Scheduling Parallel Communication: The h-relation Problem

Micah Adler *
Computer Science Division
UC Berkeley
Berkeley, CA 94720

John W. Byers †
Computer Science Division
UC Berkeley
Berkeley, CA 94720

Richard M. Karp ‡
International Computer Science Institute
and
Computer Science Division
UC Berkeley
Berkeley, CA 94720

June 5, 1995

Abstract

This paper is concerned with the efficient scheduling and routing of point-to-point messages in a distributed computing system with n processors. We examine the h-relation problem, a routing problem where each processor has at most h messages to send and at most h messages to receive. Communication is carried out in rounds. Direct communication is possible from any processor to any other, and in each round a processor can send one message and receive one message. The off-line version of the problem arises when every processor knows the source and destination of every message. In this case the messages can be routed in at most h rounds. More interesting, and more typical, is the on-line version, in which each processor has knowledge only of h and of the destinations of those messages which it must send. The on-line version of the problem is the focus of this paper.

The difficulty of the h-relation problem stems from *message conflicts*, in which two or more messages are sent to the same processor in a given round, but at most one can be received. The problem has been well studied in the OCPC optical network model, but not for other contemporary network architectures which resolve message conflicts using other techniques. In this paper, we study the h-relation problem under alternative models of conflict resolution, most notably a FIFO queue discipline motivated by wormhole routing and an arbitrary write discipline motivated by packet-switching networks. In each model the problem can be solved by a randomized algorithm in an expected number of rounds of the form $ch+o(h) + \log^{\Theta(1)} n$, and we focus on obtaining the smallest possible asymptotic constant factor c. We first present a lower bound, proving that a constant factor of 1 is not achievable in general. We then present a randomized algorithm for each discipline and show that they achieve small constant factors.

1 Introduction

We assume that a communication task to be performed by a parallel computer with n processors is specified by an $n \times n$ matrix $K = (k_{ij})$, where k_{ij} gives the number of messages originating at processor i and destined for processor j. If we let h be the maximum sum of any row or column of this matrix; then the matrix specifies

*Supported by a Schlumberger Foundation Graduate Fellowship.
†Supported by a GAANN Graduate Fellowship.
‡Supported by NSF grant number CCR-9005448

an h-relation [Val 90]. The problem of solving this communication task is called the *h-relation problem.* An h-relation can be thought of as a generic model of irregular communication. Routing an h-relation is the fundamental communication step in Valiant's BSP model of parallel computation. The on-line version of the problem is also central to the simulation of a PRAM with many processors on a PRAM with fewer processors, and, in general, to the scheduling of concurrent memory accesses in distributed memory machines.

To define the problem further, we must model the performance characteristics of the communication medium, or network, through which the messages are transmitted. Much of the existing work on the h-relation problem [AM 88], [Val1 90], [GJL+ 93] has been under the Optically Connected Parallel Computer (OCPC) model. In this model of computation, processors communicate by point-to-point messages in synchronous rounds, with the restriction that whenever two or more messages are sent concurrently to the same processor, the messages are destroyed and must be retransmitted. We provide a survey of the theoretical literature on routing h-relations under the OCPC discipline in Section 3.

In this paper, we consider the problem of routing h-relations in other communication media. It is assumed that any processor can communicate directly with any other processor. Communication occurs in synchronous rounds, during which each processor may send and receive one point-to-point message. The models are differentiated by the manner in which they handle *contention*, which occurs when several messages are sent concurrently to the same processor. We consider several disciplines for the handling of contention:

- The *FIFO discipline*, in which incoming messages destined for a given receiver are placed in a first-in first-out queue from which they are extracted by the receiving processor at the rate of one message per round. A sender is temporarily blocked after a transmission, unable to transmit more messages until the original message is extracted from the target processor's queue.
- The *arbitrary write discipline*, in which a receiving processor receives exactly one of its incoming messages at random at each time step; all other incoming messages must be retransmitted. Immediately after each round, every sender is informed as to whether its message has been received.
- The *priority queue discipline* in which incoming messages carry priorities, and are placed in a priority queue. As in the FIFO model, no two messages from the same processor may reside in queues at the same time.

In existing network architectures, message contention is primarily due to limited buffering at internal switches. The arbitrary write discipline is motivated by packet-switched architectures such as the BBN butterfly and the ATM network, where messages traverse the network as a single unit, with their final destination prepended to the message. When packet-switched messages simultaneously arrive at a switch with insufficient buffering, the switch may drop messages it cannot handle. The sender's ability to detect the successful or unsuccessful transmission of a message is often left to higher-level software, rather than the hardware itself. Modeling a packet-switched network is made more challenging by the wide variety of network topologies and internal switches used in practice. However, our arbitrary write discipline captures the essence of the conflict resolution strategy used in packet-switched networks. The contention rule in the arbitrary write discipline is similar to the contention rule of an arbitrary write CRCW PRAM.

The queued models are based on another approach used for routing messages in tightly coupled networks. These use the closely related techniques of wormhole routing and virtual cut-through routing, used for example in the J Machine [ND 90] and the CM-5 [Lei+ 94]. In wormhole routing, messages are divided into very small flow control units (flits) which are then transmitted along a fixed path through the network to the destination. A message can thereby occupy buffer space in several adjacent switches simultaneously. If the first flit of a message arrives at a switch with insufficient buffer space, the entire chain of flits stalls, often stalling the sending processor as well. This style of message-passing motivates the FIFO discipline, which stalls the sending processor until the target processor receives each transmission. Contention in the FIFO model is similar to the capacity constraint of the LogP model [CKP+ 93] and to the asynchronous version of the QRQW PRAM [GMR2 94].

To provide a preliminary comparison of the disciplines, we note that OCPC algorithms can be simulated with no slowdown in the arbitrary write model and FIFO algorithms can be simulated with no slowdown in the priority queue model. The disciplines using queues seem to be incomparable to the disciplines without queues because of the effect of stalling; in the queued disciplines, sending processors must wait until their transmission is successfully sent before proceeding.

We here concentrate our efforts on achieving the best possible leading constant for *direct* algorithms, that is algorithms where each message is routed directly from its source to its destination, without passing through any intermediate processors. Aside from their simplicity, direct algorithms require a smaller total number of successfully transmitted messages than indirect algorithms. Also, when the destinations of the messages are memory modules, messages cannot be forwarded, and direct algorithms are required.

When introducing the OCPC model, Anderson and Miller showed that in the off-line case, an arbitrary h-relation could be routed in time h [AM 88]. Recent efforts have tried to achieve *1-optimal* randomized on-line algorithms [RSTG 95]; i.e., on-line algorithms that route an h-relation in expected time $h + o(h)$. We here provide evidence that no 1-optimal algorithm exists in the FIFO discipline, the arbitrary write discipline, or the OCPC discipline. More specifically, we show that for a natural class of algorithms, there exists an h-relation input for every $h < n^{1/3}$ such that every randomized on-line algorithm for these disciplines requires time at least $1.1h$ to route that input. The class of algorithms can be informally described as those algorithms such that, if $k_{ij} = k_{ir} > 0$, then the first transmission from i to j is as likely to precede as it is to follow the first transmission from i to r.

