Performance of Circuit Switched LANs under Different Traffic Conditions

Qingming Ma

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Peter Steenkiste

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Switched LANs are become more widely used because they can provide a higher bandwidth than LANs based on shared media. Examples of packet switched LANs include HIPPI, and switched FDDI and Ethernet. A number of studies have evaluated the performance of HIPPI networks, making simplifying assumptions about both the network and the traffic load. In this paper we present the results of a simulation study of circuit-switched LANs such as HIPPI using more realistic models for the system and the traffic. We observe changes in throughput as high as a factor of ten when we change the system and traffic parameters. We also show how packet scheduling can be used to improve performance in some cases.

1 Introduction

Switched LANs are become more widely used because they potentially provide a higher bandwidth than traditional LANs based on shared media. Examples of packet switched LANs include HIPPI, ATM and switched FDDI and Ethernet. In this paper we present the results of a simulation study of packet-switched LANs under different traffic conditions. We focus on HIPPI [2][14] LANs. HIPPI networks have a link speed of 800 or 1600 Mbit/second and are widely used in supercomputer centers [10][15] to connect collections of supercomputer and high-performance storage systems [8]. Several companies sell HIPPI switches. These switches are full crossbar switches: i.e. an N node switch can support N simultaneous data transfers, assuming all N sources and all N destinations are distinct.

Communication over HIPPI is based on circuit switching: the sender first sets up a connection to the destination through one or more switches, once the connection has been established it sends the data (typically one packet) and the sender then breaks the connection. If any of the links between the source and the destination is in use by a different connection, the connection is rejected and the sender has to retry at a later time. Alternatively, the sender can specify in the connection request that the connection request should block in the switch until the link frees up (camp on).

Several studies have looked at the throughput over HIPPI, but they typically make several simplifying assumptions about both the network (no overhead for setting up connections) and the traffic conditions (fixed packet sizes and uniform packet destinations). In this paper we describe simulation results in which we relax both of these constraints for a single switch HIPPI network. In our simulations, the network adapter that connects the host to the network can store multiple packets and it can rapidly try to establish connections for these packets following a specific schedule (e.g. round robin). This is for example the approach used in the gigabit Nectar project [13][12]. We look at the impact of using different scheduling techniques on throughput.

The remainder of this paper is organized as follows. In Sections 2 and 3 we describe our methodology and earlier work on packet switching. We present results for network without overhead and fixed sized packets in Section 4 and we then relax these constraints: we look at variable packet sizes (Section 5), and add overhead to the switch (Section 6). We briefly look at non-uniform packet destinations in Section 7. We conclude in Section 8.

This research was sponsored in part by the Defense Advanced Research Projects Agency (DOD) under contract number MDA972-90-C-0035, and in part by the National Science Foundation and the Defense Advanced Research Projects Agency under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives.

2 Methodology

We describe the network architectures evaluated in this paper and our simulation methodology.

2.1 Network Model

We study packet networks that are implemented on top of a circuit switched hardware: a circuit is set up from the source to the destination for each packet. HIPPI networks follow this model. We focus on the case of distributed control: packets are stored on the network adapter that connects the node to the switch and the adapter is responsible for setting up the connection through the network, sending the packet over the connection and closing the connection. Since adapters do not have knowledge of the entire state of the network, connection requests can be rejected because the requested output port is busy. When that happens, the adapter will try to send the packet later, similar to a retry after a collision on an Ethernet.

The adapter can use different strategies to select a packet to send. We use three scheduling strategies:

- FIFO scheduling: the adapter keeps a FIFO of packets to send, and packets are sent strictly in the order in which they were submitted to the network. This is the simplest strategy.
- window control: again the adapter keeps a FIFO with packets, but the scheduler has access to the first W packets in the FIFO. It tries to send packets in this window in round robin fashion.
- logical channels: the adapter has access to the heads of L queues of packets called logical channels. When scheduling a packet for transmission, the host assigns the packet to a specific logical channel, typically based on the packet destination, e.g. different logical channels serve different destinations or groups of destinations. This approach is used in gigabit Nectar [13].

With $L = W = 1$, the window and logical channel strategies are identical to FIFO scheduling, which is similar to the use of "camp on" in HIPPI terminology.

We will also consider the case of central control, i.e. all scheduling decisions are made based on full knowledge of the current state of the network. This could be implemented by having scheduling done on the switch, or by having the information broadcasted to the adapters. In the case of central control we assume window scheduling for each port, i.e. the scheduler has access to the information of the first W packets waiting at each input port. When an output port frees up and multiple packets are available, it randomly picks one. When a packet is scheduled for transmission, the data flows from the source, through the input port to the output port (making both ports busy) and then to the destination. If another packet is available for transmission, it will take the place of the transmitted packet in the window.

In this paper, we focus on networks with a single switch. We present results for an ideal switch (no overhead), and for the case that packet scheduling and connection set up incur an overhead. If set up is free, the performance of networks with central and distributed control are identical. Our performance measure is saturation throughput, defined as the ratio of the total number of bits sent over the maximum aggregate switch bandwidth (number of ports times the link bandwidth).

2.2 Simulation methodology

We use an event driven simulator that models the operation of the network. Events fall in three categories: application events (e.g. generate a new packet), adapter events (e.g. try to establish a connection, or handle an end of packet event) and switch event (e.g. handle an incoming connection request). The accuracy of the simulator is one microsecond; all events scheduled for the same one microsecond time slot are executed in random order. All simulations were run for 5 to 20 seconds of simulated time, depending on how long it took to get steady state results. Typically, scenarios that use large packet sizes had to run longer.

3 Earlier work

In [7], Hluchyj and Karol study the impact of four different queuing organizations on the performance of packet switching networks consisting of a single nonblocking switch. They assume a fixed packet size and free connection set up, and found that with FIFO queuing, the saturation throughput is 0.586. The throughput increases with the window size. In [9][11], the above study was extended to the case that an output port can accept multiple packets simultaneously. For large switches, and assuming a uniform distribution of the destinations, the maximum throughput is more than 0.99 when more than three packets can be received at a time.

