A Fast Communication Algorithm for the Completely Connected Optical Communication Parallel Computer

Leslie Ann Goldberg†

and

Mark Jerrum‡

August 1992

Abstract In this paper we consider the problem of interprocessor communication on a *Completely Connected Optical Communication Parallel Computer* (OCPC). The particular problem that we study is the problem of realizing an *h-relation.* In this problem each processor has at most h messages to send and each processor has at most h messages to receive. It is clear that any 1-relation can be realized in $\Theta(1)$ steps on an OCPC. However, the best known p processor OCPC algorithm for realizing an arbitrary h-relation for $h > 1$ runs in $\Theta(h + \log p)$ expected time. (This algorithm is due to Valiant and is based on earlier work of Anderson and Miller.) Geréb-Grauss and Tsantilas have asked whether there is a faster algorithm for $h = \mathrm{o}(\log p)$. In particular, they have asked whether a 2-relation can be realized in $\mathrm{o}(\log p)$ expected time on a p processor OCPC. In this paper we show that a 2-relation can indeed be realized in sub-logarithmic time. More generally, we give an algorithm that realizes an arbitrary h-relation on a p processor OCPC. We show that if $h \le \log^{1/2} p$ then the expected running time of the algorithm is $\Theta(h \log^{1/2} p)$ and the probability that it takes more than $\Theta(h \log^{1/2} p)$ time can be made as small as $p^{-\alpha}$ for any fixed positive integer α. Our algorithm is *pure* in the sense of Geréb-Graus and Tsantilas.

---

† Department 1423, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185, USA. (e-mail: lagoldb@cs.sandia.gov) This material is based upon work supported under a National Science Foundation Graduate Fellowship. The work was done at the University of Edinburgh.

‡ Department of Computer Science, University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ, UK. (e-mail: mrj@dcs.ed.ac.uk)

1. Introduction

The p processor *Completely Connected Optical Communication Parallel Computer* (p-OCPC) consists of p processors, each of which has its own local memory. The p processors can perform local computations and they can communicate with each other by message passing. A *computation* on this computer consists of a sequence of *communication steps.* During each communication step each processor can perform some local computation and then it can send one message to any other processor. If a given processor is sent one message during a communication step then it receives this message successfully but if it is sent more than one message then the transmissions are garbled and it does not receive any of the messages.

The OCPC was first introduced as a model of computation by Anderson and Miller [AM 88]. (Anderson and Miller called this model the *Local Memory PRAM.*) Since then it has been studied by Valiant [Va 90] (who called the model the S^*PRAM), by Geréb-Graus and Tsantilas [GT 92], and by Gerbessiotis and Valiant [GV 92] (who also called the model the S^*PRAM). A related model has been studied by Rao [Ra 92]. The feasibility of the OCPC from a engineering point of view is discussed in [AM 88, GT 92, and Ra 92]. See also the references in [McC 92].

In this paper we study the problem of interprocessor communication on an OCPC. In particular, we study the problem of realizing *h-relations.* An h-relation (see [Va 90]) is a communication problem in which each processor has up to h messages that it wishes to send to other processors. The destinations of these messages can be arbitrary except that each processor is the destination of at most h messages. The goal is to design a fast p-OCPC algorithm that can realize an arbitrary h-relation. In the execution of this algorithm each processor knows only about the messages that it wants to send and about the messages that it has already received. (Anderson and Miller [AM 88] have observed that an h relation can easily be realized in h communication steps in the simpler case in which all of the processors are given *total* information about the h-relation to be realized†.)

It is easy to see that any 1-relation can be realized in 1 communication step on an OCPC. Anderson and Miller [AM 88] were the first to consider the problem of realizing h-relations for $h > 1$. They considered a special class of h-relations in which each of the

† To see this, consider a bipartite graph with vertex set $\{S_1, \ldots, S_p\} \cup \{D_1, \ldots, D_p\}$. Include an edge from S_i to D_j if and only if processor P_i has a message to send to processor P_j. Since this bipartite graph has maximum degree h it can be edge-colored with h colors $C_1, \ldots, C_h$. Using this edge-coloring, all of the messages can be sent without contention in h communication steps. In particular, processor P_i sends a message to P_j during communication step t if and only if there is an edge (S_i, D_j) with color C_t.

messages with a given destination has a unique label l in the range $1 \leq l \leq h$. They discovered a p-OCPC algorithm that can realize an arbitrary h-relation from this class. The expected number of communication steps that the algorithm requires is $\Theta(h + \log p)$. Anderson and Miller's algorithm consists of two phases. The first phase is probabilistic. It requires $\Theta(h)$ communication steps and it is very likely to deliver all but $\Theta(p)$ of the messages. The second phase is deterministic. It takes $\Theta(\log p)$ communication steps and it delivers the remaining $\Theta(p)$ messages. Valiant [Va 90] considered the general problem of realizing h-relations for $h > 1$. He discovered a p-OCPC algorithm that can realize an arbitrary h-relation. The expected number of communication steps that the algorithm requires is $\Theta(h + \log p)$. Valiant's algorithm consists of the first phase of Anderson and Miller's algorithm followed by a second deterministic phase which redistributes the remaining $\Theta(p)$ messages using parallel prefix, sorts them, and then sends them to the correct destinations. The second phase of Valiant's algorithm takes $\Theta(\log p)$ communication steps.

