Report on Routing Protocol Implementation

Computer Networks Sessional

CSE 324

Submitted by-

Chowdhury Md Rakin Haider

#1005009

Sabbir Ahmad

#1005010

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

1. Introduction

This report is on a simple network simulator where network devices are simulated by codes written in java programming language. The basic idea of a network simulator is to create a topology and then network devices are connected to the topology.

2. How the simulator works

2.1 Network device implementation

There are three types of devices in the project: Host, Switch, Router which are simulated by SimHost.java SimSwitch.java and SimRouter.java respectively. Each device's physical instance is simulated by SimPhy.java and DataLinkLayer.java is used as well for the devices. Each device has a device id which is supplied as command line argument which must be present in Config.txt file with necessary information.

2.2 How the topology is created

There are some configuration files in the project. The Config.txt file contains the topology information and configuration values of the simulated devices. ArpTable.txt file contains information that resolves IP address to MAC address. A new topology can be created by writing new configuration files.

2.3 How the topology is maintained

The ConnectionDaemon.java file runs the topology by reading information from Config.txt. It runs a server socket on port 9009 of local host and determines connection between devices from Config.txt file. The bottom of the protocol stack is simulated by SimPhy.java. SimPhy opens a client socket with ConnectionDaemon and sends the device id. After receiving the device id, ConnectionDaemon saves each device pair by linking input stream from one side to the output stream to other side and vice versa. The information is saved in a map in ConnectionDaemon.

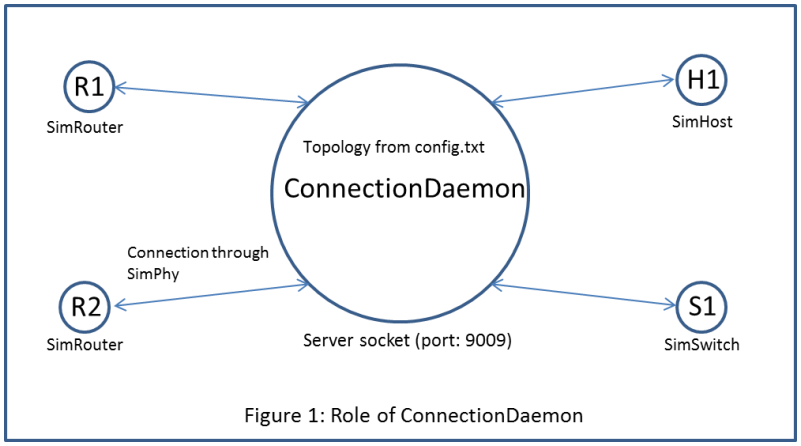

Figure 1: Role of ConnectionDaemon

2.4 Device configuration

Device id is supplied as command line argument. When a device starts it reads its configuration from Config.txt file and load connection parameters for the device. IP to MAC mapping is read form ArpTable.txt file.

3. Protocol Architecture

Each device maintains a layered protocol architecture. SimPhy is the lowest level represents physical layer of a network device. DataLinkLayer is on top of SimPhy. All the devices use these two levels in this project. The 3rd layers of the devices are different from one another and ther are simulated by SimHost, SimSwitch and SimRouter for host, switch and router respectively. SimInterface is used for representing interfaces of of SimRouter.

A device sends or receives data through physical layer. While sending data the data is send to physical layer form data link layer which had been sent form network layer. For receiving opposite work is done. So data needs to be sent and received between any two consecutive layers. For this reason between any two consecutive layers, there are two shared buffres. Send buffer and Receive buffer. There are also two threads, read thread and write thread between any two layers. Read thread continuously reads from receive buffer where lower layer has shared some data. The the data is processed and it is send to send buffer for the upper layer.

