NASA Contractor Report 198335

ICASE Report No. 96-34

ICASE

EVALUATING THE HYPERBOLIC MODEL ON A VARIETY OF ARCHITECTURES

Ion Stoica
Florin Sultan
David Keyes

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

NASA Contract No. NAS1-19480
May 1996

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23681-0001

Operated by Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-0001

DTIC QUALITY INSPECTED 1

19960807 037

Evaluating the Hyperbolic Model on a Variety of Architectures

Ion Stoica
Florin Sultan
David Keyes[1]

Department of Computer Science
Old Dominion University
Norfolk, Virginia, 23529-0162
{stoica,sultan,keyes}@cs.odu.edu

Abstract

We illustrate the application of the hyperbolic model, which generalizes standard two-parameter dedicated-link models for communication costs in message-passing environments, to four rather different distributed-memory architectures: Ethernet NOW, FDDI NOW, IBM SP2, and Intel Paragon. We first evaluate the parameters of the model from simple communication patterns. Then overall communication time estimates, which compare favorably with experimental measurements, are deduced for the message traffic in a scientific application code. For transformational computing on dedicated systems, for which message traffic is describable in terms of a finite number of regular patterns, the model offers a good compromise between the competing objectives of flexibility, tractability, and reliability of prediction.

[1]This author's work was supported in part by NSF grant ECS-9527169, and by NASA contract NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE),NASA Langley Research Center, Hampton, VA 23681-0001.

1 Introduction

Most communication models are based on an empirically inferred linear dependence of the time needed to send a message between two communicating parties on the size of the message. For example, various hardware and software overheads in a parallel environment that are modeled by a fixed component, independent of the message size, and by a variable component, proportional to the message size, are identified in [1, 3, 4, 8, 9]. However, such models (with constant coefficients) cannot accommodate contention in a general fashion. Schemes for partially avoiding contention in routing architectures (e.g., a hypercube in [15]) and for obtaining probabilistic guarantees for propagation times are proposed, but the problem of quantifying the effect of coexisting messages over the same link on the end-to-end communication performance requires more attention.

The hyperbolic model [12, 13] is a variation on the two-parameter models. Its main goal is to address in a uniform way the modularity increasingly present in modern parallel computing environments, where a message path between two communicating parties crosses multiple processing modules having clearly defined interfaces and distinct functionality. If the twin parameters of every module on a message path are known (either by measurement or functional specification), the hyperbolic model allows them to be combined by a set of simple rules into a single pair of end-to-end parameters. In contrast to models that attempt to globally characterize communication costs independently of data paths, the modular hyperbolic representation is data-driven. It can take advantage of knowledge of connectivity and component parameters along the communication paths to adapt the parameters to specific patterns of communication.

2 The Hyperbolic Communication Model

Given a set of source nodes S, a set of destination nodes D, and a set of messages M in a parallel processing environment such that:

1. *every message in M is sent by a node in S to a node in D;*
2. *every node in S sends at least one message and all messages it sends are in M;*
3. *every node in D receives at least one message and all messages it receives are in M;*

our goal is to estimate for every message in M: the time interval between the sending and the delivery of a message.

This simply described task is rendered difficult in practice by the multilayeredness of a communication network, by the possibility of contention between the messages, and by message packetization. A message can be latency-bound or bandwidth-bound, depending upon its size and packet granularity, and the layer of the network that is "critical" can shift as message size varies, since each layer may have different latency and bandwidth characteristics. In systems with message contention for network paths, the *effective* latency and bandwidth seen by a given message can be functions of the other messages present. This paper describes a means of deriving just such an effective overall pair of latency and

bandwidth parameters by algebraic combination rules of component-wise parameters. We summarize the combination rules and show how they apply to a variety of communication networks and message exchange patterns.

The sets D, S, and M determine the state of the communication system, which is represented as a directed graph called a *communication graph* (CG). A CG has two types of nodes: terminal nodes and internal nodes. The terminal nodes represent the end processes that initiate the sending (source node) and receiving (destination node) of the data. Between any pair of terminal nodes the data is passed in streams of bytes of various size, called *messages*. An internal node or *Communication Block* (CB) is an abstract module that embeds all the functions performed by the communication protocols in one or more layers of software and hardware, in order to deliver data from source to destination. A CB manipulates data in units of limited size, called *packets*. Consequently, passing a message to a CB may result in splitting it into packets. We say that two or more CBs are *dependent* if they share a common resource and therefore only one of them can process data at a given moment, and *independent* otherwise. For example, two CBs running on different processors are independent, while if they run on the same processor they are dependent.

The most important parameter characterizing a CB is the time required to process a message of size x, called the *total service time*. We consider that the packet processing time has two components: a *fixed service time* that is independent of the packet size (e.g., the overhead associated with memory management, interrupt processing and context switching, the propagation delay) and an *incremental service time* that is proportional to the packet size (e.g., data movement between different protocol layers, building and verifying of the CRC or checksum, packet transmission on the communication network).

Let us consider a CB characterized by the following parameters: the maximum packet size p (bytes), the fixed service time per packet a and the incremental service time per byte m. Then the total service time t for a message of size x is given by the following equation:

$$t(x; a, m, p) = a\lceil\frac{x}{p}\rceil + mx, \tag{1}$$

where $\lceil x/p \rceil$ is the number of packets of maximum size p being processed. We approximate the total service time t with the following monotonically increasing continuous function defined on the interval $[0, \infty)$ (see Figure 1):

$$T(x; a, b) = \frac{a^2}{a + bx} + bx, \tag{2}$$

where $b \equiv a/p + m$. This is the equation of a hyperbola in the (x, t) plane, with a horizontal tangent at $x = 0$ and an asymptote of slope b, hence the name of the model. The improvement of (2) over a linear latency (α) / reciprocal transfer rate (β) model, $T(x; \alpha, \beta) = \alpha + \beta x$, is not so much in the fit of a continuous curve to the sawtooth form of a packetized transmission, but in the analytical simplicity with which the parameters (a, b) for a CG may be derived in terms of its elemental CBs, as shown by the four combination rules in subsections 2.1 through 2.4. Using T_i to estimate the total service time required by CB_i to process a message of a given size, we derive rules for reducing n CBs interconnected in various structures to a single equivalent CB , with service time $T(a_1, b_1, a_2, b_2, \ldots, a_n, b_n)$. Evaluating the reduced CG at extreme limits of message size and number of processors permits extraction of the salient parameters for the individual CBs.

