Fast Routing on Weak Hypercubes:

Making the Valiant-Brebner Algorithm More Practical*

Ben H.H. Juurlink† Jop F. Sibeyn‡

Abstract

Routing h-relations is considered on hypercubes consisting of N processing units (PUs). In the full-port model, the randomized algorithm by Valiant and Brebner, achieves this in $\mathcal{O}(h+\log N)$ routing steps, but requires $\Omega(h\cdot\log N)$ computation steps. This implies that the power of the full-port model cannot effectively be used for non-constant h. We present a strategy, which allows to reduce the number of computation steps required for performing an h-relation. If $h=\log^c N$, $c>1$ a constant, then the total routing time becomes $\mathcal{O}(h\cdot\log N/\log\log N+\log^2 N)$. For $h=N^\epsilon$, $\epsilon>0$ a constant, we achieve the optimal time order: $\mathcal{O}(h)$.

Here the results are stated only for hypercubes, but are actually formulated for all networks for which Valiant's paradigm gives an algorithm that requires less routing steps than computation steps. We also prove that under fairly general assumptions the obtained results are optimal to within a constant factor.

1 Introduction

Communication problems are of fundamental importance on parallel computers with an interconnection network. Among the many possible communication patterns, *h-relations*, in which each processing unit (*PU*) sends up to h packets and is due to receive at most h packets, are the most important. For example, they constitute the basic communication pattern in Valiant's BSP model of parallel computation [14]. 1-relations are called *permutations*.

A *hypercube* consisting of $N=2^n$ PUs is an n-dimensional cube with side length two. Every PU of a hypercube is connected to all its n immediate neighbors. Hypercubes are fundamental networks, which have been considered extensively in theory, and which are built in practice (Connection Machine CM-2, nCUBE).

Store-and-forward routing is one of the dominant routing paradigms. In this model, in every communication step, the PUs can send packets to adjacent PUs. Packets that are not send on immediately are stored in a queue.

In this paper we consider algorithms for store-and-forward routing of h-relations on hypercubes consisting of N PUs. This problem attracted substantial research effort before. A randomized algorithm by Valiant and Brebner [13, 16] performs permutation routing in $\mathcal{O}(\log N)$ steps, with high probability. It is based on the idea that good worst-case performance can be achieved by first sending the packets to random intermediate destinations before they are routed to their actual destinations. The algorithm was extended for routing h-relations in the optimal $\mathcal{O}(h+\log N)$ steps, with high probability [15]. This strategy is not limited to hypercubes, e.g., in [10] it is applied for routing h-relations on meshes.

One drawback of this algorithm is the assumed node model. It is assumed that (1) in one routing step a node can transmit a packet on all of its edges simultaneously, and (2) the time to perform the routing function is negligible compared to the transmission time of a packet. This is sometimes referred to as the *strong-node model* or *full-port model*. In the *one-port model* a node can send only one packet along one edge in each step, and receive one packet on one of its edges. In the one-port model an h-relation requires $\Omega(h\cdot\log N)$ steps on an N-node hypercube, since only N of the $N\cdot\log N$ edges can be used simultaneously. It is questionable whether the strong-node model is compatible with a small cycle time, especially for large values of N. If at all, this could only be achieved at high cost. Yet the one-port node model seems to be too restrictive, since it sacrifices much of the communication power of the network. For further information on node design and the trade-offs involved, see [5] and the references there.

Another model is the *weak-node model*. In this model the edges connected to a node can be simultaneously active, but a node can examine only one of its edges at each step.[1] This organization resembles, for example, a node architecture in which there is a single ALU that is used to perform the routing function and to set up the packet transfers, and each link has direct memory access to a buffer space. Other node architectures are conceivable as well. A possible processor architecture is given in Figure 1.

Routing on the weak-node model is more difficult than on the strong-node model, because when several packets arrive at the same time only one of them can be processed by the ALU, while the others must be delayed. In a hypercube

*Part of this research was performed during a visit of the first author to the Max-Planck-Institut für Informatik, Saarbrücken.

