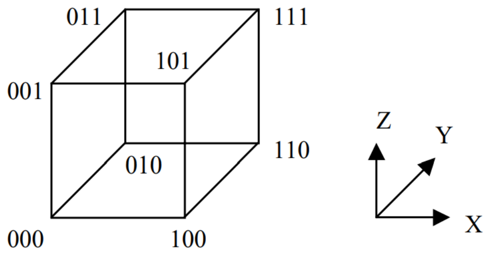
COMP 480/580 — Probabilistic Algorithms and Data Structure April 9, 2019

Lecture 22

Lecturer: Anshumali Shrivastava *Scribe By: Congmin Qiu*

1 Randomized Routing

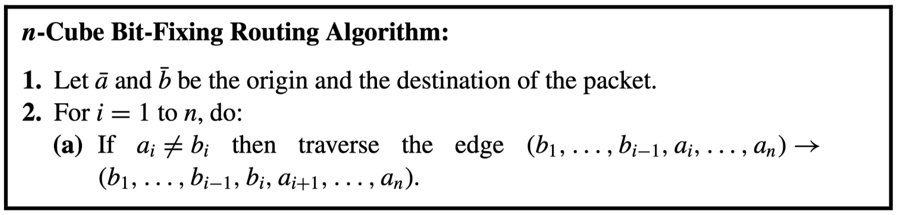
In this section, we study sending messages in a network. We take the hypercube as an example, but the methods we describe are easy to adapt to other connection networks. In the hypercube, we have 8 processors/nodes which we represent as 3-bit numbers, from 000 to 111. The lines between nodes are the links between nodes upon which we can transmit messages.



There are two constraints: 1. Every link/edge can only transmit one message at a time. 2. The current processor can only send a message directly to a processor that is 1 hamming distance from the current processor. For example, 000 can send a message to one of (001, 010, 100), but it cannot send a message to 111 in one step.

Every message consists of two parts: 1. The content of the message 2. The binary representation destination of this message, e.g., 101. Each processor can only decide which neighbor to transmit a message to based on limited information. In particular, none of the processors have enough information to calculate the shortest path to the destination.

How does each processor decide which neighbor to send the message to? The answer is known as the **Bit-fixing** strategy[**?**]:

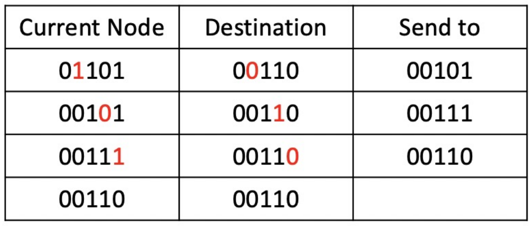


```
n-Cube Bit-Fixing Routing Algorithm:
1. Let $\bar{a}$ and $\bar{b}$ be the origin and the destination of the packet.
2. For $i = 1$ to $n$, do:
   (a) If $a_i \neq b_i$ then traverse the edge $(b_1, \ldots, b_{i-1}, a_i, \ldots, a_n) \rightarrow (b_1, \ldots, b_{i-1}, b_i, a_{i+1}, \ldots, a_n)$.
```

For example, we start from node(01101), and it holds a message whose destination is 00110. The path of the message through the network according to the Bit-Fixing strategy will be the path shown in Figure **??**. If we have N nodes in the network, the ID may be represented as $\log(N)$ bits. In expectation, half of the bits will be different between source to destination. So it will cost $\log(N)/2$ steps from source to destination.

Returning to our hypercube network, suppose that processor number i wants to send a message v_i to destination $\text{dest}(i)$. We assume that the computer runs synchronously, so that all processors try to send their message at the same time. In **permutation routing**, every processor tries to send a message to a different destination. That is, the function dest(i) is a permutation. [?]

What happens if two messages arrive at a processor and attempt to leave via the same link? In this situation, a **collision** will occur. See Figure **??** for intuition on how this can occur.



