An Introduction to Interconnection Networks and Xmulator

Sina Meraji
McGill University, Montreal, Quebec, Canada
smeraj@cs.mcgill.ca

School of Computer Science

Abstract. Today, Communication and interconnection between different components of a system limit the performance of most digital systems. In a high-end system today, most of the power is used to drive wires, and most of the clock cycle is spent on wire delay, not gate delay. As technology improves, memories and processors become smaller, faster, and less expensive. The speed of light, however, remains unchanged. The pin density and wiring density that govern interconnections between system components are scaling at a slower rate than the components themselves. Also, the frequency of communication between components is lagging far beyond the clock rates of modern processors. These factors combine to make interconnection the key factor in the success of future digital systems.

1 Introduction

One way for processors to communicate data is to use a shared memory and shared variables. A realistic assumption is that each processor has its own private memory and data communication takes place using message passing via an interconnection network. Interconnection networks are beginning to replace buses as the standard system-level interconnection. They are also replacing dedicated wiring in special-purpose systems as designers discover that routing packets is both faster and more economical than routing wires [8]. We will introduce some of the most important interconnection networks in section 2. *Routing* and *Switching* are two main concepts of computer networks (also interconnection networks) that we describe them here.

Routing algorithms specify the path to be taken by the messages sent from the source nodes to the destination nodes. A routing algorithm must provide low-latency message delivery, be aware of deadlocks, starvation, and livelock, and be able to work well under various traffic loads. The message latency is dependent to the number of hops taken by a message, hence minimal routing algorithms are usually considered as per which a message always moves closer to its destination with each hop taken; another advantage of minimal routing is that livelocks are avoided. Starvation can also be avoided by allocating resources

such as channels and buffers in a proper manner. Ensuring deadlock-freedom is a difficult issue and must be taken care of when designing the routing algorithm [14]. A simple routing algorithm is shown in section 3.

A switching mechanism determines how and when the router switch is set; that is, the input channel is connected to the output channel selected by the routing algorithm. In other words, the switching mechanism determines how network resources are allocated for message transmission [8]. One of the most popular switching techniques is Wormhole switching [6] in which 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. As a result, 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 [14].

The rest of the paper is as follows in section 2 we introduce the most famous interconnection networks. Section 3 shows two simple routing algorithms for heypercube and necklace hypercube interconnection networks. In section 4 Xmulator introduce as simulation platform for interconnection networks. Finally section 5 concludes this report.

2 Interconnection Networks

A large number of interconnection networks have been proposed and studied for highly parallel distributed-memory computers [2–5, 7, 9, 11–13, 18, 19]. Here we introduce some of the most important of them. Before introducing the network topologies let's define some important properties of interconnection networks.

Node degree: Number of channels connecting that node to its neighbors.
Diameter: The maximum distance between two nodes in the network.
Regularity: A network is regular when all the nodes have the same degree.
Symmetry: A network is symmetric when it looks alike from every node.

2.1 Fully Connected

This is one of the most power full interconnection networks in which each node is connected to all other nods via a direct path. An instance of this network is shown in figure 1.

If we have N nodes, node degree is equal to $N-1$ and the total number of edges in this network is equal to $N(N-1)/2$. Fully Connected is a regular and symmetric network with a diameter equal to 1. Although it is a very power full network the construction cost is so much when we have large number of nodes [20].

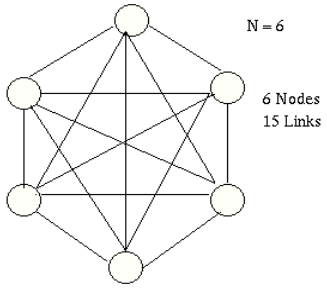


Fig. 1. An example pf fully connected interconnection network.

