Network Simulator - Final Submission

Ashutosh Rahi, Rakshit Sharma, Baljeet Singh

June 14, 2018

1 Aim

To demonstrate the working of all the layers of Network Protocol Stack using a Network Simulator.

2 Programming Language Used

Java-8

3 Kilo Lines of Code (KLOC)

2.5

4 Project Status

Deliverable developed completely on June 14, 2018. Last Submission-4 update has been shown herein.

5 Strengths of the Project

- As close as possible to the real world scenario of networking, all MAC address are of 48 bits standard, and all IP addresses follow IPv4 classful addressing scheme, 32 bit addresses.Messages being sent are duly converted to binary using ASCII and headers and trailers of intended lengths have been made.

- The simulation clearly shows movement of packet from one node to another, clearly explaining all reasons for forwarding and showing all the intricate details invloved. **GUIs have been made for the project for demonstration of various application layer protocols.**

6 Assumptions

- The user is expected to enter the english language characters and/or numerals for which proper ASCII value is defined.
- The user may enter any number of end stations upto the end limit of integer data type in java, i.e. 2^{31}-1.
- Since the project is only a **Virtual Implementation** of the Network Protocol Stack, so the propagation time measured is the time elapsed between calling the data transfer function in sender and the return of ACK, so this is only the time indicative of round trip delay.
- Although the user is given full liberty to enter as many number of routers and end devices, yet as of now the implementation is shown only for topologies with 'n' no. of routers mutually connected, and two of them connected to switches, which will further be connected to any number of devices.
- The project expects input IP address as per classful addressing scheme as of now.

7 Tasks performed by the Physical Layer

- Transmitting raw bit stream over physical medium.
- Bit synchronization.
- Determining the physical topology of the network.
- Adjusting the datarate of the data being sent.

8 Tasks performed by the Data Link Layer

- Framing
- Flow Control

- Error Control
- Access Control
- Adds sender and destination MAC address to the frame.

9 Tasks performed by the Network Layer

- Adds sender and destination IP address to the packet.
- Responsible for routing of packets at each node, that is, the decision of route.
- Address Resolution : Finding MAC address of device in case it is not known, given its IP address.

10 Tasks performed by the Transport Layer

- Adds sender and destination port numbers to the packet.
- Responsible for process to process delivery of segments.
- End-to-end flow control, congestion control and error-control

11 Tasks performed by the Application Layer

- Closest layer to end-user.
- Several protocols like **DNS**, **SMTP**, **HTTP** work in Application layer only.

12 Object Oriented Paradigm followed

- Class **Device**:The template for any kind of device in the network.For now it has only one attribute i.e. MAC Address.
- Class **end-device** inherited from Class Device which adds additional attributes wiz. identifier,data to be sent,received data and implements methods wiz. send data,received data which further make a call to subsidiary methods translate which converts the received binary data stream to the message,binarize which binarizes the data to be sent into a data stream

- Class **hub** inherited from class device which adds attributes no. of ports and ports and implements method transfer data which performs the task of receiving data from one end device and sending it to all other devices in the network.
- Class **Switch** inherited from class device which forwards data to intended receiver after properly learning addresses of stations from incoming and outgoing frames.
- Class **Router** inherited from class device, which performs the task of *routing* of packets to optimum routes across the network. The Routers maintain a routing table each.
- Class **DomainNameServer** inherited from class device, which provides the **DNS** service.
- Classes **SMTPClient/SMTPServer** inherited from class device, which provide the **SMTP** service.
- class **network simulator** which implements the above classes .

13 Project Specifications

13.1 Framing

Fixed-size framing has been used in the project. The frame size is **282 bits** which is inclusive of the following:

- Bit 0 : Data Link Layer Stop and Wait protocol Sequence No. bit : 0 or 1.
- Bits 1 through 48 : 48 bit MAC address of the sender.
- Bits 49 through 96 : 48 bit MAC address of the receiver.
- Bits 97 through 176 : 80 bit dataword.
- Bits 177 through 184 : 8 redundant bits generated through CRC.
- Bits 185 through 216 : 32 bit Sender IP address.
- Bits 217 through 248 : 32 bit destination IP address.
- Bits 249 through 264 : 16 bit Sender Port Number.

- Bits 265 through 280 : 16 bit Receiver Port Number.
- Bit 281 : 1 bit Transport Layer Stop and Wait Protocol Sequence Number.

13.2 ARP Query Format

The Address Resolution Protocol (ARP) aims at finding the MAC address of destination end station when it is not known, but destination's IP address is known. For the same task, an ARP query is broadcasted over the networks and intended receiver sends it MAC address to sender of ARP query.

The ARP query format : ***Who is A.B.C.D ? Tell X.Y.Z.W***

The said string is first converted to binary as per ASCII and broadcasted over the networks.

13.3 Static Routing

Here user is prompted to enter details of networks not known to the router, and as can be seen router configuration is essentially manual and requires admin-intervention.

13.4 Error Control

Error Control has been achieved with the help of **Cyclic Redundancy Check**, aka CRC. The divisor used for the same is standard **CRC-8**, 100000111. The number of redundant bits generated is hence, 8.

13.5 Flow Control

As of now, **Stop and Wait protocol** has been implemented, in both Data Link Layer and Transport Layer, using two sequence numbers wiz. 0 and 1. The receiver and sender window size are both 1.

13.6 Messenger

A simple message exchange applciation program named *Messenger* has been developed which shows message input from sender side and transporting it, and showing it on receiver's device.

13.7 HTTP

HTTP Client/Server Model has also been developed, wherein user types in the URL **www.server.com** and the corresponding connection request **INITIATE HTTP** in his web browser, and gets displayed google's webpage on successful connection to the server.

13.8 DNS

The DomainNameServer instances made store databases (implemented using **HashMap**) mapping domain names to corresponding IP addresses. The sender device first sends a DNS query to one of these servers (closest one) to which the DNS server responds with the corresponding IP address.

13.9 SMTP

The user is prompted to enter sender's (his own) email address and receiver's email address, and the email he wishes to send. At the receiver end, he is notified with a dialog box first, and then when he clicks OK, he can see the mail loaded on his screen.