Randomized routing on generalized hypercubes

Afonso Ferreira a,1, Miltos Grammatikakis b,*,2

a LIP-CNRS, CNRS-URA no. 1398, ENS Lyon, 69364 Lyon, Cedex 07, France
b Institute of Computer Science, FORTH, Heraklion 71110, Greece

Received June 1994; revised January 1995
Communicated by M. Nivat

Abstract

We propose and theoretically analyze a new probabilistic permutation routing algorithm, which is based on two phases of 1–1 permutation routing. Assuming the multiaccepting model, we show that the probability of routing all N packets on the N-node, base-b generalized hypercube in asymptotically optimal $C \log_b N$ time (constant $C \geq 1$) approaches exponentially one, as N increases. Furthermore, our derived upper bound on the above probability improves on previous results, especially for the binary hypercube, for which bounds can be further improved. Comparison tables of these upper bounds are also provided.

1. Introduction

Packet routing algorithms on the hypercube may be classified into oblivious and adaptive strategies.

For oblivious algorithms the routing path of a packet is uniquely determined from its source and final destination. Both the commonly used greedy algorithm, which uniformly corrects hypercube dimensions in either increasing or decreasing order, and a variant which corrects dimensions in random order belong to this class [1, 10]. Although these strategies are simple, their worst-case time complexity is $\Omega(\sqrt{N})$ [8]. A slightly better oblivious algorithm, based on many-to-one and one-to-many routing in subcubes, achieves a worst-case time delay of $O(\sqrt{N}/\log_2 N)$ [8]. Its time complexity matches the lower bound for oblivious routing on the binary hypercube.

Adaptive routing algorithms are usually disguised as sorting algorithms. Attempts at embedding on the hypercube of both Batcher's, depth $O(\log^2 N)$ sorting network, and AKS, depth $O(\log N)$ sorting network, result in $\Omega(\log^2 N)$ time delay. Recently, Cypher

* Corresponding author. Email of the authors: mdgramma@ics.forth.gr and ferreira@lip.ens-lyon.fr.
1 Partially supported by the PRC PRS and Math-Info of the French CNRS.
2 Now with Institut für Informatik, Marienburger Str. 22, Universität Hildesheim, 31141 Hildesheim, Germany.
and Plaxton [4] designed a complex on-line adaptive sorting algorithm, called sharesort, which achieves a worst-case time delay of $O(\log N \log^2 \log N)$, the best known asymptotic result for deterministic routing. Optimal adaptive (oblivious) routing schemes on the $N$-node binary hypercube have been constructed, for $N \leq 128$ ($N \leq 256$) [5, 13].

A major theoretical breakthrough was the randomized permutation routing algorithm on the binary hypercube, proposed by Valiant [11], and improved by Valiant and Brebner [12]. For this algorithm, the probability that all packets have been routed correctly within $C \log_2 N$ time is proved greater than $(1 - e^{-D \log_2 N})$, where $D$ is a constant depending only on $C$. The algorithm consists of two phases, as described below.

(A) **Randomization phase**: Packets are sent to randomly selected nodes through the network. Hence, at the end of this phase, packets are distributed randomly. The role of randomization is to reduce the gap between the average and the worst-case time complexity.

(B) **Deterministic phase**: Packets follow a shortest-path route to their final destination.

Simulation experiments have indicated the good expected performance of greedy routing for performing random permutations on Cayley graphs, such as hypercube, star, and alternating-group graph [6], and mesh topologies [9]. In this study, we provide theoretical evidence for the case of the hypercube and propose an improvement to the above randomized strategy: in the randomization phase (A) all packets are sent to intermediate destinations characterized by a random permutation. Hence, at the end of the randomization phase, unlike the algorithms in [11, 12], there is only one packet at each node, that will be deterministically routed to its final destination, as described above (B). Our main contribution is the design of new tools for modeling conflicts in a hierarchy of architectures defined by the generalized hypercube [2]. With the help of these tools we analyze our routing algorithm for generalized hypercubes. Comparing with the algorithm in [12], we show that our algorithm improves on the probabilistic time needed to route any permutation.

