Esame Completo - 11 Luglio 2017

Cognome	
Nome	
Matricola	

Tempo complessivo a disposizione per lo svolgimento: 2 ore 15 minuti
Si usi lo spazio bianco dopo ogni esercizio per la risoluzione

E1	E2	E3	Quesiti	Lab

1 - Esercizio (8 punti)

La rete di un ISP è riportata in figura. L'ISP possiede lo spazio di indirizzamento: 123.123.190.0/23
Definire un piano di indirizzamento in grado di supportare il numero di *host* indicato nella figura.

a) Indicare le sottoreti IP graficamente nella figura, mettendo in evidenza i confini tra le reti IP ed assegnando una lettera identificativa a ciascuna rete. Assegnare le lettere in ordine alfabetico iniziando dalla rete più grande e procedendo per dimensione decrescente (# indirizzi rete A ≥ # indirizzi rete B ≥ …..). Per ciascuna sottorete definire l'indirizzo di rete, la *netmask* (in formato decimale puntato), e l'indirizzo di broadcast diretto, usando la tabella 1. Assegnare gli indirizzi alle sottoreti a partire da quelli più bassi del blocco 123.123.190.0/23.

b) Scrivere nella tabella 2 la tabella di instradamento del router R2 nel modo più compatto possibile dopo aver assegnato opportunamente degli indirizzi ai router a cui R2 è connesso.

