

# Simulation of a Mesh of Clos Wormhole Network

A.D. Pimentel

L.O. Hertzberger

Dept. of Computer Systems  
University of Amsterdam  
andy@fwi.uva.nl

## Abstract

Wiring constraints often result in a trade-off between efficiency and realizability when designing communication networks for multicomputers. By combining the connectivity of multistage networks with the easy to realize mesh networks, the Mesh of Clos topology tries to narrow this gap. In this paper, a simulation study is presented in order to evaluate wormhole routed Mesh of Clos communication networks. It is shown that this type of network can substantially reduce contention as compared to flat mesh networks. Furthermore, we found that increasing the number of flitbuffers on router devices does not necessarily lead to improved communication performance. For some application loads it may even result in a loss of performance.

## 1 Introduction

Distributed memory MIMD multicomputers play an important role in the ever continuing pursuit of improving computational power. Usually these multicomputers are constructed by a network of *nodes*, where each node consists of a processor, local memory and several supporting devices. Since nodes do not share memory, communication between nodes is done via message-passing.

Two main types of networks can be distinguished. In *direct* networks each node has a direct, or point-to-point, communication channel to a number of other nodes, called *neighbours*. Neighbouring nodes may send messages to each other directly, while messages between non-neighbouring nodes have to be *routed* by other nodes in the network [3]. The routing is either done by the processor itself or by a special purpose routing device residing at the node. In *indirect* networks there are no point-to-point connections between nodes. Instead, a node is connected to one or more separate routing devices, which subsequently can be connected to other routers. Consequently, all message passing requires routing by at least one of the routing devices.

The *topology* of the network describes the interconnection scheme of the nodes. Example topologies for direct networks are 2D-grids and hypercubes. Networks with multi-

stage topologies can be regarded as examples of indirect networks.

Message passing efficiency heavily depends on the process of transferring data from source to destination, called *switching*. Many switching techniques have been proposed and implemented [11, 5, 2]. One of them, called *wormhole routing*<sup>1</sup> [2], is adopted by an increasing number of multicomputer manufacturers. The reason for this is that wormhole routing offers a low network latency while minimizing hardware expenses. In this technique, a message is divided into a sequence of constant-size *flits* (flow control digits). The first flit (the header) of the sequence holds the destination's address because it is used to determine the path the message must take. As the header advances along its route, the trailing flits follow in a pipelined fashion. Once a channel has been acquired by a message, the channel stays reserved until the last flit (the tail) has been transmitted. Whenever the header encounters a channel that is already occupied by another message, the headerflit is blocked until the channel in question is released. Instead of being buffered in a large message buffer, the trailing flits stay in the network during the stall of the header. The flow of control stops the trailing flits, and they are stored in small flitbuffers along the established route. Besides this elimination of need for large message buffers at intermediate nodes, wormhole routing exhibits a network latency that is nearly distance insensitive if there exists no channel contention among messages [9].

This paper describes the simulation and evaluation of a wormhole routed network connected in a so called Mesh of Clos topology. Our interest in this particular type of network originates from a new series of multicomputers realized by Parsytec that will be based on this communication technology. These machines are currently investigated within the Mermaid project [10], which focuses on the construction of simulation models for MIMD multicomputers with the purpose of performance evaluation. It offers a simulation environment that allows the modelling of and experimentation with a range of architectural design options, like switching and routing techniques, topologies, and variations in compu-

---

<sup>1</sup>The term routing is misleading in this context since no routing is done. The right name should have been *wormhole switching*.

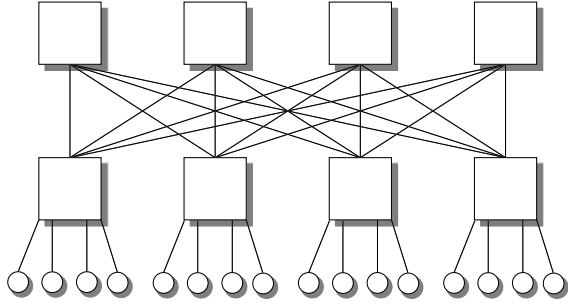


Figure 1: A Clos topology of height 2 with  $k = 4$ . Boxes refer to routers and circles to nodes.

tational and communication power.

Section 2 describes the Mesh of Clos topology and its properties. Section 3 gives an overview of the simulation environment responsible for efficient simulation of the wormhole network under varying communication loads. In section 4, an evaluation study is described with the purpose to gain insight in the behaviour of Mesh of Clos communication networks and to investigate different design options. These options include the type of network, the number of flitbuffers present on a routing device, and different routing techniques. Finally, section 5 concludes the paper and discusses some future work.

## 2 The Mesh of Clos topology

A well-known problem with communication networks is that increasing efficiency often leads to a decrease of realizability. Networks with multistage topologies can offer a small diameter, large bisection width and a large number of redundant paths but are hard to construct because of the complex wiring structure. By combining the connectivity of such multistage networks with the easy to realize mesh networks, the Mesh of Clos network [7] addresses this trade-off between efficiency and realizability. It belongs to the class of indirect networks and is based on the classical Clos network [1]. We define a Clos network of height  $h$  constructed of routers with  $2k$  bidirectional communication channels by the following recursive scheme:

- A single router with  $2k$  bidirectional communication channels of which  $k$  are connected to nodes is a Clos network of height 1.
- A Clos network of height  $h$  is built by connecting  $k$  Clos networks of height  $h - 1$  by  $k^{h-1}$  routers. Since each of the  $k$  subnetworks has  $k^{h-1}$  external communication channels,  $k^{h-1}$  routers are used at level  $h$  such that the  $i$ -th external channel of each subnetwork is connected to the  $i$ -th router at level  $h$ .

Figure 1 shows a Clos network of height 2 for  $k = 4$ . The Mesh of Clos( $h,r$ ) topology is defined by substituting the  $r$  top stages of a Clos network of height  $h$  by a  $\frac{k}{2} \times \frac{k}{2}$  mesh structure. Two examples of such networks are depicted in Figure 2. Both are configured in a  $2 \times 2$  mesh, the first having clusters of four nodes and the other having clusters of sixteen nodes. The latter is actually called a *Fat Mesh of Clos* since the mesh structure that interconnects the clusters is four channels wide. Throughout the paper we assume that routers are configured with eight communication channels of which at most four can be connected to nodes, i.e.  $k = 4$ .