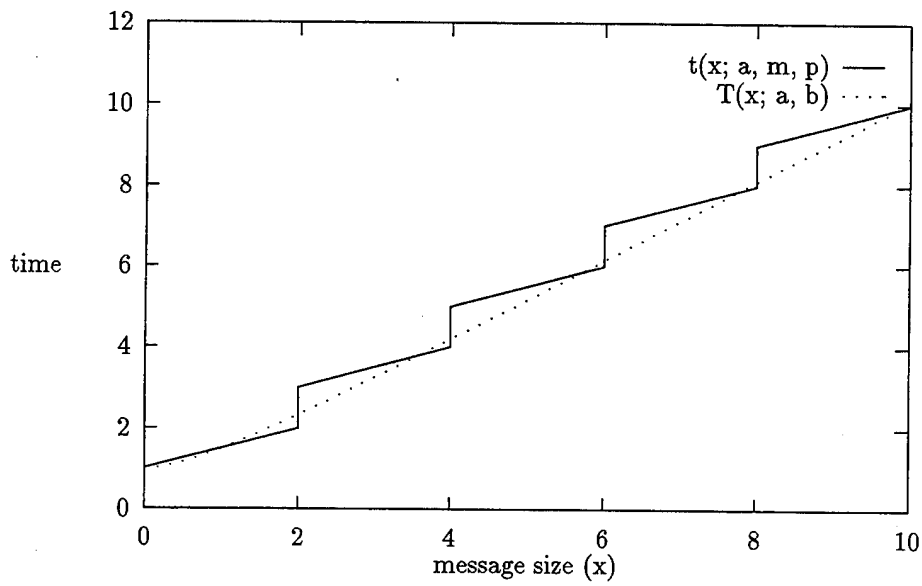

Figure 1: *The total service time t(x;a, m, p) versus the continuous function T(x; a, b) used to approximate it (a = 1, m = 0.5 and p = 2).*

A detailed discussion motivating the form of (2) and the combination rules is available in [13].

2.1 Serial Interconnection

Definition 1 *We say that n communication blocks CB_i $(1 \leq i \leq n)$ are serially interconnected with respect to a message m if every packet of m is processed sequentially by every CB_i.*

Notice that this definition does not imply that a message is processed *in its entirety* by one *CB* and only after that by the next *CB*. In fact, if the message is larger than the maximum packet size and the *CB*s are independent, as soon as CB_1 delivers a packet, CB_2 can start to process it.

Rule 1 *Given n serially interconnected communication blocks $CB_i(a_i, b_i)$, $(1 \leq i \leq n)$, this structure is equivalent to a single communication block $CB(a_I, b_I)$ (for independent blocks) or $CB(a_D, b_D)$ (for dependent blocks), where:*

$$a_I = a_D = \sum_{i=1}^{n} a_i; \quad b_I = \max\{b_1, b_2, \ldots, b_n\}; \quad b_D = \sum_{i=1}^{n} b_i \,.$$

2.2 Parallel Interconnection

Definition 2 *We say that n communication blocks CB_i $(1 \leq i \leq n)$ are parallel interconnected with respect to a message m if any packet of m can be processed by any CB_i.*

Assuming that the packets are processed such that the total service time of the message is minimized, we have the following reduction rule:

Rule 2 *Given n parallel interconnected communication blocks $CB_i(a_i, b_i)$, $(1 \leq i \leq n)$, this structure is equivalent to a single communication block $CB(a_I, b_I)$ (for independent blocks) or $CB(a_D, b_D)$ (for dependent blocks), where:*

$$a_I = a_D = \min\{a_1, a_2, \ldots, a_n\}; \quad b_I = (\sum_{i=1}^{n} \frac{1}{b_i})^{-1}; \quad b_D = \min\{b_1, b_2, \ldots, b_n\}\,.$$

We can summarize the modeling of serial and parallel interconnection on independent and dependent *CB*s as follows. In the small message limit that governs the a parameter, *CB*s in serial combine additively and *CB*s in parallel combine by taking the minimum. In the large message limit that governs the b parameter, *CB*s in serial that are dependent combine like resistors in series, and *CB*s in parallel that are independent combine like resistors in parallel. The other two subcases obey a maximum (serial, independent) or a minimum (parallel, dependent) law in deriving the overall b.

2.3 Concurrent Message Processing

We now analyze the general case in which a *CB* simultaneously receives for processing n messages m_1, m_2, ..., m_n of sizes x_1, x_2, ..., x_n. We assume that *CB* processes messages using an arbitrary policy, i.e., it first processes m_{i_1}, next m_{i_2}, and last m_{i_n}, where i_1, ..., i_n is a permutation of 1, ..., n. Since we cannot tell exactly when a particular message m_i is processed, we consider the time required to process m_i being bounded by the time required to process all messages, i.e., equivalent to the case in which m_i is the last message being processed.

Rule 3 *A communication block $CB(a, b)$ that processes n messages m_1, m_2, ..., m_n of sizes x_1, x_2, ..., x_n, respectively, is equivalent to a structure of n independent communication blocks $CB_1(a_1, b_1)$, $CB_2(a_2, b_2)$, ..., $CB_n(a_n, b_n)$ where every CB_i processes the message m_i and has parameters:*

$$a_i = na; \quad b_i = b \cdot \frac{\sum_{i=1}^{n} x_i}{x_i}\,.$$

2.4 The General Reduction Rule

The reduction rules are based on the assumption that the communication graph is identical for both small (packet size) and very large messages. Although this is true for many cases, for complex communication patterns this assumption is no longer valid (see the example of a tree-based broadcast in [13]). We therefore have the following general reduction rule, which interpolates hyperbolically between limiting cases:

Rule 4 (General Reduction Rule) *Given two terminal nodes s and d such that s sends a message m of size x to d, then the total service time for the message m is:*

$$T(x; a, b) = \frac{a^2}{a + bx} + bx,$$

where a is the service time when sending a small message from s to d ($x \to 0$), while b is the service time per data unit when sending a large message from s to d ($x \to \infty$).

