# A Flit Level Simulator for Wormhole Routing

Denvil Smith
Mississippi College
denvils@arch.com

## Abstract

Wormhole routing, the latest switching technique to be utilized by massively parallel computers, enjoys the distinct advantage of a low latency when compared to other switching techniques. More simulation tools would prove beneficial to the communication research community to aid in evaluating wormhole routed algorithms. These tools should reveal the optimum use of the network resources, which will allow for new algorithms to be compared against previously proven algorithms. Some of these resources include topology, buffers, virtual channels, and message size. The contribution of this research is a simulator that simulates network activity for three selected wormhole routed algorithms. The simulator provides an optimal network setting for each of the three simulated algorithms. The simulator will allow for settings other than reported in the literature. Thus allowing for new results for each algorithm to be achieved. Experimental results with various settings will be shown, to coincide with previous observations that will substantiate the integrity of the simulator. These results will then be used to deduce an optimal setting for each algorithm. These settings differ from previous reports in the literature for each algorithm.

## 1 Introduction

Since the inception of computers, the need has continually arisen for more and more processing power along with more storage space. This leads to joining multiple processors together so a group of processors may act like a single processor. Super computers address this need in the form of a parallel architecture or massively parallel computer, **MPC**. The communication subsystem for these processors is known as the **interconnection network** [7]. A **node** is referred to as a computing device on an interconnection network. Figure 1 illustrates the typical configuration for a node on an interconnection network. Each node contains its own processor, memory, and router. A **router** handles the communication functions. These nodes communicate with each other by passing messages. Therefore the performance of the parallel system is dependent on the interconnection network.



A **direct network** is the most popular classification of interconnection networks because it scales well. A set of nodes is connected to a subset of other nodes in this form of network [7][10]. **Switching** refers to the manner in

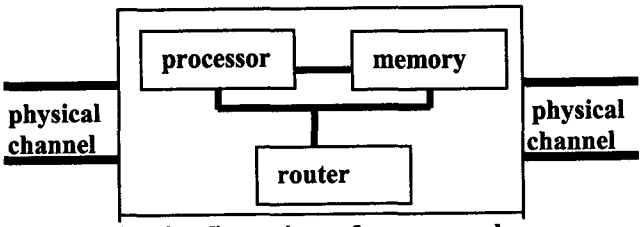



**Figure 1 Configuration for a node on an interconnection network**

which data are routed through an interconnection network of nodes. One of the main reasons for switching is that not all the nodes have a direct connection from the source to the destination node. Various switching techniques exist for data routing, such as **circuit switching, packet switching** which is also called **store and forward (SAF), virtual cut-through**, and **wormhole routing**. These techniques differ in the relationship between the size of physical unit of data and the size of the message flow control unit [7][8][17].

Wormhole routing splits a message into smaller units than packets, which are called **flits**. Each message contains one header flit which carries the routing and control information and the remaining data for the message is stored in the trailer flits. The header flit always goes first to allocate a path for the trailer flits. If the header flit encounters a wait, the message is not stored in the current node's memory but blocked in place on the network. Thus, smaller memory requirements exist for each node on the network. For this to occur small, fast, and compact routers had to be developed to allow for the flits of a message to remain on the network. Therefore, wormhole routing does not have to incur the delay of storing and retrieving the components of a message when an output channel is not available. If an output channel is available, the header flit is routed and the remaining trailer flits follow in a pipeline style fashion. During any instance of a message traversing a network, the flits of a message will be located in multiple routers. Figure 2 illustrates how these flits from a single message may be located in multiple nodes on a network. A positive attribute of this switching technique is the nearly distance insensitive trait it exhibits for the latency [7][10].

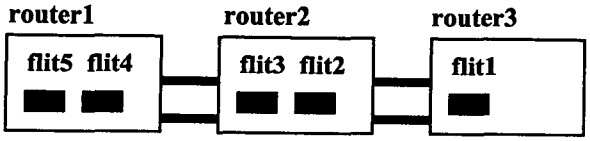



**Figure 2 Flits from a message in multiple routers**

# 2 Wormhole Routing as a Switching Technique

The main metric utilized to evaluate the performance of communication networks is **communication latency** [1][3][4][7][12][13][14][16]. This time begins when a message enters the network and ends once the message is received at the destination node. One of the most important factors to determine the communication latency is the switching technique employed by the network. The most popular switching technique today for a MPC is wormhole routing [10]. Some examples of commercial systems that utilize wormhole routing are the Cray T3D, Ametek 2010, and the J-machine, which was constructed at the Massachusetts Institute of Technology [7][10][11].

