19CSE301 - COMPUTER NETWORKS

Cab Transportation Network

Group-10

Roll Number	Name
CB.EN.U4CSE19602	Adarsh Jayan
CB.EN.U4CSE19615	Dev Divyendh
CB.EN.U4CSE19626	Karthik Shriram
CB.EN.U4CSE19649	Sri Chakra Teja

Objective:

To be able to book a cab with regards to their departure and destination places using socket programming .

Motivation:

Nowadays cabs are a major part of metropolitan transport system, where people find a balance between comfort and reliability. It is not easy for person to find a taxi stand physically present everywhere he goes. An online application that helps a person to book a cab, pick him up from his location and drop him at his destination will be of a great use in today’s world. With the help of proper networking we can build an usable application that satisfies these needs.

Protocols Used:

¬ TCP/IP
¬ HTML
¬ IMAP/POP3
¬ HTTP

Software/Operating System used:

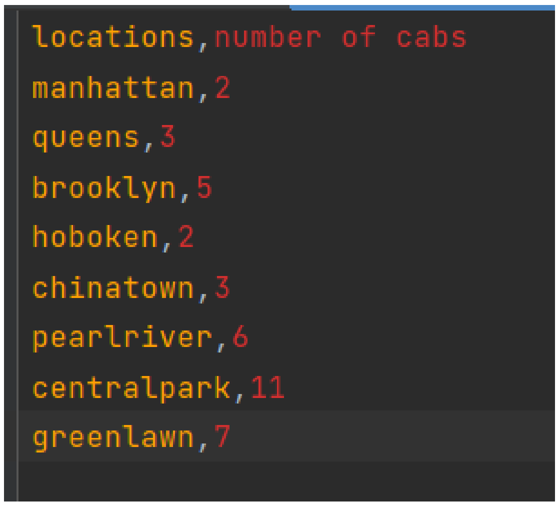
¬ Operating System: Windows 11 ,10
¬ Programming Languages: python

Abstract:

Our Project is cab transportation network whose main aim is to design networks in cab transportation system. Cabs play an important part in traffic transit convenience, which represents a city's level of economic growth and civilization. Cab booking service is a major mode of transportation provided by various transport operators in a given city. The majority of people rely on cab services for their daily transportation needs in developed cities. Companies like Uber, Ola are leading in this sector and they try to make life easier for both the customer and the company by implementing various ideas. In this field company's main tasks are to respond client's request for allocation of cab by finding the cabs which are available in a particular location requested by user and set the cab as free after the user has successfully used the cab and reached his desired location. So, in our system client request for a cab in a particular location and server allocates cab to the user if sufficient cabs are available in the client requested location. Server then decrements the cab available the particular location by 1. Then once the client has successfully reached the destination server increments the available cab in that location by 1. Once the client has reached the destination and user needs to print the bill server prints the bill with appropriate information. In this way our system works.

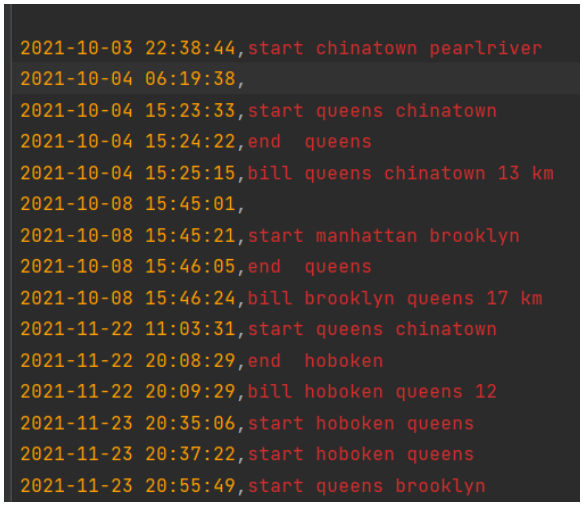
Csv files:

Locations.csv



```
locations,number of cabs
manhattan,2
queens,3
brooklyn,5
hoboken,2
chinatown,3
pearlriver,6
centralpark,11
greenlawn,7
```

Log.csv



```
2021-10-03 22:38:44,start chinatown pearlriver
2021-10-04 06:19:38,
2021-10-04 15:23:33,start queens chinatown
2021-10-04 15:24:22,end  queens
2021-10-04 15:25:15,bill queens chinatown 13 km
2021-10-08 15:45:01,
2021-10-08 15:45:21,start manhattan brooklyn
2021-10-08 15:46:05,end  queens
2021-10-08 15:46:24,bill brooklyn queens 17 km
2021-11-22 11:03:31,start queens chinatown
2021-11-22 20:08:29,end  hoboken
2021-11-22 20:09:29,bill hoboken queens 12
2021-11-23 20:35:06,start hoboken queens
2021-11-23 20:37:22,start hoboken queens
2021-11-23 20:55:49,start queens brooklyn
```

Multiple client server code

Server:

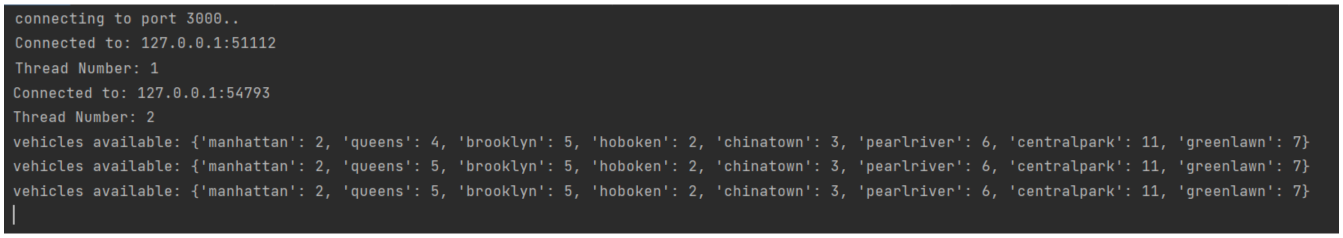
```
from socket import *
import math
import datetime
import os
from _thread import *

def read_csv():
    f=open('locations.csv','r')
    ptr=0
    avail={}
    for i in f:
        if ptr!=0:
            loc,no=i.split(',')
            loc=loc.strip()
            no=int(no.strip())
            avail[loc]=no
        ptr+=1
    return avail

def write_csv(avail):
    f=open('locations.csv','w')
    f.write('locations,number of cabs\n')
    for k in avail:
        f.write(k+','+str(avail[k])+'\n')
    f.close()

def write_log(command):
    f=open('log.csv','a')
    now = datetime.datetime.now()
    f.write(now.strftime('%Y-%m-%d %H:%M:%S')+','+command+'\n')

s = socket(AF_INET, SOCK_STREAM)
port = 3000
Threadcount = 0
s.bind(('localhost', port))
s.listen(True)
print('connecting to port 3000..')
avail=read_csv()

def multi_threaded_client(conn):
    cabs=str(avail)
    welmsg='WELCOME TO ADKT CAR RENTAL SERVICE'+'\n'+'Number Of Available
Cabs:'+'\n'+cabs
    conn.send(welmsg.encode())
    while True:
        command = conn.recv(2048).decode()

        write_log(command)
        command=command.split()
        msg=''
        skip=False
        if command[0]=='start':
            if command[1] not in avail:
                msg='Not available for asked location'
                skip=True
            if not skip and avail[command[1]]==0:
                msg='no vehicles available currently'
                skip=True
            if not skip and command[2] not in avail:
                msg='Destination is invalid'
                skip=True
            if command[1]==command[2]:
                msg='Same start and end location!!! Please change'
                skip=True
            if not skip:
                msg='Vehicle allocated'
                avail[command[1]]-=1
        elif command[0]=='end':
            if command[1] not in avail:
                skip=True
                msg='Destination is invalid'
            if not skip:
                msg='Vehicle returned'
                avail[command[1]]+=1
        elif command[0]=='bill':
            msg='Boarded at:'+command[1]+','+'Dropped at:'+command[2]
            msg=msg+'\n'+'Bill amount:RS.{}'.format(math.floor((float(command[3])*15)))
            msg=msg+'\n'+'THANK YOU!! VISIT AGAIN!!'

        else:
            msg='Invalid command'

        print('vehicles available:',avail)
        conn.send(msg.encode())
        write_csv(avail)
    conn.close()

while True:
    data, address = s.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(multi_threaded_client, (data, ))
    Threadcount += 1
    print('Thread Number: ' + str(Threadcount))
s.close()
```

