

Empirical Performance Evaluation of Adaptive Routing in Necklace Hypercubes: A Comparative Study

S. Meraji, A. Nayebi, H. Sarbazi-Azad
Sharif University of Technology and IPM School of Computer Science
Tehran, Iran.

meraji.nayebi@ce.sharif.edu, azad@sharif.edu, ipm.ir

Abstract

The necklace hypercube network has recently been introduced as an attractive alternative to the well-known hypercube. Previous research on this network topology has mainly focused on topological properties and VLSI aspects of this network. In this paper, we propose some adaptive routing algorithm for the necklace hypercubes. The performance of necklace hypercubes using the proposed routing algorithm is then evaluated by means of simulation experiments. Experiments are realized under different working loads and for different network factors. Moreover, a comparison between the necklace hypercube and the well-known hypercube network is conducted. The comparison has been done considering implementation constraints including constant pin-out and constant bisection band-width constraints.

Keywords: Multicomputers, Hypercubes, Necklace Hypercubes, Performance evaluation, Simulation, Pin-out, Bisection bandwidth.

1. Introduction

A large number of interconnection networks have been proposed and studied for highly parallel distributed-memory multicomputers [2, 4, 11]. Among them the hypercube has been one of the most famous ones having many desirable properties such as logarithmic diameter and fault-tolerance. It is not, however, hardware scalable, i.e. when adding some few nodes to it, we must duplicate the network size to reach the next specified network size. Other drawback with hypercubes was reported by Patel et al. [7] when considering VLSI layout. They showed that the minimum number of tracks for VLSI layout of an n -cube (n -dimensional hypercube) using a one-dimensional implementation has an order of network size. Their investigation revealed that for an example network of 1k processors (a 10-cube), at least 687 tracks are required for VLSI implementation. The *necklace hypercube*, introduced in [5], is a new interconnection network based on the hypercube network. While preserving most of properties of the hypercube, it has some other desirable properties such as scalability and efficient VLSI layout that make it more attractive than the hypercube network [5].

Routing algorithms specify the path to be taken by the messages sent from a source node to the destination node. A routing algorithm must provide low-latency message delivery, be aware of deadlock, livelock, and starvation, and be able to work well under various traffic loads. Early

practical multicomputers have adopted deterministic routing, where messages with the same source and destination addresses always take the same network path. This form of routing has been popular because it requires a simple deadlock-avoidance algorithm with a minimal number of virtual channels, greatly simplifying router implementation. However, messages can not use alternative paths to avoid congested channels and, thus, reduce their latency. Fully adaptive routing has often been suggested to overcome this limitation by enabling messages to explore all available paths in the network.

Wormhole switching is one of the popular switching techniques [3]. In this technique, a message is divided into a sequence of flits (the smallest fixed-size units of data). If a communication channel transmits the first flit (also called the *header* flit) of a message, it must transmit all the remaining flits of the same message before transmitting flits of another message. At any given time, the flits corresponding to a message occupy contiguous channels in the network. Thus, the message latency is proportional to the sum of the number of cycles spent in waiting for suitable channels to route message flits, number of hops from the source node to the destination node, and message length.

In this paper, we use a simulation based environment to analyze the performance of necklace hypercube interconnection network using fully adaptive routing. To this end, we first propose a fully adaptive routing algorithm and implement it. We then evaluate the performance of necklace hypercube with different sizes and under various working conditions. We also compare the performance of the necklace hypercubes with their hypercube counterparts using adaptive routing and under different working conditions.

2. The Necklace Hypercube

The *necklace hypercube* is an undirected graph that is based on a hypercube by appending a necklace of processors (or nodes, interchangeably) to each edge. That is, besides connecting adjacent nodes (according to the base hypercube topology), we connect i -th dimension neighbors by an array of nodes, as a necklace. The necklace length may be fixed or variable for different edge necklaces. In the former, there are fixed number of nodes between each two adjacent neighbors on a necklace. The network is called *regular necklace hypercube*, and can be defined as (n, k) -RNH, where n is the number of dimensions and k is the necklace size. With fixed length

necklaces, however, scaling up the network is limited to specified network sizes indicated by n and k as $2^{n-1}(nk+2)$.