2.2 Mesh and Torus

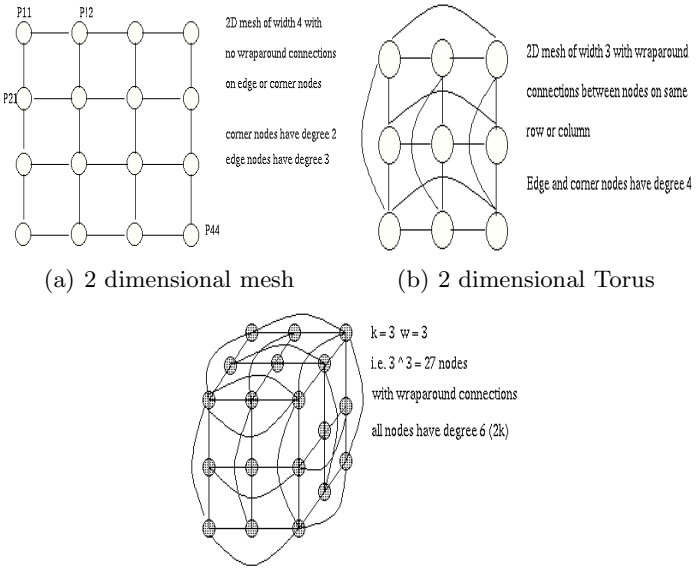
In a *Mesh* network nodes are arranged in a k dimensional lattice and we have a total number of w^k nodes. It is neither regular nor symmetric. An example of 2 dimensional mesh with width equal to 4 is shown in figure 2-a. A special case of mesh network in which we have a direct path (wraparound) between the first and the last node of each array is called *Torus*. Torus is a regular and symmetric network. Two instances of 2 dimensional and 3 dimensional Tours topologies are shown in figure 2-b and 2-c [20].

2.3 hypercube (binary n-Cube)

One of the most power full and well structured interconnection networks is *n-Cube* in which 2^k nodes are arranged in a k dimensional hypercube. The nodes are numbered $0, 1, \ldots, 2^k - 1$ and two nodes are connected if their binary labels differ by exactly one bit. Hypercube is regular and symmetric network with a diameter equal to k. an instance of 1, 2 and 3 dimensional hypercubes are shown in figure 3-a, 3-b and 3-c.

2.4 Star Graph

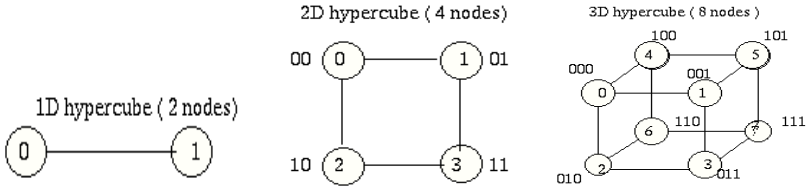
The star graph is an attractive alternative to the hypercube, and compares favorably with it in several aspects. Let V_n be the set of all $n!$ permutations of



(a) 2 dimensional mesh (b) 2 dimensional Torus

(c) 3 dimentional Torus

Fig. 2. Different examples of Mesh network.



(a) 1 dimensional (b) 2 dimensional (c) 3 dimentional

Fig. 3. Different examples of hypercube topology.

symbols $1, 2, 3, \ldots, n$. For any permutation $v \in V_n$, if we denote the ith symbol of v by $v(i)$, then v can be written as $v(1)v(2)\ldots v(n)$.
A *star graph* on n symbols, $S_n = (V_n, E_n)$, is an undirected graph with $n!$ nodes, where each node v is connected to $n-1$ nodes which can be obtained by interchanging the first and ith symbols of v, i.e. $[v(1)v(2)\ldots v(i)v(i{+}1)\ldots v(n), v(i)v(2)\ldots v(i{-}1)v(1)v(i+1)\ldots v(n)] \in E_n$, for $2 \leq i \leq n$. We call these $n-1$ connections as dimensions. Thus each node is connected to $n-1$ nodes through dimensions $2, 3, \ldots, n$. S_n is also called an n-star or an n-dimensional star. Figure 4 shows a S_4 [1].

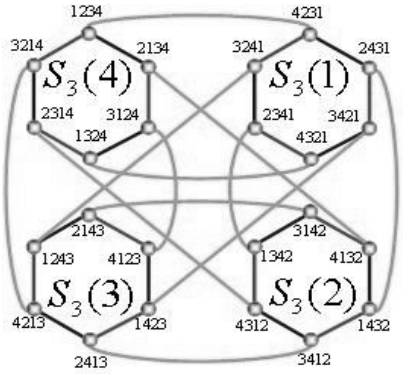


Fig. 4. An instance of Star graph (s_4).

