System Effects of Interprocessor Communication Latency in Multicomputers

An important factor in the efficiency of a distributed-memory multicomputer is the effectiveness with which data can be exchanged among its many nodes. A series of experiments and analyses on five types of hypercube and grid-topology multicomputers helped to evaluate interprocessor communication performance. Examination and comparison of system communication speed, message routing, interprocessor connectivity, and software/hardware protocols for passing messages among the five multicomputers enhanced the analysis.

Xiaodong Zhang

University of Texas at San Antonio

Parallel processing applies a simple idea: A computing job can be divided into several tasks that may be executed in parallel. Over the last 10 years designers implemented this concept using distributed-memory multicomputers in a variety of forms in different applications. This experience shows that parallel processing does not reach its anticipated speed when a large number of processors are used in solving problems.[1,2] The communications of common-state information among processors cause a major degradation of the performance (speed).

The literature[3-5] records efforts to measure and evaluate the interprocessor communication performance on the Intel hypercube and the Ncube multicomputers. In addition, Saad and Schultz[6] present several efficient algorithms for data communication on a hypercube multicomputer.

This article takes a wider view, studying various system effects of interprocessor performance, including communication speed, message routing, interprocessor connectivity, and message-passing software/hardware protocols. Both analytical and experimental results offer a clear and comprehensive understanding of the various effects, which is important for the effective use of a distributed-memory multicomputer.

Five multicomputer architectures

In a distributed-memory multiprocessor system, or multicomputer,[7] each processor has its own local memory, and tasks on separate processors coordinate their activities by sending messages through an interconnection network. However, many recent commercial distributed-memory systems vary in computing power, number of processors, type of processors, and network interconnection topology, as well as communication hardware and software.

The hypercube is one example of a distributed-memory, message-passing multicomputer. In a hypercube network 2^n processors are consecutively numbered 0 through $2^n - 1$. Each processor connects to all of the other processors, whose binary representation differs from its own by exactly one bit. This arrangement results in a network that is connected densely enough to support efficient communication between arbitrary processors. Another virtue of the hypercube network is its flexibility: Many other interconnection topologies, such as rings and trees, can be embedded in the hypercube. The dimension n of a hypercube with 2^n nodes determines the maximum number of hops needed to send messages between two nodes. Some system parameters of the five studied multicomputers are:

0272-1732/91/0400-0012$01.00 © 1991 IEEE

- **Intel iPSC/1.** The iPSC/1, one of the first commercially available hypercube computers, may support up to 128 nodes. Each node includes an 8-MHz Intel 80286 processor and 512 Kbytes of local memory. The node operating system supports message-routing asynchronous communications and multitasking within each node.[8]
- **Intel iPSC/2.** This second-generation hypercube features a 4-million-instructions-per-second Intel 80386 node processor, which is four times faster than the 286. Each node can access up to 16 Mbytes of local memory, whereas the iPSC/1 accesses 0.5 Mbytes. The NX/2 operating system supports the new message-passing protocols in the iPSC/2 besides providing a normal system environment in each node.[9]
- **Ncube/10.** This first-generation hypercube system supports up to 1,024 processors. The 32-bit, custom-chip node processor operates at a 7-MHz clock rate and contains 128 Kbytes of local memory. Since the processor includes communication channels, the number of chips per node on the Ncube is relatively low. The Axis operating system supports the transmission of messages between arbitrary nodes of the Ncube/10.[3,10]
- **Ametek 2010.** The Ametek 2010 multicomputer system is based on a 2D grid topology. Each node includes a 25-MHz Motorola 68020 processor and up to 8 Mbytes of local memory.
- **Topology 1000.** This parallel system is a transputer-based variable topology board. The interprocessor network of this Topologix system can be reconfigured. The processor in each node of the network uses the 32-bit, 20-MHz Inmos T800 transputer and up to 16 Mbytes of local memory per processor node.[11-13] The transputer's links are based upon point-to-point interprocessor communication, which eliminates bus contention when messages are transferred. Logix OS is the distributed Unix-compatible operating system supported on the Topology 1000. The Trollius operating system developed at the Cornell Theory Center forms the basis of the Logix OS.