Monien et al. have performed a study on the behaviour of several Mesh of Clos network configurations under a steady state model [7]. They found that for communication patterns exhibiting high locality only a slight decrease in performance is introduced with respect to a pure Clos network. For global communication patterns that are communicating across the mesh network more frequently, on the other hand, the per-

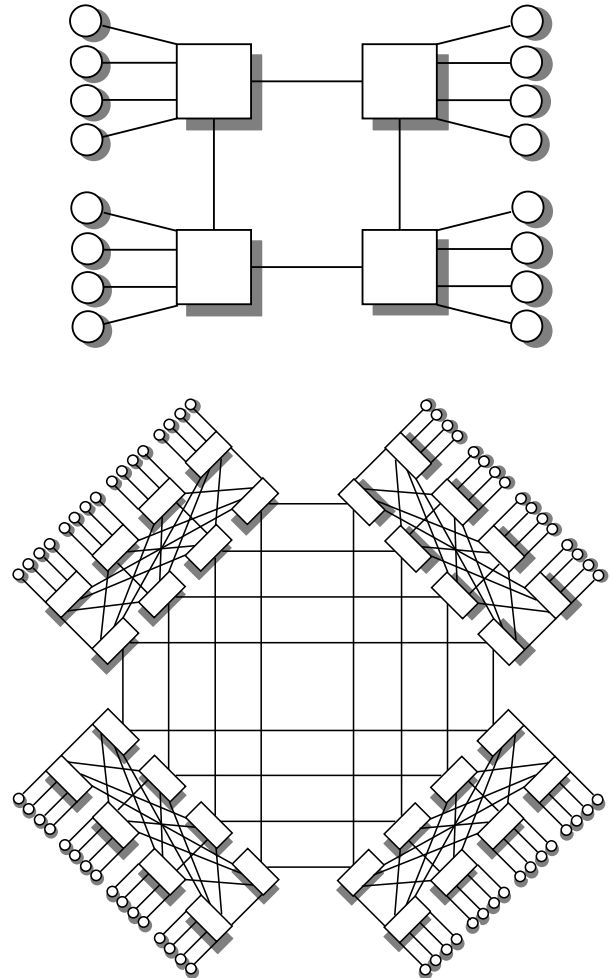


Figure 2: A Mesh of Clos(2,1) network (upper) and a Mesh of Clos(3,1) network (lower). Again the boxes refer to routers and the circles to nodes.

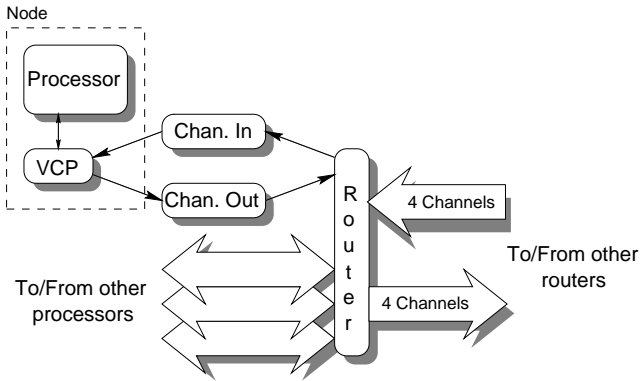


Figure 3: The simulation model of a cluster containing four nodes. An oval box refers to a basic model component.

formance decrease will be larger due to the limitations of the mesh network.

In this evaluation study we will restrict us to instances of the two Mesh of Clos topologies shown in Figure 2. This because the initial machines under investigation will be based on these topologies.

## 2.1 Routing

The routing strategy within a network determines the path messages follow in order to travel to their destination. In this paper, we assume that routing of messages within the mesh structure is performed by a deadlock free deterministic technique based on dimension ordering, called XY-routing [9]. For the Mesh of Clos(3,1) network additional routing is required within the Clos part of the network. The fat mesh structure between the clusters can be regarded as four independent layers of 2-dimensional meshes. The routers directly connected to the nodes, called *node routers*, must decide at which layer messages travel to their destination. For now, we assume a deterministic approach in which a node always sends data over a pre-defined and fixed layer. In other words, node routers use a non-adaptive routing strategy.

## 3 Simulation methodology

The wormhole network simulator used in this study is trace-driven and has been implemented in the object oriented simulation language Pearl [8]. An example of the simulation model infrastructure is shown in Figure 3. It depicts the model of a four node cluster and contains all the basic components. These components, expressed in separate Pearl objects, consist of a processor, a Virtual Communication Processor (VCP), a channel (either input or output) and a router. A processor issues instructions from the input trace. These can either be delay instructions representing computation or message-passing instructions. In the case of the latter, a VCP

is requested to send or receive the message. After this point, the VCP is responsible for handling the transmission. Message setup latency is split up and modelled within both the processor and VCP objects. The bidirectional communication channels are modelled by two unidirectional channel objects. A channel object is a straightforward forwarding mechanism with a certain latency. Finally, the router component routes all incoming flits according to a specified routing algorithm. It connects to other intermediate routers or to neighbouring clusters in order to realize the Mesh of Clos network. All components are highly parameterizable allowing the evaluation of different design options.

### 3.1 Efficient wormhole routing simulation

An important issue in simulating large-scale MIMD multi-computers is to efficiently model the behaviour of the communication network. Explicit simulation of each separate flit must be avoided because this would result in a simulation time that is linear in the message size and in the distance traveled by the message. Instead, full advantage should be taken of the pipelined fashion in which flits move through the network. This behaviour allows for explicitly simulating the header and tail flits only. The movements of the intermediate flits are implicit. This approach basically results in a simulation time that is insensitive to the message size, and thus is only linear in the distance to travel.

The presence of multiple flitbuffers on a routing device makes efficient simulation slightly more difficult. Once the header flit is blocked, the trailing flits continue to be transferred while there are enough free flitbuffers. In [6], McKinley et al. present a simulation algorithm that exploits the implicit movement of intermediate flits and that solves the problem of multiple flitbuffers. The outline of this algorithm applied to the transmission of a single message is shown in Figure 4. There are three variables defined per message. The first, *ttis*, is an array containing the amount time *ttis[i]* that channel *i* requires in order to send the flits currently stored in the flitbuffers of the sending router. Each element of *ttis* is associated with a router along the path followed by the message. The variable *blocked\_at* indicates the amount of time until all movements of trailing flits stall due to full flitbuffers. Finally, *last\_chan* refers to the channel longest held by the message. The key to efficiency for this algorithm is the use of a single value representing the state of a message with regard to each router it traverses.