Client:

```
from socket import *
tcp_soc = socket(AF_INET, SOCK_STREAM)
sendAddr = ('localhost', 3000)
tcp_soc.connect(sendAddr)
msg  = tcp_soc.recv(2048).decode()
print(msg+'\n')
while True:
    print('Enter choice:\n1)Rent a Car \n2)Return a Car \n3)Generate Bill and \nany other 
number to exit')
    ip=int(input())
    if ip==1:
        cname='start'
    elif ip==2:
        cname='end'
    elif ip==3:
        cname='bill'
    else:
        exit(0)

    if ip==1:
        print('Enter Start location:')
        loc=input().strip()
        print('Enter Destination:')
        dest=input().strip()
        command=cname+' '+loc+' '+dest
    if ip==2:
        print('Enter Destination:')
        loc=''
        dest=input().strip()
        command=cname+' '+loc+' '+dest
    if ip==3:
        print('Enter the start and drop locations , distance travelled:')
        dist=input()
        command=cname+' '+dist

    tcp_soc.send(command.encode())
    l = tcp_soc.recv(2048).decode()
    print(l)
tcp_soc.close()
```

Multiple client server Outputs

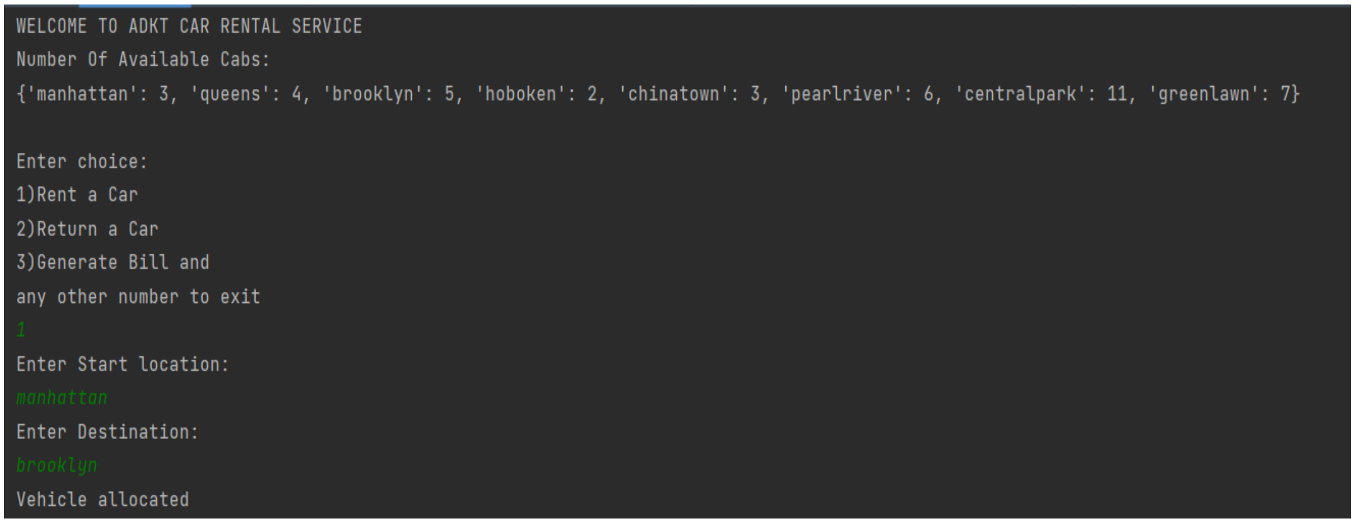
Server-side output



```
connecting to port 3000..
Connected to: 127.0.0.1:51112
Thread Number: 1
Connected to: 127.0.0.1:54793
Thread Number: 2
vehicles available: {'manhattan': 2, 'queens': 4, 'brooklyn': 5, 'hoboken': 2, 'chinatown': 3, 'pearlriver': 6, 'centralpark': 11, 'greenlawn': 7}
vehicles available: {'manhattan': 2, 'queens': 5, 'brooklyn': 5, 'hoboken': 2, 'chinatown': 3, 'pearlriver': 6, 'centralpark': 11, 'greenlawn': 7}
vehicles available: {'manhattan': 2, 'queens': 5, 'brooklyn': 5, 'hoboken': 2, 'chinatown': 3, 'pearlriver': 6, 'centralpark': 11, 'greenlawn': 7}
```

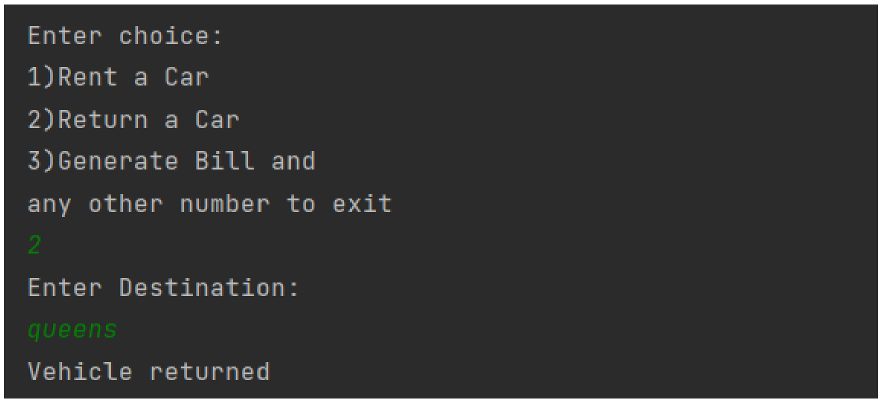
Client-side output

(Case-1, Rent a car)



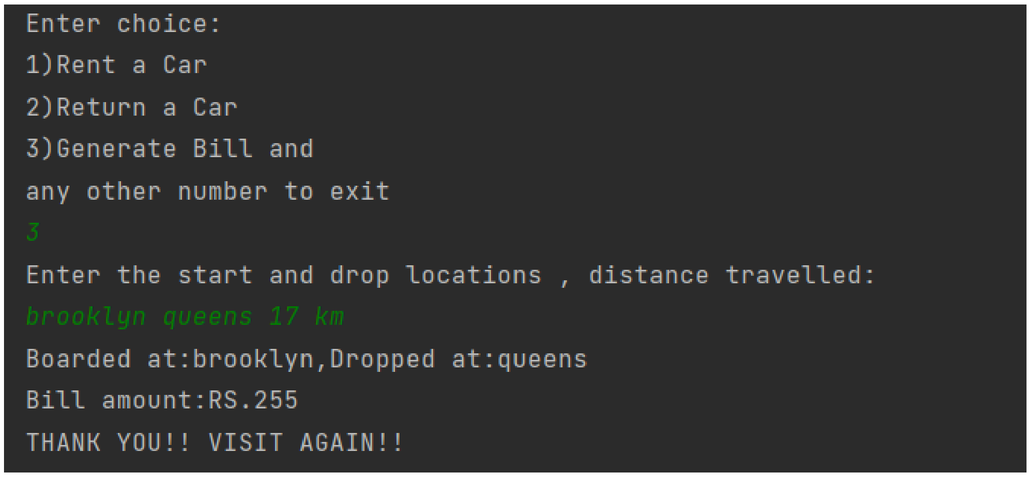
```
WELCOME TO ADKT CAR RENTAL SERVICE
Number Of Available Cabs:
{'manhattan': 3, 'queens': 4, 'brooklyn': 5, 'hoboken': 2, 'chinatown': 3, 'pearlriver': 6, 'centralpark': 11, 'greenlawn': 7}

Enter choice:
1)Rent a Car
2)Return a Car
3)Generate Bill and
any other number to exit
1
Enter Start location:
manhattan
Enter Destination:
brooklyn
Vehicle allocated
```

(Case-2, Return a car)



```
Enter choice:
1)Rent a Car
2)Return a Car
3)Generate Bill and
any other number to exit
2
Enter Destination:
queens
Vehicle returned
```

(Case-3, Generate a bill)



```
Enter choice:
1)Rent a Car
2)Return a Car
3)Generate Bill and
any other number to exit
3
Enter the start and drop locations , distance travelled:
brooklyn queens 17 km
Boarded at:brooklyn,Dropped at:queens
Bill amount:RS.255
THANK YOU!! VISIT AGAIN!!
```

