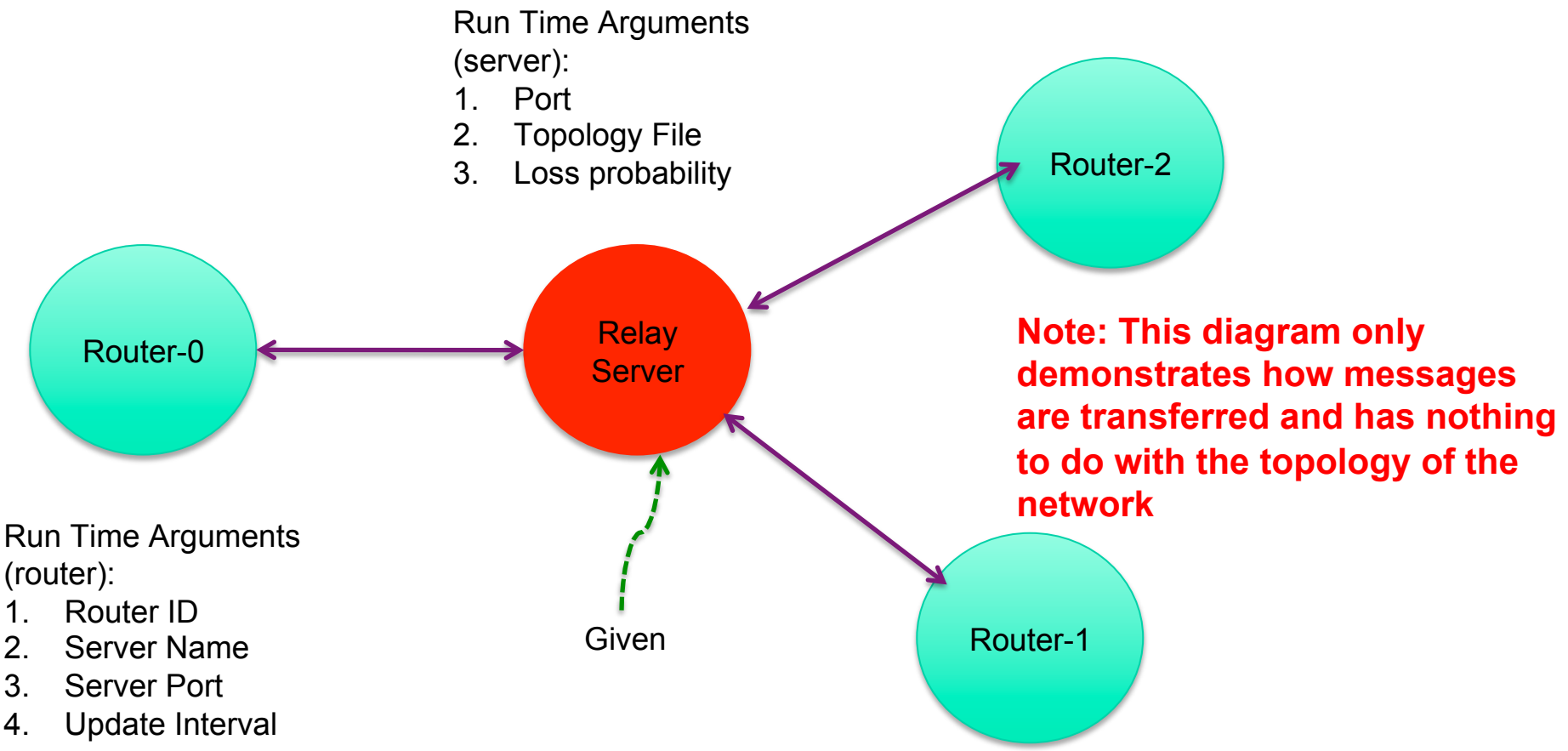
CPSC 441
Assignment-4 Discussion

Department of Computer Science
University of Calgary

Overview



You need to implement the routers

Types of Messages

- "Hello" message from router to server
  - On router startup
- "Hello" message from server to router
  - As response to "Hello" message from router
  - Contains link cost vector
  - Number of routers in the network = Length of link cost vector
- "Route" message from other routers
  - Regular routing information
- "Route" message from server
  - New topology information – for eg. new weight to link(s), link(s) added, link(s) removed or a combination of it
- "Quit" message from server to router
  - On termination

DvrPacket

- Type of packets exchanged between routers and servers
- Serializable (what it means in terms of this assignment)
  - Use **object streams** for input and output
  - Example on how to create I/O streams (socket is the **Socket** object)

```
ObjectInputStream in = new
  ObjectInputStream(socket.getInputStream());

ObjectOutputStream out = new
  ObjectOutputStream(socket.getOutputStream());
```

For more details on DvrPacket class, refer javadoc

Steps at client

Disclaimer: This may not be the best or simplest implementation. Feel free to try out other options

- Open a TCP connection to server
- Send/Receive/Process "Hello" Message
- Start Timer – sends distance vector to **neighbors** on timeout.
- While incoming packet is not "Quit"
  - Receive DvrPacket ← **Check if "Quit"**
  - Process DvrPacket ← **Based on Sender ID**
- Terminate
  - Cancel timer
  - Close socket
  - Return Routing Table