Valiant's algorithm is currently the fastest known OCPC algorithm that can realize an arbitrary h-relation for $h > 1$. It is clear that Valiant's algorithm is optimal for $h = \Omega(\log p)$. However, Geréb-Grauss and Tsantilas [GT 92] have asked whether there is a faster algorithm for $h = o(\log p)$. In particular, they have asked whether a 2-relation can be realized in $o(\log p)$ communication steps on a p-OCPC. In this paper we show that a 2-relation can indeed be realized in sub-logarithmic time. In particular, we describe a p-OCPC algorithm that can realize an arbitrary h-relation. We show that if $h \leq \log^{1/2} p$ then the expected number of communication steps that the algorithm requires is $\Theta(h \log^{1/2} p)$. Furthermore, the probability that the algorithm takes more than $\Theta(h \log^{1/2} p)$ communication steps can be made as small as $p^{-\alpha}$ for any fixed positive integer α.

Geréb-Grauss and Tsantilas [GT 92] have studied a special class of OCPC algorithms that they call *pure* algorithms. An OCPC algorithm for realizing h-relations is deemed to be *unpure* if it contains communication steps in which processors send each other information other than the original messages. Otherwise, it is deemed to be *pure*. The algorithms of Anderson and Miller and Valiant are not pure. However, Geréb-Grauss and Tsantilas have discovered a pure p-OCPC algorithm that can realize an arbitrary h-relation. The expected number of communication steps that their algorithm requires is $\Theta(h + \log p \log\log p)$. The algorithm that we describe in this paper is pure. However, we make the assumption that the destination of each message is included in its body.

In order to describe the intuition which motivates our algorithm we consider the case $h = 2$. Suppose that two processors P_i and P_j are both trying to send messages to a third processor P_ℓ and that they adopt the following "brute-force" strategy. During each communication step processors P_i and P_j both flip fair coins. If P_i's coin comes up *"heads"* then P_i sends its message to P_ℓ. Similarly, If P_j's coin comes up *"heads"* then P_j

sends its message to P_ℓ. On any given communication step P_ℓ has probability $1/2$ of successfully receiving a message. Therefore the probability that P_ℓ hasn't received any messages after t communication steps is 2^{-t}. Now suppose that we use a similar strategy to realize a 2-relation in which each processor is the destination of 2 messages. After t communication steps we will expect to have $p\,2^{-t}$ processors that haven't received any messages at all. Therefore it will take $\Omega(\log p)$ communication steps to realize the 2-relation.

In order to reduce the number of communication steps we use a probabilistic strategy that is less likely to fail. In particular, we divide the p processors into a number of disjoint "target groups". Instead of sending its message to P_ℓ, P_i sends the message to a randomly chosen processor within the target group of P_ℓ. Similarly, P_j sends its message to a randomly chosen processor within the target group of P_ℓ. During a given communication step P_i and P_j are likely to choose different destinations so their messages are unlikely to interfere with each other. The probability that neither message arrives at the target group of P_ℓ after t communication steps is substantially smaller than 2^{-t}. Once both messages have arrived at the target group we use a deterministic strategy to send them to P_ℓ.

Let k denote $\lceil e^{\log^{1/2} p} \rceil$. (All logarithms in this paper are to the base e unless a different base is specified.) In our algorithm we will divide the p processors $P_0, \ldots, P_{p-1}$ into approximately p/k target groups, each of size about k. The algorithm consists of four phases. Phase 1 uses the "brute-force" strategy outlined above to deliver most of the messages to their final destinations. Phase 2 uses the second probabilistic strategy outlined above to deliver the remaining messages to the correct target groups. The number of communication steps used during phases 1 and 2 is $\Theta(h \log^{1/2} p)$. We will show that with probability at least $1 - p^{-\alpha}$ (for any fixed positive integer α) every message is either delivered to its final destination during phase 1 or delivered to its target group during phase 2. Phase 3 uses a deterministic strategy to deliver the messages that were successfully sent to their target groups during phase 2. The number of communication steps used during phase 3 is $\Theta(h \log^{1/2} p)$. Phase 4 uses the brute-force strategy to deliver any messages that were not delivered to their final destinations during phase 1 and did not reach their target groups during phase 2. The probability that there are any messages satisfying this description is at most $p^{-\alpha}$ (for any fixed positive integer α) and the expected running time of phase 4 is $o(1)$.

In order to describe the algorithm in more detail we will need the following definitions. A $(1, h)$-relation is a communication problem in which each processor has up to 1 message to send and each processor is the destination of at most h messages. During the execution of the algorithm we will say that a message is *completely undelivered* if and only if it has not been successfully transmitted from its source to any other processor.

The structure of this paper is as follows. In section 2 we will describe phases 1 and 2 of the algorithm and we will prove that if we are given an arbitrary h-relation on a p-OCPC and we run phases 1 and 2 of the algorithm then the communication problem that remains when phases 1 and 2 terminate consists of

1. A $(1, h)$-relation within each target group, and
2. 0 or more completely undelivered messages.

We will prove that if h is less than or equal to $\log^{1/2} p$ then the probability that there are any completely undelivered messages is less than or equal to $p^{-\alpha}$ (for any fixed positive integer α). In section 3 we will describe a deterministic algorithm that can realize an arbitrary $(1, h)$-relation on a k-OCPC in $\Theta(h \log k)$ steps. Phase 3 consists of running this algorithm simultaneously within each of the size k target groups. In section 4 we will observe that the "brute-force" strategy can be used to realize an arbitrary h-relation on a p-OCPC in $O(p^2)$ expected communication steps. Phase 4 of our algorithm uses this strategy to deliver any messages that are completely undelivered when phase 2 terminates. If h is less than or equal to $\log^{1/2} p$ then the probability that any such messages exist is at most $p^{-\alpha}$ (for any fixed positive integer α) so if we define α to be greater than or equal to 2 then the expected running time of phase 4 is $o(1)$. Finally, in section 5 we will discuss some open problems.

