Lecture Notes CS:5360 Randomized Algorithms
Lectures 16 and 17: Oct 16 and Oct 18, 2018
Scribe: Shreyas Pai

1 Permutation Routing on a Hypercube

Last class we discussed the permutation routing problem in n-dimensional hypercube networks having $N = 2^n$ nodes. We analyzed the following deterministic algorithm which we called the *bit fixing* algorithm. Algorithm 1 describes the execution of this algorithm for each packet that is received and processed at a particular node.

Algorithm 1: Bit Fixing Algorithm(n: dimension, $\vec{a}$: packet source, $\vec{b}$: packet destination)

```
1 for $i \leftarrow 1$ to $n$ do
2     if $a_i \neq b_i$ then
3         send packet along edge in dimension $i$
4     end
5 end
```

Figure 1 illustrates an example of how a packet is routed using the bit fixing algorithm. We also discussed the following randomized routing protocol for the permutation routing problem on a hypercube network.

- **Phase 1:** For each packet we pick an intermediate destination independently and uniformly at random. Packets are sent to their intermediate destination using the bit fixing protocol (Algorithm 1)
- **Phase 2:** Packets are sent from their intermediate destination to their actual destination using the bit fixing protocol (Algorithm 1)

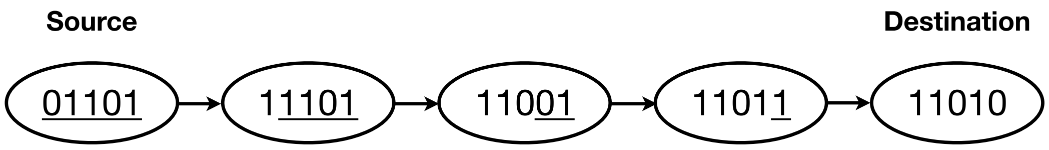

Figure 1: This figure shows the bit fixing path take by a packet with source 01101 and destination 11010 on a 5 dimensional hypercube network. The underlined portion of the ID indicates the bits that still need to be fixed by the bit fixing algorithm.

1.1 Digression: Lower Tail Chernoff Bounds

During the analysis we will require lower tail Chernoff bounds. We haven't seen the lower tail versions of Chernoff bounds so we make a digression to discuss lower tail Chernoff bounds before we launch into the analysis. The proof is similar to upper tail Chernoff bounds so we just state the different lower tail versions here without proof.

Let $X_1, X_2, \ldots, X_n$ be mutually independent binary random variables and let $\Pr[X_i = 1] = p_i \quad \forall i \in \{1, \ldots, n\}$. Let $X = \sum_i X_i$ and denote $\mathbb{E}[X]$ by μ (Note: $\mu = \sum_i p_i$). For $0 < \delta < 1$

(a) $\Pr[X \leq (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^{\mu}$

(b) $\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2}{2}\cdot\mu}$

Note that there is no equivalent of the upper tail form (c) in the lower tail regime because X takes on non-negative integer values.

Using form (b) of both upper abd lower tail bounds we get –

$$\Pr[|X - \mu| \leq \delta\mu] \leq 2e^{-\frac{\delta^2}{3}\cdot\mu}$$

1.2 Analysis

We analyze the two phases separately. The main assumption we will make in the analysis is that the nodes finish executing Phase 1 before moving on to Phase 2. This is because we don't want the Phase 2 packets causing congestion and delaying Phase 1 further than it should. This assumption does not hold in the algorithm but we will see how to relax it when we finish the analysis.

Let M be a packet (or Message) that travels along a bit fixing path P in Phase 1. Let $T_1(M)$ be the number of steps it takes for M to reach its Phase 1 destination, and let $X(e)$ be the number of packets that traverse edge e in Phase 1.

Observation 1

$$T_1(M) \leq \sum_{e \in P} X(e)$$

This is true because the packet can get unlucky and be placed at the end of the queue for every edge e it wants to traverse, therefore being delayed by $X(e)$ steps at edge e.

Our focus will now be on this upper bound of $T_1(M\backslash)$. Note that the upper bound does not depend on M at all so we define for every bit fixing path P –

$$T_1(P) = \sum_{e \in P} X(e)$$

And we now focus our attention on a particular path P.

