CE 156

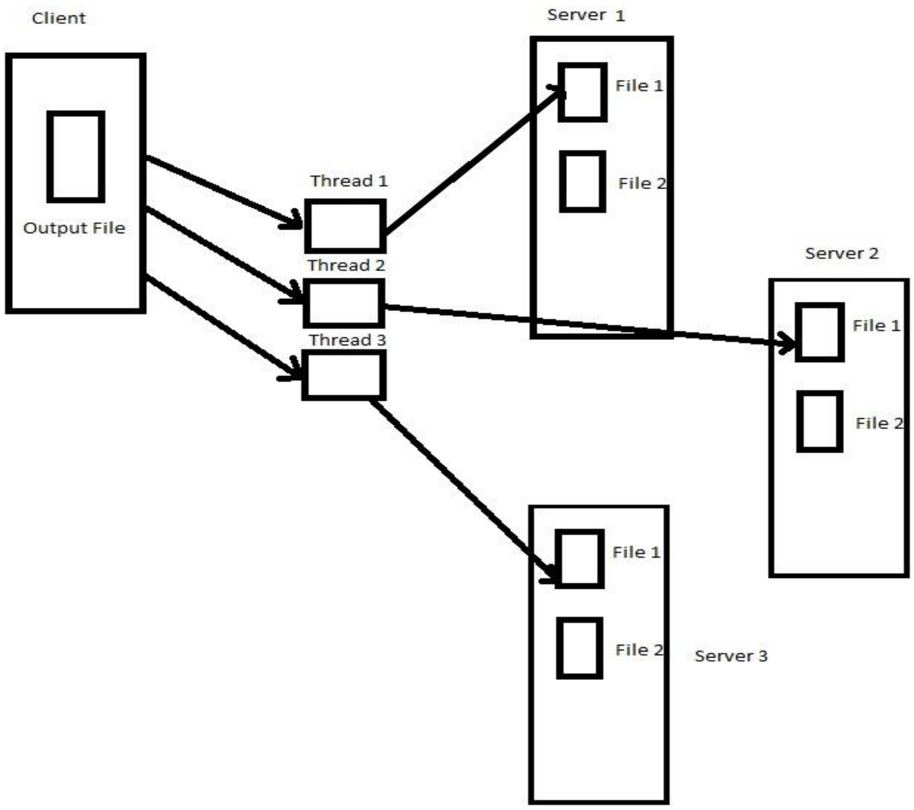
Network Programming

Assignment 3

Avirudh Kaushik

Documentation:

The architecture of the application is a basic client server architecture in which a client requests the server for a request and the server provides the client the required data. In our Assignment the client is required to get the data from a file on from different servers using threads and write the data to an output file. The transfer of data has to be done concurrently. For this Assignment I have used the pthread API. This is very similar to the previous assignment but the only difference is that we have used the User Datagram Protocol (UDP) instead of TCP for sending messages and transferring files.



The Diagram shows a representation of how the program works and how the threads access the file data.

THE WORKING OF THE APPLICATION:

1: The servers start listening for the request.

2: The client gets the text file having the Address details of the servers and the number of servers to be used by the user via command line.

3: The client parses through the text file and stores the IP addresses and the port numbers of the servers in arrays.

4: Then the client requests one of the server (1st one by default) for the size of a file.

5: The server receives the name of the file and returns its size if it exists otherwise it returns a string saying No such file exists.

6: If the server returns the size the client then spawns threads which get their own chunk sizes and the offsets.

7: The number of threads are equal to the number of servers specified by the user or the max number of servers available.

8: The threads access different servers concurrently and retrieve the chunk they are allotted form the file on the server.

9: After receiving the data back on the thread, it writes the data to the output file. The writing is also done concurrently.

10: Finally, the client waits for the threads to finish and once they finish the client program closes.

11: All the functionality is the same as previous assignment. A lot of features have not been implemented in this assignment including the server concurrency, server robustness etc.

DESIGN:

The client has a mechanism of notifying the server whether it is asking for File Size or asking for data.

By sending the “FILE-CHECK” token the client tells the server that it is currently asking whether the file exists or not and if it does then what is the size of the file.

