Yale University
Department of Computer Science

Equivalence Of Message Scheduling Algorithms For Parallel Communication

Abhiram G. Ranade

YALEU/DCS/TR-512
January 1987

This work has in part been supported by the Office of Naval Research under contracts N00014-84-K-0043 and N00014-86-K-0564. Approved for public release: distribution is unlimited.

Equivalence Of Message Scheduling Algorithms For Parallel Communication *

Abhiram G. Ranade

Yale University
Department of Computer Science
New Haven, Connecticut

Abstract: Parallel communication systems employing randomization have been widely discussed in literature [2,8,9,10,11] for accomplishing arbitrary data permutation on sparse graphs. An important component of such systems is the *message scheduling algorithm* used to select the message to be transmitted along a communication link if there are several waiting. Many of the interesting results in this area [2,6,10] require the use of complicated message scheduling algorithms viz. the message to be transmitted is chosen on the basis of preassigned *priorities.* This increases the hardware cost because of the need to support priority queues.

The main result of this paper (section 3.1) is that for randomized communication on networks like hypercubes, cube connected cycles etc., most of the interesting scheduling algorithms are equivalent to one another. This allows one to construct randomized communication schemes for bounded degree networks that accomplish message delivery in logarithmic time based on simple scheduling algorithms, *without the use of message priorities.* The equivalence of scheduling algorithms also provides better analysis techniques which can be used show that the queueing delay experienced on n dimensional hypercubes must lie in the range $n/log^{1\pm\epsilon}n$ with high probability, for arbitrary $\epsilon > 0$ and sufficiently large n. This bound is useful when the message length is large.

1 Introduction

Algorithms for fast exchange of information between processors are central to the design of large scale parallel processing systems. An important instance of this problem is data permutation, in which each processor sends a single message to an arbitrary processor, with the condition that each processor also receives just one message. In [11] Valiant and Brebner proposed a randomized solution to this problem on the hypercube interconnection

*This work has in part been supported by the Office of Naval Research under contracts N00014-84-K-0043 and N00014-86-K-0564.

network, which was later extended to bounded degree networks like butterflies and shuffle exchange networks by Aleliunas [2] and Upfal [10]. The techniques developed for this problem have also been used to solve harder problems like emulation of PRAMs [6]. In all these cases randomization has so far proved to be better than deterministic methods. For example, randomized algorithms are faster than sorting networks [3], do not need global communication as in Benes networks [4], and are useful for machines with practical sizes, unlike [1,7].

For data permutation, the randomized algorithms of [2,10,11] proceed in two phases. In the first phase the message held by each processor is sent to a randomly chosen intermediate destination processor. In the second phase, the message gets redirected to its proper destination. The communication algorithms required to perform this movement can be broken down into message routing and scheduling algorithms. Given the topology of the interconnect and the destination of the message, the routing algorithm determines a path which the message must follow. The scheduling algorithm is used to schedule message movement along the path. It ensures that two messages are not scheduled for movement along the same communication link at the same time, and thus determines the message to schedule if there are several candidates.

Many different scheduling algorithms are possible, e.g. one may schedule messages arriving earlier before the ones arriving later, the scheduling order may depend upon the distance a message is yet to travel, etc. It is clear that different scheduling algorithms have different implementation costs, depending upon their sophistication. The scheduling algorithms of [2,6,10] associate *priorities* with messages, which are used to resolve contention. Implementing this requires one to maintain priority queues at each node in the network.

The main result of this paper is that a large class of scheduling algorithms have identical performance with randomized communication, and the algorithms in this class are thus equivalent. This class contains most of the interesting scheduling algorithms, e.g. priority based, "last/first in first out" and others. Thus it is possible to attain the performance of [2,10] without using message priorities, simply by using, for example a first in first out scheduling algorithm at each node. The equivalence also makes it possible to construct complex scheduling algorithms which are especially easy to analyze and reveal the structure of parallel communication. This leads to a very tight bound on the queueing delay experienced by messages in hypercubes. For an n dimensional hypercube, this delay is shown to be in the range $n/log^{1\pm\epsilon}n$ with high probability, for arbitrary $\epsilon > 0$. For long messages, the net delay is thus $O(Ln/log^{1-\epsilon}n)$ rather than $O(Ln)$ as shown in [11].

