**Esame Completo - 12 Settembre 2017**

|  |  |
| --- | --- |
| **Cognome** |  |
| **Nome** |  |
| **Matricola** |  |

##### Tempo complessivo a disposizione per lo svolgimento: 2 ore

***Si usi lo spazio bianco dopo ogni esercizio per la risoluzione***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **E1** | **E2** | **E3** | **Quesiti** | **Lab** |
|  |  |  |  |  |

## Esercizio 1(8 punti)

Un router ha le seguenti interfacce e la seguente tabella di routing. Riceve i pacchetti con destinazione, dimensioni e impostazione del bit “Don’t Fragment” indicati sotto. Si dica per ciascuno di essi come si comporta il router: inoltro diretto o indiretto, interfaccia di uscita, riga della tabella, motivazione pacchetto scartato.

Eth0: Address: 131.175.21.254 – Netmask: 255.255.255.128 – MTU: 1500 B

Eth1: Address: 131.175.20.126 – Netmask: 255.255.255.128 – MTU: 1000 B

Eth2: Address: 131.175.20.132 – Netmask: 255.255.255.128 – MTU: 1200 B

|  |  |  |
| --- | --- | --- |
| Network | Netmask | Next-hop |
| 131.175.70.0 | 255.255.254.0 | 131.175.21.133 |
| 131.175.71.128 | 255.255.255.128 | 131.175.21.145 |
| 131.175.72.0 | 255.255.254.0 | 131.175.20.5 |
| 131.175.75.192 | 255.255.255.192 | 131.175.20.250 |
| 0.0.0.0 | 0.0.0.0 | 131.175.20.221 |

1. **131.175.20.133 (1200B, D=1) da Eth0**

Tipo inoltro: DIRETTO Interfaccia di uscita: ETH2 Riga tabella (se necessario)

Eventuale motivo di scarto

1. **131.175.21.27 (1200B, D=1) da Eth0**

Tipo inoltro: INDIRETTO Interfaccia di uscita: ETH2 Riga tabella (se necessario): 5

Eventuale motivo di scarto

1. **131.175.71.122 (1000B, D=1) da Eth1**

Tipo inoltro: INDIRETTO Interfaccia di uscita: ETH0 Riga tabella (se necessario): 1

Eventuale motivo di scarto

1. **131.175.20.133 (1000B, D=1) da Eth2**

Tipo inoltro: Interfaccia di uscita: Riga tabella (se necessario)

Eventuale motivo di scarto

SAREBBE INOLTRO DIRETTO SU ETH2, MA VIENE SCARTATO PERCHE’ INTFC IN = INTFC OUT

1. **131.175.72.72 (1200B, D=0) da Eth0**

Tipo inoltro: INDIRETTO Interfaccia di uscita: ETH1 Riga tabella (se necessario): 3

Eventuale motivo di scarto

1. **255.255.255.255 (500B, D=1) da Eth0**

Tipo inoltro: Interfaccia di uscita: Riga tabella (se necessario)

Eventuale motivo di scarto

PASSATO AI LIVELLI SUPERIORI E NON INOLTRATO PERCHE’ BROADCAST LIMITATO

1. **131.175.71.202 (1000B, D=0) da Eth0**

Tipo inoltro: INDIRETTO Interfaccia di uscita: ETH0 Riga tabella (se necessario): 2

Eventuale motivo di scarto

1. **131.175.75.15 (1000B, D=0) da Eth0**

Tipo inoltro: INDIRETTO Interfaccia di uscita: ETH2 Riga tabella (se necessario): 5

Eventuale motivo di scarto

1. **131.175.75.200 (1200 B, D=1) da Eth2**

Tipo inoltro: INDIRETTO Interfaccia di uscita: ETH2 Riga tabella (se necessario): 4

Eventuale motivo di scarto

1. **131.175.73.255 (1000 B, D=1) da Eth2**

Tipo inoltro: INDIRETTO Interfaccia di uscita: ETH1 Riga tabella (se necessario): 3

Eventuale motivo di scarto

## Esercizio 2 (5 punti)

Una connessione TCP tra l’*host* A e l’*host* H è caratterizzata dai seguenti parametri: lunghezze di *header, ack* e *segmenti di apertura* trascurabili, *link* bidirezionali simmetrici, *MSS* = 1250 Byte, *RCWND* >> *CWND, SSTHRESH* = 10000 Byte.



