**19CSE301- COMPUTER NETWORK**

**Case Study**

**Traffic Network System**

**Name: Suryanarayan.B**

**Roll no. :CB.EN.U4CSE19056**

|  |  |  |  |
| --- | --- | --- | --- |
| **Registration No** | **Name** | **Email ID** | **Contribution** |
| CB.EN. U4CSE19060 | S. Vigneshwararaj |  | Control Room |
| CB.EN. U4CSE19056 | Suryanarayan.B |  | Patrol and Toll |
| CB.EN. U4CSE19001 | Adithya Varshan |  | Judicial |
| CB.EN. U4CSE19008 | B. Sudarshan |  | RTO |

**Why do we require** **Networks in Road Transport?**

A common feature of **Networks in Road Transport** – when applied to traffic and road network management – is the use of real-time, conventional and historic data sources to produce information on the existing and future status of the road transport system. **Networks in Road Transport** applications play an important part in the way road networks are managed to improve the efficiency and reliability of transport operations and reduce negative environmental and energy consumption impacts.

The Main Departments of NRT are:

* Control Room Dept
* Road Patrol and Tolls Dept
* Judicial
* RTO

**Control Room:**

It is the place where all the traffic is controlled. It has the details about each signal and their status. It also has the Information about Road blocks and railway crossing etc. From this dept we can not only access the data but also control the traffic by auto mode. This dept also has the information about some special controls like CM Escort traffic control, Public Conference, Festival Gatherings etc…If any of these special occurs control room gives their signals to Road Patrol dept.

**Road Patrol and Toll dept:**

Highway toll and Check post department monitors the traffic and vehicle counts which passes through either a toll booth or a check post which is normally located in the forest regions (Tiger reserves, sanctuaries). It also escorts the CM or PM travel.

**RTO:**

It is the department which checks all the vehicles and their passenger’s Licence. This department has all information about the vehicles and their owners. The main duty of this dept is to check the vehicle and take actions on them. All this information is accessed by the Judicial dept

**Judicial:**

The main role of this dept is to check who are all charged by violating traffic rules and giving punishments. These information are stored in database and can be accessed by the clients.

**Software/Operating System used:**

* **OS used:**
  + Windows 10
  + Windows DOS
* **Software used:**
  + Visual Studio Code
  + Cisco Packet Tracer
* **Language:**
  + Python

**Why measure network performance?**

The demands on networks are increasing every day, and the need for proper network performance measurement is more important than ever before. Effective network performance translates into improved user satisfaction, whether that be internal employee efficiencies, or customer-facing network components such as an e-commerce website, making the business rationale for performance testing and monitoring self-evident.

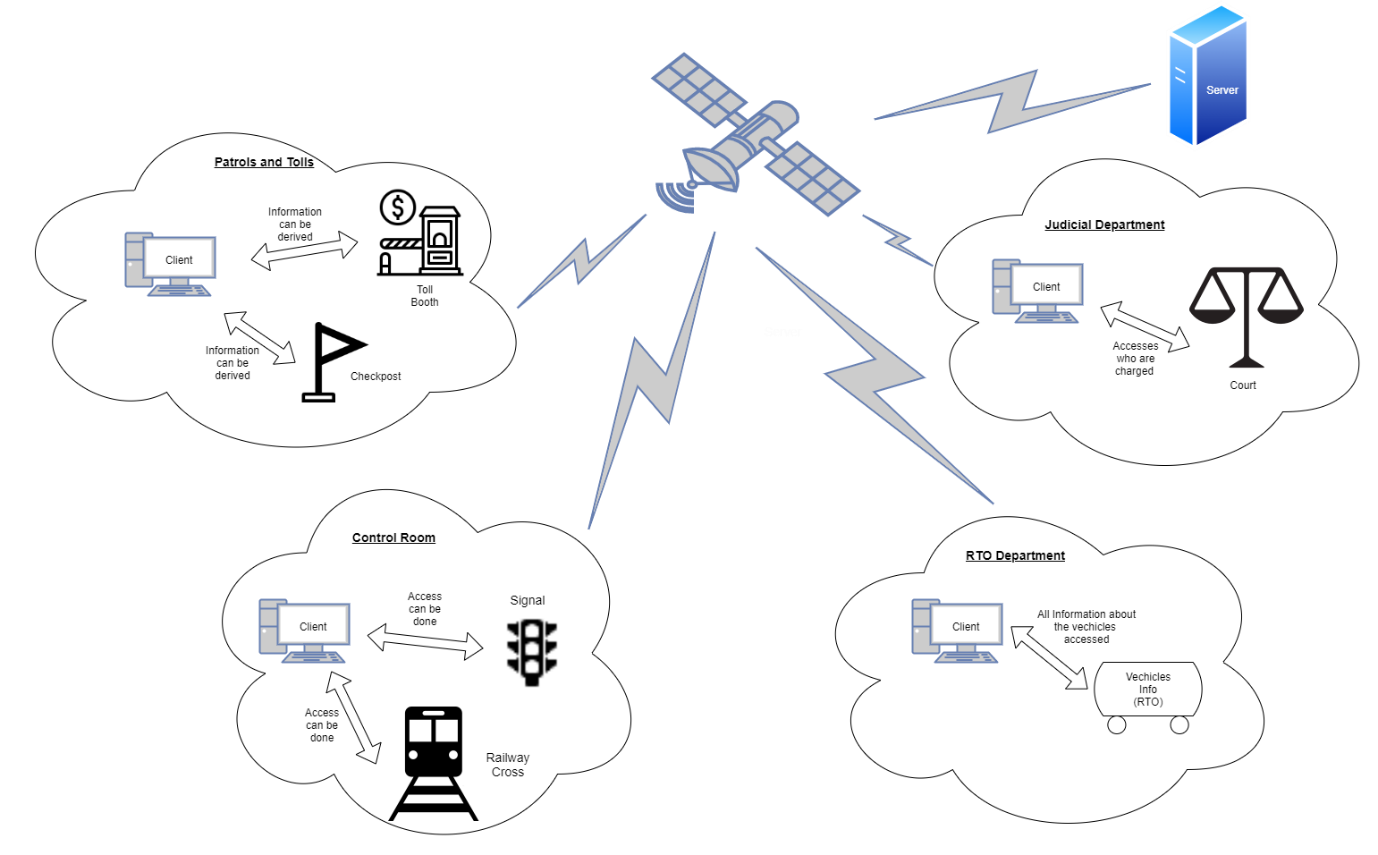
When delivering services and applications to users, bandwidth issues, network down time, and bottlenecks can quickly escalate into IT crisis mode. Proactive network performance management solutions that detect and diagnose performance issues are the best way to guarantee ongoing user satisfaction.

The performance of a network can never be fully modelled, so measuring network performance before, during, and after updates are made and monitoring performance on an ongoing basis are the only valid methods to fully ensure network quality. While measuring and monitoring network performance parameters are essential, the interpretation and actions stemming from these metrics are equally important.

**Performance parameters:**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Meaning** | **Formula** |
| **Bandwidth** | Bandwidth is the capacity of a wired or wireless network communications link to transmit the maximum amount of data from one point to another over a computer network or internet connection in a given amount of time | Expressed as [bits](https://web.archive.org/web/20190816003233/https:/whatis.techtarget.com/definition/bit-binary-digit) per second ([bps](https://web.archive.org/web/20190816003233/https:/searchnetworking.techtarget.com/definition/bits-per-second)), modern network links have greater capacity, which is typically measured in millions of bits per second ([megabits per second](https://web.archive.org/web/20190816003233/https:/searchnetworking.techtarget.com/definition/Mbps), or Mbps) or billions of bits per second ([gigabits per second](https://web.archive.org/web/20190816003233/https:/whatis.techtarget.com/definition/Gbps-billions-of-bits-per-second), or Gbps). |
| **Throughput** | Throughput measures the percentage of data packets that are successfully being sent; a low throughput means there are a lot of failed or dropped packets that need to be sent again. |  |
| **Packet Loss** | Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Due to network congestion | Efficiency = 100% \* (transferred - retransmitted) / transferred  Network Loss = 100 - Efficiency |
| **Transmission time** | The time required for transmission of a message depends on the size of the message and the bandwidth of the channel. | Transmission time=Message size / Bandwidth |
| **Propagation Time** | Propagation time measures the time required for a bit to travel from the source to the destination. The propagation time is calculated by dividing the distance by the propagation speed. | Propagation time = Distance /Propagation speed |
| **Processing Delay** | Time taken by the processor to process the data packet is called processing delay. |  |
| **Queuing Delay** | Time spent by the data packet waiting in the queue before it is taken for execution is called queuing delay. |  |
| **Jitter** | Jitter is defined as the variation in time delay for the data packets sent over a network. This variable represents an identified disruption in the normal sequencing of data packets. Jitter is related to latency, since the jitter manifests itself in increased or uneven latency between data packets, which can disrupt network performance and lead to packet loss and network congestion. Although some level of jitter is to be expected and can usually be tolerated, quantifying network jitter is an important aspect of comprehensive network | Latency=sum of all delays    To measure Jitter, we take the difference between samples, then divide by the number of samples (minus 1). |

**Architecture Diagram:**



**Socket Programming:**

**List of operations:**

**For Toll:**

1)Entering data

2)Checking the no. of vehicles passing through each toll location

3)Total Fee collected

**For Checkpost:**

1)Entering Data

2) No. of cars passed through each checkpoint:

3) No. of people passing through the chechpost each day:

**Server:**

*import* socket

*import* random

*import* pandas *as* pd

*import* numpy *as* np

*import* string

*from* pandas.core.indexes.base *import* Index

host=socket.gethostname()

port=19008

s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)

data=pd.DataFrame(pd.read\_csv('toll.csv'))

data.set\_index(['Vehicle-no.','Location','Time','Amount','Way','Highway'])

data1=pd.DataFrame(pd.read\_csv('check.csv'))

data1.set\_index(['Vehicle','Time Entered','Time Exited','Location','Date Entered','Date Exited'])

def serverBrain(clientsocket,addr,host,data,data1):

    x="--------------Welcome to Highway Control room---------------------"

    clientsocket.send(x.encode())

    ques=clientsocket.recv(1024).decode()

    print(ques," \nConnected to client")

    ques='Choose a department :\n 1. Toll\n 2.Checkpost'

    clientsocket.send(ques.encode())

    ans=clientsocket.recv(1024).decode()

    intAns=int(ans)

*if*(intAns==1):

        ques='Select an operation :\n 1.Enter Details  \n2.No of cars passed through each toll \n3. Total amount Collected'

        clientsocket.send(ques.encode())

        ans=clientsocket.recv(1024).decode()

        intAns=int(ans)

