Assignment Report

1. **Introduction**

UDP implementation for file transfer is a UPD client-server application that will implement UPD to download a big file from a server residing in different IP location. The client process will also implement Selective Repeat protocols as UPD is a connectionless protocol, and also the alternating bit protocol. Basically, a client will make an HTTP request to the server to download a file. Server will check if the requested file exists or not. If the file exists, server will split the file into many segments of specific UDP length. It will add a checksum and alternating bit in each of segment before sending the file to the client. Once the client receives the file, it will verify the checksum and will wait for another segment till all the segment have been received. It will take a necessary action based on verification, such as storing the file or requesting again for the missing segment – UPD packet loss case. Since selective repeat protocol will be implemented, the client will make multiple requests to the server before sending the acknowledgment. All the action performed during this process will be saved in a file and also will be displayed in the console – i.e. both of client and server.

1. **Architecture Description**

The perspective architecture of this project is shown below in figure 3.2. Basically, it will contain client process and a server process. The client process will have a socket handler class. It will create a socket and hold some parameters, such as IP address, port number, message, error bit, alternative bit handler and so on. It will also contain some methods such as error handler, request handler, file write, and so on.

The server process will also have a socket handler class. It will also create a socket, to read data from the client. It will contain attributes such as a checksum, length, sequence no, message, and so on. Some of the methods for server socket class will be, file reader, connection handler, segment handler and transmission methods.

The UDP client will read file name at the very beginning and will send request to the server to download the file. The server will verify the existence of the file, and if it founds the file, it will break the file UDP segments. It will perform necessary action on the file, such as an addition of checksum and will send the file to the server. Meanwhile, it will also track the send status and will continuously print the status. This process will also be carried out by the client. Since the program will also implement alternative bit protocol, server and client will perform necessary action to achieve it.

Additionally, to handle unexpected network problem, client will save the download segment information, and segment as well. This is to facilitate user to start download from the place it was broken earlier if desired. This functionality is an additional thought to the project, so it may not be achieved if more complication arises during implement.

* 1. **Class Diagram**



Figure 3.1: Class Diagram UPD file download

**2.2 Flow Diagram**

The flow diagram of the UPD file transfer is shown in the image below.



Flow Diagram: UPD file download process using client-server model

Some of the functionality may be added and some of them may be changed. Thus, down the line, there may be some changes in this diagram as the project progresses.

*Note: The symbols used in the flow diagram are adopted from the standard flow diagram design.*

**2.3 State Diagram**

****

Fig 3.3: State diagram, client server - request response process

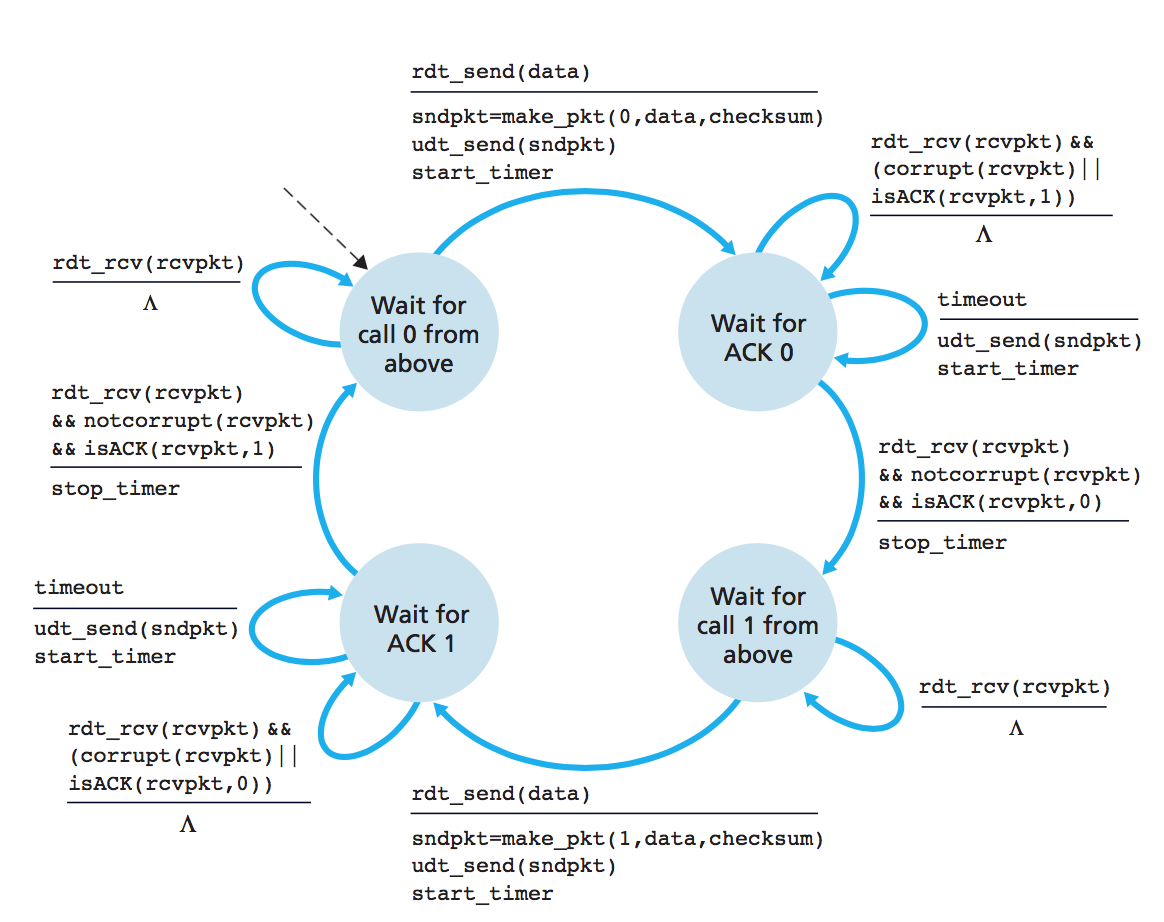


Fig 3.4: State Diagram - Alternate bit protocol

*Reference: Computer Networking – A top down approach 6th edition, Kuros, Ross*

1. **Implementation**

* Python 3.x+ for client and server programming
* Python socket library for the network socket

1. **Conclusion**

Thus, a UDP file download application will be developed in python. Along with the basic UDP implementation consideration, the application will also aim to implement UDP packet loss handling, **Alternating Bit**and **Selective Repeat protocols while download segments of a file from a server.**

1. **Results**

DELAY

100MS - 1.0056SEC 0.9944KB/SEC

400MS - 4.0070 SEC 0.2495KB/SEC

700MS - 7.0060 SEC 0.1427KB/SEC

1000MS - 10.0052SEC 0.0999KB/SEC

DUPLICATES

10% - 4.5MILLISECONDS 222.22KB/SEC

40% - 5.9 MILLISECONDS 169.49KB/SEC

70% - 6.9 MILLISECONDS 144.927KB/SEC

100% - 13.3MILLISECONDS 75.18KB/SEC

REORDER

10MS - 0.0756 SEC 13.227KB/SEC

40MS - 0.4063 SEC 2.461KB/SEC

100MS - 0.9061 SEC 1.103KB/SEC

400MS - 3.6059SEC 0.277 KB/SEC

JITTER (200MS)

10MS - 2.0236SEC 0.4941KB/SEC

40MS - 1.9714SEC 0.5072KB/SEC

70MS - 2.161 SEC 0.4627KB/SEC

100MS - 2.0940 SEC 0.477KB/SEC

150MS - 2.1750 SEC 0.4597KB/SEC

