Computer Networks Assignment II- Phase 1

Group 1:

- **Abhishek Bapna**
  2018A7PS0184H
- **Ashna Swaika**
  2018A7PS0027H
- **Siddhant Kulkarni**
  2018A7PS0185H
- **Sravani Garapati**
  2018A7PS0097H
- **Vikram S Haritas**
  2018A7PS0302H

Group 2:

- **Anany Prakhar**
  2018A7PS0211H
- **Ashish Makundilal Verma**
  2018A7PS0009H
- **Debmeet Banerjee**
  2018A7PS0385H
- **Kartikey Papnai**
  2018A7PS0228H
- **Nimish Gupta**
  2018A7PS0372H
- **Siddhant Kharode**
  2017B5A71619H

Background & Motivation

The main types of Transport Protocols in Computer Networks are TCP which is a connection-oriented protocol with error checking and recovery, and UDP being connectionless protocol without error recovery, making it unreliable but a lot faster than TCP.
There is a lot of overhead in TCP making it slower but a reliable choice. To increase the network capabilities of a TCP bound network or to give a speed bump to an existing application, using UDP is a safe bet given there is reliability ensured in the Application layer.
The Main intention behind this project is to create a reliable Communication Protocol while making sure that error-checking and data recovery is implemented.

Requirements

We propose a protocol which ensures transfer of data over UDP that is an unreliable protocol. This protocol ensures correct transfer of documents even when the connection is extremely unreliable. With this protocol, we aim to transfer documents with 0 data lost. Parallelizing the sender's send and timer functionality will make the ender much faster than if it were a stop and wait algorithm. The only way to achieve reliability is if there exists 2 way communication between the sender and receiver and this will also be implemented.

Implementation

Connection Establishment

- The machine which establishes connection can only send data. The sender is the client and receiver is server. The server can only send back acknowledgments.
- Socket is closed after each successful data transfer.

- To establish a connection, a sender sends a CONNECTION_INIT packet which has the PACKET_SIZE & WINDOW_SIZE & randomly generated SEQUENCE_NO, at intervals of 2 seconds until the CONNECTION_INIT_ACK response is received from the receiver.
- Each packet will have an INDEX number to identify retransmissions & reassemble the file from the packets once all the packets are received. If the receiver has received a CONNECTION_INIT packet again after sending the ACK packet for CONNECTION_INIT, the receiver discards all these CONNECTION_INIT packets.
- Sender Receiver pair will close connection only when the connection close packet is sent by the sender and the CLOSE_ACK response is sent by the receiver and received by the sender or 5 minutes after inactivity.

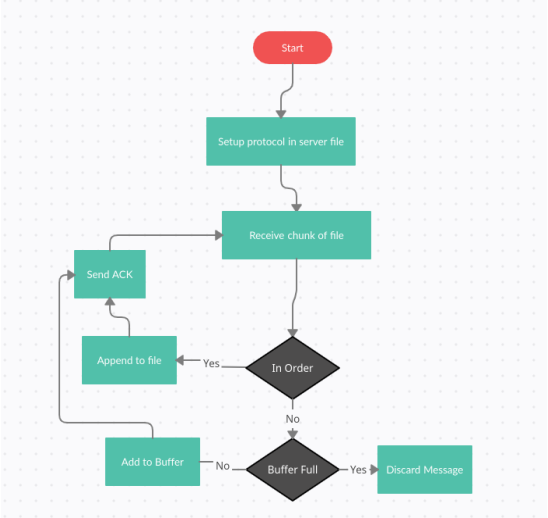
When the client accepts a CONNECTION_INIT_ACK packet, it means both client and server are online and can initiate data transfer from client to server.

The time difference between sending the first CONNECTION_INIT packet and receiving the CONNECTION_INIT_ACK packet will be RTT.

Sending Data: Selective Repeat

On an ideal test setup, there will be two PCs running, one running server file and the other client file. Both of these import protocol files. Client will upload file to server. Client file will read the data, chunk it and pass it onto the protocol file to send. Data transfer will occur between two protocol files running on different machines over the internet. Server will receive chunks from the protocol file in order. Server files will fuse these chunks.
Using the selective repeat (SR) paradigm, the files will be transferred. Each sent packet will be acknowledged by the server by sending back the ACK packet.
The choice of window size for SR will be decided in the implementation phase. There are two choices: set a predefined window size based on available information present in the problem statement, or choose window size based on RTT.

Client-side application



Server-side application

Packet Structure

A packet has structure based on its implementation.

The different information is delimited by ‘;’ by default the packet has an integer embedded in the beginning and closed by a ‘;’.

1. Regular Packets have an index at the beginning followed by a ‘;’ followed by the data that is being sent (in bytes encoded by UTF-8 format) which is then once again followed by a ‘;’ to close the packet.
2. Special Packets
    a. These packets have the index as -1, they serve as special purpose packets, there are 4 different types of special packets:
        i. Hello: This packet is used to initialize the connection which is sent by the one who wants to send a packet to the server, it has multiple fields, the first being index -1 and the rest are:
            1. String “HELLO”: Indicates the HELLO message.
            2. Filename: This is what is used as the filename for the data received.
            3. Count: The number of Packets to be transfered.
            4. Window Size: Used for transfer of packets.
        ii. HelloBack: This packet is sent by the receiver to acknowledge the connection request sent by the sender. This packet contains the string “HELLOBACK”.
        iii. Close: This is sent by the sender when all packets are Acknowledged. This packet is sent by the sender to close the connection. This packet contains the string “CLOSE”.
        iv. CloseBack: This packet is sent by the receiver to acknowledge the close connection request sent by the sender. This packet contains the string “CLOSEBACK”.

Table 1: Packet structure showing 2 chunks

Header
Data

Table 2: Descriptive packet structure

Flags	Checksum	Header
/r/n	/r/n	
Sequence number		
Data	Data	Data
Data...	Data...	

Table 3: Flags

Hello bit	HACK bit	Close bit	CACK bit	Data bit	ACK bit

Types of packets

1. Hello packet
2. ACK packet
3. Close packet
4. CACK packet

Flags

1. Hello bit: Set if hello packet
2. ACK bit: Set if ACK packet
3. Close bit: Set if Close packet
4. CACK bit: Set if CACK packet
5. Data bit: Set if Data is being sent

1. hello

   Cl= file sender- conn initiator- init rand() = ACK or init zero()= ACK+ “HELLO”

   112+HELLO

   Sv= file recv init rand() =ACK

   247+HELLOBACK

Window index: Packet Index [0, 1, 2, 3, 4, … W]

Global Index: File Packet Index(Seq no)

Seq no
Data