3 Routing Algorithm

Routing is the process of selecting paths in a network along which to send data or physical traffic. Routing directs forwarding, the passing of logically addressed packets from their source toward their ultimate destination through intermediary nodes; typically hardware devices called routers, bridges, gateways, firewalls, or switches. Ordinary computers with multiple network cards can also forward packets and perform routing, though with more limited performance [21]. Here we show a famous routing algorithm for hypercube which named e-Cube and also a routing algorithm for Necklace hypercube which is based on e-Cube routing.

For n-dimensional hypercubes, e-Cube routing produces deadlock-free paths between sources and destinations. This algorithm is described in Figures 5, where FirstOne() is a function that returns the position of the first bit set to one, and Internal is the channel connecting to the local node. The first sentence in algorithm computes the offset from the current node to the destination node. This offset is the value carried by the header when relative addressing is used. If the offset equals to 0 it means that the current node is the destination node, otherwise we send the packet on first link which has a non-zero value in the offset [8].

4 Xmulator

Simulation is the most effective way to evaluate the performance of an interconnection network. *XMulator*, is an object-oriented listener-based simulation environment for evaluating multicomputer interconnection networks. The simulator involves a toolbox of various network topologies, routing algorithms, switching techniques, and flexible router models. Xmulator uses listenerbased integration

```
Algorithm: e-Cube Routing for Hypercubes
Inputs: Current & Dest //Addresses of current node and destination node
Output: Selected output Channel
Procedure:
offset:= Current ⊕ Dest;
if offset = 0 then
        Channel := Internal;
else
        Channel := FirstOne(offset);
Endif
```

Fig. 5. e-Cube routing for hypercube.

methodology, which has a great impact on extensibility of the system. Mixed-mode event processing improves the performance of the simulator. By decoupling individual parts of the code, XMulator enables independent code development and creates a flexible and extensible environment for different aspects of network design. This simulator uses XML format to define network topologies, input parameters, and outputs reports providing a high level of flexibility. To the best of the designers' knowledge, it is the first simulator enabled to simulate any arbitrary interconnection topology under different working conditions including in the presence of faults [17]. Two main features of Xmulator in comparison with other simulators are listener-based integration and multilayered architecture which we described here briefly.

One of the main features of Xmulator is Listener-based integration which solves the problem of two-way dependency that leads to a non-extensible design. For example, consider a Queue component developed by developer A and a Generator inserts the objects to the Queue and developed by developer B. Each one must know the design of the other one because Generator must provide a link to Queue and awake it when a new object is ready to be inserted, and on the other hand, Queue must know the architecture of Generator to accept the objects. In listener-based integration, Queue provides an event, say onObjectReceived, generally to everyone who may be interested in this event. Designer of the Generator does not know about the architecture of the other components that would potentially use it later. On the other hand, Queue knows the Generator and simply registers a method on the list of listeners of the event that is called every time a new object is received by Generator.

Xmulator also uses a multilayer architecture which is one of the most important differences with other simulators. In multilayer architecture each layer provides service to upper layers and depends on the lower layers. Using the listener-based integration helps the designers to build multi-layered architecture simply. The lowest layer consists of general tools like different continues and discrete distributions. The second layer consists of base components. We have some events, basic components, and a simulation engine in this layer which are essential for a simple simulation. The engine holds and manages a queue of events. The main role of the engine is processing of events sorted by the fire time. Execution of a event may generate some other new events. The lighter the engine structure is, the more modular and more extensible the architecture could be developed. The engine does not need to know the event types and their internal processes. It is sufficient to know which event belongs to which component and when it must be fired.

Events are sorted according to the firing time in apriority queue. The data structure to save this queue is so important and has a great impact on the performance of simulator. Arrays and linked lists have lower performance than AVL trees. In this simulator the red-black tree is used, a special case of an AVL tree, as the data structure for saving the data. Each data node of the tree contains a queue of events that have the same firing time. This is one of the main differences of the current simulator and previous simulators like DEVS in which arrays or linked lists are used as date structures.
Different layers could be defined at the top of this layer to use the features of this layer for simulation any specific applications. One of the most important layers which is defined on the top of the based component layer is interconnection network layer. In this layer we can implement different types of regular and semi-regular interconnection network topologies such as meshes, hypercubes and necklace hypercubes. Different routing algorithms for different interconnection topologies can also be implemented in this layer.