Figures 1(a) and 1(b) show examples of a regular and irregular necklace hypercube. Dark nodes are the nodes in the base hypercube and the grey nodes are the necklace nodes. The index of each necklace node in its necklace is shown inside the node. Edge number of each edge in the base hypercube is shown inside a grey rectangle on it which is based on i -th dimension edge of smaller base vertex. Let us now give a more formal definition of the necklace hypercube.

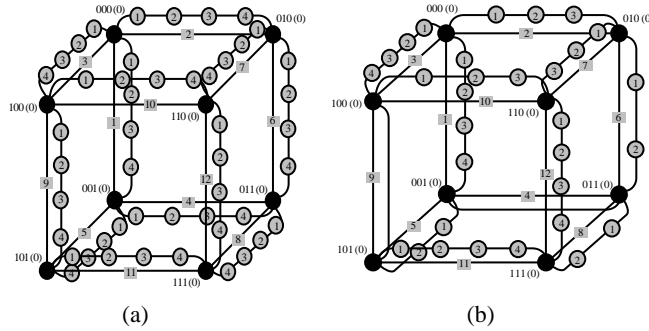


Fig. 1: a) A regular necklace hypercube with $n=3$ and $k=4$ b) An irregular necklace hypercube

3. Routing Algorithm

In order to have a deadlock-free routing algorithm in the necklace hypercube, we must use at least 2 virtual channels as follows [12]. We use two classes of virtual channels A and B. Virtual channels of class A are used when the message is at the source necklace; we use virtual channels of this class until we reach the first base vertex of the source necklace. Virtual channels of class B are used in the destination necklace; it means that when we enter the destination necklace, we use this class of virtual channels until we reach to the destination node. Each of these classes can be used for a deadlock-free routing algorithm in the base hypercube, e.g. e-cube routing. The minimum number of virtual channel sin each class is 1. Thus, we need at least 2 virtual channels per physical channel to implement a deadlock-free routing in necklace hypercubes.

According to Duato's methodology, since the base deadlock-free routing algorithm requires 2 virtual channels, we can have a fully adaptive deadlock-free routing algorithm in the necklace hypercube using at least 3 virtual channels, two of which used by the base routing algorithm and the remaining one used in any possible way that can brings the message closer to the destination node. We call the virtual channels used for base deadlock-free routing as *base virtual channels* and the remaining virtual channels as *adaptive virtual channels*.

When there are more than 3 virtual channels per physical channel, the network performance is maximized when the extra virtual channels are added to adaptive virtual channels. Thus, with V virtual channels per physical channel the best performance is achieved when we have $V-2$ adaptive virtual channels and two 2 base virtual channels. This is of course for routing in necklaces; in the

base hypercube we can use Duato's fully adaptive routing algorithm [4] which uses $V-1$ adaptive virtual channels and 1 virtual channel for deterministic deadlock-free routing.

Figure 2 shows the routing algorithm between nodes $u = (b_1, d_1, i_1)$ and $v = (b_2, d_2, i_2)$ in pseudo code. Note that the algorithm contains three main steps:

1. Move towards the nearest base neighbor using any of the $V-2$ adaptive virtual channels. If all $V-2$ virtual channels are busy use the virtual channel of class A from base virtual channels to move toward the nearest base neighbor in the source necklace.
2. Move towards the nearest neighbor of the destination using fully adaptive Duato's routing algorithm in hypercube with $V-1$ virtual channels [4]. If all $V-1$ virtual channels are busy use the 1 remaining virtual channel with e-cube routing in the base hypercube.
3. Now the current node is one of the base vertices of the destination necklace. Move towards the destination node using any of the $V-2$ virtual channels. If all $V-2$ virtual channels are busy use the virtual channel of class B from base virtual channels to move toward the destination node in the destination necklace.

```

Routing ( $u, v$ ) //  $u = (b_1, d_1, i_1)$  and  $v = (b_2, d_2, i_2)$  are the source
and destination.
 $u_1 = b_1, u_2 = b_1 \vee Location(d_1)$ ;
/* Location is a function that calculates the difference bit for the
other extreme base vertex of a necklace. */
 $v_1 = b_2, v_2 = b_2 \vee Location(d_2)$ ;
 $CC = \min(difference(i_1, u_1), difference(i_1, u_2))$ ;

If there is any free virtual channel ( $VC_1$ ) in  $V-2$  adaptive virtual
channels then set  $vc = vc_1$ 

else set  $vc = VC_2$  (class A base virtual channels);
Send message through physical channel  $CC$  and virtual channel  $vc$ ;
Set the current node  $C$ ;
 $HD_1 = H(C, v_1), HD_2 = H(C, v_2)$ ;

If ( $HD_1 < HD_2$ ) then Goto  $v_1$  using Duato's routing algorithm

else Goto  $v_2$  using Duato's routing algorithm

If there is any free virtual ( $VC_1$ ) channel in  $V-2$  adaptive virtual
channels then set  $vc = VC_1$ 

else set  $vc = VC_2$  (class B virtual channels);
Send message through physical channel  $i_2$  and virtual channel  $vc$ ;
End.

