Permutation Routing on a Hypercube

1 Deterministic Method Fails

We will show that the oblivious algorithm which considers $i \otimes \sigma(i)$ and "fixes" bits one at a time can create a lot of congestion, and hence a lot of delay. In particular, consider the transposition permutation – route from ab to ba (each of a, b are $d/2$ bits long). In other words, we swap the first and second halves of the bitstring.

Consider all sources $u = 1a'000\ldots0$ with $|a'| = d/2 - 1$ and $d/2$ zeros. They go to $000\ldots01a'$. All of these go to $000\ldots0$ and from there to $000\ldots010\ldots0$. As a result, $2^{d/2-1} = \sqrt{n}/2$ packets use the edge from $000\ldots0$ to $000\ldots010\ldots0$. So the congestion is at least $\sqrt{n}/2$, and the delay is at least $\sqrt{n}/2$ – pretty bad! More generally, for any graph G with maximum degree d and any deterministic algorithm (oblivious or not), there is a lower bound of $\Omega(\sqrt{n/d})$ on the maximum delay.

The problem is that we can carefully construct bottlenecks that cause a lot of delay. To avoid this, for each source i we will route it to a uniformly random intermediate destination $\sigma'(i)$ and from there to the actual destination $\sigma(i)$. We will continue to use the left-to-right fixing algorithm and FIFO queues at each node in each stage. The only randomness is in $\sigma'(i)$. Note that multiple i can have the same $\sigma'(i)$, but the algorithm is oblivious.

The two phases are symmetric, so we only analyze the first phase and double it. Rename $\sigma'(i)$ to $\sigma(i)$. Packet i travels from i to $\sigma(i)$ along a path $P_i = (e_{i,1}, \ldots, e_{i,k_i})$ – usually we will write $(e_1, \ldots, e_k)$ when i is clear. Since $k \leq d = \log_2 n$, this would take $\log_2 n$ steps if it weren't for delays.

2 Delay Analysis

Delays are caused when i wants to cross e_j, but has to wait for other packets to cross e_j first. Our goal is to precisely relate the resulting delay of i to the number of packets sharing an edge with i, then use randomness in $\sigma(i)$ to bound this quantity. The high-level idea is to develop a charging scheme, which charges each unit of delay to a specific packet.

Lemma 1. *Consider packets i and j. Suppose that their paths separate afer crossing some edge e. Then their paths will never rejoin.*

Proof. Say that after crossing, i fixes an earlier bit than j next. Then by the left-to-right ordering, j's path can never fix that bit, so i and j will never visit the same vertex again. □

Lemma 2. *If i and j both cross e and e', and i crosses e before j, then i also crosses e' before j.*

Proof. By Lemma 1, i and j follow the same path between e and e'. The result then follows from FIFO queues and induction on the path. □

Fix packet i with path P_i. Let S_i be the set of all packets that cross one or more edges of P_i (this includes i itself).

Lemma 3. *The delay of i is at most $|S_i|$. (The delay is the difference between the arrival time and $|P_i|$.*

The interesting thing here is that i may repeatedly wait in queues behind the same packet j, so on the surface it looks like an upper bound might only be $|S_i| \cdot |P_i|$. We will show that even if i waits behind j multiple times, each of these waits can be charged to a different packet in S_i.

Proof. Fix packet i. For any packet $i' \in S_i$, define the lag of i' at time t as $t - j$ if i' is ready to cross edge e_j at time t. Recall that $P_i = (e_1, \ldots, e_k)$ and e_j is defined with respect to P_i, not $P_{i'}$. The final lag of i (when it crosses e_k) is exactly the delay of i. Clearly, lags never decrease. The lag of i' increases if and only if i' is waiting in a queue at time t.

Consider the minimum lag

$$\ell(t) = \min\{\text{lag of } i' \text{ at time } t : i' \text{ is still on } P_i\}.$$

Consider a timestep t when $\ell(t)$ increases, *i.e.*, $\ell(t+1) = \ell(t) + 1$. Let i' be a packet that attains $\ell(t)$ at time t. Then i' must have been delayed at time t. Otherwise, it would still have lag $\ell(t)$ at time $t+1$.

