Assignment Report : Implementation of a Routing protocl

Objective:

1. Implementation of a routing protocol such as Distance vector routing.
2. Implementation of a router's forwarding Function.
3. Fill up codes in the given stub file: RoutingProtocol.java
4. Implementation of frame checksum calculation and checksum verification.
5. To verify whether checksum is working, run ConnectionDaemon with non zero error probability.
6. Implement bit-stuffing in physical layer
7. Implementation of suitable Graphical User Interface (GUI)
8. Report on the assignment

Description:

4. Implementation of frame checksum calculation and checksum verification

Data link layer has to check for errors before it pass the frame as packet to the network layer. To do this one of its strategy is to check the checksum.

While sending a frame to physical layer it adds some extra information and checksum is one of it. Checksum can be calculated in different ways. The most popular way is to calculate it by XORing all other data.

In this assignment data was considered as unit of byte. So checksum was also a byte and it was calculated this way:

checksum = source Mac address XOR destination Mac address XOR (XOR of all data bytes)

Before sending a frame to PL, DLL calculates the checksum this way and adds the checksum in checksum field.

DLL does it by calling calculateChecksum() function.

On receiving a frame DLL call the function calculateChecksum() and compare the checksum with the received ones. If they are different there is error and the frame is not passed to NL.

It was done in hasChecksumError() function.

5. Verification of checksum error:

To do this connection daemon was run with non zero parameters. In addError() function it changes some bits so that we can see checksum error. But there was bug in the given code stub. On the execution of this loop array out of bound exception occured:

```
for (int i=0;i<k;i++)

        b[i]=(byte)(~b[i]);
```

So we replaced it with :

```
for (int i = 0; i < k; i++)
                    b[i] = (byte) (~b[(int) (Math.random() * (b.length - 1))]);
```

On doing this we were able to verify checksum error.

6. Implement bit-stuffing in physical layer

To detect the frame on receiving same special preamble and post amble was added. In this assignment it was ~ (ascii 126). If this special character occurs in the middle of data the detection of frame fails.

So bit stuffing was needed to avoid this problem. The character ~ has ascii value of 01111110. After every five consecutive 1s we added an extra zero. By doing this we managed to avoid the occurrence of 01111110.

Java does not give access of bit level. So we used ArrayList of Boolean and parsed the byte array to this array list. After stuffing bits in the arrayl ist it was converted again to byte array. It was done n the method bitStuff().

On receiving bytes in PL, bit destuffing was needed to parse the actual data. The received byte array was parsed in arraylist of Boolean and after every 5 consecutive 1s the next 0 was removed. Then the array list was converted again in byte array and passes to DLL. It was done in the method bitDestuff().

We assume the state false for 0 bit and true for 1 bit in the array list.

7. Implementation of suitable Graphical User Interface (GUI)

A suitable GUI was implemented using java swing. The connection daemon output was shown in it successfully. But the output from hosts or routers could not be managed to show on this GUI.

8.**Implementation of Distance Vector Routing protocol**

- NetworkAdress class: this class declared in simrouter.java has two field ipaddress denoting network address of the network and mask denoting subnet mask of the network.
- RoutingTable : Routing table is declared in simrouter.java as `HashMap<NetworkAddress,RoutingTableEntry >.`
- Creating some Utility class in Utility.java :
  - RoutingTableEntry: This class is to make a routing table entry. Here is 3 field. Distance,NextHop and timer(invalid timer for corresponding row).
  - RIPUpdate: This class stores and process RIPupdate packet received on this router. the process() method in this class updates the routing table according to the ripUpdate packet.
  - RescheduleableTimer: This is a timer implementation for update timer and invalid timer.
  - InvalidTimerTask: this class defines what is to be done when an invalid timer times out.

- Some useful functions defined in RoutingProtocol.java :
  - PrintRoutingTable() : prints the routing table on screen.
  - sendRoutingTableToInterfaces(): this method constructs a RIPupdate packet and sends it to all interfaces of the router. The packet has source ip as ip of the interface and destination ip as RP_MULTICAST_ADDRESS. The bytes of the array is saved to a ByteArray from index 1. Where at index 0 the destination mac is placed which is BROADCAST_MAC.
- Implementation of RoutingProtocol.run() :
  - First Routing Table is populated with directly connected interfaces using the SimRouter instance. Then routing table is printed.
  - Then the Routing table is forwarded to all interfaces and update timer is set on.
  - In the update timer the timertask that is passed essentially calls the handleTimerEvent function with argument 1.
  - Then a while(true) loop starts where continually it is checked if a RIP update has been received. When the buffer for storing Rip updates is empty the current thread waits until a update is received.
  - After getting a RIP update it simply instantiate a RIPupdate object and calls process method to update the routing table according to the update. Then routing table is printed if the routing table is changed and then the contents of the RIPupdate is printed.
- Implementation of HandleTimerEvent(): when argument is 1 it simply sends routing table to all interfaces . when it is 2 it is the event of invalid timer timed-out and it deletes the row which the corresponding argument NetworkAddress maps to.
- Implementation of getNextHop():
  - The whole routing table is searched to check if there is any entry for the network corresponding to the ip passed as argument. If it finds one the returns.
- Implementation of ontifyPortStatusChange():
  - The if argument is true then the corresponding port is up and the network address is added to routing table.
  - If the argument is false then the corresponding port is set down and the nerwork address for the corresponding interface is deleted from routing table. Also all the entries that has a next hop which has this interface id is deleted.
- Implementation of notifyRouteUpdate():
  - This method simply stores the update on buffer. If the buffer is full it simply drops it.
- Implementation of RIPupdate packet format:
  - It is implemented as a byte array. Where 1^{st} 4 bytes denotes the number of row of routing table.
  - Then each row is represented as 12 bytes. 1^{st} 4 bytes the ip of network address,2^{nd} 4 bytes mask and 3^{rd} 4 bytes denotes distance. Other fields of routing table is not necessary and not send.
  - RIPupdate.getbytes() method does all the necessary computation to make this RIPpacket.
- Some important issue :
  - As the timer class implements separated thread to run. We have made the code segements synchronized on the routing table wherever any critical region is detected so that if a thread is deleting or updating routing table other thread can not print of use it simultaneously which will create race condition and result in un wanted consequences.