The resources in a wormhole switched direct network are the physical channels and buffer space [10]. For the header flit to be passed to the next node from the current node, it has to acquire both a channel and buffer space for the next node. Once the resources are acquired, they are not relinquished until the final flit of the message has exited a node. This fact leads to one of the challenges of wormhole routing, the possibility for deadlock. A **deadlock** occurs when a set of processes has resources allocated and the processes will never complete due to other processes having the remaining resources allocated [18]. Deadlock is avoided in wormhole routed networks by virtual channels and the routing algorithm used to route the flits through the network [7][11]. **Virtual channels** allow a single physical channel to be utilized by more than one message [6]. A good routing algorithm must allocate these virtual channels and buffer space in a manner that relieves the possible contention between multiple messages for these resources [7][11]. Virtual channels allows several messages to gain access to the physical channel which allows for blocked messages to be passed by non-blocked messages that have access to the necessary resources [10]. Virtual channels also allow for the network to be logically disjointed into multiple subnetworks. A disadvantage of virtual channels is the routing complexity that is added to each router. After a break-even point, additional virtual channels will only add to the latency [6][7].

A topology may be classified by the number of neighbors for each node. A popular classification of direct networks is an n-dimensional mesh. A node in a **mesh** network has either n or 2n neighbors depending on its position in the network. A **torus** is a special instance of a mesh that features a wrap-around strategy. Figure 3 illustrates a 3X3 mesh. A torus has both top to bottom along with right to left side wraps. As an example in figure 3, nodes one, two, and three have direct links to nodes seven, eight, and nine respectively. And, nodes one, four, and seven have direct links to nodes three, six, and nine respectively.

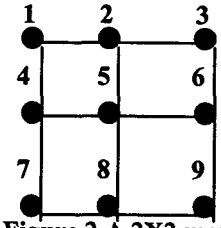



**Figure 3 A 3X3 mesh with numbered nodes**

Topologies may be compared on the basis of their bisection width. **Bisection width** is the minimum number of channels that must be removed from a network to partition the original network into two equal subnetworks [7][10]. Figure 4 illustrates the bisection of a mesh network by removing three links. The heavy dark line represents the bisection of the network.

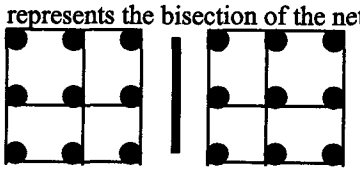


**Figure 4 Bisection of a mesh network**

Routing algorithms determine the path taken by a message while traversing the network. These algorithms may be classified by various levels. **Unicast routing** occurs when a message only has one destination and **multicast routing** is when a single message may be delivered to multiple addresses.

**Dimension ordered routing** is a classification of routing algorithms that dictates all paths in one dimension must be traversed before paths in another dimension are visited. **Adaptivity** is an algorithm's ability to route messages along other paths which are dependent on network conditions. **Deterministic algorithms** will always yield the same output channel regardless of the network activity. **Oblivious algorithms** will make a random selection to determine the output channel when a choice of output channel is available. **Minimal routing algorithms** will supply paths that will carry the message closer to the intended destination. **Nonminimal algorithms** may supply paths that take the message further away from the intended destination in the presence of faults or heavy traffic. A **fully adaptive minimal** algorithm may select from all the available minimal paths based on current network conditions [7][10].

In the past, most wormhole routed algorithms were developed as deterministic and unicast based. The current trend is to develop adaptive algorithms that support multicast message passing techniques. More tools are

needed that will allow for the study of different network settings for wormhole routed networks. These settings include the number of virtual channels, the size and number of flit buffers, the topology, and message size. These tools should also address the trade off when additional virtual channels become prohibitive due to greater communication latency [7][10][11].

# 3 Research Goals

The primary contribution of this research is a simulator that models selected wormhole routed algorithms and through the use of various input parameters, yields an optimum network setting for each algorithm. The simulator allows for other network configurations than reported in the literature, thus achieving new results. The simulator was written in C++. Other simulators have been developed and used to achieve simulated results for wormhole routing [1][3][4][7][12][13][14][16]. The input parameters simulate resources on the network. Some of the parameters include topology, message size in flits, number of buffers and size of the buffers in flits, number of virtual channels, the percentage of faulty nodes, and the number of messages. A structure variable is the central processing element for the simulator. The **structure variable** is used in C to group fields into a record structure [9]. This structure variable keeps track of the data as it pertains to each flit on the network during the simulation. Flits are simulated traversing the network by updating the value of these variables. Some of these variables include the message and flit identification number, address information, the number of wait cycles, and the current number of hops. The wait cycles variable is incremented by one when a lock on a resource is unsuccessful. The latency is a function of the number of hops and the wait cycles incurred by all the messages for a given simulation. The simulator injects messages and determines message destinations in a random fashion. The simulator also allows for a manual injection of messages to simulate hotspots.

