Virtual Channel Multiplexing in Networks of Workstations with Irregular Topology [*]

F. Silla, J. Duato
Dpto. Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
Camino de Vera, 14, 46071–Valencia, Spain
{fsilla,jduato}@gap.upv.es

A. Sivasubramaniam, C. R. Das
Dept. of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
{anand,das}@cse.psu.edu

Abstract

Networks of workstations are becoming a cost-effective alternative for small-scale parallel computing. Although they may not provide the closely coupled environment of multicomputers and multiprocessors, they meet the needs of a great variety of parallel computing problems at a lower cost. However, in order to achieve a high efficiency, the interconnects used to build the network of workstations must provide a very high bandwidth and low latencies, making their design a critical issue.

Recently, a very efficient flow control protocol for networks of workstations has been proposed by the authors. This protocol multiplexes physical channels between several virtual channels and minimizes the use of control flits by transmitting several data flits each time a virtual channel gets the link. In this protocol, a virtual channel sends data flits until the message blocks or is completely transmitted. However, it can reduce network throughput, by increasing short message latency, due to long messages monopolizing channels and hindering the progress of short messages.

In this paper, we analyze the impact of limiting the number of flits (block size) that a virtual channel can send once it gets the link. We propose a new version of the previous flow control protocol that is easily implementable on hardware. Simulation results show that limiting the maximum block size is not a good design decision, because the overall network performance decreases. Only when short message latency is crucial, it is acceptable to limit the block size.

1 Introduction and Motivation

Networks of workstations (NOWs) are being considered as a cost-effective alternative for small-scale parallel computing. Although NOWs may not provide the closely coupled environment of multicomputers and multiprocessors, they meet the needs of a great variety of parallel computing problems at a lower cost (see [1] as an example). However, in order to achieve a high efficiency, the interconnects used to build the network of workstations must provide high bandwidth and low latencies, making their design a critical issue. Recent proposals for NOW interconnects, like Autonet [11], Myrinet [2], and ServerNet [6] are switch-based and use point-to-point links between switching elements instead of the traditional bus used in computer networks. Usually, they present an irregular topology as a consequence of the needs in a local area network. This irregularity provides wiring flexibility, scalability, and incremental expansion capability required in this environment.

As a consequence of using point-to-point links, messages have to be routed through several switches until they reach their destination. Therefore, there is a need for efficient routing in switch-based networks with irregular topology. Several deadlock-free routing algorithms have been proposed [11, 10]. These algorithms are inherently different from those in regular networks, mainly due to the irregular connections between switches. In these algorithms, routing is considerably restricted in order to avoid deadlock. Recently, a general methodology for the design of adaptive routing algorithms has been presented [14, 12]. The resulting algorithms increase the maximum achievable throughput while reducing message latency. These algorithms make use of two virtual channels per physical channel.

In order to make efficient use of link bandwidth, a flow control protocol has been proposed [13]. This flow control protocol takes into account real aspects of the NOW environment, such as the long links. It relies on the use of channel pipelining and control flits. Traditionally, when a physical link is multiplexed, flits from the different virtual channels are transmitted alternatively one after another on a flit-by-flit basis, and several control lines indicate which vir-

[*]This work was supported by the Spanish CICYT under Grant TIC97–0897–C04–01

tual channel is owning the physical link during each transmission cycle. However, links in NOWs are usually narrow, and therefore, it is not recommended to use dedicated lines to signal the virtual channel owning the physical link. Thus, control flits are used to indicate which virtual channel owns the link [13]. In this case, before transmitting a data flit, a control flit must be sent in order to specify which virtual channel the following data flit belongs to. This flow control strategy wastes network bandwidth because half the bandwidth is wasted in transmitting control information. Instead of using flit-by-flit multiplexing, a better mechanism that minimizes the use of control flits is to transmit several data flits each time a virtual channel gets the physical channel. This way, a virtual channel will transmit its identifier, and then send flits until it cannot continue sending any more flits because the message blocks or it has been completely transmitted. At this time, the next ready virtual channel will transmit its identifier followed by data flits, and so on.

This flow control protocol can reduce network throughput, because short–message latency can increase if long messages monopolize channels, hindering the progress of short messages. Moreover, this protocol can increase the standard deviation of latency. A solution to this problem is to limit the number of flits a message can transmit through the physical link before it grants the link to another ready message. Limiting the maximum number of consecutive flits from the same message may result in a fairer use of the channel.

In this paper we propose and evaluate this new flow control protocol. The rest of the paper is organized as follows. In Section 2, networks of workstations are briefly introduced, in order to make the paper self-contained. Section 3 explains in more detail the insights of the proposed flow control protocol. In Section 4, the performance of this new protocol is compared with the previous one. Finally, Section 5 summarizes the conclusions from this study.

