# SEQUENCING OF CONCURRENT COMMUNICATION TRAFFIC IN A MESH MULTICOMPUTER WITH VIRTUAL CHANNELS

Bing-rung Tsai and Kang G. Shin
Real-Time Computing Laboratory
Department of EECS, The University of Michigan
Ann Arbor, MI 48109-2122
Email: {iast,kgshin}@eecs.umich.edu

Abstract — *Under the fixed-path e-cube routing in mesh multicomputers, we evaluate the performance of several low-overhead packet sequencing and flit multiplexing methods. In the presence of concurrent inter-node communication traffic, we found that unless proper packet sequencing is employed, adding more communication resources, such as links and buffers, can actually degrade the network performance. A good packet-sequencing policy combined with proper flit multiplexing is shown to improve performance by more than 30%.*

## 1 Introduction

The use of virtual channels multiplexed over each physical channel was introduced as a mechanism to accomplish deadlock-freedom by placing routing restrictions at intermediate nodes [1]. Virtual channels were also found to improve the network throughput via the increased sharing of each physical channel and the resulting reduction of packet blocking [2]. That is, when there are multiple virtual channels per physical channel, packets of these virtual channels are allowed to time-multiplexed over the physical channel, thus blocking less number of packets (waiting for the physical channel to be available).

Pipelined-communication mechanisms, such as wormhole routing [1], operate based on the principle that the overall packet latency can be reduced by pipelining the transmission of each packet when the packet must traverse multiple intermediate nodes. A packet is broken up into small *flow-control digits* or *flits*, each of which serves as the basic unit of communication. The time taken for one flit to cross a physical channel is called the *flit time*. Header flits containing routing information establish a path through the network from the source to destination. Transmission of data flits is then pipelined through the path immediately following the header. Wormhole routing also has the advantage of requiring only a small on-line buffer space per node. While the pipelined nature of wormhole routing serves to reduce delivery latency, it may also propagate the effects of such bottlenecks as blocked flits and heavily-loaded physical channels. It is therefore important to devise a means of efficient allocation and management of network bandwidth.

The network under consideration employs wormhole routing. Each pair of adjacent nodes are connected by a pair of uni-directional physical links/channels. A fixed number of uni-directional virtual channels are time-multiplexed over each physical channel. Though most of our discussion may apply to general networks, we will focus primarily on the mesh network topology, which has been widely used in evaluating the performance of virtual-channel networks [2, 3]. Especially, this paper builds on the work by Dally [2] and Gaughan [4], where wormhole routing was found to significantly reduce packet latency if it is combined with appropriate bandwidth allocation and flow control schemes. We extend their work by focusing on bandwidth allocation through packet sequencing and flit multiplexing.

In the previous related work [2, 4], communication traffic in a multicomputer network is often modeled as a number of mutually-independent, steady flows. However, this type of communication traffic does not always represent the real-world situation well, because network communication tends to be bursty. Packet arrival times are often clustered in a short period, which can temporarily saturate the network. Also, these packets may not be independent, and their delivery time as a whole is crucial to the overall performance. This tendency is exemplified by such algorithms as parallel sorting [5] and parallel Fourier-

The work reported in this paper was supported in part by the Office of Naval Research under grants N00014-92-J-1080 and N00014-91-J-1115. Any opinions, findings, and recommendations in this paper are those of the authors and do not reflect the views of the ONR.

Transform [6].

In this paper, we define a *communication mission* to be a set of packets to be exchanged among the task modules which have already been assigned to processing nodes in the network. During the execution of a parallel program, inter-node communication behaviors can be viewed as several independent communication missions. In addition to the usual mean latency, the *makespan* of a mission will also be used for performance evaluation. The makespan of a mission is defined as the maximum latency of all packets in the mission, i.e., the time span from the arrival of the first packet until all the packets reach their destination.

The main intent of this paper is to (i) explore ways of sequencing packets and flits so as to better utilize network resources, and (ii) improve the overall network performance when more network resources are added. Especially, we will focus on the case when a substantial number of packets can be transmitted through the network *concurrently*. The paper is organized as follows. Basic terms and concepts necessary for our discussion are defined in Section 2. We formulate and analyze the problem in Section 3. Simulation results are presented and discussed in Section 4. This paper concludes with Section 5.

