Measurement of Communication Rates on the Cray T3D Interprocessor Network * **

Robert W. Numrich[1] and Paul L. Springer[2] and John C. Peterson[2]

[1] Software Division, Cray Research, Inc., Eagan MN, USA
[2] Jet Propulsion Laboratory, California Institute of Technology, Pasadena CA, USA

Abstract. We measure communication rates for the Cray T3D interprocessor network. First we design experiments with no contention on the network to establish the achievable fraction of the peak hardware rate. Then we increase the contention with a sequence of experiments designed to measure the robustness of the network. We analyze the results in terms of the Hockney parameters, r_∞ and $n_{1/2}$.

1 Introduction

The Cray T3D is a massively parallel MIMD computer system with physically distributed but globally addressed memory. It is scalable from 16 to 1024 nodes with two processors per node. Each processor is a 64-bit DEC Alpha microprocessor [11] with a frequency of 150 MHz. The Alpha processor has a direct-mapped data cache organized into 256 lines with 32 bytes per line. Programs can invalidate the local cache as needed to maintain coherence. The Alpha processor manages a virtual address space through an onboard address translation buffer.

Each processor on a node owns a single bank of local DRAM memory external to the Alpha processor. The memory interface between the Alpha processor and local memory is Cray custom circuitry based on the same technology used in Cray's vector-parallel machines. This circuitry extends the local virtual address space to a global address space. It uses five bits of the virtual address to point into a look-up table containing remote processor numbers and codes defining functions to be performed remotely. Each processor can directly read from or write to any other processor's memory. To help manage cache coherency among processors, at the user's discretion, the support circuitry allows remote data entering a processor's local memory to invalidate the corresponding cache line.

The physical nodes are connected through a three dimensional torus. If there are k nodes in each of the three dimensions, such a network is known as a k-ary

* Published in High-Performance Computing and Networking, International Conference and Exhibition, Munich, Germany, April 18-20, 1994, Proceedings, Volume 2: Networking and Tools, W. Gentzsch and U. Harms, eds., Springer-Verlag, 1994, pp. 150-157.

** Supported in part by High Performance Computing and Communications Program, NASA Office of Aeronautics and Parallel Application Technology Program, Cray Research, Inc.

3-cube [3]. The network operates independently of the Alpha processors but at the same frequency. The network interface and the network switch, shared by two processors on a node, are Cray custom circuitry.

Data travels from node to node in packets with no processor interrupts along the way nor on arrival at the destination. The packets contain routing information directing them to the correct destination node. They follow a dimension-order routing algorithm, traveling first in the X dimension, then turning the corner to the Y dimension, and finally turning to the Z dimension. The operating system defines routing tables in each node with optimized paths between nodes.

Channels between nodes are two bytes wide operating at 300 MB/s, inbound and outbound in each of six directions. The sustainable rate, determined by the circuit logic in the network interface and the relative size of payload to routing information within each packet, is about 160 MB/s inbound to a node and about 130 MB/s outbound from a node. If both processors on a node are active, the sustainable rate is about 73 MB/s per processor.

We made our measurements on the machine installed at the Jet Propulsion Laboratory, California Institute of Technology. This machine has 128 processors arranged as a double copy of a 4-ary 3-cube: $2 \times (4 \times 4 \times 4)$.

2 Description of Experiments

We describe five experiments, labeled A-E, that measure the performance of the Cray T3D interconnection network. Some of the experiments generate little or no contention among packets on the network. Other experiments generate either moderate or heavy contention.

In experiment A, a single processor on one node writes data to a single processor on the adjacent node, one hop away. All other processors are quiet so there is no contention on the network. Data packets exit the node along one channel and acknowledgement packets return along the other channel.

In experiment B, all processors are active. Each processor picks a partner two nodes away in the X dimension, two away in the Y dimension, and two away in the Z dimension. Two processors share the network interface at each node. If the routing tables are set correctly, contention on the network is negligible.

In experiment C, each processor adds one to its own number and picks that processor as a partner. This kind of communication is common in many programs. Not all partners are one hop away. The partner for an even numbered processor is on the same node so data does not travel outside the node at all. The partner for an odd numbered processor, not at the end of a physical dimension in the torus, is one hop away. The partner for an odd numbered processor, at the end of a physical dimension in the torus, is two and sometimes three hops away. Contention on the network is moderate.

In experiment D, each processor picks a random partner. More than one processor may pick the same random partner so that occasionally several processors line up in front of the same inbound channel. Contention on the network is heavy.

In experiment E, processor zero is quiet while all other processors simultaneously write into its memory. The active processors transmit data at a low rate as they wait in line at the single channel into processor zero's memory. But processor zero sees a packet arriving from somewhere almost all the time so its incoming rate is high.

