

# **Fast Computation Using Faulty Hypercubes**

-

**Johan Hastad  
Tom Leighton  
Mark Newman**

Presented by: Joel Young

4 April 2000

Proceedings of the Twenty First Annual ACM  
Symposium on Theory of Computing  
May 14 - 17, 1989  
Seattle, WA USA

[http://www.acm.org/pubs/citations/  
proceedings/stoc/73007/  
p251-hastad/index.html](http://www.acm.org/pubs/citations/proceedings/stoc/73007/p251-hastad/index.html)

# Introduction

Large hypercube systems **will** experience faults. The systems need to be able to continue to operate despite faults.

There are two general strategies, one can develop fault tolerant special purpose algorithms, or one can make the system itself fault tolerant.

This paper presents special purpose algorithms for routing on a faulty hypercube as well as an algorithm to simulate a perfect cube with a faulty one.

Outline:

- The paper's contributions
- The computational model used
- Some details of one contribution: constant delay reconfiguration

## Interesting Contributions

- An  $O(\log N)$  time adaptive routing algorithm with offset routing.
- An improvement to Rabin's information dispersal with redundancy approach using a simpler algorithm.
- Constant delay embedding of a virtual  $N$ -node hypercube in a hypercube with randomly distributed node and edge faults with high probability.

“...simple, needs only local control and runs in only a polylogarithmic number of steps.”

## Model and Notation

$N$ -node hypercube  $H_n$  with dimensionality  $n = \log N$ .

Nodes are labeled by  $n$ -bit binary strings. Assume a fixed lexicographic ordering of the nodes by their labels.

Nodes are connected by an edge if the associated strings differ in one bit. If the differing bit is the  $i$ th bit ( $1 \leq i \leq n$ ) then the associated edge is called a dimension  $i$  edge.

The neighbor of node  $v$  on dimension  $i$  is denoted  $v^i$ .

$v^{i_1 i_2 \dots i_k}$  is the node reachable from  $v$  by crossing dimensions  $i_1, i_2, \dots, i_k$ .

Unbounded buffers.

Fault model: nodes and edges of  $H_n$  fail independently with probability  $p < 1$ . Most of the analysis though assumes  $p < .1$ .

**Not a worse case analysis**

# Constant Delay Reconfiguration

To get an *embedding* with constant delay, the *load*, *dilation* and *congestion* must all be constant.

**Definition 1. [embedding]** *An embedding of a virtual fault-free hypercube  $H'_n$  into a faulty hypercube  $H_n$  is a map  $\phi : H'_n \rightarrow H_n$  mapping nodes of  $H'_n$  to functioning nodes of  $H_n$  and edges of  $H'_n$  to functioning paths of  $H_n$ . A path is functioning if all nodes and edges on the path are fault-free.*

**Definition 2. [load]** *The load of an embedding is the maximum number of nodes of  $H'_n$  mapped to a single node of  $H_n$ .*

**Definition 3. [dilation]** *The dilation of an embedding is the length of the longest path in  $H_n$  corresponding to an edge in  $H'_n$ .*

**Definition 4. [congestion]** *The congestion of an embedding is the maximum number of paths that use a single edge of  $H_n$ .*

## The Task

Find an *embedding* of  $H'_n$  in the faulty  $H_n$  with constant *dilation*, *load* and *congestion*, then  $H_n$  will be able to simulate  $H'_n$  with constant slowdown.

1. Each node (live and dead) finds a live neighbor to simulate it.
2. Each pair of nodes simulating neighbors finds a live path between them of length five so that no more than a constant number of these paths congest any edge.

Let  $A_p$  and  $s_p$  be constants which depend only upon probability  $p$  of failure.

**Definition 5. [(un)saturated]** *Call a node unsaturated if it is live and it has been assigned to simulate fewer than  $A_p$  of its neighbors. Otherwise it is saturated.*

# Simulating Faulty Nodes

For  $i = 1$  to  $s_p n$

For each unassigned node  $w$

$w$  picks one of its neighbors uniformly

Each unsaturated node  $v$  agrees to simulate as many nodes as it can without exceeding its capacity, giving preference to lower labeled nodes

All excess nodes remain unassigned

This algorithm never assigns a saturated node to simulate another node, thus no node simulates more than  $A_p$  nodes giving a constant load embedding.

For ease of analysis an iterative version of the above algorithm is used.

