2017

UDP Implementation for file transfer

FOR THE FULFILMENT OF NETWORK AND INFORATION ASSURANCE PROECT

THE UNIVERSITY OF MEMPHIS

SAURAB DULAL

Table of Contents

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 on the basis of 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.

2. Problem Statement

The application should consist a client program and a server program. The server program hosts files and responds to requests for files. It breaks the requested file into segments and sends them to the client over UDP. The client program takes a file name as input and requests the file from the server. The programs should handle UDP packet losses to make sure that the entire file is correctly received by the client program. Code needs to include implementation of the **Alternating Bit** and **Selective Repeat protocols**. Refer chapter 3.4 (Principles of Reliable Data Transfer) - Computer Networking, Kurose to understand these two protocols.

Client program should print detailed information about any requests sent and data received at byte level (e.g., time X received byte Y to Z) during the file download. Server program should print detailed information about any requests received and data sent at byte level. Client program and server program should be run on different machines in different networks.

3. 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.

3.1 Class Diagram

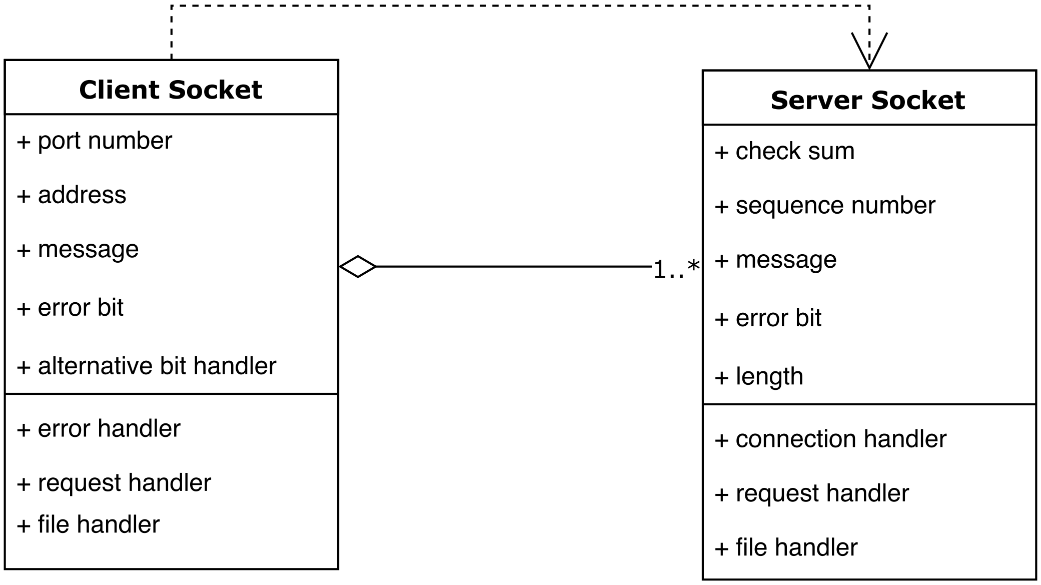


Figure 3.1: Class Diagram UPD file download

3.2 Architecture Diagram

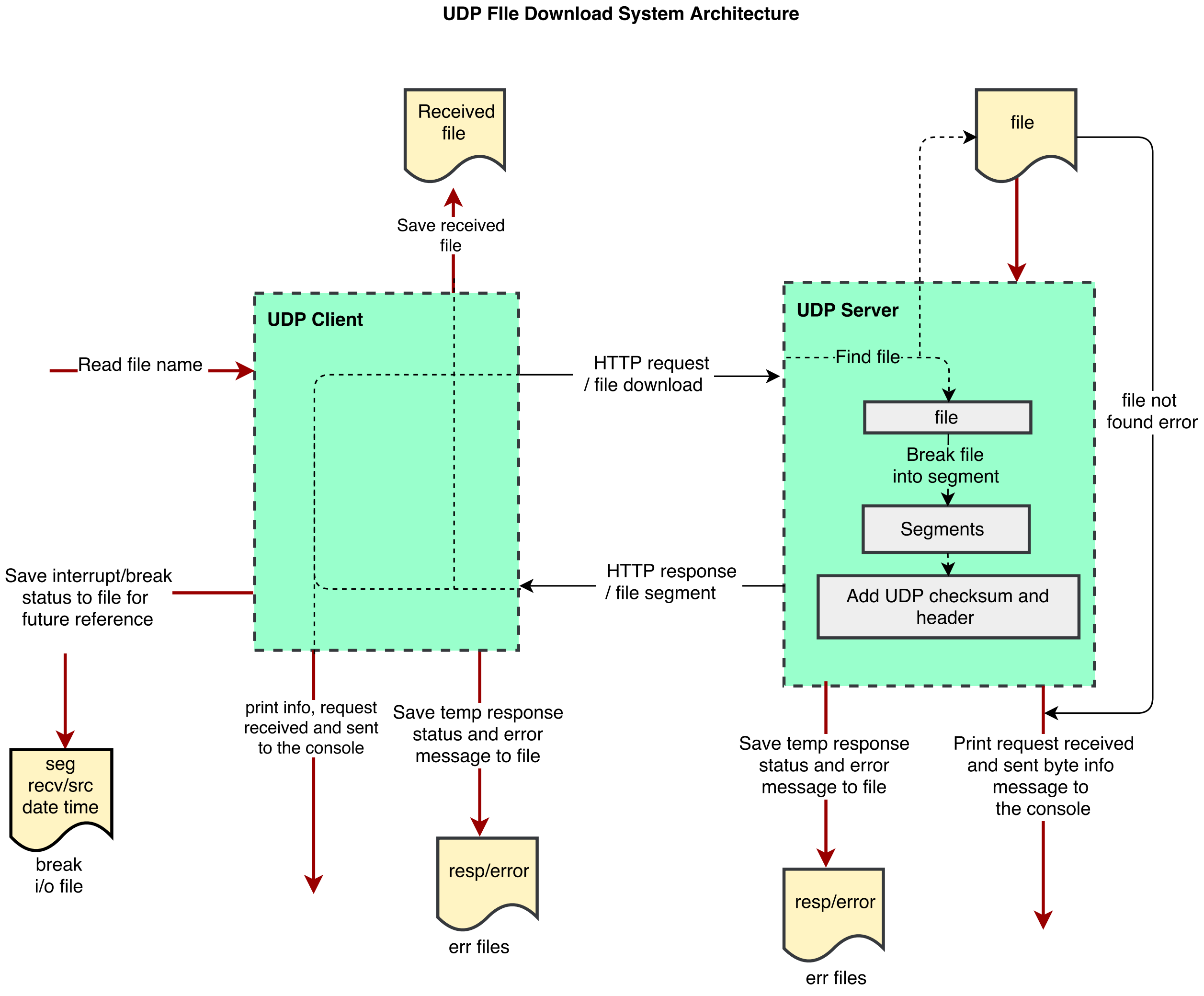
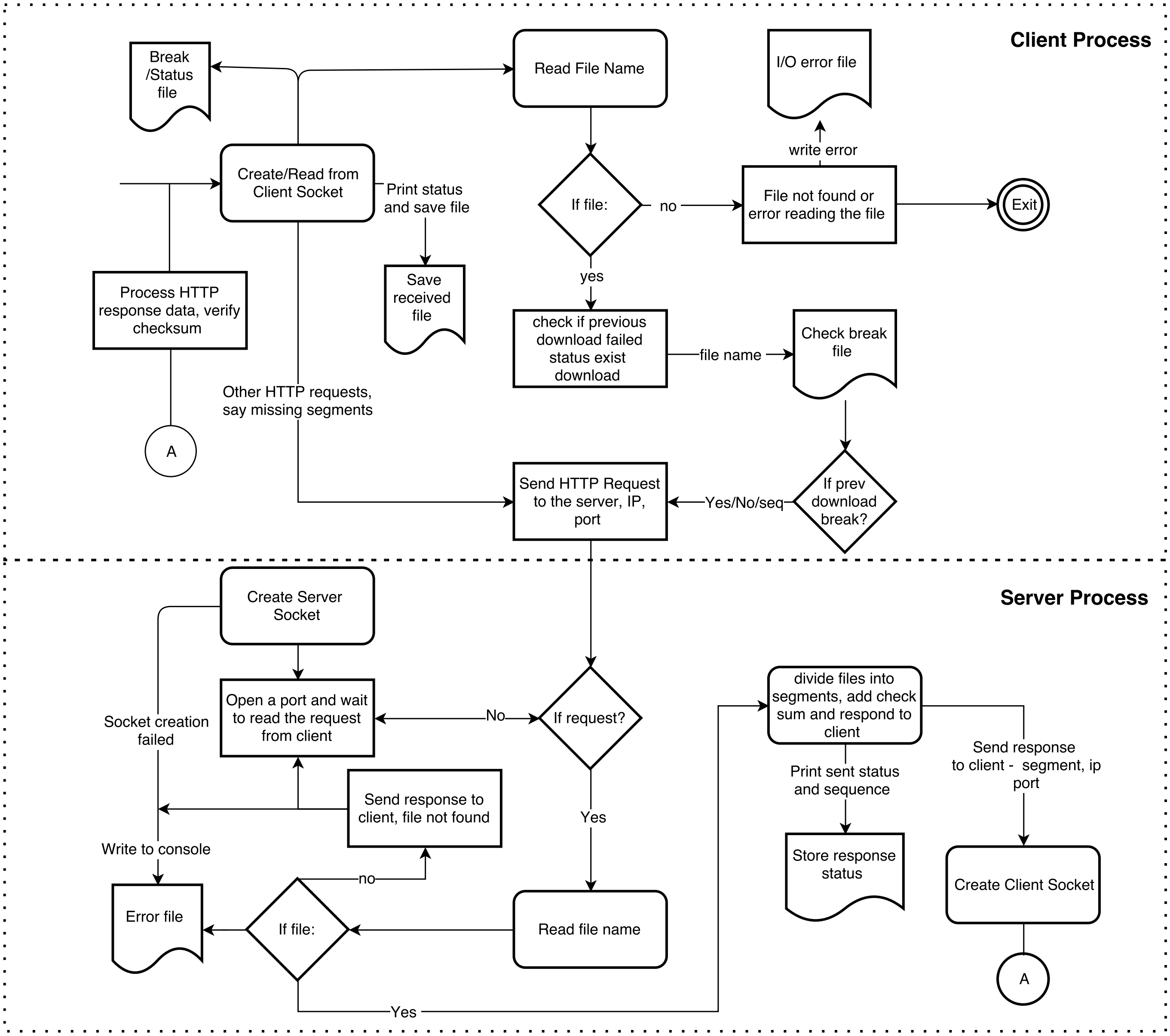

