Evaluation of a Mesh of Clos Wormhole Network

A.D. Pimentel L.O. Hertzberger
Dept. of Computer Systems, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
E-mail: {andy,bob}@fwi.uva.nl

Abstract

The pursuit of high connectivity in network design for multicomputers is often complicated by wiring constraints, resulting in a trade-off between efficiency and realizability. The Mesh of Clos topology addresses this trade-off by combining a multistage network with a mesh network. 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 in comparison with flatter 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 degradation of performance.

1. Introduction

Distributed memory MIMD multicomputers play an important role in the continuing pursuit of improving computational power. Their message passing efficiency depends heavily on the process of transferring data from one node to another, commonly referred to as *switching* [10, 2, 3]. In the past few years, one switching technique in particular, called *wormhole routing* [2], has become increasingly popular. 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 since 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. This results in a network latency that is nearly distance insensitive if there exists no channel contention between messages [7]. 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 one large message buffer, the trailing flits stay in the network during the stall of the header. They are stored in small flitbuffers along the established route, which eliminates the need for large message buffers at intermediate nodes.

This paper describes the simulation and evaluation of a wormhole-routed network connected in a so-called Mesh of Clos topology. It essentially extends the work done by Monien et al. [5], who evaluated several configurations of Mesh of Clos topologies under a steady state model. 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. The performance of these machines is currently being evaluated within the Mermaid project [9], which focuses on the construction of simulation models for MIMD multicomputers. It offers an environment that allows the modelling of and experimentation with a wide range of architectural design options.

In the next section, the Mesh of Clos topology and its properties are described. Section 3 gives a brief overview of the technique we applied in order to efficiently simulate wormhole-routed networks. In section 4, the Mesh of Clos communication network is evaluated and several design options are investigated. These options include 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

Increasing the efficiency of communication networks 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 a multistage network with an easy to realize mesh network, the Mesh of Clos network [5] addresses this trade-off between efficiency and realizability. As its name already suggests, it is based on the Clos multistage network [1]. We define a Clos network of height h, constructed of

0-8186-7557-8/96 $5.00

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 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.

Subsequently, the Mesh of Clos(h,r) topology is defined by replacing the r top stages of a Clos network of height h by a $2^r \times 2^r$ mesh structure. Here we assume that the routers are configured with eight communication channels of which at most four can be connected to nodes, i.e. $k = 4$. Two examples of a Mesh of Clos network are depicted in Figure 1. Both are configured in a 2×2 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 can be regarded as four independent layers of 2D meshes (as illustrated by the different shaded surfaces in Figure 1).

Table 1 gives a comparison of the network parameters for the Mesh of Clos and mesh topologies. Furthermore, Figure 2 shows the number of routers required to construct a Mesh of Clos network (dark surface) and a normal mesh network (light surface). The axis labeled with r defines the number of top stages that are removed from the Clos network. Note that the surface within the triangle between $r = 2$ and 1024 nodes is not defined because r would then be equal to or larger than the height of the Clos network. Figure 2 illustrates that the number of routers, which is rapidly increasing for a pure Clos ($r = 0$), can be reduced considerably by replacing the top stages of the Clos for a mesh structure, i.e. by increasing r. On the other hand, Table 1 indicates that r should not become too large in order to preserve a small network diameter and a large bisection width.

In this evaluation study, we will restrict us to instances of the two Mesh of Clos topologies shown in Figure 1. This because the machines under investigation will be based on these topologies, allowing us to validate our simulation models whenever the actual machines become available.

	Mesh of Clos(h,r)	Mesh
Router degree	8	4
Diameter	$2 \cdot (2^r - r - 1) + \log_2 N$	$2 \cdot (\sqrt{N} - 1)$
Bisection width	$N \cdot 2^{-r-2}$ (with $r > 0$)	$\sqrt{N}$

Table 1. Network parameters of the Mesh of Clos and mesh networks containing N nodes.