The network simulator has both been implemented in the naive manner (simulating every single flit) and using the algorithm of Figure 4. The first model was used to verify the more sophisticated implementation of the latter with small communication loads. The efficient simulation model showed an efficiency improvement of more than an order of magnitude compared to the naive approach.

```

tts[i]      : time to send in order to free all buffers
                at channel i.
blocked_at : time to send until transmission of all flits
                within the message is stalled due to full buffers.
last_chan  : longest held channel by the message.

/* "advance tailflit" means releasing last_chan and up- */
/* dating last_chan with the next channel along the route */

while the headerflit has not arrived at destination do
  route headerflit along channel next_chan
  while next_chan is occupied by other message do
    if tts[last_chan] < blocked_at then
      simulate time until tts[last_chan] time units
      have past or next_chan has been released
      if next_chan has not been released then
        advance tailflit
      fi
    else
      simulate time until blocked_at time units have
      past or next_chan has been released
    fi
    update tts array and recompute blocked_at
  done
  if tts[last_chan] ≤ transmission time of a flit then
    simulate transmission time of a flit, advancing
    tailflit after tts[last_chan] time units
  else
    simulate transmission time of a flit
  fi
  update tts array and recompute blocked_at
done
while tailflit has not arrived at destination do
  simulate tts[last_chan] time units
  advance tailflit and update tts array
done

```

Figure 4: Algorithm for efficient simulation of wormhole routing.

## 4 Experiments and results

A suite of four benchmarks was used to evaluate the Mesh of Clos communication network. Each program generates multiple instruction traces, dependent on the number of nodes involved, that act as input for the simulator. All programs are parameterizable for a fixed message size and are scalable to any amount of nodes. Communication used is synchronous only, which means that every message is explicitly acknowledged by a system message. Table 1 gives a short description of each benchmark.

The first benchmark, *ping-pong*, is a simple round-trip of a single message between any of two nodes in the network. The remaining three programs are more sophisticated and produce heavy communication loads. Suppose that  $N$  equals the num-

Benchmark	Description
ping-pong	A round-trip of a message between two nodes
all-to-all	An all-to-all communication load
equal-dist	A point-to-point communication load producing approximately equal-distanced routing paths
unequal-dist	A point-to-point communication load producing a large variety of routing paths

Table 1: The benchmark suite.

ber of nodes involved and  $p$  denotes a node number, then:

- *all-to-all* generates a communication load in which every processor communicates 32 times with all other processors resulting in  $32N * (N - 1)$  overall communication requests.
- *equal-dist* produces a point-to-point communication load in which every processor communicates with one fixed partner. Processors are alternately sending to and receiving from their partner. The partner tuples are formed by the mapping  $(p, ((\frac{N}{2}) + p) \bmod N)$  which results in approximately equal-distanced routing paths.
- *unequal-dist* generates a point-to-point communication load similar to the previous load but with a different partner mapping  $(p, (N - 1) - p)$  which results in a much bigger variety of routing paths.

The number of messages sent to a partner node in the *equal-dist* and *unequal-dist* benchmarks is scaled by a factor  $\frac{4096}{N}$  in order to keep the total number of messages sent constant to 4096. Because of the explicit partner mapping, these two benchmarks put a heavy burden on a fixed set of communication channels resulting in a non-uniform load.

The basic communication architecture used in this evaluation study is a 4x4 Mesh of Clos(3,2) network, which is an extended version of the Mesh of Clos(2,1) network shown in Figure 2. It is parameterized with latencies derived from the specification of one of the early models from the new series of multicomputers under investigation. By default, the routing device is equipped with one flitbuffer per incoming channel. The very limited operating system functionality that is modelled on the nodes takes care of splitting up the messages into packets of 128 bytes. After this the packets are divided into single byte flits with a header of 2 flits attached to each packet.

Simulation runs were performed for different message sizes and scaled to a variety of nodes. In order to compare the simulation results with the communication performance of an existing machine, the benchmarks were executed on a Parsytec GCel multicomputer as well. This machine consists of Inmos T805 transputers connected in a 2D grid network. Routing is implemented in software and is performed

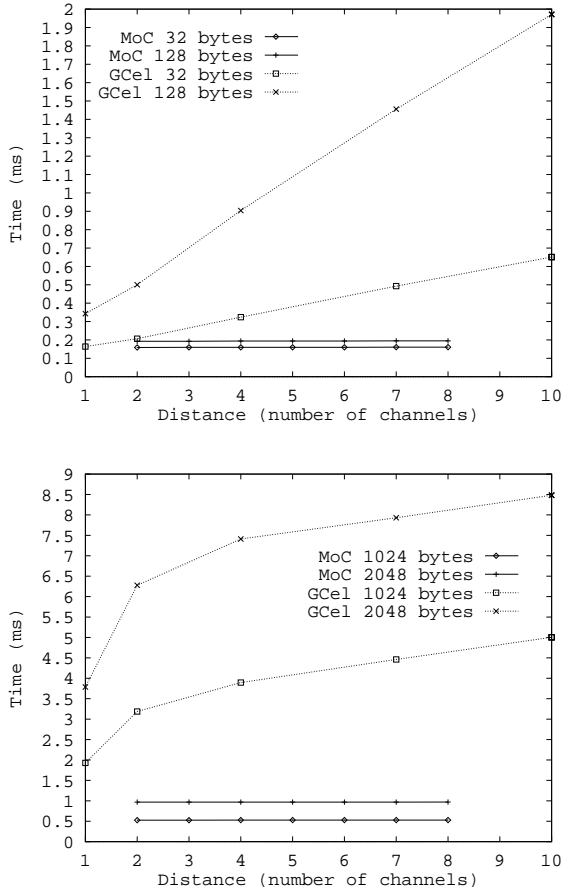


Figure 5: Results of the ping-pong benchmark.

by the transputers. Instead of wormhole routing, the GCell uses store-and-forward switching [9]. Both the performance predictions for the Mesh of Clos network (thick lines) and the execution results of the GCell machine (dotted lines) are shown in the Figures 5 and 6.

Figure 5 presents the results of *ping-pong* for several message sizes and a number of hops. The graph clearly shows that message latency in the wormhole routed Mesh of Clos is almost insensitive to the distance to travel. From two to eight hops only takes a few extra microseconds. This in contrast to the GCell where the message latency is highly distance dependent due to the store-and-forward switching.

The results of the three stress-testing benchmarks are presented in Figure 6. In order to fit all message sizes in one figure, times are shown on a base-10 logarithmic scale. Note that the *equal-dist* and *unequal-dist* programs become faster when using more nodes. This is because of the earlier mentioned scaling that is performed on these benchmarks: A larger number of participating nodes, each producing less messages in order to keep the total amount of messages constant, results in less network congestion.

