Doubly Logarithmic Communication Algorithms for Optical Communication Parallel Computers*

Leslie Ann Goldberg, Sandia National Labs[1]

Mark Jerrum, University of Edinburgh[2]

Tom Leighton, MIT[3]

Satish Rao, NEC Research Institute[4]

ABSTRACT In this paper we consider the problem of interprocessor communication on parallel computers that have optical communication networks. We consider the *Completely Connected Optical Communication Parallel Computer* (OCPC), which has a completely connected optical network and also the *Mesh of Optical Buses Parallel Computer* (MOB-PC), which has a mesh of optical buses as its communication network. The particular communication problem that we study is that of realizing an *h-relation.* In this problem, each processor has at most h messages to send and at most h messages to receive. It is clear that any 1-relation can be realized in one communication step on an OCPC. However, the best previously known p-processor OCPC algorithm for realizing an arbitrary h-relation for $h > 1$ requires $\Theta(h + \log p)$ expected communication steps. (This algorithm is due to Valiant and is based on earlier work of Anderson and Miller.) Valiant's algorithm is optimal only for $h = \Omega(\log p)$ and it is an open question of Geréb-Graus and Tsantilas whether

* A preliminary version of this paper appeared in the proceedings of the 5th annual ACM Symposium on Parallel Algorithms and Architectures.

[1] Algorithms and Discrete Math Department, Sandia National Labs, MS 1110, PO Box 5800, Albuquerque, NM 87185-1110 USA, E-mail: lagoldb@cs.sandia.gov. This work was performed at Sandia National Laboratories and was supported by the U.S. Department of Energy under contract DE-AC04-76DP00789.

[2] Department of Computer Science, The University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ United Kingdom, E-mail: mrj@dcs.ed.ac.uk. This work was performed while the author was visiting the NEC Research Institute at Princeton NJ, USA. The work was supported by grant GR/F 90363 of the UK Science and Engineering Research Council and by Esprit Working Group "RAND".

[3] Mathematics Department and Laboratory for Computer Science, MIT, Cambridge, MA 02139 USA, E-mail: ftl@math.mit.edu. Supported by Air Force contract AFOSR-F49620-92-J-0125 and DARPA contracts N00014-91-J-1698 and N00014-92-J-1799.

[4] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540 USA, E-mail: satish@research.nj.nec.com

there is a faster algorithm for $h = o(\log p)$. In this paper we answer this question in the affirmative by presenting a $\Theta(h + \log\log p)$ communication step randomized algorithm that realizes an arbitrary h-relation on a p-processor OCPC. We show that if $h \leq \log p$ then the failure probability can be made as small as $p^{-\alpha}$ for any positive constant α. In the final section of our paper we use the OCPC algorithm as a sub-routine in a $\Theta(h + \log\log p)$ communication step randomized algorithm that realizes an arbitrary h-relation on a p-processor MOB-PC. Once again, we show that if $h \leq \log p$ then the failure probability can be made as small as $p^{-\alpha}$ for any positive constant α.

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 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 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], who called this model the *Local Memory PRAM.* Since then it has been studied by Valiant [Val 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). The feasibility of the OCPC from a engineering point of view is discussed in [AM 88, GT 92, and Rao 92]. See also the references in [McC 92].

In the first part of 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 [Val 90]) is a communication problem in which each processor has up to h messages that it wishes to send to other processors (assumed distinct). 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. Anderson and Miller [AM 88] have observed that an h relation can easily be realized in h communication steps if all of the processors are given *total* information about the h-relation to be realized.† A more interesting (and perhaps more realistic) situation

† To see this, model the communications between the p processors viewed as sources, and the p processors viewed as destinations, as the edges of a bipartite graph of order $2p$. Since the graph has maximum degree h, it is edge colorable with h colors, which can be

arises if we assume that initially each processor only knows about the messages that it wants to send and the processors learn about the h-relation only by receiving messages from other processors. This is the usual assumption, and the one that will be made here.

An OCPC algorithm for realizing h-relations is said to be *direct* if it has the property that the only messages that are exchanged by the processors are the original messages of the h-relation and these messages are sent only to their destinations. In this paper we prove the following:

1. The expected number of communication steps taken by *any* direct algorithm for realizing h-relations on a p-OCPC is $\Omega(h + \log p)$.
2. An arbitrary h-relation can be realized on a p-OCPC in $\Theta(h + \log\log p)$ communication steps. (Valiant has shown that an arbitrary h-relation can be realized in $\Theta(h + \log p)$ communication steps. In this paper we describe a $\Theta(h + \log\log p)$ communication step randomized algorithm that realizes an arbitrary h-relation on a p-OCPC and we show that if $h \leq \log p$ then the failure probability can be made as small as $p^{-\alpha}$ for any positive constant α.)

It is easy to see that any 1-relation can be realized in one 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 discovered a direct p-OCPC algorithm that runs for $\Theta(h)$ communication steps and delivers most of the messages in an arbitrary h-relation. In particular, the expected number of messages remaining after Anderson and Miller's algorithm is run is $O(p)$. Anderson and Miller were interested in the special class of h-relations in which each of the messages with a given destination has a unique label ℓ in the range $1 \leq \ell \leq h$. For this class of h-relations Anderson and Miller also discovered a deterministic $\Theta(h + \log p)$ communication step algorithm that delivers all of the messages in any h-relation that contains only $O(p)$ messages. Thus, their algorithms can be combined to obtain an algorithm that realizes an arbitrary h-relation from their special class in $\Theta(h + \log p)$ expected communication steps.

Valiant [Val 90] considered the general problem of realizing h-relations for $h > 1$. He discovered a $\Theta(h + \log p)$ expected communication step p-OCPC algorithm that realizes an arbitrary h-relation. Valiant's algorithm consists of the first phase of Anderson and Miller's algorithm followed by a second phase which redistributes the remaining $O(p)$ messages using parallel prefix, sorts them, and then sends them to the correct destinations. The second phase of Valiant's algorithm takes $\Theta(h + \log p)$ communication steps.

Prior to this work, Valiant's algorithm was the fastest known OCPC algorithm that can realize an arbitrary h-relation for $h > 1$. It is not direct, however. The fastest

---

interpreted as time steps.

known *direct* OCPC algorithm for realizing arbitrary h-relations is due to Geréb-Graus and Tsantilas [GT 92] and runs in $\Theta(h + \log p \log\log p)$ expected communication steps. In this paper we show that every direct OCPC algorithm for realizing h-relations takes $\Omega(h + \log p)$ expected communication steps. Furthermore, we describe a $\Theta(h + \log\log p)$ communication step p-OCPC algorithm that can realize an arbitrary h-relation and we show that if $h \leq \log p$ then the failure probability can be made as small as $p^{-\alpha}$ for any positive constant α. (The Θ notation does not hide any large constants in the running time of our algorithm.)

In this paper we also consider a model of computation known as the *Mesh of Optical Buses Parallel Computer* (MOB-PC). The $p \times p$ MOB-PC consists of p^2 processors, organized in a $p \times p$ array. The processors can perform local computations and can communicate with each other by message passing. As in the case of the OCPC, 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 send one message. Unlike the OCPC, the MOB-PC has the restriction that the destination of each message must be in the row or the column of its sender. (The reason for considering the MOB-PC is that this restriction makes it much easier to build than a p-OCPC (see [Rao 92]).) As in the case of the OCPC, 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 $p \times p$ mesh of buses is a member of a class of networks studied by Wittie [Wit 81] and suggested by Dowd as a method for optical interconnects [Dow 91]. Rao studied the MOB-PC in [Rao 92] and used a result of Leighton and Maggs to show that for $h \geq \log p$ an arbitrary h-relation can be realized on a $p \times p$ MOB-PC in $\theta(h)$ communication steps. In this paper we describe a $\Theta(h + \log\log p)$ communication step randomized algorithm that realizes an arbitrary h-relation on a $p \times p$ MOB-PC and we show that if $h \leq \log p$ then the failure probability can be made as small as $p^{-\alpha}$ for any positive constant α.