## 2 Preliminaries

A $k$-ary $n$-cube consists of $k^n$ nodes arranged in an $n$-dimensional grid. Each node is connected to its Cartesian neighbors in the grid. A 2-dimensional $k \times k$ flat mesh is a subgraph of $k$-ary 2-cube, is not a regular graph, and has less edges than the corresponding $k$-ary 2-cubes (no wrap links at its boundary nodes). For convenience, we will call a $k$-ary 2-cube a *wrapped mesh*, or a *w-mesh* for short. Likewise, we will call a 2-dimensional flat mesh an *f-mesh*. Since an f-mesh is a subgraph of w-mesh with the same number of nodes, a w-mesh can also be made to function as an f-mesh by not using its wrap links.

Flow control in a virtual-channel network is performed at three levels: routing, packet sequencing, and flit multiplexing. Each of these can be implemented with a variety of algorithms, but we will consider only low-complexity, low-overhead flow-control mechanisms to deal with concurrent traffic in the network.

Routing: Selection of a path for each packet. A packet is routed to its destination via a fixed, shortest path. Issues related to fault-tolerance are not considered, or physical and virtual channels are assumed to be fault-free. In f-meshes, e-cube routing is used. In w-meshes, a modified version of e-cube routing is implemented to utilize the extra communication links so that each packet is routed via a shortest path. Deadlock-freedom is ensured by using the scheme proposed in [1]. That is, the virtual channels over each uni-directional physical channel are divided into high and low channels. Routing restrictions are then imposed such that either a high channel or a low channel, but not both, is allocated to each given packet. The w-meshes need at least two virtual channels per physical channel to achieve deadlock-freedom.

Packet Sequencing: Determining which packet is allowed to access a free virtual channel in case of contention. When the number of packets to access a physical channel at the same time is larger than the number of available virtual channels, some of these packets have to be queued. So, we need to determine which packets are allowed to access the virtual channels, and which packets to be queued. We will consider the FIFO policy (as default), the *largest remaining bandwidth first* (LF) policy, and the *smallest remaining bandwidth first* (SF) policy. The *remaining bandwidth* of a packet is defined as the product of packet length and the distance from the current node to its destination. SF and LF can both be easily implemented by using a priority queue instead of an FIFO queue.

Flit Multiplexing: When there are multiple virtual channels per physical channel, the packets allocated to these virtual channels are multiplexed over the physical channel. Flit multiplexing determines the order for these flits from different virtual channels to access the physical channel.

In the default, *round-robin*(RR), multiplexing, virtual channels take turns in accessing the physical channel without using any network or packet information. RR multiplexing without any modification will henceforth be called *strict* RR. Like packet sequencing, flit multiplexing can be priority-based. LP multiplexing awards priority to the virtual channel containing a packet of larger remaining bandwidth requirement. By contrast, SP multiplexing gives priority to the one of smaller remaining bandwidth requirement. As pointed out in [2], these multiplexing methods can all be implemented with combinational logic which operates on the contents of the status register associated with each virtual channel. The added hardware cost should not be a concern if the number of virtual channels is not too excessive.

If each virtual channel is allocated a fixed physical bandwidth regardless whatever the virtual channel is in use or not, this can lead to a substantial waste of physical bandwidth. *Demand-driven*(DD) allocation can be used to rectify this problem. With DD allocation, virtual channels will contend for use of a physical channel only if they have flits to send. DD allocation can be easily implemented by adding low-complexity combinational circuit to any multiplexing method.

With *CTS (Clear-To-Send) lookahead*, virtual channels only contend for use of a physical channel if each of them has a flit to send *and* the receiving node has room for it. This can further reduce the waste of physical bandwidth. When CTS lookahead is implemented, the receiving-end of each virtual channel must send a status bit back to the sending-end. These signals can be sent via separate wires [1], which requires extra hardware. Or they can be sent over the physical channel in the opposite direction, which can result in a non-negligible bandwidth overhead.

