Algorithmic aspects of communication networks	Lecture 17
OR 739 Spring 1997	February 28, 1997

In modern telecommunications networks, management of calls is done in two stages. The first stage admits the call (or rejects it) and decides on the route that will carry the call. Once the call has been accepted and a route determined, the second stage digitizes the information to be transmitted (only if necessary, such as with a voice call), partitions the data into packets and sends the packets through the network via the route selected for that call.

We have spent considerable time on the first stage, admission control and routing. We now consider the second stage.

The input to our model will be a set of paths on a graph representing the routes between the origin-destination pairs of calls accepted. These paths may or may not share edges with each other. Each path has some stream of packets that it wants to send along the path. The difficultly arises because each edge can only forward one packet per unit of time. When paths share an edge, two packets might arrive at an edge at the same time, one of which will have to wait for the other.

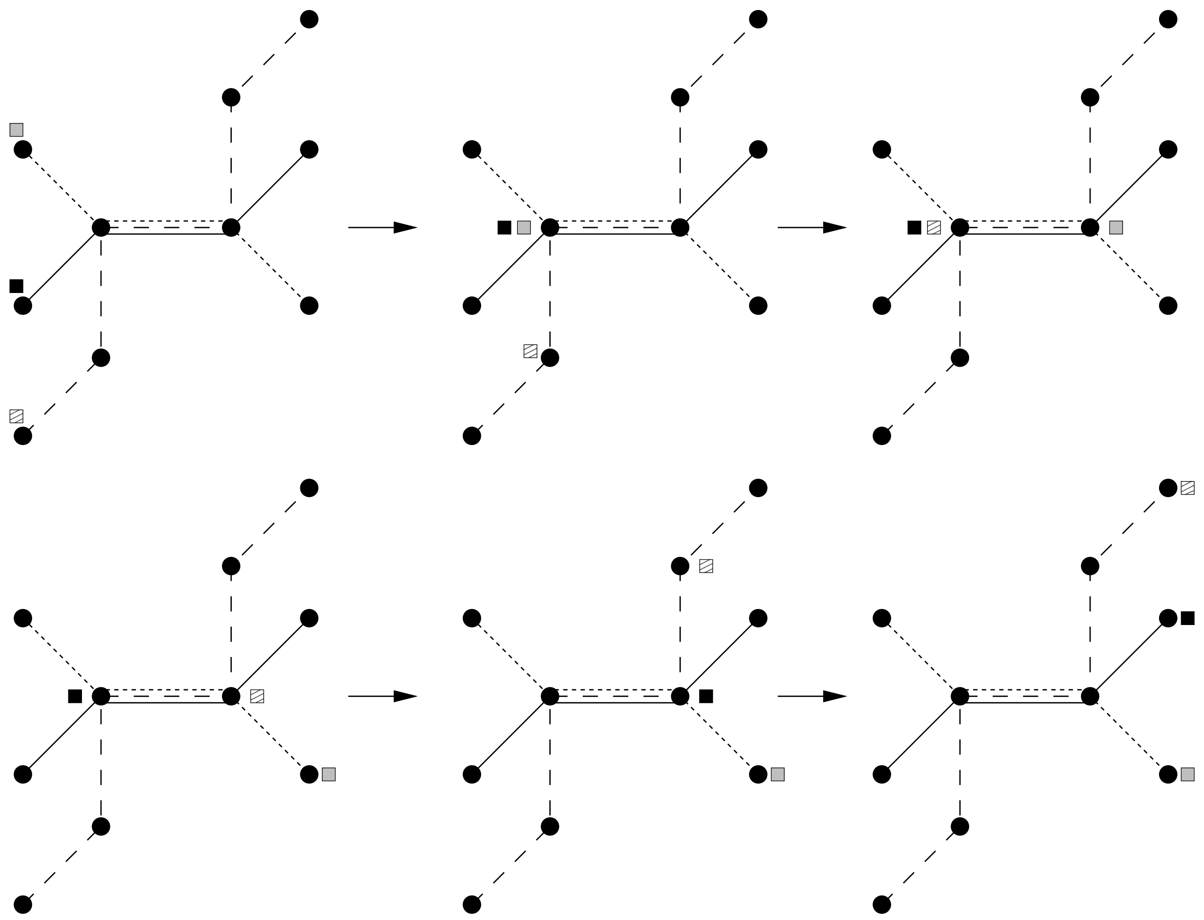

Figure 1: Example Solution

We are interested in minimizing the end-to-end delay of the packets by determining which of the packets competing for an edge goes first.

Model 1 Every path sends a single packet at time 0. Goal: minimize the maximum end-to-end delay.

Model 2 Regular stream of traffic on each path. Suitable for modeling constant bit rate traffic. Each path i declares a fixed rate $r_i < 1$ at which it will send packets. Equivalently, a new packet is sent every $\frac{1}{r_i}$ time steps.

Model 3 Every path i declares a rate r_i and a buffer (bucket) size b_i. Each path i guarantees that for all time windows of t time steps, at most $r_i t + b_i$ packets are sent.

Model 4 For all time windows t, the expected number of packets sent is at most $r_i t + b_i$. Usually, some guarantee is required on the nature of the distribution and/or the variance of the distribution of the packets on each edge.