Current Node	Destination	Send to
01101	00110	00101
00101	00110	00111
00111	00110	00110
00110	00110	

Figure 1: From 01101 to 00110

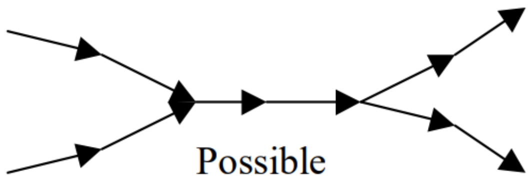


Figure 2: Collision

During the lecture, a student mentioned if it's possible for two roads to collide twice, the slides [?] mean that Figure **??** is impossible. In response, we refer the reader to the proof on page 4 and 5 of [?].

Due to collisions, using only the bit-fixing routes can lead to high levels of congestion and poor performance. There are certain permutations that cause the bit-fixing routes to behave poorly. However, it turns out that the network performs well if each message is sent from a source to a destination chosen uniformly at random. This motivates the two-phase approach: first route each message to a randomly chosen intermediate point and then route it from this intermediate point to its final destination.

First, we define an indicator random variable H_{ij}. $H_{ij} = 1$ when message i's and message j's roads share one link. $H_{ij} = 0$ if there is no shared link. Then for a certain message, the number of steps from source to destination is the sum of (1) regular step using Bit-fixing strategy and (2) delays caused by collisions. The first part is $\log(N)/2$ as we mentioned in the first page. The second part can be calculated as the expectation of the number of messages that shared a link with message i.

$$\text{Total Steps} = \log(N)/2 + \mathbb{E}[\sum_{j=1}^{N} H_{ij}]$$

Now we need to compute the expectation. First, suppose the route of message v_i consists of the edges $(e_1, e_2, ..., e_k)$. We will use $T(e)$ to denote the number of routes that pass through a given edge e. We have

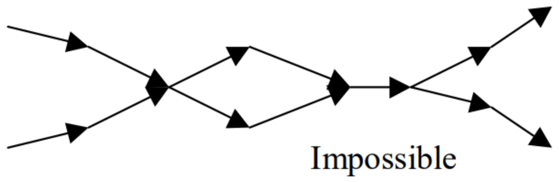


Figure 3: It's impossible to collide twice for two messages.

Source Processor Id	1	2	...	N
Intermediate Points	random(1)	random(2)	...	random(N)
Destination	dest(1)	dest(2)	...	dest(N)

$$\sum_{j-1}^{N} H_{ij} < \sum_{l=1}^{k} T(e_l)$$

While the expectation of the number of messages per link is

$$E[T(e)] = \frac{\text{sum of lengths of all routes}}{\text{total edges in the network}}$$

Our hypercube network has 12 links. Since each link is bi-directional, there are 24 edges, which is the denominator of the equation above. There will be 8 messages sent out from 8 processors. In expectation, they will travel through $n/2$ links before reaching their destination, where n is the number of bits in the address ($n = 3here$). So $\mathbb{E}[T(e)] = (N \times n/2)/(N \times n) = \frac{1}{2}$. Therefore $\sum_{j-1}^{N} H_{ij}$ is upper bounded by $k/2$.

$$\sum_{j-1}^{N} H_{ij} \leq \frac{k}{2} \leq \frac{n}{2}$$

where k is the number of links that message i passed through, $n = \log N$ is the number of bits, and N is the number of nodes. Using the Chernoff bound, we have

$$Pr[\sum H_{ij} > (1 - \delta)\mu] \leq 2^{-\mu\delta}$$

This inequality is useful to reason about the performance of the system. For example, suppose we want to calculate the probability for message i to be delayed by at least $3n$ steps. This is

$$Pr[\sum H_{ij} > 3n]$$

Then using our inequality we have that $(1 + \delta)\mu = \mu + \delta\mu = 3n$ and $\mu \leq 0.5n$, so $\mu\delta \geq 2.5n$. This means that probability of more than $3n$ steps is bounded by $2^{-2.5n}$. For our hypercube example, the number of steps if we do not consider collisions is 1.5 steps. The probability for a message to be delayed by at least $3n = 9$ steps is less than $2^{-2.5*3} = 0.00552427$.

References

[1] Mitzenmacher, Michael, and Eli Upfal. Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis. Cambridge university press, 2017.

[2] https://people.eecs.berkeley.edu/ jfc/cs174/lecs/lec11/lec11.pdf