Lecture 7

1 Overview

Last time, we showed that since for a one to random destination routing problem, that with high probability the congestion is $O(\log n)$ (actually $O(\log N/\log\log N)$), and therefore any "greedy" routing algorithm will certainly route in $O(\log^2 N/\log\log N)$ time. We then showed that this could be improved to the $O(\log N)$ time by using the notion of path congestion rather than edge congestion.

We also note that we used "infinite" queues at each node.

2 Loaded Butterflies

.

The analysis of the previous lecture is optimal to within a constant factor since one cannot route in fewer than $\log N$ steps, and it routes in $O(\log N)$ steps. But is it efficient? Could one route more packets in this time? Well, the average congestion on each edge is $O(1)$, so most edges are idle most of the time? Can we use them to route other packets.

For example, what if $\log N$ packets started at each input and were destined for each output? Can this be accomplished in $O(\log N)$ time? This, we will show (to borrow a phrase from Yoda.)

In the previous section, we showed how to select routes that are good in the sense of congestion. Here we assume that we have a routing that gives congestion at most C. Furthermore, we assume that the network has a special structure (like the Benes network) where packets flow monotonically between levels of a network. For example, the butterfly network each packet flows from one level to the next, the depth D of the network is $\log N$. We call these levelled networks.

We will use an algorithm that assigns random priorities to each packet. That is, if one is at a node, the highest priority packet is sent. Note this allows packets to jump over other packets in queues.

First, think of a packet's progress. It either moves or is delayed. This time it is delayed by a higher priority packet. So, if the packet is delayed L times it must have met up with L higher priority packets. We will write down this thing as a delay sequence which we define as follows with respect to a priority function $r(\cdot)$ on the packets.

Definition 1 *A priority respecting delay sequence for a packet routing is a sequence of pairs (p_i, e_i) where p_i is a packet and e_i is an edge in the network where packet p_i uses edge e_i and either*

1. $p_i = p_{i+1}$ *and* $e_i \neq e_{i+1}$ *and* e_{i+1}, e_i *form a path in the network, or*

2. $p_i \not p_{i+1}$ *and* $e_i = e_{i+1}$*, and* $r(p_i) > r(p_{i+1})$.

Lemma 1

If a packet p is delayed L time units in a priority based routing, one can construct a priority respecting delay sequence of length $D + L$ where D is the depth of the network.

One can start with the packet, (p, e), where e is the last edge that the packet traversed. Now, if the packet was delayed in the previous step, that would mean that it waited to use edge e. This would mean some other higher priority packet p' used e. Thus, we (p', e) to the delay sequence. Otherwise, the packet must of used another edge e' in the previous step, and we add (p, e') to the delay sequence. We continue to extend the delay sequence in this manner. Note, we change the packet that we follow backwards. When we switch edges we go backwards in the network. The ith iteration corresponds to the $T - i$th step of operation of the network. Since $T = D + L$, we can continue for this many steps.

Notice, further that the number of times we switch packets is L and the number of times we switch edges is D.

Now, we can prove the following theorem.

Theorem 2

For any levelled network of bounded degree and of depth D, the priority routing algorithm routes in time $O(C + D + \log N)$ time.

Proof:

We will show that the probability of a very long, i.e., $>> C + D + \log N$, is very low. Here, the probability space is defined by the choice of the random priorities.

We assume that a packet arrives at time t, and note that there will be a delay sequence of length t.

The probability that a delay sequence of length t is sorted is at most

$$\frac{1}{(t-D)!}.$$

For convenience, we define $L = (t - D)$.

Here, we assume that the priorities are chosen in a large enough universe so that there are no ties.

Question 1: How large a range must someone choose the packets priorities from before one can conclude that each packet has a different priority, with probability at least $1/N^2$.

The calculation then follows by noting that the number of packets is $L = t - D$ and they must be sorted. Then we observe, that there are $L!$ possible orders for the priorities of these packets. (Formally, we are taken a conditional probability conditioning on the fact that no 2 packets have the same priority.)

What is the number of delay sequences of length t? Well each delay sequence consists of a path of length D in the network, a choice of whether to switch packets or edges at each point, and a selection of packets on $L = t - D$ edges on the path. The number of paths is at most

$$Nd^D$$

, where d is the maximum in-degree in the network, N starting points, and d choices of edges to switch in each step. The number of choices for switching packets is

$$\binom{t}{D}.$$

The number of choices of packets at each place where packets are chosen is C. Overall this amounts to C^L possibilities. All together the total number of delay sequences is bounded by

$$Nd^D\binom{t}{D}C^L.$$

By the union bound, the probability that there is any sorted delay sequence given a randomly chosen set of priorities at at most

$$Nd^D\binom{t}{D}C^L\frac{1}{L!}.$$

Question 2: Show that for some $L = \Omega(D + C + \log N)$, this probability is at most $1/N$. □

This result was first proven by Alelunias and Upfal. This presentation was based on Ranade's algorithm, which was invented to solve the problem for routing with bounded queues.