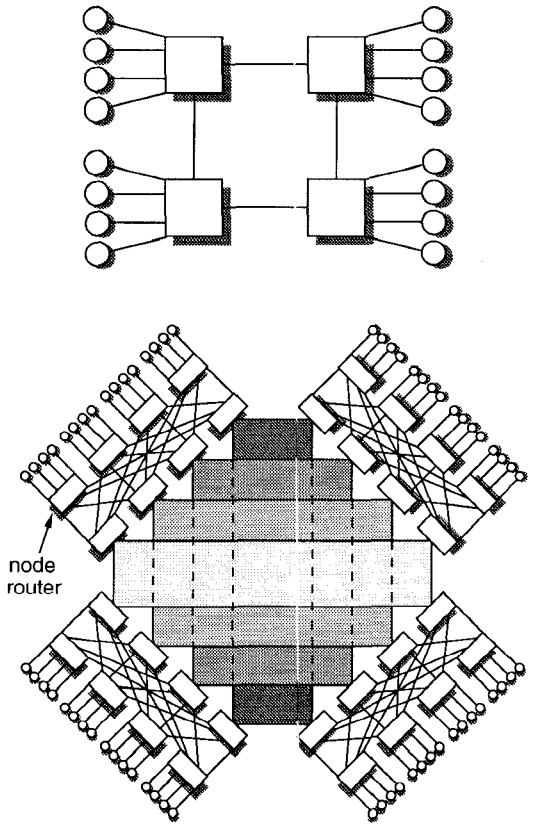

Figure 1. A Mesh of Clos(2,1) network (top) and a Mesh of Clos(3,1) network (bottom). The boxes refer to routers and the circles to nodes.

2.1. Routing

Throughout this paper, we assume that routing of messages within the mesh structure is performed by a deadlock free deterministic technique based on dimension ordering [7]. For the Mesh of Clos networks containing Clos structures of height 2, additional routing is required within the Clos parts. The routers directly connected to the nodes, called *node routers* (Figure 1), must decide at which mesh-layer messages should 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 according to the following scheme: if a node router connects to nodes n_i (with $0 \leq i \leq 3$) and the four mesh-layers are numbered 0 to 3 then messages from node n_i are routed to mesh-layer i.

3. Efficient simulation

An important issue in simulating large-scale MIMD multicomputers is to model efficiently the behaviour of the communication network. Explicit simulation of each separate flit must be avoided because this would result in a simula-

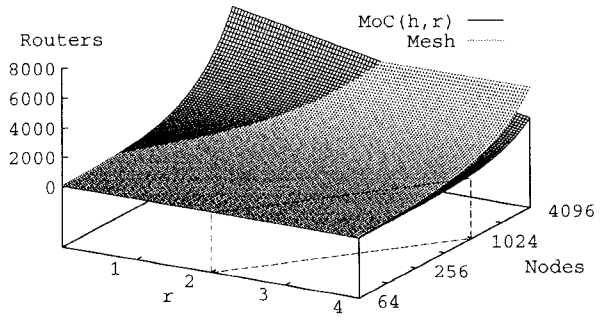

Figure 2. Number of routers required for a Mesh of Clos (dark surface) and a normal mesh network (light surface).

tion 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 movement of the intermediate flits is implicit. This approach results in a simulation time that is basically 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 [4], McKinley and Trefftz describe a simulation algorithm that exploits the implicit movement of intermediate flits and that deals with the problem of multiple flitbuffers.

Our network simulator, written in the architecture simulation language Pearl [6], has been implemented in both the naive manner (simulating every single flit) and using the optimized algorithm from [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.

4. Experiments and results

To evaluate the Mesh of Clos network, four types of communication loads were used. All loads are synthetic and therefore only marginally represent the behaviour of real applications. Furthermore, the loads perform synchronous communication, which means that every message must be explicitly acknowledged by a system message. As a result, the acknowledgements put an additional load onto the network. The reason for using synchronous communication originates from our interest in SPMD (Single-Program Multiple-Data) applications. Typically, these data parallel programs communicate via synchronous message passing.

In the first communication load, called *uniform*, messages are sent to destinations which are uniformly distributed. The second load, referred to as *hotspot*, represents a less ideal model of communication. It defines several non-neighbouring nodes that receive a disproportionately large number of messages, causing hotspots to appear within the network. This type of load is similar to the one used by Pfister and Norton to study hotspots in shared memory systems [8]. The third load, called *hotregion*, distinguishes a whole cluster of nodes forming a "hot region", rather than defining single nodes as hotspots. In both *hotspot* and *hotregion*, the total hotspot area receives 40% of all messages. Finally, *partner* represents a point-to-point load, in which every processor communicates with one fixed partner. The partner tuples are formed such that there exists a variety of routing-path distances. For all communication loads, messages are generated at fixed intervals of time.