# 4 Selected Wormhole Routed Algorithms

All wormhole routed algorithms address the two global issues facing this switching technique, deadlock and flow control. The manner in which each algorithm addresses these issues are the differentiating factor between the algorithms. In other words, each algorithm differs in how the flits are routed through the network and how resources are allocated to a message [10][11]. The objective of this study is to develop a flit level simulator that simulates the network activity of selected wormhole routed algorithms. The simulator accepts input parameters, which are the resources for a wormhole routed network. Some of the output parameters of the simulator include latency, virtual channel utilization, and bisection utilization. The range of input parameters are constructed such that various network settings may be utilized for experimentation purposes to arrive at an optimal configuration for each simulated algorithm. The simulator has provided some previously established results for wormhole networks. These results will be displayed as evidence of the integrity of the simulator. And, these results will also be used to determine the most efficient use of the network resources to provide an optimal setting for each simulated algorithm. A survey of the literature was conducted to arrive at three wormhole routed algorithms. The selected algorithms display the current trends in wormhole routed networks. The algorithms are ***Channels**, **NHop**, and **f-cube2**. Each of these algorithms uses a dependency graph to detect deadlock [1][3][4].

**Channels* was selected because it is a fully adaptive minimal deadlock free algorithm that is not dependent on the size of the network or the size of the message. The **Channels* algorithm was developed for the N dimensional torus. The number of virtual channels utilized is not dependent on the dimensions of the torus. **Channels* avoids deadlock by way of using two subnetworks. These two subnetworks are actually a means of segregating the virtual channels associated with a physical channel into two groups. The two groups are the **nonstar virtual channels** and the **star virtual channels**. A message initially utilizes the nonstar subnetwork by way of the nonstar virtual channels. While traversing the nonstar subnetwork a message is fully adaptive minimal. If a message encounters a deadlock while being classified as a nonstar message, the message is simply converted to a star message and begins to utilize the star subnetwork through the star virtual channels. A wait is recorded for the message during the conversion process. Star network messages do not carry the fully adaptive minimal privilege. These messages employ dimension ordered routing beginning with the x-axis. A nonstar message may also be converted to a star message, if the message encounters a wrap-around, which is associated with the torus topology. Twenty-five percent of the virtual channels are used to compose the nonstar subnetwork and the remainder of the virtual channels makes up the star subnetwork [1].

The negative hop algorithm, NHop, was selected due to its investigation into the possibility of developing wormhole algorithms from previously proven SAF algorithms. The flexibility of NHop is demonstrated in its ability to segregate a physical channel into multiple virtual channels based on the topology. The formula used to derive the necessary number of virtual channels per node for the mesh is $\lfloor n(k-1)/2 \rfloor + 1$, and $\lfloor n(k/2)/2 \rfloor + 1$ for the torus, where k is the number of nodes and n is the number of dimensions. These formulas must provide at least one more virtual channel than the total number of negative hops possible for a topology for the algorithm to guarantee deadlock free routing. **Hop schemes** refer to the current number of hops of a message. The NHop algorithm labels nodes in a fashion that allows for a

message to take a hop from a node with a higher label to a node with a lower label or, from a lower labeled node to a higher labeled node almost intermittently. Figure 5 illustrates this concept. A **negative hop** occurs when a message goes from a node of a higher label to a node with a lower label. Otherwise, a hop is deemed a **non-negative hop**.

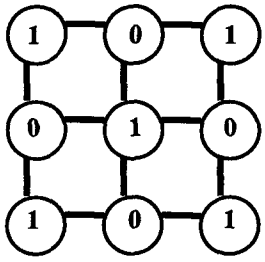
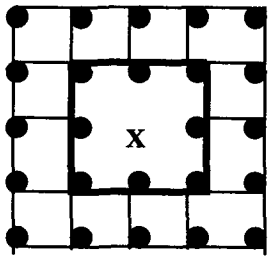



**Figure 5 A possible labeling of a 3X3 2D mesh for the NHop algorithm**

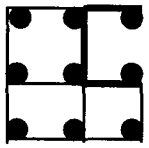
Each time a message takes a negative hop the class of the message is incremented by one. The virtual channels are segregated by class. A message will always use the corresponding class of the virtual channel as the message itself. This means if a message has taken three negative hops, it will have a class of three. And hence attempt a lock on a virtual channel of class three for its next hop. An unsuccessful lock on a virtual channel will lead to a wait being recorded. This algorithm has deficiencies of not favoring hot spots and possibly necessitating the need for too many virtual channels for most networks [3].