1. Si calcoli la lunghezza della finestra che permette la trasmissione continua Wc
2. Si calcoli il tempo necessario (da prima dell’apertura della connessione alla ricezione dell’ultimo ACK) a trasferire un file di 50 KByte dall’*host* A all’*host* H
3. Si ripeta il calcolo assumendo che tutti segmenti della quarta finestra vadano persi e il timeout corrispondente sia Tout=100 ms

**SOLUZIONE**

a)

SSHTRESH = 10000 [byte] / 1250 [byte] = 8 MSS

T1 = MSS / C1 = 1250 \* 8 / 20 \* 106 = 0.5 [ms]

T2 = 2 \* T1 = 1 [ms]

T3 = 5 \* T2 = 5 [ms]

RTT = T1 + T2 +T3 + 2\*(tau1 + tau2 + tau3) = 6.5 + 2\*20 = 46.5 [ms]

Il link colloid bottiglia è il link 3, dunque

Wc = RTT / T3 = 46.5 / 5 = 9.3, dunque la trasmissione continua inizia con finestra pari a 10 MSS.

b)

File = 50 [kbyte] / 1250 [byte] = 40 MSS

Tsetup = 2\*(tau1 + tau2 + tau3) = 40 [ms]

Ttot = Tsetup + 5RTT (1-2-4-8-9) + 15\*T3 + RTT = 394 [ms]

c)

Ttot = Tsetup + 3RTT (1-2-4) + Tout (persi 8 MSS, nuova SSHTHRESH = 8/2 MSS) + 7RTT (1-2-4-5-6-7-8) + 7T3 (arrive degli ACK dell’ultima finestra) = 640 [ms]

## Esercizio 3 (5 punti)



Si consideri la rete in figura dove sono indicati *router*, reti e costo associato alle interfacce dei *router*. Si supponga di utilizzare il protocollo di *routing* OSPF. Si divida come mostrato in figura la rete in 3 aree e si disegnino i grafi che rappresentano la rete vista dal *router* R6, R1, ed R3. Si indichino chiaramente cosa rappresenta ciascun nodo ed i costi associati agli archi.

**Router R6**



**Router R1**



**Router R3**



## Domande (9 punti)

## D1 - La figura riporta la topologia di rete, la tabella di routing di R1 ed il successivo Distance Vector (DV) proveniente da R2 ricevuto da R1. Si assuma che i collegamenti tra i vari router abbiano costo unitario.

## Scrivere:

## la nuova tabella di routing del router R1 dopo la ricezione del DV da R2

## il DV con Split Horizon (no Poisonous Reverse) inviato da R1 a R2 e R3, dopo l’arrivo del DV da R2

## Routing Table R1 Distance Vector da R2

|  |  |  |
| --- | --- | --- |
| Network | Cost | Next-hop |
| 1.1.1.0/24 | 1 | R4 |
| 1.1.2.0/24 | 3 | R2 |
| 1.1.3.0/24 | 2 | R2 |
| 1.1.4.0/24 | 4 | R3 |
| 1.1.5.0/24 | 3 | R3 |

|  |  |
| --- | --- |
| Network | Cost |
| 1.1.1.0/24 | 5 |
| 1.1.3.0/24 | 4 |
| 1.1.4.0/24 | 2 |
| 1.1.6.0/24 | 7 |

a)

|  |  |  |
| --- | --- | --- |
| Network | Cost | Next-hop |
| 1.1.1.0/24 | 1 | R4 |
| 1.1.2.0/24 | 3 | R2 |
| 1.1.3.0/24 | 5 | R2 |
| 1.1.4.0/24 | 3 | R2 |
| 1.1.5.0/24 | 3 | R3 |
| 1.1.6.0/24 | 8 | R2 |
|  |  |  |