**Claim 1.** *With high probability no nodes are ever dedicated during the iterative algorithm below thus with high probability the iterative algorithm produces the same result as the original above.*

## Simulating Faulty Nodes

**Definition 6. [dedicated]** *Let  $\alpha_p$  be dependent only on  $p$ . If a node  $w$  has less than  $\alpha_p n$  unsaturated neighbors to choose from during its turn, select an arbitrary set of saturated neighbors as dedicated to  $w$  during its turn. A dedicated node, if selected, will agree to simulate  $w$  even though it is saturated.*

The iterative algorithm:

For  $i = 1$  to  $s_p n$

For unassigned nodes  $w$  in lexicographic order

If  $w$  has fewer than  $\alpha_p n$  unsaturated neighbors  
arbitrarily dedicate enough (saturated)  
neighbors

$w$  picks one of its neighbors uniformly

If the chosen node is unsaturated or dedicated

$w$  is assigned to that node

Else  $w$  remains unassigned

# Simulating Faulty Nodes

**Lemma 1.** *With high probability all nodes have been assigned after  $s_p n$  steps for sufficiently large  $s_p$ .*

1. each node always has at least  $\alpha_p n$  neighbors which will simulate it if chosen
2. thus probability that a given node is assigned during a step is at least  $\alpha_p$  independent of what has occurred in previous steps.
3. thus probability a node is unassigned after  $s_p n$  steps is no more than  $(1 - \alpha_p)^{s_p n} < \frac{1}{N^k}$  given  $s_p > \frac{k}{\alpha_p}$

## Simulating Faulty Nodes

**Lemma 2.** *For  $p < 1 - \sqrt[4]{0.5} \approx 0.16$ , there exists an  $\epsilon_p$  such that with high probability each node has at least  $\epsilon_p n$  live neighbors.*

**Lemma 3.** *Given a failure rate  $p$ , with high probability a given node  $v$  never has fewer than  $\alpha_p n$  unsaturated neighbors available during the iterative algorithm for  $\alpha_p = \frac{\epsilon_p}{2}$ .*

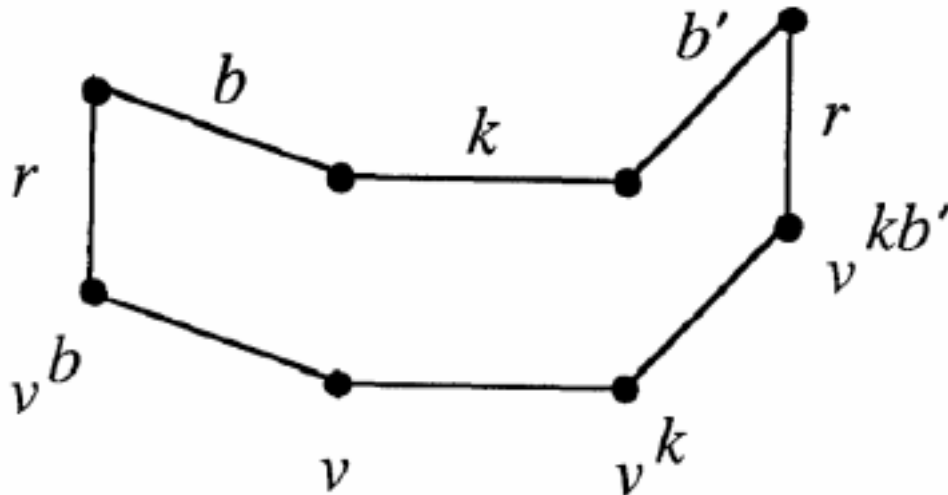
1. Since the iterative algorithm behaves identically (whp) to the original algorithm and
2. with high probability successfully assigns each node to a neighbor and
3. the original algorithm never requires any node to simulate more than  $A_p$  nodes,

We conclude that the original algorithm achieves a constant load embedding with high probability.

## Assigning Paths to Simulate Edges

Now we need to find paths to simulate each edge  $(v, v^k)$  between pairs of adjacent nodes. Let  $v^b$  simulate  $v$  and  $v^{kb'}$  simulate  $v^k$ . Then to simulate  $(v, v^k)$  nodes  $v^b$  and  $v^{kb'}$  choose a path between them of the form

$$P(v, v^k, b, b', r) = (v^b, v^{br}, v^r, v^{rk}, v^{rkb'}, v^{kb'})$$



This gives us our constant *dilation* of 5.

## Assigning Paths to Simulate Edges

$p < 1 - \sqrt[4]{0.5}$  implies

$$\Pr\{P(v, v^k, b, b', r) \text{ is live}\} = s = (1 - p)^4 > \frac{1}{2}.$$

Also, the paths able to simulate  $(v, v^k)$  with  $(r \neq k)$  are node-disjoint for a fixed choice of  $v, v^k, b$ , and  $b'$ , thus the probability that any one of them is live is independent of the other paths.

