CS273 Scribe Notes for 10 February 2003

Instructor: Satish Rao (satishr@cs.berkeley.edu)
Scribe: Satrajit Chatterjee (satrajit@cs.berkeley.edu)

Do not distribute without consent of the instructor.

1. Last Time: Random *1*-mapping

We saw that random *1*-mapping can be done in $O(\log N)$ time. With high probability the path congestion was $\Theta(\log N)$.

The reasoning behind that is as follows. In random routing, with probability ½ every packet chooses the upper path or the lower path at a switch. Therefore the expected path congestion along a specific path is $(1/2) + 1\,(1/4) + 2\,(1/8) + 4\,(1/16) + \ldots = \Theta(\log N)$

Using a Chernoff bound, we showed that the path congestion was this with high probability.

2. Random (*log N)*-mapping: Naïve Bound

Instead of one packet, now try to route *log N* packets at each input. (You hold each packet in a queue at the beginning and then send one out on each time step.) It is easy to see the upper bound of $(\log N/4)(\log N)$ i.e. $O(\log^2 N)$ for path congestion in this case. But while this argument led to an optimal result for the random 1-mapping case (since the diameter of the network was *log N,* we cannot hope to do much better), in the random *(log N)*-mapping case we get a sub-optimal result. Can we do better?

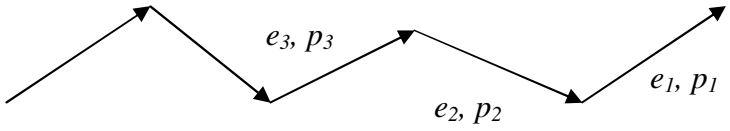
Yes, we can! The naïve bound is not tight because we are being too pessimistic. We are assuming that packets get delayed at every possible location. By having a smarter algorithm and a smarter argument, we can prove a better bound.

3. Random (*log N)*-mapping: Smarter Algorithm, Better Bound

In the smarter algorithm, assign a random priority to every packet and at a switch send the packet having the higher priority first.

Recap of the delay-sequence algorithm. You would ask a delayed packet why it was late. The packet could have one of two replies: 1. I was on edge *e* or 2. I was waiting for packet *p* on edge *e*. If the reply was the latter, you'd start following packet *p* ("Why were you messin' with ma boy!?!").

Now consider a delay sequence:



Here, p3 delays p2, so p3 had higher priority then p2. Similarly p2 must have higher priority than p1. Thus a delay sequence sorts the priorities. If *D* is the depth of the (level) network, then a delay sequence of length t has *(t-D)* different packets. The probability that this delay sequence is sorted is *1/(t-D)!*
Therefore, given a delay sequence, it is unlikely to exist!

Now,

$$P(\exists \text{ a sorted delay sequence of length} \geq t) \leq \left\{ N \cdot 2^D \cdot C^{t-D} \cdot \binom{t}{D} \right\} \cdot \frac{1}{(t-D)!}$$

The probability is bounded by the product of the total number of sequences of length *t* possible (the first term in curly braces) and the probability of each individual sequence of length *t* existing.

The total number of sequences of length *t* is computed as follows. The first two terms fix the total number of paths in the network (*N* is the number of output nodes and 2^D selects for every level which way we route). The third term fixes the number of sequences of packets and the last chooses the places on the sequence where we move back an edge.

Now choose t so that the above probability bound is inverse polynomial in N. It can be verified that $t >> D + C + 10 \log N$ does the trick.

Thus for the butterfly network, randomized *(log N)*-mapping is *O(log N)* time with high probability using random priorities. In fact this is true for any level network of depth *D*.

4. Finite Queues and a Flawed Analysis

From the above consideration, the communication for *p log p* virtual processor EREW PRAM (having *p* real processors) can be implemented using a *p*-input butterfly with high probability in *log p* steps.

But right now queue sizes are assumed to be finite. What happens when queue sizes are bounded? The analysis becomes significantly harder since a packet can be delayed some other packets which are far away. This was an open problem for sometime.

Let us try the delay-argument to see where it leads us in this case. Now when we ask a packet the reason for the delay, in addition to the two answers we saw in Section 3, it could have an additional answer: I was waiting for the queue in front of me which was full (I was ready, so was the edge.).

So now while tracing, we need to move forwards along an edge to ask why the queue at the destination node was full.

Let f be the number of forward edges in the delay sequence. Now the expression in Section 3 gets modified to:

$$P(\exists \text{a sorted delay sequence of length} \geq t)$$
$$\leq \left\{ N \cdot 2^{F} \cdot 2^{D+f} \cdot C^{t-D+qf} \cdot \binom{t}{f} \cdot \binom{t-f}{D+f} \right\} \cdot \frac{1}{(t-D+qf)!}$$

With a generous helping of algebra, one can show that the above is inverse polynomial in n for $t >> D + C + \log N$.

But this analysis is wrong! Since forward edges are present, the delay sequence need not be sorted.

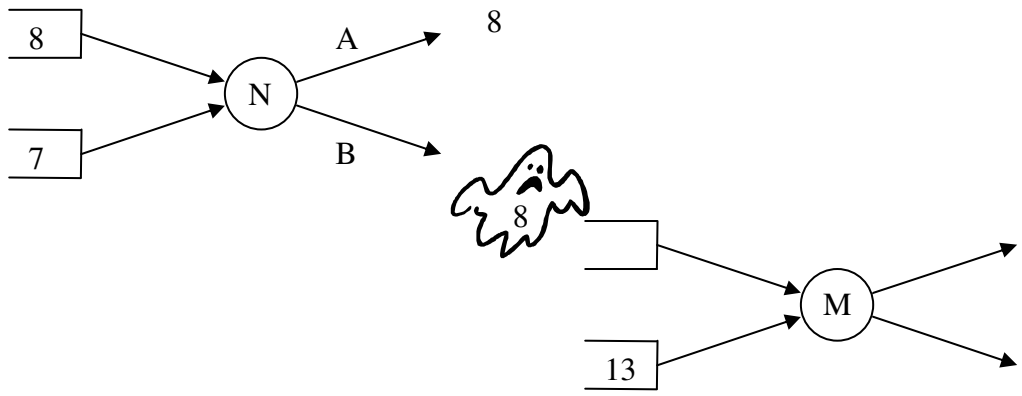
People got stuck here. The solution came a few years later, though from a different direction altogether.

5. CRCW PRAMs and Serendipity

We now look at an algorithm for routing memory requests in an interconnection network for CRCW PRAMs. The CRCW model considered here is the $\oplus$ model, i.e. if two guys write to the same location simultaneously, the sum of the two writes is written to the memory location.

Consider the following system:

1. Two packets destined for the same memory location have the same priority.
2. Two simultaneous read requests for the same location get merged at some internal node (where they collide) and the node remembers the merge so that the reply packets can be unfolded.
3. Also every node only sends a packet out when it can prove that it will not get a higher priority packet.



Node N above can send packet 8 along edge A, since it knows that nothing higher than 8 is going to come on either of its inputs after this point. The node also sends out a ghost packet of 8 on its other output edge B. This information allows the node connected to B, i.e. node M to send 13 out (since now M can guarantee that nothing higher than 13 is going to come).

Ghost packets take no space in the queues, since if there are any real packets that follow them, the ghost packets do not need to be stored.

It turns out that this idea which originated in the CRCW world can be used to analyze bounded queues. The proof and more in the next lecture.