

Access Control List (ACL)

1

Muhammad Zen Samsono Hadi, ST. MSc.

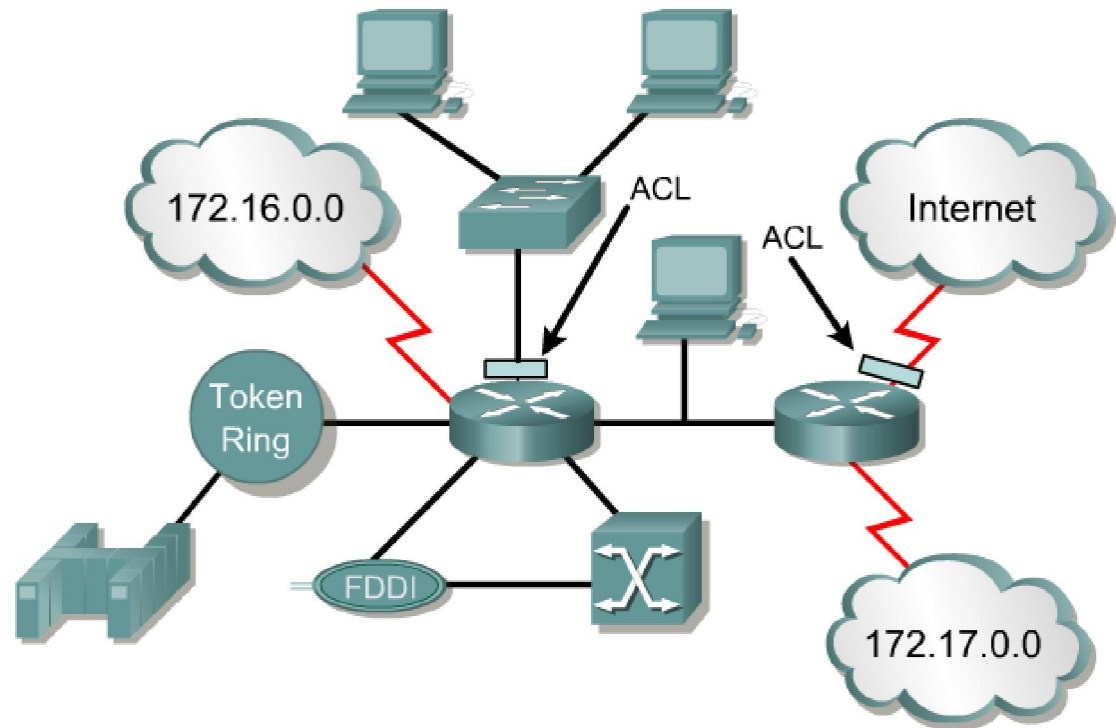
**Lab. Komunikasi Digital
Gedung D4 Lt. 1
EEPIS-ITS**

Overview

2

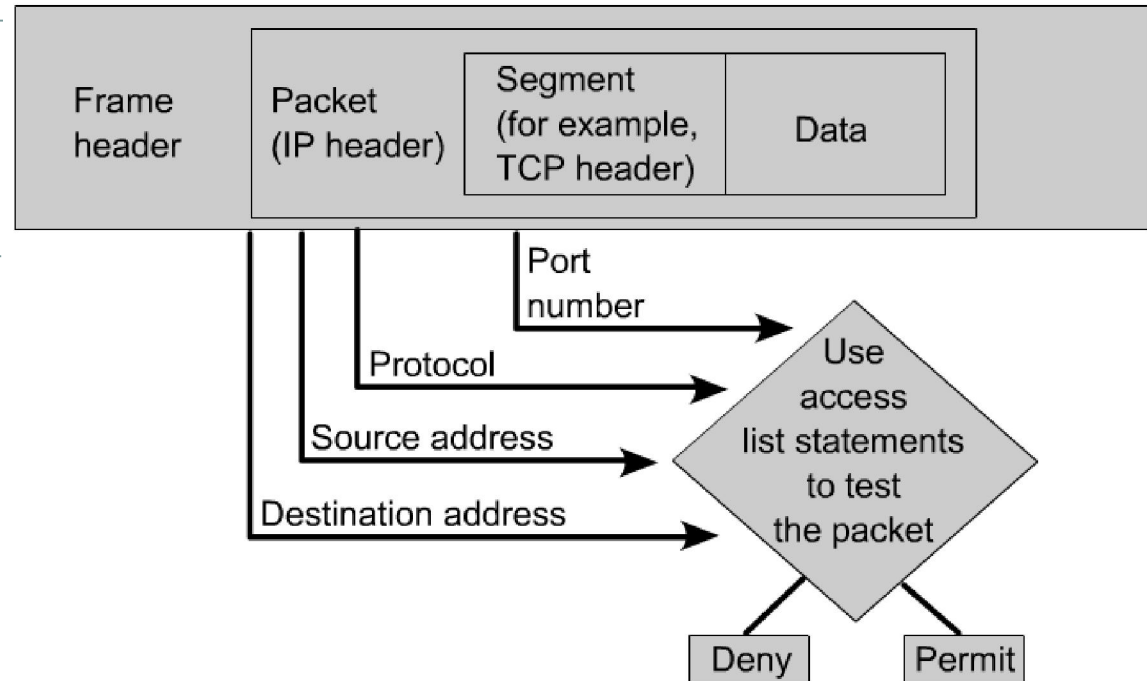
- Network administrators must figure out how to deny unwanted access to the network while allowing internal users appropriate access to necessary services.
- Although security tools, such as passwords, callback equipment, and physical security devices are helpful, they often lack the flexibility of basic traffic filtering and the specific controls most administrators prefer.
- For example, a network administrator may want to allow users access to the Internet, but not permit external users telnet access into the LAN.
- Routers provide basic traffic filtering capabilities, such as blocking Internet traffic, with **access control lists (ACLs)**.
- An ACL is a sequential list of permit or deny statements that apply to addresses or upper-layer protocols.
- This module will introduce **standard** and **extended ACLs** as a means to control network traffic, and how ACLs are used as part of a security solution.