2 Networks of Workstations

Networks of workstations are usually arranged as switch-based networks with irregular topology. In these networks each switch is shared by several workstations, which are connected to the switch through some of its ports. The remaining ports of the switch are either left open or connected to other switches to provide connectivity between the workstations. Links in a NOW are typically bidirectional full-duplex, and multiple links between two switches are allowed. Figure 1 shows a typical NOW using a switch-based interconnect with irregular topology. It is assumed in this figure that switches have eight ports and each workstation has a single port.

Different switching techniques, like wormhole, virtual cut-through, or ATM, can be implemented in NOWs. Re-

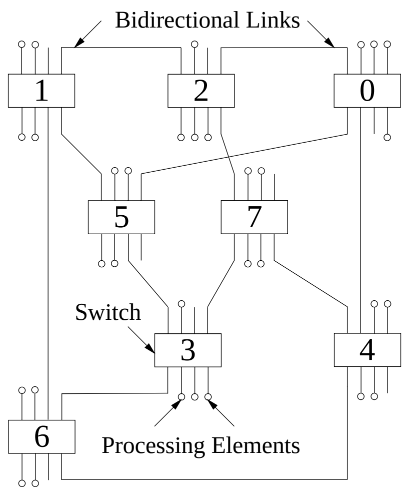

Figure 1. A network with switch-based interconnect and irregular topology.

cently proposed networks like Myrinet [2] and ServerNet [6] use wormhole switching. Therefore, we will restrict ourselves to wormhole switching in this paper.

Routing decisions can be based on source routing or on distributed routing. In the former, the message header contains the sequence of ports to be crossed along the path to the destination [2]. In the latter, each switch has a routing table that stores the output ports that can be taken by the incoming message. Regardless of where decisions are taken, a routing algorithm must determine the path to be followed. Several deadlock-free routing schemes have been proposed for irregular networks [11, 2, 6, 10]. Moreover, a general methodology for the design of adaptive routing algorithms for irregular networks has been recently proposed [14, 12]. In order to make this paper self-contained, we briefly present the routing algorithm used in the simulations carried out in the performance evaluation section.

2.1 Up/Down Routing

The up/down routing scheme is used in Autonet [11] networks. It is a distributed deadlock-free routing algorithm that provides partially adaptive routing in irregular networks. Since it is based on table lookup, routing tables must be filled before messages can be routed. In order to fill these tables, a breadth-first spanning tree (BFS) on the graph G of the network is computed first using a distributed algorithm. Routing is based on a direction assignment to all the operational links in the network. In particular, the "up" end of each link is defined as: (1) the end whose switch is closer to the root in the spanning tree; or (2) the end whose switch has the lower ID, if both ends are at switches at the same tree level. After this assignment, each cycle in the network has at least one link in the "up" direction and one link in the "down" direction. To avoid deadlocks, routing is based on the following up/down rule: a message cannot traverse a link in the "up" direction after having traversed a link in the "down" direction.

When a message arrives at a switch, the routing algorithm is computed by accessing the routing table. If there are several suitable outgoing ports, one of them is selected.

Up/down routing is not always able to provide a minimal path between every pair of nodes due to the restriction imposed by the up/down rule. As the network size increases, this effect becomes more important.

2.2 Adaptive Routing

A design methodology for adaptive routing algorithms on irregular networks has been proposed in [14, 12]. This methodology starts from a deadlock-free routing algorithm for a given interconnection network. In the first step, all the physical channels in the network are split into two virtual channels, called the *original* channel and the *new* channel. In the second step, the routing function is extended so that it can use all the virtual channels. New channels are used with the only restriction that they must bring messages closer to their destination. Original channels are used in the same way as in the original routing function. When a message is injected into the network, it can only leave the source switch by using new channels, since they provide a higher degree of adaptivity and, usually, a shorter path. When a message arrives at an intermediate switch, it first tries to reserve a new channel. If all the suitable outgoing new channels are busy, then an original channel belonging to a minimal path is selected. To ensure that the new routing function is deadlock-free, if none of the original channels provides a minimal path to the destination, then the original channel that provides the shortest path will be used as escape path [4]. In case several outgoing original channels belong to shortest paths, only one of them will be selected. Once a message reserves an original channel, it can no longer reserve a new one. This message will be routed through original channels until it arrives at the destination switch.