Table 1 summarizes the five types of architectures.

Interprocessor communication

Communication efficiency, one of the most important factors to be considered when designing a multicomputer architecture, often becomes one of the main obstacles to increased performance of parallel algorithms on distributed systems. When a message passes between a pair of nodes in a network, it may be routed through a connected circuit in a number of hops. In addition, intermediate processors may be interrupted to store and then forward the message, or the message may be directly transferred by communication-processing data links through a connected circuit. Thus, the communication speed of the interprocessor network depends on the communication-routing protocols, processor speed, data link speed, and topology of the network.

A comparison of the various effects of different routing models, different interprocessor connections, and other factors to the performance of interprocessor communication on the five types of distributed memory architectures follows.

Communication models. Consider the store-and-forward mechanism[14] used as a typical communication model for first-generation multicomputers such as the iPSC/1, Ncube/10, and Ametek/14. In this communication model, messages pass indirectly between a pair of nodes that are not directly connected via other connected nodes. Each node in the communication path temporarily stores the message in its memory. The processor on each node in that path interrupts work on a task to forward the stored message either to its neighbor or the destination node. Thus, while messages move between a pair of nodes across the network, memory bandwidth and computing cycles in the intermediate nodes are consumed.

The communication latency of this model is also very sensitive to the distance a message must be passed, or it is linearly proportional to the number of hops of the communication. We can express the communication latency of the store-and-forward model as:

$$T_{lat1} = T_{d1} H \quad (1)$$

where $T_{d1} = K/B_1$, which is the time for a message of size K (bytes) to pass through the channel of bandwidth B_1 (bytes/s) in one hop. H equals the distance in the number of hops,

Table 1. Architecture overviews.

Features	iPSC/1	iPSC/2	Ncube/10	Ametek 2010	Topology 1000
Node CPU	Intel 286	Intel 386	Custom 32-bit	Motorola 68020	Inmos T800
Clock rate (MHz)	8	16	7	25	20
Node operating system	Axis 3.0	NX/2	Axis 2.3	R Kernel	Logix OS
Node memory (bytes)	512K	Up to 16M	128K	8M	Up to 16M
Data rate (Mbytes/s)	1.25	4	0.875	20	5

and we can view T_{d1} as the routing delay of each node.

Kermani and Kleinrock[14] and Athas and Seitz[15] called the basic routing model used in second-generation multicomputers (for example, the iPSC/2 and Ametek 2010) wormhole routing. Instead of storing a packet completely in a node and then transmitting it to the next node, wormhole routing operates by advancing the head of a packet directly from incoming to outgoing channels. Only a few flow-control digits are buffered at each node. These digits, or flits, are the smallest units of information that a queue or channel can accept or refuse. A message consists of a sequence of flits, in which the flit at the head of the message governs the route, and the remaining flits follow in pipeline fashion. Besides avoiding the use of storage bandwidth in the nodes through which messages are routed, wormhole routing and its flow control also reduce the message latency caused by distance in the network. Therefore, the data transfer rate becomes the limiting factor for message-passing speed.

We can express the communication latency of the wormhole model as:

$$T_{lat2} = T_{d2}H + (K/B_2) \quad (2)$$

where $T_{d2} = K_h/B_2$ is the routing delay in each node for sending the packet head in K_h (bytes) to pass through the channels of bandwidth B (bytes/s). K/B_2 is the time required to transmit the whole packet K (bytes) continuously through the wormhole channels of bandwidth B_2 (bytes/s), and H is the communication distance. The ratio between Equations 1 and 2 is a quantitative comparison of the two models:

$$R = \frac{T_{lat1}}{T_{lat2}} = \left(\frac{B_2}{B_1}\right)\frac{KH}{K_hH + K} \quad (3)$$

The size of the packet head K_h is trivial in comparison with the size of the whole packet K. For example, the packet head size in the Ametek 2010 is only 2 bytes. Therefore the ratio R in Equation 3 may be expressed as:

$$R \approx (B_2/B_1)H \quad (4)$$

This equation indicates that the wormhole model reduces the communication latency up to $B_2/B_1 \times H$ times over the store-and-forward model. In this case, we assumed the message size K and the communication distance H to be the same in both communication architectures, and the bandwidth of the second-generation multicomputer B_2 to be higher than the one of the first-generation B_1.