*if*(intAns==1):

            s=[]

            enterData="Vehicle-no.: "

            clientsocket.send(enterData.encode())

            vehicle=clientsocket.recv(1024).decode()

            s.append(vehicle)

            enterData="Location: "

            clientsocket.send(enterData.encode())

            loc=clientsocket.recv(1024).decode()

            s.append(loc)

            enterData="Time: "

            clientsocket.send(enterData.encode())

            tiem=clientsocket.recv(1024).decode()

            s.append(tiem)

            enterData="Amount: "

            clientsocket.send(enterData.encode())

            amt=clientsocket.recv(1024).decode()

            s.append(amt)

            enterData="Way: "

            clientsocket.send(enterData.encode())

            way=clientsocket.recv(1024).decode()

            s.append(way)

            enterData="Highway: "

            clientsocket.send(enterData.encode())

            hw=clientsocket.recv(1024).decode()

            s.append(hw)

            n=len(data)

            data.loc[n]=s

            i=data.to\_string()

            clientsocket.send(i.encode())

            data.to\_csv('toll.csv',index=False)

*elif*(intAns==2):

            x=data['Location'].value\_counts()

            x=x.to\_string()

            clientsocket.send(bytes(x,"utf-8"))

*elif*(intAns==3):

            x=data['Amount'].sum()

            x=str(x)

            clientsocket.send(bytes(x,"utf-8"))

*elif*(intAns==2):

        ques='Select an operation :\n 1.Enter Details \n2. No of cars passed through each checkpoint \n3.no. of people passed through each day'

        clientsocket.send(ques.encode())

        ans=clientsocket.recv(1024).decode()

        intAns=int(ans)

*if*(intAns==1):

            u=[]

            enterData="Vehicle-no.: "

            clientsocket.send(enterData.encode())

            vehicle=clientsocket.recv(1024).decode()

            u.append(vehicle)

            enterData="Time Entered: "

            clientsocket.send(enterData.encode())

            te=clientsocket.recv(1024).decode()

            u.append(te)

            enterData="Time Exited: "

            clientsocket.send(enterData.encode())

            tex=clientsocket.recv(1024).decode()

            u.append(tex)

            enterData="Location: "

            clientsocket.send(enterData.encode())

            loc=clientsocket.recv(1024).decode()

            u.append(loc)

            enterData="Date Entered: "

            clientsocket.send(enterData.encode())

            de=clientsocket.recv(1024).decode()

            u.append(de)

            enterData="Date Exited: "

            clientsocket.send(enterData.encode())

            dex=clientsocket.recv(1024).decode()

            u.append(dex)

            n=len(data)

            data.loc[n]=u

            i=data.to\_string()

            clientsocket.send(i.encode())

            data.to\_csv('check.csv',index=False)

*elif*(intAns==2):

            x=data1['Location'].value\_counts()

            x=x.to\_string()

            clientsocket.send(bytes(x,"utf-8"))

*elif*(intAns==3):

            x=data1['Date Entered'].value\_counts()

            y=data1['Date Exited'].value\_counts()

            a="No.of people entered during a particular day\n"

            clientsocket.send(a.encode())

            x=x.to\_string()

            y=y.to\_string()

            clientsocket.send(bytes(x,"utf-8"))

            b="No.of people exited during a particular day\n"

            clientsocket.send(b.encode())

            clientsocket.send(bytes(y,"utf-8"))

*else*:

        errorMsg="Wrong Option! Please Select any of the above options only"

        clientsocket.send(errorMsg.encode())

    clientsocket.close()

def initiate():

    s.bind((host,port))

    s.listen(5)

    x=True

*while*(x):

        c,addr=s.accept()

        serverBrain(c,addr,host,data,data1)

        ch=input("\nDo You want to continue(y/n): ")

*if*(ch=='y'):

*pass*

*else*:

            x=False

print('----------------Server Initiated !-----------------')

initiate()

**Client:**

*import* threading

*import* socket

sck=socket.socket()

host=socket.gethostname()

port=19008

sck.connect((host,port))

print(sck.recv(1024).decode())

sck.send(bytes("-----Client Initiated-----","utf-8"))

print(sck.recv(1024).decode())

ch=input()

*if*(ch=="1"):

    sck.send(bytes(ch,"utf-8"))

    rply=sck.recv(1024).decode()

    print(rply)

    ch=input()

*if*(ch=="1"):

        sck.send(bytes(ch,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        vno=input()

        sck.send(bytes(vno,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        loc=input()

        sck.send(bytes(loc,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        t=input()

        sck.send(bytes(t,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        a=input()

        sck.send(bytes(a,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        w=input()

        sck.send(bytes(w,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        hw=input()

        sck.send(bytes(hw,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

*elif*(ch=="2"):

        sck.send(bytes(ch,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

*elif*(ch=="3"):

        sck.send(bytes(ch,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

*elif*(ch=='2'):

    sck.send(bytes(ch,"utf-8"))

    rply=sck.recv(1024).decode()

    print(rply)

    ch=input()

*if*(ch=="1"):

        sck.send(bytes(ch,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        vno=input()

        sck.send(bytes(vno,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        te=input()

        sck.send(bytes(te,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        tex=input()

        sck.send(bytes(tex,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        loc=input()

        sck.send(bytes(loc,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        de=input()

        sck.send(bytes(de,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

        dex=input()

        sck.send(bytes(dex,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

*elif*(ch=="2"):

        sck.send(bytes(ch,"utf-8"))

        rply=sck.recv(1024).decode()

        print(rply)

*elif*(ch=="3"):

        sck.send(bytes(ch,"utf-8"))

        rply=sck.recv(1024).decode()

        rply1=sck.recv(1024).decode()

        rply2=sck.recv(1024).decode()

        rply3=sck.recv(1024).decode()

        print(rply ,'\n',rply1,'\n',rply2,'\n',rply3)

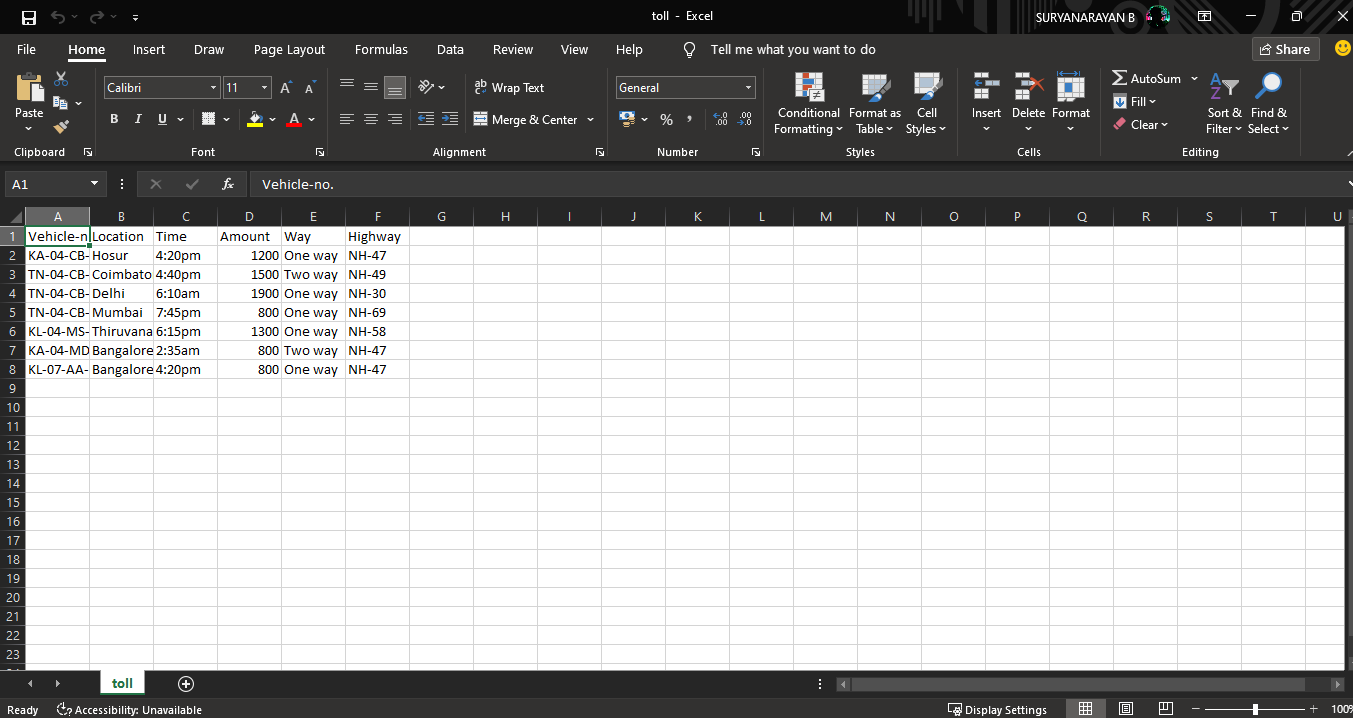
*else*:

*pass*

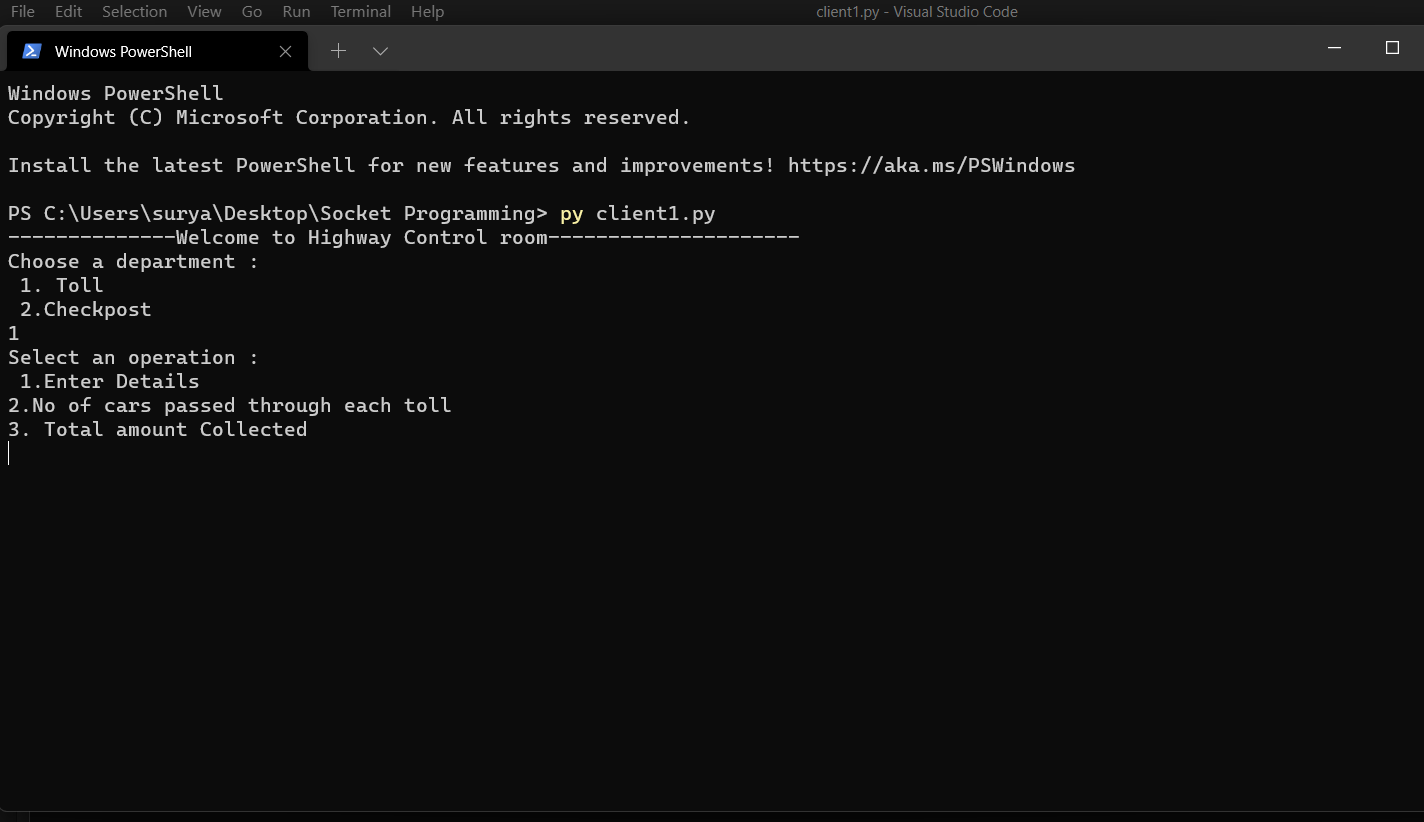
sck.close()