We can, however, achieve within a small constant factor of optimal for all three disciplines. We present the following.

- In the FIFO discipline, a direct randomized algorithm that runs in time $3.41h+o(h)+O(\log^3 n \log\log n)$ with high probability. The analysis of this algorithm is based on introducing martingales associated with the progress of the algorithm, showing that the martingales have bounded differences with high probability, and then applying Azuma's tail inequality for martingales with bounded differences.
- In the arbitrary write discipline, a direct randomized algorithm that runs in time $1.62h + o(h) + O(\log n \log\log n)$ with high probability.
- In the priority queue discipline, a simple and direct randomized algorithm that runs in time $(2e-1)h + o(h) + O(\log n)$ with high probability, where e denotes the base of the natural logarithm. The analysis of this algorithm is based on a delay sequence argument.

We also report on simulations in which the following constant factors are observed: $2.08h$ for the FIFO algorithm, $1.57h$ for the arbitrary write algorithm, and $1.85h$ for the priority queue algorithm.

Throughout the paper, we say that an event holds with high probability if, for some $c > 1$, the event holds with probability $\geq 1 - 1/n^c$.

2 Previous Work

In this section, we survey the body of work on routing h-relations, most of which has been done in the Optically Connected Parallel Computer (OCPC). In the OCPC, each of n processors can transmit a message to any of the other processors at each time step. If a processor receives two or more messages at a given time step, the data in those messages is lost and the messages must be retransmitted. Successful transmissions are acknowledged by the receiving processor, and failure to acknowledge implies that the data in the message was lost. This model was introduced by Anderson and Miller [AM 88].

Anderson and Miller were also the first to observe that if all processors have complete information about a given h-relation, there exists a schedule which routes the relation using exactly h communication steps. This observation follows by viewing the relation as a bipartite multigraph of maximum degree h, which is edge-colorable using h colors. Drawing a correspondence between colors and communication steps gives the communication pattern that can be routed in time exactly h, which is also an immediate lower bound.

2.1 Asymptotically Efficient Protocols

A primary theme in the literature on routing h-relations has been to provide routing protocols which are asymptotically efficient, especially for small values of h. Two types of algorithms are considered, *direct* algorithms, in which messages are always transmitted from the sender directly to the final receiver and *indirect* algorithms, in which senders may forward messages through intermediate locations en route to the final destination. The simplicity of direct algorithms makes them appealing from a practical standpoint, but they have theoretical limitations, which will be described momentarily. In all the algorithms we describe, the use of randomness is fundamental; to our knowledge, the best upper bound for deterministically routing an arbitrary h-relation on an OCPC is $\mathrm{O}(h \log n)$.

The direct algorithm with the best asymptotic running time for the OCPC is the simple randomized algorithm of Gereb-Graus and Tsantilas [GT 92]. For any fixed $\epsilon > 0$, their algorithm with high probability transmits an arbitrary h-relation in time

$$\frac{e}{1-\epsilon}h + \Theta(\sqrt{h \log n} + \log n \log\log n).$$

This is done with $\log_{1/(1-\epsilon)} h = \frac{\log h}{-\log(1-\epsilon)}$ phases, where the problem remaining at the start of phase i is w.h.p. a $k_i = (1-\epsilon)^i h$-relation, and at the end of phase i is a $k_{i+1} = (1-\epsilon)^{i+1}h$-relation. Each phase transmits some ϵk_i-relation w.h.p. as follows. At each step, every processor uniformly at random picks one of its $r \leq k_i$ packets to send, and attempts to transmit that packet with probability $\frac{r}{k_i}$. With probability $\frac{k_i - r}{k_i}$, the processor does nothing. Using the fact that each processor successfully transmits a packet that has been sent with probability at least $\frac{1-\epsilon}{e}$, they show that the number of steps round i requires is no more than

$$\frac{e}{1-\epsilon}(\epsilon k_i + O(\sqrt{\epsilon k_i \log n} + \log n).$$

The bound on the total amount of time follows.

In [GJLR 92], Goldberg, Jerrum, Leighton and Rao prove a lower bound of $\Omega(\log n)$ for direct algorithms for the h-relation problem on the OCPC. The lower bound motivates study of indirect algorithms as a means to achieve sub-logarithmic time for routing h-relations for small values of h. The bound is a formalization of the following intuition. Consider a 2-relation in which $2n$ senders each transmit a single message to n receivers and where the senders do not know the whereabouts or the sending strategy of their partner. Random direct sending strategies are likely to lead to a pair which conflicts for $\Omega(\log n)$ rounds, thereby acheiving the lower bound. The same authors then present an indirect protocol for routing h-relations on the OCPC which we sketch below that runs in time $\mathrm{O}(h + \log\log n)$.

In the first phase of their protocol, the h-relation is thinned by a direct algorithm which routes many of the messages to their final destinations, using a procedure somewhat similar to that of [AM 88]. With high probability, the procedure transmits all but $\mathrm{O}(n/h \log\log n)$ messages in time $\mathrm{O}(h + \log h \log\log\log n)$. The protocol runs for $\mathrm{O}(\log h)$ rounds, round i of which reduces an $\frac{h}{2^{i-1}}$ relation to an $\frac{h}{2^i}$ relation across all but a small fraction of the processors who drop out of the remainder of this phase of the protocol if they are left with too many undelivered messages. During round i, for each of $\mathrm{O}(h/2^{i-1} + \log h + \log\log\log n)$ steps, each sender which has survived transmits each of its messages with probability $\frac{2^{i-1}}{h}$. The analysis uses the method of bounded differences to derive the bound on the number of undelivered messages.

Next, in the second phase of the protocol, the undelivered messages are smoothly redistributed across all the processors, so that each processor has at most one undelivered message. The final two phases employ the concept of target groups, $\frac{n}{k}$ disjoint sets of processors of size $k = \log^{c_1} n$. In phase three, all messages are delivered to the target groups of the destination processor, but perhaps not to the destination processor itself. The technique employed is dart-throwing, where those messages which cannot be routed directly to the target group with high probability are copied to many processors, each of which then attempts to transmit the message to the target group. Messages for which more than one copy arrive at the target group select a leader and eliminate duplicates. With high probability, each of the messages has now been delivered at its target group, and furthermore, every processor has at most two messages. In the final phase, running a deterministic procedure such as Valiant's algorithm [Val2 90] (a simple indirect protocol outlined below) within each target group sorts and routes each of the messages to their final destinations in time $O(\log k) = O(\log\log n)$.