In [1], the authors use simulation to determine the saturation throughput of a switch-based network using central control. There is a cost for connection set up and fixed-sized packets are used. The results show that the switch performs better under congestion than a bus and that performance degrades as the packet size decreases.

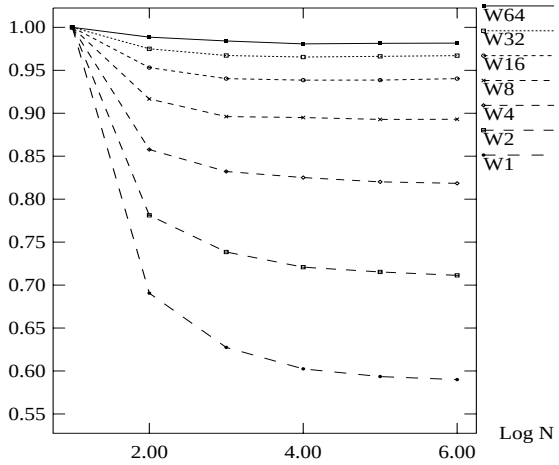

Figure 1: Throughput with ideal switch and fixed packet size

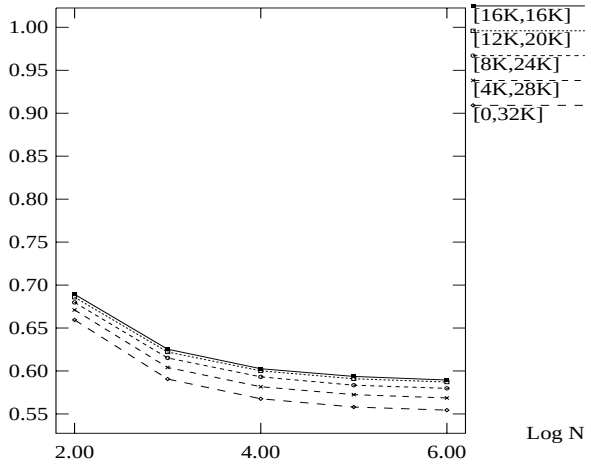

Figure 2: Throughput with ideal switch, FIFO scheduling and uniform packet size

In [4][6], the authors model a single switch network with negligible connection setup cost. Both the packet size and the inter-packet arrival times follow an exponential distribution. Different connection management strategies and service disciplines are studied. The results show that central connection management outperforms distributed control, and that a large window size is superior to FIFO queuing.

In [5], three different policies for setting up connections in HIPPI systems are compared. Besides the centralized and the distributed strategies, which are similar to the ones we study, the authors consider a broadcast strategy, where all nodes are notified when an outport frees up, allowing them to retry a destination as soon as it frees up. The study shows that the broadcast strategy performs better than the centralized approach, but worse than the distributed one. However, similar to most other studies, only scenarios with fixed packet sizes, FIFO scheduling, uniform destination distribution, and free connection set up are considered.

4 Simple Network and Traffic Model

We first consider the case that the set up connection time is 0, all packets are of the same size and packet destinations are uniformly distributed across all destinations. Figure 1 shows the saturation throughput as a function of the switch size (logarithmic scale) for different window sizes. Our simulation results match the analytical and simulation results in [7].

For a fixed window size, the throughput decreases with increasing switch sizes. The reason is that the chances of a conflict for an output port increase with the switch size. With FIFO scheduling (window size equal to 1), the throughput flattens out at about 58% percent. Using a window scheduling policy improves the throughput. With a window size of 16, the throughput is over 90% for all switch sizes, while the throughput is over 98% with a window size of 64. We conclude that window scheduling is effective at increasing throughput in this case.

5 Traffic with variable packet size

We relax the constraint that packet sizes are fixed, but we still assume an ideal switch and uniform packet destinations. We consider uniform and bimodal packet size distributions, and we study FIFO and window scheduling strategies. Note that under these conditions, logical channels (with the number of logical channels equal to the number of ports) have a saturation throughput of 100%.

5.1 Uniform packet size distribution

Figure 2 shows the saturation throughput with the packet size uniformly distributed over an interval $[l, u]$, and FIFO queuing. The graph shows that the throughput decreases as the interval length $u - l$ increases. This performance loss is a result of the FIFO scheduling strategy: when a packet occupies an output port, it prevent any node with a packet for that output port at the head of the queue from sending. Intuitively, as the difference in the packet sizes grows, it becomes more likely that a single (or a small number of) large packets will force senders to be idle, since senders can on average send more packets during the transmission of a single large packet, thus increasing the chance that they will block on a busy output port.

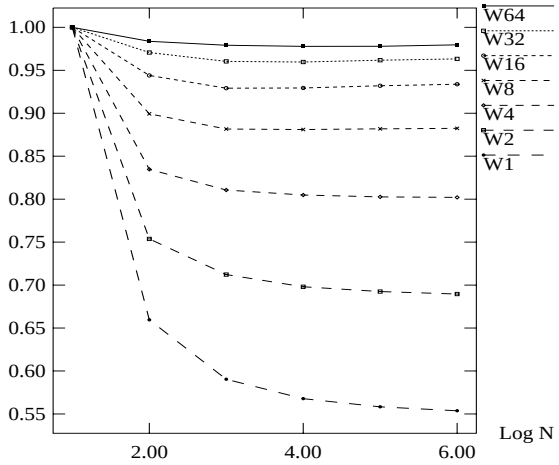

Figure 3: Throughput with ideal switch, window scheduling, and uniform packet size [0,32000]