## 3 Formulation and Analysis

In this section, we discuss the tradeoffs among different packet-sequencing policies and flit-multiplexing methods under the following assumptions.

- A physical channel takes one unit of time to transmit a single flit. This unit of time will also be called a physical-channel *cycle*.
- There is a single-flit buffer associated with each virtual channel.
- A packet arriving at its destination is consumed immediately without waiting.
- There are an even number of virtual channels associated with each physical channel in a w-mesh.

The *latency* of a packet $p_{ij}$ from node $i$ to node $j$, denoted as $t_{ij}$, is the time span from a packet's arrival to acceptance of the last flit of the packet by its destination. We will use $\bar{t}$ to represent the mean latency of a mission. The *makespan*, denoted as $\hat{t}$, of a communication mission is the maximum latency of all packets in the mission. We will evaluate the performance of a network with both the mean mission latency and makespan. Then,

$$t_{ij} = t^0_{ij} + (1/r_{ij})(\ell_{ij} - 1).$$

The first term, $t^0_{ij}$, denotes the time span between the arrival of $p_{ij}$ at the source node $i$ and the arrival of its header flit at the destination node $j$. $t^0_{ij}$ is composed of two components: accumulated queueing delay $t^q_{ij}$ and accumulated head flit-multiplexing delay $t^x_{ij}$. $t^q_{ij}$ is the sum of queueing times at all nodes in the path for available virtual channels. $t^x_{ij}$ is the sum of times $p_{ij}$'s header flit waits at all nodes on its path for use of physical channels. The second term, $(1/r_{ij})(\ell_{ij} - 1)$, represents the time required for all other flits of $p_{ij}$ to reach node $j$, which is determined by the length of $p_{ij}$, denoted as $\ell_{ij}$, and the transmission rate, $r_{ij}$, of the pipeline setup for $p_{ij}$. Depending on the flit-multiplexing method used, $r_{ij}$ may change with time during a mission.

### Packet Sequencing

Given a communication mission and a fixed number of virtual channels, $t^q$ will be affected by the underlying packet-sequencing scheme. Under the LF policy, packets requiring larger bandwidths are given priority. Since those packets farther away from their destinations are more likely to have larger $t^x$'s, by minimizing their $t^q$'s, we may minimize the variance of packet latencies. Similarly, the second term of $t_{ij}$ is larger for longer packets. By giving these packets higher priority in using virtual channels, the balancing effect of smaller $t^q$'s and hence smaller $t^0$'s can also minimize the variance of packet latencies. However, in wormhole routing, a blocked packet does not release resources already allocated to it. A packet farther away from its destination is more likely to be blocked and may therefore result in more resources being held. Also, a longer packet can hold up resources in the path for a longer time, thus blocking more of the other packets. Under the SF policy, packets requiring smaller bandwidths are given higher priority. This heuristic is proposed based on the conjecture that if these packets are delivered to their destinations quickly then there will be less packets contending for communication resources and the overall concurrent communication traffic can be reduced, and thus, the remaining packets may be sent through the network encountering less contention.

### Flit Multiplexing

Generally, $t^q$ decreases with the increase of the number of virtual channels per physical link, denoted by $v$. However, under strict RR multiplexing without

DD allocation or CTS lookahead, $r_{ij} = 1/v \; \forall i, j$. So, $1/r_{ij}$ increases with $v$ and $t^x$ may also increase because a flit may need to wait more cycles for use of the physical channel. That is, there is a tradeoff between $t^q$ and $r_{ij}$, and also between $t^q$ and $t^x$, when more virtual channels are added. Usually when $v$ is increased to a certain point, the improvement in reducing $t^q$ reaches a plateau, and adding more virtual channels only increases packet latency.