A note about the proof techniques used in this paper and their evolution from Valiant and Brebner's original paper [11] is in order. Their methods only used spatial information viz. if a message is delayed by d there must be at least d messages touching its path. The bounds obtained by Aleliunas[2] and Upfal[10] were based on the use of temporal information, i.e. if one message delays another, then it is also necessary that they be in the same message queue at the same time. The analysis in this paper also incorporates information about the scheduling algorithm i.e. not only is it necessary that the two messages be in the same queue at the same time, but further the scheduling algorithm used at the node must give preference to the proper message. It is felt that the techniques used in this paper may be useful for the analysis of other problems involving randomized communication.

1.1 Outline

This document is organized as follows. Section 2 formally defines the communication model and the message permutation problem. Section 3 defines *non predictive* scheduling algorithms and *tree based routing algorithms* and proves the central theorem viz. if tree based routing algorithms are used, then all non predictive scheduling algorithms have the same performance. Section 4 constructs a scheduling algorithm called *random priority based* scheduling algorithm which is shown to have the same performance as non predictive scheduling algorithms, but which is especially suitable for analysis. Section 5 analyzes the random priority based scheduling algorithm on hypercubes, and section 6 on butterflies and the cube connected cycles. The results are are valid for all non predictive scheduling algorithms, because of the central theorem.

2 The Communication Model

In this paper, the function log is used to denote the logarithm of a number to the base 2. Also, the bits in an n bit binary number are numbered 0 through $n-1$ from the most significant bit through the least significant.

A parallel communication system (PCS) is a five-tuple $(\mathcal{G}, \mathcal{R}, \mathcal{I}, \mathcal{Q}, \mathcal{D})$ having the following features:

1. There is a directed graph $\mathcal{G} = (V, E)$ where V is a set of nodes, and E the set of edges between them. Each node has queues for holding messages, one queue for every outgoing edge. The queues have infinite length. E_q denotes the set of queues.

2. There is a set M of messages in the system at all times. The **initial message distribution** $\mathcal{I}$ indicates the number of messages initially at each node: $\mathcal{I} : V \rightarrow \mathrm{N}$ where N is the set of natural numbers. The messages are labelled using a canonical ordering which depends only upon their initial positions. The message **destination list** $\mathcal{D} : M \rightarrow V$ indicates the destination for each message.

3. At every integral instant (i.e. $t = 0, 1, 2 \ldots$) each message is at some node. During each unit interval (i.e. time period $(t, t+1)$ for integral t), each edge can transmit one message in the sense of its direction.

4. At the end of every unit interval, or at system initialization the messages at a node may consist of those that have just arrived from a neighbouring node, others that were waiting at one of the queues during the last interval, others that originate at that node, and others that were already finished (i.e. had arrived at their final destinations) before this interval. At the end of every integral instant, and also at system initialisation the **Routing algorithm** $\mathcal{R}$ decides for every unfinished message what queue it is to be in. In general the routing algorithm can use all possible information about the messages being routed i.e. it has access to $\mathcal{I}$ and $\mathcal{D}$.

5. At the start of each unit interval, each queue contains a set of messages, and the **Scheduling Algorithm** $\mathcal{Q}$ determines which one is to be at the head of the queue, provided the queue is non-empty. In general the scheduling algorithm may use all available information i.e. $\mathcal{I}$, $\mathcal{D}$, $\mathcal{R}$ for message selection. During the unit interval, the message at the head of every non-empty queue is transmitted along the associated directed edge to the neighbouring node.

The **route** is the path a message follows in $\mathcal{G}$. Its **delay** is the total time it waits unserved in the queues. A message A is said to **delay** a message B at time t if at time t A and B are present in the same queue at some node and A is chosen for transmission. The **(maximal) delay** of the parallel communication system is the maximum delay suffered by any message. A message is said to touch a path in $\mathcal{G}$ if its route shares at least one edge with the path. The **state** S_t of a Parallel Communication Scheme is a specification of the node at which a message is at a time t, i.e $S_t : M \rightarrow V$. Thus given $\mathcal{G}$, $\mathcal{R}$, $\mathcal{Q}$, $\mathcal{I}$, $\mathcal{D}$ it is possible to determine S_t for all t. Note that S_0 depends only upon $\mathcal{I}$.

Notice that the above description allows a very wide range of routing algorithms and scheduling algorithms. Also the above does not describe *tickets* for messages, these deal with the implementation of a PCS rather than its functional specification.

