assignment 2

Abdelaziz Mohamed Abdelaziz - 19015941

Omar Khairat Mohamed Abodeif - 19016063

O

**Assignment Statement**

Since you’ve learned about the socket interface and how it is used by an application; by now, you’re pretty much an expert in how to use the socket interface over a reliable transport layer, so now seems like a good time to implement your own socket layer and reliable transport layer! You’ll get to learn how the socket interface is implemented by the kernel and how a reliable transport protocol like TCP runs on top of an unreliable delivery mechanism (which is the real world, since in real world networks nothing is reliable). This lab should be fun since your implementation will differ very little from what would be required in a real-world situation.

The network communication in last assignment was provided through a reliable transfer protocol (TCP/IP). In this assignment, you are required to implement a reliable transfer service on top of the UDP/IP protocol. In other words, you need to implement a service that guarantees the arrival of datagrams in the correct order on top of the UDP/IP protocol, along with congestion control.

**Specifications**

Suppose you’ve a file and you want to send this file from one side to the other (server to client). You will need to split the file into chunks of data of fixed length and add the data of one chunk to a UDP datagram packet in the data field of the packet. You need to implement TCP with congestion control.

**Data Structures Used**

* Array of characters (string) to save our request information.
* Vector of Strings to store our requests independent of each other.
* Struct socket\_address to utilize our WebSocket.
* struct arg\_struct { int client\_socket; long long\* timer; }.

**User Guide**

* Run the server from terminal after changing the directory to its directory and type the following in terminal:
* g++ -o HTTP-Server main.cpp
* ./HTTP-Server
* Run the client from terminal after changing the directory to its directory and type the following in terminal:
* g++ -o HTTP-Client main.cpp
* ./HTTP-Client

**Packet type and fields (Server) :**

* There are two kinds of packets:

Data packets :

Text

Description automatically generated

Ack packets :

Text

Description automatically generated

**Main functions Server:**

readFileData() : this function is used to read content of the file.

Text

Description automatically generated

getAckChecksum() = > this function is used to get Ack Checksum.

A screenshot of a computer

Description automatically generated

gerDataChecksum ()=> this function is used to get Data of Checksum.

A screenshot of a computer

Description automatically generated

createPacket() => in this function is used to create packet that will be send to be client

Text

Description automatically generated

corruptDatagram ()=> this function is used to check if datagram is corrupted or not

Text

Description automatically generated

readInfo() => this function is used to readInfo of command file

Text

Description automatically generated

checkfileExistence() => check if the file is exist or not

A screenshot of a computer

Description automatically generated with medium confidence

main()

Text

Description automatically generated

Text

Description automatically generated

send\_ack\_file\_name(client,fname,numberofPackets,client\_addr)=> it is used to send ack of file.

Text

Description automatically generated

handle\_client\_request () => it is used to handle client request .

A screenshot of a computer screen

Description automatically generated with medium confidence

handle\_time\_out ()

Text

Description automatically generated

handle\_check\_sum ()

Text

Description automatically generated

Retransmit\_loss\_packet ()

Text

Description automatically generated

sendTheData\_HandleCongestion ()

Text

Description automatically generated

Text

Description automatically generated

Text

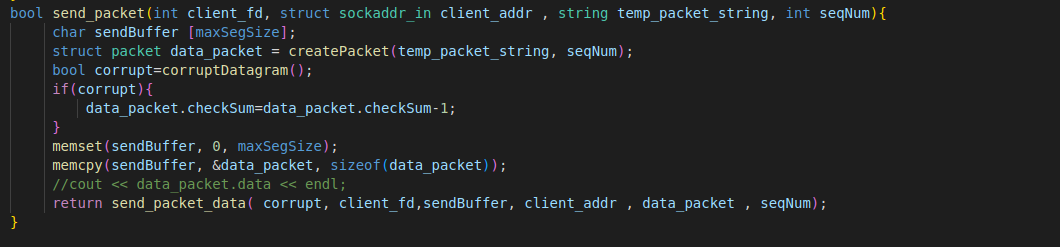
Description automatically generated

send\_packet\_data()

Text

Description automatically generated

send\_packet ()



**Design Decisions Server :**

* Multiple users the server forks off a child process to handle the client.
* The server (child) creates a UDP socket to handle file transfer to the client.
* Handling client requests as follows :

-The client sends the name of file that he wants the server to send to him.

- when the server receives the file name checks if file is available if not it appear error message

- if file is available the server send the number of packet need to client and transmission will be as following:

Diagram

Description automatically generated

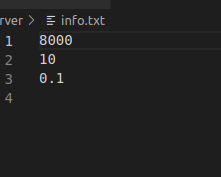
* To set the Arguments of the server we make commands file (info.txt)

Where we put the following arguments :

Well - known port number .

Random generator seed value .

Probability p of datagram loss (real number in the range [0.0,1.0]).



* Simulating corrupted packets we subtract one from the check sum of the packet so that client knows that the packet is corrupted.