Go back N code

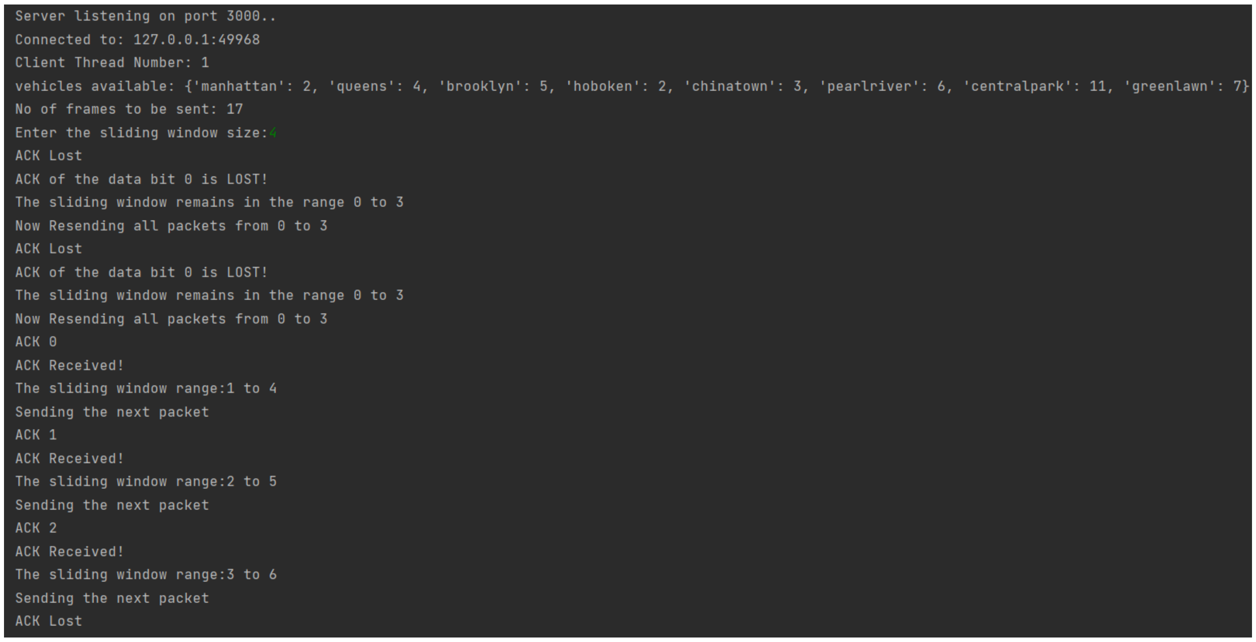
Server:

```
from socket import *
import math
import datetime
import os
import time
import random
from _thread import *


def GBNsend(conn,message):
    conn.send(str(len(message)).encode())
    time.sleep(1)
    i=0
    frames=len(message)
    print("No of frames to be sent:",frames)
    wsize=int(input("Enter the sliding window size:"))
    # wsize=4
    ack=""
    wsize=wsize-1
    k=wsize
    while i!=frames:
        while(i!=(frames-wsize)):
            conn.send(message[i].encode())
            ack=conn.recv(1024)
            ack=ack.decode()
            print(ack)
            if(ack!="ACK Lost"):
                print("ACK Received!")
                print("The sliding window range:"+(str(i+1))+" to "+str(k+1)+"")
                print("Sending the next packet")
                i=i+1
                k=k+1
            else:
                print("ACK of the data bit "+(str(i))+" is LOST!")
                print("The sliding window remains in the range "+(str(i))+" to "+str(k)+"")
                print("Now Resending all packets from "+(str(i))+" to "+(str(k))+"")
            time.sleep(0.1)
        while(i!=frames):
            conn.send(message[i].encode())
            ack=conn.recv(1024)
            ack=ack.decode()
            print(ack)
            if(ack!="ACK Lost"):
                print("ACK Received!")
                print("The sliding window range "+(str(i+1))+" to "+str(k)+"")
                if(i<frames-1):
                    print("Sending the next packet")
                i=i+1
            else:
                print("ACK of the data bit "+(str(i))+" is LOST!")
                print("The sliding window remains in the range "+(str(i))+" to "+str(k)+"")
                print("Now Resending all packets from "+(str(i))+" to "+(str(k))+"")
            time.sleep(0.1)

def GBNrecv(connection):
    k=connection.recv(2048).decode()
    k=int(k)
    i=0
    originalmsg=''
    ack=''
    f=random.randint(0,1)
    nxtchr=''
    while i!=k:
        f=abs(random.randint(0,1) - random.randint(0,1))
        if(f==0):
            ack='ACK '+str(i)
            nxtchr = connection.recv(1024)
            nxtchr = nxtchr.decode()
            connection.send(ack.encode())
            originalmsg=originalmsg+nxtchr
            i=i+1
        else:
            ack='ACK Lost'
            nxtchr = connection.recv(1024)
            nxtchr = nxtchr.decode()
            connection.send(ack.encode())
    # print('The message received is :', originalmsg)
    return originalmsg

def read_csv():
    f=open('locations.csv','r')
    ptr=0
    avail={}
    for i in f:
        if ptr!=0:
            loc,no=i.split(',')
            loc=loc.strip()
            no=int(no.strip())
            avail[loc]=no
        ptr+=1
    return avail

def write_csv(avail):
    f=open('locations.csv','w')
    f.write('locations,number of cabs\n')
    for k in avail:
        f.write(k+','+str(avail[k])+'\n')
    f.close()

def write_log(command):
    f=open('log.csv','a')
    now = datetime.datetime.now()
    f.write(now.strftime('%Y-%m-%d %H:%M:%S')+','+command+'\n')

s = socket(AF_INET, SOCK_STREAM)
port = 3000
Threadcount = 0
s.bind(('localhost', port))
s.listen(True)
print('Server listening on port 3000..')
avail=read_csv()

def multi_threaded_client(conn):
    cabs=str(avail)
    welmsg='WELCOME TO ALS CABS!!!'+'\n'+'Available Cabs:'+'\n'+cabs
    conn.send(welmsg.encode())
    # GBNsend(conn,welmsg)
    while True:
        command = GBNrecv(conn)
        write_log(command)
        command=command.split()
        msg=''
        skip=False
        if command[0]=='start':
            if command[1] not in avail:
                msg='Not available for asked location'
                skip=True
            if not skip and avail[command[1]]==0:
                msg='no vehicles available currently'
                skip=True
            if not skip and command[2] not in avail:
                msg='Destination is invalid'
                skip=True
            if command[1]==command[2]:
                msg='Same start and end location!!! Please change'
                skip=True
            if not skip:
                msg='Vehicle allocated'
                avail[command[1]]-=1
        elif command[0]=='end':
            if command[1] not in avail:
                skip=True
                msg='Destination is invalid'
            if not skip:
                msg='Vehicle returned'
                avail[command[1]]+=1
        elif command[0]=='bill':
            msg='Boarded at:'+command[1]+','+'Droped at:'+command[2]
            msg=msg+'\n'+'Bill amount:₹.{}'.format(math.floor((float(command[3])*22)))
            msg=msg+'\n'+'THANK YOU!! VISIT AGAIN!!'

        else:
            msg='Invalid command'

        print('vehicles available:',avail)
        # conn.send(msg.encode())
        GBNsend(conn, msg)
        write_csv(avail)

        # if not command:
        #     break
        # conn.sendall(str.encode(response))
    conn.close()

while True:
    data, address = s.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(multi_threaded_client, (data, ))
    Threadcount += 1
    print('Client Thread Number: ' + str(Threadcount))
s.close()
```