2. Phases 1 and 2

For simplicity we will assume that k divides p and we will define the ℓth target group (for ℓ in the range $0 \leq \ell < n/k$) to be the set $\{P_{k\ell}, \ldots, P_{k\ell+k-1}\}$†. An M *messages per target group h-relation* is defined to be an h-relation in which the number of messages that are destined for the members of any given target group is at most M.

We will use the notation $m_{i,j}$ to refer to the jth message that processor P_i has to send. For any message m we will use the notation $d(m)$ to denote the index of the processor to which message m is destined. Processor $P_{d(m)}$ is a member of the $\lfloor d(m)/k \rfloor$th target group. We will refer to this group as the "target group of m".

Let t_1 denote $\lceil 2eh \log^{1/2} p \rceil$. Let α be any constant that is greater than zero and let t_2 denote $\lceil 3(\alpha+1) \log^{1/2} p \rceil$. In our algorithm each processor has local variables $s_1, \ldots, s_h$, *full*, and m. For j in the range $\{1, \ldots, h\}$ s_j is initialized to "`true`" if the processor has at least j messages to send and it is initialized to "`false`" otherwise. *full* is initialized to "`false`". The symbol i is used to refer to the index of the processor running the algorithm. The first two phases of the algorithm are as follows:

---

† The case in which k does not divide p presents no real difficulty. In this case the target groups should be defined in such a way that all but one of the groups has size k and the size of the remaining group is between k and $2k$.

Phase 1

```
For All Processors in Parallel Do
    For t ⟵ 1 To t_1
        Choose j uniformly at random from {1, ..., h}
        /* Send Step */
        If s_j = true
                Send m_{i,j} to P_{d(m_{i,j})}
        /* Receive Step */
        If exactly one message is received
            Send an acknowledgement back to the sender
        /* Receive Acknowledgement Step */
        If an acknowledgement is received
            s_j ⟵ false
```

Phase 2

```
For All Processors in Parallel Do
    For t ⟵ 1 To t_2
        For j ⟵ 1 To h
            /* Send Step */
            If s_j = true
                Choose a processor P uniformly at random
                    from the target group of m_{i,j}
                Send m_{i,j} to P
            /* Receive Step */
            If exactly one message is received
                If full = false
                    m ⟵ the message received
                    full ⟵ true
                    Send an acknowledgement back to the sender
            /* Receive Acknowledgement Step */
            If an acknowledgement is received
                s_j ⟵ false
```

We will prove the following lemmas.

Lemma 1. Suppose that we are given an arbitrary h-relation on a p-OCPC and that we run phase 1. Phase 1 delivers 0 or more of the messages in the h-relation to their final destinations and then it terminates after $\Theta(h \log^{1/2} p)$ communication steps. If h is less than or equal to $\log^{1/2} p$ then with probability at least $1 - 1/2p^{\alpha}$ the communication problem that remains when phase 1 terminates consists of a $k^{1/2} \log^2 p$ messages per target group h-relation.

Lemma 2. Suppose that we are given an arbitrary h-relation on a p-OCPC and that we run phase 2. After $\Theta(h \log^{1/2} p)$ communication steps phase 2 terminates and the remaining communication problem consists of

1. A $(1, h)$-relation within each target group (the messages in the $(1, h)$-relation are stored in the local variable "m" of the processors), and
2. 0 or more completely undelivered messages.

If the h-relation is a $k^{1/2} \log^2 p$ messages per target group h-relation then the probability that there are any completely undelivered messages is less than or equal to $1/2p^{\alpha}$.

We can combine these lemmas to obtain the following theorem.

Theorem 1. Suppose that we are given an arbitrary h-relation on a p-OCPC and that we run phases 1 and 2. After $\Theta(h \log^{1/2} p)$ communication steps phases 1 and 2 terminate and the remaining communication problem consists of

1. A $(1, h)$-relation within each target group, and
2. 0 or more completely undelivered messages.

If h is less than or equal to $\log^{1/2} p$ then the probability that there are any completely undelivered messages is less than or equal to $p^{-\alpha}$.

In the remainder of this section we will prove lemmas 1 and 2. We will use the following bounded difference inequality.

Theorem 2 (Independent Bounded Differences Inequality) [McD 89]. Let $x_1, \ldots, x_n$ be independent random variables, with x_k taking values in a set A_k for each k. Suppose that the (measurable) function $f : \prod A_k \to \mathbb{R}$ satisfies $|f(\overline{x}) - f(\overline{x}')| \leq c_k$ whenever the vectors $\overline{x}$ and $\overline{x}'$ differ only in the kth coordinate. Let Y be the random variable $f(x_1, \ldots, x_n)$. Then for any $t > 0$, $\Pr(|Y - E(Y)| \geq t) \leq 2\exp(-2t^2 / \sum c_k^2)$.

