COMPSCI 630: Randomized Algorithms Feb 11, 2018

Lecture Oblivious Routing

Lecturer: Rong Ge *Scribe: Xingyu Chen*

1 Introduction

In this class, we will mainly focus on oblivious routing problem. This lecture will cover the design and analysis of the randomized algorithms solving this problem with the promise that each message will only incur linear delay.

2 Problem Definition

2.1 Routing Problem

The routing problem is defined as follows:

- Suppose we have N machines connected by a network
- Each link can send 1 message/packet synchronously at every time step
- Goal is to minimize the latency for each message

2.2 Oblivious Routing Problem

However, the problem we are going to look at is a special category of routing problem called **oblivious routing**, which is defined as:

- Given a permutation $\pi : \{0,1,...N-1\} \to \{0,1,...N-1\}$.
- Given a network $G = (V,E)$, where V contains $N = 2^n$ machines.
- Given each machine a unique n-bit binary identifier. There exists an edge $e = (i,j)$ if the binary number of machine i and machine j differs in only 1 bit.
- Machine i requests to send one message to machine $\pi(i)$.
- Goal is to find an algorithm to minimize the latency for each message.

Remark 1. *Here, the word **oblivious** means that we have no knowledge of other machines' request or decisions. The machine needs to decide a path locally and independently.*

3 Deterministic Oblivious Routing Algorithm

First, we try to show that all attempts of deterministic algorithm will actually fail.

One algorithm falls into the category of the deterministic oblivious routing algorithm if

1. For every pair (i, j), the algorithm computes a deterministic path from i to j.
2. The algorithm always uses this precomputed path to send the message from i to j.

3.1 Example

One example of deterministic oblivious routing algorithm is the **Bit Walk** Algorithm. The algorithm can be defined below:

1. Give each machine a binary number representation. E.g. Machine 7 should be 0111, Machine 9 should be 1001.
2. For each i to j path, scan from left to right (in binary) and flip bits that are different. For example consider the path from Machine 5 to Machine 14. Then the path will be

   $$0101 \rightarrow 1101 \rightarrow 1111 \rightarrow 1110$$

   which is,

   $$Machine5 \rightarrow Machine13 \rightarrow Machine15 \rightarrow Machine14$$

3.2 Failure for All Deterministic Algorithm

For all deterministic routing strategy, we have the following theorem,

Theorem 1. *For any deterministic routing strategy, there is a permutation that recalls at least* $\sqrt{\frac{N}{deg}}$ *time in delay, where deg is the degree of the graph.*

The theorem isn't proved during the lecture. But just from its conclusion, we can see that all deterministic algorithms are ruled out for our problem. Since for any deterministic algorithm we design, there's a permutation that can incur $\sqrt{\frac{N}{n}}$ delay. This delay can be exponential over n.

4 Randomized Oblivious Routing Algorithm

So we switch our attention to randomized algorithm. The Algorithm we are going analyze is:

1. For each i, pick $\sigma(i) \in \{0, 1, \ldots N-1\}$ uniformly at random.
2. Deliver the message using this path, $i \rightarrow \sigma(i) \rightarrow \pi(i)$.
3. Send the message from machine i to machine $\sigma(i)$, then machine $\sigma(i) \rightarrow \pi(i)$ using **Bit Walk** algorithm we mentioned before

We will propose the main lemma but delay the proof. The main lemma is stated below in **Lemma 2**,

Lemma 2. *Let S be the set of paths* $(j \rightarrow \sigma(j))$ *that intersects with the path from i to* $\sigma(i)$*, then the path from* $i \rightarrow \sigma(i)$ *takes at most* $|S| + n$ *time.*

4.1 From Main Lemma to Theorem

Assume the main lemma is correct, let's try to prove the following result, as in **Theorem 3**,

Theorem 3. *For each path from i to $\sigma(i)$, the maximum delay, in expectation, is at most $6n$ time.*