```

Fig. 2: fully adaptive routing algorithm

4. Simulation Results

The XMulator [6] was used to implement the proposed routing algorithm for the necklace hypercube and to perform simulation experiments. XMulator is an event-based flit-level simulator which can provide various detailed results after each run.

In the simulator program (based on XMulator) developed for the sake of this research, virtual channels are demand

time-multiplexed on a physical channel. Moreover, only virtual channels that have messages to transmit use the physical channel in a round-robin manner. In our simulation experiments, the delays for switching and routing are ignored and only delay of physical channels is considered (1 cycle). Different number of virtual channels (for different network sizes) are used for each physical channel and the injection channels. Message consuming at the destination node is not bandwidth limited to avoid performance bottlenecks.

We consider three main groups of necklace hypercubes for our simulation experiments: large, medium and small networks. The large networks (which have more than 1000 nodes) are (7,3)-RNH and (7,8)-RNH, the medium networks (with about 100-300 nodes) are (4,3)-RNH and (4,8)-RNH, and the small networks (with 15-50 nodes) are (2,3)-RNH and (2,8)-RNH. We consider the effect of different number of virtual channels and different message sizes in each of these networks. The performance metrics calculated and reported here is the *average message latency*. It is the average of all message latencies transmitted over the network for at least 1000000 messages. The first 10% of messages are excluded from the calculation to avoid warm up effects.

Figure 3 (a-f) shows the effect of the number of virtual channels on the overall performance when the message length is assumed to be 64 flits. As can be seen in the figure, increasing the number of virtual channels results in a better performance. This is a rational effect because when we increase the number of virtual channels, there are some extra ways for the messages to continue their journey to the destination nodes, and hence the generation rate at which network saturation happens is increased. Another important point is the small effect of virtual channels when we use more than 2 virtual channels. In low traffics, the fully adaptive routing algorithm in the base hypercube works well and increases the saturation generation rate, but in high traffics the effect of necklaces concurs it and increasing virtual channels per physical channel can not help noticeably.

Let us now consider the effect of the message length on the overall performance. To this end, we fix the number of virtual channels per physical channel in our simulation experiments. For large networks (Figures 4.a, 4.b) 8 virtual channels per physical channels, for medium-sized networks (Figures 4.c, 4.d) 6 virtual channels per physical channel, and for small networks (Figures 4.e, 4.f) 2 virtual channels per physical channel are used. As can be seen in the figures, increasing the message size increases the average message latency and results in earlier saturation for the network. Note that the smaller networks work better in high traffic regions. Another interesting result that can be achieved from figures 3 and 4 is the effect of the necklace size for networks with similar base hypercube. As can be seen in all these figures, increasing the necklace size reduces the overall performance of the network. This is a rational result because when we increase the necklace size the diameter of the network

increases, and hence the average time to deliver the messages to their destination nodes will be increased.

