CMPUT 675: Randomized Algorithms **Fall 2005**

Lecture 6: Sept 27

Lecturer: Mohammad R. Salavatipour *Scribe: Jun Ma*

6.1 Packet Routing Problem

6.1.1 Properties of the Network

Recall the definition of problem from previous lecture. We have a network with N nodes. The nodes communicate with each other through directed edges. Every node i has a packet to be sent to a distinct destination $d(i)$, i.e. the destinations form a permutation of the nodes. The system is synchronous and works in rounds. In every round vertices may send some packets to some of their neighbors. Every edge can transmit only one packet in every time step and it takes exactly one time step for a packet to travel an edge. If there are several packets at a node that want to go through an edge we buffer them (queue) and send them one by one. We usually use FIFO queuing at buffers. We are looking for a routing scheme for these packets with minimum number of rounds. The scheme must be *oblivious*; i.e. every packet is routed independently, without regard to where other packets are routed. At every node we decide where to send each packet based only on its destination $d(i)$. The model of network we consider is a k-dimensional hypercube.

Definition 6.1 *A hypercube or k-cube has 2^k nodes, with k-bit binary strings as labels for each node. Two nodes are connected with directed edges if and only if their labels differ in exactly one bit. Thus in a k-cube, each node is of degree k.*

Given packet v_i to be sent to destination $d(i)$, assuming the current node is $\sigma(i)$, the bit-fixing algorithm compares $d(i)$ and $\sigma(i)$ and finds the leftmost bit that they differ, say bit j. Then sends the packet along the edge from $\sigma(i)$ to the node that differs from $\sigma(i)$ in bit j. Thus bit j of the new node is the same as $d(i)$. Clearly by this scheme, the total number of rounds for a single packet (not considering delays that may be caused by other packets) is at most k (i.e. diameter of a k-cube).

Theorem 6.2 *For any deterministic algorithm, there is an instance that requires $\Omega\left(\sqrt{\frac{N}{d}}\right)$ steps.*

In the case of the hypercube, the bound is $\Omega\left(\frac{\sqrt{2^k}}{k}\right)$. Our goal is to route all the packets in $O(\log N)$ rounds, where $N = 2^k$ (i.e. $O(k)$ rounds). The idea was to first route every packet to an arbitrary location in the network and then route it from that intermediate destination to its original destination. So we will have the following two-phase algorithm:

Two Phase Algorithm
Phase 1: Route packets to a random location (not a permutation!) using bit-fixing.
Phase 2: Route packets from the random intermediate locations to their destination using bit-fixing.

Let's assume that all the packets finish phase I before any packet starts phase II and focus on phase I. We show that w.h.p. all packets will arrive at their intermediate destination (finish phase I) in $O(k)$ rounds. Phase II is symmetric to phase I in analysis.

Definition 6.3 *For any edge e_i, let $T(e_i)$ be the number of routes that use edge e_i.*

By Symmetry, $\mathrm{E}[T(e_i)]$ is independent of e_i. Expected length of a path is $\frac{k}{2}$, and total length of paths for all packets is $\frac{kN}{2}$. Since total number of edges is kN, we get $\mathrm{E}[T(e_i)] = \frac{1}{2}$. We will use the following fact (whose proof is simple).

Fact: Once two routes that intersect separate from each other, they cannot intersect again.

Consider a fixed packet v_i and assume that $\rho_i = e_1, e_2, \ldots, e_k$ is the route for v_i. Denote the set of packets that intersect ρ_i by S.

Lemma 6.4 *The delay for v_i is at most $|S|$.*

Proof: Charge each delay that we have for packet v_i to the separation of a route from ρ_i. A packet that is waiting to follow edge e_j at time t is called to have lag $t - j$. The delay for v_i is the lag when crossing e_k. Note that when the delay for v_i goes up from l to $l+1$, some packet from S has lag l because some packet from S is using edge e_j instead of v_i at time t (otherwise v_i would have used e_j). Let time t' be the last time step where there is some packet w with lag l. Say w crosses some edge $e_{j'}$ at time t' and has lag l. All other packets that were waiting to cross $e_{j'}$ the lag is increased by 1. We claim that w departs in this step. Otherwise, in the next time step w still has lag t', contradicting the definition of t'. Thus we can charge w the one increase in the lag value. ■

Define an indicator random variable $H_{i,j} = 1$, if and only if packet i and packet j share an edge. Let D_i be the value of delay for packet i. Then $D_i = |S|$ and

$$\begin{aligned}
\mathrm{E}[D_i] = \mathrm{E}\left[\sum_{j=1}^{N} H_{i,j}\right] &= \sum_{j=1}^{N} \mathrm{E}\left[H_{i,j}\right] \\
&\leq \sum_{l=1}^{k} \mathrm{E}\left[T(e_l)\right] \\
&\leq \frac{k}{2}.
\end{aligned}$$

Note that now H_{ij} are independent random variables because the intermediate destinations are selected randomly. So we can apply Chernoff bound:

$$\Pr[D_i \geq 6k]] < 2^{-6k}.$$

Lemma 6.5 *With probability at least $1 - 2^{-5k}$, every packet gets to its intermediate destination in at most $7k$ steps.*

Proof: We have $N = 2^k$ packets each is delayed by more than $6k$ steps with probability at most 2^{-6k} (By Chernoff Bound). Thus the probability of at least one delay being more than $6k$ is at most $2^k 2^{-6k} = 2^{-5k}$. ■

For phase II, similar analysis applies. Thus the total number of time steps is w.h.p. $14k \in O(k)$.

