IT352-Lab-Program-1

Create network topology given the subsequent slides using open source tool called packet tracer, assign IP address as mentioned in the given topology, configure dynamic routing protocol RIPv1 on all routers shown in the topology. Verify the connectivity that connectivity that communication takes place between any system to any system in the created topology using “ping” command. Configure ACL as given in the network topology.

IT352-Lab-Program-1 Submission

File name of the program : RegisterNo_IT352_P1 (P1 indicates Lab Program Number-1)

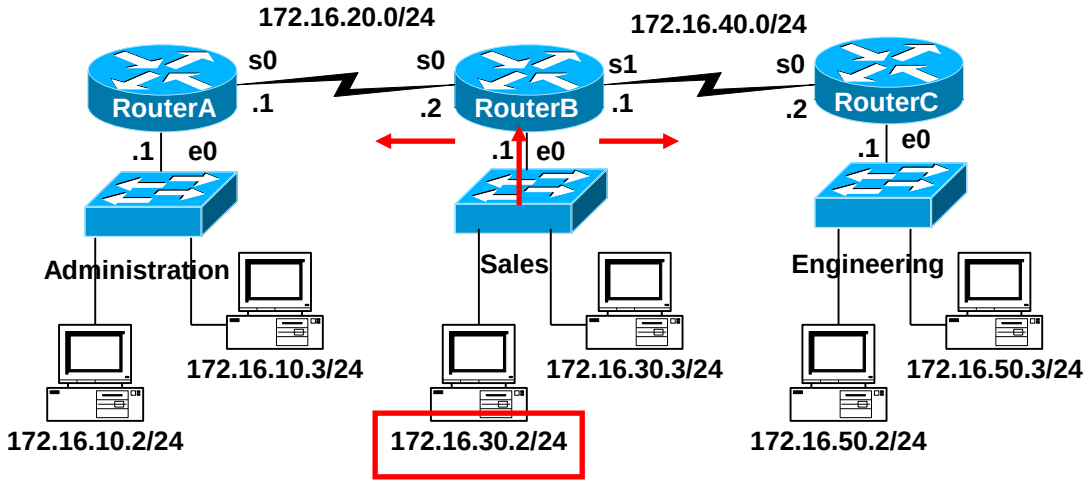
Deadline of Lab Program : 19th January 2022 Wednesday

Deadline of Submission : 20th January 2022 Thursday on or before 6:00PM.

Submit created and configured network topology using packet-tracer file to the Moodle under the web-link title “IT352-Lab-Program-1-Submisison Web Link”.

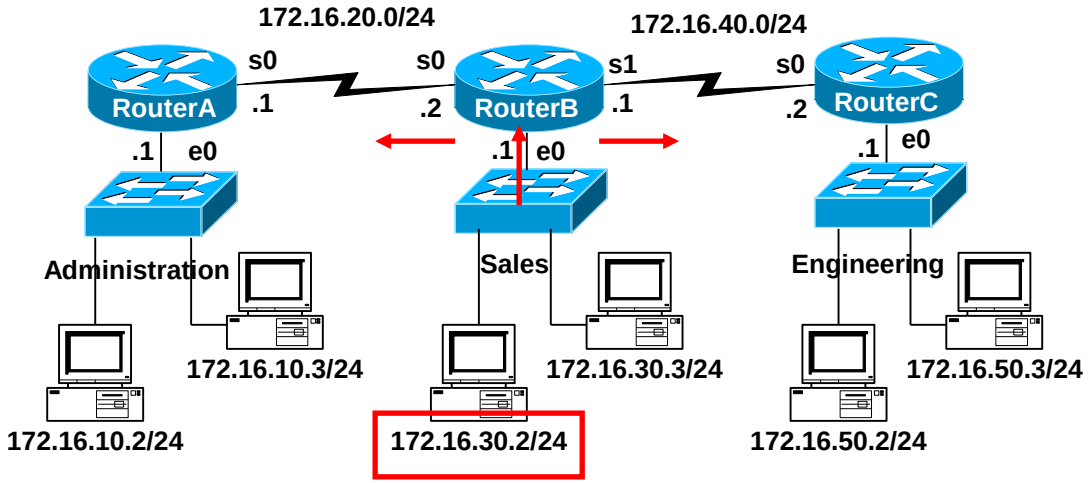
Note : No/Zero marks for incomplete submission/late submission/incomplete program.

For Reg. No 181560181IT245, 191300191IT101- 191154191IT134



- Task:
  - Permit only the host 172.16.30.2 from exiting the Sales network.
  - Deny all other hosts on the Sales network from leaving the 172.16.30.0/24 network.

For Reg. No 181560181IT245, 191300191IT101- 191154191IT134



Step 1 – ACL statements Implicit deny any, which is automatically added.

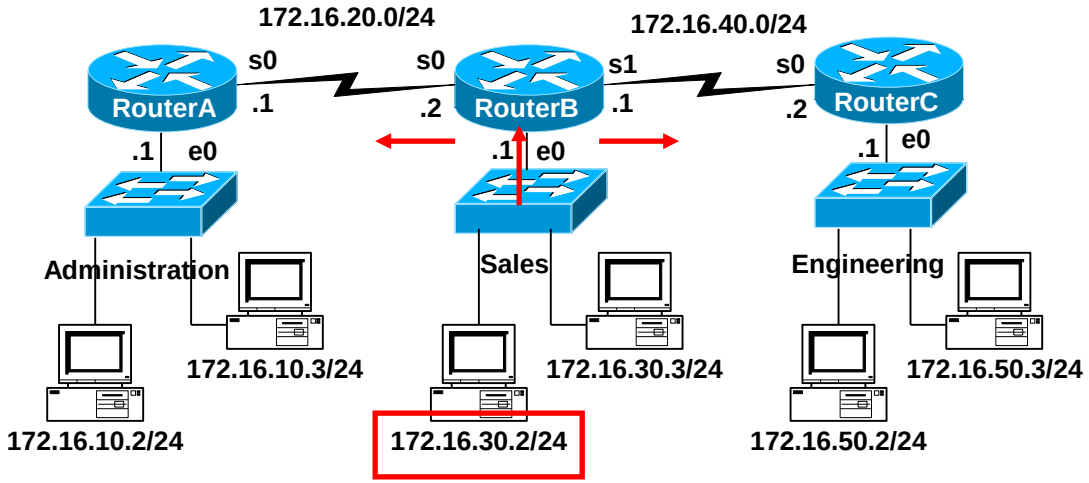
Test Condition

```
RouterB(config)#access-list 10 permit 172.16.30.2
Implicit "deny any" -do not need to add this, discussed later
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255
```

```
Router(config)#access-list access-list-number
  {permit | deny} {test-conditions}
```

Protocol	Range
IP (Standard IP)	1-99

For Reg. No 181560181IT245, 191300191IT101- 191154191IT134



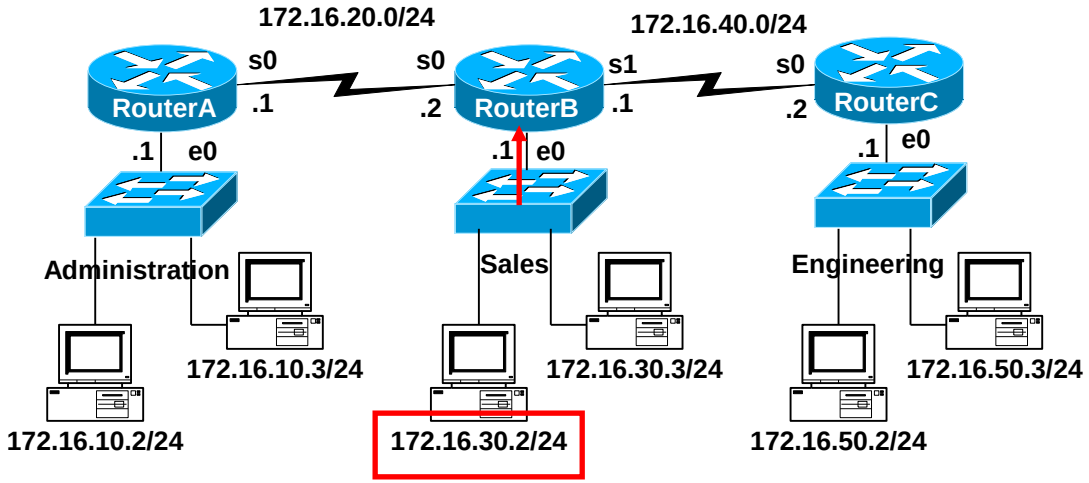
Applying ACLs

- You can define ACLs without applying them.
- However, **the ACLs will have no effect until they are applied to the router's interface.**
- It is a good practice to apply the Standard ACLs on the interface closest to the destination of the traffic and Extended ACLs on the interface closest to the source.

Defining In, Out, Source, and Destination

- **Out** - Traffic that has already been routed by the router and is leaving the interface
- **In** - Traffic that is arriving on the interface and which will be routed router.