2.1 The routing problem

The following message routing problem is considered. Initially the vertices in $\mathcal{G}$ are assumed to have a number T of messages specified by an input distribution $\mathcal{I}$. The function $H : M \rightarrow V$ gives the destination for each message. The problem considered in this paper is to deliver these messages to their destinations using some routing algorithm $\mathcal{R}$ and some scheduling algorithm $\mathcal{Q}$. Let $\mathcal{O}$ represent the output distribution function for H, i.e. the number of messages received by each output, $\mathcal{O} : V \rightarrow \mathrm{N}$. It is easy to see that $\mathcal{O}$ can be computed from H. The following two phase algorithm is analyzed:

Phase 1: Each message is sent to an independently randomly chosen destination. Thus, this corresponds to a parallel communication system $(\mathcal{G}, \mathcal{R}, \mathcal{I}, \mathcal{Q}, \mathcal{D}_1)$ where $\mathcal{D}_1$ is randomly chosen.

Phase 2: Each message is forwarded from the destination chosen in phase 2 to the appropriate destination indicated by H. Thus in this phase, the source for each message is randomly chosen, given the destinations distributed as per $\mathcal{O}$. Let $\mathcal{I}_2$ and $\mathcal{D}_2$ represent the initial distribution and the destination list randomly chosen adhering to this restriction. Thus phase 2 corresponds to a parallel communication system $(\mathcal{G}, \mathcal{R}, \mathcal{I}_2, \mathcal{Q}, \mathcal{D}_2)$.

3 Restricted communication model

The communication systems considered in this paper have *tree based* routing algorithms and *non predictive* scheduling algorithms, defined below.

Definition 1 *Let $\mathcal{G}$ =(V,E) be the graph of a parallel communication system, with routing algorithm $\mathcal{R}$. Let $\mathcal{G}'$ be a graph with the vertex set $V \bigcup E_q$. Let g and g' be two distinct vertices in $\mathcal{G}'$. Then there is an edge in $\mathcal{G}'$ from g to g' iff:*

1. *$g \in V$ and $g' \in E_q$ and g' is the first queue on the path of a possible message originating at g, given $\mathcal{R}$.*
2. *$g \in V$ and $g' \in V$ and g and g' are consecutive queues on the path of some possible message as per $\mathcal{R}$.*
3. *$g \in E_q$ and $g' \in V$ such that g is the last queue on the path of some message terminating in g' as per $\mathcal{R}$.*

Then $\mathcal{G}'$ is called the **resource graph** *of $\mathcal{R}$.*

The resource graph of of a parallel communication system has been called the *channel dependency graph* in [5] in connection with deadlock analysis. Indeed, if the resource graph is acyclic, then the parallel communication system is deadlock free, even if each queue is of unit length [5]. The motivation for the present paper is different, however.

Definition 2 *Let $\mathcal{R}$ be a routing algorithm for a graph $\mathcal{G} = (V, E)$ with E_q being the set of queues. Let $\mathcal{G}'$ be its resource graph. Suppose that for all $v, w \in V$, there is exactly one path in $\mathcal{G}'$ from v to w passing through only vertices in E_q. Then $\mathcal{R}$ is called a* **tree-based** *routing algorithm.*

Note that this is much stronger than oblivious routing algorithms [11]. The interesting property of tree based routing algorithms is that if a queue contains messages which are performing random walks in the network, then the scheduling algorithm need not distinguish between them. This idea is formalised below.

Definition 3 *A scheduling algorithm is said to be* **non predictive** *if the choice of a message at time t depends only upon the state S_i for all $0 \leq i \leq t$. (In particular the choice is independent of $\mathcal{D}$)*

The state information allows the scheduling algorithm to compute how the message reached the queue, i.e. its source, and also how long it waited in different queues. Thus a "first in first out" scheduling algorithm is non predictive. Another example of non predictive scheduling algorithms are those based on *priority.* These scheduling algorithms do not even use the state information.

3.1 Equivalence of non predictive scheduling algorithms

For analyzing characteristics like maximal message delay etc. it is useful to disregard message identity and consider "message traffic". The notion of a *population descriptor* allows one to do this.