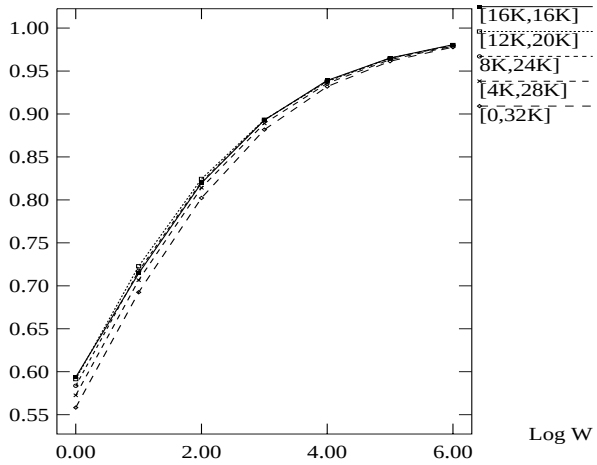

Figure 4: Throughput with ideal switch (size 32), window scheduling and uniform packet size

Figure 3 shows what happens if we use a window scheduling strategy with a uniform packet size distribution; the packet size distribution intervals is [0, 32000]. We observe that window scheduling helps in increasing the saturation throughput considerably. A comparison of Figures 1 and 3 shows that the difference in throughput compared with fixed-sized traffic is negligible for large window sizes. Figure 4 confirms this: it shows the throughput as a function of the window size for different uniform packet size distributions.

5.2 Bimodal distribution

Studies of network traffic show that the packet size distribution is typically bimodal: the packet size is either the maximum transfer unit (MTU) for the network, or the packets are short. In this section we look

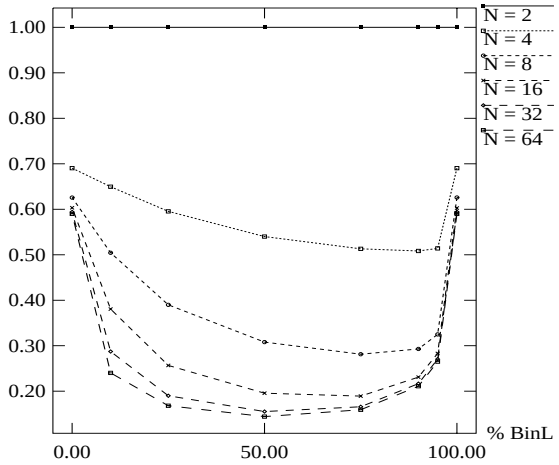

Figure 5: Throughput with ideal switch, FIFO scheduling and bimodal packet size (L = 16000)

at the impact of having a bimodal packet distribution on saturation throughput. We approximate the distribution found on existing networks by using a fixed size for short packets (100 bytes).

Two parameters control the traffic distribution: the percentage of long (short) packets, and the long packet size. We use the "percentage of bytes in long packets" as a parameter instead of the more obvious "percentage of long packets", because it is more characteristic of a traffic load than the number of long packets. For example, if the MTU of the network changes, the percentage of long packets will change dramatically. There will also be a change in the number of short packets (e.g. acks), since it depend on the number of long packets, but overall, the impact on the percentage of bytes in long packets will be small.

In practice, the distribution of bytes between long and short depends on the environment in which the network is used. One traffic study [3] for general internet traffic indicates that about 80% of the bytes are in long packets. The average short packet size is 100 bytes in that study. We expect a higher percentage of bytes in long packets in supercomputer or distributed computing environments.

Figure 5 shows the saturation throughput with FIFO scheduling for different switch sizes and different distributions of bytes between long and short packets (BinL is Bytes in Long packets); long packets are 16000 bytes. The extreme cases when all bytes are in long or short packets correspond to the fixed-sized packet case. Figure 5 shows that for switch sizes larger than 2, a bimodal packet size distribution results in a much lower throughput than fixed packet

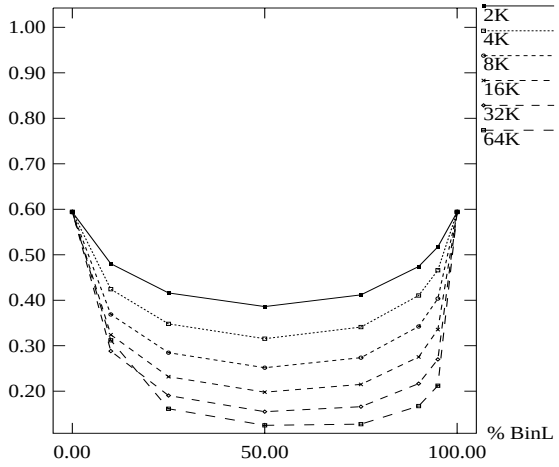

Figure 6: Throughput with ideal switch (size 32), FIFO scheduling and bimodal packet size

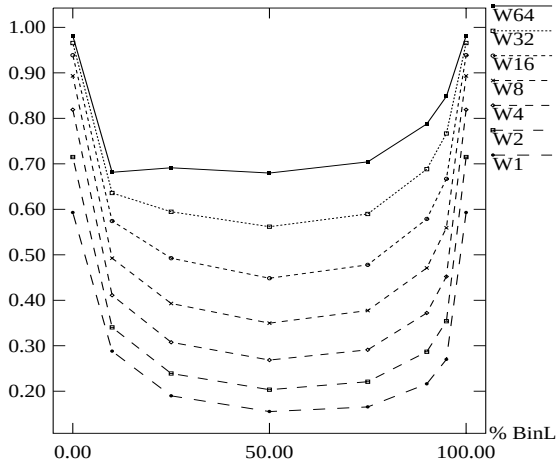

Figure 7: Throughput with ideal switch (size 32), window scheduling and bimodal packet size (L = 32000)

sizes. For example, when the switch size is 16, the throughput decreases form 0.60 (100% BinL) to 0.22 (90% BinL). The drop in throughput is most dramatic when, starting with all bytes in long packets, we move just a few percent of the bytes into short packets. For large switches, the throughput is minimal when bytes are about equally distributed between long and short packets.