This design methodology can be applied to any deadlock-free routing algorithm. In particular, the routing scheme used in this paper is the result of applying this methodology to the Autonet routing algorithm. Concisely, newly injected messages can only leave the source switch using new channels belonging to minimal paths. When a message arrives at a switch through a new channel, the routing function gives a higher priority to the new channels belonging to minimal paths. If all of them are busy, then the up/down routing algorithm is used, selecting an original channel belonging to a minimal path (if any). To ensure deadlock-freedom, if none of the original channels supplied provides minimal routing, then the one that provides the shortest path will be used. Once a message reserves an original channel, it will be routed using only original channels according to the up/down routing function until it is delivered. We would like to remark that this routing algorithm provides fully adaptive minimal routing between all pairs of nodes until messages are forced to move to original channels. When a message starts using original channels, it provides the same adaptivity as the up/down routing algorithm.

2.3 Flow Control

The routing algorithm described above makes use of two virtual channels per physical channel. There are several implementation issues for virtual channels in a NOW environment. In this section we review the flow control mechanism proposed by the authors in [13].

NOWs usually have long links that differ in their length. In order to make clock frequency completely independent of wire length, a technique called channel pipelining can be used. In this technique, flits are clocked into the communication link without waiting for the reception of the previous flit at the other end of the link. Channel pipelining supports long wires as well as wires with different lengths.

On the other hand, links in NOWs are, in general, narrow. Therefore, instead of adding several wires per link for flow control, it is preferable to use control flits. In summary, the virtual channel flow control mechanism can be stated as follows: The sending switch, before transmitting data flits through one of the virtual channels, sends a control flit called "Select n" (where n indicates which virtual channel is to follow) and then starts sending data flits until the message has been completely transferred or it blocks. Input buffers have two watermarks, "Stop" and "Go" [2], so that when the input buffer fills over the "Stop" mark, the receiving switch sends a "Stop n" control flit to the previous switch. Once the sending switch receives the "Stop n" control flit, it stops the transmission on this virtual channel and switches to the other virtual channel (if ready), sending a "Select n" control flit followed by data flits. When the input buffer at the receiving switch empties below the "Go" mark, it sends a "Go n" control flit to the sending switch. Therefore, while sending flits on a virtual channel, a "Go n" control flit on the other virtual channel may arrive. In this case, the sending switch does not switch to that virtual channel immediately, but records this change of status.

3 Block Multiplexing

The flow control protocol revisited in Section 2.3 fits the needs of NOWs, where links are generally long and narrow, and have different lengths. However, it is not very suitable when there is a mix of short and long messages in the network. In this case, network throughput may be reduced because of long messages that monopolize physical channels and hinder the progress of short messages, increasing their latency.

One way to make short messages progress in spite of long messages is to limit the number of transmission cycles a virtual channel is allowed to own the physical channel. In this case, virtual channels containing long messages, after transmitting a predetermined number of flits (maximum block size[1]), will be forced to grant the physical channel to the next ready virtual channel. If the other virtual channels in that physical channel are not ready to transmit, then the same virtual channel will resume transmission and the process repeats. In this case, the virtual channel identifier is not sent again because the virtual channel that owns the link has not changed and the receiving switch will store incoming flits in the same input buffer. However, if any of the other virtual channels is ready and is assigned the link, the control flit specifying its identifier should be transmitted.

The old flow control protocol (described in Section 2.3) assigned physical bandwidth to virtual channels until the corresponding message blocks or is completely transmitted, while the new flow control protocol assigns physical bandwidth until the message blocks, has been completely transmitted or it has transmitted the maximum block size. With this new flow control protocol no virtual channel can monopolize the physical link. The old flow control protocol focused on minimizing control traffic. The new one focuses more on making a fairer use of the link.

Limiting the number of consecutive flits a virtual channel can transmit is easily implementable. Only a counter and a comparator per physical channel are needed. Each time a virtual channel gets the physical channel, the counter is reset. This counter is incremented with every data flit transmission. When the output of the counter is equal to the value set in the comparator, this virtual channel must grant the link to the next ready virtual channel (if any). If none of the other virtual channels are ready, it will resume transmission after resetting the counter.

4 Performance Evaluation

In this section, we compare the behavior of the new flow control protocol described in Section 3 with the one described in Section 2.3. We have considered several maximum block sizes. More specifically, block sizes of 8, 16, and 32 flits have been evaluated. We have labeled them as "b8", "b16", and "b32", respectively. Moreover, we have labeled the results obtained for the flow control protocol described in Section 2.3 as NB. We have obtained results for the adaptive routing algorithm described in Section 2.2 (referred to as MA2) and also for the up/down routing algorithm when used in a network with two virtual channels per physical channel, which will be referred to as UD2.

[1]We use the term "maximum block size" because once the entire message is transmitted, the virtual channel gives up transmission, even if it has sent fewer flits than the block size.