We assume that there is a buffer for every edge adjacent to a node (d-port communication), and a processing element may accept one packet from each of its neighbors at the same time (multi-accepting model) [1]. Our focus is on time complexity, assuming that buffers are long enough to accommodate as many packets as necessary. A brief description of the contents of this paper follows.

In Section 2 we recall the definition of the generalized hypercube architecture, provide the mathematical lemmas necessary for the analysis, and describe the new routing algorithm with its properties. In Section 3, we derive necessary and sufficient conditions for conflicts in a base-$b$ hypercube, for our routing algorithm. Using this analysis, in Theorem 3.1 we obtain a Chernoff upper bound on the probability of routing on the $N$-node, base-$b$ generalized hypercube in optimal $C \log_b N$ time, for any constant $C \geq 1$. In Section 4 and Proposition 4.1, we show that our bound of Theorem 3.1 is better than a similar Chernoff upper bound obtained by applying the randomized routing strategy from [12] to the generalized hypercube; we justify why upper bounds can
be compared instead of the exact probabilities. Further, in Proposition 4.2 we prove that there is an exponentially increasing probability (as $N$ increases) for our routing algorithm on the binary hypercube to be completed in $4.44 \log_2 N$ total (queuing + propagation) routing time, while the corresponding time for the algorithm in [12] is $5.90 \log_2 N$. The difference in favor of our routing method is maximum for the binary hypercube ($b = 2$). Comparison tables of the upper bounds on these two algorithms illustrate our analysis. We conclude this paper by providing relevant questions for further research.

2. Architecture and routing algorithms

2.1. Generalized hypercube architecture

An $N$-node, base-$b$ hypercube of dimension $k$ is a multicomputer, which consists of $N$ nodes (PEs) numbered as $0, 1, 2, \ldots, N - 1$, where $N = b^k$ [2]. A node $x$ can be represented as $x = x_1x_2 \ldots x_k$, where $x_i \in \{0, 1, \ldots, b - 1\}$, for $1 \leq i \leq k$. Two nodes of the hypercube $x = x_1x_2 \ldots x_k$ and $y = y_1y_2 \ldots y_k$ are connected by an edge if and only if $x_i \neq y_i$ and $x_j = y_j$ for all $j \neq i$ and $1 \leq i, j \leq k$.

We also use the following notations: $x: (x_j = \tau)$ defines a node obtained by changing the $j$th digit in $x$'s $b$-ary representation to $\tau$. Finally, $H(x, y)$ denotes the relative Hamming distance of the two nodes $x, y$.

2.2. Mathematical preliminaries

Lemma 2.1 (Hoeffding [7]). If we make $n$ independent Poisson trials with corresponding probabilities $p_i$, where $1 \leq i \leq n$, the probability of $m$ or more successes is

$$P[\text{At least } m \text{ successes}] \leq B(m, n, p),$$

where $p = \sum_{i=1}^{n} p_i/n$, and $B(m, n, p)$ denotes the binomial distribution (the probability of at least $m$ successes in $n$ trials, when $p$ is the success probability of a single trial).

Lemma 2.2 (Chernoff [3]). For $m \geq np + 1$, Chernoff's tail approximation of the binomial distribution is

$$B(m, n, p) \leq (\frac{np}{m})^m e^{-np}.$$

Lemma 2.3. The average node distance in an $N$-node generalized hypercube of base $b$ is $\eta = ((b - 1)/b) \log_b N$.

Proof. There are

$$g(i) = \binom{\log_b N}{i} (b - 1)^i$$
nodes at distance \( i \) from any given node. Hence, the average distance is

\[
\eta = \frac{1}{N} \sum_{i=1}^{\log_b N} ig(i) = \frac{b-1}{b} \log_b N. \quad \square
\]

2.3. A new probabilistic permutation routing algorithm

We propose a new probabilistic permutation routing algorithm, which is a variant to previous randomized routing schemes. It consists of two phases of shortest-path permutation routing: probabilistic phase A, followed by deterministic phase B. Our algorithm is as follows.