The communication loads have been simulated for four different Mesh of Clos (MoC) topologies. Two of them are based on Clos structures of height 1, whereas the other two contain Clos structures of height 2. The Mesh of Clos networks of height 1, the MoC(3,2) and MoC(4,3), are similar to the MoC(2,1) network shown in Figure 1. However, they contain meshes of 4×4 and 8×8 respectively. The Mesh of Clos networks of height 2 are the MoC(3,1) (see Figure 1) and the MoC(4,2), of which the latter contains a 4×4 mesh. Every network model is parameterized with latencies derived from the specification of an early model of the new multicomputers under investigation. By default, the routing devices are equipped with one flitbuffer per incoming channel. Furthermore, the very limited operating system functionality that is modelled on the nodes, takes care of splitting up messages into packets of 128 bytes. After this is done, the packets are divided into single byte flits with a header of 2 flits attached to each packet.

Figures 3 and 4 show the estimated network throughputs for the different types of communication loads. The solid lines refer to the networks containing 256 nodes, whereas the dotted lines refer to the networks of 64 nodes. To compare the Mesh of Clos results, Figures 3 and 4 also depict the simulation results obtained for a normal mesh network. This mesh network model was parameterized using exactly the same communication parameters as the Mesh of Clos model.

The point after which no additional throughput is obtained for increasing message sizes, indicates that the network is saturated. For the *hotspot* and *hotregion* loads, Figures 3 and 4 show that the networks containing 256 nodes reach this point of saturation earlier than their counterparts of 64 nodes. The reason for this is that the larger number of nodes can stress the specific hotspot areas more intensively. Moreover, the graphs clearly show the decrease of throughput when communication is not ideally, uniformly, distributed. For example, the hotspots created by the *hotspot*

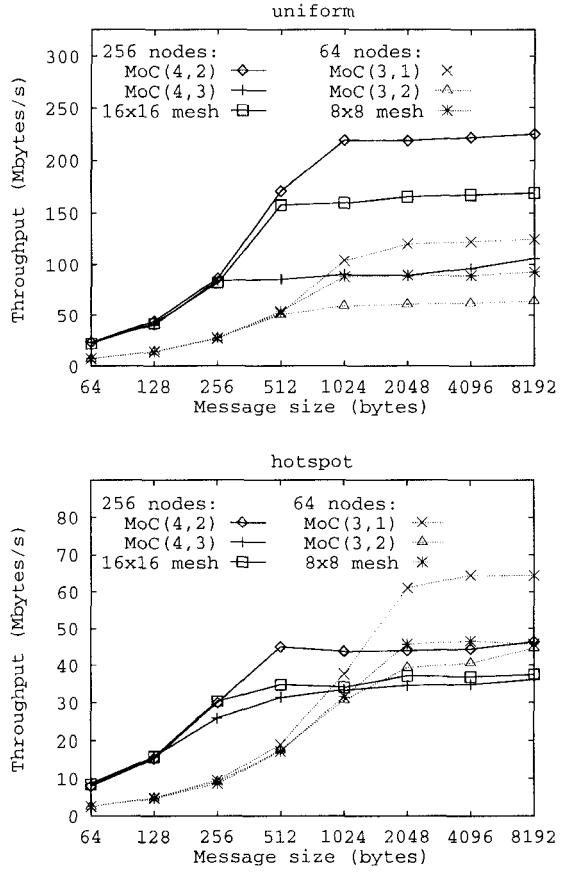

Figure 3. Estimated throughputs for the *uniform* and *hotspot* communication loads.

