Lower Bounds on Average Time for Random Destination Mesh Routing and Their Utility as Performance Predictors for PRAM Simulation *

George Chochia, Murray Cole and Todd Heywood

Department of Computer Science

The University of Edinburgh

King's Buildings, Edinburgh EH9 3JZ

Scotland

email: {gac,mic,thh}@dcs.ed.ac.uk

Abstract

This paper presents lower bounds on the expected time for random destination routing on a mesh, valid for any routing scheme, queueing discipline and queue size. We show that the lower bounds are applicable to probabilistic simulation of the P processor EREW PRAM with shared memory M on a $\mathcal{P}$ processor mesh with distributed memory $\mathcal{M} = \mathrm{M}$, $\mathcal{M} = \mathcal{P}^{\epsilon}$, $\epsilon > 1$, and consider two cases: where the number of packets q per processor is one, which corresponds to PRAM simulation with $\mathrm{P} = \mathcal{P}$, and where $q \gg 1$, which corresponds to PRAM simulation with parallel slackness. Experimental results are given showing that the bounds give good analytical predictions of the actual performance of both random destination routing and practical probabilistic PRAM simulation on meshes. The experiments are carried out on a mesh with small fixed queues and memory randomized by pseudo-random hash functions. Simulating PRAM memory accesses with random strides show that $\Theta(\ln \mathcal{P} / \ln \ln \mathcal{P})$-universal hash functions perform better than linear, 2-universal hash functions.

*Work supported by EPSRC grant GR/J43295

1 Introduction

The Parallel Random Access Model (PRAM) provides an ideal platform for the design and analysis of parallel algorithms, as demonstrated by the large body of results on it which has accumulated over the past fifteen years. This model consist of P processors and M shared memory locations, where the processors operate synchronously, and in every parallel step each processor may store/fetch a data element to/from any one memory location. The EREW (Exclusive Read Exclusive Write) PRAM allows at most one processor to access any single shared memory location in one step, while concurrent access variants allow simultaneous reads (CREW) and writes (CRCW) to memory locations.

However, actually implementing a PRAM with a significant number of processors is infeasible due to the P-way fan-in requirement to each memory location. To overcome this, a number of techniques, both deterministic and randomized, for simulating PRAMs on distributed memory architectures have been proposed (see [6] for a survey). Randomized techniques appear to be the most realistic, and involve the mapping of the logical shared memory to the physical distributed memory by means of a pseudo-random hash function such that for (almost) all parallel access patterns the memory references are distributed evenly among the modules comprising the physical memory. Ranade's simulation of a PRAM on a butterfly network [10] is probably the best known example of this technique.

Provably efficient probabilistic simulations of PRAMs on meshes are considered in [8, 10, 11]. In these simulations the logical to physical memory mapping is randomized by means of $\Theta(\frac{\ln M}{\ln \ln M})$-universal [9] hash functions. The known *fast injective* $\Theta(\frac{\ln M}{\ln \ln M})$*-universal* hash functions require space $O(\mathrm{M})$, where the constant factor is more than 3. To minimize memory required for simulation it is desirable to have a *bijective mapping.* For example, 2-universal hash functions $h(x) = (a \cdot x + b)\ mod\ \mathrm{M},\ a \in [1, ..., \mathrm{M}-1], b = [0, .., \mathrm{M}-1],\ x \in [0 : \mathrm{M}-1]$ are bijective if a is relatively prime to M.

We consider probabilistic simulation of a P processor EREW PRAM with shared memory M on a $\mathcal{P}$ processor mesh with distributed memory $\mathcal{M} = \mathrm{M}$. We assume that each communication step of the mesh takes unit time. The physical memory is organized in $\mathcal{P}$ memory banks of size $\mathcal{M}/\mathcal{P}$ each. Each bank is capable of serving at most one request at a time. Functions of the form h_{lin}: $M \to \mathcal{P}$, $h_{lin}(x) = h(x)\ (mod\ \mathcal{P})$ will be called *linear hash functions* and specify the memory bank to which the address $a \in [0...\mathrm{M}-1]$ is mapped. *Memory bank congestion* is the number of requests mapped to the same bank for some PRAM simulation step.

Note that linear hash functions are not bijective and so as many as $\mathcal{P}$ addresses can go to some memory bank in the worst case. However if instead of $h(x)$ we pick up (at random) a $\Theta(\frac{\ln M}{\ln\ln M})$-universal hash function the probability of observing this situation will be sufficiently small [9] to allow simulations of large numbers of EREW PRAM steps with at most $O(\frac{\ln \mathcal{P}}{\ln\ln \mathcal{P}})$ maximal congestion in some memory module.

Dietzfelbing [4] proved that linear hash functions may be not reliable i.e. given a subset $S \subset [0...M-1]$, $|S| \leq \mathcal{P}$ of pairwise distinct elements, and randomly chosen a and b in the linear hash function, the probability to observe congestion $\geq u$ is $\Omega(u^{-2})$. If we repeat this, picking up random sets $\Theta(u^2)$ times, then the probability of observing congestion u will be $\Theta(1)$ for some case. However, the performance of simulations with linear hash functions has been observed to be basically the same as with $\Theta(\frac{\ln M}{\ln\ln M})$-universal hash functions in at least three different experimental studies:

- Ranade [10]: PRAM algorithms for radix and bitonic sorting, matrix multiplication, and FFT.
- Englemann and Keller [5]: linear array with random patterns and with strides 1, 13, 32; and PRAM algorithms for list ranking, matrix multiplication, and connected components.
- Chochia, Cole and Heywood [2]: linear hash functions were tested on the linear array reference problem on a randomly partitioned mesh.