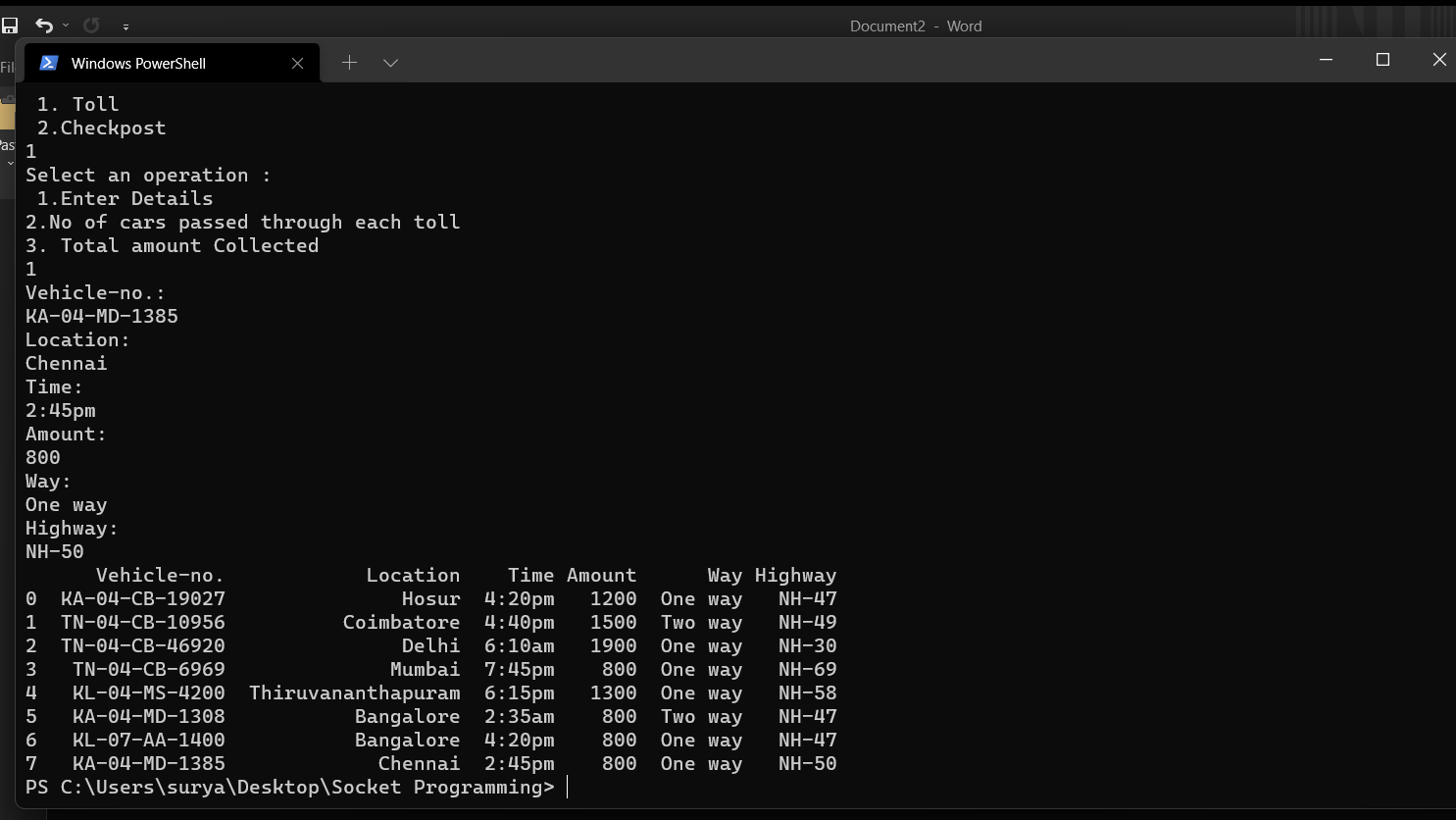
**Output:**

**Toll:**

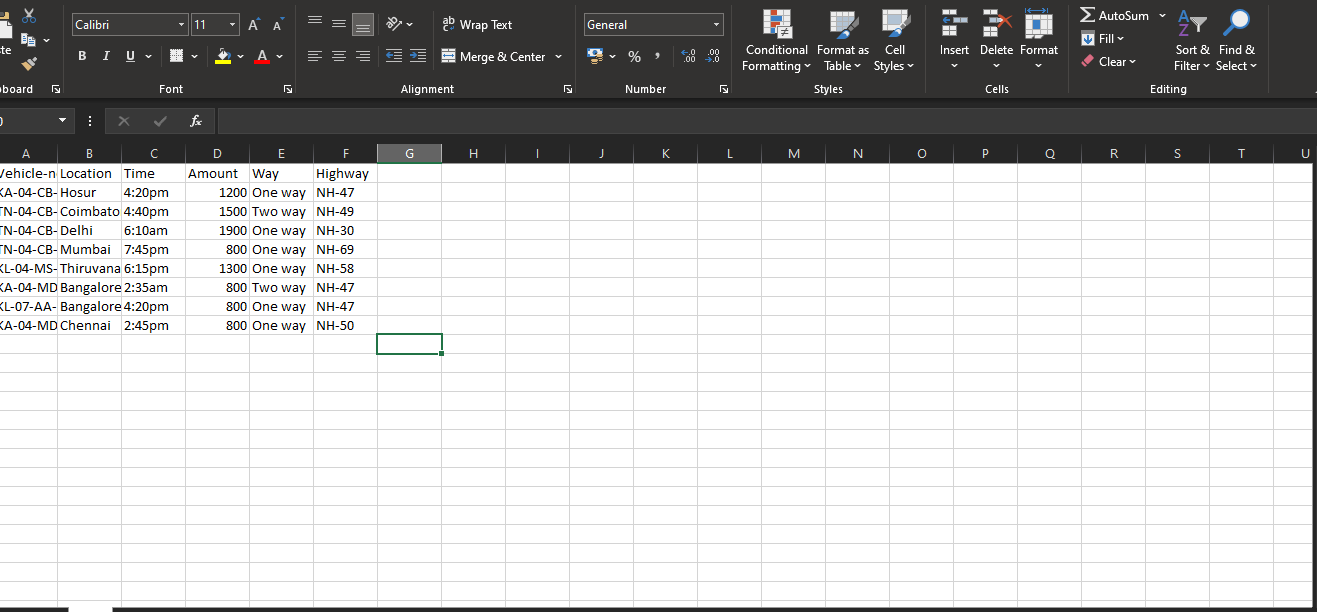
Toll csv before input:

Toll client:





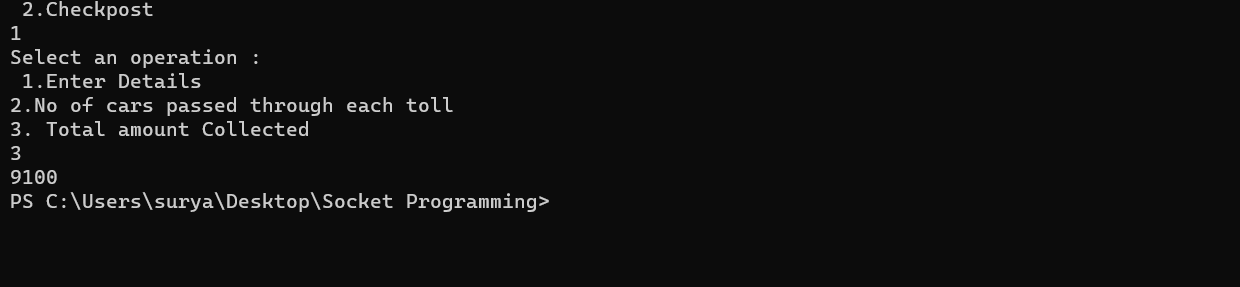
Csv after Entry:



No. of cars passed through each toll:

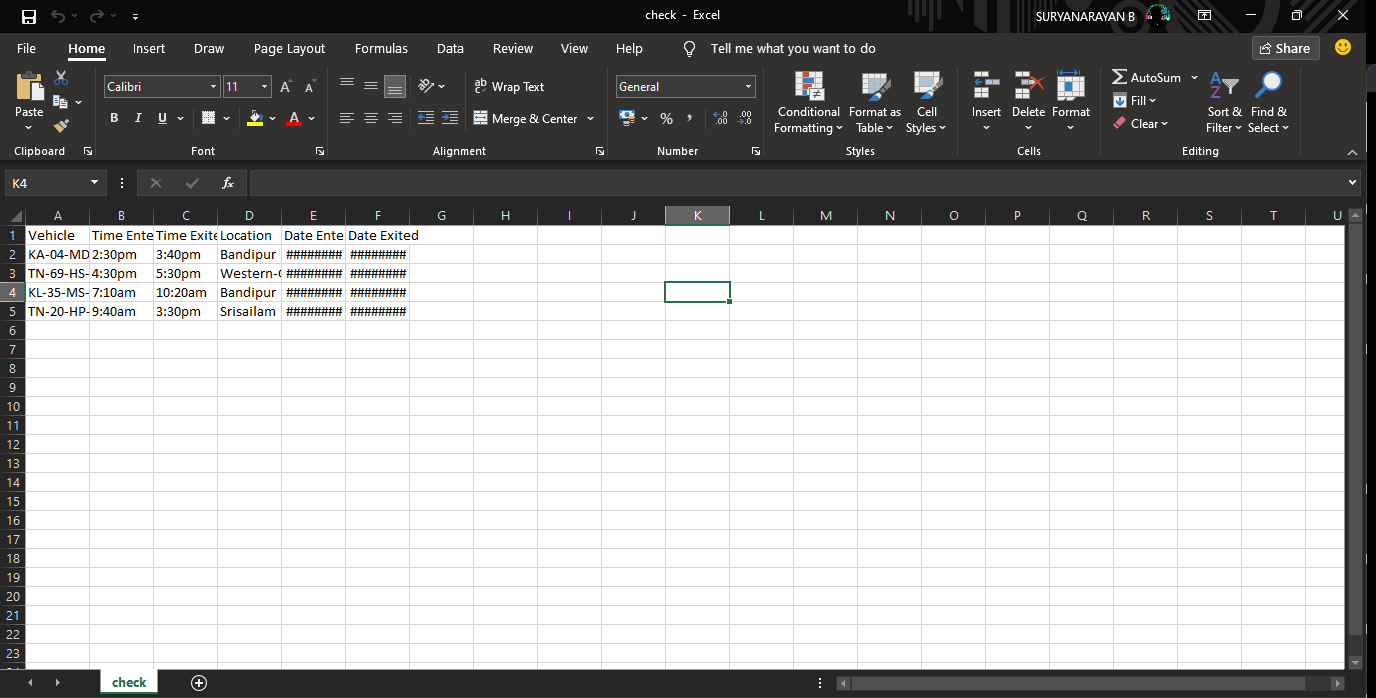


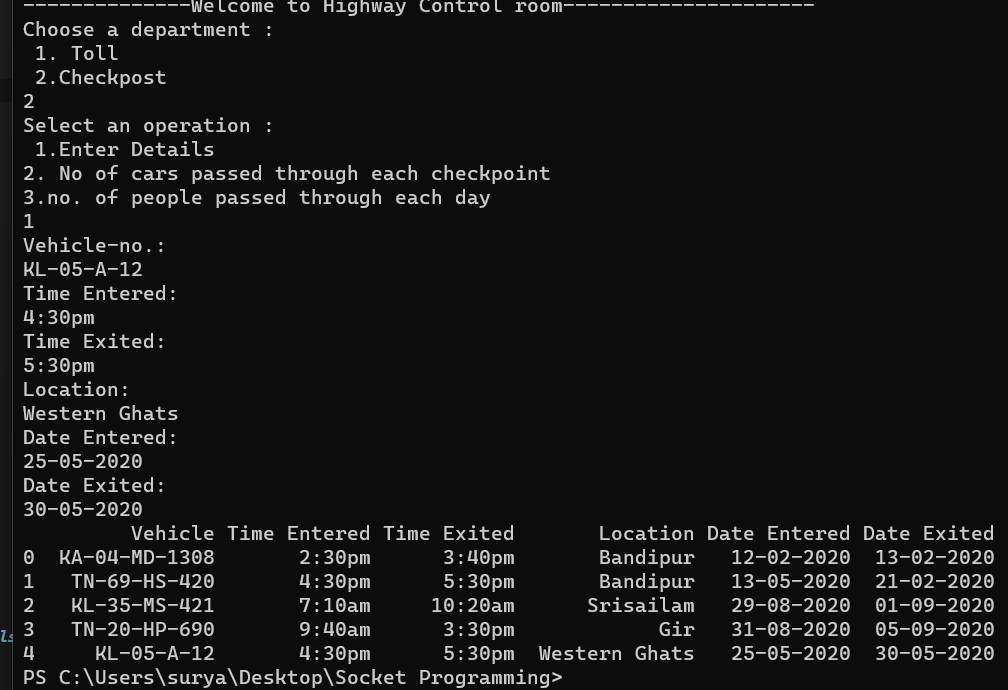
Total Amount collected:

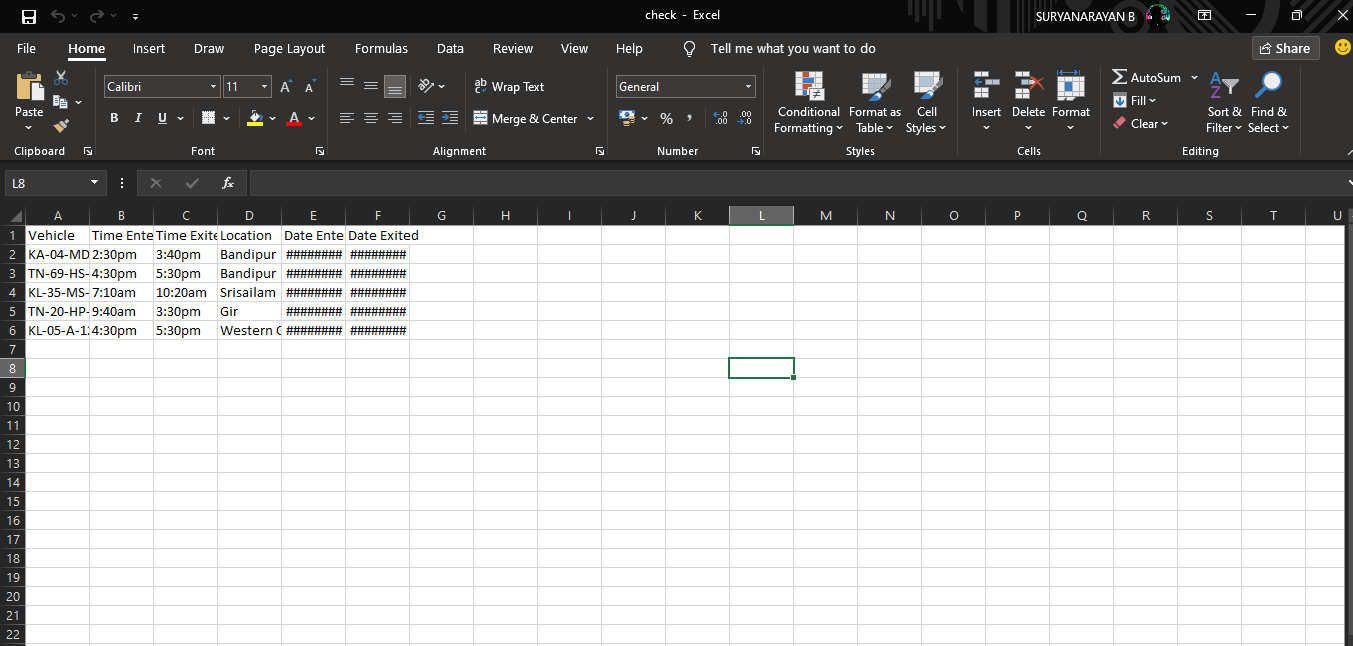


**Checkpoint:**

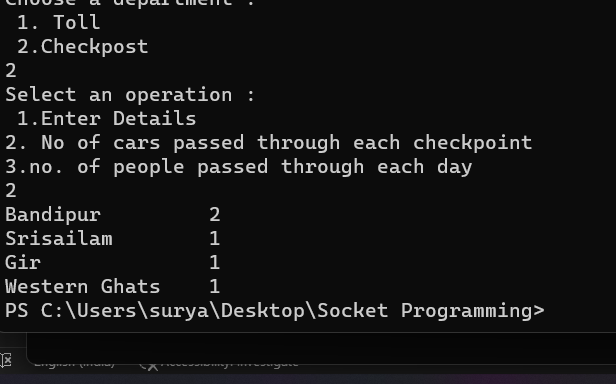
Checkpoint csv before input:

****

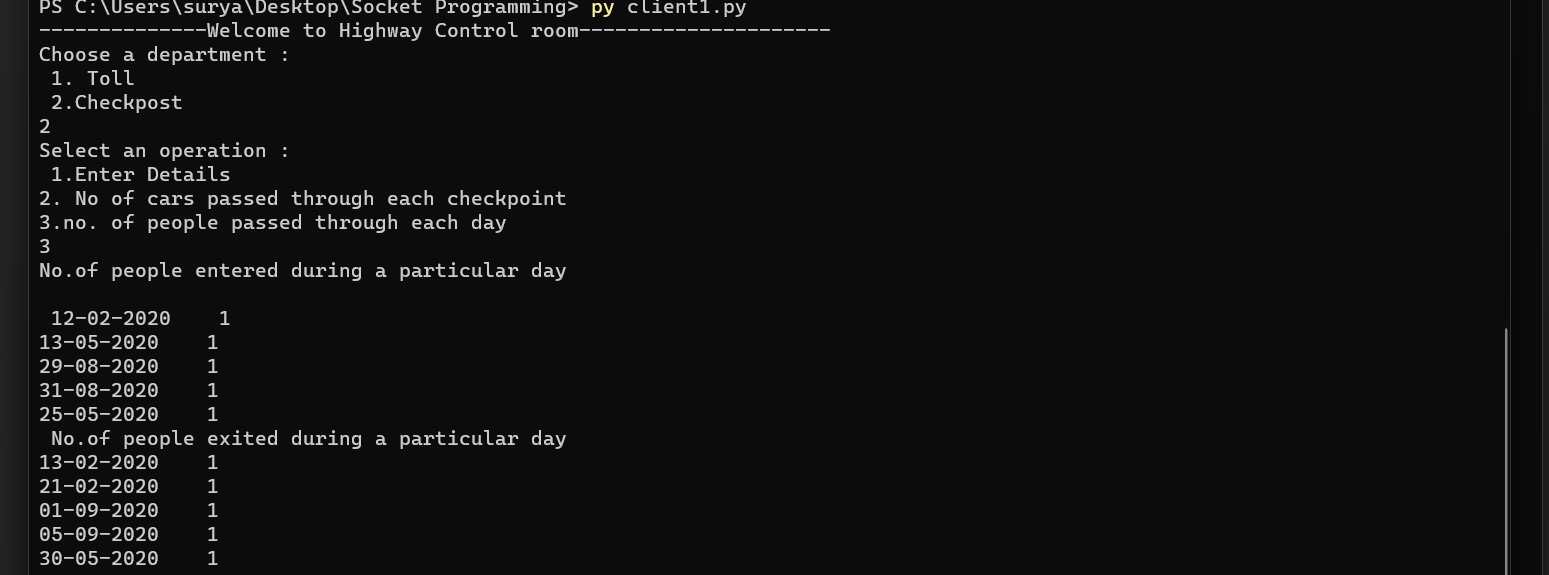
Checkpoint client:Csv after Modification:



No. of cars passed through each checkpoint:

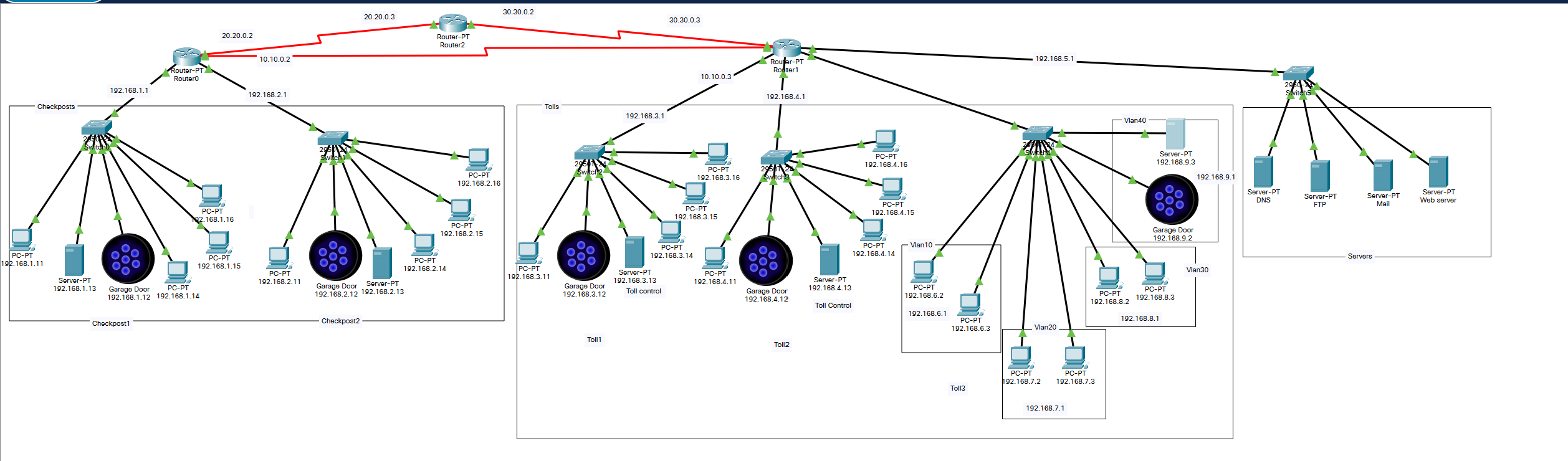


No. of people passing through the chechpost each day:

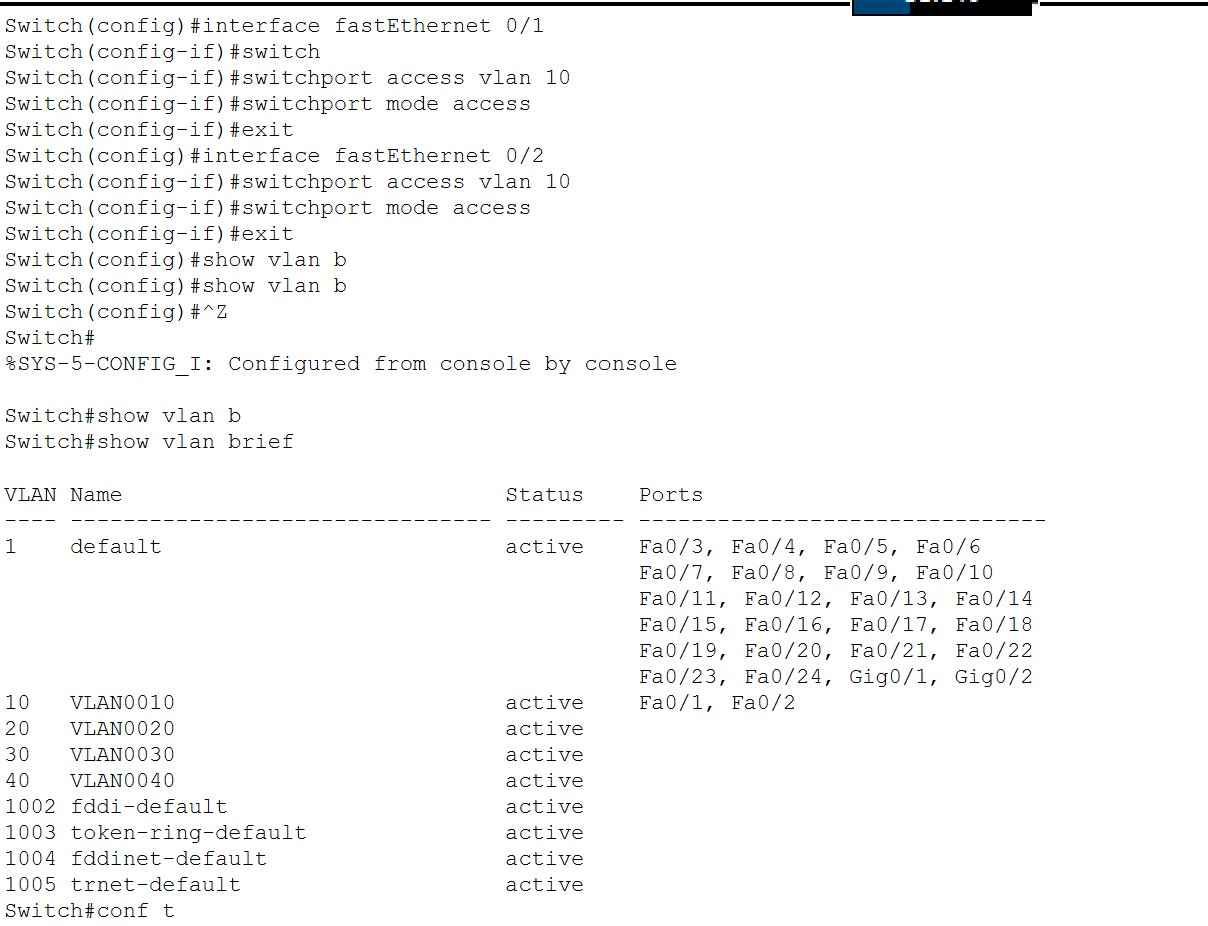


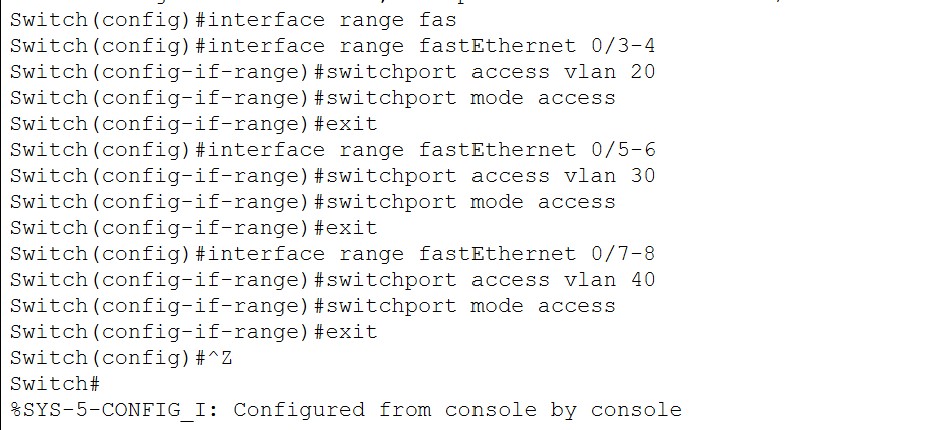
**Cisco Packet Tracer**

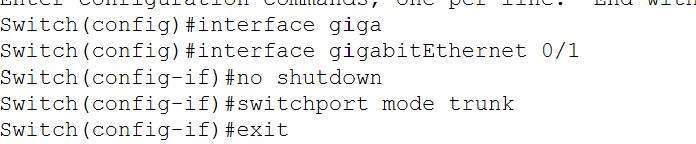
Checkpost and Toll Model

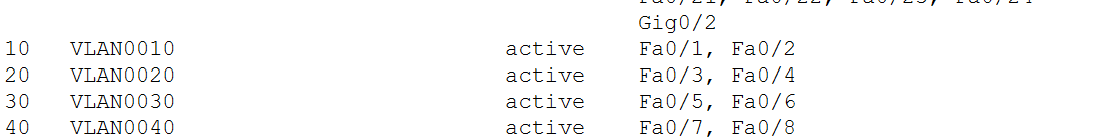


Vlan:

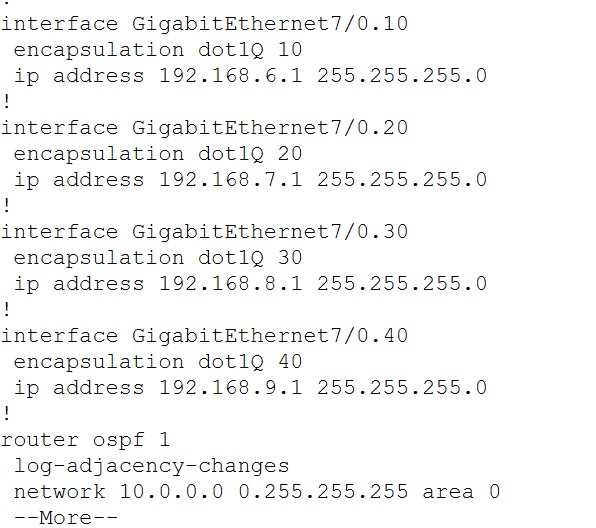


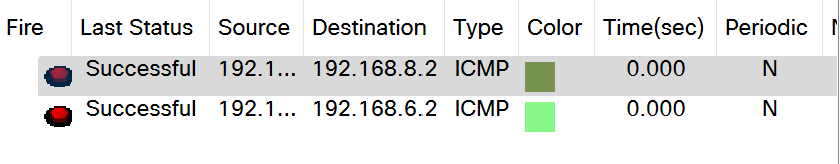






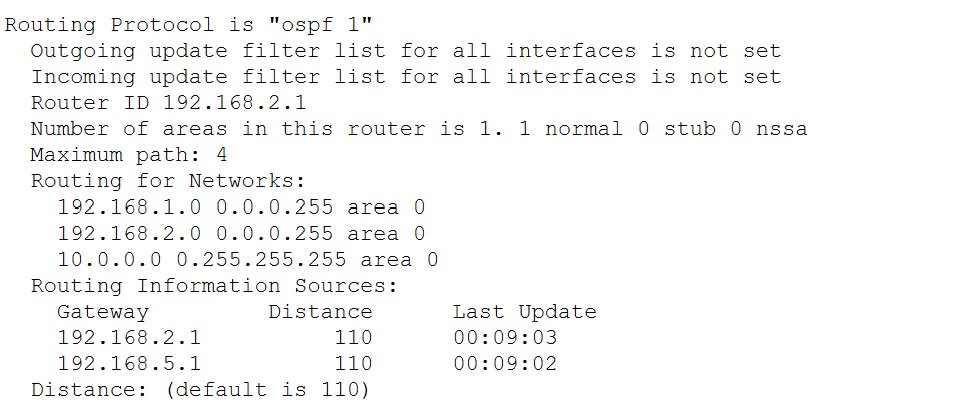
Router:



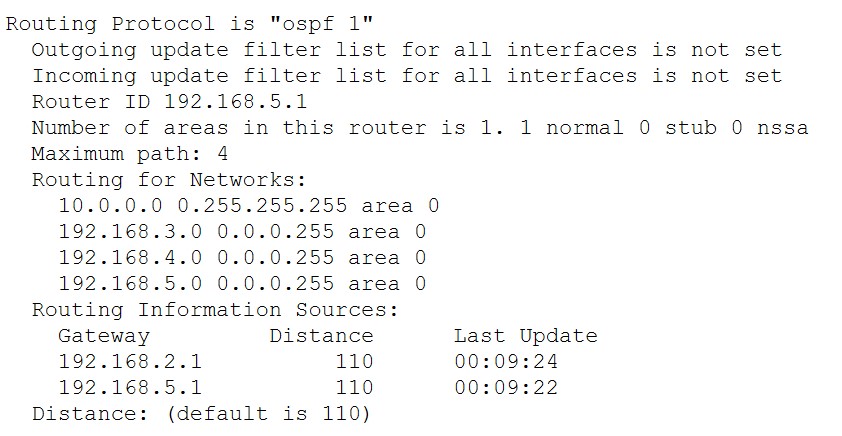


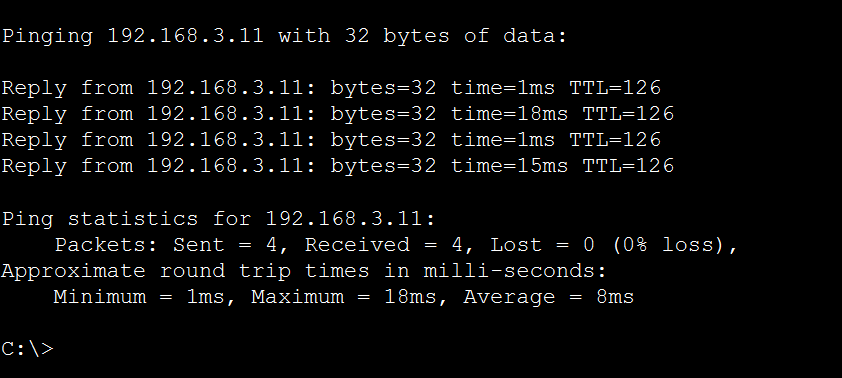
OSPF:

Router 1

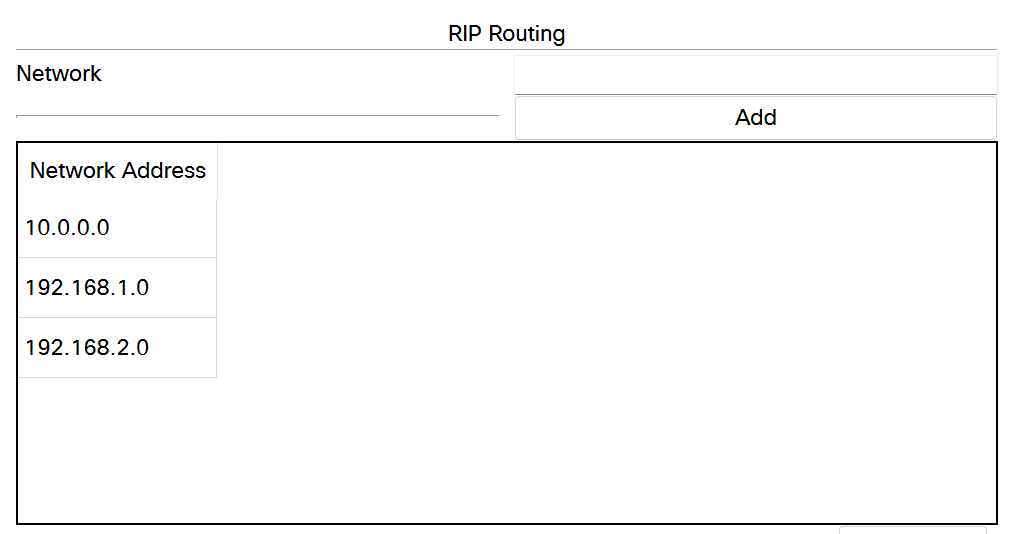


Router 2

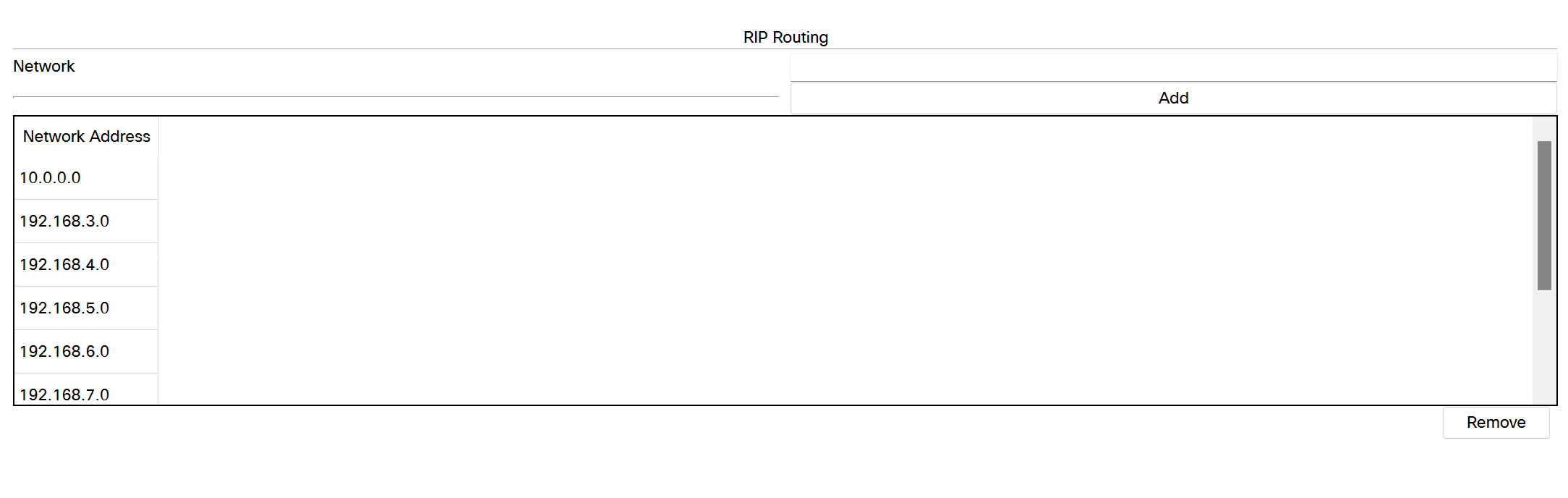


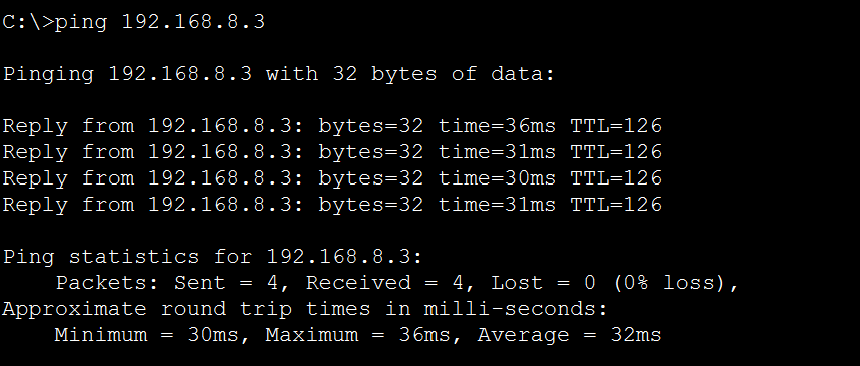


RIP:

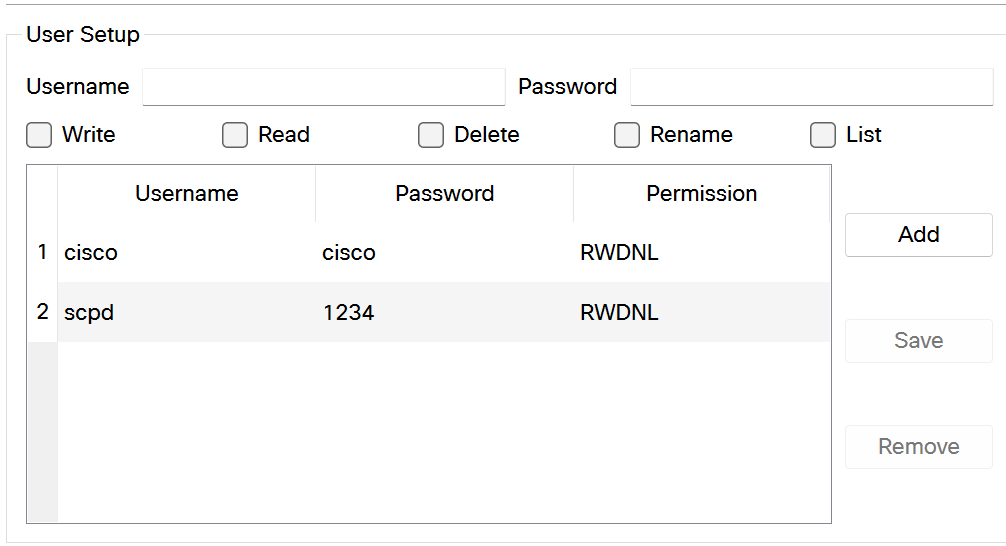
Router 1  


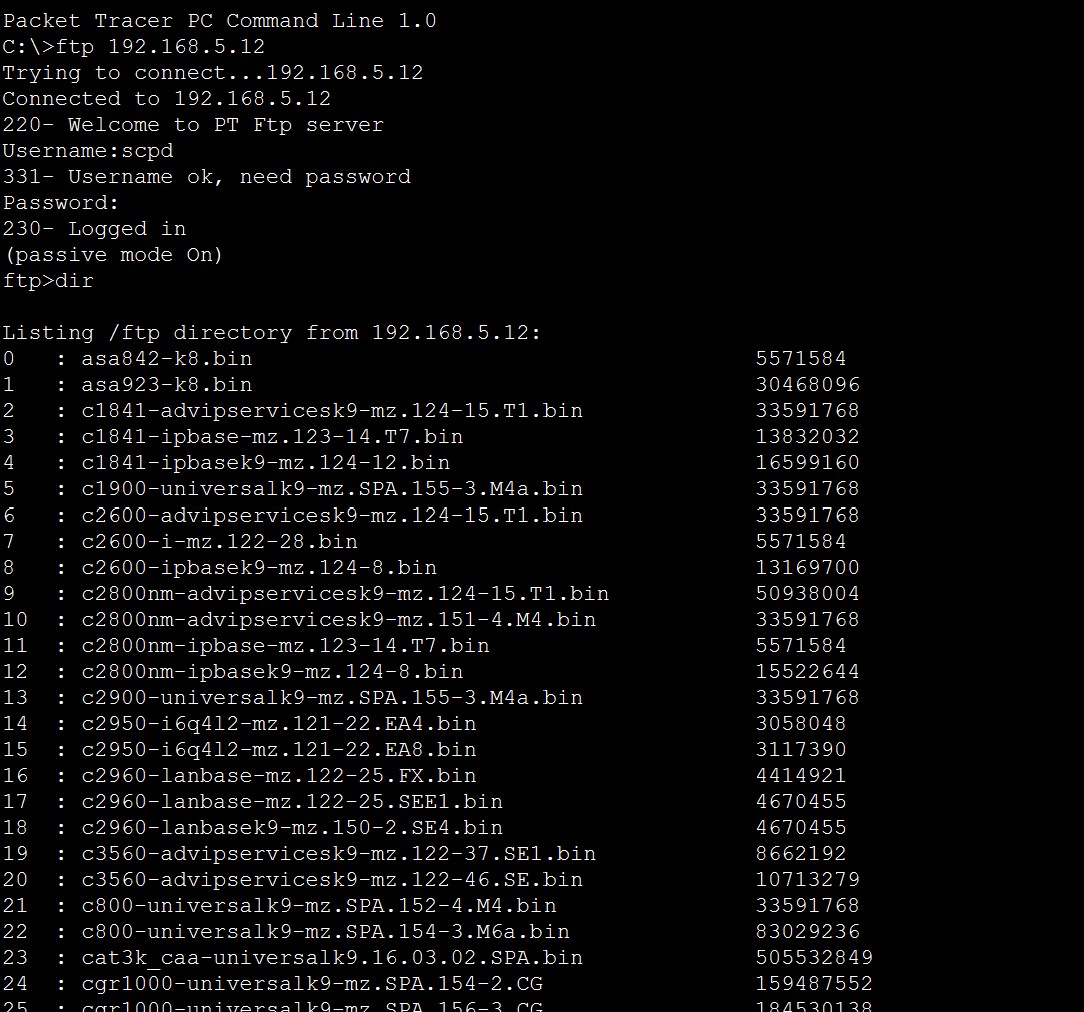
Router 2:



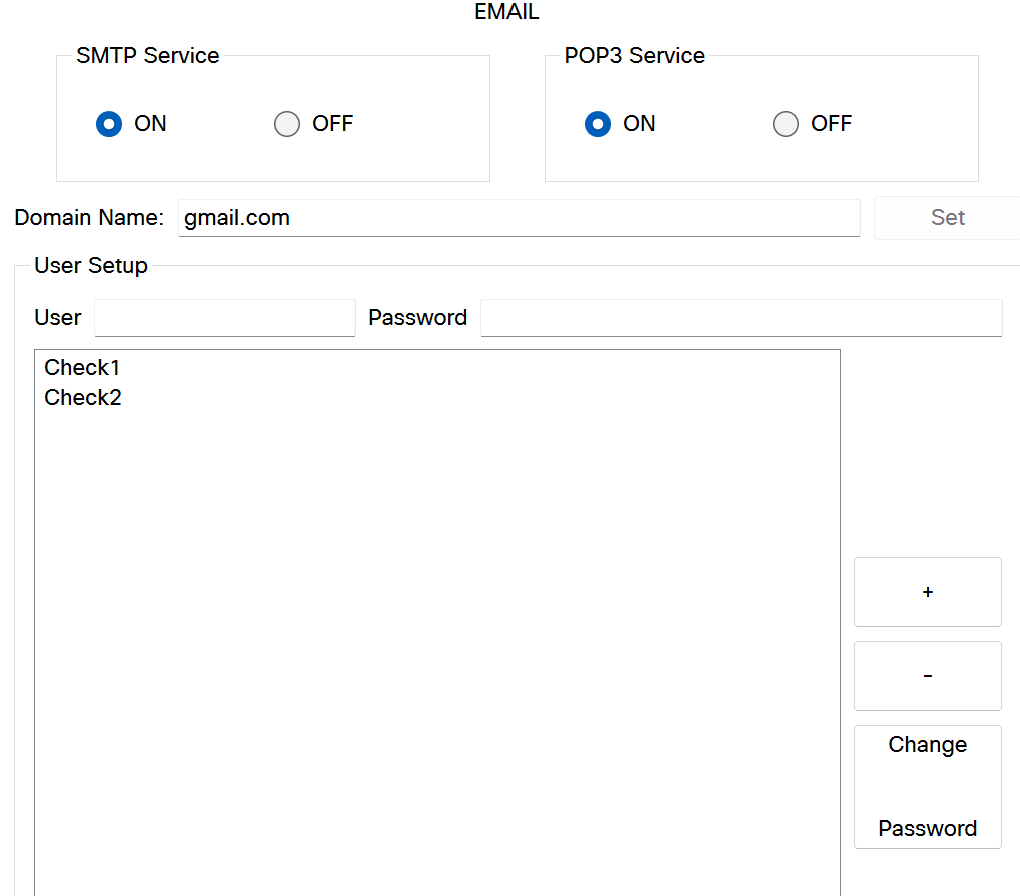


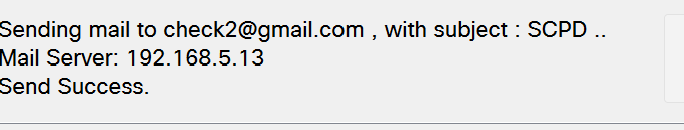
FTP:

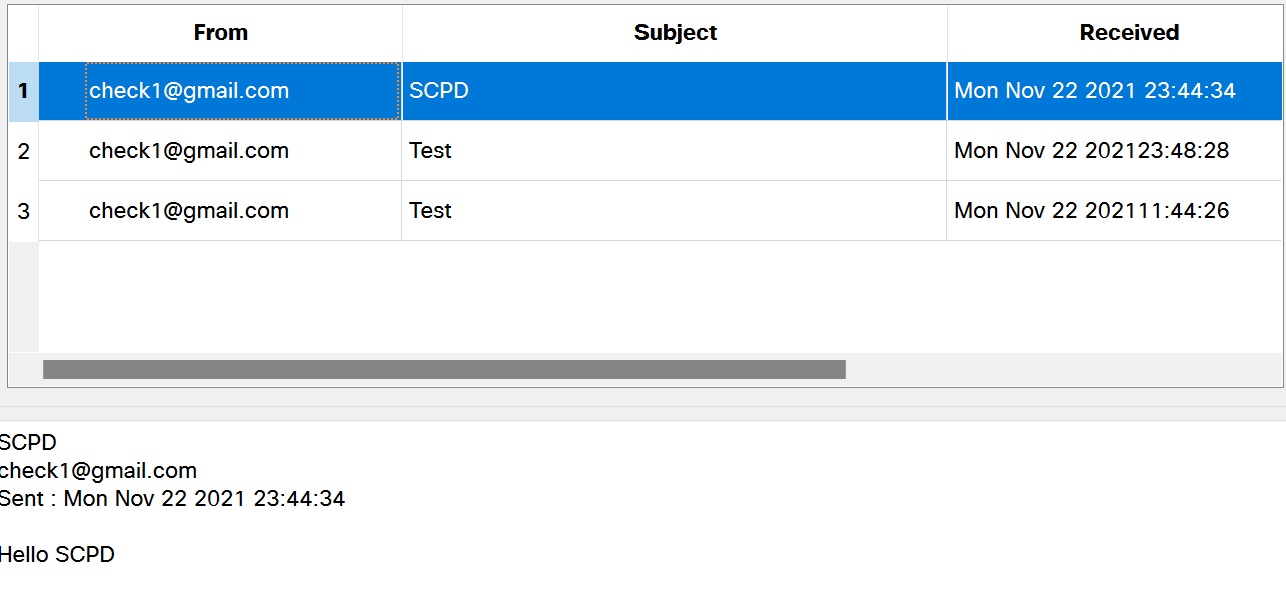




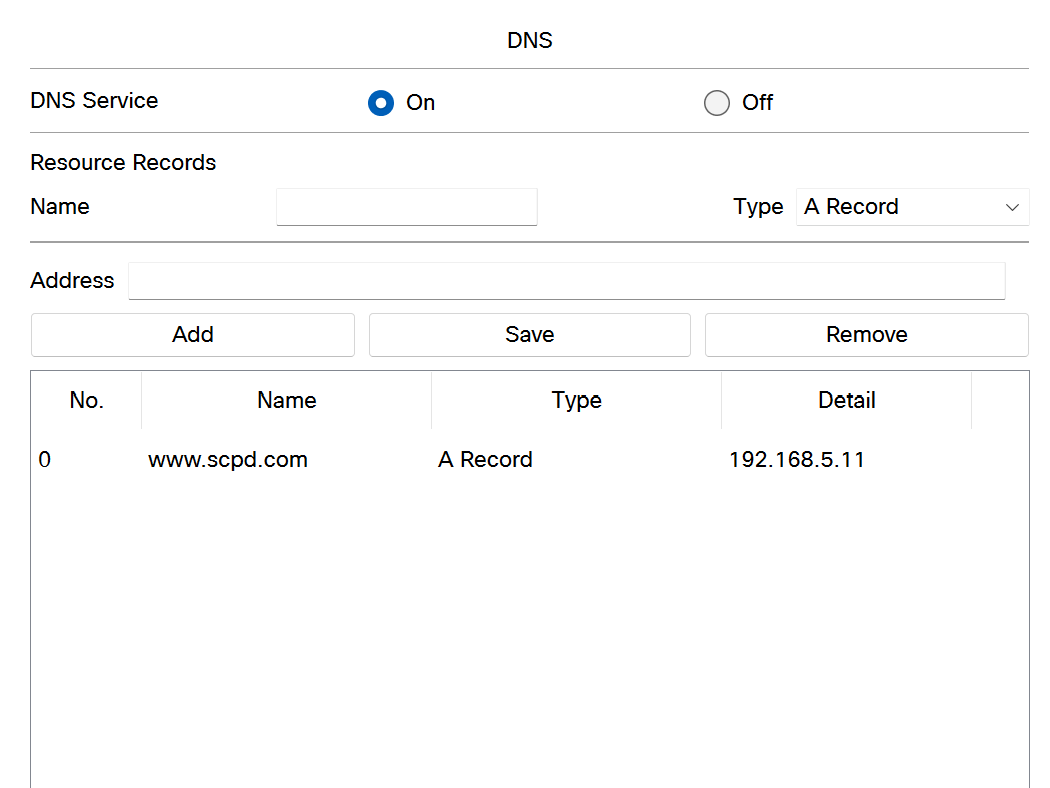
Mail Server:

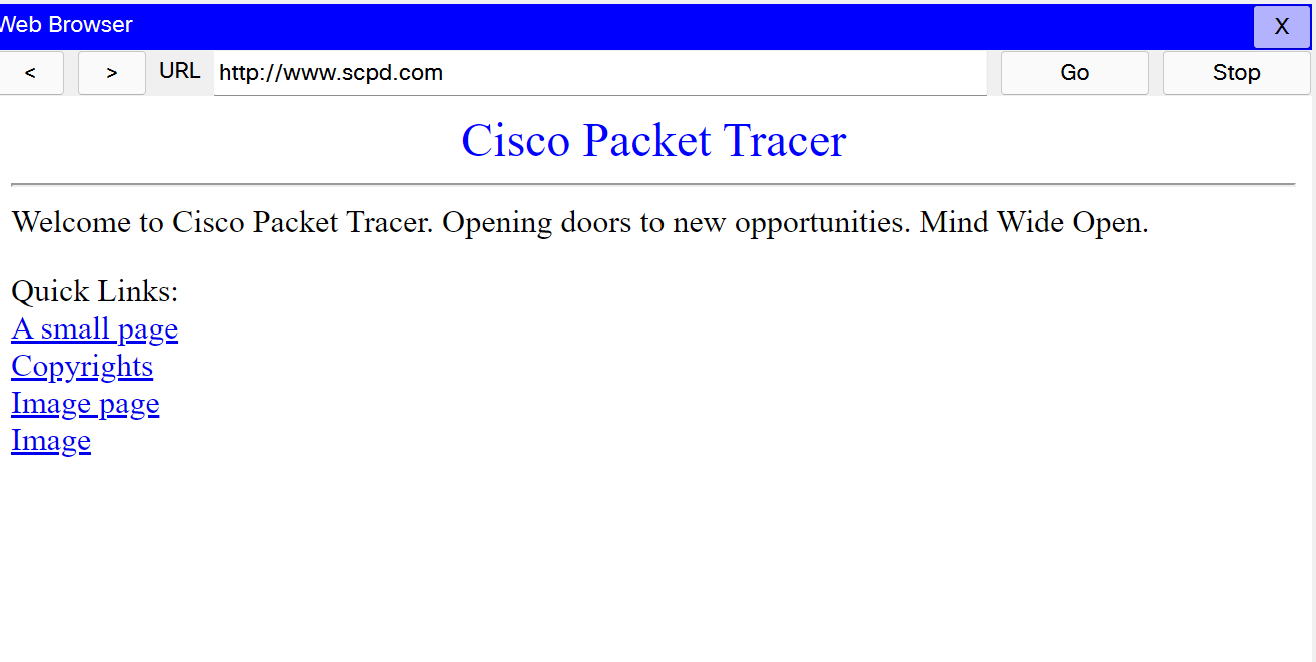






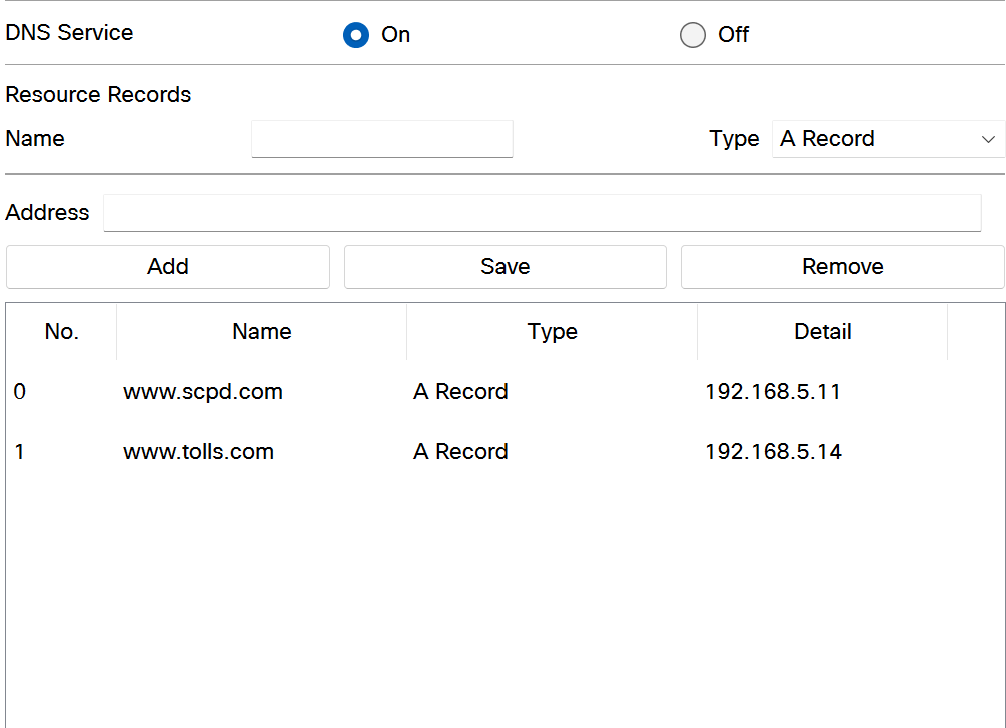
DNS:

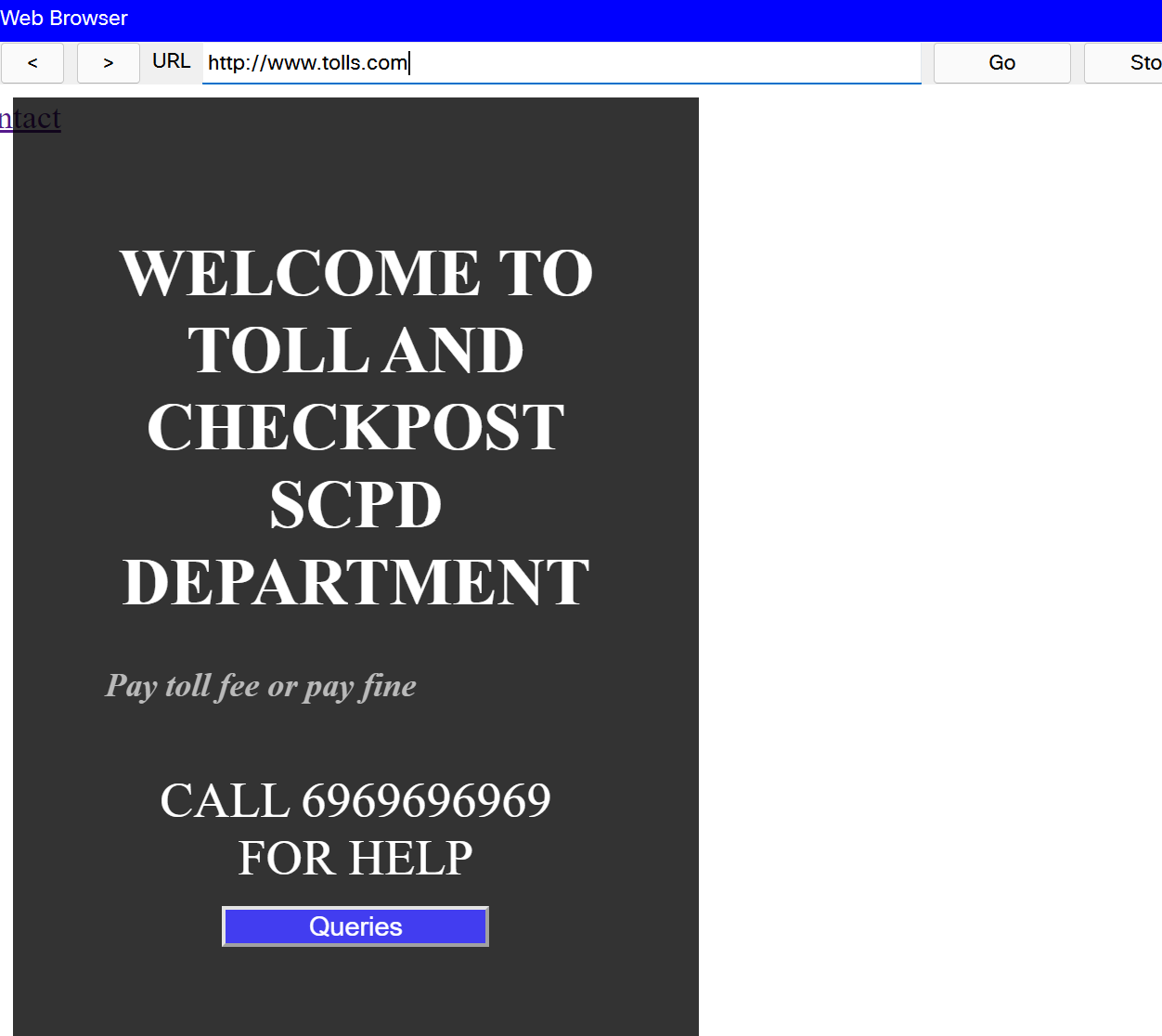




**Web Server:**

**Configured Web server:**

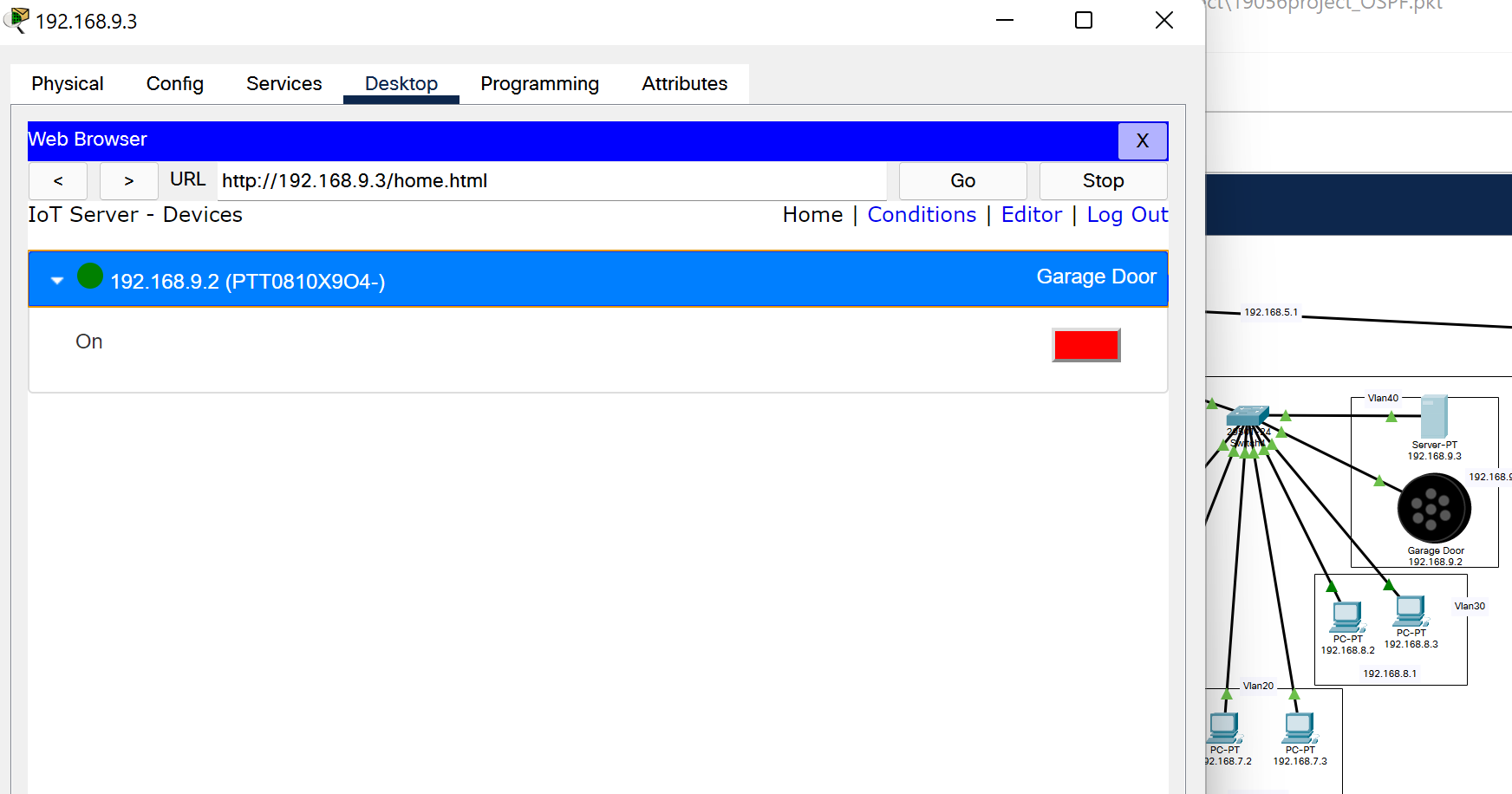
****

**Output:**

**IOT Devices:**

**To show the closing and opening of Toll and Checkpost Gates**

**Gate Closed:**



Gate Open:

