UNIVERSITY OF DHAKA

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

LAB Viva

Submitted By:

Name : Sudipto Das Sukanto

Roll No : 29

Name: Ahanaf Ahmed Shawn

Roll No : 47

Submitted To :

Dr. Md. Abdur Razzaque

Dr. Md. Mamun Or Rashid

Dr. Muhammad Ibrahim

Mr. Md. Redwan Ahmed Rizvee

1 LAB3:

1.1 Task 1: File Transfer via Socket Programming

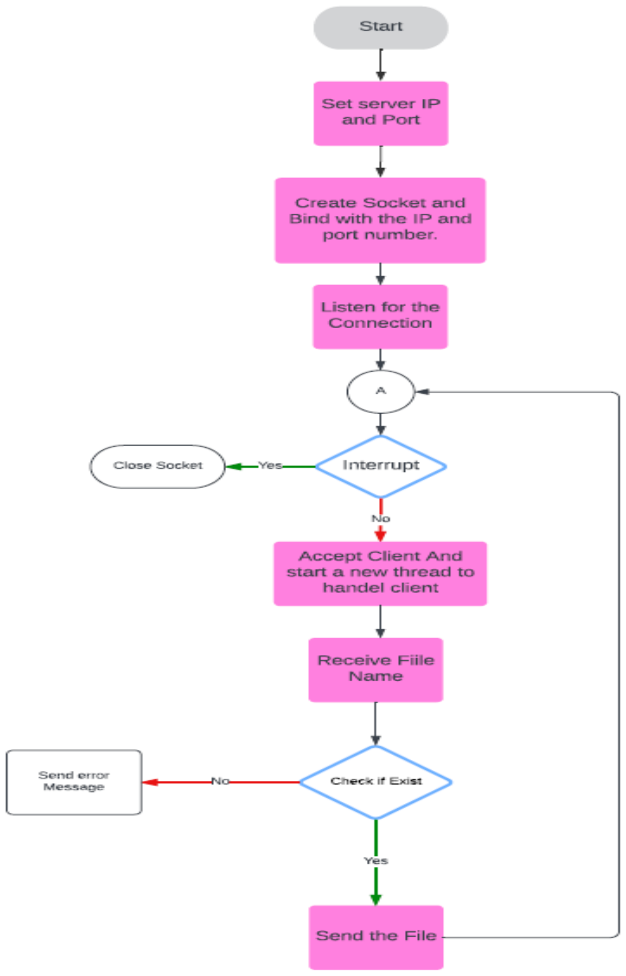
Start
Set server IP and Port
Create Socket and Bind with the IP and port number.
Listen for the Connection
A
Close Socket
Yes
Interrupt
No
Accept Client And start a new thread to handel client
Receive Fiile Name
Send error Message
No
Check if Exist
Yes
Send the File

Figure 1: Server FlowChart

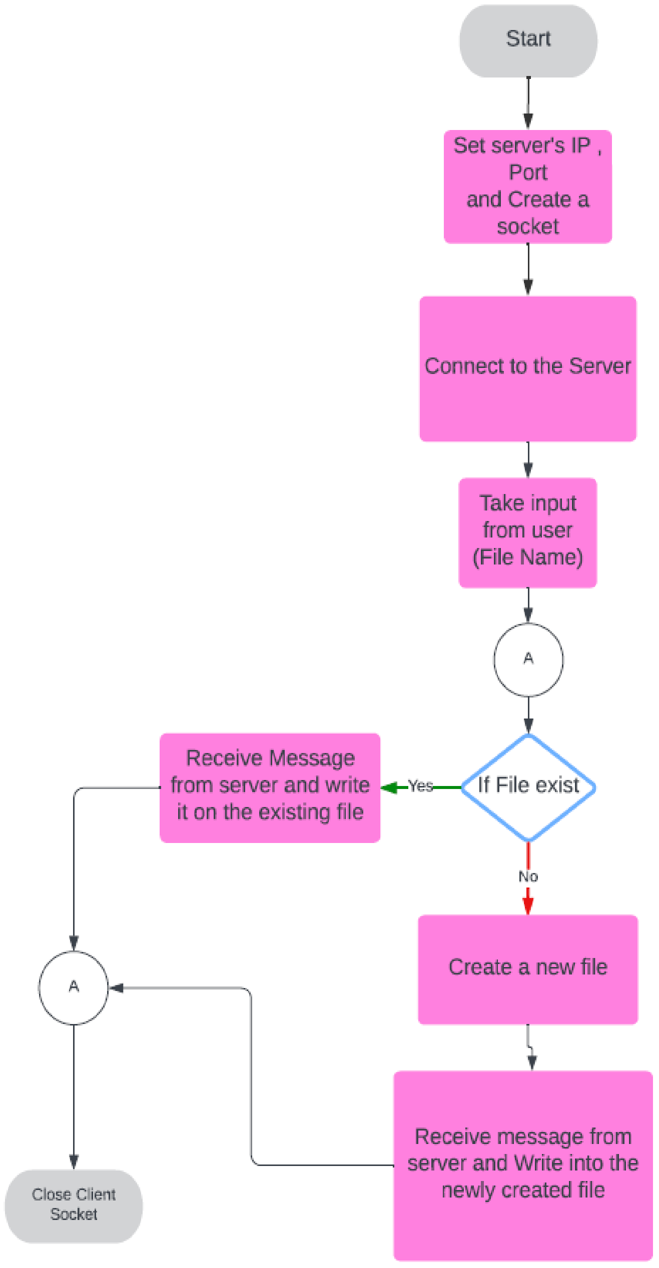
Start
Set server's IP , Port and Create a socket
Connect to the Server
Take input from user (File Name)
A
If File exist
Yes
Receive Message from server and write it on the existing file
No
Create a new file
Receive message from server and Write into the newly created file
A
Close Client Socket