Let $H_{i,j} = 1$ if path $i \to \sigma(i)$ and path $j \to \sigma(j)$ intersect. So we have,

$$|S| = \sum_{j \neq i} H_{i,j}$$

Since the number of edges in all N paths is $N \cdot \frac{n}{2}$, the number of edges in graph is as well $N \cdot \frac{n}{2}$, the expected number of paths containing every edge is 1. If we take the expectation on both sides, we have

$$\begin{aligned}
\mathbf{E}[|S|] &= \sum_{j \neq i} \mathbf{E}[H_{i,j}] \\
&\leq \sum_{e \text{ on } i \to \sigma(i)} \mathbf{E}[\text{Number of paths containing e}] \\
&\leq \sum_{e \text{ on } i \to \sigma(i)} 1 \\
&\leq n
\end{aligned}$$

Then use Chernoff bound and apply it to $|S|$, we get,

$$\begin{aligned}
Pr[|S| > (1+\delta)n] &\leq [\frac{e^\delta}{(1+\delta)^{1+\delta}}]^n \qquad \text{Choose } \delta = 4 \\
&\leq [\frac{e^4}{5^5}]^n \\
&<< 2^{-n}
\end{aligned}$$

Finally, we apply Union Bound over all $N = 2^n$ paths. Then we get,

Lemma 4. *With high probability, for all path from i to $\sigma(i)$,*

$$|S| \leq 5n$$

After applying our main lemma, we obtain,

Theorem 5. *For each path from i to $\sigma(i)$, the maximum delay, with high probability, is at most $6n$ time.*

This result is significantly better than any deterministic algorithm can obtain since it is linear instead of exponential over n.

4.2 Proof of Main Lemma

Now the only thing left is the proof of our main lemma. First, we prove a claim,

Claim 6. *For two paths $i \to \sigma(i)$ and $j \to \sigma(j)$, they can only intersect in a consecutive subpath.*

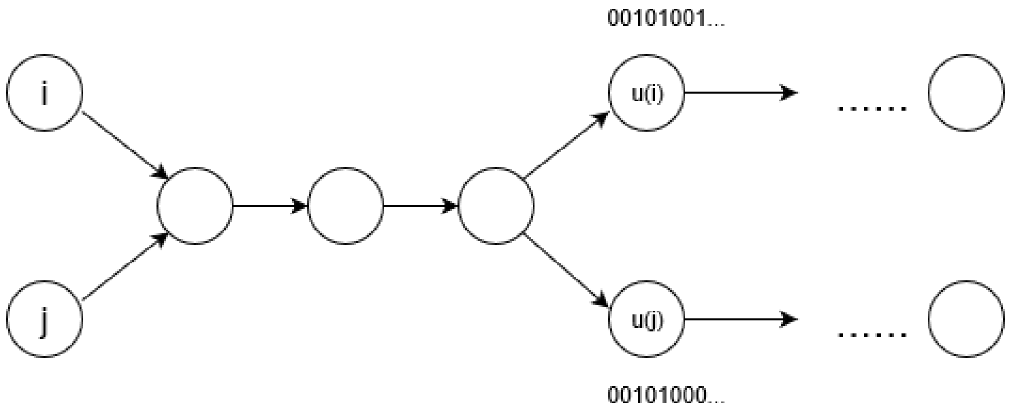

Figure 1: Two Paths Separate

Proof. Consider the first time two paths separate, call the two points u_i and u_j as in Figure 1. Without loss of generality, assume u_i is flipped in an earlier bit. Then after u_i is flipped, if two paths are converged again, it must happen when two binary representations become the same after some flips of **Bit Walk** algorithm. However, at the bit u_i, two paths must have already been different so they can't converge again. ☐

In the later proof, the following definition of delay is used,

Definition 1. *Delay of a message, with respect to path $i \to \sigma(i)$, is defined as $t - j$, where j is the location on path $i \to \sigma(i)$.*

With this definition of delay, we can prove the following claim,

Claim 7. *When delay of message X increases from l to $l+1$, we must have another message with delay l.*

Proof. In order to delay message X from l to $l+1$, there must be another message Y transmitting at the same time on this node. In order for this message Y to show up at this node at this time, it must have delay l. After blocking X, message Y will still have delay l and the message X from $i \to \sigma(i)$ will increase the delay from l to $l+1$. ☐

