Inside Parallel Computers: Trends in Interconnection Networks

Howard Jay Siegel, *Purdue University*
Craig B. Stunkel, *IBM T.J. Watson Research Center*

Computational scientists who depend on parallel computing to let them run larger models in less time will be disappointed unless the processors can pass information back and forth quickly. The interconnection networks through which processors communicate in tightly coupled parallel machines thus remain a vital research topic for computer architects.

Over the last two decades we have seen an evolution in the demands placed on these networks. Early SIMD machines required the simultaneous transfer of data from each network input to each output for a relatively small set of communication configurations or permutations; whereas the SIMD and MIMD machines of today need to support varied patterns of synchronous and asynchronous traffic, respectively. 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, their fault tolerance, their simplicity, their amenability to partitioning, and their aggregate bandwidth for local and nonlocal traffic patterns.[1–3]

Topologies

Interconnection networks are built up of switching elements (*switches*), which are devices that contain multiple input and output ports with a crossbar interconnection between them. *Topology* is the pattern in which the individual switches of the network are connected to other switches and to processors and memories (*nodes*). Connecting all nodes via a single nonblocking 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 many 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 connect only to other switches.

Figure 1 illustrates several direct networks, with each circle representing a node–switch combination. The 2D mesh of the 1970s-era Illiac IV[4] is similar to the 1990s-era 2D mesh of the Intel Paragon and the toroidal 3D mesh of the Cray T3D[5] and T3E.

Figure 2 shows some indirect networks, with circles representing nodes and rectangles indicating switches. The indirect cube network, often referred to as a *multistage interconnection network* or MIN,[6,7] has been used in a variety of machines. This topology was used in the Staran SIMD machine of the 1970s, in which data would traverse the network from one side to the other.[8] Allowing data to reverse direction at any stage in a bidirectional MIN (or BMIN) leads to a variation sometimes called a *fat-tree*. MINs and their fat-tree variants are used in MIMD machines of the 1990s such as the Meiko CS-2,[9] the IBM SP2,[10] and the Thinking Machines CM-5.[11]

Direct networks often excel at routing local traffic patterns such as the passing of boundary data in grids. Higher-dimensional direct networks such as hypercubes are also adept at certain nonlocal traffic patterns and permutations.

Indirect networks (such as k-node MINs) can provide a variety of global communication paths by passing though the multiple stages of switches. They typically have a communication delay proportional to $\log_2 k$. This is in contrast to a direct network such as a 2D mesh, which has a worst-case delay proportional to $\sqrt{k}$. Indirect networks can be constructed from a fixed-size building block for a range of values of k. This contrasts with a direct network such as a hypercube, whose optimal switch size is $\log_2 k \times \log_2 k$ (that is, a function of the machine size). Although unidirectional MINs are not optimized for local traffic, BMINs profit from the shorter routes between nodes within the same subtree.

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 a message is transferred along that

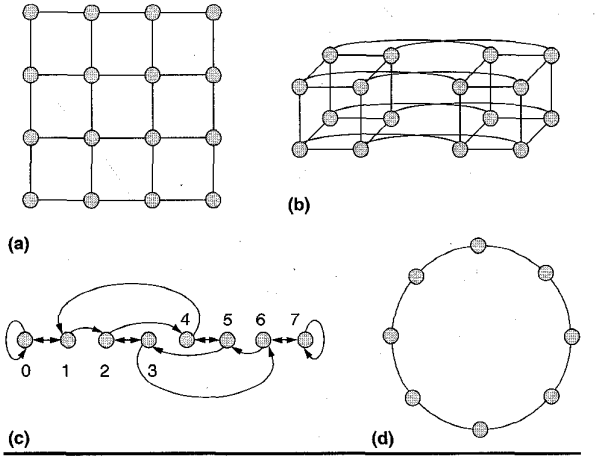

Figure 1. Direct topologies: (a) 2D mesh, (b) 4D hypercube, (c) shuffle-exchange, and (d) ring.

1070-9924/96/$5.00 © 1996 IEEE

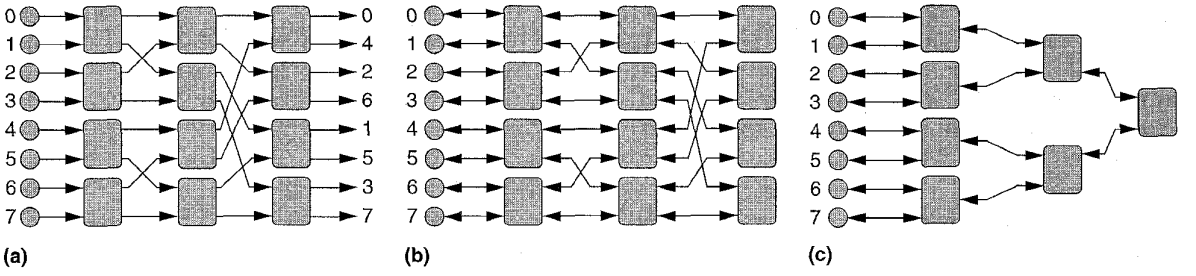

Figure 2. Indirect topologies: (a) multistage interconnection network (MIN), (b) bidirectional MIN (or fat-tree), and (c) regular tree.

path. This technique is efficient for large data transfers but it may reserve network links for a long time, blocking the formation of other circuits and increasing the variance in message latency. For small messages, circuit setup and release causes a disproportionately large overhead.