We use a basic set of communication primitives to move data from local memory on one processor to local memory on another processor. We wrote these primitives in Alpha assembler for speed but we call them from C or Fortran. Using the normal Alpha load and store instructions, they read data from local memory into local registers and write to remote memory; or they read data from remote memory into local registers and write into local memory. By setting a function code in the external circuitry, they process a long transfer by cutting it into cache line pieces and transmitting one cache line per packet onto the network. The entire transfer requires no interrupt to the operating system on either the sending or receiving side. Writing to remote memory is faster than reading from remote memory because the Alpha processor stalls on a local cache miss.

For a fixed transfer length, we measure the average time to write 1000 times to the remote processor. We flush the cache before each transfer to be sure the data travels from memory to memory and not from cache to memory. We do not include the time to flush the cache in the measured time. We subtract the overhead of the timer routine, about 0.5 μs, but do not subtract the overhead of the routine performing the actual data transfer. The measured time includes the time for an acknowledgement to return from the remote processor.

3 Results of the Experiments

From the measured time t and the transfer length n, we compute the transfer rate, $r = n/t$. Figure 1 shows the results for the transfer rates as functions of transfer size. Marked points represent the measured rates and the curves through the points are the result of a linear regression analysis described in the next section.

For experiment A, only processor zero is active and the rate approaches 126 MB/s, a value just below the hardware limit of 130 MB/s. For experiment B, the rate approaches 73 MB/s, the peak sustainable hardware limit with two active processors, implying that the routing tables are set so that there is no contention on the network. For experiment C, the rate approaches 57 MB/s, slightly below experiment B, because some processors go two or three hops rather than just one causing mild contention. For experiment D, the rate approaches 31 MB/s because random traffic causes heavy contention on the network. For experiment E, the rate into processor zero approaches 153 MB/s very close to the peak hardware limit of 160 MB/s. On the other hand, the rate out of the active processors is 127 times smaller than this value.

The measured results fluctuate slightly about their average values. To look at this behavior, we modified experiment A so that processor zero writes to each remote processor in turn. Figure 2 shows the fluctuation of the zero-length latency, defined in the next section, as a function of the location of the remote processor. The latency changes at most by four per cent for any direction in the machine. The erratic, not quite periodic, fluctuations about the average are a reflection of the physical geometry of the 4-ary 3-cube.

We also modified experiment D. For fixed length $n = 1000$ w_{64}, we measured times for a large number of transfers. Figure 3 shows the resulting discrete probability density function. The density function is fairly tight around the average $\bar{r} = 35$ MB/s with standard deviation $\sigma = 12$ MB/s. It is nonuniform with many packets hitting a wall of contention as they randomly work their way through the network.

4 Analysis of the Results

The Hockney parameters, r_∞ and $n_{1/2}$, are an excellent pair of parameters for characterizing our results [8]. To obtain these parameters, we perform linear regression on the measured data using the form

$$t = t_0 + n/r_\infty . \tag{1}$$

From the two primary quantities, t_0 the zero-length latency and r_∞ the asymptotic transfer rate, we compute the derived length

$$n_{1/2} = t_0 r_\infty . \tag{2}$$

Table 1 lists the results of the linear regression.

Table 1. Hockney Parameters

Experiment	A	B	C	D	E
r_∞ (MB/s)	126	73	57	31	153
t_0 (μ s)	2.7	2.5	2.3	2.2	0.2
$n_{1/2}$ (w_{64})	43	23	16	9	4

Dividing the transfer length n by the time t of definition (1) and using definition (2) for $n_{1/2}$, we obtain the rate in the form

$$r = r_\infty/(1 + n_{1/2}/n). \tag{3}$$

We might have performed a nonlinear regression using equation (3) to find r_∞ and $n_{1/2}$ as primary quantities. In that case, $t_0 = n_{1/2}/r_\infty$ is a derived quantity. We did not perform the nonlinear regression. We drew the curves in Figure 1 using the function defined by equation (3) with the values of the parameters in Table 1.

Sometimes the use of dimensionless variables reveals similarities among different physical experiments [1]. We have three quantities, r_∞, $n_{1/2}$ and t_0, that characterize our experiments. Only two are independent. Expressed in units such that $r_\infty = n_{1/2} = 1$, so that rates are scaled by $1/r_\infty$ and lengths by $1/n_{1/2}$, all five experiments are described by the same dimensionless function. Following Hockney [5], we define the pipe function

$$\mathrm{pipe(x)} = 1/(1 + 1/\mathrm{x}). \tag{4}$$

Then equation (3) becomes

$$r/r_\infty = \mathrm{pipe(n/n_{1/2})}. \tag{5}$$

Figure 4 shows the results of the five experiments in dimensionless form. Our five experiments are scaled variations of the same experiment described by the pipe function.