The reason for this throughput loss is the same as with the uniform packet distribution: occasional large packets may occupy an output ports for a long time, thus blocking the transmission of many small packets to these output ports from other input ports. When the switch size becomes larger, the number of input ports that are blocked increases, so the throughput decreases faster. This throughput loss becomes worse when the size of long packets increases, as is shown in Figure 6. For instance, when the size of long packets is 64000 bytes, the performance can be worse than 15%. An important conclusion is that for a specific traffic mix (i.e. certain percentage of bytes in long packets), it is not necessarily advantageous to use the largest possible packet size.

Figure 7 shows the effect of using window scheduling with different window sizes on the saturation throughput; the results are for a 32 node switch and a long packet size of 32000 bytes. We again see that window scheduling is effective at increasing throughput, although the throughput remains well below the throughput with fixed-sized packets. For example, with a window of size of 32, the throughput drops from 97% with fixed sized packets to 59% with 75% of the bytes in long packets, to 56% with 50% of the bytes in long packets (the worst case).

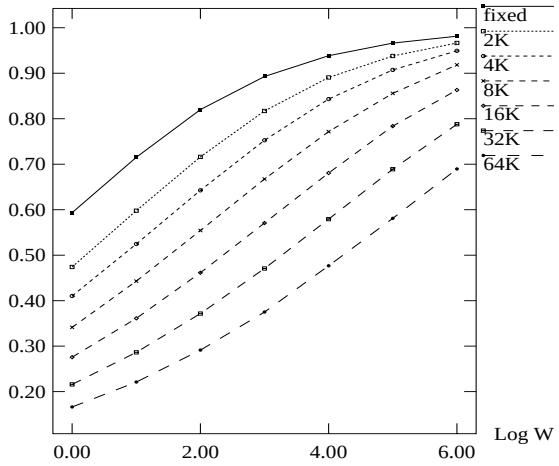

Figure 8: Throughput with ideal switch (size 32), window scheduling and bimodal packet size (90% BinL)

Figure 8 shows the impact of the size of long packets on the throughput with 90% of the bytes in long packets. We see the same pattern as with FIFO scheduling: reducing the size of the large packets results in a more effective sharing of the network. A combination of large windows and a relatively small MTU size gives the best performance. However, the impact of the packet size is smaller for large window sizes, in other words, with large window size, the packet size can be increased with relatively small impact on performance.

In the previous discussion, we used a short packet size of 100 bytes. The size of short packets does affect the saturation throughput in bimodal distributions.

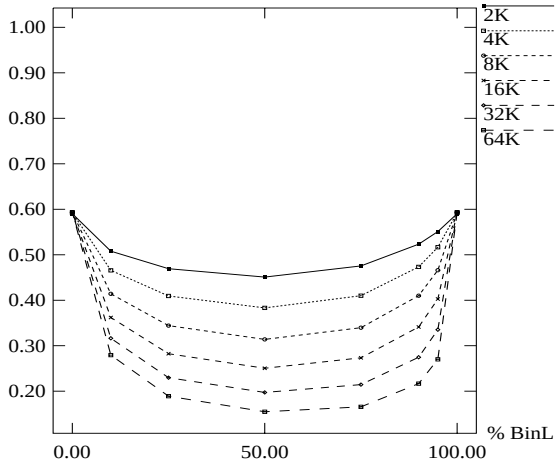

Figure 9: Throughput with ideal switch (size 32), FIFO scheduling and bimodal packet size with short packet size of 200.

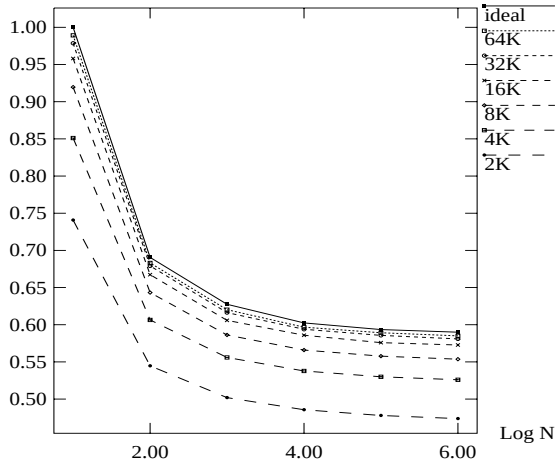

Figure 10: Throughput with overheads and distributed FIFO scheduling

In Figure 9, we use a short packet size of 200 bytes. A comparison with with Figure 6 shows that a larger short packet size improves performance slightly. This should not be a surprise since this effectively reduces the variance in the packet size distribution. The impact is however small so we will only use a single short packet size (100 bytes) in the remainder of the paper.

5.3 Summary

We showed that the packet size distribution has a big impact on throughput. In general, the larger the variance in packet size, the worse the performance. However, the simulations show that window scheduling can be used to counteract the adverse effects of the packet size distribution.

We observed that, given a certain percentage of bytes in long packets, increasing the maximum packet size reduces throughput, because the sharing of the network is less effective. Note however we have not yet taken the effect of per-packet overhead into account. We also observed that, in most cases, throughput increases as more bytes travel in long packets. This is good news, since we expect a high percentage of bytes in long packets in supercomputer applications and distributed computing, e.g. in the environments where HIPPI is most commonly used.

6 Scheduling overhead

We make the model of the network more realistic by adding scheduling overhead. In the case of distributed control, we add two microseconds to select the next packet, and five microseconds to attempt to set up a connection. Note that these costs occur independent from whether the packet can be sent or not. Since, a request for a connection from an adapter may be refused many times before a packet can be sent, these overheads can be incurred many times per packet. We consider the impact of different overheads on the performance later in the section.

6.1 Fixed packet size

Figure 10 shows the throughput as a function of the switch size using FIFO scheduling and for fixed packet sizes, and compares it with the case of no overhead (labeled ideal in the graph). As expected, the scheduling overhead reduces the throughput and the reduction is more severe for smaller packets. Figure 11 shows the impact of window scheduling. We again see that window scheduling increases throughput, although it cannot make up for the scheduling overhead. For packet sizes of 8000 bytes of smaller, the throughput remains under 75%.