3 Communication Parameters

In principle, one can determine *CB* parameters a and b by considering the hardware characteristics of the computation nodes and the communication network (e.g., the processor speed, the memory access time, the internal bus speed, etc.) and the communication protocol implementation details (e.g., the number of times a data buffer is copied while passed through various protocol layers, the algorithm used to compute the checksum, etc.). Although this approach appears to allow accurate evaluation of *CB* parameters, it is hard to apply in practice because of several factors:

- Various layers of the communication architecture are embedded in the general purpose operating systems running on the processing nodes. This makes them compete for system resources with other processes in the multitasking environment. It also means that various factors like interrupt processing, context switching, memory management, etc., combined with hardware features like the presence of a cache memory system, would have to be considered when trying to model the communication.
- Systems may be heterogeneous (made up of machines from different vendors, with different characteristics and running different operating systems).
- Software packages, such as the support for communication between end processes (at the application level), each having their own characteristics and introducing their own overhead, would have to be represented in a detailed model.

For four distributed-memory computing systems, Ethernet NOW (Network of Workstations), FDDI NOW, the IBM SP2, and the Intel Paragon, we illustrate the combination rules, and invert them to derive the salient parameters for individual *CB*s from convenient end-to-end measurements of limiting cases.

3.1 Ethernet Network of Workstations

Let us consider a network of n identical workstations linked by a communication network $CB_c(a_c, b_c)$. For simplicity, assume that the overheads for sending and receiving messages are equal. Thus, all workstations are modeled by the same $CB(a_w, b_w)$ irrespective of whether a message is being sent or received. We consider two communication patterns.

For the first pattern, we measure the round-trip time between two workstations. Let $RTT(x)$ be the round-trip time measured for a message of size x. Then, by symmetry, the transmission time of a message from one workstation to another, which is not directly

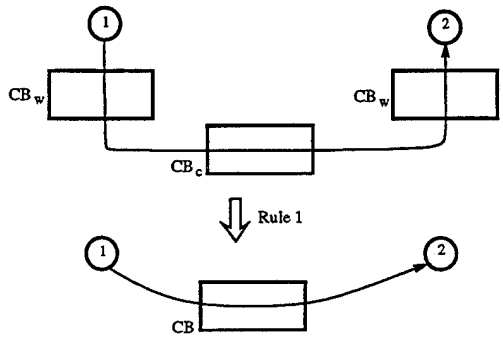

Figure 2: *The communication graph and its equivalent CB for sending one message from process 1 to process 2.*

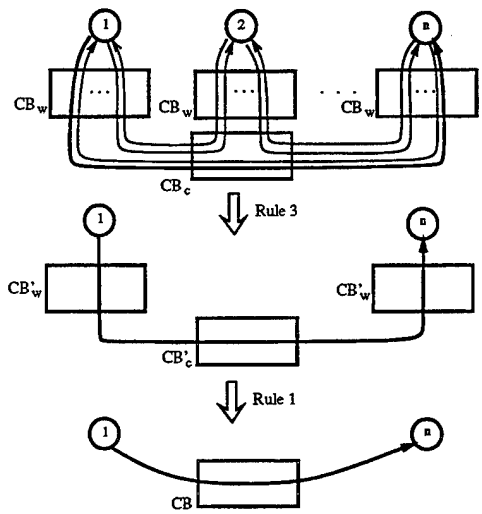

Figure 3: *The communication graph and its reduction to an equivalent CB for the case when each process sends a message to all the other processes over an Ethernet network.*

measurable on a single clock, is $RTT(x)/2$. Since, as shown in Figure 2, the corresponding communication graph can easily be reduced (using Rule 1) to an equivalent communication block $CB(a, b)$, we have

$$\begin{aligned} a &\equiv \frac{RTT(0)}{2} = 2a_w + a_c, \\ b &\equiv \lim_{x \to \infty} \frac{RTT(x)}{2} = \max\{b_w, b_c\}. \end{aligned} \tag{3}$$

The second pattern consists of every workstation sending/receiving a message to/from all the other workstations. More precisely, given n workstations, each workstation sends $n-1$ messages of the same size to each other workstation, after which it waits to receive all the messages addressed to it. Figure 3 depicts the communication pattern, as well as the transformations for the communication graph involving sending one message from node 1 to node n. First, the original graph is reduced by using Rule 3 to an intermediate graph

consisting of three modules: two modules $CB'_w(a'_w, b'_w)$ which represent the end-nodes, and one $CB'_c(a'_c, b'_c)$ representing the communication network. Since all the messages have the same size, and since a workstation sends $n-1$ messages and receives another $n-1$ messages, by Rule 3 we obtain $a'_w = 2(n-1)a_w$, $b'_w = 2(n-1)b_w$. Similarly, it is easy to see that the total number of messages "injected" into the communication network (i.e., into block CB_c) is $n(n-1)$, and therefore we obtain: $a'_c = n(n-1)a_c$, $b'_c = n(n-1)b_c$. Next, by using Rule 1 we reduce the intermediate communication block to one equivalent CB with the following parameters:

$$\begin{aligned} a(n) &= 4(n-1)a_w + n(n-1)a_c, \\ b(n) &= \max\{(2(n-1)b_w, n(n-1)b_c\}, \end{aligned} \tag{4}$$

where the parameters of the resulting CB are expressed as a function of the number of nodes.

Since in a distributed network of workstations we do not have a global clock, we cannot directly measure the time taken to send a message from workstation 1 to workstation n. Instead, we synchronize all the workstations to begin transmission at the same time, and we average the times measured on *each* workstation from the moment the first message is sent, until the last one is received.[2] The decision to consider these times is motivated by the fact that in an "ideal" system all the workstations will begin sending and will finish receiving at the same time. Moreover, notice that this is in accordance with our assumptions made in deriving Rule 3, i.e., when a CB processes concurrent messages we assume conservatively that the message we are studying is the *last* one that is processed. Let $T(x,n)$ be the average time measured between the moment an workstation initiates the sending of the first message, and the moment it receives the last message. Since, by Rule 4, $T(0,n) = a(n)$, and $\lim_{x\to\infty} T(x,n) = b(n)$, from Eqs. (4) we obtain the following relations:

$$\begin{aligned} a_c &= \lim_{n\to\infty} \frac{a(n)}{n(n-1)} = \lim_{n\to\infty} \frac{T(0,n)}{n(n-1)}, \\ b_c &= \lim_{n\to\infty} \frac{b(n)}{n(n-1)} = \lim_{n\to\infty} \frac{\lim_{x\to\infty} T(x,n)}{n(n-1)}. \end{aligned} \tag{5}$$

