of congestion is physically close to the cause of the congestion, as is the case with this benchmark.

8.5.7. Benchmark IV: Four Senders with Long Messages

In this benchmark, the members of G_{NUT} transmit long (1024-byte) messages to randomly chosen destinations. These messages are broken into a sequence of thirty-two thirty-two byte packets — the packets which constitute a message are transmitted back-to-back at saturation throughput, and in this benchmark the messages themselves are generated with no idle time in between (saturation throughput). By attempting to utilize 100% of the available bandwidth on the path to their destination, each long message will cause congestion. There are four members of G_{NUT}: senders <240>, <244>, <248> and <252>. Due to the senders which were chosen to constitute G_{NUT}, two packets from G_{NUT} senders will contend for the output port of a switch only at the forth (final) switching stage. In other words, intra-G_{NUT} contention only occurs when two members of G_{NUT} transmit messages to the same destination in overlapping time periods. Thus, for the majority of the simulation time, there will be four independent heavily used paths through the network (one from each member of G_{NUT}).

The remaining senders are divided into three groups based upon their interaction with G_{NUT}. The twelve senders whose traffic intersects G_{NUT}'s traffic at the first switching stage are G_0, the forty-eight whose traffic intersects at the second stage are G_1, and the remaining 192 senders, whose traffic intersects G_{NUT}'s at the third switching stage, are G_2; this is identical to the sender grouping for Benchmark I. This benchmark is intended to test the flow control mechanisms in a dynamic environment, where the cause of congestion is transient. Also,

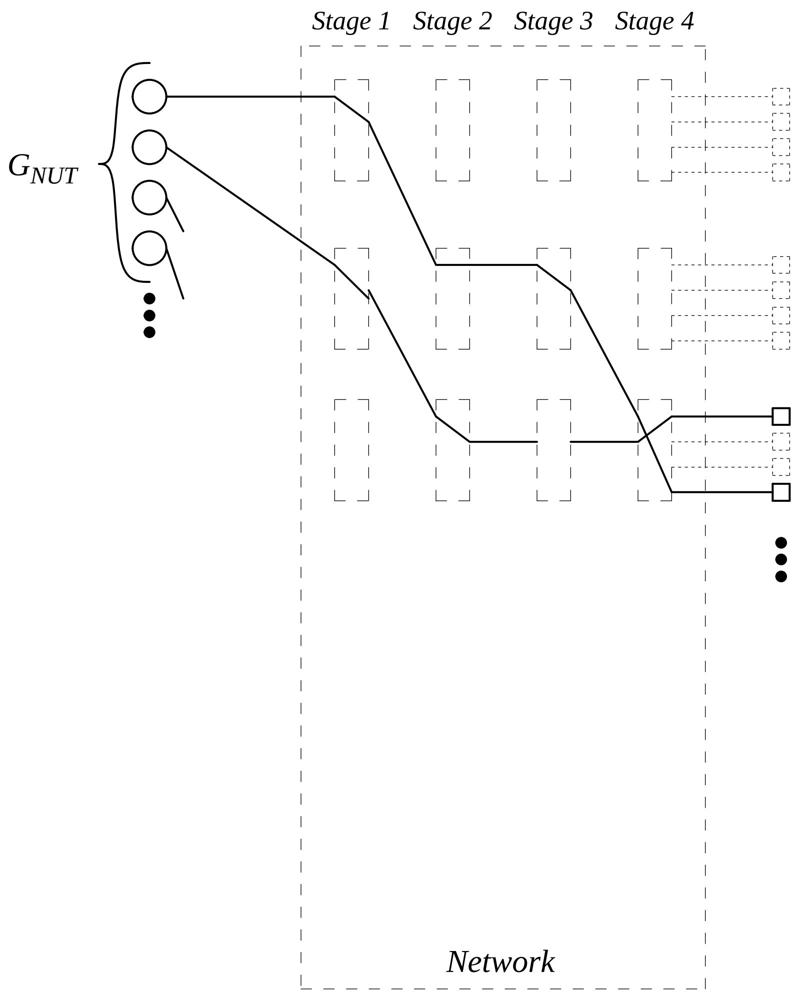


Figure 8.22: *Benchmark IV: Dynamic Serial Transmission.* Four senders (G_{NUT}) operating at saturation, assigning uniformly-distributed destinations to 1024 byte messages.

creating streams of packets from a source to a particular destination mimics events which occur in multicomputers such as remote paging, cache flushing or file I/O.

Figure 8.23 shows the throughput and the average latency of traffic from each of the four groups of senders for a network of DAMQ switches using blocking flow control. The results suggest that the congestion caused by this benchmark degrades the performance to the network as a whole far less than did the static hot or NUT

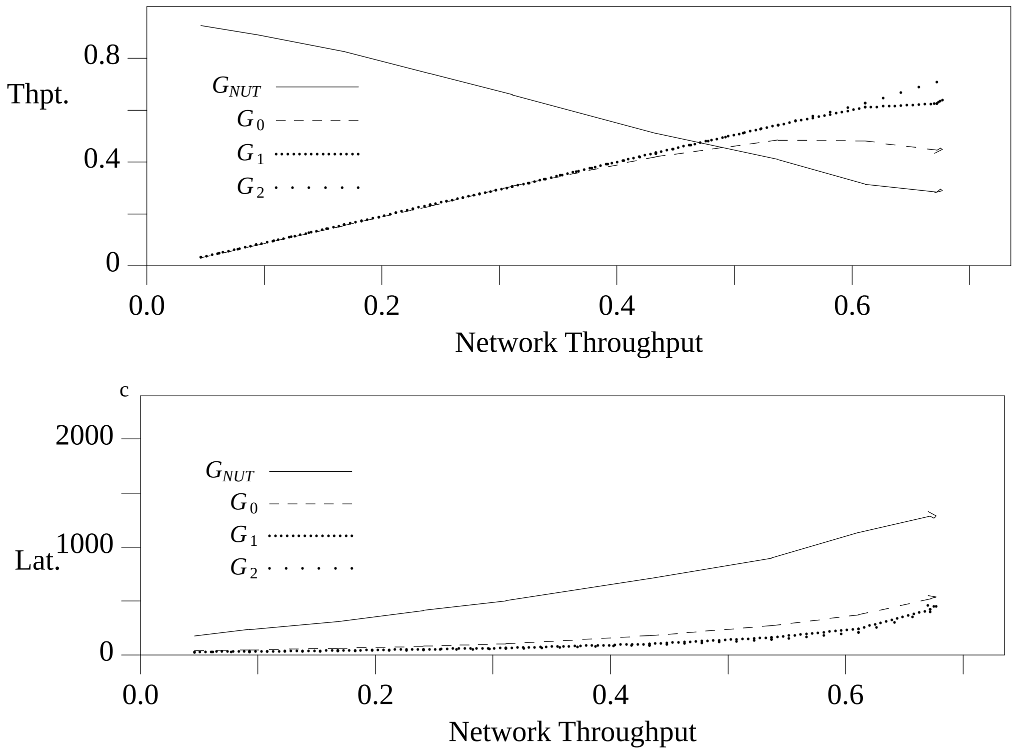


Figure 8.23: *Benchmark IV: DAMQ Buffers, Blocking Flow Control.* Serial traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_{0-2}). Switches with DAMQ buffers, blocking flow control. Throughput and average latency vs. total network throughput, G_{NUT} and G_{0-2}.

spots of the previous benchmarks. The graphs show G_0 suffers the greatest impact, with its saturation throughput reduced to a maximum of 0.49, a forty percent reduction compared to the throughput of a system with a uniform traffic distribution. In addition, the point of maximum G_0 throughput does not correspond to the maximum network throughput; beyond the point at which the G_0 throughput peaks, contention from $G_{2,3}$ (which compete with $G_{0,1}$ for network resources only *after* the first switching stage) reduces the G_0 throughput.

G_1 is minimally impacted by G_{NUT}, with its throughput dropped to 0.64, and G_2 is not impacted at all, achieving a throughput of 0.71. The reason for this is that, while G_{NUT} reduces the throughput of G_0, G_0 does the same to G_{NUT}. Congestion at the first switching stage reduces the applied load of G_{NUT} traffic on the second switching stage, decreasing its impact on G_1. Similarly, contention between G_{NUT} and G_1 at the second switching stage reduces G_{NUT} traffic to the point that G_2's throughput is not effected.

Figure 8.24 compares the G_0 results of blocking flow control to results for maximum usage flow control (with thresholds of ten and six buffer blocks) and to static buffer allocation (SAMQ and SAFC buffering). Static buffer allocation and maximum usage flow control provide similar protection for the performance of G_0; the maximum throughput of G_0 is increased to ≈ 0.58 (slightly higher for SAFC buffers, slightly lower for SAMQ buffers and maximum usage flow control), and the average latency of G_0 is cut to nearly half of what it is for blocking flow control. By limiting the amount of buffer space which can be occupied by a single queue, these forms of flow control prevent the buffers of the second-stage switches from being filled with G_{NUT} packets, thus allowing the G_0 traffic which will take a different second-stage output port to bypass the congestion at that switching stage. This improvement is not enough, however, to offset the performance degradation caused by static buffer allocation under uniform traffic conditions; SAMQ and SAFC buffering support significantly lower maximum network throughputs under this benchmark than does DAMQ buffers with blocking flow control. Maximum usage flow control, on the other hand, increases the maximum throughputs of both G_0 and the network as a whole relative to blocking flow control; the impact of

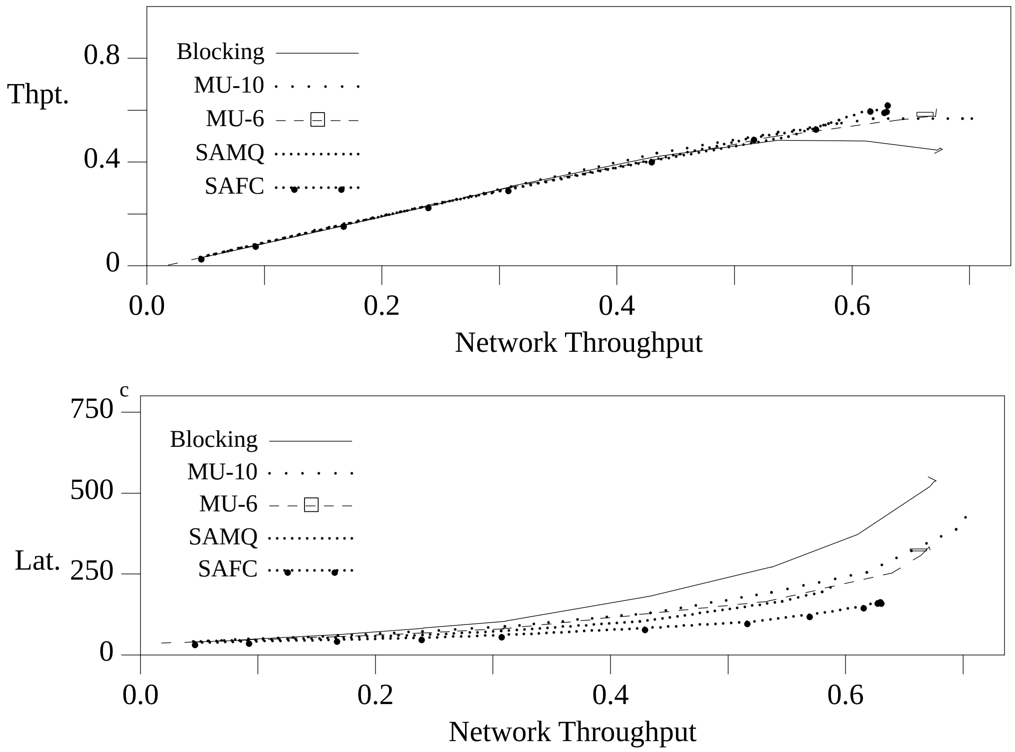
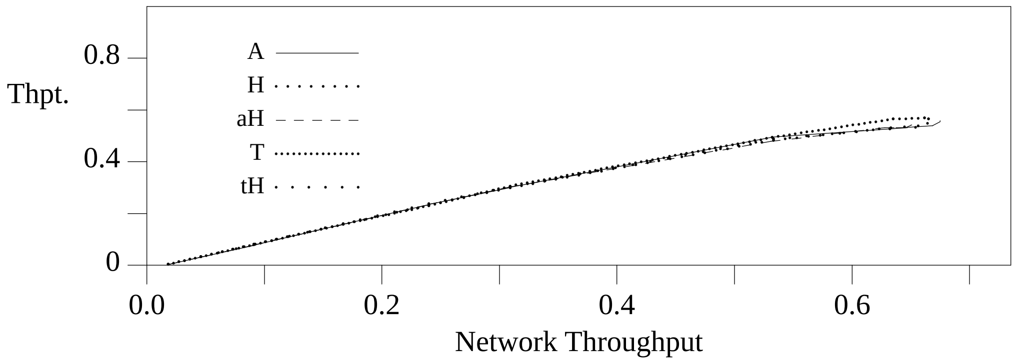


Figure 8.24: *Benchmark IV: DAMQ Buffers, Maximum-Usage Flow Control.* Serial traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_{0-2}). Blocking, maximum-usage flow control and statically allocated (SAMQ and SAFC) buffers. Throughput and average latency vs. total network throughput, G_0.

maximum usage flow control on uniformly-distributed traffic is presented in Sec. 8.5.9.