For Reg. No 181560181IT245, 191300191IT101- 191154191IT134

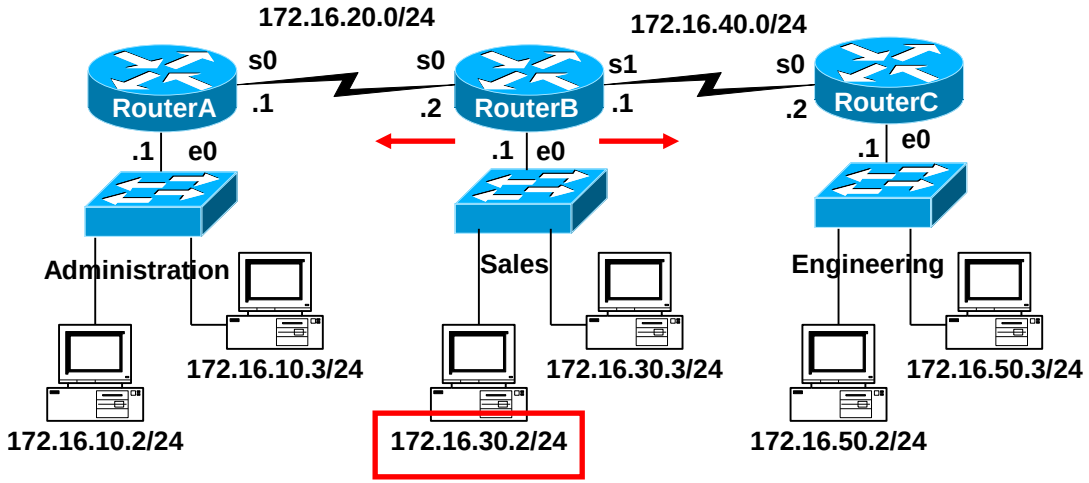


Step 2 – Apply to an interface(s)

```
RouterB(config)#access-list 10 permit 172.16.30.2
Implicit "deny any" -do not need to add this, discussed later
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255


RouterB(config)# interface e 0
```

```
Router(config-if)#{protocol} access-group access-list-
number
```

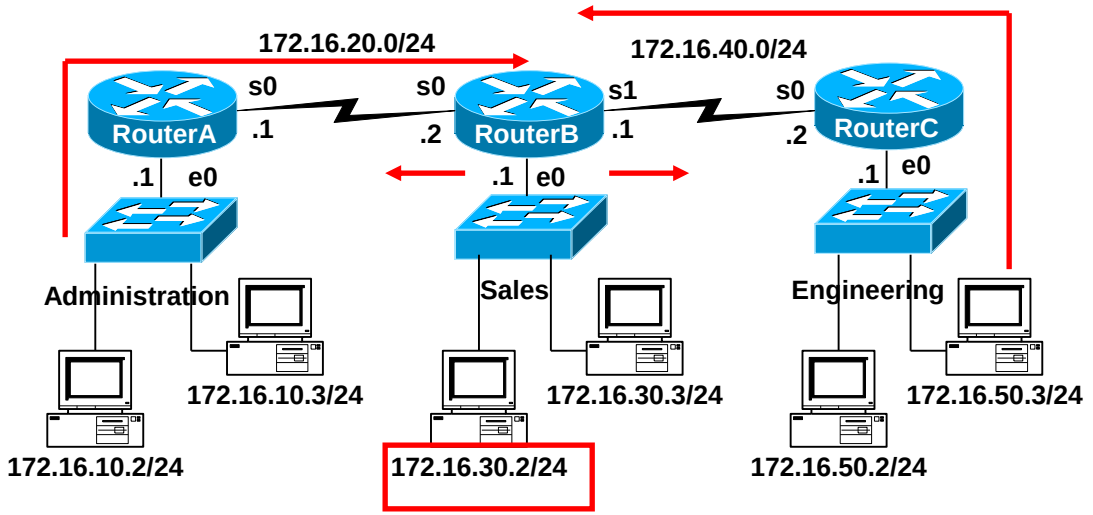
For Reg. No 181560181IT245, 191300191IT101- 191154191IT134



Step 2 – Or the outgoing interfaces… Which is preferable and why?

```
RouterB(config)#access-list 10 permit 172.16.30.2
Implicit "deny any" -do not need to add this, discussed later
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255

RouterB(config)# interface s 0
RouterB(config-if)# ip access-group 10 out
RouterB(config)# interface s 1
RouterB(config-if)# ip access-group 10 out
```

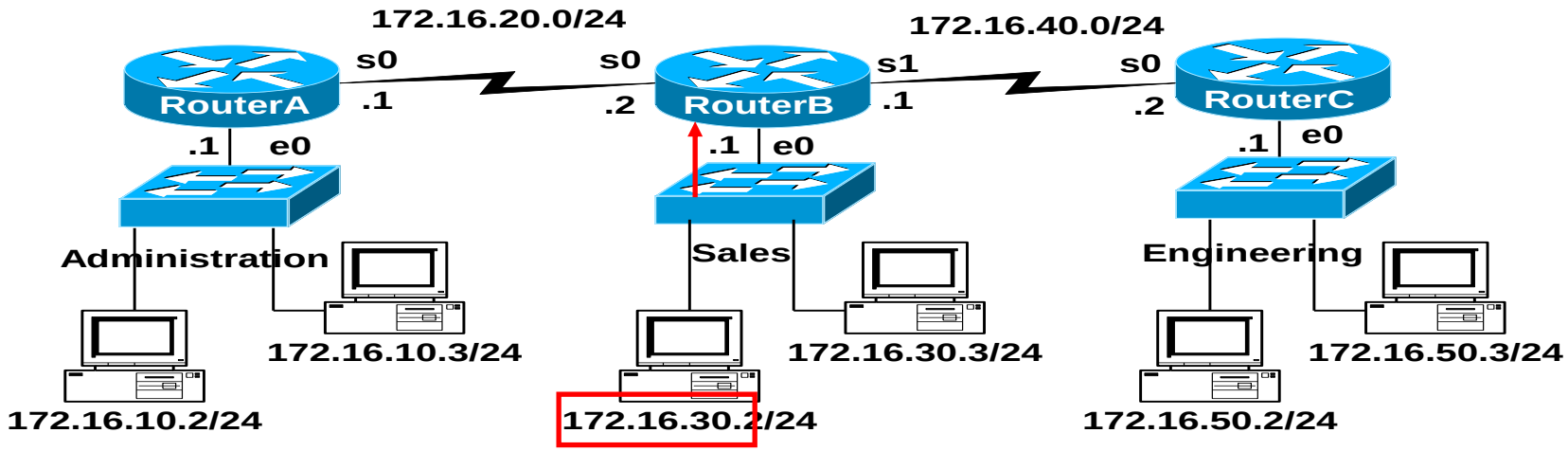
For Reg. No 181560181IT245, 191300191IT101- 191154191IT134



Because of the implicit deny any, this has an adverse affect of also denying packets from Administration from reaching Engineering, and denying packets from Engineering from reaching Administration.

```
RouterB(config)#access-list 10 permit 172.16.30.2
Implicit "deny any" -do not need to add this, discussed later
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255

RouterB(config)# interface s 0
RouterB(config-if)# ip access-group 10 out
RouterB(config)# interface s 1
RouterB(config-if)# ip access-group 10 out
```

For Reg. No 181560181IT245, 191300191IT101- 191154191IT134



Preferred, this access list will work to all existing and new interfaces on RouterB.

RouterB(config)#**access-list 10 permit 172.16.30.2**

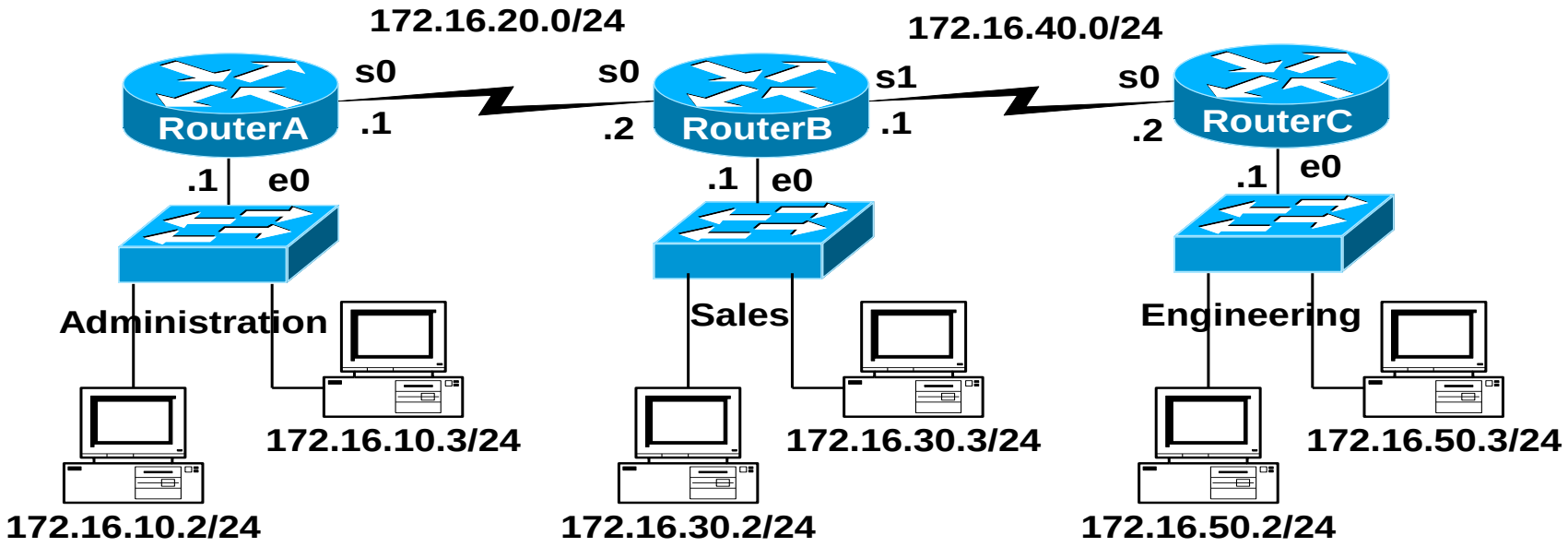
*Implicit "**deny any**"* -do not need to add this, discussed later