It is interesting that the impact of the overhead is more significant for window scheduling than for FIFO scheduling. The change in performance when the long packet size is changed from 2000 bytes to 64000 bytes is less than 15% with FIFO scheduling, but is more than 35% with window scheduling with the window size is 32. The reason is simple. With FIFO scheduling, the adapter is always trying to send the same packet at the head of the queue, so if that destination frees up, the connection will be established almost immediately, and the cost of the overhead is at most the cost of one connection attempt. With window scheduling, however, the adapter tries to send packets in round robin fashion, and when a connec-

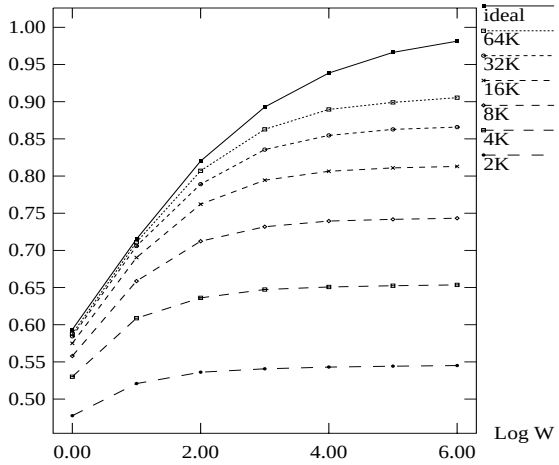

Figure 11: Throughput for 32 node switch with overhead, and different window and packet sizes

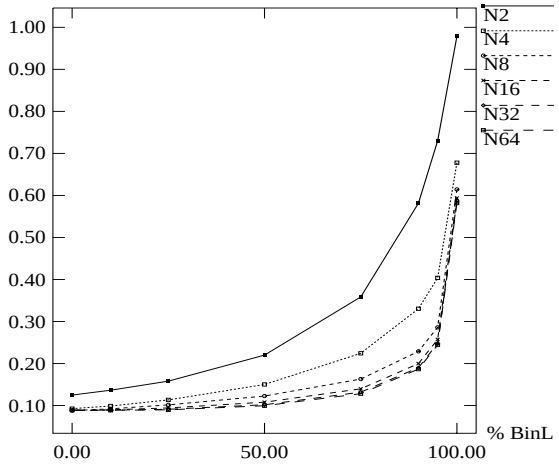

Figure 12: Throughput for a 32 node switch with overhead, FIFO scheduling, and bimodal distribution

tion frees up, several other attempts might be made for other packets before this is observed, so the delay can be the cost of several connection attempts. Note that with overhead, window scheduling still performs better, but the advantage is smaller.

6.2 Bimodal packet size distribution

Figure 12 shows the throughput as a function of the percentage of bytes in long packets, for different switch sizes; the long packet size is 32000 bytes and FIFO scheduling is used. We observe that the throughput decreases monotonically with the decreasing percentage of bytes in long packets. The effect is quite dramatic and the throughput drops to less than 10% if most of the bytes are in small packets. This trend is in contrast with the results of Figure 5 (no overhead), where the throughput was minimal with about 50% of the bytes in long packets. The reason is that as more bytes travel in short packets, the number of packets increases, and so does the impact of the (per-packet) overhead.

Note that the curve corresponding to a switch size of 2 ($N2$) in Figure 12 corresponds to the case that there is no queuing delay (since there is only a single destination). The curve effectively divides the "lost throughput" in two components: the space above the grey curve is throughput that is lost due to (minimal) scheduling overhead, while the space between the $N2$ curve and the other throughput curves is throughput that is lost as a result of destination conflicts and resulting queuing delays. We see that the scheduling overhead is quite large, unless almost all the bytes are carried in long packets, so the low throughput should not be a surprise.

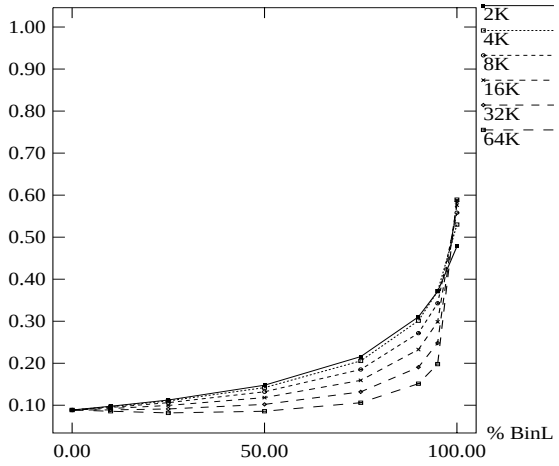

Figure 13: Throughput for a 32 node switch with overhead, FIFO scheduling, and bimodal distribution

Figure 13 shows the throughput as a function of the percentage of bytes in long packets for different long packet sizes. We see that the optimal long packet size depends on the percentage of bytes in long packets. In most cases, smaller large packet sizes work better. The reason is that the sharing of the network is more effective with smaller long packets, as was discussed in Section 5. However, if almost all bytes are in long packets, it becomes more attractive to use larger large packet sizes. The reason is that this scenario is similar to that of fixed packet sizes, and using a larger packet size reduces the effect of the scheduling overhead.