Definition 4 *Given a parallel communication system $(\mathcal{G}, \mathcal{R}, \mathcal{I}, \mathcal{Q}, \mathcal{D})$ let $\mathcal{G}' = (V', E')$ be its resource graph. Let v be a vertex in V' at time t such that there is at least one message at v at time t. Then the* **population descriptor** *$\mathcal{P}$ indicates the vertex $w \in V'$ to which a message moves from v at time t. Thus $\mathcal{P} : V' \times \mathrm{N} \rightarrow V' \bigcup \{\perp\}$ where $\mathcal{P}$ evaluates to $\perp$ (undefined) if there is no message in v at time t.*

Lemma 1 *The population descriptor $\mathcal{P}_1$ for $(\mathcal{G}, \mathcal{R}, \mathcal{I}, \mathcal{Q}, \mathcal{D}_1)$ is different from that for the population descriptor $\mathcal{P}_2$ for $(\mathcal{G}, \mathcal{R}, \mathcal{I}, \mathcal{Q}, \mathcal{D}_2)$, for $\mathcal{D}_1 \neq \mathcal{D}_2$.*

Proof: Let t be the earliest at which the position of a message is different in the two systems. And let m be such a message. Let $\mathcal{G}'$ be the resource graph for $\mathcal{R}$. Then at time $t-1$ the message m was at the same $v \in \mathcal{G}'$ in both systems, and further both the sytems had the same messages at v at time $t-1$. Because $\mathcal{Q}$ is non predictive, m must have been selected in both the systems in v at time t and v. But since its position at time t differs in the two systems $\mathcal{P}_1(v,t) \neq \mathcal{P}_2(v,t)$ ▮

Theorem 1 *Let $\mathcal{P}$ be the population descriptor for $(\mathcal{G}, \mathcal{R}, \mathcal{I}, \mathcal{Q}_1, \mathcal{D}_1)$. Then for every $\mathcal{D}_1$ there is a unique $\mathcal{D}_2$ such that for any non predictive $\mathcal{Q}_2$, the population descriptor of $(\mathcal{G}, \mathcal{R}, \mathcal{I}, \mathcal{Q}_2, \mathcal{D}_2)$ is $\mathcal{P}$.*

Proof: The following is a way of constructing $\mathcal{D}_2$. Given $\mathcal{I}$ it is possible to construct S_0 for $(\mathcal{G}, \mathcal{R}, \mathcal{I}, \mathcal{Q}_2, \mathcal{D}_2)$. Given S_0 and $\mathcal{Q}_2$ it is possible to determine the messages selected at each queue. But for every message m selected at a queue q, the destination $\mathcal{P}(q,t)$ is known. Because $\mathcal{R}$ is a tree based routing algorithm, this forms a valid *partial message path. Thus it is possible to compute S_1, whereupon the procedure is repeated, until the the entire path for each message is known. This gives $\mathcal{D}_2$.* ▮

Theorem 1 indicates that different non predictive scheduling algorithms do not generate fundamentally new data movement patterns or population descriptors, but choose from a single set of data movement patterns in a different order.

Theorem 2 *For phases 1 and 2, the probability of having a given maximal delay d is identical for all non predictive scheduling algorithms.*

Proof: For phase 1, the initial message distribution $\mathcal{I}$ is h for every node. For a scheduling algorithm $\mathcal{Q}_1$ let $\mathcal{D}_1$ be a destination lists that results in a delay d. But by theorem 1 it is possible to construct $\mathcal{D}_2$ for every scheduling algorithm $\mathcal{Q}_2$ such that it has the same population descriptor and hence the same delay d. But since the destination list is chosen at random, the probability of choosing $\mathcal{D}_1$ is the same as that of choosing $\mathcal{D}_2$. But $\mathcal{D}_2$ is unique for every $\mathcal{D}_1$, and thus the probability of choosing a destination list having maximal delay d with $\mathcal{Q}_1$ is the same as that with $\mathcal{Q}_2$.

For phase 2, let $\mathcal{D}_1$ be a destination list that results in a delay d for a scheduling algorithm $\mathcal{Q}_1$. The number of messages received by each output is h. By theorem 1 it is possible to construct $\mathcal{D}_2$ for every scheduling algorithm $\mathcal{Q}_2$ such that it has the same population descriptor. Thus it must also have the same delay d and also the same number of messages destined for each output. Thus with $\mathcal{Q}_2$ and $\mathcal{D}_2$ each output receives exactly h messages. But for phase 2, a source node is chosen at random for each message, and thus the probability of choosing $\mathcal{D}_1$ is the same as that of choosing $\mathcal{D}_2$. Again, because there is a unique $\mathcal{D}_2$ for every $\mathcal{D}_1$, it follows that the probability of having a maximal delay of d is the same for any two non predictive scheduling algorithms $\mathcal{Q}_1$ and $\mathcal{Q}_2$. ▮

4 Priority based scheduling algorithms

The scheduling algorithm analyzed in this paper is called *random priority based* scheduling algorithm, which is randomly chosen from *priority based* scheduling algorithms.

