Project

Foundations of Computer Networks

Srinath Kanna Dhandapani (sd2689)

Protocol Messages

Data Packet:-

Client IP	Server IP
Client Port	Server Port
Sequence Number	Checksum
Number of Packets	File Size
Data (500 bytes)	

Every packet contains Client IP, Client Port, Server IP, Server Port, Number of packet in total and file size.

And other details relevant to each packet like sequence number, checksum and data. The checksum is calculated from the data field alone. Each packet is approximately 800 bytes.

Acknowledgement Packet:-

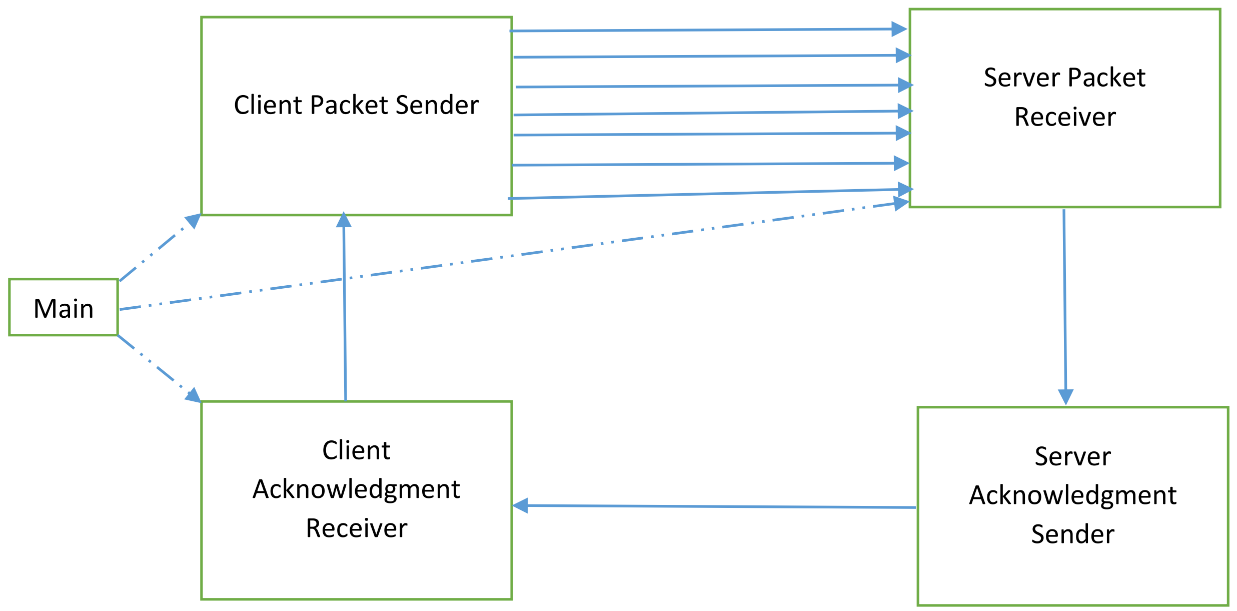
Next Required Sequence Number
End Bit
Optional data

Every Acknowledgment packet contains the next packets sequence number required and if all the packets are transferred then the end bit is set and transferred.

Flow Diagram:-

Three threads are started by the main class. The communication between the client and the server are maintained by the UDP connection. Packet travels from Client Packet Sender to server packet sender.

The received packets are handled by the Server Ack Sender and then transmitted to Client Ack Receiver.



All there four threads communicated interconnected by sharing common variables and transfer of messages which in turn produce the reliable transfer protocol.

Implementation:-

Handshake

With the transmission of any packet that is being the first to reach the server does the handshake on its own by initializing all the essential variables and sockets for the transmission between the client and the server. This is even more efficient than the three way handshake as every packet in the designed packet contains every data required in the process of handshake.

Slow start

The communication starts from the client to server with the initial window to be 1 and the first packet being transmitted from client to server. Once when this file is reached the server handles this packet by storing it in an array. For all the packets that is missing from this point to the previously handled window is being checked and then an acknowledgement denoting the first missing packet in the array in increasing order is then transmitted to the client receiver.

Once the acknowledgment is received then the client changes its list of last three previously received Acknowledgments. Then if the Acknowledgment denotes the successful transmission of all the elements in the window then the window size is doubled and the transmission of packet starts by sliding the window forward completely. The window size is double even in the case when few packets are lost but the window does not slide. This mechanism is being set as the transmission loss could even be because of noise in the channel. This happens till the window size reaches the threshold of 128. After that the window size is incremented along with the threshold linearly by 1 for each successful transmission.

3- Acknowledgment Scenario:-

If same acknowledgment is received thrice then the window size is decreased to 1, along with reducing the threshold to half and then the first packet in the previous window is transmitted.

Timeout Scenario:-

If the acknowledgements don't arrive before the timeout set at the compile time then the window size is set as 1, along with reducing the threshold to half and the first element of the previous window is sent to the server. And if this packets acknowledgement reaches on time then the window size is doubled and the transmission picks up the pace and regains the transmission speed.

This on the whole gives the feature of slow start, fast retransmission and flow control.