4.1 Network Model

We have used a simulator to evaluate the performance of the flow control protocols. The simulator models the network at the flit level using the evaluation methodology proposed in [4]. We have measured message latency, in clock cycles, and throughput, measured in flits per cycle per switch. We have also measured the standard deviation of the message latency (in clock cycles) and the channel utilization.

Message latency should also include software overhead at source and destination processors (system call, buffer allocation, buffer-to-buffer copies, etc.). This overhead has traditionally accounted for a high percentage of message latency [9, 7]. However, we did not consider such an overhead because our focus here is on the network hardware. Further, some recent proposals reduce and/or hide the software overhead [5, 8, 3].

In the switch model, we have assumed that it takes one clock cycle to compute the routing algorithm, one clock cycle to transmit one flit across the internal crossbar and that the switch needs one clock cycle to decode the control flits. Simulations have been carried out with propagation times along the physical channel of 4, 8, and 16 cycles, while data are injected into the physical channel, which is pipelined, at a rate of one flit per cycle. According to [13], the input buffer size of each virtual channel has been set to 27, 51, and 99 flits with fly times of 4, 8, and 16 cycles, respectively. With respect to the output buffer size, we have set it to 4 flits.

Each switch has a routing control unit that selects the output channel for a message as a function of its destination node, the input channel and the output channel status. Table look-up routing is used. The routing control unit can only process one message header at a time. It is assigned to waiting messages in a demand-slotted round-robin fashion. When a message gets the routing control unit but it cannot be routed because all the alternative output channels are busy, it must wait in the input buffer until its next turn. A crossbar inside the switch allows simultaneous multiple message traversal. It is configured by the routing control unit each time a successful route is established.

With respect to the network model, network topology has been generated randomly and is completely irregular. For the sake of simplicity, all the switches in the network have the same size. We have assumed 8-port switches. Four of the ports are used to connect to different processors and the other four ports are used to connect to other switches. Also, two neighboring switches are connected by a single link.

Message traffic and message length depend on the applications. For each simulation run, the message generation rate is kept constant and the same for all the nodes. We have evaluated the full range of traffic, from low load to saturation. Message destination is randomly chosen among all the nodes in the network. For message length, 16-flit, 64-flit, and 256-flit messages are considered. Also, mixtures of 16-flit and 256-flit messages with 80% of short messages and mixtures of 16-flit and 1024-flit messages, and 102-flit and 1024-flit messages with 50% of short messages have been evaluated. For network size, 16-switch (64 workstations), and 64-switch (256 workstations) networks are studied.

4.2 Simulation Results

Figure 2 shows some results for a network with 16 switches. Figure 2(a) shows the average message latency when 64-flit messages are used and the propagation time along the physical channel is four cycles. The routing algorithm used is MA2. As can be seen, the worst performance is obtained when the maximum block size is limited to 8 flits, while the NB protocol achieves the best performance for the whole load range. As the maximum block size increases, performance improves. We have run simulations varying the maximum allowed block size from 8 to 20,000 flits for 16 and 64-flit messages in networks with 16 and 64 switches and the NB protocol always gave the best results. When the maximum block size is very large, the performance is close to the NB protocol (the NB protocol behaves as the new flow control protocol when the maximum block size is infinite). The reason for this is the control traffic, that becomes more pronounced at small block size. As the maximum block size gets smaller, the control traffic due to control flits specifying the number of the virtual channel owning the link ("Select n" control flit) is more significant, as can be seen in Figure 2(c). This delays even more the arrival of messages. It should be noted that it is worth using a maximum block size larger than message size. The reason is that several consecutive messages can be transmitted using the same virtual channel without sending any control flit to select the virtual channel. Figure 2(b) shows the standard deviation of the message latency. Again, the NB protocol achieves the best standard deviation for the entire load range. Results when message length is 16 or 256 flits and when the propagation time is 8 or 16 cycles are similar to those presented in Figures 2(a), (b), and (c). When the routing algorithm is UD2 instead of MA2, results are also similar.

