NATIONAL SCHOOL OF BUSINESS MANAGEMENT

COMPUTER NETWORKS

GROUP ASSIGNMENT

Contents

INTRODUCTION

Networking fundamentals

A **computer network**, or **data network**, is a digital telecommunications network which allows nodes to share resources. In computer networks, computing devices exchange data with each other using connections (data links) between nodes. These data links are established over cable media such as wires or optic cables, or wireless media such as Wi-Fi.

Network computer devices that originate, route and terminate the data are called network nodes.[1] Nodes can include hosts such as personal computers, phones, servers as well as networking hardware. Two such devices can be said to be networked together when one device is able to exchange information with the other device, whether they have a direct connection to each other. In most cases, application-specific communications protocols are layered (i.e. carried as payload) over other more general communications protocols. This formidable collection of information technology requires skilled network management to keep it all running reliably.

Computer networks support an enormous number of applications and services such as access to the World Wide Web, digital video, digital audio, shared use of application and storage servers, printers, and fax machines, and use of email and instant messaging applications as well as many others. Computer networks differ in the transmission medium used to carry their signals, communications protocols to organize network traffic, the network's size, topology, traffic control mechanism and organizational intent. The best-known computer network is the Internet.Z

Overview

A company consists of three branches dispersed across the island, the headquarters (HQ) of the company is in Colombo and two branch offices are situated at various remote locations in the Island. There are 80 nodes in HQ and nodes are spread equally over four functional departments namely Management, Sales, Finance, HR & Admin. Each branch has 12 nodes. we've been tasked to design a new network for the company by developing a Local Area Network (LAN) and a Wide Area Network (WAN) architecture that aligns with its data communication requirements.

Configuring the LAN and Switching Topology

A local area network (LAN) is a group of computers and associated devices that share a common communications line or wireless link to a server. Typically, a LAN encompasses computers and peripherals connected to a server within a distinct geographic area such as an office or a commercial establishment. Computers and other mobile devices use a LAN connection to share resources such as a printer or network storage.

Subnetting

From the 3 class A, B, C IP blocks which have been given to us as following,

1. 192.168.15.0/24- headquarters
2. 172.16.10.0/24-branches
3. 10.1.1.0/28- WAN/links routers

Headquarters

We have divided the headquarters ip block into 4 subnets for its Management, Sales, Finance, HR & Admin which are its functional departments. We have allocated 2 bits to the network side and retained 6 bits on the host side to accommodate the 20 pcs each department.

Subnet Mask- 255.255.255.192

Network ID	Department	Default gateway
192.168.15.0/26	Management	192.168.15.1
192.168.15.64/26	Sales	192.168.15.65
192.168.15.128/26	Finance	192.168.15.129
192.168.15.192/26	Admin & HR	192.168.15.193

Remote branch 1/2

We have divided the branch ip block into 2 subnets. We have allocated 1 bit to the network side and retained 7 bits on the host side to accommodate the 12 pcs each department.

Network id	Department	Default gateway
172.16.10.0/25	Branch 1	172.16.10.1
172.16.10.128/25	Branch 2	172.16.10.129

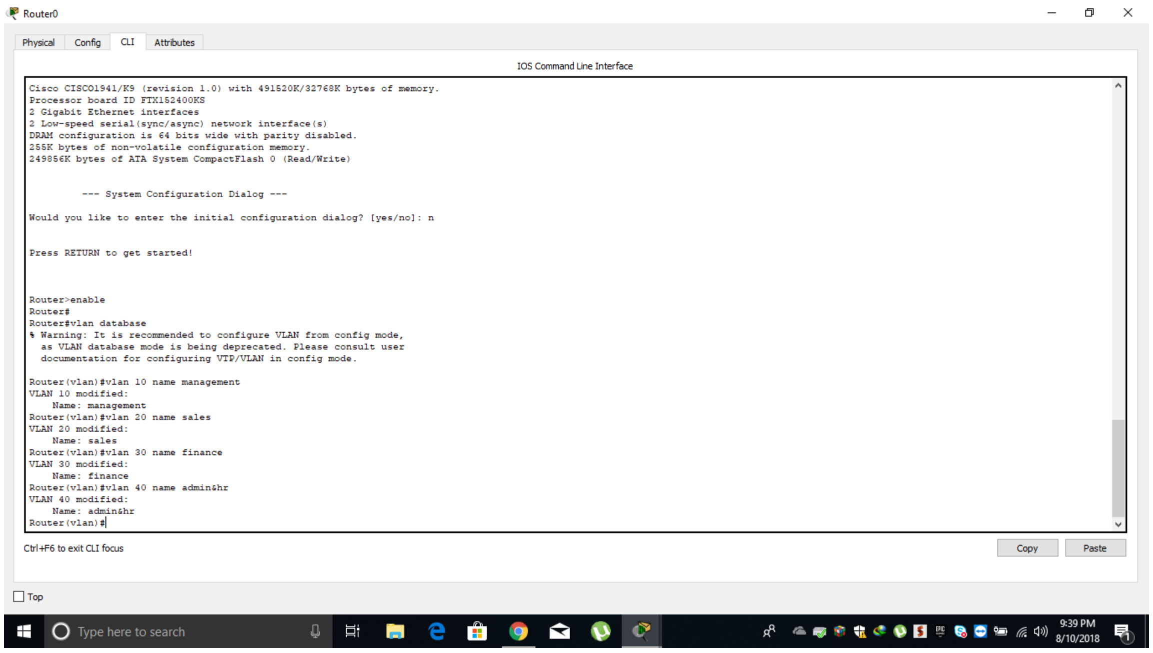
WAN/links routers

We have divided the ip block into 2 subnets. We have allocated 2 bit to the network side and retained 6 bits on the host side to accommodate the links and routers

Creating virtual local area networks (VLANs)

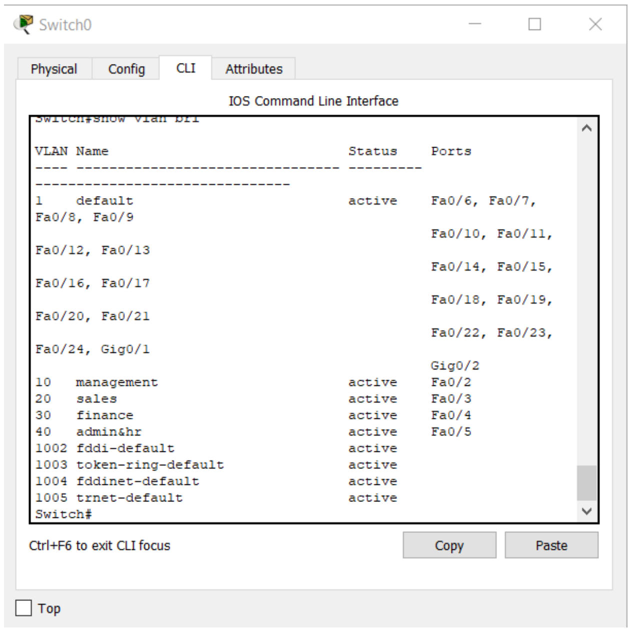
A VLAN is a subgroup of a LAN, in a VLAN computers or other devices behaves as if they are connected using the same wire together although they are on different segments of a LAN. A VLAN is created using software other than physically creating it.

Creating 3 VLANs in each switch for 3 departments:



```
Cisco CISCO1941/K9 (revision 1.0) with 491520K/32768K bytes of memory.
Processor board ID FTX152400KS
2 Gigabit Ethernet interfaces
2 Low-speed serial(sync/async) network interface(s)
DRAM configuration is 64 bits wide with parity disabled.
255K bytes of non-volatile configuration memory.
249856K bytes of ATA System CompactFlash 0 (Read/Write)


         --- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: n


Press RETURN to get started!



Router>enable
Router#
Router#vlan database
% Warning: It is recommended to configure VLAN from config mode,
  as VLAN database mode is being deprecated. Please consult user
  documentation for configuring VTP/VLAN in config mode.

Router(vlan)#vlan 10 name management
VLAN 10 modified:
    Name: management
Router(vlan)#vlan 20 name sales
VLAN 20 modified:
    Name: sales
Router(vlan)#vlan 30 name finance
VLAN 30 modified:
    Name: finance
Router(vlan)#vlan 40 name admin&hr
VLAN 40 modified:
    Name: admin&hr
Router(vlan)#
```

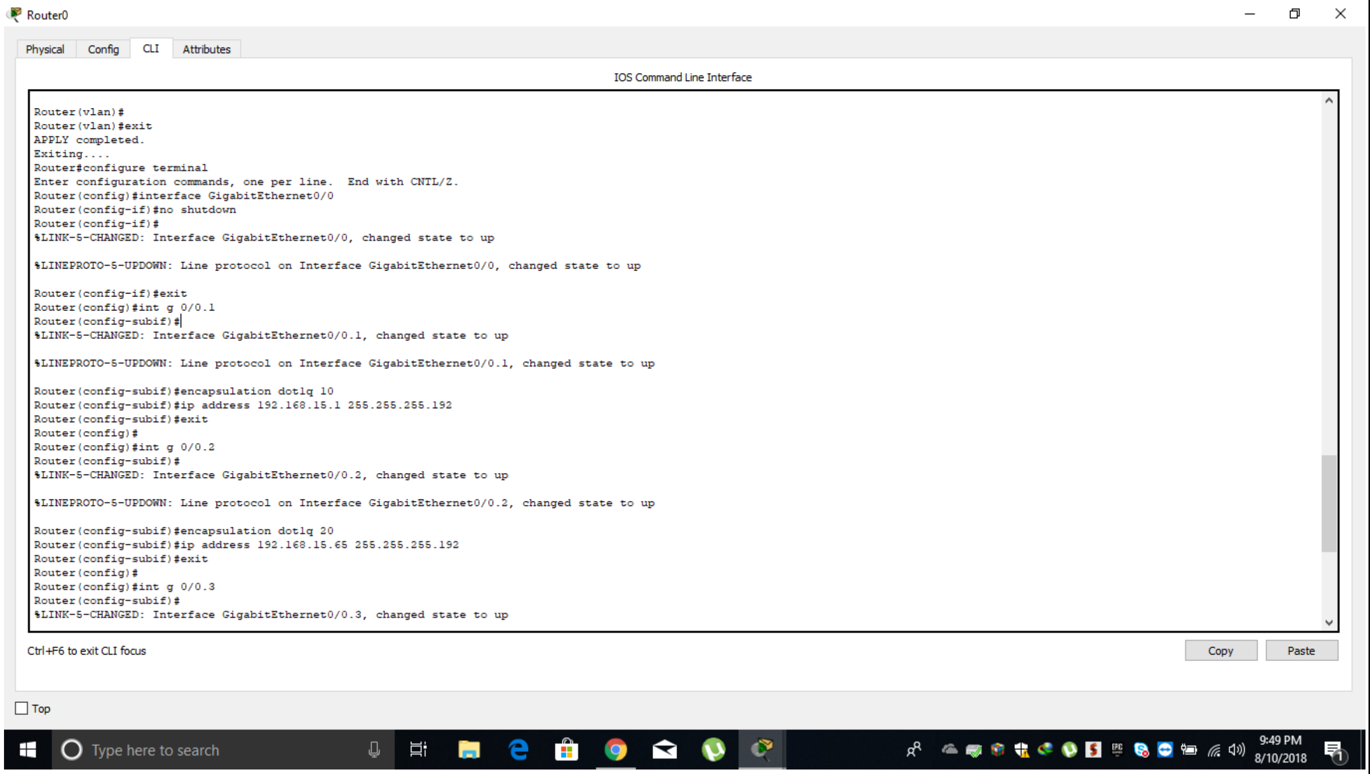
VLANs created successfully in switch and ports assigned successfully.



```
Switch#show vlan bri

VLAN Name                             Status    Ports
---- -------------------------------- ---------
-------------------------------
1    default                          active    Fa0/6, Fa0/7,
Fa0/8, Fa0/9
                                                Fa0/10, Fa0/11,
Fa0/12, Fa0/13
                                                Fa0/14, Fa0/15,
Fa0/16, Fa0/17
                                                Fa0/18, Fa0/19,
Fa0/20, Fa0/21
                                                Fa0/22, Fa0/23,
Fa0/24, Gig0/1
                                                Gig0/2
10   management                       active    Fa0/2
20   sales                            active    Fa0/3
30   finance                          active    Fa0/4
40   admin&hr                         active    Fa0/5
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
Switch#
```

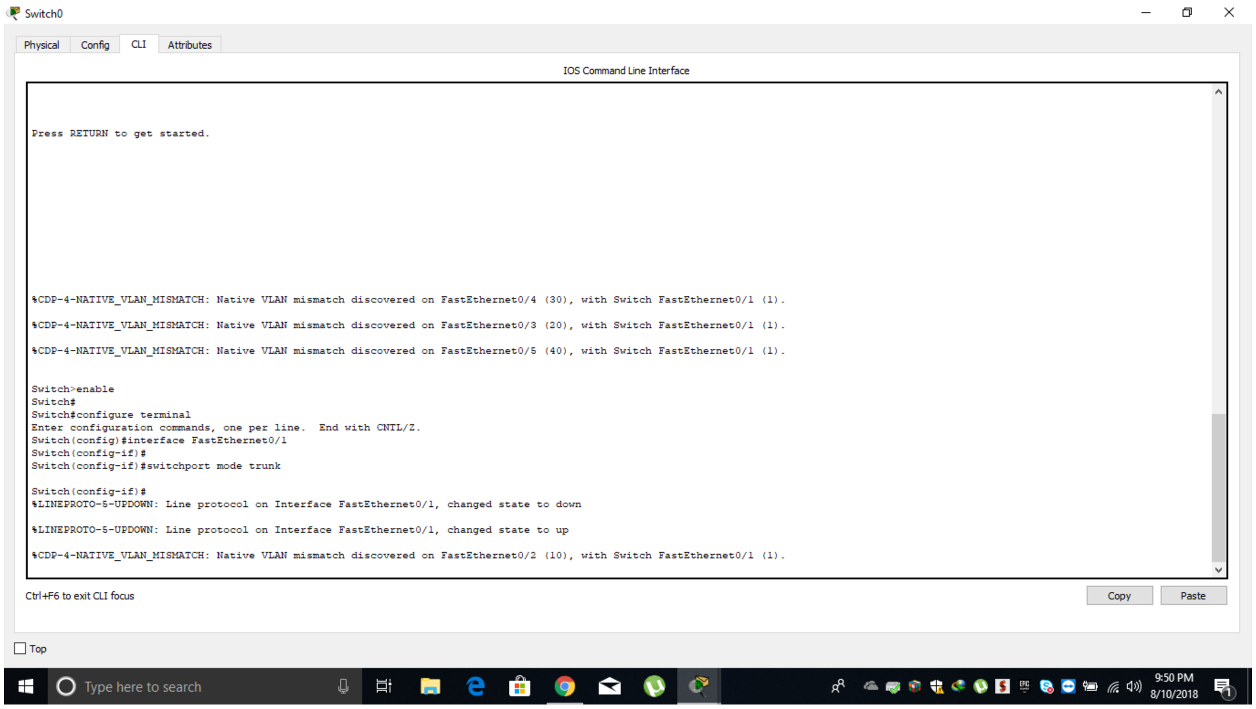
Inter -VLAN routing

Inter-VLAN routing can be defined to forward traffic between different VLAN by implementing a router in the network. As we learnt previously, VLANs logically segment the switch into different subnets, when a router is connected to the switch, an administrator can configure the router to forward the traffic between the various VLANs configured on the switch. The user nodes in the VLANs forwards traffic to the router which then forwards the traffic to the destination network regardless of the VLAN configured on the switch.



```
Router(vlan)#
Router(vlan)#exit
APPLY completed.
Exiting....
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface GigabitEthernet0/0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0, changed state to up

Router(config-if)#exit
Router(config)#int g 0/0.1
Router(config-subif)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0.1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0.1, changed state to up

Router(config-subif)#encapsulation dot1q 10
Router(config-subif)#ip address 192.168.15.1 255.255.255.192
Router(config-subif)#exit
Router(config)#
Router(config)#int g 0/0.2
Router(config-subif)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0.2, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0.2, changed state to up

Router(config-subif)#encapsulation dot1q 20
Router(config-subif)#ip address 192.168.15.65 255.255.255.192
Router(config-subif)#exit
Router(config)#
Router(config)#int g 0/0.3
Router(config-subif)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0.3, changed state to up
```

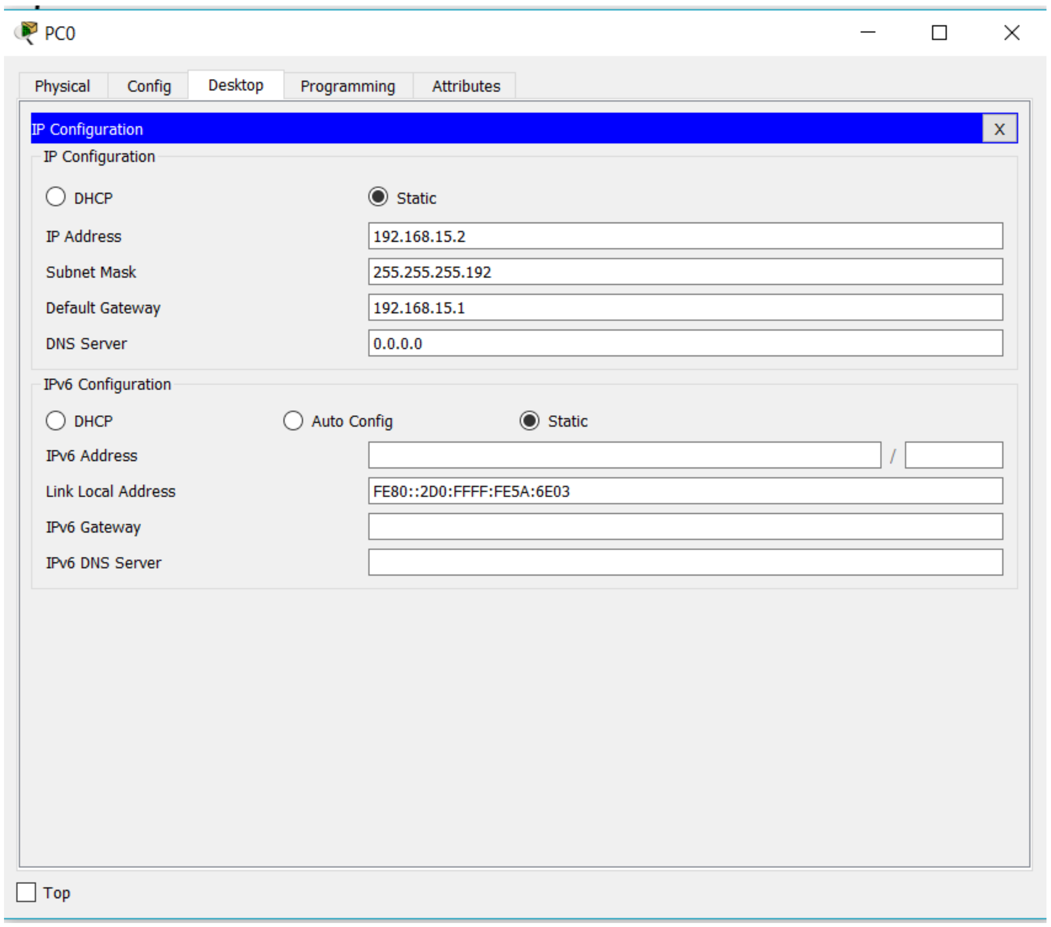
Assigning ports to each VLAN



```
Press RETURN to get started.







%CDP-4-NATIVE_VLAN_MISMATCH: Native VLAN mismatch discovered on FastEthernet0/4 (30), with Switch FastEthernet0/1 (1).

%CDP-4-NATIVE_VLAN_MISMATCH: Native VLAN mismatch discovered on FastEthernet0/3 (20), with Switch FastEthernet0/1 (1).

%CDP-4-NATIVE_VLAN_MISMATCH: Native VLAN mismatch discovered on FastEthernet0/5 (40), with Switch FastEthernet0/1 (1).


Switch>enable
Switch#
Switch#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)#interface FastEthernet0/1
Switch(config-if)#
Switch(config-if)#switchport mode trunk

Switch(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to down

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1, changed state to up

%CDP-4-NATIVE_VLAN_MISMATCH: Native VLAN mismatch discovered on FastEthernet0/2 (10), with Switch FastEthernet0/1 (1).
```