RouterB(config)#**access-list 10 deny 0.0.0.0 255.255.255.255**

RouterB(config)# **interface e 0**

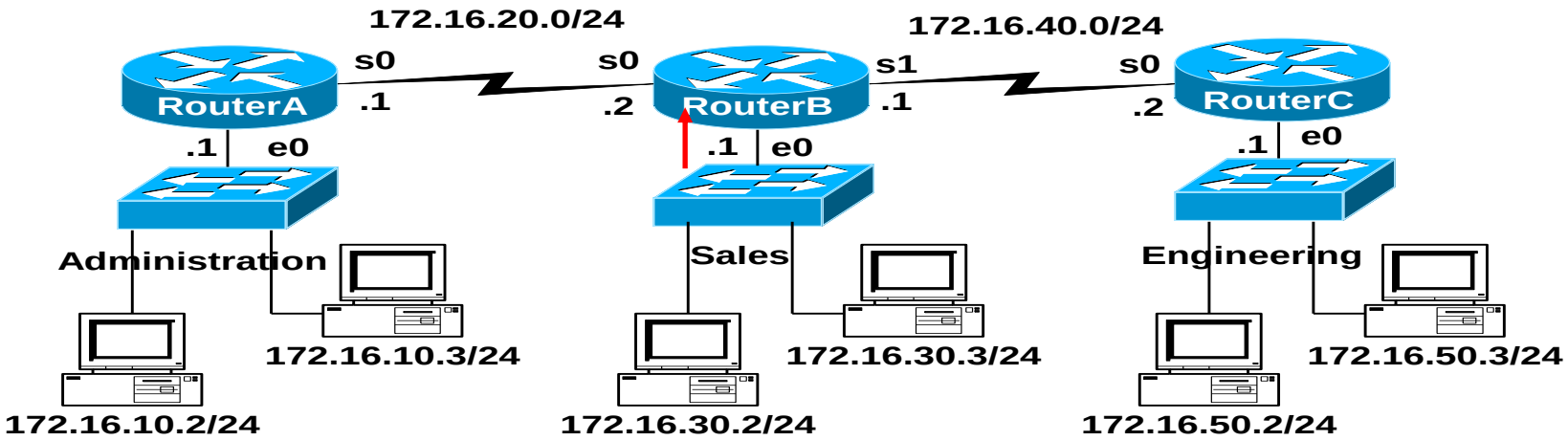
RouterB(config-if)# **ip access-group 10 in**

For Reg. No 191063191IT135 - 191023191IT212



- Task:
  - Permit only the hosts 172.16.30.2, 172.16.30.3, 172.16.30.4, 172.16.30.5 from exiting the Sales network.
  - Deny all other hosts on the Sales network from leaving the 172.16.30.0/24 network.

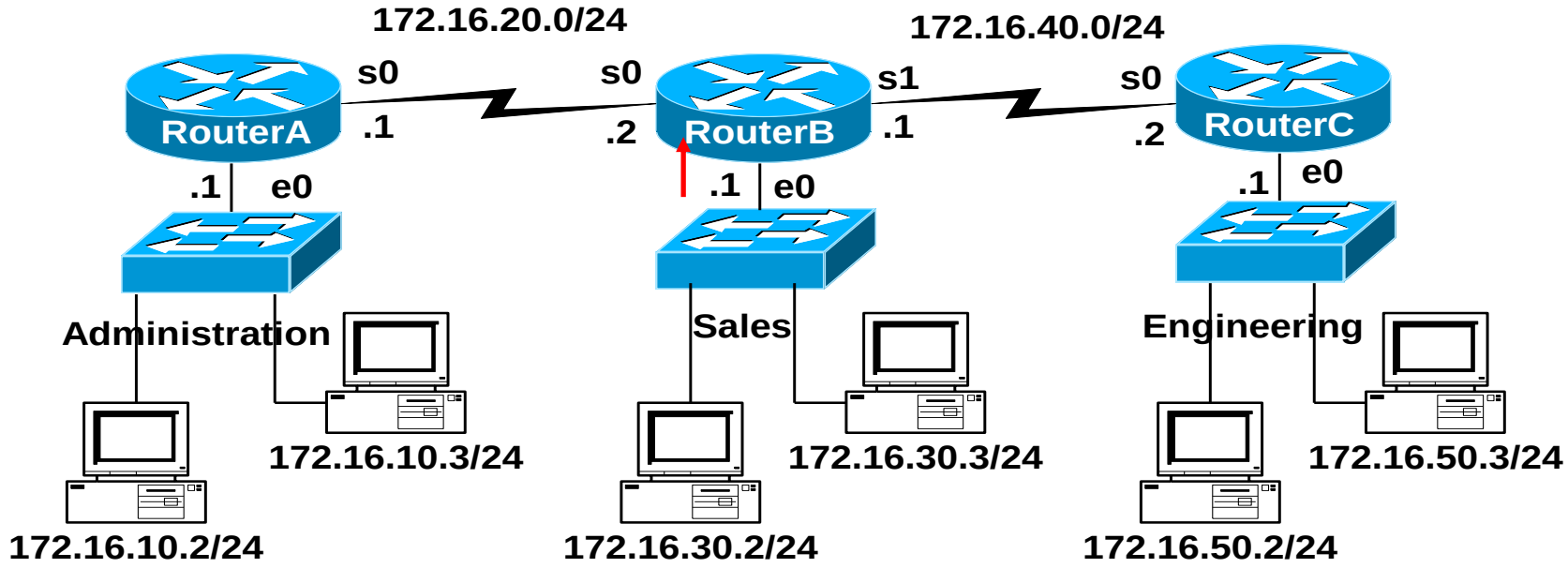
For Reg. No 191063191IT135 - 191023191IT212



Once a condition is met, all other statements are ignored, so the implicit *deny any* only applies to not-matched packets.

```
RouterB(config)#access-list 10 permit 172.16.30.2
RouterB(config)#access-list 10 permit 172.16.30.3
RouterB(config)#access-list 10 permit 172.16.30.4
RouterB(config)#access-list 10 permit 172.16.30.5
Implicit "deny any" -do not need to add this, discussed later
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255

RouterB(config)# interface e 0
RouterB(config-if)# ip access-group 10 in
```

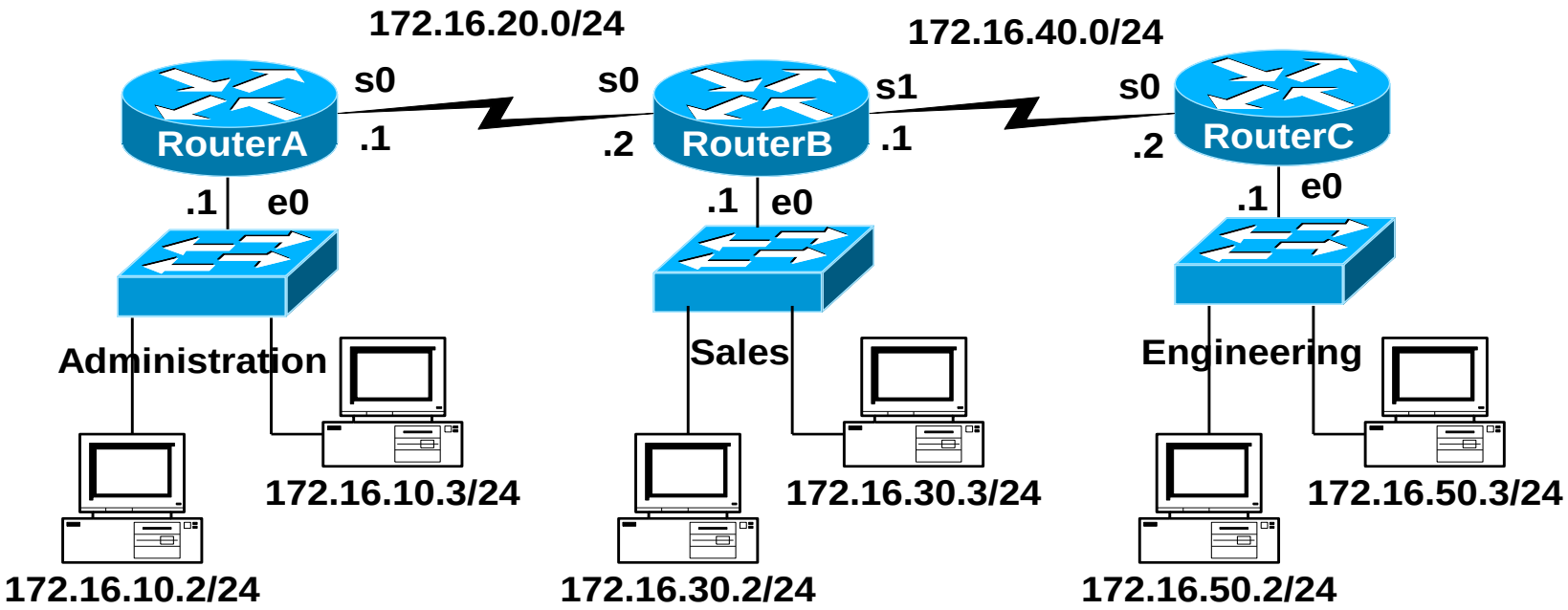
For Reg. No 191063191IT135 - 191023191IT212



To remove an Access List, use the no access-list command. Removing the access-group only from the interface leaves the access list, but they are not currently being applied. Usually, best to remove it from both.

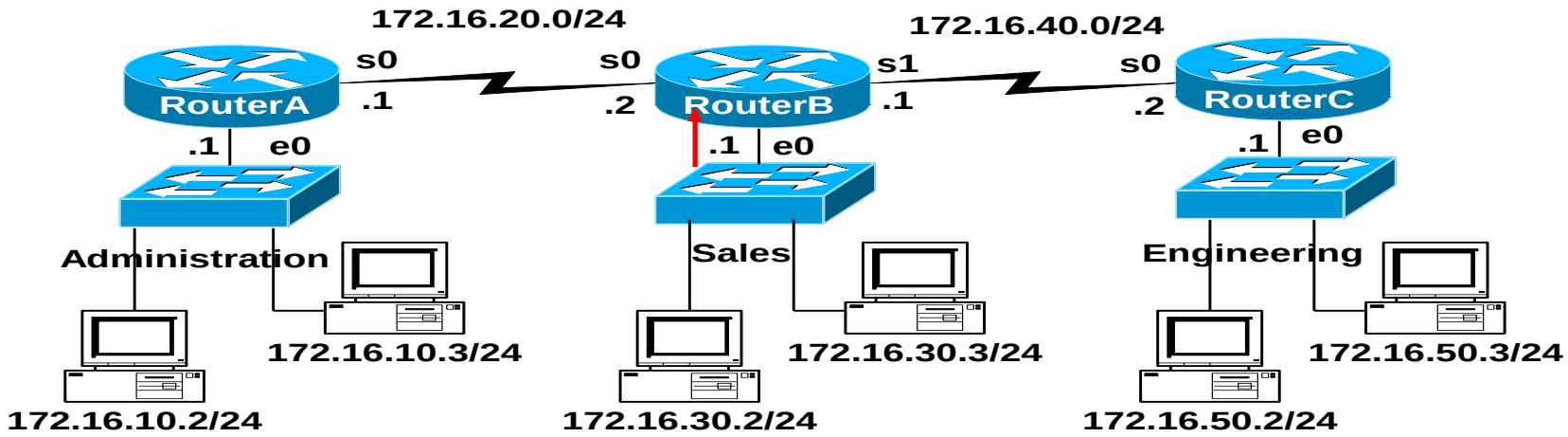
```
RouterB(config)#no access-list 10


RouterB(config)# interface e 0
RouterB(config-if)# no ip access-group 10 in
```

For Reg. No 191063191IT135 - 191023191IT212



- Task:
  - Deny only the host 172.16.30.2 from exiting the Sales network.
  - Permit all other hosts on the Sales network to leave the 172.16.30.0/24 network.
- **Keyword "any" can be used to represent all IP Addresses**.