b) A R2 A R3

|  |  |
| --- | --- |
| Network | Cost |
| 1.1.1.0/24 | 1 |
| 1.1.5.0/24 | 3 |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Network | Cost |
| 1.1.1.0/24 | 1 |
| 1.1.2.0/24 | 3 |
| 1.1.3.0/24 | 5 |
| 1.1.4.0/24 | 3 |
| 1.1.6.0/24 | 8 |
|  |  |

## D2 - Un sistema di accesso multiplo centralizzato a divisione di tempo (TDMA) è caratterizzato da una trama a slot, ciascuno con un tempo di burst (di trasmissione) TB=10[us] e un tempo di guardia TG=2[us]. Il sistema serve 15 utenti e ha una rate trasmissivo del segnale multiplato di C=2[Mb/s].

## Si chiede di:

## indicare il numero di bit di ciascun tributario trasmessi in ogni slot

## indicare il massimo rate possibile per ciascun tributario in ingresso

**SOLUZIONE**

1. n = TB \* C = 10 \* 10-6 \* 2 106 = 20 [bit]
2. v = n / Ttrama = 20 / 15 \* (TB + TG) = 20 / 15 \* 12 \* 10-6 = 111.11 [kbit/s]

## D3 – Con riferimento ai protocolli di livello applicativo POP3 e SMTP. Si indichi in che cosa differiscono e in quali colloqui del servizio di posta elettronica vengono utilizzati.

**SOLUZIONE**

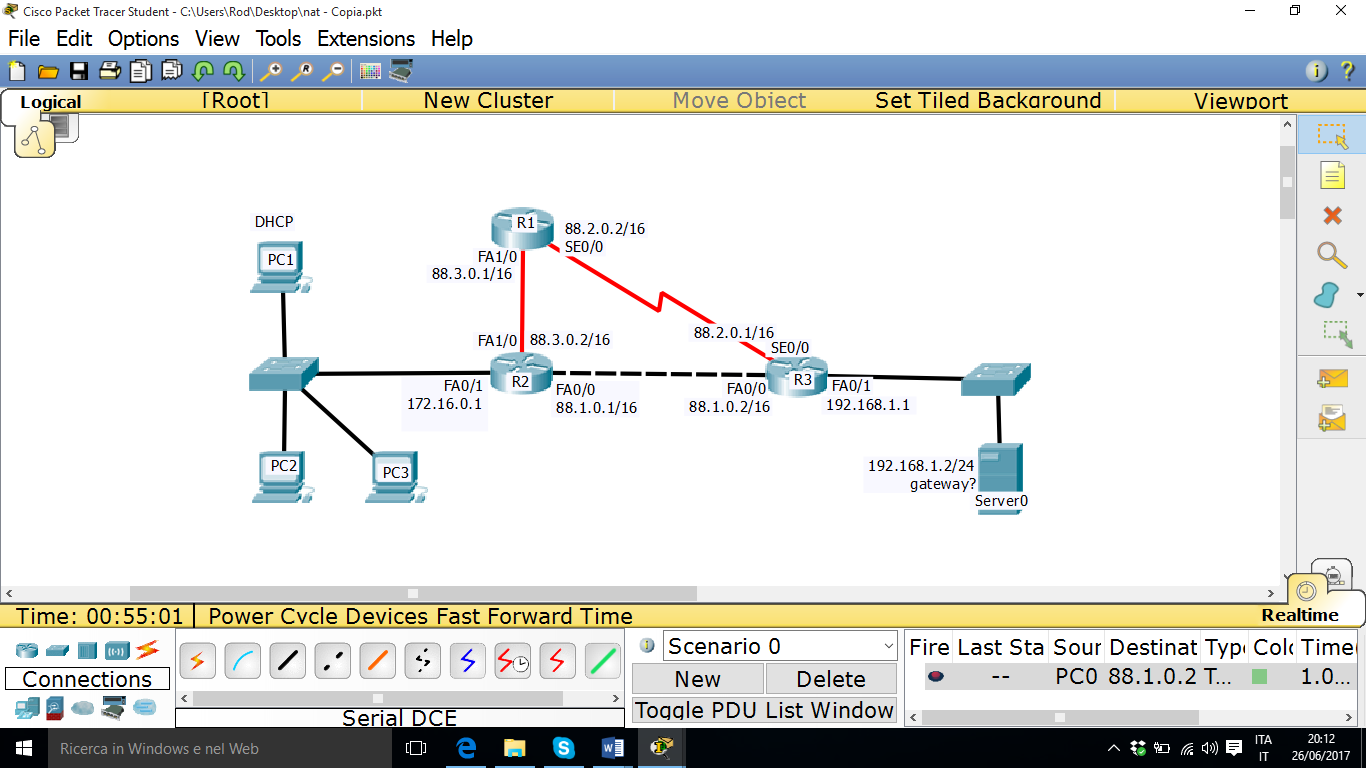
Si vedano le slide. I punti principali sono: POP3 è utilizzato per leggere e scaricare i messaggi di posta, mentre SMTP è utilizzato per inviare i messaggi di posta.

POP3 è tipicamente utilizzato dai client di posta per interrogare il server di posta in arrivo e scaricare i messaggi di posta. SMTP è tipicamente utilizzato 1) dal client di posta per inviare i messaggi di posta al server di posta in uscita 2) dai server di posta per inoltrare i messaggi al serve di posta destinazione

## Laboratorio (6 punti)

*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 PUBBLICHE ad esso connesse, e configurare come **passiva** la interfaccia verso la rete privata ad esso connessa.

(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

R2(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 172.16.0.0 0.0.255.255

R3(config)# ip nat inside source list 1 interface fa1/0 overload

***Socket programming***

Si vuole scrivere un'applicazione client/server per emulare l’applicazione PING utilizzando socket UDP 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:

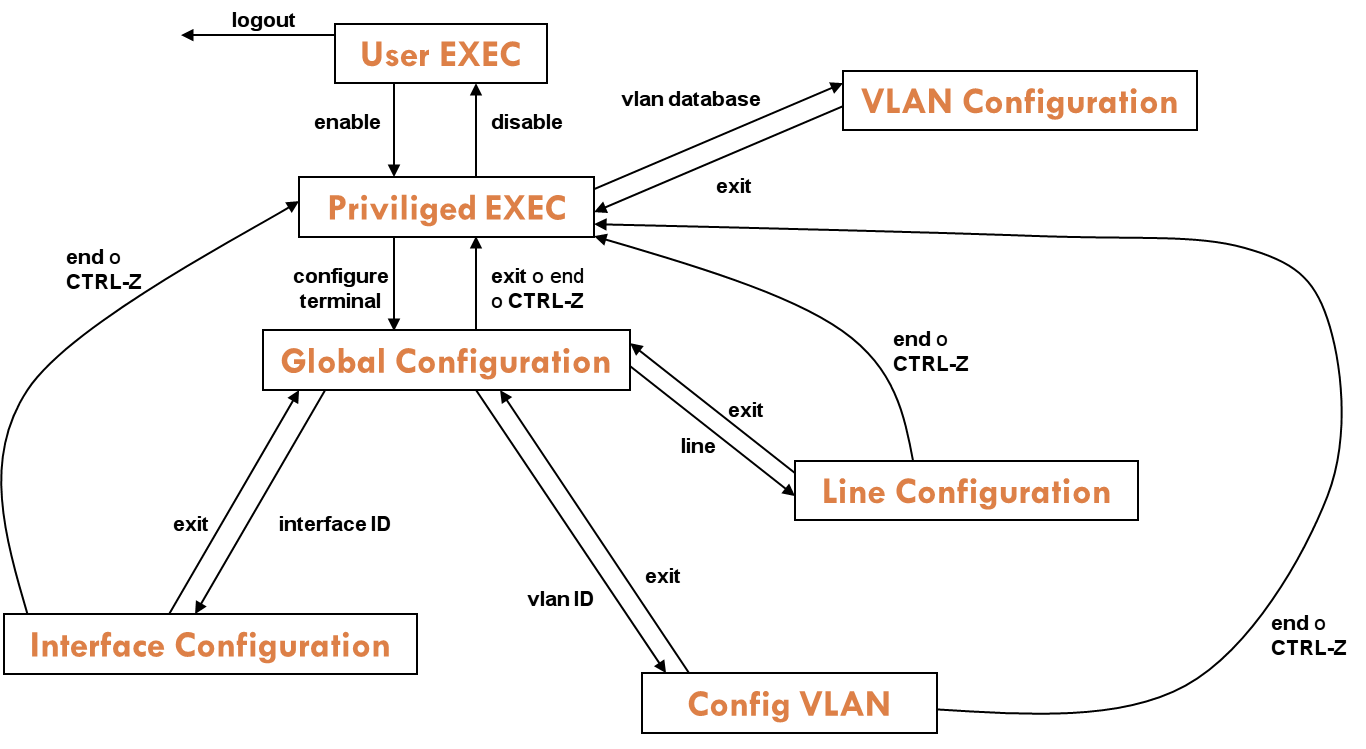
message, address = serverSocket.recvfrom(1024)

message = message.upper()

rand = random.randint(0, 20)

if rand > 9

serverSocket.sendto(message, address)



**Comandi**

|  |  |
| --- | --- |
| Router>  Router> show   cdp  clock  controllers  frame-relay  history  interfaces  ip  version | **Modalità User EXEC**  -CDP information  -Display the system clock  -Interface controllers status  -Frame-Relay information  -Display the session command history  -Interface status and configuration  -IP information  -System hardware and software |
| Router> enable  Router#  Router# show  access-lists  arp  cdp  clock  controllers  frame-relay  history  interfaces  ip  running-config  startup-config  version  Router# copy running-config startup-config | **Modalità Privileged EXEC**  -List access lists  -Arp table  -CDP information  -Display the system clock  -Interface controllers status  -Frame-Relay information  -Display the session command history  -Interface status and configuration  -IP information  -Current operating configuration  -Contents of startup configuration  -System hardware and software status  -Salvare la configurazione corrente |
| Router# configure terminal  Router(config)#  Router(config)# hostname HOST*NAME*  Router(config)# banner motd  Router(config)# enable secret *PASSWORD*  Router(config)# no enable secret | **Modalità Global Configuration**  -Cambiare nome al router  -Impostare messaggio del giorno  -Impostare password  -Disabilitare password |
| Router(config)# interface *TYPE SLOT/PORT*  Router(config-if)# no shutdown  Router(config-if)# shutdown  Router(config-if)# ip address *IP\_ADDRESS NETMASK*  Router(config-if)# clock rate *CLOCK\_RATE* | **Configurare interfaccia**  *-Attivare interfaccia*  *-Disattivare interfaccia*  *-Assegnare IP*  *-Clock seriale* |
| Router(config)# line vty 0 4  Router(config-line)# password *PASSWORD*  Router(config-line)# login  Router(config-line)# ^Z | **-Accesso via rete (remoto).**  -Impostare la password per l’accesso via rete |
| Router(config)# line console 0 | **Accesso via porta console** |
| Router(config)# ip dhcp pool *NAME\_POOL*  Router(dhcp-config)# default-router *ROUTER\_IP\_ADDRESS*  Router(dhcp-config)# network *NETWORK\_IP\_ADDRESS NETMASK*  Router(dhcp-config)# ip dhcp excluded-address *EXCLUDED\_IP\_ADDRESS* | **DHCP**  -Nome pool indirizzi  -Assegnare il default gateway al pool  -Definire la rete a cui appartengono gli indirizzi  -Escludere un indirizzo dal pool |
| Router(config)# ip route *DEST\_PREFIX* *DEST\_NETMASK NEXTHOP/INTERFACE*  Router(config)# no ip route *DEST\_PREFIX* *DEST\_NETMASK NEXTHOP/INTERFACE* | -Aggiungere una **rotta statica**  -Rimuovere una rotta statica |