Assigning IP addresses

VLAN 10



IP Configuration

DHCP [✓ Static]

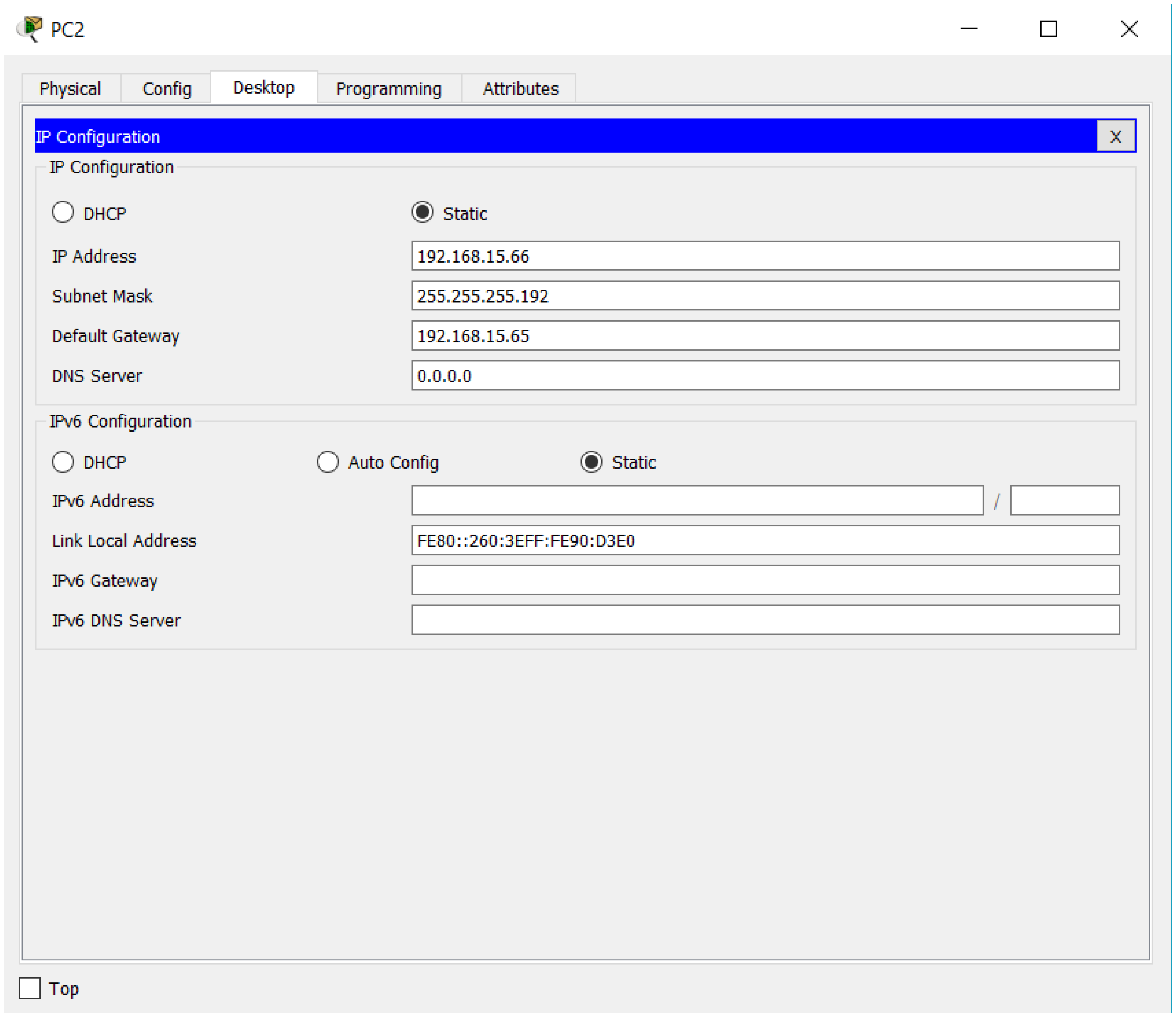
IP Address	192.168.15.2
Subnet Mask	255.255.255.192
Default Gateway	192.168.15.1
DNS Server	0.0.0.0

IPv6 Configuration

DHCP Auto Config [✓ Static]

IPv6 Address	/
Link Local Address	FE80::2D0:FFFF:FE5A:6E03
IPv6 Gateway	
IPv6 DNS Server	

VLAN 20



PC2

IP Configuration

IP Configuration

DHCP [✓ Static]

IP Address	192.168.15.66
Subnet Mask	255.255.255.192
Default Gateway	192.168.15.65
DNS Server	0.0.0.0

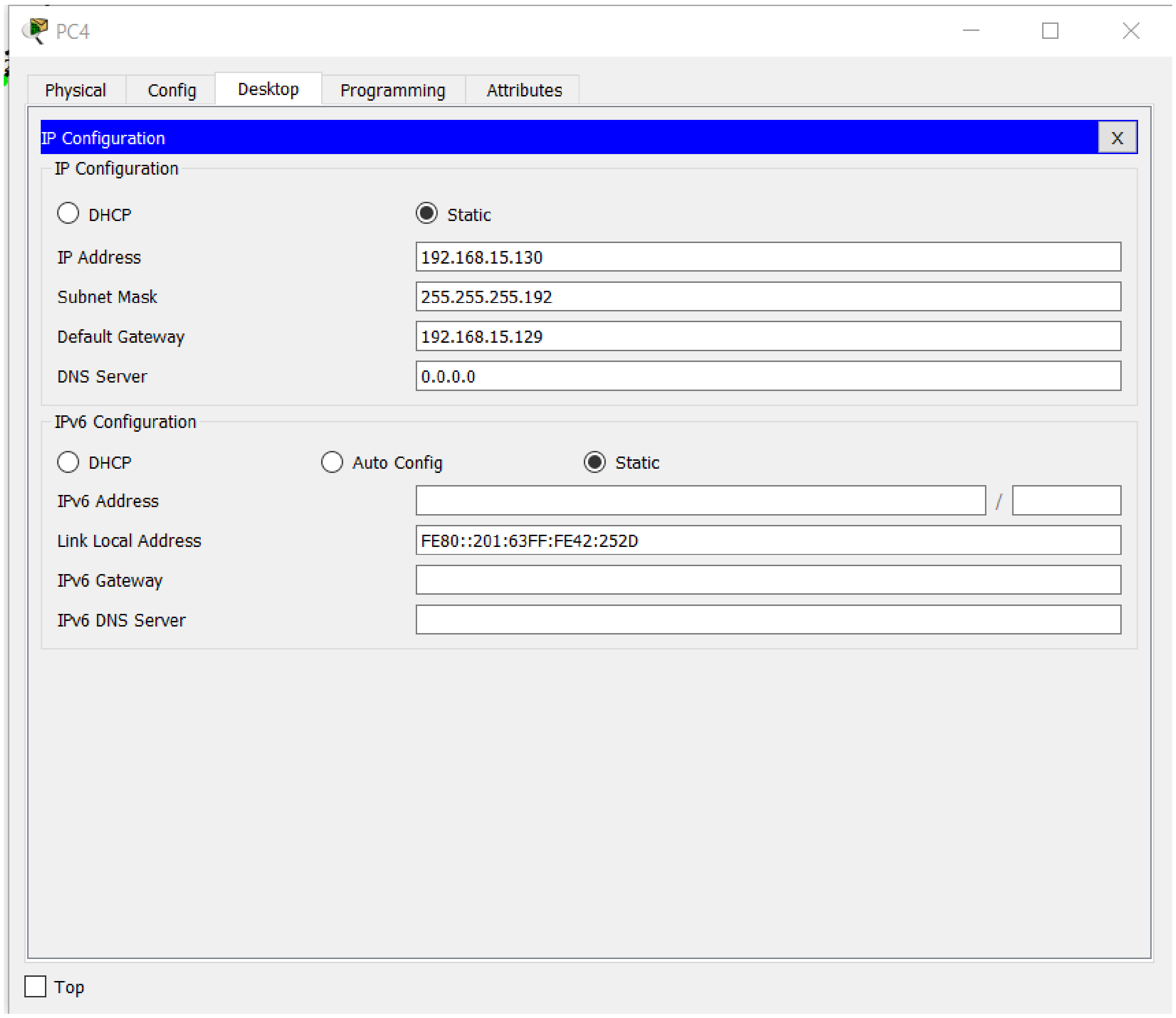
IPv6 Configuration

DHCP Auto Config [✓ Static]

IPv6 Address	/
Link Local Address	FE80::260:3EFF:FE90:D3E0
IPv6 Gateway	
IPv6 DNS Server	

Top

VLAN 30


PC4

Physical | Config | Desktop | Programming | Attributes

IP Configuration

IP Configuration

DHCP [✓ Static]

IP Address	192.168.15.130
Subnet Mask	255.255.255.192
Default Gateway	192.168.15.129
DNS Server	0.0.0.0

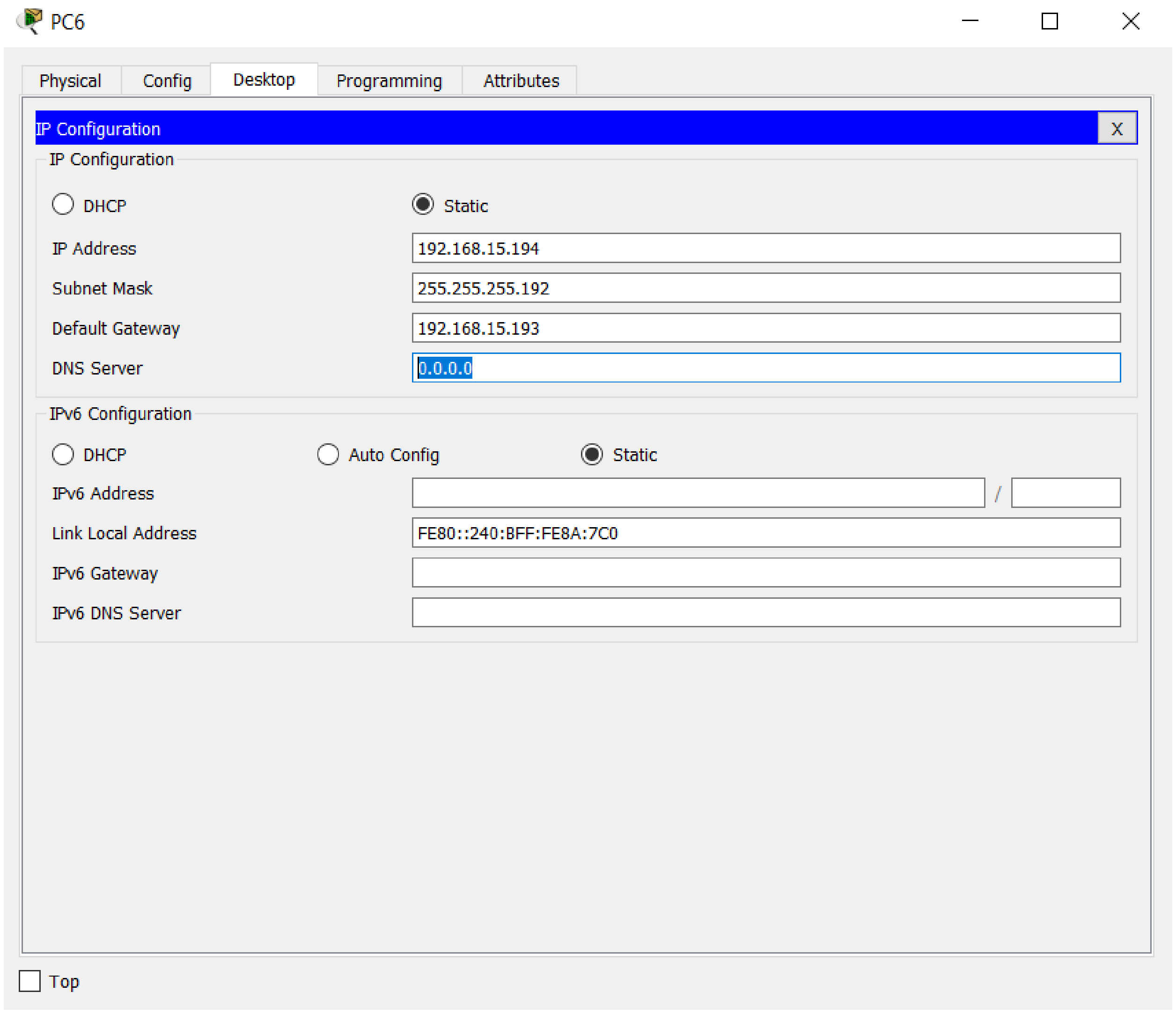
IPv6 Configuration

DHCP Auto Config [✓ Static]

IPv6 Address	/
Link Local Address	FE80::201:63FF:FE42:252D
IPv6 Gateway	
IPv6 DNS Server	

Top

VLAN 40



IP Configuration

DHCP [✓ Static]

IP Address	192.168.15.194
Subnet Mask	255.255.255.192
Default Gateway	192.168.15.193
DNS Server	0.0.0.0

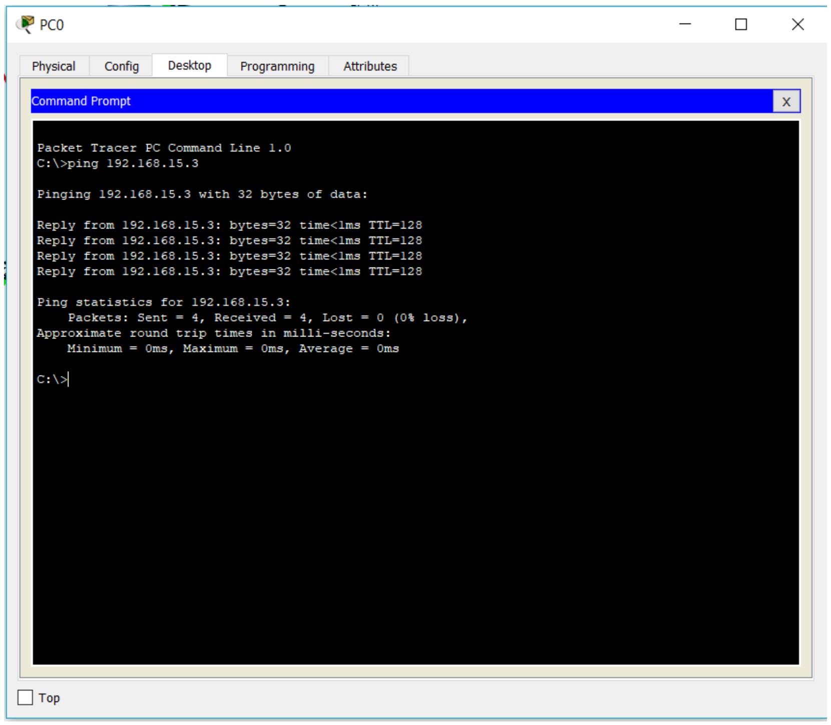
IPv6 Configuration

DHCP Auto Config [✓ Static]

IPv6 Address	/
Link Local Address	FE80::240:BFF:FE8A:7C0
IPv6 Gateway	
IPv6 DNS Server	

