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()
```

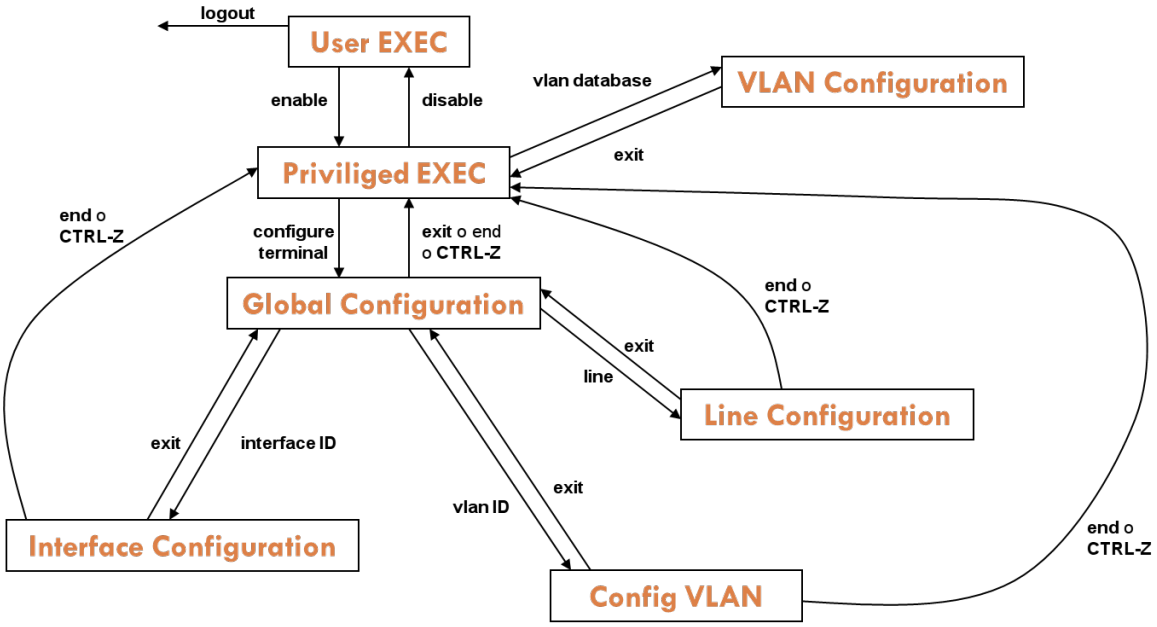
TCP auto server

```
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,))
```

Modalità operative



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(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