Even if the data bandwidth of the two models were the same, $B_1 = B_2$, the communication latency would be reduced to H times in the wormhole model. For example, the first-generation hypercube Intel iPSC/1 uses the store-and-forward model; its data bandwidth is 1.25 Mbytes/s. The second-generation hypercube Intel iPSC/2 uses the wormhole model; its data bandwidth is 4 Mbytes/s. If we substitute $B_1 = 1.25$, $B_2 = 4$, and H = 5 for a 32-node hypercube in Equation 4, we obtain $R \approx 16$. This ratio indicates that a 32-node iPSC/2 hypercube may reduce the communication latency time up to 16 times over a 32-node iPSC/1 hypercube.

Hardware implementation. The communication mechanisms based on the store-and-forward technique used in the Intel iPSC/1 and Ncube/10 are typical first-generation message-passing protocols on a distributed-memory multiprocessing system. The processor on each node in that path participates in handling communications, stopping other processing tasks during message-passing periods.

The iPSC/1 and the Ncube/10 consume the local memory bandwidth and computing cycles in the routing nodes while accumulating a latency of several hundred microseconds per hop. Thus, the computing speed and bandwidth in each processor mainly determine the store-and-forward communication speed. The higher the clock rate of each processor, the lower the latency in communication will be, since the processor more speedily accomplishes the store-and-forward operation. The experiment's results discussed in the next section show the low efficiency of the store-and-forward techniques on the Intel iPSC/1 and Ncube/10.

We can implement the wormhole model differently on a multicomputer. The hardware structures on the iPSC/2 and Ametek 2010 are two typical implementations for the wormhole routing model on a interconnecting network.

The wormhole routing model greatly reduces communication latency.

The direct-connect router, a hardware-controlled message-passing system in the Intel iPSC/2, forms the basis of the communication system. Think of the router as a switching network. When one node wants to communicate with another, the sending node closes a series of switches and establishes the communication path. Then, messages proceed at the full hardware speed of 4 Mbytes/s. Only the sending and destination processors participate in the communication; the other processors in the routing path continue with their normal activities. Since it takes only a few microseconds per node to build the path, the additional overhead for multihop communications is insignificant. In addition, the hardware routes messages independently, and the iPSC/2 communication latency is significantly reduced over that in the iPSC/1.

The Ametek 2010 communication network is the most efficient one among the five multiprocessing architectures. The message network consists of a 2D grid of custom mesh routing chips. Message packets advance directly from one of these chips to another in a blocking variant of cut-through routing of the wormhole routing. At the 20-MHz rate, the 8-bit-wide channels yield a communication bandwidth of 20 Mbytes/s per channel. Thus, the network quickly establishes a connection circuit between two remote nodes, and the mesh routing chips transfer messages in a byte-serial fashion in one operation.

The Topology 1000 implements the store-and-forward technique differently. The communication system is tied into each transputer at a very low level. The transputer employs a hardware scheduler to schedule the communication of messages. Therefore, setting up a communication takes just a few microseconds.

On the other hand, the transputer implements synchronized message passing. Both sender and receiver must be ready before a communication can take place. This coordination occurs at the lowest level of the communication protocol and results in the absence of problems with data overruns or buffer overflows. In addition, the store operation acts the same as it does in the iPSC/1 and Ncube/10, storing the message in the local memory of the routing node. However, each processor is only responsible for initiating the forward operation. Then the DMA data link carries out the message transfer without further interruption of the processor.