High Level Idea

In order to achieve our goal, we will proceed as follows.

(1) We first show that the number of packets that are candidates for traversing an edge $e \in P$ is less than $6n$ with high probability.

(2) Conditioning on (1), we show that $T_1(P) \leq 30n$ with high probability.

Definition 2 *Let $P = (v_0, v_1, \ldots, v_{m-1}, v_m)$ be a path with $m+1$ vertices (and therefore m edges). Suppose that traversing the edge (v_{i-1}, v_i) fixes the j^{th} bit of a packet. We say that a packet M is* active at node v_{i-1} *id it reaches v_{i-1} before its j^{th} bit is fixed.*

A packet M is active for a path P *is it is active for some node v_{i-1} in P.*

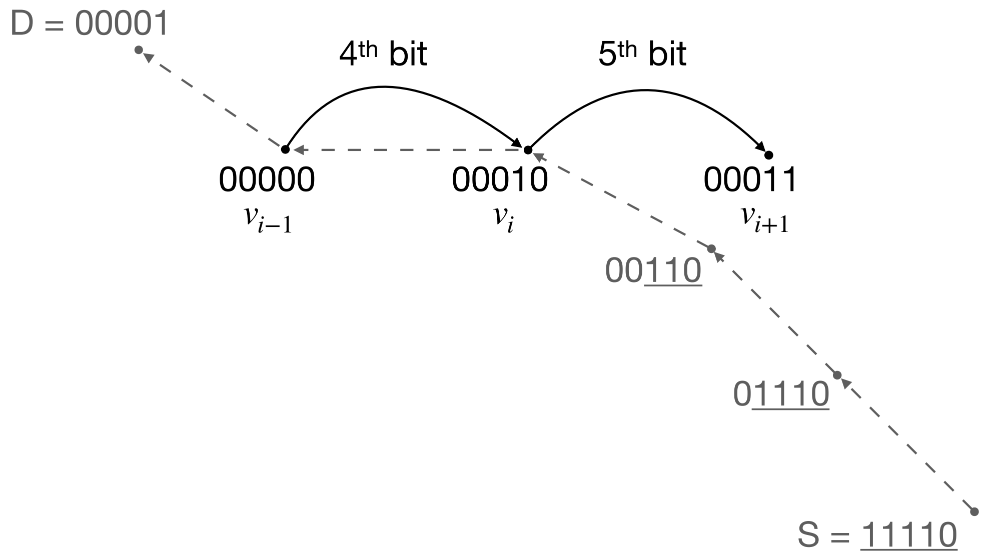

Figure 2: Example of active and non-active packets in a 5-dimensional hypercube network. In bold are two edges of the bit fixing path P where (v_{i-1}, v_i) fixes the 4^{th} bit and (v_i, v_{i+1}) fixes the 5^{th} bit. The packet M (trajectory indicated by dashed lines) is not active for v_{i-1} because it reaches there after its 4^{th} bit is fixed but it is active for v_i because it reaches there before its 5^{th} bit is fixed. Therefore M is active for the path P

A packet is a candidate for traversing an edge in a path (as in (1)) if it is active for the path. Note that active packets may or may not travel along the path but inactive packets will definitely not travel along the path. Therefore, counting the number of active packets gives us an upper bound on the number of packets that travers a path P. So let H denote the number of packets that are active for path P.

Using this new notation, we can rewrite our high level idea mathematically. Recall our goal is to show that $\Pr[T_1(P) \geq 30n]$ is very small. Therefore we can write using the law of total probability –

$$\begin{aligned}\Pr[T_1(P) \geq 30n] &= \Pr[T_1(P) \geq 30n \mid H < 6n] \cdot \Pr[H < 6n] + \Pr[T_1(P) \geq 30n \mid H \geq 6n] \cdot \Pr[H \geq 6n] \\ &\leq 1 \cdot \Pr[H < 6n] + \Pr[T_1(P) \geq 30n \mid H \geq 6n] \cdot 1 \\ &= \Pr[H < 6n] + \Pr[T_1(P) \geq 30n \mid H \geq 6n]\end{aligned}$$

Therefore, our goal now is to bound $\Pr[H < 6n]$. We wish to use Chernoff bounds and therefore need to express H as a sum of independent binary random variables. For $i = 1, \ldots, N$ let,

$$H_i = \begin{cases} 1 & \text{if packet with source } i \text{ is active for } P \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that $H = \sum_i H_i$. Whether a packet is active or not depends entirely on its source and destination and since the destinations are picked independently for each packet, the $H_i's$ are mutually independent. We now calculate $\mathbb{E}[H]$.