†High Performance Computing Division, Dept. of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands. Email: benj@cs.LeidenUniv.nl

‡Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken Germany. Email: jopsi@mpi-sb.mpg.de

[1] Some authors, e.g. in [11], refer to the one-port model as the weak hypercube model. We follow the terminology of [1].

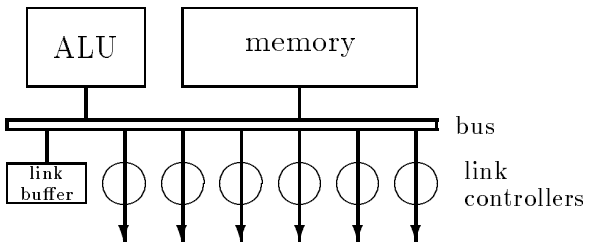

Figure 1: The architecture of a PU in the weak-node model.

of size N, up to $\log N$ packets can arrive simultaneously. In a direct implementation of the Valiant-Brebner algorithm, $\Omega(h \cdot \log N)$ computation steps need to be performed by each PU. This has led several researchers to conclude that the communication capabilities of the hypercube do not scale when the number of nodes is increased [6, 9]. The only way to remedy this problem is to pack several packets into one message so that the routing decisions can be overlapped with message transmissions. However, it is not clear at first sight how this must be done, since in general only few packets originating in the same node share their destinations.

The remainder of this paper is structured as follows. After basic facts from probability theory and our main routing hypothesis, we describe an algorithm for routing h-relations on the weak-node model. The algorithm achieves the same performance as on the strong-node model provided $h \geq N^\epsilon$ for some constant $\epsilon > 0$. For smaller h, the algorithm still beats the Valiant-Brebner algorithm on the weak-node model by more than a constant factor. In Section 5 we show that under light assumptions the obtained results are sharp. In Section 6, we use the routing results for the weak-node model in order to efficiently simulate PRAM models.

2 Chernoff Bounds

In the analysis of the randomization technique we use the following facts, known as Chernoff bounds [3]. These bounds provide estimates of the tail probabilities of binomial distributions.

Let X be the number of heads in n independent flips of a biased coin, the probability of a head in a single flip being p. Such an X has the binomial distribution $B(n,p)$. In [7] it is shown, that for all $0 < h < n \cdot p$

$$Pr\{X \geq n \cdot p + h\} \leq e^{-h^2/(3 \cdot n \cdot p)}, \quad (1)$$

$$Pr\{X \leq n \cdot p - h\} \leq e^{-h^2/(2 \cdot n \cdot p)}. \quad (2)$$

The results of this paper hold with 'high probability':

Definition 1 *An event A happens with* high probability, *if $Pr\{A\} \geq 1 - n^{-\alpha}$, for some constant $\alpha > 0$.*

Usually we need that a polynomial number of (not necessarily independent) binomial random variables, $X_1, \ldots, X_M$, $X_i = B(n,p)$ for all $1 \leq i \leq M$, are all bounded at the same time. Let h_1 be the h value, that is required to bound one of the X_i with high probability; and let h_M be the h value to bound all X_i at the same time. The definition of 'high probability' is such, that with (1) and (2) it is easy to show that $h_M = \mathcal{O}(h_1)$. The use of this fact is often left implicit. Formally, we have

Lemma 1 *If M is polynomial in n, then there is an $h_M = \mathcal{O}((p \cdot n \cdot \log n)^{1/2})$, such that $|X_i - p \cdot n| \leq h_M$, for all $1 \leq i \leq M$ simultaneously, with high probability.*

3 Main Routing Hypothesis

Our analysis is based on a natural hypothesis concerning the routing time for h-relations. It is inspired by the following lemma, which holds when applying the algorithm of Valiant and Brebner:

Lemma 2 [15] *For all $h > 0$, routing h-relations on a strong hypercube with N PUs, can be performed in $\mathcal{O}(h + \log N)$ routing steps, with high probability.*

We want to state our results as general as possible. Therefore, we consider a general network of which we specify some parameters. The number of PUs is denoted by N. The diameter of an N-node network is $D(N)$, and $T_r(h,N)$ denotes the time required for performing an h-relation. $t_r(m)$ is the time to transfer a message consisting of m packets from one PU to a neighbor. t_c is the time for performing a computation step. We assume that $t_r(m) \leq m \cdot t_r(1)$, and that $t_r(1) = \Theta(t_c)$.