The f-cube2 algorithm was selected to model since it displays resiliency in the face of network faults. The algorithm is also fully adaptive minimal however, this requirement is relaxed in the face of network faults. A network fault(s) is described to be a **fault set** during any instance of the algorithm. A fault set may be described as an inoperable node, or a link, which connects two nodes. A message traverses the network until a fault set is encountered. Once a fault set is encountered the message is merely routed around the faulty region or **f-region**. To aid in the routing of messages around the f-region the f-cube2 algorithm uses fault rings and fault chains, **f-rings** and **f-chains** respectively. These rings are actually the non-faulty links and non-faulty nodes that are around a faulty node or a faulty link. Figure 6 illustrates the concept of f-rings for a 5X5 2D mesh and figure 7 illustrates an f-chain for a 3X3 2D mesh.




● **a non-faulty node**
X **a faulty node**
— **a non-faulty link**
— **a non-faulty link used on the f-ring**

**Figure 6 Formation of f-rings**



— **a non-faulty link**
— **a non-faulty link used on the f-chain**
**the faulty link on the network boundary is not shown**

**Figure 7 Formation of f-chains**

The f-ring may only resemble a rectangular region. Routing on the boundary of the network is accomplished by the formation of f-chains. An f-chain is the non-faulty links around a faulty link, which exists on the boundary of the network. An f-chain from one fault set may not overlap with the f-chains or the f-rings of another fault set. The algorithm mandates that at least two virtual channels are available per node to accommodate fault tolerance and non-overlapping fault rings. Each virtual channel is labeled as being within a class as either horizontal or vertical. A message also has a type of either horizontal or vertical. A message may only use the class of virtual channels to coincide with the type of the message. Since virtual channels will aid in the overall network performance [6], f-cube2 allows for a pool of virtual channels to be available at each node. A message will first attempt to lock a virtual channel that coincides with its class. If this virtual channel is not available, then a request is made to obtain a virtual channel from the free pool. If this request is unsuccessful, a wait is recorded for the message [4].

# 5 Simulation Results

Some results obtained from the simulator will be presented in this section. Some of these results coincide with results obtained from other simulators with the same settings, thus providing integrity for this simulator. The presented results will provide a relationship between the network setting and latency. These results will then be used to arrive at an optimal setting for each algorithm.

The **Channels* algorithm was simulated in [1] with a (31,2) torus, 250 flits per buffer, sixteen virtual channels per router, message sizes of fifteen and thirty one flits, and the network volume was increased up to sixty messages. The simulator extends these network configurations to include a (50,2) torus, fifty to 500 flits per buffer, from four to fifty two virtual channels per router, and up to 100 messages for the network volume. The buffer space is segregated evenly into the input and output buffers [1]. The experimental process was divided into four groups where the flit size of the messages and the number of nodes for the topology differentiated the groups. Within each group six experiments were performed. In these six experiments, the flits per buffer and the number of virtual channels were varied. The simulator was executed at various levels of network traffic and at each level an average was taken to arrive at the latency for a given setting. The latency increased as the

network volume increased [1][7]. This is due to more flits being on the network competing for resources and thus the model is recording more wait states. This denotes messages as they are converted from a nonstar message to a star message. When the size of the topology increases, more nodes exist on the network. Thus, each message has the possibility of traveling a further distance to reach its destination. By traveling a further distance a message has to lock more resources and may encounter more wait cycles in an attempt to lock these resources. If resource contention were not encountered, then the latency for a small topology would be nearly the same as a larger topology, since the simulator simulates a wormhole routed network. Figure 8 illustrates that more wait cycles were encountered by the larger 2500 node topology than the smaller 961 node topology.