(1) **Probabilistic phase A:** Let \( \psi \) be a random permutation over the set of all \( N \) hypercube nodes. We assume that the permutation \( \psi \) has been precomputed off-line; therefore the destination of each node is available prior to executing probabilistic phase \( A \). Then, a packet \( X \) at node \( x = x_1x_2\ldots x_k \) is sent to a distinct intermediate destination node \( \psi(x) = \psi_1^x \psi_2^x \ldots \psi_k^x \), using shortest-path routing. Hence, at the end of this phase we have one packet at each network node. Let the "cobegin ... coend" refer to all packets. Formally,

\[
\text{cobegin} \\
i := 1 \\
\text{while } ((x_i \neq \psi_i^x) \text{ and } (i < k)) \text{ do} \\
\quad \text{Transmit } X \text{ from } x \text{ to } x : (x_i = \psi_i^x) \\
i := i + 1 \\
\text{coend}
\]

(2) **Deterministic phase B:** Packets are sent deterministically from the distinct intermediate nodes of phase \( A \) to their final destinations, using shortest-path routing. Assuming that an arbitrary packet \( X \) must be moved from source node \( \psi(x) = \psi_1^x \psi_2^x \ldots \psi_k^x \) to destination node \( x' = x_1'x_2'\ldots x_k' \), we have

\[
\text{cobegin} \\
i := 1 \\
\text{while } ((\psi_i^x \neq x_i') \text{ and } (i < k)) \text{ do} \\
\quad \text{Transmit } X \text{ from } \psi(x) \text{ to } \psi(x) : (\psi_i^x = x_i') \\
i := i + 1 \\
\text{coend}
\]

2.4. Properties of our algorithm

The following four properties (oblivious, symmetric, nonrepeating, and maximum delay) hold for each phase of our routing algorithm. They are used in the analysis of Section 3. Notice that the same properties also hold for Valiant and Brebner’s [12] probabilistic routing scheme.

1. **Oblivious property:** The route of any packet does not depend on the route of any other packet.
2. **Symmetric property**: The expected number of packets crossing an edge is independent of which edge is chosen. This property holds, since the permutation $\psi$ is chosen randomly over the set of all hypercube nodes.

3. **Nonrepeating scheme**: After two routes intersect and depart from each other, they can no longer intersect. Formally, for routes $R_1 = (r_1, r_2, \ldots, r_k)$ and $R_2 = (r'_1, r'_2, \ldots, r'_k)$, if $r_i = r'_j$, $r_k = r'_m$, $k > i$, then $k - i = m - j$ and $r_p = r'_{p+j-i}$, $i \leq p \leq k$. This property follows from the fact that each dimension is considered only once in each routing phase.

4. **Maximum delay**: If $L$ packets intersect over a route $R$, the maximum queuing delay of a packet following route $R$ is $L - 1$. To see this, notice that the worst-case queuing delay occurs when all other packets are given priority over a certain packet. Then, this packet submits priority to another packet $L - 1$ times.

3. **Analysis of the new routing algorithm**

We proceed to derive an upper bound on the probability that all $N$ packets are routed to their final destination within $C \log_b N$ time, where constant $C \geq 1$. Notice that, initially, there is one packet at each hypercube node. Similarly, at the end of phase A, all packets are at distinct hypercube nodes, since a random permutation $\psi$ is used for routing. Phase B, in turn, moves the packets to their final destination. We can think of phase B as one of moving backwards in time, wherein the packets are routed from their final destination to the distinct nodes specified by the random permutation $\psi$. This viewpoint, based on time reversal, simplifies the estimation of the total delay time, since phases A and B are quite similar. Hence, from now on we concentrate on the analysis of phase A, computing the probability of finishing in optimal time. The same idea was also used in the analysis in [12].