In case of strict RR, a large portion of physical bandwidth can be wasted on idle virtual channels. With DD allocation (denoted as DD-RR), $r_{ij}$ is bounded below by $\max\{1/v, 1/k_{ij}\}$, where $k_{ij}$ is the degree of *congestion* of the path, i.e., the maximal number of packets to share a physical link in the path of $p_{ij}$. In the worst case, $r_{ij} = 1/v$ and latency $t_{ij}$ is the same as in the case of strict RR. But, if at any instant there are less than $v$ packets contending for a physical channel, then no physical channel will be allocated to any idle virtual channel. Packets can be transmitted at a rate $\geq 1/v$.

Even with DD allocation, physical bandwidth can still be wasted if the corresponding input buffer is not ready to receive a new flit. With CTS lookahead, a virtual channel will not contend for a physical channel unless it has a flit ready to be sent and the receiving end has room for accepting it. Therefore, for a given multiplexing method, ($r_{ij}$ |with neither DD nor CTS) $\leq$ ($r_{ij}$ |with DD) $\leq$ ($r_{ij}$ | with DD and CTS).

Another advantage of CTS is deadlock-freedom. In priority-based flit multiplexing, a deadlock may occur if the method is not carefully implemented. For example, in Fig. 1, each physical channel has two virtual channels. All packets are routed in the same direction on node 2. Packet A and B have established pipelines from node 0 to node 2, and node 1 to node 2, respectively. So on node 2, the two packets have occupied both of the output buffers in the direction which all packets need to be routed to. Suppose packet C arrives at node 0 later than A, and occupies the other virtual channel, and the same situation occurs on node 1 when packet D arrives after B. If C is given priority over A, then C will access the physical channel between node 0 and 2. Similarly, D also has higher priority than B, and monopolizes the physical channel between node 1 and 2. But on node 2, both of the output buffers are already occupied by packet A and B, and they cannot be preempted since wormhole routing is used. Therefore, C and D will be queued at node 2 indefinitely waiting for free output buffers, while A and B cannot access the physical channels which they need to finish sending their remaining flits and, release the output buffers on node 2 that C and D are waiting for. So, a deadlock follows.

If CTS lookahead is used in the above example, then one can avoid the deadlock. Since packet C and D will not be allowed to contend for physical channels, A and B will be able to continue sending their remaining flits. In this paper, we will always evaluate LP and SP flit multiplexing with CTS lookahead to avoid deadlock.

W- and F- meshes

When compared to an f-mesh with the same number of nodes, a w-mesh has the advantages of more physical channels and smaller communication diameter. Also, its regular connectivity may lead to better communication load balancing, especially in the case of uniformly-distributed traffic.

However, one main drawback of w-meshes is the potential deadlock resulting from the addition of wrap links. To ensure deadlock-freedom, virtual channels running over each physical channel must be divided into two halves. When packets are sent between a pair of nodes between which the Hamming distance is $< k/2$ in a $k$-ary 2-cube, only $v/2$ virtual channels are available. If the other $v/2$ virtual channels are not in use, they are left idle and their bandwidth wasted. Thus, there exists a tradeoff between these two topologies. Depending on traffic density and distribution, one can outperform the other. Our simulation results in the next section demonstrate this tradeoff.

# 4 Simulation Results

Under the following assumptions, we developed a program that simulates the flit-level communication behavior.

- Transferring a flit between two nodes via a physical channel takes one unit of time.
- At any instant of time, all flits that have been allocated channels are transferred synchronously in a single physical channel cycle.
- Each virtual channel is assigned a single-flit buffer.
- Traffic is uniformly-distributed. For a given mission, the probability that node $i$ may send a packet to node $j$ is fixed.

| | $\hat{t}$ | | $\bar{t}$ | |
|---|---|---|---|---|
| $v$ | w-mesh | f-mesh | w-mesh | f-mesh |
| 1 | n/a | 801 | n/a | 280 |
| 2 | 1389 | 652 | 470 | 228 |
| 4 | 1090 | 546 | 345 | 209 |
| 6 | 945 | 508 | 305 | 223 |
| 8 | 812 | 502 | 305 | 251 |
| 12 | 707 | 533 | 341 | 304 |
| 16 | 688 | 629 | 377 | 414 |

Table 1: Performance with strict RR multiplexing.

| | $\hat{t}$ | | $\bar{t}$ | |
|---|---|---|---|---|
| $v$ | w-mesh | f-mesh | w-mesh | f-mesh |
| 1 | n/a | 801 | n/a | 280 |
| 2 | 889 | 616 | 316 | 216 |
| 4 | 635 | 507 | 200 | 188 |
| 6 | 516 | 457 | 162 | 189 |
| 8 | 401 | 432 | 147 | 198 |
| 12 | 310 | 418 | 144 | 216 |
| 16 | 290 | 410 | 151 | 229 |

Table 2: Performance with DD-RR multiplexing.