Figure 14 shows that adding window scheduling improves performance, but only in a limited way. There is a benefit of 20% or more if most bytes are in long

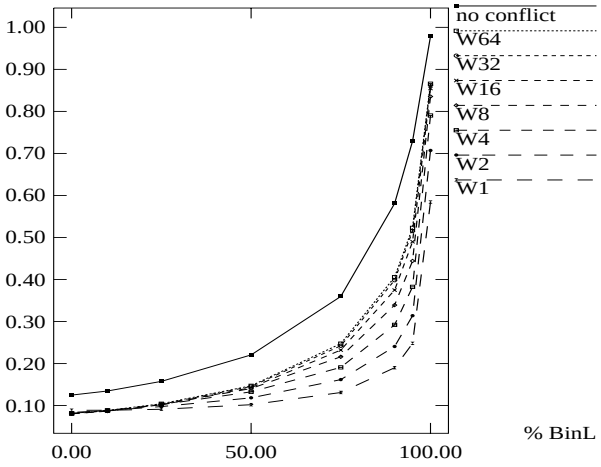

Figure 14: Throughput with overhead, bimodal packet size (L = 32000), and 32 node switch

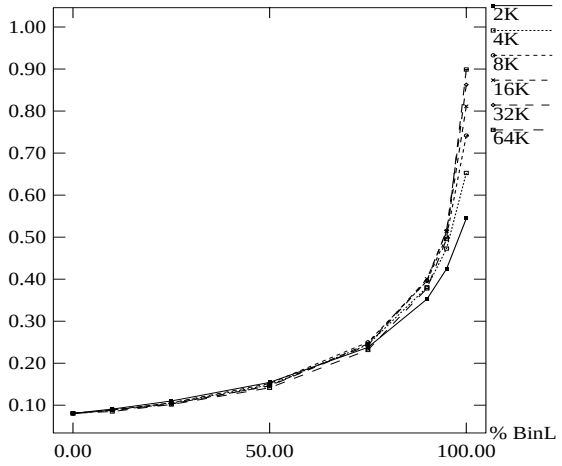

Figure 15: Throughput for a 32 node switch with overhead, bimodal distribution, and window scheduling (W = 32)

packets, but as soon as a reasonable percentage of the bytes are in short packets, the value of window scheduling disappears. The curve labeled "no conflict" perfect shows the throughput that we would have if scheduling were perfect (all connections can be made immediately and there is never a queuing delay) and the only overhead is scheduling overhead. We observe that the scheduling overhead reduces the throughput considerably, and window scheduling cannot recover that loss: it can only reduce the queuing delay.

Figure 15 shows the impact of the long packet size on the throughput for a 32 node switch and and a window size of 32. We can make several observations. First, window scheduling has more impact if

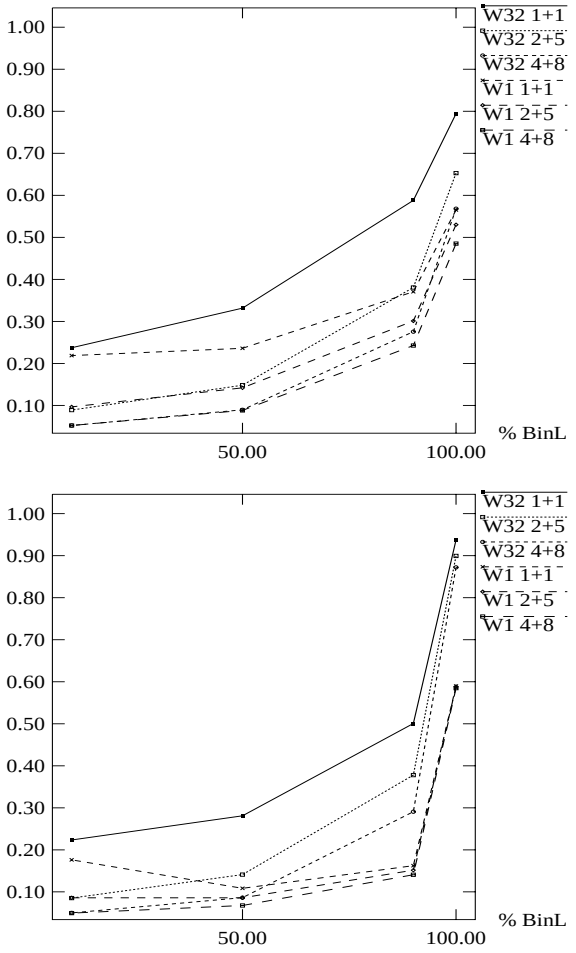

Figure 16: Throughput with different overheads, L = 4000 (top) and 64000 (bottom), and N = 32

more bytes are in long packets (compare with Figure 13). The reason is that with smaller large packet sizes, more throughput is lost due to scheduling overhead (and less to scheduling conflicts), and window scheduling cannot recover that bandwidth. Second, the best large packet size depends on percentage of bytes in long packets, as we already observed in Figure 13 for FIFO scheduling. Finally, if less than 75% of the bytes are in long packets, the size of the long packets has little impact on the throughput.

6.3 Changing the overhead

All earlier results use the same control overhead. Figure 16 shows the impact on performance of changing the overhead for two different maximum packet sizes. The impact of a higher overhead is quite substantial. We also see that changing the overhead has more impact when the average packet size is small, i.e., when the percentage of bytes in long packets is

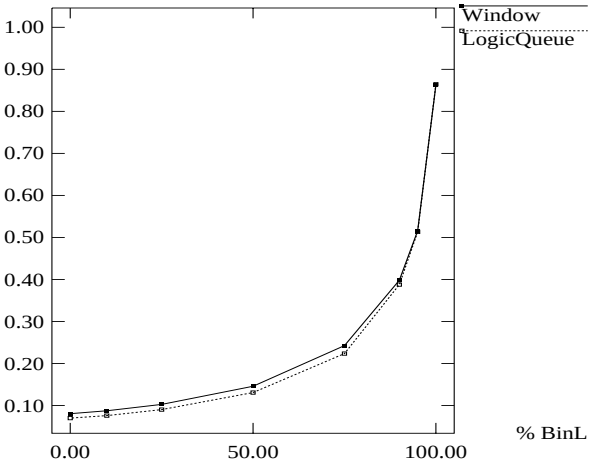

Figure 17: Throughput for switch with overhead using logical channel and window scheduling and bimodal packet size (L = 32000)

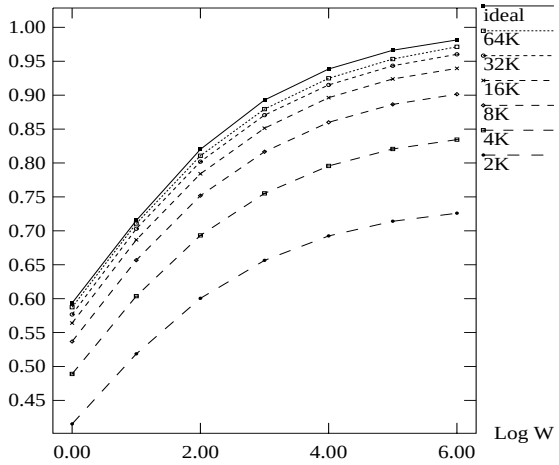

Figure 18: Throughput with overheads and using central control

small, or when the long packet size is small.