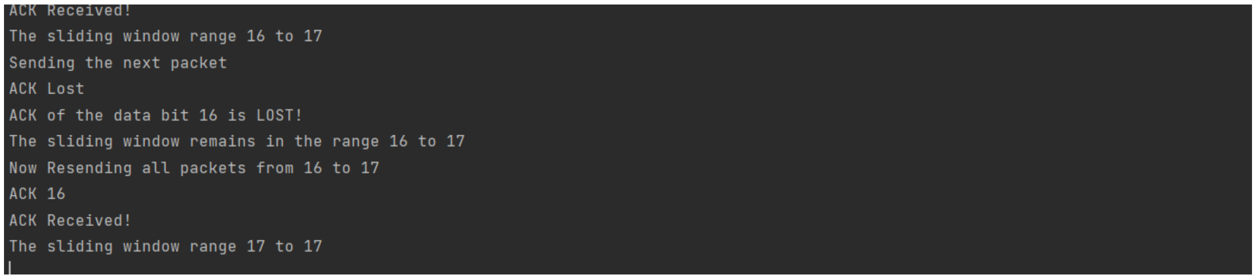
Client:

```
from socket import *
import time
import random
def GBNsend(conn,message):
    conn.send(str(len(message)).encode())
    time.sleep(1)
    i=0
    frames=len(message)
    print("No of frames to be sent:",frames)
    wsize=int(input("Enter the sliding window size:"))
    # wsize=4
    ack=""
    wsize=wsize-1
    k=wsize
    while i!=frames:
        while(i!=(frames-wsize)):
            conn.send(message[i].encode())
            ack=conn.recv(1024)
            ack=ack.decode()
            print(ack)
            if(ack!="ACK Lost"):
                print("ACK Received!")
                print("The sliding window range:"+(str(i+1))+" to "+str(k+1)+"")
                print("Sending the next packet")
                i=i+1
                k=k+1
            else:
                print("ACK of the data bit "+(str(i))+" is LOST!")
                print("The sliding window remains in the range "+(str(i))+" to "+str(k)+"")
                print("Now Resending all packets from "+(str(i))+" to "+(str(k))+"")
            time.sleep(0.1)
        while(i!=frames):
            conn.send(message[i].encode())
            ack=conn.recv(1024)
            ack=ack.decode()
            print(ack)
            if(ack!="ACK Lost"):
                print("ACK Received!")
                print("The sliding window range "+(str(i+1))+" to "+str(k)+"")
                if(i<frames-1):
                    print("Sending the next packet")
                i=i+1
            else:
                print("ACK of the data bit "+(str(i))+" is LOST!")
                print("The sliding window remains in the range "+(str(i))+" to "+str(k)+"")
                print("Now Resending all packets from "+(str(i))+" to "+(str(k))+"")
            time.sleep(0.1)

def GBNrecv(connection):
    k=connection.recv(2048).decode()
    # print(k,'len of msg')
    k=int(k)
    i=0
    originalmsg=''
    ack=''
    f=random.randint(0,1)
    nxtchr=''
    while i!=k:
        f=abs(random.randint(0,1) - random.randint(0,1))
        if(f==0):
            ack='ACK '+str(i)
            nxtchr = connection.recv(1024)
            nxtchr = nxtchr.decode()
            # print(nxtchr)
            connection.send(ack.encode())
            originalmsg=originalmsg+nxtchr
            i=i+1
        else:
            ack='ACK Lost'
            nxtchr = connection.recv(1024)
            nxtchr = nxtchr.decode()
            connection.send(ack.encode())
    # print('The message received is :', originalmsg)
    return originalmsg

tcp_soc = socket(AF_INET, SOCK_STREAM)
sendAddr = ('localhost', 3000)
tcp_soc.connect(sendAddr)
msg  = tcp_soc.recv(2048).decode()
# msg=GBNrecv(tcp_soc)
print(msg+'\n')
while True:
    print('Enter choice: 1)Rent 2)Return 3)Generate Bill or any other number to exit')
    ip=int(input())
    if ip==1:
        cname='start'
    elif ip==2:
        cname='end'
    elif ip==3:
        cname='bill'
    else:
        exit(0)

    if ip==1:
        print('Enter Start location:')
        loc=input().strip()
        print('Enter Destination:')
        dest=input().strip()
        command=cname+' '+loc+' '+dest
    if ip==2:
        print('Enter Destination:')
        loc=''
        dest=input().strip()
        command=cname+' '+loc+' '+dest
    if ip==3:
        print('Enter the start,drop locations and distance travelled:')
        dist=input()
        command=cname+' '+dist

    # tcp_soc.send(command.encode())
    GBNsend(tcp_soc,command)
    # l = tcp_soc.recv(2048).decode()
    l=GBNrecv(tcp_soc)
    print(l)
tcp_soc.close()
```

Server-side output:

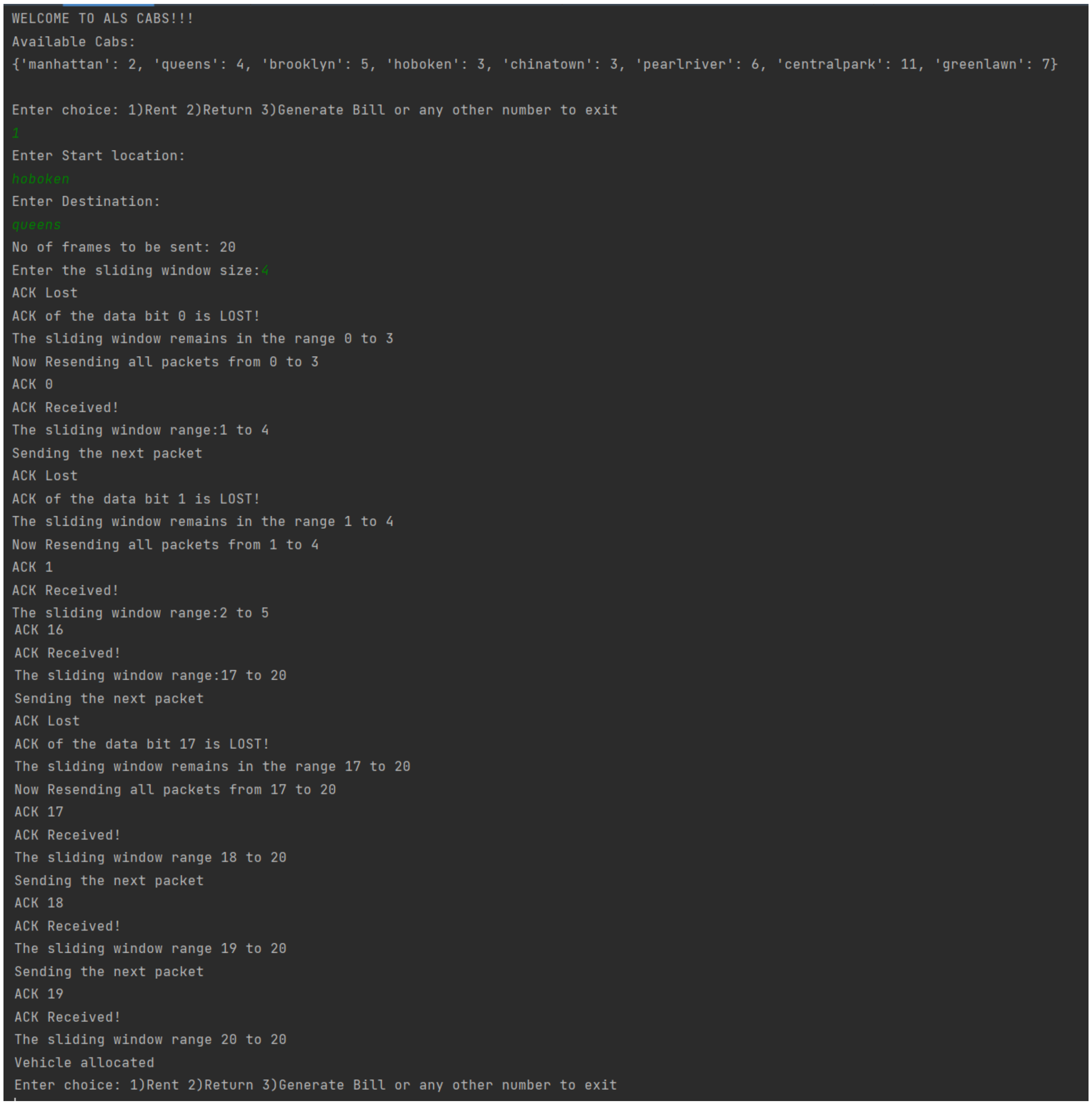


```
Server listening on port 3000..
Connected to: 127.0.0.1:49968
Client Thread Number: 1
vehicles available: {'manhattan': 2, 'queens': 4, 'brooklyn': 5, 'hoboken': 2, 'chinatown': 3, 'pearlriver': 6, 'centralpark': 11, 'greenlawn': 7}
No of frames to be sent: 17
Enter the sliding window size:4
ACK Lost
ACK of the data bit 0 is LOST!
The sliding window remains in the range 0 to 3
Now Resending all packets from 0 to 3
ACK Lost
ACK of the data bit 0 is LOST!
The sliding window remains in the range 0 to 3
Now Resending all packets from 0 to 3
ACK 0
ACK Received!
The sliding window range:1 to 4
Sending the next packet
ACK 1
ACK Received!
The sliding window range:2 to 5
Sending the next packet
ACK 2
ACK Received!
The sliding window range:3 to 6
Sending the next packet
ACK Lost
```



```
ACK Received!
The sliding window range 16 to 17
Sending the next packet
ACK Lost
ACK of the data bit 16 is LOST!
The sliding window remains in the range 16 to 17
Now Resending all packets from 16 to 17
ACK 16
ACK Received!
The sliding window range 17 to 17
```

