National University of Sciences & Technology
School of Electrical Engineering and Computer Science
Department of Computing
EE353: Computer Networks, BESE-9A Spring 2020

Project Specifications (V1)	
CLO 4: Design and implement solutions to contemporary networking issues (through hands on programming)	
Maximum Marks: 20	Instructor: Dr. Arsalan Ahmad
Date: 27th April 2020	Due Date: 17th May 2020

1 General

This is a group project. Maximum size of the group is restricted to two members.
You are advised to avoid plagiarism; any such case would result in award of zero marks both to the "sharer"and the "acquirer".

2 Learning Objectives

In this project, your task is to build a system which downloads a large size file from multiple servers simultaneously. Each server should have a separate copy of the file and should operate independently. The file on each server should be fragmented and the client should download the fragments of file from different servers simultaneously based on availability of the server for service and then combine them on the client side. On completion of this project, you will gain sufficient expertise in the following skills:

- Socket programming in python
- File segmentation and recombination
- Server setup and handling the server failure
- Download resuming

3 Functionalities

3.1 Servers

You have to create virtual servers with a copy of a video file on each server. The file should be segmented on each server.

Your server program file should be named 'server.py'. The command line syntax for the server is given below:

server.py -i <status_interval> -n <num_servers> -f <file_location> -p <list_of_server_port_numbers>

-i	(Required)	Time interval in seconds between server status reporting
-n	(Required)	Total number of virtual servers
-f	(Required)	Address pointing to the file location
-p	(Required)	List of port numbers('n' port numbers, one for each server)

3.2 Client

Your client should be named client.py. The command line syntax for the client is given below:

Client.py -r -i <metric_interval> -o <output_location> -a <server_ip_address> -p <list_of_server_port_numbers>

-i	(Required)	Time interval in seconds between metric reporting (section 3.4)
-o	(Required)	path of output directory
-a	(Required)	IP address of server
-p	(Required)	List of port numbers (one for each server)
-r	(optional)	Whether to resume the existing download in progress

Once the client is executed, it should actively seek all the servers available at the given port numbers, upon unsuccessful communication to any of the the server should follow the Server Failure Handling explained in section 3.3.

When the client finishes downloading the file segments, it should combine all the segments of the file and place the file in the output directory.

3.3 Handling Server Failure and Load Balancing

The program should have basic server failure tolerance i.e upon any server crashing or simply closing any server, the client shouldn't lose that chunk of data and rather send request to the currently live servers and divide the load among them in a way that ensures load balancing and still successfully obtain the video file at the client side.

3.4 Output and metric reporting

3.4.1 Client

The client should print an output that indicates the download status according to the interval specified to -i flag. Suppose the file is being downloaded from 4 simultaneous servers, the output in this case should be according to the follow format:

Server 1: <downloaded_bytes>/<total_bytes>, download speed: <speed>kb/s
Server 2: <downloaded_bytes>/<total_bytes>, download speed: <speed>kb/s
Server 3: <downloaded_bytes>/<total_bytes>, download speed: <speed>kb/s
Server 4: <downloaded_bytes>/<total_bytes>, download speed: <speed>kb/s
Total: <downloaded_bytes>/<total_bytes>, download speed: <speed>kb/s

If the interval specified to -i flag is 2, then this output should be printed every 2 seconds. You should report the download speed from each server as specified.

3.4.2 Server

The server should print an output that indicates the port number, options to shutdown each server and the status of each server according to the interval specified to -i flag. Suppose there are four servers, the output in this case should be according to the following format:
Server 1: Port: <port_no> Status: <dead/alive>, To Shutdown Server 1 Enter: E1
Server 2: Port: <port_no> Status: <dead/alive>, To Shutdown Server 2 Enter: E2
Server 3: Port: <port_no> Status: <dead/alive>, To Shutdown Server 3 Enter: E3
Server 4: Port: <port_no> Status: <dead/alive>, To Shutdown Server 4 Enter: E4

3.5 Download Resuming

In a scenario where we lose all servers or connection to client is simply lost, upon re-establishing connection the download should resume from the progress where the connection was lost between client and server.

4 Specifications

Following are some of the points you need to keep in mind before working on your project:

- You are required to use the Python programming language while working on this project.
- You cannot use any high level abstract libraries to achieve the full functionality via simple use, you need to create your own program for the core functionalities of this project.
- You can use sockets library for basic TCP functionality.
- You need to make sure the client makes use of all the servers available
- The Request mechanism should not be static in nature i.e the client should make use of the available servers and apply load balancing
- The file used for transferring should specifically be a video file and the minimum size should be at least 5Mb and at most 10Mb.

- The servers should **not** be aware of other servers and **not** communicate with them in any capacity.
- For demo client and server both should be on separate machines also bring the client code in a flash drive.

5 Grading Criteria

You will be graded based on the following grading criteria:

Criteria	Weightage (%)
Downloading from single server using TCP	15
Downloading from multiple servers using TCP + File Segmentation + File Recombination	40
Handling Server Failure and Load Balancing	15
Download Resuming	15
Correct input, output and metric reporting	10
Code is well documented, clean and readable	5

Late penalty (on your received marks) will be applied as follows:

- 1 day after deadline: 25% reduction
- 2 days after deadline: 50% reduction
- 3 days after deadline: Not accepted

6 Submission and report

You are required to submit your source code and a short report to an upload link that would be made available on the LMS. Zip your source code files and your report document in a file named Project_Surname_Surname (Surname of both Group members). Nominate one of the group members to submit on behalf of the group. Please note that you must upload your submission BEFORE the deadline. The LMS would continue accepting submissions after the due date. Late submissions would carry penalty per day with maximum of 2 days late submission allowed (see section 5). Students who fail to submit would not be allowed to appear in viva and those who miss the viva would not be allowed to retake the viva.

Your submitted code would be checked for similarities and any instances of plagiarism would result in award of ZERO marks for all such students, irrespective of who has shared with whom.

You must write down group members' Registration No‟s and Names at the beginning of the report. All reports will be read for marking purposes.

The size of your report MUST be under 2 pages. Your report should briefly document your techniques and methodology used for implementation and how you combat the relevant problems in development. Treat it as a summary document only (point form is acceptable). You will be asked to demonstrate your program during your viva.