In first one corruption packet as lost val > 5.9

In second one Corruption packet as lost <5.9

Text

Description automatically generated

* Simulating lost packets the server does not send them so , no ack for them from the client will be received

In first one corruption packet as lost val < 5.9

In second one Corruption packet as lost >5.9

Text

Description automatically generated

**For the Congestion Window:**

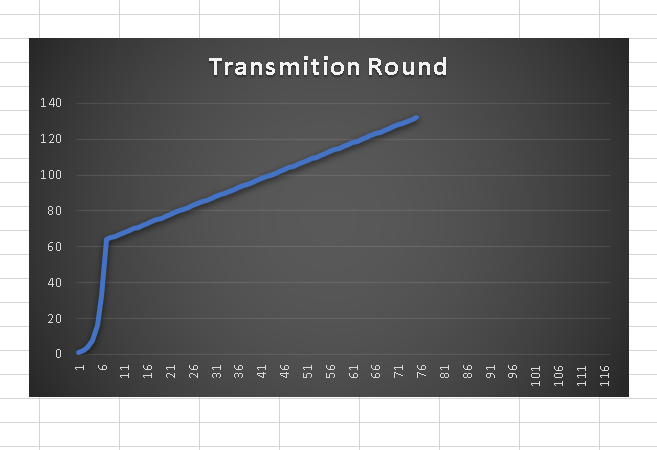
* The CWND increasing exponentially in the stage of slow start .
* When CWND value is more than the value of ssthresh which we set initially 128 , it starts to increase linearly in the stage of congestion window avoidance.
* When there is packet lost the CWND is drop to equal 1 and start increasing again.
* When there is triple duplicate Ack the CWND size drop to half of its size and the ssthresh will equal the half of CWND just before losing the packet.

**Network System analysis:**

|  |  |  |
| --- | --- | --- |
| Points of Comparison | Stop -AND-WAIT | Selective Repeat |
| Basic | Retransmits all the frames after frame suspect to be lost | Retransmits only frames that are suspect to be lost |
| Complexity | Less complicated | More Complex |
| Window Size | N-1 | <= (N+1)/2 |
| Sorting | Sorting is neither need in sender or receiver | Receiver must be able sort . |
| Storing | Receiver do not stored frames after damaged until the frame is retransmit | Receiver stores the frames received after damaged frame is replaced. |
| Searching | No searching | Sender must be able to search. |
| Bandwidth  Utilization | If error rate is high, it waste bandwidth | Less bandwidth is wasted  In retransmission |

**Congestion Window Analysis:**

Probability is 0.01:

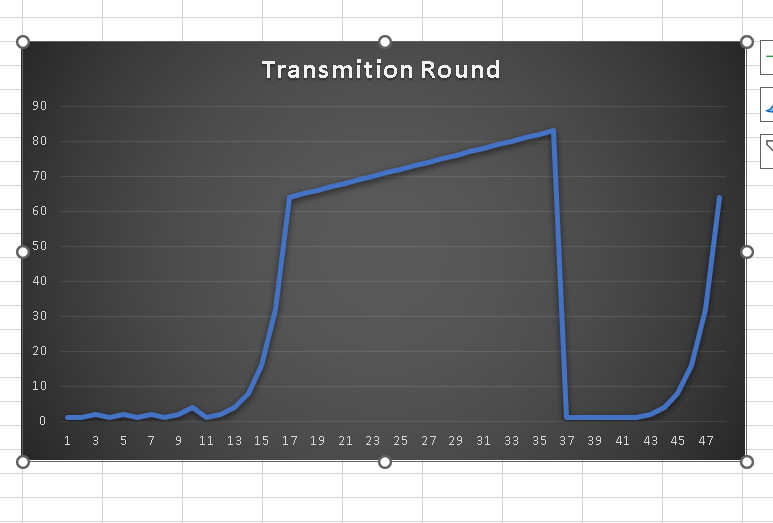


Probability is 0.05:

Graphical user interface, application

Description automatically generated

Probability is 0.1:



Probability is 0.3:

**A screenshot of a computer

Description automatically generated with medium confidence**

**Packet type and fields (Client) :**

* There are two kinds of packets:

Data packets :

Text

Description automatically generated

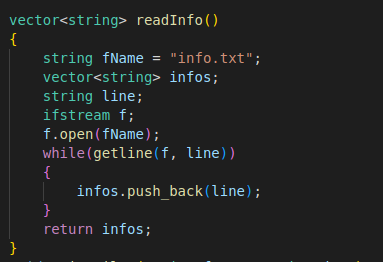
Ack packets:

Text

Description automatically generated

**Main functions Client:**

readInfo ()

****

writefile ()

**Text

Description automatically generated**

getAckChecksum ()

A screenshot of a computer

Description automatically generated with medium confidence

getDataChecksum ()

A screenshot of a computer

Description automatically generated

createPacket()

Text

Description automatically generated

main ()

Text

Description automatically generated

Text

Description automatically generated

**Design Decisions Client :**

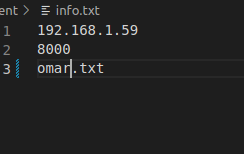
* To set the Arguments of the server we make commands file (info.txt)

Where we put the following arguments :

IP address of server .

Well known port number in server .

Filename to be transferred (large file).

****

**Sample Runs :**

(Client side)

**Text

Description automatically generated**

**Text

Description automatically generated**

(Server Side)

Text

Description automatically generated

Text

Description automatically generated with medium confidence