Proof of Lemma 1: It is easy to see that if we are given an arbitrary h-relation on a p-OCPC and we run phase 1 then phase 1 delivers 0 or more of the messages in the h-relation to their final destinations and it terminates after $\Theta(h \log^{1/2} p)$ communication steps. We will show that if h is less than or equal to $\log^{1/2} p$ then with probability at least $1 - 1/2p^{\alpha}$ the communication problem that remains when phase 1 terminates consists of a $k^{1/2} \log^2 p$ messages per target group h-relation.

Let S_r denote the set of processors that have messages to send to processor P_r. Suppose that P_i is a member of S_r. We will say that an iteration of phase 1 is *good* for P_i to send to P_r if and only if during this iteration processor P_i selects a value of j such that $d(m_{i,j}) = r$ and no other processor $P_l \in S_r$ selects a value of j such that $d(m_{l,j}) = r$. The probability that an iteration is good for P_i to send to P_r is $\frac{1}{h}(1-\frac{1}{h})^{|S_r|-1}$. So the probability that phase 1 does not contain an iteration that is good for P_i to send to P_r is at most

$$\left(1 - \tfrac{1}{h}(1-\tfrac{1}{h})^{|S_r|-1}\right)^{t_1} \le \left(1 - \tfrac{1}{eh}\right)^{t_1} \le e^{-t_1/eh}.$$

We will say that processor P_r is *slow* if and only if there is a processor $P_i \in S_r$ such that phase 1 does not contain an iteration that is good for P_i to send to P_r. The probability that P_r is slow is at most $|S_r|e^{-t_1/eh} \le he^{-t_1/eh}$. If P_r is not slow then it receives all of the messages that are destined for it during phase 1.

Let us now fix our attention on a particular target group T. Let S_T denote $\cup_{P_r \in T} S_r$. For convenience we will consider the processors in S_T to be ordered by their indices and we will use the notation $S_T[i]$ to denote the member of S_T with the ith largest index. For $1 \le i \le |S_T|$ let x_i be a random variable that denotes the sequence of values that the local variable j takes when $S_T[i]$ runs phase 1. (The value of x_i is chosen uniformly at random from the set $\{1,\ldots,h\}^{t_1}$.) Let $f(x_1,\ldots,x_{|S_T|})$ denote the number of slow processors in T. Suppose that P_r is a member of T and that l is an integer in the range $1 \le l \le |S_T|$ such that $S_T[l] \notin S_r$. It is clear from the definition of “slow” that changing the value of x_l cannot change whether or not processor P_r is slow. Since there are at most h processors $P_{r'}$ such that $S_T[l] \in S_{r'}$ the choice of x_l effects the slow-ness of at most h processors. We conclude that if the vectors $\overline{x} = (x_1,\ldots,x_{|S_T|})$ and $\overline{x}' = (x'_1,\ldots,x'_{|S_T|})$ differ only in the lth component then $|f(\overline{x}) - f(\overline{x}')| \le h$.

Let Y be the random variable $f(x_1,\ldots,x_{|S_T|})$. If Y is less than or equal to $k^{1/2}\log^{3/2} p$ then target group T has at most $k^{1/2}\log^{3/2} p$ slow processors so there are at most $hk^{1/2}\log^{3/2} p \le k^{1/2}\log^2 p$ messages destined for the members of T which are undelivered at the end of phase 1.

From our previous calculations we can conclude that $E(Y) \le khe^{-t_1/eh} \le 1$. The probability that $Y \ge k^{1/2}\log^{3/2} p$ is less than or equal to the probability that $Y \ge 1 + ((\alpha+1)kh^3 \log p/2)^{1/2}$ and by the independent bounded differences inequality this is at most $2\exp(-2(\alpha+1)kh^3 \log p/2kh^3) = 2p^{-(\alpha+1)}$.

We have now shown that for any given target group T the probability that there are more than $k^{1/2}\log^2 p$ messages which are destined for the members of T and are undelivered at the end of phase 1 is at most $2/p^{(\alpha+1)}$. We can conclude that the probability that there exists such a target group is at most $2p/kp^{(\alpha+1)}$ which is less than or equal to $1/2p^{\alpha}$ for big enough values of p. □

Proof of Lemma 2: Suppose that we are given an arbitrary h-relation on a p-OCPC and that we run phase 2. It is easy to see that phase 2 terminates after $\Theta(h \log^{1/2} p)$ communication steps. After phase 2 terminates the messages that are stored in the local variable "m" of the processors in each target group form a $(1, h)$-relation within the target group.

Let M denote $k^{1/2} \log^2 p$ and consider the case in which the h-relation is an M messages per target group h-relation. We wish to show that the probability that there are any completely undelivered messages when phase 2 terminates is less than or equal to $1/2p^\alpha$.