In this paper we show that expected time for *random destination routing* is a lower bound on the expected time for probabilistic simulation of an EREW PRAM step when M $= \mathcal{M}$, and $\mathcal{M} = \mathcal{P}^{\epsilon}$, $\epsilon > 1$, $\mathcal{P} \to \infty$. We compare lower bounds obtained for random destination routing with experiments on simulation of the EREW PRAM on a mesh. In our experiments we use a model of a mesh with small (size one) *first in first out* (FIFO) input and output queues.

Leighton [7], showed that the greedy algorithm for random destination routing has a maximal number of packets queued at any edge which is at most 4 with probability $1 - O(\frac{\log^4 \mathcal{P}}{\sqrt{\mathcal{P}}})$, where $\mathcal{P}$ is the number of processors in a 2D mesh. The probability that any particular packet is delayed Δ steps while on its shortest path to its destination is at most $O(e^{-\Delta/6})$. Note that Leighton's analysis used *furthest first* queuing discipline and did not use small, fixed queues.

The paper is organized as follows. In the first section we establish the lower bound on the expected time for random destination routing on a mesh, where each processor generates at most one request. The proof is based on consideration of the *wide channel model* where multiple packets are allowed to cross a link simultaneously. In the second section we experiment with a more realistic model where only one packet may cross a link at a time in one direction. The two case studies presented are the random destination routing and EREW PRAM simulation on *random arithmetic progression* (RAP). (N.B. This memory access pattern is common for a broad class of algorithms manipulating vectors). In the latter case memory was randomized with linear and 10-universal hash functions. We show that lower bound found in the wide channel model is close to the observed time to simulate PRAM step in the realistic model. The experimental study revealed a significant deviation between worst case performances of linear and 10-universal hash functions for RAP.

In the third part of the paper, we turn to finding the lower bound on expected time to simulate a step of an EREW PRAM with $\mathrm{P} > \mathcal{P}$, also known as simulation with parallel slackness. It is shown in [11] that this simulation results in latency hiding. As was observed in [1] full latency hiding is possible on a hypercube but not on a mesh where it improves the performance by at most a constant factor. We derive the lower bound $\frac{\mathcal{P} \cdot q}{2b}$ on the expected time for random destination routing in a network of $\mathcal{P}$ processors of bisection width b where each processor generates q requests to random destinations. This gives us the lower bound on the expected length of a superstep in probabilistic simulation of a $q\mathcal{P}$ processor EREW PRAM on a $\mathcal{P}$ processor network with bisection b, when $\mathrm{M} = \mathcal{M}$, $\mathcal{M} = \mathcal{P}^{\epsilon}$, $\epsilon > 1$, $\mathcal{P} \to \infty$.

2 Random Destination Routing in the Wide Channel Model

Random destination routing has two phases: (A) each processor sends a packet to some other node (random destination) with equal probability, and (B) each processor receives back an acknowledgement. Let $\delta_{i,j}$, $i, j = 0, ..., n-1$ be the distance, due to the number of links a packet issued by processor (i, j) has to traverse, to the destination and back, and $t_{i,j}$ be the time when that processor receives back an acknowledgement. In the wide channel model $t_{i,j} = \delta_{i,j}$. In general given a model of a mesh where at most one packet is allowed to cross a link (in any direction) at a time $t_{i,j} \geq \delta_{i,j}$. Let $D = \max_{i,j} \delta_{i,j}$ denote the longest distance and $T = \max_{i,j} t_{i,j}$ the completion time for some routing instance.

Theorem 1 *In the wide channel model of a $\mathcal{P}$-processor square mesh for any routing scheme, the expected time T_A to complete phase A, $\boldsymbol{E}(T_A)$ is*

$$(1) \quad \geq 2\cdot\sqrt{\mathcal{P}} - 2 - \sum_{k=1}^{n-1}\prod_{s=0}^{k-1}\left(1-\frac{(k-s)\cdot(k-s+1)}{2\cdot\mathcal{P}}\right)^{4\cdot(s+1)} - \sqrt{\mathcal{P}}\cdot e^{-O(\mathcal{P})}$$

Proof: Let $p_{i,j}(l)$ be the probability that processor (i,j) generates a request to the node at distance $\leq l$. In the wide channel model each packet is advanced towards its destination in every step. Therefore $\mathbf{E}(T_A)=\mathbf{E}(D/2)$, and the probability

$$(2) \qquad Pr(D/2\leq l) = \prod_{i,j=0}^{n-1} p_{i,j}(l)\ .$$

The expected time to complete phase A is

$$\begin{aligned}\mathbf{E}(D/2) &= \sum_{l=1}^{2\cdot n-2} l\cdot(Pr(D/2\leq l)-Pr(D/2\leq l-1)) \\ (3) \qquad &= 2\cdot n - 2 - \sum_{l=0}^{2\cdot n-3} Pr(D/2\leq l)\ ,\end{aligned}$$

where $n=\sqrt{\mathcal{P}}$. Let $l=2\,n-1-k,\ k=1,...,2\,n-1$. We proceed by case analysis on the three ranges of k.