Thus, if we have enough workstations, by sending enough large messages, we should be able to compute the communication network parameters from Eqs. (3) and (5). Unfortunately, in practice we cannot use unlimited resources to compute these parameters exactly. We remark, however, that it is easier to compute b_c accurately than a_c. For computing b_c we use very large messages, and since the time to do the synchronization is much smaller than the time required to send/receive such messages, we can neglect the synchronization time even for a small number of workstations. In fact, in our Ethernet setting, we can accurately compute b_c using only 3 workstations. This is not true when computing a_c, since the time to send a small message is comparable to the time to do the synchronization; in fact, the synchronization is, itself, implemented by using very small messages. Therefore, to accurately compute a_c, we need a much larger number of nodes. In practice, as a rule of thumb, we consider that we have enough workstations to compute the value of a_c when

[2]The reason we take the average here is to account for the probabilistic behavior of the CSMA/CD protocol employed by the Ethernet [10]; on all the other platforms (Sections 3.2, 3.3. and 3.4) we take the maximum over the measured times.

adding a new workstation changes the value of a_c by less than 5%. In our experiments we use eight workstations for computing a_c.

From Eq. (3), we can easily determine a_w, and $\max\{b_c, b_w\}$, in addition to a_c and b_c. Note that if b_w is smaller than b_c, then b_w is not directly available from these experiments. However, this is not a problem in practice, since this means that when large messages are sent, the communication network is *always* the bottleneck, and therefore b_w is "shadowed" by b_c.

For a group of Sun SPARCstation 20s at ICASE running SunOS 4.1.3, using MPI (as implemented in Argonne's MPICH [7]) as the communication layer, the parameters are:

$$\begin{aligned} a_c &= 250\ \mu\text{sec}, \\ b_c &= 0.95\ \mu\text{sec/byte}, \\ a_w &= 750\ \mu\text{sec}, \\ b_w &= 1.05\ \mu\text{sec/byte}. \end{aligned} \tag{6}$$

We note that $1/b_c$ is only about 10% slower than the theoretical peak performance of Ethernet, virtually the same performance realization reported in [11] in a different workstation environment. We expect the b_w parameter of the present workstations to be visible only when there is low contention, since it is within a factor of two of b_c.

3.2 FDDI Network of Workstations

To determine the corresponding parameters for FDDI we use a similar experimental setting: eight Sun SPARCstation 20s running SunOS 4.1.3, and using MPI on FDDI as the communication layer. Of the many differences between Ethernet and FDDI technology, the most apparent to the user is FDDI's 100 Mb/sec peak bandwidth versus 10 Mb/sec for Ethernet.

By using the same two communication patterns, we can compute a_c, b_c, and $2a_w + a_c$ in the same manner as above. However, it is much harder to compute a_c accurately. The reason is that the access time to the network in the FDDI case is much shorter than for Ethernet, since there are no collisions to delay the packet transmission[3]. In fact, it can be shown that the maximum access time to the network, a_c, is given by the propagation delay of a *token* along the entire network. Since in our case, the length of the network is under 1000 meters, the a_c should be less than 5 μsec, which is two orders of magnitude less than the value of $2a_w + a_c$. Therefore, we will simply neglect a_c. With this, the parameters for FDDI are:

$$\begin{aligned} a_c &= 5\ \mu\text{sec}, \\ b_c &= 0.11\ \mu\text{sec/byte}, \\ a_w &= 380\ \mu\text{sec}, \\ b_w &= 0.13\ \mu\text{sec/byte}. \end{aligned} \tag{7}$$

The throughput for FDDI is one order of magnitude larger than for Ethernet, as expected. However, we point out that these results were possible only after patching the MPICH

[3]Note that the access time here does not include the time that an workstation has to wait while the communication network is used by another workstation, since this effect is already modeled in the hyperbolic model by Rule 3.

release [7] of the MPI software. Specifically, we had to increase the size of the TCP frame buffer, which is configured through MPI, from 4 KB to 40 KB. After this change, the throughput increased by a factor of almost *five*.

3.3 IBM SP2

The SP2 communication architecture [14] is based on a high-performance (low latency, high bandwidth) switching network called the High-Performance Switch. The topology consists of non-switching nodes (mainly processors for our purpose) connected through a multistage network of switching elements. Each element is a 4×4 bidirectional switch with 8 input and 8 output ports.

Each node of an SP2 system belongs to a *logical frame*. A frame is a two-stage interconnect that provides any permutation of 16 bidirectional links to/from 16 processors. Multiple frames can be further interconnected, allowing full connectivity throughout the resulting network. A property of this network is that for each pair of nodes there are at least four possible paths (but not all of the same length), unless the nodes are attached to the same switching element. Another observation is that the basic switching elements are potential bottleneck points due to multiplexing of packets from multiple sources on their limited resources. Thus, contention may occur for two message paths traversing the same link (input port to output port) in a switch, with the side effects of increased delay and reduced throughput due to buffering in the switch.

To determine the componentwise hyperbolic model parameters for the SP2 system[4], we initially attempted to use the same communication patterns as for FDDI and Ethernet. Unfortunately, we are not able to run the experiments for the second communication pattern, in which every node is supposed to send messages to all the others, for large messages. We suspect buffer overflow within the communication switch is the cause. This is not a practical problem from the point of view of evaluating CB parameters, however; we simply replace the second communication pattern with a new one in which each node sends/receives a message to/from exactly one other node.[5] More precisely, consider $2n$ nodes numbered from 1 to $2n$, and let each node i from the first half send a message to the corresponding node $n+i$ from the second half, $i = 1, 2, \ldots n$.

Next, let us consider a pair of nodes that communicate with each other, such as nodes 1 and $n+1$. Then, we can determine the service time $T(x, n)$ required to deliver a message of size x from node 1 to node $n+1$ by reducing the initial communication graph to an equivalent CB, as shown in Figure 4. It is easy to check that the communication parameters of the resulting CB as a function of the number of nodes are:

$$\begin{aligned} a(n) &= 2a_p + na_c, \\ b(n) &= \max\{b_p, nb_c\}. \end{aligned} \tag{8}$$

Since $T(0, n) = a(n)$ and $\lim_{x\to\infty} T(x, n) = b(n)$, by using the Eq. (8) we can write:

[4] We ran our experiments on the NASA LaRC 48-node (wide-node) SP2 system.

