Efficient Communication Using Total-Exchange

Satish Rao
NEC Research Institute
4 Independence Way
Princeton, NJ 08540

Torsten Suel
NEC Research Institute
4 Independence Way
Princeton, NJ 08540

Thanasis Tsantilas[†]
Dept. of Computer Science
Columbia University
New York, NY 10027

Mark Goudreau
Dept. of Computer Science
University of Central Florida
Orlando, FL 32816

Summary

A central question in parallel computing is to determine the extent to which one can write parallel programs using a high-level, general-purpose, and architecture-independent programming language and have them executed on a variety of parallel and distributed architectures without sacrificing efficiency. A large body of research suggests that, at least in theory, general-purpose parallel computing is indeed possible provided certain conditions are met: an excess of logical parallelism in the program, and the ability of the target architecture to efficiently realize balanced communication patterns.

The canonical example of a balanced communication pattern is an h-relation, in which each processor is the origin and destination of at most h messages. A plethora of protocols has been designed for routing h-relations in a variety of networks. The goal has been to minimize the value of h while guaranteeing delivery of the messages within time a constant factor from optimal. In this paper we describe protocols that meet the most stringent efficiency requirement, namely delivery of messages within time that is a lower order additive term from the best achievable. Such protocols are called 1-optimal. While these protocols achieve 1-optimality only for heavily loaded networks, that is, for large values of h, they are remarkable for their simplicity in that they only use the total-exchange communication primitive. The total-exchange can be realized in many networks using very simple, contention-free, and extremely efficient schemes.

The technical contribution of this paper is a protocol to route random h-relations in an N-processor network using $\frac{h}{N}(1+o(1)) + O(\log\log N)$ total-exchange rounds with high probability. Using message duplication, we can improve the bound to $\frac{h}{N}(1+o(1)) + O(\log^* N)$. This improves upon the $\frac{h}{N}(1+o(1)) + O(\log N)$ bound of Gerbessiotis and Valiant. While our theoretical improvements are modest, our experimental results show an improvement over the protocol of Gerebessiotis and Valiant.

1 Introduction

In this paper we study the total-exchange communication pattern in the context of general-purpose parallel computing. Under this communication pattern, every processor has a distinct message to send to every other processor. The efficient implementation of the total-exchange in a variety of networks has been extensively studied [16, 4, 14]. These studies were primarily motivated by the ubiquitousness of the total-exchange in many parallel algorithms, particularly in the field of scientific computing [15, 7, 6, 3].

Our motivation is, however, different. We study the total-exchange communication primitive for its potential to implement general communication patterns in parallel computers. Specifically, we consider a class of balanced communication patterns, called h-relations, in which each processor is the origin and destination of at most h messages. The efficient realization of h-relations is very important, as it is a necessary condition for the optimal implementation of high-level programming models such as Valiant's BSP model and the PRAM [24, 26, 25]. Thus, the main idea is to write computer programs in a language based on these models and, provided the degree of virtual parallelism is sufficiently large compared to the number of processors available, have them executed with optimal efficiency. (Intuitively, the value of h corresponds to the ratio of the degree of virtual parallelism required in a program to the number of processors in the underlying machine.) For a detailed description of this approach, as well as discussions on efficient general-purpose parallel computing, see [26, 25, 7, 19, 20].

In this paper, we present theoretical and experimental results in support of total-exchange based communication protocols for routing h-relations. Our goal is to minimize the number of total-exchange rounds.

[†] Supported by NEC Research Institute.

We consider the case where h is large (about the size of the network) and compare the performance of our protocols with existing ones considering two architectures. The strengths of our approach are detailed in the following arguments:

1. The total-exchange communication pattern can be implemented efficiently in a variety of interconnection topologies. For example, there are simple protocols for the circuit-switched butterfly and the Optically Connected Parallel Computer (OCPC) model that run in $N/2$ and N steps, respectively. We emphasize that these protocols are contention-free and require very simple network control.