Let t be an integer in the range $1 \le t \le t_2$ and suppose that $m_{i,j}$ is an message that is completely undelivered at the beginning of iteration t of phase 2. Since there are only M messages destined for members of the target group of $m_{i,j}$ the probability that $m_{i,j}$ is not delivered to its target group during iteration t of phase 2 is at most $(M-1)/k$. Therefore the probability that $m_{i,j}$ is still completely undelivered after t_2 iterations of phase 2 is at most $((M-1)/k)^{t_2} \le p^{-(\alpha+1)}$. Finally, the probability that there exists a message $m_{i,j}$ that is completely undelivered after phase 2 terminates is at most $(p/k) \times M \times p^{-(\alpha+1)} \le 1/2p^\alpha$. □

3. Phase 3

In this section we will describe a deterministic algorithm that can realize an arbitrary $(1, h)$-relation on a k-OCPC in $\Theta(h \log k)$ communication steps. Phase 3 of our main algorithm consists of running this algorithm simultaneously within each of the size k target groups. Therefore, phase 3 realizes a $(1, h)$-relation within each target group and it requires $\Theta(h \log^{1/2} p)$ communication steps.

We start by observing that on a k-OCPC every $(1, h)$-relation is also a $(1, k)$-relation since there are at most k messages to be sent. Therefore, we can assume that $h \le k$. In order to describe our algorithm we require some definitions. Suppose that g is a positive integer, that $\Psi_1, \ldots, \Psi_\kappa$ are subsets of $\{1, \ldots, g\}$, and that λ is an integer in the range $1 \le \lambda \le \kappa$. We will say that the set $\{\Psi_1, \ldots, \Psi_\kappa\}$ forms a λ-*halver* if and only if it is the case that for any size λ set $I \subseteq \{1, \ldots, \kappa\}$ there are at most $\lfloor \lambda/2 \rfloor$ integers $\ell \in I$ such that

$$\Psi_\ell - \bigcup_{\ell' \in I - \{\ell\}} \Psi_{\ell'} = \emptyset.$$

To motivate this definition suppose that we are given an arbitrary $(1, h)$-relation to realize on a k-OCPC. Suppose further that g is a positive integer and that $\{\Psi_1, \ldots, \Psi_k\}$ is a set of subsets of $\{1, \ldots, g\}$ that forms an h-halver. Consider the following simple k-OCPC algorithm which runs for g communication steps. On step t of the algorithm (for $1 \le t \le g$) processor P_i checks whether $t \in \Psi_i$. If so, P_i sends its message to its destination. Otherwise, it does not send.

Consider any processor P_r and let I_r be the set containing the indices of all processors that have messages destined for P_r. Suppose that ℓ is a member of I_r and that $\Psi_\ell - \bigcup_{\ell' \in I-\{\ell\}} \Psi_{\ell'} \neq \emptyset$. Then there is a step t during which P_ℓ sends its message to P_r and no other processor sends a message to P_r. Therefore the algorithm successfully delivers the message from P_ℓ to P_r.

Since the set $\{\Psi_1, \ldots, \Psi_k\}$ forms an h-halver there are at most $\lfloor h/2 \rfloor$ integers $\ell \in I_r$ such that $\Psi_\ell - \bigcup_{\ell' \in I_r-\{\ell\}} \Psi_{\ell'} = \emptyset$. Therefore there are at most $\lfloor h/2 \rfloor$ messages that are destined for P_r and are not successfully delivered by the algorithm. We conclude that the communication problem that remains after the algorithm terminates is a $(1, \lfloor h/2 \rfloor)$-relation.

For every pair of positive integers (λ, κ) let $g(\lambda, \kappa)$ denote $\lceil 2e(1 + \lambda \log(2\kappa)) \rceil$. In this section we will prove the following lemma.

Lemma 3. Suppose that λ and κ are positive integers with $\lambda \leq \kappa$. Then there is a set $\{\Psi_1, \ldots, \Psi_\kappa\}$ of subsets of $\{1, \ldots, g(\lambda, \kappa)\}$ that forms a λ-halver.

Assume that lemma 3 is true and consider the problem of realizing a $(1, h)$-relation on a k-OCPC. For every integer j in the range $0 \leq j \leq \lfloor \log_2 h \rfloor + 1$ let λ_j denote $\lfloor h/2^j \rfloor$. Note that $\lambda_0 = h$. For every integer j in the range $0 \leq j \leq \lfloor \log_2 h \rfloor$ let $\{\Psi_{1,j}, \ldots, \Psi_{k,j}\}$ be a set of subsets of $\{1, \ldots, g(\lambda_j, k)\}$ that forms a λ_j-halver.

Consider the following algorithm for realizing a $(1, h)$-relation. In this algorithm each processor has local variables *full* and m. Initially the value of *full* is "`true`" if the processor has a message to send (in which case the message is stored in m). Otherwise, the value of *full* is "`false`". As before, the symbol i is used to refer to the index of the processor running the algorithm.

```
Algorithm Realize_(1,h)-Relation
For All Processors in Parallel Do
    For j ⟵ 0 To ⌊log₂ h⌋
        /* The current communication problem is a (1, λ_j)-relation */
        For t ⟵ 1 To g(λ_j, k)
            /* Send Step */
            If t ∈ Ψ_{i,j}
                If full = true
                    Send m to P_{d(m)}
            /* Receive Step */
            If exactly one message is received
                Send an acknowledgement back to the sender
            /* Receive Acknowledgement Step */
            If an acknowledgement is received
                full ⟵ false
        /* The communication problem is now a (1, λ_{j+1})-relation */
```

The analysis that we gave earlier in this section shows that if the current communication problem is a $(1, \lambda_j)$-relation at the start of the jth loop then it will be a $(1, \lfloor \lambda_j/2 \rfloor)$-relation after the termination of the jth loop. Note that $\lfloor \lambda_j/2 \rfloor \le \lambda_{j+1}$ so every $(1, \lfloor \lambda_j/2 \rfloor)$-relation is a $(1, \lambda_{j+1})$-relation. When the loop terminates the communication problem that remains is a $(1, \lambda_{\lfloor \log_2 h \rfloor + 1})$-relation. But $\lambda_{\lfloor \log_2 h \rfloor + 1} = 0$ so there are no messages left to send when the loop terminates.