Say i' was ready to cross e_j, so $\ell(t) = t - j$. Instead, some $\hat{\imath}$ crossed e_j. We know $\hat{\imath} \in S_i$ and $\hat{\imath}$ is on P_i at time t. Because $\hat{\imath}$ was ready to cross e_j at time t, it too had lag $\ell(t) = t - j$. If $\hat{\imath}$ did not diverge from P_i next, it would have to next cross e_{j+1}. But then its lag at time $t+1$ is $(t+1) - (j+1) = t - j = \ell(t)$, and we chose t such that $\ell(t+1) > \ell(t)$. But $\hat{\imath}$ would ensure $\ell(t+1) = \ell(t)$. So $\hat{\imath}$ must diverge, and by Lemma 1 will never rejoin P_i.

We charge the lag increase to $\hat{\imath}$, and every packet $\hat{\imath} \in S_i$ is charged at most once. So, $\ell(t) \leq |S_i|$ for all i. Finally, at the step t when i crosses e_k, it has minimum lag (because it is furthest ahead on P_i) so it has lag $\ell(t) \leq |S_i|$, and its lag equals its delay. $\square$

3 Tail Bound Application

The remaining step is to use the random destination $\sigma(i)$ to bound the size of $|S_i|$. Let H_{ij} be an indicator variable representing whether $P_i \cap P_j \neq \emptyset$. Then

$$|S_i| = \sum_j H_{ij},$$

$$\mathbb{E}[|S_i|] = \sum_j \Pr[P_i \cap P_j \neq 0]$$

We are interested in the maximum delay of any packet, which is upper-bounded by $\max_i |S_i|$. For fixed i, the H_{ij} as we vary j are mutually independent, so we will apply Chernoff bounds, then take a union bound.

Calculating $\Pr[P_i \cap P_j \neq 0]$ is not obvious, so we will upper-bound it by $H_{ij} \leq |P_i \cap P_j|$. Thus

$$\begin{aligned}
\mathbb{E}[|S_i| \mid P_i] &\leq \sum_j \mathbb{E}[|P_i \cap P_j|] \\
&= \sum_{e \in P_i} \mathbb{E}[\#j : e \in P_j]
\end{aligned}$$

Focus on one edge $e = (u, v)$ and calculate $\mathbb{E}[\#j : e \in P_j]$. Without loss of generality, $u = a0b$, $v = a1b$ because e just flips one bit. Because e is on the path P_i, we know $i = a'0b$ and $\sigma(i) = a1b'$ for some a', b'. If e is on the path P_j, then $j = a''0b$ and $\sigma(j) = a1b''$ for some a'', b''.

How many candidate nodes j are there? If the sequence a has length k, then there are 2^k such nodes – one for each bitstring a''. For any such node, the probability of choosing $\sigma(j)$ that will route through e is

$$\frac{2^{|b|}}{2^d} = 2^{-(k+1)}$$

So,

$$\mathbb{E}[\#j : e \in P_j] = 2^k \cdot 2^{-(k+1)} = \frac{1}{2}.$$

Thus

$$\mathbb{E}[|S_i| \mid P_i] \leq \frac{|P_i|}{2} \leq \frac{d}{2}.$$

So $\mathbb{E}[|S_i|] \leq d/2$ and $|S_i|$ is a sum of independent Bernoullis, so we apply Chernoff bounds (upper bound on expectation version):

$$\Pr[X \geq \Delta] \leq \left(\frac{e\mu}{\Delta}\right)^{\Delta}$$

if $\mu \geq \mathbb{E}[X]$. With $\mu = d/2$, choose $\Delta = 3d$, so

$$\begin{aligned}\Pr[X \geq 3d] &\leq \left(\frac{ed}{6}\right)^{3d} \\ &\leq 2^{-3d}\end{aligned}$$

and since $d = \log_2 n$, this is n^{-3}. Finally, a union bound over n nodes and two routing phases implies that with probability at least $1 - 2/n^2$, all delays in each phase are at most $3d$. Add to this at most $d + d = 2d$ routing steps along edges of P_i.

Therefore with probability at least $1 - 2/n^2$, all packets reach their destinations in at most $8d$ steps. Note that $8d = \mathcal{O}(\log n)$ is exponentially faster $\Omega(\sqrt{n})$.