2. We show how a random h-relation can be routed using $\frac{h}{N}(1+o(1)) + O(\log\log N)$ total-exchange rounds, with high probability. This can be improved to $\frac{h}{N}(1+o(1)) + O(\log^* N)$ if duplication of messages is used. Arbitrary h-relations can be routed in about twice as much time, by first routing to random intermediate destinations as proposed by Valiant [23].

3. We compare the total-exchange based protocols against other protocols in two cases: The N-node OCPC, for which there is a protocol to realize a random h-relation within $h(5.5+o(1))$ communication steps provided that $h = \omega(\log\log N)$ [10], and the N-input circuit-switched butterfly for which there is a protocol to realize a random h-relation within $O(h \log N \log\log N)$ parallel communication steps [21]. These protocols were engineered towards achieving efficiency for small values of h, rather than 1-optimality for large values of h.
   In contrast, our protocols are 1-optimal for large values of h. Specifically for $h = \omega(N \log\log N)$ and $h = \omega(N \log^* N)$, we present two 1-optimal protocols for routing random h-relations in both architectures. This improves upon a result by Gerbessiotis and Valiant [7], who gave a 1-optimal total-exchange based algorithm for $h = \omega(N \log N)$[*].

4. Experiments based on a variation of our protocols show its efficacy for routing large h-relations.

The remainder of the paper is organized as follows. In the next section we discuss the importance of h-relations in general-purpose parallel computing, and we review simple protocols for realizing the total-exchange on the butterfly and the OCPC. Section 3 contains our new protocols. Finally, in the last section, we present experimental results using a variation of one of our protocols.

[*]We remark that these algorithms can also be used hierarchically to lower the value of h at the expense of the protocol's optimality. For example, there is a two-level total-exchange protocol that is 2-optimal for $h = \omega(\sqrt{N} \log\log N)$.

2 Communication primitives and general-purpose parallel computing

In this section we discuss the importance of h-relations in the theory of general-purpose parallel computing. Much of this discussion is drawn from [22]. Then we survey routing protocols in two contexts, viz., the circuit-switched butterfly and the OCPC (Optically Connected Parallel Computer). Finally, we review two simple protocols for realizing a total-exchange in these networks. None of the results in this section are new.

One of the major challenges in parallel computing is to determine the extent to which general-purpose parallel computing can be achieved. The goal is to deliver both scalable parallel performance and architecture-independent parallel software. One approach towards achieving this goal has been proposed by Valiant [24, 26]. (Other approaches have been proposed; see, for example, Kruskal, Rudolph, and Snir [17].)

Valiant introduced the Bulk-Synchronous Parallel (BSP) model which abstracts the characteristics of a parallel machine into three numerical parameters corresponding to the number of processors, bandwidth, and latency. The BSP model was suggested by Valiant as a possible "bridging model" to serve as a standard interface between the language and architecture levels in parallel computation. It can support two powerful programming styles: a shared-memory PRAM-like programming style and a direct programming style where the programmer maintains control of memory allocations. As already mentioned, any parallel architecture that can efficiently realize h-relations can also efficiently support programming languages written for the BSP model[†]. The value of h affects the latency parameter of the BSP model, and the efficiency of the implementation of an h-relation affects the bandwidth parameter of the BSP model. Specifically, higher values of h produce higher values for the latency parameter. On the other hand, assuming a higher value of h, one can derive more efficient solutions to the h-relation routing problem, and thus obtain better values for the bandwidth parameter of the BSP model.

While there are numerous existing protocols for routing h-relations [26, 18], they are primarily concerned with minimizing the value of h while guaranteeing that a random h-relation is delivered within a constant factor of the optimal time. For the BSP

[†]Arguments for the BSP model have been extensively presented and supported by theoretical analysis. For precise definition of the BSP model, examples of BSP-style algorithms, and arguments for the candidacy of BSP as a universal model, we refer to Valiant [24, 26, 25, 7] and McColl [19, 20].

model, this translates to minimizing the latency parameter while only sacrificing the bandwidth by a constant factor. A different approach is to require that the h-relation be routed in time that is optimal to within a factor of $1+o(1)$. These protocols are called *1-optimal* by Gerbessiotis and Valiant [7]. For the BSP model, such an approach corresponds to minimizing the latency while essentially not sacrificing any bandwidth at all.