The DMA data links on the Topology 1000 operate at a maximum unidirectional rate of 1.75 Mbytes/s or a bidirectional rate of 2.5 Mbytes/s. Four links per transputer produce a 10-Mbyte/s rate. The basic idea of this model is to use *excess* parallelism to hide the latency in the data transfer. For very short messages, a low transfer rate is possible because most of the time spent in the communication occurs in the processor cycles upon initializing a data transfer. However, the communication can take advantage of a large message transfer when the processor's initialization time is trivial (compared with the data transfer time used by the DMA data links).

Experimental results on a Topology 1000 with the DMA data links show improvement in communication efficiency. The communication speed of the Topology 1000 is much higher than on the Intel iPSC/1 and Ncube/10, although all three multiprocessing systems use general store-and-forward techniques.

Comparing the two topologies. The Ncube and both iPSC systems use the hypercube interconnection topology. The Ametek 2010 uses a 2D grid as the interprocessor connecting topology. We can compare these two network topologies in terms of the communication efficiency.

We can make a hypercube of arbitrary dimension by using a linear arrangement with connecting wires. We obtain the cube of each dimension by replicating the one in the next-lower dimension and then connecting corresponding nodes. For example, directly connecting two nodes labeled 0 and 1 between the two nodes gives us a one-dimensional hypercube (2^1). We make a 2D hypercube by duplicating the bisection, or the 1D hypercube, by directly connecting the corresponding node of each bisection together. Adding a high-order bit to the node number sets it to 0 for the lower order bisection and 1 for the other. We construct the higher dimensional hypercube by further connecting the bisections of the hypercube. As Figure 1a shows, each processor in a hypercube connects to all other processors whose binary tags differ by exactly one bit. We can make a hypercube of arbitrary dimension by using a linear arrangement with connecting wires, as shown in Figure 1b.

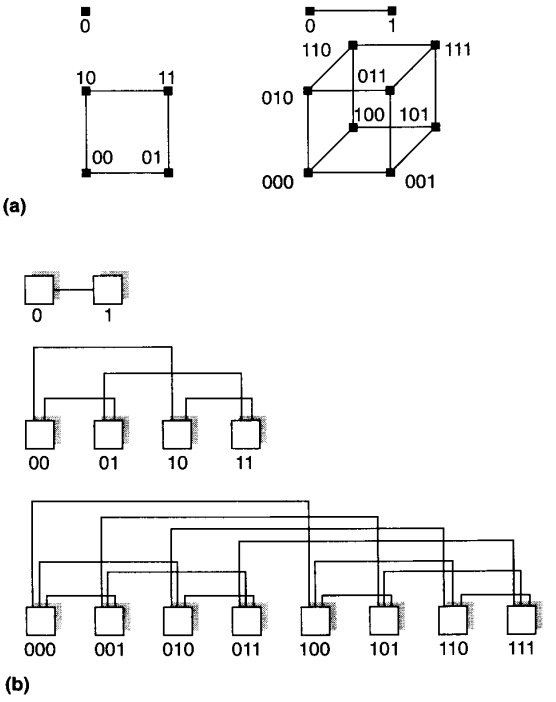

Figure 1. Construction of a hypercube.

We can make a channel that physically links two directly connected nodes from a bundle of wires consisting of data bits and any necessary control bits. We need $N/2$ channels across the bisection to construct a hypercube, where N is number of nodes in the hypercube. However, using the same method to construct a 2D grid requires $O\sqrt{N}$ channels across the bisection, where N is the number of nodes in the 2D grid. We can determine the maximum distance between a pair of

