Application Layer Protocol for UDP packets

Gotam Dahiya
2017A7PS0223H

Hariharan R
2017A7PS0065H

Srikruthi
2017A7PS0086H

Arun Parimi
2017A7PS0221H

Sricharan
2017A7PSyyyyH

Abstract:

For providing reliability to the UDP sockets an application layer protocol was developed which provides reliability via retransmission, checksum, sequence and acknowledgement numbers. These aspects will be explained in detail in this paper.

Assumptions about the Application and Network:

One basic assumption taken in this assignment is that the server and the client are connected to each other on the same network. This is taken into consideration as the server runs on the local machine, with the client working on another local machine connected to the network. No images, audio or video messages are passed between the server and client. Only one client and server talk to each other until either one of them closes the connection by passing a particular keyword. The application does not run until the server is running on a local machine.

Due to encryption of messages the public key entered at both the client and server should be the same otherwise erroneous messages are displayed at either end. Only the message is encrypted, not the whole packet, so as to avoid unnecessary packet loss.

Only a small buffer of the most 20 recent messages are stored in a dictionary with corresponding sequence and packet numbers for retransmission in case of packet loss.

If the client closes the connection by passing the keyword “FIN” then both the client and server shut down. If the server passes the keyword “FIN” then only the client shuts down.

Strategies to Tackle different Network Issues:

1) Checking for Parity Errors:

    The checksum of the packet to be sent is calculated. This helps in verifying whether there has been any corruption in the packet or not while being transmitted.

    The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. This is given by RFC 791.

    This basically means that every 16 bits are added to each other and the final summations answer is taken as one's complement. The same thing happens at the receiver side and the 2 values are compared. If equal then no parity errors, otherwise resend packet.

2) Correct order of packets:

    For correct sequencing of packets a protocol similar to Real Time Protocol(RTP) is used in which the sequence is for client to server and acknowledgement is for server to client. If the difference between current sequence/acknowledgement and previous sequence/acknowledgement is not less than equal to one then the packet which was lost is rendered to the receiver through the buffer.

    The sequence numbers are updated at the server side while the acknowledgement numbers are updated at the client side. This allows for easier debugging and a simpler protocol to implement than TCP.

    The default values for sequence and acknowledgement numbers are taken as zero.

Encryption of Messages:

The messages are encrypted from all the ASCII characters(0,127). A seed number for the random generator is taken by the server and client. These two numbers have to be the same as they will create the same random number for the server and client. The seed number is like a public key for the connection.

A list of all characters is shuffled using the random key generated. The corresponding characters index is taken to generate a new string which is passed to the receiver. This is decrypted on the receiver's side after checking for parity error and sequencing of packet transmitted.

References:

1) TCP packet built from scratch in Python 3 https://gist.github.com/NickKaramoff/b06520e3cb458ac7264cab1c51fa33d6
2) Creating UDP packet from scratch https://github.com/houluy/UDP/blob/master/udp.py
3) Checksum RFC 791 https://tools.ietf.org/html/rfc791