The performance of two-counter (hysteresis) flow control is shown in Fig. 8.25. Simulations were run for this benchmark using a variety of parameter values (Tab. 8.2). As was seen in the previous benchmarks, there is little difference in the degree to which two-counter flow control with the different parameter values promote the performance of the groups of senders transmitting



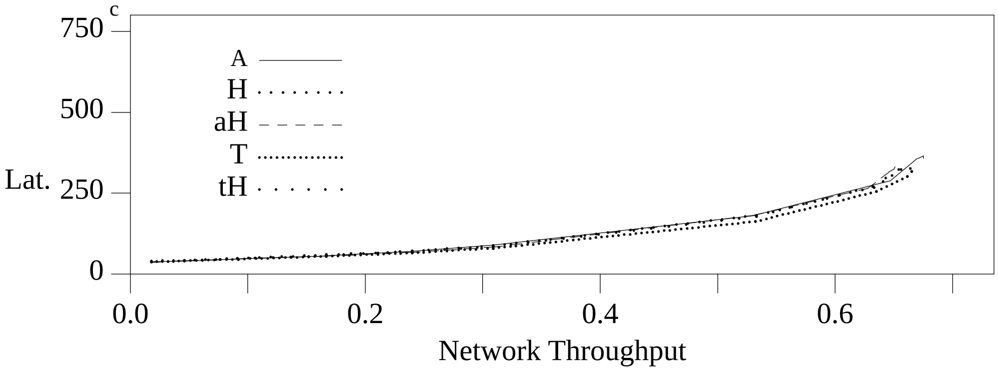


Figure 8.25: *Benchmark IV: DAMQ Buffers, Two-Counter Flow Control.* Serial traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_{0-2}). Shown is the throughput vs. network throughput and latency vs. network throughput of G_0 for two-counter flow control with a variety of parameters (see Tab. 8.2 for details of the parameters).

uniformly-distributed traffic. In this benchmark, the best two-counter flow control results are similar to that of maximum usage flow control with a threshold of six buffer blocks. With the previous three benchmarks the lack of impact of hysteresis on network performance was not surprising, due to the static nature of the congestion in those benchmarks. Benchmark IV does create dynamic congestion patterns. The lack of impact of hysteresis here indicates that, perhaps, the congestion patterns or not ‘‘dynamic enough’’ for hysteresis to be helpful. This possibility is examined further with Benchmark V, where the congestion patterns are more dynamic.

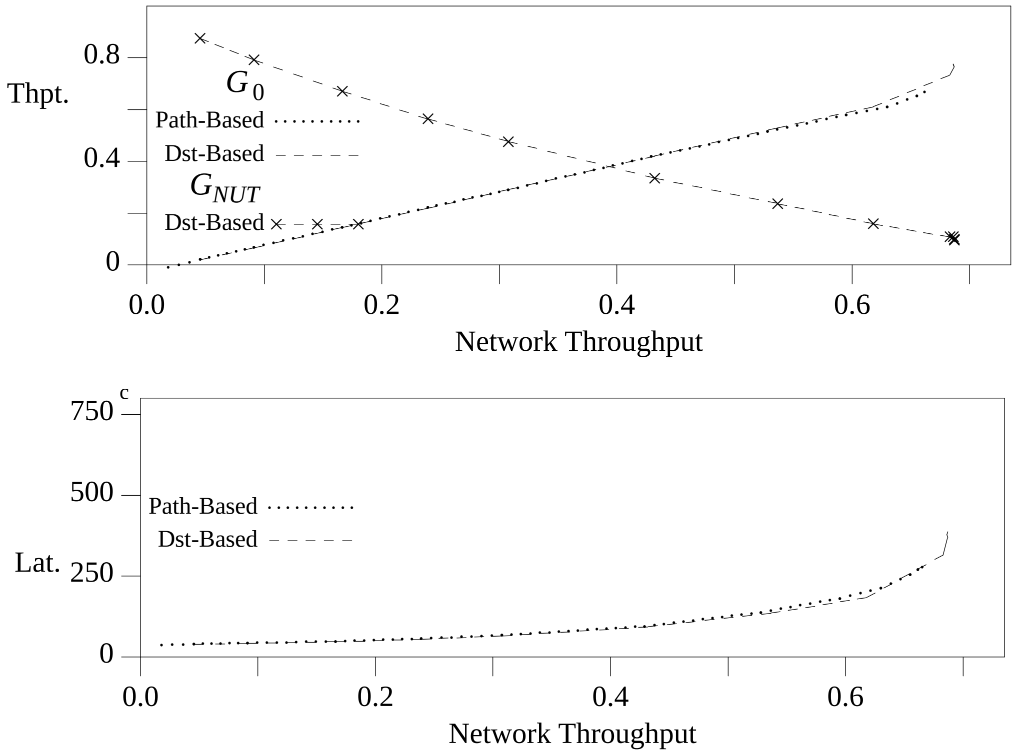


Figure 8.26: *Benchmark IV: DAMQ Buffers, Link-Based Flow Control.* Serial traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic ($G_{0–2}$). Destination- and path-based flow control. Throughput and average latency vs. total network throughput, G_0.

Figure 8.26 shows the performance of G_0 for networks implemented with destination-based flow control and path-based flow control. Path-based flow control allows G_0 to transmit data at a throughput of ≈ 0.69, and destination-based flow control supports a G_0 saturation throughput of 0.79. These flow control mechanisms achieve high throughputs for G_0 by throttling the throughput of G_{NUT}. As Fig. 8.26 shows, destination-based flow control throttles the G_{NUT} traffic as low as 0.11 in response to contention with traffic from the other sender groups. Whether or not throttling G_{NUT} to this degree creates a ‘‘fairness problem’’ is an application-specific question; in the context of this evaluation, the total network throughput must be the basis for comparing the flow control mechanisms. While the high throughput achieved by G_0 under both destination- and path-based flow control almost offsets the severe throttling of G_{NUT}, the maximum total network throughput is lower for these flow control mechanisms than it is for maximum usage flow control with a threshold of ten buffer blocks.

This benchmark was a departure from the previous three benchmarks on two fronts. First, it created only mild network congestion — DAMQ buffers with blocking flow control achieve a network throughput of 0.68, only a bit shy of the 0.72 achieved under uniform traffic conditions. Benchmark III also minimally impacted network throughput, but for a very different reason. In Benchmark III, the point of congestion was unreachable by three fourths of the senders. Here, the congestion impacts every sender; it simply does not impact them very much. Second, the traffic pattern was dynamic. This caused points of congestion to be created and destroyed over the course of the simulation. In theory, one of the flow control characteristics this benchmark was to demonstrate was their ability to adapt under dynamic conditions.

This benchmark exposed the tendency of destination-based and path-based flow controls to over-throttle serial communication. This extreme throttling was a positive feature in Benchmarks I and II, but under traffic conditions in which high-throughput serial transmissions can coexist with the uniformly distributed traffic, this can cause a drop in performance.

In contrast, maximum usage flow control with a threshold of ten buffer blocks minimally throttles G_{NUT} and achieves the highest maximum network throughput of any of the flow control mechanisms for this benchmark. This mechanism depends upon congestion between G_{NUT} and G_0 at the first switching stage to throttle G_{NUT}, which allows G_{NUT} to utilize as much network bandwidth as it can. Further, by preventing any single packet queue from occupying an entire buffer, G_0 is not severely impacted by the G_{NUT} traffic (as it is under blocking flow control).

8.5.8. Benchmark V: 128 Senders Transmitting Multi-Packet Messages

For the fifth benchmark, G_{NUT} consisted of 128 (half) of the senders in the network. These senders cause congestion by transmitting 256-byte messages to uniformly distributed destinations at saturation throughput. Each message is split into eight-packet sequences. The other half of the senders transmit thirty-two-byte packets to uniformly-distributed destinations with a graduated applied load. As with Benchmark II, the senders are chosen such that G_{NUT} and G_0 ‘‘mix’’ in the first switching stage; each switch in *stage #1* receives input from two members of G_{NUT} and two members of G_0.

In this benchmark, there are two sources of congestion. The first is that the 256-byte messages generated by G_{NUT} can completely fill a buffer if the packet sequence encounters contention for a link. The second source is that the packet sequences which constitute a 256-byte message heavily utilize individual links for extended periods of time. This is similar to the congestion in Benchmark IV, except that (1) the congestion pattern in this benchmark is more dynamic and

(2) messages from G_{NUT} can collide with one another at any point in the network, and do so more frequently in this benchmark than in the previous. As with the previous benchmark, the three metrics used to evaluate the flow control mechanism's performance are the maximum total network throughput (the throughput of G_{NUT} and G_0 combined), average latency of the uniform traffic vs. network throughput, and the saturation throughput of G_0.

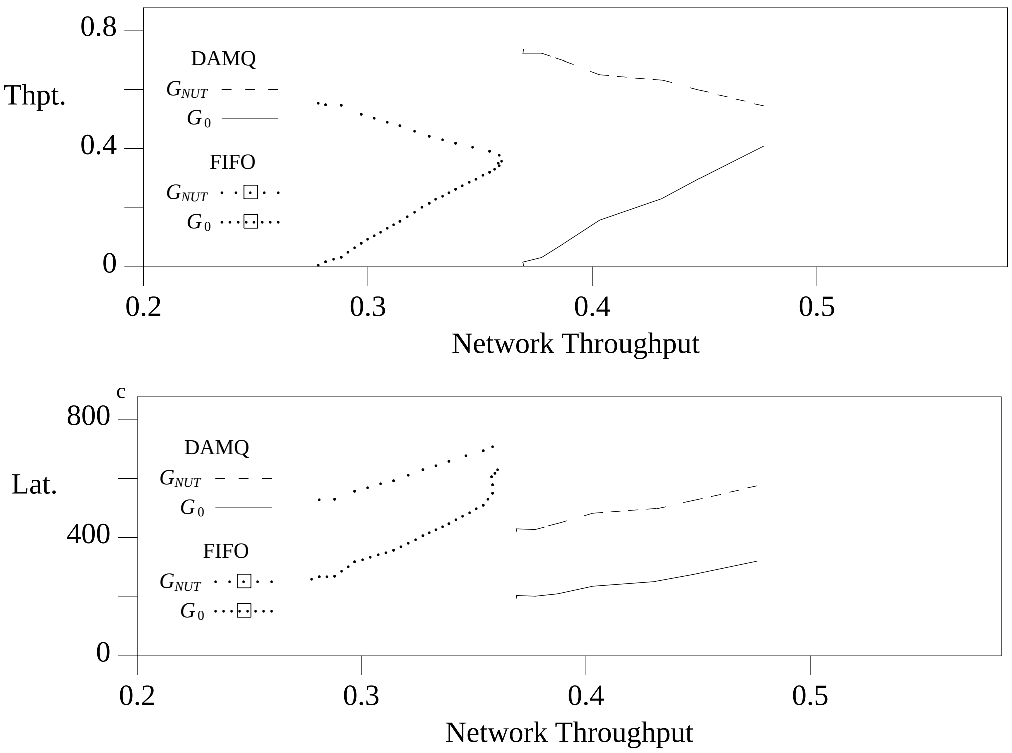


Figure 8.27: *Benchmark V: Blocking Flow Control.* Congestion traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). Switches with FIFO and DAMQ buffer, blocking flow control. Throughput and average latency vs. total network throughput, G_0 and G_{NUT}.

Fig. 8.27 shows the throughput and average latency vs. system throughput of

G_{NUT} and G_0 for networks of FIFO and DAMQ switches with blocking flow control. As previously mentioned, G_{NUT} operates at saturation throughput while the applied load of G_0 is graduated, G_{NUT} achieves its highest throughput and lowest latency when the applied load of G_0 is 0.0 m/sc (G_0 throughput equals 0.0 b/ls). As the applied load of G_0 is increased, the G_{NUT} traffic encounters more contention, and its throughput drops. Unlike the first four benchmarks, the G_{NUT} throughput remains higher than that of G_0 under blocking flow control, even when G_0 is at saturation throughput.

As can be seen from the figure, DAMQ buffers provide superior performance over FIFO buffers for both G_0 and G_{NUT} senders. Since the network congestion is highly dynamic, a buffering scheme which reduces the effect of output port contention makes a significant impact on network performance. The network of DAMQ buffers achieves a total network throughput of 0.48, compared to 0.36 for the network of FIFO buffers. While the DAMQ buffered network improves the throughput of both G_0 and G_{NUT} traffic, it does *not* provide a significant improvement in latency for the traffic of either group. Further, both the total network throughput and the G_0 throughput fall well short of the 0.71 throughput achievable under uniform traffic conditions.

Fig. 8.28 shows G_0 throughput vs. system throughput and G_0 latency vs. system throughput for DAMQ switches with blocking flow control, maximum usage flow control with thresholds of ten and six blocks, and static buffer allocation (SAMQ and SAFC buffers).