Other designs use *packet-switching*, in which large messages are broken into smaller self-routing units called packets. The first implementations of packet-switching were of the *store-and-forward* variety, 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* was proposed.[12] 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*[13] relaxes the constraint of buffering entire blocked packets in a single switch, 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 (for instance, one queue for each possible destination switch output[14]). Another strategy is to create a central buffer shared among all switch inputs and outputs.[10] 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 commercial parallel network implementations employ input queues or central buffering.

Routing techniques

Network routing techniques can be categorized according to path length, adaptivity, and mechanisms for deadlock avoidance and recovery. Packet routing can adhere to minimal or nonminimal 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 nonadaptive (also called deterministic or oblivious). Some adaptive schemes are nonminimal.

Deadlock occurs in networks when cyclic dependencies form among packets, and 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 owing to differences in performance criteria, application domains, operating environments, cost constraints, and implementation technologies.

No clear consensus exists on optimal topologies. However, 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[10] 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 (asynchronous transfer mode) 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. I.D. Scherson and A.S. Youssef, eds., *Interconnection Networks for High-Performance Parallel Computers*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1994.
2. H.J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing*, 2nd ed., McGraw Hill, New York, 1990.
3. A. Varma and C.S. Raghavendra, eds., *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1994.
4. W.J. Bouknight et al., "The ILLIAC IV System," *Proc. IEEE*, Vol. 60, No. 4, Apr. 1972, pp. 369–388.
5. R.E. Kessler and J.L. Schwarzmeier, "Cray T3D: A New Dimension for Cray Research," *Digest of Papers, COMPCON Spring '93 [Proc. 38th IEEE Computer Soc. Int'l Computer Conf.]*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1993, pp. 176–182.
6. D.H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Computers*, Vol. C-24, No. 12, Dec. 1975, pp. 1,145–1,155.
7. H.J. Siegel, "Interconnection Networks for SIMD Machines," *Computer*, Vol. 12, No. 6, June 1979, pp. 57–65.
8. K.E. Batcher, "The Flip Network in STARAN," *Proc. 1976 Int'l Conf. on Parallel Processing*, IEEE, New York, 1976, pp. 65–71.
9. J. Beecroft, M. Homewood, and M. McLaren, "Meiko CS-2 Interconnect Elan-Elite Design," *Parallel Computing*, Vol. 20, Nov. 1994, pp. 1,627–1,638.
10. C.B. Stunkel et al., "The SP2 High-Performance Switch," *IBM System J.*, Vol. 34, No. 2, 1995, pp. 185–204.
11. C.E. Leiserson et al., "The Network Architecture of the Connection Machine CM-5," *Proc. 4th Ann. Symp. Parallel Algorithms and Architectures*, ACM Press, New York, 1992, pp. 272–285.
12. P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communications Switching Technique," *Computer Networks*, Vol. 3, No. 4, Sept. 1979, pp. 267–286.
13. W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers*, Vol. C-36, No. 5, May 1987, pp. 547–553.
14. Y. Tamir and G.L. Frazier, "High-Performance Multi-Queue Buffers for VLSI Communication Switches," *Proc. 15th Ann. Int'l Symp. on Computer Architecture*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1988, pp. 343–354.

Howard Jay Siegel is a professor of electrical and computer engineering at Purdue University. He received BS degrees from MIT and the PhD degree from Princeton. He published his first paper about interconnection networks in 1975 and has since coauthored over 230 papers on parallel and heterogeneous computing. Siegel is a fellow of the IEEE and is on the editorial board of IEEE Transactions on Parallel and Distributed Systems. *School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-1285; e-mail, hj@purdue.edu.*

Craig B. Stunkel is a research staff member at IBM's T.J. Watson Research Center. He received BS and MS degrees from Oklahoma State University, and the PhD in electrical engineering from the University of Illinois in 1990. At IBM Research he was a codesigner of the Vulcan network used in the IBM SP1 and SP2 parallel machines, and he continues to investigate interconnection network issues. IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598; e-mail, stunkel@watson.ibm.com.

Modernizing High-Performance Computing for the Military

Anita K. Jones
US Department of Defense

For five decades the Department of Defense has helped build and sustain US world leadership in computing. The first modern digital computer—the Electronic Numerical Integrator and Computer, or Eniac—was developed under the direction of US Army Colonel Paul N. Gillon 50 years ago at the Army Ballistic Research Laboratory, Aberdeen Proving Ground, Aberdeen, Maryland. Eniac was designed to compute