First consider range $1\leq k\leq n/2$. Choose any corner, introduce Cartesian coordinate system with the origin in that corner and coordinate axes going along the edges of the mesh. Choose the set of $s+1,\ s=0,...,k-1$ nodes with coordinates (i,j) such that $i+j=s$. For each element of the set there are $\frac{(k-s)(k-s+1)}{2}$ nodes in the mesh at distance $\geq l$. This is case A in Figure (1). Thus with respect to any corner

$$\prod_{i+j=s} p_{i,j}(l) = \left(\frac{\mathcal{P}-(k-s)\cdot(k-s+1)/2}{\mathcal{P}}\right)^{s+1}\ .$$

In order to compute (2) we find a product over all $s=0,...,k-1$ for all corners,

$$(4) \qquad Pr(D/2\leq l)\leq \prod_{s=0}^{k-1}\left(1-\frac{(k-s)\cdot(k-s+1)}{2\cdot\mathcal{P}}\right)^{4\cdot(s+1)}\ ,$$

for $l\geq 3/2\,n-1$.

Now consider range $n/2<k<n$. In this case there are two types of nodes: first, those for which only one triangle domain exists; second, those with two domains. These are cases B and C in Figure (1). For the first type of nodes the above analysis works. Consider the node of the second type which is at distance $s<k$

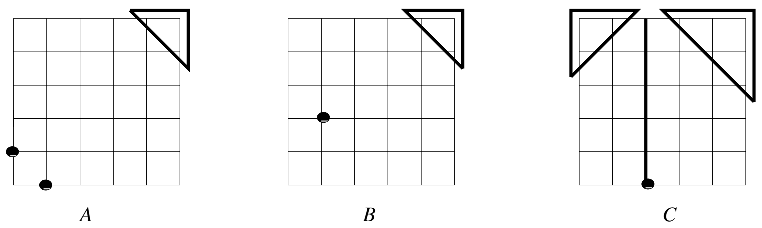

Figure 1: *All mesh nodes are assumed to be at junctions of horizontal and vertical lines. Case A corresponds to the range $1 \leq k \leq n/2$, cases B and C to the range $n/2 < k < n$. Case A: $k = 3$, $s = 1$, two nodes with coordinates $i + j = s$ are marked with dots, the triangle contains nodes at distance ≥ 8; case B: $k = 5$, $s = 3$, there exist one triangle domain of nodes at distance ≥ 6 (first type); case 3: $k = 5$, $s = 2$, $s' = 3$, there exist two triangle domains at distance ≥ 6 separated by a column (second type).*

from one corner and distance $s' < k$ from another (each corner is opposite one triangular domain). Then there exist $z + z'$ nodes at distance $\geq l$ from that node, where $z = \frac{(k-s)\cdot(k-s+1)}{2}$, and $z' = \frac{(k-s')\cdot(k-s'+1)}{2}$. Since $k \leq n - 1$ it follows that minimal l is n, therefore two triangular domains are separated by a column (row) of nodes at distance $< n$ from the node. Hence $p_{i,j}(l) = 1 - \frac{z+z'}{\mathcal{P}}$. From the inequality $1 - \frac{z+z'}{\mathcal{P}} \leq \left(1 - \frac{z}{\mathcal{P}}\right) \cdot \left(1 - \frac{z'}{\mathcal{P}}\right)$ valid for $z, z' \geq 0$ it follows that the contribution of the factor $p_{i,j}$ in (2) can be upper bounded by two factors each one associated with its triangular domain. This allows us to count contributions of factors associated with the triangular domains independently. This leads to (4) which is therefore valid for $l \geq n$.

Finally consider range $k \geq n$ ($l \leq n - 1$). Using the inequality $Pr(D/2 \leq l) \leq Pr(D/2 \leq l + 1)$, valid for $l = 0, ..., 2 \cdot n - 2$, we have

$$\begin{aligned}
\sum_{l=0}^{n-1} Pr\left(D/2 \leq l\right) &\leq n \cdot Pr(D/2 \leq n) \\
&= n \cdot \prod_{s=0}^{n-2} \cdot \left(1 - \frac{(n-s)\cdot(n+1-s)}{2\cdot\mathcal{P}}\right)^{4\cdot(s+1)} \\
&\leq n \cdot e^{-\frac{2}{n^2}\cdot\sum_{s=0}^{n-2}(n-s)\cdot(n+1-s)\cdot(s+1)} \\
&= \sqrt{\mathcal{P}} \cdot e^{-(\frac{\mathcal{P}}{6}+\mathcal{P}^{1/2}+O(1))} \\
&\leq \sqrt{\mathcal{P}} \cdot e^{-O(\mathcal{P})}\ .
\end{aligned}$$

Substituting the bounds on probabilities for the three regions in (3) we obtain the inequality (1). ∎

Using the facts that in the wide channel model packets are never delayed and the minimality of paths under greedy routing, we may conclude that Theorem (1) gives the lower bound on the time to complete the first phase for *any* routing scheme, queue size, and queueing discipline. From the fact $T \geq D$, we find the lower bound on expected time to solve the random destination routing problem $l.b.(\mathbf{E}(T)) = 2 \cdot \mathbf{E}(D/2)$.