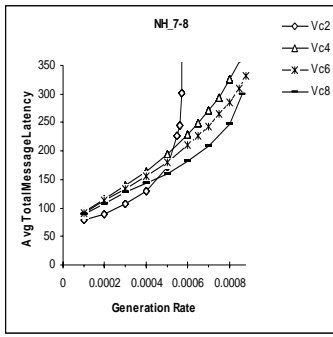
Let us now investigate the effect of network configuration when a fixed network size is considered. Again, we consider three sizes: large, medium, and small. The large networks considered are (7,2)-RNH, (6,5)-RNH, and (5,12)-RNH and simulation results are shown in Figure 5.a. The medium networks include (6,2)-RNH, (5,5)-RNH and (4,13)-RNH (Figure 5.b) and the small networks considered here include (4,1)-RNH, (3,4)-RNH, (2,11)-RNH (see Figure 5.c). As can be seen in the figures, increasing the dimension of the base hypercube results in a better performance. This is because for lower dimensional networks (assuming that the network size is fixed) the necklace size and network diameter grow, and therefore the average message latency increases. This can be seen better in cases where more than 10 nodes are used in necklaces.

Now, we compare the performance of the necklace hypercube with an equivalent hypercube network. We first compare the networks with equal number of nodes without considering any technological or implementation constraint. To do so, we consider 3 different networks the 10-dimensional hypercube, (6,5)-RNH and (7,2)-RNH, all with 1024 nodes. Figure 6.a shows the simulation results of these networks when the message size is fixed at 32 flits and the number of virtual channels per physical channel is 8. As can be seen in the figure, the hypercube performs better than the other two equivalent networks. One reason is the excessive number of channels existing in the hypercube which enable the network to tolerate higher traffic rates. Moreover, comparing the average distance of Necklace hypercube [5] and a binary $n/2$ -cube (i.e. $n/2$) reveals a big difference especially when the necklace size, k , is big.

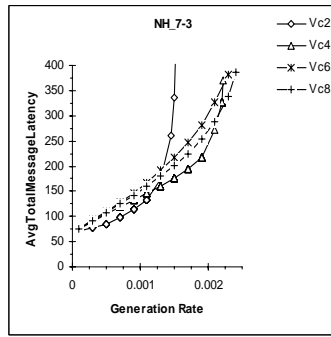
Similar results are shown in Fig.6.b for medium sized networks. Here, the considered networks are the 8-dimensional hypercube, (4,7)-RNH, (4,8)-RNH, (5,2)-RNH and (6,1)-RNH, all with a size around 256 nodes. Fig.6.c shows the results for the small networks: the 6-dimensional hypercube and (3,5)-RNH, each with 64 nodes. In all these three figures, the hypercube has shown a better performance. This is because of the higher network bandwidth and smaller diameter of the hypercube compared to their necklace counterparts.

When an interconnection network is implemented on a chip or over a board, the bisection bandwidth has the most important effect on the performance. Thus, many studies [8, 9, 10] consider a constant bisection bandwidth when comparing two equivalent networks (with the same number of nodes). Also pin-out constraint has an important impact on the performance when an interconnection network is implemented by wiring between chips, boards, and cabinets [1].

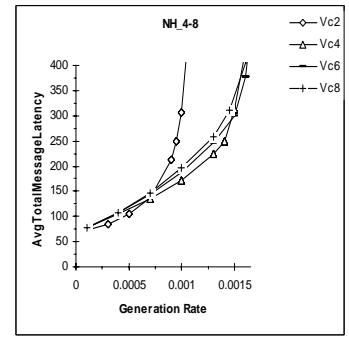
Comparison results under constant bisection bandwidth and node pin-out reveal that the necklace hypercube can perform much better than its hypercube counterpart [12].