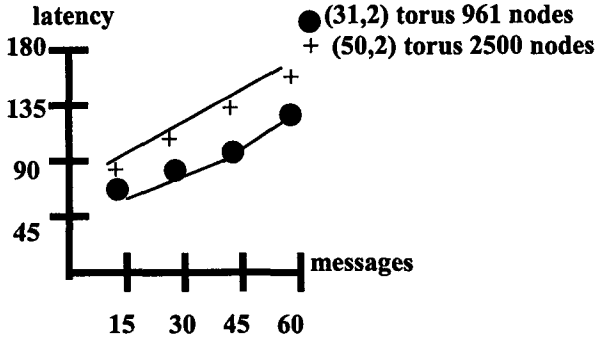



**Figure 8 Effect of number of nodes on latency for the **Channels* algorithm**

To provide an optimal setting, all twenty four experiments were compared to select the setting that provided the overall lowest latency for the **Channels* algorithm. This was a fifteen flit message size, (31,2) topology, 150 flit buffer space, and five virtual channels.

The literature settings for the NHop algorithm are an (8,3) mesh, an (8,3) torus, a (16,2) torus, either sixteen, eighteen, or twenty four buffers per router, a message size of twenty flits, and four flits per buffer [3]. The simulator extends these configurations to include a (16,2) mesh, either eight, thirty six, or forty eight buffers per router, a message size from five flits to forty flits in increments of five, and from one to 100 messages for the network volume. The empirical structure for the NHop experiments were structured to evaluate the latency and the virtual channel utilization. The first latency experiment began with an (8,3) torus that featured twenty messages for the network volume and sixteen buffers per node. As expected, the latency increased with the message size [3][7]. This also revealed the effect of the message size on the latency for the NHop algorithm. A larger message size contributes to more flits on the network, which yields more congestion and latency for the network. A larger message size also requires more time for the router to handle the additional flits of a message. The relationship of message size and network volume to latency is supported by the results in [1][7]. The next configuration, was the same topology but the network volume was doubled. And, in an effort to reduce the expected increase in latency for the extra volume, the buffers per router were increased to forty eight. The initial configuration continued to produce a lower latency, which means more network traffic can not necessarily be compensated for by more buffers for this algorithm. Next, a (16,2) mesh with the same flits per message and buffers per router was compared against the initial configuration. The initial configuration again displayed a lower latency than the new configuration. This reveals the importance of the wrap around strategy of the torus [7]. Also, the distance insensitive routing traits of wormhole routing are evident here by traversing 512 nodes with a lower latency than traversing 256 nodes [1][3][4][7]. These results were also achieved in [3] for the literature settings. Figure 9 illustrates how a torus with a small message size may out perform other configurations that have more buffers per router and less nodes for the topology for the NHop algorithm.

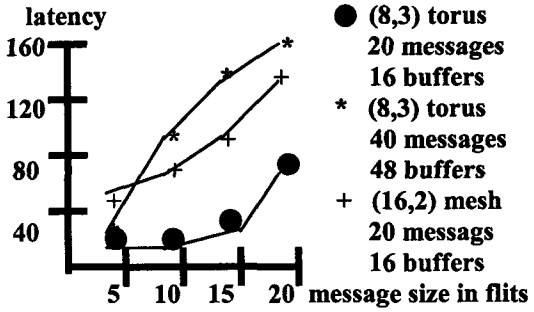



**Figure 9 Effect of message size on latency for the NHop algorithm**

An issue with the NHop algorithm is the possibility of too many virtual channels for most networks [3]. Various topologies were used for this experiment with fifteen flits per message and thirty-six buffers per router. Each configuration displayed a virtual channel utilization rate of less than twenty five percent. From these low utilization rates we may conclude the NHop algorithm in its base form may indeed require far too many virtual channels for some networks. This also leads to the assumption that the upper classes of virtual channels are not being utilized. From the above results, we may conclude the optimal configuration for the NHop algorithm is an (8,3) torus, thirty six buffers per router and a message size of ten flits. The algorithm dictates eleven virtual channels for the (8,3) torus.

The literature settings for the f-cube2 algorithm are a (16,2) mesh, fault sets of 0%, 1%, 5%, and 10%, eight virtual channels per router, and a message size of twenty flits [4]. The simulator extends the settings to include an (8,2) mesh and a (32,2) mesh, a 15% fault set, and from two to twenty virtual channels per router. Also, up to fifty buffers per router, up to fifty flits per buffer, up to fifty