In order to motivate both our lower bound for direct OCPC algorithms and our OCPC algorithm (which is a sub-routine in our MOB-PC algorithm) consider the following experiment on an OCPC. Suppose that two processors P_i and P_j are both trying to send messages to a third processor P_d and that they adopt the following direct 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_d. Similarly, if P_j's coin comes up "heads" then P_j sends its message to P_d. On any given communication step P_d has probability $\frac{1}{2}$ of successfully receiving a message. Therefore the probability that P_d has not 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 two messages. After t communication steps we will expect to have $p2^{-t}$ processors that have received

no messages at all. Therefore it will take $\Omega(\log p)$ communication steps to realize the 2-relation.

Intuitively, the reason that so much time is needed is that the events are "too independent". In particular, the fact that most of the other messages are already delivered will not make it easier for P_i and P_j to send their messages to P_d. In order to obtain a sub-logarithmic OCPC algorithm we adopt the following strategy. We divide the set of p destinations into disjoint "target groups". During the first part of our algorithm we send each message in the h-relation to a randomly chosen processor within the target group containing its destination. As more and more messages are delivered to a given target group the probability that any remaining message is successfully delivered to the group in one communication step increases. Once all of the messages have been delivered to their target groups we solve the smaller problem of realizing an h-relation within each target group.

Our OCPC algorithm consists of four procedures. The first three procedures deliver the messages to their target groups and the last procedure realizes smaller h-relations within the target groups.

The methods that we use to deliver messages to target groups rely upon the fact that the number of messages being sent to each group is small compared to the size of the group. The first procedure of our algorithm (the "thinning" procedure) establishes this condition by delivering most of the messages in the h-relation to their final destinations. The thinning procedure is a direct OCPC algorithm and it is based on Anderson and Miller's algorithm. Proving that it satisfies the appropriate conditions requires a probabilistic analysis of dependent events. To do the analysis we use the "method of bounded differences" [McD 89, Bol 88].

After the thinning procedure has terminated the number of messages remaining will be $O(p/(h \log\log p))$ with high probability. The purpose of the second procedure (the "spreading" procedure) is to re-distribute these messages so that each sender has at most 1 message to send. After the spreading procedure terminates the third procedure delivers the remaining messages to their target groups. The bulk of the messages are delivered using a probabilistic tool called "approximate compaction". After the approximate compaction terminates the number of messages that have not been delivered to their target groups will be $O(p/\log^2 p)$ with high probability. Each remaining message is copied $\log p$ times and the processors are re-allocated so that $\log p$ processors can work together to send each message to its target group. (The approximate compaction technique and the copying technique were first used in PRAM algorithms such as those described in [CDHR 89] and in [GM 91] and [MV 91]. In this work we require a smaller failure probability for approximate compaction than previous authors because our target groups are only polylogarithmic

in size and we need to bound the probability of failure in any group.)

At the end of the third procedure the communication problem that remains consists of one h-relation within each target group. These h-relations could be realized in $\Theta(h + \log\log p)$ communication steps by simultaneously running the second phase of Valiant's algorithm within each target group, substituting a deterministic EREW sorting algorithm such as Cole's parallel merge sort (see [Col 88]) for the randomized sorting algorithm that Valiant uses.

Our fourth procedure is an alternative algorithm for realizing the h-relations within the target groups. It does not rely on efficient deterministic $O(\log p)$-time EREW sorting and it is therefore likely to be faster in practice. The algorithm is as follows. Each target group is sub-divided into disjoint sub-groups. Our "thinning", "spreading", and "deliver to target group" procedures are run simultaneously in each target group to deliver the messages in that group to the appropriate sub-groups. The communication problem remaining with each sub-group is an h-relation and this h-relation is realized using the second phase of Valiant's algorithm in which the sorting is done by Bitonic sort. With high probability the proportion of target groups for which this strategy delivers all of the messages is at least $1 - 1/\log^c p$ for a sufficiently large constant c. The processors from these target groups are then re-allocated and used to help the unsuccessful target groups finish realizing their h-relations. After the processors are re-allocated each unsuccessful target group sorts its messages using an enumeration sort due to Muller and Preparata [MP 75] which is fast in practice as well as in theory. The sorted messages are then delivered to their destinations.

The structure of this paper is as follows. In Section 2 we describe the OCPC algorithm in detail. We demonstrate that it uses $\Theta(h + \log\log p)$ communication steps and we prove that if $h \leq \log p$ then the probability that any messages are left undelivered can be made as small as $p^{-\alpha}$ for any positive constant α. In Section 3 we give the proof of the lower bound for direct OCPC algorithms. Finally, in Section 4 we describe the MOB-PC algorithm. We demonstrate that it uses $\Theta(h + \log\log p)$ communication steps and we prove that if $h \leq \log p$ then the probability that any messages are left undelivered can be made as small as $p^{-\alpha}$ for any positive constant α.

2. The OCPC Algorithm

Before we can define the OCPC algorithm we must describe the partition of the set of p processors into disjoint "target groups". The size of each target group will be a polynomial in $\log(p)$. To be precise, let c_1 denote a sufficiently large integer (the size of c_1 will depend upon the failure probability that we wish to obtain) and let k denote $\lceil \log^{c_1} p \rceil$. We will divide the p processors into approximately p/k target groups, each of size about k. To

simplify the presentation 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 $\{P_{k\ell}, \ldots, P_{k\ell+k-1}\}$. We will define the target group of any given message to be the target group containing the destination of the message and we will say that the message is destined for that target group.

The algorithm consists of the following four procedures:

- **Thinning.** At the beginning of the algorithm the number of messages destined for any given target group may be as high as hk. The goal of the thinning procedure is to deliver most of the messages to their final destinations so that by the end of the procedure the number of undelivered messages destined for any given target group is at most $k/(h\lceil c_2 \log\log p\rceil)$ for a sufficiently large constant c_2. If $h \leq \log p$ then this can be done in $\Theta(h + \log(h)\log\log\log(p))$ steps with probability at least $1 - p^{-\alpha}$ where the constant in the running time depends upon α and c_2.
- **Spreading.** At the end of the thinning procedure there will only be $O(p/(h\log\log p))$ undelivered messages. However, some senders may have as many as h undelivered messages. The spreading procedure spreads these out so that each sender has at most one to send. This can be done in $\Theta(h + \log\log p)$ communication steps with probability at least $1 - p^{-\alpha}$ where the constant in the running time depends upon α.
- **Deliver to Target Groups.** This procedure delivers all of the undelivered messages to their target groups. After it terminates each sender will have at most 2 undelivered messages to send and the destination of each undelivered message will be within the target group containing its sender. The procedure can be implemented in $\Theta(\log\log p)$ communication steps with probability at least $1 - p^{-\alpha}$ where the constant in the running time depends upon α.
- **Deliver within Target Groups.** This procedure delivers all messages to their final destinations. It can be implemented deterministically in $\Theta(h + \log\log p)$ steps by running the second phase of Valiant's algorithm twice in each target group. However this implementation may be slow in practice. In section 2.4 we describe an alternate implementation which runs in $\Theta(h + \log\log p)$ communication steps and succeeds with probability at least $1 - p^{-\alpha}$. (The constant in the running time depends upon α.)

We will use the following tool in the implementation of our algorithm. (For similar tools see [CDHR 89, GM 91, and MV 91].)

Definition 1. *The* (s, β, Δ) approximate compaction problem *is defined 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 $2k$.

Given

- *a p-OCPC in which at most s senders each have one message to send,*
- *a set of βs receivers which is known to all of the senders,*

deliver all but up to Δ of the messages to the set of receivers in such a way that each receiver receives at most one message. (During the delivery messages may only be sent from the original senders to the βs receivers.)