The number of communication steps used by the algorithm is $\sum_{j=0}^{\lfloor \log_2 h \rfloor} 2\, g(\lambda_j, k)$ which is easily shown to be $\Theta(h \log k)$. Therefore we have proved the following theorem.

Theorem 3. Algorithm *Realize_(1, h)-Relation* can realize an arbitrary $(1, h)$-relation on a k-OCPC in $\Theta(h \log k)$ communication steps.

In the remainder of this section we will prove lemma 3. Then we will discuss the problem of implementing algorithm *Realize_(1, h)-Relation* on a k-OCPC.

Proof of Lemma 3: We will prove lemma 3 using the probabilistic method. Suppose that λ and κ are positive integers with $\lambda \le \kappa$ and let $\Psi_1, \ldots, \Psi_\kappa$ be random variables denoting κ subsets of $\{1, \ldots, g(\lambda, \kappa)\}$ constructed by the following process. For each $t \in \{1, \ldots, g(\lambda, \kappa)\}$ and for each integer ℓ in the range $1 \le \ell \le \kappa$ put t in Ψ_ℓ with probability $1/\lambda$.

Suppose that $\{\Psi_1,\ldots,\Psi_\kappa\}$ does *not* form a λ-halver. Then there is a size λ set $I \subseteq \{1,\ldots,\kappa\}$ and a size $\lfloor\lambda/2\rfloor+1$ set $J \subseteq I$ such that for every $\ell \in J$ we have $\Psi_\ell - \bigcup_{\ell' \in I-\{\ell\}} \Psi_{\ell'} = \emptyset$. Therefore the probability that $\{\Psi_1,\ldots,\Psi_\kappa\}$ does *not* form a λ-halver is at most

$$\sum_{\substack{I\subseteq\{1,\ldots,k\}\\|I|=\lambda}} \; \sum_{\substack{J\subseteq I\\|J|=\lfloor\lambda/2\rfloor+1}} \; \prod_{t\in\{1,\ldots,g(\lambda,\kappa)\}} \left(1-\sum_{\ell\in J}\Pr(t\in\Psi_\ell - \bigcup_{\ell'\in I-\{\ell\}}\Psi_{\ell'})\right).$$

It is straightforward to verify that this is at most

$$(2k)^\lambda\left(1-(\lfloor\lambda/2\rfloor+1)\tfrac{1}{\lambda}(1-\tfrac{1}{\lambda})^{\lambda-1}\right)^{g(\lambda,\kappa)}$$

which is less than or equal to $1/e$. We conclude that there is a set $\{\Psi_1,\ldots,\Psi_\kappa\}$ of subsets of $\{1,\ldots,g(\lambda,\kappa)\}$ that forms a λ-halver. □

In the remainder of this section we will discuss the problem of implementing algorithm *Realize_*$(1,h)$*-Relation*. So far in this paper we have assumed that processor P_i knows the values of p and h. Storing these values requires $\Omega(\log p)$ bits. In algorithm *Realize_*$(1,h)$*-Relation* processor P_i must also know the value of the sets $\Psi_{i,0},\ldots,\Psi_{i,\lfloor\log_2 h\rfloor}$. If h is less than or equal to $\log^{1/2} p$ then these sets can be represented in $O(\log p)$ bits so it would be reasonable to assume that these sets are also "hard-wired" in to the algorithm.

In fact, it suffices to assume that P_i knows the value of i. For each j in the range $0 \le j \le \lfloor\log_2 h\rfloor$ each processor could use the method of conditional probabilities [Spe 87, Rag 88] to construct a set $\{\Psi_{1,j},\ldots,\Psi_{k,j}\}$ of subsets of $\{1,\ldots,g(\lambda_j,k)\}$ that forms a λ_j-halver. Processor P_i could then use the ith member of this set.

For $\lambda_j = 1$ and $\lambda_j = 2$ it is not necessary to use the method of conditional probabilities. In particular, *any* set of non-empty subsets of $\{1,\ldots,g(1,\kappa)\}$ is a 1-halver. Furthermore, any set of subsets of $\{1,\ldots,g(2,\kappa)\}$ that is a *Sperner System* (see [Bol 86]) is a 2-halver. So, for example, if we define Ψ_i to be the ith size-$\lfloor g(2,\kappa)/2\rfloor$ subset of $\{1,\ldots,g(2,\kappa)\}$ (in some ordering such as the lexicographical ordering) then the set $\{\Psi_1,\ldots,\Psi_\kappa\}$ forms a 2-halver.

We will conclude this section by describing a sequential algorithm that takes as input two positive integers λ and κ with $\lambda \le \kappa$ and outputs a set $\{\Psi_1,\ldots,\Psi_\kappa\}$ of subsets of $\{1,\ldots,g(\lambda,\kappa)\}$ that forms a λ-halver. The algorithm is based on the method of conditional probabilities.