In addition to defining 1-optimality, [7] presents a very simple 1-optimal protocol based on the total-exchange primitive. Here we improve on [7] and give protocols, also based on the total-exchange, which are 1-optimal for a slightly wider range of h. These protocols are very simple to implement, contention-free, require minimal hardware control, and are efficient.

To illustrate the strengths of total-exchange protocols against those designed for small values of h, we consider two abstract networks. The first is the N-input circuit-switched butterfly, where any set of disjoint paths can be set up in one step. For such a network, the results of Ranade, Schleimer and Wilkerson [21] imply that there is a protocol that routes a random h-relation in $O(h \log N \log\log N)$ steps. For large h, the total-exchange based protocols are better by a factor of $O(\log N \log\log N)$. This is achieved by the protocol of Gerbessiotis and Valiant [7] when $h = \omega(N \log N)$, or by our protocols when $h = \omega(N \log\log N)$ and $h = \omega(N \log^* N)$, respectively.

Our second example is the OCPC (Optically Connected Parallel Computer), a model designed to capture the features of optical communication systems. This model was previously used by Eshaghian [5] and Anderson and Miller [1]. (A similar model was also discussed by Hartmann and Redfield [13].) An N-OCPC consists of N processors, each of which has its own local memory. In any step, a processor can send at most one message to any other processor. If two processors transmit messages to the same processor, neither transmission is successful and the messages have to be retransmitted. A successful transmission is acknowledged by sending a confirmation message to the sending processor; hence, a non-successful transmission is detected by the absence of such an acknowledgement message. Thus, in constant time, all the processors requesting accesses are informed whether they have succeeded. A very simple randomized protocol for routing h-relations on an N-OCPC is described by Geréb-Graus and Tsantilas in [8]. Their protocol runs in time $\Theta(h+\log N \log\log N)$. A much more complicated protocol that runs in time $\Theta(h + \log\log N)$ is described in [10]. For either protocol, however, the constant factor is about 5.5. In contrast, for large h, our total-exchange based protocols achieve 1-optimality on the OCPC.

Finally, we describe how the total-exchange primitive itself can be realized in these two networks. We emphasize the simplicity of these protocols, the fact that they are contention-free, and the low hardware control requirements for their realization.

Fact 1 *There exist simple, deterministic and contention-free protocols that realize the total-exchange in the N-input multi-port butterfly and the N-OCPC within $N/2$ and N parallel communication steps, respectively.*

Proof. The protocol of [7] realizes the total-exchange in the OCPC as follows. It runs for N phases, and during phase j processor i transmits its message destined for processor $(i + j) \bmod N$.

For the butterfly, one can implement the previous protocol in N steps, in the single port model of the circuit switched butterfly. (That is, the set of circuits that can be routed in one step correspond to a set of *node* disjoint paths.) In a multi-port circuit switched model, where the set of circuits that can be routed in one step correspond to a set of *edge* disjoint paths, one can implement a total-exchange in $N/2$ rounds as follows. Consider the N permutations obtained as compositions of a sequence of $\log N$ shuffle or exchange permutations as defined by the shuffle-exchange network. The result follows by observing that these permutations constitute a total-exchange and that they can be divided in pairs each of which can be realized in a conflict-free manner in the butterfly. $\square$

3 Two Routing Protocols

In this section we present two protocols for routing random h-relations in networks equipped with the total-exchange communication primitive. These protocols can also be used to route *arbitrary* h-relations, by using Valiant's two-stage routing strategy [23], where messages are first sent to a randomly chosen intermediate destination.

Gerbessiotis and Valiant [7] observed that the maximum number of messages over all origin-destination pairs in a random h-relation is $h/N + O(\sqrt{(h/N)\log\log N}) + O(\log N)$, with high probability. This implies that $\frac{h}{N}(1 + o(1)) + O(\log N)$ total-exchange rounds suffice to route all messages to their destinations.