The results presented in Figure 2(a) are obtained with message length of 64 flits. As we explained in Section 3, the NB protocol behaves poorly when there are messages with different lengths in the network. Figure 2(d) shows the average message latency with an equal mixture of 16 flit and 1024 flit messages and the propagation time along the physical channel is four cycles. The routing algorithm used is the MA2. It can be observed that the NB protocol achieves again the lowest latency and the highest throughput. Again, as the maximum block size decreases, performance worsens. In Figure 2(e) one can see that the standard deviation of the message latency is also lower for the NB protocol. However, in this case we expected the NB protocol to present a worse performance than the new protocol. The reason for this disparity is again the overhead for control traffic. As can be seen in Figure 2(f), for the NB flow control protocol the channel utilization produced by control traffic is really low. As the maximum block size decreases, channel utilization derived from the control traffic increases. This increase does not compensate the improvement obtained from a fairer use of the link. Moreover, for medium to high network loads, with the NB protocol links are shared properly among the virtual channels multiplexed on them. Effectively, at very low loads, long messages can monopolize physical channels, since they may not block due to fewer messages in the network. However, as network load increases, the likelihood of long messages blocking increases, thus potentially granting the physical channel to the other virtual channel (if ready). At high loads, long messages are also likely to block, and short messages get the physical channel automatically. This is the best way to implement block multiplexing, since the traffic of the "Select n" control flits is minimized, making NB flow control protocol perform better.

For the rest of message length mixtures results are similar. Also, when fly time is 8 or 16 cycles instead of 4 cycles and when the UD2 routing algorithm is used instead of the MA2 routing algorithm, relative results are again similar.

Figure 3 shows the results for a network with 64 switches. Figure 3(a) displays the average message latency when 64-flit messages are used. Figure 3(b) shows the standard deviation of the message latency. In Figure 3(c), the channel utilization produced by control traffic is displayed. As can be seen, results are similar to those when network size was 16 switches. Figures 3(d), (e), and (f) show the results when 16 and 1024-flit messages are used. Results are similar, again, to those obtained with a network size of 16 switches.

Results displayed in Figures 2 and 3 show that when we limit the maximum block size, the overall network performance decreases and the standard deviation of the message latency increases, contrary to our initial assumption. Why? What happens with short and long messages individually? Figure 4 shows the performance for short and long messages separately. Figure 4(a) displays the average short message latency for a network with 16 switches when mes-

