CS 590R: Algorithms for Communications Networks February 6th, 2003

Lecture 8

Lecturer: Gopal Pandurangan Scribe: Jayesh Pandey

Packet Routing in Parallel Computer

1 Properties of a Communication Network

1.1 Nodes

The nodes can be considered as switches or processors in the network. If there are N nodes in the network, each node has an unique identifier between 1 and N. Also the nodes have a buffer to store packets.

1.2 Edges

The communication links between the nodes form the edges. We assume the links are unidirectional. In case the links are bidirectional, the same analysis holds by taking into account a factor of 2.

1.3 Synchronous Operation

In a tightly coupled system like a parallel computer, all communication proceeds in a synchronous fashion.

1.4 Capacity

Each link can carry 1 packet (in one direction).

1.5 Congestion

In one step a node can send out at most one packet to each of its neighbor. Thus in case more than 1 packet has to follow a link, then they will be queued using some associated queuing mechanism.

2 Routing

Given that node i wants to send packet v_i to destination $d(i)$ for $1 \leq i \leq N$. A routing algorithm routes the packets to their destination. Since at each step at most one packet can be served by a given edge, every edge has an associated queue, specified by a queuing disciple.

2.1 Permutation Routing

In permutation routing each node sends a packet to a different destination. That is $d(i)$ is a permutation of 1 to N. For example

src 1 2 3 4
des 2 4 1 3

Remark: This can be considered as a good case as every node gets one packet.

2.2 Oblivious Routing

The route taken by a packet v_i from i to $d(i)$ depends only on its destination $d(i)$ and does not depend on the destinations of any other packets nor on the state of the network at any time step.
Remarks: It is very simple as you don't need to worry about other packets. Hence can be implemented in hardware.
The Goal is to minimize time required for routing.

3 Lower Bound for Deterministic Routing

Theorem 1 *For any N-node network with maximum degree d, and any deterministic oblivious packet routing algorithm, there is a permutation that requires $\Omega(\sqrt{N/d^3})$ steps. [1].*

We will do this in the next class.

4 The Hypercube

We do the analysis for a hypercube network.

4.1 Hypercube

The n-cube has $N = 2^n$ nodes and dimension n. The nodes are all the bit vectors of length n i.e. $x = (x_1, \ldots, x_n) x_i \in \{0, 1\}$. Two nodes are connected iff their binary representation differ in exactly one bit. (So degree of each node is n). You can create a hypercube in $N + 1$ dimensions from an N-dimensional by making an extra copy of the N-dimensional cube, and adding an extra bit which is zero in one copy and one in the other. Then join corresponding vertices. E.g. a $4D$ hypercube can be built from the $3D$ hypercube this way as shown in Figure 1.

4.2 Bitwise Routing

The path from i to $d(i)$ is just the path we get by going over the bits of i one by one from left to right and'fixing' them whenever needed to be the bits of $d(i)$. This is an oblivious routing.

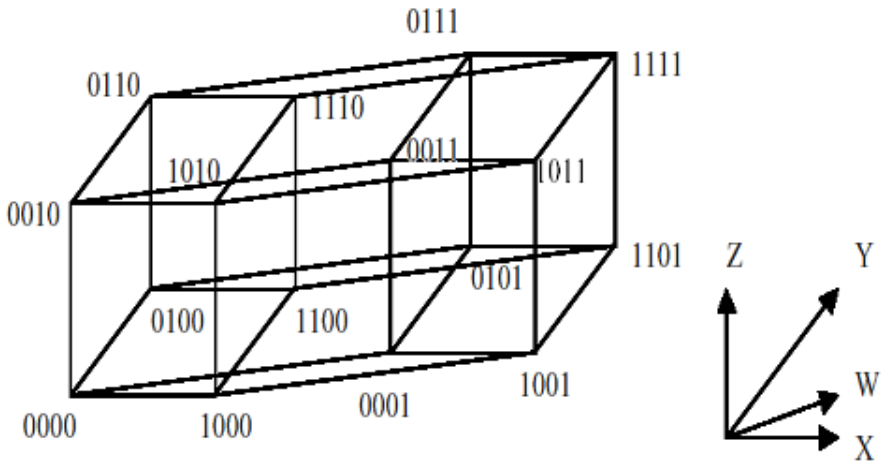

Figure 1: $4D$ Hypercube

5 Randomized Routing Algorithm

As theorem 1 states, the randomized algorithm performs exponentially better than any deterministic algorithm. This randomized algorithm is due to Valiant's trick [2].

1. For each packet v_i independently choose an intermediate destination (i) uniformly randomly from the set of nodes.
2. Phase 1: Send the packet from i to (i).
3. Phase 2: Send the packet from (i) to $d(i)$.

Remarks: The routing in each of the phases is done using bitwise routing.
The second phase is symmetric to first phase in analysis.
The queuing mechanism can be any greedy method e.g. FIFO.