We now demonstrate the relationship between random destination routing and EREW PRAM simulation. Consider random destination routing on a $\mathcal{P}$ processor machine. The probability that any $r \geq 0$ processors generate requests to a given destination whereas the other processors generate requests to some other destinations is $\binom{\mathcal{P}}{r}\frac{1}{\mathcal{P}^r}(1-\frac{1}{\mathcal{P}})^{\mathcal{P}-r}$. Now consider probabilistic simulation of a P processor EREW PRAM with shared memory of size M on a $\mathcal{P}$ processor mesh with P $= \mathcal{P}$ and distributed physical memory of total size $\mathcal{M} =$ M. Let H denote a class of M! permutations. In a single step each PRAM processor reads/writes a single memory location therefore the probability that any r shared memory locations are in a given memory module (node) and the other $\mathcal{P} - r$ locations in some other modules for a randomly chosen permutation $h \in H$ is $\binom{\mathcal{P}}{r}\left(\frac{\mathcal{M}/\mathcal{P}}{\mathcal{M}}\right)\ldots\left(\frac{\mathcal{M}/\mathcal{P}-r}{\mathcal{M}}\right)\left(1-\frac{\mathcal{M}/\mathcal{P}-r}{\mathcal{M}}\right)^{\mathcal{P}-r}$. The probabilities are the same if $\mathcal{P} \to \infty$ and $\mathcal{M}/\mathcal{P} \to \infty$. Thus we may conclude that in that limit random destination routing generates a set of communication patterns identical to that observed with a randomly chosen hash function. The class of hash functions H is the largest possible, therefore the expected step time in probabilistic simulation with hash functions (permutations) from H can not be more than that in simulations with any $h \in H'$, $H' \subseteq H$, i.e. this is a lower bound.

3 Experiments with the Realistic Model

Though the wide channel model is simple enough for analytical analysis it is not realistic. In this section we describe a model that can be implemented practically and compare the predictions obtained in the wide channel model with numerical simulation in the realistic model. The model we are dealing with is a variant of the Schnorr-Shamir model with fixed queues. A special communication protocol is implemented to preserve link and memory queues from overflowing or losing packets. We call networks possessing this property *self-stabilized.* Each node has four input and four output links. Each link has a queue of size one. Communications are allowed in both directions simultaneously. Packets are routed using the greedy routing scheme. In that scheme at most three packets may compete for the queue. Therefore in addition to the queuing discipline (FIFO in our case) we must specify a contention resolution rule which we choose to be *random winner*. Transferring a packet to the adjacent node takes one time unit.

In the previous section we analysed the performance of random destination routing. Our practical simulations are carried out for random destination routing and EREW PRAM simulation, with comparisons justified by the arguments in the previous section. Memory is randomized by means of the pseudo random hash function. Recall the definition from [9].

Definition 1 *The class H of hash functions $h : [0...M-1] \to [0...\mathcal{M}-1]$ is (c,k) universal if for all pairwise $a_i \in [0...M-1]$ and $b_i \in [0...\mathcal{M}-1]$, $i = 1, ..., k$, and a hash function $h \in H$ drawn at random, $Pr\ (\ h(a_i) = b_i,\ i = 1, ..., k\ ) \le \frac{c}{\mathcal{M}^k}$, $c \in \mathbb{R}$, $k \in \mathbb{N}$.*

Given any $\mathcal{P}$ pairwise distinct requests hashed to $\mathcal{P}$ memory banks by means of a randomly chosen (c,k)-universal hash function $h \in H$, define $R^{\mathcal{P}}_{max}$ to be the maximal expected congestion in one memory bank. From [9] (Corollary from Theorem[2], p. 346) we have

Fact 1 *If H is (c,k)-universal, $h \in H$: $[0...M-1] \to [0...\mathcal{P}-1]$ and $k = 3\ln\mathcal{P}/\ln\ln\mathcal{P}$, then for any set $S \subset M$, $|S| \le \mathcal{P}$ of pairwise distinct addresses $R^{\mathcal{P}}_{max} \le k + c\mathcal{P}^2/k! = O(\ln\mathcal{P}/\ln\ln\mathcal{P})$.*

The following fact [3] gives a probabilistic estimate for congestions.

Fact 2 *If H is (c,k)-universal, and $h \in H$: $[0...M-1] \to [0...\mathcal{P}-1]$ is chosen at random with equal probability, then for any set $S \subset M$, $|S| \le \mathcal{P}$ of pairwise distinct addresses the congestion b_i, $i = 0...\mathcal{P}-1$ in any memory module is*

$$\forall i,\ Pr(b_i > u) \le \begin{cases} c \cdot e^{u-1} \cdot u^{-u}, & \text{if } u \le k, \\ c \cdot e^{k-1} \cdot u^{-k}, & \text{if } u > k. \end{cases}$$

In our experiments we used (c,k)-universal hash functions of the form $h(x) = \left(\left(\sum_{i=0,k-1} a_i\, x^i\right)\ mod\ \mathcal{M}'\right)\ mod\ \mathcal{P}$, where $a_i \in [1, ..., \mathcal{M}'-1]$, and $\mathcal{M}'$ is the largest prime less than $\mathcal{M}$. Obviously, if $k = 2$ we have a linear hash function. Note, that these hash functions evaluate the destination node or memory bank only. The address within the bank can be easily found but we do not need to know it. For the hash function in use $c < 2$, so the direct evaluation based on Fact 1 gives $R^{\mathcal{P}}_{max} < 11$ for $k = 10$ and $\mathcal{P}$ in the range 16 to 1024.

The RAP memory reference pattern can be described as follows. Each processor generates memory request to the virtual address $\zeta \cdot \#$, where $\zeta = \left[1, ..., \lfloor \mathcal{M}'/\mathcal{P} \rfloor\right]$, and # is a processor number from 0 to $\mathcal{P}-1$. For each run a new value of ζ is picked at random with equal probability. The following statistics has been collected over a number of runs: $\mathbf{A}(T)$ – the average time to complete a simulation step, $\mathbf{Max}(T)$ – maximal time to complete the simulation step, $\mathbf{A}(\Delta)$ – average delay over all packets, $\mathbf{Max}(\Delta)$ – maximal delay over all packets. All statistics were collected twice: for linear and 10-universal hash functions. The results are summarized in Figure 2. From this table we observe that the lower bound $l.b.\mathbf{E}(T)$ and

mesh size	16	64	256	1024	4096
$l.b.\mathbf{E}(T)$	9.90	23.68	52.44	111.80	233.20
$\mathbf{A}(T)$:rand. dest.	10.13	24.11	52.80	112.11	—
$\mathbf{A}(T)$: 10-univ.	10.74	26.37	52.86	112.17	—
$\mathbf{A}(T)$: linear	10.58	26.42	53.05	112.24	—
$\mathbf{Max}(T)$: 10-univ.	14	30	62	124	—
$\mathbf{Max}(T)$: linear	18	70	201	125	—
$\mathbf{A}(\Delta)$: 10-univ.	.24	.44	.63	.78	—
$\mathbf{A}(\Delta)$: linear	.26	.48	.73	.78	—
$\mathbf{Max}(\Delta)$: 10-univ.	4	6	10	12	—
$\mathbf{Max}(\Delta)$: linear	7	44	157	41	—

Figure 2: *The first row is the lower bound on average time for random destination routing on a mesh in wide channel model, obtained using theorem (1). The second row is the average time for random destination routing in the realistic model with queues of fixed size one. The rest of the table presents statistics collected for experiments with linear and ten universal hash functions in the realistic model. The statistics were collected over 3000 runs per case.*

the average observed time for random destination routing in the realistic model $\mathbf{A}(T)$ differ by less than one for all $\mathcal{P}$ in the range 16 to 1024. The difference remains small for the RAP reference problem under pseudo-random hashing and it does not grow with the dimension of a mesh. The average delay $\mathbf{A}(\Delta)$ grows slowly with the dimension. 10-universal hashing performed better than linear hashing for all simulated dimensions. Figure 3 presents histograms for packet delays. Packet delays for 10-universal hash functions fit Leighton's predictions though we are using queues of size one and a FIFO queueing discipline. The distributions become less sharp with the dimension but continue to decrement rapidly

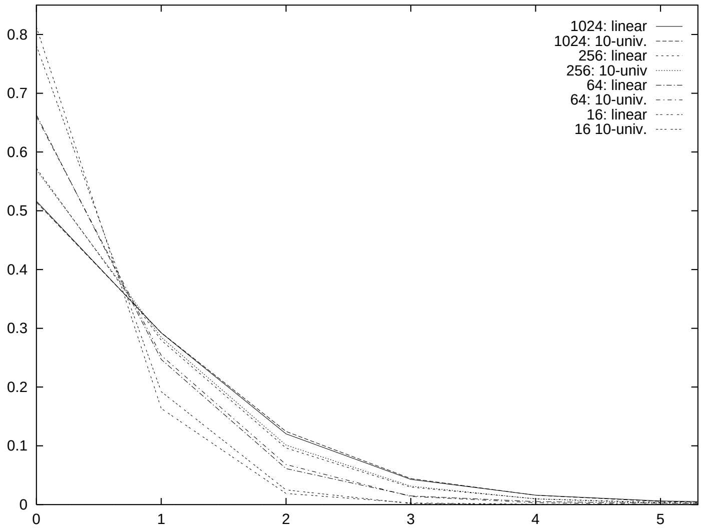

Figure 3: *Packet delay distributions in simulation of EREW PRAM on RAP with memory access randomized by linear and* 10*-universal hash functions with statistics collected over 3000 runs.*