Lemma 1. *For any positive constant α there is a positive constant c_2 such that the $(s, \lceil c_2 \log\log p \rceil, \Delta)$ approximate compaction problem can be solved in $O(\log\log p)$ communication steps with failure probability at most $\alpha^{-\sqrt{s}} + s^{-\alpha(\Delta+1)}$ †.*

Using the (s, β, Δ) approximate compaction algorithm we can accomplish a variety of tasks. For example (following [CDHR 89] and [GM 91]) we use the algorithm to allocate $\lfloor \log p \rfloor$ processors to each message once the number of undelivered messages is reduced to $p/\lfloor \log p \rfloor^2$. We use the following definition in the proof of lemma 1.

Definition 2. *The (s, β, Δ)* approximate collection problem *is defined to be the same as the (s, β, Δ) approximate compaction problem except that we remove the requirement that each receiver receives at most one message.*

Lemma 2. *For any positive constant α there is a positive constant c_2' such that the $(s, 36, \Delta)$ approximate collection problem can be solved in at most $\lceil c_2' \log\log p \rceil$ communication steps with failure probability at most $\alpha^{-\sqrt{s}} + s^{-\alpha(\Delta+1)}$.*

Proof of Lemma 1. Let α be any positive constant and let $c_2 = 36c_2' + 1$, where c_2' is the constant associated with α in Lemma 2. Suppose that we are given an instance of the $(s, \lceil c_2 \log\log p \rceil, \Delta)$ approximate compaction problem. Partition the set of receivers into $\lceil c_2' \log\log p \rceil$ disjoint sets $R_1, R_2, \ldots$, each of size at least $36s$. Since the $(s, 36, \Delta)$ approximate collection problem can be solved in at most $\lceil c_2' \log\log p \rceil$ communication steps with failure probability at most $\alpha^{-\sqrt{s}} + s^{-\alpha(\Delta+1)}$, there is an algorithm with this failure probability that delivers all but up to Δ of the messages to the receivers in R_1 in only $\lceil c_2' \log\log p \rceil$ steps. To solve the $(s, \lceil c_2 \log\log p \rceil, \Delta)$ approximate compaction problem simply run this algorithm substituting the set R_i for R_1 on the ith communication step of the algorithm. □

Proof of Lemma 2. We say a sender is *active* initially if it contains a message. Our algorithm proceeds in a number of similar communication steps, where in step i each

† In fact there is a positive constant c_2 such that the (s, c_2, Δ) approximate compaction problem can be solved in $O(\log\log s)$ communication steps with small failure probability but lemma 1 is sufficient for our purposes.

active sender sends its message to a random location in the set of receivers. Each sender that successfully transmitted a message is considered *inactive.*

Let m denote $36s$. We must show that there are at most Δ active messages when the algorithm terminates. We use the following claim.

Claim 1. *Let c be a positive integer. If there at most m/r active senders left at step i, then the probability that there will be $f = \max\{\lceil m/r^{3/2} \rceil, \Delta + 1\}$ or more active senders left at step $i + 2c$ is at most $(2\mathrm{e}/\sqrt{r}\,)^{cf}$.*

We prove Claim 1 by imagining that in a certain step the m/r active senders make their random choice of destination in some fixed order. For there to be f active senders that do not transmit their message, there must be $\lceil f/2 \rceil$ times at which a sender chooses the same receiver as one chosen by a previous sender in this order. The probability of choosing the same receiver as a previous sender is at most $(m/r)/m = 1/r$. Thus, the probability of $\lceil f/2 \rceil$ such events occurring is bounded above by

$$\begin{aligned}
\binom{\lfloor m/r \rfloor}{\lceil f/2 \rceil} \left(\frac{1}{r}\right)^{\lceil f/2 \rceil} &\leq \left(\frac{2em}{rf}\right)^{\lceil f/2 \rceil} \left(\frac{1}{r}\right)^{\lceil f/2 \rceil} \\
&\leq \left(\frac{2em}{r^2 \max\left\{\lceil m/r^{3/2} \rceil, \Delta + 1\right\}}\right)^{\lceil f/2 \rceil} \\
&\leq \left(\frac{2em}{r^2 (m/r^{3/2})}\right)^{f/2} \\
&\leq \left(\frac{2e}{\sqrt{r}}\right)^{f/2}
\end{aligned}$$

We proceed by computing the probability that f active senders remain after $2c$ steps. It is easy to verify that the probability that f senders remain active after $2c$ steps in our algorithm is less than the probability that f senders remain active if each of the $2c$ successive steps is implemented by sending from all the processors that were active at the initial step. In this situation, the successive steps are independent thus the probability that there are f senders that never got a message through on any of the steps is at most the probability above raised to the $2c$th power. This proves Claim 1.

Now we define $r_0 = 36$, $r_j = r_{j-1}^{3/2}$, $f_j = \max\left\{\lceil m/r_j^{3/2} \rceil, \Delta + 1\right\}$, and $t = \min\{j : f_j = \Delta + 1\}$. The algorithm will run for $t + 1$ "supersteps" $0, 1, \ldots, t$, each superstep consisting of $2c$ steps as described above, with c a constant to be chosen later. Observe that the number of supersteps, and hence the total number of steps, is $\mathrm{O}(\log\log s)$ and is therefore $\mathrm{O}(\log\log p)$.

We say that superstep j is *successful* if, starting with at most m/r_j active senders, it finishes with (strictly) fewer than f_j active senders. Note that if supersteps $0, 1, \ldots, j$

are all successful, then the number of active senders remaining at the end of superstep j is strictly less than f_j. If all $t+1$ supersteps are successful then the number of active senders remaining at the end is at most Δ, as required.

Using Claim 1, we can bound the probability that some superstep fails by

$$\sum_{j=0}^{t} \left(\frac{2\mathrm{e}}{\sqrt{r_j}}\right)^{cf_j}.$$

Notice that each term where $r_j \leq m^{1/3}$ is at most $(\mathrm{e}/3)^{6c\sqrt{s}}$, and every other term is at most $(16\mathrm{e}^6/(9s))^{c(\Delta+1)/6}$. Thus the probability that *some* superstep fails is at most

$$(t+1)\left\{\left(\frac{\mathrm{e}}{3}\right)^{6c\sqrt{s}} + \left(\frac{16\mathrm{e}^6}{9s}\right)^{c(\Delta+1)/6}\right\}.$$

Observe that $t+1 = \mathrm{O}(\log\log s)$ so if c is chosen to be big enough relative to α this is at most $\alpha^{-\sqrt{s}} + s^{-\alpha(\Delta+1)}$ as required. □

We proceed by describing the implementation of the various steps of the algorithm.

2.1 Thinning

The thinning procedure is a direct OCPC algorithm which is based on Anderson and Miller's algorithm [AM 88]. It consists of $\mathrm{O}(\log h)$ phases. Intuitively, the goal of the ith phase is to reduce the problem of realizing a $h/2^{i-1}$-relation to the problem of realizing a $h/2^i$-relation. That is, the ith phase should get so many of the messages delivered that the remaining communication problem is "essentially" a $h/2^i$-relation. After the last phase the h-relation will be mostly realized except that there will be small number (at most $k/(h\lceil c_2 \log\log p\rceil)$) of undelivered messages destined for each target group.

Let c_3 be a sufficiently large constant (depending on c_1 and c_2 and the constant α in the desired failure probability) and let t_i denote $c_3\lceil h/2^{i-1} + \log h + \log\log\log p\rceil$. ($t_i$ denotes the number of communication steps in phase i.) Before phase i it will be the case that each participating sender has at most $h/2^{i-1}$ undelivered messages to send. During phase i each participating sender executes the following communication step t_i times.

```
Choose an integer j uniformly at random
from the set {1, ..., h/2^{i-1}}
    If there are at least j undelivered msgs. to be sent
        Send the jth undelivered msg. to its destination
```

After each communication step there is an acknowledgment step in which every receiver that receives a message sends an acknowledgment back to the sender indicating that the

message was delivered successfully. At the end of phase i any sender that has more than $h/2^i$ undelivered messages left to send stops participating.