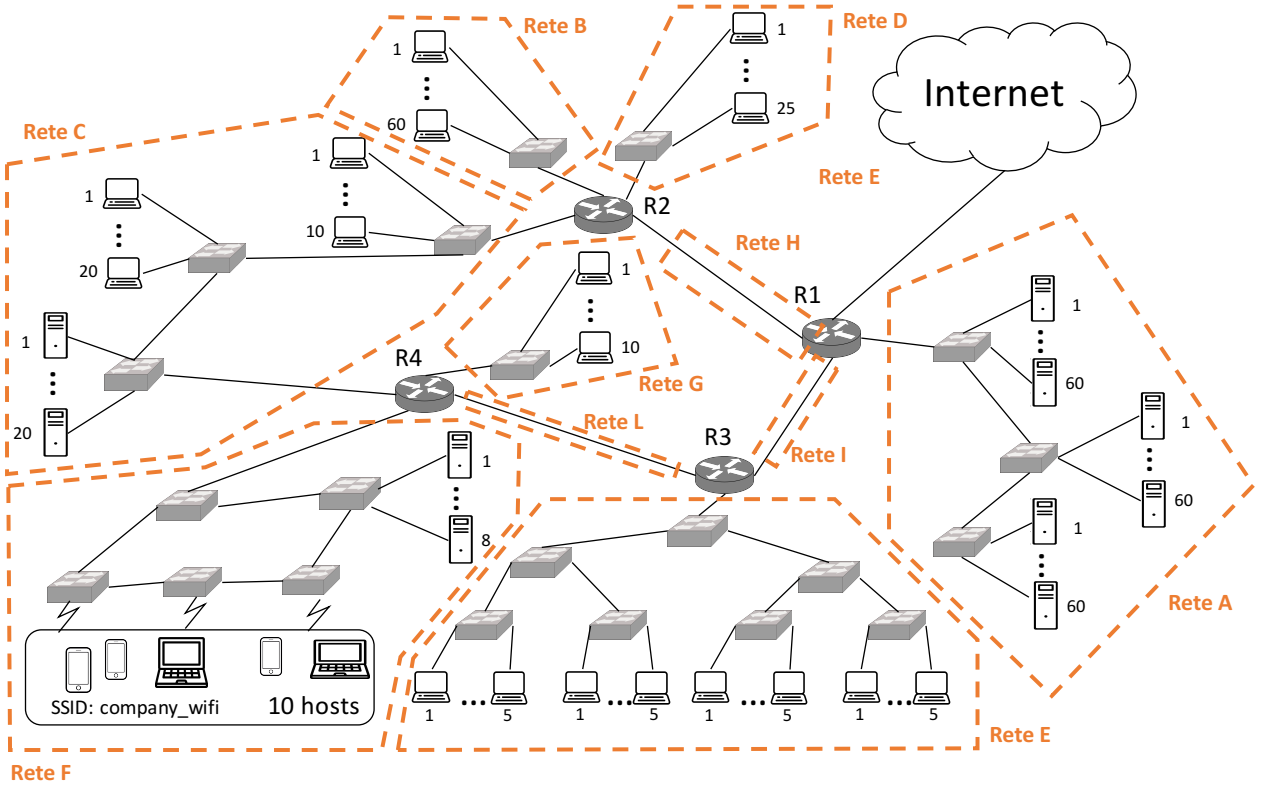


Tabella 1

Rete	Indirizzo di rete	Netmask	Ind. broadcast diretto
A	190.0	255.255.255.0	190.255
B	191.0	255.255.255.192	191.63
C	191.64	255.255.255.192	191.127
D	191.128	255.255.255.224	191.159
E	191.160	255.255.255.224	191.191
F	191.192	255.255.255.224	191.223
G	191.224	255.255.255.240	191.239
H	191.240	255.255.255.252	191.243
I	191.244	255.255.255.252	191.247
L	191.248	255.255.255.252	191.251

Tabella 2

Network	Netmask	Next Hop
0.0.0.0	0.0.0.0	191.242

a)
Rete A: 180 indirizzi ➔/24
Rete B: 60 indirizzi ➔/26
Rete C: 50 indirizzi ➔/26
Rete D: 25 indirizzi ➔ /27
Rete E: 20 indirizzi ➔/27
Rete F: 18 indirizzi ➔ /27
Rete G: 10 indirizzi ➔/28
Rete H: 2 indirizzi ➔ /30
Rete I: 2 indirizzi ➔ /30
Rete L: 2 indirizzi ➔ /30

