CSEE 4119: Computer Networks

Spring 2016

Assignment 6

Academic Honesty Policy

You are permitted and encouraged to help each other through Piazza's web board. This only means that you can discuss and understand concepts learnt in class. However, you may NOT share source code or hard copies of source code. Refrain from sharing any material that could cause your source code to APPEAR TO BE similar to another student's source code enrolled in this or previous years. Refrain from getting any code off the Internet. Source code should be yours and yours only. Do not cheat.

Mini-programming Assignment

You will build a very basic implementation of a distance-vector (RIP-like) protocol over UDP. Each node runs an instance of your code, invoked as

```
$ router listen_port interface1 interface2 […]
```

```
$ router 2000 128.59.16.1:2001:3 128:59.16.2:2003:4 […]
```

where the link tuples are peer IP address, port and link metric ("hops"), with interfaces numbered starting at 1.

Each router listens at the listen_port for incoming routing messages, and sends update messages to the neighbors designated on the command line, using a separate UDP socket. This allows you to run all nodes on the same host, if you prefer. Thus, the typical loop would be "listen for message – compute routing table -- send updates to neighbors."

Links are created and changed by re-starting the application.

You do **not** have to worry about the count-to-infinity problem nor implement poisoned reverse. Links and routes can change weights, but they do not disappear, i.e., you do not have to time out links.

Each router sends periodic text messages to its peers, at start-up and once every 5 seconds, using any packet format you like. Links do not lose packets. When a router receives a new message from one of its neighbors, it re-computes the routing table and shows a modified routing table on its terminal output, with a timestamp, similar to

```
Node 128:59.16.1:2000 @ 21:35:17
```

```
host        port  distance    interface
128.59.16.1 2000  3           1
128.59.16.5 2001  5           2
```

You should support host names in addition to IP addresses, but can restrict yourself to IPv4. The command line should accept any number of link tuples.

To test your algorithm, you can use the graph in problem 26 (pg. 422). You can assume that the links are bidirectional and symmetric, or you can use the directed graph.

You should submit a description of your packet format, a sample topology graph you used for testing and the output of your nodes. You can use the Unix **tee** command to capture output while still seeing the output, as in

```
$ router … | tee logfile.txt
```

Written Problems

1. *Routing*: Draw an example network that shows that poisoned reverse does not prevent loops for loops involving three a more nodes.
2. *CSMA/CD*: Even assuming no electrical propagation loss, a CSMA/CD network cannot be arbitrarily large. Why not? Given a 10 Mb/s channel, what would you have to do in your link layer protocol to ensure that the network, made from copper cable, can span 100 meters?
3. *ARP*: Use the **arp** command to find out what the MAC address of your local access point or router is. Describe how you determined this.