SAMQ buffering provides a lower G_0 average latency and slightly higher G_0 throughput than DAMQ buffering with blocking flow control. The graph indicates,

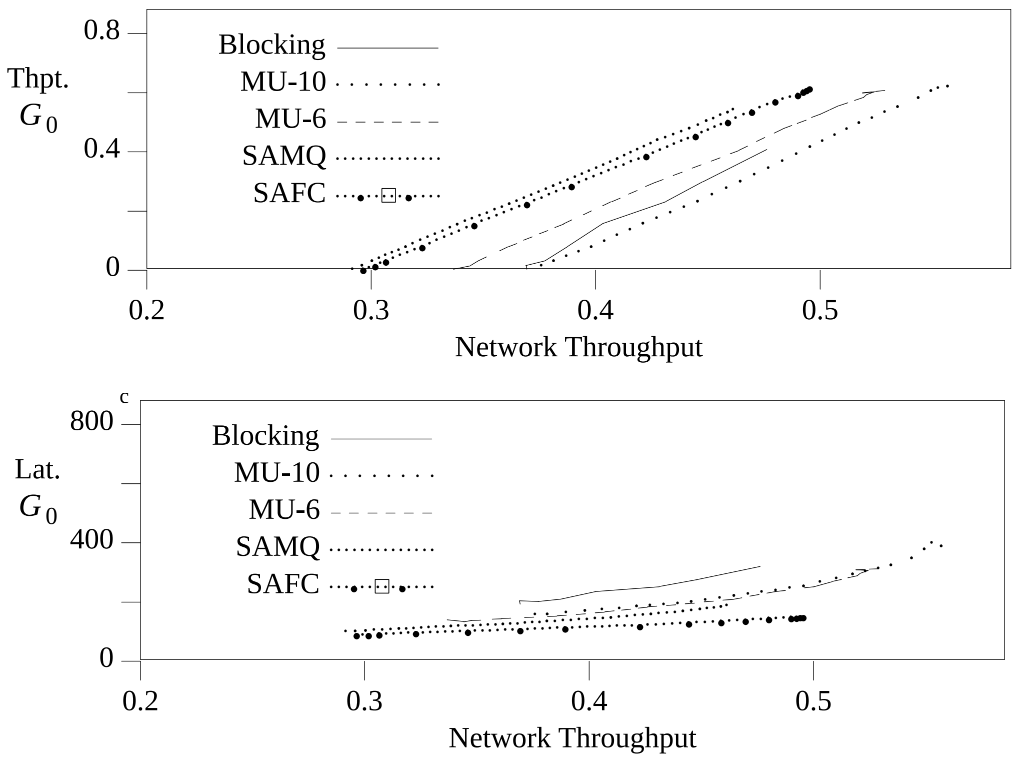


Figure 8.28: *Benchmark V: Maximum Usage Flow Control.* Non-uniform traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). Shown is the G_0 throughput vs. system throughput and G_0 latency vs. system throughput for blocking, static allocation and maximum usage flow control (six and ten buffer maximums).

however, that the SAMQ switch network has a lower network throughput — it has a much lower saturation throughput for G_{NUT}, which lowers network maximum throughput. The SAMQ switch network performs as well as the DAMQ switch network on G_0 traffic only because it so strongly throttles the G_{NUT} traffic, reducing the contention that the G_0 traffic encounters from the G_{NUT} messages.

While exhibiting better performance, the SAFC switch displays behavior similar to that of the SAMQ switch. By over-throttling G_{NUT}, the network of SAFC switches achieves a maximum throughput for G_0 traffic equal to that for maximum usage flow control, and the average latency is significantly lower (Fig. 8.28). However, the total network throughput is lower for the SAFC switch network (0.50 vs. 0.55 for maximum usage flow control). If one considers the maximum network throughput to be *the* metric of interest, SAFC switch performance falls measurably short of the mark. If the latency of G_0 traffic is the primary concern, then SAFC buffering is preferable over maximum usage flow control for this benchmark.

Maximum usage flow control performance is superior to blocking flow control for this benchmark; the average G_0 latency is lower, and the maximum throughputs for both G_0 and the network as a whole are higher for maximum usage than for blocking flow control. Comparing maximum usage blocking thresholds, the ten-block congestion threshold supports higher total network throughput than a threshold of six buffer blocks while they both achieve approximately the same maximum G_0 throughput. Thus, it is the throughput of G_{NUT} traffic which benefits the most from the higher threshold. The higher threshold reduces the number of switches that the packets which constitute a message occupies; this, in turn, increases the probability that a G_{NUT} sender will be able to inject a message into the network in its entirety, without halting. The throughput and latency of G_0 seem to be indifferent to whether the threshold is ten or six blocks.

If there is a condition under which hysteresis is beneficial to a hop-level flow control mechanism, it is under dynamic congestion conditions such as this benchmark. We found, however, that two-counter (hysteresis) flow control is less

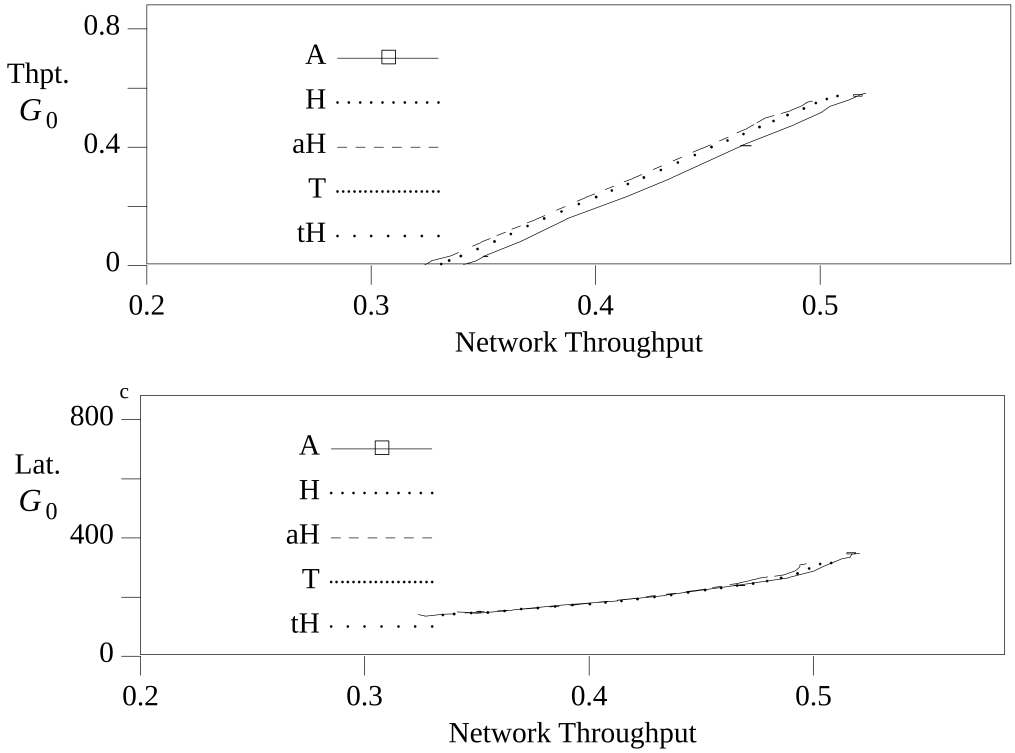


Figure 8.29: *Benchmark V: Two-Counter Flow Control.* Non-uniform traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). Shown is the G_0 throughput vs. system throughput and G_0 latency vs. system throughput for two-counter flow control with a variety of parameters.

effective than maximum usage. Fig. 8.29 shows the results for networks using two-counter flow control with a variety of parameters (see Tab. 8.2 for an explanation of parameter settings). G_0 performance for two-counter flow control is almost as good as that for maximum usage, but G_{NUT} throughput falls short of the throughputs achieved by maximum usage flow control (as can be seen by the reduction in the total network throughput).

As has been previously discussed, the concept behind hysteresis flow control is that one wants to change flow control states only when the network state (specifically, the point of congestion) has ‘‘really’’ changed, and thus avoiding both throttling flow unnecessarily and allowing flow when one should not. Since these are situations which only arise under dynamic congestion conditions, we expected this benchmark to be where hysteresis flow control demonstrated its worth. Our results indicate that either (a) ‘‘false’’ throttles and releases do not significantly effect performance, (b) adding hysteresis to the flow control does not prevent ‘‘false’’ throttles and releases or (c) one must carefully match the thresholds of the hysteresis mechanism to the dynamics of the traffic pattern, and we did find the match. If any of those hypotheses are true, it suggests that hysteresis is not a beneficial characteristic of a flow control mechanism for a general purpose communication network.

Figure 8.30 shows simulator results for the link-based flow control mechanisms (destination- and path-based flow control). The link-based flow control mechanisms achieve very high throughputs for G_0 traffic — destination-based flow control supports a G_0 throughput of 0.85. Further, at a network throughput of 0.50, the average latency of G_0 packets is ≈ 150 c, markedly lower than the average G_0 latency for maximum usage flow control operating at 0.50 (≈ 260 c).

The saturation throughput of G_0 under destination-based flow control in this benchmark is significantly higher than the saturation network throughput obtainable under uniform traffic conditions — how destination-based flow control is able to achieve such a high throughput for G_0 is evident from the fact that the maximum network throughput (which, under this benchmark, is the average of the

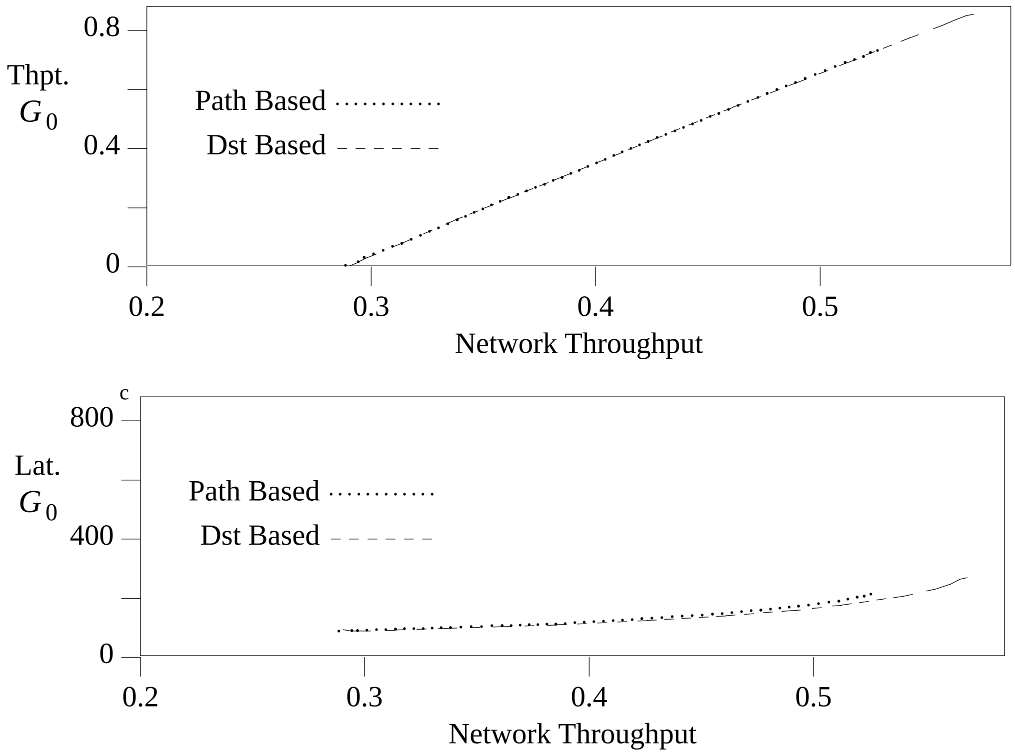


Figure 8.30: *Benchmark V: Link-Based Flow Control.* Non-uniform traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). Shown is the G_0 throughput vs. system throughput and G_0 latency vs. system throughput for destination- and path-based flow control.

G_{NUT} and G_0 maximum throughputs for this benchmark) is significantly lower than the maximum G_0 throughput. Destination-based flow control heavily penalizes the G_{NUT} traffic in favor of the G_0 traffic. G_{NUT} is restricted to 0.29 when G_0 is operating at saturation ($[0.85+0.29]/2=0.57$). Whether or not this behavior is desirable depends upon the applications being executed; while this flow control may seem to be ‘‘unfair’’ in the degree to which it is throttling G_{NUT} traffic in favor of G_0, the saturation throughput for the network as a whole is 0.57, which is slightly higher than the total network saturation throughput for maximum usage flow control with a ten-block threshold (0.55).

While path-based flow control throttles G_{NUT} traffic as much or more than does destination-based flow control, G_{NUT} achieves a higher saturation throughput under path-based flow control. This is due to the fact that path-based flow control throttles G_0 more heavily than does destination-based (throttling on the basis of the next two hops to be traversed as opposed to the destination address). Thus, G_{NUT} encounters less network contention from G_0 packets under path-based flow control and achieves a higher throughput. The net result is that path-based flow control has a lower total network throughput than either destination-based or maximum usage flow control.