190.0/23 – 1011111||0

/24 1011111||0|.00000000: Rete A 190.0/24 BC: 190.255

/24 1011111||1

/26 1011111||1.00|000000: Rete B 191.0/26 BC: 191.63

/26 1011111||1.01|000000: Rete C 191.64/26 BC: 191.127

/26 1011111||1.10|000000

/27 1011111||1.100|00000: Rete D 191.128/27 BC: 191.159

/27 1011111||1.101|00000: Rete E 191.160/27 BC: 191.191

/26 1011111||1.11|000000

/27 1011111||1.110|00000: Rete F 191.192/27 BC: 191.223

/27 1011111||1.111|00000

/28 1011111||1.1110|0000: Rete F 191.224/28 BC: 191.239

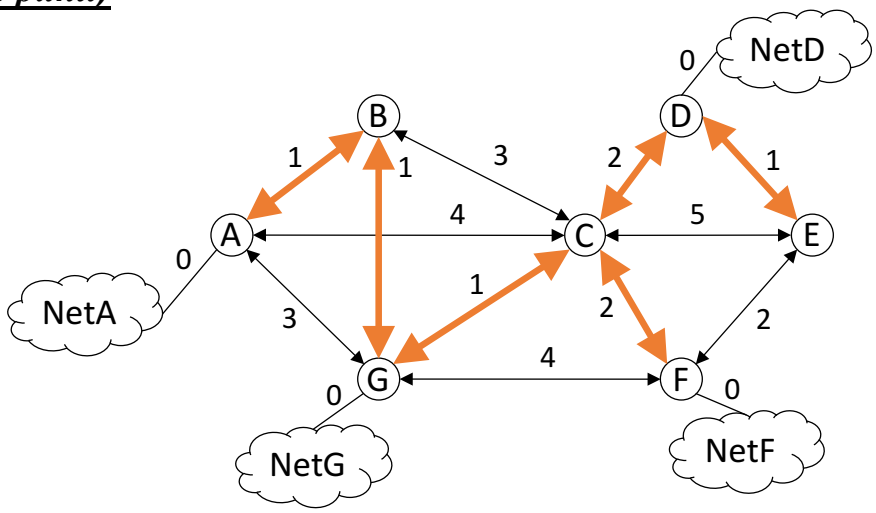
/28 1011111||1.1111|0000

/30 1011111||1.111100|00: Rete G 191.240/28 BC: 191.243

/30 1011111||1.111101|00: Rete G 191.244/28 BC: 191.247

/30 1011111||1.111110|00: Rete G 191.248/28 BC: 191.251

Esercizio 2 (6 punti)



Nella rete in figura è rappresentato il grafo di una rete in cui sono presenti dei router (A, B, C, D, E, F, G) e 4 reti (NetA, NetD, NetF, NetG). I costi di attraversamento sono indicati accanto ad ogni link, i link sono bidirezionali e simmetrici.
Si chiede di:

a) Calcolare mediante l'algoritmo di Bellman-Ford l'albero dei cammini minimi con sorgente A e destinazioni tutti gli altri router (si omettano le reti nel grafo). Indicare:
   - nella Tabella A, il valore dell'etichetta ad ogni step in cui il nodo viene analizzato: nel caso lo step successivo non modifichi l'etichetta dello step precedente occorre riscrivere l'etichetta dello step precedente.
   - nella figura sopra, l'albero trovato

b) Sulla base dell'albero dei cammini calcolato al punto precedente, indicare i Distance Vector (DV) relativi alle reti NetA, NetD, NetF e NetG, inviati dal router G nella modalità Split Horizon senza Poisonous Reverse. Per ogni DV inviato indicare chiaramente: il destinatario del DV, le reti raggiungibili comunicate ed i rispettivi costi.

Tabella A

Nodo A	Nodo B	Nodo C	Nodo D	Nodo E	Nodo F	Nodo G
A,0	-,inf	-,inf	-,inf	-,inf	-,inf	-,inf
A,0	A,1	A,4	-,inf	-,inf	-,inf	A,3
A,0	A,1	A,4	C,6	C,9	C,6	B,2
A,0	A,1	G,3	C,6	D,7	C,6	B,2
A,0	A,1	G,3	C,5	D,7	C,5	B,2
A,0	A,1	G,3	C,5	D,6	C,5	B,2
A,0	A,1	G,3	C,5	D,6	C,5	B,2

b)

Verso A: (NetA,2), (NetD,3), (NetF,3), (NetG0)

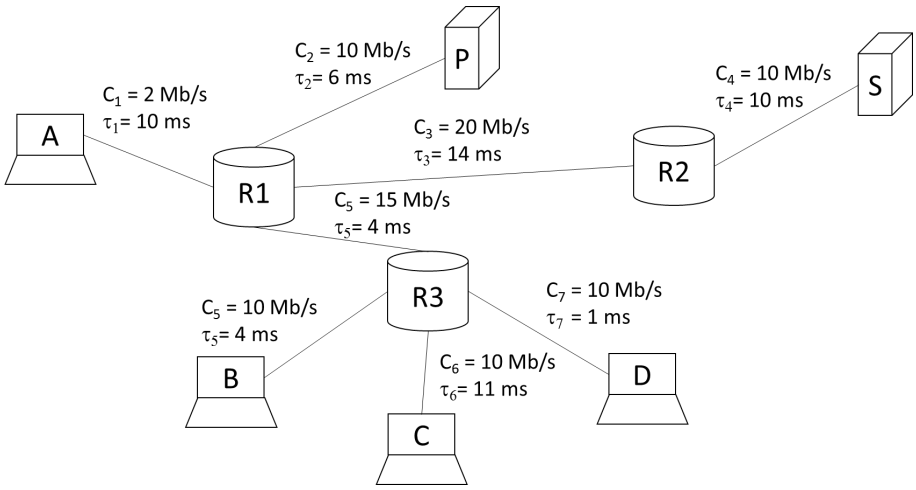
Verso B: (NetD,3), (NetF,3), (NetG0)