Client-side output:



```
WELCOME TO ALS CABS!!!
Available Cabs:
{'manhattan': 2, 'queens': 4, 'brooklyn': 5, 'hoboken': 3, 'chinatown': 3, 'pearlriver': 6, 'centralpark': 11, 'greenlawn': 7}

Enter choice: 1)Rent 2)Return 3)Generate Bill or any other number to exit
1
Enter Start location:
hoboken
Enter Destination:
queens
No of frames to be sent: 20
Enter the sliding window size:4
ACK Lost
ACK of the data bit 0 is LOST!
The sliding window remains in the range 0 to 3
Now Resending all packets from 0 to 3
ACK 0
ACK Received!
The sliding window range:1 to 4
Sending the next packet
ACK Lost
ACK of the data bit 1 is LOST!
The sliding window remains in the range 1 to 4
Now Resending all packets from 1 to 4
ACK 1
ACK Received!
The sliding window range:2 to 5
ACK 16
ACK Received!
The sliding window range:17 to 20
Sending the next packet
ACK Lost
ACK of the data bit 17 is LOST!
The sliding window remains in the range 17 to 20
Now Resending all packets from 17 to 20
ACK 17
ACK Received!
The sliding window range 18 to 20
Sending the next packet
ACK 18
ACK Received!
The sliding window range 19 to 20
Sending the next packet
ACK 19
ACK Received!
The sliding window range 20 to 20
Vehicle allocated
Enter choice: 1)Rent 2)Return 3)Generate Bill or any other number to exit
```

Select Repeat code

Server:

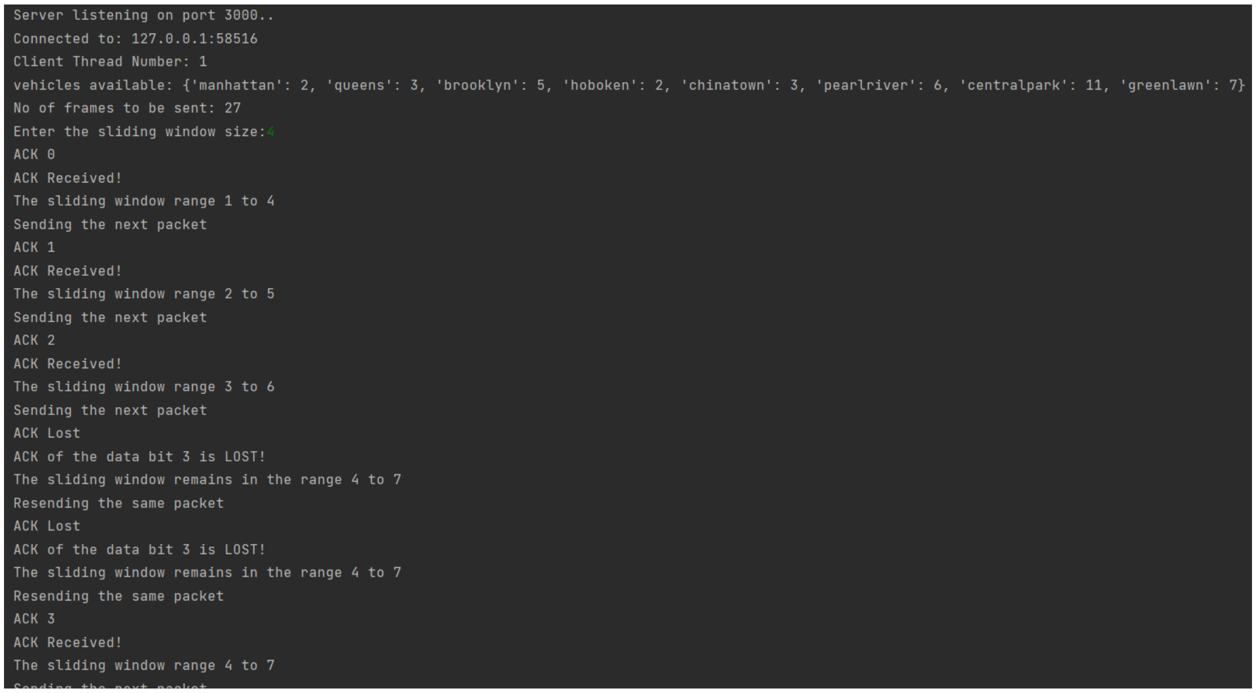
```
from socket import *
import math
import datetime
import os
import time
import random
from _thread import *


def SRrecv(conn):
    k=conn.recv(2048).decode()
    k=int(k)
    i=0
    orig_msg=[""] * (k-1)
    b=""
    ack=[]
    f=random.randint(0,1)
    message=""
    while(i!=4):
        message = conn.recv(1024).decode()
        orig_msg[i] = message
        # print(message)
        ack.append(i)
        i+=1
    while i!=k-1:
        f=abs(random.randint(0,1) - random.randint(0,1))
        if(f==0):
            b="ACK "+str(ack[0])
            conn.send(b.encode())
            message = conn.recv(1024).decode()
            # print(message)
            orig_msg[i] = message
            ack.append(i)
            ack.pop(0)
            i=i+1

        else:
            b="ACK Lost"
            conn.send(b.encode())
            message = conn.recv(1024)
            message = message.decode()
            orig_msg[ack[0]] = message
    while(len(ack) != 0):
        b="ACK "+str(ack[0])
        conn.send(b.encode())
        ack.pop(0)
        i+=1
        time.sleep(0.5)
    conn.send('Final ACK'.encode())
    return ''.join(orig_msg).strip()

def SRsend(conn,message):
    message+=' '*10
    conn.send(str(len(message)).encode())
    time.sleep(1)
    i=0
    f=len(message)
    print("No of frames to be sent:",f)
    wsize=int(input("Enter the sliding window size:"))
    b=""
    wsize=wsize-1
    ack=[]
    while(i!=wsize+1):
        conn.send(message[i].encode())
        ack.append(i)
        i+=1
        time.sleep(0.1)
    time.sleep(1)
    while i!=f-1:
        b=conn.recv(1024)
        b=b.decode()
        print(b)
        if(b!="ACK Lost"):
            print("ACK Received!")
            print("The sliding window range "+(str(i+1-4))+" to "+str(min(wsize+1,f-1))+"")
            print("Sending the next packet")
            ack.pop(0)
            conn.send(message[i].encode())
            ack.append(i)
            i+=1
            wsize+=1
        else:
            print("ACK of the data bit " + str(ack[0]) + " is LOST!")
            print("The sliding window remains in the range "+(str(i+1-4))+" to 
"+str(min(wsize+1,f-1))+"")
            print("Resending the same packet")
            conn.send(message[ack[0]].encode())
        time.sleep(0.1)
    k=1
    message = conn.recv(1024).decode()
    while(message!="Final ACK"):
        print(message)
        print("ACK Received!")
        print("The sliding window range "+(str(i+1-4))+" to "+str(min(wsize+1,f-1))+"")
        print("Sending the next packet")
        ack.pop(0)
        message = conn.recv(1024).decode()
        k+=1
    print("ACK",f-1)
    print("ACK Received!")
    print("The sliding window range "+(str(i+1-4))+" to "+str(min(wsize+1,f-1))+"")
    # print("Now sending the next packet")
    print("\nAll Packets Acknowledged")

def read_csv():
    f=open('locations.csv','r')
    ptr=0
    avail={}
    for i in f:
        if ptr!=0:
            loc,no=i.split(',')
            loc=loc.strip()
            no=int(no.strip())
            avail[loc]=no
        ptr+=1
    return avail

def write_csv(avail):
    f=open('locations.csv','w')
    f.write('locations,number of cabs\n')
    for k in avail:
        f.write(k+','+str(avail[k])+'\n')
    f.close()

def write_log(command):
    f=open('log.csv','a')
    now = datetime.datetime.now()
    f.write(now.strftime('%Y-%m-%d %H:%M:%S')+','+command+'\n')

s = socket(AF_INET, SOCK_STREAM)
port = 3000
Threadcount = 0
s.bind(('localhost', port))
```

```
s.listen(True)
print('Server listening on port 3000..')
avail=read_csv()

def multi_threaded_client(conn):
    # conn.send(str.encode('Server is working:'))
    cabs=str(avail)
    welmsg='WELCOME TO ALS CABS!!!'+'\n'+'Available Cabs:'+'\n'+cabs
    # SRsend(conn,welmsg)
    conn.send(welmsg.encode())
    while True:
        command = SRrecv(conn)
        write_log(command)
        command=command.split()
        msg=''
        skip=False
        if command[0]=='start':
            if command[1] not in avail:
                msg='Not available for asked location'
                skip=True
            if not skip and avail[command[1]]==0:
                msg='no vehicles available currently'
                skip=True
            if not skip and command[2] not in avail:
                msg='Destination is invalid'
                skip=True
            if command[1]==command[2]:
                msg='Same start and end location!!! Please change'
                skip=True
            if not skip:
                msg='Vehicle allocated'
                avail[command[1]]-=1
        elif command[0]=='end':
            if command[1] not in avail:
                skip=True
                msg='Destination is invalid'
            if not skip:
                msg='Vehicle returned'
                avail[command[1]]+=1
        elif command[0]=='bill':
            msg='Boarded at:'+command[1]+','+'Droped at:'+command[2]
            msg=msg+'\n'+'Bill amount:₹.{}'.format(math.floor((float(command[3])*22)))
            msg=msg+'\n'+'THANK YOU!! VISIT AGAIN!!'

        else:
            msg='Invalid command'
        print('vehicles available:',avail)
        SRsend(conn,msg)
        write_csv(avail)
    conn.close()

while True:
    data, address = s.accept()
    print('Connected to: ' + address[0] + ':' + str(address[1]))
    start_new_thread(multi_threaded_client, (data, ))
    Threadcount += 1
    print('Client Thread Number: ' + str(Threadcount))
s.close()
```