Suppose $v_{i-1} = (a_1, a_2, \ldots, a_{j-1}, b_j, b_{j+1}, \ldots, b_n)$ and $v_i = (a_1, a_2, \ldots, a_{j-1}, a_j, b_{j+1}, \ldots, b_n)$

Packets that are active for v_{i-1} have sources of the form – $(*, *, \ldots, *, b_j, b_{j+1}, \ldots, b_n)$. Therefore there are at most 2^{j-1} such packets. Each of these packets will have a destination of the form $(a_1, a_2, \ldots, a_{j-1}, *, *, \ldots, *)$. The probability that a packet has destination of such a form is $1/2^{j-1}$.

Therefore, the expected number of packets active at v_{i-1} is 1. Note that H is just the sum of active packets at each node in the path and we have

$$\mathbb{E}[H] = \sum_{i=0}^{m-1} \mathbb{E}[\text{ of packets at } v_i] = m \leq n$$

Using form (c) of the upper tail Chernoff bounds with $R = 6n \geq 6\mathbb{E}[H]$, we get–

$$\Pr[H \geq 6n] \leq 2^{-6n} = \frac{1}{N^6}$$

Therefore we have –

$$\Pr[T_1(P) \geq 30n] \leq \frac{1}{N^6} + \Pr[T_1(P) \geq 30n \mid H \geq 6n]$$

Now we wish to bound $\Pr[T_1(P) \geq 30n \mid H \geq 6n]$.

Observation 3 *Let M be an active for P. If M leaves P, then it does not return back to P*

This is because the bit fixing algorithm does not touch the bits that it has already fixed. If M leaves the path P when the j^{th} bit is fixed then the j^{th} bit of M's trajectory will be different from the j^{th} bit of all the subsequent nodes in the path P. Note that M can also leave P without ever traversing along an edge in P.

Consider the following randomized trial as a thought experiment: For each active packet M of P, when M is at a node in P, we toss a biased coin and decide whether M traverses the next edge of P or leaves P based on the coin toss. A *success* or *heads* imples the packet leaves the path (to never come back) and a *failure* or *tails* implies the packet stays on the path.

Observation 4 $Pr[success] \geq 1/2$

To see this, consider an equivalent version of the algorithm where the destination bits are computed "on the fly" as opposed to being determined at the source. In particular, when a node v_{i-1} receives a packet whose $j-1$ destination bits are determined (or fixed), it computes the j^{th} bit by tossing a coin. Therefore, in order for the packet to stay on the path, v_{i-1} needs to get the correct outcome that sends the packet along the edge (v_{i-1}, v_i). And the success probability will be at least $1/2$ because the active packets that v_{i-1} receives need not have all $j-1$ bits fixed, so it has to make more coin tosses first that allow the packet to stay at v_{i-1} before the j^{th} bit is fixed.

Note that $T_1(P)$ is the total number of failures and the total number of successes is $\leq H < 6n$. But we don't know the total number of trials so we can't directly bound $\Pr[T_1(P) \geq 30n \mid H < 6n]$.

Let Z = number of successes when an unbiased coin is tossed $36n$ times.

Claim 5 $Pr[T_1(P) \geq 30n | H < 6n] \leq Pr[Z < 6n]$

We skipped the proof in class and noted that it can be proved using induction.

