CS590R: Algorithms for Communications Networks
Notes for Lectures 11 and 12

Gopal Pandurangan

Notes taken By:
Srijan Chakraborty
ID: 999-07-4123
email: schakrab@cs.purdue.edu

Universal Packet Routing

We are given an arbitrary topology network and we are given N packets to route. For each packet we have a source node and a destination node and the route.

Assumptions:

We assume that:

1. Packet routes are *edge-simple*. i.e., one packet does not traverse the same edge twice.
2. In each time step only one packet can traverse an edge.
3. Routes have already been picked as well as the sources and destinations.
4. Every source and destination has an initial queue.
5. Queue sizes in the intermediate steps are restricted. Initial and final queues at the source and destination, respectively, are determined by the particular packet routing problem to be solved, independent of the routing algorithm used.

A *schedule* for a set of packets specifies which move and which wait at each time step. i.e., a schedule specifies the timing for the movement of the packets [1].

Given any underlying network and any selection of routes for the packets, our goal is to produce a schedule for the packets that minimizes the total time and the maximum queue size needed to route all the packets to their respective destinations.

We define the two terms *dilation* and *congestion* as:

Definition 1 Dilation d**:** *The maximum distance from any packet's source to its destination.*

Definition 2 Congestion c**:** *The maximum number of routes through any edge.*

In the wide channel model, where any number of packets can traverse the same edge at the same time, a trivial lower bound on the time required to route all the packets is: $\Omega(d)$. However, according to our assumptions, since only one packet can traverse an edge at one time step, a lower bound on routing all the packets is: $\Omega(c+d)$.

Given any set of paths with congestion c and dilation d in any network, it is straightforward to route all the packets to their destinations in cd steps using queues of size c at each edge. Each packet can be delayed at most $c-1$ steps at each of at most d edges on the way to its destination. However, There are much better schedules. Theorem 1 gives a randomized on-line algorithm that produces a schedule of length $O(c+d\log(Nd))$ using queues of size $O(\log(Nd))$, where N is the total number of packets.

Theorem 1 *There is an on-line algorithm for producing a schedule of length $O(c+d\log(Nd))$ using queues of size $O(\log(Nd))$ with high probability.*

Proof: The algorithm is: For each packet, assign a delay chosen randomly, independently, and uniformly from the interval $[1, \frac{\alpha c}{\log(Nd)}]$, where α is a sufficiently large constant that will be specified later.

First we analyze the *unconstrained* schedule, where more than one packets may traverse the same edge in a single step. In this case, A packet that is assigned a delay of x, waits in its initial queue for x time steps, and then moves on to its final destination without ever stopping.

Consider a particular edge g at a particular time step t. Since at most c different packets pass through g, and for each of these, at most one of the $\frac{\alpha c}{\log(Nd)}$ possible delays sends it through g at time step t, the expected number of nodes traversing g at t is at most: $\frac{\log(Nd)}{\alpha c} \times c = \frac{\log(Nd)}{\alpha}$. If X is the number of packets passing through edge g in a given time step, then using the Chernoff bound we get:

$$\Pr(X > \log(Nd)) = \Pr(X > \alpha.\frac{\log(Nd)}{\alpha}) \leq (\frac{e^{\alpha-1}}{\alpha^{\alpha}})^{\frac{\log(Nd)}{\alpha}} < (\frac{e}{\alpha})^{\log(Nd)}$$

Since, α is sufficiently large, we can set $\alpha = e^{\beta+1}$ for a sufficiently large β. Substituting this value in the above inequality we get:

$$\Pr(X > \log(Nd)) < e^{-\beta\log(Nd)} = \frac{1}{(Nd)^{\beta}}$$

Thus we see that with high probability no more than $O(\log(Nd))$ packets pass through any edge in a given time step.

The schedule we have formulated is not valid, since it is unconstrained. However, each step of the unconstrained schedule can be simulated by $O(\log(Nd))$ steps of a real schedule. This schedule length will be $O(c+d\log(Nd))$ steps using queues of size $O(\log(Nd))$. □

The above online algorithm is still far from the optimal order: $O(c+d)$. There are better schedules possible. Theorem 2 states a much better schedule.

Theorem 2 *For any set of packets whose paths are edge-simple and have congestion c and dilation d, there is a schedule having length $(c+d)2^{O(\log^*(c+d))}$ and maximum queue size $\log(c+d)2^{O(\log^*(c+d))}$ in which at most one packet traverses each edge at each step.*

To prove the above theorem we need some preliminary results. We begin with some terminology.