Traffic that respects the restrictions of **Model 3** is common in real networks, since the hardware often enforces this discipline anyway, dropping the packets of traffic which does not respect it. Queues in a network are usually served by constant rate servers and have fixed finite buffer size. If the server handles packets at rate r_i and the buffer size is b_i, then if more than $r_i t + b_i$ packets arrive in t time units, then some packets arrived when the buffer was full, and were discarded. This method of traffic flow control is called the leaky bucket. The bucket leaks water at a constant rate of r_i and has a capacity of b_i litres. If more than $r_i t + b_i$ litres of water arrive in t time units then water overflows from the bucket and is lost.

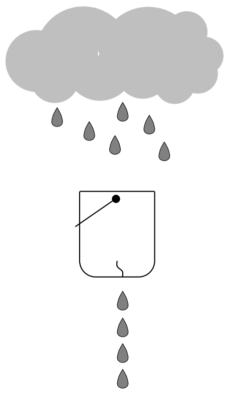

Figure 2: The leaky bucket

We will consider the analysis of **Model 1** where we seek to minimize the maximum end-to-end delay. Let C be the maximum number of paths sharing an edge and D be the maximum number of edges in a path. Similarly, let c_i be the maximum number of paths sharing an edge of path i and d_i be the number of edges on path i.

Easy lower bounds (independent of edge protocol):

$$\begin{aligned} \text{Maximum end-to-end delay} &\geq C \\ &\geq D \end{aligned}$$

For each model, we can consider two versions of the analysis:

Version A Determine a guarantee on the maximum end-to-end delay for all paths as a function of C and D.

Version B Determine a guarantee on the end-to-end delay for path i as a function of c_i and d_i.

Theorem 1 *There exists an edge protocol such that the maximum end-to-end delay is bounded by DC and the end-to-end delay of path i is bounded by $d_i c_i$.*

Proof.

Let λ_e be the number of paths that share edge e. Edge e arbitrarily selects a periodic schedule with period λ_e in which it accepts a packet from each path that uses it in turn. If there is no packet on a particular path when its slot is available, leave the slot idle.

Then as each packet arrives at an edge, it must wait at most λ_e time units before proceeding. For path i, $\lambda_e \leq c_i$ and there are d_i edges in path i. Therefore the end-to-end delay of path i is bounded by $d_i c_i$. This bound implies that the maximum end-to-end delay is bounded by DC. ■

Theorem 2 *There exists a packet schedule with end-to-end guarantee of $\mathcal{O}(c_i + d_i)$.*

Proof.