Prior to the [GJLR93] result, the algorithm that achieved optimality for the smallest value of h was that of Valiant, an indirect algorithm which routes an h-relation in time $O(h + \log n)$, for any $h = \Omega(\log n)$. The algorithm consists of two steps, the first of which is a thinning step, which insures that within $O(h)$ rounds, the total number of unsent messages is $O(\frac{hn}{\log n})$ w.h.p. The remaining messages are sent by a deterministic subroutine requiring time $O(h + \log n)$. This subroutine proceeds by first performing a parallel prefix with all the processors that informs the processors of the total number of messages, along with the rank of the processor's messages within that total. This allows the processors to redistribute the messages so that every processor has exactly the same number of messages. Once this has been accomplished, the messages can be sorted by various means, such as Cole's parallel merge sort [Col 88], in time $O(h + \log n)$. The keys to be sorted consist of the identifier of the destination processor concatenated with the message itself, and thus all messages destined for the same processor will end up in consecutive processors. Thus, the remaining messages can be deterministically routed to their final destinations in time $O(h)$.

Finally, it is worth remarking that even for indirect algorithms, time bounds of the form $O(h)$ are not possible for small values of h. Goldberg, Jerrum and MacKenzie [GJM 94] prove a lower bound of $\Omega(h + \sqrt{\log\log n})$ for realizing an h-relation on the OCPC.

2.2 1-Optimal Protocols

For much larger values of h, recent research has focused not on asymptotic behavior, but on leading constant factors of asymptotically optimal algorithms. Gerbessiotis and Valiant [GV 94] define a 1-optimal protocol for the OCPC as a protocol which can route a random h-relation in time at most $(1 + o(1))h$ with high probability. To derive 1-optimal algorithms, they employ the total-exchange communication primitive (also known as all-to-all personalized communication), in which every processor has a distinct message to transmit to every other processor. Extremely simple and contention-free strategies solve this communication problem in time n on the OCPC. Using these tools, the same authors present a protocol to route random h-relations using at most $\frac{h}{n}(1 + o(1) + O(\log\log n))$ total-exchange rounds with high probability, and so this algorithm is 1-optimal. Recent work [RSTG 95] has improved this bound to $\frac{h}{n}(1 + o(1) + O(\log^* n))$ total-exchange rounds and has provided experimental results which confirm that the algorithm is very fast in practice. A drawback of this approach is that h must be extremely large for the algorithms to achieve 1-optimality, i.e. at least $\omega(n)$.

3 Direct 1-Optimal Algorithms Are Not Possible

In this section, we prove a lower bound for direct, on-line algorithms that are *uniform*, or, informally, algorithms where, at a given sender, a target destination is treated identically to other target destination receiving the same number of messages until the first attempt to send. To be more formal, let an *untried* destination for a processor be a destination to which that processor has not yet attempted to transmit a message. Then, we say that an algorithm is *uniform* if, at every time step, the probability that a processor sends to a given untried destination is the same for all untried destinations receiving the same number of messages. Restricting our attention to the class of uniform algorithms is justified by the fact that all untried destinations receiving the same number of messages look equivalent to a sending processor routing an arbitrary h-relation.

In addition, we assume that each processor is using the same program, and if sender A attempts the first transmission of a message to receiver X at the same time t as at least one other sender, then the probability that the message from A is successfully received at t is $\leq \frac{1}{2}$. If it is not successfully received, it will take at least one more time step for A to successfully send that message. We note that the last assumption holds for

$$
\begin{bmatrix}
1&&&&1&&&&1&&&&1&&&&1 \\
11111 \\
1&1111 \\
1&&1111 \\
1&&&1111 \\
&&&&11111 \\
&&&&1&1111 \\
&&&&1&&1111 \\
&&&&1&&&1111 \\
&&&&&&&&11111 \\
&&&&&&&&1&1111 \\
&&&&&&&&1&&1111 \\
&&&&&&&&1&&&1111 \\
&&&&&&&&&&&&11111 \\
&&&&&&&&&&&&1&1111 \\
&&&&&&&&&&&&1&&1111 \\
&&&&&&&&&&&&1&&&1111 \\
&&&&&&&&&&&&&&&&11111 \\
&&&&&&&&&&&&&&&&1&1111 \\
&&&&&&&&&&&&&&&&1&&1111 \\
&&&&&&&&&&&&&&&&1&&&1111
\end{bmatrix}
$$

Figure 1: A difficult matrix

the concurrent write, FIFO and OCPC disciplines, but not the priority queue discipline. We prove a lower bound of $1.1h$ for any uniform, direct, on-line algorithm for any model meeting these assumptions, provided that $h \leq n^{1/3}$. Thus, no 1-optimal algorithms exist which meet these criteria.

Theorem 1 *When $h \leq n^{1/3}$, there does not exist a uniform and direct on-line algorithm that routes every h-relation in time less than $1.1h$ with probability greater than $\frac{1}{2}$.*

We assume such an algorithm exists, and work towards a contradiction. Any such algorithm must successfully route h-relations of the form diagrammed below, which we will refer to as *difficult* h-relations. In such an h-relation, Processor 1 has h messages to send to distinct destination processors $j_1, \ldots j_h$. Each of these destinations receives an additional $h-1$ *conflicting* messages from distinct processors. The $h(h-1)$ processors which transmit conflicting messages are referred to as *blocking* processors. Each blocking processor transmits h messages in all, and all non-conflicting messages which they transmit have distinct destinations.

We call a time slot *crowded* if all blocking processors attempt the first transmission of a conflicting message during that time slot with probability at least $\frac{1}{2h} - \frac{1}{h^2}$.

Lemma 1 *In any uniform algorithm that routes an h-relation in time $1.1h$ with probability at least $\frac{1}{2}$, there are at least $0.9h$ crowded time slots in the $1.1h$ possible time slots.*

Proof: Since the algorithm is uniform and all processors execute the same code, it suffices to show that the lemma holds for any given blocking processor. To see that this is the case, consider the behavior of an *isolated* processor, i.e. a processor that sends h messages to distinct destinations, none of which conflict with any other messages. Until the time of the first attempted transmission of a conflicting message, the behavior of each blocking processor is identical to that of the isolated processor. For all of n/h isolated processors to succeed with probability $1/2$, each of the isolated processors must succeed with probability $(1/2)^{h/n}$, which is at least $1 - \frac{1}{h^2}$ for h sufficiently large. To complete in time $1.1h$ an isolated processor must choose h time slots to send its messages. If more than $0.2h$ slots are selected with probability $< \frac{1}{2}$, the number of slots utilized is strictly less than $.2h * .5 + .8h * 1 = h$. Thus, to complete in time $1.1h$, there must exist at least $0.9h$ slots that the isolated processor chooses with probability at least $\frac{1}{2}$. Furthermore, each of the $0.9h$ slots has a $\frac{1}{h}$ chance of being the slot chosen for a blocking processor's conflicting message, so the probability that a given conflicting message is scheduled in a crowded time slot is $\geq \frac{1}{2h} - \frac{1}{h^2}$.