So now we focus on upper bounding $\Pr[Z < 6n]$. Note that $\mathbb{E}[Z] = 18n$, therefore by form (b) of lower tail Chernoff bound we get –

$$\begin{aligned}\Pr[Z < 6n] &\leq \Pr[Z \leq (1-2/3)18n] \\ &\leq e^{-\frac{(2/3)^2}{2}\cdot 18n} \\ &= e^{-4n} = \frac{1}{N^4}\end{aligned}$$

Therefore, $\Pr[T_1(P) \geq 30n | H < 6n] \leq N^{-4}$ which implies $\Pr[T_1(P) \geq 30n] \leq N^{-6} + N^{-4}$

Now consider all possible bit fixing paths P: there are $N \times N = N^2$ different bit fixing paths. Therefore, by the union bound –

$$\Pr[\exists P : T_1(P) \geq 30n] \leq N^2(N^{-6} + N^{-4}) \leq \frac{2}{N^2} \leq \frac{1}{N}$$

Notice that for Phase 2, the same analysis will go through but the only thing we would need to change is that the sources are picked independently and uniformly at random instead of the destinations. Therefore we get

$$\Pr[\exists P : T_2(P) \geq 30n] \leq \frac{1}{N}$$

We had assumed that Phase 2 starts after Phase 1 has completed. Note that $T_1(P)+T_2(P) \geq 60n$ implies either $T_1(P) \geq 30n$ or $T_2(P) \geq 30n$. Therefore –

$$Pr[\exists P : T_1(P) + T_2(P) \geq 60n] \leq \max\{\Pr[\exists P : T_1(P) \geq 30n], \Pr[\exists P : T_2(P) \geq 30n]\} \leq \frac{1}{N}$$

Therefore, we get a probabilistic bound on the running time of the randomized routing algorithm. This result is encapsulated in the following theorem –

Theorem 6 *With high probability, the randomized routing algorithm delivers all packets to their destination in $O(n)$ time steps.*

2 Randomized Data Structures: Skip Lists

We would like to design a data structure that takes $O(\log n)$ time for *insert*, *delete*, and *search* operations. There are many deterministic data structures that provide such guarantees. Examples are variants of balanced binary search trees like AVL and Red-Black trees.

Skip lists are a simple[1] alternative. The idea is to create different levels of linked lists. The list at level 0 (L_0) contains all elements in sorted order. The list at level i (L_i) is a sublist of the list at level $i-1$. The lists at higher levels serve as "highways" for the query to "skip" significant portions of the lists at lower levels.

There are many ways to implement skip lists. Figure **??** illustrates an implementation where L_i is created by picking every other element from the list L_{i-1}. Having this invariant allows us to perform search in $O(\log n)$ time but it is difficult to efficiently maintain the invariant under insertions and deletetions.

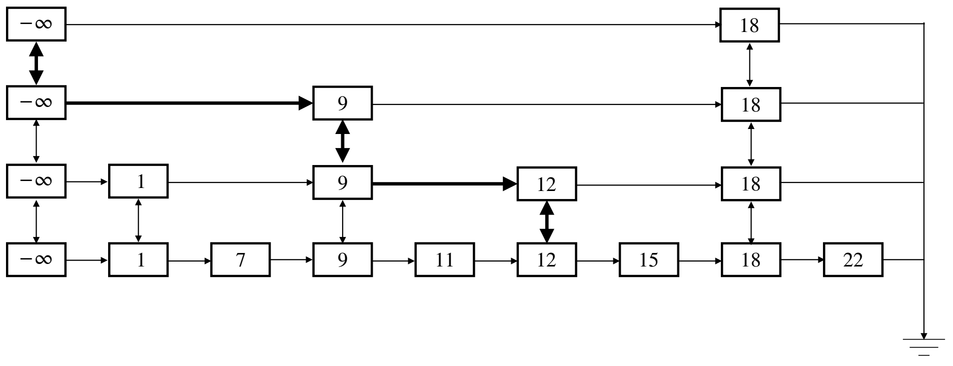

Figure 3: Example of a deterministic skip list where the higher level is created by picking alternate elements from the lower level. The bold arrows show the path taken by a search query for element 14 (since 14 is not in the list, we return the closest smaller element 12). The $-\infty$ elements are added at the beginning for convenience and to give the search algorithm a concrete starting point irrespective of the changes in the data structure.

In the next lecture we will look at the performance of a randomized variant of skip lists where we create the list L_i by picking each element from L_{i-1} independently with probability $1/2$.

[1]The reader can (and should) decide for themselves whether this data structure is really that simple.