continued on p. 52

Oct. 1984, pp. 1733-1749.
33. P. Corsini, B. Lazzerini, and C.A. Prete, "A Kernel for a Multiprocessor System with Anonymous Processes," *Proc. Int'l Conf. Parallel Processing and Applications,* North-Holland, Amsterdam, Sept. 1987, pp. 71-78.
34. Q. Yang, L.N. Bhuyan, and B.-C. Liu, "Analysis and Comparison of Cache Coherence Protocols for a Packet-Switched Multiprocessor," *IEEE Trans. Computers,* Vol. 38, No. 8, Aug. 1989, pp. 1143-1153.
35. B. Lazzerini, L. Lopriore, and C.A. Prete, "A Programmable Debugging Aid for Real-Time Software Development," *IEEE Micro,* Vol. 6, No. 3, June 1986, pp. 34-42.
36. P. Corsini and C.A. Prete, "Multibug: Interactive Debugging in Distributing Systems," *IEEE Micro,* Vol. 6, No. 3, June 1986, pp. 26-33.
37. B. Lazzerini and C.A. Prete, "Event-Driven Debugging for Distributed Software," *Microprocessors and Microsystems,* Vol. 12, No. 1, Jan./Feb. 1988, pp. 33-39.
38. *PAL Device Data Book,* Advanced Micro Devices, Inc., 1988.
39. *CMOS Data Book,* Cypress Semiconductor Corp., San Jose, Calif., Jan. 1986.

Cosimo A. Prete is a research fellow at the Institute of Electronics and Telecommunications of the University of Pisa. He is involved in the Italian National Research Council's Nonconventional Parallel Systems project, which is conducting comparative analysis and performance evaluation of operating systems and programming environments for nonconventional parallel systems. His main interests include multiprocessor organization, cache memories, and software development methodologies.

Prete holds a degree in electronic engineering and a PhD (Dottore di Ricerca) from the University of Pisa, Italy.

Address questions concerning this article to Cosimo A. Prete, Università di Pisa, Dipartimento di Ingegneria dell' Informazione: Elettronica, Informatica e Telecommunicazioni, 56126 Pisa, Via Diotisalvi, 2, Italy; e-mail: prete@mv3500.eit.unipi.it.

Communication latency

continued from p. 15

nodes in a hypercube if we know the dimension of the hypercube, or $\log_2(N)$. The same factor in a 2D grid is $O(\sqrt{N})$, which increases faster than $\log_2(N)$.

Recall that the communication latency is dependent on the channel width, distance (number of hops), and size of the message. The network latency in the wormhole model precisely equals the time it takes the head of a message to enter the network at the source and the tail to emerge at the destination:

$$T_{lat} = t_d H + (K/B) \tag{5}$$

Here, T_d is the delay of the individual routing nodes encountered on the path, H is the number of hops needed in passing messages, and K/B is the time required for a message of size K to pass through the channels of bandwidth B.

In lower dimensional hypercube topology, the number of hops increases, but so does the channel width. The optimization to minimize latency simply minimizes Equation 5. In this equation, higher dimensional networks reduce the first term at the expense of the second, while lower dimensional networks reduce the second term at the expense of the first. The 2D grid has $O(\sqrt{N})$ times more wires per channel for a fixed number N of nodes than an equivalent N-node topology. The following numerical comparisons indicate the advantage of lower latency in the 2D grid network. Assume the routing delay T_d in both the hypercube and 2D grid networks is identical. We can express the time needed to send a message with K bytes between a pair of nodes in the maximum distance $\log_2(N)$ in the hypercube with N nodes as:

$$T_{cube} = T_d \log_2(N) + (K/B) \tag{6}$$

We express the same timing factor in the 2D grid network of N nodes to send a K-size message differently, since the bandwidth in each channel is $O(\sqrt{N})$ times wider, and the maximum distance $O(\sqrt{N})$ is also a faster-increasing function:

$$T_{grid} = T_d O(\sqrt{N}) + \frac{K}{O(\sqrt{N})B} \tag{7}$$

We derive the ratio of T_{cube}/T_{grid} by

$$R = O(\sqrt{N}) \frac{T_d \log(N) + T_c}{T_d N + T_c} \tag{8}$$

where $T_c = K/B$.