Data Structure @ Router

- **linkcost** array
  - For storing cost of link to other routers
  - Updated ONLY when “Hello” and “Route” message received from **the server**
  - **Remember to update this vector when “Route” message is received from the server**
- **mincost** array
  - To store minimum cost to each router
- **nexthop** array
  - Next hop to reach each router

Sending DvrPacket to Socket

```
out.writeObject(packet);
```

packet is of type **DvrPacket**

Note the method “writeObject”

Reading DvrPacket from Socket

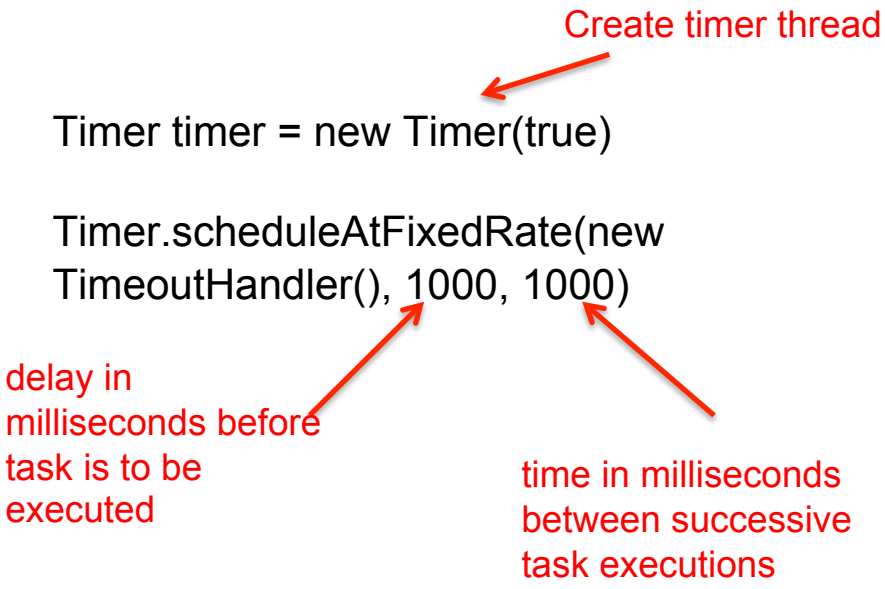
packet = (DvrPacket) in.readObject()

packet is of type **DvrPacket**

Note the method “readObject”

For sending periodic "route" message

- Use **Timer** class as in assignment-3
  - Periodic
- Use **scheduleAtFixedRate** method to schedule timer at regular intervals




```
class TimeoutHandler extends TimerTask
{

   // define constructor

public void run()
{
      // send min cost vector to neighboring
            routers
}
```

Compute Min. Cost

- Use **"modified" Bellman-Ford algorithm**
- Distance vector (**mincost** array) of a router (id: x) is updated as follows upon receiving a "Route" message from neighboring router

$$Dx(y) = \min_{v}\{C(x,v) + Dv(y)\}$$

x = router id

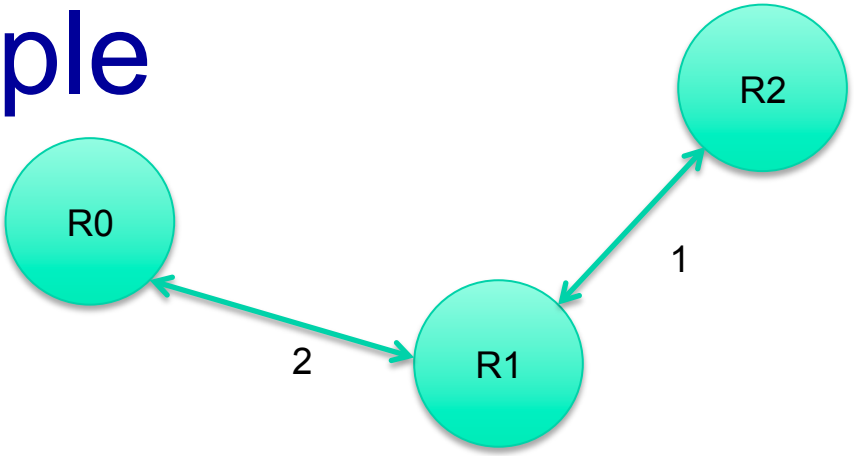
y = Set of routers in the topology

v = Set of neighbors of router x

Don't forget to update other data structures used based on the type of message received

Note: Recommend using 'synchronized' receive (by main thread) and send (by time thread) methods since both uses 'mincost' data structure so as to avoid any concurrency problem

Example



Data Structure with Respect to R0:

Initial:

linkcost = [0 2 999]
mincost = [0 2 999]
nexthop = [0 1 -1]

Initialized upon receiving "Hello" or "Route" message from the server

Final (upon termination):

linkcost = [0 2 999]
mincost = [0 2 3]
nexthop = [0 1 1]

To denote that information is not known currently