Hypothesis 1

$$T_r(h,N) = \mathcal{O}((h + D(N)) \cdot t_r(1) + h \cdot D(N) \cdot t_c).$$

The term $(h + D(N)) \cdot t_r(1)$ accounts for the number of routing steps performed by the algorithm, and is derived from Lemma 2. The second term $h \cdot D(N) \cdot t_c$ is the number of computation steps performed by each PU, and is motivated as follows. Each time a packet is sent along an edge, its addressing information is inspected in order to determine the next output edge. Since there are $h \cdot N$ packets in total, each of which follows a path of length $\mathcal{O}(D(N))$, the number of computation steps performed by each PU is $\mathcal{O}(h \cdot D(N))$.

On many networks, Hypothesis 1 is satisfied with high probability. We shortly consider this for networks of logarithmic diameter and sufficiently large h. We slightly reformulate a result by Valiant [15]: there are constants α_0, β and γ, such that for all $\alpha \geq \alpha_0$,

$$Pr\{T_r(h,N) > \alpha \cdot h\} \leq N^\gamma \cdot e^{-\alpha \cdot \beta \cdot h}.$$

For $h \geq \delta \cdot \log N$, for sufficiently large constant δ, this probability is very small. For other networks the expression is

different, but still we may assume that randomized greedy routing according to the strategy of Valiant and Brebner finishes in the number of routing steps from Hypothesis 1 with overwhelming probability. The overall failure probability remains negligible even when many disjoint local routings are performed as a result of recursion (there can be at most N such local routings).

4 The Algorithm

In the remainder of the paper our analysis is based on Hypothesis 1. We do not further consider how the routing is performed. The word *message* will be reserved for a bundle of packets.

Large h. For $h = \omega(1)$ (non constant), the computation time, $\mathcal{O}(h \cdot D(N) \cdot t_c)$, exceeds the routing time, $\mathcal{O}((h + D(N)) \cdot t_r(1))$. But, for sufficiently large h, it is easy to achieve optimality:

Lemma 3 *If $h \geq D(N) \cdot N$, then h-relations can be routed in $\mathcal{O}(h + D^2)$ time.*

Proof: Fill the packets going from a PU to another PU into messages of size D. This takes $\mathcal{O}(h + N) = \mathcal{O}(h)$ local computation steps by using bucket sort. Then, N messages may remain partially empty; one for each destination. Thus, the routing of an h-relation is transformed to routing an $h/D+N = \mathcal{O}(h/D)$-relation of messages consisting of D packets each. Applying Hypothesis 1, we see that the time for this is bounded by

$$\mathcal{O}((h/D + D) \cdot t_r(D) + h/D \cdot D \cdot t_c) = \mathcal{O}(h + D^2). \quad \square$$

Moderate h. Now, suppose that $h < D(N) \cdot N$. The idea is to bundle packets, which have destinations in the same *block* consisting of S PUs, into messages of size m. The values of m and S are determined later.

The algorithm consists of recursive routing phases. The recursion continues until the block size is reduced to two. In each phase the following steps are performed:

1. In every PU, compose messages consisting of up to m packets going to the same block. Pack the messages as full as possible, leaving at most one partially filled message for every block.

2. Send the messages to random destinations in their blocks.

Call the complete algorithm BLOCK-ROUTE. We analyze its performance as a function of h, m and S.

Lemma 4 *If $h/m \geq \log N$, then for $S = N/(h/m)$, no PU sends or receives more than $\mathcal{O}(h/m)$ messages, with high probability.*

Proof: Within a PU, h/m messages can remain partially empty, one for each block. Thus, each PU sends at most $2 \cdot h/m$ messages. Also notice, that at most N partially filled messages can be destined for any target group, and therefore no group receives more than $h \cdot S/m + N = 2 \cdot N$ messages. So, the expected number of messages received by a PU is at most $2 \cdot h/m$. Applying Chernoff bounds, the actual number can be bounded to $2 \cdot h/m + \mathcal{O}((h/m \cdot \log N)^{1/2})$. $\square$