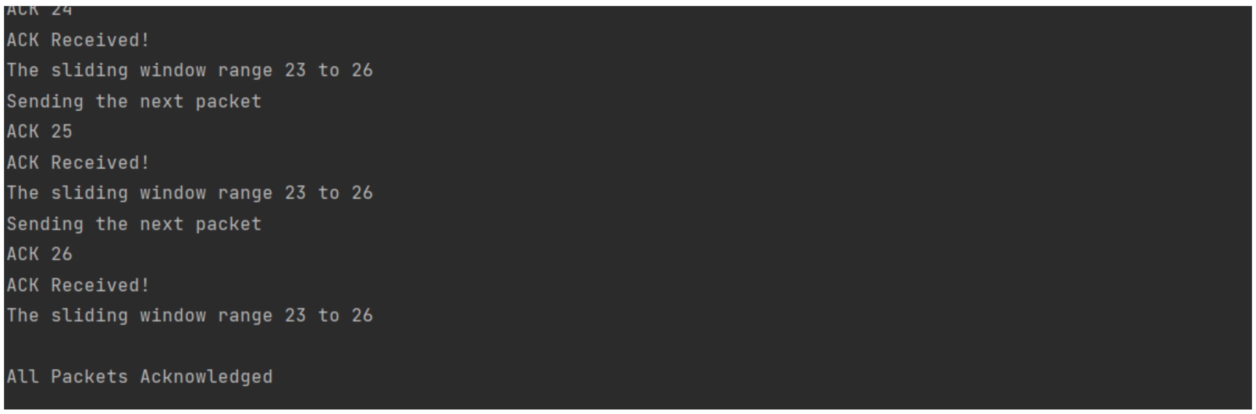
Client:

```
from socket import *
import random
import time

def SRrecv(conn):
    k=conn.recv(2048).decode()
    k=int(k)
    i=0
    orig_msg=[""] * (k-1)
    b=""
    ack=[]
    f=random.randint(0,1)
    message=""
    while(i!=4):
        message = conn.recv(1024).decode()
        orig_msg[i] = message
        # print(message)
        ack.append(i)
        i+=1
    while i!=k-1:
        f=abs(random.randint(0,1) - random.randint(0,1))
        if(f==0):
            b="ACK "+str(ack[0])
            conn.send(b.encode())
            message = conn.recv(1024).decode()
            # print(message)
            orig_msg[i] = message
            ack.append(i)
            ack.pop(0)
            i=i+1

        else:
            b="ACK Lost"
            conn.send(b.encode())
            message = conn.recv(1024)
            message = message.decode()
            orig_msg[ack[0]] = message
    while(len(ack) != 0):
        b="ACK "+str(ack[0])
        conn.send(b.encode())
        ack.pop(0)
        i+=1
        time.sleep(0.5)
    conn.send('Final ACK'.encode())
    return ''.join(orig_msg).strip()

def SRsend(conn,message):
    message+=' '*10
    conn.send(str(len(message)).encode())
    time.sleep(1)
    i=0
    f=len(message)
    print("No of frames to be sent:",f)
    wsize=int(input("Enter the sliding window size:"))
    b=""
    wsize=wsize-1
    ack=[]
    while(i!=wsize+1):
        conn.send(message[i].encode())
        ack.append(i)
        i+=1
        time.sleep(0.1)
    time.sleep(1)
    while i!=f-1:
        b=conn.recv(1024)
        b=b.decode()
        print(b)
        if(b!="ACK Lost"):
            print("ACK Received!")
            print("The sliding window range "+(str(i+1-4))+" to "+str(min(wsize+1,f-1))+"")
            print("Sending the next packet")
            ack.pop(0)
            conn.send(message[i].encode())
            ack.append(i)
            i+=1
            wsize+=1
        else:
            print("ACK of the data bit " + str(ack[0]) + " is LOST!")
            print("The sliding window remains in the range "+(str(i+1-4))+" to
"+str(min(wsize+1,f-1))+"")
            print("Resending the same packet")
            conn.send(message[ack[0]].encode())
        time.sleep(0.1)
    k=1
    message = conn.recv(1024).decode()
    while(message!="Final ACK"):
        print(message)
        print("ACK Received!")
        print("The sliding window range "+(str(i+1-4))+" to "+str(min(wsize+1,f-1))+"")
        print("Sending the next packet")
        ack.pop(0)
        message = conn.recv(1024).decode()
        k+=1
    print("ACK",f-1)
    print("ACK Received!")
    print("The sliding window range "+(str(i+1-4))+" to "+str(min(wsize+1,f-1))+"")
    # print("Now sending the next packet")
    print("\nAll Packets Acknowledged")

tcp_soc = socket(AF_INET, SOCK_STREAM)
sendAddr = ('localhost', 3000)
tcp_soc.connect(sendAddr)
msg  = tcp_soc.recv(2048).decode()
print(msg+'\n')
while True:
    print('Enter choice: 1)Rent 2)Return 3)Generate Bill or any other number to exit')
    ip=int(input())
    if ip==1:
        cname='start'
    elif ip==2:
        cname='end'
    elif ip==3:
        cname='bill'
    else:
        exit(0)

    if ip==1:
        print('Enter Start location:')
        loc=input().strip()
        print('Enter Destination:')
        dest=input().strip()
        command=cname+' '+loc+' '+dest
    if ip==2:
        print('Enter Destination:')
        loc=''
        dest=input().strip()
        command=cname+' '+loc+' '+dest
    if ip==3:
        print('Enter the start,drop locations and distance travelled:')
        dist=input()
        command=cname+' '+dist

    SRsend(tcp_soc,command)
    l = SRrecv(tcp_soc)
    print(l)
tcp_soc.close()
```

Server-side Output:

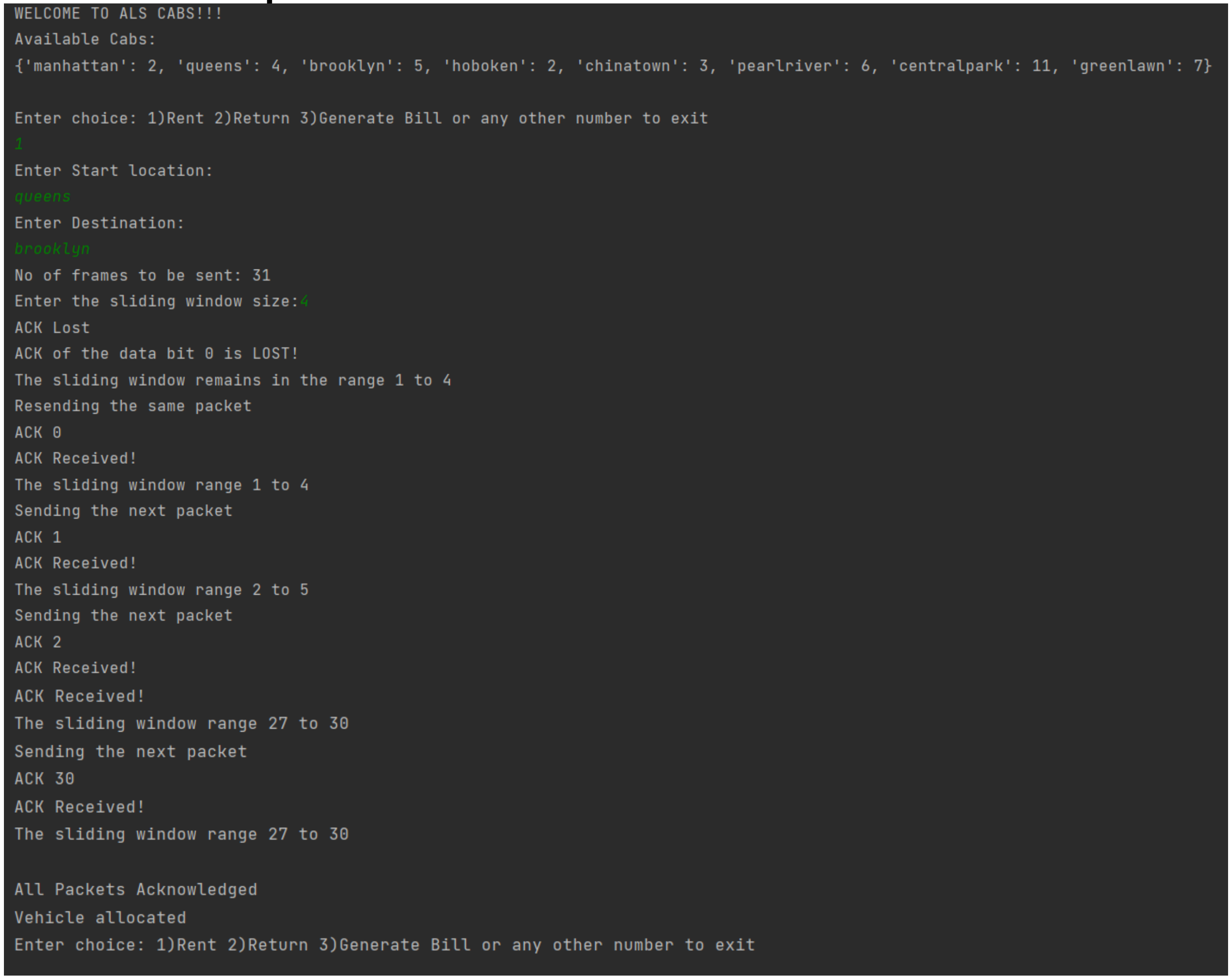


```
Server listening on port 3000..
Connected to: 127.0.0.1:58516
Client Thread Number: 1
vehicles available: {'manhattan': 2, 'queens': 3, 'brooklyn': 5, 'hoboken': 2, 'chinatown': 3, 'pearlriver': 6, 'centralpark': 11, 'greenlawn': 7}
No of frames to be sent: 27
Enter the sliding window size:4
ACK 0
ACK Received!
The sliding window range 1 to 4
Sending the next packet
ACK 1
ACK Received!
The sliding window range 2 to 5
Sending the next packet
ACK 2
ACK Received!
The sliding window range 3 to 6
Sending the next packet
ACK Lost
ACK of the data bit 3 is LOST!
The sliding window remains in the range 4 to 7
Resending the same packet
ACK Lost
ACK of the data bit 3 is LOST!
The sliding window remains in the range 4 to 7
Resending the same packet
ACK 3
ACK Received!
The sliding window range 4 to 7
Sending the next packet
```



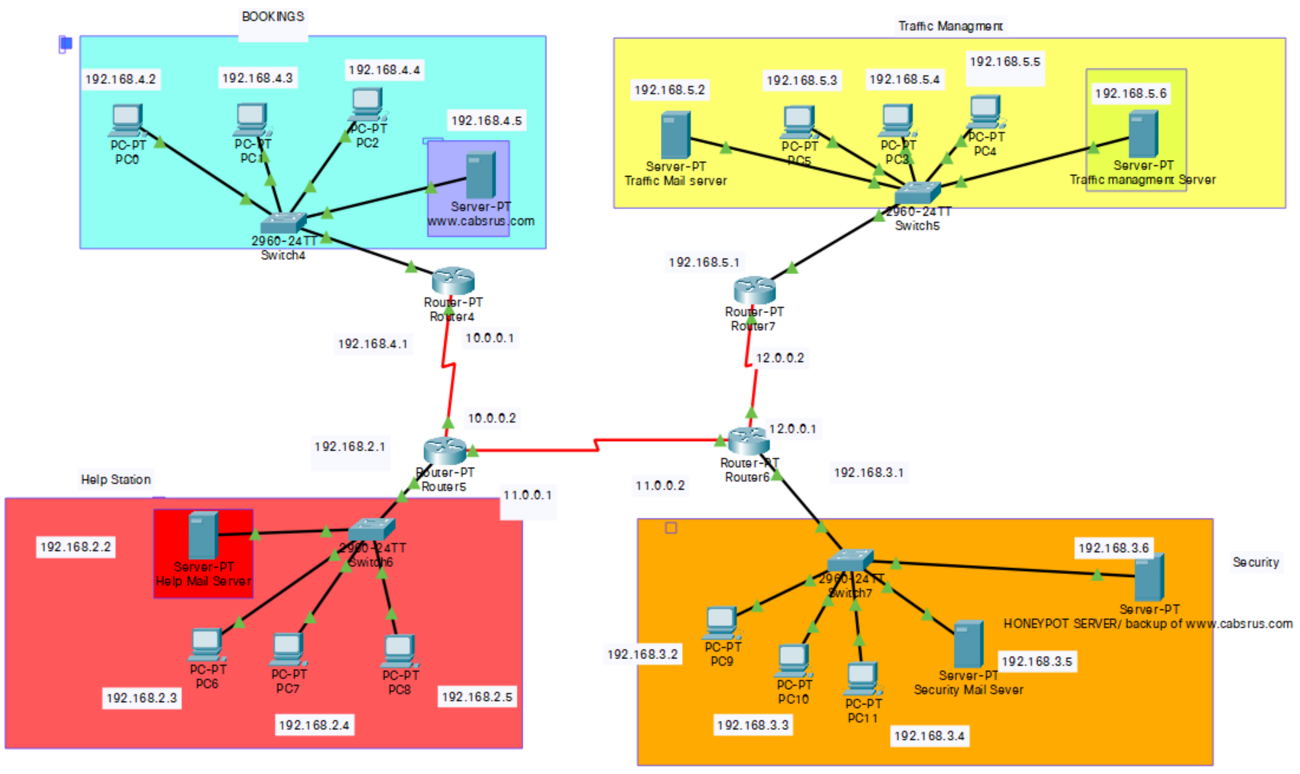
```
ACK 24
ACK Received!
The sliding window range 23 to 26
Sending the next packet
ACK 25
ACK Received!
The sliding window range 23 to 26
Sending the next packet
ACK 26
ACK Received!
The sliding window range 23 to 26

All Packets Acknowledged
```