We will prove the following theorem.

Theorem 1. *Suppose that $h \leq \log p$. Then with probability at least $1 - p^{-\alpha}$ the number of undelivered messages destined for any given target group is at most $k/(h\lceil c_2 \log\log p\rceil)$ after the thinning procedure terminates.*

In order to prove theorem 1 we will use the following notation. We will say that a given message is "participating" at any point in time if it is undelivered at that time and its sender is participating. We will say that a receiver is "overloaded" in phase i if at the start of phase i the number of participating messages with that destination is more than $h/2^{i-1}$. We will say that the receiver becomes overloaded in phase i if it is not overloaded in phases 1 through i but it is overloaded in phase $i+1$. We will say that a sender is "good" in phase i if it does not have a message to send to an overloaded receiver. For every target group T let $S(T)$ denote the set containing all senders in the h-relation with messages destined for T and let $N(T)$ denote the set containing all destinations of messages from processors in $S(T)$. Finally, let $S(N(T))$ be the set containing all senders with messages destined for members of $N(T)$. (Note that $|S(T)| \leq h\,|T|$, $|N(T)| \leq h^2|T|$, and $|S(N(T))| \leq h^3|T|$.) The theorem follows from the following lemma.

Lemma 3. *Suppose that $h \leq \log p$. Let i be an arbitrary phase of the thinning procedure and let T be any target group. With probability at least $1 - p^{-(\alpha+1)}$*

1. *At most $|N(T)|/(h^6\lceil c_2 \log\log p\rceil)$ receivers in $N(T)$ become overloaded in phase i*
2. *At most $|S(T)|/(h^6\lceil c_2 \log\log p\rceil)$ good senders in $S(T)$ stop participating at the end of phase i.*

Proof of Theorem 1. To see that the theorem follows from lemma 3 note that the number of target groups is at most p/k and the number of phases is $O(\log h)$ so with probability at least $1 - p^{-\alpha}$ (1.) and (2.) hold for all phases i and target groups T. Suppose that this is the case and consider any particular target group T. A message that is destined for T will be delivered by the thinning procedure unless either (1) there is a phase in which its sender is not good (in which case the sender could possibly stop participating) or (2) its sender stops participating even though it is good. The number of messages that are destined for T and are not delivered is therefore at most

$$\log(h) \times \big(\; h^2|N(T)|/(h^6\lceil c_2 \log\log p\rceil) + \\ h|S(T)|/(h^6\lceil c_2 \log\log p\rceil) \; \big).$$

This is at most $k/(h\lceil c_2 \log\log p\rceil)$. □

The proof of lemma 3 will use the following "independent bounded differences inequal-

ity" of McDiarmid [McD 89]. (The inequality is a development of the "Azuma martingale inequality"; a similar formulation was also derived by Bollobás as [Bol 88].)

Theorem 2. [McDiarmid] *Let $x_1, \ldots, x_n$ be independent random variables, with x_i taking values in a set A_i for each i. Suppose that the (measurable) function $f : \prod A_i \to \mathbb{R}$ satisfies $|f(\overline{x}) - f(\overline{x}')| \leq c_i$ whenever the vectors $\overline{x}$ and $\overline{x}'$ differ only in the ith coordinate. Let Y be the random variable $f(x_1, \ldots, x_n)$. Then for any $t > 0$,*

$$\Pr\left(|Y - \mathrm{E}(Y)| \geq t\right) \leq 2\exp\left(-2t^2 / \textstyle\sum_{i=1}^{n} c_i^2\right). \quad \square$$

Proof of Lemma 3. Suppose that $h \leq \log p$, let i be an arbitrary phase of the thinning procedure, and let T be any target group. Let x_j denote the sequence of integers randomly chosen by processor P_j during phase i.

We will start by proving that with probability at least $1 - p^{-(\alpha+2)}$ at most $|N(T)|/(h^6 \lceil c_2 \log\log p \rceil)$ receivers in $N(T)$ become overloaded in phase i.

Let $Y = f(\{x_j \mid P_j \in S(N(T))\})$ be the number of receivers in $N(T)$ that become overloaded during phase i. Let R be any receiver in $N(T)$ that is not overloaded in phases 1 through i and let s_j denote the number of participating messages that are destined for R at the jth communication step of phase i. (Note that these messages are not necessarily *sent* on the jth communication step.) The probability that R receives a message on this step is $s_j\,(2^{i-1}/h)\,(1 - 2^{i-1}/h)^{s_j - 1}$. There is a positive constant ρ such that this probability is greater than or equal to ρ for every s_j that is greater than or equal to $h/2^i$. (Note that R cannot become overloaded in phase i if s_j is ever less than $h/2^i$.) Therefore, the probability that R becomes overloaded is at most

$$\sum_{j=0}^{h/2^i - 1} \binom{t_i}{j} \rho^j (1-\rho)^{t_i - j}.$$

Furthermore, as long as c_3 is sufficiently large (i.e., t_i is sufficiently large compared to j) there is a constant $c_4 > 1$ such that the above sum is at most $c_4^{-t_i}$. Therefore the expected number of processors in $N(T)$ that become overloaded in phase i is at most $N(T)c_4^{-t_i}$ which is at most $|N(T)|/(2h^6 \lceil c_2 \log\log p \rceil)$ as long as c_3 is sufficiently large.

If the value of x_j changes for any j then Y changes by at most h. Therefore, by the bounded differences inequality of theorem 2, the probability that Y is greater than $|N(T)|/(h^6 \lceil c_2 \log\log p \rceil)$ is at most

$$2\exp(-2\,|N(T)|^2/(4h^{12}\,\lceil c_2 \log\log p \rceil^2\,|S(N(T))|\,h^2)).$$

This is at most $p^{-(\alpha+2)}$ as long as the constant c_1 is sufficiently large (i.e., the target groups are sufficiently large). (Here we use the fact that $h \leq \log p$.)

We now prove that with probability at least $1 - p^{-(\alpha+2)}$ at most $|S(T)|/(h^6 \lceil c_2 \log\log p \rceil)$ good senders in $S(T)$ stop participating at the end of phase i.

Let $Y = f(\{x_j \mid P_j \in S(N(T))\})$ be the number of good senders in $S(T)$ that stop participating at the end of phase i.

Let S be any good sender in $S(T)$ that participates in phase i and let s_j denote the number of participating messages that S has to send at the jth communication step of phase i. Let $d_{\ell,j}$ denote the number of participating messages at the jth communication step that have the same destination as the ℓth message that S has to send. (Since S is good each $d_{\ell,j}$ is less than or equal to $h/2^{i-1}$.) The probability that S sends a message successfully on the jth communication step is $\sum_{\ell=1}^{s_j} (2^{i-1}/h)\,(1 - 2^{i-1}/h)^{d_{\ell,j}-1}$. As before, there is a positive constant ρ such that this probability is greater than or equal to ρ for every s_j that is greater than or equal to $h/2^i$. Therefore, the probability that S stops participating is at most

$$\sum_{j=0}^{h/2^i-1} \binom{t_i}{j} \rho^j (1-\rho)^{t_i - j}.$$

As in the proof of the first part of the lemma, we conclude that the expected number of good senders in $S(T)$ that stop participating at the end of phase i is at most $|S(T)|/(2h^6 \lceil c_2 \log\log p \rceil)$.