Verso C: (NetA,2), (NetG0)

Verso F: (NetA,2), (NetD,3), (NetF,3), (NetG0)

Esercizio 3 (4 punti)

Si assuma B, P e S siano rispettivamente client, proxy e server HTTP. B richiede a P un contenuto che non è presente nella cache e che quindi P deve richiedere a S. Il contenuto consiste in un documento HTML base di 40 KByte e 9 immagini di 0.6 MByte ciascuna. Per tutto il tempo sono presenti e attivi i seguenti flussi TCP: A-S, C-S, D-S.



Assumiamo che:

- i messaggi di apertura connessione TCP e richieste HTTP sono di lunghezza trascurabile,
- si trascurano i tempi di accodamento nei nodi,
- la trasmissione dei file avviene a un ritmo medio di trasmissione pari al valore di condivisione equa delle risorse.

Si calcoli il tempo di trasferimento necessario, inteso come intervallo tra l'istante in cui B richiede il primo file a P e l'istante in cui B riceve interamente l'ultimo file da P, nelle due modalità:

a) connessione HTTP persistente (una singola connessione) e
b) connessione HTTP non persistente (con trasmissione in parallelo delle immagini).

Si assuma che le connessioni HTTP tra S e P e tra P ed A avvengano con la stessa modalità.

		Flusso B-P	Flusso P-S	Unità mis.
Caso a	RTT	28	60	ms
	R_{HTML}	5	2.5	Mb/s
	T_{HTML}	64	128	ms
	R_{OBJ}	5	2.5	Mb/s
	T_{OBJ}	960	1920	ms
Caso b	RTT	28	60	ms
	R_{HTML}	5	2.5	Mb/s
	T_{HTML}	64	128	ms
	R_{OBJ}	1.11	0.83	Mb/s
	T_{OBJ}	4324.32	5783.13	ms

T_{Tot_a} =

T_{Tot_b} =

a)
1 flusso B-P:
link 5 10/1 = 10
link 5 15/3 = 5
link 2 10/1 = 10
collo di bottiglia: 5 Mb/s

1 flusso P-S:
link 2 10/1 = 10
link 3 20/4 = 5
link 4 10/4 = 2.5
collo di bottiglia: 2.5 Mb/s

$$T_{Tot} = RTT^{BP} + RTT^{BP} + T_{HTML}^{BP} + RTT^{PS} + RTT^{PS} + T_{HTML}^{PS} + 9\left(RTT^{BP} + T_{OBJ,1}^{BP} + RTT^{PS} + T_{OBJ,1}^{PS}\right) = 11RTT^{BP} + 11RTT^{PS} + T_{HTML}^{BP} + T_{HTML}^{PS} + 9\left(T_{OBJ,1}^{BP} + T_{OBJ,1}^{BP}\right) = 27.08\ s$$

b)
1 flusso B-P:
link 5 10/1 = 10
link 5 15/3 = 5
link 2 10/1 = 10
collo di bottiglia: 5 Mb/s

1 flusso P-S:
link 2 10/1 = 10
link 3 20/4 = 5
link 4 10/4 = 2.5
collo di bottiglia: 2.5 Mb/s

9 flussi B-P:
link 5 10/9 = 1.11
link 5 15/12 = 1.36
link 2 10/9 = 1.11
collo di bottiglia: 1.11 Mb/s

9 flussi P-S:
link 2 10/9 = 1.11
link 3 20/12 = 1.67
link 4 10/12 = 0.83
collo di bottiglia: 0.83 Mb/s

$$T_{Tot} = RTT^{BP} + RTT^{BP} + T_{HTML}^{BP} + RTT^{PS} + RTT^{PS} + T_{HTML}^{PS} + RTT^{BP} + RTT^{BP} + T_{OBJ,9}^{BP} + RTT^{PS} + RTT^{PS} + T_{OBJ,9}^{PS} = 4RTT^{BP} + 4RTT^{PS} + T_{HTML}^{BP} + T_{HTML}^{PS} + T_{OBJ,9}^{BP} + T_{OBJ,9}^{BP} = 10.65\ s$$