[5] On the other hand, we were not able to use the same sparse communication pattern for the Ethernet NOW and FDDI NOW, since the number of workstations available was insufficient to produce the communication bottleneck that determines the a_c and b_c parameters (see (5)).

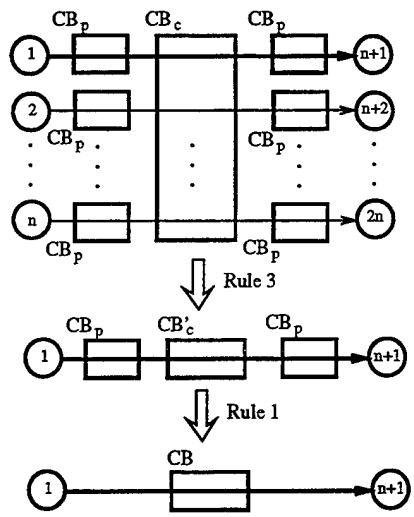

Figure 4: *The communication graph and its reduction to an equivalent CB for the case when each process with index i ($i \leq n$) sends a message to the process with index $n+i$.*

$$a_c = \lim_{n\to\infty} \frac{T(0,n)}{n}, \tag{9}$$

$$b_c = \lim_{n\to\infty} \frac{\lim_{x\to\infty} T(x,n)}{n}. \tag{10}$$

By the nature of its interconnection topology, bisection bandwidth in an SP2 system increases with the number of nodes. This is different from FDDI and Ethernet, where the communication bandwidth is independent of the number of nodes. (In fact, for Ethernet, the bandwidth might slightly decrease as we add more nodes due to the increasing number of collisions.) Thus, the CB_c parameters for SP2 are dependent on the interconnection topology.

In the remainder of this subsection we consider two basic configurations, one with 14 nodes located on the same frame[6], and the other with 28 nodes equally divided among two frames. Since each frame provides any permutation of 16 bidirectional links, in the first case we can assume that each pair of nodes communicates through its own communication channel. Since there are seven such pairs, b_c can be simply written as $b_l/7$, where b_l denotes the inverse of an individual channel bandwidth. By performing measurements for messages as large as 0.5MB, we obtain $b(7) = \max\{b_p, 7b_c\} = \max\{b_p, b_l\} = 0.029$ μsec/byte. Also, by measuring the round-trip time for very short messages we obtain $2a_p + a_c = 124\mu sec$.

In the second case, we consider 14 processors located on one frame that communicate with 14 processors located on another frame. Since, as shown in [14], there are only eight communication channels available between two adjacent frames, the overall bandwidth will *not* increase when the number of pairs that communicate between frames exceeds eight. Thus, the overall value of b_c in this case should be $b_l/8$, for any number

[6]We avoid using the base nodes in each frame because they participate in other facility-wide networks (FDDI, HiPPI, Ethernet) in addition to the internal network.

of pairs greater or equal to eight. Finally, since there are 14 pairs of processors, we have $b(14) = \max\{b_p, 14b_c\} = \max\{b_p, 7b_l/4\}$. According to our measurements we obtain $b(14) = 0.049$ μsec/byte. Now notice that since from the previous experiment $b(7) = 0.029$ μsec/byte, b_p cannot be larger than $7b_l/4$, and therefore we have $b_l = 0.028$ μsec/byte. Since this value is very close to the one obtained for $b(7)$, we will assume that $b_l = 0.29$, and that b_p is less or equal to b_l.

The values above are very close to those found in various technical papers describing the architecture and the performances of SP2 communication system, and of the MPI package running on SP2 [2]. In fact, the corresponding parameters quoted in [14] are:

$$\begin{aligned} a_c &= 5 - 35\ \mu\text{sec}, \\ b_l &= 0.0285\ \mu\text{sec/byte}, \\ a_p &= 40\ \mu\text{sec}, \\ b_p &= 0.025\ \mu\text{sec/byte}. \end{aligned} \tag{11}$$

which are consistent with those we measure via the hyperbolic model, i.e., $b_l = 0.029$ μsec/byte, and $2a_p + a_c = 124\mu$sec.

3.4 Intel Paragon

The Intel Paragon communication architecture is based on a rectangular 2-D mesh interconnection network. Each computing node is attached to an associated communication processor through a private bidirectional communication link. Every communication processor is connected with other four adjacent communication processors over bidirectional network channels. It implements wormhole routing functions by forwarding packets received on incoming links from its neighbors and from its attached computing node.

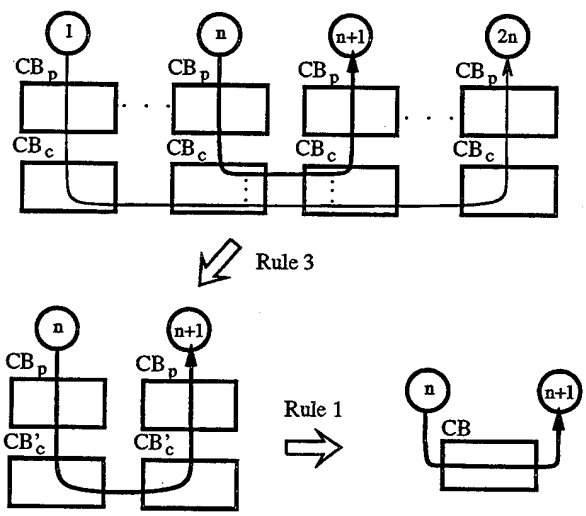

Figure 5: *The communication graph and its reduction to an equivalent CB for the case when each process with the index i ($i \leq n$) sends a message to the process with the index $2n - i + 1$.*

To determine the Paragon[7] parameters we use the same communication patterns as for the SP2. In our experiments, all the processing nodes involved in the communication are placed along the same row of the communication mesh. In this way we are able to maximize contention since all the messages are sent along the communication links corresponding to that row, according to the X-Y routing policy of the Paragon. Figure 5 shows the original communication graph as well as the derivation of the equivalent CB involving the end-nodes n and $n+1$. The communication blocks CB_c model the communication processors that perform routing functions. Since n messages pass through the corresponding CB_cs of nodes n and $n+1$, the resulting communication parameters for the equivalent CB are:

$$\begin{aligned} a(n) &= 2a_p + 2na_c, \\ b(n) &= \max\{b_p, nb_c\}. \end{aligned} \tag{12}$$

By denoting the time required to send a message of size x between two end-nodes as $T(x,n)$, we obtain from Eq. (12) that

$$a_c = \lim_{n\to\infty} \frac{T(0,n)}{2n}, \tag{13}$$

$$b_c = \lim_{n\to\infty} \frac{\lim_{x\to\infty} T(x,n)}{n}. \tag{14}$$

By using similar observations as in the SP2 case we determine the following parameters:

$$\begin{aligned} a_p + a_c &= 120\ \mu\text{sec}, \\ b_c &= 0.012\ \mu\text{sec/byte}, \\ b_p &= 0.031\ \mu\text{sec/byte}. \end{aligned} \tag{15}$$

Again, we cannot isolate a_c because it is not possible to achieve a bottleneck for very small messages.