Another observation is that increasing window size does not reduce the performance loss caused by increasing the overhead. In fact, with higher overheads, window scheduling becomes less effective. This confirms that window scheduling cannot overcome scheduling overheads, and that the polling that takes place as part of window scheduling becomes more expensive when overheads increase. Note that when HIPPI networks are used for MAN or WAN interconnects, the cost of setting up connections increases dramatically as a result of the propagation delays. Our results indicate that this is in general not a good solution.

6.4 Logical channels

In this section we compare window-based scheduling with scheduling based on logical channels. As discussed earlier, with logical channels, the adapter keeps packet in a per-destination queue, and tries to send packets from the queues in round-robin fashion. Figure 17 shows the throughput with logical channels (L = 32) and window scheduling (W = 32) for a 32 node switch with long packets of size 32000. The results are almost identical. Under saturation conditions, we expect the two strategies to behave in a similar way, since both always have packets to send. The only difference is that when packets for a busy destination accumulate on the adapter, that destination will be retried at a higher rate with window-scheduling.

6.5 Central control

With central control, the packet scheduler has complete knowledge of the state of the input queues (or windows), when selecting the next packet to be sent, so the connection request will always succeed. We use a cost of two microseconds to select a packet and to set up a connection.

Figure 18 shows the saturation throughput when central control and window scheduling is used. The curve for the ideal case without overhead is also displayed. The graph shows that the performance loss due to overhead in the central scheduling is proportional to the time sending an average sized packet. The reason is that with central control, the overhead for selecting a packet is incurred exactly once for every packet.

A comparison of Figure 18 with Figure 11, which shows the throughput under the same conditions but with distributed control (which also has higher overhead), shows that central control outperforms distributed control, as one would expect. We also observe that window scheduling pays off more in the case of central control than in the case of distributed control. The reason is that the central controller has access to the windows of all ports, so increasing the window size has a larger impact.

6.6 Summary

We showed that scheduling overhead reduces throughput substantially. The impact is dramatic when a lot of the bytes travel in short packets. Window scheduling still pays off in comparison with FIFO scheduling, but it is not effective at reducing scheduling overhead.

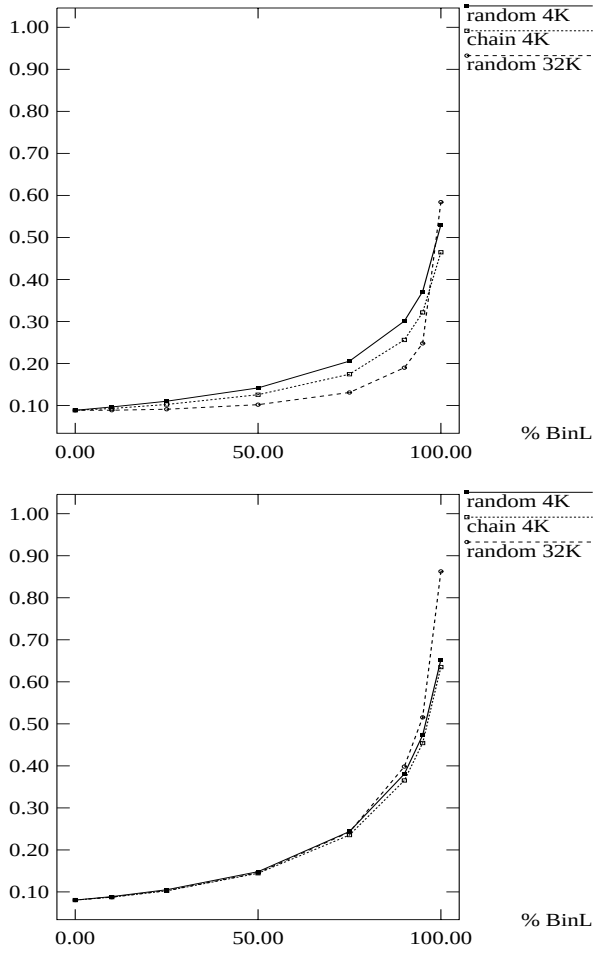

Figure 19: Throughput with packet trains, window size 1 (top) and 32 (bottom), and N = 32

7 Effect of correlation between packet destinations

Up to this point we have a uniform distribution of packet destinations. In this section we first look at the effect of packet trains, sequences of packets to the same destination. This is for example a model of the effect of packetization and fragmentation. We then look at what happens if a lot of the traffic is within a subset of the nodes, for example a set of file servers. The scheduling overhead is $5 + 2$.

7.1 Packet trains

In Figure 19, we compare two different methods for transmitting 32000 byte packets (bimodal packet distribution): as a single packet and as a sequence of eight 4000 byte packets. We compare these approaches with the case of long packets of size 4000 bytes. With FIFO scheduling (top), the packet train approach outperforms sending 32000 byte packets. Both behave worse than (random) 4000 byte long packets. The reason is that the less efficient sharing of the network that results from sending very large packets has more impact on the performance than the increased overhead that results from breaking up the large packets into a sequence of smaller ones.