Lemma 4 is used in the proof of our main theorem:

Theorem 1 *If $h/m \geq \log N$, then* BLOCK-ROUTE *with $S = N/(h/m)$ and $m = D$ routes h-relations in*

$$T_r(h, N) = \mathcal{O}(\log N / \log(h/D) \cdot (h + D^2)). \qquad (3)$$

Proof: Assume for a moment that the communication pattern is completely balanced, i.e., each PU receives exactly h/m messages, holding exactly h packets in total. Then, applying Hypothesis 1, the running time of the sketched algorithm is given by the recurrence

$$\begin{aligned} T_r(h, N) &= T_r(h, N/(h/m)) + \mathcal{O}(h + m \cdot D + (h/m) \cdot D), \\ T_r(h, 2) &= \mathcal{O}(h). \end{aligned}$$

Working the recurrence out gives

$$T_r(h, N) = \mathcal{O}(\log N / \log(h/m) \cdot (h + m \cdot D + (h/m) \cdot D)).$$

Substituting $m = D$ gives the stated result.

Of course, the pattern is not perfectly regular. But, from Lemma 4 we know that under the given condition a PU receives at most $\mathcal{O}(h/m)$ messages, each consisting of at most m packets. It is important to notice that the factors do not multiply during the recursion. Consider the packets with destination in some block B of size $S_i = N/(h/m)^i$, to which the packets are routed in phase i. Independently of their distribution over the PUs of the enveloping block of size $S_{i-1} = N/(h/m)^{i-1}$, these packets are packed into at most $2 \cdot h/m$ messages and sent to random destinations in B. $\square$

We give three extremal cases:

Corollary 1 *If $h = D \cdot N^\epsilon$, $\epsilon > 0$ a constant, then h-relations can be routed in $\mathcal{O}(h + D^2)$ time. If $h = D \cdot 2^{\sqrt{\log N}}$, they require $\mathcal{O}(\sqrt{\log N} \cdot (h + D^2))$ time, and for $h = D \cdot \log^\epsilon N$, $\epsilon > 0$, $\mathcal{O}(\log N / \log\log N \cdot (h + D^2))$ time.*

If the network is recursively decomposable, then some improvement can be obtained if the number of packets which are packed into a message is reduced along with the reduction of the diameters of the blocks in which the routing is performed. This allows to reduce the size of the blocks more strongly, which results in less recursion steps. However, the gain is not substantial for networks with a small diameter like hypercubes.

5 Lower Bounds

It is extremely hard to prove lower bounds on the routing time of all possible algorithms. Therefore, we only consider 'naturally structured' algorithms:

Definition 2 *A routing algorithm is* naturally structured *if it operates in routing phases. Prior to each phase, messages are composed from the packets residing in the same PU. In each phase, all messages are routed within subnetworks of the same size.*

Basic Result. First suppose, that the PUs hold exactly h packets after each phase, that all messages consist of m packets, and that all routings are performed in the whole network. That is, for the time being, we assume that each phase costs as much as routing an h/m-relation. Denote the number of required phases by f. Hence, for a given m, according to Hypothesis 1

$$\begin{aligned} & T_r(h,N) \\ &= f(m)\cdot\Omega((h/m+D(N))\cdot t_r(m) \\ &\qquad + h/m\cdot D(N)\cdot t_c) \\ &= \Omega(f(m)\cdot(h+m\cdot D(N)+h/m\cdot D(N))). \end{aligned} \quad (4)$$

Lemma 5 *The number of distinct h-relations, $A(h,N)$, satisfies*

$$A(h,N)=\Theta((N^h/\sqrt{h})^N) > (N/2)^{h\cdot N}.$$

Proof: $A(h,N)=\prod_{i=1}^{N}\binom{h\cdot i}{h}=(h\cdot N)!/h!^N$. Applying Stirlings formula, $x!=\Theta((x/e)^x\cdot\sqrt{x})$, gives the result. □

By estimating the number of distinct h-relations that can be obtained in one phase, we get the following lower bound on the required number of routing phases f.