The main difficulty in analyzing routing algorithms is namely how to model contention. Our analysis follows three basic steps. First, we compute an upper bound on the probability of a conflict between two packets (with source and destination chosen arbitrarily from a random permutation). Then, since conflicts can be considered independent (oblivious and symmetric property, Section 2.4), we derive a binomial distribution describing the number of conflicts of a packet. Finally, using Lemma 2.2, we obtain an upper bound on the probability of more than $C \log_b N$ conflicts for any of the $N$ packets. Assuming a worst-case queuing effect, this bound also reflects the probability of more than $C \log_b N$ queuing delay.

3.1. **Conditions for congestion**

Suppose two arbitrary packets $X$ and $Y$ originating at nodes $x \equiv x_1x_2 \ldots x_k$ and $y \equiv y_1y_2 \ldots y_k$ are destined (after phase A) to nodes $x' = x'_1x'_2 \ldots x'_k$ and $y' = y'_1y'_2 \ldots y'_k$, respectively.
The symmetry of our routing scheme (hypercube edge transitivity, obliviousness, random permutation) implies that the expected delay from the remaining packets is the same at any node in the routes of X and Y. Therefore, Lemma 3.1 below considers two arbitrary packets, ignoring the contention arising from the remaining packets in the system.

Lemma 3.1. Packets X and Y will conflict if and only if all of the following conditions (α), (β), (γ), and (δ) are satisfied for some $s$, $1 \leq s \leq k - 2$, $k > 2$.

(α) $x_i = y_i$, where $s < i < k$.

(β) $x'_i = y'_i$, where $1 \leq i \leq s$.

(γ) $H(x_1x_2...x_s, x'_1x'_2...x'_s) = H(y_1y_2...y_s, y'_1y'_2...y'_s)$.

(δ) $x'_{s+v} = y'_{s+v}$ and $x_{s+v} \neq y'_{s+v}$, when $x_{s+j} = x'_{s+j}$ and $y_{s+j} = y'_{s+j}$, for some $1 \leq v \leq k - s - 1$ and $1 \leq j \leq v - 1$.

Proof. Congestion occurs when the two packets are routed through the same link, at the same time. This can only happen once (nonrepeating property, Section 2.4). Since digits in the deterministic algorithm are changed in a left to right order, packets X and Y will meet at the same node $z = x_1x_2...x_sx_{s+1}x_{s+2}...x_k$, if and only if for some $s$, the first $s$ destination digits and the last $k - s$ source digits of $x$ equal those of $y$ (conditions (α) and (β)). Also, the two packets meet at node $z$ at the same time, if and only if the number of digits changed in their path to node $z$ is the same (condition (γ)). Notice that we ignore delays from remaining packets; the symmetry of our approach implies that, at any node in the paths of packets X and Y to node $z$, expected delays from remaining packets are the same. In addition, for a congestion to occur, we must also require that both packets traverse the same edge at the same time. Thus, condition (δ) must be true.

As an example, consider routing on the $N = 16$ node binary hypercube. If the final destinations for packets 1000 and 0100 are 1111 and 1110, respectively, there will be a conflict at node 1100, since both packets follow the same edge after meeting at node 1100. Note that conditions (α), (β), (γ), (δ) of Lemma 3.1 are all true for $s = 2$.

1000 $\rightarrow$ 1100 $\rightarrow$ 1110 $\rightarrow$ 1111
0100 $\rightarrow$ 1100 $\rightarrow$ 1110

3.2. Probability of finishing in optimal time

In Lemma 3.2 below, we obtain an approximation of the probability of a conflict between two packets X and Y with (source, destination) pair chosen arbitrarily from the random permutation $\psi$. We are ignoring temporarily contention with the remaining packets. Conflicts with these packets are statistically evaluated in Theorem 3.1.

Lemma 3.2. The probability of a conflict between two arbitrary packets routed on the $N$-node, base-$b$ generalized hypercube is $p_1 < \log_b N/((b + 1)N)$. 
Proof. Consider a random pair of packets $X$ and $Y$, with each packet's source randomly matched to a destination. Then, the sources $x$ and $y$ are picked at random, without replacement, from the set $\{0, 1, \ldots, N - 1\}$, and similarly for destinations $x'$ and $y'$.