UDP FIle Download System Architecture
Received file
Save received file
file
UDP Client
UDP Server
Read file name
HTTP request / file download
Find file
file
Break file into segment
Segments
Add UDP checksum and header
file not found error
HTTP response / file segment
Save interrupt/break status to file for future reference
seg recv/src date time
break i/o file
print info, request received and sent to the console
Save temp response status and error message to file
resp/error
err files
Save temp response status and error message to file
resp/error
err files
Print request received and sent byte info message to the console

Figure 3.2: UDP file download system architecture

3.3 Flow Diagram

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


Client Process
Break /Status file
Read File Name
I/O error file
write error
Create/Read from Client Socket
Print status and save file
If file:
no
File not found or error reading the file
Exit
Process HTTP response data, verify checksum
Save received file
yes
check if previous download failed status exist download
file name
Check break file
Other HTTP requests, say missing segments
A
Send HTTP Request to the server, IP, port
Yes/No/seq
If prev download break?
Server Process
Create Server Socket
Open a port and wait to read the request from client
No
If request?
divide files into segments, add check sum and respond to client
Socket creation failed
Print sent status and sequence
Send response to client - segment, ip port
Send response to client, file not found
Yes
Write to console
Store response status
Create Client Socket
no
Error file
If file:
Read file name
A
Yes

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.