As expected, the graphs show that the Mesh of Clos communication architecture is several times faster than the GCell

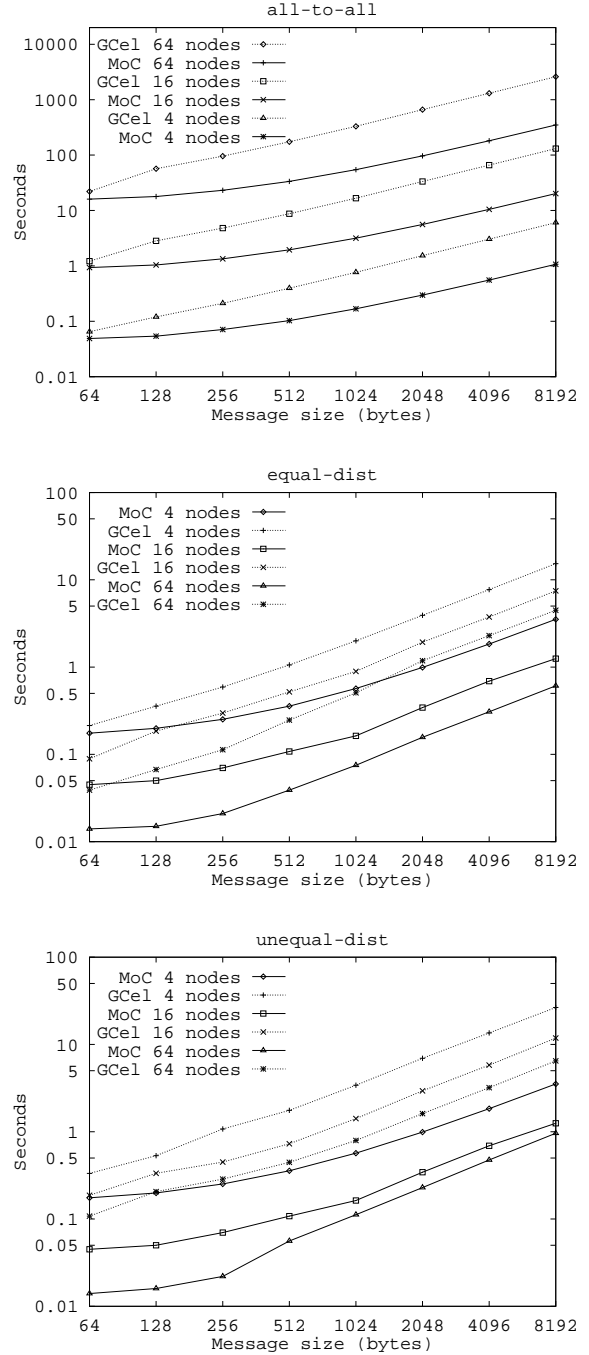


Figure 6: Results of the three stress-testing benchmarks. The top graph shows prediction and execution times of *all-to-all*. The graphs in the middle and at the bottom depict the results of *equal-dist* and *unequal-dist* respectively.

architecture. Whereas the execution times for the GCell are approximately linear to the message size, the predicted execution times for the Mesh of Clos architecture generally exhibit a smoother increase with respect to message size. For smaller messages, the wormhole routing can fully exploit

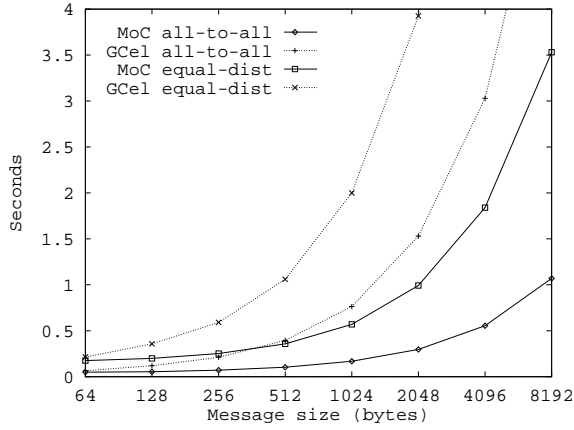


Figure 7: The results of *all-to-all* and *equal-dist* using 4 nodes shown on a normal time scale.

its low communication latency. But as the message size increases, the additional network traffic results in more contention and thus in higher latencies. The graphs in the middle and at the bottom of Figure 6 suggest that this turning point lies at a message size of approximately 512 bytes for *equal-dist* and *unequal-dist* in case of 64 nodes. After this particular message size contention starts to constrain the communication performance.

Figure 7 shows the combined results of *all-to-all* and *equal-dist* using 4 nodes on a normal time scale instead of a logarithmic scale. The steeper slope of the graphs for *equal-dist* with respect to *all-to-all* indicates that there is higher contention in the first benchmark. Identical behaviour was measured for simulation runs using 16 or 64 nodes. And as the similarity of the results of *equal-dist* and *unequal-dist* in Figure 6 already suggests, the same holds for the *unequal-dist* benchmark. The higher contention in these two benchmarks can be explained by the pair-wise communication producing a constant load on a fixed and small set of routing paths.

For further evaluation only the three stress-testing benchmarks were used. So if not explicitly stated, references to benchmarks refer to these benchmarks only.

#### 4.1 Multiple flitbuffers

The use of multiple flitbuffers per incoming link on a router, working as a FIFO buffer, can have a positive effect on the communication performance. By increasing the buffer space channels may be unblocked earlier, and thereby potentially improving the throughput. To explore the influence of multiple flitbuffers in the Mesh of Clos network, we simulated the benchmarks using several router configurations.

Figure 8 depicts the relative speedup due to multiple flitbuffers of the benchmarks simulated for 64 nodes. It shows that the *all-to-all* benchmark greatly benefits from increasing the buffer space. For all message sizes an improved performance was measured due to the larger amount of flitbuffers.