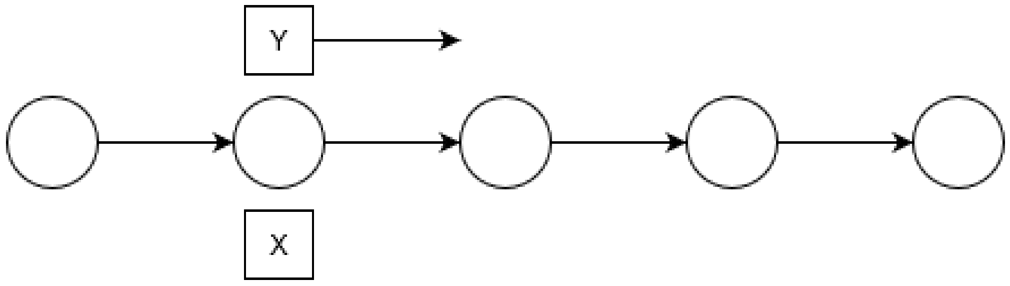

Figure 2: Delay by another message

Claim 8. *If message i from i to $\sigma(i)$ is delayed by delay(i), then for any $l = 0, 1, 2, \ldots, delay(i) - 1$, there must be a message of delay l that has exited from the path.*

Proof. See Figure 3 for graph representation. We keep track of the message X with delay l.

- If X exits, then our assumption is satisfied since this message with delay l has exited.
- If X goes to the next step, X will still have delay l, then we still keep tracking it.
- If X stays at the current location and has delay $l+1$, we can keep track of the message Y that blocked X. Since from the previous claim, we know, if delay of X goes from l to $l+1$, then another message Y must have delay l. We instead keep track of message Y.

To increase the delay of message i to delay(i), there must have been messages with delay ranging from 0 to delay(i) according to **Claim 7**. Further since all messages must have exited the path $i \to \sigma(i)$ at some point, and for each delay l, we have at least one path to keep track of, then for all delay l less than delay(i), there must be a message of delay l that has exited from the path. ☐

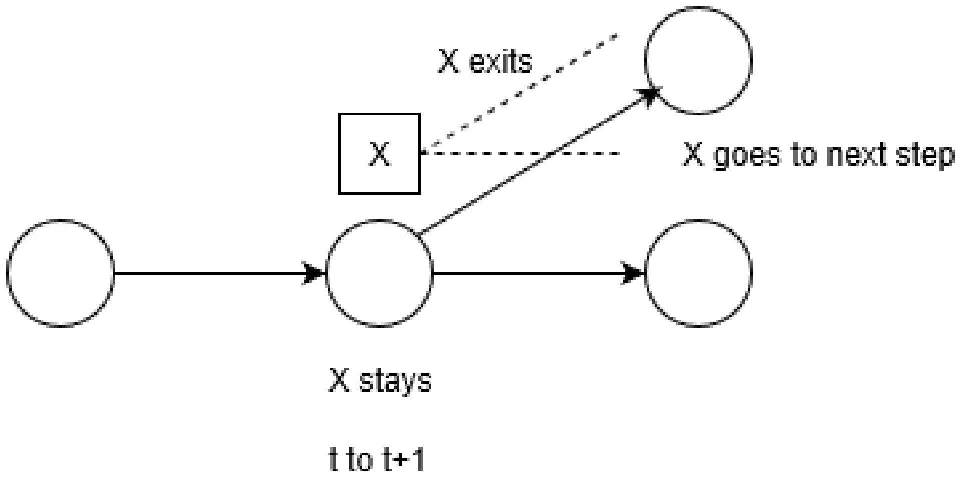

Figure 3: Delay by another message

There are only $|S|$ number of messages intersecting with $i \to \sigma(i)$. If our message from $i \to \sigma(i)$ is delayed by delay(i), then there must have been messages of delay $1, 2, \ldots delay(i) - 1$ exiting the path. So to satisfy **Claim 8**, the maximum delay we can have is $|S|$. If we add the original transmission time n back, we have our main lemma,

Lemma 9. *Let S be the set of paths ($j \to \sigma(j)$) that intersects with the path from i to $\sigma(i)$, then the path $i \to \sigma(i)$ takes at most $|S| + n$ time.*

5 Conclusion

In this lecture, we talk about a randomized algorithm for oblivious routing problem. This algorithm guarantees a linear delay, which is good comparing to the exponential delay that any deterministic algorithm must always incur. In next lecture, we will talk about Power of Two Choices.