In the discussion that follows, the term *configuration* is used to represent a combination of certain packet-sequencing policy with a flit-multiplexing method. The simulation results presented here were obtained using the following parameters:

- Unless stated otherwise, all packets are 20 flits long.
- Both w- and f- meshes are of size $16 \times 16$.
- The probability, *density*, that node $i$ sends a packet to node $j$ is 0.01. In a $16 \times 16$ network, the total number of concurrent packets during a mission is $\approx 0.01 \cdot (16^2 - 1)^2$.
- Each data point is obtained by averaging results from 10,000 iterations. Deviation from the mean values is found to be ($< 5\%$). Due to the page limit, results on inputs with variable packet lengths and other traffic distribution are not presented. However, general trends of the results obtained from these alternative inputs do not deviate significantly from the data shown.

Table 1 shows the makespan ($\hat{t}$) and the mean latency ($\bar{t}$) of w- and f- meshes with FIFO packet sequencing and strict RR flit multiplexing. Clearly, with strict RR, w-meshes not only perform worse than f-meshes with the same $v$, but also worse than f-meshes with $v/2$ virtual channels. Thus, addition of physical and virtual channels in w-meshes actually degrades the performance if strict RR is used.

Table 2 shows the case of DD allocation. Obviously, DD allocation greatly improves the performance, particularly in the case of w-meshes with larger $v$'s. In a certain situation, $\hat{t}$ and $\bar{t}$ are reduced by more than 50%. For f-meshes, DD also makes a monotonic improvement of $\hat{t}$ with the increase of $v$. Note that with DD-RR, w-meshes start to have smaller makespans than f-meshes with the same $v \geq 8$. Nevertheless, in the case where both types of network have equal number of nodes, we have to take into account that w-meshes have more physical channels. In a network of $16 \times 16$ nodes, a w-mesh has 1024 uni-directional physical channels while an f-mesh has only 960. Considering this, w-meshes still have poorer physical channel utilization, even with $v \geq 12$.

From the results in Tables 1 and 2, we conclude that physical bandwidth is used much more efficiently with DD-RR than strict RR. Since DD allocation can be implemented with minimum hardware overhead over any configuration, all configurations will be evaluated with DD allocation. DD-RR with FIFO will be used as our "default" configuration. The data in Table 2 will be used as the reference for other configurations. The makespan and the mean latency of each configuration are plotted.

Fig. 2 shows the comparison of the LF and SF packet-sequencing policies with FIFO in w- and f-meshes. DD-RR multiplexing is assumed in all three configurations. In w-meshes, SF shows a significant improvement ($> 30\%$) over FIFO for $v = 2$. The margin of improvement decreases with larger $v$'s, which drops below 10% when $v > 8$, and reduces to near 0% after $v > 12$. The sharp drop after $v > 6$ can be attributed to the fact that, when $v > 6$, adding virtual channels improves FIFO, and makes the effect of packet sequencing less significant. On the other hand, LF does not make any notable improvement over FIFO. Only when $v = 2$ it shows $\approx 8\%$ improvement.

In f-meshes, SF shows 10% to 17% improvements over FIFO for $v = 1$ and $v = 2$, and quickly drops to the same with FIFO when $v \geq 6$, while LF is only marginally effective when $v = 1$ ($\approx 7\%$ improvement) and is virtually the same with FIFO for any $v \geq 4$.