Lemma 6 *For a naturally structured algorithm for routing h-relations in which the messages consist of m packets in all phases and the PUs hold at most h packets,*

$$f \geq \frac{\log N - 1}{(\log N)/m + \log(h/m)}.$$

Proof: In every phase, for a message there are at most N possible destination PUs. This gives less than $N^{h\cdot N/m}$ recombination possibilities. Before every phase the messages in every PU are composed. This gives less than $(h/m)^{h\cdot N}$ recombination possibilities. Thus the number of distinct h-relations that can be realized in f phases is at most $(N^{h\cdot N/m}\cdot(h/m)^{h\cdot N})^f$. By Lemma 5, we require this number to be at least $(N/2)^{h\cdot N}$. Therefore, we get the following condition on f:

$$(N^{h\cdot N/m}\cdot(h/m)^{h\cdot N})^f \geq (N/2)^{h\cdot N},$$

and the results follows. □

It follows from (4) and Lemma 6 that if the routing is performed within the whole network in all phases, then

$$T_r(h,N)=\frac{\Omega(h+m\cdot D(N)+h/m\cdot D(N))}{(\log N)/m+\log(h/m)}\cdot\log N.$$

Minimizing over all m, we get

Theorem 2 *Assume that $h \geq D^2(N)$. For a naturally structured algorithm for routing h-relations in which the routing is performed with a fixed message size within the whole network in all phases and in which the PUs hold at most h packets,*

$$T_r(h,N)=\Omega(h\cdot\min\{D,\log N/\log h\}).$$

Proof: We must minimize

$$\frac{h\cdot D+m\cdot h+m^2\cdot D}{\log N+m\cdot\log(h/m)}\cdot\log N, \quad (5)$$

for $1\leq m\leq h$. For our estimate we may replace the factor $\log(h/m)$ by $\log h$. We consider several cases to simplify the computation. Under the assumption $h\geq D^2$, we can distinguish three cases for the upper part of (5):

1. $m\leq D$: $h\cdot D+m\cdot h+m^2\cdot D\simeq h\cdot D$.

2. $D\leq m\leq h/D$: $h\cdot D+m\cdot h+m^2\cdot D\simeq m\cdot h$.

3. $h/D\leq m$: $h\cdot D+m\cdot h+m^2\cdot D\simeq m^2\cdot D$.

For every case we distinguish two subcases

a. $m\leq\log N/\log h$: $\log N+m\cdot\log h\simeq\log N$.

b. $\log N/\log h\leq m$: $\log N+m\cdot\log h\simeq m\cdot\log h$.

We check all six combinations:

1.a. $h\cdot D$.

2.a. $m\cdot h\geq h\cdot D$.

3.a. $m^2\cdot D\geq h^2/D\geq h\cdot D$. Here we used that $h\geq D^2(N)$.

1.b. $h\cdot D\cdot\log N/(m\cdot\log h)\geq h\cdot\log N/\log h$.

2.b. $h\cdot\log N/\log h$.

3.b. $m\cdot D\cdot\log N/\log h\geq h\cdot\log N/\log h$. □

Comparing with (3), we see that our algorithm is close to optimal for most values of h. For networks of logarithmic diameter, the lower bound is $T_r(h,N)=\Omega(h\cdot\log N/\log h)$, while for $h\geq D^2$, (3) gives $T_r(h,N)=\mathcal{O}(h\cdot\log N/\log h)$. Also observe that for $D\leq\log N/\log h$, the lower bound becomes $T_r(h,N)=\Omega(h\cdot D)$, which is the same as the running time of a direct implementation of the Valiant-Brebner algorithm.

Stronger Results. In the remainder of this section we weaken the conditions imposed on the routing algorithm. Here we address the following points.

- Allow that after every phase the PUs hold more than h packets. At higher routing costs, this gives more possibilities to recombine the packets before a routing phase.
- Allow that the message size m_i in phase i, $1 \leq i \leq f$, is different per phase.

A further extension would be to allow that the packets are recombined much more often, getting away from the naturally-structuredness of the algorithm. Possibly, a message could be decomposed after it has been transferred over just a single connection. Of course this cannot be done with all packets after every step, as this would cause a tremendous computational burden.