| Router(config)# router rip  Router(config)# no router rip  Router(config-router)# version N  Router(config-router)# network A.B.C.D  Router(config-router)# passive-interface TYPE SLOT/PORT  Router# debug ip rip  Router# no debug ip rip  Router# show ip route  Router# show ip route rip  Router# show ip protocols  Router# show ip rip database | *-Abilitare* ***RIP***  *-Disabilitare RIP*  *-Scegliere la versione*  *-Definire le reti che usano RIP*  *-Configurare un’interfaccia in modalità passiva.*  *-Abilitare/disabilitare il debug per il protocollo RIP*  *-* *Ottenere la tabella di routing*  *-Visualizzare le entry nella tabella di routing ottenute con RIP*  *-* *Ottenere l'elenco dei protocolli di routing attivi e il loro stato*  *-* *Visualizzare le informazione raccolte dal routing RIP* |
| Router(config)# router ospf *ID-PROCESS*  Router(config)# no router ospf *ID-PROCESS*  Router(config-router)# network *A.B.C.D* *NET\_WILDCARD* area *N*  Router(config-router)# auto-cost reference-bandwidth *BANDWIDTH\_VALUE*  Router(config)# interface *TYPE SLOT/PORT*  Router(config-if)# ip ospf cost *COST\_VALUE* | -Abilitare **OSPF**  -Disabilitare OSPF  -Definire le reti che usano OSPF  -Modificare il valore di banda di riferimento  -Modificare la metrica costo |
| Router(config)# router eigrp *N* Router(config)# no router eigrp N  Router(config-router)# network *A.B.C.D* Router(config-router)# metric weights *TOS K1 K2*  *K3 K4 K5* | -Abilitare **EIGRP**  -Disabilitare OSPF  -Definire le reti che usano EIGRP  -Modificare i pesi delle metriche |
| Router(config)# interface *TYPE* *PORT*/*SLOT*  Router(config-if)# ip nat inside  Router(config-if)# ip nat outside  Router(config)# access-list *LIST\_NUM* permit *NET\_ADDR* *NET\_WILDCARD*  Router(config)# ip nat inside source list *LIST\_NUM* interface *OUTSIDE\_INTERFACE\_NAME* overload | **Configurazione NAT**  -definizione ruolo porte  - Creare una lista di indirizzi a cui sarà permesso il NAT  - Associare il NAT alla lista indicata prima |
| Router(config)# interface *TYPE* *PORT*/*SLOT*  Router(config-if)# ip nat inside  Router(config-if)# ip nat outside  Router(config)# ip nat inside source static tcp *IP\_INSIDE* PORT\_*INSIDE* *IP\_OUTSIDE PORT\_OUTSIDE* | **Configurazione Port Forwarding**  -definizione ruolo porte  - Associare staticamente l'indirizzo e la porta esterna a quelli interni |
| Switch> enable  Switch# show spanning-tree  Switch> enable  Switch# config  Switch(config)# spanning-tree vlan 1 priority 0 | **SPANNING TREE**  -Controllare lo stato del protocollo STP  -Impostazione di uno switch come Root Bridge |

**Codice esercizi laboratorio**

**UDP client**

from socket import \*

serverName = 'localhost'

serverPort = 12000

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

message = raw\_input('Input lowercase sentence:')

clientSocket.sendto(message, (serverName, serverPort))

modifiedMessage, serverAddress = clientSocket.recvfrom(2048)

print modifiedMessage

clientSocket.close()