Fig. 3 compares the mean latency, $\bar{t}$, of the three packet-sequencing policies. In w-meshes, SF reduced the mean latency by more than 30% for $v = 2$ but the margin reduces gradually down to less than 10% when $v = 6$ and later drops to near 0% after $v \geq 8$. Similarly, in f-meshes, SF is effective for a small number of virtual channels($v = 1$ and $v = 2$), reducing $\bar{t}$ by at least 12%. But it performs virtually the same as FIFO for $v \geq 4$. In both the topologies, LF makes virtually no improvement over FIFO in terms of $\bar{t}$ at any value of $v$ and is not shown in the plot.

CTS lookahead can effectively minimize the waste of physical channel cycles, but its higher implementation cost may not be justifiable if the margin of improvement is small. In [4], it is shown that with pipelined circuit-switching, CTS is not very effective. As for wormhole routing, it was shown in [2] that in a network with 32-bit flits and $v = 15$, without adding extra wires for the lookahead signals, an additional 12.5% traffic overhead is required to implement CTS lookahead. Therefore, CTS lookahead should provide at least 12.5% improvement to justify its implementation overhead.

The effects of adding CTS lookahead on the three packet-sequencing policies are plotted in Figs. 4 and 5. CTS lookahead is very effective in w-meshes, reducing $\hat{t}$ over the corresponding non-CTS lookahead version for 10% to 30% when $2 \leq v \leq 12$, and $\bar{t}$ for at least 15% when $2 \leq v \leq 8$.

CTS lookahead is less effective in f-meshes, however. The greatest improvement ($\approx 10\%$) in $\hat{t}$ over non-CTS versions occurred when $v = 4$. The improvement margin gradually decreases down to 0% when $v$ is increased to 16. It reduces $\bar{t}$ by at most 10% when $v = 2$ or $v = 4$.

The effects of SP and LP multiplexing (with CTS lookahead to avoid deadlocks) are plotted in Figs. 6–9. In w-meshes, as compared to the corresponding configurations with CTS lookahead only, the SP version further reduces $\hat{t}$ by $\approx 10\%$ when $4 \leq v \leq 8$. But the margin of improvement gradually decreases when $v$ reaches 12, and actually has worse performance for larger $v$'s. In f-meshes, SP multiplexing improves the CTS-only version slightly. The maximum margin of improvement occurs at $v = 4$.

SP multiplexing is very effective in reducing $\bar{t}$. As can be seen in Fig. 7, for $2 \leq v \leq 32$, this multiplexing method reduces $\bar{t}$ significantly for FIFO, and clearly outperforms the CTS-only counterpart for larger $v$'s, i.e., $v > 4$ in w-meshes and $v \geq 4$ in f-meshes. The performance in $\bar{t}$ of LF and SF policies with SP multiplexing is very close to FIFO-SP and hence is not plotted.

LP multiplexing is less effective overall than SP: it has virtually no improvement in $\hat{t}$ over the CTS-only counterpart. But it still reduces $\bar{t}$ effectively when $v > 8$ for both w- and f- meshes, though not to the same extent of SP multiplexing.

From the data discussed above for SP and LP multiplexing, one can conclude that they are quite effective in reducing the variance of packet latencies, and the makespan is sacrificed somewhat for large $v$'s.

We also ran simulations for the general case where packet length is not uniform. Results are found to be consistent with the uniform packet-length case, and bandwidth-sensitive packet sequencing and flit-multiplexing methods are found to be more effective when the variance of packet length is increased.

The simulation results discussed thus far are summarized as follows.

- SF packet sequencing outperforms LF in almost all situations. This is surprising since in [7], LF sequencing is shown to be much more effective than SF in a network with large-buffer switching methods like store-and-forward and virtual cut-through. We can thus conclude that in a wormhole routing network, resource management should be quite different from large-buffer switching networks.
- Demand-driven allocation and CTS lookahead are extremely effective in reducing the waste of physical bandwidth, especially when the number of virtual channels is large.
- If reducing the mean latency is the main goal, then priority-based multiplexing is most effective. Especially, in the case of f-meshes with a large number of virtual channels, no other packet-sequencing policy or flit-multiplexing methods can stop the trend of increasing $\bar{t}$'s with larger $v$'s. With SP multiplexing, $\hat{t}$ is also reduced when $v$ is small.
- When $v = 1$ or $v = 2$, f-meshes should be considered a better topology than w-meshes. For the case of $v = 2$, f-meshes outperform w-meshes in both $\hat{t}$ and $\bar{t}$ with less resources. Moreover, w-meshes cannot function with $v = 1$ unless they are used as f-meshes.
- Reducing the makespan of a mission does not necessarily reduce $\bar{t}$, and vice versa. Network configurations should be evaluated carefully with both measures before making any conclusion on their performance.