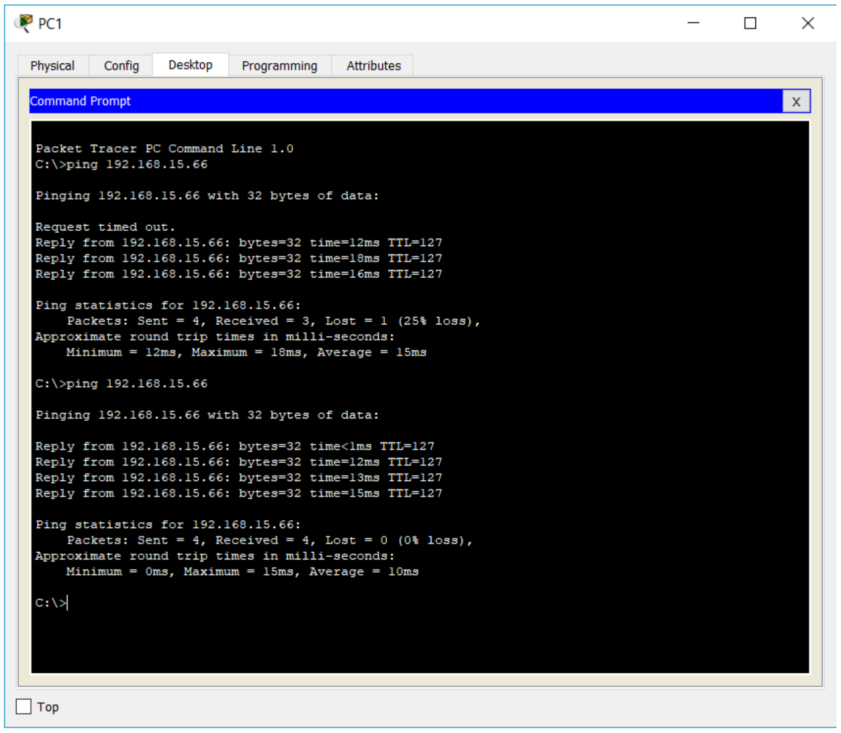
Ping command outputs

Ping in same VLAN



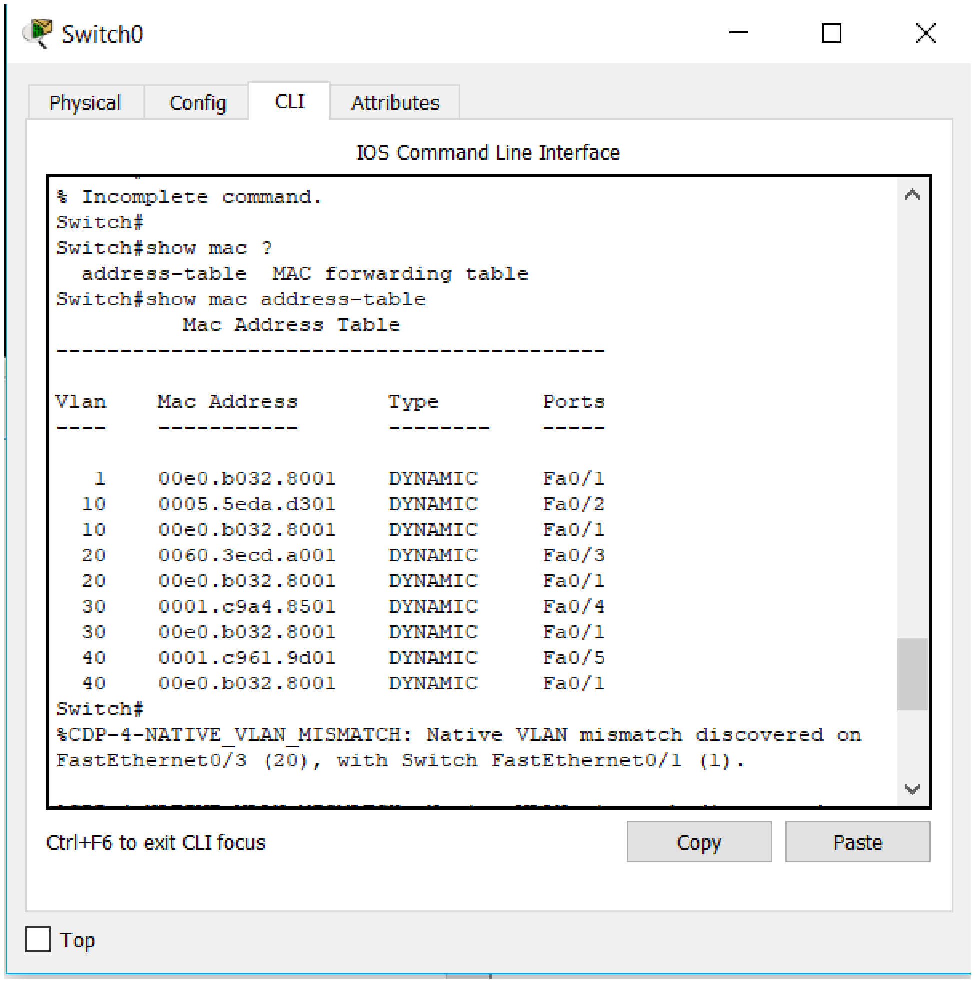
```
Packet Tracer PC Command Line 1.0
C:\>ping 192.168.15.3

Pinging 192.168.15.3 with 32 bytes of data:

Reply from 192.168.15.3: bytes=32 time<1ms TTL=128
Reply from 192.168.15.3: bytes=32 time<1ms TTL=128
Reply from 192.168.15.3: bytes=32 time<1ms TTL=128
Reply from 192.168.15.3: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.15.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

Ping in different VLAN



```
Packet Tracer PC Command Line 1.0
C:\>ping 192.168.15.66

Pinging 192.168.15.66 with 32 bytes of data:

Request timed out.
Reply from 192.168.15.66: bytes=32 time=12ms TTL=127
Reply from 192.168.15.66: bytes=32 time=18ms TTL=127
Reply from 192.168.15.66: bytes=32 time=16ms TTL=127

Ping statistics for 192.168.15.66:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 12ms, Maximum = 18ms, Average = 15ms

C:\>ping 192.168.15.66

Pinging 192.168.15.66 with 32 bytes of data:

Reply from 192.168.15.66: bytes=32 time<1ms TTL=127
Reply from 192.168.15.66: bytes=32 time=12ms TTL=127
Reply from 192.168.15.66: bytes=32 time=13ms TTL=127
Reply from 192.168.15.66: bytes=32 time=15ms TTL=127

Ping statistics for 192.168.15.66:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 15ms, Average = 10ms

C:\>
```

MAC-address table information



```
% Incomplete command.
Switch#
Switch#show mac ?
  address-table  MAC forwarding table
Switch#show mac address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----

   1    00e0.b032.8001    DYNAMIC     Fa0/1
  10    0005.5eda.d301    DYNAMIC     Fa0/2
  10    00e0.b032.8001    DYNAMIC     Fa0/1
  20    0060.3ecd.a001    DYNAMIC     Fa0/3
  20    00e0.b032.8001    DYNAMIC     Fa0/1
  30    0001.c9a4.8501    DYNAMIC     Fa0/4
  30    00e0.b032.8001    DYNAMIC     Fa0/1
  40    0001.c961.9d01    DYNAMIC     Fa0/5
  40    00e0.b032.8001    DYNAMIC     Fa0/1
Switch#
%CDP-4-NATIVE_VLAN_MISMATCH: Native VLAN mismatch discovered on
FastEthernet0/3 (20), with Switch FastEthernet0/1 (1).
```

A MAC address table is used to determine where to forward traffic on a LAN in Ethernet switches.

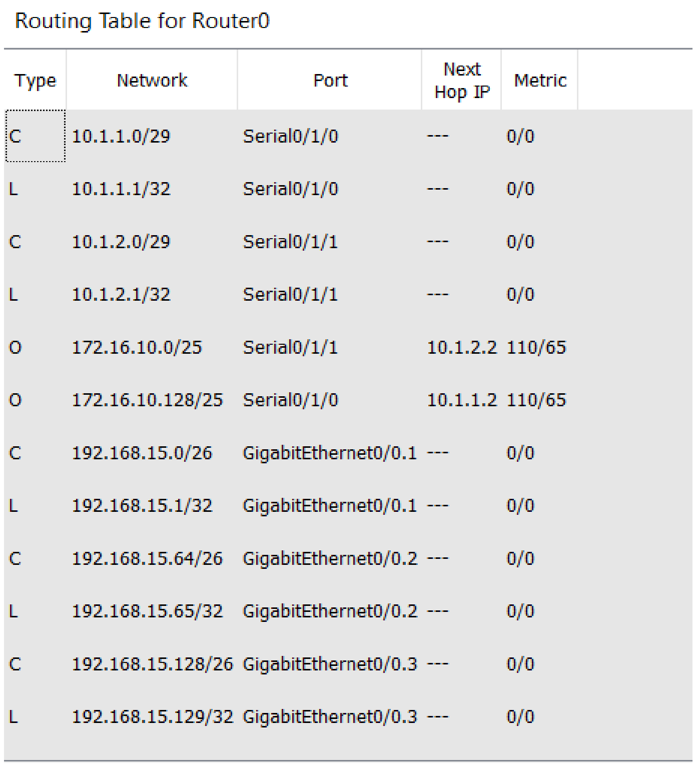
When a packet comes to this router, it will check the mac address table and based on the information it will switch the packet to the intended destination.

Configuring WAN and Routing Architecture

A Wide Area Network (WAN) is a collection of computer and network resources that are connected through a network over a large geographical area, WANs are typically connected using the Internet or special connections provided by service providers.

Routing tables

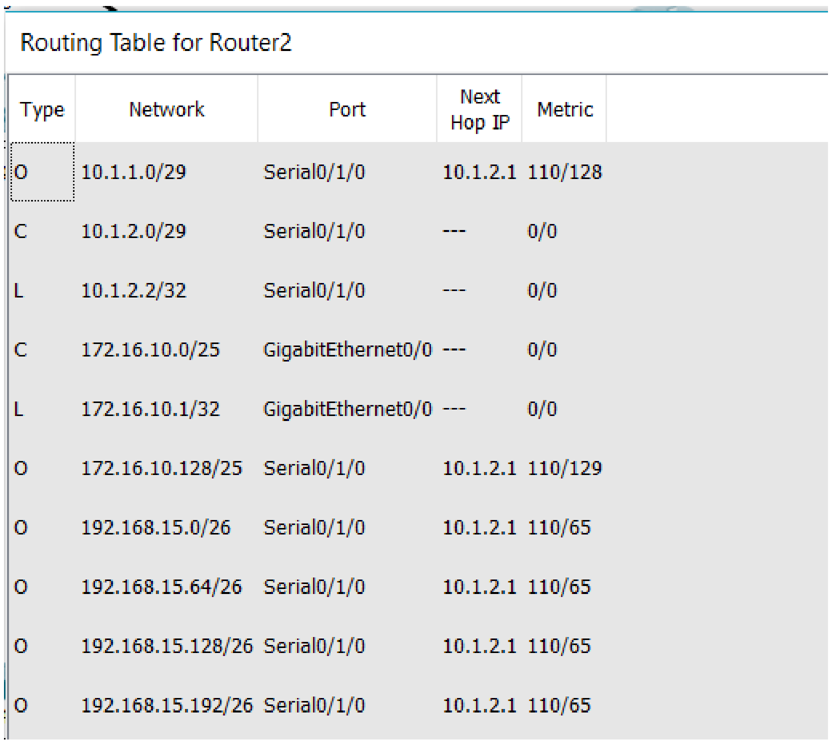
Router0



Routing Table for Router0

Type	Network	Port	Next Hop IP	Metric
C	10.1.1.0/29	Serial0/1/0	---	0/0
L	10.1.1.1/32	Serial0/1/0	---	0/0
C	10.1.2.0/29	Serial0/1/1	---	0/0
L	10.1.2.1/32	Serial0/1/1	---	0/0
O	172.16.10.0/25	Serial0/1/1	10.1.2.2	110/65
O	172.16.10.128/25	Serial0/1/0	10.1.1.2	110/65
C	192.168.15.0/26	GigabitEthernet0/0.1	---	0/0
L	192.168.15.1/32	GigabitEthernet0/0.1	---	0/0
C	192.168.15.64/26	GigabitEthernet0/0.2	---	0/0
L	192.168.15.65/32	GigabitEthernet0/0.2	---	0/0
C	192.168.15.128/26	GigabitEthernet0/0.3	---	0/0
L	192.168.15.129/32	GigabitEthernet0/0.3	---	0/0

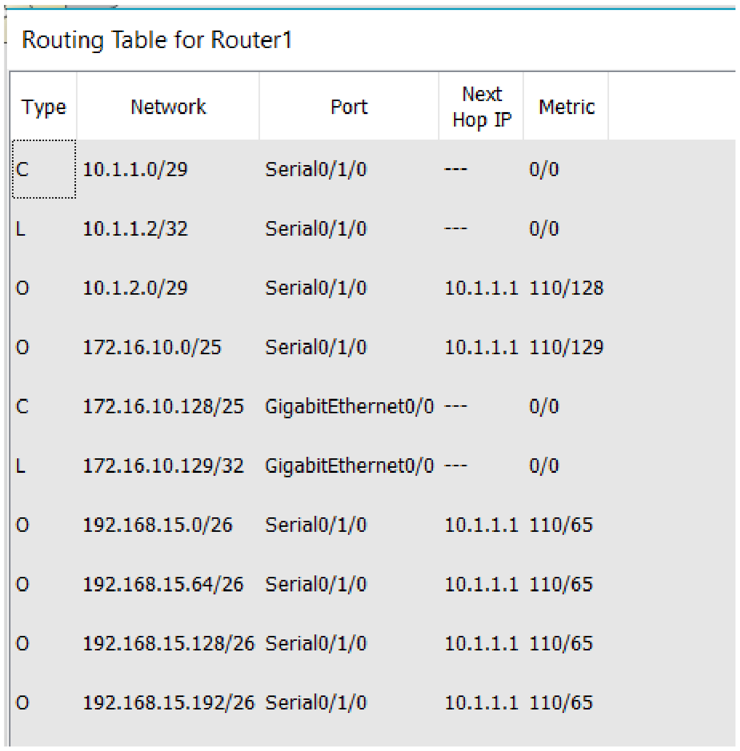
Router1



Routing Table for Router2

Type	Network	Port	Next Hop IP	Metric
O	10.1.1.0/29	Serial0/1/0	10.1.2.1	110/128
C	10.1.2.0/29	Serial0/1/0	---	0/0
L	10.1.2.2/32	Serial0/1/0	---	0/0
C	172.16.10.0/25	GigabitEthernet0/0	---	0/0
L	172.16.10.1/32	GigabitEthernet0/0	---	0/0
O	172.16.10.128/25	Serial0/1/0	10.1.2.1	110/129
O	192.168.15.0/26	Serial0/1/0	10.1.2.1	110/65
O	192.168.15.64/26	Serial0/1/0	10.1.2.1	110/65
O	192.168.15.128/26	Serial0/1/0	10.1.2.1	110/65
O	192.168.15.192/26	Serial0/1/0	10.1.2.1	110/65

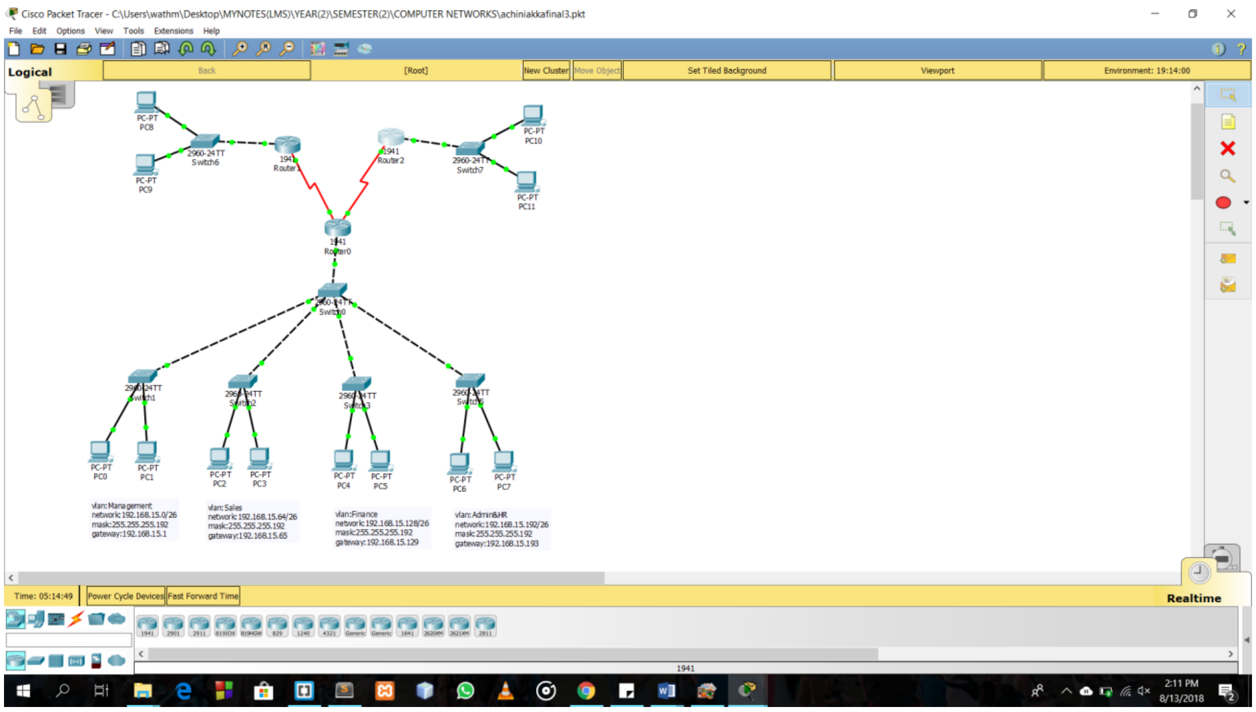
Router2



Routing Table for Router1

Type	Network	Port	Next Hop IP	Metric
C	10.1.1.0/29	Serial0/1/0	---	0/0
L	10.1.1.2/32	Serial0/1/0	---	0/0
O	10.1.2.0/29	Serial0/1/0	10.1.1.1	110/128
O	172.16.10.0/25	Serial0/1/0	10.1.1.1	110/129
C	172.16.10.128/25	GigabitEthernet0/0	---	0/0
L	172.16.10.129/32	GigabitEthernet0/0	---	0/0
O	192.168.15.0/26	Serial0/1/0	10.1.1.1	110/65
O	192.168.15.64/26	Serial0/1/0	10.1.1.1	110/65
O	192.168.15.128/26	Serial0/1/0	10.1.1.1	110/65
O	192.168.15.192/26	Serial0/1/0	10.1.1.1	110/65

Connecting headquarters and other two branches via serial DTE cable



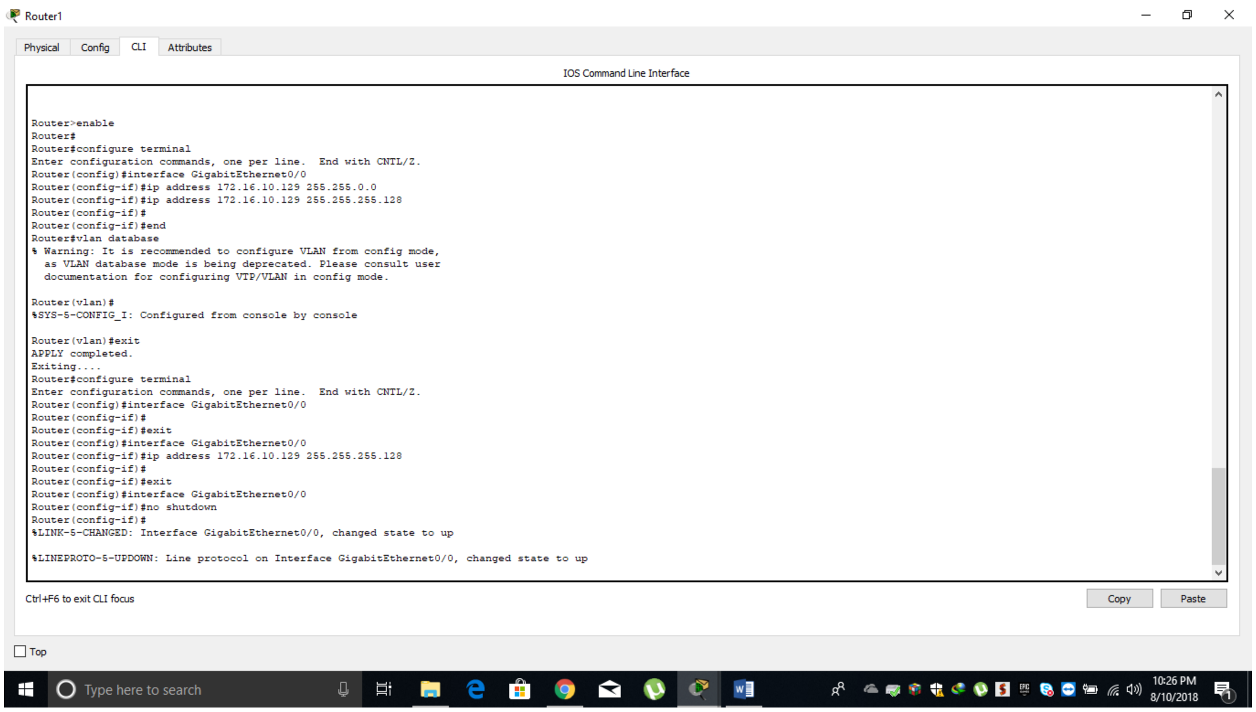
vlan:Management
network:192.168.15.0/26
mask:255.255.255.192
gateway:192.168.15.1

vlan:Sales
network:192.168.15.64/26
mask:255.255.255.192
gateway:192.168.15.65

vlan:Finance
network:192.168.15.128/26
mask:255.255.255.192
gateway:192.168.15.129

vlan:Admin&HR
network:192.168.15.192/26
mask:255.255.255.192
gateway:192.168.15.193

Configuring OSPF as the routing protocol in all routers


Router1

Physical | Config | CLI | Attributes

IOS Command Line Interface

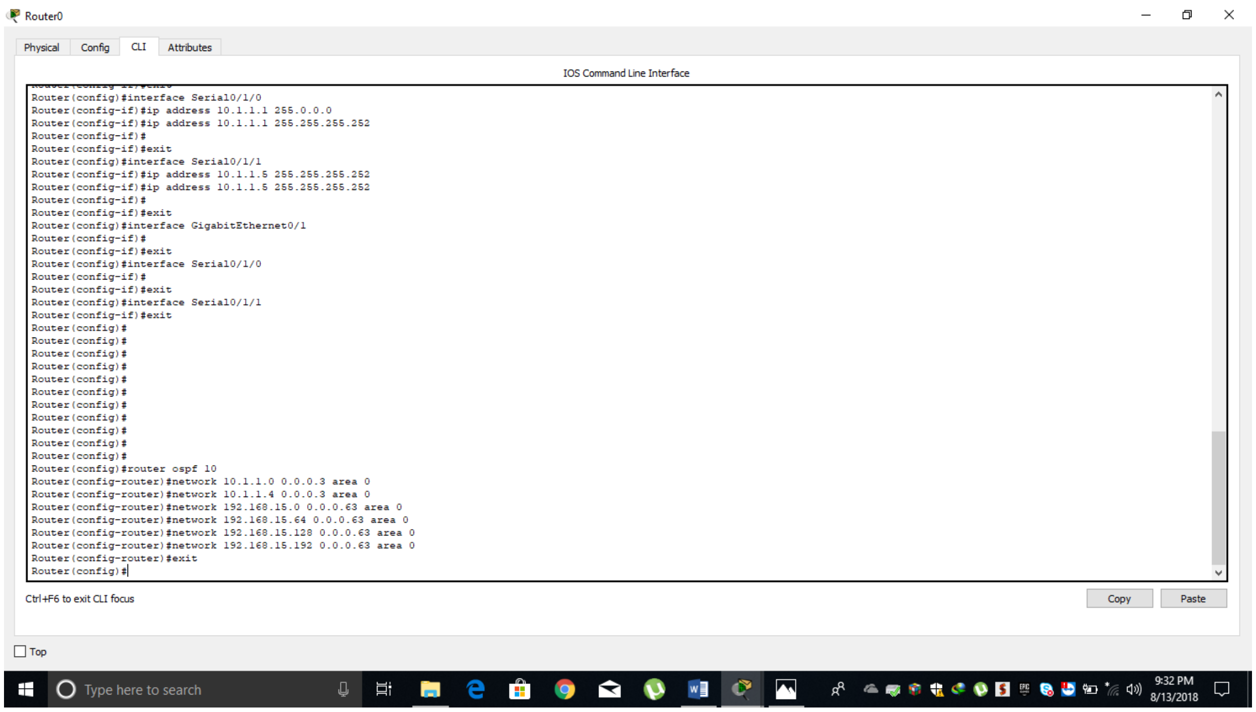
```
Router>enable
Router#
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface GigabitEthernet0/0
Router(config-if)#ip address 172.16.10.129 255.255.0.0
Router(config-if)#ip address 172.16.10.129 255.255.255.128
Router(config-if)#
Router(config-if)#end
Router#vlan database
% Warning: It is recommended to configure VLAN from config mode,
  as VLAN database mode is being deprecated. Please consult user
  documentation for configuring VTP/VLAN in config mode.