Lemma 2 *The probability that processor 1 successfully sends a message on the first attempt, given that the attempt is during a crowded time slot, is no more than* $\frac{1+\exp(-\frac{h-3}{2h})}{2}$.

Proof: There are $h-1$ blocking processors sending conflicting messages coinciding with each message from processor 1. Thus, the probability that processor 1 sends during a time slot where no blocking processor attempts its first transmission is

$$\leq \left(1 - \frac{1}{2h} + \frac{1}{h^2}\right)^{h-1} \leq \exp\left(-\frac{h-3}{2h}\right)$$

During the other time slots, the probability that processor 1 sends successfully is no more than $\frac{1}{2}$. Thus, the total probability that processor 1 sends successfully is

$$\leq \exp\left(-\frac{h-3}{2h}\right) + \frac{1}{2}\left(1 - \exp\left(-\frac{h-3}{2h}\right)\right) = \frac{1}{2}\left(1 + \exp\left(-\frac{h-3}{2h}\right)\right).$$

Proof of Theorem 1 : To complete in time $1.1h$, processor 1 must attempt at least once to send each of its h messages during the first $1.1h$ time slots. But, by Lemma 1, at most $0.2h$ of these slots are not crowded, so processor 1 would have to attempt to send during at least $0.8h$ crowded time slots. Using lemma 2, it is a simple exercise in Chernoff bounds to show that with high probability, at least $0.11h$ of the messages from processor 1 are not transmitted successfully on the first try. Thus, processor 1 needs total time at least $1.11h$, and we have reached a contradiction.

4 Scheduling h-relations under the FIFO discipline

We now turn our attention to the FIFO discipline. In this discipline, transmission of the h-relation is divided into synchronous rounds, where each processor is allowed to send and receive one message during each round. Messages in transit to a given target processor are viewed as residing in a First-In First-Out queue. In each round a processor receives the message, if any, at the head of its queue, and that message is deleted from the queue. Network capacity constraints are enforced by ensuring that at most one message is in transit from any given processor at any time. Thus, as long as a message from a sending processor resides in the input queue of some processor, the sending processor is stalled and cannot send further messages, but it can receive messages. As described in the introduction, the FIFO discipline models contemporary machines that use wormhole routing or virtual cut-through routing to transmit messages.

Theorem 2 *There exists a direct randomized algorithm which routes an arbitrary h-relation in the FIFO discipline within time* $3.41h + o(h) +$
$O(log^3 n \log\log n)$ *w.h.p.*

We begin by describing a generic parameterized algorithm for the FIFO discipline with parameters k and μ. The algorithm runs in $m = log_{\frac{1}{\mu}} h$ stages, where with high probability, the following invariant is maintained: at the end of each stage $i, 1 \leq i \leq m$, the undelivered messages form an h_i relation, where $h_i = h\mu^i$. Stage i consists of kh_{i-1} rounds of communication scheduled in a manner described below.

Before stage i begins, each processor schedules the sending times of all of its undelivered messages. These sending times are chosen by sampling without replacement from the uniform distribution over the integers between 1 and kh_{i-1}. Thus the sending time of any given message is uniformly distributed over $[1..khi-1]$, and no two messages from the same processor have the same sending time. During stage i, each processor sends its messages according to their scheduled sending times. However, if a processor is stalled at the scheduled sending time of one of its messages, then the message is not sent, but is deferred to the next stage. Thus, if a message sent at time t_1 by a processor ρ resides in the input queue of its target processor until time t_2, then all messages scheduled to be sent by ρ in the interval $[t_1+1..t_2]$ are deferred to the next stage, and ρ resumes its transmissions at time t_2+1. The algorithm halts at a stage i such that $h_i \leq h^{2/5}$, at which time all remaining messages can be transmitted deterministically in time $o(h)$. The algorithm described in Theorem 2 is an instantiation of the parameterized algorithm with $k = 2.5$ and $\mu = 0.267$. In the following discussion ϵ is a positive constant which may be chosen as small as desired.

Theorem 2 is a consequence of the following lemma, which we will prove momentarily.

Lemma 3 *In stage i, with probability $1 - 2n \cdot \exp(-\Theta(h_i^{\frac{1}{3}}))$:*

- *All processors with at least $0.267h_i$ unsent messages transmit at least a $(0.733-\epsilon)$ fraction of those messages.*
- *All processors with at least $0.267h_i$ unreceived messages receive at least a $(0.733-\epsilon)$ fraction of those messages.*

Proof of Theorem 2 : From Lemma 3, we see that during stage i, which runs in $2.5h_i$ rounds, we have reduced the problem size by a factor of $0.733-\epsilon$ w.h.p. We recurse until the resulting problem forms an $h^{2/5}$-relation, which takes time $2.5 \cdot \sum_{i=0}^{m} (0.267+\epsilon)^i\, h = \frac{3}{0.733-\epsilon}h$. Then, all messages are sent in any order, and since no message can be delayed by more than $h^{2/5}$ time slots, and no processor needs to send more than $h^{2/5}$ messages, the algorithm finishes in time $o(h)$. Since the failure probability during stage i is at most $2n \cdot exp(-\Theta(h_i^{0.267}))$, the m stages succeed with high probability when for all i, $h_i = \Omega(\log^3 n)$, or equivalently, $h = \Omega(\log^{15/2} n)$. When $h \in [\log^3 n \ldots \log^{15/2} n]$, we can run until $h_i = \Theta(\log^3 n)$, and then complete in time $O(\log^3 n \log\log n)$. Thus, the algorithm completes w.h.p. in time $3.41h + o(h) + O(\log^3 n \log\log n)$. □

In the proof of Lemma 3, we use the following terminology. We say that a schedule chosen for stage i of our protocol is *well-distributed* if no processor receives more than $0.9h_i^{\frac{1}{3}}$ messages in any $h_i^{\frac{1}{3}}$ consecutive rounds. We say that a receiving processor j is *idle* at time t if its input queue is empty during the tth round. Otherwise, we say that the processor is *active* at time t. If the schedule for stage i is well-distributed, no receiving processor is active for more than $h_i^{1/3}$ consecutive rounds in stage i.

The proof also uses the following Martingale Tail Inequality due to Azuma. Recall that a Martingale is a sequence of random variables Y_i, $i = 0, 1, \cdots, n$, such that, for all i, $E[Y_i | Y_0, Y_1, \ldots, Y_{i-1}] = Y_{i-1}$.

Theorem 3 (Azuma) *Let $\{Y_i\}$ be a Martingale, and let $a_1, a_2, \ldots, a_n$ be such that, for $i = 1, 2, \cdots, n$, $|Y_i - Y_{i-1}| \leq a_i$. Then the probability that $Y_n - Y_0 > \lambda\sqrt{a_1^2 + \cdots a_n^2}$ is less than or equal to $e^{-\frac{\lambda^2}{2}}$.*

The application of this theorem is often called the method of bounded differences.