with the value of delay, so that delays larger than 6 become exceptional. In fact, the largest observed delay for the 1024 processor mesh over all packets and runs was only 12. Note that we measured delays on the packet route to the destination and back. Though linear hashing pictorially exhibits identical behavior with 10-universal hashing, the worst case observed delay is much larger (157 units for 256 processor mesh). It is important to note that this delay is due to "bad" hashing resulting in a large memory congestion rather than an example of "bad" routing with uniform hashing. Maximal congestion is not presented in the table but it was observed to be almost as large as maximal delay. Note that the small congestion observed for simulations with 10-universal hash functions are due to the strong probabilistic guarantee based on facts (1) and (2). On the other hand the "large" congestion for linear hash functions was observed almost always within 3000 runs with various randomized seeds. The only experimental result which has to be explained is the "small" maximal congestion for simulation of the 1024 processor mesh with linear hash functions. In fact "large" congestion was observed in that case as well in direct experiments with hash function if the number of runs was more than 10000. Below we show how "large" congestion may appear in linear hashing. First recall that $\mathcal{M}$ is assumed to be polynomial in $\mathcal{P}$, i.e. $\mathcal{M} \gg \mathcal{P}$. Linear hashing $h_{lin}(x)$ distributes the set of RAP addresses $\{\zeta \cdot \# | \# = 0, ..., \mathcal{P}-1\}$, over the banks $((a_1 \cdot \zeta \cdot \# + a_0) \ mod \ \mathcal{M}') \ mod \ \mathcal{P}$, where $\mathcal{P} = 2^r, \ r > 0$. Suppose, without loss of generality, that $a_0 = 0$. If $a_1 \cdot \zeta \equiv 2^k q \ (mod\ \mathcal{M}')$, and $1 \le 2^k q < \lfloor \mathcal{M}'/\mathcal{P} \rfloor$, then g.c.d.$(2^k q, \mathcal{P}) \ge 2^k$ requests go to the same memory bank. On the other hand for any $a \in [1, ..., \mathcal{M}'-1]$ there exist $\rho_{k,q} \in [1, ..., \mathcal{M}'-1]$ such that $a_1 \cdot \rho_{k,q} \equiv 2^k q \ (mod\ \mathcal{M}')$, $k, q \in \mathbb{N}$. If $a_1 \cdot \zeta \ (mod\ \mathcal{M}') > \lfloor \mathcal{M}'/\mathcal{P} \rfloor$ then "bad" congestion may be observed as well. For more analysis of the probabilities of these events we refer the reader to [4]. Figure 4 represents histograms for the

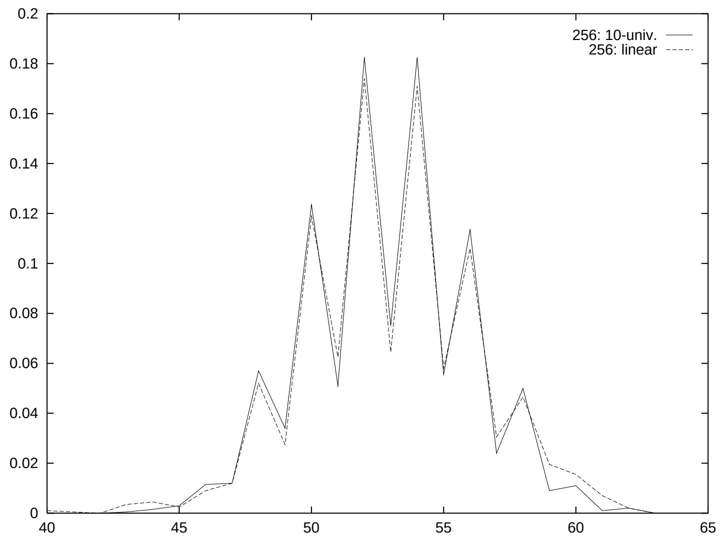

Figure 4: *EREW PRAM time step distribution for RAP simulated on a 256 node mesh with memory access randomized by linear and* 10*-universal hash functions, with statistics collected over 3000 runs.*

time to simulate a PRAM step on a RAP problem. The peaks occur at even numbers of steps. As the delay distribution is a rapidly decreasing function, the sum of probabilities for packets delayed an even number of times (including non delayed packets) is noticeably larger than that for packets delayed an odd number of times.

4 Random Destination Routing with Parallel Slackness

Consider the following instance of random destination routing. Each processor generates q packets destined to any other node with equal probability and receives back an acknowledgement for each packet. The time to generate a packet is set to zero. In what follows we will be looking for the lower bound on average expected time required to perform this routing.

As will be shown later this bound is valid for PRAM simulation [12] in the case P $= q\mathcal{P}$ (in other words, with parallel slackness). In that setting this is the lower bound on the average length of a superstep. Due to the pipelining of the outstanding requests this technique leads to latency hiding in a network. Full latency hiding requires that the network has a capacity which can handle $q\mathcal{P}$ packets simultaneously. Otherwise, the degree of latency hiding of the self-stabilized network is limited by the rate at which packets are injected into the network. First we consider an r-dimensional mesh and applying "edge use" arguments suggested in [1] derive the lower bound on expected time to execute the random destination routing instance described above. Later we improve upon that bound and give a general expression valid for any network.