4–Domande (9 punti)

D1 - Un *router* è caratterizzato dalla seguente configurazione delle interfacce locali e della seguente tabella di *routing*. Per ciascuno dei pacchetti indicati di seguito (caratterizzati da interfaccia di provenienza, indirizzo di destinazione, dimensione e valore dei *flag* Do-not-Fragment) dire come si comporta il router specificando se procede con inoltro diretto, indiretto o se scarta il pacchetto (**tipo inoltro**). Indicare **chiaramente l'interfaccia di inoltro, la riga della tabella di *routing* "scelta" per l'inoltro indiretto ed eventualmente il motivo per cui il pacchetto viene scartato**.

eth0: 123.123.144.254, 255.255.240.0 MTU=500 [byte]
eth1: 212.111.128.254, 255.255.252.0 MTU=800 [byte]

Riga #	*Destinazione*	*Netmask*	*Next Hop*
1	131.175.32.0	255.255.224.0	212.111.128.222
2	131.175.64.0	255.255.192.0	123.123.150.254
3	0.0.0.0	0.0.0.0	212.111.130.254

212.111.131.23 da eth1, L=400 [byte], DF=1

Tipo inoltro: SCART Interfaccia inoltro: Riga tabella (se necessario):
Eventuale motivo di scarto:
Inoltro diretto con interfaccia d'ingresso uguale a interfaccia d'uscita

131.175.96.44 da eth1, L=600 [byte], DF=1

Tipo inoltro: SCART Interfaccia inoltro: Riga tabella (se necessario):
Eventuale motivo di scarto:
Inoltro indiretto su eth0 da riga 2, ma L > MTU e DF=1

131.175.192.34 da eth1, L=600 [byte], DF=0

Tipo inoltro: IND Interfaccia inoltro: eth1 Riga tabella (se necessario): 3
Eventuale motivo di scarto:

D2 - Un sistema di accesso multiplo a divisione di tempo (TDMA) è caratterizzato da slot di durata Ts= 1 [ms], con un rapporto T_{utile} (dati) / $T_{guardia}$ = 4. Il sistema serve 20 stazioni e ciascuna ha una velocità di tributario pari a v=1 [kb/s]. Indicare

- la durata temporale della trama, T_{TRAMA};
- il numero di bit di ciascuna stazione trasmessi in ogni slot, k;
- il rate di trasmissione del segnale multiplato, C.

$T_{guardia} + 4T_{guardia} = Ts \Rightarrow T_{guardia} = 0.2\ ms,\ \ T_{utile} = 0.8\ ms$
$T_{TRAMA} = 20 \cdot Ts = 20\ ms$
$k = v \cdot T_{TRAMA} = 20\ bit$
$C = \frac{k}{T_{utile}} = 25 \frac{kb}{s}$

D3 - A proposito del parametro Maximum Segment Size (MSS) nel protocollo TCP [rispondere negli spazi previsti]:

a) A che cosa serve?
Serve ad indicare la lunghezza massima consentita per un segmento TCP

b) Durante quale momento della connessione TCP viene configurato?

Durante la fase di setup della connessione

c) In quale campo del header del segmento TCP viene trasmesso?

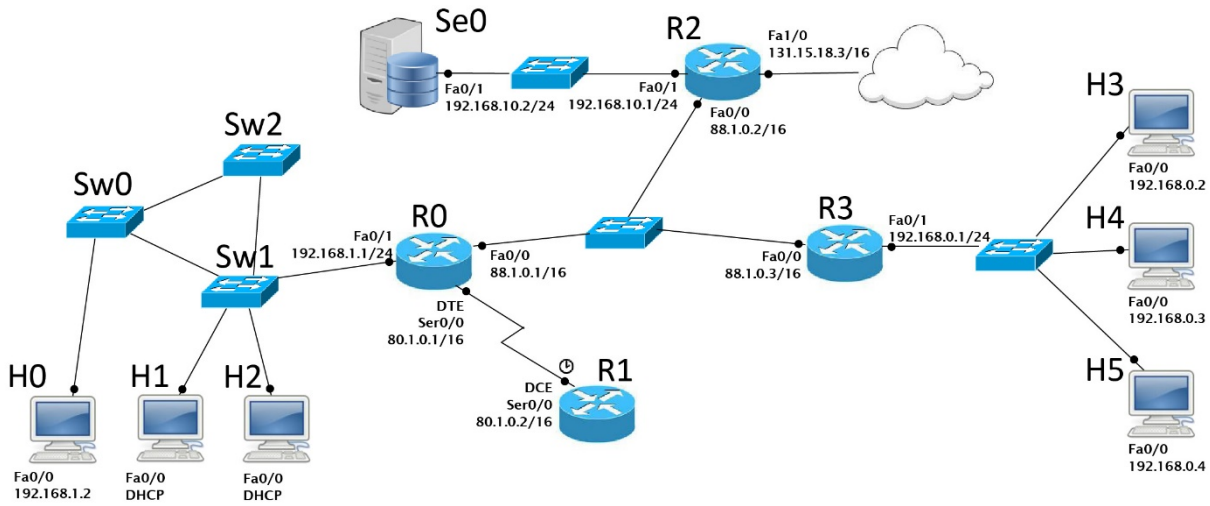
Fa parte delle opzioni dell'header TCP

d) In base a quale caratteristica della rete viene normalmente scelto il suo valore?

Il suo valore dipende dai livelli inferiori al TCP, in particolare dal valore di MTU

5 – Laboratorio (6 punti)
Packet Tracer

Si consideri la rete in figura



Attenzione:

- **Indirizzi IP e gateway sono già stati configurati per i 6 host.**
- **Le interfacce dei router R0, R2 e R3 sono già state configurate ed attivate come in figura.**
- **Le reti /24 sono reti private**
- **Indicare sempre prima del comando il prompt visualizzato dal sistema, prestando attenzione alla modalità di partenza in ciascuna richiesta**

Q1) Configurare ed attivare l'interfaccia seriale Ser0/0 del router **R1** assumendo un collegamento a 20 Mbit/s.

```
R1> enable
R1# configure terminal
R1(config)# interface Ser0/0
R1(config-if)# ip address 80.1.0.2 255.255.0.0
R1(config-if)# clock rate 20000000
R1(config-if)# no shutdown
```

Q2) Configurare il routing statico sul router **R3** in modo che possa raggiungere tutte le reti pubbliche e internet, minimizzando il numero di regole necessarie.

```
R3(config)# ip route 80.1.0.0 255.255.0.0 88.1.0.1
R3(config)#ip route 0.0.0.0 0.0.0.0 88.1.0.2
```

.

Socket programming

Si vuole scrivere un'applicazione client/server UDP per conteggiare il numero di vocali presenti in una stringa.
Il client chiede all'utente di inserire una stringa, il server risponde indicando il numero di vocali presenti nella stringa (sia maiuscole che minuscole). **Hint**: y.count(x) conta quante volte appare l'elemento x nella lista y.

UDP client

```
from socket import *

serverName = 'localhost'
serverPort = 9999
clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.settimeout(5)

message = raw_input('Inserisci una frase:')
clientSocket.sendto(message, (serverName, serverPort))

try:
    reply, serverAddress = clientSocket.recvfrom(2048)
    print reply
except error, v:
    print "Il server non ha risposto entro il timeout..."
finally:
    clientSocket.close()
```

UDP server

```
from socket import *

serverPort = 9999
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))

print 'Server pronto a ricevere!'

vocali = ['A','E','I','O','U']

while 1:
    messaggio, clientAddress = serverSocket.recvfrom(2048)

    num = 0
    for voc in vocali:
        num = num + messaggio.count(voc)
    risposta = "Il messaggio contiene " + str(num) + " vocali."
    serverSocket.sendto(risposta, clientAddress)
```

Q1) Completare lo script "UDP client" date le seguenti specifiche:

- Utilizzare indirizzi IPv4
- Time-out in ricezione: 5 secondi.
- Lunghezza buffer di ricezione: 2048 byte.