Definition 3 T-frame: *A T-frame is a sequence of T consecutive time steps.*

Definition 4 Frame congestion: *The* frame congestion, *C, in a T-frame is the largest number of packets that traverse any edge in the frame.*

Definition 5 Relative Congestion: *The* relative congestion, *R in a T-frame is the ratio C/T of the congestion in the frame to the size of the frame.*

The tool we use to prove theorem 2 is *Lovász Local Lemma.* Given a set of "bad" events in a probability space, the lemma provides a simple inequality which, when satisfied, guarantees that with probability greater than zero, no bad event occurs. The inequality relates the probability that each bad event occurs with the dependence among them. A set of events $A_1, A_2, \ldots, A_m$ in a probability space has *dependence* at most d if every event is mutually independent of some set of $m-d-1$ other bad events. The formal definitions of mutually independent events and dependency graph is given below.

Definition 6 *A event E is mutually independent of the events $E_1, \ldots, E_n$ if for any $T \subset [1, \ldots, n]$,*

$$\Pr(E| \cap_{j \in T} E_j) = \Pr(E)$$

Definition 7 *A dependency graph for a set of events $E_1, \ldots, E_n$ has n vertices $1, \ldots, n$. Events E_i is mutually independent of any set of events $\{E_j | j \in T\}$ iff there is no edge in the graph connecting i to any $j \in T$.*

Let $A_1, \ldots, A_n$ be a set of bad events. We want to show that

$$\Pr(\cap_{i=1}^{n} A_i^c) > 0$$

1. If $\sum_{i=1}^{n} \Pr(A_i) < 1$ then $\Pr(\cap_{i=1}^{n} A_i^c) > 0$.
2. If all the A_i's are mutually independent (which means that all the A_i^c's are also independent) and for all i, $\Pr(A_i) < 1$ then $\Pr(\cap_{i=1}^{n} A_i^c) = \prod_{i=1}^{n}(1 - A_i) > 0$.
3. If each A_i depends only on a few other events then we apply the Lovász Local Lemma.

Lovász local lemma has many versions. The following is the symmetric version of the lemma.

Lemma 1 *(**Lovász**) Let $E_1, \ldots, E_n$ be a set of events. Assume that*

1. *For all i, $\Pr(E_i) \leq p$;*
2. *The degree of the dependency graph is bounded by d.*
3. *$4pd \leq 1$.*

Then

$$\Pr(\cap_{1=1}^{n} E_i^c) > 0$$

Proof: Let $S \subset \{1, \ldots, n\}$. We prove by induction on $s = 0, 1, \ldots, n$ that if $|S| \leq s$, for all k

$$\Pr(E_k| \cap_{j \in S} E_j^c) \leq 2p$$

For $s = 0$, $S = \phi$, and the proof is trivial.

Without loss of generality, renumber so that $S = \{1, \ldots, s\}$, and (k, j) is not an edge of the dependency graph for $j > d$. i.e., we renumber such that events that depend on k are at the beginning of the list.

$$\Pr(E_k | E_1^c, \ldots, E_s^c)$$

$$= \frac{\Pr(E_k E_1^c \ldots E_s^c)}{\Pr(E_1^c \ldots E_s^c)}$$

$$= \frac{\Pr(E_k E_1^c \ldots E_d^c | E_{d+1}^c \ldots E_s^c).\Pr(E_{d+1}^c \ldots E_s^c)}{\Pr(E_1^c \ldots E_d^c | E_{d+1}^c \ldots E_s^c).\Pr(E_{d+1}^c \ldots E_s^c)}$$

Now, since $\Pr(A \cap B) \leq \Pr(A)$, for any two events A and B,

$$\Pr(E_k E_1^c \ldots E_d^c | E_{d+1}^c \ldots E_s^c)$$

$$\leq \Pr(E_k | E_{d+1}^c \ldots E_s^c) = \Pr(E_k) \leq p$$

Using the induction hypothesis and applying De-Morgan's law and the union bound, we get:

$$\Pr(E_1^c \ldots E_d^c | E_{d+1}^c \ldots E_s^c)$$

$$\geq 1 - \sum_{i=1}^{d} \Pr(E_i | E_{d+1}^c \ldots E_s^c) \geq 1 - \sum_{i=1}^{d} 2p \geq 1 - 2pd \geq 1/2,$$

since $4pd \leq 1$. Thus,

$$\Pr(E_k | E_1^c, \ldots, E_s^c) \leq \frac{p}{1/2} = 2p$$

$$\Pr(E_1^c \ldots E_n^c) = \Pi_{i=1}^{n} \Pr(E_i^c | E_1^c \ldots E_{i-1}^c)$$

$$= \Pi_{i=1}^{n} (1 - \Pr(E_i | E_1^c \ldots E_{i-1}^c)) \geq \Pi_{i=1}^{n} (1 - 2p) > 0$$

□

The lemma is nonconstructive; for a discrete probability space, it shows only that there exists some elementary outcome that is not in any bad event.
Now we prove the following key lemma that we use for the proof of theorem 2:

Lemma 2 *For any set of packets whose paths are edge-simple and have congestion c and dilation d, there is a schedule of length $O(c + d)$ in which packets never wait in queues and in which the relative congestion in any frame of size* $\log d$ *or greater is at most 1.*

Proof: Without loss of generality we assume that $c = d$ (because we can take the maximum among c and d and replace the smaller one with that).