4 Test Application

A model parallel scientific application originally written for the Intel Hypercube by Horton [5] and rewritten to take advantage of the multiplatform implementation of the MPI standard is instrumented and used as a test program for the hyperbolic model. A multigrid (MG) code for transient flow in a cavity with an oscillating lid was chosen among conveniently available codes for its simplicity and for its scaling properties in communication requirements. We describe the application just sufficiently to expose the leading-order communication complexity and to appreciate its general context. To verify the accuracy of our model, we select for graphical comparison estimates of the communication times and corresponding measurements. The estimates derive from the archetypal communication operations as described in [13], with parameters evaluated for each platform as in section 3.

[7] We ran our experiments on the NASA LaRC 72-node Paragon system.

4.1 Multigrid

The model application is transient simulation of incompressible Navier-Stokes flow in a two-dimensional square cavity filled with fluid, driven by a sinusoidally oscillating rigid lid. The numerical method is based on a standard uniform-grid spatial discretization and implicit time discretization for a velocity-pressure formulation of Navier-Stokes with a hybrid space-parallel/time-parallel multigrid (MG) solver. A MG solver uses a succession of grids presenting different refinements of the same problem, in order to iteratively damp the component of the error at each wavenumber on the grid for which its particular damping factor is most rapid, rather than damping all error components on the same grid. Space parallelism is achieved through domain partitioning, with one processor per subdomain; we permit both stripwise and boxwise decomposition in order to obtain more flexibility in the number of subdomains while still preserving the uniformity of each subdomain. Time parallelism is achieved by assigning identically spatially decomposed time planes to disjoint sets of processors. The motivation for time parallelism is the degradation of efficiency in space parallelism that is due both to degrading perimeter-to-area (or surface-to-volume) ratios of conventional implicit methods, and to degrading convergence rate as global coupling is sacrificed in the MG "smoother," which is the error-reducing operation at the heart of MG, performed on a partition of a grid at a given refinement level. The effectiveness of time parallelism is physically counter-intuitive because of causality. Nonetheless, it is more effective than space parallelism in many practical physical and numerical parameter ranges, when time-accurate resolution of a transient flow is required.

In the limit of pure time parallelism, p processors work concurrently on p different time planes of the transient solution. In the limit of pure space parallelism, only one time plane is computed at a time. Multigrid is used in the spatial direction only; there is no time coarsening. (Time coarsening is worthy of attention in other contexts (see, e.g., [6]), but is irrelevant to our immediate purpose for this application, namely to introduce a communication complexity that scales to the same asymptotic order in problem size as the computation complexity.)

The communication patterns and the amount of traffic vary with the allocation of available processors between space and time, as well as with the refinement of the spatial grid, with the result that a wide range of message sizes, message numbers and message patterns are observed, depending on three factors: the number of physical time steps simultaneously solved for, the number of domain partitions, and the number of spatial coarsening levels. The most important observation about the computation and communication complexity, however, is that their asymptotic sizes are of equal order. Consider the purely time parallel limit of p planes of $n \times n$ gridpoints. Transferring the full plane of data between time levels is an $\mathcal{O}(n^2)$ operation, which is the same as the $\mathcal{O}(n^2)$ arithmetic complexity of the stencil operations of residual evaluation and ILU smoothing in the fine grid sweep of the MG algorithm.

4.2 Communication Parameters

In this section we present the derivations of the communication parameters for the traffic consisting of inter-plane grid transfers which, as shown before, is dominant in overall communication complexity. On a homogeneous set of processors this dominant pattern should

exhibit contention since all transfers will start at "almost" the same time.

Figure 6 shows the communication graphs corresponding to this communication pattern for Ethernet/FDDI, IBM SP2, and Intel Paragon, as well as the reduction to an equivalent CB. In the case of the SP2 we assume that every pair of nodes communicates through a "dedicated" link (represented by CB_l). It is, in fact, possible to arrange that consecutive nodes be allocated on the same frame, which drastically reduces the inter-frame communication load. On the other hand, recall that for the processing nodes that reside on the same frame, the communication bottleneck is not a problem, since the SP2 communication system ensures 16 communication links per frame.

Table 1 shows the expressions of the communication parameters for the intermediate communication graph, as well as for the final CB. These parameters are computed for the bi-directional case where neither nodes i nor $i+1$ are associated with the first or last planes (i.e., $i = 2, 3, \ldots, n-2$). For the first and last planes, communication is uni-directional only, the parameters can be computed in the same manner.

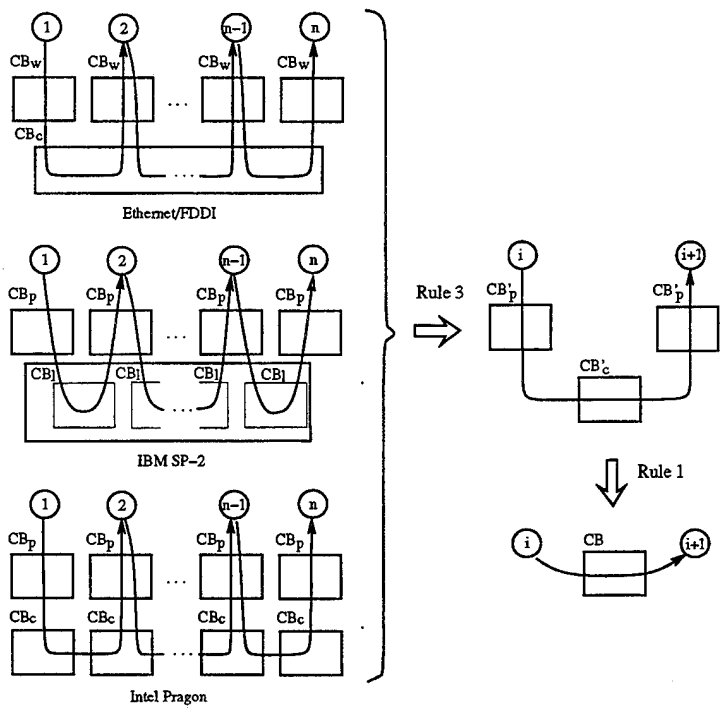

Figure 6: *The communication graph and its reduction to an equivalent CB for the main communication pattern induced by the multigrid application in the time-parallel limit for: Ethernet/FDDI, IBM SP2 and Intel Paragon*