In order to describe the algorithm we need some notation. Suppose that λ and κ are positive integers with $\lambda \le \kappa$ and that $U = \{U_1,\ldots,U_\kappa\}$ and $W = \{W_1,\ldots,W_\kappa\}$ are two sets of subsets of $\{1,\ldots,g(\lambda,\kappa)\}$ such that $U_\ell \cap W_\ell = \emptyset$ for $1 \le \ell \le \kappa$.

Let $\Psi_1(U,W),\ldots,\Psi_\kappa(U,W)$ be random variables denoting κ subsets of $\{1,\ldots,g(\lambda,\kappa)\}$ constructed by the following process. For each $t \in \{1,\ldots,g(\lambda,\kappa)\}$ and for each integer ℓ in the range $1 \leq \ell \leq k$

1. If $t \in U_\ell$ then put $t \in \Psi_\ell(U,W)$.
2. If $t \notin U_\ell \cup W_\ell$ then put $t \in \Psi_\ell(U,W)$ with probability $1/\lambda$.

Let $\rho(U,W)$ denote

$$\sum_{\substack{I\subseteq\{1,\ldots,k\}\\|I|=\lambda}} \sum_{\substack{J\subseteq I\\|J|=\lfloor\lambda/2\rfloor+1}} \prod_{t\in\{1,\ldots,g(\lambda,\kappa)\}} \left(1-\sum_{\ell\in J}\Pr(t \in \Psi_\ell(U,W) - \bigcup_{\ell'\in I-\{\ell\}} \Psi_{\ell'}(U,W))\right).$$

The argument used in the proof of lemma 3 shows that $\rho(U,W)$ is an upper bound on the probability that $\{\Psi_1(U,W),\ldots,\Psi_\kappa(U,W)\}$ does *not* form a λ-halver.

Suppose that t is a member of $\{1,\ldots,g(\lambda,\kappa)\}$ and that ℓ is an integer in the range $1 \leq \ell \leq \kappa$ such that $t \notin U_\ell \cup W_\ell$. Let $U[\ell,t]$ denote the set obtained from U by adding t to U_ℓ and let $W[\ell,t]$ denote the set obtained from W by adding t to W_ℓ. It is straightforward to verify that $\rho(U,W) = \frac{1}{h}\times\rho(U[\ell,t],W)+(1-\frac{1}{h})\times\rho(U,W[\ell,t])$. Therefore at least one of $\rho(U[\ell,t],W)$ and $\rho(U,W[\ell,t])$ is less than or equal to $\rho(U,W)$.

Consider the following algorithm.

```
Algorithm Construct(λ, κ, U, W)
/* λ and κ are positive integers with λ ≤ κ.                       */
/* U = {U_1,...,U_κ} and W = {W_1,...,W_κ} are two sets of         */
/* subsets of {1,...,g(λ,κ)} such that U_ℓ ∩ W_ℓ = ∅               */
/* for 1 ≤ ℓ ≤ κ.                                                  */
While there is an integer t ∈ {1,...,g(λ,κ)} such that for some
integer ℓ in the range 1 ≤ ℓ ≤ κ we have t ∉ U_ℓ ∪ W_ℓ
    Let t and ℓ be integers meeting the above description
    If ρ(U[ℓ,t],W) ≤ ρ(U,W)
        U_ℓ ⟵ U_ℓ ∪ {t}
    Else
        W_ℓ ⟵ W_ℓ ∪ {t}
Else
    /* {Ψ_1(U,W),...,Ψ_κ(U,W)} = {U_1,...,U_κ} */
    Output {Ψ_1(U,W),...,Ψ_κ(U,W)}
```

Suppose that λ and κ are positive integers with $\lambda \leq \kappa$ and that $U' = \{U'_1, \ldots, U'_\kappa\}$ and $W' = \{W'_1, \ldots, W'_\kappa\}$ are two identical sets, each consisting of κ empty sets. Suppose that we run *Construct*$(\lambda, \kappa, U', W')$. The proof of lemma 3 demonstrates that $\rho(U', W')$ is less than or equal to $1/e$. The value of $\rho(U, W)$ does not increase during the execution of algorithm *Construct,* so *Construct*$(\lambda, \kappa, U', W')$ terminates with two sets U and W such that $\rho(U, W) \leq 1/e$ and $U_\ell \cup W_\ell = \{1, \ldots, g(\lambda, \kappa)\}$ for every integer ℓ in the range $1 \leq \ell \leq \kappa$. Since $\rho(U, W)$ is an upper bound on the probability that $\{\Psi_1(U, W), \ldots, \Psi_\kappa(U, W)\}$ does *not* form a λ-halver we conclude that this probability is at most $1/e$. However, since $U_\ell \cup W_\ell = \{1, \ldots, g(\lambda, \kappa)\}$ for every integer ℓ in the range $1 \leq \ell \leq \kappa$, $\Psi_\ell(U, W) = U_\ell$ for every ℓ. Therefore the probability that $\{\Psi_1(U, W), \ldots, \Psi_\kappa(U, W)\}$ does not form a λ-halver is either 0 (if $\{U_1, \ldots, U_\kappa\}$ is a λ-halver) or 1 (otherwise). We conclude that this probability is 0 and that $\{U_1, \ldots, U_\kappa\}$ is a λ-halver.