For a given $s$, let $P(\alpha)$ denote the probability of the event corresponding to condition (a), and similarly for conditions (\beta), (\gamma), (\delta) of Lemma 3.1. Then, we have:

$$P(\alpha) = P[\text{Last } k - s \text{ digits of } x, y \text{ are the same}] = \frac{1}{b^{k-s}}.$$  

$$P(\beta) = P[\text{First } s \text{ digits of } x', y' \text{ are the same}] = \frac{1}{b^s}.$$  

$$P(\gamma) = \sum_{i=1}^{s} P[\text{Packet } X \text{ changes } i \text{ digits}] \cdot P[\text{Packet } Y \text{ changes } i \text{ digits}]$$

$$= \sum_{i=1}^{s} \frac{(b-1)^i \binom{s}{i} (b-1)^i \binom{s}{i}}{b^i} = \frac{1}{b^{2s}} \sum_{i=1}^{s} (b-1)^{2i} \binom{s}{i}^2.$$  

$$P(\delta) = \sum_{i=1}^{k-s-1} P[X \text{ changes } i \text{ digit } 1] \cdot P[Y \text{ changes } i \text{ digit } 1 \text{ similarly}]$$

$$= \sum_{i=1}^{k-s-1} \frac{1}{b^{2(i-1)}} \frac{b-1}{b^2} = 1 - \frac{b^{2(s-k+1)}}{b+1}.$$  

The if and only if condition of Lemma 3.1 implies that the four events (a), (\beta), (\gamma), and (\delta) are independent. Therefore, the probability of a conflict between two arbitrary packets is given by the following sum of products:

$$p_1 = \sum_{s=1}^{k-2} P(\alpha)P(\beta)P(\gamma)P(\delta) = \sum_{s=1}^{k-2} \frac{1}{\sqrt{N}} \frac{1}{b^{2s}} \left[ \sum_{i=1}^{s} (b-1)^{2i} \binom{s}{i}^2 \right] \frac{1 - b^{2(s-k+1)}}{b+1}.$$  

After some simplification we obtain

$$p_1 = \frac{1}{(b+1)N} \sum_{s=1}^{k-2} \left( \frac{1}{b^{2s}} - \frac{b^2}{N^2} \right) \sum_{i=1}^{s} (b-1)^{2i} \binom{s}{i}^2.$$  

(3.1)

Note that the sum of squares of a set of nonnegative integers is always less than or equal to the square of their sum. Hence,

$$p_1 \leq \frac{1}{(b+1)N} \sum_{s=1}^{k-2} \left( \frac{1}{b^{2s}} - \frac{b^2}{N^2} \right) \left[ \sum_{i=1}^{s} (b-1)^{2i} \binom{s}{i}^2 \right]^2.$$  

Since $\sum_{i=0}^{s} (b-1)^i \binom{s}{i} = b^s$, we have

$$p_1 \leq \frac{1}{(b+1)N} \sum_{s=1}^{k-2} \left( \frac{1}{b^{2s}} - \frac{1}{N^2} \right) (b^s - 1)^2 \leq \frac{1}{(b+1)N} \sum_{s=1}^{k-2} \frac{1}{b^{2s}} (b^s - 1)^2.$$  

Therefore,

$$p_1 \leq \frac{1}{(b+1)N} \sum_{s=1}^{k-2} \left( 1 - \frac{2}{b^s} + \frac{1}{b^{2s}} \right).$$
Recalling that $k = \log_b N$, we have

$$p_1 \leq \frac{\log_b N}{(b + 1)^N}. \quad \square$$

**Theorem 3.1.** For our probabilistic routing strategy the probability that all $N$ packets have been routed on the $N$-node, base-$b$ generalized hypercube within $C \log_b N$ time, $C \geq 1$, is greater than $(1 - e^{-D \log_b N})$, where the constant $D$ is

$$D = C \ln \left( (b + 1) e^{1/C(b+1)-1} \right) - \ln(b)$$

and $e = 2.718 \ldots$ represents the natural logarithm base.