flits per message, and the network volume may be increased up to 100 messages. The empirical structure for the f-cube2 experiments were constructed to show the latency as it relates to the fault sets within a topology. Also, utilization experiments were executed to display the virtual channel utilization and the buffer utilization. The latency reacted much as would be expected for the f-cube2 algorithm as faults were injected into the network. The latency increased for all the fault sets for the topologies that were simulated [4][7]. This means as more fault sets were encountered by the algorithm, messages came into contention for virtual channels and buffer space which meant logging more waits for resources as they were routed around the fault sets. Since a large number of waits were being encountered, the (32,2) mesh had a greater latency than the two smaller meshes. These waits negated the distance insensitive routing traits of wormhole routing. The bisection utilization also increased along with the latency for most configurations [4][7]. This was to be expected since more fault sets were encountered the original path of a message would be disrupted, thus causing more messages to utilize the bisection. The (16,2) mesh revealed much the same effect on latency for the 1% and 5% fault sets as it did with the 0% fault set [4]. The f-cube2 algorithm allows for a pool of virtual channels. This is an excellent opportunity to determine the optimum number of virtual channels for the algorithm. For this experiment to show the resiliency of the algorithm in the face of faults, a 5% fault set was injected into the network. Figure 10 displays the strength of virtual channels to reduce latency for the f-cube2 algorithm.

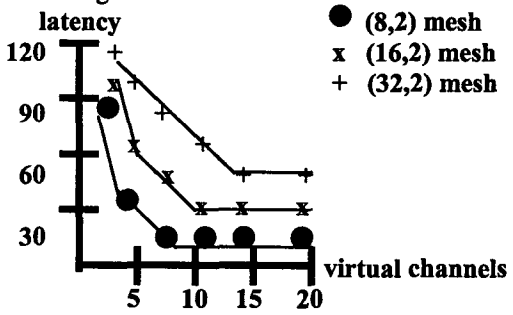



**Figure 10 Latency effect for the simulated topologies and virtual channels for the f-cube2 algorithm**

By simply adding two virtual channels, the latency is reduced significantly, particularly in the smaller mesh. In [4][7] similar results were achieved. The virtual channel utilization experiment displayed much the same shape curve for all the topologies. The utilization decreased considerably above six virtual channels. Buffers were added to each router to determine the optimum number of buffers per router. This experiment began with three buffers per router and was increased up to fifty buffers per router with the most efficient utilization being realized at six buffers per router [7]. As with the virtual channels, the larger mesh continued to display a small decrease in latency above six buffers. The (8,2) mesh and the (16,2) mesh displayed the same latency curve from six to fifty buffers. The results from the simulator have proven the optimum topology for the f-cube2 algorithm to be the (16,2) mesh. This topology displayed less of a reaction in latency to the fault sets. The virtual channel and buffer utilization experiments have shown that six virtual channels and six buffers are the better setting for the algorithm. Further simulations confirmed a small message size of ten flits and ten flits per router to complete the optimum configuration for the f-cube2 algorithm.

# 6 A Comparison of the Algorithms

The **Channels* algorithm employs a relatively simple routing function. A weakness of the **Channels* algorithm is that it is dependent on the topology. The **Channels* algorithm was developed for the n-dimensional torus [1]. A strong point of the NHop algorithm is the simplicity of its virtual channel allocation policy. This logic is embedded in the router at each node [3]. With the simplicity of the virtual channel allocation policy comes a weakness of the algorithm. Even though the algorithm is not dependent on a topology, it may require too many virtual channels for some networks [3]. The results produced by this simulator confirm this to be true. The number of virtual channels are directly dependent on the topology [3]. The f-cube2 algorithm is only one of the few current wormhole routed algorithms to display a degree of fault tolerance [4]. This algorithm allows for a certain number of faults on a network while continuing to maintain a relatively low degree of latency. A weakness of the f-cube2 algorithm is the placement and shape of the faulty regions [4]. Even with these liabilities, the f-cube2 algorithm remains a good base algorithm for studying and developing other fault free wormhole routed algorithms. This fact may be attributed to the simplicity of the algorithm [4].

An experiment was developed to compare the algorithms on the basis of latency. The parameters for each simulation were chosen with regard to fairness of each algorithm while continuing to conform to the requirements of the respective algorithms. Since the only algorithm to display any resiliency to faults was the f-cube2 algorithm, no faults were injected into the network. For this comparison experiment the (16,2) torus was chosen based on the other algorithm's topology [1][3][4]. The f-cube2 algorithm was not developed for the torus, so a (16,2) mesh was chosen for this algorithm [4]. The minimal number of virtual channels required by each algorithm was selected. This was done in regard to NHop since this algorithm necessitates the need for a specific number of virtual channels per node [3]. Thus, the comparison used four virtual channels for **Channels*, nine virtual channels for NHop, and two virtual channels for f-cube2. The number of buffers was selected

to be sixteen for each node and the flits per buffer was chosen to be four. This is an average configuration for wormhole routed simulations [3]. This will allow the routers for each simulation to hold the same number of flits. A small message size of fifteen flits was chosen to keep the latency low for all the algorithms. Hotspots were intentionally injected into the network to insure resource contention to compare the algorithms.