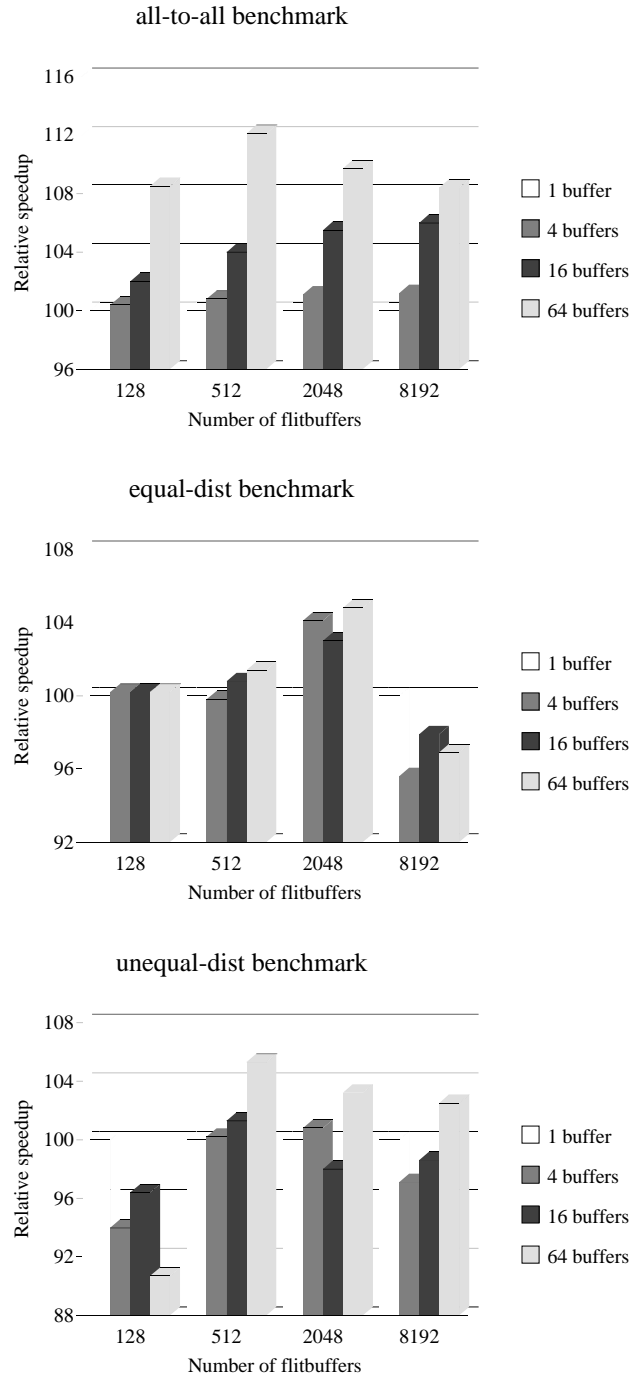


Figure 8: Relative speedup due to multiple flitbuffers in the case of 64 nodes.

The *equal-dist* and *unequal-dist* benchmarks, on the other hand, show a much less predictable performance behaviour. In some cases, even speeddowns were measured for these programs. The sometimes radical behaviour of the benchmarks is due to a number of effects. The multiple flitbuffers cause channels to be unblocked earlier, enabling new messages to enter the network and thereby increasing the network

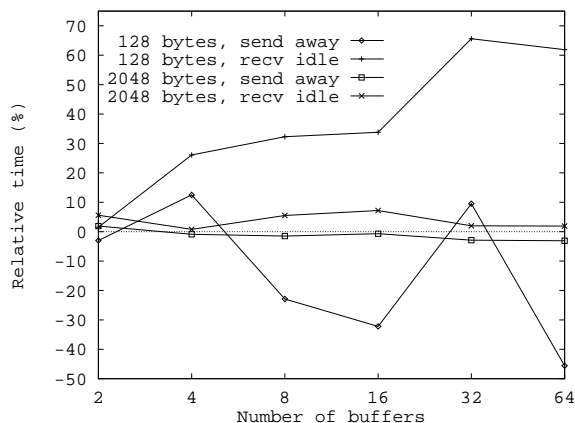


Figure 9: The relative send away time for a packet in the *unequal-dist* benchmark and the relative time that a node idles on the network in order to receive a packet. These results were obtained on 64 nodes and are relative to simulation using one flitbuffer.

traffic. This may lead to longer headerflit stall times and can result in a different *packet flow*. By packet flow we mean the interleaving pattern of packets routed across communication channels within the network. Especially for the *equal-dist* and *unequal-dist* programs, which put a non-uniform load on the communication channels, a change of packet flow may have a substantial performance impact. For example, an increasing number of packets with a nearby destination waiting for packets with a distant destination, can have negative performance consequences. This because of the greater chance of the latter packets being blocked during their journey, and thus a greater probability of keeping the channels in their paths occupied.

The change of packet flow can have another effect on communication performance. Since there are several possibilities for latency hiding within our communication architecture, a different packet flow may result in distinct latency hiding patterns. For instance, during message setup times of the processor messages may be received and processed in parallel by the VCP. In order to demonstrate the influence of the aforementioned effects on the performance of the benchmarks, we first present an abstract view of the causes of performance fluctuations due to different buffer space sizes. Thereafter we descend in abstraction level and give a more low-level description of the changes in communication behaviour.

Figure 9 gives an impression of the influence of some macro effects on the performance of *unequal-dist*. For two message sizes it shows the relative difference in time it takes for a node to send a packet away with respect to the single flitbuffer situation. Additionally it gives the relative increase in time a node idles on the network in order to receive a packet. The first gives more insight into the effectiveness of the multiple flitbuffers allowing packets to enter the network earlier.

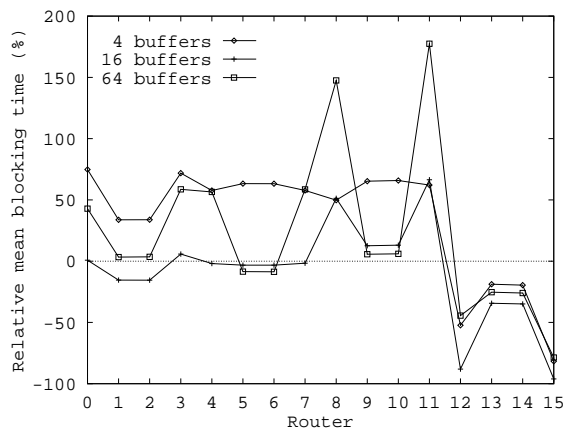


Figure 10: The relative difference in mean time that headers are blocked at a certain router within the mesh structure compared to the single flitbuffer situation. These results are for *unequal-dist* on 64 nodes with a message size of 128 bytes.

The lower the send away time the earlier new packets can be injected into the network. The second effect deals with the consequences of a changed packet flow. The receive idle time is affected by both the blocking times of packets during their journey and by the occurrence of latency hiding.

The irregular behaviour shown in Figure 9 illustrates the impact of changing buffer size on the packet flow in the network, and thus on the communication performance shown in Figure 8. It suggests that nodes are idling longer if using more flitbuffers. For messages of 2048 bytes using routers with 4, 32 or 64 flitbuffers, this behaviour is compensated by a higher relative decrease in send away time. Hence, a performance speedup is realized in these cases (see Figure 8). For messages of 128 bytes on the other hand, the compensation to overcome the higher receive idle times is only large enough in the case of 2 flitbuffers. This results in the speeddowns as shown in Figure 8.

To give an impression of the changes in contention behaviour due to the different packet flow patterns, Figure 10 presents a more microscopic image of what is happening within the network. For several flitbuffer sizes, it shows the relative difference in mean blocking time of headers at the individual routers compared to a single flitbuffer situation. Routers within the mesh structure are numbered row-wise, starting at the top leftmost router. The depicted results were obtained for *unequal-dist* on 64 nodes with a message size of 128 bytes. The trend of the graph in Figure 10 gives some additional explanation of the speeddowns shown in Figure 8. For most routers in the network an increased mean blocking time of headers was measured, with the 16 buffer configuration performing somewhat better than the other configurations. And especially in the case of 64 flitbuffers, the figure shows that different packet flows can amplify hot-spots within the network. This explains the large speeddown mea-

sured for the case of 64 flitbuffers (see Figure 8).