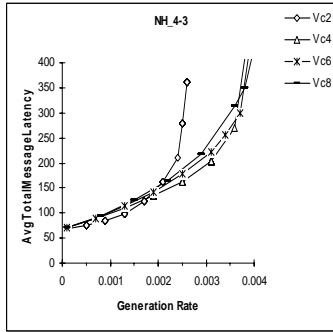
a



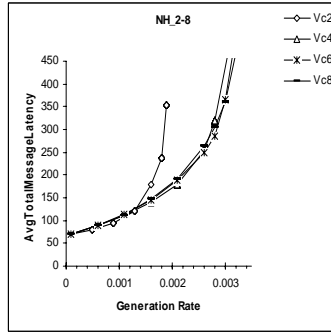
b



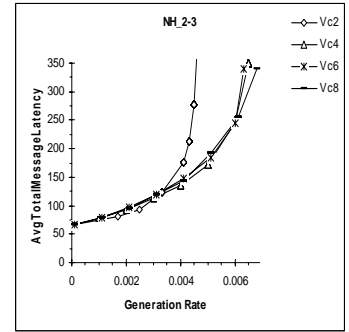
c



d

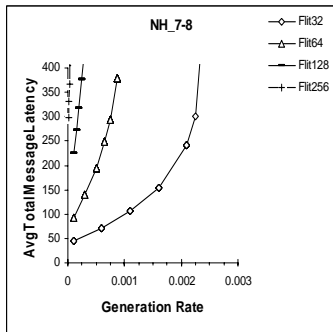


e

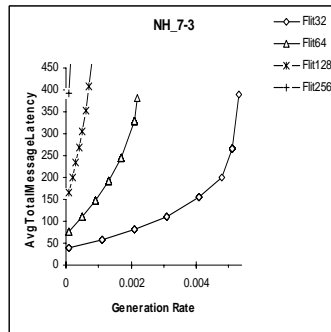


f

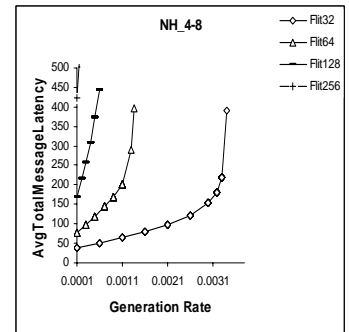
Fig.3: The average message latency in different necklace hypercubes, for different number of virtual channels per physical channel, as a function of message generation rate when message length is fixed at 64 flits.



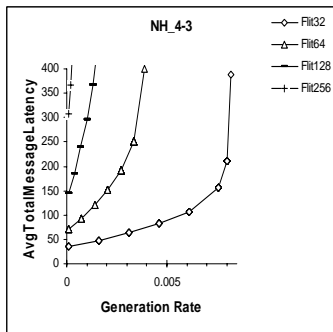
a



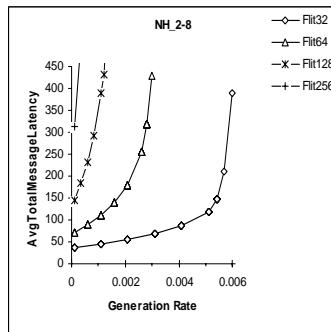
b



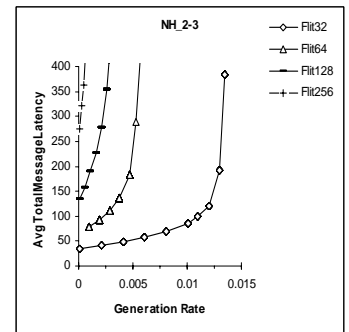
c



d

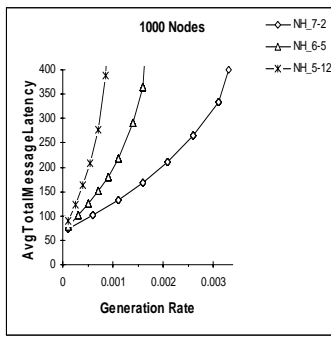


e

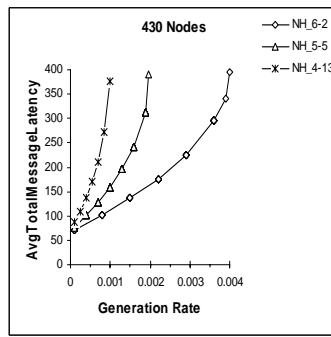


f

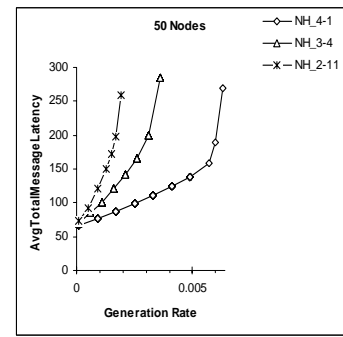
Fig.4: The average message latency in different necklace hypercubes, for different message sizes, as a function of message generation rate.



