Socket Programming in python

Go Back-N

Ahmed El-Halawany 2689

Farida Adham 2616

Mariam Soffar 3066

Veronica Gawish 2620

Introduction:

The sender continues to send a number of [packets](https://en.wikipedia.org/wiki/Data_frame) to receiver specified by a window size even without receiving an [acknowledgement](https://en.wikipedia.org/wiki/Acknowledgement_(data_networks)) packet from the receiver. It can transmit N packets to the client before requiring an ACK. The receiver keeps track of the sequence number of the next packet it expects to receive, and sends that sequence number with every ACK it sends. The receiver will discard any frame that does not have the exact sequence number it expects (either a duplicate frame it already acknowledged, or an out-of-order frame it expects to receive later) and will resend an ACK for the last correct in-order frame. Once the sender has sent all of the frames in its window, it will detect that all of the frames since the first lost frame are outstanding, and will go back to the sequence number of the last ACK it received from the receiver process and fill its window starting with that frame and continue the process over again.

Part 1: Multi-threaded Web Server

The server waits for the client request containing the file name. Then it divides this file into chunks which are placed in an array. The server gives each chunk a sequence number and then it starts sending them to the client. The server waits for the ack response with the sequence number of the last sent packet. In case of a lost packet or an ack, the server waits until timeout and resends the packets after the last sequence number received. And with each client request, the server runs a new thread.

- Functions used:

def chsum(self, datagram): Returns the check sum of the packet

def wsize(self, datagram): Returns the window size of the packet

def checkonchecksum(self,datagram): Checks if the received check sum is equal to check sum of the packet received

def split(self, data, n): Divides the data into chunks

def canAddToWindow(self): Checks if there is space in sender window

def makeDatagram(self, seqno, packetdata): Forms the packet that will be sent, by returning the checksum, sequence number, packet size and data

def sendDatagram(self, packet): Sends the packets

def addToWindow(self, byte): Adds the packets to the window

def checkingsum(self,data): Calculates the check sum of data

def seqno(self, datagram): Gets packet number from packet

def rwnd(self, datagram): Gets the size of the receiver’s window

def removeFromWindow(self, datagram): Takes packets from sender window

def acceptAcks(self): Checks Acknowledgements

def resendWindow(self): Resends the complete window

def sendEOF(self): Sends ‘End of File’ message

def sendMessage(self, datalist, listlength): Manages the flow of packets

def run(self): Runs each thread

Part 2: Web Client

When running the client, it generates a random port, it then takes the request from the user containing the filename and sends this request to the server. For each received packet, the client checks if it has the sequence number of the expected packet. If it’s the expected packet, the client sends an acknowledgment with this sequence number, if not, the client waits for the timeout of the server so it resends the expected packets.

- Functions used:

def receiving(self): Receives data from server and writes it in file

def chsum(self, datagram): Returns the check sum of the packet

def checkonchecksum(self,datagram): Checks if the received check sum is equal to check sum of the packet received

def removeFromWindow(self, byte): Removes the packet from window

def seqno(self, datagram): Returns sequence number of the packet

def data(self, datagram): Returns the data in the packet

def checkingsum(self,data): Calculates the check sum of data

def canAddToWindow(self): Checks if there is space in sender window

def mkDatagram(self, seqno, winsize): Creates the packet that will be sent

def addToWindow(self, datagram): Adds the packets to the window

def sendAck(self, datagram, counter): Sends acknowledgement to the server

def MakeWholeData(self): Writes the packets from window to file

def receiveMessage(self): Receives the packets

Bonus Features:

Checksum function in both client and server