Goodbye Handshake:-

Finally after all the packets in the file are received in the server side then an acknowledgement in the form of a string containing the word "done" is sent after which the server terminates. The packets of the file that is collected from client is then written in a file sequentially and the details of Hash Code and file size is printed.

After receiving the "done" acknowledgment from the server the client terminates printing the details of the file that is sent.

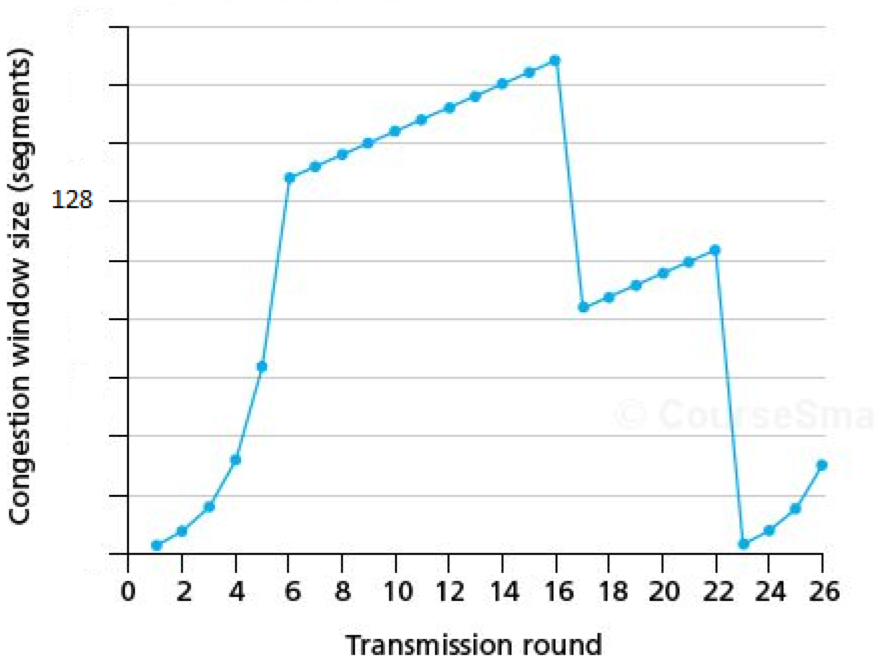


Fig 1.1 Tahoe window size illustration (www.eng.utah.edu)

Helper modules:-

Quiet

A method to handle the print during the application execution. When quiet is initialized to 1 through the command line argument then the print statements in the application doesn’t get executed.

Packet Maker

Reads the input file and converts it into an array of packets that contains all the data sequentially. Each packet gets assigned with its own sequence numbers, checksums and data.

File Writer

A module to write all the packets read from the client to be written correctly and sequentially so that the Hash Code of the files remains the same.

File Hash Code Printer

To read the file from corresponding server and client sides and print its details like file size and Hash Code to verify the similarity among the file read and file written.

Sample Test Cases and Outputs:-

TestCase 1:

-s 5000

-c –f resume.txt localhost 5000

Client Console:-

```
Creating packet : 1
Creating packet : 2
Creating packet : 3
Creating packet : 4
Creating packet : 5
Creating packet : 6
Creating packet : 7
Creating packet : 8
Creating packet : 9
Client Started:
Packet generation in progress:
Client sending packet : 1
Doubling the window size: 2
Ack received as : 1
Client sending packet : 2
Client sending packet : 3
Doubling the window size: 2
Ack received as : 3
Client sending packet : 4
Client sending packet : 5
Doubling the window size: 4
```

```
Ack received as : 5
Client sending packet : 6
Client sending packet : 7
Client sending packet : 8
Client sending packet : 9
Doubling the window size: 4
Ack received as : 9
No ack from server: Timeout: Setting window size as 1
Client sending packet : 9
Client transferred the file:
Client File details
Size of file is: 4447
Hashcode for file is: 9d31a285
```

Server Console:-

```
Server started:
Server accepting packet : 0
Server sending ack for : 1
Server accepting packet : 1
Server accepting packet : 2
Server sending ack for : 3
Server accepting packet : 3
Server accepting packet : 4
Server sending ack for : 5
Server accepting packet : 5
Server accepting packet : 6
Server accepting packet : 7
Server accepting packet : 8
Server sending ack for : 9
Server accepting packet : 8
Sending file received acknowledgement: done
All packets received now writing in file
Writing packet in file : 1
Writing packet in file : 2
Writing packet in file : 3
Writing packet in file : 4
Writing packet in file : 5
Writing packet in file : 6
Writing packet in file : 7
Writing packet in file : 8
Writing packet in file : 9
Server File details
Size of file is: 4447
Server sending ack for : 9
Hashcode for file is: 9d31a285
```

TestCase 2: (With timeout and quiet)

-s –q 500

-s –q –t 800 –f resume.txt localhost 5000

Client Console:-

```
Packet generation in progress:
Client File details
Size of file is: 4447
Hashcode for file is: 9d31a285
```

Server Console:-

```
Server File details
Size of file is: 4447
Hashcode for file is: 9d31a285
```