CS F303 Computer Assignment 3

Team Members

NAME	ID NUMBER
Prateek Agarwal	2017A7PS0075H
Utkarsh Grover	2017A7PS1428H
Shreeya Bharat Neelekar	2017A7PS0093H
Rashi Jain	2017A7PS0082H
Ankit Bansal	2017A7PS0159H

Project Info

Source Code:
https://github.com/prat-bphc52/ReliableUDPProtocol

Source Language:
Java

MyReliableUDP Protocol

Introduction:
UDP sockets are used to send data over the network however compromising the guarantee of delivery of the packets.
A MyReliableUDPProtocol designed by us is an implementation of UDP Protocol in the Application Layer which makes it a reliable one. This protocol is designed using UDP sockets and ensures the packets and ensures that the entire data is transmitted successfully taking care of Packet Loss, packet corruption or other issues that leads into the loss of Data.

Protocol Specification:

MAX_PACKET_DATA_SIZE = **512 bytes**
Types of Packets:
There are majorly 3 types of packets that are used to transfer data using this protocol :

1. **Data Packet** :
   Packet Size = 512 Bytes
   Packet Data Contents:

4 bytes Sequence Number	**8 bytes** Packet ID (long value)	**480 bytes** Data	**20 bytes** SHA1 key hash

2. **Acknowledgement Packet** :
   Packet Size = 32 Bytes
   Packet Data Contents:

4 bytes Sequence Number	**8 bytes** Packet ID (long value)	**20 bytes** SHA1 key hash

3. **Transmission ID Packet :**
   Packet Size = 40 Bytes ()
   Packet Data Contents:

4 bytes Sequence Number	**8 bytes** Packet ID (long value)	**4 bytes** Total number of packets	**4 bytes** Total data length	**20 bytes** SHA1 key hash

** Note - The above mentioned sizes only refers to the JAVA DatagramPacket **DATA** size and not the other headers and other info of a packet used by UDP Protocol

Protocol APIs:

1) MyReliableUDPSocket.create(int port, InetAddress hostAddr)
   This function creates a JAVA DatagramSocket and binds it to the specified port and address

2) MyReliableUDPSocket.send(byte[] arr, InetAddress destAddr, int destPort)
   This function sends all the bytes present in **arr** to the socket bound at the specified address and port. This function is blocking and waits until the entire data has been sent successfully.

3) MyReliableUDPSocket.receive()
   This function receives the entire data that has been sent using the above specified send() function from any socket. It stores the IP Address and packet ID obtained from the first packet and subsequently accepts only those packets which match the source IP address, port and packet ID.
   Return type - Returns a byte array storing the entire data received by it

Protocol Working and handling of packet loss, corruption:
Initially whenever a transmission begins, a **TRANSMISSION_ID** type packet is created and assigned the sequence number **1**. This packet stores all the info related to this transmission such as the number of packets that would be required to complete this transmission i.e. entire data to be sent, and also the total length of data i.e. length of byte array sent using send() function. It also creates a transmission ID equal to the system's current timestamp to uniquely identify each packet which transmission it belongs to. This is to ensure that packets of one data type don't get mixed up with other packets of another data type.

Next the entire data is divided into the number of required **DATA** type packets. The packets are serially generated and assigned sequence numbers starting from **2** (1 is reserved for Transmission ID type packet only).

Finally whenever a packet(transmission_id or data type) is received at the receiver's end, an **ACKNOWLEDGEMENT** type packet is created for that packet and this packet is assigned a sequence number which is negative of the sequence number original packets (this is done to distinguish between the 3 types of packets). This acknowledgement is only sent if the received packet is not corrupt which is verified using the hash key stored in the packet as specified below.

All of the above packets are also appended with a **20 byte long SHA-1** hash key for that particular packet. The hashkey is generated from the rest of the data bytes present in this packet.

When the send() function is called, a new thread is created to receive acknowledgements.
All the packets (1 Transmission ID packet and remaining Data packets) are sent initially.
The acknowledgement receiver thread maintains a HashMap of sequence numbers as keys and boolean values to keep a track of the packets whose acknowledgements have been received.
Once all packets have been sent, the hashmap maintained by the receiver thread is looked up and all those packets whose acknowledgement haven't been received are resent. This ensures that there is no packet loss in the long run.