## 5 Conclusion

We have evaluated the performance of several packet-sequencing policies and flit-multiplexing methods in a mesh network with wormhole routing and virtual channels. We focused on w-meshes ($k$-ary 2-cubes) and f-meshes($k \times k$ meshes) with $e$-cube routing under concurrent traffic. We considered primarily low-complexity flow control mechanisms. Simulations have been performed for three packet-sequencing policies, FIFO, SF and LF, and their combinations with flit-multiplexing methods such as demand-driven (DD) allocation, CTS lookahead, and priority-based multiplexing.

We used two performance measures in evaluating these configurations: $\hat{t}$, the makespan of a communication mission, and $\bar{t}$, the mean latency. It was found that DD allocation and CTS lookahead are both essential to minimize the waste of physical bandwidth. With a small amount of extra hardware, SF packet sequencing and the SP flit multiplexing can improve network performance significantly. Also, w-meshes, though with more communication resources, may perform worse than f-meshes in certain situations.

## References


[1] W. J. Dally, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. on Computers*, vol. C-36, no. 5, pp. 547–553, May 1987.

[2] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, March 1992.

[3] W. J. Dally, "Performance analysis of $k$-ary $n$-cube interconnection networks," *IEEE Trans. on Computers*, vol. 39, no. 6, pp. 775–785, June 1990.

[4] P. T. Gaughan and S. Yalamanchili, "Analytical models of bandwidth allocation in pipelined $k$-ary $n$-cubes," submitted to publication, 1993.

[5] T. Tang, "Parallel sorting on the hypercube concurrent processor," in *Proc. of the 5th Distributed Memory Computing Conference*, pp. 237–240, April 1990.

[6] L. Desbat and D. Trystram, "Implementing the discrete Fourier Transform on a hypercube vector-parallel computer," in *Proc. of the 4th Distributed Memory Computing Conference*, pp. 407–410, March 1989.

[7] B. R. Tsai and K. G. Shin, "Combined routing and scheduling of concurrent communication traffic in hypercube multicomputers," submitted to publication, 1993.


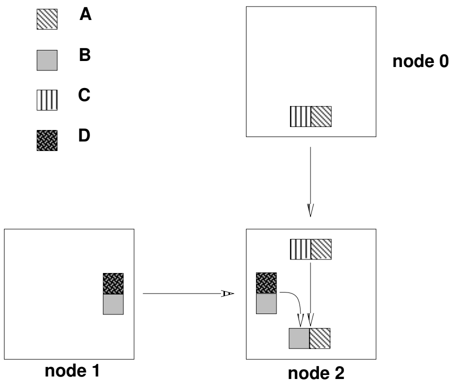


Figure 1: A deadlock caused by flit multiplexing.

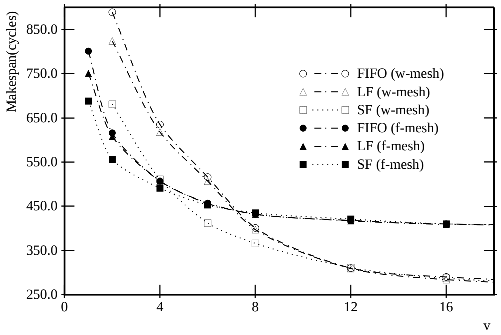


Figure 2: Makespan comparison.