Furthermore, the smaller relative blocking times for routers 12 up to 15 indicates an improved contention behaviour at the bottom row of the mesh structure for all buffer sizes. Nevertheless, this improvement does not compensate the deteriorated contention behaviour at the other routers.

To conclude, uniform communication loads like *all-to-all* may benefit from a larger buffer space while for non-uniform loads an increased number of flitbuffers can lead to a loss of performance as underlined by the results for *equal-dist* and *unequal-dist*. As shown in Figure 10, the extra flitbuffers can cause hot-spots within the network to be amplified, resulting in a degrading overall communication performance.

#### 4.1.1 Raw communication

To gain additional insight in the direct influence of different packet flow patterns on communication performance, we simulated the *unequal-dist* benchmark on a so called *raw* communication architecture. In this architecture, which is again a Mesh of Clos(3,2), there are no other latencies than the raw transmission and routing latencies. Message setup times and message processing times have been nullified. This approach

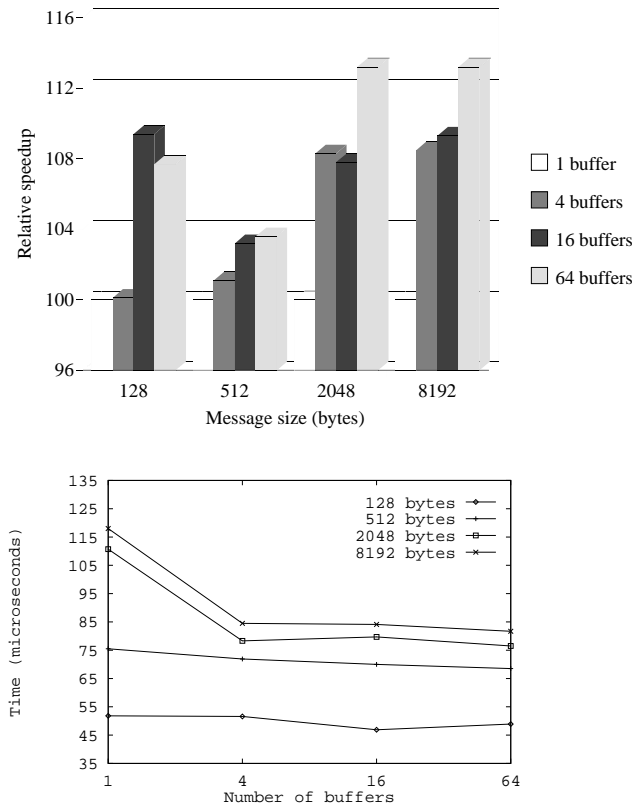


Figure 11: *Raw* multiple buffer performance of *unequal-dist* on 64 nodes (top). And the mean *effective* blocking time of a packet within the mesh structure (bottom).

has the advantage that measurements are not disrupted by latency hiding effects.

Figure 11 shows both the speedup due to multiple flitbuffers for *unequal-dist* on the raw network and the mean *effective* blocking time of a packet within the mesh structure. The effective blocking time is defined as the time a headerflit has to wait for a channel to become unblocked while it is *ready* to send. This means that the headerflit has to reside in the front flitbuffer of the incoming link. With this effective blocking time one has a means to measure the direct influence on network performance of the different packet flow patterns that are caused by the multiple flitbuffers.

Figure 11 suggests that the elimination of latency hiding effects results in a less irregular performance behaviour. Increasing the number of flitbuffers more or less improves performance in all cases. Nevertheless, like was shown in the previous section, the improvement of raw communication does not guarantee a better overall performance. As illustrated by the good fit between the speedup graph at the top of Figure 11 and the effective blocking time graph at the bottom, the communication performance within the raw communication network is dominated by the effective blocking time of packets traversing through the mesh structure. The relative speedup increases whenever the mean effective blocking time per packet decreases and vice versa.

#### 4.2 The Mesh of Clos(3,1) topology

The basic Mesh of Clos(3,2) network used in the previous section is very similar to a normal mesh network. This means that its communication performance is still highly constrained by the global mesh structure. Extending the basic network architecture to a Mesh of Clos(3,1) results in a larger bisection width and increases the number of redundant paths, which subsequently may lead to improved communication performance.

In this section we evaluate a 2x2 Mesh of Clos(3,1) network, of which the total number of nodes is consistent with the original basic network. As mentioned in section 2.1, the node routers decide at which layer packets travel to their destination. By default, we use a deterministic approach in which a node always sends data over a pre-defined and fixed layer. Further, there is still one flitbuffer per incoming link on the routers, and XY-routing is performed within the layered 2-dimensional meshes.

The benchmarks have been simulated for the new network architecture with all 64 nodes participating. Figure 12 depicts the relative speedup compared to the runs on the basic Mesh of Clos(3,2) network. Whereas speedups up to 60 percent for *unequal-dist* and up to 40 percent for *equal-dist* were obtained, there is only little improvement of the *all-to-all* benchmark. This can be explained by the higher contention that appeared for the first two programs on the original Mesh of Clos(3,2) network. We measured, for example, a mean ef-

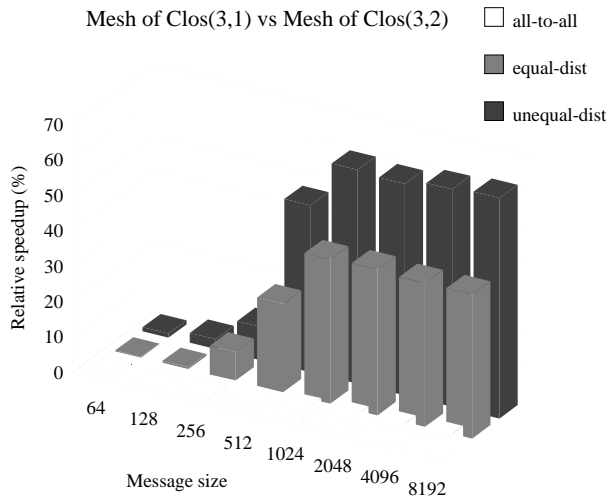


Figure 12: Relative speedup of the benchmarks on a Mesh of Clos(3,1) compared to the basic Mesh of Clos(3,2) network.

fective blocking time of only 2 microseconds per packet for messages of 2048 bytes in *all-to-all*, while for the same message size the mean effective blocking times of *equal-dist* and *unequal-dist* are 44 and 61 microseconds per packet respectively. Because of the improved communication structure of the new network this contention may be dramatically diminished. Figure 12 shows that the large speedups for *equal-dist* and *unequal-dist* are obtained for message sizes of 512 bytes and larger. This perfectly matches with our earlier observation that contention starts to constrain the performance of these benchmarks after this particular message size in the basic communication network. So one can conclude that the Mesh of Clos(3,1) substantially reduces contention as compared to the basic network, which can improve performance significantly. Naturally, this improvement will be even bigger with respect to normal mesh networks.