8.5.9. Benchmark VI: Uniform Traffic

Our final benchmark is to apply uniformly distributed traffic to networks implemented with different flow control mechanisms. The purpose of this benchmark is to measure the degree to which latency is increased and saturation throughput is reduced by the flow control mechanisms when there are no NUT spots in the network. For this benchmark, all senders transmit thirty-two byte messages, and each message's destination address is randomly selected from a uniform distribution of all possible destinations. Since the senders are not divided into separate groups, the performance metrics are the maximum (total) network throughput and the average latency per throughput.

Figure 8.5, presented in Sec. 8.5.2, showed latency vs. throughput for networks of FIFO and DAMQ switches with blocking flow control. The network

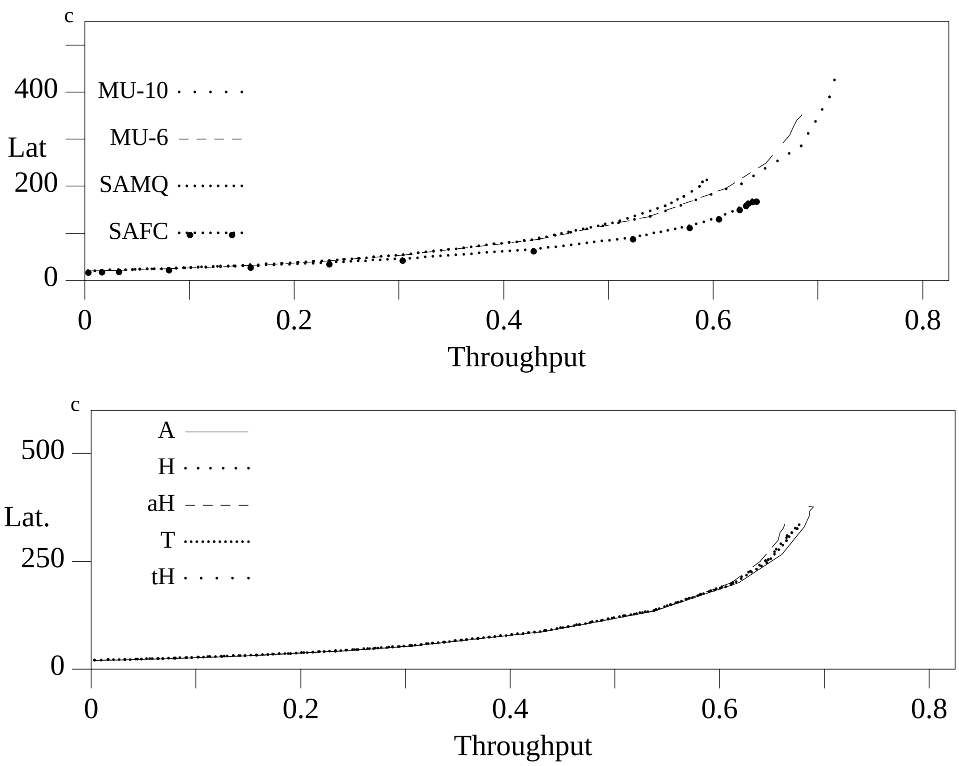


Figure 8.31: *Benchmark VI: Maximum Usage and Two-Counter Flow Control.* All senders transmitting 32-byte messages with uniformly distributed destinations. The average latency vs. throughput is presented; top graph shows static allocation and maximum-usage flow control (six and ten buffer maximums), bottom graph shows two-counter flow control with a variety of parameters (see Tab. 8.2 for explanation of parameters).

of FIFO switches saturates at a throughput of 0.49, while the network of DAMQ switches saturates at 0.71. At 0.20, both networks display an average latency of ≈ 35 c (the minimum latency through the four-stage omega network is 20 c). Fig. 8.31 presents two graphs. The top graph shows the average latency vs. throughput for networks with static buffer allocation (SAMQ and SAFC buffers) and maximum usage flow control (thresholds of six and ten buffer blocks). The bottom graph shows the performance of networks with two-counter flow control with a variety of parameter values (see Tab. 8.2 for explanation of parameter settings). For networks operating below saturation throughput, under uniform traffic conditions, these flow control mechanisms do not impact communication latency. The results indicate that the more restrictive flow control policies do reduce the saturation throughput. The flow control policy which significantly reduces the saturation throughput is static buffer allocation; SAMQ buffering drops the maximum throughput to 0.60, SAFC buffering drops it to 0.63.

Maximum usage flow control with a ten-block threshold actually improved the saturation throughput of uniformly distributed traffic. This mechanism achieved a network throughput of 0.72. An improvement of 0.01 is insignificant, and is within the error margin of the simulator (97.5% confidence intervals for simulation results of the two networks overlap), but one can understand why this mechanism would perform as well or better than blocking flow control under uniformly distributed traffic. The only time it throttles traffic is when a queue already has three packets in it — it does not cause output ports to be under-utilized, and it prevents a single queue from occupying the entire buffer, which eliminates the multi-queueing function of the buffer.

Two-counter flow control and maximum usage flow control with a threshold of six buffer blocks perform slightly worse than blocking flow control, saturating in the region of 0.68 (with some variance among the two-counter parameterizations). These mechanisms either throttle traffic more often then blocking flow control, or unblock blocked queues more slowly, and thus reduce the network throughput slightly.

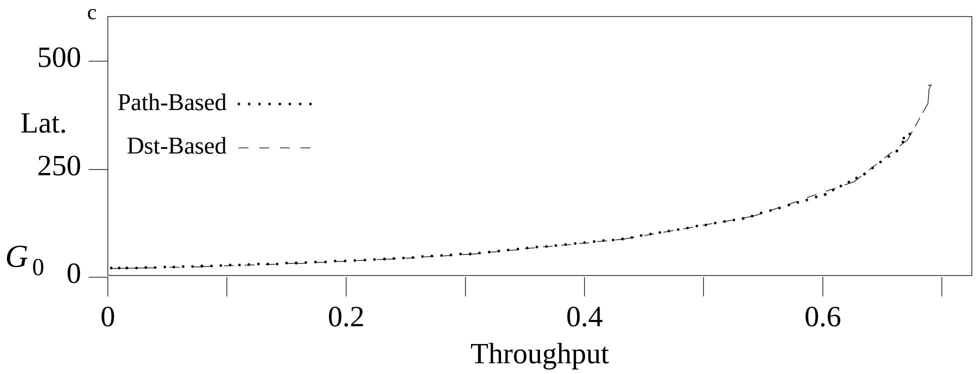


Figure 8.32: *Benchmark VI: Link-Based Flow Control.* All senders transmitting 32-byte messages with uniformly distributed destinations. Shown is the latency vs. throughput for Destination- and path-based flow control.

Figure 8.32 shows the performance of path-based and destination-based flow control. Destination-based flow control achieves a saturation throughput just slightly lower than that of blocking and maximum usage flow controls (0.69). Path-based flow control is slightly worse (0.67) — this mechanism is more likely to commit ‘‘false throttling’’ (throttling traffic when there is no congestion) than is destination-based flow control.

8.5.10. Combining Hop-Level Flow Control Mechanisms

In Secs. 8.5.4 through 8.5.9, a set of hop-level flow control mechanisms were evaluated. The results of this evaluation indicate that different flow control mechanisms protect the performance of G_{0-2} senders and the throughput of the network as a whole under different congestion patterns. For Benchmarks I and II (the hot spot benchmarks), destination-based flow control outperforms every other flow control mechanism examined, both in terms of total network throughput and in terms of protection provided for G_0. On Benchmarks IV and V, destination-based flow control provides a greater degree of protection for the G_0 senders than does maximum usage flow control, but destination-based flow control and maximum usage flow control with a ten-buffer-block threshold achieve approximately the same maximum network throughput.

In Benchmark III, a static non-uniform traffic spot not associated with a single destination was generated in the network. Since only sixteen of the 256 senders were impacted by this congestion, there was little difference between the maximum total network throughput achieved by the different flow control mechanisms. Under destination-based flow control, however, the G_0 traffic was severely impacted by this congestion. As was discussed in Sec. 8.5.6, destination-based flow control is ineffective against congestion which is not destination-oriented. The buffer-based flow control mechanisms (maximum usage, hysteresis and static buffer allocation) were all significantly more effective than destination-based flow control in this case.

Benchmark VI addressed the performance of the flow control mechanisms under uniform traffic conditions. Static buffer allocation was the only flow control mechanism which significantly impacted network throughput under uniform traffic conditions; the rest of the mechanisms we examined had little impact. SAMQ buffers reduced throughput and had no redeeming features; SAFC buffers had a lower maximum throughput than the other flow control mechanisms, but it also supported a lower average latency at high throughputs.

The flow control mechanisms which we have examined are not all mutually exclusive. That is to say, one can implement a flow control protocol which combines two or more of the mechanisms which we have discussed. In this section, we explore the results of combining destination-based flow control with maximum usage flow control and with SAFC buffering. Destination-based flow control is an obvious candidate for combining, as it provided the highest performance on most of the benchmarks. Maximum usage flow control also did well on many of the benchmarks, including Benchmark III — the benchmark that destination-based flow control did so poorly on. Although the SAFC buffer failed to achieve throughputs as high as the other flow control mechanisms, it was consistently able to maintain low average latencies.

Combining destination-based flow control and maximum usage flow control into a single flow control protocol is relatively straight forward. As with destination-based flow control, if a packet attempts to enter a buffer in which there is already a packet with the same destination, the incoming packet is rejected. Rejected packets are moved to the tail of the queue from which they were transmitted. Rejected packets are marked as BLOCKED, and packets which ‘‘cause’’ incoming packets to be rejected are marked as BLOCKER. When a BLOCKED packet moves to the front of a packet queue, it is moved to the tail without attempting to transmit it. When a BLOCKER packet is transmitted, this information is sent to the previous switch which clears the BLOCKED mark from all of the packets queued for the BLOCKER’s buffer. As with maximum usage flow control, any of the individual queues within a buffer may block reception. Full / not full signals for each of the queues within a buffer are fed back to the neighboring switch which transmits to that buffer (as opposed to a single full / not full signal for the entire buffer). Switches pre-route packets to determine which queue they will be appended to in the next switch, and compare this information with the per-queue flow control to determine whether or not packets can be transmitted. Under this flow control combination, if the packet at the front of a queue is destined for a full queue and so cannot be transmitted, the packet is moved to the tail of the queue. This mechanism is called *blocked-packet-requeueing*. We did not provide blocked-packet-requeueing for plain maximum usage flow control, as (a) it requires additional hardware and firmware to support moving packets from the head to the tail of their queues and (b) maximum usage flow control would no longer support a FIFO packet ordering between source / destination pairs. Since destination-based flow control requires the presence of this hardware / firmware and preserves the FIFO ordering in the presence of re-queueing, it only makes sense to evaluate the performance of this flow control protocol *with* this optimization.

The implementation of destination-based flow control with SAFC buffers is identical to the implementation with maximum usage flow control just described. Combining destination-based flow control and SAFC buffering only makes sense, however, if the buffers are large enough that more than one packet can be in a single queue. For the network simulations presented previously in this chapter, the switches each have 128 bytes of buffer memory at each input port. Statically allocating a buffer memory of this size gives each queue within the buffer thirty-two bytes — enough memory to store a single packet. Destination-based flow control is not compatible with a buffer architecture in which the maximum queue length is a single packet, so the results presented in this section are from network

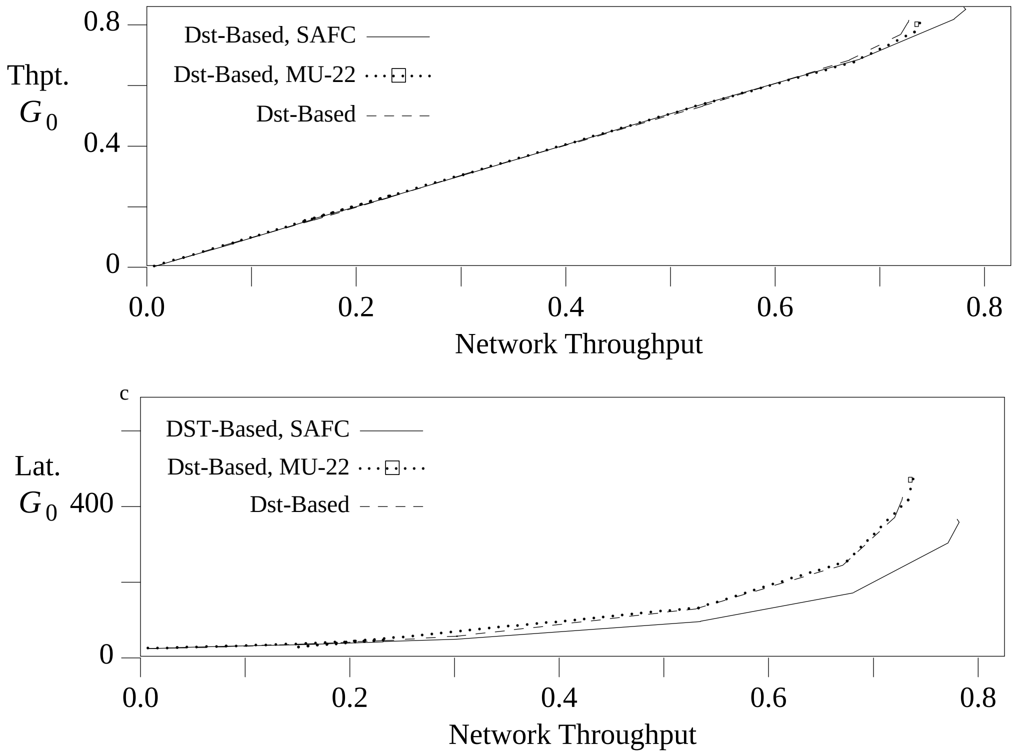
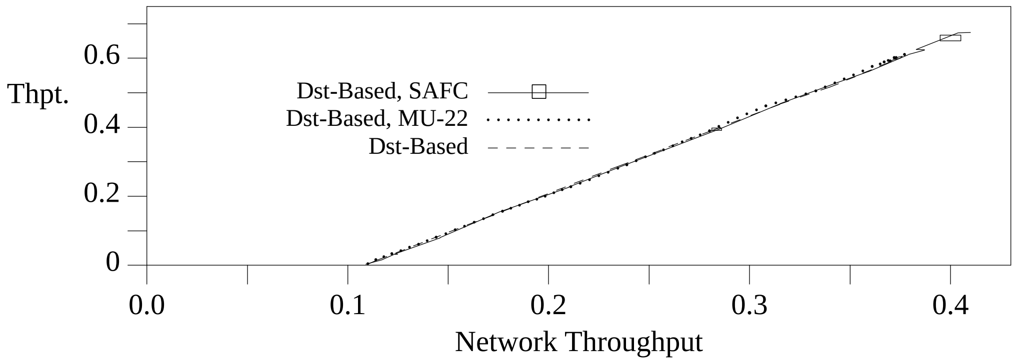