On the other hand, the standard deviation of the number of messages over any origin-destination pair is $\Theta(\sqrt{h/N})$, and it can be shown that at least $h/N + \Omega(\sqrt{h/N})$ total-exchange rounds are needed to route a random h-relation, with high probability. Thus, in the design of our protocols, we have to deal with the fact that the number of messages over all origin-destination pairs in a random h-relation is not uniform. We do this by routing some of those messages that belong to heavily loaded origin-destination pairs to random intermediate destinations.

In our analysis, we assume that every processor initially holds h messages, and that the destinations of the messages are chosen independently and uniformly at random from among the N processors. Note that the resulting communication pattern will actually not be an h-relation in the strict sense. However, our results can also be extended to the case of a "true" h-relation chosen uniformly at random from the set of all h-relations.

3.1 Analysis of a Simple Routing Protocol

In this subsection, we present and analyse a simple protocol that achieves 1-optimality for $h = \omega(N \log\log N)$. For the sake of simplicity, we assume $h = O(N \log N)$. However, the result can be fairly easily extended to larger values of h, either by running the protocol repeatedly on small subsets of the input, or by adapting it appropriately. The protocol runs in the following steps:

1. Perform $h/N + \epsilon\sqrt{(h/N)\log\log N} + \alpha \log\log N$ rounds of total-exchange to route messages to their final destinations, for some appropriately chosen constants α and ϵ.

   Condition A: With probability $1 - N^{-\Theta(1)}$, the remaining messages form an $(N/\log^{c-1} N)$-relation, where c is a constant depending on α.

2. a. Consider a partition of the processors into consecutive blocks of size $\log^c N$. Each processor assigns a random block to each of its messages.

   Condition B: The number of messages that choose the same destination block is at most $\log^c N$with probability $1 - N^{-\Theta(1)}$.

   b. If Condition B does not hold then break and deliver the messages with as many rounds of total-exchange as necessary. Otherwise, for each processor pick an intermediate destination for each message by injectively assigning destinations within the block to messages destined for the same block.

   Condition C: For each possible intermediate destination, each processor has at most one message to deliver.

   c. Using one total-exchange round, deliver the messages to their intermediate destinations.

3. Using one total-exchange round deliver, some of the messages to their destinations. The choice of which message to send is done arbitrarily.

4. Repeat Steps 2 and 3 until all messages are delivered to their destination.

Theorem 1 *A random h-relation can be routed in an N-node network using $\frac{h}{N}(1+o(1))+O(\log\log N)$ total-exchange rounds with probability $1 - N^{-\Theta(1)}$.*

Proof. We prove that Conditions A, B, and C are true with high probability and that $O(\log\log N)$ iterations of Steps 2 to 4 suffice to deliver all the messages to their destinations. This is established by a series of claims.

Claim 1 *Condition A is true.*

Proof. For the probabilistic analysis, we use the standard Bernstein-Chernoff estimate for bounding the tail of the binomial distribution with parameters m and p (e.g., see [2]). Suppose we toss m independent coins each having probability p of landing heads. Then, the probability that the total number of heads is greater than or equal to k, $k \geq mp$, is at most $\exp\{-k\ln(k/mp) + k - mp\}$.

In what follows, "with high probability" means with probability $1 - N^{-\Theta(1)}$. In a random h-relation, the above bound implies that, with high probability, no source processor has more than $h/N + \epsilon\sqrt{(h/N)\log\log N} + O(\log N)$ messages destined for any given processor.