After all, this is what differentiates wormhole routed algorithms, the manner in which each algorithm assigns resources to multiple messages. If no channel and buffer contention were encountered, each algorithm would display approximately the same latency. The **Channels* algorithm clearly displayed a lower latency at all traffic levels. This denotes the efficiency of the virtual channel selection policy, which alludes to less time waiting on resources. The NHop algorithm logged wait cycles in an attempt to allocate the same class of virtual channels. The f-cube2 model is at somewhat of a disadvantage in this experiment since it may not utilize the torus topology and the simulator for this algorithm may only access one virtual channel per class of a message. This means both horizontal and vertical messages spent time waiting to allocate a virtual channel. Figure 11 displays the results of this experiment.

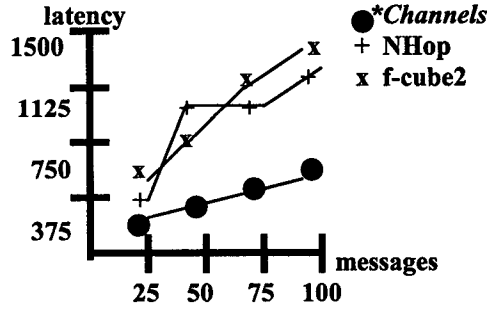



**Figure 11 A comparison of the simulated algorithms in regards to latency**

All of these experiments have shown that via a small number of virtual channels, the overall latency may be reduced. This is attributable to an asset of the wormhole routing switching technique, with the addition of minimal network resources wormhole routing may reduce the overall network latency. **Channels* and NHop does not consider faults on the network, but both have good virtual channel allocation policies [1] [3]. The f-cube2 algorithm was designed specifically to acknowledge faults on the network [4]. Thus, if faults were injected into the network for the comparison experiment, f-cube2 would be favored. Further, by adding several virtual channels to the free pool, f-cube2 would probably have an edge over NHop.

# 7 Future Research

The future research will be discussed in its relationship to the deficiencies of each algorithm. Some of this work has already begun and is in various stages of development [4][5][15][18]. After this future work has been presented, modifications to the simulator will be discussed which will allow for simulations of this new research.

A weakness of the **Channels* algorithm is, it was only developed for the torus [1]. Future investigations could be initiated to determine the algorithms reaction to the mesh topology. The current simulator for this algorithm was developed specifically for the torus and its wrap-around strategy. Modifications to the simulator would need to be made to the function, which calculates the next possible node for a message. The manner in which the function deals with the edge of a network would depend on the topology. More buffer space would probably be required to keep the latency low for the mesh since messages could not use the wrap-around links of the torus to shorten the distance traveled by the messages. A good comparison experiment would be the extra buffers required by the mesh to keep the latency as low as the torus.

A variation of the base NHop algorithm was presented that centered on the number of upgrades for a message [3]. That is, a message was allotted a certain number of bonus upgrades, which was calculated as the possible number of negative hops minus the actual number of negative hops to be taken for the message. Therefore, if a message was unable to allocate a given class of virtual channel, the message could use a bonus upgrade and attempt to allocate a class higher than its current class. This modification for the NHop algorithm is supported in the current design of this simulator. The simulator could be modified to attempt a lock on the highest class of virtual channel available, if the message's current class is not available.

A liability of the f-cube2 algorithm is, the f-regions must be rectangular. This is addressed in [5], and is a modification to the f-cube2 algorithm, that requires a virtual channel to be available for each direction of travel EW, WE, NS, SN. This algorithm allows for f-regions with the shape of T, L or +. The simulator may be modified for this improvement. The main modification would be to acknowledge four message types. Currently messages are of the horizontal or vertical type and the modification would be to place a message in the NS, SN, WE, or EW type depending on the message's direction of travel.