In the remainder of this section we assume that the network has degree and diameter $\Theta(\log N)$. An example is, of course, the hypercube network. Another example is an $\mathcal{O}(\log N)$-dilated butterfly [1]. A *b-dilated* butterfly is derived from an ordinary butterfly by replacing each edge with a bundle of b edges. On this network, Hypothesis 1 is also satisfied with high probability.

Suppose that PU i contains $h_{i,j}$ packets prior to the start of phase j, and let m_j denote the message size during phase j. Then

Lemma 7 *There is a constant c, such that the number of distinct h-relations that can be produced in phase j is bounded by*

$$\frac{N^{h \cdot N/m_j}}{{m_j}^{h \cdot N}} \cdot (c \cdot h \cdot \log^2 N)^{h \cdot N}.$$

Proof: In PU i, $h_{i,j}/m_j$ messages are composed from the $h_{i,j}$ packets residing there. This gives less than $(h_{i,j}/m_j)^{h_{i,j}}$ recombination possibilities. Each of the $h_{i,j}/m_j$ messages can be sent to N destinations. This gives less than $N^{h_{i,j}/m_j}$ recombination possibilities. Thus the number of distinct h-relations that can be obtained in phase j is at most

$$\prod_{i=0}^{N} (h_{i,j}/m_j)^{h_{i,j}} \cdot N^{h_{i,j}/m_j},$$

subject to the constraints $h_{i,j} \geq 0$, $\sum_{i=0}^{N-1} h_{i,j} \leq h \cdot N$, and $h_{i,j} \leq c \cdot h \cdot \log^2 N$, for some constant c. The last condition must hold because we know that h-relations can be realized in time $\mathcal{O}(h \cdot \log N)$ and it takes at least $k/\log N$ steps to concentrate k packets into a node. Thus,

$$\prod_{i=0}^{N} (h_{i,j}/m_j)^{h_{i,j}} \cdot N^{h_{i,j}/m_j} \leq \frac{N^{h \cdot N/m_j}}{{m_j}^{h \cdot N}} \cdot (c \cdot h \cdot \log^2 N)^{h \cdot N}. \square$$

Lemma 8 *There are constants c_1, c_2 and c_3 such that the total number of distinct h-relations that can be achieved in phases $1, 2, \ldots, f$ can be estimated on*

$$\frac{N^{c_1 \cdot h \cdot N/\log(h/\log N)}}{(c_2 \cdot \log(h/\log N))^{h \cdot N \cdot f}} \cdot (c_3 \cdot h \cdot \log^2 N)^{h \cdot N \cdot f}.$$

Proof: Using Lemma 7, the total number of h-relations that can be produced in phases 1 through f is at most

$$\prod_{j=1}^{f} \left(\frac{N^{h \cdot N/m_j}}{{m_j}^{h \cdot N}} \cdot (c \cdot h \cdot \log^2 N)^{h \cdot N} \right) = \frac{N^{h \cdot N \cdot (1/m_1 + \cdots + 1/m_f)}}{(m_1 \cdot \ \cdots \ \cdot m_f)^{h \cdot N}} \cdot (c \cdot h \cdot \log^2 N)^{h \cdot N \cdot f}. \quad (6)$$

In order to estimate this number we must look at the running time of the algorithm. By assumption, the cost of phase j equals the routing time of an (h/m_j)-relation of messages consisting of m_j packets each. So, the sketched algorithm takes time

$$\Theta\left(h \cdot f + \log N \cdot \sum_{j=1}^{f} m_j + h \cdot \log N \cdot \sum_{j=1}^{f} 1/m_j \right).$$

Since by Theorem 1 we know how to route h-relations in $\mathcal{O}(h \cdot \log N/\log(h/\log N))$ time, the sketched approach will only result in less routing time if $\sum_{j=1}^{f} 1/m_j = \mathcal{O}(1/\log(h/\log N))$. This, in turn, implies that for all j, $1 \leq j \leq f$, $m_j = \Omega(\log(h/\log N))$. Substituting in (6) gives the result. $\square$

In [8] we also consider the possibility of routing in subnetworks and sending only partially filled messages. Taking these results for granted, we obtain a fairly general lower bound:

Theorem 3 *For all naturally structured algorithms for routing h-relations, $h \geq \log^2 N$, on an N-node network with degree and diameter $\Theta(\log N)$.*

$$T_r(h, N) = \Omega\left(h \cdot \log N/\log h\right).$$

Proof: We again require the number of distinct h-relations that can be obtained in phases $1, 2, \ldots, f$ to be at least $(N/2)^{h \cdot N}$. By using Lemma 8, we get the following lower bound on the number of required routing phases

$$f \geq \frac{\log N - 1 - c_1 \cdot \log N/\log(h/\log N)}{\log(c_3 \cdot h \cdot \log^2 N) - \log(c_2 \cdot \log(h/\log N))}.$$

Clearly, for $\log^2 N \leq h \leq N^{\epsilon}$ and sufficiently large N, the numerator is $\Theta(\log N)$ and the denominator is $\Theta(\log h)$. Since each phase takes time at least $\Omega(h)$, the theorem follows. $\square$

6 PRAM Simulations

Valiant's BSP [14] or XPRAM [15] model employs the routing results for the strong hypercube in order to efficiently simulate an exclusive-read exclusive-write (EREW)

PRAM. In these simulations, a shared memory access is implemented by sending a packet through the communication network between the processor accessing a variable and the processor storing that variable. In order to assure that the requested variables are equally spread among the memory modules, the shared memory is distributed according to a hash function selected randomly from a class of universal hash functions [2]. Let v denote the number of PRAM processors and let p denote the size of the strong hypercube. Then, with high probability, the v memory accesses correspond to an $\mathcal{O}(v/p)$-relation which, by Lemma 2, can be realized in time $\mathcal{O}(v/p)$ provided that $v \geq p \cdot \log p$. We now shortly discuss how these ideas apply to weak hypercubes. In fact, using Theorem 1, it is easy to establish

Theorem 4 *A p-processor weak hypercube can simulate a v-processor EREW PRAM in expected optimal time $\mathcal{O}(v/p)$ if $v \geq p^{1+\epsilon}$ for some constant $\epsilon > 0$.*

In case concurrent read and write operations are allowed, the simulation scheme is different. In that case, the requests are first *semisorted* in an array of length v such that requests to the same memory address will be adjacent in the array. Each processor will be given a segment of length v/p of the semisorted array. Then the first request of each segment is selected by a parallel prefix operation and the processor storing that request performs the access. After that, the variables read are duplicated to the other requests by another parallel prefix. Semisorting an array of length v on a p-processor strong hypercube can be done in expected time $\mathcal{O}(v/p \cdot (\log v)/\log(v/p)) = \mathcal{O}(v/p)$ if $v \geq p^{1+\epsilon}$ by using a stable sorting algorithm due to Reif [12] and by hashing the requests to an array of size $2 \cdot v^2$ [15]. Hence it can be shown that a p-processor strong hypercube can simulate a v-processor CRCW PRAM in expected optimal $\mathcal{O}(v/p)$ time if $v \geq p^{1+\epsilon}$ for some $\epsilon > 0$.

Lemma 9 *Let B be an array of size v of items from $\{0, 1, \ldots, M-1\}$. A p-processor weak hypercube can semisort B in expected optimal time $\mathcal{O}(v/p)$ if $v \geq p^{1+\epsilon}$, $\epsilon > 0$ a constant.*

Proof: Similar to [15]. The algorithm consists of a constant number of phases (expected). In each phase an $\mathcal{O}(v/p)$-relation needs to be realized, and two parallel prefix operations on an array of size v have to be performed. Since $v \geq p^{1+\epsilon}$, the routing can be done in time $\mathcal{O}(v/p)$ by Theorem 1. Also, it is not hard to show that the prefixes can be computed in $\mathcal{O}(v/p)$ steps if $v \geq p \cdot \log p$ by using recursive doubling, because in each step a processor needs to send or receive only one packet on one of its edges. □

The simulation result now follows.