The second term K/B of Equation 5 dominates the network latency for all but very short messages in the second-generation multicomputers. For example, in one implementation conducted by the California Institute of Technology,[16] the routing delay in one node T_d was 80 nanoseconds. Even the fast bandwidth of the Ametek 2010—B = 20 Mbytes/s needed to transfer a 160-byte message—would take 8 microseconds, which is 100 times longer than T_d. When K is reasonably large, T_d may be ignored, and the ratio R of Equation 8 is $O(\sqrt{N})$. That is, the communication latency in a 2D grid network may be reduced up to $O(\sqrt{N})$ times over a hypercube network.

In summary, given a constant bisection width, the 2D grid network produces lower latency and higher throughput than a higher dimensional hypercube. Mainly, fewer channels contribute to the bisection, which permits each channel to be made wider. On the other hand, the throughput is bounded by allowing more channels crossing the bisection in a higher dimensional hypercube.

A transputer is a good candidate for constructing a 2D grid network.

The Topology 1000 provides hardware reconfigurability of the network topology under software control through the use of the Inmos C004 link crossbar adapter. Thus, a user can define an interprocessor communication topology, and the hardware and software can implement it. Since each transputer has four links to connect with other transputers in the network, a transputer becomes a good candidate for constructing a 2D grid network. We can achieve full connectivity or high connectivity in a lower dimensional topology with a small number of nodes and construct a 4D hypercube by connecting 16 transputers properly.

However, we cannot build higher dimensional hypercubes out of transputers exclusively since they are limited to four links per node, and hypercubes of five or more dimensions require five or more links per node. Such topologies are possible with the addition of hardware link switches such as the Inmos C004 crossbar adapter used in the Topologix system. Performance losses occur with the use of such switches, however.

The experiment

A distributed-memory multicomputer is a collection of processors or nodes connected by a communication network. Thus, the basic communication timing test for distributed-memory multicomputers requires measurement of the time required to transmit a message packet from one node to its nearest neighbor. This test, also known as an echo test, directs a test node to send a message to an echo node that is directly connected to the test node. The echo node receives the message and sends it back to the test node. We can express the interprocessor communication time required to transmit a message between two directly connected nodes as:

$$T_{comm} = \alpha + \beta K$$

where K is the number of bytes contained in the message. Here, α equals the overhead or the start-up time for sending a packet in microseconds, and β equals the bandwidth of the communication channel (microseconds/byte). The experiment used different sizes of message packets and a least square fit to approximate α and β. Table 2 reproduced from Zhang and Beguelin[17] lists the αs and βs of the five types of multicomputers.

Table 2. Alphas and betas (in microseconds/byte) for one-hop communication.

Multiprocessor	α	β
iPSC/1	893	1.51
iPSC/2	349	2.30×10^{-1}
Ncube/10	447	2.40
Ametek 2010	168	1.01×10^{-1}
Topology 1000	215	1.02×10^{-1}

Since multiple-hop communications occur more often in most applications on a multiprocessor system, the one-hop communication measurements do not let us sufficiently evaluate the performance of the interprocessor communication. For this reason, we[17] constructed a comprehensive experiment to measure the overall communication performance on a multiprocessor system for a given topology network.

In the experiment, a test node sent n messages to and received n messages from all nodes in the network. We measured the time it took for a test node to send a message to every node in the network and return. We repeated this process m times and continued the whole process until every node had become the test node. We obtained the average communication time in the network from the p timing measures, where p is number of processors in the network. We chose the message size from a minimum of 1 byte to a maximum of 8 Kbytes. The communication distances in this experiment range from a minimum zero hop (a node to itself) to a maximum H_{max} hops. $H_{max} = n$, for an n-dimensional hypercube topology, and $H_{max} = O(\sqrt{N})$, for an N-node 2D grid topology.