Definition 5 *Let $\mathcal{Q}$ be a scheduling algorithm for a parallel communication system and let T be the number of messages. Let each message be assigned a unique integer in the range $1..T$ called* priority. *Further, assume that the priority of messages does not change with different destination lists. $\mathcal{Q}$ is said to be a* **priority based** *scheduling algorithm if for every set M' of messages in any queue q at any time t, the message selected is the one having the greatest priority.*

It is easy to see that priority based scheduling algorithms are non predictive. They have another interesting property.

Theorem 3 *Consider a parallel communications* $(\mathcal{G}, \mathcal{R}, \mathcal{I}, \mathcal{Q}, \mathcal{D})$ *such that* $\mathcal{Q}$ *is a priority based scheduling algorithm. Let* M *be a message with delay at least* d*. Then there exists a sequence of messages* $M_1..M_d$ *such that with* $M = M_0$, M_i *delays* M_{i-1} *in queue* Q_i*. Further the path from the source of* M_d *to the destination of* M *passes through queues* $Q_d..Q_1$ *in that order.*

Proof: *This can be proved by assuming the following induction hypothesis:*

> *There exists a sequence of messages* $M_0..M_k$ *such that* M_i *delays* M_{i-1} *at switch* Q_i*. The path from the input of* M_i *to the output of* M *passes through queues* Q_i *through* Q_1 *in that order. Further the the delay of* M_i *when it leaves* Q_i *is at least* $d - i$*.*

Let M_{i+1} *be the last message to delay* M_i *before* M_i *delayed* M_{i-1}*. Let this happen at queue* Q_{i+1}*. Then both* Q_i *and* Q_{i+1} *are on the path of* M_i*, with* Q_{i+1} *possibly on the input side. The delay of* M_i *is at least* $d-i$ *after being delayed by* M_{i+1}*, and hence the delay of* M_{i+1} *when it leaves* Q_{i+1} *must be at least* $(d-i)-1 = d-(i+1)$*. It is easy to see that queues* Q_{i+1} *through* Q_1 *lie on the path from the source of* M_{i+1} *to the destination of* M_0 *in the right order. The base case can be proved in a similar manner by constructing* M_1*.* ▮

4.1 The Random priority based scheduling algorithm

Definition 6 *Consider a parallel communication system with* T *messages. Suppose each message is assigned a randomly chosen unique* priority *in the range* $1..T$*. Let* $\mathcal{Q}$ *be the scheduling algorithm such that if* M' *is the set of messages in any queue* Q *at any time* t*, then the message selected is the one having the greatest priority. Then* $\mathcal{Q}$ *will be called the* **random priority based** *scheduling algorithm.*

Because each message is assigned a unique priority, then with T messages, the total number of priority assignments is $T!$. All priority assignments are equally probable.

Theorem 4 *The random priority based scheduling algorithm has the same probability of having a delay* d *for phase 1 and phase 2 as any non predictive scheduling algorithm.*

Proof: Consider any particular priority assignment. The scheduling algorithm obtained using this assignment is a priority based scheduling algorithm, which is non predictive. Thus randomly choosing priorities is equivalent to

```
for every message $M$ at node $P$ at time $t$
  if $\mathcal{D}(M) \neq P$ then        {if the message has not reached its destination}
    do
    Let $i$ be the smallest integer such that $\mathcal{D}(m) \oplus P = d_0..d_{n-1}$ and $d_i$ is 1
    Place $M$ in queue $i|p_0..p_{n-1}$
    od
```

Figure 1: Hypercube routing algorithm

chosing a non predictive scheduling algorithm at random. But all non predictive scheduling algorithms have the same probability of having a delay d. ▮

5 Randomized Routing On Hypercubes

The routing algorithm of section 2.1 is now analyzed for binary hypercubes.