The first step is to assign initial delay to each packet. The delays are chosen randomly, independently, and uniformly from the range $[1, \alpha d]$, where α is a sufficiently large constant that will be determined later. In the resulting schedule, S_1, a packet that is assigned a delay of x waits in its initial queue for x steps, then moves on without waiting again until it reaches its destination. Thus the length of S_1 (i.e., number of steps required by S_1 to route all packets) is at most $(1+\alpha)d$.

We use the Lovász local lemma to show that with nonzero probability the relative congestion in any frame of size $\log d$ or greater is at most 1. Thus, such a set of delays must exist.

To apply the Lovász local lemma, we associate a bad event with each edge. The bad event for edge g is: more than T packets use g in some T-frame, for $T \geq \log d$. To show that there is a way of choosing the delays so that no bad event occurs, we need to bound the dependence, b, among the bad events and the probability, p, of each individual bad event occuring.

Whether or not a bad event occurs depends solely on the delays assigned to the packets that pass through the corresponding edge. Since at most c packets pass through an edge, and each of these packets passes through at most d other edges, the dependence, b, of the bad events is at most $cd = d^2$.

Now we need to bound the probability p that more than T packets use the same edge g in some T-frame. Let, X is the number of packets that go through an edge g in a frame of size T. Then $E(X)$ is $\frac{1}{\alpha d} dT = T/\alpha$. Using the Chernoff bound we get:

$$\Pr(X > T) = \Pr(X > \alpha.\frac{T}{\alpha}) \leq (\frac{e^{\alpha-1}}{\alpha^{\alpha}})^{T/\alpha} < (\frac{e}{\alpha})^{T}$$

Since, α is sufficiently large, we can set $\alpha = e^{\beta+1}$ for a sufficiently large β. Substituting this value in the above inequality we get:

$$\Pr(X > T) < e^{-\beta T}$$

Since, $T \geq \log d$, we get:

$$\Pr(X > T) < \frac{1}{d^{\beta}}$$

Since we have at most $(1+\alpha)d$ T-frames, the bad event for an edge occurs with probability at most $1/d^{\beta-2}$. therefore, we have:

$$4pb < 4\frac{1}{d^{\beta-2}}d^2 < \frac{4}{d^{\beta-4}}$$

For a sufficiently large β, $4pb < 1$. Thus, by Lovász local lemma, there is some assignment of delays such that the relative congestion in any frame of size $\log d$ or greater is at most 1. □

Now we are ready to prove theorem 2.

Proof: (Theorem 2) For simplicity, without loss of generality, we shall assume that $c = d$, so that the bounds on the length of schedule and queue size are $d2^{O(\log^* d)}$ and $(\log d)2^{O(\log^* d)}$, respectively.

We begin by using Lemma 2 to produce a schedule S_1 in which the number of packets that use an edge in any $\log d$-frame is at most $\log d$. Next we break the schedule into $(1+\alpha)d/\log d \log d$-frames. Finally we view each $\log d$-frame as a routing problem with dilation $\log d$, and congestion $\log d$, and solve it recursively.

Each $\log d$-frame in S_1 can be viewed as a separate scheduling problem where the origin of a packet is its location at the beginning of the frame, and its destination is its location at the end of the frame. If at most $\log d$ packets use each edge in a $\log d$-frame, then the congestion of the problem is $\log d$. The dilation of the problem is also $\log d$ because in $\log d$ time steps a packet can move a distance of at most $\log d$. In order to schedule each frame independently, a packet that arrives at its destination before the last step in the rescheduled frame is forced to wait there until the next frame begins.

Now we can bound the length of the schedule and the size of the queues. The recursion proceeds to a depth of $O(\log^* d)$ at which point the frames have constant size, and at most a constant number of packtes use each edge in each frame. The resulting schedule can be converted to one in which at most one packet uses each edge in each time step by slowing it down by a constant factor. Since the length of the schedule increases by a constant factor during each recursive step, the length of the final schedule is $d2^{O(\log^* d)}$. The bound on the queue size follows from the observation that no packet waits at any one spot (other than its origin and destination) for more than $(\log d)2^{O(\log^* d)}$ consecutive time steps, and in the final schedule at most one packet traverses each edge at each time step. □

We can actually find an even better schedule. Theorem 3 presents the asymptotically best schedule that we can get.

Theorem 3 *For any set of packets with edge-simple paths having congestion c and dilation d, there is a schedule having length $O(c+d)$ and constant maximum queue size in which at most one packet traverses each edge of the network at each step.*

Note that there are no restrictions on the size, topology, or degree of the network or on the number of packets.

References

[1] T. Leighton, B. Maggs, S. Rao. Packet Routing and Job-Shop Scheduling in O(Congestion + Dilation) Steps. Combinatorica 14, pp. 167-186, 1994.