Proof of Lemma 3: We begin by deriving an upper bound of $\frac{2}{3}$ on the a priori expected time a processor stalls during the transmission of any message. We then apply the method of bounded differences to show that the schedules chosen in stage i (which are well-behaved w.h.p.) cause a processor which schedules x messages to experience a delay of at most $\frac{2x}{3}$ rounds with high probability. A final application of the method of bounded differences reveals that with high probability in stage i, each sender which begins with more than h_i undelivered messages successfully transmits at least $0.733h_i$ of these messages with high probability. A similar statement holds for the receivers and the lemma then follows.

Let x messages to be sent to processor j be partitioned into m sets, where set r contains n_r messages, and $\sum_{r=1}^{m} n_r = x$. The *arrival time* of a message is defined as the number of the round in which it is sent to processor j, or, equivalently, the number of the round in which it arrives at the input queue of processor j. The arrival times of the messages within set r are n_r slots chosen uniformly at random without replacement from the range $[1..S]$. The arrival times of the messages in each set are independent of the arrival times in the other sets. We call the distribution of arrival times associated with any partition an *arrival distribution.* We sometimes use the same notation to refer to both a partition and its arrival distribution.

A message is said to be stalled in a given round if, at the end of the round, it resides in the input queue of processor j. For any message p among the x messages, let $D(p)$, the *delay* of p, be the number of rounds in which message p is stalled.

Lemma 4 *For any message p and any arrival distribution for x messages including p, onto the range $[1 \ldots S]$, the expected value of $D(p)$ is less than or equal to $\frac{x}{S-x}$.*

Proof: We first consider the partition consisting of x singleton sets. In this case the arrival times are independent identically distributed random variables, each with the uniform distribution over $[1..S]$. We call this the *uniform and independent arrival distribution.* We say that message p is in the system for $D(p)+1$ rounds. Let $S(j)$ be the time spent in the system of message j. Let $Y(i)$ be the number of messages in the system at time i. By counting the number of message-time pairs, we have that $\sum_{j=1}^{x} S(j) \geq \sum_{i=1}^{S} Y(i)$. For p to be in the system for z rounds, it must be stalled for $z-1$ rounds, and so there must be at least $z-1$ other messages that are either already in the system at the time p arrives, or that have the same arrival time as p. Since the arrival time of p is uniform and independent of the other arrivals,

$$\mathrm{E}[S(p)] \leq 1 + \mathrm{E}[\frac{1}{S}\sum_{i=1}^{S} Y(i)] \leq 1 + \mathrm{E}[\frac{1}{S}\sum_{j=1}^{x} S(j)].$$

By symmetry, $\mathrm{E}[S(p)] \leq 1 + \frac{1}{S}x\mathrm{E}[S(p)]$. This gives us $\mathrm{E}[S(p)] \leq \frac{S}{S-x}$, or that $\mathrm{E}[D(p)] \leq \frac{x}{S-x}$.

To complete the proof we show that of all possible arrival distributions, the expected delay of each message is greatest in the case of the uniform and independent arrival distribution. We compare two partitions: M, which contains a singleton set $\{s\}$, and M_d, which is the same as M, except that s has been merged with some other set U. We show that the expected delay of every message in M is at least as large as the corresponding message in M_d. This shows that the expected delay of any message in any arrival partition is no greater than the expected delay of a message in the uniform and independent partition, since any arrival partition can be made by a series of such transformations, starting with the uniform and independent partition.

To see that the expected delay of any message in M is no greater than that of the corresponding message in M_d, we compare M_d with M_o, where M_o is the same as M with the additional constraint that the message s must be scheduled at the same time as a message in U. Since the arrival distribution M is just a weighted sum of M_o and M_d, showing that each message has smaller expected delay in M_d than in M_o shows that each message has smaller expected delay in M_d than in M.

We first assign a number to each of the x messages in M_o and in M_d, with the constraint that a message in M_o has the same number as its corresponding message in M_d. Also, s is assigned x in M_o, and the message in U that overlaps with s is assigned $x-1$. In M_d, x and $x-1$ are any two elements of U. We henceforth refer to messages by their assigned numbers.

Let $\mathrm{E}_o[i]$ and $\mathrm{E}_d[i]$ be the expected delay of message i when the schedule is defined by arrival distributions M_o and M_d respectively. We shall show that $\forall i$, $\mathrm{E}_o[i] \geq \mathrm{E}_d[i]$. This is proved in two parts.

Claim 1: $\mathrm{E}_o[x] \geq \mathrm{E}_d[x]$ and $\mathrm{E}_o[x-1] \geq \mathrm{E}_d[x-1]$.

Proof: Fix the location of messages $1 \ldots x-2$. Let Z be the expected delay of a single additional message arriving with these fixed messages. Then, $\mathrm{E}_o[x] = \mathrm{E}_o[x-1] = Z + \frac{1}{2}$, since the messages have an expected delay of Z due to the other messages, and $\frac{1}{2}$ due to each other. $\mathrm{E}_d[x] = \mathrm{E}_d[x-1] \leq Z + \frac{1}{2}$, since message x is scheduled after $x-1$ with probability $\frac{1}{2}$.

To analyze the delay of the other messages, we introduce a method of calculating the delay of a given message i, given a fixed schedule. Set a counter to 0. Start at the beginning of the scheduling period. Scan from left to right, one time slot at a time. For every time slot, increment the counter by the number of arrivals, and then decrement it by 1, never letting the counter decrease below 0. The value of this counter when we reach the time slot that i arrives in (before accounting for that time slot), plus $\frac{1}{2}$ times the number of other arrivals during time slot i, is the expected delay of i, with the given schedule.

We can do the same thing by maintaining a *delay list*, where at every time step, each arrival for that time step is placed on the list, after which the smallest numbered message is removed from the list. The number of messages in the delay list when we reach i will be the delay of the first message serviced of those arriving at the same time slot as i. Note that the messages in the delay list are not necessarily the messages that are waiting for service at the time. But, from this we see that if, for a schedule chosen from arrival distribution M_d, we define indicator variables I_{ij}, where $I_{ij} = 1$ if message i is in the delay list when it reaches message j, $\frac{1}{2}$ if message i arrives at the same time as j, and 0 otherwise, then we see that $\mathrm{E}_d[j] = \sum_{i=1}^{x} \mathrm{E}[I_{ij}]$. We can define the same indicator variable J_{ij} for arrival distribution M_o, which gives us $\mathrm{E}_o[j] = \sum_{i=1}^{x} \mathrm{E}[J_{ij}]$, from which we see that lemma 4 follows from the following claim.