Consider now the h messages destined for a given processor. We want to estimate how many of them do not get delivered after Step 1 of the algorithm. This can be done by analysing the underlying urn process where we toss h balls into N urns, each of capacity $h/N + \epsilon\sqrt{(h/N)\log\log N} + \alpha\log\log N$. Using the Bernstein-Chernoff bound, we can show that the probability that a particular urn is overflowed is at most $1/\log^\beta N$, and that the number of overflowed urns is with high probability at most $N/\log^\gamma N$, where β and γ are constants that depend on α. We now observe that the number of excess balls in an overflowed urn is $h/N + \epsilon\sqrt{(h/N)\log\log N} + O(\log N) - (h/N + \epsilon\sqrt{(h/N)\log\log N} + \alpha\log\log N) = O(\log N)$ with high probability. We conclude that, with high probability, any destination processor will have at most $N/\log^{c-1} N$ messages after Step 1, where c is a constant that depends on α.

A similar analysis shows that the total number of undelivered messages in any source processor is $N/\log^{c-1} N$ with high probability. □

Claim 2 *Condition B is true.*

Proof. We estimate the probability that in any of the N source processors one of the $N/\log^c N$ blocks is chosen by more than $\log^c N$ messages. By Condition A the number of messages residing at a processor after Step 1 of the algorithm is at most $N/\log^{c-1} N$.

Therefore this probability is bounded from above by

$$N \binom{N}{\log^c N} \sum_{k \ge \log^c N} \binom{\frac{N}{\log^{c-1} N}}{k} \left(\frac{\log^c N}{N}\right)^k$$

This can easily be shown to be $1/N^{\Theta(1)}$. Notice that Condition B implies Condition C. □

Claim 3 *With probability $1 - N^{-\Theta(1)}$, Steps 2 and 3 will be repeated $O(\log\log N)$ times.*

Proof. Assuming the message distribution after Step 1 of the algorithm, consider the $N/\log^{c-1} N$ messages for a particular destination processor π. We estimate the number of messages delivered to π in an iteration of Steps 2 to 4, and show that $O(\log\log N)$ iterations are enough, with high probability.

Recall that in Step 2a, each message is sent to an intermediate destination in a randomly chosen block. The probability that a message is successfully transmitted to its actual destination in Step 2c is bounded from below by the probability that no other message with the same destination is assigned to the same block.

To estimate the number of such messages, we set $B = N/\log^c n$ and proceed as follows. We assume that in the beginning of iteration i, there are at most B/ℓ_i messages destined for π, and we show that with probability $1 - 1/N^{\Theta(1)}$ at most $B/{\ell_i}^{3/2}$ messages are undelivered at the end of the iteration. Since ℓ_0 is at least $\log N$, it follows that after $O(\log\log N)$ iterations all the messages have been successfully tranmitted.

The analysis follows by considering the urn process where we toss B/ℓ_i balls into B urns. The goal is to show that not too many balls share urns. The number of balls that share an urn with some other ball corresponds to the number of messages that are undelivered after iteration i. We consider the process sequentially. If a ball shares an urn with other balls, we "charge" this event to this ball if it is the last ball to land in the urn; otherwise we charge it to the next ball that lands in the urn. Thus, each ball is charged at most twice, and it can only be charged if it lands in an urn that was previously chosen by a ball. Thus, the probability that k balls share urns is upper-bounded by the probability that $k/2$ balls choose a previously occupied urn. The probability that a ball chooses an occupied urn is at most $(B/\ell_i)/B = 1/\ell_i$. Thus, the probability that at least $k/2$ balls choose previously occupied urns is

$$\sum_{j \ge k/2} \binom{B/\ell_i}{j} \left(\frac{1}{\ell_i}\right)^j \left(1 - \frac{1}{\ell_i}\right)^{B/\ell_i - j}$$
$$\le \binom{B/\ell_i}{k/2} \left(\frac{1}{\ell_i}\right)^{k/2} \le \left(\frac{eB/\ell_i}{k/2}\right)^{k/2} \left(\frac{1}{\ell_i}\right)^{k/2}.$$

For the last derivation we used the inequality $\binom{x}{s} \le (\frac{ex}{s})^s$. By choosing $k/2 = B/2\ell_i^{3/2}$, we have that with probability $1 - N^{-\Theta(1)}$ at most $B/\ell_i^{3/2}$ balls share urns. This completes the proof of the claim. □