5 Summary and Conclusions

We have described a sequence of experiments to measure communication rates for the Cray T3D interconnection network. We have demonstrated that a single processor can sustain inbound transfer rates of about 153 MB/s and outbound transfer rates of about 126 MB/s. With all processors active, we have measured sustained rates of 57-73 MB/s depending on the level of contention on the network. With heavy contention, such as that encountered for random traffic, we have measured rates of about 31 MB/s. For comparison, an incomplete list of reports on communication measurements on other machines includes [2, 10, 4, 7, 6].

We have measured zero-length latencies equal to about 2.5 microseconds for memory to memory transfers. The latencies are small because the hardware supports a global address space directly available to user code. Because the startup time is small, the length required to reach half the asymptotic rate is small and the rise to the asymptote is fast. Moving small amounts of data around the machine is efficient.

We have demonstrated that the three dimensional structure of the network results in only small variations in latencies across the machine. The time required to read or write data anywhere in global memory is essentially uniform. Furthermore, the network maintains robust rates even under random traffic with heavy contention on the network. The discrete probability density functions for random traffic have interesting, nonuniform structure similar to results observed in shared memory vector machines [9].

References

1. P. W. Bridgman, *Dimensional Analysis*, Yale University Press, New Haven, 2nd ed., 1931.
2. S. Chittor and R. Enbody, *Performance evaluation of mesh-connected wormhole-routed networks for interprocessor communication in multiprocessors*, Supercomputing '90, (1990), pp. 647–656.

3. W. J. Dally, *Performance analysis of k-ary n-cube interconnection networks*, IEEE Transactions on Computers, 39 (1990), pp. 775–785.
4. R. W. Hockney, *Synchronization and Communication Overheads on the LCAP multiple FPS-164 computer System*, Parallel Computing, 9 (1988/9), pp. 279–290.
5. ———, *Performance Parameters and Benchmarking of Supercomputers*, Parallel Computing, 17 (1991), pp. 1111–1130.
6. ———, *The Communication Challenge for MPP: Intel Paragon and Meiko CS-2*, Parallel Computing, (1994), pp. –. to appear.
7. R. W. Hockney and E. A. Carmona, *Comparison of Communication on the Intel iPSC/860 and Touchstone Delta*, Parallel Computing, 18 (1992), pp. 1067–1072.
8. R. W. Hockney and C. R. Jesshope, *Parallel Computers 2: Architecture, Programming and Algorithms*, Adam Hilger/IOP Publishing, Bristol & Philadelphia, 1988.
9. R. W. Numrich, *Cray-2 memory organization and interprocessor memory contention*, in Linear Algebra, Markov Chains, and Queuing Models, C. Meyer and R. Plemmons, eds., Springer-Verlag, 1992, pp. 267–294.
10. R. Ponnusamy, R. Thakur, A. Choudhary, and G. Fox, *Scheduling regular and irregular communication patterns on the CM-5*, Supercomputing '92, (1992), pp. 394–402.
11. R. L. Sites, *Alpha Architecture Reference Manual*, Digital Press, 1992.

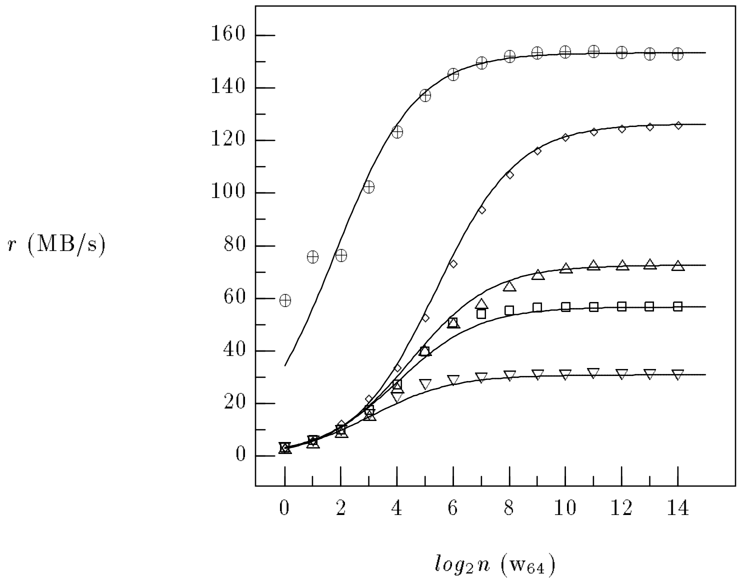

Fig. 1. Transfer rate r as a function of transfer length n for the five experiments described in section 2: ($\diamond$ A), ($\triangle$ B), ($\square$ C), ($\bigtriangledown$ D), ($\oplus$ E). The rates are average rates over all active processors except for experiment E. Its rates are the total rates into processor zero from the other 127 active processors.