Claim 2: $\forall i, 1 \leq i \leq x, \forall j 1 \leq i \leq x-2$, $\mathrm{E}[I_{ij}] \leq \mathrm{E}[J_{ij}]$

Proof: When $i \leq x-1$, then $\mathrm{E}[I_{ij}] = \mathrm{E}[J_{ij}]$ follows from the fact that the location of messages numbered higher than i have no effect on either I_{ij} or J_{ij}, and thus neither does the distribution of those messages. To see that $\mathrm{E}[I_{xj}] \leq \mathrm{E}[J_{xj}]$, fix the location of all messages but $x-1$. This schedule of messages $1 \ldots x-2, x$ is as likely in M_d as it is in M_o. The only effect on I_{xj} or J_{xj} that adding $x-1$ to the other messages can have is changing I_{xj} or J_{xj} from a 0 to a 1. But, if it changes I_{xj} from a 0 to a 1, then J_{xj} will also be changed to a 1, and thus $\mathrm{E}[I_{xj}] \leq \mathrm{E}[J_{xj}]$.

□

Given a bound on the expected delay incurred by a given message, we now let D_j denote the total delay incurred by sending processor j during stage i. Setting $x = h$ and $S = 2.5h$ in Lemma 2, we find that the expected delay of each message transmitted by processor j is at most 2/3, and thus $\mathrm{E}[D_j] \leq \frac{2h_i}{3}$. We now show that w.h.p. the actual delay D_j is unlikely to deviate significantly from its expectation. The following lemma assumes that the schedule for stage i is well-distributed; using a Chernoff bound, this can be verified to hold for all processors with high probability, assuming that $h = \Omega(\log^3 n)$.

Lemma 5 *Given that the schedule for stage i is well-distributed, $Pr[|D_j - E[D_j]| > \beta h_i] \leq exp(-\beta^2 h_i^{\frac{1}{3}})$.*

<u>Proof:</u> We define the Martingale A_t, where $A(t) = \mathrm{E}[D_j | S_1 \ldots S_t]$ and S_t represents the scheduled actions of all processors at time t. To apply Azuma's inequality, we must provide a bound on $|A_{t+1} - A_t|$, the increase in expected delay at processor j after seeing a single additional round of the algorithm. Since we are given that the schedule is well-distributed, every receiving processor is guaranteed to be idle some time before time $t + 1 + h_i^{1/3}$, and therefore $h_i^{1/3}$ serves as a bound on $|A_{t+1} - A_t|$. The lemma follows immediately by an application of Azuma's inequality. □

Lemma 6 *Given that $D_j \leq \frac{2}{3}h_i$, the a priori probability that a given message from processor j is successfully sent during stage i is at least $\frac{11}{15}$.*

<u>Proof:</u> When $D_j \leq \frac{2}{3}h_i$, at most $\frac{4}{11}$ of the scheduled sending slots for processor j are invalidated due to processor j being stalled while sending other messages. Therefore, each message has probability of at least 0.733 of being sent in stage i. □

We can now sketch the proof of Lemma 3 for a sending processor; the proof for a receiving processor is similar. Let j denote an arbitrary processor with $h' \geq (0.267 + \epsilon)h_i$ messages left to send at the start of stage i, and let R_j be the random variable that denotes how many of j's messages are actually sent during stage i. From Lemma 5, we see that when $h = \Omega(\log^3 n)$, then for each processor j, $D_j \leq \frac{2}{3}h_i$ with high probability, and so we may restrict attention to the case where this inequality holds. Thus, from the previous lemma, before we choose the schedule for stage i, $\mathrm{E}[R_j] \geq 0.733h'$. We then define the Martingale $B_t =$

$\mathrm{E}[R_j|S_1 \ldots S_t]$, where S_t again represents the actions which all processors take at time t. Since the schedule is well distributed with high probability, we have that $|B_{t+1} - B_t| \leq h^{1/3}$, so Azuma's inequality shows that with high probability, R_j does not deviate substantially from $E[R_j]$. Therefore, processor j transmits at least $(0.733 - \epsilon)\, h'$ of its messages with probability $\exp(\frac{-\epsilon^2}{4} h_i^{1/3})$ and the lemma follows. $\square$

5 Scheduling h-relations under the Arbitrary Write discipline

In this section we analyze algorithms employing the arbitrary write discipline, which is analogous to the concurrent-write PRAM discipline of the same name. Each processor is able to send and receive up to one message during each synchronous round. If more than one processor sends a message to the same destination processor in a single round, then that destination processor receives one of the messages chosen at random, while the other messages are lost. The arbitrary write discipline can be contrasted with the somewhat more pessimistic OCPC model, in which all conflicting messages are lost. As in the OCPC discipline, successful receipt of a message can be acknowledged in unit time; those processors which do not receive an acknowledgment can assume their messages were not transmitted successfully.

The algorithm we employ is similar to the direct algorithm for routing h-relations on the OCPC due to Geréb-Graus and Tsantilas [GT 92]. By continuously updating weighted transmission probabilities for messages from sender i to receiver j, we obtain an algorithm which routes h-relations in the arbitrary write model with a small leading constant. We work toward the proof of the following theorem, which gives us a bound on routing h-relations in the arbitrary write model of $1.62h$ + o(h) + $\mathrm{O}(\log n \log\log n)$ as compared with the OCPC protocol which runs in time eh+ $\mathrm{o}(h)$ + $\mathrm{O}(\log n \log\log n)$ achieved by [GT 92] in a less favorable model.

Theorem 4 *There exists a protocol which solves any instance of the h-relation problem under the arbitrary write discipline within time $1.62h + o(h) + O(\log n \log\log n)$ rounds, with high probability.*

The first phase of the protocol is a thinning procedure which runs in a sequence of m stages, where in stage i, the problem is reduced from an h_{i-1}-relation to an h_i-relation with high probability, where $h_i = (1-\beta)^i h$ for $i \leq m$ and $h_0 = h$. The parameter β is a positive real parameter chosen arbitrarily close to zero and the parameter m is chosen as the smallest integer such that $h(1-\beta)^m < h^{2/5}$. After each of the stages, the transmission probabilities at each of the senders are recomputed in a method that will be described momentarily. After m such stages, the second phase of the algorithm routes the remaining messages (of which there are at most $h^{2/5}$) in time $\mathrm{O}(h^{4/5})$ using an obvious direct deterministic algorithm which processes any t-relation in time t^2 under the arbitrary write discipline.

Stage k of the first phase of the protocol consists of $t_k = \frac{\alpha\beta(1+\beta)}{1-\beta}(h_k + \log n)$ rounds where α is the reciprocal of the quantity $4(1-e^{-1/2})^2)$, or approximately 1.62. With high probability, the following invariant assertion holds: no processor has more than h_{k-1} messages left to send or receive at the beginning of stage k. If we let d_{ij} denote the number of undelivered messages originating at processor i and destined for processor j at some instant during stage k, we have by the invariant that the row and column sums of the matrix (d_{ij}) are less than or equal to h_{k-1}. At each round of stage k, processor i chooses at most one message to send; the probability that it sends to processor j is $1 - \exp\left(-\frac{d_{ij}}{(1-a)^{k-1}h}\right)$. It is easily verified that these probabilities sum to at most 1. The following sequence of lemmas culminate in the main theorem.