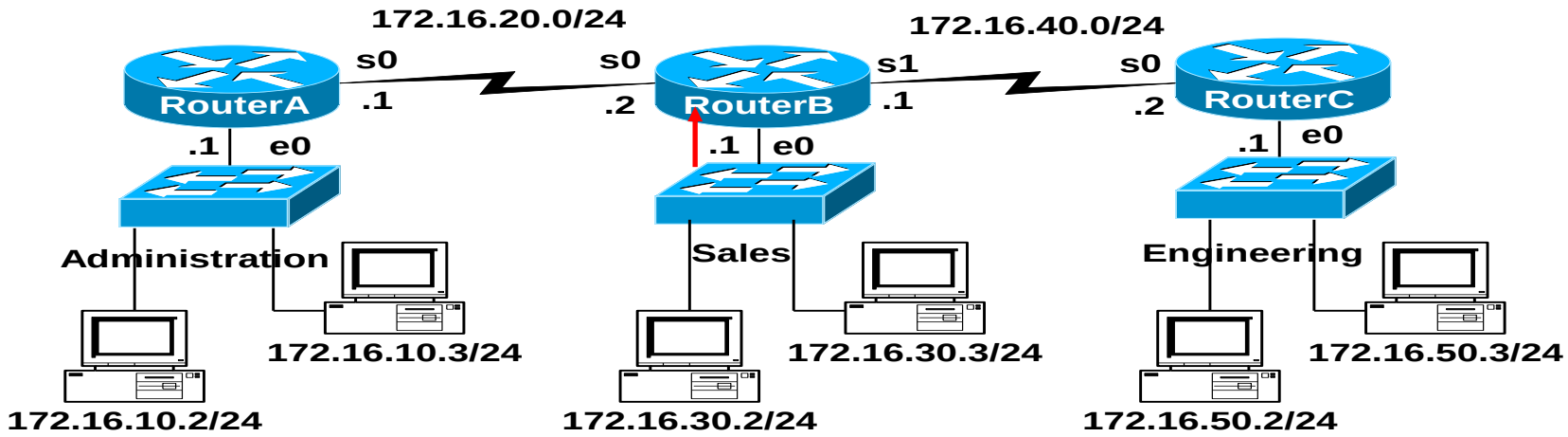
For Reg. No 191063191IT135 - 191023191IT212



Order matters! What if these two statements were reversed? Does the *implicit deny any* ever get a match? No, the permit any will cover all other packets.

```
RouterB(config)#access-list 10 deny 172.16.30.2
RouterB(config)#access-list 10 permit any
Implicit "deny any" -do not need to add this, discussed later
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255

RouterB(config)# interface e 0
RouterB(config-if)# ip access-group 10 in
```

For Reg. No 191063191IT135 - 191023191IT212



Order matters! In this case all packets would be permitted, because all packets would match the first access list statement. Once a condition is met, all other statements are ignored. The second access list statement and the implicit deny any would never be used. This would not do what we want.

RouterB(config)#**access-list 10 permit any**

RouterB(config)#**access-list 10 deny 172.16.30.2**

Implicit "deny any" -do not need to add this, discussed later

RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255

RouterB(config)# **interface e 0**

RouterB(config-if)# **ip access-group 10 in**

Note on Inbound Access Lists

- When an access lists applied to an **inbound** interface, the packets are checked against the access list before any routing table lookup process occurs.
- We will see how **outbound access list** work in a moment, but they are applied after the forwarding decision is made, after the routing table lookup process takes place and an exit interface is determined.
- **Once a packet is denied by an ACL, the router sends an ICMP "Destination Unreachable" message, with the code value set to "Administratively Prohibited"** to the source of the packet.

```
RouterB(config)#access-list 10 deny 172.16.30.2
RouterB(config)#access-list 10 permit any
Implicit "deny any" (do not need to add this, discussed later):
RouterB(config)#access-list 10 deny 0.0.0.0 255.255.255.255
RouterB(config)# interface e 0
RouterB(config-if)# ip access-group 10 in
```

Notes from www.cisco.com

- Traffic coming into the router is compared to ACL entries based on the order that the entries occur in the router.
- New statements are added to the end of the list.
- The router keeps looking until it has a match.
- If no matches are found when the router reaches the end of the list, the traffic is denied.
- For this reason, you should have the frequently hit entries at the top of the list.
- There is an "implied deny" for traffic that is not permitted.
- A single-entry ACL with only one "deny" entry has the effect of denying all traffic.
- You must have at least one "permit" statement in an ACL or all traffic will be blocked.

access-list 10 permit 10.1.1.1 0.0.0.255

access-list 10 deny ip any (implicit)

Wildcard Masks

```
Access-list 1 permit 172.16.0.0  0.0.255.255
```

A **wildcard mask** address:

- Tells how much of the packet's source IP address (or destination IP address) needs to match for this condition to be true.

Wildcard Masks Contd.

Access-list 1 permit 172.16.0.0 0.0.255.255

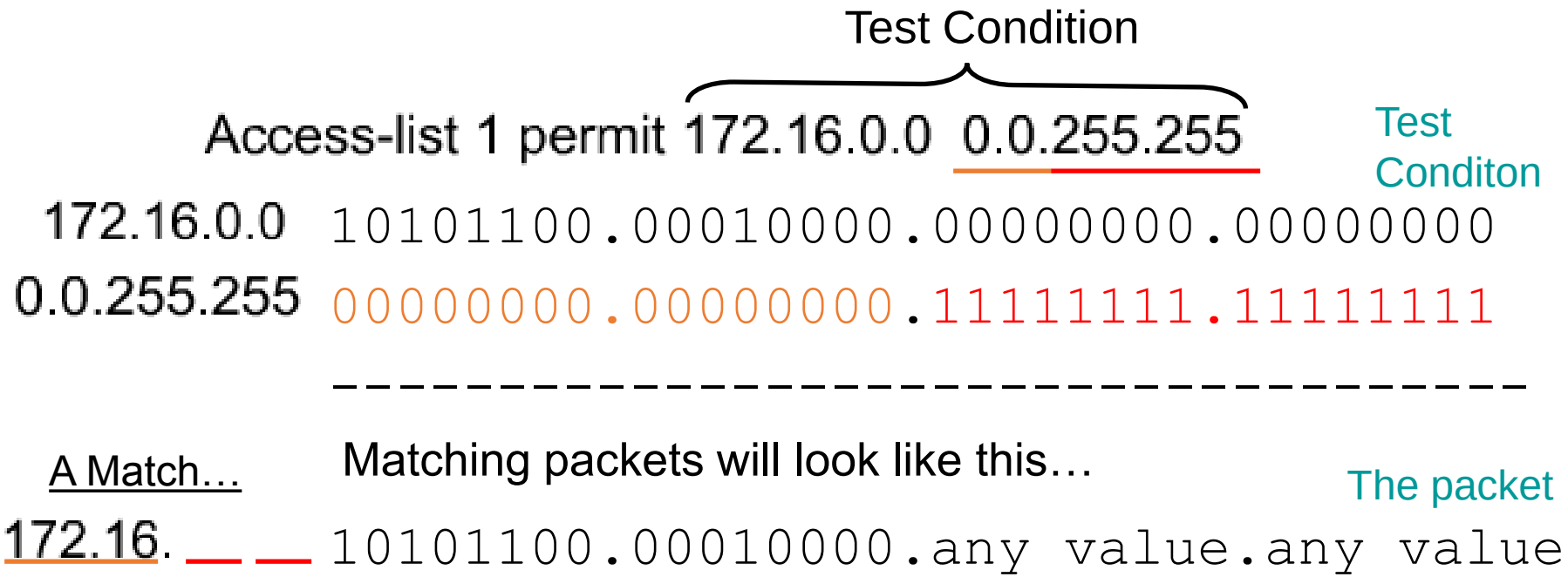
- A **wildcard mask** is a 32-bit quantity that is divided into four octets.
- A wildcard mask is paired with an IP address.
- The numbers one and zero in the mask are used to identify how to treat the corresponding IP address bits.
- The term wildcard masking is a nickname for the ACL mask-bit matching process and comes from of an analogy of a wildcard that matches any other card in the game of poker.
- Wildcard masks have no functional relationship with subnet masks.
  - They are used for different purposes and follow different rules.
- Subnet masks start from the left side of an IP address and work towards the right to extend the network field by borrowing bits from the host field.
- Wildcard masks are designed to filter individual or groups of IP addresses permitting or denying access to resources based on the address.

Wildcard Masks Contd.

Access-list 1 permit 172.16.0.0 0.0.255.255

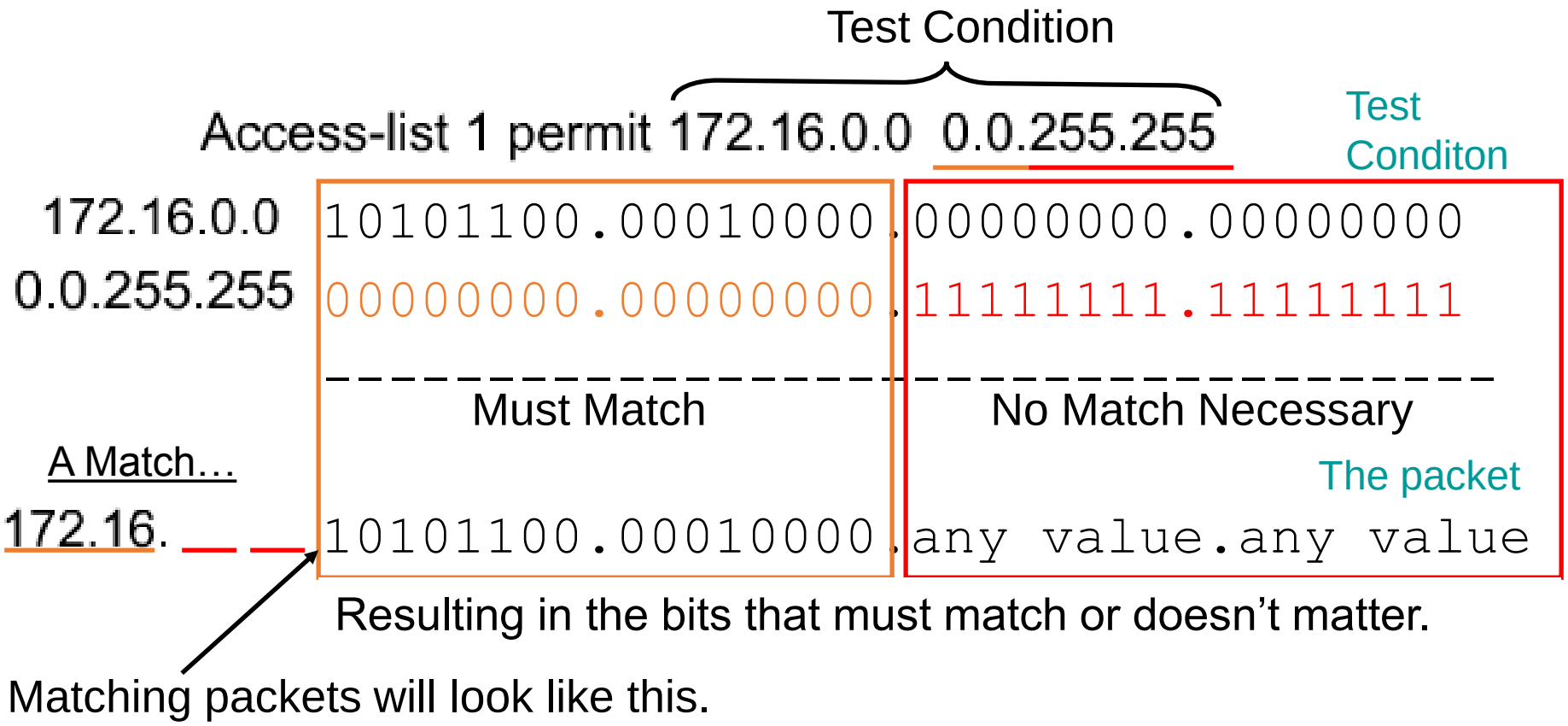
- “Trying to figure out how wildcard masks work by relating them to subnet masking will only confuse the entire matter. The only similarity between a wildcard mask and a subnet mask is that they are both thirty-two bits long and use ones and zeros for the mask.”
- This is not entirely true.
- Although it is very important that you understand how a wildcard mask works, it can also be thought as an **inverse subnet mask**.

Wildcard Masks



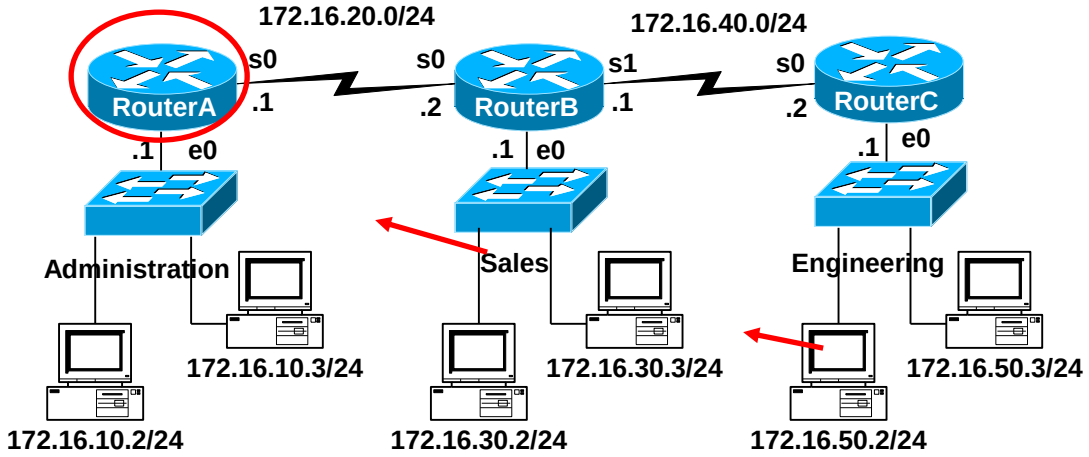
- Wildcard masking used to identify how to treat the corresponding IP address bits.
  - **0** - "**check** the corresponding bit value."
  - **1** - "**do not check** (ignore) that corresponding bit value."
- A **zero** in a bit position of the access list mask indicates that the corresponding bit in the address must be checked and must match for condition to be true.
- A **one** in a bit position of the access list mask indicates the corresponding bit in the address is not "interesting", does not need to match, and can be ignored.

Wildcard Masks Contd.



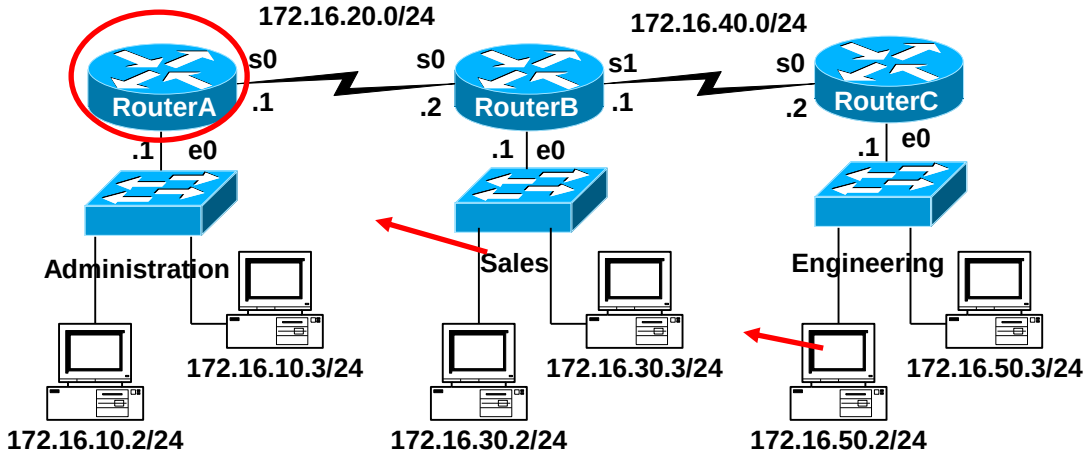
- **0** - "**check** the corresponding bit value."
- **1** - "**do not check** (ignore) that corresponding bit value."

For Reg. No 191552191IT215 - 191455191IT250 Using Wildcard Masks



- Task:
  - Want Router A to permit entire sales network and just the 172.16.50.2 station.
  - Deny all other traffic from entering Administrative network.

For Reg. No 191552191IT215 - 191455191IT250 Using Wildcard Masks




```
RouterA(config)#access-list 11 permit 172.16.30.0 0.0.0.255
RouterA(config)#access-list 11 permit 172.16.50.2 0.0.0.0
```

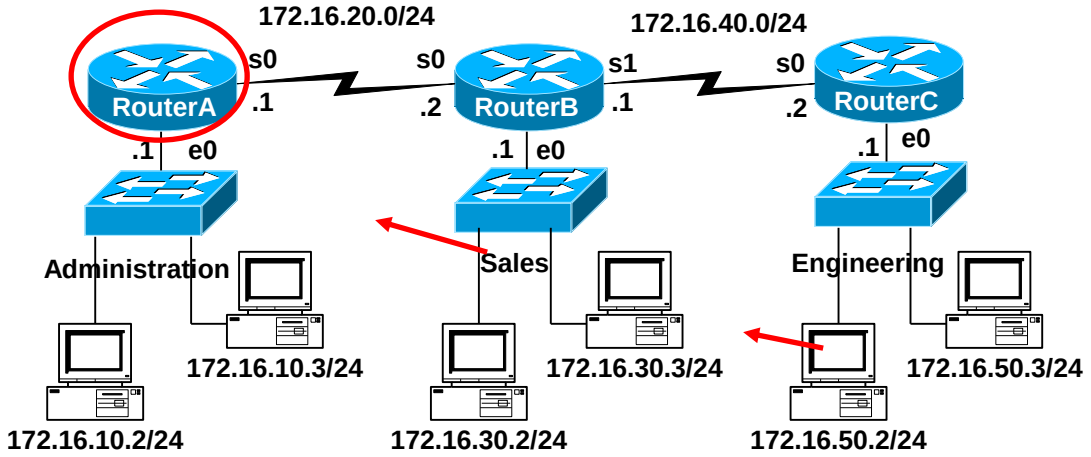
172.16.30.0 **0.0.0.255**

- 0 check - make sure first octet is 172
- 0 check - make sure second octet is 16
- 0 check - make sure third octet is 30
- 255 - don't check (*permit* any fourth octet)

172.16.50.2 0.0.0.0

- 0 check - make sure first octet is 172
- 0 check - make sure second octet is 16
- 0 check - make sure third octet is 50
- 0 check - make sure fourth octet is 2

For Reg. No 191552191IT215 - 191455191IT250 Using Wildcard Masks



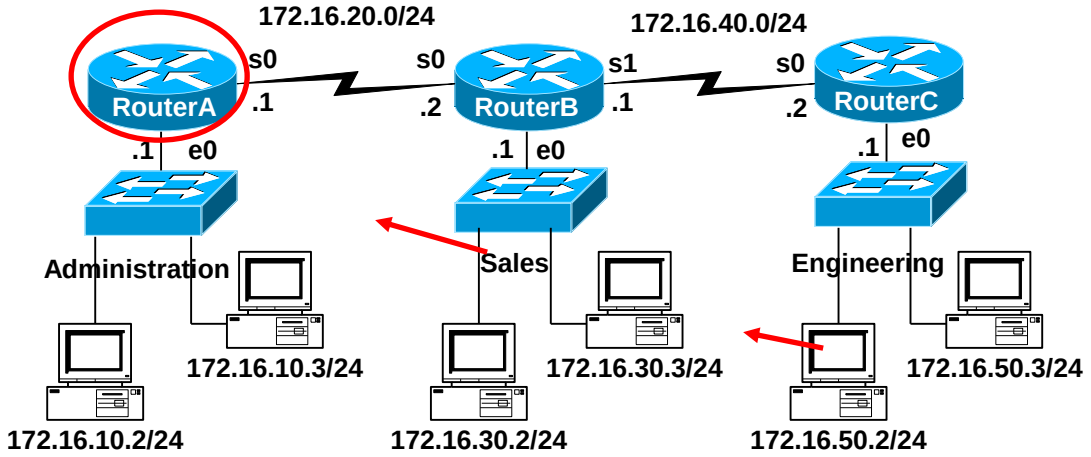

```
RouterA(config)#access-list 11 permit 172.16.30.0 0.0.0.255
0 = check, we want this to match, 1 = don't check (don't care)
```

```
172.16.30.0      10101100 . 00010000 . 00011110 . 00000000
0.0.0.255        00000000 . 00000000 . 00000000 . 11111111
                 -----------------------------------
172.16.30.0      10101100 . 00010000 . 00011110 . 00000000
172.16.30.1      10101100 . 00010000 . 00011110 . 00000001
                                                 ... (through)
172.16.30.255    10101100 . 00010000 . 00011110 . 11111111
```

Test Conditon

The packet(s)

For Reg. No 191552191IT215 - 191455191IT250 Using Wildcard Masks



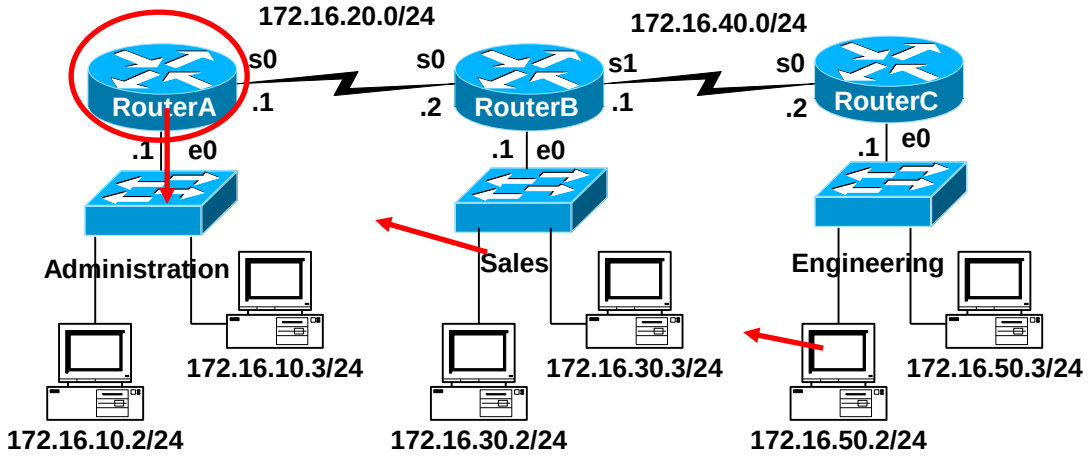

```
RouterA(config)#access-list 11 permit 172.16.50.2 0.0.0.0
0 = check, we want this to match, 1 = don't check (don't care)
```

```
172.16.50.2      10101100 . 00010000 . 00110010 . 00000010
0.0.0.0          00000000 . 00000000 . 00000000 . 00000000
                 ---------------------------------------
172.16.50.2      10101100 . 00010000 . 00110010 . 00000010
```

Test Conditon

The packet(s)

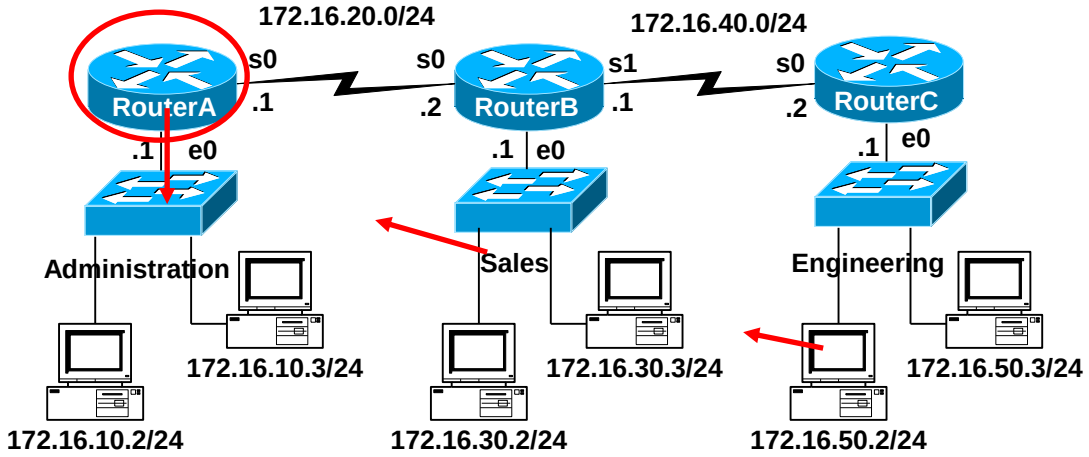
For Reg. No 191552191IT215 - 191455191IT250 Using Wildcard Masks



Don’t forget to apply the access-list to an interface.

```
RouterA(config)#access-list 11 permit 172.16.30.0 0.0.0.255
RouterA(config)#access-list 11 permit 172.16.50.2 0.0.0.0

RouterA(config)# interface e 0
RouterA(config-if)#ip access-group 11 out
```

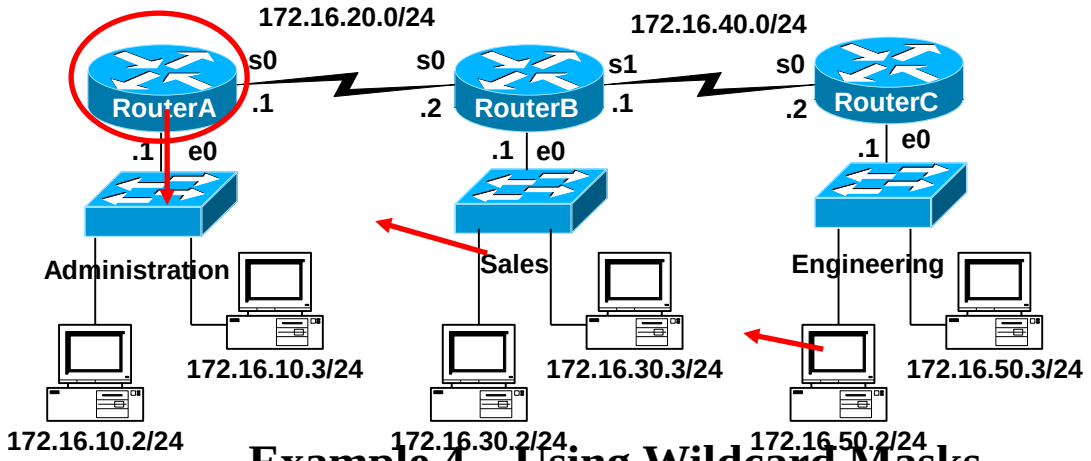
For Reg. No 191552191IT215 - 191455191IT250 Using Wildcard Masks



Remember that implicit deny any? It's a good idea for beginners to include the deny any statement just as a reminder.

```
RouterA(config)#access-list 11 permit 172.16.30.0 0.0.0.255
RouterA(config)#access-list 11 permit 172.16.50.2 0.0.0.0
RouterA(config)#access-list 11 deny 0.0.0.0 255.255.255.255

RouterA(config)# interface e 0
RouterA(config-if)#ip access-group 11 out
```

For Reg. No 191552191IT215 - 191455191IT250 Using Wildcard Masks



Example 4 – Using Wildcard Masks

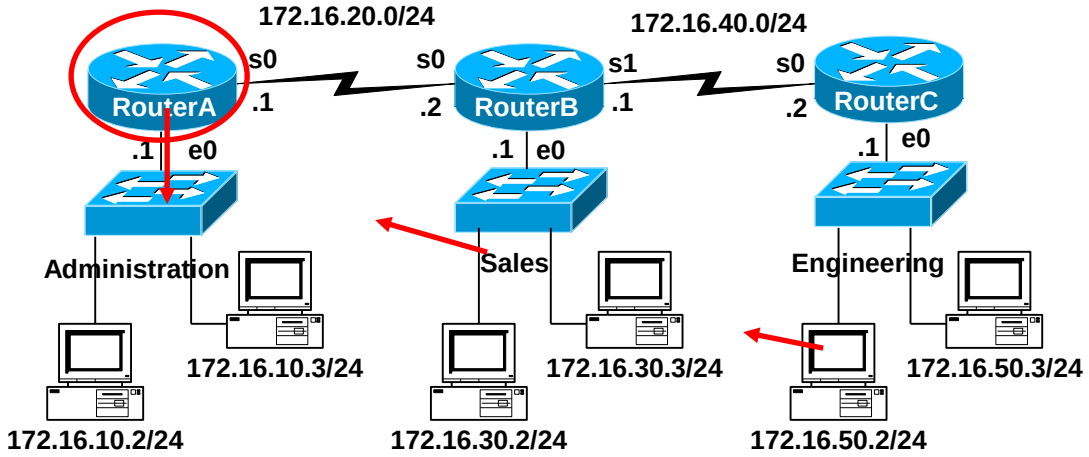
```
RouterA(config)#access-list 11 deny 0.0.0.0 255.255.255.255
0 = check, we want this to match, 1 = don't check (don't care)
```

```
0.0.0.0           00000000 . 00000000 . 00000000 . 00000000
255.255.255.255   11111111 . 11111111 . 11111111 . 11111111
                  ---------------------------------------

0.0.0.0           00000000 . 00000000 . 00000000 . 00000000
0.0.0.1           00000000 . 00000000 . 00000000 . 00000001
                                 ... (through)
255.255.255.255   11111111 . 11111111 . 11111111 . 11111111
```

Test Conditon

The packet(s)

For Reg. No 191061191IT251- 191185191IT258 Using “any” keyword

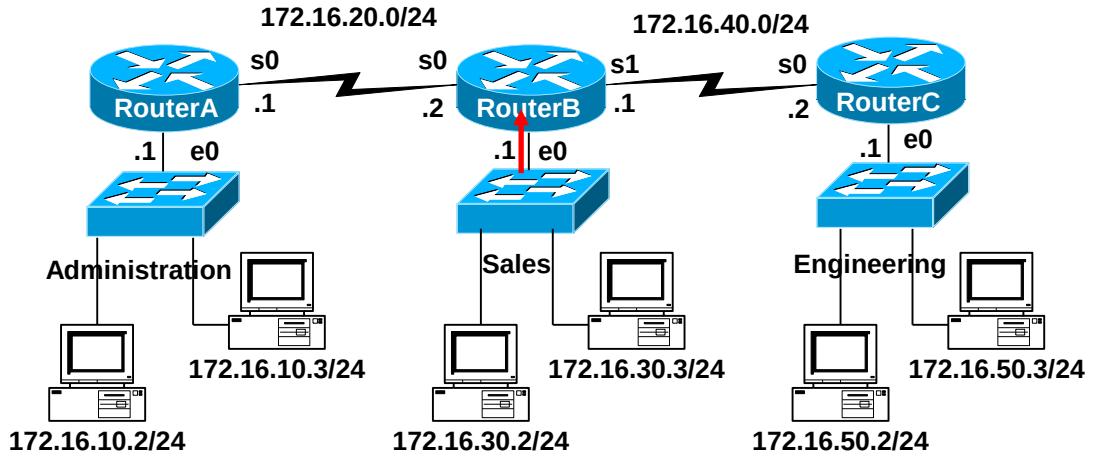



```
RouterA(config)#access-list 11 deny 0.0.0.0 255.255.255.255
Or
RouterA(config)#access-list 11 deny any
```

any = 0.0.0.0 255.255.255.255

- Simply put, the **any** option substitutes 0.0.0.0 for the IP address and 255.255.255.255 for the wildcard mask.
- This option will match any address that it is compared against.

For Reg. No 191061191IT251- 191185191IT258 Using “any” keyword

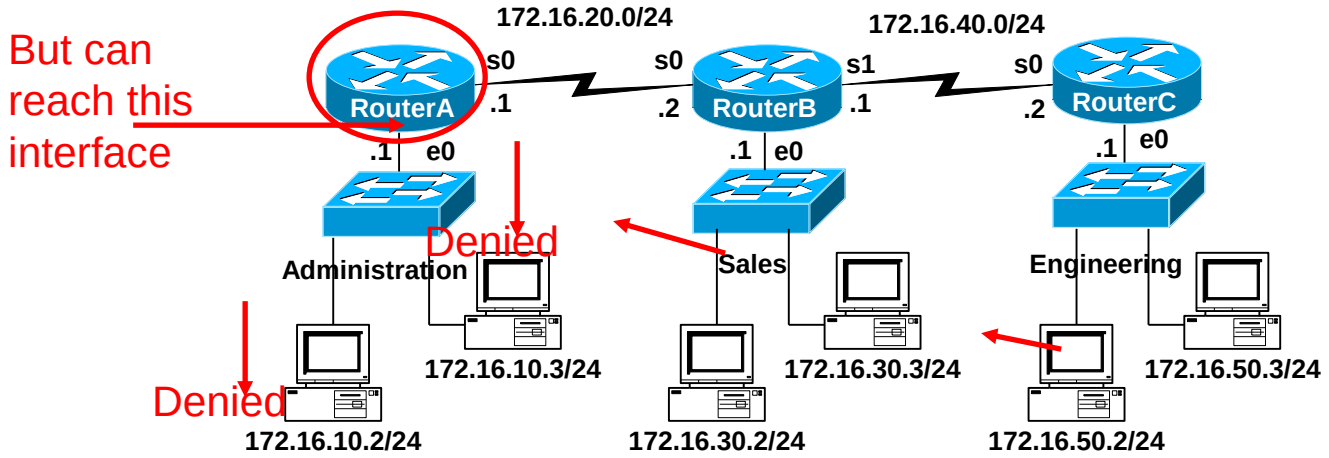



```
RouterB(config)#access-list 10 deny 172.16.30.2
RouterB(config)#access-list 10 permit any
or
RouterB(config)#access-list 10 permit 0.0.0.0 255.255.255.255
```

Previous example:

- Deny only the host 172.16.30.2 from exiting the Sales network.
- Permit all other hosts on the Sales network to leave the 172.16.30.0/24 network.
- Keyword “any” can be used to represent all IP Addresses.

A note about outbound access lists



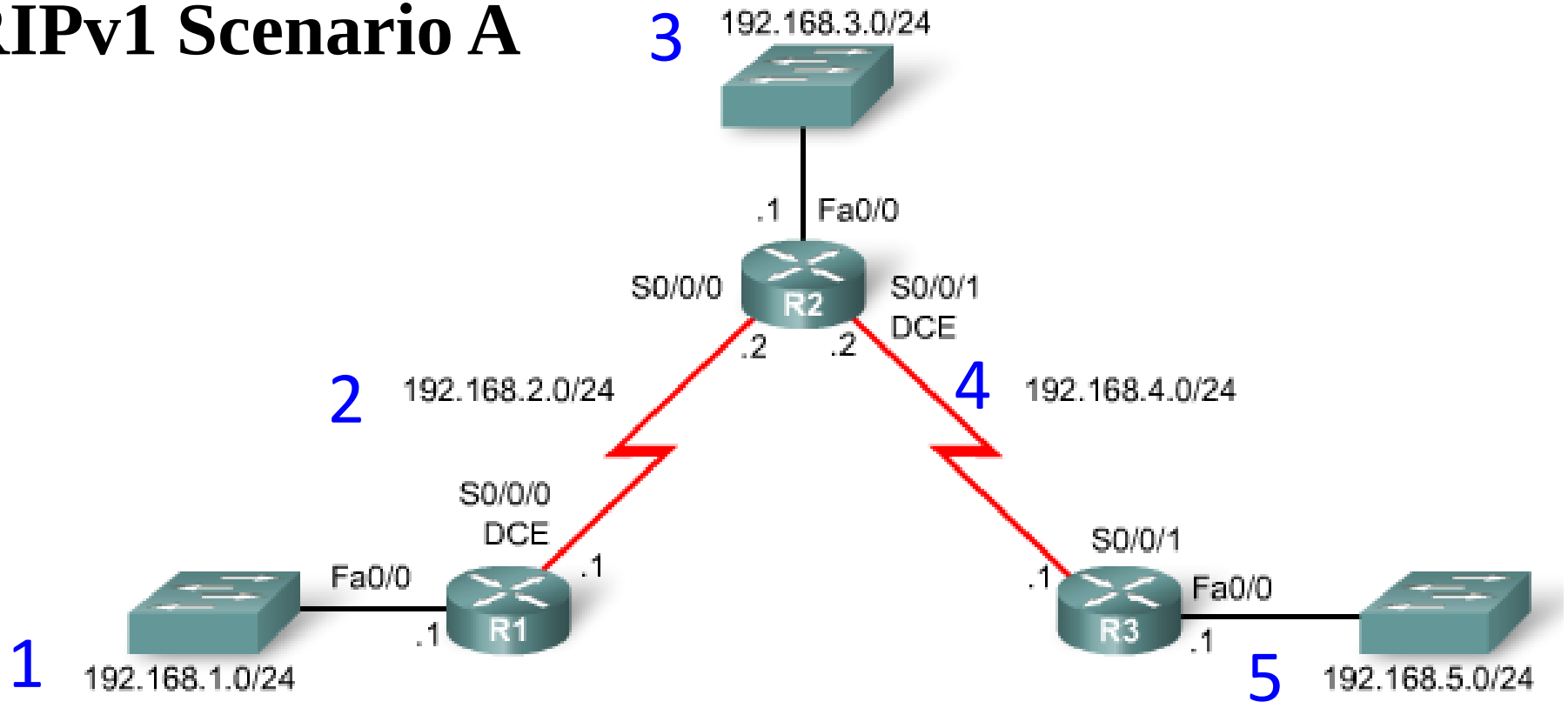

```
RouterA(config)#access-list 11 permit 172.16.30.0 0.0.0.255
RouterA(config)#access-list 11 permit 172.16.50.2 0.0.0.0
RouterA(config)#access-list 11 deny 0.0.0.0 255.255.255.255
RouterA(config)# interface e 0
RouterA(config-if)#ip access-group 11 out
```

This will deny packets from 172.16.30.0/24 from reaching all devices in the 172.16.10.0/24 Administration LAN, except RouterA's Ethernet 0 interface, of 172.16.10.1. The access list will need to be applied on Router A's Serial 0 interface for it to be denied on RouterA's Ethernet 0 interface. A better soluton is to use an Extended Access list. (coming)

Basic RIPv1 Configuration

- RIPv1 Scenario A
- Enable RIP: router rip Command
- Specifying Networks

RIPv1 Scenario A



- RIPv1 is a classful or classless routing protocol?
  - Classful
- How many classful networks are there and of what class?
  - 5 Class C network addresses.
- We will see that the class of the network is used by RIPv1 to determine the subnet mask.

