# 22C:296 Seminar on Randomization

Scribe: Shuquan Hou

Oct 1st

### Application of Chernoff Bounds (cont'd)

**Oblivious routing problem:**
Last time we introduced oblivious routing problem and bit-fixing protocol. Today we will continue to work on oblivious routing problem. First we want to show that the bit-fixing protocol performs badly.

**Claim:** There is an instance of $(i, d_i)$, i= 1, 2, $\cdots$,N, such that $\Omega(\sqrt{\frac{N}{n}})$ time steps are needed for all packets to reach their destinations
**Proof:** Suppose n is even. Say $b(i) = a_i b_i$, where $b(i)$ denotes the bit-code of node i. $|a_i| = |b_i| = \frac{n}{2}$. Set $d_i = b_i a_i$. Consider path $a_i b_i \rightsquigarrow b_i b_i \rightsquigarrow b_i a_i$, where $b_i b_i$ is an intermediate node on the path from $a_i b_i$ to $b_i a_i$. Fix $b_i$, for each of the $2^{\frac{n}{2}}$ $a_i b_i$'s the packets originating at node $a_i b_i$ has to pass through node $b_i b_i$. At each time node $b_i b_i$ can at most send $\frac{n}{2}$ packets. This is because of the constraint that each node can send at most one packet to each out-neighbor in each time step. Hence $2^{\frac{n}{2}}$ packets pass through node $b_i b_i$ need at least $\frac{2^{\frac{n}{2}}}{\frac{n}{2}}$ time steps to get their destinations. Since $N = 2^n$, then it at least needs $\frac{\sqrt{N}}{\frac{n}{2}}$ time steps. $\square$

### Randomize bit-fixing protocol

1. For each i, pick an intermediate destination $\sigma_i$ uniformly at random from $1, 2, \cdots, N$. Use the bit-fixing protocol to send packet $v_i$ from i to $\sigma_i$.
2. Use bit-fixing protocol to send packet $v_i$ from $\sigma_i$ to $d_i$.

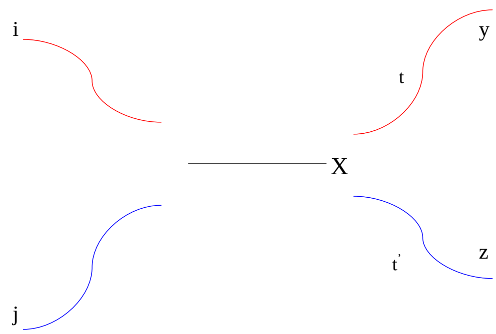
### Queueing dicipline

Each node maintains a FIFO queue for each out-going edge. At each step, packet at the front of each queue is sent off. At each step all arriving packets are queued into their out-going edge queue. Ties are broken arbitrarily.

Let phase 1 denote packets travel from i to $\sigma_i$. We will first analyze phase 1.

**Claim:** If a pair of path $i \rightsquigarrow \sigma_i \text{ and } j \rightsquigarrow \sigma_j$ seperates, then they never rejoin.
**Proof:** Assume node i and j share some nodes until node X. Then two pathes separate from node X. Since two pathes differ from $t^{th}$ bit, then two future pathes differ at least 1 bit, so they

can be same again, then they can't rejoin.



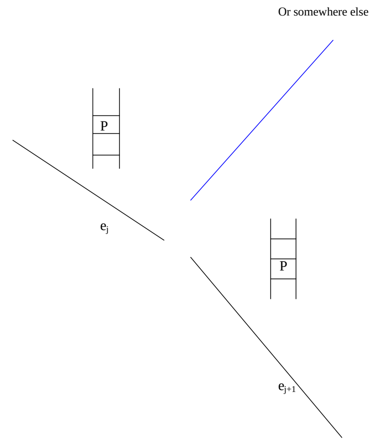

□

**Definition 1** *Suppose at time step t, a packet is in a queue on edge $e_i$, then $t +$ (# of elements in the queue before the packet $v_i$) - $i$ is the lag of the packet at time $t$.*

**Definition 2** *The delay of $v_i$ is the lag at the time at which $v_i$ passes through $e_k$.*

**Claim:** Let the route of $v_i$ follows the sequence of edges – $\rho_i = (e_1,\ e_2,\ \cdots, e_k)$. Let $S$ be the set of packets (other than $v_i$) whose routes pass through at least one of $e_1,\ e_2,\ \cdots, e_k$, then the delay of $v_i\ \le |S|$.

**Proof:** What causes the lag of $v_i$ to increase from $l$ to $(l\ +\ 1)$? If $v_i$'s lag has grown from $l$ to$(l\ +\ 1)$, there must be a packet in $S$ whose lag is $l$ at some time. Consider the packet in $S$ whose lag is $l$ last. Let us say P $\in S$, P is the last packet ever to have lag $l$, has to been at front of queue (otherwise it can't delay $v_i$ or P is the last packet to have lag $l$). When $v_i$ is from $e_j$ to $e_{j+1}$, there are three possible cases.




1. When P is on edge $e_{j+1}$, P is the on the top of the queue. If no other packets are behind P, then P can't delay $v_i$ anymore. If there are other packets are behind P, then P can't be the last packet ever to have lag $l$, so it is impossible.

2. When P is on the edge $e_{j+1}$, P is behind some packets. This is impossible. Because we claim that P is the last packet ever to have lag $l$.

3. P is going other edges which doesn't share any path with $v_i$

Either case will charge the increase in lag of $v_i$ from $l$ to $(l\ +\ 1)$ to this packet P. Overall every packet in $S$ get charged $\leq$ once.

You can also think it in this way. Each packet in $S$ at most get charged at once. Let us say if Packet P delay $v_i$ once, if P can delay $v_i$ again, it must because that other packet Q delays P. So we can charge Q once instead of charging P. From either way we can show that the delay of $v_i\ \leq |S|$. $\square$