Like was done for the basic network, we investigated the performance consequences of multiple flitbuffer configurations for the Mesh of Clos(3,1). Figure 13 shows the relative speedup measured under varying buffer space sizes. The *all-to-all* benchmark shows similar results as obtained in the case of the basic network. This means that there is still enough contention to benefit from the extra flitbuffers. The decreasing speedup after a message size of 512 bytes using 64 flitbuffers still indicates some influence of different packet flows. For *equal-dist* and *unequal-dist* again a more irregular behaviour was measured. The great similarity between the speedup graphs of these benchmarks suggest that the reduction of contention degrades the two programs to almost identical communication loads. Dramatic speeddowns of nearly 30 percent were obtained with a message size of 2048 bytes and 4 flitbuffers. Further examination showed a substantial increase of effective blocking time of packets in this particular case. This suggests a deteriorating packet flow within the

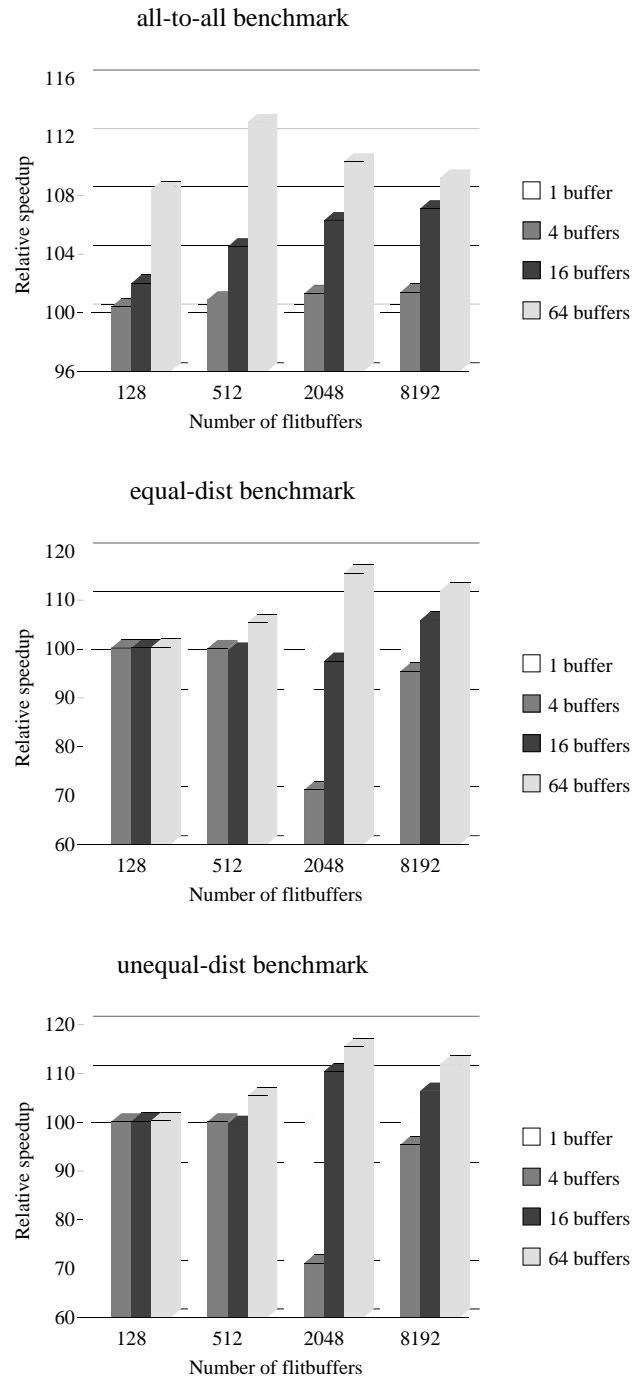


Figure 13: Relative speedup due to multiple flitbuffers in the Mesh of Clos(3,1) network.

network.

The absence of speedup for small messages in *equal-dist* and *unequal-dist* is another indication that contention among messages is diminished. Without contention, communication does not benefit from a larger amount of flitbuffers. Only for larger messages enough network traffic is generated to pro-

Strategy	Description
<i>Random</i>	Randomly choose the link to a layer
<i>RoundRobin</i>	Choose the link to a layer in round robin fashion
<i>IdleFixed</i>	Choose an idle link to a layer. If no link is idle then choose a fixed and pre-defined link, depending on the originating node
<i>IdleRnd</i>	Same as <i>IdleFixed</i> , but if no link is idle then randomly choose a link

Table 2: The adaptive node router strategies.

duce some contention, as illustrated by the more irregular behaviour after a message size of 512 bytes in Figure 13.

#### 4.2.1 Routing strategies

The non-adaptive routing strategy of the node routers is reasonable simple to implement and does not require extra hardware or software support by the VCP's. Nevertheless, this strategy might not fully exploit the potentials of the four layered mesh structures. Adaptively routing the packets among the four layered meshes on the other hand may result in better utilization of network resources. When a link to a certain mesh layer is busy while one or more links to other layers are idle, packets can be transmitted along these idle links. Because in this scheme packets within a message may travel through distinct mesh layers with potentially different latencies, packets can arrive out of order at the destination VCP. Therefore, additional support by the VCP is required in order to reconstruct a message using the correct sequence of packets.

Several adaptive node router strategies have been simulated to investigate the performance effects on the benchmark suite. To model the reconstruction of messages, an extra latency has been added for each message receipt at a VCP. Routing within the 2-dimensional meshes is still performed deterministically. This means that no deadlock prevention is necessary as packets cannot switch from mesh layer during their journey and thus no "inter-layer" dependency cycles can be formed. The different adaptive routing strategies that have been tested are listed and described in Table 2. Because the *RoundRobin* approach chooses the link with the greatest chance of being idle, it effectively resolves into some sort of *IdleRoundRobin* strategy: when no link is idle then a link is chosen in round robin fashion.

The performance effects of the adaptive strategies as compared to the non-adaptive scheme are shown in Figure 14. These relative performance figures have been averaged over different message sizes using the geometric mean, which is correct for normalized numbers [4]. The graphs suggest that for our benchmarks the adaptive strategies can not compete with the original deterministic scheme. With a few exceptions, communication performance has decreased. In most

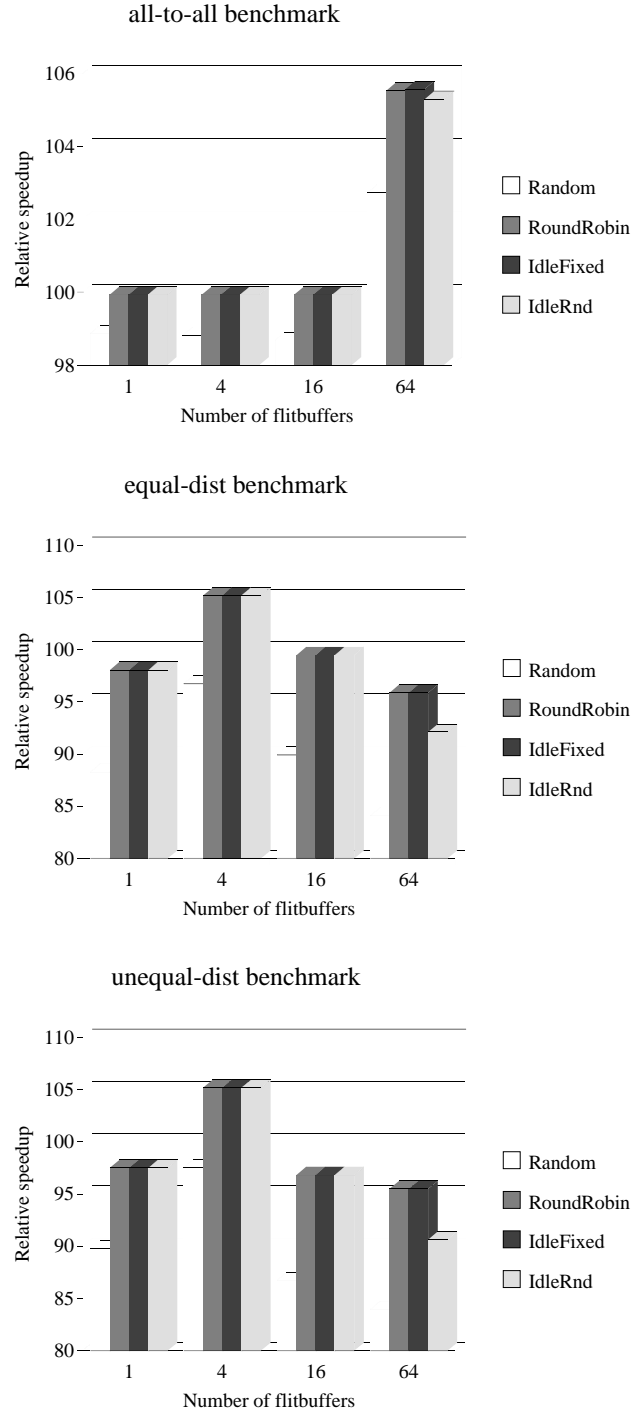


Figure 14: Relative speedup due to adaptive node router strategies compared to the non-adaptive approach.

cases the overhead due to the reconstruction of messages can not be compensated by a higher throughput. The few speedups were mainly obtained in the cases where the original network shows deteriorated performance for multiple flit-buffer configurations (see Figure 13).

Comparing the adaptive strategies, *Random* clearly has the worst performance. The remaining approaches exhibit identical behaviour. No or very small differences between *RoundRobin* and *IdleFixed* were measured, and *IdleRnd* only shows decreased performance in case of a large buffer space.

## 5 Conclusions

In this paper we presented an evaluation study of wormhole routed Mesh of Clos networks. To efficiently simulate communication within the network the implicit movement of intermediate flits in packets has been exploited. This allowed us to only simulate the header and tail flits, instead of simulating every single flit explicitly.

Obtained results for a communication benchmark suite show that Mesh of Clos networks are potentially less prone to contention than more purely mesh based networks. Increasing the size of the Clos part of a network resulted in speedups of up to 60 percent. This was obtained for non-uniform communication loads producing high contention on more flat Mesh of Clos networks. Evidently, the performance improvement is even bigger when compared with normal mesh networks.

It was shown that increasing the buffer space on router devices does not guarantee a better performance. While uniform communication loads may fully benefit from a larger number of flitbuffers, the impact on performance for non-uniform loads is much less predictable. In some cases a larger buffer space can even lead to a loss in performance. The extra flitbuffers may affect latency hiding patterns in the communication architecture and can change the packet flow within the network, which subsequently can cause hot-spots to be amplified. This results in a degrading overall communication performance.

For a Mesh of Clos network containing multiple independent layers of small 2-dimensional meshes we also investigated several strategies to route packets among these layers. As a result we found that for the benchmarks used none of the adaptive routing strategies was able to compete with a straightforward deterministic scheme. So on first sight it might not be worthwhile to invest in extra hardware or software necessary for adaptive routing. However, whether this behaviour also holds for Mesh of Clos networks with larger multi-layered mesh structures still has to be investigated. Furthermore, evaluation of adaptive routing strategies within the meshes as compared to the deterministic XY-routing, used so far, requires additional research.

## Acknowledgments

We would like to thank Marcel Beemster for his comments on a draft version of this paper.

## References

- [1] C. Clos. A study of non blocking switching networks. *Bell System Technical Journal*, pages 775–785, March 1953.
- [2] W. J. Dally and C. L. Seitz. The torus routing chip. *Journal of Distributed Computing*, 1(3):187–196, 1986.
- [3] S. A. Felperin, L. Gravano, G. D. Pifarre, and J. L. Sanz. Routing techniques for massively parallel communication. In *Proceedings of the IEEE*, volume 79, pages 488–503, Apr 1991.
- [4] Ph. J. Fleming and J. J. Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *Communications of the ACM*, 29(3):218–221, Mar 1986.
- [5] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3(4):267–286, 1979.
- [6] P. K. McKinley and C. Trefftz. Multisim: A simulation tool for the study of large-scale multiprocessors. In *Proceedings of the 1993 International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Networks (MASCOTS)*, pages 57–62, Jan 1993.
- [7] B. Monien, R. Lüling, and F. Langhammer. A realizable efficient parallel architecture. In *Proceedings of the first International Heinz Nixdorf Symposium: Parallel Architectures and their Efficient Use*, volume 678, pages 93–109, 1992.
- [8] H. L. Muller. *Simulating computer architectures*. PhD thesis, Dept. of Comp. Sys, Univ. of Amsterdam, Feb 1993.
- [9] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, 26:62–76, Feb 1993.
- [10] A. D. Pimentel, J. van Brummen, Th. Papathanassiadis, P. M. A. Sloot, and L. O. Hertzberger. Mermaid: Modelling and Evaluation Research in MIMD Architecture Design. In *Proc. of the High Performance Computing and Networking Conference, Lecture Notes in Computer Science*, volume 919, pages 335–340. Springer Verlag, Berlin, May 1995.
- [11] C. L. Seitz. The cosmic cube. *Communications of the ACM*, 28:22–33, Jan 1985.