4.3 Experimental Data

We have ported the multigrid application to all four platforms evaluated in Section 3: Ethernet NOW, FDDI NOW, IBM SP2, and Intel Paragon. Since it is available on all platforms,

Architecture	CB'_p		CB'_c		CB	
	a'_p	b'_p	a'_c	b'_c	a	b
Ethernet/FDDI	$2\ a_w$	$2\ b_w$	$(n-1)a_c$	$(n-1)b_c$	$4a_w + (n-1)a_c$	$\max\{2b_w, (n-1)b_c\}$
IBM SP2	$2\ a_p$	$2\ b_p$	a_l	b_l	$4\ a_p + a_l$	$\max\{2b_p, b_l\}$
Intel Paragon	$2\ a_p$	$2\ b_p$	$2\ a_c$	$2\ b_c$	$4\ a_p + 2\ a_c$	$\max\{2b_p, 2b_c\}$

Table 1: *The communication parameters for the CBs shown in Figure 6.*

we use MPI as the communication library. To predict the overall communication times, we use the hyperbolic model to estimate the times for the basic communication pattern, by reducing the corresponding communication graphs (see Figure 6) to an equivalent *CB*, whose parameters were computed based on the values determined in Section 3.

For Ethernet and FDDI we run the experiments on up to 8 SUN SPARCstation 20s. For the SP2 and Paragon, in choosing the maximum numbers of nodes, our goal is to minimize as much as possible the interference of other users. Therefore, on the SP2 we run the experiments by using up to the maximum number of processors on a frame (i.e., 16), while for Paragon we run the experiments on up to 12 nodes, since this is the maximum number of nodes we can allocate along a mesh column. (By allocating all the nodes on one side of the mesh we eliminate any potential interference.)

Figure 7 shows the predicted versus the measured values for the total communication times corresponding to one iteration in the algorithm. We note that for FDDI NOW, IBM SP2, and Intel Paragon the predicted data are within 20% of the measurements, with the exception of the experiment running on seven processors on FDDI. We believe that this was primarily due to communication interference from other workstations on the same subnet. (We reserved only the individual workstations; we could not reserve the entire subnet.) Upon running multiple tests, we expect, statistically, to draw the 7-processor point down to the rest of the curve.

On the other hand, for Ethernet the difference between the predicted data and the measurements (the "asynchronous" curve) is much larger and tends to increase with the number of processors. The main reason is that the hyperbolic model assumes that all processors send data at the same time, which yields theoretically an upper bound on the communication time. While the multigrid application is inherently synchronous, in practice the probabilistic protocol employed by Ethernet "destroys" the synchronicity. This is because at the beginning of each iteration all workstations attempt to send messages "almost" at the same time, and therefore the probability of collision is high. When a workstation detects such a collision, it backs off and waits for a random amount of time [10], before retrying to send the message. In time, this results in workstations sending out messages at slightly staggered intervals. Consequently, the degree of overlap between messages sent by different processors is much lower than is assumed in the model, which results in the observed communication time discrepancy. For validation purposes, we can change the algorithm to force synchronization at intermediate points during an iteration. As shown in Figure 7, the measured overall communication time in this case (the "synchronous" curve) is very close to the predicted value.

The key contrast between the NOWs and the tightly-coupled machines, as predicted

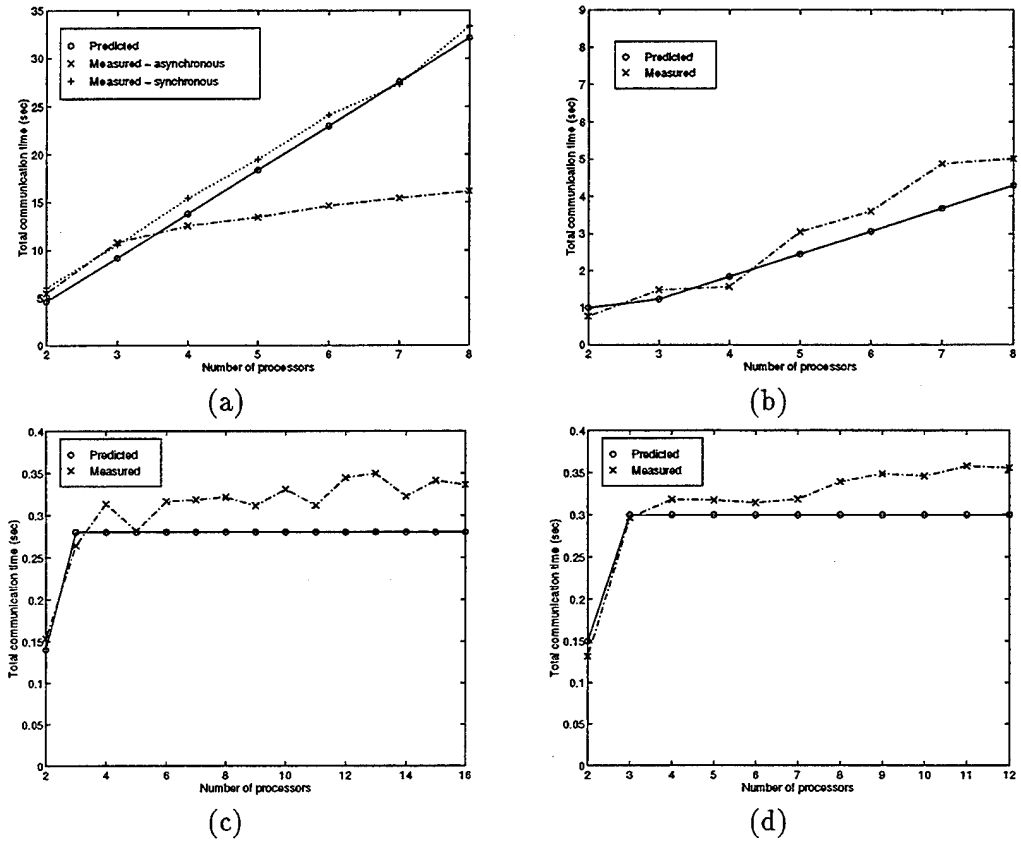

Figure 7: *Predicted versus measured communication times for: (a) Ethernet NOW, (b) FDDI NOW, (c) IBM SP2, (d) Intel Paragon.*