Together, Claims 1 to 3 establish the correctness of the algorithm, and show that its running time is $h/N(1 + o(1)) + O(\log\log N)$ total-exchange rounds, with high probability. □

3.2 A Faster Protocol

In this subsection we describe a protocol that routes a random h-relation in $h/N + O(\sqrt{h/N}) + O(\log^* N)$ total-exchange rounds, thus achieving 1-optimality for all $h = \omega(N \log^* N)$. The improvement is made possible by the use of a fast contention-resolution technique recently developed in the context of randomized computation on the CRCW PRAM [9, 11, 12]. The basic idea underlying this technique, sometimes referred to as the "dart-throwing technique" [12], is to achieve faster contention-resolution by creating multiple instances of requests that remain unresolved after each step of the protocol. In our routing protocol, we create multiple copies of those messages that remain undelivered after the number of messages per source and destination has been decreased below N. To deliver a message to its destination, it then suffices to successfully route one of its copies.

Following is the description of our protocol. If k rounds of total-exchange are performed in a given step of the protocol, and there are more than k messages that have to be routed from some processor i to some processor j, then a random subset of k of these messages is routed.

```
1. Perform $h/N$ rounds of total-exchange to route
   messages directly to their final destinations

2. $h := 2\sqrt{hN}$ (Claim: The remaining messages
   form an $h$-relation, w.h.p.)

3. Repeat until $h \le N$
   a. Every message chooses an intermediate desti-
      nation uniformly at random from among the
      $N$ processors
   b. Perform $h/N$ rounds of total-exchange to
      route messages to their intermediate destina-
      tions
   c. Perform $h/N$ rounds of total-exchange to
      route messages to their final destinations
   d. $h := 2\sqrt{hN}$ (Claim: The remaining messages
      form an $h$-relation, w.h.p.)

4. Repeat until $h < 1/N^{10}$
```

a. Make N/h copies of each undelivered message. Then every copy chooses an intermediate destination uniformly at random from among the N processors

b. Perform one round of total-exchange to route messages to their intermediate destinations

c. Perform one round of total-exchange to route messages to their final destinations, and eliminate all copies afterwards.

d. $h := h/2^{N/(8h)}$ (Claim: The remaining messages form an h-relation, w.h.p.)

Note that Step 3 of the protocol is iterated $\Theta(\log\log(h/N))$ times, and that the total time spent on Step 3 is dominated by the time for the first iteration, which involves $\Theta(\sqrt{h/N})$ total-exchange steps. Step 4 is iterated $\Theta(\log^* N)$ times, and hence the total running time of the protocol is $h/N + O(\sqrt{h/N}) + O(\log^* N)$.

A proof of correctness for the protocol will be given in the full version of the paper. We end this subsection with a few remarks about some possible modifications of the above protocol.

(1) Note that we do not have to send actual copies of the messages during Step 4 of the protocol. Instead, it would suffice to generate a number of empty "dummies" for each message. Once one of these "dummies" has reached the destination, we can then send the full message along the same path.

(2) If we would not make any copies in Step 4, and continue along the lines of Step 3, then the resulting, slightly simpler protocol would run in time $h/N + O(\sqrt{h/N}) + O(\log\log N)$, which is 1-optimal for $h = \omega(N \log\log N)$.

(3) Finally, note that we could route an arbitrary h-relation in about twice the time by simply omitting Steps 1 and 2 of the protocol.

4 Experimental Results

In this section we experimentally test total-exchange based protocols for routing random h-relations when h is large. We do not test them against other routing protocols because, as already mentioned, these were engineered towards minimizing h and have larger constant factors. The first protocol, which we call the direct protocol, is based on the observation that if m is the maximum number of packets over all origin-destination pairs, then m total-exchange rounds suffice to route all the packets. Gerbessiotis and Valiant [7] showed that in a random h-relation, we have $m = h/N(1 + o(1)) + O(\log N)$ with high probability.