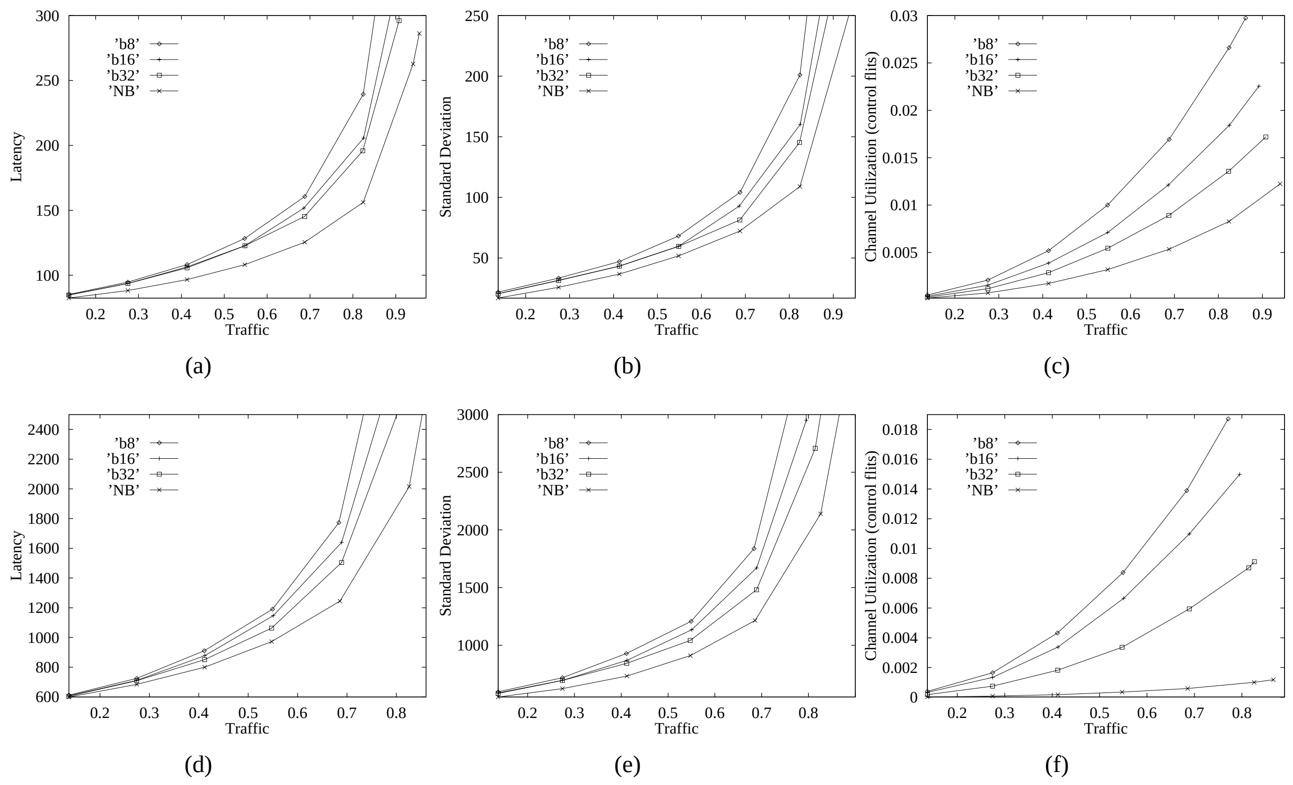

Figure 2. Simulation results for a 16-switch network. Fly time is 4 cycles. The MA2 routing algorithm is used. Message length is 64 flits in (a), (b), and (c), and 50% 16 flits 50% 1024 flits in (d), (e), and (f).

sage length is 50% 16 flits and 50% 1024 flits (the same simulation parameters as in Figure 2(d)). It can be seen that the NB protocol exhibits the highest latency for low load, an intermediate behavior for moderate load, and gives the lowest latency and highest throughput at high load. This result agrees with the discussion in Section 3. At low loads, long messages monopolize physical channels, because the probability for them to block is low since there are very few messages in the network. Therefore, when there is a conflict between a long and a short message, the short message must wait until the complete transmission of the long message finishes, unless it finds an alternative path. This waiting increases short message latency and its standard deviation (as shown in Figure 4(c)). Fortunately, at low loads short messages are not always delayed by long ones because adaptivity helps to find alternative paths. On the other hand, at high loads long messages are continuously blocking, and granting the physical channel to the next ready virtual channel in a way that minimizes the use of control flits. Thus, short messages can advance toward their destination without suffering from the overhead introduced by the "Select n" control flits. At moderate load, long messages block less frequently than at high load, but they block nevertheless, allowing short messages to progress. With respect to long messages, limiting the maximum block size produces a detrimental effect on throughput, as can be seen in Figure 4(b). This is due to the fact that they relinquish the physical channel frequently and send more "Select n" control flits than strictly necessary.

When network size is 64 switches, results are similar. Figure 4(d) displays the average short message latency when message length is 50% 16 flits and 50% 1024 flits (as in Figure 3(d)). In this case, differences in latency are more noticeable, both with short and long messages. The behavior of short messages is worse with the NB flow control protocol, while the behavior of long messages is much better. With respect to the standard deviation of the short message latency (in Figures 4(c) and 4(f)), note that in both cases, small and large networks, it is much greater than the message latency itself. Also, it is similar to the standard deviation of long message latency (not shown). This means that long messages dominate the performance of the network.

For the rest of simulations (80% 16-flit messages and 20% 256-flit messages, 50% 102-flit messages and 50% 1024 flit-messages, in both small and large networks with the UD2 and MA2 routing algorithms using the different propagation times), results are again similar. In general,