6.2 Randomized Rounding

Consider the following multicommodity network problem. Given a graph $G = (V, E)$, and k pairs of vertices (source/sink) $\{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\}$. The problem is to find a feasible solution which consists of a set of n paths $p_1, p_2, ..., p_n$ such that p_i is a path from s_i to t_i. The congestion of each edge e is the number of paths using the edge. Our goal is to minimize the maximum number of paths that use any edge, that is find a feasible solution with minimum maximum congestion.

Denote the set of all paths between s_i and t_i by $\mathcal{P}_i$ and let $x_{i,p}$ be the indicator variable that is 1 if we pick path $p \in \mathcal{P}_i$. Our goal is to assign $0/1$ values to these variables and minimize the parameter C such that:

$$\begin{array}{ll} \text{subject to} & \sum_{p \in \mathcal{P}_i} x_{i,p} = 1 \quad \forall i \\ & \sum_{p \in \mathcal{P}_i : e \in p} x_{i,p} \le C \quad \forall e \in E \\ & x_{i,p} \in \{0, 1\} \end{array}$$

This is an integer program formulation of our problem. If we relax the constraint on $x_{i,p}$ to be real value between 0 and 1, i.e. $0 \le x_{i,p} \le 1$ (instead of $x_{i,p} \in \{0,1\}$) we obtain a linear programming relaxation of the problem. Note that the solution to linear program (LP) is no more than the solution to the integral program (IP). Now we can solve this LP using LP solvers in polynomial time. One property of the solution to this LP is that it has only a polynomially many number of nonzero variables $x_{i,p}$.

Denote by vector x^* the optimal fractional solution and denote by C^* the value of the optimal fractional solution. For every i, we are going to choose exactly one path from $\mathcal{P}_i$. The probability that we choose $p \in \mathcal{P}_i$ is exactly equal to $x^*_{i,p}$ (note that the sum of these values is 1). Let random variable $Y_e^i = 1$ if and only if the selected path for s_i, t_i contains edge e. Thus, the congestion of edge e is $Y_e = \sum_{i=1}^n Y_e^i$.

$$\begin{aligned} \mathrm{E}[Y_e] &= \mathrm{E}\left[\sum_{i=1}^n Y_e^i\right] \\ &= \sum_{i=1}^n \mathrm{E}[Y_e^i] \\ &= \sum_{i=1}^n \Pr[Y_e^i = 1] \\ &= \sum_{i=1}^n \sum_{p \in \mathcal{P}_i : e \in p} x_p^* \\ &\le C^* \end{aligned}$$

If $\mu = E[Y_e] \ge 1$ then let $1+\alpha = \frac{d \ln n}{\ln \ln n}$ for some constant $d > 0$ where $n = |V|$. So $(1+\alpha)\ln(1+\alpha) - \alpha \ge 3 \ln n$ if d is large enough, which implies:

$$\Pr[Y_e \ge (1+\alpha)\mu] \le e^{-(3 \ln n)} \le \frac{1}{n^3}. \tag{6.1}$$

Since $\mu \le C^* \le OPT$, (6.1) implies that $\Pr[Y_e \ge (1+\alpha)OPT] \le \frac{1}{n^3}$. If $\mu < 1$ then let $\alpha\mu = \frac{d \ln n}{\ln \ln n}$ for large enough d. Then $((1+\alpha)\ln(1+\alpha) - \alpha)\mu \ge 3 \ln n$ for large enough d. Therefore:

$$\Pr[Y_e \ge (1+\alpha)\mu] \le e^{-(3 \ln n)} \le \frac{1}{n^3}. \tag{6.2}$$

Note that in this case the congestion in optimal (integral) solution is at least 1. So the probability that congestion of e is larger than $O(\frac{\ln n}{\ln \ln n} OPT)$ is at most $\frac{1}{n^3}$ by (6.2). In either case, the probability that for edge e, the congestion is larger than OPT by a factor of $O(\frac{\ln n}{\ln \ln n})$ is at most $\frac{1}{n^3}$. Summing this probability over all edges, we see that with probability at least $1 - n^2 \frac{1}{n^3} = 1 - o(1)$ every edge has congestion at most $O(\frac{\ln n}{\ln \ln n} OPT)$.