Figure 2: Client FlowChart

1.2 Task2: File transfer via HTTP

Algorithm for Client:

1. Import the `requests` library.
2. Define a function `download_file` that takes a URL and a local filename as input parameters.
3. Send a GET request to the specified URL with streaming enabled.
4. Check if the response status code is 200 (indicating success).
5. If the status code is 200, open the local file in binary write mode.
6. Iterate over the response content in chunks (each of size 1024 bytes) and write them to the local file.
7. Print a success message if the file is downloaded successfully.
8. If the response status code is not 200, print an error message indicating the status code and the response text.
9. Define a function `upload_file` that takes a URL and a local filename as input parameters.
10. Open the local file in binary read mode.
11. Prepare a dictionary `files` containing the file content with the key 'file' and the filename as the value.
12. Send a POST request to the specified URL with the file content as part of the request payload.
13. Check if the response status code is 200 (indicating success).
14. If the status code is 200, print a success message indicating the file upload was successful.
15. If the response status code is not 200, print an error message indicating the status code and the response text.
16. In the main block, define the server URL, download path, and upload path.
17. Call the `download_file` function with the appropriate arguments to download a file from the server.

18. Call the `upload_file` function with the appropriate arguments to upload a file to the server.

Algorithm for Server:

1. Import the required modules: `BaseHTTPRequestHandler` and `HTTPServer` from `http.server`.

2. Define a custom request handler class named `get_post_handler`, which subclasses `BaseHTTPRequestHandler`.

3. Define the `do_GET` method within the `get_post_handler` class to handle GET requests.

   (a) Extract the file path from the request URL.

   (b) Attempt to open the requested file in binary read mode.

   (c) If the file is found, send a response with status code 200 (OK).

   (d) Set the response header to indicate the content type as `application/octet-stream`.

   (e) End the response headers.

   (f) Write the contents of the file to the response body.

   (g) If the file is not found, send a 404 (File Not Found) error response.

4. Define the `do_POST` method within the `get_post_handler` class to handle POST requests.

   (a) Extract the content length from the request headers.

   (b) Read the file content from the request body based on the content length.

   (c) Extract the file name from the request URL.

   (d) Write the received file content to the specified file in binary write mode.

   (e) Send a response with status code 200 (OK) to indicate successful file upload.

   (f) Set the response header to indicate the content type as `text/plain`.

   (g) End the response headers.

   (h) Write a success message to the response body.

5. Define the `start_server` function to start the HTTP server on the specified port.

   (a) Create an instance of `HTTPServer`, passing it an empty string (indicating to listen on all available interfaces) and the specified port.
   (b) Print a message indicating that the server is listening on the specified port.
   (c) Start serving requests indefinitely.

6. In the main block:
   (a) Define the port number for the server.
   (b) Call the `start_server` function with the specified port number.

2 LAB4:

2.1 Task 1: Iterative DNS server

1. **Client sends DNS query to Root Server:**
   (a) The client sends a DNS query for the domain to the root server.
2. **Root Server Response:**
   (a) The root server responds with a referral to the TLD server responsible for the domain's top-level domain.
3. **Client sends DNS query to TLD Server:**
   (a) The client sends the same DNS query to the TLD server received from the root server.
4. **TLD Server Response:**
   (a) The TLD server responds with a referral to the authoritative DNS server for the domain.
5. **Client sends DNS query to Authoritative Server:**
   (a) The client sends the same DNS query to the authoritative DNS server received from the TLD server.
6. **Authoritative Server Response:**
   (a) The authoritative DNS server responds with the IP address for the domain.

7. **IP Address Returned to Client:**
   (a) The client receives the IP address for the domain from the authoritative DNS server.
8. **Resolution Complete:**
   (a) The DNS resolution process is complete, and the client can use the obtained IP address to communicate with the desired domain.

2.2 Task 2: Recursive DNS server

1. **Define a function to send a recursive DNS query to a recursive DNS resolver:**
   (a) Create a DNS query packet specifying the domain name and type of query.
   (b) Send the query packet to the recursive DNS resolver's IP address.
   (c) Receive the response from the recursive DNS resolver.
2. **The recursive DNS resolver:**
   (a) Receives the DNS query from the client.
   (b) Sends a query to the root DNS server for the domain.
   (c) Receives the response from the root DNS server.
   (d) If the response contains a referral to a TLD DNS server:
      i. Sends a query to the TLD DNS server.
      ii. Receives the response from the TLD DNS server.
      iii. If the response contains a referral to an authoritative DNS server:
         A. Sends a query to the authoritative DNS server.
         B. Receives the response from the authoritative DNS server.
         C. If the response contains the IP address for the domain:
            - Returns the IP address to the client.
   (e) If the response from the root DNS server contains the IP address for the domain:
      - Returns the IP address to the client.
3. **The recursive DNS resolver:**

  - Sends the response containing the IP address for the domain to the client.

4. **The client receives the IP address for the domain from the recursive DNS resolver.**

5. **Verification:**

    - Verify that the client receives the correct IP address for the domain from the recursive DNS resolver.