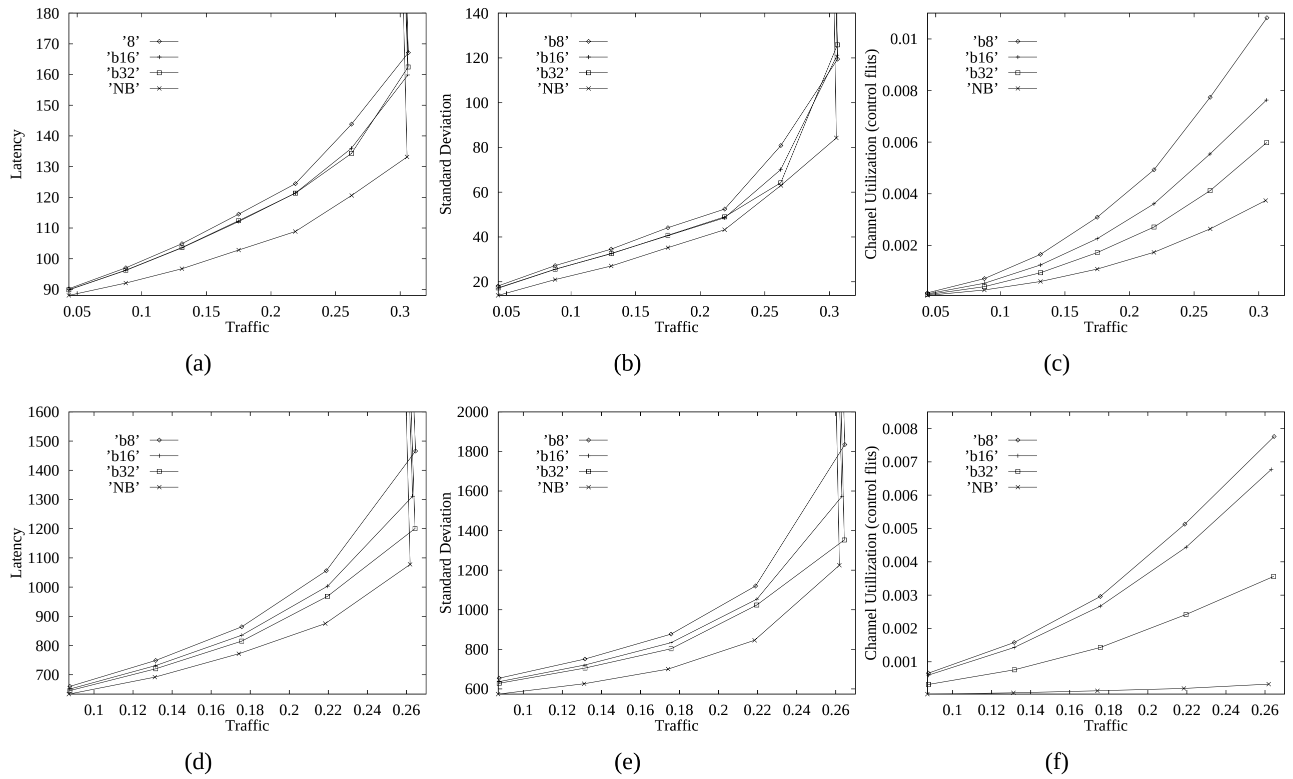

Figure 3. Simulation results for a 64-switch network. Fly time is 4 cycles. The MA2 routing algorithm is used. Message length is 64 flits in (a), (b), and (c), and 50% 16 flits 50% 1024 flits in (d), (e), and (f).

we can summarize that the improvement achieved for short messages by limiting the maximum block size does not compensate the additional overhead for long messages.

5 Conclusions

Networks of workstations are increasingly becoming more attractive because they are a cost-effective alternative for small-scale parallel computing. However, it is necessary that the underlying interconnect provides high bandwidth and low latency. In this paper we have taken a step towards this objective. Specifically, we have studied the impact of limiting the number of flits transmitted (block size) for a message on a virtual channel over a physical channel before switching to another virtual channel. To do so, we have proposed a new version of the flow control mechanism proposed by us in [13] that is easily implementable in hardware. This study has provided the following insights:

- If the maximum block size is not limited, long messages can monopolize physical channels at low loads. At high load, since long messages block, it is unnecessary to limit block size.
- In general, limiting the maximum block size decreases network performance. As the block size becomes smaller, message latency increases and throughput decreases. Also, the standard deviation of latency increases.
- Limiting the maximum block size increases control traffic because as the maximum block size decreases, the number of control flits required to select the virtual channel increases considerably.
- Limiting the block size in multiplexing virtual channels slightly favors short messages, while penalizing long messages.

In summary, limiting the maximum block size is not a good design decision, because the overall network performance decreases. Only when applications produce bimodal traffic and short message latency is crucial, would it be acceptable to limit the maximum block size.

References

[1] A. C. Arpaci-Dusseau et al., "High–performance sorting on networks of workstations," in *Proc. of ACM SIGMOD'97*,