This completes our discussion about the problem of implementing algorithm *Realize_*$(1, h)$*-Relation*. As we mentioned at the beginning of this section, phase 3 of our main algorithm consists of running algorithm *Realize_*$(1, h)$*-Relation* simultaneously within each of the size k target groups. Therefore, phase 3 realizes a $(1, h)$-relation within each target group and it requires $\Theta(h \log^{1/2} p)$ communication steps. Combining this observation with theorem 1 we obtain the following.

Theorem 4. Suppose that we are given an arbitrary h-relation on a p-OCPC and that we run phases 1, 2, and 3. After $\Theta(h \log^{1/2} p)$ communication steps phases 1–3 terminate. At this point every message is either completely undelivered or it is delivered to its final destination. If h is less than or equal to $\log^{1/2} p$ then the probability that there are any completely undelivered messages is less than or equal to $p^{-\alpha}$.

4. Phase 4

In this section we will observe that the "brute-force" strategy can be used to realize an arbitrary h-relation on a p-OCPC in $O(p^2)$ expected communication steps. Phase 4 of our algorithm uses this strategy to delivery any messages that are still undelivered when phase 3 terminates.

We start by proving the following theorem.

Theorem 5. Suppose that we are given an arbitrary h-relation on a p-OCPC and that each processor runs successive iterations of the loop in phase 1. The expected number of iterations that will be run before all of the messages are delivered to their final destinations is $O(p^2)$.

Proof of Theorem 5: Let (P_i, P_r) be any pair of processors such that the h-relation includes a message from P_i to P_r. The analysis given in the proof of lemma 1 shows that the probability that the message from P_i to P_r is still undelivered after t iterations is at most $e^{-t/eh}$. There are at most p^2 pairs of processors so the probability that there are any messages undelivered after t iterations is at most $p^2 e^{-t/eh}$. Therefore the expected number of iterations that will be run before all of the messages are delivered is at most $1 + p^2 \sum_{t=1}^{\infty} e^{-t/eh} = \mathrm{O}(p^2)$. □

Let phase 4 be the algorithm in which each processor runs successive iterations of the loop in phase 1. Suppose that α is greater than or equal to 2. Then we can combine theorems 4 and 5 to obtain the following.

Theorem 6. Suppose that we are given an arbitrary h-relation on a p-OCPC and that we run phases 1, 2, 3, and 4. If h is less than or equal to $\log^{1/2} p$ then the expected number of communication steps that will be executed before all of the messages are delivered to their final destinations is $\Theta(h \log^{1/2} p)$.

5. Open Problems

In this paper we have described a pure p-OCPC algorithm that can realize an arbitrary h-relation and we have shown that its expected running time is $\Theta(h \log^{1/2} p)$. While this algorithm is faster than previously known h-relation realization algorithms for $h = \mathrm{o}(\log^{1/2} p)$, it is not *optimal* for any h. It would be interesting to know whether the algorithm could be modified so that its expected running time is reduced to $\mathrm{O}(h + \log^{1/2} p)$. Such an improvement would yield an optimal algorithm for realizing h-relations for $h = \Omega(\log^{1/2} p)$.

The main challenge involved would be to design a $\mathrm{O}(h + \log^{1/2} p)$ algorithm to replace phase 1. Phase 3 can already be replaced with an unpure $\Theta(h + \log^{1/2} p)$ algorithm which realizes a $(1, h)$-relation within each target group by sorting the messages according to their destinations. As we observed in section 4, the expected running time of phase 4 is already o(1).

Other open problems are discussed in [GT 92].

References

[AM 88] R. J. Anderson and G. L. Miller, Optical Communication for Pointer Based Algorithms, Technical Report CRI 88-14, Computer Science Department, University of Southern California, Los Angeles, CA 90089-0782 USA, 1988.

[Bol 86] B. Bollobás, *Combinatorics* (Cambridge Univ. Press, 1986).

[GT 92] M. Geréb-Graus and T. Tsantilas, Efficient Optical Communication in Parallel Computers, *Proc. ACM Symp. On Parallel Algorithms and Architectures* **4** (1992) 41–48.

[GV 92] A. V. Gerbessiotis and L. G. Valiant, Direct Bulk-Synchronous Parallel Algorithms, *Proc. Scandinavian Workshop on Algorithm Theory* **3** (1992).

[McC 92] W. F. McColl. General Purpose Parallel Computing, pre-print (1992).

[McD 89] C. McDiarmid, On the Method of Bounded Differences, Surveys in Combinatorics, London Math. Soc. Lecture Notes Series **141** (Cambridge Univ. Press, 1989) 148–188.

[Rag 88] P. Raghavan, Probabilistic Construction of Deterministic Algorithms: Approximate Packing Integer Programs, *Journal of Computer and System Sciences* **37** (1988) 130–143.

[Ra 92] S. B. Rao, Properties of an Interconnection Architecture Based on Wavelength Division Multiplexing, Technical Report TR-92-009-3–0054-2, NEC Research Institute, 4 Independence Way, Princeton, NJ 08540 USA, 1992.

[Spe 87] J. Spencer, *Ten Lectures on the Probabilistic Method* (SIAM, 1987).

[Va 90] L. G. Valiant, General Purpose Parallel Architectures, Chapter 18 of Handbook of Theoretical Computer Science, Edited by J. van Leeuwen (Elsevier 1990) (see especially p. 967)