RELIABLE UDP:
Not So Unreliable UDP: NSUU

Computer Networks: Assignment 2, Phase 1

CS F303, Second Semester 2020-21

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI,

HYDERABAD CAMPUS.

Mudit Chaturvedi	2018A7PS0248H	Group 1
Hardik Parnami	2018A7PS0062H	Group 1
Kriti Jethlia	2018A7PS0223H	Group 1
Sristi Sharma	2018A7PS0299H	Group 1
Mereddy Aishwwarya Reddi	2018A7PS0276H	Group 2
Rupsa Dhar	2018A7PS0376H	Group 2
Pranavi Marripudi	2018A7PS0507H	Group 2
Adesh Kumar Pradhan	2017B3A70960H	Group 2
Rishabh Jain	2018A7PS0275H	Group 2

Assumptions:

- ❏ The protocol itself is neutral towards client-server terminology and allows 2-way file transfer. However, the report follows the convention that the server is the sender of the file(s) and the client is the receiver of the file(s). The other way file transfer follows a vice versa case.
- ❏ Timeout is assumed to be 1 second.
- ❏ Maximum Retransmission for any packet is 13 times.
- ❏ Window Size is advertised at the start of the 3-way handshake and passed in the SYN Flag as a non-zero value.
- ❏ Maximum Segment Size (MSS) is 64 bytes.

1. Introduction:

NSUU is an application layer transport protocol designed along the lines of the traditional unreliable UDP but with an assurance of the reliability of data transmission.
Employing the Selective Repeat Paradigm, the protocol can successfully transfer files between client and server running on reachable IP addresses.
The primary concern involved in using the classic UDP is the unreliability of data transmission since the datagrams are sent without any provision of acknowledgments and error connection. To combat these shortcomings, our protocol employs numerous features, as discussed in the subsequent sections.

2. Features:

NSUU provides the following features to ensure reliability:

- ❏ **Three-way Handshake**: This is used to ensure that both client and server have synchronized their sequence numbers and have a reliable connection established between them. A conventional SYN, SYN+ACK, ACK method is implemented, with the client initially sending a packet with the SYN bits set (SYN is of size 4bytes and will contain the window size, if SYN is set then it will be non-zero and equal to the window size). To which, the server responds with a packet with both SYN and ACK bits set. Upon receiving this SYN-ACK packet, the client responds with an ACK of its own along with the request.
- ❏ **Flow Control**: During the handshake, both Client and Server advertise the Window size to convey how much data they can accept at once. This is done to deal with the flow control problem.
- ❏ **Timeout**: To deal with lost packets, upon sending a packet, the server starts the timer. When this timer exceeds the stipulated period, it is termed as a timeout. If a timeout occurs, then the packet needs to be retransmitted.
- ❏ **Triple DupACK**: When the sender receives ACKs with the same sequence number 3 times, it assumes the packet to be lost and retransmits it.
- ❏ **Maximum Retransmission**: When a particular packet's ACK is not received even after retransmitting it numerous times (more than the maximum retransmission

threshold), the connection is assumed to be closed.

- ❏ **Error Detection**: A 84-byte hashed checksum is used to ascertain data integrity in the transmitted packet.
- ❏ **Maximum Segment Size (MSS):** The maximum size of the data that can be received by the receiver in a particular packet.

3. Packets:

NSUU supports two types of packets, Data and ACK, namely, with the Data packet being much heavier than ACK. If the packet received has more than two delimiters, then it is a Data packet else ACK packet.

A. Data Packet:

4 bytes	1 bit	1 bit	4 bytes
SYN Flag	ACK Flag	Fin Flag	Sequence Number
64 bytes	Actual encoded Message		
84 bytes	Checksum		

Size of Data Packet = 157 bytes.
Each data packet will contain the following information:

- ❏ SYN Flag
- ❏ ACK Flag
- ❏ Fin Flag
- ❏ Sequence number of the packet
- ❏ Actual encoded message
- ❏ Checksum

The message to be sent is encoded, and its checksum is calculated using the sha1 hash function in python. Sequence number shall serve as the packet number, and as per Selective Repeat requirements, available Sequence numbers must be at least twice the window size.
SYN Flag contains the window size and will be non-zero (equal to window size) whenever the SYN bit needs to be set. The SYN and ACK flags are used in conjunction to depict the SYN, SYN-ACK, and ACK phases of the 3-way handshake. The client will close the connection by replacing the FIN flag with the SYN flag.

B. ACK Packet:

4 bytes	4 bytes
ACK number	Sequence Number

Size of the ACK packet = 8 bytes (<< size of Data packet)
Each ACK packet will contain the following information:

- ❏ ACK Number
- ❏ Sequence Number

ACK number will refer to the sequence number of the packet for which this ACK is being sent, and the sequence number field will contain the next expected packet number.

4. Client and Server Roles:

A. **Server**:

- The server will receive a request from the client for connection, along with the initiation of a 3-way handshake.
- Sending the next packet: The server will send all the packets in the current window at least once, and if there are no retransmission requests, it will wait for the window size to expand.
- Checksum calculation: The server will calculate the message's checksum and append it to the header before sending the packet to the client.
- Expansion of window: The server will expand its window towards the right once it gets ACK for the packet at the window's extreme left end.
- Retransmission: Retransmission of packet would occur in 2 cases, Timeout and Triple Duplicate ACK(a.k.a. Triple DupACK)
- Closing connection: The server will close the connection if a packet has been retransmitted for K times, K here being an upper limit for the number of retransmissions.
- The server will receive a request from the client for closing the connection.

B. **Client**:

- When the server's packet is received, the checksum value is verified to see whether it matches with the data received.
- Given that the checksum is correct, we check the sequence number of the packet received and match it with the expected sequence number, i.e., one greater than the last sequence number received. We may come across two cases.

Case 1: Received and expected sequence number match

In this case, we will update the expected sequence number by incrementing it by one. The window will be moved from the given location by as many places as the number of packets received, i.e., whenever we come across an empty slot in the window, we stop the window there. Whatever we have passed while moving the window is sent to the application layer.

Case 2: Received and expected sequence number do not match

- **The packet is a duplicate**: If the received sequence number is less than the expected sequence number, the packet is a duplicate, and we will discard it. Since our expected sequence number always updates to the minimum packet number in the receiver window, which is yet to be received. If the received sequence number is greater than the expected sequence number but within the window size then we check if we have received this earlier or not. If yes then the packet is a duplicate and needs to be discarded.

- **The packet is out of order**: In this case, the received sequence number is greater than the expected sequence number but within the window size and we have not received it earlier. We will accept the packet and send an ACK with the received sequence number and the expected sequence number from before.

- Given checksum is wrong:

  This means that the packet has been corrupted.

  ❏ **Case 1:** Received sequence number is equal to Expected sequence number. In this case, the expected sequence number is kept the same, and we need not send an ACK.

  ❏ **Case 2:** Received sequence number is not equal to Expected sequence number. Again we won't send an ACK.

5. References:

❏ RFC 768: User Datagram Protocol

❏ Reliable UDP: Wikipedia

❏ Draft-IETF: Reliable UDP

❏ Reliable UDP: Kansas State University