Figure 8.33: *Benchmark I: Group 0 Performance, Combined Flow Control.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_{0-2}). Switches with 256 bytes per buffer. Shown is the G_0 throughput vs. network throughput and G_0 latency vs. network throughput for destination-based flow control alone, combined with maximum-usage flow control (threshold of twenty-two buffer blocks), and implemented with SAFC switches.

simulations in which each buffer has 256 bytes of storage.

Figures 8.33-8.37 present the throughput and average latency of G_0 vs. the network throughput for destination-based flow control combined with maximum usage flow control with a throttling threshold of twenty-two buffer blocks and destination-based flow control implemented on SAFC buffered switches. In order to compare the performance of these combined flow control protocols, results for



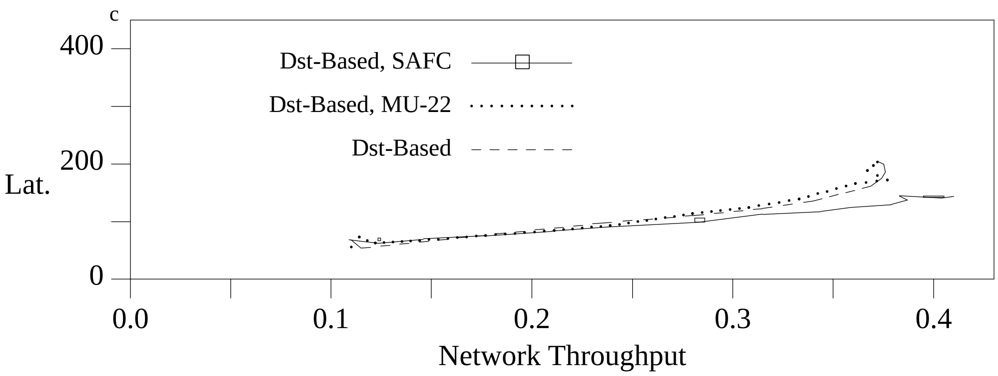


Figure 8.34: *Benchmark II: Combined Flow Control*, *Group 0.* Non-uniform traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). 256-byte buffers. Shown is the G_0 throughput vs. network throughput and G_0 latency vs. network throughput for destination-based flow control alone, combined with maximum-usage flow control (threshold of twenty-two blocks), and implemented with SAFC switches.

DAMQ switches with plain destination-based flow control are included in the figures, and Fig. 8.35 (results for Benchmark III) also shows the results for DAMQ switches with plain maximum usage flow control with a threshold of twenty-two blocks. Finally, Fig. 8.38 compares the performance of the combined flow control mechanisms to plain destination-based, maximum usage and blocking flow control under uniformly distributed traffic (Benchmark VI).

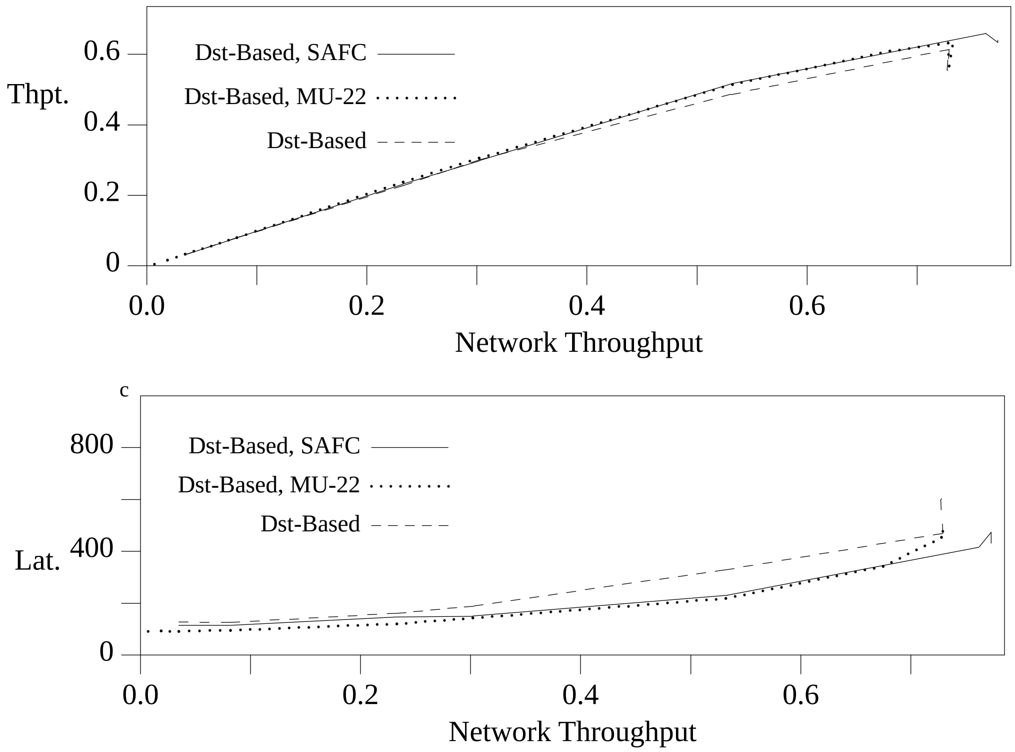


Figure 8.35: *Benchmark III: Combined Flow Control, Group 0.* Congestion traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic ($G_{0,1}$). 256-byte buffers. Destination-based flow control alone, combined with maximum-usage flow control (threshold of twenty-two blocks), and implemented with SAFC switches. Throughput and average latency vs. total network throughput, G_0.

The results of our simulations of combined flow control mechanisms can be succinctly summarized. Combining destination-based flow control with SAFC buffering provides superior network performance under uniform traffic conditions and in the presence of a wide variety of congestion patterns. The high performance of this combination has two sources. The first is the fact that it combines flow control mechanisms which make flow control decisions based upon different

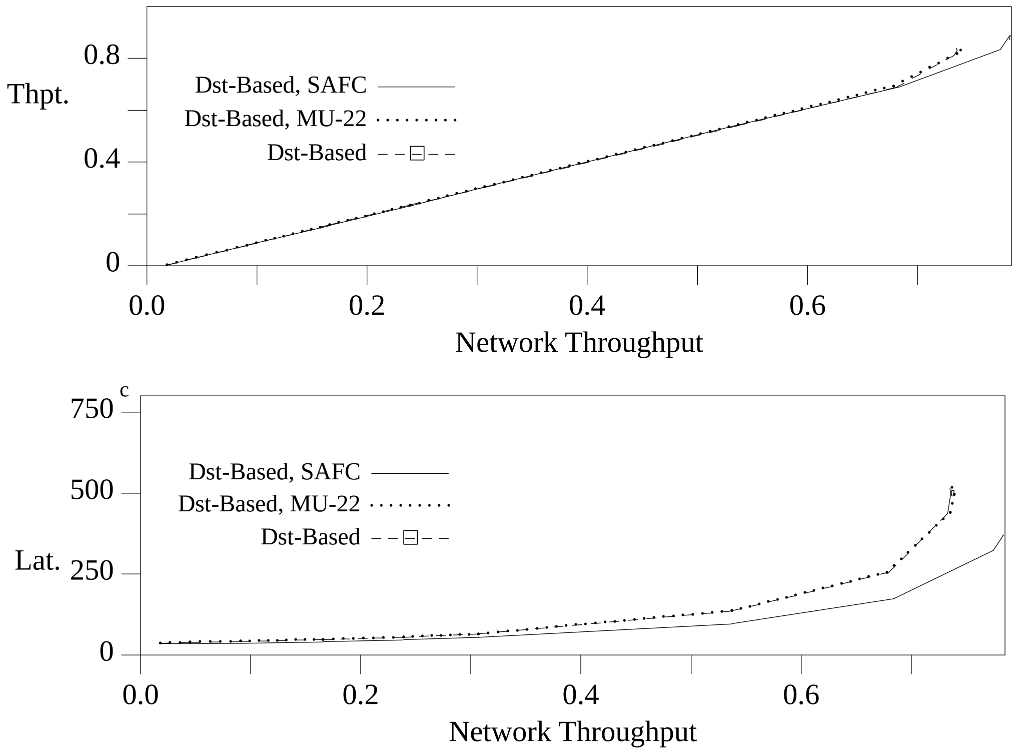


Figure 8.36: *Benchmark IV: Combined Flow Control, Group 0.* Congestion traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_{0-2}). 256-byte buffers. Destination-based flow control alone, with maximum-usage flow control (threshold of twenty-two buffer blocks), and with SAFC switches. Throughput and average latency vs. total network throughput, G_0.

aspects of communication traffic. Static buffer allocation provides feedback as to the current buffer utilization; by limiting the number of packets in any one queue, the number of queues per buffer that have packets in them is increased, which in turn increases the link utilization. Destination-based flow control provides a predictive mechanism which (a) detects potential congestion multiple hops before the point of congestion and (b) provides highly directed back-pressure. The

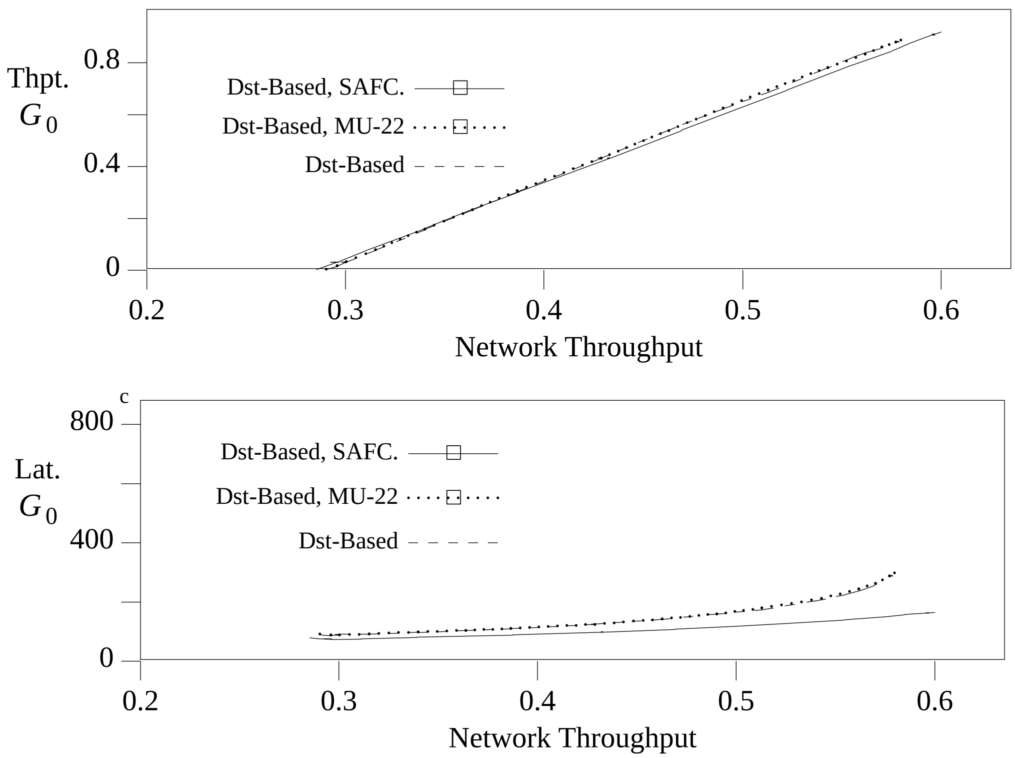


Figure 8.37: *Benchmark V: Combined Flow Control.* Non-uniform traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). 256-byte buffers. Shown is the G_0 throughput vs. system throughput and G_0 latency vs. system throughput for destination-based flow control alone, combined with maximum-usage flow control (threshold of twenty-two blocks), and implemented with SAFC switches.

combination of the two flow control mechanisms creates a congestion ‘‘firewall’’ which provides superior protection to the ‘‘well-behaved’’ senders in the system.