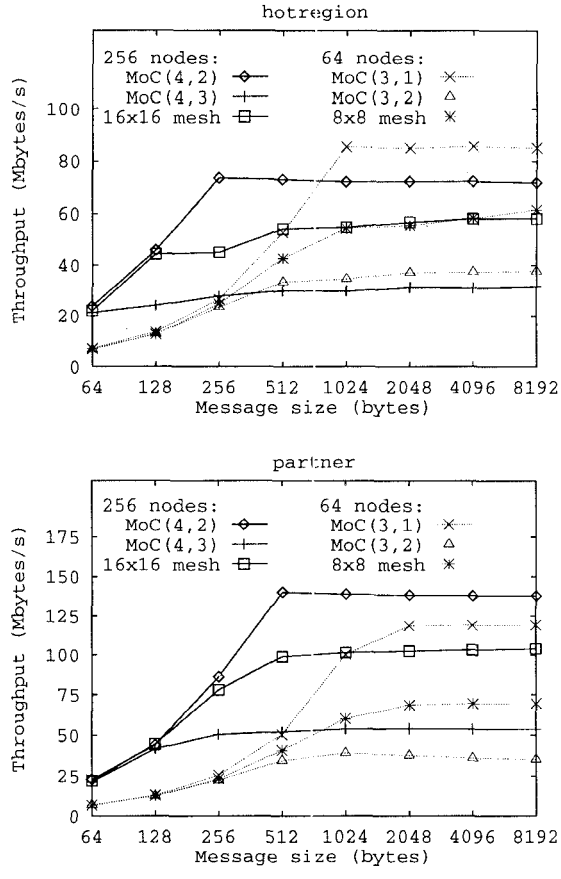

Figure 4. Estimated throughputs for the *hotregion* and *partner* communication loads.

load typically cause a throughput deterioration of more than 50% compared to uniform communication.

The MoC(4,2) and MoC(3,1) networks, being the Mesh of Clos networks of height 2, achieve the highest throughputs for all communication loads. Using these topologies, we measured throughputs that are up to 75% higher than those for the normal mesh network. This suggests that both networks are less prone to contention than the more flat networks that have been examined. As a consequence, the Mesh of Clos networks of height 2 often saturate more slowly than the other networks. To illustrate this, consider the graph of *uniform*. It shows that the 16×16 mesh saturates at a message size of 512 bytes, while the MoC(4,2) reaches the point of saturation at a message size of 1Kb.

The results also indicate that the performance of the mesh networks is superior to that of the Mesh of Clos networks of height 1 (i.e. the MoC(4,3) and MoC(3,2) networks). Apparently, the routers within the latter type of network suffer from high contention. This can be explained by the fact that they must handle traffic from both their four nodes and from their neighbouring routers within the flat mesh structure. As the Mesh of Clos of height 1 shows such poor performance, we will not use this type of network for further evaluation. The next sections, which address the buffering and routing characteristics of Mesh of Clos networks, will therefore only focus on the MoC(4,2) topology.

4.1. Multiple flitbuffers

The use of multiple flitbuffers per incoming channel on a router can have a positive effect on the communication performance. By increasing the buffer space, channels may be unblocked earlier, thereby potentially improving the throughput. To investigate the influence of multiple flitbuffers in the Mesh of Clos network, we simulated the communication loads using various router configurations.

Table 2 displays the relative increase of throughput for several message sizes due to multiple flitbuffers in a MoC(4,2) network. It shows that many communication loads modestly benefit from a larger buffer space. The highest measured gain of throughput equals to 14%. Naturally, the larger number of flitbuffers has the greatest impact on the *hotspot* and *hotregion* loads, as their hotspot areas cause high contention. For small message sizes (i.e. 128 bytes),

none of the communication loads does improve. This is because not enough network traffic is generated to cause the contention that is needed to benefit from the extra flitbuffers.

As can be seen from Table 2, the performance results of the multi-flitbuffer configurations are less predictable than one would expect. For some instances of the communication loads, we even measured a decrease of throughput when adding flitbuffers. This unexpected behaviour is due to a number of effects. The multiple flitbuffers cause channels to be unblocked earlier, enabling new packets to enter the network and thereby increasing the network traffic. Subsequently, the header stall delays that packets encounter within the network will change due to the effects of the extra network traffic and the enlarged buffer space: packet headers are potentially more often blocked because of the higher network traffic, but have a shorter mean stall time due to the larger number of flitbuffers. This change of header stall delays means that the order in which packets arrive at and are handled by routers will change as well. If such a change of packet handling order results in a delay of the packets that are on the critical path of the application load, then the overall throughput may decrease. This critical path can be formed by synchronizations, either at the operating system level (e.g. acknowledgements) or at the application level.