Let T_q denote the time required to execute an instance of our random destination routing problem. Consider an r-dimensional rectangular mesh $G_r = (V_r, E_r)$ with set of nodes V_r and set of edges E_r. Put $\mathcal{P} = |V_r| = d_1 \times \ldots \times d_r$, $d_i > 1$, $i = 1, \ldots, r$ and $\mathcal{L} = |E_r|$. The manhattan distance between two nodes $x, y \in V_r$ is the function $\rho(x,y) = \sum_{i=1}^{r} |x_i - y_i|$. Define $l(x)$ to be the number of links crossed by a packet issued at node $x \in V_r$ on its way to its destination. $\sum_{x\in V_r} l(x)$ is total path length for a given instance of the routing problem. If at any time step until the routing completes some packet crosses each edge and each packet is routed along the minimal path, then for that routing problem the total time is as small as possible. Hence for expected time $\mathbf{E}(T_q)$ we have the inequality

$$\text{(5)} \qquad \mathbf{E}(T_q) \geq \frac{\mathbf{E}\left(2 \cdot \sum_{x\in V_r} l(x)\right)}{\mathcal{L}} = \frac{2 \cdot \sum_{x\in V_r} \mathbf{E}(l(x))}{\mathcal{L}} = \frac{2 \cdot q}{\mathcal{L} \cdot \mathcal{P}} \sum_{x,y \in V_r} \rho(x,y) \ .$$

The following Lemma gives $\sum \rho(x,y)$ for the r-dimensional rectangular mesh.

Lemma 1 *For the r-dimensional rectangular mesh*

$$\text{(6)} \qquad \sum_{x,y\in V_r} \rho(x,y) = \frac{1}{3} \cdot \prod_{i=1}^{r} d_i^2 \cdot \left(\sum_{j=1}^{r} d_j - \frac{1}{d_j} \right)$$

$$\text{(7)} \qquad \mathcal{L} = \prod_{i=1}^{r} d_i \cdot \left(r - \sum_{j=1}^{r} \frac{1}{d_j} \right)$$

Proof: *Base of induction*: Consider a one-dimensional array of d nodes. Then $\sum_{i,j=0}^{d-1} |i-j| = \frac{1}{3} \cdot d^2 \cdot \left(d - \frac{1}{d}\right)$ and the lemma holds.

Inductive step: Suppose the lemma holds up to dimension $k > 1$. Consider $k+1$ dimensional mesh $G_{k+1} = (V_{k+1}, E_{k+1})$. Partition it into d_{k+1} isomorphic k-dimensional sub-meshes $G_k^i = (V_k^i, E_k^i)$, $i = 1, \ldots, d_{k+1}$, along the $k+1$ coordinate.

Let $\mathcal{P}_k = \prod_{i=1}^{k} d_i$, then

$$\sum_{x,y\in V_{k+1}} \rho(x,y) = \sum_{i,j=1}^{d_{k+1}} \sum_{\substack{x\in V_k^i \\ y\in V_k^j}} \rho(x,y) \;=\; \sum_{i,j=1}^{d_{k+1}} \left(\sum_{x,y\in V_k} \rho(x,y) + |i-j|\cdot \mathcal{P}_k^2 \right)$$

$$= \frac{1}{3}\cdot \prod_{i=1}^{k+1} d_i^2 \cdot \left(\sum_{j=1}^{k+1} d_j - \frac{1}{d_j} \right) \;.$$

Here $G_k = (V_k, E_k)$ is a sub-mesh isomorphic to $G_k^i,\ i = 1, ..., d_{k+1}$. The proof of (7) is based on the fact that one-dimensional array of d nodes has $d-1$ edges. ■

Substituting $\sum \rho(x,y)$ for a square mesh from Lemma (1) in (5) we can prove the following

Theorem 2 *For a square $\mathcal{P}$-processor mesh expected time for random destination routing is $\boldsymbol{E}(T_q) \geq \frac{q}{3}(\sqrt{\mathcal{P}} - \frac{1}{\sqrt{\mathcal{P}}})$ for any $q \geq 1$, routing scheme, queueing discipline and queue size.*

The following theorem takes into account bisection width arguments and gives a better bound than Theorem (2) for a square mesh.

Theorem 3 *Given a $\mathcal{P}$ processor distributed network with bidirectional links and bisection width b, the expected time for random destination routing $\boldsymbol{E}(T_q) \geq \frac{\mathcal{P}\,q}{2\cdot b}$ for any $q \geq 1$, routing scheme, queueing discipline and queue size.*

Proof: Consider some arbitrary node and suppose that the network is split into two halves. The half to which the node belongs and the other "distant" half. The probability that a node generates a request to a node in the distant half is $\frac{1}{2}$. Each node generates q requests. The probability that k requests cross the boundary is $\binom{q\cdot\mathcal{P}/2}{k}\left(\frac{1}{2}\right)^{q\cdot\mathcal{P}/2}$. Hence

$$\mathbf{E}(T_q) \;\geq\; 2\cdot \frac{\sum_k k\cdot \binom{q\cdot\mathcal{P}/2}{k}\cdot\left(\frac{1}{2}\right)^{q\cdot\mathcal{P}/2}}{b} = \frac{q\,\mathcal{P}}{2\,b}\sum_k \binom{q\cdot\mathcal{P}/2-1}{k-1}\cdot\left(\frac{1}{2}\right)^{q\cdot\mathcal{P}/2-1} = \frac{\mathcal{P}\,q}{2\,b}\;.$$