by the model and as borne out in the experiments, is in the asymptotic behavior of the communication time with respect to the number of processors. For the NOWs it is linear, since the communication network is a shared resource of limited capacity: adding more nodes decreases the share of communication bandwidth allocated to each processor, and the communication time consequently increases. On the other hand, for the SP2 and the Paragon, the communication times remain practically constant as the number of nodes increases. This is expected from the scalability of the communication subsystems employed by these platforms.

The difference in the time to complete the communication in going from two to three processors for the SP2 and the Paragon is due to the fact that for more than two processors there is at least one processor (the one in the middle) which both sends and receives data. On the other hand, in the two-processor case each processor either sends or receives (but not both), which reduces by nearly half the overall message processing time at the end nodes.

5 Conclusions

The two-parameter hyperbolic model [12, 13] for parallel communication complexity on general dedicated networks has been applied, using a uniform set of rules, to different communication architectures, including clusters of workstations and dedicated parallel machines. Under different message topologies on each platform the rules reduce to different analytical expressions of overall communication parameters. Sample experimental procedures for deriving parameters of elemental communication blocks are demonstrated. The model has proved to be flexible and reasonably accurate in predicting the communication times on a variety of distributed-memory systems in the context of a real-world application.

References

[1] Dunigan, T. H. Performance of the Intel iPSC/860 and Ncube 6400 hypercubes. *Parallel Computing*, **17** (1991), 1285-1302.

[2] Franke H., Wu C. E., Riviere M., Pattnaik P., Snir M., MPI Programming Environment for IBM SP1/2, URL: http://ibm.tc.cornell.edu/ibm/pps/doc/mpif/paper_mpif.ps.

[3] Hockney, R. W. Performance Parameters and Benchmarking of Supercomputers. *Parallel Computing*, **17** (1991), 1111-1130.

[4] Hockney, R. W., and Curington, I. J. $f_{1/2}$: A Parameter to Characterize Memory and Communication Bottlenecks. *Parallel Computing*, **10** (1989), 277-286.

[5] Horton, G. Time-parallelism for the massively parallel solution of parabolic PDEs, *Applications of High Performance Computers in Science and Engineering, Computational Mechanics Publications*, Southampton (UK), December 1994.

[6] Horton G., and Vandewalle S. A Space-Time Multigrid Method for Parabolic PDEs, Report 6/93, IMMD 3, Universitaet Erlangen, July 1993 (to appear in *SIAM J. Sci. Comput.*).

[7] *MPICH World Wide Web home page.* http://www.mcs.anl.gov/home/lusk/mpich/index.html.

[8] Maly, K., Khanna, S., Foudriat, E. C., Overstreet, C. M., Mukkamala, R., Zubair, M., Yerraballi R., and Sudheer, D. Parallel Communications: An Experimental Study, TR-93-20 Dept. of Comp. Sci., Old Dominion Univ., 1993.

[9] Ramakrishnan, K. K. Performance studies in designing Network Interfaces: A Case Study, *Proc. of the 4th IFIP Conference on High Performance Networking '92*, A. Danthine and O. Spaniol, (Eds.). *Int. Fed. for Information Processing*, 1992, pp. F3-1 – F3-15.

[10] Stallings, W. *Data and Computer Communications*, Macmillan, New York, 1991.

[11] Stevens, W. R. *TCP/IP Illustrated vol. 1, The Protocols*, Addison-Wesley, Reading, MA, 1994.

[12] Stoica, I., Sultan, F., Keyes, D. Modeling Communication in Cluster Computing. *Proceedings of the 7th SIAM Conference on Parallel Processing for Scientific Computing*, San Francisco, CA, February 15-17 1995, pp. 820-825.

[13] Stoica, I., Sultan, F., Keyes, D. A Simple Hyperbolic Model for Communication in Parallel Processing Environments, *ICASE TR 94-78*, Institute for Computer Applications in Science and Engineering, September 1994. To appear in *J. Par. Dist. Comput.*, 1996, under the title "A Hyperbolic Model for Communication in Layered Parallel Processing Environments".

[14] Stunkel, C. B., Shea G. D., Abali B., Atkins M., Bender C. A., Grice D. G., Hochschild P. H., Joseph D. J., Nathanson B. J., Swetz R. A., Stucke R. F., Tsao M., Varker P. R. The SP2 Communication Subsystem, URL: http://ibm.tc.cornell.edu/ibm/pps/doc/css/css.ps.

[15] Valiant, L. G. A Bridging Model for Parallel Computation. *Communications of the ACM*, **33** (1990), 103-111.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
May 1996

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
EVALUATING THE HYPERBOLIC MODEL
ON A VARIETY OF ARCHITECTURES

5. FUNDING NUMBERS
C NAS1-19480
WU 505-90-52-01

6. AUTHOR(S)
Ion Stoica
Florin Sultan
David Keyes

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Computer Applications in Science and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
ICASE Report No. 96-34

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-198335
ICASE Report No. 96-34

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Dennis M. Bushnell
Final Report
To appear in the Proceedings of Europar '96.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited

Subject Category 60, 61

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

We illustrate the application of the hyperbolic model, which generalizes standard two-parameter dedicated-link models for communication costs in message-passing environments, to four rather different distributed-memory architectures: Ethernet NOW, FDDI NOW, IBM SP2, and Intel Paragon. We first evaluate the parameters of the model from simple communication patterns. Then over-all communication time estimates, which compare favorably with experimental measurements, are deduced for the message traffic in a scientific application code. For transformational computing on dedicated systems, for which message traffic is describable in terms of a finite number of regular patterns, the model offers a good compromise between the competing objectives of flexibility, tractability, and reliability of prediction.

14. SUBJECT TERMS
communication modeling; parallel scientific computing; network of workstations; Ethernet; FDDI; IBM SP2; Intel Paragon

15. NUMBER OF PAGES
20

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT