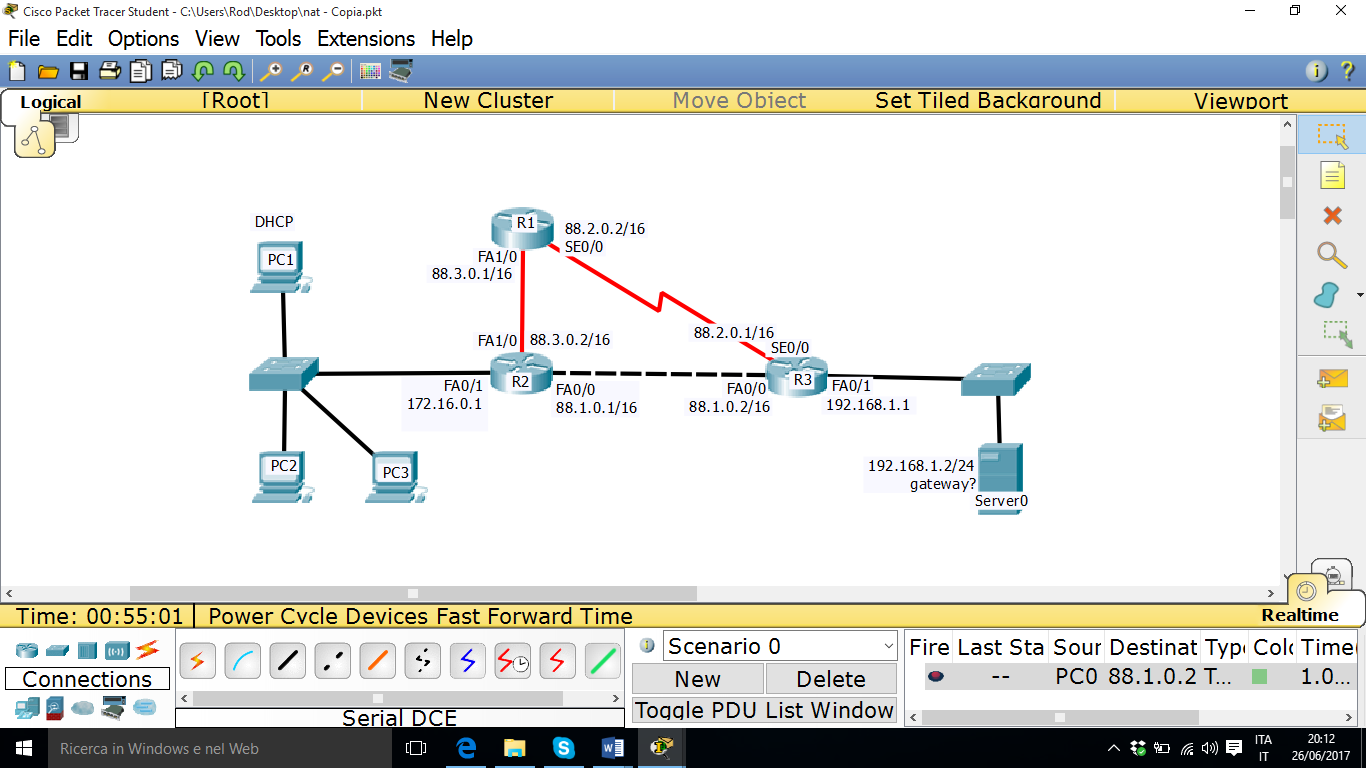
*Laboratorio PKT tracer*

Si consideri la rete in figura e il suo piano di indirizzamento.



**Q1)** Configurare il protocollo **RIP versione 2** sul router **R2**, abilitare solo l’annuncio delle reti PUBLICHE ad esso connesse, e configurare come **passiva** la interfaccia verso la rete privata ad esso connesse.

(NB: Scrivere in modo esplicito la modalità del router in cui deve essere eseguito ogni comando)

R2> enable

R2# configure terminal

R2(config)#router RIP

R2(config-router)#version 2

R2(config-router)#network 88.1.0.0

R2(config-router)#network 88.3.0.0

R2(config-router)#passive-interface fa0/1

**Q2)** Configurare NAT su **R2** per permettere l'accesso a Internet attraverso FA1/0 dalla rete privata

**R3(config)# interface fa0/0**

**R3(config-if)# ip nat outside**

**R3(config-if)# exit**

**R3(config)# interface fa0/1**

**R3(config-if)# ip nat inside**

**R3(config-if)# exit**

**R3(config)# interface fa1/0**

**R3(config-if)# ip nat outside**

**R3(config-if)# exit**

**R3(config)# access-list** 1 **permit** **172.16.0.1** **0.0.255.255**

**R3(config)# ip nat inside source list 1 interface** **FastEthernet1/0** **overload**

***Laboratorio Socket programming***

Si vuole scrivere un'applicazione client/server per emulare l’applicazione PING utilizzando socket UDP, e per calcolare il RTT e la percentuale di perdita di pacchetti nel collegamento fra client e server. Si ipotizzi che il server, una volta ricevuta la ECHO Request proveniente dal client, estragga un numero casuale (rand) e decida di rispondere al PING con una ECHO REPLY SOLO nel caso in cui il numero estratto sia maggiore di 9.

**Q1) Completare lo script "UDP server"**

**Q2) Quanti PING sono scambiati prima di chiudere la connessione? \_\_\_\_\_\_\_\_\_\_\_\_**10

**UDP client**

import socket

import time

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_addr = ('localhost', 12000)

sock.settimeout(1)

try:

for i in range(1, 11):

start = time.time()

message = 'Ping #' + str(i) + " " + time.ctime(start)

try:

sent = sock.sendto(message, server\_addr)

print("Sent " + message)

data, server = sock.recvfrom(4096)

print("Received " + data)

end = time.time();

elapsed = end - start

print("RTT: " + str(elapsed) + " seconds\n")

except socket.timeout:

print("#" + str(i) + " Requested Time out\n")

finally:

print("closing socket")

sock.close()

**UDP server**

import random

from socket import \*

serverSocket = socket(AF\_INET, SOCK\_DGRAM)

serverSocket.bind(('', 12000))

while True:

message, address = serverSocket.recvfrom(1024)

message = message.upper()

rand = random.randint(0, 20)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ if rand < 9:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ continue

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ serverSocket.sendto(message, address)