Project Report

**Run Instruction:**

Main.java is the entry point of the program. The program can be compiled and run using a Java IDE such eclipse and netbeans, or from the terminal using “javac” (to build) and “java” (to run) commands. The output file will be created in the root directory of the project and will be named “outputfile”. Please add appropriate extension to view the file.

**Output Explanation**

Sample output:

HostC accepted packet with priority 1632

HostC: layer 2: received data from layer 1

HostC: layer 2: passing data to layer 3

HostC: layer 3: received data from layer 2

HostC: layer 3: passing data to layer 4

HostC: layer 4: received data from layer 3

Router accepted packet with priority 787

Router: layer 2: received data from layer 1

Router: layer 2: passing data to layer 3

Router: layer 2: received data from layer 3

Router: layer 2: passing data to layer 1

Router sent packet with priority 1633

As you can see, every layer of a host prints out message upon receiving data from the other layers and passing data to the other layers. Host also prints out a message when it accepts a packet, meaning when layer1 verifies that the packet is for that host and sends it to layer2, and also when it sends a packet, meaning when layer 1 puts the data on the “wire”. Packet priority in the sent message is to track the order in which the information was sent.

**Code Explanation**

1. **Main.java**

Like mentioned earlier, this is entry point of the program. The first thing it does is set up hosts by assigning them proper IP Addresses, MAC Addresses, and creating threads for each host. There’s a variable a called “**inputFilePath**”, which is the path to the file that we want to transfer. This can be modified with an appropriate file path on your specific machine. Right below, there is thread that will initiate file transfer. This can be modified to specify source host and destination host. For now, let us assume that we are transferring file from HostA to HostC. So after specifying the input file, you can run the program and it will start the file transfer. Once the transfer is complete, it will check MD5 checksum for both input and output file and will print out the result of the match.

To initiate the file transfer we will call sendFile method of Host.java with hostA object. This means that HostA will send file to the given destination IP Address (taken as parameter).

1. **Host.java**

Host.java represents the host and contains host information such as IP Address, Name, etc. The sendFile method will give layer4 the necessary information to send file, such as destination IP Address and file that needs to be sent.

The receive method will give Layer1 the packet that the host received over the network. Basically the way “receive” works is that each host will keep looking for packets in the broadcast queue. For example, if a host is member of Class A network, it will only look for any packet that are put in classABroadcast queue. Once it finds a packet, it will remove it and call receive method to send that packet to layer 1 for evaluation.

1. **Layer1.java**

Layer1.java represents the functionality of Layer1. The processReceivedPacket method compares destination MAC address of the packet to its MAC address. If both addresses match, then it will send the packet to layer2. Else, it will put the packet in the queue.

receiveFromLayer2 method receives data from Layer2 and puts it in the broadcast queue of the network that it is member of, so that the intended receiver and can receive the packet.

1. **Layer2Frame.java**

This class represents data in layer2 frame. It contains information such as CRC, body, preamble, source address, destination address, etc. The toByteArray method returns all these information in bytes. To see all the Layer2 headers that will be set, please refer to the variables declared in the first 20 line of the code.

1. **Layer2.java**

Represents functionality of Layer2. recieveFromLayer1 processes data received from layer1. It basically checks the checksum and if the checksum matches, it removes its headers from the data and sends the information to layer3.

recieveFromLayer3 method receives data from layer3. It calculates CRC, sets destination address (MAC from its ARP table), adds its headers and send the data to layer1.

1. **Layer3Packet.java**

Represents data in layer3 packet. Please refer to the variables declared in first 30 lines of code, for packet structure.

1. **Layer3.java**

receiveFromLayer2 method receives data from layer2. It verifies internet checksum. If the checksum matches than it will analyze the destination IP Address. If the destination IP Address matches to host’s IP address, then it sends data to the layer 4. Otherwise, it will look up next hop IP in its routing tables, set new headers and send data back to layer2, so the packet can be sent to the next hop.

receiveFromLayer4 method receives data from layer4. It calculates checksum, sets next hop IP address based on the routing tables, sets it headers, and sends information to layer2.

1. **Layer4.java**

receiveFromHost method receives information from host to initiate the file transfer. It reads the given file in chunks (Size = MTU – Layer2 header size – Layer3 Header size) and sends the read file chunk to layer3, along with the destination IP address.

receiveFromLayer3 method will receive data from layer3. It will read body of the packet to rebuild the file.

1. **Packet.java**

Just a wrapper class that contains data that will be read or written by layer 1 on the “wire”.