The second source of this combination’s high performance is shown in Fig. 8.38 (Benchmark VI, uniform traffic). Under uniform traffic conditions, destination-based flow control with SAFC buffers supports as high a saturation throughput as *any* flow control mechanism / buffer architecture we have examined,

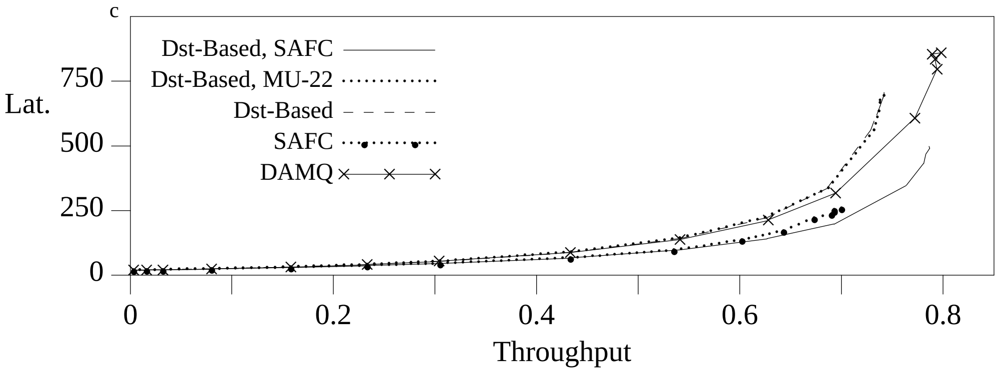


Figure 8.38: *Benchmark VI: Combined Flow Control.* All senders transmitting 32-byte messages with uniformly distributed destinations. 256-byte buffers. Shown is the latency vs. throughput for destination-based flow control alone, combined with maximum-usage flow control (blocking threshold of twenty-two blocks) and implemented with SAFC switches. Also shown are DAMQ and SAFC switches with blocking flow control.

and does so with a significantly lower average packet latency. Thus, even if it is only equally effective as the other flow control mechanisms which we have examined for preserving network performance in the presence of congestion, its superior performance under uniform traffic conditions would result in superior performance under congestion conditions.

Figure 8.38 suggests that destination-based flow control and SAFC buffers have a synergistic relationship — the combination results in a network whose performance is superior to that of either SAFC-buffered networks with blocking flow control or destination-based flow control with DAMQ buffers. In Chs. 4 and 6, we presented simulation results which indicate that DAMQ buffers provide performance superior to that of SAFC buffers under uniform traffic conditions. It is clear that SAFC buffers provide a greater degree of connectivity between buffers and output ports, which gives SAFC buffers the potential to achieve a higher throughput than DAMQ buffers. However, they utilize their buffer memory less efficiently than do DAMQ buffers, which results in SAFC-buffered networks having a lower throughput under uniform traffic conditions than do networks with DAMQ buffers, for buffers up to 1024 bytes in size (Fig. 6.4).

The inefficient buffer utilization of SAFC buffers (and other statically allocated buffers) stems from the fact that queues can become full when there is still free memory in the buffer. If the packet at the head of a queue in an SAFC buffer (Q^0) is destined for a full queue in the next switch (Q^1), under blocking flow control Q^0 cannot transmit until Q^1 transmits (frees some of its buffer memory). With blocked-packet-requeueing, the packet at the head of Q^0 can be moved to the tail of Q^0; while the next packet in the queue is destined for the same buffer that the requeued packet was, it is not necessarily destined for Q^1, and thus may be transmittable.

The resulting increase in the utilization of SAFC buffers is shown by the fact that both the saturation throughput *and* the average latency at the saturation throughput are higher for SAFC buffers with destination-based flow control than for SAFC buffers with blocking flow control. Comparing the points of saturation throughput for SAFC-buffered networks with blocking and with destination-based flow control, we see that the average number of packets existing within the network† increases from 1445 to 3154.

---

† The average number of packets stored in the network equals the throughput per link times the number of links connected to destinations times the average latency divided by the number of bytes per packet.

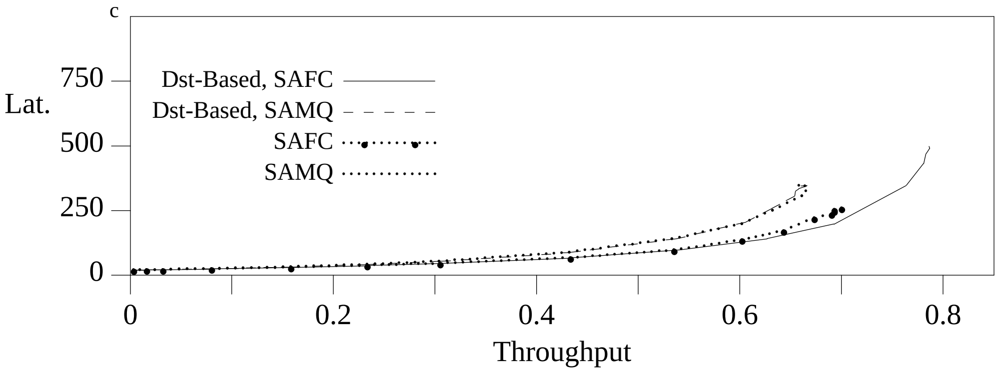


Figure 8.39: *Benchmark VI: Destination-Based Flow Control with SAMQ Buffers.* All senders transmitting 32-byte messages with uniformly distributed destinations. 256-byte buffers. Shown is the latency vs. throughput for SAFC- and SAMQ-buffered networks with blocking and destination-based flow control.

The phenomena of blocked-packet-requeueing does not totally explain the superior performance of destination-based flow control with SAFC buffers, however. Fig. 8.39 shows the average latency vs. throughput under uniform traffic conditions for 256-byte SAMQ- and SAFC-buffered networks, with both blocking and destination-based flow control. If the change in performance of the SAFC buffers was solely driven by the ability to move blocked packets to the rear of queues, then the network of switches with SAMQ buffers would also experience higher buffer utilization with destination-based flow control. The figures for average latency and saturation throughput suggest that SAMQ buffers received little or no increase in buffer utilization or network throughput under destination-based flow control. Thus, it requires more than blocked-packet-requeueing to improve network throughput under static buffer allocation.

The difference between the impact of destination-based flow control on

SAMQ and SAFC buffers stems from the number of packets which are available for transmission at any point in time. With SAMQ buffers, only the packets in buffers which are currently idle are considered. With SAFC buffers, on the other hand, every queue associated with a particular output port can be considered for every crossbar arbitration that the output port is available, since multiple queues within an SAFC buffer can transmit simultaneously. Since destination-based flow control moves blocked packets to the tail of queues, under SAFC buffering, when an output port becomes available, *every* packet currently on the chip destined for that output port is a candidate for transmission. This both increases link utilization relative to SAMQ buffers *and* reduces the average transmission latency.

In examining the results presented in Figs. 8.38 and 8.39, it is seen that, under uniform traffic conditions, destination-based flow control (a) increases the saturation throughput of SAFC buffers, (b) has little effect on SAMQ buffers and (c) reduces the saturation throughput of DAMQ buffers. We have explained the increased performance of the SAFC-buffered network, and the negligible impact destination-based flow control has on SAMQ-buffered networks. What remains to be explored is the negative impact destination-based flow control has on networks of switches implemented with DAMQ buffers.

Destination-based flow control *significantly* reduces the throughput of the DAMQ-buffer network transmitting uniformly-distributed traffic. While it impacts performance throughout the network, the most significant effect of destination-based flow control occurs in the final switching stage of the network. There, the output ports of the switches are connected directly to the destination nodes. Since destination-based flow control allows at most one packet per destination per buffer, this means that there can be at most one packet per queue in the buffers of the last switching stage. This causes the 256-byte DAMQ buffers in the last switching stage of a network using destination-based flow control to operate as though they were 128-byte SAMQ buffers. This phenomena occurs in SAMQ and SAFC buffers, as well. It does not impact the SAMQ-buffered network, however, as the applied load on the last switching stage is low enough that the reduction from 256-byte SAMQ buffers to 128-byte SAMQ buffers does not reduce the throughput. It *does* impact SAFC buffers, but to a lesser degree than DAMQ buffers. Since SAFC buffers allow multiple packets to be transmitted from a single buffer, 128-byte SAFC buffers support a higher packet throughput than do 128-byte SAMQ buffers.

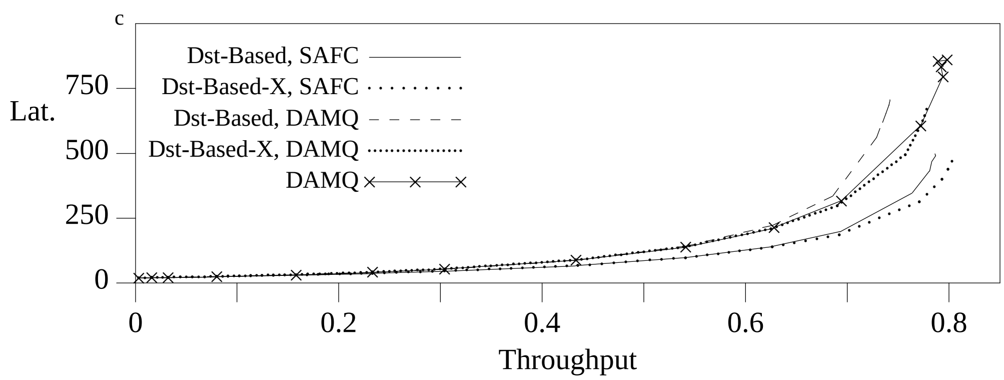


Figure 8.40: *Benchmark VI: Destination-Based Flow Control, Suppressed at Destination.* All senders transmitting 32-byte messages with uniformly distributed destinations. 256-byte buffers. Average latency vs. throughput for blocking flow control with DAMQ buffers, destination-based flow control, and destination-based flow control with discarding suppressed at the destination switches (‘‘Dst-Based-X’’ in the graph legend).

This hypothesis can be verified by simulating a multistage interconnection network in which every switching stage except the final one implements destination-based flow control. Fig. 8.40 shows the average packet latency vs. network throughput for uniformly-distributed traffic in DAMQ- and SAFC-buffered networks (a) with destination-based flow control throughout the network and (b) with destination-based flow control operating in the first three switching stages, and blocking flow control in the final switching stage (labeled *Dst-Based-X* in Fig. 8.40).

As can be seen from Fig. 8.40, suppressing destination-based flow control in the last switching stage improves the SAFC-buffered network only slightly but has a significantly positive effect on the DAMQ-buffered network. Still, SAFC buffers with destination-based flow control achieve a higher throughput than do DAMQ buffers with destination-based flow control, even with discarding disabled for the final switching stage. This is simply a matter of there being more bandwidth spent transmitting packets which are subsequently discarded in the DAMQ-buffered network than in the SAFC-buffered network.

The reason that the DAMQ-buffered network discards more packets is that dynamic buffer allocation more efficiently utilizes the available buffer space than does static buffer allocation. This is true even with blocked-packet-requeueing. This can be seen from the fact that the product of the average latency at the maximum network throughput and the maximum network throughput is higher for the DAMQ-buffered network than the SAFC buffered network; there are more packets in a DAMQ buffered network than there are in an SAFC buffered network at a given throughput. Thus, in a DAMQ buffered network, when a packet arrives at a buffer, there are (on average) more packets in the buffer and a higher chance that the incoming packet will encounter a packet with the same destination and be discarded.

8.6. Flow Control Mechanism Implementation

In the previous sections of this chapter, the performance of a number of hop-level flow control mechanisms have been evaluated. This section describes their implementation costs and discusses the implications this cost has on their feasibility and effectiveness.

8.6.1. Blocking Flow Control

Blocking flow control is the simplest of the flow control mechanisms examined in this dissertation. The function of blocking flow control is to halt transmissions to a buffer when it cannot guarantee that there is enough free memory to store the packet. The hardware to implement blocking flow control consists of subunits to perform three separate functions: (1) track the amount of free space in the buffer, (2) transmit flow control information to the neighboring switch, and (3) use the flow control information received from a neighbor to allow / prevent transmissions to that neighbor.

Under blocking flow control, the buffer cannot accept a new transmission if there is not enough free memory to store a maximum-size packet (thirty-two bytes, in our implementation of the DAMQ buffer — Ch. 7 and[Fraz89]). A counter is used to track the amount of free space in the buffer memory (the DAMQ buffer tracks buffer blocks, FIFO, SAMQ and SAFC buffers track bytes). When the value in this counter indicates that there are fewer than thirty-two bytes available in the buffer, the flow control mechanism signals that the buffer is full.