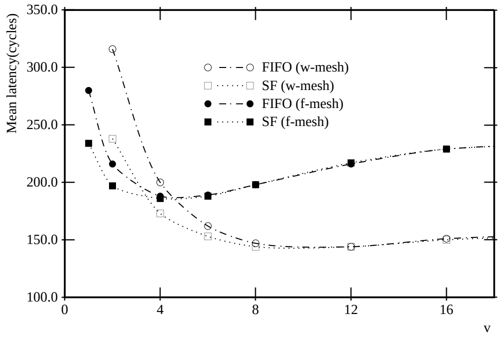


Figure 3: Mean latency comparison.

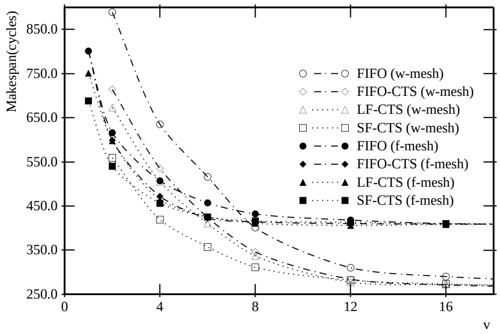


Figure 4: Makespan comparison.

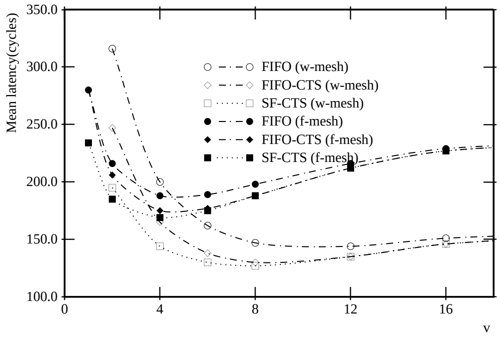


Figure 5: Mean latency comparison.

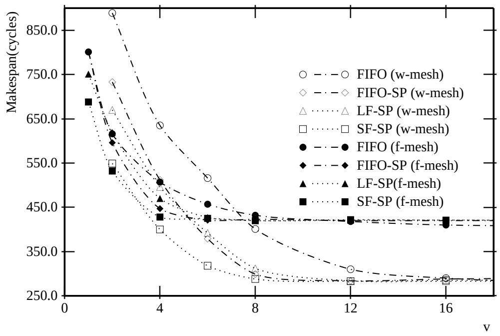


Figure 6: Makespan comparison.

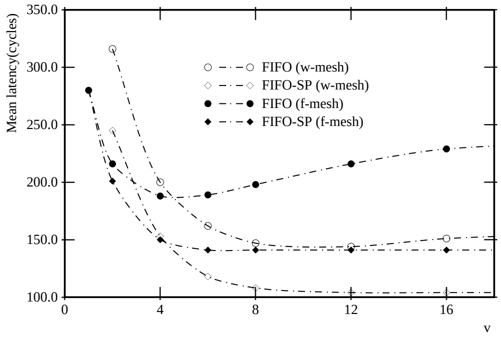


Figure 7: Mean latency comparison.

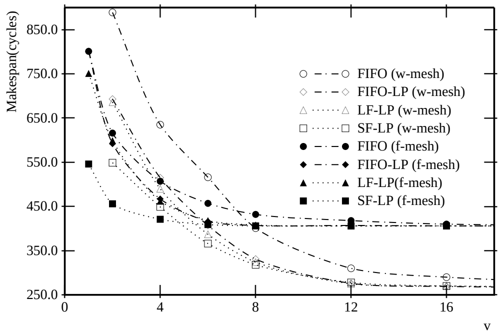


Figure 8: Makespan comparison.

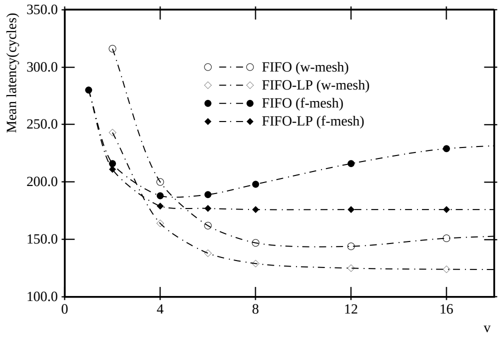


Figure 9: Mean latency comparison.