With a large window size (bottom), the three methods result in very similar performance when the percent of bytes in long packets drops under a certain level (e.g., 75%). With window scheduling, the scheduling overhead has more impact than with FIFO scheduling, and the two effects (less efficient sharing versus increased overhead) cancel each other out.

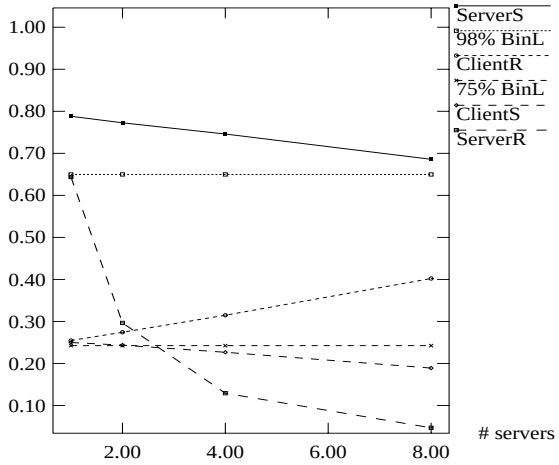

Figure 20: Throughput with concentrate traffic, window 32, and bimodal packet size (L = 32000)

7.2 Concentration of traffic

We now analyze a more complicated scenario. The network has a number of file servers (shown on the x-axis) and a number of clients; the total number of nodes is 32. Clients send 80% of their traffic to the file servers (98% of bytes in long packets) and 20% of their traffic to other clients (75% of bytes in long packets). The file servers only sends to clients (98% of bytes in long packets). Figure 20 shows the throughput of the different traffic streams. The picture also shows the throughput for uniform packet distributions with 98% and 75% of the bytes in long packets (from Figure 15).

We observe that the file servers can send at high rates; this is a result of the high percentage of bytes in long packets for that traffic stream. However, the servers are receiving data at very low rates. The reason is that the clients can only send at a low rate, since their communication with other clients is inefficient; this slows down the rate of the more efficient

communication with the server. We expect that better scheduling techniques (i.e. other than round robin) can be used to improve performance of the communication between the client and servers.

8 Conclusion

We showed that the traffic characteristics and overhead have an enormous impact on performance. The throughputs observed differ by as much as a factor of 10. In general, a high variance in the packet size distribution and high scheduling overhead reduce throughput. More intelligent scheduling on the adapter, e.g. window scheduling, can overcome most of the problems created by the packet size distribution, but the negative effect of scheduling overhead cannot be reduced that way.

These results are only a first step in understanding the performance of circuit switched LANs as a function of the traffic characteristics. More work is needed to find better scheduling techniques that improve the throughput of realistic traffic loads.

References

[1] D. Abensour, H. Meleis, A.N. Tantawi, and D. Zumbo. Structure for High Speed Network Node. Technical Report RC 16421 (72968), IBM, January 1991.

[2] ANSI. High-Performance Parallel Interface - Mechanical, Electrical and Signalling Protocol Specification (HIPPI-PH). ANSI X3.183-1991, 1991.

[3] Ramon Caceres. Efficiency of ATM Networks in Transporting Wide-Area Data Traffic. Submitted for publication, 1992.

[4] Imrich Chlamtac and Aura Ganz. Performance Evaluation of an HIPPI Interconnection System. Technical Report RC 16482 (71965), IBM, October 1990.

[5] Imrich Chlamtac, Aura Ganz, and Martin G. Kienzle. An HIPPI Interconnection Systems. *IEEE Transactions on Computers*, 42(2):138–150, February 1993.

[6] Imrich Chlamtac and Martin G. Kienzle. Multitasking in High-Speed Interconnection Systems. Technical Report RC 16599 (73369), IBM, February 1991.

[7] Michael G. Hluchyi and Mark J.Karol. Queueing in High-Performance Packet Switching. *IEEE Journal on Selected Areas in Communications*, 6(9):1587–1597, December 1988.

[8] James P. Hughes. HIPPI. In *Proceedings of the 17th Conference on Local Computer Networks*, pages –. IEEE, October 1992.

[9] M.J. Karol, M.G. Hluchyj, and S.P. Morgan. Input versus Output Queueing on a Space-division Packet Switch. *IEEE Transactions on Communications*, COM-35:1347–1356, December 1987.

[10] Jamshid Mahdavi, Gwendolyn L. Huntoon, and Matthew B. Mathis. Deployment of a HIPPI-based Distributed Supercomputing Environment at the Pittsburgh Supercomputing Center. In *International Parallel Processing Symposium*, pages –, Los Angeles, April 1992. IEEE.

[11] Yuji Oie, Masayuki Murata, Koji Kubota, and Hideo Hiyahara. Performance Analysis of Non-blocking Packet Switch with Input and Output Buffers. *IEEE Transactions on Communications*, 40(8):1294–1297, August 1992.

[12] Peter A. Steenkiste, Michael Hemy, Todd Mummert, and Brian Zill. Architecture and Evaluation of a High-Speed Networking Subsystem for Distributed-Memory Systems. In *Proceedings of the 21th Annual International Symposium on Computer Architecture*, page to appear. IEEE, May 1994.

[13] Peter A. Steenkiste, Brian D. Zill, H.T. Kung, Steven J. Schlick, Jim Hughes, Bob Kowalski, and John Mullaney. A Host Interface Architecture for High-Speed Networks. In *Proceedings of the 4th IFIP Conference on High Performance Networks*, pages A3 1–16, Liege, Belgium, December 1992. IFIP, Elsevier.

[14] Don E. Tolmie and Marty G. Halvorson. HIPPI / Serial-HIPPI. In *Thirty Seventh IEEE Computer Society International Conference*, pages 222–228. IEEE, February 1992.

[15] Ronald J. Vetter, David H. C. Du, and Alan E. Klietz. Network Supercomputing. *IEEE Network Magazine*, 6(3):38–44, May 1992. Describes the use of the Cray-CM2 connection at the Minnesota Supercomputing Center.