**Proof.** Our probabilistic routing strategy is oblivious. The assumption that the routed permutation $\psi$ is random and the edge transitivity property of the hypercube imply that the expected traffic through any hypercube edge is the same (symmetric property). Therefore, conflicts between packets can be considered independent. The probability $p_1$ of a conflict between two arbitrary packet routes was computed in Lemma 3.2. Further, the probability of a given number of conflicts between a packet and the remaining packets is the result of $N$ Poisson trials, with average success probability $p_1$. Using Lemma 2.1, the probability that the total amount of conflicts is at least $C \log_b N$ is bounded above by the binomial distribution $B(C \log_b N, N, p_1)$. Using the maximum delay property of Section 2.4, the probability $\Phi_1$ of at least $C \log_b N$ queuing delay in routing a given packet is also bounded by $\Phi_1 \leq B(C \log_b N, N, p_1)$. Therefore, if $\Phi_1$ represents the probability that at least one packet will be delayed more than $C \log_b N$ time during phase A (and phase B), we have

$$\Phi_1 = 1 - (1 - \Phi_1)^N \leq 1 - (1 - p_1)^N = N\Phi_1. \quad (3.3)$$

Using Lemma 5, and applying Lemma 2.2 (which holds true, since $C \log_b N \geq N \log_b N / ((b + 1)N)$, for any $C \geq 1$), the probability $\Phi_1$ is given by the following formula,

$$\Phi_1 \leq N \exp \left[ -C \log_b N \ln \left( (b + 1) \exp(1/C(b + 1) - 1) \right) \right] \quad (3.4)$$

By substituting $N = e^{\log_b N \ln b}$ into Eq. (3.4), we observe that in order to have $\Phi_1 \leq e^{-D \log_b N}$, the constant $D$ is

$$D = C \ln \left( (b + 1) e^{1/C(b+1)-1} \right) - \ln(b). \quad \square$$

**4. Comparisons with Valiant and Brebner’s strategy**

We specialize the following general theorem (due to Valiant and Brebner) to the case of permutation routing on the generalized hypercube. The term *initialized scheme* refers to having initially at most $\lambda$ packets at any node, with no destination occurring on more than $\lambda$ packets.
Theorem 4.1 (Valiant and Brebner [12]). In any initialized scheme that is oblivious, nonrepeating, and symmetric, with (i) N nodes, (ii) degree d, (iii) \( T = \lambda N \) packets in total, (iv) maximal route length \( \mu \), and (v) expected route length \( \eta \), the probability \( \Phi_2 \) that some packets are delayed at least \( \Delta \geq \lambda \eta / d \) units is 
\[
\Phi_2 \leq T (e^{\eta / \eta / ((\Delta d))^{1/2}}).
\]

Applying Theorem 4.1 to the generalized hypercube, we have

Corollary 4.1. For the probabilistic routing from [12], the probability that all \( N \) packets have been routed on the \( N \)-node, base-\( b \) generalised hypercube within \( C \log_b N \) time, \( C \geq 1 \), is greater than 
\[
1 - e^{-D' \log_b N},
\]
where the constant \( D' \) is
\[
D' = C \ln (\frac{bC}{e}) - \ln(b).
\]

Proof. For permutation routing on the generalized hypercube, \( \lambda = 1 \), \( d = (b-1) \log_b N \), \( \Delta = C \log_b N \), \( \mu = \log_b N \), and \( \eta = ((b-1)/b) \log_b N \) (Lemma 2.3). From Theorem 4.1, the probability \( \Phi_2 \) that some packets have a queuing delay of at least \( C \log_b N \) steps, \( C \geq 1 \), is
\[
\Phi_2 \leq N \left( e^{\log_b N} \right).
\]
The probability \( 1 - \Phi_2 \) can be rewritten as given and the result follows. \( \square \)

Further, since both Valiant and Brebner's, and our analysis consider queuing delays for only one of the two phases of the probabilistic algorithm, the study of Section 3 implies

Proposition 4.1. For any \( N \)-node, base-\( b \) generalized hypercube, the upper bound of Theorem 3.1 for our new probabilistic routing algorithm is better than the corresponding upper bound for the probabilistic routing strategy in [12].