What are ACLs?



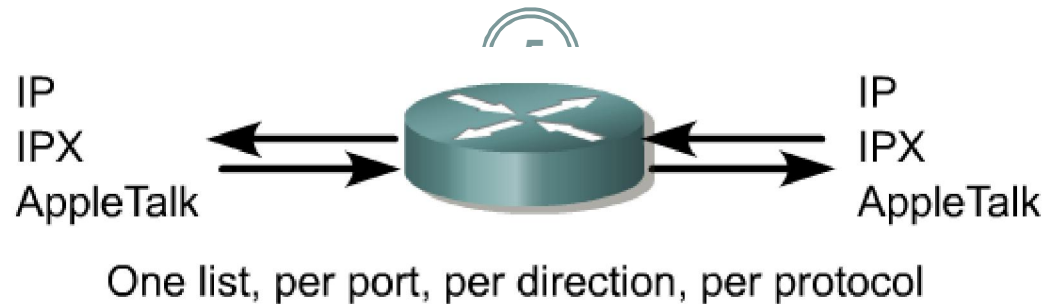
- An access list is a sequential series of commands or filters.
- These lists tell the router what types of packets to:
 - accept or
 - deny
- Acceptance and denial can be based on specified conditions.
- ACLs applied on the router's interfaces.

What are ACLs?



- The router examines each packet to determine whether to forward or drop it, based on the conditions specified in the ACL.
- *Some* ACL decision points are:
 - IP source address
 - IP destination addresses
 - UDP or TCP protocols
 - upper-layer (TCP/UDP) port numbers

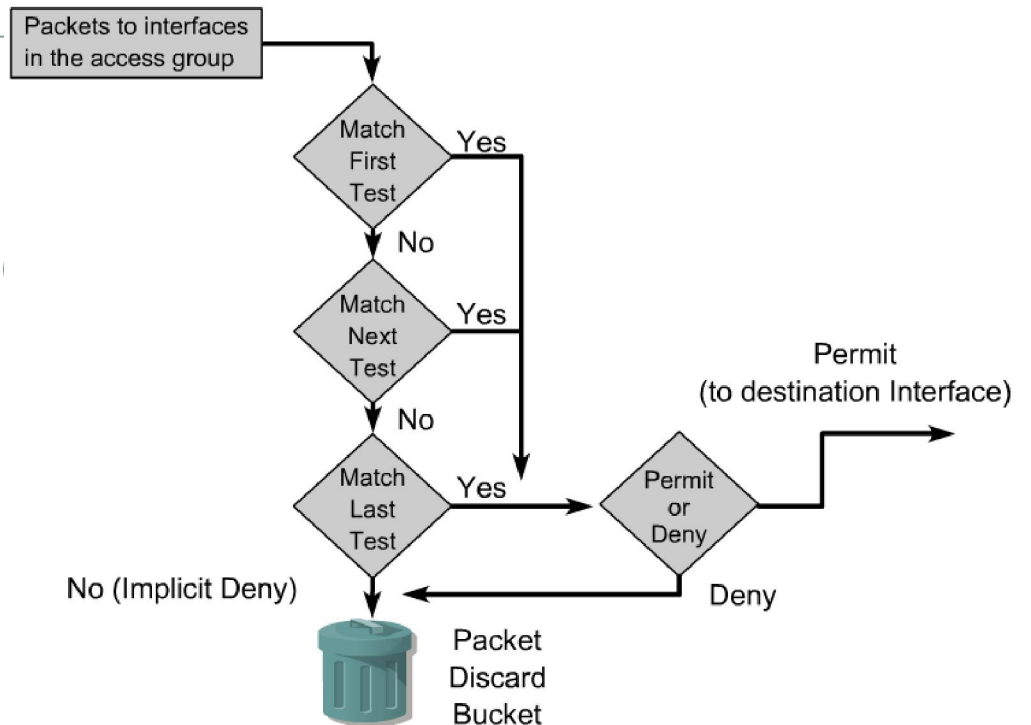
What are ACLs?