Q2) Con quale messaggio risponde il server se il client invia la stringa *"aEniU"* ?

La risposta del server è: Il messaggio contiene 2 vocali.

Q3) Se si considera la seguente versione del server UDP (**in grassetto ci sono evidenziate le modifiche rispetto a UDP server1)**, con quale messaggio risponde il server se il client invia la stringa *"aEniU"* ?

La risposta del server è: Il messaggio contiene 4 vocali.

UDP server2

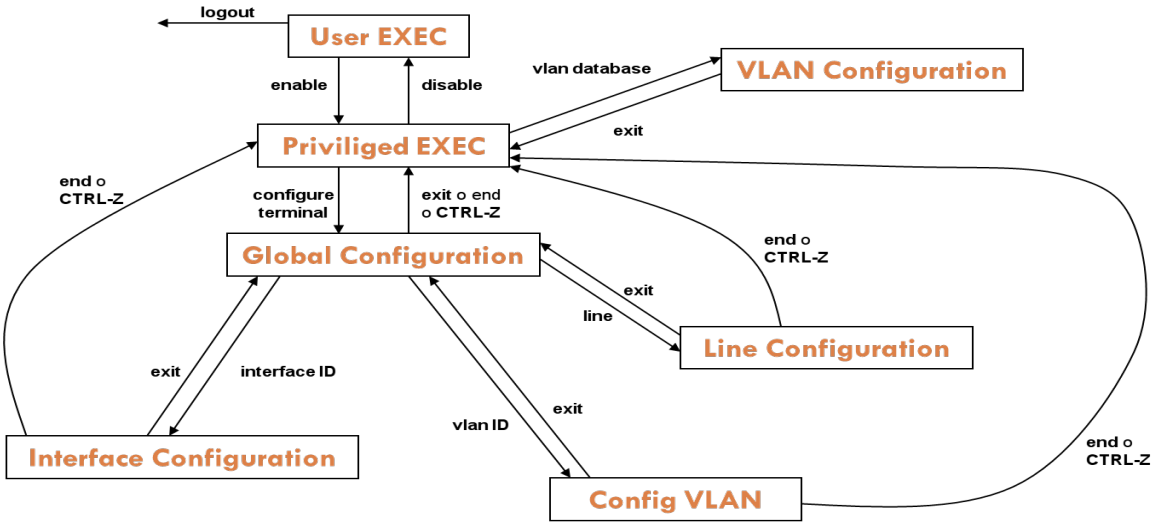
```
from socket import *

serverPort = 9999
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))

print 'Server pronto a ricevere!'

vocali = ['A','E','I','O','U']

while 1:
    messaggio, clientAddress = serverSocket.recvfrom(2048)
    messggioUp =  messagio.upper()
    num = 0
    for voc in vocali:
        num = num + messaggioUp.count(voc)
    risposta = "Il messaggio contiene " + str(num) + " vocali."
    serverSocket.sendto(risposta, clientAddress)
```



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*	**DHCP** -Nome pool indirizzi

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*	-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
Configurazione NAT 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	-definizione ruolo porte - Creare una lista di indirizzi a cui sarà permesso il NAT - Associare il NAT alla lista indicata prima
Configurazione Port Forwarding 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*	-definizione ruolo porte - Associare staticamente l'indirizzo e la porta esterna a quelli interni
SPANNING TREE Switch> enable Switch# show spanning-tree Switch> enable Switch# config Switch(config)# spanning-tree vlan 1 priority 0	-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,))
```