Our simulations assume that there is one wire for flow control which accompanies each communication link. This wire transmits the flow control state of the destination buffer to the neighboring switch. With unidirectional links such as those of the multistage interconnection networks simulated for this dissertation, this wire is necessary. In the general case, however, wires dedicated to flow control represent bandwidth which is not being used to transmit data. It is possible to avoid dedicating bandwidth to flow control in networks whose connections are bidirectional (i.e. neighboring switches each having a link transmitting to the other). In these networks, one can ‘‘piggyback’’ flow control information on packets traversing the link. Hence, the flow control information competes for network bandwidth with normal traffic. In this case, minimizing the bandwidth required for flow control is desirable. As discussed earlier (Sec. 8.2), schemes with hysteresis reduce the required bandwidth for flow control. Such schemes may thus be desirable if there are no dedicated lines for flow control. Another consideration when evaluating piggybacked flow control is the latency that this adds to the flow control feedback; if the latency is too high, the link idle time between packets may have to be increased in order to prevent buffer overflow.

Having received the flow control feedback from their neighbors, the switches’ crossbars must have a mechanism to prevent connecting buffers to output ports whose destination buffer is full. How this is done depends directly upon the implementation of the crossbar arbiter and the buffer / crossbar interface. An example is shown in Fig. 8.41. For a 4×4 switch, each buffer has four request lines to the crossbar indicating to which output ports the buffer has packets destined.

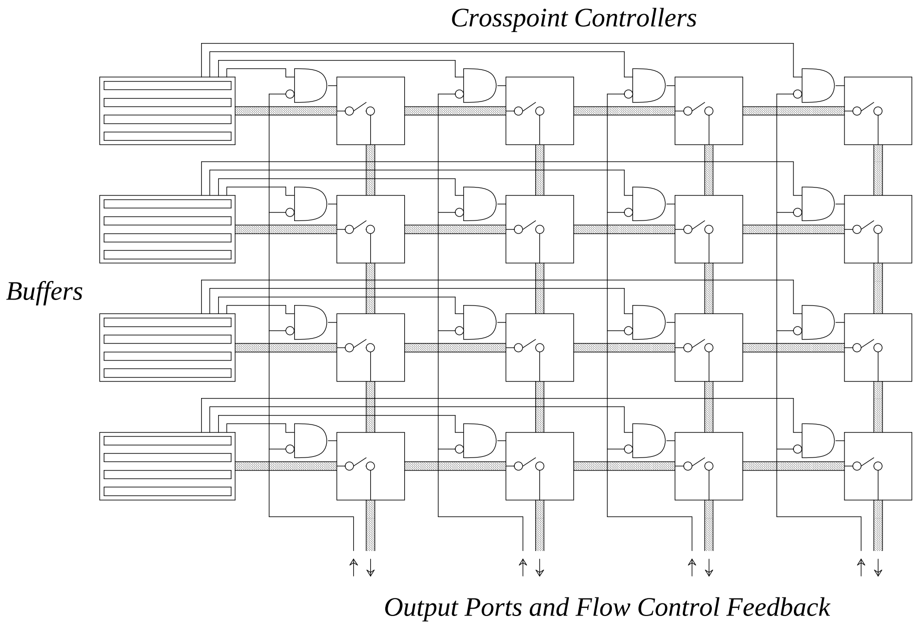


Figure 8.41: *Crossbar Hardware for Blocking Flow Control.* AND-gates are used to suppress connection requests from buffers to output ports whose destination input ports have blocked transmissions.

For FIFO buffers, at most one of the four lines can be asserted at any given point in time (assuming static routing). For non-FIFO buffers, the request lines correspond to non-empty packet queues within the buffer. In either case, the flow control signal associated with an output port can be used to mask requests for that output port (Fig. 8.41), preventing packets from being transmitted to that output port.

The total hardware overhead for blocking flow control is minimal. The most significant hardware addition is the wire dedicated to transmitting flow control information. Since one is only interested in changes in the flow control state, and since these changes do not happen on the majority of the clock cycles, dedicating the wire to flow control wastes some amount of bandwidth. The alternative (piggy-backing flow control information) (a) is topology dependent (b) is considerably more complex and (c) may lose bandwidth due to the added latency of the feedback.

8.6.2. Queue-Based Flow Control

Our evaluation of hop-level flow control mechanisms included maximum usage flow control, statically allocated buffers, and two-counter flow control. These flow control mechanisms are similar to one another, in that they all block packets on a *per-queue* basis. Thus, the functional units which implement these mechanisms are very similar to one another. Each queue has its own counter to track the amount of data in the queue. Maximum usage and two-counter flow control also need a central counter to track free space in the DAMQ buffer. In the simulations presented in this chapter, it was assumed that each communication link was accompanied by an n-bit flow control channel, where n is the number of packet queues in the buffer. If $n = 4$ and the links are eight bits wide, twelve wires connect neighboring switches and flow control occupies 33% of the communication bandwidth. In other words, going from blocking flow control to queue-based flow control requires an increase in the link bandwidth of 33% (nine wires to twelve wires). This is an excessive amount of overhead; none of the queue-based flow control mechanisms increased the maximum network throughput by 33% in *any* of the benchmarks presented in this chapter.

An alternative to dedicating a separate wire to each queue in the buffer is to maintain a single wire for flow control feedback, and to sequentially transmit separate bits of flow control information over this wire. While this does increase the latency of the flow control feedback, it is an incremental increase, and would have negligible impact on network performance. It does increase the complexity of the flow control logic, in that there must be logic on the transmitting end of the communication link that can parse incoming flow control information and decode it for the rest of the switch, but this is simple logic.

The queue-based flow control mechanisms all require that packets be *pre-routed,* and that the crossbar arbiter not connect buffers to output ports if the packet at the head of the queue associated with that output port is destined to a blocked queue. The complexity of pre-routing packets is dependent upon whether the results of pre-routing are transmitted with a packet. If the pre-routing result is not transmitted with the packet, two routing operations must be performed at each switch. The tradeoff between these two options is the relative overhead of dedicating $\log_2(n)$ bits of bandwidth per packet to transmit port information vs. an increase in the amount and complexity of the packet reception and routing hardware.

The complexity of the pre-routing mechanism is also dependent upon the routing protocol used by the network. It is at its worst for table-lookup routing. For it, each switch must maintain routing tables for each of its neighbors; after the output port and new header are determined for a packet, then the new header would be used to access the table associated with the neighbor connected to the destined output port to discover the destination queue. Even if the results of pre-routing are attached to packets, a new packet header must be determined before the pre-routing can take place. For topology-based routing algorithms, the hardware which determines the output port of a packet can be enhanced to return the destined output ports of both the current and the next switch without a major increase in complexity. Pre-routing becomes a particularly complex operation when used in conjunction with dynamic routing. In this case, a switch must make intelligent routing decisions (or, at least, avoid *bad* routing decisions) one hop away from where the switching will actually take place.

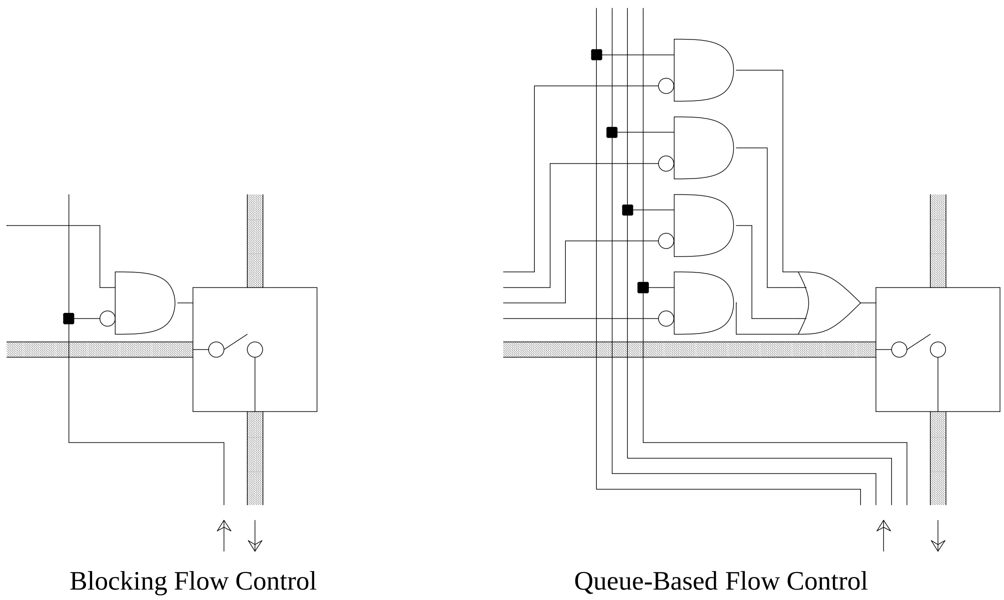


Figure 8.42: *Crosspoint Controller Hardware for Queue-Based Flow Control.* A single crosspoint controller from the crossbar of a switch using blocking flow control (Fig. 8.41) and queue-based flow control.

Figure 8.42 compares the complexity of the crosspoint controllers of the crossbar of a switch using queue-based flow control to that of a switch using blocking flow control. Pictured are single crosspoint controllers of a 4×4 switch for both queue-based and blocking flow control systems. When requesting an output port connection in a queue-based system, a buffer must choose one of four wires to assert — the asserted wire indicates which queue in the *next* buffer the packet is destined for. Combinatorial logic compares the request signal to the four flow control lines associated with the output port. If the flow control signal does not mask the request, then the arbiter may make the connection, whereupon the buffer transmits the packet.

A difference between maximum usage and statically allocated buffer flow control is that, with maximum usage flow control, an incoming packet can be stored to *any* available buffer memory. This allows the buffer to determine where a packet will be stored before it actually arrives. With a statically allocated buffers, on the other hand, packets must arrive at an input port with the identifier of the queue to which they are to be appended in their header, or there must be a *staging buffer* at the input port to store the packet while it is being routed. The use of a staging buffer may add to the minimum cut through latency of the switch (the speed with which a packet can move through a switch). On the other hand, including the queue identifier in the header increases bandwidth overhead per packet.

8.6.3. Destination- and Path-Based Flow Control

Since the only difference between destination-based flow control and path-based flow control is the hardware to determine whether or not to accept a packet, this subsection focuses on destination-based flow control. Destination-based flow control requires considerably more enhancements to the basic switching hardware than do the other flow control mechanisms discussed in this chapter; their use alters the fundamental operation of a switch. When a packet is transmitted, the buffer space it occupies is not immediately deallocated under destination-based flow control. Rather, the buffer waits for an **ACK** signal from the next buffer, indicating that the packet is being accepted, before it deallocates its buffer space.

If it instead receives a **NACK**, the buffer must relink the packet at the tail of the queue and not deallocate the buffer space. We will first examine the the hardware required to determine whether a packet is to be ACK-ed or NACK-ed, then the communication overhead of sending ACK / NACK signals to neighboring switches, followed by the hardware required to handle requeueing NACK-ed packets, and finally we will examine the hardware implications of our enhancements to destination-based flow control (*abortive transmissions* and *restricted retry*).

Determining whether a packet is to be ACK-ed or NACK-ed requires comparing the *new* header of an arriving packet (i.e. the header returned by the router) to the headers of the packets currently stored in the buffer. This requires storing packet headers in a content-addressable memory (CAM). Every time a new header is read from the router interface, it is driven over the CAM — if a match is made, the reception is aborted and a NACK is generated. Altering the header register array to be content-addressable significantly increases the size of the array, but since it is a relatively small functional unit to begin with (in our example DAMQ buffer implementation — Ch. 7[Fraz89], — there are twelve header registers for a ninety-size-byte buffer), then this is not a major issue.

With destination-based flow control, the entire address portion of the packet headers is compared. For path-based flow control, however, only that part of the header which relates to the next n hops is compared. This is the only difference between the two, and for multistage interconnection networks, this simply means that a smaller CAM is needed for path-based flow control.

Since destination-based flow control must be accompanied by some form of blocking or queue-based flow control which will prevent buffer overflow, the number of wires required to indicate ACK / NACK of a transmission is an open question. If a queue-based flow control mechanism is using a single wire, transmitting flow control information serially, then one option is to increase the number of codes to include ACK and NACK signals. Another option is to modify the accompanying flow control mechanism such that no pre-routing or blocking is performed; the ACK / NACK signal would indicate not only whether the packet collided with another destined to the same address, but also whether an attempt was made to append the packet to a full queue. This allows a single wire to easily transmit full / not-full information, as well as conveying the ACK (since the target application of this flow control mechanism is a tightly-coupled multicomputer, the NACK can be implemented as a fast timeout). This represents a different flow control mechanism than was described in Sec. 8.5.10, where the combined flow control mechanisms were evaluated. This would drop performance somewhat relative to the simulation results presented in that section, since the crossbar controller of a switch would not be able to distinguish between packets destined for full vs. not-full queues, but it would substantially reduce the amount of hardware required to implement the flow control, as well as eliminate the pre-routing (which, as was previously mentioned, is problematic under certain circumstances).