Router(vlan)#
%SYS-5-CONFIG_I: Configured from console by console

Router(vlan)#exit
APPLY completed.
Exiting....
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface GigabitEthernet0/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0
Router(config-if)#ip address 172.16.10.129 255.255.255.128
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0, changed state to up
```

Ctrl+F6 to exit CLI focus

Copy Paste

Top

10:26 PM 8/10/2018


Router0

Physical | Config | CLI | Attributes

IOS Command Line Interface

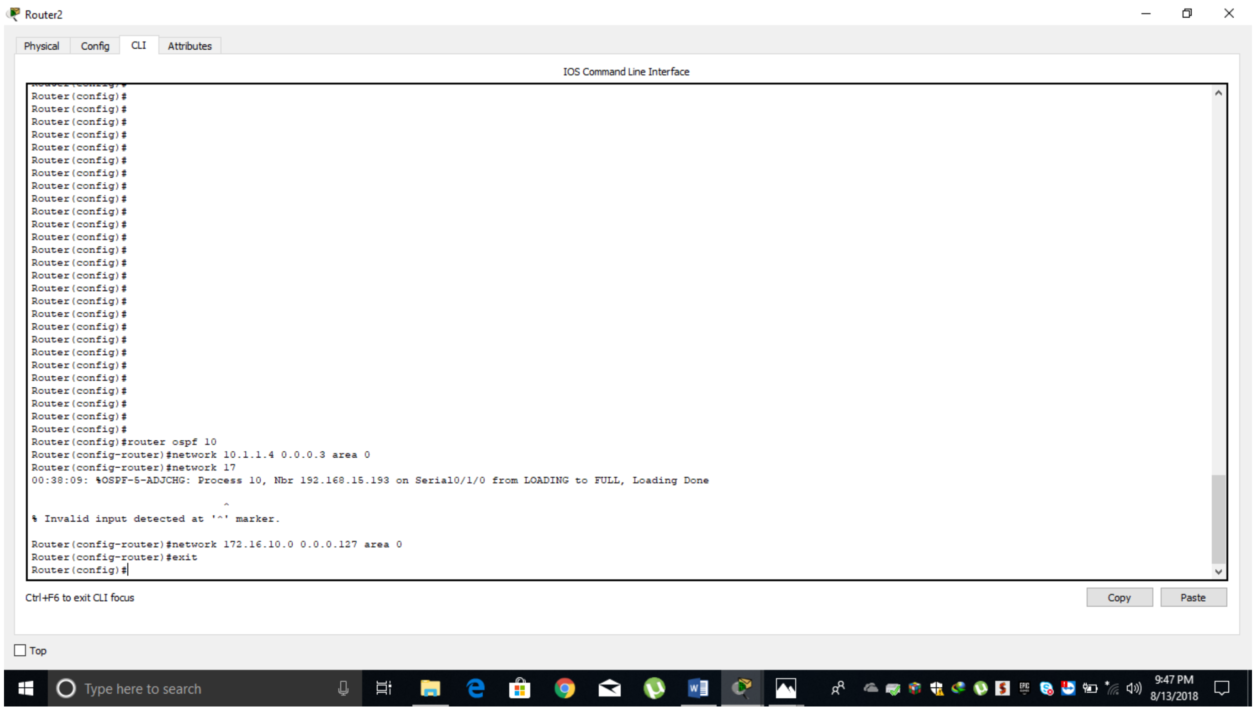
```
Router(config)#interface Serial0/1/0
Router(config-if)#ip address 10.1.1.1 255.0.0.0
Router(config-if)#ip address 10.1.1.1 255.255.255.252
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/1/1
Router(config-if)#ip address 10.1.1.5 255.255.255.252
Router(config-if)#ip address 10.1.1.5 255.255.255.252
Router(config-if)#
Router(config-if)#exit
Router(config)#interface GigabitEthernet0/1
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/1/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/1/1
Router(config-if)#exit
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#router ospf 10
Router(config-router)#network 10.1.1.0 0.0.0.3 area 0
Router(config-router)#network 10.1.1.4 0.0.0.3 area 0
Router(config-router)#network 192.168.15.0 0.0.0.63 area 0
Router(config-router)#network 192.168.15.64 0.0.0.63 area 0
Router(config-router)#network 192.168.15.128 0.0.0.63 area 0
Router(config-router)#network 192.168.15.192 0.0.0.63 area 0
Router(config-router)#exit
Router(config)#
```

Ctrl+F6 to exit CLI focus

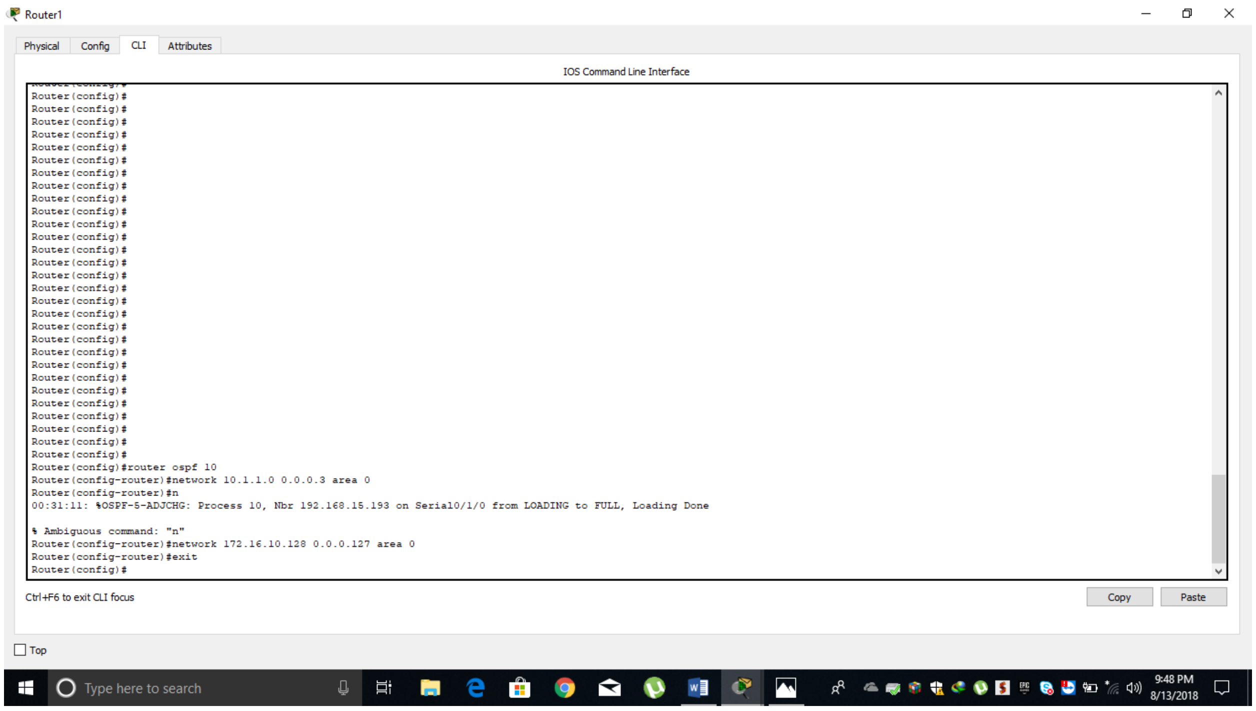
Copy Paste

Top

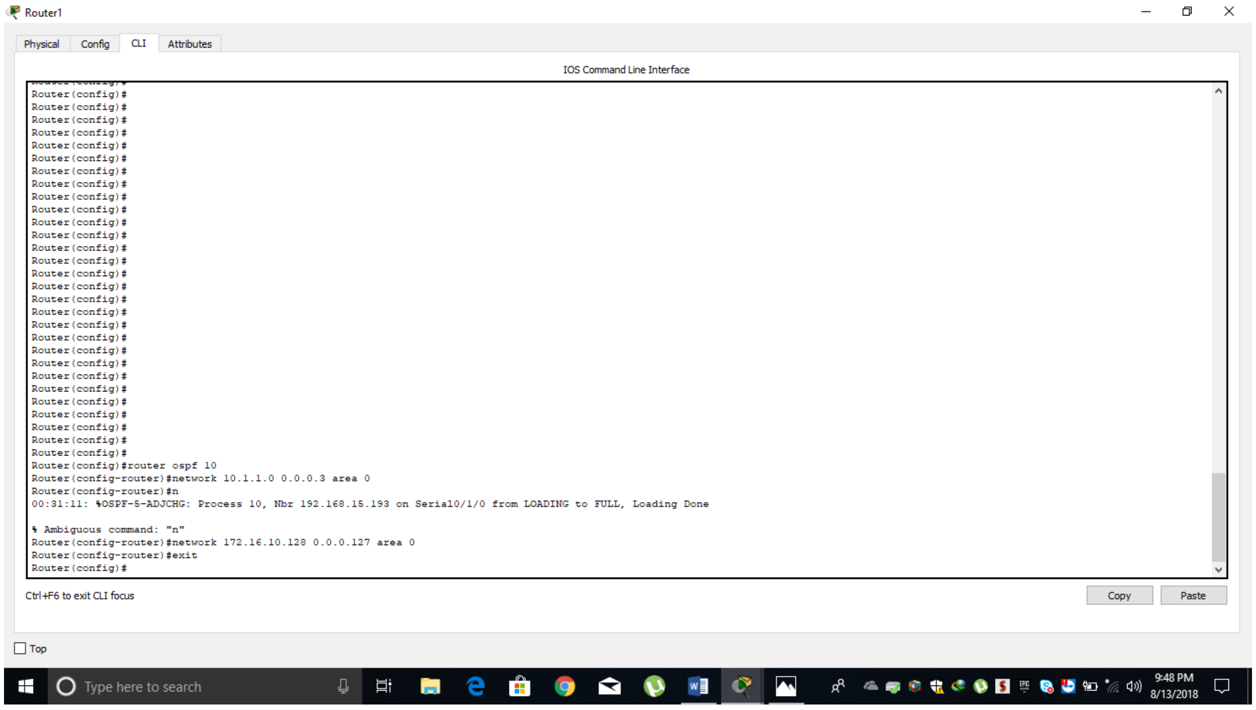
9:32 PM 8/13/2018



```
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#router ospf 10
Router(config-router)#network 10.1.1.4 0.0.0.3 area 0
Router(config-router)#network 17
00:38:09: %OSPF-5-ADJCHG: Process 10, Nbr 192.168.15.193 on Serial0/1/0 from LOADING to FULL, Loading Done

                              ^
% Invalid input detected at '^' marker.

Router(config-router)#network 172.16.10.0 0.0.0.127 area 0
Router(config-router)#exit
Router(config)#
```



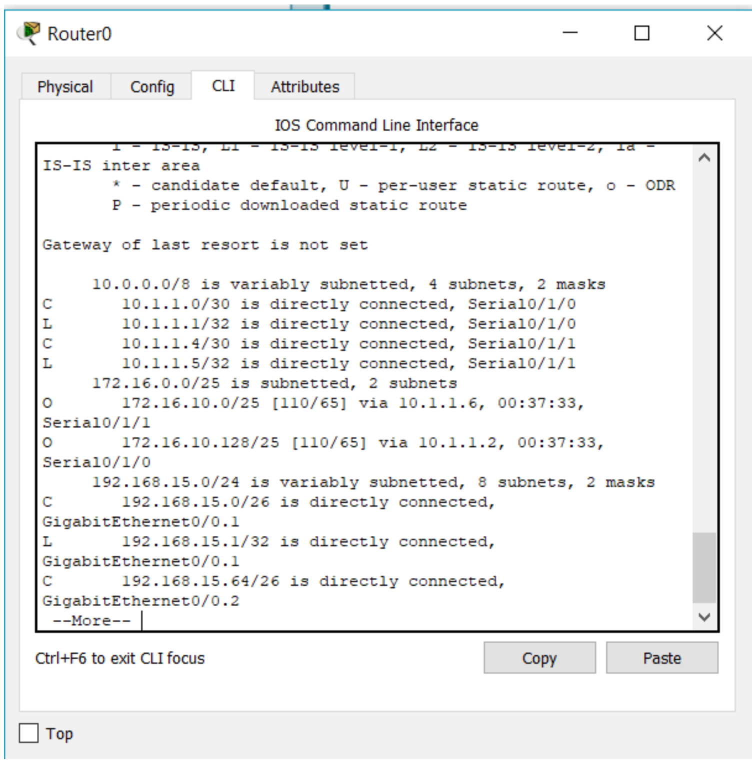
```
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#router ospf 10
Router(config-router)#network 10.1.1.0 0.0.0.3 area 0
Router(config-router)#n
00:31:11: %OSPF-5-ADJCHG: Process 10, Nbr 192.168.15.193 on Serial0/1/0 from LOADING to FULL, Loading Done

% Ambiguous command: "n"
Router(config-router)#network 172.16.10.128 0.0.0.127 area 0
Router(config-router)#exit
Router(config)#
```



```
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#router ospf 10
Router(config-router)#network 10.1.1.0 0.0.0.3 area 0
Router(config-router)#n
00:31:11: %OSPF-5-ADJCHG: Process 10, Nbr 192.168.15.193 on Serial0/1/0 from LOADING to FULL, Loading Done

% Ambiguous command: "n"
Router(config-router)#network 172.16.10.128 0.0.0.127 area 0
Router(config-router)#exit
Router(config)#
```

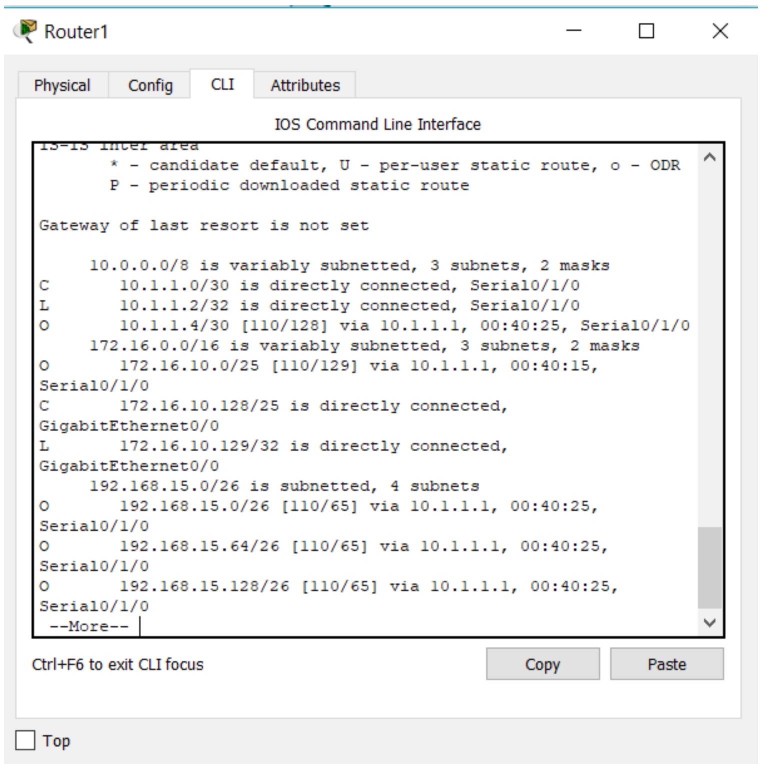
Routing tables output

Router 0



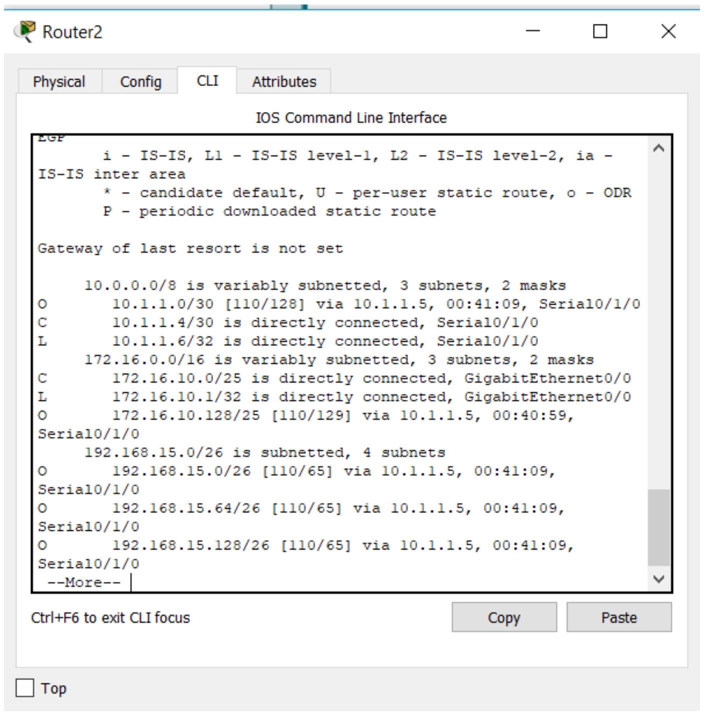
```
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia -
IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C       10.1.1.0/30 is directly connected, Serial0/1/0
L       10.1.1.1/32 is directly connected, Serial0/1/0
C       10.1.1.4/30 is directly connected, Serial0/1/1
L       10.1.1.5/32 is directly connected, Serial0/1/1
     172.16.0.0/25 is subnetted, 2 subnets
O       172.16.10.0/25 [110/65] via 10.1.1.6, 00:37:33,
Serial0/1/1
O       172.16.10.128/25 [110/65] via 10.1.1.2, 00:37:33,
Serial0/1/0
     192.168.15.0/24 is variably subnetted, 8 subnets, 2 masks
C       192.168.15.0/26 is directly connected,
GigabitEthernet0/0.1
L       192.168.15.1/32 is directly connected,
GigabitEthernet0/0.1
C       192.168.15.64/26 is directly connected,
GigabitEthernet0/0.2
 --More--
```

Router 1



```
IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C       10.1.1.0/30 is directly connected, Serial0/1/0
L       10.1.1.2/32 is directly connected, Serial0/1/0
O       10.1.1.4/30 [110/128] via 10.1.1.1, 00:40:25, Serial0/1/0
     172.16.0.0/16 is variably subnetted, 3 subnets, 2 masks
O       172.16.10.0/25 [110/129] via 10.1.1.1, 00:40:15,
Serial0/1/0
C       172.16.10.128/25 is directly connected,
GigabitEthernet0/0
L       172.16.10.129/32 is directly connected,
GigabitEthernet0/0
     192.168.15.0/26 is subnetted, 4 subnets
O       192.168.15.0/26 [110/65] via 10.1.1.1, 00:40:25,
Serial0/1/0
O       192.168.15.64/26 [110/65] via 10.1.1.1, 00:40:25,
Serial0/1/0
O       192.168.15.128/26 [110/65] via 10.1.1.1, 00:40:25,
Serial0/1/0
 --More--
```

Router 2


Router2

Physical Config CLI Attributes

IOS Command Line Interface

```
EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia -
IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
O       10.1.1.0/30 [110/128] via 10.1.1.5, 00:41:09, Serial0/1/0
C       10.1.1.4/30 is directly connected, Serial0/1/0
L       10.1.1.6/32 is directly connected, Serial0/1/0
     172.16.0.0/16 is variably subnetted, 3 subnets, 2 masks
C       172.16.10.0/25 is directly connected, GigabitEthernet0/0
L       172.16.10.1/32 is directly connected, GigabitEthernet0/0
O       172.16.10.128/25 [110/129] via 10.1.1.5, 00:40:59,
Serial0/1/0
     192.168.15.0/26 is subnetted, 4 subnets
O       192.168.15.0/26 [110/65] via 10.1.1.5, 00:41:09,
Serial0/1/0
O       192.168.15.64/26 [110/65] via 10.1.1.5, 00:41:09,
Serial0/1/0
O       192.168.15.128/26 [110/65] via 10.1.1.5, 00:41:09,
Serial0/1/0
 --More--
```

Ctrl+F6 to exit CLI focus

Describe all routing table parameters

Destination: The IP address of the packet's destination

Next hop: The IP address to that the packet is forwarded

Interface: The outgoing network interface the device ought to use when forwarding the packet to consequent hop or destination

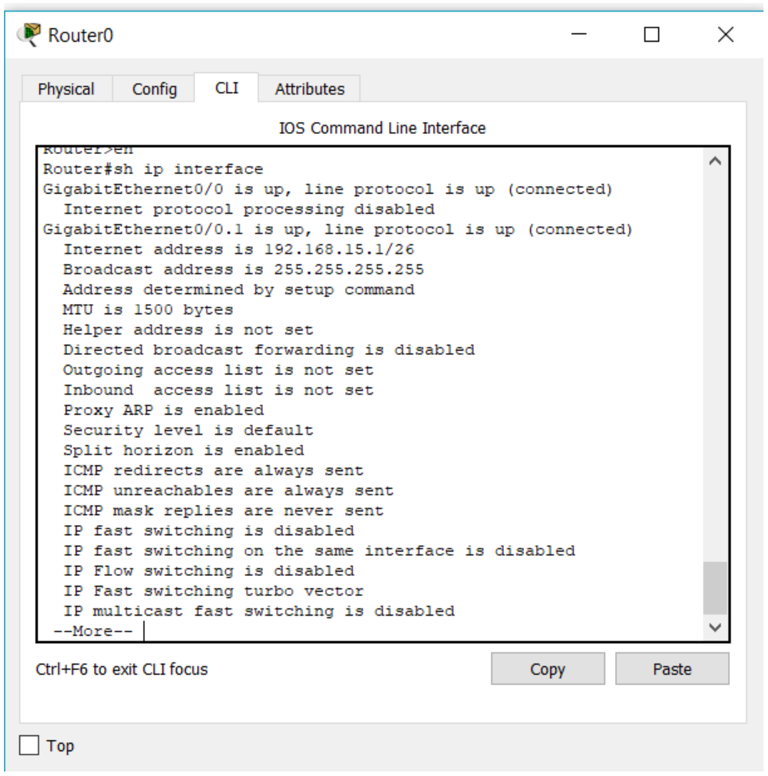
Metric: Assigns a cost to every accessible route so that the foremost cost-effective path can be chosen

Routes: Includes directly-attached subnets, indirect subnets that aren't hooked up to the device but may be accessed through one or additional hops, and default routes to use for certain forms of traffic or when info is lacking.

Cost: The link state cost of the trail to the destination. For all ways except sort a pair of external paths this describes the whole path's cost. For sort a pair of external paths, this field describes the cost of the portion of the trail internal to the AS. This cost is calculated because the total of the costs of the path's constituent links.

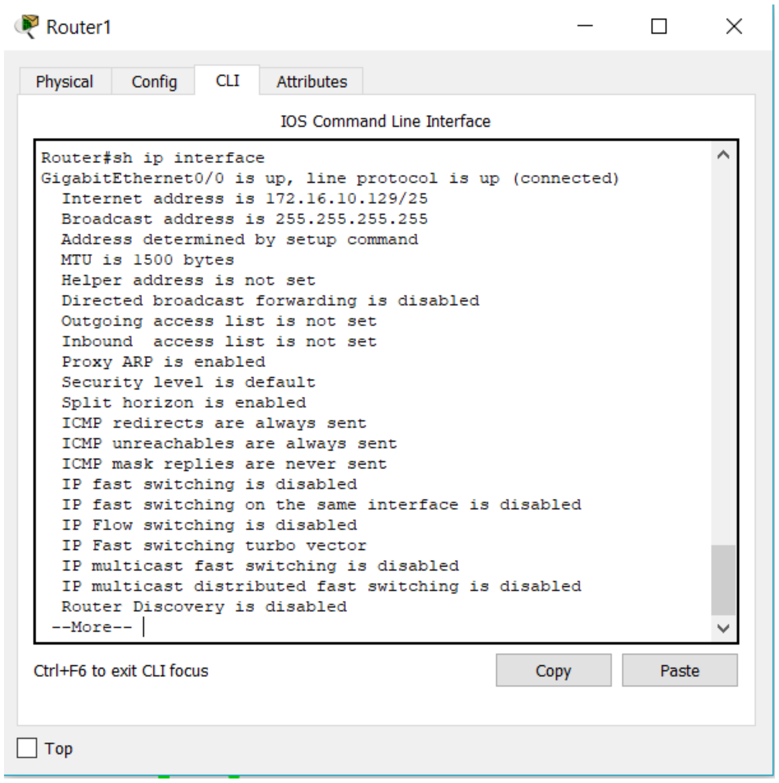
Router's interface IP address details and status(up/down)

Router0



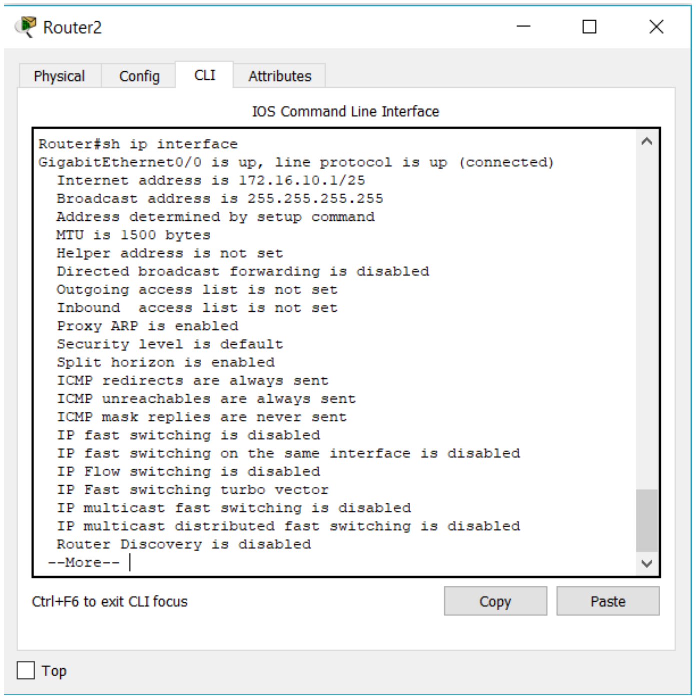
```
Router>en
Router#sh ip interface
GigabitEthernet0/0 is up, line protocol is up (connected)
  Internet protocol processing disabled
GigabitEthernet0/0.1 is up, line protocol is up (connected)
  Internet address is 192.168.15.1/26
  Broadcast address is 255.255.255.255
  Address determined by setup command
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is enabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  IP fast switching is disabled
  IP fast switching on the same interface is disabled
  IP Flow switching is disabled
  IP Fast switching turbo vector
  IP multicast fast switching is disabled
 --More--
```

Router1



```
Router#sh ip interface
GigabitEthernet0/0 is up, line protocol is up (connected)
  Internet address is 172.16.10.129/25
  Broadcast address is 255.255.255.255
  Address determined by setup command
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is enabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  IP fast switching is disabled
  IP fast switching on the same interface is disabled
  IP Flow switching is disabled
  IP Fast switching turbo vector
  IP multicast fast switching is disabled
  IP multicast distributed fast switching is disabled
  Router Discovery is disabled
 --More--
```

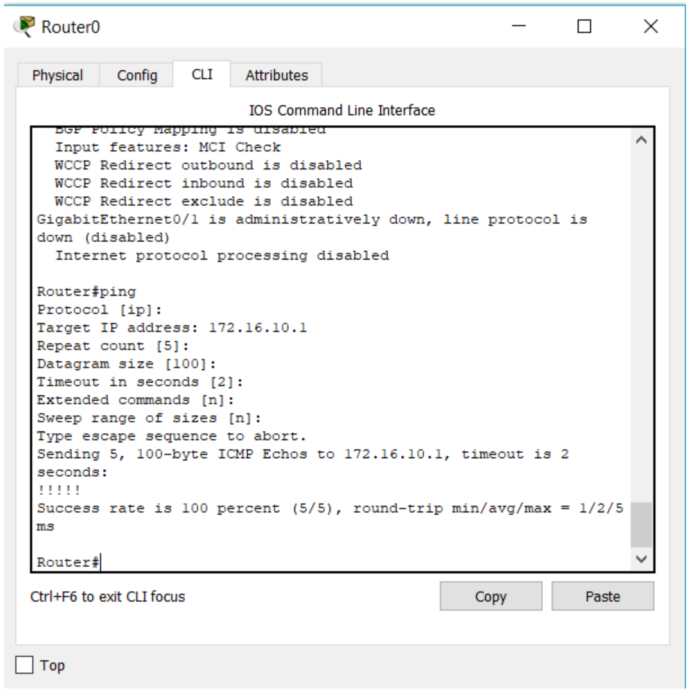
Router2



```
Router#sh ip interface
GigabitEthernet0/0 is up, line protocol is up (connected)
  Internet address is 172.16.10.1/25
  Broadcast address is 255.255.255.255
  Address determined by setup command
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is enabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  IP fast switching is disabled
  IP fast switching on the same interface is disabled
  IP Flow switching is disabled
  IP Fast switching turbo vector
  IP multicast fast switching is disabled
  IP multicast distributed fast switching is disabled
  Router Discovery is disabled
 --More--
```

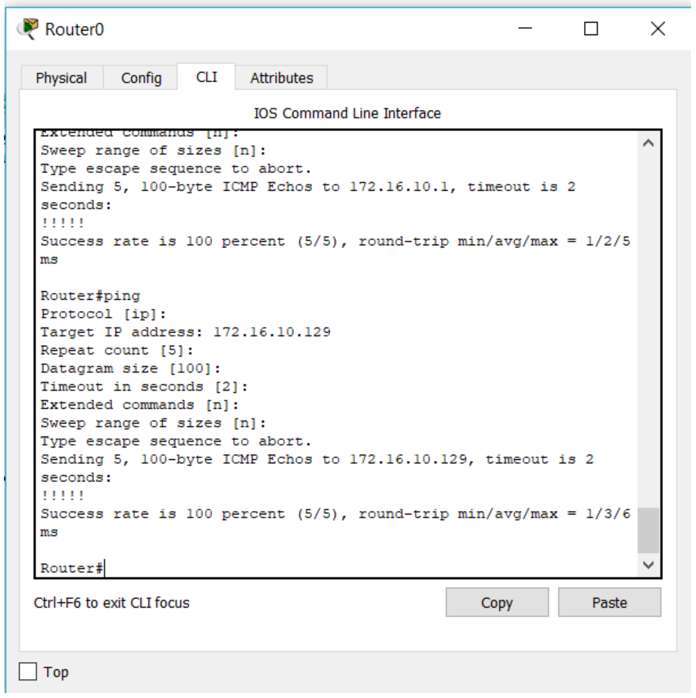
Ping command outputs form each router to other two routers

Ping from router0 to router2



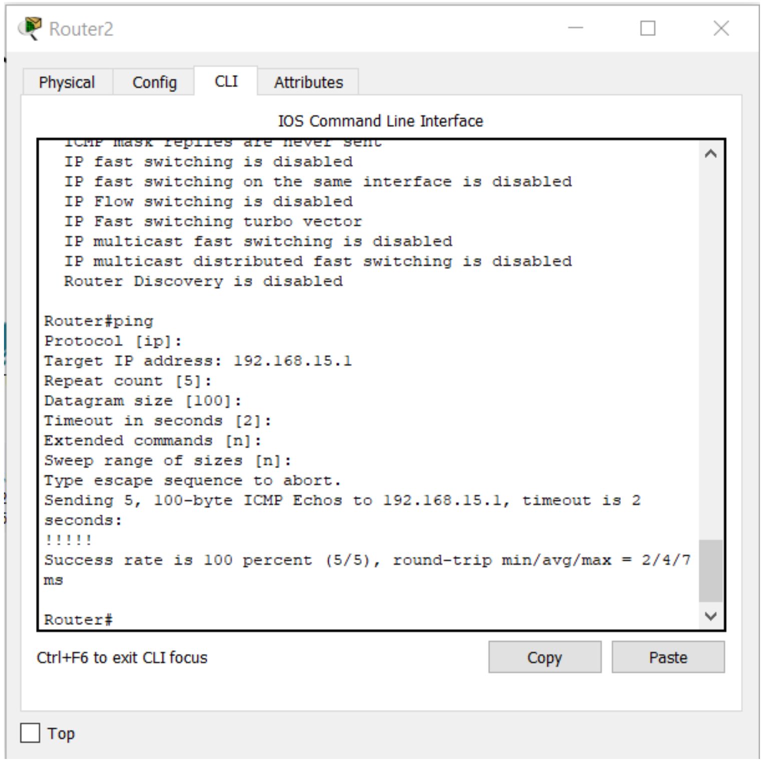
```
  BGP Policy Mapping is disabled
  Input features: MCI Check
  WCCP Redirect outbound is disabled
  WCCP Redirect inbound is disabled
  WCCP Redirect exclude is disabled
GigabitEthernet0/1 is administratively down, line protocol is
down (disabled)
  Internet protocol processing disabled

Router#ping
Protocol [ip]:
Target IP address: 172.16.10.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.10.1, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/5
ms

Router#
```

Ping from router0 to router1



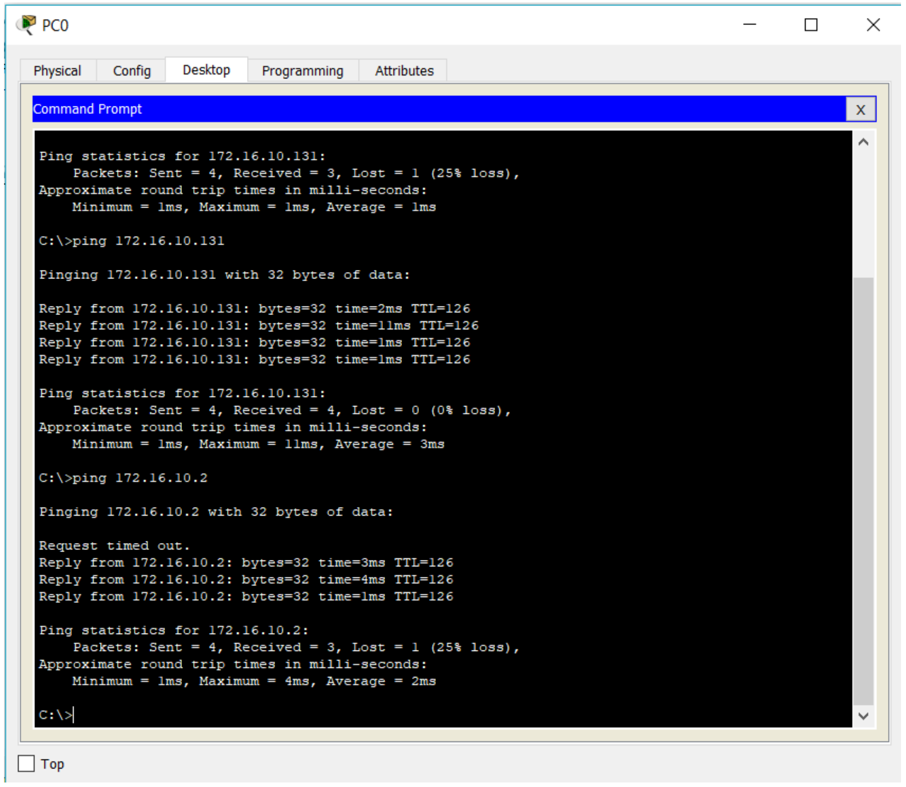
```
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.10.1, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/5
ms

Router#ping
Protocol [ip]:
Target IP address: 172.16.10.129
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.10.129, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/3/6
ms

Router#
```

Ping from router2 to router0

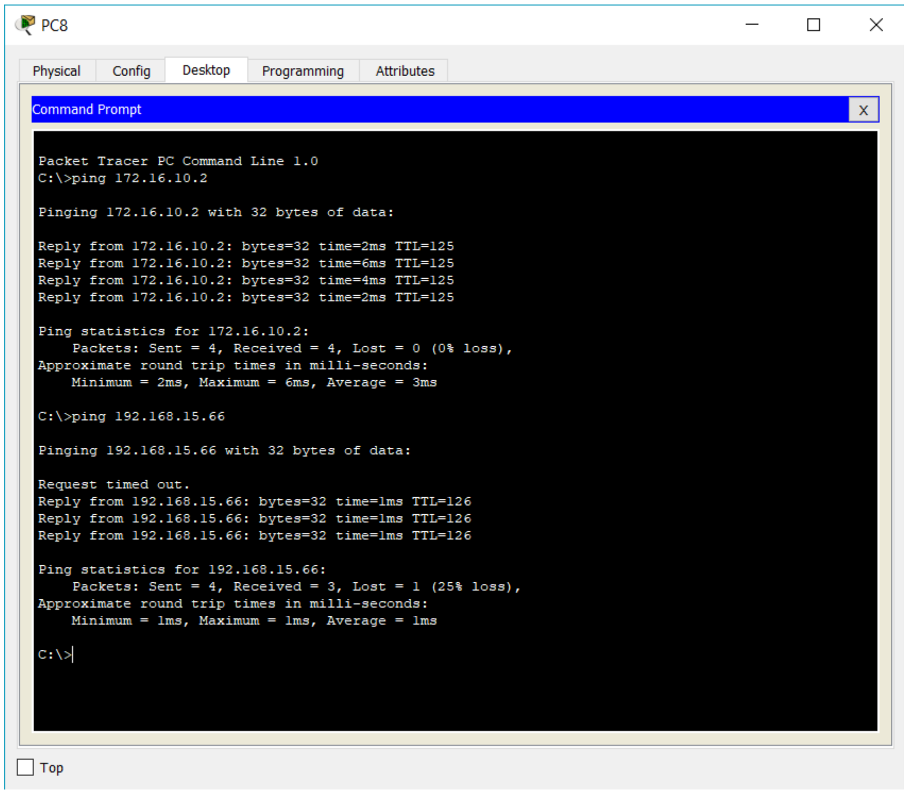


```
  IP fast switching is disabled
  IP fast switching on the same interface is disabled
  IP Flow switching is disabled
  IP Fast switching turbo vector
  IP multicast fast switching is disabled
  IP multicast distributed fast switching is disabled
  Router Discovery is disabled

Router#ping
Protocol [ip]:
Target IP address: 192.168.15.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.15.1, timeout is 2
seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/4/7
ms

Router#
```

Ping from one location PC to other two locations' few pcs and get the ping output



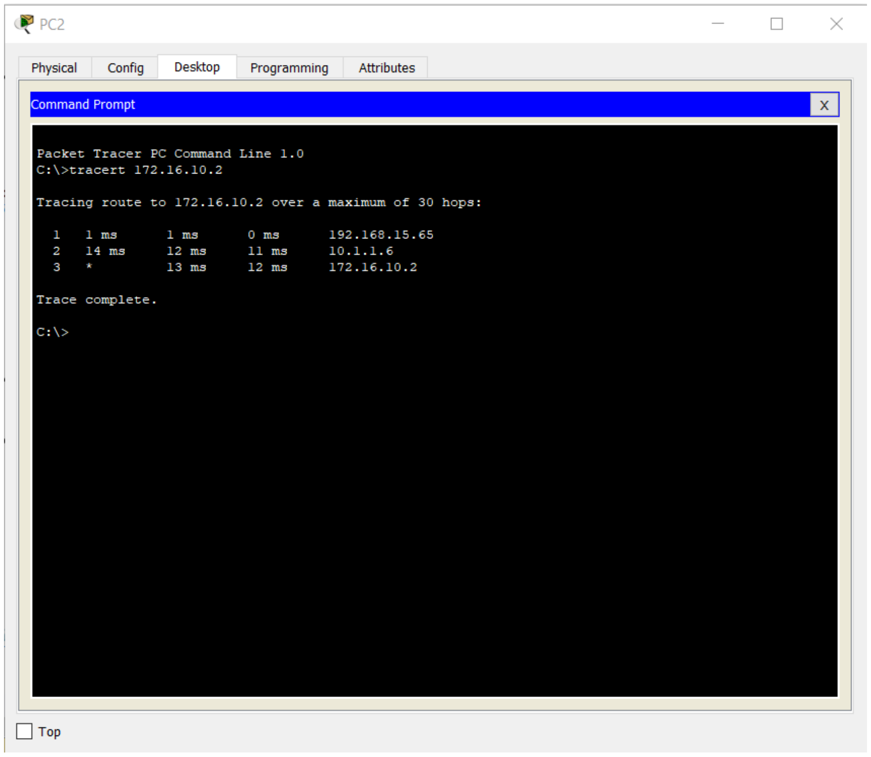
```
Ping statistics for 172.16.10.131:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms

C:\>ping 172.16.10.131

Pinging 172.16.10.131 with 32 bytes of data:

Reply from 172.16.10.131: bytes=32 time=2ms TTL=126
Reply from 172.16.10.131: bytes=32 time=11ms TTL=126
Reply from 172.16.10.131: bytes=32 time=1ms TTL=126
Reply from 172.16.10.131: bytes=32 time=1ms TTL=126

Ping statistics for 172.16.10.131:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 11ms, Average = 3ms

C:\>ping 172.16.10.2

Pinging 172.16.10.2 with 32 bytes of data:

Request timed out.
Reply from 172.16.10.2: bytes=32 time=3ms TTL=126
Reply from 172.16.10.2: bytes=32 time=4ms TTL=126
Reply from 172.16.10.2: bytes=32 time=1ms TTL=126

Ping statistics for 172.16.10.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 4ms, Average = 2ms

C:\>
```



```
Packet Tracer PC Command Line 1.0
C:\>ping 172.16.10.2

Pinging 172.16.10.2 with 32 bytes of data:

Reply from 172.16.10.2: bytes=32 time=2ms TTL=125
Reply from 172.16.10.2: bytes=32 time=6ms TTL=125
Reply from 172.16.10.2: bytes=32 time=4ms TTL=125
Reply from 172.16.10.2: bytes=32 time=2ms TTL=125

Ping statistics for 172.16.10.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 6ms, Average = 3ms

C:\>ping 192.168.15.66

Pinging 192.168.15.66 with 32 bytes of data:

Request timed out.
Reply from 192.168.15.66: bytes=32 time=1ms TTL=126
Reply from 192.168.15.66: bytes=32 time=1ms TTL=126
Reply from 192.168.15.66: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.15.66:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms

C:\>
```

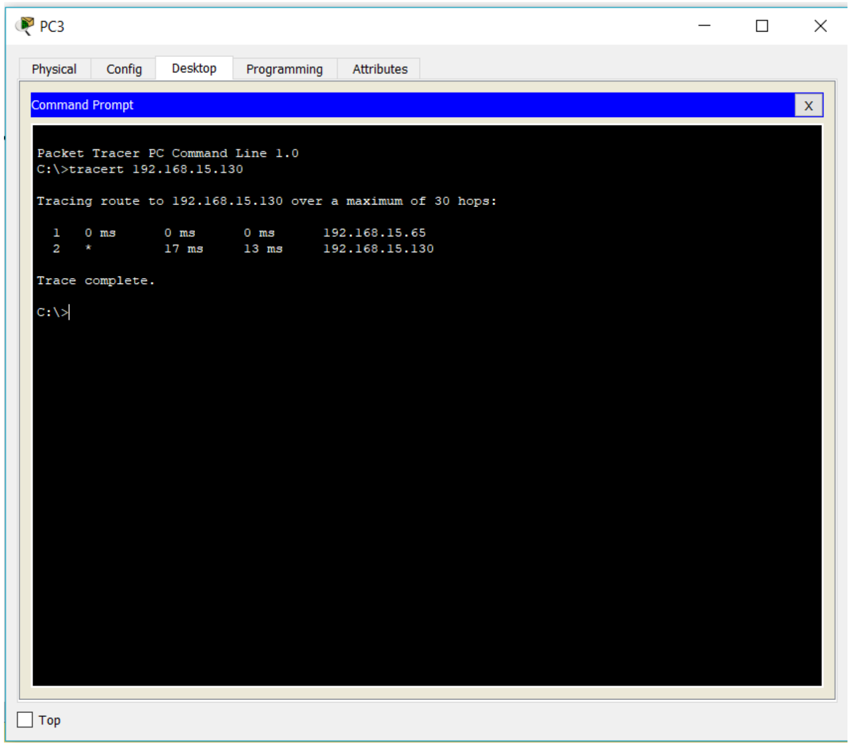
Trace Route output

Pc2 to pc10



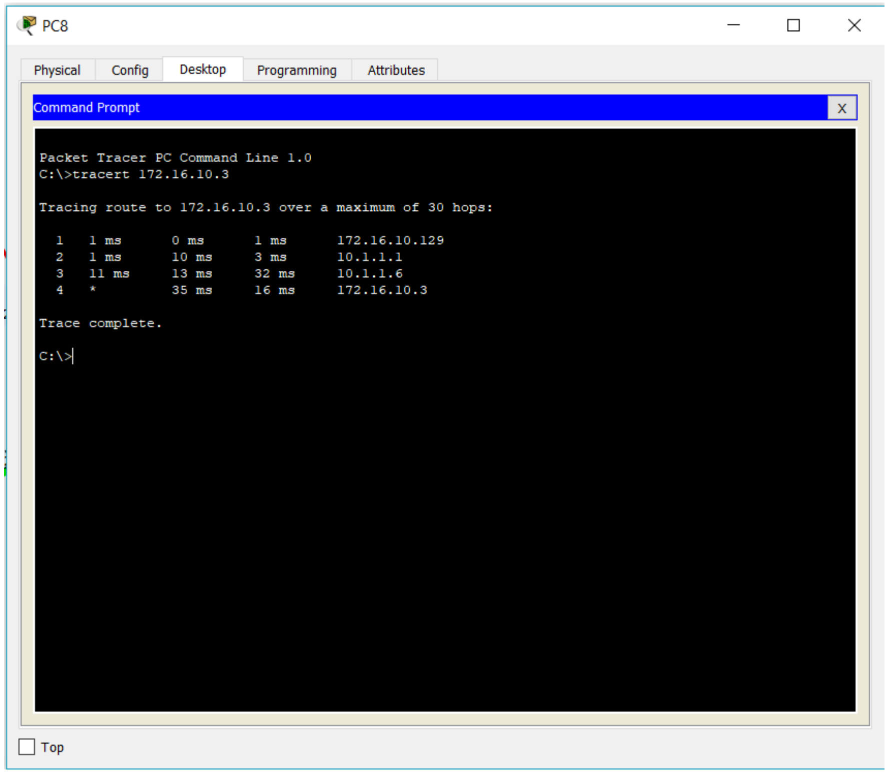
```
Packet Tracer PC Command Line 1.0
C:\>tracert 172.16.10.2

Tracing route to 172.16.10.2 over a maximum of 30 hops:

  1   1 ms      1 ms      0 ms      192.168.15.65
  2   14 ms     12 ms     11 ms     10.1.1.6
  3   *         13 ms     12 ms     172.16.10.2

Trace complete.

C:\>
```

Pc3 to pc 4



```
Packet Tracer PC Command Line 1.0
C:\>tracert 192.168.15.130

Tracing route to 192.168.15.130 over a maximum of 30 hops:

  1   0 ms      0 ms      0 ms      192.168.15.65
  2   *         17 ms     13 ms     192.168.15.130

Trace complete.

C:\>
```

Pc8 to pc11



```
Packet Tracer PC Command Line 1.0
C:\>tracert 172.16.10.3

Tracing route to 172.16.10.3 over a maximum of 30 hops:

  1   1 ms      0 ms      1 ms      172.16.10.129
  2   1 ms      10 ms     3 ms      10.1.1.1
  3   11 ms     13 ms     32 ms     10.1.1.6
  4   *         35 ms     16 ms     172.16.10.3

Trace complete.