Proof. Our upper bound for probabilistic routing (Theorem 3.1) is better than Valiant and Brebner's bound (Corollary 4.1), provided that \( D > D' \), or, equivalently,
\[
C \ln \left( (C(b+1)e^{1/(C(b+1))^{-1}} - \ln(b) \right) > C \ln \left( \frac{bC}{e} \right) - \ln(b).
\]
Since \( C \geq 1 \) and the natural logarithm (ln) is monotonically increasing, we simplify Eq. (4.2),
\[
C(b+1)e^{1/(C(b+1))^{-1}} > \frac{bC}{e}.
\]
Equivalently we have
\[
e^{1/(C(b+1))} > \frac{b}{b+1}.
\]
Table 1
Probability of larger than $C \log_b N$ delay for $N = 1024$, $C = 2.0$

<table>
<thead>
<tr>
<th>Base $b$</th>
<th>$\Phi_1$</th>
<th>$\Phi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$4.8 \times 10^{-6}$</td>
<td>0.45</td>
</tr>
<tr>
<td>4</td>
<td>$8.3 \times 10^{-4}$</td>
<td>$2.1 \times 10^{-2}$</td>
</tr>
<tr>
<td>32</td>
<td>$2.3 \times 10^{-3}$</td>
<td>$3.3 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

Table 2
Probability of larger than $C \log_b N$ delay for $N = 1024$, $C = 3.0$

<table>
<thead>
<tr>
<th>Base $b$</th>
<th>$\Phi_1$</th>
<th>$\Phi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$7.4 \times 10^{-4}$</td>
<td>$4.9 \times 10^{-8}$</td>
</tr>
<tr>
<td>4</td>
<td>$2.8 \times 10^{-9}$</td>
<td>$2.2 \times 10^{-7}$</td>
</tr>
<tr>
<td>32</td>
<td>$4.1 \times 10^{-7}$</td>
<td>$5.3 \times 10^{-7}$</td>
</tr>
</tbody>
</table>

Table 3
Probability of larger than $C \log_b N$ delay for $N = 4096$, $C = 2.0$

<table>
<thead>
<tr>
<th>Base $b$</th>
<th>$\Phi_1$</th>
<th>$\Phi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$4.2 \times 10^{-7}$</td>
<td>$9.9 \times 10^{-2}$</td>
</tr>
<tr>
<td>4</td>
<td>$2.0 \times 10^{-4}$</td>
<td>$9.7 \times 10^{-3}$</td>
</tr>
<tr>
<td>16</td>
<td>$5.4 \times 10^{-6}$</td>
<td>$1.1 \times 10^{-5}$</td>
</tr>
<tr>
<td>64</td>
<td>$7.6 \times 10^{-4}$</td>
<td>$8.3 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

Since $b \geq 2$ and $C \geq 1$, the left-hand side in Eq. (4.4) is always greater than 1, while the right hand side is always less than 1. Therefore, Eq. (4.2) is always true.

In order to further illustrate our analysis, in Tables 1–3 we compare

- the upper bound for our probabilistic routing algorithm (probability $\Phi_1$ of Eq. (3.4)) versus
- the upper bound (probability $\Phi_2$ of Eq. (4.1)) for the algorithm in [12].

The constant $C$ in the tables refers to a total queuing time during phase A (or phase B) of $C \log_b N$. In each table, $N$ and $C$ are kept constant, and we vary the hypercube base $b$. From these tables and our analysis, we note that, for a given number of nodes $N$, and constant $C$, the probability ($1 - \Phi$) of routing all packets within $C \log_b N$ time is much higher for our routing strategy, especially when the base $b = 2$.

The validity of comparing upper bounds instead of exact probabilities for the two routing algorithms stems from the following three facts.