5 Conclusion and future work

In the current report, we introduced the main basics of interconnection networks. The main topologies of interconnection networks are introduced. We also show the e-cube routing algorithm for hypercube topology which is one of the famous topologies in this area. Necklace hypercube and Stretched hypercube are two subsets of hypercube topology which has a better VLSI layout than hypercube and as a result they can be constructed with a lower cost. In the future work, we use atom3 which is a tool for multi-paradigm modelling to model different instances of hypercube topology. A comparison between these topologies will be shown in the final report.

References

1. Kiasari A., Sarbazi-Azad H., and M. Rezazad. Performance comparison of adaptive routing algorithms in the star interconnection network. In *HPC-Asia05: Proceedings of the 8th International Conference on High Performance Computing in Asia Pacific Region*, pages 257–264, 2005.
2. Pascal Berthomé, Afonso Ferreira, and Stéphane Pérennès. Optimal information dissemination in star and pancake networks. *IEEE Trans. Parallel Distrib. Syst.*, 7(12):1292–1300, 1996.
3. Chi-Chang Chen and Jianer Chen. Nearly optimal one-to-many parallel routing in star networks. *IEEE Trans. Parallel Distrib. Syst.*, 8(12):1196–1202, 1997.
4. Chi-Chang Chen and Jianer Chen. Optimal parallel routing in star networks. *IEEE Transactions on Computers*, 46(12):1293–1303, 1997.
5. Yuh-Shyan Chen, Yu-Chee Tseng, and et. al. Embedding of congestion-free complete binary trees with dilation two in star graphs. 33(3):221–231, 1999.
6. W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Comput.*, 36(5):547–553, 1987.
7. K. Day and A. Tripathi. A comparative study of topological properties of hypercubes and star graphs. *IEEE Trans. Parallel Distrib. Syst.*, 5(1):31–38, 1994.
8. Jose Duato, Sudhakar Yalamanchili, and Ni Lionel. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.
9. Satoshi Fujita. Neighborhood information dissemination in the star graph. *IEEE Trans. Comput.*, 49(12):1366–1370, 2000.
10. Sidney W. Graham and Steven R. Seidel. The cost of broadcasting on star graphs and k-ary hypercubes. *IEEE Trans. Comput.*, 42(6):756–759, 1993.
11. M. C. Heydemann, J. Opatrny, and D. Sotteau. Embeddings of complete binary trees into star graphs with congestion 1. In *HICSS '95: Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS'95)*, page 546, Washington, DC, USA, 1995. IEEE Computer Society.
12. J. Jwo, S. Lakshmivarahan, and S. Dhall. Embedding of cycles and grids in star graphs. In *Proc. Second IEEE Parallel and Distributed.Processing Symp*, 1990.
13. Nigam M., Sahni S., and Krishnamurthy B. Embedding hamiltonians and hypercubes in star interconnection graphs. *Proc. Intl. Conf. Parallel Processing*, 3:340–343, 1990.
14. S. Meraji, A. Nayebi, and H. Sarbazi-Azad. Simulation-based performance evaluation of deterministic routing in necklace hypercubes. *aiccsa*, 0:343–350, 2007.
15. S. Meraji, H. Sarbazi-Azad, and A. Patooghy. Performance modelling of necklace hypercubes. *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8, 26-30 March 2007.
16. M. Monemizadeh and H. Sarbazi-Azad. The necklace-hypercube: a well scalable hypercube-based interconnection network for multiprocessors. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 729–733, New York, NY, USA, 2005. ACM.
17. A. Nayebi, S. Meraji, A. Shamaei, and H. Sarbazi-Azad. Xmulator: A listener-based integrated simulation platform for interconnection networks. *Modelling and Simulation, 2007. AMS '07. First Asia International Conference on*, pages 128–132, 27-30 March 2007.
18. S. Ranka, J. Wang, and N. Yeh. Embedding meshes on the star graph. *Supercomputing '90. Proceedings of*, pages 476–485, 12-16 Nov 1990.

19. Jang-Ping Sheu, Wen-Hwa Liaw, and Tzung-Shi Chen. A broadcasting algorithm in star graph interconnection networks. *Inf. Process. Lett.*, 48(5):237–241, 1993.
20. www.cm.cf.ac.uk/Parallel/Year2/section5.html.
21. www.wikipedia.org.