Theorem 5 *A p-processor weak hypercube can simulate a v-processor CRCW PRAM in expected optimal time $\mathcal{O}(v/p)$ if $v \geq p^{1+\epsilon}$ for some constant $\epsilon > 0$.*

Thus, in order to efficiently simulate an EREW PRAM on a weak hypercube, the simulated program has to have more parallelism than is required for a strong hypercube. For CRCW PRAM simulation, the power (and extra complexity and cost) of the strong model is not needed.

7 Concluding Remarks

In this paper we have presented a routing strategy that reduces the number of computation steps required for performing an h-relation. For $h \geq N^{\epsilon}$, $\epsilon > 0$ a constant, the algorithm achieves the same performance as on the strong-node model. If $h = \log^c N$ for some constant $c > 1$, the total routing time becomes $\mathcal{O}(h \cdot \log N / \log\log N + \log^2 N)$. We also showed that under fairly general assumptions the obtained results cannot be improved by more than a constant factor.

Although we have described our results in the store-and-forward packet routing model, we believe that our techniques can also be used in the virtual cut-through or wormhole routing models. As observed by Culler et al. [4], these modern flow control techniques have made network latency largely independent of the interconnection structure. Instead, the message communication time in current parallel architectures is dominated by the send and receive overheads. This is the time it takes for a processor to inject (receive) a message into (from) the communication network. Our algorithm bundles packets into larger messages, thereby reducing the overhead caused by each packet.

References

[1] B. Aiello, T. Leighton, B. Maggs, and M. Newman. Fast Algorithms for Bit-Serial Routing on a Hypercube. In *Proc. 2nd Symp. on Parallel Algorithms and Architectures*, pages 55–64. ACM, 1990.

[2] J.L. Carter and M.N. Wegman. Universal classes of hash functions. *J. of Computers and System Sci.*, 18:145–154, 1979.

[3] H. Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[4] D. Culler, R. Karp, D. Patterson, A. Sahay, K. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a realistic model of parallel computation. In *Proc. 4th SIGPLAN Symp. on Principles and Practice of Parallel Programming*, pages 1–12. ACM, May 1993.

[5] S.R. Dickey and R. Kenner. Hardware Combining and Scalability. In *Proc. 4th Symp. on Parallel Algorithms and Architectures*, pages 296–305. ACM, 1992.

[6] P.W. Dowd and M. Carrato. High Speed Routing in a Parallel Processing Environment: A Simulation Study. In *The 24th Annual Simulation Symposium*, pages 60–72, 1991.

[7] T. Hagerup and C. Rüb. A Guided Tour of Chernoff Bounds. *Inf. Proc. Lett.*, 33:305–308, 1990.

[8] B.H.H. Juurlink and J.F. Sibeyn. Fast Routing on Weak Hypercubes. Technical Report 95-23, Leiden University, 1995.

[9] B.H.H. Juurlink and H.A.G. Wijshoff. Experiences with a Model for Parallel Computation. In *12th Annual Symposium on Principles of Distributed Computing*, pages 87–96. ACM, 1993.

[10] M. Kaufmann, S. Rajasekaran, and J.F. Sibeyn. Matching the Bisection Bound for Routing and Sorting on the Mesh. In *Proc. 4th Symposium on Parallel Algorithms and Architectures*, pages 31–40. ACM, 1992.

[11] C.G. Plaxton. Load Balancing, Selection and Sorting on the Hypercube. In *Proc. Symp. on Parallel Algorithms and Architectures*, pages 64–73. ACM, 1989.

[12] J.H. Reif. An optimal parallel algorithm for integer sorting. In *Proc. 26th Annual Symp. on Foundations of Comp. Sci.*, pages 496–504. IEEE, 1985.

[13] L.G. Valiant. A Scheme for Fast Parallel Communication. *SIAM J. Comput.*, 11, 1982.

[14] L.G. Valiant. A Bridging Model for Parallel Computation. *Comm. of the ACM*, 33(8), 1990.

[15] L.G. Valiant. General Purpose Parallel Architectures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North Holland, Amsterdam, 1990.

[16] L.G. Valiant and G.J. Brebner. Universal Schemes for Parallel Communication. In *Proc. 13th Symp. on Theory of Computing*, pages 263–277. ACM, 1981.