CS 536 (Park) **Assignment XIII** **Due: Dec. 12 (Fri.), 1997**

PROBLEM 1

(a) Read Sections 4.2, 4.3, 4.4, 5.3, 9.1, 9.2, 9.3 from P & D.

(b) Read "ATM concepts, architectures, and protocols," R. Vetter. *Communications of the ACM*, 38(2):30–38, 1995. Give a one-page pros/cons analysis of ATM networking technology based upon the material you have learned so far. Include your own views.

PROBLEM 2

(a) Implement the distance vector (Bellman-Ford) routing algorithm for the network shown in Figure 1. Assume

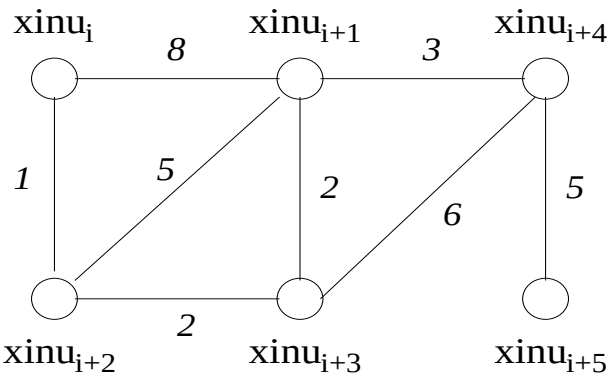

Figure 1: Logical network topology with link latencies.

symmetric links where the labels on the edges represent latencies. At every instance where the receipt of a routing table control packet (*destination*, *delay*) from a neighboring node induces an update in the current routing table, log the current routing table into a suitable file for later examination (make each xinu node log into a unique file by using `gethostname` and crafting an appropriate file name; this is needed due to your file systems being NFS mounted).

Check that your implementation works correctly by comparing the final table entries against your hand-computed solution. You may assume that the tables have stabilized if no more routing table control packets are received for 5 seconds after the last update. If so, let xinu$_i$ send a packet to xinu$_{i+5}$; at each intermediate node (including the sender and receiver), record the host ID onto the payload of the packet. At the receiver, output the payload of the probe packet and verify that the shortest path has been taken. To clean up, each node should terminate itself after about 30 seconds.

How long do the routing tables take to stabilize (i.e., measured with respect to the start time)? For xinu$_i$, plot the evolution of the routing table entry values as a function of time (5 graphs, one for each destination). What is the convergence rate?

(b) As a continuation of part (a), after stabilization, let xinu$_i$ send probe packets to xinu$_{i+5}$ (where again the entire route taken is recorded and output at the destination) 200 ms seconds apart for 10 seconds. 500 ms after initial stabilization, perturb the latency on the link from xinu$_i$ to xinu$_{i+1}$ by changing it from 8 to 3. From the output of the probe packets at xinu$_{i+5}$, observe if the path taken by the packets has changed over the course of time. Verify that this is the new shortest path.

(c) As a further continuation of part (a), what happens if from the very start, each link latency is perturbed around the values given in Figure 1? That is, for each link (xinu$_i$, xinu$_j$) and its latency ℓ_{ij}, at X second intervals, the latency is set to

$$\ell_{ij} + d$$

where $d \in \{0, 1, -1\}$ is a random variable taking on each value of its domain with probability 1/3. Here, ℓ_{ij} denotes the *original* link latency given and not the cumulative one. Set X to be the convergence time found in part (a). Additionally, send probe packets from xinu$_i$ to to xinu$_{i+5}$ at 1 second intervals recording the route taken in the

payload. Track the entries of xinu$_i$'s routing table as in (a). Do the values ever converge? Run the experiment for 60 seconds.

What happens to the route of the probe packets sent? Do they remain invariant in the presence of the perturbation?

PROBLEM 3

Extend Dijkstra's algorithm for shortest path calculation such that the three quality of service (QoS) requirements—minimum bandwidth, maximum end-to-end delay, maximum packet loss rate—are satisfied (each link has three values associated with it, latency, bandwidth, and packet loss rate). That is, an application indicates its QoS requirement by presenting the three bounds. Give a concise argument of why your algorithm works correctly.

How would such an algorithm be used in practice? Is additional router support beyond what is provided currently needed? If so, what are they? Describe the events that your system would take to achieve per-application QoS-sensitive routing using the modified SPF algorithm as a subroutine.