# 8 Conclusions

A summarization of the study will now be presented. This will include the observations already made in the literature, which were used to demonstrate the integrity of the simulator. Also, these results were then used to arrive at the optimal network setting for each algorithm. Network volume is directly proportional to the latency [1][3][4][7][12][16]. The more messages that are placed on a network the more difficult it becomes to allocate the necessary resources for a message to achieve another hop toward its destination. This causes messages to wait and attempt to lock the resources again during the next communication cycle. The message size is also directly related to the latency [1][3][12]. A larger message size

contributes to more flits on the network, which leads to higher network congestion. A larger network topology will produce a higher latency than a smaller network topology [7]. This is relative to the waits encountered. If no waits are encountered, the distance insensitive traits of wormhole routing will produce nearly the same latency regardless of the topology size [10][11]. Therefore, without contention, a 512 node topology will deliver approximately the same performance as a 256 node topology. A torus topology will deliver a lower overall latency when compared to a mesh topology [3][7]. This is due to the wrap-around strategy of the torus, which allows messages to take a single hop to go from one edge of the network to the other edge of the network as opposed to having to traverse the middle area of the network. More buffers will not necessarily compensate for a larger message size [7]. Time will be spent at each router to handle the larger message size. The simulator offered a relatively small number of virtual channels as being the point of diminishing returns. This reveals the nature of wormhole routing. With just minimal additional resources, this switching technique may reduce the latency of a network [10][11]. Figure 12 lists the optimum network setting for each of the three simulated algorithms.

| resource | *Channels | NHop | f-cube2 |
|---|---|---|---|
| topology | (31,2) torus | (8,3) torus | (16,2) mesh |
| buffers | 1 | 36 | 6 |
| buffer size | 150 flits | 4 flits | 10 flits |
| virtual channels | 5 | 11 | 6 |
| message size | 15 flits | 10 flits | 10 flits |

**Figure 12 Optimum network settings for the simulated algorithms**

# References


1 Berman P., Gravano L., and Pifarre D. Adaptive Deadlock and Livelock Free Routing with All Minimal Paths in Torus Networks in *Proceedings of ACM Conference on Parallel Algorithms and Architectures* (1992) pp. 3-12.

2 Bertsekas D. and Gallager R. *Data Networks* Prentice Hall pp. 1-14, pp. 363-379, pp. 493-500, 1992.

3 Boppana R. V. and Chalasani S., A Framework for Designing Deadlock Free Wormhole Routing Algorithms in *IEEE Transactions on Parallel and Distributed Systems* (1997) vol.7, no.2.

4 Boppana R. V. and Chalasani S., Fault Tolerant Wormhole Routing Algorithms for Mesh Networks in *IEEE Transactions on Computers* (1995) vol.44, no.7.

5 Boppana R. V. and Chalasani S., Communication in Multicomputers with Nonconvex Faults in *IEEE Transactions on Computers* (1997) vol.46, no.5.

6 Dally W. J., Virtual Channel Flow Control in *IEEE Transactions on Parallel and Distributed Systems* (1992) vol.3, no. 2.

7 Duato J., Yalmanchili S., and Ni L. *Interconnection Networks, An Engineering Approach* IEEE Press pp. 1-65, pp. 401-472, 1997.

8 Freer J. *Computer Communications and Networks* IEEE Press pp. 1-3, pp. 3-16, pp. 73-82, pp. 91-112, pp. 121-150, 1996.

9 Lafore R. *Object-Oriented Programming in MicroSoft C++* Waite Group Press pp. 7-16, 1992.

10 McKinely P. K. and Ni L. A Survey of Wormhole Routing Techniques in Direct Networks in *Computer* (February 1993) pp. 62-76.

11 McKinely P. K. and Robinson D. F. Collective Communication in Wormhole Routed Massively Parallel Computers in *Computer* (December 1995) pp. 39-50.

12 McKinely P. K. and Tsai Y. An Extended Dominating Node Approach to Broadcast and Global Combine in Multiport Wormhole-Routed Mesh Networks in *IEEE Transactions on Parallel and Distributed Systems* (1997) vol.8, no.1.

13 McKinely P. K. and Tsai Y. A Broadcast Algorithm for All-Port Wormhole Routed Torus Networks in *IEEE Transactions on Parallel and Distributed Systems* (1997) vol.7, no.8.

14 McKinley P. K., et. al. Deadlock-Free Multicast Wormhole Routing in 2-D Mesh Multicomputers in *IEEE Transactions on Parallel and Distributed Systems* (1994) vol.5, no.8.

15 Park S. and Bose B., All-to-All Broadcasting in Faulty Hypercubes in
*IEEE Transactions on Computers* (1997) vol. 46, no.7.

16 Rexford J. et. al. PP-MESS-SIM : A Flexible and Extensible Simulator for Evaluating Multicomputer Networks in *IEEE Transactions on Parallel and Distributed Systems* (1997) vol.8, no.1.

17 Stalling W. *Data and Computer Communications* MacMillan pp. 436-462, 1994.

18 Tanenbaum A. S. *Operating Systems, Design and Implementation* Prentice Hall pp. 122-127, 1987.