a

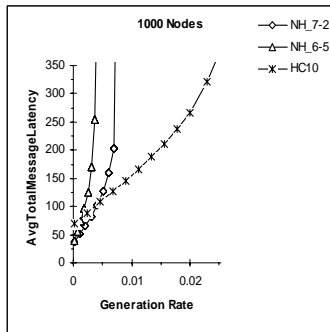


b

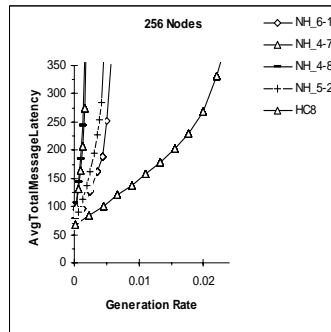


c

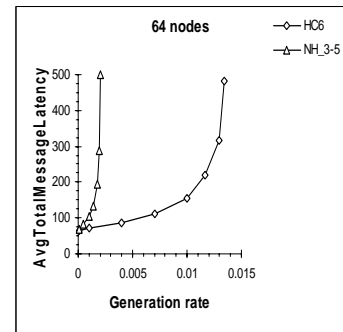
Fig.5: The effect of necklace size on network performance.



a



b



c

Fig.6: Comparison of different necklace hypercubes with their equivalent hypercubes.

5. Conclusion

In this paper, we proposed a fully adaptive deadlock-free routing algorithm for necklace hypercube interconnection networks. The algorithm is based on Duato's fully adaptive routing in hypercube for routing within the base graph but can be replaced with any other deadlock-free routing algorithm. The performance of the proposed routing algorithm was evaluated through simulation experiments. We then compared the performance of necklace hypercubes with their equivalent hypercubes without and with technological constraints. Two implementation constraints namely the constant bisection bandwidth and constant pin-out constraints were used when comparing 3 classes of large, medium and small networks. Our analysis showed that under no implementation constraints, the hypercube network performs better than the necklace hypercube while the opposite conclusion achieved when implementation constraints were considered.

References

- [1] Agarwal A., Limits on interconnection network performance, *IEEE Trans. Parallel Distr. Systems*, vol.2, no.4, pp. 398-412, 1991.
- [2] Dally W. J., Performance analysis of k-ary n-cubes interconnection networks, *IEEE Trans. Comput.*, vol.39, no.6, pp. 775-85, 1990.
- [3] Dally W. J., and Seitz C., Deadlock-free message routing in multiprocessor interconnection networks, *IEEE Trans. Computers*, vol. C-36, no. 5, pp. 547-553, 1987.
- [4] Duato J., Yalamanchili C., Ni L., *Interconnection Networks: an engineering approach*, IEEE computer society press, 1997.
- [5] Monemizadeh M., and Sarbazi-Azad, H., The necklace hypercube: a well scalable hypercube-based interconnection network for multiprocessors, *ACM SAC 2005*, pp.729-733, 2005.
- [6] Nayebi A., Sarbazi Azad, H., Shamaei, A., Meraji, S., XMulator: An Object Oriented XML-Based Simulator, accessible at <http://www.XMulator.org>, 2006.
- [7] Patel A., Kusalik A. and McCrosky C., Area-Efficient VLSI Layout for Binary Hypercubes, *IEEE Trans. on Computers*, vol. 49, no. 2, 2000.
- [8] Rezazad M., Sarbazi-Azad H., A Constraint-Based Performance Comparison of Hypercube and Star Multicomputers with Failures, *19th International Conference on Advanced Information Networking and Applications (AINA 2005)*, pp. 841-846, 2005.
- [9] Sarbazi-Azad H., Ould-Khaoua M., Constraint-based performance analysis of k-ary n-cube networks Using a new cost model, *International Journal of Computers and Their Applications*, vol.12, pp. 83-92, 2005.
- [10] Sarbazi-Azad H., Constraint-based performance comparison of multi-dimensional interconnection networks with deterministic and adaptive routing strategies, *Journal of Computers and Electrical Engineering*, vol. 30, pp.167-82, 2004.
- [11] Tseng Y. C., Chen Y. S. and Chang C. J., Congestion-Free, Dilation-2 Embedding of Complete Binary Tree in Star Graphs, *Networks*, vol. 33, no. 3, pp. 221-231, 1999.
- [12] Meraji S., Sarbazi-Azad H., Nayebi A., Message routing and performance issues in necklace hypercubes, Technical Report, School of Computer Science, IPM, Tehran, Iran, 2006.