Figure 2 charts the average communication time for different message sizes on different types of multiprocessors. The iPSC/1, iPSC/2, and Topology 1000 have a hypercube topology, and the Ametek 2010 has 2D grid topology. The results of the experiment showed that communication timing differences are very close to the results predicted earlier by the latency models. For example, Equation 4 predicted the iPSC/1 and iPSC/2 communication latency ratio for a 16-node system to be 12.8. The experiment's results in Figure 2 also show that the timing ratio was more than 10.

To show that the communication latency of the wormhole model exhibits little sensitivity to message distance, we conducted another experiment on the five types of multicomputers. In this experiment we fixed the message size, let the communication time become the function of the distance, and set the number of hops as H. We ran this experiment with message-packet sizes of 1 Kbytes to 8 Kbytes and used the average timing value from eight runs as the measure to cover a wide range of message sizes. Figure 3 describes this timing function based on the experimental data.

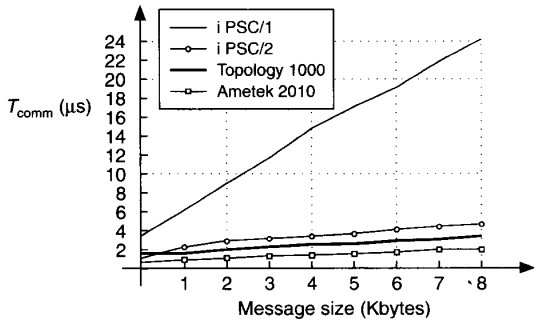

Figure 2. Average multihop communication time on the Intel, Topologix, and Ametek multiprocessors.

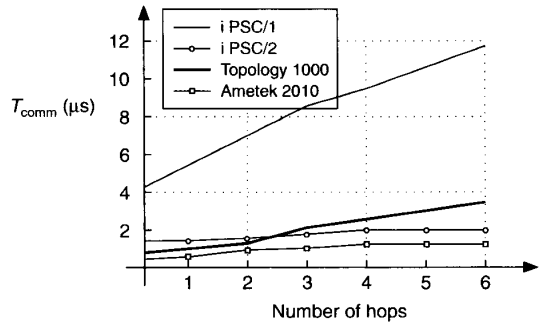

Figure 3. Average communication time with different hops on the same multiprocessors.

The experiment's results clearly show the performance difference of the interprocessor communication between the first-generation multicomputer systems and the second-generation distributed multiprocessor systems. The traditional store-and-forward technique for interprocessor communication greatly limits the communication speed among the processors. In addition, the processors of the first-generation multiprocessing systems are not very powerful, which is another major reason communication proceeds slowly in these systems.

To transfer a message in a store-and-forward network, such as the iPSC/1 or the Ncube/10, the processor must move each byte of data through its own memory, thus consuming both storage bandwidth and computing cycles in the routing nodes. The Intel iPSC/2 uses more powerful processors and, more importantly, uses direct switches as the interprocessor connections. Thus, the communication performance is greatly improved over that of the iPSC/1 and the Ncube/10.

The Topology 1000's high-performance interprocessor communication occurs especially when the number of processors in the network is not very large and the message size is not too small. The four links of each transputer, which may create more hops and a low number of direct connections for a large number of transputer networks, limit the inter-transputer connectivity. The DMA data links in the multiple-transputer system play an important role in transferring data at high speeds. However, as the graph in Figure 3 shows, the communication latency of the Topology 1000 is more sensitive than are the iPSC/2 and Ametek 2010 when the number of hops increases. The Topology 1000 uses the store-and-forward model, after all. We obtained the timing results from a 16-node hypercube network on both the iPSC system and the Topology 1000. Finally, the point-to-point communication established on the Ametek 2010, which contains a powerful mesh routing chip on each node, produces the best interprocessor communication performance among the five multiprocessor architectures.

