RC 20454 (May 7, 1996)
Computer Sciences/Mathematics

IBM Research Report

Trends in Parallel Machine Interconnection Networks

Howard Jay Siegel

School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285

Craig B. Stunkel

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

an edited version to appear in IEEE Computational Science and Engineering, Summer 1996

Trends in Parallel Machine Interconnection Networks

Howard Jay Siegel
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285
hj@purdue.edu

Craig B. Stunkel
IBM T. J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
stunkel@watson.ibm.com

The interconnection networks that provide communication among processors in large-scale tightly-coupled parallel machines continue to be a vital topic of research for computer architects. The demands placed on these networks have evolved over the last two decades, from SIMD machines that required the simultaneous transfer of data from each input to each output for a relatively small set of different communication paths (permutations), to the MIMD machines of today that need support for delivering varied and asynchronous traffic patterns. Interconnection networks can be categorized according to a number of criteria such as topology, routing strategy, and switching technique. They vary widely in their cost, fault tolerance, simplicity, partitionability, and aggregate bandwidth for local and non-local traffic patterns [9, 11, 14].

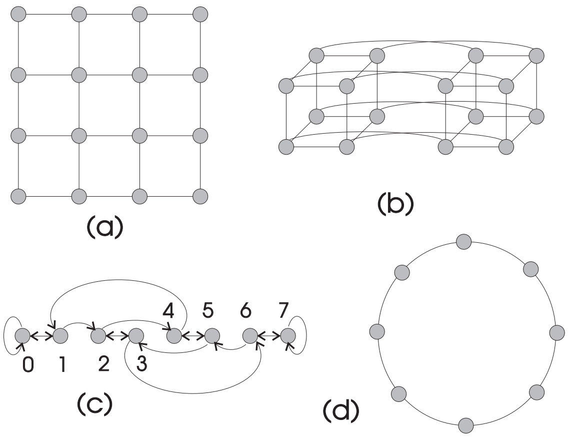

Figure 1: Direct topologies: (a) 2D mesh, (b) 4D hypercube, (c) shuffle-exchange, and (d) ring.

Topologies

Topology is the pattern in which the individual switching elements[1] (switches) of the network are connected to other switches and processors/memories (nodes). Connecting all nodes via a single non-blocking crossbar switch provides optimal performance. However, a crossbar to connect k nodes requires k I/O ports and has complexity that grows with k^2, making it impractical for large-scale parallel systems. Thus larger systems employ multiple smaller switches in direct or indirect networks. Direct topologies connect each switch directly to a node, while in indirect topologies at least some of the switches are connected only to other switches.

Figure 1 illustrates several direct networks, with each circle representing a node-switch combination. The 2D mesh of the 1970s' Illiac IV [3] is similar to the 1990s' 2D mesh of the Intel Paragon and the toroidal 3D mesh of the Cray T3D [6] and T3E.

Figure 2 displays some indirect networks, with circles representing nodes and rectangles indicating switches. The indirect cube network, often referred to as a multistage interconnection network (MIN) [7, 10], has been used in a variety of machines. This topology was used in the 1970s' STARAN SIMD machine, where data would traverse the network from one side to the other [1]. Allowing data to reverse direction at any stage in a bidirectional MIN (BMIN) leads to a variation sometimes called a fat-tree. MINs and their fat-tree variants are used in 1990s' MIMD machines such as the Meiko CS-2 [2], the IBM SP2 [12], and the Thinking Machines CM-5 [8].

Direct networks often excel at routing local traffic patterns such as passing boundary data in grids, and higher-dimensional direct networks such as hypercubes are also adept at certain non-local traffic patterns and permutations. Indirect networks, such as a k-node MIN, typically have a communications delay proportional to $\log_2 k$, can provide a variety of global communication paths by passing though the multiple stages of switches, and can be constructed from a fixed size building block for a range of values for k. This is in contrast to a direct network such as a 2D mesh that has a worst case delay proportional to $\sqrt{k}$, and to a direct network such as a hypercube whose optimal switch size is $\log_2 k$ by $\log_2 k$ (i.e., a function of the machine size). Unidirectional MINs are not

[1] A switching element is a device containing multiple input and output ports, with a crossbar interconnection between them.

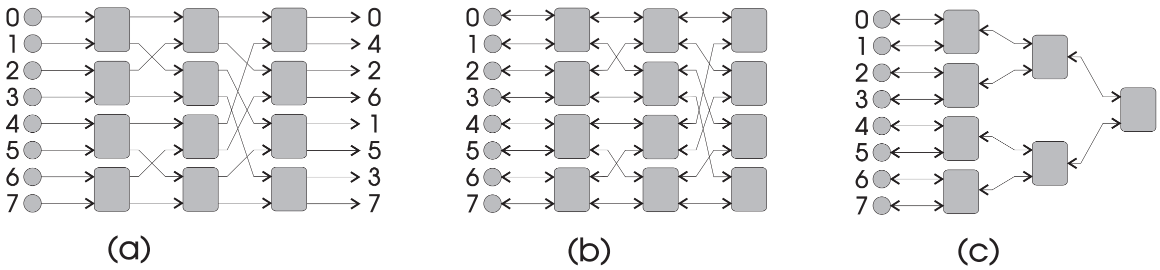

Figure 2: Indirect topologies: (a) multistage interconnection network (MIN), (b) bidirectional MIN, and (c) tree.

optimized for local traffic, although BMINs profit from the shorter routes between nodes within the same sub-tree.

Switching techniques

One attribute of networks that has evolved significantly is switching technique. Many early designs used circuit-switching, in which an entire circuit (path) through the network is reserved before message data is transferred along that path. This technique is efficient for large data transfers, but may reserve network links for an extended period of time, which blocks the formation of other circuits and increases the variance in message latency. In addition, small messages incur a large overhead for the setup and release of the circuit.

Other designs use packet-switching, in which large messages are typically broken into multiple smaller self-routing units termed packets. The first implementations of packet-switching were store-and-forward, in which an entire packet is received into a switch buffer before being forwarded to the next switch in the path. A switch output port forwards the packet when a buffer for the packet is available in the next switch. This method is very efficient when a full packet can be passed in one switch cycle.

To minimize latency for large packets, virtual cut-through (VCT) was proposed [5]. In VCT, each switch can begin forwarding the packet immediately after it determines an appropriate switch output, but blocked packets are completely buffered as in store-and-forward. To reduce switch cost, wormhole routing [4] relaxes the constraint of buffering entire blocked packets, and instead blocks packets in place in multiple switches along the currently allocated path. Wormhole routing is currently the most common switching technique for networks in commercial parallel machines.

Queuing techniques

The queuing technique within each switch has an enormous impact on the potential aggregate bandwidth of the network. The simplest technique to implement is input queuing, in which packets queue at the switch inputs awaiting the availability of the desired switch output. Packets blocked at the head of a queue also block the packets behind it, even if some of these packets are destined for idle switch outputs. This is often referred to as the "head-of-the-line blocking" problem. A higher-performing but more complex technique is output queuing. Output queuing is difficult to implement because, to maintain bandwidth through a switch, each switch output queue must be able to accept packet data from all switch inputs concurrently.

A strategy for gaining performance is to form multiple queues at each switch input (e.g., one queue for each possible destination switch output [13]). Another strategy is to create a central buffer shared among all switch inputs and outputs [12]. This latter technique allows an alternative implementation of output queuing: only one entity—the central buffer—needs to receive and send packet data from each switch input and to each switch output concurrently. In addition, the space within this central buffer can be shared dynamically among the inputs and outputs on a demand basis, further improving bandwidth through the switch. Most recent parallel network implementations employ multiple input queues or central buffering.

Routing techniques

Network routing techniques can be categorized according to path length, adaptivity, and deadlock avoidance and recovery mechanisms. Packet routing can adhere to minimal or non-minimal path length. Individual switches can adaptively react to network congestion or faults by choosing among more than one available switch output, or packet routing can be non-adaptive (also called deterministic or oblivious). Some adaptive schemes are non-minimal.

Deadlock occurs in networks when cyclic dependencies form among packets. Deadlock can be avoided by methods such as restricted route paths. Another strategy is to allow deadlock to occur, but to detect it and recover from it. To date, commercial parallel systems have exclusively used deadlock avoidance as opposed to deadlock recovery, or have employed networks such as MINs that create no cyclic dependencies for minimal routing.

Developments to watch

Comparing networks to determine which is "best" is extremely difficult due to differences in performance criteria, application domains, operating environments, cost constraints, and implementation technologies. While there is currently no clear consensus on optimal topologies, meshes, hypercubes, MINs, and BMINs have been the most widely used in recent implementations. Circuit-switching and store-and-forward switching have yielded to wormhole routing, and it is possible that, as improvements in VLSI increase the buffer capacity per switch, virtual cut-through or hybrid wormhole-VCT solutions [12] will dominate in the future. Likewise, these VLSI improvements will favor the increasing use of multiple switch input queues or central buffering to attack head-of-the-line blocking. Other developments to watch include the influence of ATM[2] architecture and technology, the emergence of low-cost optical interconnections, and optical switching.

A major issue inseparable from parallel networks is the component of message latency due to sending and receiving overhead at the nodes. This overhead dominates the latency for traversing a lightly-loaded network in many current parallel machines. Thus, the improvement of communication protocols and the processor-network interface is fundamental to the successful evolution of parallel networks.

References

[1] K. E. Batcher. The flip network in STARAN. In *Proc. Int. Conf. on Parallel Processing*, pages 65–71, Aug. 1976.

[2] J. Beecroft, M. Homewood, and M. McLaren. Meiko CS-2 interconnect Elan-Elite design. *Parallel Computing*, 20:1627–1638, Nov. 1994.

[3] W. J. Bouknight, S. A. Denenberg, D. E. McIntyre, J. M. Randall, A. H. Sameh, and D. L. Slotnick. The ILLIAC IV system. *Proc. of IEEE*, 60(4):369–388, April 1972.

[4] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. on Computers*, C-36(5):547–553, May 1987.

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

[2] Asynchronous Transfer Mode

[6] R. E. Kessler and J. L. Schwarzmeier. Cray T3D: A new dimension for Cray Research. In *Proc. 38th IEEE Int. Computer Conf.*, pages 176–182, Spring 1993.

[7] D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Trans. on Computers*, C-24(12):1145–1155, Dec. 1975.

[8] C. E. Leiserson, et al. The network architecture of the Connection Machine CM-5. In *Proc. 1992 Symp. Parallel Algorithms and Architectures*, pages 272–285. ACM, 1992.

[9] I. D. Scherson and A. S. Youssef, editors. *Interconnection Networks for High-Performance Parallel Computers.* IEEE Computer Society Press, Los Alamitos, CA, 1994.

[10] H. J. Siegel. Interconnection networks for SIMD machines. *IEEE Computer*, pages 57–65, June 1979.

[11] H. J. Siegel. *Interconnection Networks for Large-scale Parallel Processing.* McGraw Hill, New York, NY, 2nd edition, 1990.

[12] C. B. Stunkel, D. G. Shea, B. Abali, et al. The SP2 high-performance switch. *IBM System Journal*, 34(2):185–204, 1995.

[13] Y. Tamir and G. L. Frazier. High-performance multi-queue buffers for VLSI communication switches. In *Proc. 15st Ann. Int. Symp. on Computer Architecture*, pages 343–354, May 1988.

[14] A. Varma and C. S. Raghavendra, editors. *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice.* IEEE Computer Society Press, Los Alamitos, CA, 1994.