Theorem 2 *The two-phase routing algorithm routes an arbitrary permutation on the n-cube in $O(\log N)$ steps with high probability (w.h.p.).*

Proof:
We bound the routing time of a given packet v_i.
Let $\rho_i = e_1, \ldots, e_k$ be the edges traversed by v_i in phase 1.
The number of steps taken by v_i is equal to the length of ρ_i, which is at most n, plus queueing delay in the intermediate nodes.

Lemma 3 *If a packet leaves the path of another packet it cannot return to that path in the same phase.*

Proof: That is, a pair of routes can look like the left picture but not the right in Figure 2.

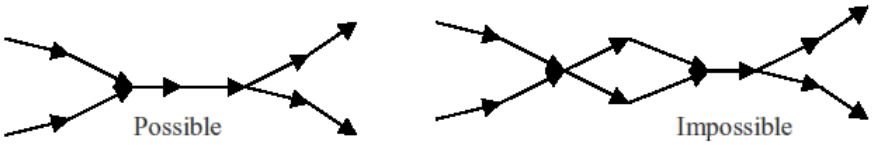

Figure 2: Packet Path

To see this, notice that during bitwise routing, an intermediate address always looks like $D_1, , D_k, S_k + 1, , S_n$ where S_i is a bit of the Source address, and D_j is a bit of the destination address. If two routes collide at the k^{th} step, that means their destination addresses agree in their first k bits and the source addresses agree in their last $n - k$ bits. Let k_0 be the first value of k for which the packets collide. At each time step, we add one more bit of the destination address, which means we increment k. Eventually, the destination bits must disagree (because the destinations are different). Let k_1 be the value of k at which this happens. Then the D_{k_1} destination bit is different for the two packets. At that point, the two packets separate. They will never collide again, because all the later intermediate destinations will include the D_{k_1} bit. ■

This observation is the crux of the proof of the following Lemma:

Lemma 4 *Let S be the set of packets (other than v_i) whose routes pass through at least one of the edges in ρ_i. Then the delay incurred by v_i is at most $|S|$.*

Proof: Its enough to notice that whenever the routes of two packets intersect, one of the packets may be delayed by one time step. Once that packet is delayed by one time step at the first shared node, it will flow along the shared route behind the other packet, and will not be delayed any more by that packet. If the same route intersects other routes, each of them may add a delay of one time step. This happens either (i) because another packet collides with the current packet along a shared route, or (ii) because another packet collides with a packet that is ahead of the current packet along a part of the route which is shared by all three. In either case, an extra delay of at most one results.
So the worst case delay is at most $|S|$. ■

To get the running time of this scheme, we compute the expected value of the size of the set S above. Define an indicator random variable H_{ij} which is 1 when the routes of packet v_i and packet v_j share at least one edge, and H_{ij} is 0 otherwise. Then by the above theorem, the expected delay of packet v_i is the expected size of S which is

$$\sum_{j=1}^{N} H_{ij}$$

Its rather difficult to an exact estimate of this quantity because H_{ij} is a complicated condition. Its easier to think about $X(e)$ which is the number of routes that pass through a given edge e. Now suppose the route of packet v_i consists

of the edges $(e_1, e_2, \ldots, e_k)$. Then we have

$$\sum_{j=1}^{N} H_{ij} \leq \sum_{l=1}^{k} X(e_l)$$

This is an upper bound as a packet could be counted multiple times.
For the $N-1$ packets other than v_i let $F_l^j = 1$ iff packet j traversed edge e_l, else $F_l^j = 0$. So,

$$E[X(e_l)] = \sum_{j=1}^{N-1} \Pr(F_l^i = 1)$$

Since traversing e_l "fixes" the l^{th} bit, a packet can cross that edge only in its l^{th} transition. So number of packets traversing an edge is $2^{l-1} * 2^{-l} = \frac{1}{2}$.
Hence $E[X[e_l]] \leq \frac{1}{2}$.

$$E[\sum_{j=1}^{N} H_{ij}] \leq E[\sum_{l=1}^{k} X(e_l)] \leq \frac{n}{2}$$

Now we can apply Chernoff bounds to the probability of there being a substantial number of paths intersecting v_is path. The Chernoff bound is

$$\Pr(\sum_{j=1}^{N} H_{ij} \geq 3n) \leq 2^{-3n} = 1/N^3$$

Thus with probability at least $1 - 1/N^2$, every packet finishes phase 1 in $4n$ or fewer steps. The $4n$ comes from the delay time ($3n$) plus the time for the bit fixing steps, which is $\leq n$. Notice that all of this applies to just one phase of the algorithm (recall that phase one routes a packet to a random location, and phase two routes from there to the actual destination). So the full algorithm routes all packets to their destinations with high probability in $8n$ or fewer steps. ■

References

[1] A. Borodin and J. Hopcroft, *Routing, Merging and Sorting on Parallel Models of Computation*, JCSS 30, (1985), pp. 130-145.

[2] L. G. Valiant, *A Scheme for Fast Parallel Communication*, SIAM J. Comp 11 (1982), pp. 350-361.