A binary hypercube of n dimensions has $N = 2^n$ processors numbered 0 through $N-1$. There is a bidirectional edge connecting each processor $P = p_0p_1..p_{n-1}$ where p_i are the bits in the binary representation of P, to processor $p_0p_1..\bar{p_i}..p_{n-1}$ for all $0 \leq i < n$. The queue associated with the edge from $p_0..p_i..p_{n-1}$ to $p_0..\bar{p_i}..p_{n-1}$ is numbered $i|p_0..p_i..p_{n-1}$. Also $dim(i|p_0..p_{n-1}) = i$.

The routing algorithm shown in figure 1 is used. This routing algorithm is used in [11] and is well studied. It is easily seen that it is tree based. The random priority based scheduling algorithm is used in both phases, but as discussed earlier, the conclusions is applicable to all non predictive scheduling algorithms.

The case of h-permutations is considered, i.e. each node in the hypercube is the source and the destination for exactly h messages.

5.1 Analysis of phase 1

For phase 1, the destination for each message is uniformly randomly chosen from the N different destinations. There are $(hN)!$ different ways of ordering hN messages, and one of these priority orders is chosen at random. Thus phase 1 has $N^{hN}(hN)!$ different outcomes. Of these, the outcomes which have a delay of d must satisfy theorem 3. This gives a way of estimating their number.

For what follows, let $Pr(E)$ denote the probability of an event E, and δ the maximal delay.

Lemma 2 *For phase 1, if the random priority based scheduling algorithm is used then*

$$Pr(\delta \geq d) \quad \leq \quad \frac{2N^2\binom{n-1+d}{d}(\frac{h}{2})^{d+1}}{(d+1)!}$$

Proof: If a given outcome has a message with delay at least d, then there must be a distinguished path and messages $M_0..M_d$ with increasing priorities and queues $Q_1..Q_d$ satisfying theorem 3. Each such outcome satisfying theorem 3 can be constructed as follows:

1. *Select the source and the destination nodes for the distinguished path. This can be done in in N^2 ways.*

2. *Select $Q_1..Q_d$ from the queues on the distinguished path. There can be at most n queues on the distinguished path. Because $Q_1..Q_d$ occur along the path in that order starting from the destination, this is equivalent to choosing d objects from n allowing repetition. This can be done in $\binom{n-1+d}{d}$ ways. Note that this is equivalent to specifying the distance of each Q_i from the destination of the distinguished path.*

3. *Select each message $M_0..M_d$ and their destinations. For $i > 0$ M_i is known to pass through $Q_i = k|p_0p_1..p_{n-1}$. All the $h2^k$ messages originating at nodes of the form $**..*p_k..p_{n-1}$ can reach Q_i. But because these messages pass through Q_i, they can reach only reach destinations of the form $p_0..p_{k-1}\bar{p}_k**..*$ i.e. at most 2^{n-k-1} destinations. Thus the total number of ways in which a message M_i and its destination can be chosen is at most $h2^k2^{n-k-1} = h2^{n-1} = hN/2$. M_0 is known to have the same destination as the distingushed path. Its source can be chosen in at most hN ways.*

4. *The above leaves behind $hN - d - 1$ messages, and the destination for each can be chosen in N ways.*

5. *The priorities for $M_0..M_d$ can be assigned arbitrarily, provided they are increasing and in the range $1..hN$. This can be done in $\binom{hN}{d+1}$ ways. The remaining messages must be assigned the remaining priorities, and this can be done in $(hN - d - 1)!$ ways.*

Thus the total number of outcomes which can satisfy theorem 3 must be at most:

$$N^2\binom{n-1+d}{d}(hN/2)^d hN N^{hN-d-1}\binom{hN}{d+1}(hN-d-1)!$$
$$\leq \frac{2N^2\binom{n-1+d}{d}(\frac{h}{2})^{d+1}N^{hN}(hN)!}{(d+1)!}$$

Thus the number of outcomes having maximal delay at least d is also less than the above expressions. Because there are $N^{hN}(hN)!$ outcomes to phase 1, the bound follows. ∎

Theorem 5 *The bound of lemma 2 can be improved to:*

$$Pr(\delta \geq d) \quad \leq \quad \frac{2N\binom{n-1+d}{d}(\frac{h}{2})^{d+1}}{(d+1)!} \tag{1}$$

$$\leq \frac{N\binom{n+d+1}{d+1}\left(\frac{h}{2}\right)^{d+1}}{(d+1)!}$$

Proof Outline: It is easy to see that in the characterization of outcomes possibly having maximal delay at least d in lemma 2, it is not necessary to select the source for the distinguished path. If M_i and the distance of Q_{i+1} from Q_i is known it is possible to compute Q_{i+1} and similarly the entire distinguished path. The second expression results using $2\binom{n-1+d}{d} < \binom{n+d+1}{d+1}$.