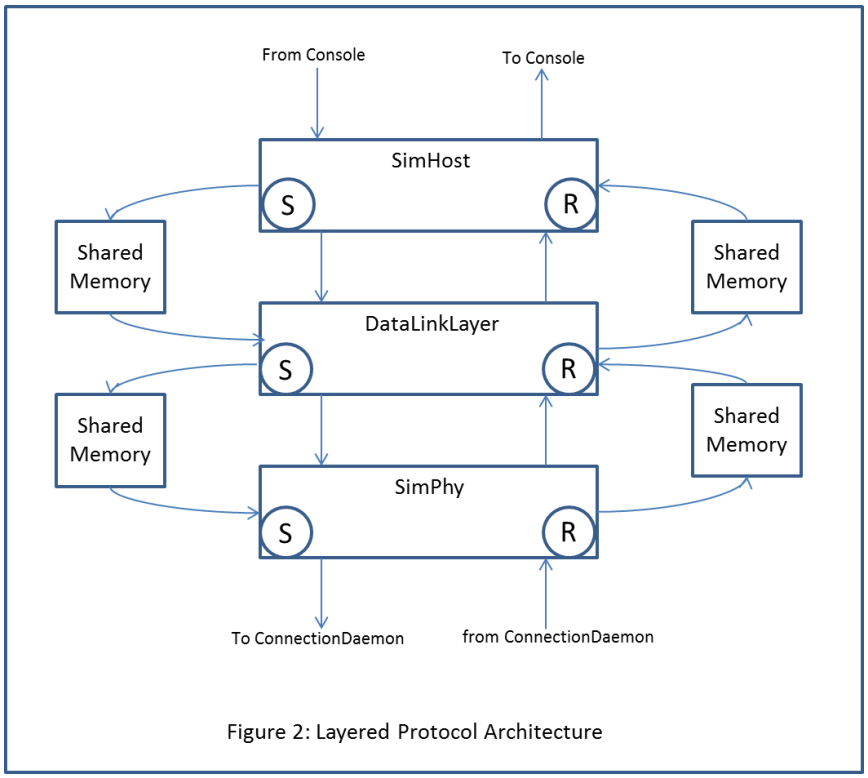

Figure 2: Layered Protocol Architecture

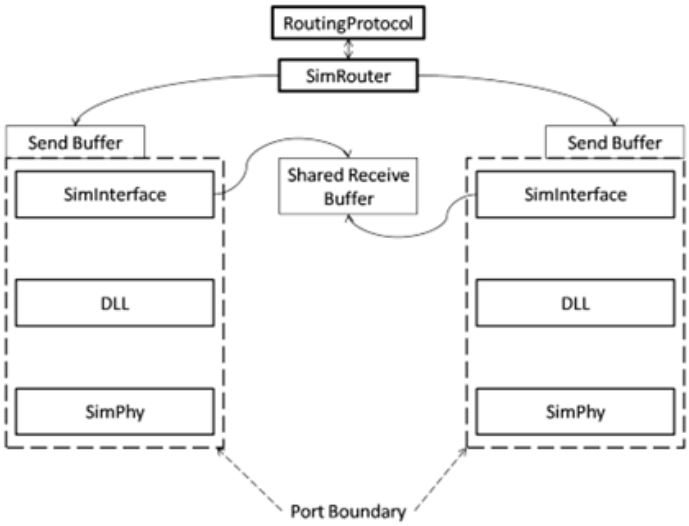

Figure 3: Schematic representation of Simulated Router

4. Implemented Routing Protocol

The protocol that is implemented here is Routing Information Protocol (RIP) which is one of the oldest distance-vector routing (DVR) protocols. It employs the hop count as a routing metric. RIP prevents routing loops by implementing a limit on the number of hops allowed in a path from the source to a destination. The maximum number of hops allowed for RIP is 15. This hop limit, however, also limits the size of networks that RIP can support. A hop count of 16 is considered an infinite distance, in other words the route is considered unreachable.

4.1 Rules

RIP defines two types of messages: Request Message and Response Message. In RIP router sends its routing table information to through its interfaces to all its neighbors. Upon receiving the information a router processes the information to update its routing table as per following rules:

1. If there are no route entry matching the one received then the route entry is added to the routing table automatically, along with the information about the router from which it received the routing table
2. If there are matching entries but the hop count metric is lower than the one already in its routing table, then the routing table is updated with the new route.

3. If there are matching entry but the hop count metric is higher than the one already in its routing table, then the routing entry is updated with hop count of 16 (infinite hop). The packets are still forwarded to the old route. A Holddown timer is started and all the updates for that from other routers are ignored. If after the Holddown timer expires and still the router is advertising with the same higher hop count then the value is updated into its routing table. Only after the timer expires, the updates from other routers are accepted for that route.

Timers are used for different events in RIP. The timers that are implemented are given in the following:

4.2 Timers

- **Update Timer**

  The update timer controls the interval between two gratuitous Response Message. By default the value is 30 seconds. The response message is broadcast to all its RIP enabled interface.

- **Invalid Timer**

  The invalid timer specifies how long a routing entry can be in the routing table without being updated. This is also called as expiration Timer. By default, the value is 180 seconds. After the timer expires the hop count of the routing entry will be set to 16, marking the destination as unreachable.

5. Code Modification

5.1 Additional Variables:

1. buffer: A buffer with the name "Upadate Packet Buffer" is kept to store the received update packets.

2. updateTimer: A router needs to transmit its routingTable to its neighbours after regular interval so that all the routers can update their own routing table. To do this updateTimer is used. It expires after 30 seconds and calls handelTimerEvent funciton which send routing packet to every neighbour routers. The updateTimer variable is created in the constructor of RoutingProtocol. Then inside the run function a TimerTask is scheduled with the required time interval.