We proceed to use it thusly. (Is that even a word?)

3 Bounded Queues

What about bounded queues? How do we solve this problem? Well, Ranade invented the random priority algorithm to solve precisely this problem. There is one additional condition he enforces.

Ranade's sorting condition. The packets that use a switch must emerge in sorted order in terms of the priorities.

This seems a global condition. That is before one can send a message out a link, one needs to know if anyone with better priority is ever going to use the switch!! How can we implement this.

Ranade used "ghost packets", whenever one sends a packet of priority γ out one edge, one should send a "ghost packet" of priority γ out all the other edges.

Consider the first level. We start by sending out a packet on one edge, and we send ghost packets out the other.

For internal nodes, it gets a bound on the priorities from the incoming packets. By induction, each input has packets with worse and worse priorities over time. Thus, one sends a packet from one input, when the other inputs verify that no better packet it to come. That is, one can always send the best packet at all the inputs.

Here, we use the idea of the levelled network of depth D. After D steps everyone, has at least a ghost packet on the inputs.

Finally, there is a bounded buffer, say of size 5 at each edge, so a switch may not be allowed to send a packet at all!

So, to review, the algorithm at each node is to send the best priority packet from all its inputs, *if* there is room at the next switch. Moreover, it sends a ghost copy of the packet on all edges.

Notice, that one never needs to have a ghost packet being internal to a queue. If a packet arrives after it, it has worse priority and can simply destroy the ghost packet. Moreover, if it does wait it simply informs the previous packet in the queue of its lower bound. (Thus, each packet carries a lower bound and a priority.)

Now, we can state a generalization of Ranade's theorem.

THEOREM 3
For any bounded degree, levelled network of depth D, one can route any set of paths with congestion C, in time $O(C + D + \log N)$ using switches with constant sized buffers.

PROOF:
Now, the concept of the delay sequence needs to be modified. A packet p can be delayed by a packet that uses a node that it wants to, or by the fact that a queue is full ahead of it, or by the fact that it is waiting to get a signal that its priority is high enough, i.e., a ghost packet arrived. Note that the ghost packet, never waits.

Now, we will define a definition of delay sequence that corresponds to this situation.

DEFINITION 2 *Given a set of packets with priority function $r(\cdot)$ a sorted delay q-bounded buffer delay sequence is a sequence $\{(p_1, e_1), (p_2, e_2), \ldots,\}$ where p_i uses edge e_i and*

- *$p_i = p_{i+1}$ and $e_i \neq e_{i+1}$ and e_{i+1}, e_i form a path in the network, or*
- *$p_i = p_{i+1}$ and $e_i \neq e_{i+1}$ and e_{i+1}, e_i leave the same switch in the network, or*
- *$p_i \not p_{i+1}$, e_{i+1} is a predecessor of edge e_i in the levelled network and $r(p_i) > r(p_{i+1})$, or*
- *$p_i \not p_{i+1}$, e_i is a descendant edge of e_{i+1}. We define this to be a forward edge.*

The third case is more complicated because we follow the ghost packet to the real packet initiator of the ghost packet. We note the ghost packet never waits so this process is easy in a sense. The fourth condition, models waiting for a packet in a previous queue. The second case models waiting in a queue with other packets. The first models the case of waiting for another packet to use your edge.

Using this reasoning, we have the following lemma.

LEMMA 4
If a packet arrives at time t in a levelled network of depth D, one can construct a bounded queue, sorted delay sequence of length $t + f(q-1)$, where f is the number of forward edges in the sequence.

PROOF:
We start with a delayed packet p. We follow output (p, e) where e was the last edge it used. If in the previous step it moved, we output (p, e'), if not it was delayed. If another

packet used e, we output (p, e). If it waited in a queue, we output (p', e') where the previous packet p' wishes to use e' in the queue. If it waited for a queue on the next edge, we output the a sequence of q pairs corresponding to the packets in the queue ahead. (Note, this is a forward edge.) If a packet was allowed to go by a ghost packet, we follow the ghost packet to the last real packet that used it.

Each step of this process goes back in time, one unit. The forward edges add q packets to the delay sequence for one unit of time.

□

Again, we count the number of possible delay sequences and then multiply by the probability that each is sorted to upper bound the probability that any packet arrives after time t.

The number of delay sequences is at most

$$N(d)^{D+f}C^{t-(D)+(q-1)f}.$$

This uses the fact that we trace out a path by going backward or by going forward, and the fact that when we move backward we get no new packets. Again, the probability that it is sorted is

$$\frac{1}{(t-D+(q-1)f)!},$$

since $t - D + (q-1)f$ is the number of packets and there is one sorted order.

Plugging in some value of $t = O(C + D + \log N)$ can ensure that the probability is less than $1/N^2$.

□