# Routing in a Parallel Computer

November 7, 2017

# Model

- model parallel machine as a graph
  - $N$ nodes
  - nodes: processing elements
  - unique identifier in $1, ..., N$
  - edges: communication links
- communication in synchronous steps
- in each step, send at most one packet over a link

# The Permutation Routing Problem

- Each processor initially contains one packet destined for some processor in the network.
- Let $v_i$ denote the packet originating at processor $i$.
- We denote its destination by $d(i)$.
- $d(i)$ forms a permutation of $\{1, ..., N\}$.
- How many steps are necessary and sufficient to route an arbitrary permutation request?

# Oblivious Algorithms

- Oblivious strategy: The route chosen for each packet does not depend on the routes of other packets.
- That is, the path from $i$ to $d(i)$ is a function of $i$ and $d(i)$ only.

# Lower Bound

### Theorem (Theorem 4.4, MR95)

*For any deterministic oblivious permutation routing algorithm on a network of $N$ nodes each of out-degree $d$, there is an instance of permutation routing requiring $\Omega(\sqrt{N/d})$ steps.*

# Hypercubes

- A popular network for parallel processing is the Boolean hypercube.
- $N = 2^n$ nodes
- connected in the following manner:
  - Let $(i_0, ..., i_{n-1}) \in \{0, 1\}^n$ be the (ordered) binary representation of node $i$
  - There is a directed edge from node $i$ to node $j$ if and only if their binary representations differ in exactly one position.
- Every node in the hypercube has $n = \log_2 N$ directed outgoing edges.
- Theorem 4.4 then tells us that for any deterministic oblivious routing algorithm on the hypercube, there is a permutation requiring $\Omega(\sqrt{N/n})$ steps.

# The Bit-Fixing Algorithm

- source and destination addresses are $n$-bit vectors
- scan the bits of $d(i)$ from left to right
- compare them with the address of the current location of the packet.
- Send the packet along the edge corresponding to the left-most bit in which the current position and $d(i)$ differ.

# A Randomized Oblivious Algorithm

- **Phase 1:** Pick a random intermediate destination $\sigma(i)$ from $\{1, ..., n\}$. Packet $v_i$ travels to node $\sigma(i)$.
- **Phase 2:** Packet $v_i$ travels from $\sigma(i)$ on to its destination $d(i)$.
- Each phase uses the bit-fixing strategy to determine its route.
- Nodes use a FIFO queue to store incoming packets.

# Analysis

- Observation: View each route in Phase 1 as a directed path in the hypercube from the source to the intermediate destination. Once two routes separate, they do not rejoin.

## Lemma

*Let the route of $v_i$ follow the sequence of edges $\rho = (e_1, e_2, ..., e_k)$. Let $S$ be the set of packets (other than $v_i$) whose routes pass through at least one of $\{e_1, e_2, ..., e_k\}$. Then, the delay incurred by $v_i$ is at most $|S|$.*

# Theorem

## Theorem

*With probability at least* $1 - 1/N$*, every packet reaches its destination in* $14n$ *or fewer steps.*