THE WORMHOLE ROUTING MODEL greatly reduces communication latency and is no longer sensitive to the distance involved in passing messages. In addition, the high-data bandwidth and high-speed nodes of the second-generation multicomputers such as the iPSC/2 and Ametek 2010 increase communication speed. The Topology 1000 interprocessor communication may perform at a rate similar to that of the iPSC/2 and Ametek 2010 on a medium-

size network since the system takes advantage of the high-speed transputer data links. The 2D grid topology is a more efficient structure than a higher dimensional hypercube topology in terms of reducing communication latency, as long as the routing delay in each node is small, such as the one in the second-generation multicomputers.

Acknowledgments

The National Science Foundation under grants CCR-9008991 and CCR-9047481 partially supported this research.

References

1. C. Moler, "Matrix Computation on Distributed-Memory Multiprocessors," *SIAM Proc. Hypercube Multiprocessors*, Soc. Industrial and Applied Mathematics, Philadelphia, 1986, pp. 181-195.
2. X. Zhang, R. Byrd, and R. Schnabel, "Solving Nonlinear Block-Bordered Circuit Equations on Hypercube Multicomputers," *Proc. Fourth Conf. Hypercubes, Concurrent Computers, and Applications*, Vol. I, 1989, pp. 701-707.
3. A. Beguelin and D. Vasicek, "Communication Between Nodes of a Hypercube," *SIAM Proc. Hypercube Multiprocessors*, 1987, pp. 162-168.
4. D.A. Reed and R.M. Fujimoto, *Multicomputer Network: Message-Based Parallel Processing*, MIT Press, Cambridge, Mass., 1987.
5. D.C. Grunwald and D.A. Reed, "Benchmarking Hypercube Hardware and Software," *SIAM Proc. Hypercube Multiprocessors*, 1987, pp. 169-175.
6. Y. Saad and M.H. Schultz, "Data Communication in Hypercubes," *J. Parallel Distributed Computing*, Vol. 6, 1989, pp. 115-135.
7. C.L. Seitz, "The Cosmic Cube," *Comm. ACM*, Vol. 28, No. 1, 1985, pp. 22-33.
8. *iPSC User's Guide*, No. 17455-3, Intel Corp., Portland, Ore., 1985.
9. P. Pierce, "The NX/2 Operating System," *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, ACM Press, 1988, pp. 384-390.
10. *Ncube Handbook, Version 1.1*, Ncube Corp., Beaverton, Ore., 1986.
11. *The Transputer Data Book*, Inmos Corp., Bristol, UK, 1989.
12. *The Transputer Application's Notebook: Architecture and Software*, Inmos Corp., 1989.
13. *The Transputer Application's Notebook: Systems and Performance*, Inmos Corp., 1989.
14. A.S. Tanenbaum, *Computer Network*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1981.
15. P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks*, Vol. 13, 1979, pp. 267-286.
16. W.C. Athas and C.L. Seitz, "Multicomputers: Message-Passing Concurrent Computers," *Computer*, Vol. 21, No. 8, 1988, pp. 9-24.
17. X. Zhang and A. Beguelin, "Interprocessor Communication Performance on Different Types of Multicomputers," *Intelligent Distributed Processing*, R. Ammar, ed., ACTA Press, Anaheim, Calif., 1989, pp. 73-76.

Xiaodong Zhang is an assistant professor of computer science at the University of Texas at San Antonio and holds a visiting faculty position at the Center for Research on Parallel Computation at Rice University. Earlier, he had worked as a member of the technical staff for Topologix Inc., Denver. His research interests lie primarily in the areas of parallel and distributed computation, parallel system performance evaluation and VLSI simulation, and numerical analysis of nonlinear equations and optimization problems.

Zhang received the BS degree in electrical engineering from Beijing Polytechnic University and the MS and PhD degrees in computer science from the University of Colorado at Boulder. He is a member of the IEEE Computer Society, the Association of Computing Machinery, and the Society for Industrial and Applied Mathematics.

Address questions concerning this article to the author at the Division of Mathematics and Computer Science, University of Texas at San Antonio, San Antonio, TX 78285-0664; or via Internet at zhang@ringer.cs.utsa.edu.