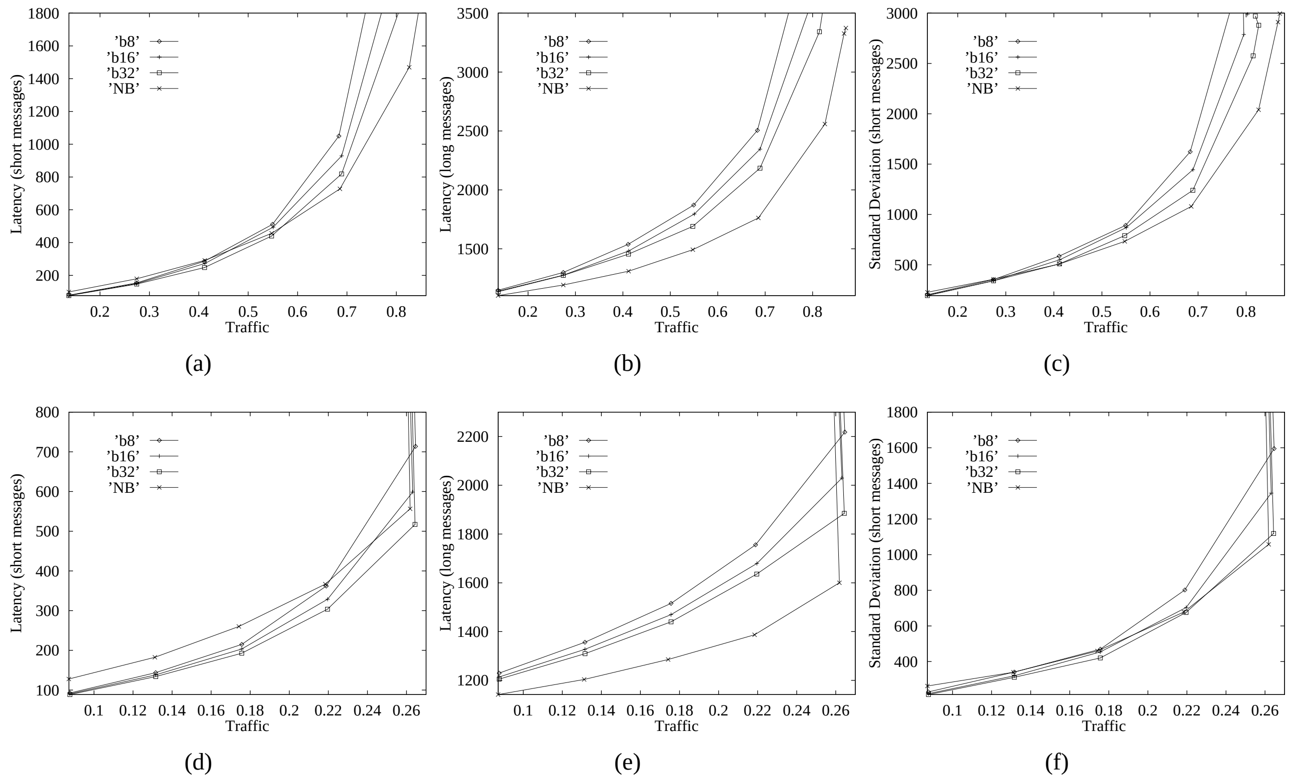

Figure 4. Results obtained for short and long messages separately. Message length is 50% 16 flits 50% 1024 flits. Fly time is 4 cycles. The MA2 routing algorithm is used. Network size is 16 switches in (a), (b), and (c) and 64 switches in (d), (e), and (f).

May 1997.

[2] N. J. Boden et al., "Myrinet - A gigabit per second local area network," in *IEEE Micro*, Feb. 1995.

[3] B. V. Dao, S. Yalamanchili and J. Duato, "Architectural support for reducing communication overhead in pipelined networks," *Third International Symposium on High Performance Computer Architecture*, February 1997.

[4] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Tran. on Par. and Dist. Syst.*, vol. 4, no. 12, pp. 1320–1331, Dec. 1993.

[5] T. von Eicken, et al., "Active messages: A mechanism for integrated communication and computation," in *Proc. of the 19th Int. Symp. on Computer Architecture*, June 1992.

[6] R. Horst, "ServerNet deadlock avoidance and fractahedral topologies," in *Proc. of the Int. Parallel Processing Symp.*, pp. 274–280, April 1996.

[7] J.-M. Hsu and P. Banerjee, "Performance measurement and trace driven simulation of parallel CAD and numeric applications on a hypercube multicomputer," *IEEE Trans. on Parallel and Dist. Systems*, vol. 3, no. 4, pp. 451–464, July 1992.

[8] V. Karamcheti and A. A. Chien, "Do faster routers imply faster communication?," in *Proceedings of the Workshop on Parallel Computer Routing and Communication*, May 1994.

[9] R. J. Littlefield, "Characterizing and tuning communications performance for real applications," in *Proceedings of the First Intel DELTA Applications Workshop*, February 1992.

[10] W. Qiao and L. M. Ni, "Adaptive routing in irregular networks using cut-through switches," in *Proc. of the 1996 Int. Conf. on Parallel Processing*, Aug. 1996.

[11] M. D. Schroeder et al., "Autonet: A high-speed, self-configuring local area network using point-to-point links," Technical Report SRC research report 59, DEC, April 1990.

[12] F. Silla and J. Duato, "Improving the efficiency of adaptive routing in networks with irregular topology," in *Proceedings of the 1997 Int. Conference on High Performance Computing*, December 1997.

[13] F. Silla and J. Duato, "On the use of virtual channels in networks of workstations with irregular topology," in *1997 Parallel Computing, Routing, and Communication Workshop*, June 1997.

[14] F. Silla et al., "Efficient adaptive routing in networks of workstations with irregular topology," in *Proc. of the Workshop on Comm. and Architectural Support for Network-based Parallel Computing*, Feb. 1997.