For the DAMQ buffer to have the option of either re-linking a packet to the tail of its queue or deallocating the memory blocks to the linked list, one would add an additional head and tail register to the buffer. These registers define an additional linked list of buffer blocks for the buffer — the *currently-transmitting list*. When a packet is transmitted, instead of moving its buffer blocks directly from the head of its queue to the tail of the free list, they are appended to the currently-transmitting list. If an ACK is received, the block at the end of the currently-transmitting list is appended to the free list, the value in the tail register of the currently-transmitting list is copied to the tail register of the free list, the null register for the currently-transmitting list is set (this empties the list), and the memory is deallocated. If a NACK is received, the currently-transmitting list is appended to the queue's linked list by the same process. While this will add considerable complexity to the firmware which controls the buffer, the addition of a head / tail register pair is an insignificant increase in hardware.

We previously described two enhancements to the path-based flow control schemes: *abortive transmissions* and *restricted retry*. Since these enhancements measurably improve the performance of both destination-based and path-based flow control, the link-based flow control simulation results presented in this chapter were with both enhancements. As can be seen in Fig. 8.43, both *abortive transmissions* and *restricted retry* contribute to the performance of DAMQ buffers with destination-based flow control under uniformly distributed traffic conditions. Further, when evaluating SAFC buffers with destination-based flow control, the blocked-packet-requeueing mechanism was critical to the performance of the combination (blocked-packet-requeueing and abortive transmissions use the same hardware mechanisms).

The abortive transmissions enhances network performance to a significantly greater extent than does restricted retry under uniform traffic conditions. This is because restrictive retry only improves performance when a packet would, without restrictive retry, make multiple attempts to traverse a link and be repeatedly rejected due to collisions with the same packet. This is unlikely to occur in a

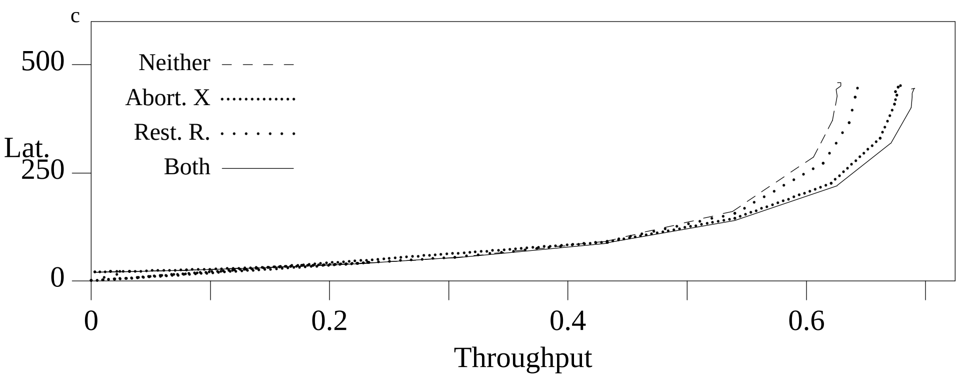
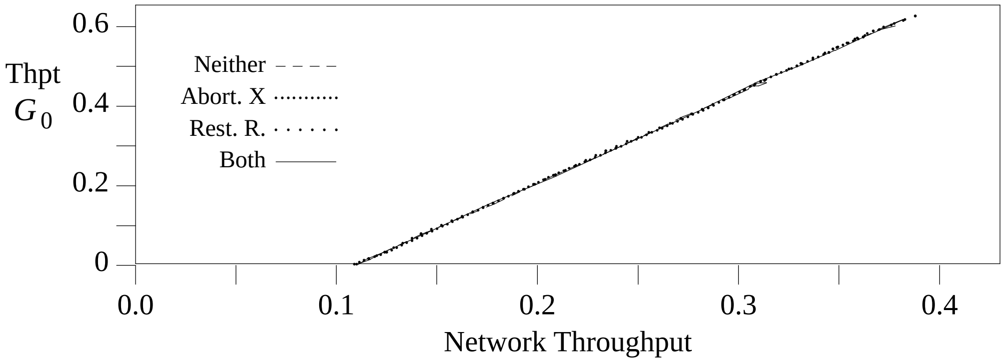


Figure 8.43: *Evaluating Link-Based Flow Control Enhancements: Benchmark VI.* All senders transmitting 32-byte messages with uniformly distributed destinations. Shown is the latency vs. throughput for Destination-based flow control with no enhancements, with abortive transmissions, with restricted retry, and with both enhancements.

uniformly distributed traffic pattern.

Figure 8.44 shows the performance of abortive transmissions and restricted retry for destination-based flow control under Benchmark II (a static hot spot created by half of the senders transmitting at saturation throughput, 3% of the packets to a hot spot, 97% of the packets uniformly distributed). Under this congestion pattern there is a point of congestion (the hot spot) which exists for an extended period of time. These are the circumstances in which individual packets may attempt to traverse a single link multiple times before being accepted by the next buffer. Restricted retry alone performs slightly better than does abortive retry alone, and the performance of the two enhancements together is no better.

The functionality of abortive transmissions significantly overlaps that of restricted retry. The sole purpose of restricted retry is to avoid ‘‘spending’’ link



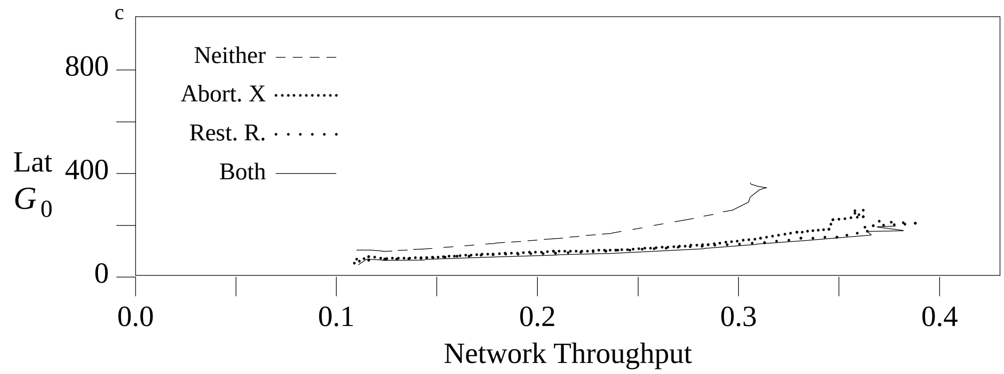


Figure 8.44: *Evaluating Link-Based Flow Control Enhancements: Benchmark II.* Hot spot traffic (G_{NUT}) operating at saturation, graduating applied load of uniform traffic (G_0). Shown is the G_0 throughput vs. network throughput and G_0 latency vs. network throughput for destination-based flow control with neither enhancement, abortive transmissions, restricted retry, and with both enhancements.

bandwidth on packets for which it is known that the packet will be rejected by the next buffer. Since abortive transmissions significantly reduce the amount of bandwidth spent on *all* rejected packets, restricted retry can at best provide an incremental improvement in performance.

Of the two enhancements we have proposed for link-based flow control, abortive transmissions requires the least amount of additional hardware. Abortive transmissions involves simply ceasing to transmit a packet once a NACK has been received (the buffer must continue to requeue the packet — the simulations take this into account). The amount of hardware required to implement this is dependent upon the ‘‘strictness’’ desired of the mechanism — if one allows the transmission to continue for some cycles after the NACK’s reception, the hardware required to implement abortive transmissions is negligible. Given the significant improvement resulting from its implementation, it is a clear positive tradeoff to implement link-based flow control *with* abortive transmissions / blocked-packet-requeueing.

Restricted retry associates two bits of status with each packet queued in a buffer. One bit indicates whether or not a packet is a BLOCKER. BLOCKER-s are packets which have caused incoming packets to be discarded, i.e. they *blocked* a packet. The other bit indicates whether a packet is a BLOCKEE. BLOCKEE-s are packets which have attempted a transmission and been rejected. These bits should be both physically and logically located near the header registers, since their access pattern is similar to that of the header registers. When a packet is successfully received into a buffer, both its BLOCKER and BLOCKEE bits are cleared. When a CAM lookup matches one of the headers in the header register array, the BLOCKER bit of the matched packet is set. When a packet is rejected and requeued, the buffer firmware must explicitly set the BLOCKEE bit associated with the packet. This can be done when the head register of the currently-transmitting list is read to requeue the packet. When a packet whose BLOCKEE bit is set moves to the front of a queue, this must be detected and the packet moved to the rear of the queue without attempting to transmit it. Finally, when transmitting a packet whose BLOCKER bit is set, a flow control signal must be sent to the neighboring switch to clear the BLOCKEE bits of all packets destined for this buffer.

While no single aspect of restricted retry is overly complex, taken as a whole this enhancement adds considerable hardware and complexity to the switch. One of its most difficult aspects is the amount of concurrency required. A large number of separate and sophisticated actions occur in parallel with the possibility of it being interrupted in the middle of one activity in order to perform another. Much of this complexity is eliminated if the queues are physically independent (i.e. SAMQ and SAFC buffers). In this case, each queue has its own transmission control logic and arbiter interface, which greatly simplifies the queue manipulations. However, since abortive transmissions provides the majority of the ‘‘enhancement’’ of enhanced path-based flow control, one would be disinclined to implement restricted retry.

8.7. Flow Control: Summary and Conclusions

This chapter presented an analysis of flow control mechanisms intended for use in scalable multicomputers. *Flow control* regulates the flow of data through the network by managing the allocation of resources (e.g., wires and buffers) to information units (e.g., packets) [Dall90a]. In many systems, the flow control mechanism exists solely to prevent the loss of data due to overflowing packet buffers. In this chapter, we explored the ability of flow control to optimize the performance of communication networks under uniform and non-uniform traffic patterns.

A network link is a *point of congestion* when the applied load of traffic attempting to traverse the link exceeds the bandwidth of the link. This usually occurs due to non-uniform traffic patterns, where there is a group of senders that are all transmitting packets that need to traverse the congested link. If packets are forced to wait to use the critical resource (the congested link) for an extended period of time, they will cause other packets to queue up waiting to use the network resource that they are occupying (the packet buffer). It is in this way that congestion spreads through a network. We termed this *congestion propagation.*

The bandwidth of the points of congestion directly limits the throughput of the non-uniform traffic group (G_{NUT} — the senders causing the point(s) of congestion). A flow control mechanism cannot, on its own, improve this. However, if congestion is allowed to propagate, other senders in the system which are not members of G_{NUT} may have their communication performance reduced by the congestion caused by G_{NUT}. A flow control mechanism can improve network performance by preventing congestion propagation, protecting well-behaved senders from those that cause congestion.

To support our analysis, a taxonomy for flow control mechanisms was developed. It uses three characteristics to classify flow control mechanisms. The first is the action taken to control traffic (blocking packets vs. discarding them). Second, where in the network the flow control decisions are made (global vs. end-to-end vs. hop-level flow control) was considered. The final characteristic used is the trigger which activates the flow control mechanism (predictive flow control, which is triggered by particular transmission patterns, vs. reactive flow control, which responds to congestion). In most cases, tightly-coupled interconnection networks for scalable multicomputers utilize flow control mechanisms which are blocking, hop-level, and reactive.

In our evaluations of hop-level flow control mechanisms, we found the queue-based flow control mechanisms (maximum usage, static buffer allocation and hysteresis flow control) to be effective when the point of congestion is close to the senders. These flow control mechanisms are less effective when the point of congestion is more then two hops away from the senders because they do not provide a means for directing back-pressure, and thus do not differentiate between senders in G_{NUT} and others.

Queue-based flow control is also effective when the congestion is highly dynamic (i.e. bursty traffic, where the bursts are short-lived). Under these circumstances, the congestion has little time to propagate; if the packets which comprise the burst can be stored in the buffers of one or a few switches, the lack of directed back-pressure is not an issue. Since a queue-based flow control mechanism prevents the congestion-causing packets from entirely filling the buffers, they support a higher throughput (under these conditions) than does straight blocking flow control.

We found destination-based flow control [Dias89] to be highly effective against congestion which is associated with a particular destination. Even when the packets collide a large number of hops away from their destination, if they have the same destination, this flow control mechanism identifies the non-uniform traffic and prevents congestion propagation. This includes non-uniformities caused by long messages or any burst of traffic to a single destination (a situation which may be common in a scalable multicomputer with distributed secondary memory).

The situation in which destination-based flow control is ineffective is when the traffic from multiple nodes collides at some point internal to the network, where the packets do not share the same destination (an internal non-uniform traffic spot). Under this circumstance, destination-based flow control is ‘‘blind’’ to the congestion, and performs as blocking flow control would.

Perhaps the most interesting result was the effectiveness of SAFC buffers combined with destination-based flow control. In previous chapters, we showed how static buffer allocation reduces the degree to which buffers are utilized, limiting the maximum throughput of a network. Sec. 8.5.10 showed that, by implementing destination-based flow control with SAFC buffers, buffer utilization is dramatically improved. The result is a buffer architecture / flow control mechanism combination whose performance in every benchmark is superior to that of any other examined in this dissertation. It even outperforms DAMQ buffers with blocking flow control under uniform traffic conditions (Fig. 8.38).

This chapter makes a number of contributions to the field of communication network architectures for scalable multicomputers. Among them is the means by which we evaluated the flow control mechanisms. Segregating the senders into groups, having one group to cause congestion within the network, and then measuring the performance of the individual groups and the network as a whole reveals a wealth of information as to the performance characteristics of the flow control mechanism. When combined with our congestion benchmark suite, this methodology provides a thorough evaluation of a flow control mechanism’s ability to support high performance communication.