Lemma 7 *The following holds at each round of stage k: if processor 1 sends to processor j, then the probability that its message gets delivered successfully is at least $\frac{1-e^{-\lambda_j}}{\lambda_j}$, where $\lambda_j = (1/h_{k-1}) \sum_{k=2}^{n} d_{kj}$.*

<u>Proof:</u> Let the random variable X be the number of sources other than 1 that send to j. Then the probability that the message from source 1 is selected for transmission, given that source 1 sends to destination j, is

$E[\frac{1}{1+X}]$, where E denotes expectation. It is easily seen that the distribution of X is stochastically larger than the Poisson distribution with rate λ_j. This follows from the fact that the random variable X becomes stochastically larger whenever a source i is split into two sources whose demand vectors sum to the original demand vector of source i. If we split each source until each remaining source has exactly one message to send, and then recursively split each of these, then as the number of times the sources are split approaches infinity, the distribution of X_i approaches a Poisson distribution. The lemma then follows from the fact that, when X has the Poisson distribution with mean λ_j, $E[\frac{1}{1+X}] = \frac{1-e^{-\lambda_j}}{\lambda_j}$. □

Lemma 8 *Consider some fixed processor ρ for which at some round during stage k, $\sum_{j=1}^{n} d_{\rho j} = V$ and $\sum_{i=1}^{n} d_{i\rho} = W$. Then during this round:*

- *Processor ρ successfully transmits a message with probability at least $\frac{V}{\alpha^2 h_{k-1}}$.*
- *Processor ρ successfully receives a message with probability at least $1 - e^{-W/h_k}$.*

Proof: The second claim in the lemma is immediate. To prove the first claim, let $F(x)$ be defined as $\frac{1-e^{-x}}{x}$. Then, by Lemma 5, the probability of processor ρ successfully transmitting a message is at least $\sum_{j=1}^{n} \mu_j F(\mu_j) F(\lambda_j)$ where $\mu_j = \frac{d_{\rho j}}{h_{k-1}}$ and λ_j is as defined in the above lemma. Since F is a log-convex function, the product $F(\mu_j)F(\lambda_j)$ is minimized, subject to $\mu_j \geq 0$, $\lambda_j \geq 0$ and $\mu_j + \lambda_j \leq 1$, when $\mu_j = \lambda_j = \frac{1}{2}$. The value of the product at the minimum is $\frac{1}{\alpha^2}$, and the result follows from the fact that $\sum_j \mu_j = \frac{V}{h_{k-1}}$. □

Lemma 9 *Given that the invariant assertion holds at the beginning of stage k, the probability that it fails to hold at the beginning of stage $k+1$ is at most $2n \exp\left(-\frac{(\beta^3 h_k)}{3}\right)$.*

Proof: (Sketch) By Lemma 2 we see that at each round within stage k at which a processor has at least h_k messages left to send, its probability of sending successfully is at least $\frac{1-\beta}{\alpha}$. Now by transmitting with this probability for $t_k = \frac{\alpha\beta(1+\beta)}{1-\beta}(h_k + \log n)$ rounds, by a Chernoff bound, each of the n processors achieves the goal of transmitting at least βh_k of these messages with failure probability only $\exp\left(-\frac{(\beta^3(h_k + \log n))}{3}\right)$ for $h_k = \omega(\log n)$. Likewise, since a processor with at least h_k unreceived messages in stage k receives a message at each round with probability at least $1 - \exp(-\frac{1}{1-\beta}) > \frac{1-\beta}{\alpha}$, the goal of receiving at least βh_k of these messages is also realized with even smaller failure probability in the same time bound, and the lemma follows from a union bound. □

Proof of Theorem:

By Lemma 8, the invariant assertion holds throughout the $O(\log h)$ stages of the first phase of the protocol with high probability. To compute the amount of time that the protocol requires, we first bound the total time measured in rounds used through stage j, where j is the largest integer such that $h_j > \log n$ by

$$\frac{\alpha\beta(1+\beta)}{1-\beta} \sum_{i=0}^{j} h_i < \left(\frac{1+\beta}{1-\beta}\right) \alpha h.$$

The remaining $O(\log\log n)$ stages, $h_{j+1}, \ldots, h_m$, each take $O(\log n)$ rounds, so the first phase of the protocol completes in $\frac{1+\beta}{1-\beta}\alpha h + O(\log n \log\log n)$ rounds. In the second and final phase of the protocol, we route the remaining $h^{2/5}$-relation in time $o(h)$. To accomplish this, note that a single undelivered message will be transmitted unsuccessfully at most $h^{2/5}$ times. Therefore, each processor simply transmits each of its at most $h^{2/5}$ messages until they all arrive successfully, in total time $h^{4/5}$. □

6 Scheduling h-relations under the Priority Queue discipline

The priority queue discipline is similar to the FIFO discipline in that messages wait in a queue until they are transmitted to their respective destinations, and the sending processor stalls until the message has been sent. The distinction between the two disciplines is that in the priority queue discipline, a sending processor assigns a priority to each sent message, and the message with the highest priority is the first to leave each queue, with any ties being broken arbitrarily. It is easy to see that a protocol for the FIFO discipline can be simulated in the priority queue discipline by assigning each message a priority equal to the time at which it was transmitted. Therefore, the FIFO protocol presented in section 4 can be easily modified to run in this discipline. However, the analysis of the following algorithm is very simple, and the additive term dependent on the number of processors is much smaller.

Under this discipline, we present an algorithm for scheduling an h-relation that yields a running time of $(2e-1)h + o(h) + \log n$. Each of the P processors assigns each message a priority r, chosen independently and uniformly from $[1 \dots R]$, R sufficiently large to guarantee that all priorities selected are distinct with high probability. Each processor then sends its messages in order of the chosen priorities, from highest to lowest.

Theorem 5 *An h-relation can be routed under the priority queue discipline in time $(2e-1)h + o(h) + \log n$ w.h.p.*

Proof: We begin by providing a simple proof for a bound of $(2e+\epsilon)h$ which uses the delay sequence argument of [Ran 88]. We focus attention on the last message to arrive at its destination, and retrace the sequence of delays which resulted in this message's delayed arrival. In the context of this problem, a delay sequence for a message p_0 with priority r_0 consists of a sequence of messages with increasing priorities all greater than r_0 in which the messages either have the same source as p_0 or the same destination as p_0.