5.2 Analysis of Phase 2

For phase 2, the final message distribution, rather than the initial is known. Because each output is known to receive h messages, the number of ways in which a message destination can be chosen so that the message passes through a queue in stage k is $2^{n-1-k}h$. But its source is randomly distributed over all nodes. But because only messages from 2^k nodes can reach the given queue, the total number of ways in which M_i can be chosen is $2^k 2^{n-1-k}h = hN/2$. Thus phase 2 has similar behaviour, and that the same bounds apply.

5.3 Permutation Routing

The case $h = 1$ is considered:

$$Pr(\delta \geq d) \leq \frac{N\binom{n+d+1}{d+1}}{(d+1)!2^{d+1}}$$

Using $\log\binom{n}{r} \leq r\log ne/r$ and $log n! \geq n\log n/e$, and with $c = d+1$ it follows:

$$\begin{aligned}\log Pr(\delta \geq c) &\leq n + c\log\frac{(n+c)e}{c} - c\log\frac{c}{e} - c \\ &\leq n - c\log\frac{2c^2}{(n+c)e^2}\end{aligned}$$

Let $c = n/\log^{1-\epsilon} n$. Then, for large enough n:

$$\begin{aligned}\log Pr(\delta \geq n/\log^{1-\epsilon} n) &\leq n - \frac{n}{\log^{1-\epsilon} n}\log\frac{2(n/\log^{1-\epsilon} n)^2}{(n + (n/\log^{1-\epsilon} n)e^2)} \\ &\leq n - \frac{n}{\log^{1-\epsilon} n}\log\sqrt{n} \\ &= -(\frac{n}{2}\log^{\epsilon} n - n)\end{aligned}$$

Thus asymptotically, the probability that the delay is greater than $\frac{n}{log^{1-\epsilon}n}$ tends to zero, for arbitrary $\epsilon > 0$.

5.4 High probability asymptotic lower bound for the delay

Consider the processors in the hypercube numbered $**..*p_k..p_{n-1}$. The 2^k messages originating in these processors can reach queues $k|**..*p_k..p_{n-1}$. There are 2^k different queues, and the probability of any given message reaching a given such queue is $2^{-(k+1)}$. Thus the probability of all the messages reaching any single queue from this set is $2^k(2^{-(k+1)})^{2^k}$. Let $c = 2^k$. Then this probability is $c(1/2c)^c = c^{1-c}2^{-c}$. This is the probability of having c messages pass through a single queue, for a given set of messages. But there is such a set for each choice of $p_k..p_{n-1}$, i.e. there are N/c such sets. Then the probability of not having a message delayed by c in the entire hypercube i.e.

$$\begin{aligned} 1 - Pr(\delta \geq c) &< (1 - c^{1-c}2^{-c})^{N/c} \\ &= (1 - c^{1-c}2^{-c})^{c^{c-1}2^cN/(2c)^c} \\ &\leq e^{-N/(2c)^c} \end{aligned}$$

Because $(1 - h^{-1})^h$ is at most $1/e$ for $h > 1$. With $c = n/(\log^{1+\epsilon} n)$ it is easy to see that for sufficiently large n the above expression can be made as small as necessary. Thus, asymptotically, the probability that the delay is at least $n/(\log^{1+\epsilon} n)$ can be made as close to 1 as desired, for arbitrary $\epsilon > 0$.

Thus for sufficiently large n, with probability tending to 1, the maximum delay for phase 1 or phase 2 lies within $\frac{n}{\log^{1\pm\epsilon} n}$ for arbitrary $\epsilon > 0$.

5.5 Analysis of h-permutation routing

$$\begin{aligned} Pr(\delta \geq d) &\leq \frac{N\binom{n+d+1}{d+1}(\frac{h}{2})^{d+1}}{(d+1)!} \\ &\leq \frac{N\binom{n+kh}{n}(\frac{h}{2})^{kh}}{(kh)!} \end{aligned}$$

with $kh = d + 1$, where $kh \geq n$. Using $\log\binom{n}{r} \leq r \log ne/r$ and $\log n! \geq n \log n/e$ it follows:

$$\begin{aligned} \log Pr(\delta \geq kh) &\leq n + n\log\frac{(n+kh)e}{n} + kh\log\frac{h}{2} - kh\log\frac{kh}{e} \\ &\leq -(kh\log\frac{2kh}{he} - n\log\frac{2e(n+kh)}{n}) \\ &\leq -(kh\log\frac{2k}{e} - n\log 2e - n\log(1+\frac{kh}{n})) \\ &\leq -(kh\log\frac{2k}{e} - kh\log(2e)^{\frac{n}{kh}} - kh\log(1+\frac{kh}{n})^{\frac{n}{kh}}) \\ &\leq -(kh\log\frac{2k}{e} - kh\log(2e)^{\frac{1}{k}} - kh\log(1+k)^{\frac{1}{k}}) \end{aligned}$$

Because $(1+x)^{-x}$ is a decreasing function for $x > 1$. Thus

$$\log Pr(\delta \geq kh) \leq -(kh\log\frac{2k}{e(2e(1+k))^{1/k}})$$

It is easy to see that $k \geq 6$ gives $\log Pr(\delta \geq kh) < -kh$. But $h > n$. Thus

$$Pr(\delta \geq kh) \leq 2^{-kn} = N^{-k}$$

6 Routing on bounded degree networks

A butterfly routing network is considered. Butterflies are very well studied and their relationship to hypercubes is well known. For this reason the following discussion is kept very brief.

An n stage butterfly has $n2^n$ nodes, with $N = 2^n$ nodes in each stage. Each node in stage i is numbered $i|P$ where P is in the range 0 through $N-1$. There is an edge from $i|p_0..p_{n-1}$ to nodes $j|p_0..p_j..p_{n-1}$ and $j|p_0..\bar{p_j}..p_{n-1}$, where $j = i + 1 \bmod p$. It is assumed that each node sends and receives exactly 1 message.

The following three phase algorithm is used:

Phase 1 The messages are first moved to stage 0, i.e. processor $i|P$ sends its message to processor $0|P$.

Phase 2 Messages destined for processor $i|P$ are sent to $0|P$.

Phase 3 Messages destined for processor $i|P$ are sent to $i|P$ from $0|P$.

It is easy to see that phases 1 and 3 require $n-1$ cycles each, and phase 3 can be implemented by using randomized communication. In fact, because of the well known transformations between hypercube algorithms and butterfly algorithms, it may be observed that the analysis of section 5.5 applies with minor modifications.

It should be mentioned here that the above three phase algorithm completes in logarithmic time without the use of message priorities.

Acknowledgements

I am very grateful to Lennart Johnsson for numerous discussions and for his encouragement. I would also like to thank Sandeep Bhatt for many discussions.

References

[1] M. Ajtai, J. Komlos, and E. Szemeredi. An O(n log n) sorting network. In *Proceedings of STOC 83*, pages 1–10, 1983.

[2] R. Aleliunas. Randomized parallel communication. In *PODC*, pages 60–72, 1982.

[3] K. Batcher. Sorting networks and their applications. In *AFIPS Spring Joint Comp. Conf.*, pages 307–314, 1968.

[4] V. Benes. *Mathematical theory of connecting networks and telephone traffic.* Academic Press, 1965.

[5] William Dally and Charles Seitz. *Deadlock Free Message Routing in Multiprocessor Interconnection Networks.* Technical Report 5206:TR:86, California Institute of Technology, 1986.

[6] Anna Karlin and Eli Upfal. Parallel hashing - an efficient implementation of shared memory. In *Proceedings of STOC 86*, 1986.

[7] F. T. Leighton. Tight bounds on the complexity of sorting. In *Proceedings of STOC 84*, pages 71–80, 1984.

[8] Nicholas Pippenger. Parallel communication with limited buffers. In *Proceedings of FOCS 84*, pages 127–136, 1984.

[9] Stefan Reisch and Georg Schnitger. Three applications of kolmogorov-complexity. In *Proceedings of FOCS 82*, pages 45–52, 1982.

[10] E. Upfal. Efficient schemes for parallel communication. In *PODC*, pages 55–59, 1982.

[11] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of STOC 81*, pages 263–277, 1981.