COMPUTER NETWORKS

Assignment - 2

Name	ID Number
Gaurang Gupta	2018A7PS0225H
Rushabh Musthyala	2018A7PS0433H
Mihir Bansal	2018A7PS0215H
Jhaveri Aditya Alok	2018A7PS0209H
Dev Gupta	2017B3A71082H
Rishi Saimshu Reddy	2018A7PS0181H
Abhinav Bandaru	2018A7PS0236H
Pranay Tarigopula	2018A7PS0237H
Pranav Reddy Pesaladinne	2018A7PS0238H
Vishal Dheeraj Donkada	2018A7PS0239H
Dhruv Adlakha	2018A7PS0303H

TABLE OF CONTENTS

Introduction

Standalone UDP as a transport layer protocol does not implement reliability. So to implement reliability on top of UDP, we have to design an application layer middle-ware protocol. This document describes the remote function calls interface for the UDP transport layer, which helps communicate between the client and the server.

The proposed protocol uses the 3-way handshaking concept to establish connections reliably. It uses timers, ACKs, and sequence numbers to ensure lost/corrupted packets can be recovered and delivered in order.

Packet Structure and Description

To implement reliability over UDP, we define a modified packet in our application layer that the server and client will receive and will have the following structure.

Application Layer Header - This contains the information necessary for the proper functioning of the protocol. It is divided into (number of bits mentioned in brackets) -

- *Single bit flags* - SYN, FIN, ACK flags.
    - Dummy flag - value is always set to 0. (1)
    - SYN - Used to Indicate the establishment of the connection. (1)
    - FIN - Indicates the termination of the connection. (1)
    - ACK - Indicates if the packet carries a valid acknowledgment number. (1)
- *Packet Number* - Indicates the packet number being sent by the sender to the receiver (4)
- *ACK Number* - Indicates the packet number of the packet being acknowledged. (4)
- *Checksum* - Hash computed on the packet data to ensure integrity. (16)
- *Body Size* - Indicates the size of the actual data being transmitted. (12)
- *End Of The Header* - Special symbols to indicate the end of the header. (4)

Body - This is the data that needs to be sent from the server to the client.

Implementation Specifics

We had to add the following parameters in the application layer over the pre-existing selective repeat paradigm for implementing reliability over UDP.

- **3-way Handshake** - To implement the above protocol, the client and the server have to start a connection by a 3-way handshake. Once the connection has been established, the transmission of the packets can begin.
- **Global timer** - Ensures that there is no stagnation in the connection by abruptly closing the connection if either side receives no packets in a specified interval.
- **Packet Size** - The client and server have to decide on a standard packet size that will not overflow the buffers on either side.
- **Buffer Size** - The client and server's buffer size is set such that it is greater than the product of the packet size and the sending window size.
- **Window Size** - Denotes the maximum number of outstanding packets in the connection.
- **Maximum Retransmission Count** - The maximum number of times a packet can be retransmitted before the connection is closed.

Client-Side

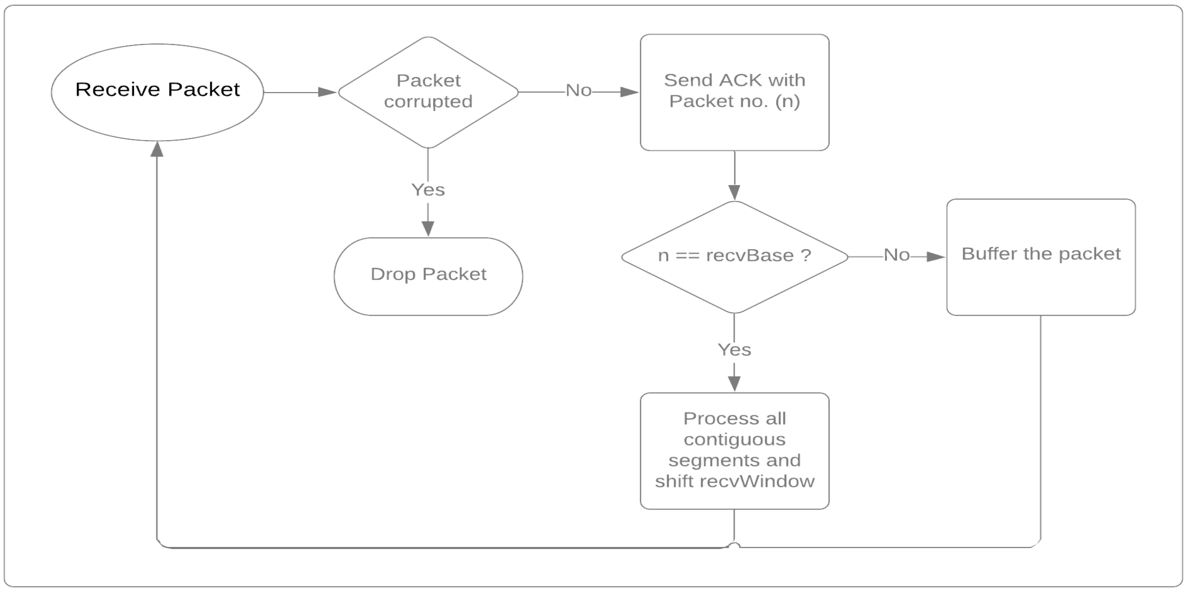
- The client sends the first packet to establish a connection with the server by setting the SYN bit to 1 and starts the timer. Two possibilities can occur here.
    - Response is received from the server: In this case, the client can now send the request.
    - Response is not received, and timer times out: In this case, the client resends the connection request and resets the timer.