■

The square $\mathcal{P}$-processor mesh has a bisection width $\sqrt{\mathcal{P}}$, therefore $\mathbf{E}(T_q) \geq \frac{q\sqrt{\mathcal{P}}}{2}$. Repeating arguments of section (2) for the case when each processor of a mesh simulates $q\mathcal{P}$ PRAM processors we can show that if $\mathcal{M} = \mathcal{P}^{\epsilon},\ \epsilon > 1$ and $\mathcal{P} \to \infty$

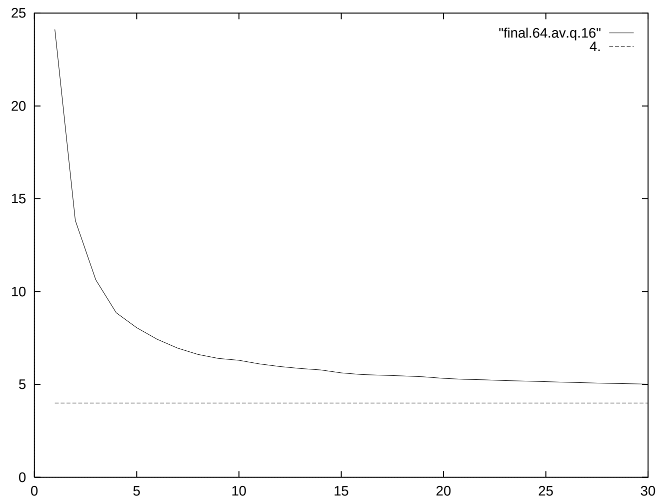

Figure 5: *1 — average (over 500 observations per point) time, $\boldsymbol{A}(T_q/q)$ for random destination routing on a* 64 *node square mesh (realistic model with greedy routing, FIFO queues of size 16) as a function of number requests per node q; 2 — lower bound from theorem 3.*

then the expected time for random destination routing is equal to that for probabilistic PRAM simulations with memory randomized by means of hash function $h \in H$, where H is a class of all permutations. Therefore we conclude that in that limit the expected length of a superstep is $\geq \frac{\mathcal{P}\,q}{2\,b}$ for any probabilistic simulations of a $q\mathcal{P}$ processor EREW PRAM on a $\mathcal{P}$ processor mesh with $\mathcal{M} = \mathrm{M}$.

References

[1] Bilardi, G., and Preparata, F.P., "Horizons of Parallel computation." in: A. Bensoussam, J.-P. Verjus (eds.), *Future Tendencies in Computer Science, Control and Applied Mathematics, Int. Conf. on the Occasion of the 25th Anniversary of INRIA*, LNCS 653, 1992, pp. 155-174.

[2] Chochia, G., Cole, M., and Heywood, T., "Implementing the Hierarchical PRAM on the 2D Mesh: Analyses and Experiments", *In Proc. of the Seventh IEEE Symposium on Parallel and Distributed Processing*, 1995, to appear.

[3] Dietzfelbinger, M., Karlin, A., Mehlhorn, K., Meyer auf der Heide, M., Rohnert, H., and Tarjan, R.E., "Dynamic Perfect Hashing: Upper and Lower Bounds ", *Proc. of the 29th IEEE FOCS*, 1988, pp. 524-531.

[4] Dietzfelbinger, M.,"On Limitations of the Performance of Universal Hashing with Linear Functions" *Technical Report No. 84*, Reihe Informatik, Universität Paderborn, 1991.

[5] Engelmann, C., and Keller, J., "Comparison of Hash Functions for Emulated Shared Memory", *In Proc. Parallel Architectures and Languages Europe*, LNCS 694, Springer, 1993, pp. 1-11.

[6] Harris, T., "A Survey of PRAM Simulation Techniques", *ACM Computing Surveys*, June 1994, pp. 187-206.

[7] Leighton, T.,"Average case analysis of greedy routing algorithms on arrays" *Proc. ACM Symp. on Parallel Algorithms and Architectures*, 1990, pp. 2-10.

[8] Leighton, T., Maggs, B.M, Ranade, A.G., and Rao, S.B., "Randomized Routing and Sorting on Fixed-Connected Networks", Journal of Algorithms, July 1994.

[9] Mehlhorn, K., and Vishkin, U.,"Randomized and Deterministic Simulations of PRAMs by Parallel Machines with Restricted Granularity of Parallel Memories", *Acta Informatica*, 21, 1984, pp. 339-374.

[10] Ranade, A.G., Bhatt, S.N., and Johnsson, S.L., "The Fluent Abstract Machine", *Advanced Research in VLSI: Proceedings of the 5th MIT Conference*, J. Allen and T. Leighton, (eds.), MIT Press, April, 1988, pp. 71-94.

[11] Valiant, L.G., "General purpose parallel architectures", *In J. van Leewen, editor, Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, chapter 18, 1990, Elsiever, Amsterdam, pp. 943-971.

[12] Valiant, L.G., "Bulk-synchronous Parallel Computers", *In Parallel Processing and Artificial Intelligence*, M. Reeve and S. Zenith, eds. (John Wiley, New York, 1989) pp.15-22.