Laboratorio – Esame 25 Giugno 2022

Docenti: Proff. A. Capone, M. Cesana, G. Maier, F. Musumeci

Cognome	**BRAVO**
Nome	**STUDENTE**
Matricola	

Python (3 punti)

Il codice sotto riportato implementa un'applicazione Client-Server basata sul protocollo di trasporto TCP.
Il Server implementa una versione multi-threaded di un *supremo CHATBOT*, ascoltando le richieste dei vari client.
Il client dell'applicazione dopo essersi connesso può inviare dei messaggi al Server, che risponde con un messaggio di risposta generato dopo una fase di processing, che per semplicità è riportata nella funzione *process_request*. **(3 punti)**

Client

```
from socket import *
serverName = 'localhost'
serverPort = 14000
clientSocket = socket(AF_INET,SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
while True:
    request = input('Inserisci una richiesta per il supremo CHATBOT')
    clientSocket.send(request.encode('utf-8'))
    if request == 'arrivederci':
        break
    response = clientSocket.recv(1024)
    print('Il supremo CHATBOT dice:', response.decode('utf-8')
clientSocket.close()
```

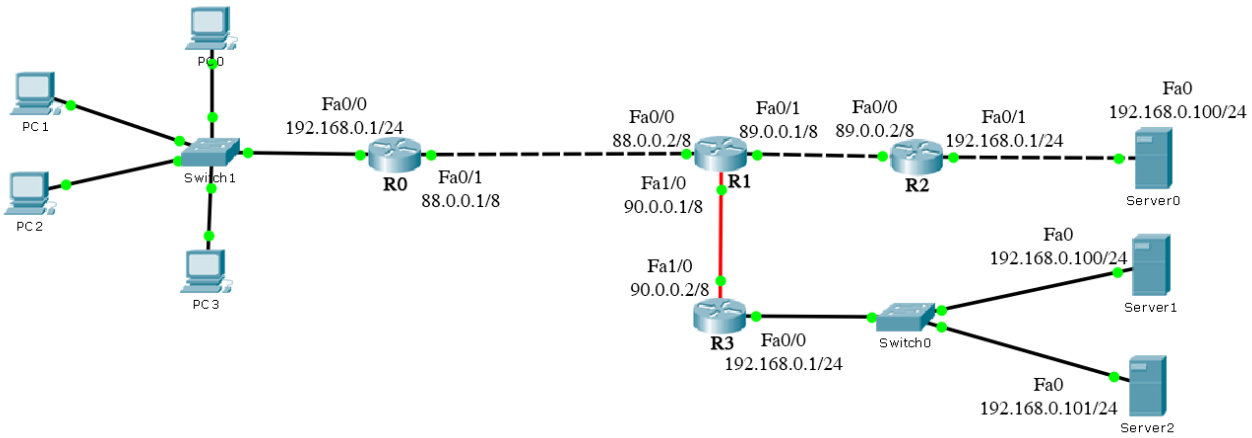
Server

```
from socket import *
from threading import Thread

def process_request(request):
    # Questa funzione processa la stringa request (mediante sofisticati
    # meccanismi di intelligenza artificiale) e produce in risposta
    # la stringa response che ritorna come valore
    return response

def handler(connectionSocket):
    while True:
        richiesta = connectionSocket.recv(1024)
        richiesta = richiesta.decode('utf-8')
        if richiesta == 'arrivederci':
            break
        risposta = process_request(richiesta)
        connectionSocket.send(risposta.encode('utf-8'))
    connectionSocket.close()


serverPort = 14000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
serverSocket.bind(("",serverPort))
serverSocket.listen(1)
while True:
    newSocket,  addr = serverSocket.accept()
    thread = Thread(target=handler, args=(newSocket,))
    thread.start()
```

Q1. (2.5pt) Si completino i codici di Client e Server.
Q2. (0.5pt) Quale messaggio deve inviare il Client per terminare la sessione?

Packet Tracer (3 punti)

Si consideri la rete in figura e il suo piano di indirizzamento.
Si considerino pubbliche le reti 88.0.0.0/8, 89.0.0.0/8 e 90.0.0.0/8.
Si considerino private tutte le altre reti.
NB: Indicare sempre prima del comando il prompt visualizzato dal sistema, prestando attenzione alla modalità di partenza in ciascun quesito.



Q1) Configurare il **DHCP** in R0 in modo tale da assegnare ai 4 PC (PC0-PC3) un indirizzo appartenente alla rete 192.168.0.0/24. *(1 punto)*

```
R0> enable
R0# configure terminal
R0 (config)#ip dhcp pool 1
R0 (dhcp-config)# default-router 192.168.0.1
R0 (dhcp-config)# network 192.168.0.0 255.255.255.0
```

Q2) Configurare su R3 una **rotta statica** per raggiungere l'interfaccia pubblica di R2. *(1 punto)*

```
R3>enable
R3# configure terminal
R3(config)# ip route 89.0.0.0 255.0.0.0 90.0.0.1
```

Q3) Configurare il **NAT** su R2 per permettere al Server0 di comunicare con la rete pubblica. *(1 punto)*

```
R2(config)# interface Fa0/0
R2(config-if)# ip nat outside
R2(config-if)# exit
R2(config)# interface Fa0/1
R2(config-if)# ip nat inside
R2(config-if)# exit
R2(config)# access-list 1 permit  192.168.0.  0 0.0.0.255
R2(config)# ip nat inside source list 1 interface Fa0/0 overload
```