If the value of x_j changes for any j then Y changes by at most h^2. Therefore, by the bounded differences inequality of theorem 2, the probability that Y is greater than $|S(T)|/(h^6 \lceil c_2 \log\log p \rceil)$ is at most

$$2\exp(-2\,|S(T)|^2/(4\,h^{12}\,\lceil c_2 \log\log p \rceil^2\,|S(N(T))|\,h^4)).$$

This is at most $p^{-(\alpha+2)}$ as long as the constant c_1 is sufficiently large (i.e., the target groups are sufficiently large). (Once again, we use the fact that $h \leq \log p$.) □

2.2 Spreading

Let α be any positive constant and let c_2 be the constant associated with α that is defined in lemma 1. At the end of the thinning procedure there will be at most $p/(h \lceil c_2 \log\log p \rceil)$ undelivered messages. We wish to spread these out so that each sender has at most one to send. To do this we observe that there are at most $p/(h \lceil c_2 \log\log p \rceil)$ senders with undelivered messages. Suppose (without loss of generality) that h divides p and partition the set of p receivers into h disjoint sets $R_1, \ldots, R_h$ of size p/h. Perform

a $(p/(h\lceil c_2 \log\log p\rceil), \lceil c_2 \log\log p\rceil, 0)$ approximate compaction to send the first message from each sender to a unique processor in R_1. (The probability that this will succeed is at least

$$1 - \alpha^{-\sqrt{p/(h\lceil c_2 \log\log p\rceil)}} - (p/(h\lceil c_2 \log\log p\rceil))^{-\alpha}.)$$

Finally, send the remaining messages to $R_2, \ldots, R_h$ in $\Theta(h)$ communication steps with no contention using the following strategy. If the 1st message of sender i was sent to the jth cell of R_1 by the approximate compaction then send the ℓth message of sender i to the jth cell of R_ℓ for $1 < \ell \leq h$.

2.3 Deliver to Target Groups

Let α be any positive constant and let c_2 be the constant associated with α that is defined in lemma 1. At the end of the spreading procedure each sender will have at most one undelivered message to send and each target group will have at most $k/(h\lceil c_2 \log\log p\rceil)$ undelivered messages to receive. Our goal is to deliver the messages to the target groups. After this procedure terminates each processor will have at most 2 undelivered messages to send and the destination of each undelivered message will be within the target group containing its sender.

We have two methods for implementing this procedure in $\Theta(\log\log p)$ communication steps. The simpler method (which we describe here) involves making copies of messages but the other method does not. The simpler of the two methods consists of two phases.

We first describe phase 1. Consider any target group T. At the start of the procedure there are at most $k/\lceil c_2 \log\log p\rceil$ senders each of which has one message to send to the target group. Let ℓ denote $\lfloor \log p \rfloor$. We send all but up to k/ℓ^2 of these messages to T in $O(\log\log p)$ steps by doing a $(k/\lceil c_2 \log\log p\rceil, \lceil c_2 \log\log p\rceil, k/\ell^2)$ approximate compaction. We can do this in parallel for each target group and the probability that it fails for any target group is at most

$$\tfrac{p}{k}(\alpha^{-\sqrt{k/\lceil c_2 \log\log p\rceil}} + (k/\lceil c_2 \log\log p\rceil)^{-\alpha(k/\ell^2+1)})$$

which is sufficiently small as long as the constant c_1 in the definition of k is sufficiently large.

We will use the phrase "completely undelivered" to describe all messages that were undelivered before phase 1 and were not delivered to their target groups during phase 1. At the end of phase 1 each sender has at most one completely undelivered message to send, each member of each target group has received at most one message, and the number of completely undelivered messages is at most p/ℓ^2. Choose ℓ disjoint sets $R_1, \ldots, R_\ell$ of size $\lfloor p/\ell \rfloor$ from the set of p receivers and let Q_j denote the set consisting of the jth

receiver from each of $R_1, \ldots, R_\ell$. Next, send all of the completely undelivered messages to R_1 by performing a $(p/\ell^2, \lceil c_2 \log\log p \rceil, 0)$ approximate compaction. (This fails with probability at most $\alpha^{-\sqrt{p/\ell^2}} + (p/\ell^2)^{-\alpha}$.) Finally (for each j in parallel) the processors in Q_j copy the message received at the jth receiver in R_1 (if there is one) to the other processors in Q_j. (This takes $\Theta(\log\log p)$ communication steps.)

At this point each completely undelivered message is stored at each of the ℓ processors in Q_j (for some j) and each processor stores at most one completely undelivered message. The following communication step is now performed in parallel by all processors. If the ith processor in Q_j has a completely undelivered message to send then it chooses an integer uniformly at random from the set $\{\gamma \mid (1 \leq \gamma \leq k) \text{ and } (\gamma \bmod \ell = i)\}$ and it sends the message to the γth processor in its target group. The probability that the ith processor in Q_j is unsuccessful is at most $1/\ell$ and this probability is independent of the probability that the other processors in Q_j succeed so the probability that there is a completely undelivered message that is not delivered at least once to its target group in this communication step is at most $p\ell^{-\ell}$ which is sufficiently small.

For each j in parallel the processors in Q_j perform parallel prefix to select one of the delivered copies. They then send messages "cancelling" any other copies that were delivered to their target group. This takes $\Theta(\log\log p)$ communication steps. Note that each processor receives at most 2 messages during the procedure — one in phase 1 and one in phase 2.

2.4 Deliver within Target Groups

When this procedure begins each sender has at most 2 undelivered messages to send and the destination of each undelivered message is within the target group containing its sender. Our goal is to deliver all of the undelivered messages.

This procedure can be implemented deterministically in $\Theta(h + \log\log p)$ steps by running the second phase of Valiant's algorithm [Val 90] twice within each target group. The algorithm within each target group is as follows. First we consider only one undelivered message per sender. These messages are sorted by destination in $\Theta(\log\log p)$ communication steps using an EREW sorting algorithm such as Cole's parallel merge sort [Col 88]†. Then the sorted messages are delivered to their destinations without contention in $\Theta(h)$ communication steps. Next the process is repeated for the remaining undelivered messages.

In this section we describe an alternative implementation of the procedure. It does not rely on efficient deterministic $O(\log p)$-time EREW sorting and it is therefore likely to

† Valiant uses a randomized parallel sorting algorithm instead of using parallel merge sort. We cannot do that here because we want to be able to claim that (with high probability) the messages are successfully (and quickly) sorted in all of our target groups.

be faster in practice.

The main idea is as follows. We start by sub-dividing each target group into target sub-groups. We then run the "thinning", "spreading", and "deliver to target group" procedures within each target group to deliver the messages to their target sub-groups. If these three procedures succeed within a target group then each sender in the group will have at most 2 undelivered messages to send and the destination of each undelivered message will be within the target *sub-group* of its sender. We can now run the second phase of Valiant's algorithm twice within each target sub-group to deliver the messages in the target group to their final destinations. Since the sub-groups are very small we can use Bitonic sort (which is fast in practice) to do the sorting. With high probability the proportion of target groups for which the "thinning", "spreading", or "deliver to target group" procedures fail will be $O(k^{-3})$. We now allocate a group of k^2 extra processors to each of these target groups and we use these extra processors to sort the messages using a counting sort that is fast in practice as well as in theory.

We now describe the procedure in more detail. The communication problem within each target group can be viewed as the problem of realizing an h-relation on a k-OCPC. Therefore we can run the "thinning", "spreading", and "deliver to target group" procedures simultaneously *within* each target group. Before we can do that we must partition each target group into target sub-groups. Let the size of the target sub-groups be $k' = \lceil \log^{c_5} k \rceil$ where c_5 is a constant that is sufficiently large that the probability that the "thinning", "spreading", and "deliver to target group" procedures fail within a target group is at most k^{-3}. (In order to simplify the presentation in this section we will assume that k' divides k. The case in which k' does not divide k is no more difficult – it is simply messier. Similarly, we will assume that k^3 divides p.) After the "deliver to target group" procedure terminates within each target group run the second phase of Valiant's algorithm twice within each target sub-group, using Bitonic sort to do the sorting. (This takes $\Theta(h + \log^2 k')$ communication steps.) If the "thinning", "spreading", and "deliver to target group" procedures succeeded within a target group then all of its messages are now delivered. (This will happen with probability at least $1 - k^{-3}$.)

We now describe the second part of the procedure — the allocation of extra processors to help target groups that have not finished. Partition the set of target groups into p/k^2 disjoint sets $S_1, \ldots, S_{p/k^2}$. Each set S_ℓ contains k target groups and is called a *target super-group*. Partition the set of target super-groups into k disjoint sets $\mathcal{C}_1, \ldots, \mathcal{C}_k$. Each set $\mathcal{C}_\ell$ contains p/k^3 target super groups (and therefore p/k^2 target groups) and is called a *collection* of target super-groups. Note that with probability at least $1 - k\exp(-p/3k^5)$ each collection of target super-groups contains at most $2p/k^5$ un-finished target groups. Suppose that this is the case. Each target group and each target super-group performs a parallel prefix to determine whether or not it has finished. (This takes $\Theta(\log\log p)$

communication steps.) Next each processor that is part of an un-finished target group attempts to find a finished target super-group. In particular, if the processor is the jth member of the target group then it chooses a target super-group uniformly at random from $\mathcal{C}_j$ and it sends a message to the first processor in the target super-group asking whether the target super-group is finished. The probability that a given member of a given unfinished target group fails to find a finished super-group is at most $3/k$ (the probability that the super-group chosen is not finished is at most $2/k^2$ and the probability that the query is sent to the same destination as some other query is at most $2/k$). Furthermore the queries from any given target group are independent of each other so the probability that every processor in a given unfinished target group fails to find a finished super-group is at most $(3/k)^k$ and the probability that there exists an unfinished target group that fails to find a finished super-group is at most $p(3/k)^k$ which is sufficiently small. Each unfinished target group then performs a parallel prefix to choose a single finished super-group.

At this point each un-finished target group has identified a single finished super-group containing k^2 processors. Consider the k^2 processors to be organized in a k by k matrix. We now run Valiant's algorithm twice in each un-finished target group. The message are sorted using Muller and Preparata's algorithm [MP 75] which works as follows. The ith processor of the un-finished target group sends its message (if it has one) to all of the processors in the ith row. (This takes $\Theta(\log\log p)$ communication steps.) If the processor in the ith row of the ith column gets a message then it sends this message to all of the processors in the ith column and the processors in the ith column perform parallel prefix to determine its rank. (Again, this takes $\Theta(\log\log p)$ communication steps.) Finally (in 1 communication step) the message with rank i is sent to the ith processor in the un-finished target group.

3. A Lower Bound for Direct OCPC Algorithms

The algorithm described in the previous section often sends a message to a processor other than its final destination, i.e., the algorithm is not *direct*. Using a non-direct strategy in a network that allows direct routing may seem strange at first, and one might question its necessity. In this section we prove a lower bound that demonstrates that any sublogarithmic OCPC algorithm must necessarily use non-direct routing.

Theorem 3. *Let $\mathcal{A}$ be any direct (randomized) OCPC algorithm that can realize any 2-relation with success probability at least $\frac{1}{2}$. Then there is a 2-relation which $\mathcal{A}$ takes $\Omega(\log p)$ communication steps to realize.*

Proof of Theorem 3. Consider any direct randomized OCPC algorithm that runs for $t \leq \lfloor \frac{1}{3}\log p \rfloor$ steps. We shall construct a 2-relation ρ such that the probability that the algorithm successfully realizes ρ is exponentially small (in p). In the 2-relation ρ, each

processor has at most one message to send.

Consider a processor P_i that is not itself the destination of any messages and has a single message to send to P_d, but is blocked every time it attempts to transmit. Since P_i receives no external stimulus, we can imagine that P_i selects its transmission strategy at random in advance of the first time step. A strategy for P_i to transmit to P_d (under the blocking regime) can be coded as a binary word of length t, where a 1 in position t' indicates that P_i is to attempt to send its message at time step t'.

For convenience, assume that p is divisible by 4. The 2-relation ρ is the union of $p/4$ subrelations, each consisting of a pair of sending processors attempting to send a single message each to a common destination. The $3p/4$ processors in the 2-relation are distinct. The $p/4$ subrelations will be selected sequentially. Note that at any stage there will be $f \geq p/4$ "free" processors from which the next pair of senders may be selected. To make the selection, first choose a free destination processor P_d. Observe that, since the number of possible transmission strategies is 2^t, there must exist a strategy $\sigma \in \{0,1\}^t$ such that the expected number of free senders that choose strategy σ to send to P_d under the blocking regime is at least $f2^{-t}$. Thus there is a free sender, say P_i, that chooses strategy σ with probability at least $2^{-t} \geq p^{-1/3}$; and a different free sender, say P_j, that chooses σ with probability at least

$$(f2^{-t} - 1)/(f - 1) \geq 2^{-t} - f^{-1} \geq p^{-1/3} - 4p^{-1},$$

which is at least $\frac{1}{2}p^{-1/3}$ for $p \geq 24$. Now add to ρ the subrelation that requires P_i and P_j each to send a single message to P_d.

Note that P_i and P_j select strategies independently, so the probability that they both select σ is at least $\frac{1}{2}p^{-2/3}$; thus the probability that P_i and P_j fail to get rid of their messages is also at least $\frac{1}{2}p^{-2/3}$. Since there are $p/4$ subrelations forming ρ, the probability that ρ is successfully realized is at most $(1 - \frac{1}{2}p^{-2/3})^{p/4}$, which is less than $\exp(-p^{1/3}/8)$. □

It may be observed from the proof that a direct algorithm requires a logarithmic number of steps to achieve even inverse polynomial success probability.

4. The MOB-PC Algorithm

In this section we describe a $\theta(h + \log\log p)$ communication step algorithm that realizes an arbitrary h-relation on a $p \times p$ MOB-PC. We show that if $h \leq \log p$ then the failure probability can be made as small as $p^{-\alpha}$ for any positive constant α.

As each row and each column of a $p \times p$ MOB-PC is itself a p-OCPC we start by considering a p-OCPC. As in Section 2, we divide the p processors into target groups of

size $k = \lceil \log^{c_1} p \rceil$. A *target group h-relation* is defined to be a communication problem in which each processor has up to h messages that it wishes to send. The destinations of these messages are target groups, and each target group is the destination of at most hk messages. We will use the following lemma.

Lemma 4. *Suppose that $h \leq \log p$ and let α be any positive constant. Then there is a p-OCPC algorithm that can realize an arbitrary target group h-relation in $O(h + \log\log p)$ steps with failure probability $3p^{-\alpha}$.*

Proof of Lemma 4: Suppose that we are given a target group h-relation. As in Section 2, we will let $S(T)$ denote the set containing all senders that have messages destined for target group T. Let $M(S(T))$ denote the set of messages that are to be sent by these senders. Let each message choose a destination uniformly at random from within its target group. Let $h' = 8eh + \log\log p$ and let $h'' = h'/2$. We will say that a message is *externally bad* with respect to a target group T if the message has the same destination as at least h'' other messages that are not sent from senders in $S(T)$. We will say that a message is *internally bad* with respect to a target group T if it has the same destination as at least h'' other messages that are sent from senders in $S(T)$. We will say that a sender is *initially good* unless one or more of its messages is (externally or internally) bad. We will prove the following claim.

Claim 2. *With probability at least $1 - p^{-\alpha}$ every set $S(T)$ contains at most $k/(2h'^2\lceil c_2 \log\log p\rceil)$ senders that are not initially good.*

Suppose that every set $S(T)$ contains at most $k/(2h'^2\lceil c_2 \log\log p\rceil)$ senders that are not initially good and that we start to deliver messages to their destinations by running the thinning procedure from Section 2.1 using h' as the value of the variable "h". It is easy to see that we can modify the proof of Lemma 3 to obtain the following. (The following lemma is the same as Lemma 3 except for the factor of 2 in the denominator.)

Lemma 3′. *Suppose that $h' \leq \log p$. Let i be an arbitrary phase of the thinning procedure and let T be any target group. With probability at least $1 - p^{-(\alpha+1)}$*

1. *At most $|N(T)|/(2\,h'^6\lceil c_2 \log\log p\rceil)$ receivers in $N(T)$ become overloaded in phase i*
2. *At most $|S(T)|/(2\,h'^6\lceil c_2 \log\log p\rceil)$ good senders in $S(T)$ stop participating at the end of phase i.*

We conclude that with probability at least $1 - 2p^{-\alpha}$ the number of messages that are not delivered to a given target group is at most the sum of

1. $k/(2h'\lceil c_2 \log\log p\rceil)$ (these messages may not be delivered because their sender is not initially good)

2. $k/(2h'\lceil c_2 \log\log p\rceil)$ (these messages may not be delivered because their sender stops participating or stops being good during the thinning).

We conclude that with probability at least $1 - 2p^{-\alpha}$ the number of undelivered messages destined for any given target group is at most $k/(h'\lceil c_2 \log\log p\rceil)$ after the thinning procedure terminates. Therefore, we can deliver the rest of the messages to their target groups using the "Spreading" procedure from Section 2.2 and the "Deliver to Target Groups" procedure from Section 2.3. In the remainder of this section it will be important to have our algorithm for realizing target group h-relations behave symmetrically with respect to the different destinations within a target group. We can achieve this goal by modifying the "Deliver to Target Group" procedure from Section 2.3 as follows.

1. In the first part of the procedure we deliver messages to their target groups using "approximate collection" rather than "approximate compaction".
2. In the second part of the procedure (the part involving copies) the "winner" is chosen uniformly at random (rather than arbitrarily) from amongst the successfully delivered copies.

We now finish the proof of Lemma 4 by proving Claim 2. Let T be any target group. We will show that the probability that $M(S(T))$ contains more than $k/(4h'^2\lceil c_2 \log\log p\rceil)$ externally bad messages is at most $\frac{1}{2}p^{-\alpha}(k/p)$. Then we will show that the probability that $M(S(T))$ contains more than $k/(4h'^2\lceil c_2 \log\log p\rceil)$ internally bad messages is at most $\frac{1}{2}p^{-\alpha}(k/p)$.

First we consider externally bad messages. We will say that a processor P is *externally crowded* with respect to a target group T if there are at least h'' messages which are not in $M(S(T))$ and have destination P. A set of b members of a target group are all externally crowded only if at least bh'' messages have destinations in the set. Therefore, the probability that there is a set of b members of a target group that are all externally crowded is at most

$$\binom{p}{k}\binom{k}{b}\binom{kh}{bh''}\left(\frac{b}{k}\right)^{bh''}.$$

We can use Stirling's approximation to show that for $b = k/h''^6$ this quantity is at most $(p/k)2^{-k/h''^5}$. Therefore, with probability at least $1 - (p/k)2^{-k/h''^5}$ every target group has at most k/h''^6 processors which are externally crowded with respect the T. Suppose that this is the case. Then the probability that a message in $M(S(T))$ chooses a destination which is externally crowded with respect to T is at most h''^{-6}. Using a Chernoff bound, we see that with probability at least $1 - \exp(-|M(S(T))|/(3 \times h''^6))$ at most $2|M(S(T))|/h''^6$ messages in $M(S(T))$ choose

a destination which is externally crowded with respect to T. Note that as long as p is sufficiently large then $2\,|M(S(T))|/h''^6 \leq k/(4h'^2\lceil c_2 \log\log p\rceil)$. Also, as long as $|M(S(T))| \geq k/(4h'^2\lceil c_2 \log\log p\rceil)$ and $h' \leq \log p$ and the constant c_1 is sufficiently large the sum of $(p/k)2^{-k/h''^5}$ and $\exp(-|M(S(T))|\,/\,(3 \times h''^6))$ is at most $\frac{1}{2}p^{-\alpha}\,(k/p)$.

We now consider internally bad messages. We start by calculating an upper bound on the probability that a message is internally bad. This probability is at most

$$\sum_{j=h''}^{hk-1} \binom{hk-1}{j} \frac{1}{k^j} \left(1-\frac{1}{k}\right)^{hk-1-j}.$$

We can use Stirling's approximation to show that this sum is $O(2^{-h''})$. So the expected number of messages in $M(S(T))$ which are internally bad is $O(|M(S(T))|2^{-h''})$.

Let x_i be a random variable which denotes the destination of the ith message in $M(S(T))$ and let Y be a random variable denoting the number of internally bad messages in $M(S(T))$. (Y is a function of $x_1, \ldots, x_{|M(S(T))|}$.) If we change one of the x_i's then we change Y by at most $h''+1$. Therefore, by Theorem 2 (the bounded differences inequality),

$$\Pr(Y \geq k/(4h'^2\lceil c_2 \log\log p\rceil)) \leq 2\exp\left(-2\left(\frac{k}{4h'^2\lceil c_2 \log\log p\rceil} - \mathrm{E}(Y)\right)^2 \Big/ \left(|M(S(T))|\,(h''+1)^2\right)\right)$$

Since $\mathrm{E}(Y) \leq \frac{k}{8h'^2\lceil c_2 \log\log p\rceil}$ (for big enough p) the probability is at most

$$2\exp(-k/(32h'^4\lceil c_2 \log\log p\rceil^2 h^2 (h''+1)^2)).$$

This quantity is at most $\frac{1}{2}p^{-\alpha}\,(k/p)$ as long as c_1 is sufficiently large and h' is at most $\log p$. □

Now that we have proved Lemma 4 we are ready to describe our algorithm for realizing h-relations on a $p \times p$ MOB-PC. We have already observed that each row and each column of a MOB-PC is a p-OCPC. We will divide each row and each column of the MOB-PC into target groups of size k. A *block* of the MOB-PC is defined to be a $k \times k$ sub-MOB-PC in which each row is a row target group of the original MOB-PC and each column is a column target group of the original MOB-PC. We will use the phrase *column of blocks* to refer to a collection of p/k blocks which together make up k columns of the MOB. Finally, we will sub-divide each column of blocks into p/k^2 *super-blocks* in which each super-block consists of k blocks. (As in Section 2.4 we will simplify the presentation by assuming that k^3 divides p. We will also assume that $h \leq \log p$.)

The algorithm has five steps.

1. On each Row: Each message picks a random row target group and the messages are routed to the target groups.
2. On each Column: Each message chooses as its immediate destination the column target group that intersects the row of its final destination. The messages are routed to the target groups.
3. Within each Block: Each message chooses an immediate destination uniformly at random from its row. The messages are routed to their immediate destinations using the OCPC algorithm in each row. Each message that was successfully delivered to its immediate destination chooses as its new immediate destination the processor which is in its column and in the row of its final destination. The messages are routed to their immediate destinations using the OCPC algorithm in each column. If the block contains a message that was not successfully delivered to the row of its final destination then we say that the block *failed.* Every processor in the block is notified of the failure.
4. If any block of a super-block failed then we say that the super-block failed. Every processor in the super-block is notified of the failure. Each failed block attempts to allocate a super-block which has not failed from within its column of blocks. After allocating a super-block, the failed block copies all of its messages to each of the blocks in the super-block. Each of these blocks then repeats Step 3. If there is a block in the super-block which does not fail then the first such block copies the (delivered) messages back to the original failed block.
5. On each Row: Each message is routed to its final destination.

We will conclude the section by considering each of the five steps. For each step we will discuss the method that is used to implement the step and also the failure probability of the method.

At the beginning of Step 1 each processor has at most h messages. Each message then picks a random row target group. Using a Chernoff bound we see that the probability that a given target group is the destination of more than $2hk$ messages is at most $e^{-hk/3}$ so the probability that there is such a target group is at most $p \times (p/k) \times e^{-hk/3}$ which is at most $p^{-\alpha}$ as long as c_1 is sufficiently large. Suppose that every target group is the destination of at most $2hk$ messages. Then we can use the method described in the proof of Lemma 4 to deliver the messages to their target groups in $O(h + \log\log p)$ steps. The probability that this method fails is at most $3p^{-\alpha}$ for any positive constant α.

At the beginning of Step 2 each processor has at most h_2 messages where h_2 is h plus the number of time-steps used in Step 1. If it is also true that every target group is the destination of $O(hk)$ messages in Step 2 then we can use the method described in the

proof of Lemma 4 to deliver the messages to the target groups in $O(h + \log\log p)$ steps. We will conclude our discussion of Step 2 by showing that with high probability each target group is the destination of $O(hk)$ messages.

Let T be any column target group and let C be the column of T. There are at most hkp messages which have final destinations in rows which intersect T. These are the only messages which could be destined for T on Step 2. We will refer to them as the set of "potentially relevant" messages. Each potentially relevant message will be destined for T on Step 2 if and only if it is delivered to column C on Step 1. Therefore, our goal is to prove that with high probability only $O(hk)$ of the potentially relevant messages are delivered to column C on Step 1.

We start out by using a Chernoff bound to prove that with probability at least $1 - \exp(-hk^2/3)$ only $2hk^2$ of the potentially relevant messages select target groups that intersect C in Step 1. We refer to these messages as "relevant" messages. Our goal is to prove that with high probability only $O(hk)$ of the relevant messages are delivered to column C on Step 1.

We will use the following theorem of Hoeffding which is included in McDiarmid's paper [McD 89].

Theorem 4. [Hoeffding] *Let the random variables $X_1, \ldots, X_p$ be independent, with $0 \leq X_i \leq 1$ for each i. Let $\overline{X} = \frac{1}{p}\sum_i X_i$ and $\mu = \mathrm{E}[\overline{X}]$. Then for $0 \leq t < 1 - \mu$,*

$$\Pr(\overline{X} \geq t + \mu) \leq \left[\left[\frac{\mu}{\mu + t}\right]^{\mu+t}\left[\frac{1-\mu}{1-\mu-t}\right]^{1-\mu-t}\right]^{p}. \quad \square$$

To apply Hoeffding's inequality, let X_i be h_2^{-1} times the number of relevant messages that are delivered to row i of column C on Step 1. Observe that $0 \leq X_i \leq 1$ and that the X_i's are independent. Note that $\overline{X}$ is $(h_2 p)^{-1}$ times the number of relevant messages that are delivered to column C in Step 1. Recall that the algorithm for realizing target group h-relations behaves symmetrically with respect to the destinations forming a particular target group; thus the expected number of relevant messages delivered to column C on Step 1 is k^{-1} times the expected number of relevant messages. Therefore, μ is at most $2hk/(h_2 p)$. Let t denote $4hk/(h_2 p)$. Observe that $t \geq 2\mu$ and that $0 \leq t < 1 - \mu$. By Hoeffding's inequality, the probability that $\overline{X}$ is at least $6hk/h_2 p$ is at most

$$\left[\left[\frac{\mu}{\mu + t}\right]^{\mu+t}\left[\frac{1-\mu}{1-\mu-t}\right]^{1-\mu-t}\right]^{p} \leq 3^{-tp}e^{tp} = e^{-\Omega(tp)}.$$

We conclude that high probability at most $6hk$ messages are destined for any target

group during Step 2. In this case the messages can be delivered in $O(h + \log\log p)$ steps using the method described in the proof of Lemma 4.

At the beginning of Step 3 each processor has at most h_3 messages where h_3 is h plus the number of time-steps used in Steps 1 and 2. Using a Chernoff bound (as in Step 1) we see that with probability at least $1 - p \times (p/k) \times e^{-hk/3}$ each row of each block is the destination of at most $2hk$ messages in Step 3. We now consider each particular block. Following Rao [Rao 92] we can use a Chernoff bound to show that with probability at least $1 - k^2 \exp(-h_3/3)$ the communication problem on each row is a $2h_3$ relation. Similarly, with high probability the communication problem on each column is a $2h_3$-relation. Therefore, the probability of failure can be made as small as k^{-3} . The processors in the block use parallel prefix to notify each other of failure. Similarly, the processors in each super-block use parallel prefix to notify each other of failure.

The implementation and analysis of Step 4 closely follows that of Section 2.4. The probability that there is a failed block that fails to allocate a super-block is at most $(p^2/k^2)(3/k)^k$. The probability that there is a super-block in which every block fails when it repeats Step 3 is at most $(p^2/k^3)(1/k^3)^k$.

If Steps 1 through 4 are successful then at the start of Step 5 all of the messages will be in the correct row. Furthermore, there will be at most h_5 messages at any processor, where h_5 is h plus the number of time-steps used in steps 1–4. Since the communication problem is an h-relation, each processor will be the destination of at most h messages. Therefore the p-OCPC algorithm described in Section 2 can be used to deliver the messages on each row. The probability that this algorithm fails is at most p (the number of rows) multiplied by the probability that the p-OCPC algorithm fails, which is at most $p^{-\alpha}$ for any positive constant α.

In the introduction to this paper we pointed out that the MOB-PC is easier to build than an OCPC because it restricts the number of processors that a given processor can send to directly. Nevertheless, we have provided an algorithm for realizing h-relations on a MOB-PC which is asymptotically as fast as the fastest known algorithm for realizing h-relations on an OCPC. Similarly, we could define a new machine by replacing each row and each column of a $p \times p$ MOB-PC with a $p^{1/2} \times p^{1/2}$ MOB-PC. Our algorithm could be used recursively to realize h-relations in $O(h + \log\log p)$ steps on the new machine. Clearly, this recursion could be carried out to any constant depth.

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 88] B. Bollobás, Martingales, Isoperimetric Inequalities and Random Graphs, in *Combinatorics* (eds A. Hajnal, L. Lovász, and V. T. Sós), *Colloq. Math. Soc. János Bolyai* **52** (North Holland 1988) 113–139.

[CDHR 89] B. S. Chlebus, K. Diks, T. Hagerup, and T. Radzik, New Simulations between CRCW PRAMs, *Proc. Foundations of Computation Theory* **7** , Lecture Notes in Computer Science **380** (Springer-Verlag 1989) 95–104.

[Col 88] R. Cole, Parallel Merge Sort, *SIAM Journal of Computing* **17(4)** (1988) 770–785.

[Dow 91] P. W. Dowd, High Performance Interprocessor Communication Through Optical Wavelength Division Multiple Access Channels, *Proceedings of the ACM International Symposium on Computer Architecture* **18** (1991) 96–105.

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

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

[GM 91] J. Gil and Y. Matias, Fast Hashing on a PRAM, *Proceedings of the ACM-SIAM Symposium On Discrete Algorithms* **2** (1991) 271–280.

[MV 91] Y. Matias and U. Vishkin, Converting High Probability into Nearly-Constant Time — with Applications to Parallel Hashing, *Proceedings of the ACM Symposium On Theory of Computing* **23** (1991) 307–316.

[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.

[MP 75] D. E. Muller and F. P. Preparata, Bounds to Complexities of Networks for Sorting and for Switching, *Journal of the ACM* **22** (1975) 195–201.

[Rao 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.

[Val 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)

[Wit 81] L. D. Wittie, Communication Structures for Large Networks of Microcomputers, *IEEE Transactions on Computers* **C-30(4)** (1981) 264–273.