The proof of this result uses the Lovász Local Lemma to show existence. While the proof can be made constructive using the algorithmic version of the local lemma, the resulting algorithm is *very* complicated. ■

In the following theorems, C and D are defined as above and n is the number of nodes in the graph.

Theorem 3

a) There is a packet schedule for **Model 1** *with end-to-end delay guarantee of* $\mathcal{O}\left(c_i + d_i \log nCD\right)$.

b) There is a packet schedule for **Model 2** *with end-to-end delay guarantee of* $\mathcal{O}\left(\frac{1}{r_i} + d_i \log n\right)$.

c) There is a packet schedule for **Model 3** *with end-to-end delay guarantee of* $\mathcal{O}\left(\frac{b_i}{r_i} + d_i \log n\right)$.

We will provide a "cheating" proof of the result for **Model 1**.

Consider the schedule constructed by:

1. Each path i waits w_i time steps at the origin, where w_i is selected independently and uniformly at random from $\{0, \ldots, 2c_i\}$.
2. Construct the "schedule" (may ignore edge conflicts) in which the packet waits w_i time steps and then proceeds along its path one edge per time step.

Lemma 1 *Let $\gamma = 1/\log \frac{4}{e}$. Then, for each edge e and each time window $[\tau, \tau + 6\gamma \log nCD]$,*

$$Pr\left(\textit{The number of packets on edge } e \textit{ during } [\tau, \tau + 6\gamma \log nCD] \; > 6\gamma \log nCD\right) < \frac{1}{n^3 CD}$$

Proof.

Consider the packet associated with the path i and the probability that it crosses edge e during the time period $[\tau, \tau + 6\gamma \log nCD]$. The number of choices of w_i that allow the packet to cross edge e in this time period is $\leq 6\gamma \log nCD$. We also know that for all paths i through edge e, $c_i \geq \lambda_e$. Therefore

$$Pr\left(\text{Packet } i \text{ crosses } e \text{ during the time period}\right) \leq \frac{6\gamma \log nCD}{2\lambda_e}$$

and so

$$E\left(\text{Number of packets crossing } e \text{ during the time period}\right) \leq 3\gamma \log nCD.$$

We will use the following lemma to complete the proof:

Lemma 2 (Chernoff bound) *Let $X_1, \ldots, X_n$ be independent and identically distributed 0-1 random variables and suppose that $E(\sum_i X_i) \leq M$. Then*

$$Pr\left(\sum_i X_i > (1+\delta)M\right) < \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^M$$

Set $\delta = 1$ and $M = 3\gamma \log nCD$ in the Chernoff Bound. Then

$$Pr\left(\text{Number of packets crossing } e \text{ during the time period} > 6\gamma \log nCD\right) < \left(\frac{e}{4}\right)^{3\gamma \log nCD} \leq \frac{1}{n^3 CD}$$

■

Corollary 1

$$Pr\left(\textit{There exists an edge and a time window with too many packets}\right) < \frac{1}{n}.$$

Now we must resolve the fact that we ignore concurrent use of the edges. For each packet on path i, we impose the condition that it must cross the jth edge of path i by the deadline $w_i + j6\gamma \log nCD$. Then, at each edge, schedule the waiting packet with the earliest deadline for that edge. If there are no edges and no time windows with too many packets, then it is not hard to show that all packets make their deadlines at each edge. Corollary 1 shows that this event has high probability. The end-to-end delay of packet i in this schedule is at most $O(c_i + d_i \log(nCD))$ as the deadline on the first edge is at most $O(c_i + \log(nCD))$, and the deadlines on subsequent edges get incremented by $\log(nCD))$ per edge. Hence with high probability all packets arrive at the end destination in at most $O(c_i + d_i \log(nCD))$ steps. The tradeoff is that each packet must carry its own deadlines as additional data in the packet. This overhead is undesirable.