**UDP server**

from socket import \*

serverPort = 12000

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

serverSocket.bind(('', serverPort))

print "The server is ready to receive"

while 1:

message, clientAddress = serverSocket.recvfrom(2048)

print "Datagram from: ", clientAddress

modifiedMessage = message.upper()

serverSocket.sendto(modifiedMessage, clientAddress)

**UDP error management**

from socket import \*

serverName = 'localhost'

serverPort = 12001

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

clientSocket.settimeout(5)

message = raw\_input('Input lowercase sentence:')

try:

clientSocket.sendto(message, (serverName, serverPort))

modifiedMessage, serverAddress = clientSocket.recvfrom(2048)

# in case of error blocks forever

print modifiedMessage

except error, v:

print "Failure"

print v

finally:

clientSocket.close()

**TCP client**

from socket import \*

serverName = 'localhost'

serverPort = 12000

clientSocket = socket(AF\_INET, SOCK\_STREAM)

clientSocket.connect((serverName, serverPort))

sentence = raw\_input('Input lowercase sentence:')

clientSocket.send(sentence)

modifiedSentence = clientSocket.recv(1024)

print 'From Server:', modifiedSentence

clientSocket.close()

**TCP server**

from socket import \*

serverPort = 12000

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

serverSocket.bind(('', serverPort))

serverSocket.listen(1)

print 'The server is ready to receive'

while True:

connectionSocket, clientAddress = serverSocket.accept()

print "Connection form: ", clientAddress

sentence = connectionSocket.recv(1024)

capitalizedSentence = sentence.upper()

connectionSocket.send(capitalizedSentence)

connectionSocket.close()

**TCP client persistent**

from socket import \*

serverName = 'localhost'

serverPort = 12000

clientSocket = socket(AF\_INET, SOCK\_STREAM)

clientSocket.connect((serverName, serverPort))

while True:

sentence = raw\_input('Input lowercase sentence ( . to stop):')

clientSocket.send(sentence)

if sentence == '.':

break

modifiedSentence = clientSocket.recv(1024)

print 'From Server:', modifiedSentence

clientSocket.close()

**TCP server persistent**

from socket import \*

serverPort = 12000

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

serverSocket.bind(('', serverPort))

serverSocket.listen(1)

while True:

print 'The server is ready to receive'

connectionSocket, clientAddress = serverSocket.accept()

print "Connection form: ", clientAddress

while True:

sentence = connectionSocket.recv(1024)

if sentence == '.':

break

capitalizedSentence = sentence.upper()

connectionSocket.send(capitalizedSentence)

connectionSocket.close()

**TCP auto client**

from socket import \*

import time

serverName = 'localhost'

serverPort = 12000

clientSocket = socket(AF\_INET, SOCK\_STREAM)

clientSocket.connect((serverName, serverPort))

for a in range(100):

clientSocket.send('A')

time.sleep(1)

clientSocket.send('.')

#clientSocket.recv(1024)

clientSocket.close()

**TCP auto server**

from socket import \*

serverPort = 12000

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

serverSocket.bind(('', serverPort))

serverSocket.listen(1)

while True:

print 'The server is ready to receive'

connectionSocket, clientAddress = serverSocket.accept()

print "Connection form: ", clientAddress

while True:

sentence = connectionSocket.recv(1024)

if sentence == '.':

break

print len(sentence)

# connectionSocket.send(capitalizedSentence)

connectionSocket.close()

**TCP server thread**

from socket import \*

import thread

def handler(connectionSocket):

while True:

sentence = connectionSocket.recv(1024)

if sentence == '.':

break

capitalizedSentence = sentence.upper()

connectionSocket.send(capitalizedSentence)

connectionSocket.close()

serverPort = 12000

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

serverSocket.setsockopt(SOL\_SOCKET, SO\_REUSEADDR, 1)

serverSocket.bind(('', serverPort))

serverSocket.listen(1)

while True:

print 'The server is ready to receive'

newSocket, addr = serverSocket.accept()

thread.start\_new\_thread(handler, (newSocket,))