Let  $S(v, v^k, b, b')$  be the set of dimensions  $r \neq k$  such that  $P(v, v^k, b, b', r)$  is a live path.

**Lemma 4.** *With high probability, for all quads  $(v, v^k, b, b')$ ,  $|S(v, v^k, b, b')| > \eta_p n$  for constant  $\eta_p$ .*

So we for each edge we have many paths to choose from, but which one do we choose to give a constant *congestion*?

## Assigning Paths to Simulate Edges

**Definition 7. [unsaturated]** *An edge is saturated if it has accepted  $B_p$  paths through it, otherwise it is unsaturated.*

For  $i = 1$  to  $s'_p n$

For each unassigned adjacent pair of nodes

Pick a path between them uniformly

Each unsaturated edge agrees to as many paths routed through as it can without exceeding its capacity, giving preference to lower labeled pairs

All excess pairs remain unassigned

Let  $S'(v, b) = \{r \mid \text{more than } \gamma_p \text{ nodes can send a path through the edge } (v^{br}, v^r)\}$ .  $|S'(v, b)| \leq \frac{\eta_p}{4} n$

Let  $T(v, v^k, b, b') = S(v, v^k, b, b') - S'(v, b) - S'(v^k, b')$ .

Let  $U(v, v^k, b, b')$  be the subset of  $T(v, v^k, b, b')$  consisting of dimensions  $r$  for which all edges along  $P(v, v^k, b, b', r)$  are unsaturated.

## Assigning Paths to Simulate Edges

**Definition 8. [dedicated]** *a path is dedicated to a pair  $(v, v^k)$  when there are not  $\beta_p n$  choices for a simulating path.*

Iterative Version:

For  $i = 1$  to  $s'_p n$

For all unassigned pairs  $(v, v^k)$  in order

If  $|U(v, v^k, b, b')| < \beta_p n$

dedicate enough  $r \in T(v, v^k, b, b')$

Pick a path between them uniformly

If the chosen path is unsaturated or dedicated

$(v, v^k)$  is assigned to the path

else  $(v, v^k)$  remains unassigned

## Assigning Paths to Simulate Edges

**Lemma 5.** *For a suitably large choice of the constant  $s'_p$ , with high probability all pairs of nodes searching for an assignment to a path have been assigned one after  $s'_p n$  steps of the iterative algorithm.*

**Lemma 6.** *whp no set  $U(v, v^k, b, b')$  ever has cardinality less than  $\beta_p n$  at the beginning of some step of the iterative algorithm, given  $\beta_p = \frac{\eta_p}{4}$*

## Putting it all together

**Theorem 1.** *For each  $p < 1 - \sqrt[4]{.5}$  there is an  $O(\log N)$  step algorithm such that if each of the nodes fails with prob.  $p$  then whp the algorithm finds an embedded fully functioning  $N$ -node cube with constant load, dilation, and congestion. The paths which simulate the edges of the cube use only live nodes.*

Edge faults: Let  $p_n$  be the probability of a faulty node and  $p_e$  be probability of an edge being faulty. Then if

$$P_n + p_e - p_n p_e < 1 - \sqrt[5]{.5} \text{ (about 0.13)}$$

the algorithms work with high probability. Just need to reason that if one node tries to communicate with a neighbor node, it fails both when the neighbor node is faulty and when the link between them is faulty.

## Putting it all together

By showing that most local areas of a cube retain *good* structure even with very high constant probability of failure:

**Theorem 2.** *If each node of an  $N$ -node hypercube fails independently and with constant probability  $p < 1$ , then with high probability, the faulty hypercube can simulate a completely functioning  $N$ -node hypercube with only a constant slowdown.*

# Issues

The high constant probability results are interesting theoretically, but I don't see the probability being that high in practice (unless faults have accumulated over an extended period of time).

While theoretically very important, implementing the algorithm does not appear obvious:

1. What extra information needs to be added to the headers to support routing on the embedding?
2. Given that a particular node is simulating a constant number of other nodes, which nodes computations does it simulate first? Lexicographic ordering?
3. How overbuilt do the nodes need to be?

# Conclusion

- An  $O(\log N)$  time adaptive routing algorithm with offset routing.
- An improvement to Rabin's information dispersal with redundancy technique with a simpler algorithm.
- Constant delay embedding of a virtual  $N$ -node hypercube in a hypercube with randomly distributed node and edge faults with high probability.
  - Simulate the nodes (constant *load*)
  - Simulate the edges (constant *load* and *congestion*)
  - Some questions of practicality