- After receiving the server's confirmation message, the client can now send its request to the server (SYN bit is set to 0). The client must initialize a buffer to receive data sent from the server. The client must also establish a receiver window to understand the identity of the packet sent by the server (using recvBase pointer).

- From this point onwards, packets received from the server contain parts of a response to the request sent. When a packet is received, there are two possibilities:
  The packet's integrity is checked using a checksum -
    - The packet is not corrupted, the client sends an ACK for the packet number.
        - If the packet number is present in the current receive window:
            - If the packet is out of order, then the packet is buffered.

            - If the packet number equals recvBase, then the packet along with all contiguous packets in the buffer are processed simultaneously, and the window shifted accordingly.
        - If the packet number is not in the current receive window, the client sends an ACK with an ACK number of the current packet number.
    - The packet is corrupted, in which case the client drops the packet without sending any ACK.

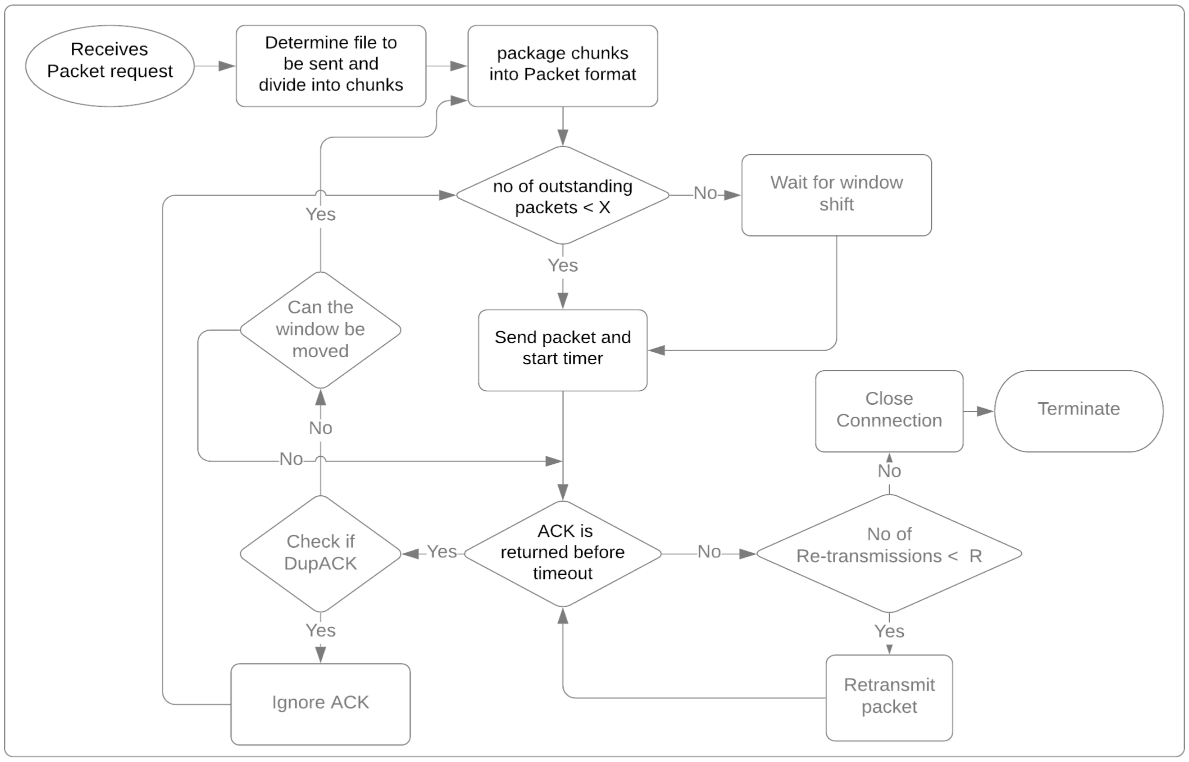
- When all packets are ACKed and received by the server, a termination is sent by the server. Upon receiving which, the client must send an ACK and deallocate all its resources.



Server-Side

- At the start of the connection, receive the packet/request from the client and check if the SYN bit is set to 1.
    - *If Yes* - verify the connection parameters and send an ACK, and set SYN bit.
- Receive request message from the client
    - Determine which file is to be sent and divide it into chunks according to the pre-defined packet size agreed upon by the client and server.
- From this point onwards, the server can start sending the packets to the client.
    - The chunk of data is isolated, and a packet is made from it by setting the appropriate flags and computing the checksum.
    - Once the packet has been made, it can be sent to the client, and the corresponding packet's timer can be started.

  - There can be a maximum of X outstanding packets in the connection.
  - After sending a packet, the server waits for an ACK, four things can happen here -
    - *Expected ACK comes* - check if the window can be moved ahead, if yes, send more packet(s).
    - *Timeout* - Resend the packet which caused the timeout, provided the number of retransmissions for that packet does not exceed the limit.
    - *Duplicate ACK* - Ignore
    - *If a corrupted ACK is received* - Ignore
- When all the packets are ACKed, the server sends a FIN bit enabled packet to the client to indicate the close of the connection. The server waits for an ACK from the client for the same and then closes the connection or closes the connection upon timeout.


Receives Packet request
Determine file to be sent and divide into chunks
package chunks into Packet format
no of outstanding packets < X
No
Wait for window shift
Yes
Yes
Can the window be moved
Send packet and start timer
Close Connnection
Terminate
No
No
No
Check if DupACK
Yes
ACK is returned before timeout
No
No of Re-transmissions < R
Yes
Yes
Ignore ACK
Retransmit packet