Enabling RIP: router rip Command (don't do this yet)

```
R1# conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)# router ?
  bgp        Border Gateway Protocol (BGP)
  egp        Exterior Gateway Protocol (EGP)
  eigrp      Enhanced Interior Gateway Routing Protocol (EIGRP)
  igrp       Interior Gateway Routing Protocol (IGRP)
  isis       ISO IS-IS
  iso-igrp   IGRP for OSI networks
  mobile     Mobile routes
  odr        On Demand stub Routes
  ospf       Open Shortest Path First (OSPF)
  rip        Routing Information Protocol (RIP)
R1(config)# router rip
R1(config-router)#
```

- What routing protocols does this router support? (PT is limited)
- Configure RIP…

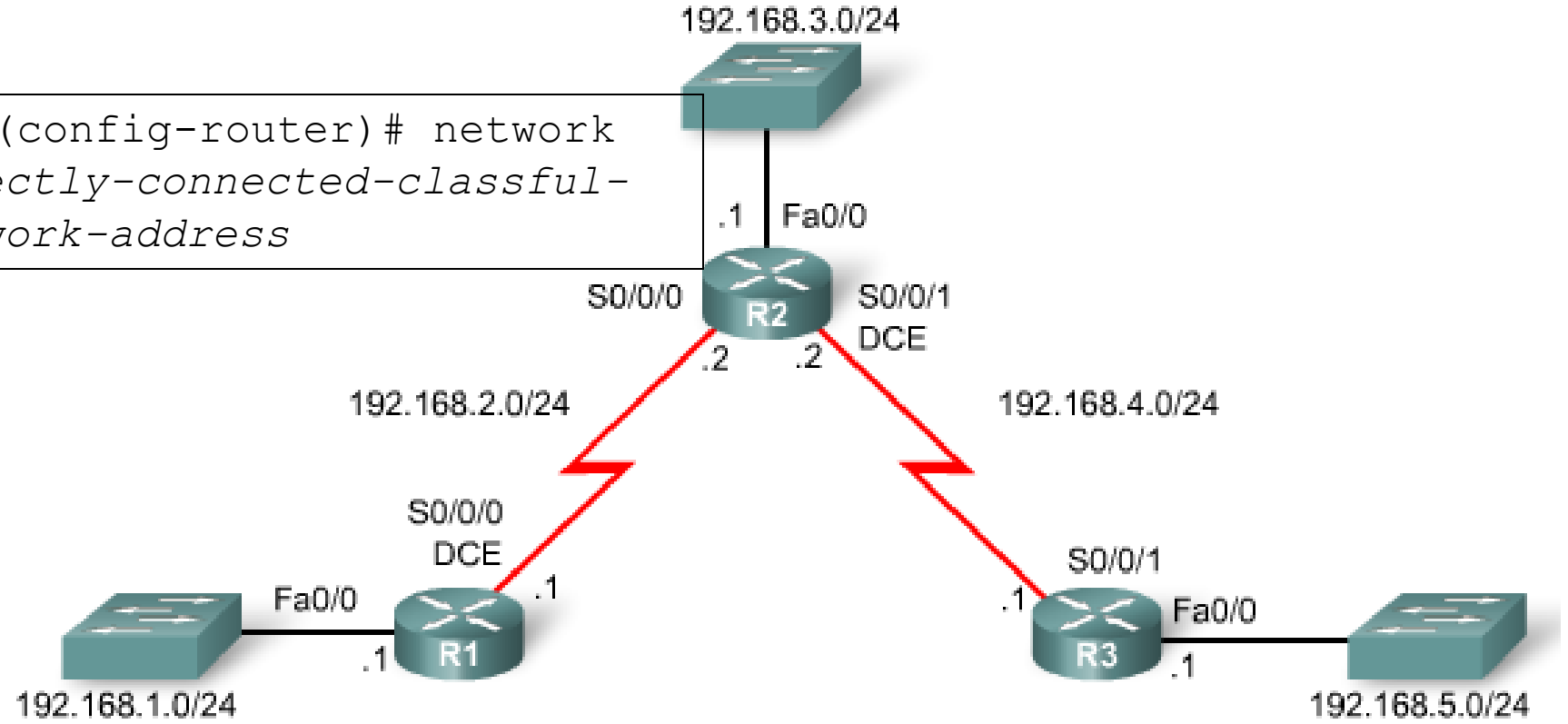
Enabling RIP: router rip Command (don’t do this yet)

- **no router rip**
  - To remove the RIP routing process from a device
  - Stops the RIP process
  - Erases all existing RIP configuration commands.

```
R1# conf t
R1(config)# router rip
R1(config-router)# network 192.168.1.0
R1(config-router)# network 192.168.2.0
R1(config-router)# exit
R1(config)# no router rip
```

Specifying Networks

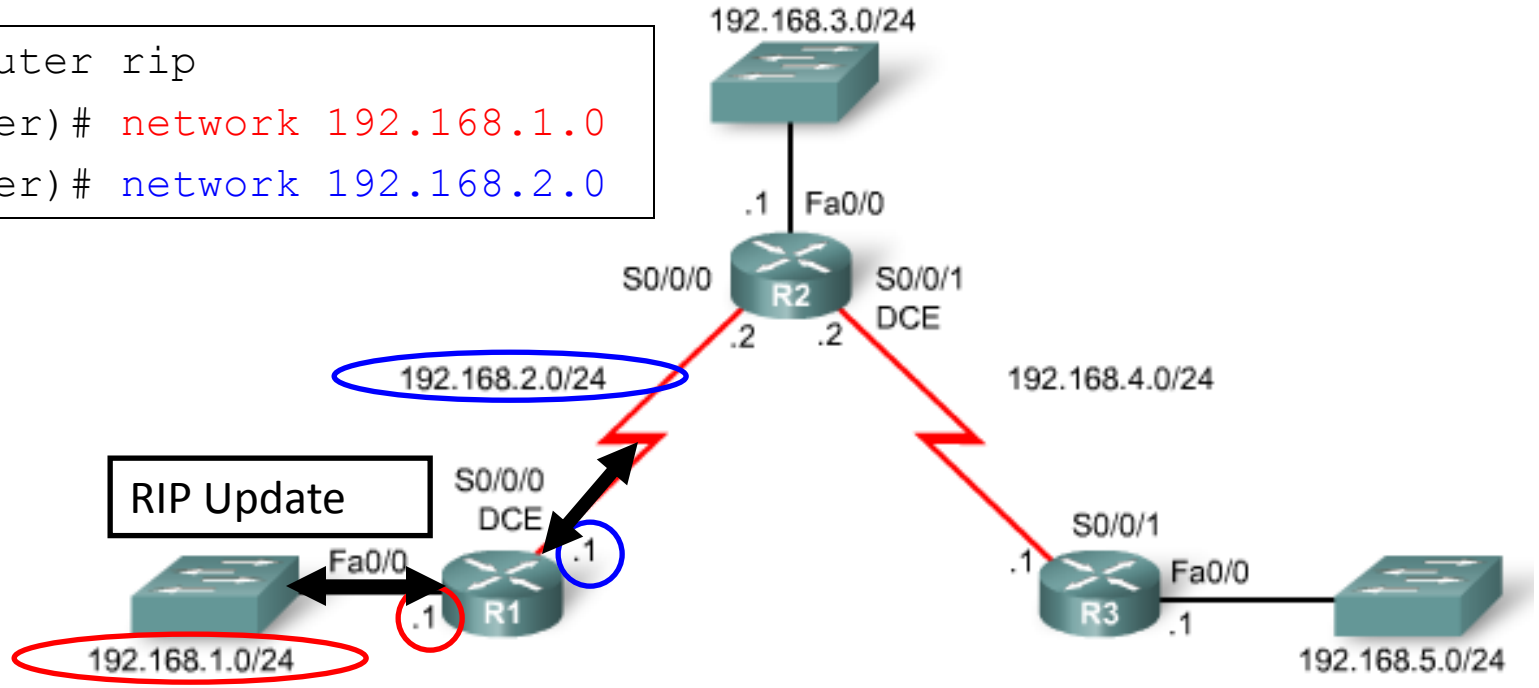
```
Router(config-router)# network
   directly-connected-classful-
   network-address
```



- To enable RIP routing for a network, use the **network** command in router configuration mode
- Enter the classful network address for each directly connected network.

Specifying Networks

```
R1(config)# router rip
R1(config-router)# network 192.168.1.0
R1(config-router)# network 192.168.2.0
```



- The **network** command performs the following functions:
  - Enables RIP on all interfaces that belong to a specific network.
    - Associated interfaces will now both send and receive RIP updates.
  - Advertises the specified network in RIP routing updates sent to other routers every 30 seconds (no mask).

Specifying Networks

Only directly connected classful network addresses!

```
R1(config)# router rip
R1(config-router)# network 192.168.1.0
R1(config-router)# network 192.168.2.0
```

```
R2(config)# router rip
R2(config-router)# network 192.168.2.0
R2(config-router)# network 192.168.3.0
R2(config-router)# network 192.168.4.0
```

```
R3(config)# router rip
R3(config-router)# network 192.168.4.0
R3(config-router)# network 192.168.5.0
```

- Configure RIP for all three routers
- What happens if you enter a subnet or host IP address? (Try it)
  - IOS automatically converts it to a classful network address.
  - For example, if you enter the command **network 192.168.1.32**, the router will convert it to **network 192.168.1.0**.

```
R1(config)# router rip
R1(config-router)# network 192.168.1.0
R1(config-router)# network 192.168.2.0
```

```
R2(config)# router rip
R2(config-router)# network 192.168.2.0
R2(config-router)# network 192.168.3.0
R2(config-router)# network 192.168.4.0
```

```
R3(config)# router rip
R3(config-router)# network 192.168.4.0
R3(config-router)# network 192.168.5.0
```

Only directly connected classful network addresses!