3.4 State Diagram

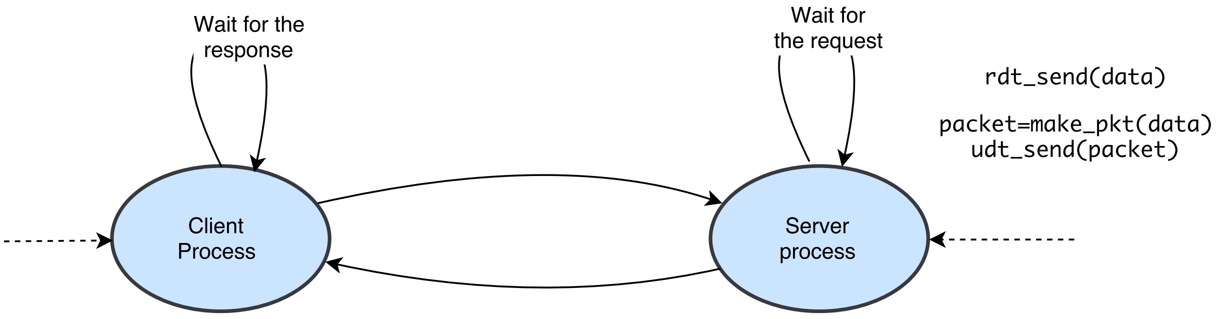


Fig 3.3: State diagram, client server - request response process

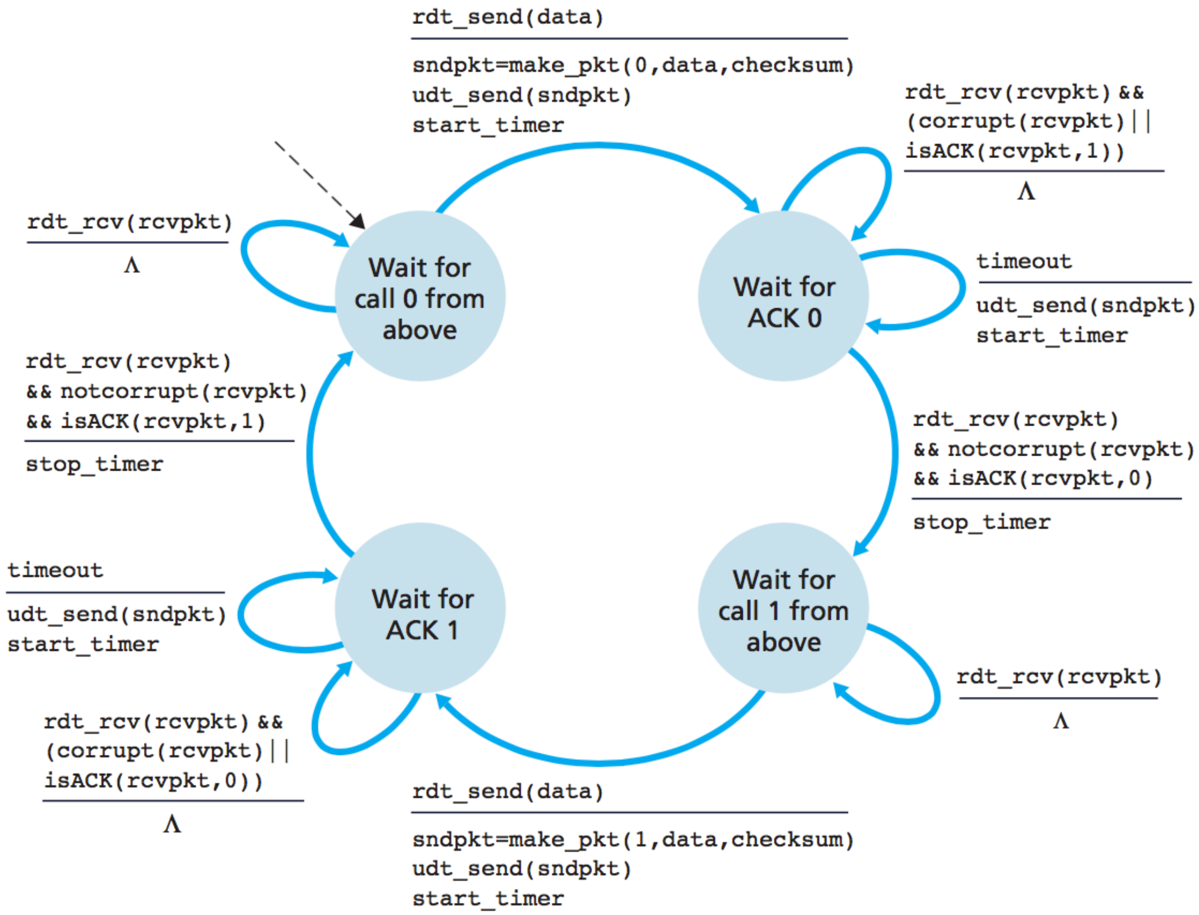


Fig 3.4: State Diagram - Alternate bit protocol

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

4. Implementation

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

5. 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.