Computer Networks (CS4310) - Spring 2014

Project One

A Discrete-time Event Simulator for Distance Vector Routing

Due Nov 4, 2015 @ 11:59 PM

This project would take a good amount of work and time, so please start early. You would not be able to finish if you start few days before the due date. Late submissions would incur a penalty of 20% per day for up to 2 days, and then they will not be accepted. Leave the last few days for documentation, further testing and formatting the results. Please read the description below carefully, understand the requirements and the expected outcomes. Please remember that it is an act of plagiarism to submit someone else's work as part of your own. Please come see me/email me, if you have questions. This project is to be done individually.

An Overview:

In this project, you are asked to build a discrete-time event simulator to simulate Distance Vector routing for a given network. The main goal of this project is to study the impact of different factors on the convergence times to optimal routes. You will be provided with multiple files that represent different network topologies. Your simulator would need to deal with data packets, DV packets, and link failure events. For example, on DV packet arrivals, each node would update its internal routing table. On data packet arrivals, each node would forward the packet towards the destination, after consulting its routing table.

The Network Topology Files:

Each network topology file consists of a number of rows, each row represents an edge in the network. There are four entries per row. The first entry is the source node ID (address), the second is the destination node ID, the third entry is the cost of the link between the source and destination. The last column represents the delay it takes to send and receive packets on that link. Notice that the third entry is the one that will be used to find the shortest paths in the network. For example, a row with these values: 2 12 23 0.3, would mean that there is a link between node 2 and node 12. The link has a cost 23 and it would take 0.3 seconds for a packet (DV or data) to be sent from one end to the other. Here are three topologies to use:

Toplogy1.txt (5 nodes, 7 edges)

Toplogy2.txt (10 nodes, 20 edges)

Toplogy3.txt (30 nodes, 60 edges)

These topologies were generated using BRITE. The grader may test/grade your code on different topologies than the ones provided above, so make sure *nothing* is hard-coded (even the number of nodes!). Your simulator should take as a line argument, which topology file to use along with the duration to run your simulation.

Distance Vector:

Recall that in a distance vector-based routing algorithm, each node would tell its neighbors,

what it knows about the whole network. As nodes start receiving DV packets, they update their routing tables if better routes are discovered. Each node would maintain a routing table that consists of <destination, cost, next hop>. The DV packet each node sends is simply the destination and cost pairs (no need to send the next hop). Your distance vector should implement split horizon (A node should not send an entry in its DV to a particular neighbor, if it learned this route from that neighbor). Once a node receives a data packet, it consults its routing table and forwards the packet to the next hop, which would do the same until the packet reaches the destination. Nodes every 1 second will send their DV to their neighbors (periodic update). When a given node changes its routing table, it sends its new DV immediately to its neighbors (triggered update).

Where to Start:

- First, you need to fully understand how distance vector works.
- Build a simple simulator, define the events that are going to be used and make sure events are inserted in the right order and that the corresponding routine would be called correctly. Examples of routines are: processDVPacket, forwardDataPacket, sendDVPacket, etc...
- Start with a simple topology (topology1.txt) and correctly fill in the data structures you are going to use.
- Write trace/debug functions so you can make sure the simulation is behaving correctly, following the time constraints given in the topology. For example, you can print the routing table for each node, print any route changes and print the next hop on data packet forwarding, making sure the next hop receives the data packet and forwards it correctly. Always print out the simulation time corresponding to each event.

What to Turn in:

- A short design document, stating the overall design decisions made, data structures used and events that are handled by the simulator.
- The source code.
- The "results" document that includes the full routing table for each node (for the above 3 topology files) at the simulation end time (should be set long enough to ensure convergence).
- Please provide a single command line to compile your program and another single command line to test your code and make sure to provide a working example of the command line. You can assume that the topology file will be in the same directory as your simulator.
- The grader must be able to compile and run the simulator on the CS Linux Servers.
- You need to submit a single zipped file via email to me with the subject "cs4310-project1"

- In the "Results document", answer the following questions:

1. How long did it take each network to converge?
2. What is the ID of the last node to converge in each network?
3. How many routing-control messages were sent before each network converged?

- Insert the following events in your simulator and trace the events (times, and next hops) until the packets make it to their destinations.

1. For the first topology: at time 10, node 0 receives a data packet destined to node 3.
2. For the second topology: at time 20, node 0 receives a data packet destined to node 7.
3. For the third topology: at time 30, node 0 receives a data packet destined to node 23.

- Insert the following events in your simulator and print down how long it takes for each network to converge back.

1. For the first topology: at time 10, node 0 and node 2 discover that the link between them went down.
2. For the second topology: at time 20, node 5 and node 7 discover that the link between them went down.
3. For the third topology: at time 30, node 9 and node 17 discover that the link between them went down.