C:\>
```

Interior gateway routing protocols

Routing is one of the most important parts of a network which enables large networks to communicate with each other. This is facilitated by a device called router. A router will pass routing information to its neighboring routers and so forth thus materializing communication between networks.

There are basically two types of routing.

- Static routing
- Dynamic routing

In static routing the network administrator or person responsible will have to manually create and maintain the routing tables in routers. If it’s a large network this task would be tedious where the network administrator will have to go to individual router and create the routing table.

A way out of this is dynamic routing which a functionality is built into the routers which allows them to share routing information on their own automatically and create routing tables for themselves. Routing protocols aid routers in this respect.

There are two major routing protocols

- Interior gateway protocol
- Exterior gateway protocol

Interior gateway protocol deals with exchange of routing information within routers of a single autonomous system where as exterior gateway protocol deals with exchange of routing information between different autonomous systems. (Kozierok, 2005)

Interior gateway protocol can be divided into two parts as

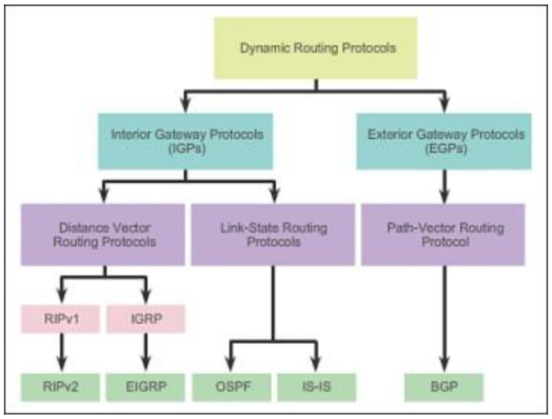
- Distance vector routing protocol
- Link state routing protocol

Distance vector routing protocol determines the routes based on the distances between routers. This distance is calculated based on the number of hops between routers. Routers using distance vector routing protocol will only have knowledge about routers connected to it and the routers which it can reach via the connected routers hence they do not have knowledge about the network topology as a whole. (Kozierok, 2005)

Link state routing protocol is different in that it selects routes based on the shortest path between networks. Routers using this routing protocol will have a map of the current network topology and based on advanced algorithms the routers will determine the shortest path for a particular route. (Kozierok, 2005)

There are several popular routing protocols belonging to these two main internal gateway protocols. Those are

- RIP v1, RIP v2, IGRP and EIGRP for distance vector routing protocol.
- OSPF and IS-IS for link state routing protocol


Dynamic Routing Protocols
Interior Gateway Protocols (IGPs)
Exterior Gateway Protocols (EGPs)
Distance Vector Routing Protocols
Link-State Routing Protocols
Path-Vector Routing Protocol
RIPv1
IGRP
RIPv2
EIGRP
OSPF
IS-IS
BGP

RIP version 1

RIP (Routing information protocol) is a distance vector routing protocol and it is one of the oldest routing protocols that exist today. It uses the Bellman-Ford algorithm which is named after its inventors to determine the routes. It is a classful protocol which broadcasts information through the UDP protocol on address 255.255.255.255. To prevent packets from looping in the network RIP uses a technique called counting to infinity which limits the maximum hop count to 15 hops per packet. RIP protocol transmits network information updates every 30 seconds amongst its neighbors hence causing heaving bandwidth consumption.

RIP version 2

Router information protocol version 2 is also a distance vector routing protocol same as its predecessor but with several improvements built into to it. It is a classless protocol which supports classless inter domain routing thus enabling RIP v2 to use variable length subnet masking. It differs from RIP v1 in that it multicasts information to its neighbors on address 224.0.0.9 instead of broadcasting. Because of this feature It supports triggered updates. (Thomas, 2015)

IGRP

Interior gateway routing protocol (IGRP) is also a distance vector routing protocol which is a proprietary protocol developed by Cisco. Compared to RIP protocol IGRP can be used in larger networks. IGRP is a classful routing protocol similar to RIP. When selecting routes IGRP uses multiple metrics such as bandwidth, load and reliability among others and uses an algorithm to select the best route possible. IGRP broadcasts update information to its neighbors every 90 seconds which is an improvement over the RIP protocol.

EIGRP

Enhanced interior gateway routing protocol (EIGRP) is also a distance vector routing protocol but with improvements to overcome the shortcomings of IGRP. It is a classless protocol which supports variable length subnet masking (VLSM). It also uses a more advanced algorithm called diffusing update algorithm (DUA) to determine the best and efficient route and also to reduce packet looping. In this protocol, if only there is a change in one of the routers in the network the neighboring routers will be notified of it instead of sending out all the information.

OSPF

Open shortest path first (OSPF) is a link state routing protocol which uses an algorithm called Dijkstra's algorithm or shortest path first algorithm to determine the best possible path to a destination. Compared to other protocols, routers using this will converge quickly and also will be less prone to errors. Routers in a network using this protocol will have a map of the current topology of the network stored in them which they will use in determining the shortest path to a destination.

IS-IS

Intermediate system to intermediate system (IS-IS) is also a link state routing protocol which is more or less similar to OSPF protocol but compared to OSPF, it is more scalable hence can be used in larger networks and it converges faster than OSPF. IS-IS uses the same shortest path first algorithm to determine the best path to a destination and shares the map of the current network topology with all the converged routers.

Advantages of using OSPF over other protocols

- Relatively the converging time is faster
- Highly scalable

- Supports variable length subnet masking (VLSM)
- Uses multicasting instead of broadcasting thus reduces bandwidth consumption
- It is an open standard not a proprietary standard
- OSPF only sends updates on routing tables instead of the entire table

Operation of OSPF routing protocol

1. Each router using link state protocol will identify its directly connected networks known as neighbors.
2. Then routers will exchange hello packets with their neighbors.
3. After each router receives hello packets from its directly connected neighbors, it will create its own link state packet (LSP) which includes the neighbors network ID, link type and bandwidth.
4. After the LSP is created, it will be flooded to all neighbors who then will save it in a link state database and forward it to their neighbors until all networks have knowledge about each other. (LSPs are sent mainly on two occasions (1) When a router starts up (2) when there is a change in topology).
5. Once all routers have information about every other router, using their link state database they create a map of the network topology.
6. Using this network topology, they calculate the shortest path to each destination by considering the costs of each route. The route with least cost will be selected.
7. Routers use link state advertisements to share this topology with other routers in the network.
8. Once the shortest paths for each route is determined, routers will create routing tables based on these paths.

Sample Proxy Server

A proxy server is a dedicated computer or a software system running on a computer that acts as an intermediary between an endpoint device, such as a computer, and another server from which a user or client is requesting a service. The proxy server may exist in the same machine as a firewall server or it may be on a separate server, which forwards requests through the firewall.

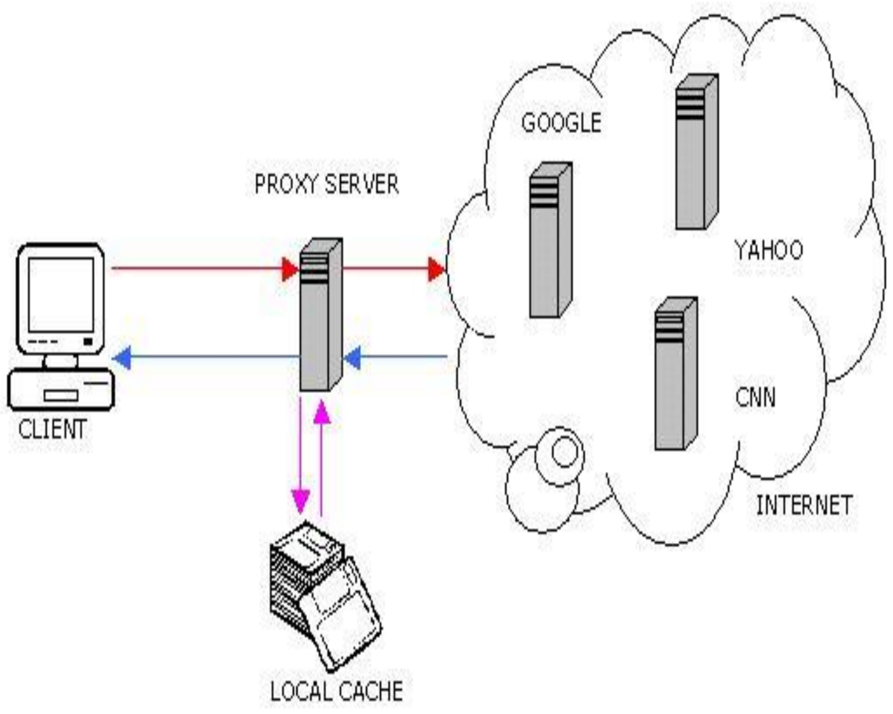
An advantage of a proxy server is that its cache can serve all users. If one or more Internet sites are frequently requested, these are likely to be in the proxy's cache, which will improve user response time. A proxy can also log its interactions, which can be helpful for troubleshooting.

Simple example for how proxy servers work

When a proxy server receives a request for an Internet resource (such as a Web page), it looks in its local cache of previously pages. If it finds the page, it returns it to the user without needing to forward the request to the Internet. If the page is not in the cache, the proxy server, acting as a client on behalf of the user, uses one of its own IP addresses to request the page from the server out on the Internet. When the page is returned, the proxy server relates it to the original request and forwards it on to the user.

Proxy servers are used for both legal and illegal purposes. In the enterprise, a proxy server is used to facilitate security, administrative control or caching services, among other purposes. In a personal computing context, proxy servers are used to enable user privacy and anonymous surfing. Proxy servers can also be used for the opposite purpose: To monitor traffic and undermine user privacy.

To the user, the proxy server is invisible; all Internet requests and returned responses appear to be directly with the addressed Internet server. (The proxy is not actually invisible; its IP address has to be specified as a configuration option to the browser or other protocol program.)



Java proxy server

Part one

Proxy functionality

1. The proxy listens for requests from clients
2. When there is a request, the proxy spawns a new thread for handling the request and creates and HttpRequest-object which contains the request.
3. The new thread sends the request to the server and reads the server's reply intoan HttpResponse-object.
4. The thread sends the response back to the requesting client.

Fill in code is **bolded** below.

HttpRequest.java

```
/**

 * HttpRequest - HTTP request container and parser

 *

 * $Id: HttpRequest.java,v 1.2 2003/11/26 18:11:53 kangasha Exp $

 *

 */


import java.io.*;

import java.net.*;

import java.util.*;


public class HttpRequest {

   /** Help variables */

   final static String CRLF = "\r\n";

   final static int HTTP_PORT = 80;

   /** Store the request parameters */

   String method;

   String URI;

   String version;

   String headers = "";
```

```
/** Server and port */
private String host;
private int port;

/** Create HttpRequest by reading it from the client socket */
public HttpRequest(BufferedReader from) {
        String firstLine = "";
        try {
           firstLine = from.readLine();
        } catch (IOException e) {
           System.out.println("Error reading request line: " + e);
        }

        String[] tmp = firstLine.split(" ");

        /* Format GET = GET http://www.google.pt HTTP/1.0 */
        method = tmp[0]; /* method GET */
        URI =  tmp[1]; /* URI */
        version =  tmp[2]; /* HTTP version */

        System.out.println("URI is: " + URI);



        try {
           String line = from.readLine();
           while (line.length() != 0) {
                   headers += line + CRLF;
                   /* We need to find host header to know which server to
                    * contact in case the request URI is not complete. */
                   if (line.startsWith("Host:")) {
```

```
                tmp = line.split(" ");
                if (tmp[1].indexOf(':') > 0) {
                    String[] tmp2 = tmp[1].split(":");
                    host = tmp2[0];
                    port = Integer.parseInt(tmp2[1]);
                } else {
                    host = tmp[1];
                    port = HTTP_PORT;
                }
              }
              line = from.readLine();
          }
        } catch (IOException e) {
          System.out.println("Error reading from socket: " + e);
          return;
        }
        System.out.println("Host to contact is: " + host + " at port " + port);
    }

    /** Return host for which this request is intended */
    public String getHost() {
        return host;
    }

    /** Return port for server */
    public int getPort() {
        return port;
    }

    /**
```

```
     * Convert request into a string for easy re-sending.
     */
    public String toString() {
            String req = "";

            req = method + " " + URI + " " + version + CRLF;
            req += headers;
            /* This proxy does not support persistent connections */
            req += "Connection: close" + CRLF;
            req += CRLF;

            return req;
    }
}
```

HttpResponse.java

```
/**
 * HttpResponse - Handle HTTP replies
 *
 * $Id: HttpResponse.java,v 1.2 2003/11/26 18:12:42 kangasha Exp $
 *
 */

import java.io.*;
import java.net.*;
import java.util.*;

public class HttpResponse {
```

```
final static String CRLF = "\r\n";
/** How big is the buffer used for reading the object */
final static int BUF_SIZE = 8192;
/** Maximum size of objects that this proxy can handle. For the
 * moment set to 100 KB. You can adjust this as needed. */
final static int MAX_OBJECT_SIZE = 100000;
/** Reply status and headers */
String version;
int status;
String statusLine = "";
String headers = "";
/* Body of reply */
byte[] body = new byte[MAX_OBJECT_SIZE];

/** Read response from server. */
public HttpResponse(DataInputStream fromServer) {
        /* Length of the object */
        int length = -1;
        boolean gotStatusLine = false;

        /* First read status line and response headers */
        try {
           String line =fromServer.readLine();
           while (line.length() != 0) {
                   if (!gotStatusLine) {
                      statusLine = line;
                      gotStatusLine = true;
                   } else {
                      headers += line + CRLF;
                   }
```

```
            /* Get length of content as indicated by
             * Content-Length header. Unfortunately this is not
             * present in every response. Some servers return the
             * header "Content-Length", others return
             * "Content-length". You need to check for both
             * here. */
            if (line.startsWith("Content-Length:") ||
               line.startsWith("Content-length:")) {
               String[] tmp = line.split(" ");
               length = Integer.parseInt(tmp[1]);
            }
            line = fromServer.readLine();
   }
} catch (IOException e) {
            System.out.println("Error reading headers from server: " + e);
}

try {
   int bytesRead = 0;
   byte buf[] = new byte[BUF_SIZE];
   boolean loop = false;

   /* If we didn't get Content-Length header, just loop until
    * the connection is closed. */
   if (length == -1) {
            loop = true;
   }

   /* Read the body in chunks of BUF_SIZE and copy the chunk
```

```
                 * into body. Usually replies come back in smaller chunks
                 * than BUF_SIZE. The while-loop ends when either we have
                 * read Content-Length bytes or when the connection is
                 * closed (when there is no Connection-Length in the
                 * response. */
                while (bytesRead < length || loop) {
                    /* Read it in as binary data */
                    int res = fromServer.read(buf, 0, BUF_SIZE); /* Read binary date until BUFF_SIZE to buf
(array in bytes) */
                    if (res == -1) {
                        break;
                    }
                    /* Copy the bytes into body. Make sure we don't exceed
                     * the maximum object size. */
                    for (int i = 0;
                         i < res && (i + bytesRead) < MAX_OBJECT_SIZE;
                         i++) {
                        body[bytesRead + i] = buf[i]; /* copy bytes read to buf to body */
                    }
                    bytesRead += res;
                }
            } catch (IOException e) {
                System.out.println("Error reading response body: " + e);
                return;
            }


    }


    /**
```

```
     * Convert response into a string for easy re-sending. Only
     * converts the response headers, body is not converted to a
     * string.
     */
    public String toString() {
            String res = "";

            res = statusLine + CRLF;
            res += headers;
            res += CRLF;

            return res;
    }
}
```

ProxyCache.java

```
/*
* ProxyCache.java - Simple caching proxy
*
* $Id: ProxyCache.java,v 1.3 2004/02/16 15:22:00 kangasha Exp $
*/



import java.net.*;
import java.io.*;
import java.util.*;
import java.lang.*;

public class ProxyCache {
    /** Port for the proxy */
    private static int port;
    /** Socket for client connections */
    private static ServerSocket socket;
    /** Create the ProxyCache object and the socket */
    private static Map<String, String> cache = new Hashtable<String, String>();



    public  static void caching(HttpRequest req, HttpResponse res) throws IOException{
            File cachedSites;
            DataOutputStream output;


            cachedSites = new File("cachedSites.txt");
            FileOutputStream fileOutputStream = new FileOutputStream(cachedSites);
                            ObjectOutputStream objectOutputStream = new
  ObjectOutputStream(fileOutputStream);

                            objectOutputStream.writeObject(cache);
                            objectOutputStream.close();
                            fileOutputStream.close();
            cache.put(req.URI, cachedSites.getAbsolutePath());
            System.out.println("Caching from: "+req.URI+" para "+cachedSites.getAbsolutePath());
    }


            public  static byte[] uncaching(String name) throws IOException{
                    File file1cached;
                    FileInputStream fileInput;
                    String hashfile;
                    byte[] bytescached;
```

```
        if((hashfile = cache.get(name))!=null){
                file1cached = new File(hashfile);
                fileInput = new FileInputStream(file1cached);
                bytescached = new byte[(int)file1cached.length()];
                fileInput.read(bytescached);
                System.out.println("Caching: Hit on "+name+" returning cache to user");
                return bytescached;
        }
        else {
                System.out.println("Caching: No hit on "+name);
                return bytescached = new byte[0];
        }

}



public static void init(int p) {
port = p;
        try {
        socket = new ServerSocket(port);
        } catch (IOException e) {
                System.out.println("Error creating socket: " + e);
                System.exit(-1);
                }
}
```

```
/** Read command line arguments and start proxy */
public static void main(String args[]) {
        int myPort = 0;
        File cachedir = new File("cache/");
        if (!cachedir.exists()){cachedir.mkdir();}

        try {
           myPort = Integer.parseInt(args[0]);
        } catch (ArrayIndexOutOfBoundsException e) {
           System.out.println("Need port number as argument");
           System.exit(-1);
        } catch (NumberFormatException e) {
           System.out.println("Please give port number as integer.");
           System.exit(-1);
        }

        init(myPort);

        /** Main loop. Listen for incoming connections and spawn a new
         * thread for handling them */
        Socket client = null;

        while (true) {
           try {
                   client = socket.accept(); /* Accepts new customers */
                   (new Thread(new Threads(client))).start(); /* Create threads for each new client */
           } catch (IOException e) {
                   System.out.println("Error reading request from client: " + e);
                   /* Definitely cannot continue processing this request,
                         * so skip to next iteration of while loop. */
                        continue;
                    }
                }

    }
}
```

Threads.java

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.lang.*;

public class Threads implements Runnable{

            private final Socket client;

    public Threads(Socket client) {
        this.client = client;
    }



            public void run() {
            Socket server = null;
            HttpRequest request = null;
            HttpResponse response = null;

            /* Process request. If there are any exceptions, then simply
```

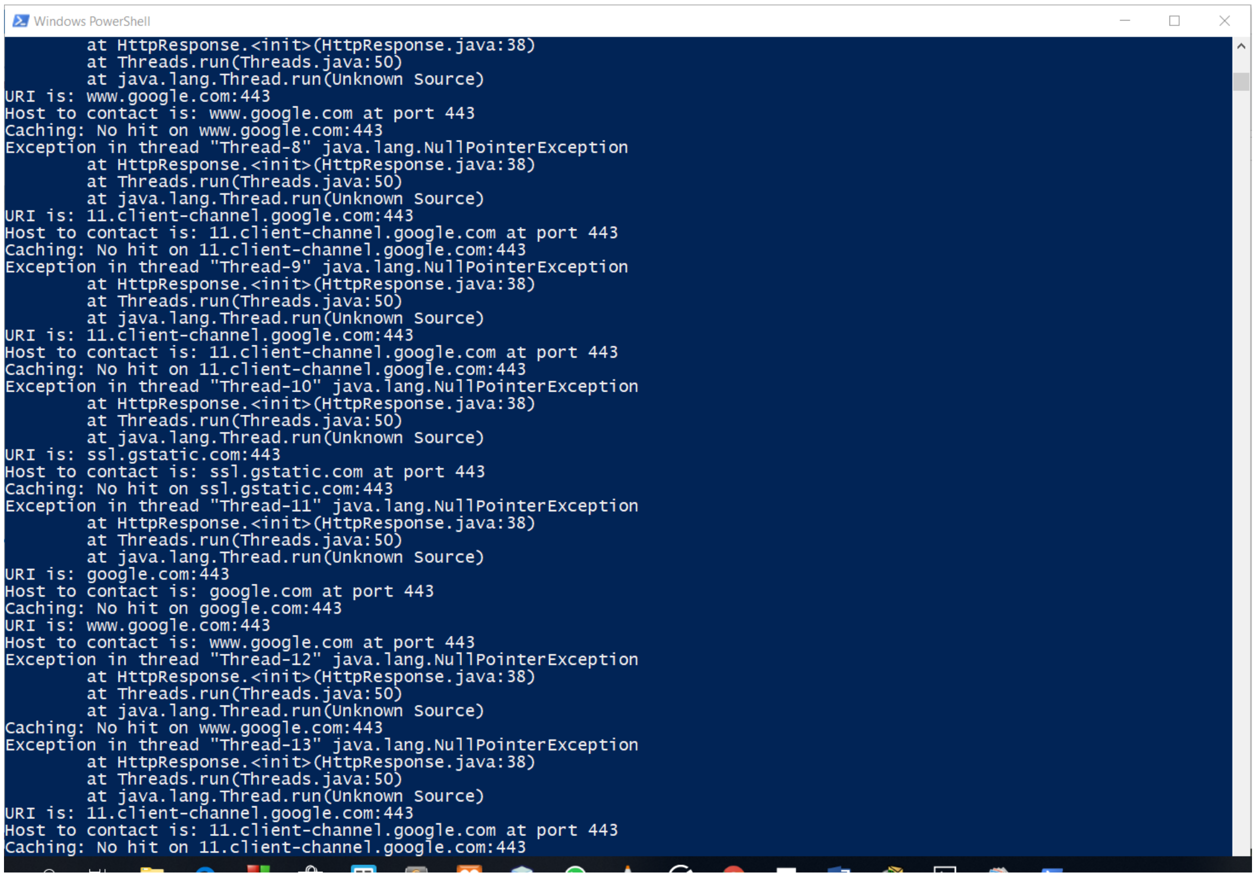
```
             * return and end this request. This unfortunately means the
             * client will hang for a while, until it timeouts. */

            /* Read request */
            try {
                BufferedReader fromClient = new BufferedReader(new
InputStreamReader(client.getInputStream()));
                request = new HttpRequest(fromClient);
            } catch (IOException e) {
                System.out.println("Error reading request from client: " + e);
                return;
            }
            /* Send request to server */
            try {
                /* Open socket and write request to socket */
                server = new Socket(request.getHost(), request.getPort()); /* Create socket */
                DataOutputStream toServer = new DataOutputStream(server.getOutputStream()); /* Create
outputstream for server on socket */
        toServer.writeBytes(request.toString()); /* Create outputstream for the serviceScribe request to the
outputstreamidor on the socket */
            } catch (UnknownHostException e) {
                System.out.println("Unknown host: " + request.getHost());
                System.out.println(e);
                return;
            } catch (IOException e) {
                return;
            }
            /* Read response and forward it to client */
            try {
                    byte[] cache = ProxyCache.uncaching(request.URI);
                    if(cache.length==0) {
                        DataInputStream fromServer = new DataInputStream(server.getInputStream()); /*
Create server inputstream */
                        response = new HttpResponse(fromServer); /* Create object with server response */
                        DataOutputStream toClient = new DataOutputStream(client.getOutputStream());




                        toClient.writeBytes(response.toString()); /* Write headers */
                        toClient.write(response.body); /* Write body */
                        /* Write response to client. First headers, then body */

                        ProxyCache.caching(request, response); /* Save to cache */

                        client.close();
                        server.close();
                        /* Insert object into the cache */
                        /* Fill in (optional exercise only) */
                    }
                    else{
                            DataOutputStream toClient = new
DataOutputStream(client.getOutputStream());
                            toClient.write(cache);
                            client.close();
                            server.close();
                    }

            } catch (IOException e) {
               System.out.println("Error writing response to client: " + e);
            }
    }
}
```

Part two

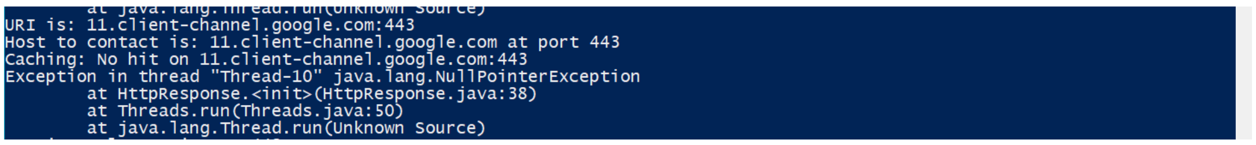
Add better error handling

Task 2: Add better error handling
The simple proxy which you have completed in task 1 does no error handling. This can be a problem especially when the client requests an object which is not available, since the "404 Not found" response usually has no response body and the proxy assumes there is a body and tries to read it. You need to modify code to so that it can HTTP responses with error codes.

As you can see in the ProxyCache.java above, the error handling is done by using exceptions. Additional class apart from the given classes name Threads.java has also fulfilled with error handling.



```
        at HttpResponse.<init>(HttpResponse.java:38)
        at Threads.run(Threads.java:50)
        at java.lang.Thread.run(Unknown Source)
URI is: www.google.com:443
Host to contact is: www.google.com at port 443
Caching: No hit on www.google.com:443
Exception in thread "Thread-8" java.lang.NullPointerException
        at HttpResponse.<init>(HttpResponse.java:38)
        at Threads.run(Threads.java:50)
        at java.lang.Thread.run(Unknown Source)
URI is: 11.client-channel.google.com:443
Host to contact is: 11.client-channel.google.com at port 443
Caching: No hit on 11.client-channel.google.com:443
Exception in thread "Thread-9" java.lang.NullPointerException
        at HttpResponse.<init>(HttpResponse.java:38)
        at Threads.run(Threads.java:50)
        at java.lang.Thread.run(Unknown Source)
URI is: 11.client-channel.google.com:443
Host to contact is: 11.client-channel.google.com at port 443
Caching: No hit on 11.client-channel.google.com:443
Exception in thread "Thread-10" java.lang.NullPointerException
        at HttpResponse.<init>(HttpResponse.java:38)
        at Threads.run(Threads.java:50)
        at java.lang.Thread.run(Unknown Source)
URI is: ssl.gstatic.com:443
Host to contact is: ssl.gstatic.com at port 443
Caching: No hit on ssl.gstatic.com:443
Exception in thread "Thread-11" java.lang.NullPointerException
        at HttpResponse.<init>(HttpResponse.java:38)
        at Threads.run(Threads.java:50)
        at java.lang.Thread.run(Unknown Source)
URI is: google.com:443
Host to contact is: google.com at port 443
Caching: No hit on google.com:443
URI is: www.google.com:443
Host to contact is: www.google.com at port 443
Exception in thread "Thread-12" java.lang.NullPointerException
        at HttpResponse.<init>(HttpResponse.java:38)
        at Threads.run(Threads.java:50)
        at java.lang.Thread.run(Unknown Source)
Caching: No hit on www.google.com:443
Exception in thread "Thread-13" java.lang.NullPointerException
        at HttpResponse.<init>(HttpResponse.java:38)
        at Threads.run(Threads.java:50)
        at java.lang.Thread.run(Unknown Source)
URI is: 11.client-channel.google.com:443
Host to contact is: 11.client-channel.google.com at port 443
Caching: No hit on 11.client-channel.google.com:443
```



```
URI is: 11.client-channel.google.com:443
Host to contact is: 11.client-channel.google.com at port 443
Caching: No hit on 11.client-channel.google.com:443
Exception in thread "Thread-10" java.lang.NullPointerException
        at HttpResponse.<init>(HttpResponse.java:38)
        at Threads.run(Threads.java:50)
        at java.lang.Thread.run(Unknown Source)
```

ProxyCache.java

```
catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Need port number as argument");
        System.exit(-1);
    } catch (NumberFormatException e) {
        System.out.println("Please give port number as integer.");
        System.exit(-1);
    }
```

Threads.java

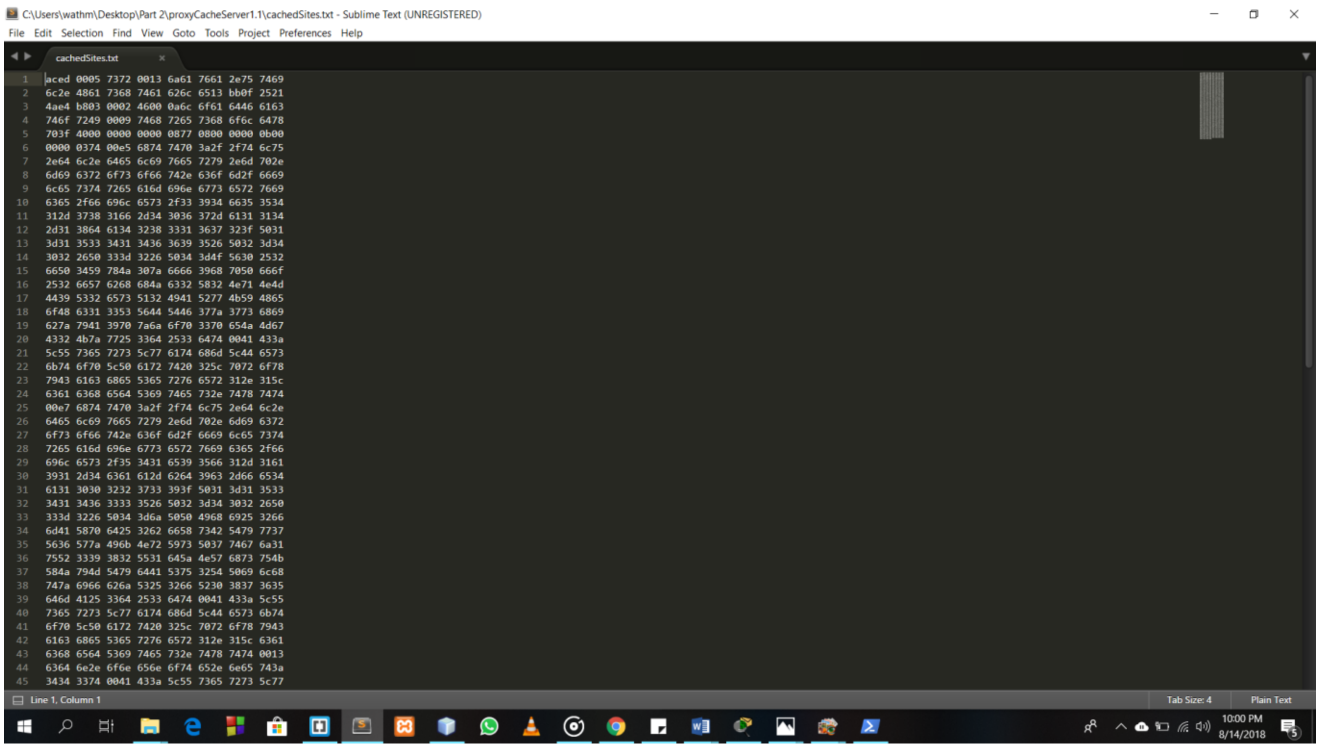
```
} catch (UnknownHostException e) {
        System.out.println("Unknown host: " + request.getHost());
        System.out.println(e);
        return;
    } catch (IOException e) {
        return;
    }
```

Part 3

Add caching

Task 3: Add caching
Add the simple caching functionality to the proxy you completed in task 2 above. You do not need to implement any replacement or validation policies. Your implementation will need to be able to write responses to the disk (i.e., the cache) and fetch them from disk when you get a cache hit. For this you need to implement some internal data structure in the proxy to keep track of which objects are cached and where they are on disk. You can keep this data structure in main memory; there is no need to make it persist across shutdowns.

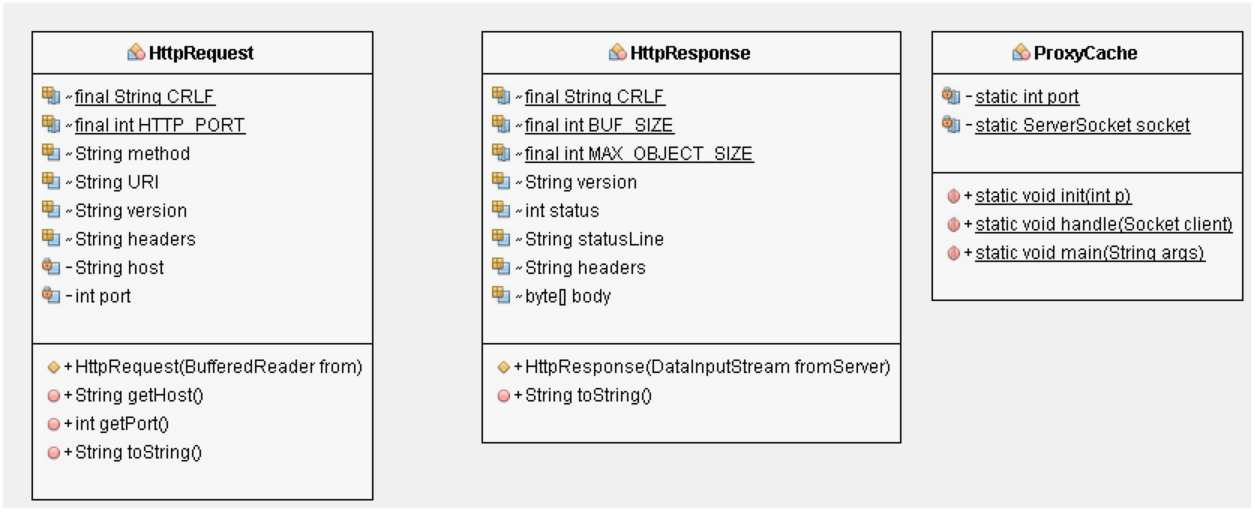
Here is the screenshot for cached packets in the text file.


C:\Users\wathm\Desktop\Part 2\proxyCacheServer1.1\cachedSites.txt - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
cachedSites.txt
Line 1, Column 1
Tab Size: 4
Plain Text
10:00 PM
8/14/2018

```
aced 0005 7372 0013 6a61 7661 2e75 7469
6c2e 4861 7368 7461 626c 6513 bb0f 2521
4ae4 b803 0002 4600 0a6c 6f61 6446 6163
746f 7249 0009 7468 7265 7368 6f6c 6478
703f 4000 0000 0000 0877 0800 0000 0b00
0000 0374 00e5 6874 7470 3a2f 2f74 6c75
2e64 6c2e 6465 6c69 7665 7279 2e6d 702e
6d69 6372 6f73 6f66 742e 636f 6d2f 6669
6c65 7374 7265 616d 696e 6773 6572 7669
6365 2f66 696c 6573 2f33 3934 6635 3534
312d 3738 3166 2d34 3036 372d 6131 3134
2d31 3864 6134 3238 3331 3637 323f 5031
3d31 3533 3431 3436 3639 3526 5032 3d34
3032 2650 333d 3226 5034 3d4f 5630 2532
6650 3459 784a 307a 6666 3968 7050 666f
2532 6657 6268 684a 6332 5832 4e71 4e4d
4439 5332 6573 5132 4941 5277 4b59 4865
6f48 6331 3353 5644 5446 377a 3773 6869
627a 7941 3970 7a6a 6f70 3370 654a 4d67
4332 4b7a 7725 3364 2533 6474 0041 433a
5c55 7365 7273 5c77 6174 686d 5c44 6573
6b74 6f70 5c50 6172 7420 325c 7072 6f78
7943 6163 6865 5365 7276 6572 312e 315c
6361 6368 6564 5369 7465 732e 7478 7474
00e7 6874 7470 3a2f 2f74 6c75 2e64 6c2e
6465 6c69 7665 7279 2e6d 702e 6d69 6372
6f73 6f66 742e 636f 6d2f 6669 6c65 7374
7265 616d 696e 6773 6572 7669 6365 2f66
696c 6573 2f35 3431 6539 3566 312d 3161
3931 2d34 6361 612d 6264 3963 2d66 6534
6131 3030 3232 3733 393f 5031 3d31 3533
3431 3436 3333 3526 5032 3d34 3032 2650
333d 3226 5034 3d6a 5050 4968 6925 3266
6d41 5870 6425 3262 6658 7342 5479 7737
5636 577a 496b 4e72 5973 5037 7467 6a31
7552 3339 3832 5531 645a 4e57 6873 754b
584a 794d 5479 6441 5375 3254 5069 6c68
747a 6966 626a 5325 3266 5230 3837 3635
646d 4125 3364 2533 6474 0041 433a 5c55
7365 7273 5c77 6174 686d 5c44 6573 6b74
6f70 5c50 6172 7420 325c 7072 6f78 7943
6163 6865 5365 7276 6572 312e 315c 6361
6368 6564 5369 7465 732e 7478 7474 0013
6364 6e2e 6f6e 656e 6f74 652e 6e65 743a
3434 3374 0041 433a 5c55 7365 7273 5c77
```

```
public  static void caching(HttpRequest req, HttpResponse res) throws IOException{
        File cachedSites;
        DataOutputStream output;



        cachedSites = new File("cachedSites.txt");
        FileOutputStream fileOutputStream = new FileOutputStream(cachedSites);
                    ObjectOutputStream objectOutputStream = new
ObjectOutputStream(fileOutputStream);

                    objectOutputStream.writeObject(cache);
                    objectOutputStream.close();
                    fileOutputStream.close();
        cache.put(req.URI, cachedSites.getAbsolutePath());
        System.out.println("Caching from: "+req.URI+" para "+cachedSites.getAbsolutePath());
  }
```

Class diagram



References

1. https://github.com/Insatisfeito/ProxyServer/blob/master/ProxyCache.java
2. https://stackoverflow.com/questions/29091648/cache-webpage-using-java-proxy-server
3. www.youtube.com
4. http://study-ccna.com/

Group members

1. Pamodi Rathnayke-BSC-UCD-CSC-16.2-016/10009210
2. Achini Rathnayke-BSC-UGC-MIS-16.2-042/10009227
3. Helitha Thalagalaarachchi-BSC-UCD-CSC-16.2-025/10009190
4. Kavindu Fonseka-BSC-UCD-CSC-004/10009215
5. Hashan Jayalath- BSC-UCD-CSC-16.2-009/10009207
6. Buddi Perera-BSC-PLY-COM-16.2-199/10009004