If the client sends the token “GET-DATA” then the client is requesting the server for bytes of data from the file.

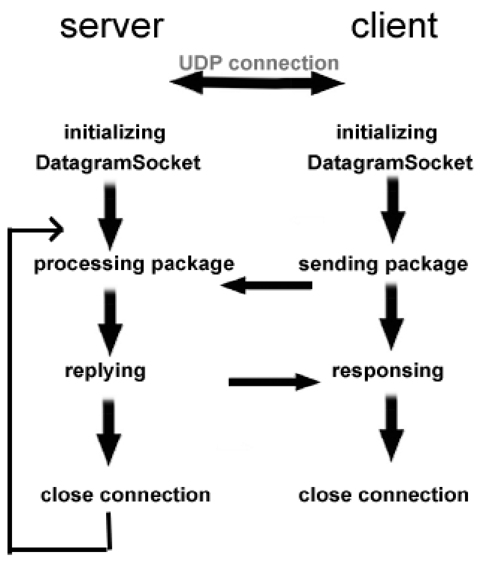
So the first message sent from the client to the server always have a HEADER with a token containing either “FILE-CHECK” or “GET-DATA”. The “FILE-CHECK” token is usually sent by the main client and the “GET-DATA” token is usually sent by the threads.

In my program there are constant messages exchanged between the client and the servers. Most of the messages sent using the Write and Read calls are followed by an ACK. This was done to ensure the client or the server does not write asynchronously.

The major difference in this assignments design is that I have used a sendto() and recvfrom() system calls to transfer messages and data between the client and the server. The client starts the sending process by specifying the “FILE-CHECK” token which is followed by an ACK from the server. This is followed by the File name of the file to be retrieved. The server checks for the file in the directory and responds with the file size if the file exists or with “NO-FILE” token to tell the file does not exist. In this assignment almost every communication between the client/client thread and server is followed by an ACK. This made the application more reliable and made sure the data is reaching at the right time to the right socket.

Another change in this assignment is the use of select() system call in conjugation with the recvfrom() system call. The select() call enabled to us to keep track of our socket descriptor and to check whether an input is still available on the socket. This call also enabled us to set a timer on the recvfrom() call i.e. helped to us to come out of the blocking recvfrom() call when no input was being received on the socket. I am very confident that if the retransmission feature was added in the application then the select command would be used along with a sequence numbers. The program sticks to assignment specifications but has not been fully equipped with all the features required.

The following figure shows how the UDP datagrams have been transmitted across server and client.



ERROR CHECKING:

Both the client and server applications check for errors whenever there is a possibility for an error. The following cases of errors are checked in the application.

1: The IP addresses and the port number not specified error for client.

2: The Port number not being specified for the server.

3: If the number of servers mentioned by the user greater than the servers available.

4: Error from creation of a socket in both client and server.

5: Error in Binding the socket to the network address in server.

6: The client application crashing

7: Error in reading the data

8: Error in opening the file

9: Error in reading from the file pointer

10: Checking for a client disconnection

11: Error in converting the IP address provided as a String into a Network order address on the client

12: Error in establishing a connection between client and server

13: Error in writing to the send buffer in client.

14: Error in creating a Thread

15: Error in seeking a particular a place in a file using fseek()

16: Error when Threads join to the main thread

17: Error If the number of servers are not specified on the client

18: Error if the number of servers specified are greater than the number of servers available.

19: Error in using the select() system call

20: Checking for timeouts from the recvfrom() call

21: Error in the recvfrom() call

22: Error in the sendto() system call

23: Error in conversion of IP address from string to network format

HOW TO USE THE APPLICATION:

The application will run perfectly by following these steps:

1: Unzip the tar.gz file.

2: Go in the ex3 folder

3: Run the make command when in the ex3 directory.

4: Then in 4 terminals go to the bin directory.

5: In 1-3 terminals type: ./myserver.c <portnumber>

The port number is the port on which you intend to run the server

6: On the other terminal type: ./myclient.c <serve-info.txt> <number of servers>

7: Enter the name of the File you wish to transfer