Design Notes
Assignment 4

Majid Ghaderi

Disclaimer

These notes are based on my own implementation. I do not claim that my implementation is the simplest or the best. Feel free to use or disregard any of these suggestions as you wish.

Tipis

- Start with a simple topology, for example, just two nodes.
- Print enough information to trace the operation of your router.
- Change the update interval to slow down the router if you are tracing the output.

Program Structure

The high-level structure of my Router.start() method looks like the following:

```
1: open a TCP connection to the server
2: send/receive/process HELLO
3: start Timer
4: while not QUIT packet do
5:     dvr = receive a DvrPacket
6:     processDvr(dvr)
7: end while
8: cancel timer, close socket, clean up
9: return routing table
```

The method processDvr() is at the heart of the distance vector routing algorithm. It essentially implements the Bellman-Ford algorithm. The operation of this method depends on the sender of the DvrPacket:

```
void processDvr(DvrPacket dvr) {
  // if dvr.sourceid == DvrPacket.SERVER
  //    this is a link cost change message
  //    update link cost vector
  //    update min cost vector
  // else
  //    this is a regulsr routing update from a neighbor
  //    update min cost vector
}
```

Setting up a Recurring Timer

The method scheduleAtFixedRate() in Class Timer can be used to schedule a recurring timer at fixed intervals. In the associated timer task, the min cost vector at the router should be transmitted to its directly connected neighbors.

Routing Data Structures

In my Router implementation, I have defined three arrays to keep track of the link cost vector, next hop vector, and distance vectors of the router and its neighbors:

```
  int[] linkcost;   // the cost of link to other routers
  int[] nexthop;    // the next hop to reach each router
  int[][] mincost;  // the array of mincost vectors
```