5.2 Additional Class:

RoutingTableEntry:

We implemented the routing table by using an CopyOnWriteArrayList of object of type RoutingTableEntry. RoutingTableEntry class has fields named type, ip, mask,nextHop, interfaceId, hopCount, invalidTimer. The type field denotes whether this router is directly connected to the network related to the ip address in ip field or not. The other fields stores information about the next router a packet should be routed to reach corresponding network, number of hops to reach destination network, and finally an invalid timer. The invalid timer is the one which expires when no update is received for this entry between a certain times.

5.3 Changes to given stub code:

Function **notifyRouteUpdate():**

This function is only called when the simrouter encounters a packet whose destination ip is RP_MULTICAST_ADDRESS. That means when the simrouter recieves an update packte it passes the update packet to this function. This function is solely responsible for storing the update packet in buffer. This buffer is read inside the infinite while loop inside run function. While storing the packet to the buffer proper synchronization is ensured as the buffer can be accessed from other thread. If the buffer is full the function will wait untill there is empty space to store the packet.

Function **notifyPortStatusChange():**

This is the function that is called when some port of the simrouter has gone down or has become up. If the function is called with the status true it means the port denoted by the interfaceId has become up. Therefore a RoutingTableEntry object with information about this port is created and then added to the routingTable. On the other hand when status has false as its value it means the port has gone down and the RoutingTableEntry corresponding to this port has been removed from the routing table.

Function **getNextHop():**

The function is passed a destination ip address. So the routingTable is searched for an entry with a matching ip address. As our routingTable is sorted in descending order of mask we will be taking the longest possible match as our next hop. If a next hop is found its returned otherwise null is returned.

Function **run():**

The first change that is made to the given code is a Buffer for ByteArray is kept inside the simrouter. This buffer serves the purpose of passing information about port,ip and mask of each interface to RoutingProtocol class. When informations about a new interface is found inside loadParameters function it is stored in the buffer. Then this buffer is read from within the run function of

RoutingProtocol class. The first while loop inside run, reads information about each interface and creates a new RoutingTableEntry for that interface. These RoutingTableEntry are inserted to the routingTable arraylist and thus the routing table for directly connected network is formed.

After creating routing table for directly connected network all the table entries are shown as described by the to-do list in the function. The next job is to send initial routing update packet to neighbours. This is done by the sendUpdatePacketToNeighbours(). After sending initial update packet to neighbours the a timer named updateTimer is started. This timer expires after every 30 second and calls a function handleTimerEvent(1, null). The description of handleTimerEvent is given in the later part of this report.

After initializing the update timer the run function enters an infinite loop which tries to read from the Buffer with the name "Upadate Packet Buffer" in which all the update buffer are kept. If there is a new update packet then the packet is proccessed. The nested for loop checks every entry in the routing table to find out a match for each ip in update packet. If a match is found then it is checked whether the hopCount stored in the routers own routing table is less than the hopCount found from update packet. If its less than the current hopCount then the table entry is updated and invalidTimer is restarted. Otherwise only the invalidTImer is restarted nothing else done. If no match is founc for an ip in the update packet then this must be added to the routingTable, which is done inside the if block with the condition (!found).

Function **handleTimerEvent():**

This function handles all the timer expiration event. When a timer expires this function is called with appropriate argument.

If the type parameter has 1 in it means update timer has expired and we need to send the routing table to neighbours. To do this first we convert the routingTable to a ByteArray. For each entry in routingTable we only use the ip address and hopCount for update packet. Then for each interface in the simRouter we create a packet where the srcip is the interfaces ip, destination ip is RP_MULTICAST_ADDRESS and payload is the ByteArray respresentation of routingTable. Then this packet is sent to the interface using sendPacketToInterface().After this the updateTimer is restarted.

On the other hand if the type parameter has 2 as its value, it means that an invalidTimer has expired and that timer is related to the RoutingTableEntry passed to this function. Therefore this RoutingTableEntry is deleted from routingTable.

5.4 Additional Functions:

Function **add():**

This function is used to insert RoutingTableEntry to routingTable in a sorted order.

Function **showAllEntry():**

Showing all the entries of the routingTable is done by this function.

Function **forTableView():**

This function is required to show the routing table in GUI.

Function **sendUpdatePacketToNeighbours():**

When we need to send the routing table to neighbours this function is called. To do this first the routingTable is converted to a ByteArray. For each entry in routingTable we only use the ip address and hopCount for update packet. Then for each interface in the simRouter we create a packet where the srcip is the interfaces ip, destination ip is RP_MULTICAST_ADDRESS and payload is the ByteArray respresentation of routingTable. Then this packet is sent to the interface using sendPacketToInterface().

Function **intToBytes()** and **bytesToInt():**

These two functions are needed when we convert routingTable to ByteArray. We need to convert hopCount to byte and vice versa.