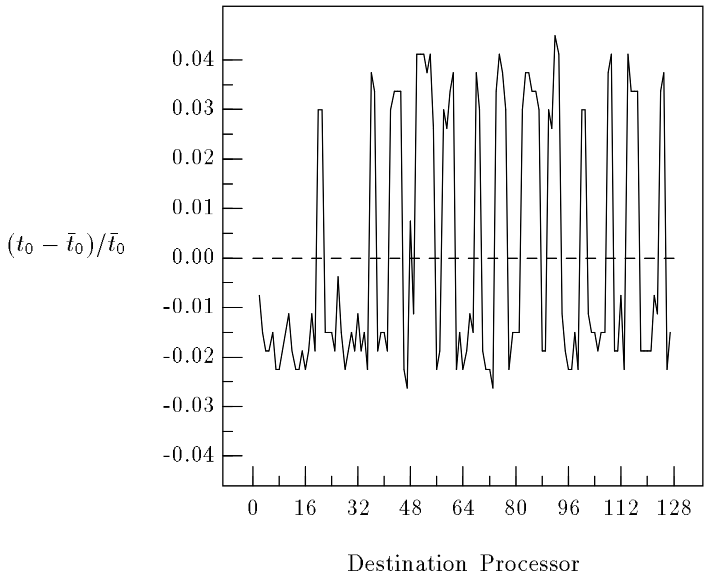

Fig. 2. Fluctuation about the average latency $\bar{t}_0 = 2.7\mu s$ as a function of destination processor as measured by modified experiment A.

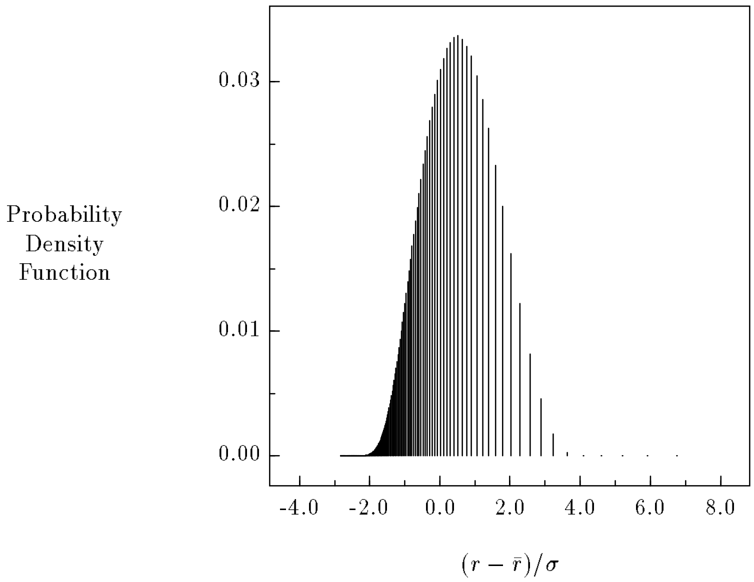

Fig. 3. Discrete probability density function, obtained from modified experiment D, for transfers of length $n = 1000$ w_{64} to random processors. The average rate is $\bar{r}$=34.6 MB/s and the standard deviation is σ=11.8 MB/s.

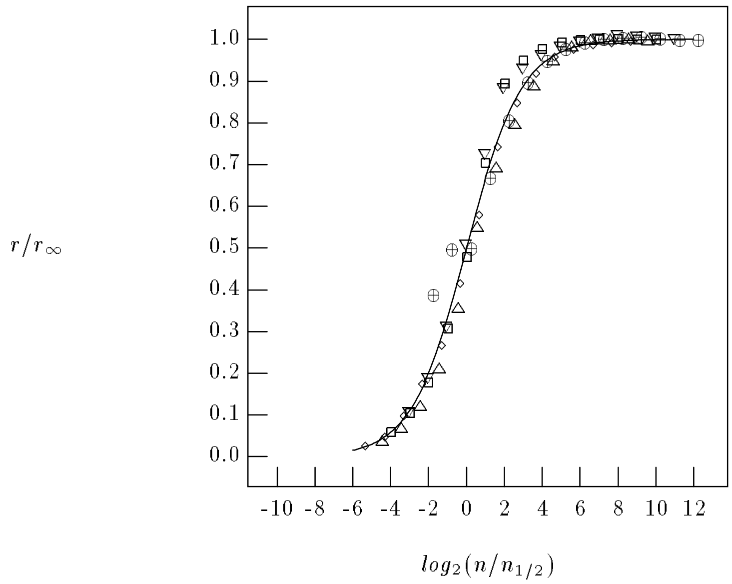

Fig. 4. Scaled transfer rate, r/r_∞, as a function of scaled transfer length, $n/n_{1/2}$ for all five experiments: ($\diamond$ A), ($\triangle$ B), ($\square$ C), ($\triangledown$ D), ($\oplus$ E). The solid curve is the pipe function defined by equation (4).