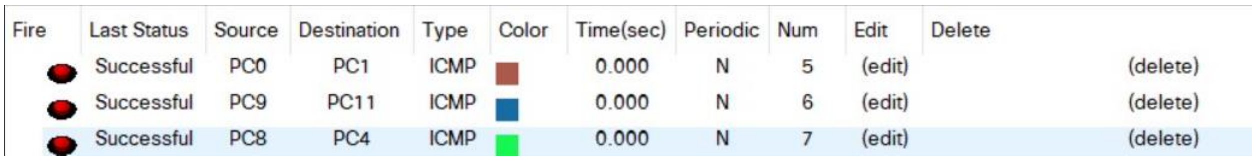
Client-side Output:



```
WELCOME TO ALS CABS!!!
Available Cabs:
{'manhattan': 2, 'queens': 4, 'brooklyn': 5, 'hoboken': 2, 'chinatown': 3, 'pearlriver': 6, 'centralpark': 11, 'greenlawn': 7}

Enter choice: 1)Rent 2)Return 3)Generate Bill or any other number to exit
1
Enter Start location:
queens
Enter Destination:
brooklyn
No of frames to be sent: 31
Enter the sliding window size:4
ACK Lost
ACK of the data bit 0 is LOST!
The sliding window remains in the range 1 to 4
Resending the same packet
ACK 0
ACK Received!
The sliding window range 1 to 4
Sending the next packet
ACK 1
ACK Received!
The sliding window range 2 to 5
Sending the next packet
ACK 2
ACK Received!
ACK Received!
The sliding window range 27 to 30
Sending the next packet
ACK 30
ACK Received!
The sliding window range 27 to 30

All Packets Acknowledged
Vehicle allocated
Enter choice: 1)Rent 2)Return 3)Generate Bill or any other number to exit
```

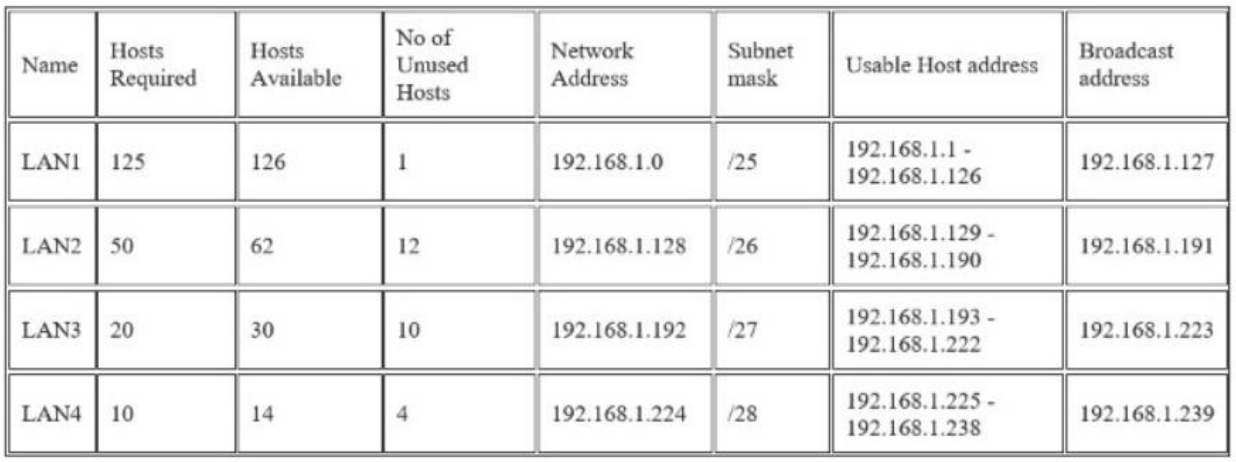
Cisco packet tracer diagram:



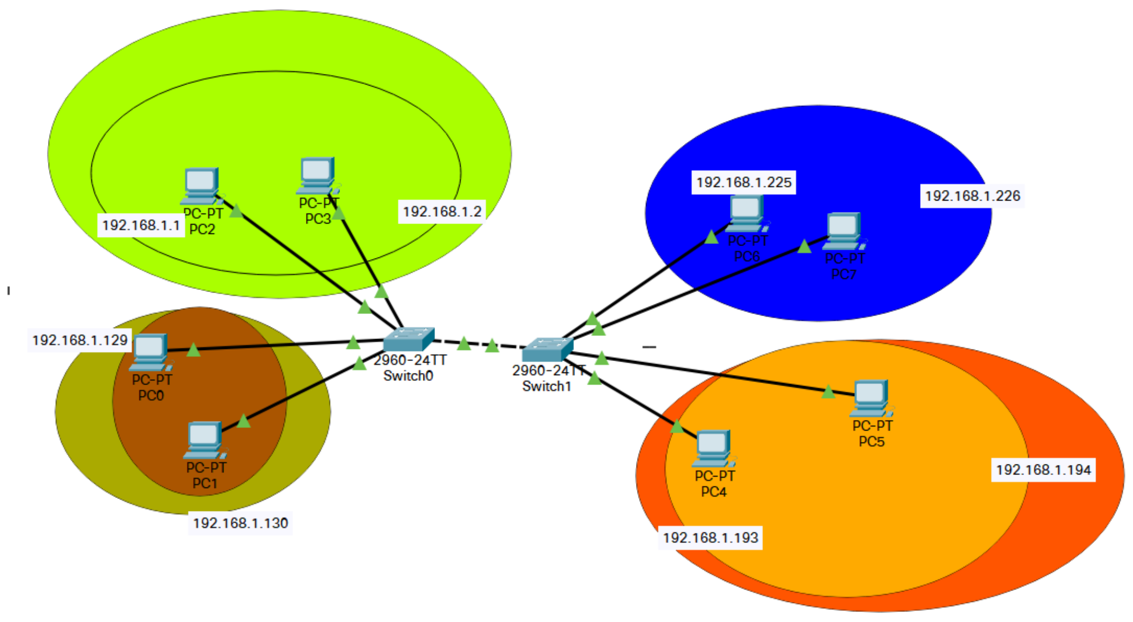


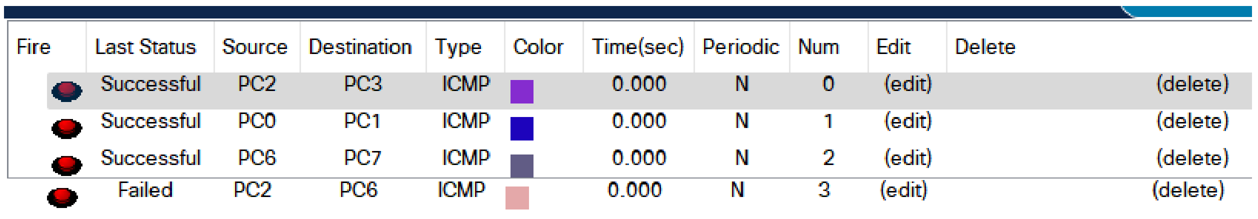
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC0	PC1	ICMP	🟫	0.000	N	5	(edit)	(delete)
●	Successful	PC9	PC11	ICMP	🟦	0.000	N	6	(edit)	(delete)
●	Successful	PC8	PC4	ICMP	🟩	0.000	N	7	(edit)	(delete)

Vlan diagram:



Name	Hosts Required	Hosts Available	No of Unused Hosts	Network Address	Subnet mask	Usable Host address	Broadcast address
LAN1	125	126	1	192.168.1.0	/25	192.168.1.1 - 192.168.1.126	192.168.1.127
LAN2	50	62	12	192.168.1.128	/26	192.168.1.129 - 192.168.1.190	192.168.1.191
LAN3	20	30	10	192.168.1.192	/27	192.168.1.193 - 192.168.1.222	192.168.1.223
LAN4	10	14	4	192.168.1.224	/28	192.168.1.225 - 192.168.1.238	192.168.1.239





Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC2	PC3	ICMP	🟪	0.000	N	0	(edit)	(delete)
●	Successful	PC0	PC1	ICMP	🟦	0.000	N	1	(edit)	(delete)
●	Successful	PC6	PC7	ICMP	■	0.000	N	2	(edit)	(delete)
●	Failed	PC2	PC6	ICMP	■	0.000	N	3	(edit)	(delete)

Conclusion:

Our objective to be able to book a cab with regards to their departure and destination places using socket programming, Vlan , rip routing also to create a track record of all the fare details and also to show the structure of the network .In case the message has failed to reach from server to client, the message will be re-transmitted via Selective Repeat & GoBackN. And the model has been implemented in Cisco Packet Tracer.

References:

1)https://w7cloud.com/nat-configuration-on-packet-tracer/#more-1669
2)https://www.practicalnetworking.net/stand-alone/routing-between-vlans/
3)https://www.slideshare.net/arzsy91/lab-practice-1-configuring-basic-routing-and-switching-with-answer

4)https://www.eventhelix.com/networking/ip-routing