The other protocol is a variation of those described in the previous section. It also consists of a number of total-exchange rounds. The key idea is to notice that in each round the load of the processor-destination pairs is not balanced: some pairs have an "excess" of packets, and some a "deficiency" with respect to the average value. The deficiencies imply that, in the direct protocol, some total-exchange round would have "idle" transmissions. We take advantage of these "idle" transmissions to deliver excess packets to "random" intermediate destinations. This in effect creates a more load-balanced situation.

In the table below we compare the performance of the two protocols. We measure the number of total-exchange rounds (referred to as rounds in the table), as well as the total number of parallel communication steps (referred to as time). For the latter we assume an N-node network in which total-exchange can be implemented within N steps.

		our protocol		direct protocol	
N	h	rounds	time/h	rounds	time/h
64	$4N$	8.0	2.00	12.7	3.18
	$8N$	12.3	1.54	19.9	2.49
	$10N$	14.4	1.44	23.8	2.38
	$16N$	21.0	1.31	31.6	1.98
	$32N$	38.1	1.19	53.1	1.66
	$50N$	57.0	1.14	78.2	1.56
	$64N$	71.0	1.10	93.8	1.47
128	$4N$	8.0	2.00	13.6	3.40
	$8N$	12.4	1.55	21.6	2.70
	$10N$	14.9	1.49	25.1	2.51
	$16N$	21.0	1.31	34.2	2.14
	$32N$	38.0	1.19	56.4	1.76
	$50N$	57.0	1.14	81.1	1.62
	$64N$	71.0	1.11	93.8	1.47
256	$4N$	8.0	2.00	15.0	3.75
	$8N$	12.9	1.61	22.4	2.80
	$10N$	15.0	1.50	26.6	2.66
	$16N$	21.1	1.32	36.0	2.25
	$32N$	38.1	1.19	59.1	1.85
	$50N$	57.0	1.14	83.2	1.67
	$64N$	71.3	1.11	100.3	1.57
512	$4N$	8.0	2.00	16.0	4.00
	$8N$	13.0	1.63	23.8	2.98
	$10N$	15.0	1.50	28.0	2.80
	$16N$	21.0	1.31	36.9	2.31
	$32N$	38.0	1.19	62.2	1.95
	$50N$	57.0	1.14	84.9	1.70
	$64N$	71.5	1.12	103.1	1.61
1024	$4N$	8.0	2.00	17.3	4.33
	$8N$	13.0	1.63	24.9	3.11
	$10N$	15.1	1.51	28.4	2.84
	$16N$	21.7	1.36	38.7	2.42
	$32N$	38.6	1.21	62.8	1.96
	$50N$	57.2	1.14	88.9	1.78
	$64N$	72.0	1.12	106.0	1.66

The second-to-last column is just the maximum load over all origin-destination pairs in a random h-relation. It is remarkable that the running time of our protocol, which is given in the column labelled "time/h", essentially depends only on the value of h/N. This phenomenon can be explained by the fact that for small N a whole total-exchange round is wasted to route only a few packets. In such cases it would be beneficial to use an algorithm not based on total-exchange to route those few packets.

References

[1] R. J. Anderson and G. L. Miller, *Optical communication for pointer based algorithms*, Tech. Report CRI-88-14, Computer Science Department, University of Southern California, 1988.

[2] D. Angluin and L. G. Valiant, *Fast probabilistic algorithms for hamiltonian circuits and matchings*, Journal of Computer and Systems Sciences, 18 (1979), pp. 155–193.

[3] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation : Numerical Methods*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[4] H. Bruck, C.-T. Ho, S. Kipnis, and D. Weathersby, *Efficient algorithms for all-to-all communications in multi-port message-passing systems*, in 6th Symposium on Parallel Algorithms and Architectures (SPAA '94), 1994, pp. 298–309.

[5] M. M. Eshaghian, *Parallel algorithms for image processing on OMC*, IEEE Transactions on Computers, 40 (1991), pp. 827–833.

[6] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker, *Solving Problems on Concurrent Processors, Vol. 1: General Techniques and Regular Problems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[7] A. Gerbessiotis and L. G. Valiant, *Direct bulk-synchronous parallel algorithms*, Journal of Parallel and Distributed Computing, 22 (1994), pp. 251–267.

[8] M. Geréb-Graus and T. Tsantilas, *Efficient optical communication in parallel computers*, in 4th Symposium on Parallel Algorithms and Architectures (SPAA '92), 1992.

[9] J. Gil, Y. Matias, and U. Vishkin, *Towards a theory of nearly constant time parallel algorithms*, in 32nd IEEE Symposium on Foundations of Computer Science (FOCS '91), 1991, pp. 698–710.

[10] L. Goldberg, M. Jerrum, T. Leighton, and S. Rao, *A doubly-logarithmic communication algorithm for the completely connected optical communication parallel computer*, in 5th Symposium on Parallel Algorithms and Architectures (SPAA '93), 1993.

[11] T. Hagerup, *Fast parallel space allocation, estimation and integer sorting*, Tech. Report MPI-I-91-106, Max-Planck-Institut für Informatik, Saarbrücken, 1991.

[12] ———, *The log-star revolution*, in STACS '92, Lecture Notes in Computer Science 577, Springer, 1992, pp. 259–278.

[13] A. Hartmann and S. Redfield, *Design sketches for optical crossbar switches intended for large scale parallel processing applications*, Optical Engineering, 29 (1989), pp. 315–327.

[14] S. Hinrichs, C. Kosak, D. O'Hallaron, T. M. Stricker, and R. Take, *An architecture for optimal all-to-all personalized communication*, in 6th Symposium on Parallel Algorithms and Architectures (SPAA '94), 1994, pp. 310–319.

[15] S. L. Johnsson, *Massively parallel computing: Data distribution and communication*, in Parallel Architectures and their Efficient Use, F. Meyer auf der Heide, B. Monien, and A. Rosenberg, eds., Lecture Notes in Computer Science 678, Springer-Verlag, 1993, pp. 68–92.

[16] S. L. Johnsson and C. T. Ho, *Optimal broadcasting and personalized communications in hypercubes*, IEEE Transactions on Computers, 38 (1989), pp. 1249–1268.

[17] C. P. Kruskal, L. Rudolph, and M. Snir, *A complexity theory of efficient parallel algorithms*, Theoretical Computer Science, 71 (1990), pp. 95–132.

[18] F. T. Leighton, *Introduction To Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, San Mateo, California, 1992.

[19] W. F. McColl, *General purpose parallel computing*, in Lectures on Parallel Computation. Proc. 1991 ALCOM Spring School on Parallel Computation, A. M. Gibbons and P. Spirakis, eds., Cambridge University Press, Cambridge, UK, 1992.

[20] W. F. McColl, *Scalable parallel computing: A grand unified theory and its practical development*, in Proc. 13th IFIP World Computer Congress. Volume I (Invited Paper), B. Pehrson and I. Simon, eds., Elsevier, 1994, pp. 539–546.

[21] A. G. Ranade, S. Schleimer, and D. Wilkerson, *Nearly tight bounds for wormhole routing*, in 35th IEEE Symposium on Foundations of Computer Science, 1994, pp. 347–355.

[22] S. B. Rao and T. Tsantilas, *Optical interprocessor communication protocols*, in First International Workshop on Massively Parallel Processing Using Optical Interconnections, 1994, pp. 266–274.

[23] L. G. Valiant, *A scheme for fast parallel communication*, SIAM Journal on Computing, 11 (1982), pp. 350–361.

[24] ———, *Optimally universal parallel computers*, Phil. Trans. R. Soc. Lond., A 326 (1988), pp. 373–376.

[25] ———, *A bridging model for parallel computation*, Communications of the ACM, 33 (1990), pp. 103–111.

[26] ———, *General purpose parallel architectures*, in Handbook of Theoretical Computer Science: Vol. A (J. van Leeuwen, ed.), North-Holland, 1990.