To illustrate the above, consider Figure 5. It shows a routing scenario at two routers *R1* and *R2* in which a packet *A* has to be routed in western direction, while two other packets *B* and *C* need to be routed to the north. The packets *A* and *B* will contend for the eastern input channel at *R1*, and the packets *B* and *C* will contend for the northern output channel at *R2*. It is assumed that router *R1* routes packet *A* first and that packet *C* is on the critical path of the application. The latter implies that the transmission of packet *C* forms the critical path delay in this example. This scenario allows two different packet handling orders. If, due to the lack of free flitbuffers, packet *B* is still blocked before router *R1*, then *R2* routes packet *C* first. After this packet has left *R2*,

Message size (bytes)	Number of flitbuffers	*uniform* (%)	*hotspot* (%)	*hotregion* (%)	*partner* (%)
128	4	0	0	0	0
	16	0	0	0	0
	64	0	0	0	0
512	4	0	-3.5	0.4	0.6
	16	0	7.8	2.9	2.6
	64	0	6.4	6.1	2.7
2048	4	0.8	-0.5	0.9	0.1
	16	4.6	6.3	3.9	0.7
	64	7.2	7.5	5.2	0.3
8192	4	-1.2	-3.9	1.1	0
	16	2.5	13.9	3.6	0.2
	64	1.5	9.8	5.6	-0.1

Table 2. Relative increase of throughput due to multiple flitbuffers.

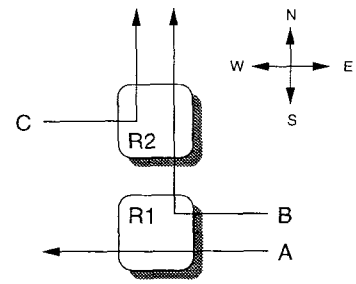

Figure 5. Different packet handling orders.

packet *B* may be routed at the moment enough flitbuffers become free. In this scheme, packet *B* does not interfere with the packets on the critical path (packet *C* in this example). On the other hand, if we add more flitbuffers at the routers then this would allow packet *B* to arrive first at router *R2*. In that case, packet *C* has to wait for packet *B* and is blocked at router *R2*. Now, the critical path delay has been increased by the transmission delay of packet *B* from router *R2*. Hence, this order of packet handling will result in a decrease of overall throughput.

As communication in our loads is synchronous, the synchronizations at the operating system level form an essential part of the critical path: a node cannot continue sending until it has received the acknowledgment of the previous message. To demonstrate the influence of changes in the order of packet handling on these synchronizations, the *hotspot* load will be subject to closer examination (as this load shows the most irregular behaviour). For *hotspot* using a message size of 512 bytes, Figure 6 shows a histogram of the cumulative periods of time that nodes have been waiting for acknowledgements. Each counted period of time correlates with one of the 256 nodes in the MoC(4,2) network. The histogram is drawn in a continuous shape to enhance the readability.

From the histogram can be seen that the peaks for 4 flitbuffers are shifted slightly towards the right with respect to the peaks for one flitbuffer. This implies that the network latencies of acknowledgements are generally higher in the configuration of 4 flitbuffers than in the single-flitbuffer configuration. Evidently, this contributes to the decrease of throughput for 4 flitbuffers (see Table 2). Moreover, the histogram also shows that the smallest latencies are achieved with 16 flitbuffers, followed by the configuration of 64 flitbuffers. This corresponds with the results of Table 2.

To investigate the reason behind the increased latency of the acknowledgements in the case of 4 flitbuffers, Figure 7 gives a more microscopic view of what happens within the network. It shows the differences in the average time that packet headers are blocked within the 16 Clos clusters of the MoC(4,2) for the configuration of 4 flitbuffers compared to the single-flitbuffer configuration. Although in many clusters a decrease of header stall delay is measured

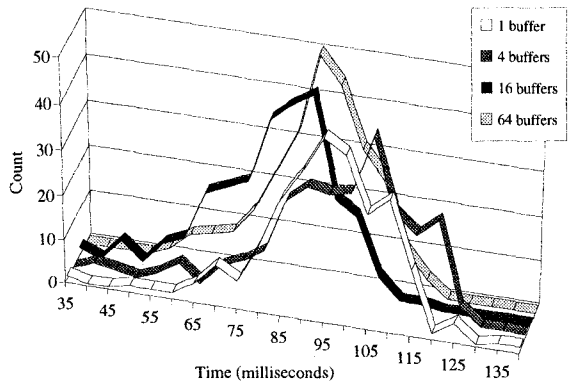

Figure 6. Histogram of the cumulative time that nodes waited for acknowledgements in *hotspot* on a MoC(4,2) using messages of 512 bytes.

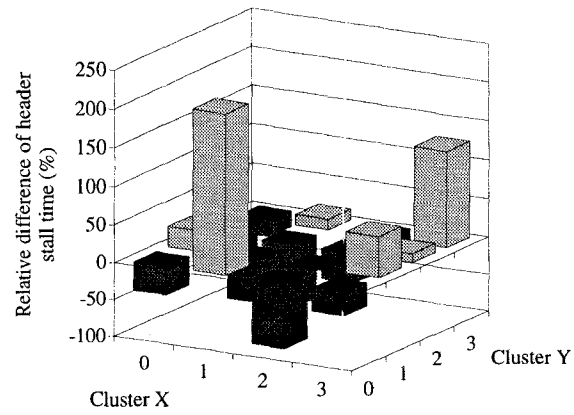

Figure 7. The relative difference in average header blocktime within a certain cluster of the MoC(4,2) network. These results are for *hotspot* with messages of 512 bytes and 4 flitbuffers and are relative to the results of the single-flitbuffer configuration.

(dark bars), the stall delays in several other clusters have increased considerably (light bars). One cluster in specific (the big peak at the front) is forming a large bottleneck, as we measured an increase of roughly 210% for this cluster. Apparently, the larger number of flitbuffers creates new or amplifies existing hotspots within the network. These hotspots may on their turn affect a large quantity of packets, including those on the critical path of the application (e.g. acknowledgements). This can subsequently result in a decrease of overall throughput, as shown in Table 2.

4.2. Routing strategies

The non-adaptive routing strategy of the node routers is reasonable simple to implement. Nevertheless, this strategy might not fully exploit the potentials of the layered mesh structure in the Mesh of Clos(4,2) network. Adaptively routing the packets to one of the four layered meshes may result in a better utilization of network resources. By doing so, the packets within a message can arrive out of order at the destination, as they may travel through distinct mesh-layers with potentially different latencies. Therefore, additional support 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 communication loads. To model the reconstruction of messages, an extra latency has been added for each message receipt. The routing within the 2D meshes is still performed deterministically. So deadlock prevention is not necessary as packets cannot switch from mesh-layer during their journey and thus no "inter-layer" dependency cycles can be formed.

The first two adaptive routing strategies that have been examined, are *Random* and *RoundRobin*. They route packets unconditionally, which means that they do not examine the state of the output channels. *Random* randomly selects a channel to a layer and *RoundRobin* selects a channel in round robin fashion. Without contention, the selected channel by *RoundRobin*, which is the least recently used channel, will have the greatest chance of being idle. In the third strategy, called *IdleFixed*, packets are routed along an idle channel. If more idle channels are available, then one is selected by random. Are all channels busy, then this strategy falls back on the original deterministic scheme (see Section 2.1). Finally, the *IdleRnd* strategy is similar to *IdleFixed* with the difference that a channel is randomly selected if there are no idle channels available.

Table 3 gives an overview of the communication performance effects due to adaptive routing, as compared to the performance of the original non-adaptive routing strategy. It contains the average throughput gain for each communication load, the overall average increase of throughput and the standard deviation of this overall average. Furthermore, the last column indicates whether a routing strategy is statistically equal to ($\equiv$), more effective ($\uparrow$) or less effective ($\downarrow$) than non-adaptive routing when using a confidence interval of 98%. All numbers are percentages and have been averaged over a wide range of message sizes.

The results clearly show that *Random* and *RoundRobin* are the least effective adaptive routing strategies. For most non-uniform communication loads, they actually decrease the throughput. Statistically, the performance of *Random* is not significantly different from that of the non-adaptive strategy. This is caused by the irregular performance behaviour of the *Random* strategy, as illustrated by its large standard deviation. So besides the many throughput decreases, there were measured quite a few increases of throughput.

The *RoundRobin* strategy, on the other hand, cannot com-

Routing strategy	*uniform* (%)	*hotspot* (%)	*hotregion* (%)	*partner* (%)	Overall av.(%)	σ	$\equiv/\uparrow/\downarrow$
Random	0.2	-1.1	-2.1	1.1	-0.5	4.0	$\equiv$
RoundRobin	0.4	-1.7	-0.2	-1.3	-0.7	1.7	$\downarrow$
IdleFixed	4.7	1.5	1.4	1.9	2.4	3.0	$\uparrow$
IdleRandom	2.8	2.6	0.5	3.2	2.3	3.2	$\uparrow$

Table 3. Average increase of throughput due to adaptive routing for the different types of communication loads.

pete with the non-adaptive scheme. Within the given interval of confidence, its communication performance is inferior to that of non-adaptive routing.

IdleRand and *IdleFixed* achieve the highest average throughputs for all communication loads. The overall average performance of both strategies is nearly identical, with *IdleFixed* performing slightly better. Using the confidence level of 98%, it can be concluded that these adaptive routing strategies are more effective than the non-adaptive scheme. Despite this, the throughput improvements are still marginal. Judging from these results, it might not be worthwhile to add extra complexity to the hardware or to the software in order to support adaptive routing at the node routers.

5. Conclusions

In this paper, we presented an evaluation study of wormhole-routed Mesh of Clos networks. This type of network, which combines the mesh and Clos topologies, addresses the trade-off between realizability and efficiency. Simulation experiments with different types of communication loads indicate that the Mesh of Clos networks with Clos structures of height 2 are potentially less prone to contention than flatter mesh-based networks. In circumstances of congestion, throughputs of up to 75% higher than those for a normal mesh network were measured. Moreover, it was found that Mesh of Clos network of height 1 generally suffers from high contention. For all applied communication loads, its communication performance is inferior to that of a normal mesh network.

It was shown that adding flitbuffers to router devices does not guarantee a better communication performance. Although many communication loads gain some benefit from a larger number of flitbuffers, the performance impact is not as predictable as one would expect. In several cases, a decrease of throughput was measured when enlarging the buffer space. This is due to the additional network traffic and the altered stall delays of packets that the extra flitbuffers cause. These effects may change the order in which packets are handled by the routers within the network, which subsequently can affect the delays of packets that are on the critical path of the application. This may, in some extreme cases, lead to a degraded overall communication performance.

For a Mesh of Clos network of height 2, which contains four independent layers of 2D meshes, we investigated several strategies to adaptively route packets to these mesh-layers. Of the four adaptive schemes that have been examined, only two obtain slightly higher throughputs compared to a straightforward non-adaptive strategy. These two strategies, which both make routing decisions based on the state of the communication channels, achieve an average throughput increase of roughly 2.4%. So, at first sight it might not be worthwhile to invest in the extra hardware or software necessary for adaptive routing.

Throughout this study, we have assumed that routing within the mesh structures is performed deterministically. However, applying (partially) adaptive routing within a mesh, combined with a strategy to adaptively route packets to a particular mesh-layer, may result in another gain of throughput. This issue requires additional research.

Acknowledgments

We would like to thank Marcel Beemster and Hugh McEvoy for their 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, Mar. 1953.

[2] W. J. Dally and C. L. Seitz. The torus routing chip. *Journal of Distributed Computing*, 1(3):187–196, 1986.

[3] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3(4):267–286, 1979.

[4] P. K. McKinley and C. Trefftz. Multisim: A simulation tool for the study of large-scale multiprocessors. In *Proc. of the 1993 Int. Workshop on Modeling, Analysis, and Simulation of Comp. and Telecom. Networks*, pages 57–62, Jan. 1993.

[5] B. Monien, R. Lüling, and F. Langhammer. A realizable efficient parallel architecture. In *Proc. of the 1st Int. Heinz Nixdorf Symposium, LNCS*, volume 678, pages 93–109, 1992.

[6] H. L. Muller. *Simulating computer architectures*. PhD thesis, Dept. of Comp. Sys, Univ. of Amsterdam, Feb. 1993.

[7] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, 26:62–76, Feb. 1993.

[8] G. Pfister and V. Norton. Hot spot contention and combining in multistage interconnection networks. *IEEE Transactions on Computers*, 34(10):943–948, Oct. 1985.

[9] A. D. Pimentel, J. van Brummen, T. Papathanassiadis, P. M. A. Sloot, and L. O. Hertzberger. Mermaid: Modelling and Evaluation Research in MIMD ArchItecture Design. In *Proc. of the High Performance and Networking Conference, LNCS*, volume 919, pages 335–340, 1995.

[10] C. L. Seitz. The cosmic cube. *Communications of the ACM*, 28:22–33, Jan. 1985.