We derive a bound on the probability that there exists a delay sequence of length t for any message. For any fixed message p_0, there are fewer than $2h$ messages with either the same source or the same destination, so each successive message in the delay sequence could be one of at most $2h$. Therefore, there are at most $(2h)^t$ possible delay sequences for each message, and at most hP messages, for a total of at most $hP(2h)^t$ possible delay sequences. Now since each possible delay sequence is actually a delay sequence only when the priorities are strictly increasing, this happens with probability $\frac{1}{t!}$. Using Stirling's approximation, we approximate the probability of any delay sequence actually occurring by

$$\frac{hP(2h)^t}{t!} \leq \frac{hP(2h)^t}{(\frac{t}{e})^t\sqrt{2\pi t}}$$

Thus, when $t > (2e+\epsilon)h = 5.44h$, we see that a delay sequence of length t does not exist w.h.p. By refining the technique used to count the number of delay sequences of a given length to give an improved bound, the leading constant can be improved to $2e-1$, as stated in Theorem 5. $\square$

7 Simulation Results and Conclusion

We have obtained some preliminary simulation results on the algorithms described and analyzed in the previous sections. Our empirical results focus on the 1-relation also known as all-to-all personalized communication, a special case of the h-relation in which each processor pair must exchange a single distinct message. Our analysis of the parameterized algorithm for the FIFO discipline predicted a minimum running time of $3.4h$ by setting $K = 1.9$. Our empirical results show that for this setting of K, the theoretical analysis closely predicts the observed running time. However, as K decreases below 1.9, our analysis does not model the empirical performance of the algorithm, whose running time continues to improve, reaching a minimum of $2.08h$ at $K = 1$. A comparison of theoretically predicted performance vs. empirically observed performance appears in Figure 2.

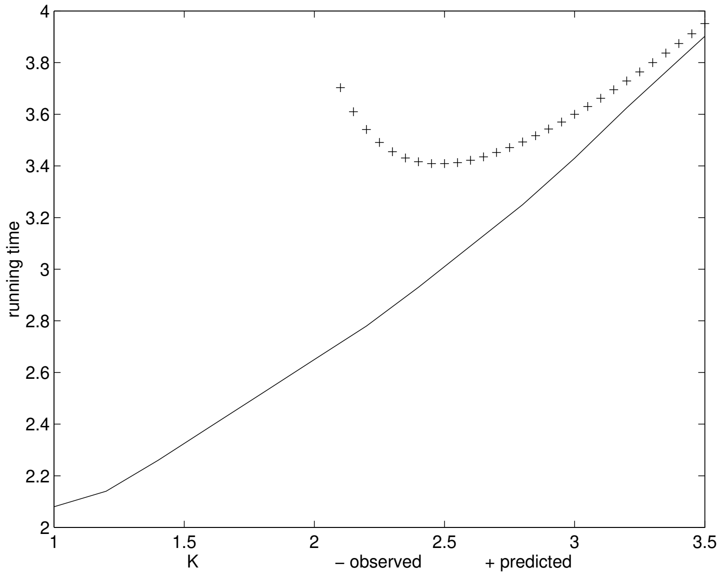

Figure 2: The dependence of running time on K.

For the arbitrary write discipline, we discovered that the theoretical bound of $1.62h$ closely predicts the empirically observed running time. Furthermore, we found that the probabilistic weighting scheme was not merely an artifact of the proof; we were not able to achieve running times better than $1.7h$ for any algorithm in the arbitrary write discipline which transmitted all unsent messages with equal probability. Finally, we found that the algorithm used in the priority queue discipline with priorities chosen uniformly at random worked extremely well in practice. Observed running times were approximately $1.85h$, a marked improvement over the running time of the algorithm for the FIFO discipline.

It remains an open problem to reduce the gap between the upper and lower bounds for all the disciplines, most notably the sizable gap remaining for the FIFO discipline. A related problem of interest would be to improve the analysis to reduce the additive polylogarithmic factors associated with the running times of algorithms in the FIFO discipline.

8 Acknowledgments

We would like to thank David Culler for a fruitful discussion of contemporary parallel architectures and network routing techniques.

References

[AM 88] R. A. 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, 1988.

[BHK+ 94] J. Bruck, C. Ho, S. Kipnis, D. Weathersby. "Efficient Algorithms for All-to-All Communications in Multi-Port Message-Passing Systems," *Proc. 6th ACM Symp. on Parallel Algorithms and Architectures*, pp. 298-309, June 1994.

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

[CKP+ 93] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. "LogP: Towards a Realistic Model of Parallel Computation," *Proc. 4th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming,* pp. 1-12, January 1993.

[GV 94] A. Gerbessiotis and L. Valiant. "Direct Bulk-Synchronous Parallel Algorithms," *Journal of Parallel and Distributed Computing 22,* pp. 251-267, 1994.

[GT 92] M. Geréb-Graus and T. Tsantilas. "Efficient Optical Communication in Parallel Computers," *Proc. 4th ACM Symp. on Parallel Algorithms and Architectures*, pp. 41-48, June 1992.

[GMR1 94] P. Gibbons, Y. Matias and V. Ramachandran. "The QRQW PRAM: Accounting for contention in parallel algorithms," *Proc. 5th ACM Symp. on Discrete Algorithms*, pp. 638-648, January 1994.

[GMR2 94] P. Gibbons, Y. Matias and V. Ramachandran. "Efficient Low-Contention Parallel Algorithms," *Proc. 6th ACM Symp. on Parallel Algorithms and Architectures,* pp. 236-247, June 1994.

[GJL+ 93] L. A. Goldberg, M. Jerrum, T. Leighton, and S. Rao. "A Doubly Logarithmic Communication Algorithm for the Completely Connected Optical Communication Parallel Computer," *Proc. 5th ACM Symp. on Parallel Algorithms and Architectures,* pp. 300-309, June-July 1993.

[GJM 94] L. A. Goldberg, M. Jerrum and P. MacKenzie. "An $\Omega(\sqrt{\log\log n})$ Lower Bound for Routing on Optical Networks," *Proc. 6th ACM Symp. on Parallel Algorithms and Architectures,* pp. 147-156, June 1994.

[Lei 92] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes* Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[Lei+ 94] C. E. Leiserson, et al. "The Network Architecture of the Connection Machine CM-5," *Proc. 4th ACM Symp. on Parallel Algorithms and Architectures,* June 1992. Revised March 21, 1994.

[ND 90] M. Noakes and W. J. Dally. "System Design of the J Machine," *Sixth MIT Conference on Advanced Research in VLSI,* pp. 179-194. MIT Press, 1990.

[Ran 88] A. Ranade. *Fluent Parallel Computation.* PhD thesis, Yale University, New Haven, CT, 1988.

[RSTG 95] S. Rao, T. Suel, T. Tsantilas and M. Goudreau. "Efficient Communication Using Total-Exchange," *Proc. 9th IEEE International Parallel Processing Symposium,* pp. 544-555, April 1995.

[Val1 90] L. G. Valiant. "General Purpose Parallel Architectures." In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Volume A,* pp. 943-972. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1990.

[Val2 90] L. G. Valiant. "A Bridging Model for Parallel Computation," *Communications of the ACM 33,* pp. 103-111, 1990.

[Wol 89] R. W. Wolff, *Stochastic Modeling and the Theory of Queues,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1989.