(1) We first compare the probability of conflict between two packets chosen arbitrarily either from permutations, as in our scheme, or from general relations, as in Valiant and Brebner’s scheme. The probability of conflict for our scheme ($p_1$) is given by Eq. (3.1), while only upper bounds have been considered for the corresponding probability $p_2$ in Valiant and Brebner’s scheme. Nevertheless, it is obvious that
With general relations, we must consider packets heading to different destinations (as in our routing algorithm), and also packets heading to similar destinations. A conflict is much more likely for the latter packets, since they eventually meet at the common destination.

(2) The upper bound of Lemma 1, which is used to evaluate the probability of more than $C \log_2 N$ queuing delay for both algorithms, is not only tight, but is in fact attainable, when all Poisson trials have equal success probabilities (cf. Theorem 5, Eq. (30), and remark of [7]).

(3) Although Chernoff's approximation on the probability of $m = C \log_2 N$ queuing delay is within one order of magnitude from the actual value, the same approximation (with same sample size $N$, and number of successes $m$) is used for both algorithms. Since both the binomial distribution and Chernoff's upper bound approximation are monotonic increasing functions, with increasing number of successes $m$, instead of comparing exact values of the binomial distributions, we can relatively compare their corresponding upper bounds.

We now compare the two algorithms, assuming that the probability of routing in $C \log_2 N$ steps approaches exponentially 1, as $N$ becomes large. In Proposition 4.2 below, we show that the bound on the expected total routing time is again in favor of our probabilistic routing algorithm.

**Proposition 4.2.** The critical value of $C$ for probabilistic routing on the binary hypercube in optimal $C \log_2 N$ time, with an exponentially increasing probability (as $N$ increases), is much smaller for our algorithm than for the strategy proposed in [12].

**Proof.** For the algorithm in [12], from Corollary 4.1, we find that, for all values of $C \geq C_2 = 1.95$, there is a positive constant $D'$, such that for any base $b$, $\Phi_2 \leq e^{-D' \log_b N}$. Then, $C_2 = 1.95$ is the critical value for $C$, and corresponds to $b = 2$; for larger base $b$ this critical value becomes even smaller. Since the value of $C_2$ corresponds to one phase only, and the propagation time is $2 \log_2 N$ in the worst-case, for the binary hypercube there is exponentially increasing probability of finishing in $5.90 \log_2 N$ total routing time.

Similarly, for our routing method, from Theorem 3.1, we can easily check that for all values of $C \geq C_1 = 1.22$, there is a positive constant $D$, such that for any base $b$, $\Phi_1 \leq e^{-D \log_b N}$. Then $C_1 = 1.22$ is the critical value for $C$ for an exponentially increasing probability of finishing in optimal $C_1 \log_b N$ time, and corresponds to $b = 2$. For larger bases $b$ this critical value would become smaller. Adding the propagation time which is at most $2 \log_2 N$, the total routing time becomes $4.44 \log_2 N$. □

5. Concluding remarks

When $b = 2$, it is possible to reduce Eq. (3.1) to a single combinatorial quantity, and obtain better upper bounds on the probability $p_1$ (the bound is proportional
to $\sqrt{\log_2 N/N}$, instead of $\log_2 N/N$). These bounds further reduce queuing delays in Proposition 4.2, and improve our results in Tables 1-3 for $b = 2$.

Our analysis can be extended to prove that the buffer size for our routing algorithm is $O(\log_b N)$ with high probability (same as the routing method in [11, 12]). However, our results can be improved to $O(\sqrt{\log_2 N})$ for the binary hypercube (see the above paragraph).

Another extension to the single-port model is obvious, since only three conditions $(\alpha, \beta, \gamma)$ must be satisfied for a conflict, with the given probabilities.

It is an open question whether a general theorem concerning the asymptotic optimality of our probabilistic routing method on any graph can be proved, in the same sense as Theorem 4.1.

Acknowledgements

We are grateful to Prof. S.K. Dhall, and S. Lakshmivarahan for stimulating discussions during early stages of this research. We also wish to thank the referees for valuable comments and suggestions.

References