With two interfaces and three protocols running, this router could have a total of 12 separate ACLs applied.

- ACLs must be defined on a:
 - per-protocol (IP, IPX, AppleTalk)
 - per direction (in or out)
 - per port (interface) basis.
- ACLs control traffic in one direction at a time on an interface.
- A separate ACL would need to be created for each direction, one for inbound and one for outbound traffic.
- Finally every interface can have multiple protocols and directions defined.

How ACLs work



- An ACL is a group of statements that define whether packets are accepted or rejected coming into an interface or leaving an interface.
- ACL statements operate in sequential, logical order (top down).
- If a condition match is true, the packet is permitted or denied and the rest of the ACL statements are not checked.
- If all the ACL statements are unmatched, an implicit **"deny any"** statement is placed at the end of the list by **default**. (not visible)
- When first learning how to create ACLs, it is a good idea to add the **implicit deny** at the end of ACLs to reinforce the dynamic presence of the command line.

Two types of ACLs

7

- Standard IP ACLs
 - Can only filter on source IP addresses
- Extended IP ACLs
 - Can filter on:
 - ✦ Source IP address
 - ✦ Destination IP address
 - ✦ Protocol (TCP, UDP)
 - ✦ Port Numbers (Telnet – 23, http – 80, etc.)
 - ✦ *and other parameters*

Creating Standard ACLs – 2 Steps



Step 1

Define the ACL by using the following command:

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

A global statement identifies the ACL. Specifically, the 1-99 range is reserved for standard IP. This number refers to the type of ACL. In Cisco IOS Release 11.2 or newer, ACLs can also use an ACL name, such as `education_group`, rather than a number.

The **permit** or **deny** term in the global ACL statement indicates how packets that meet the test conditions are handled by Cisco IOS software. **permit** usually means the packet will be allowed to use one or more interfaces that you will specify later. The final term or terms specifies the test conditions used by the ACL statement.

Step 2

Next, you need to apply ACLs to an interface by using the **access-group** command, as in this example:

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

All the ACL statements identified by *access-list-number* are associated with one or more interfaces. Any packets that pass the ACL test conditions can be permitted to use any interface in the access group of interfaces.

Creating ACLs – 2 Steps



Step 1

Define the ACL by using the following command:

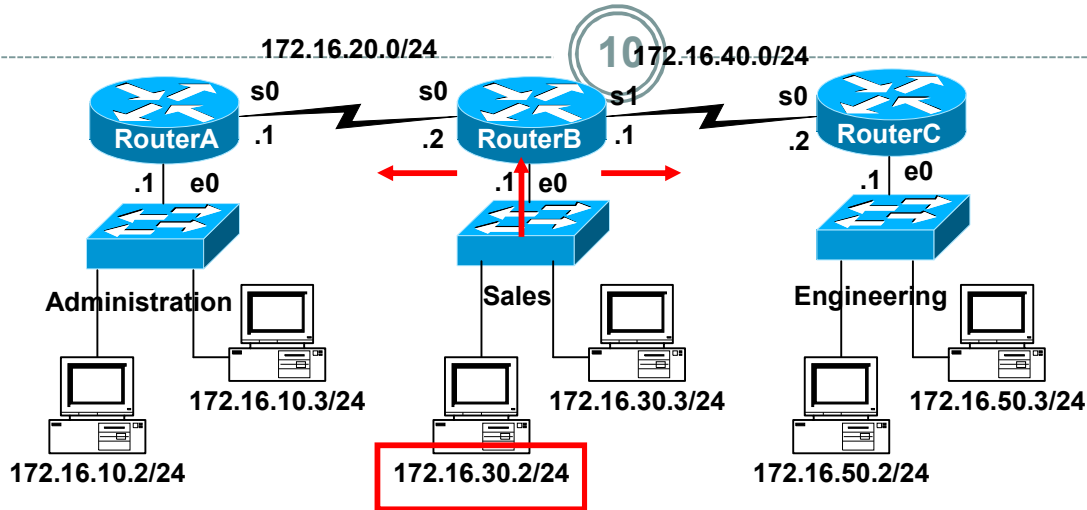
```
Router(config)#access-list access-list-number  
          {permit | deny} {test-conditions}
```

A global statement identifies the ACL. Specifically, the 1-99 range is reserved for standard IP. This number refers to the type of ACL. In Cisco IOS Release 11.2 or newer, ACLs can also use an ACL name, such as `education_group`, rather than a number.

The **permit** or **deny** term in the global ACL statement indicates how packets that meet the test conditions are handled by Cisco IOS software. **permit** usually means the packet will be allowed to use one or more interfaces that you will specify later. The final term or terms specifies the test conditions used by the ACL statement.

Protocol	Range
IP (Standard IP)	1-99
Extended IP	100-199
AppleTalk	600-699
IPX	800-899
Extended IPX	900-999
IPX Service Advertising Protocol	1000-1099

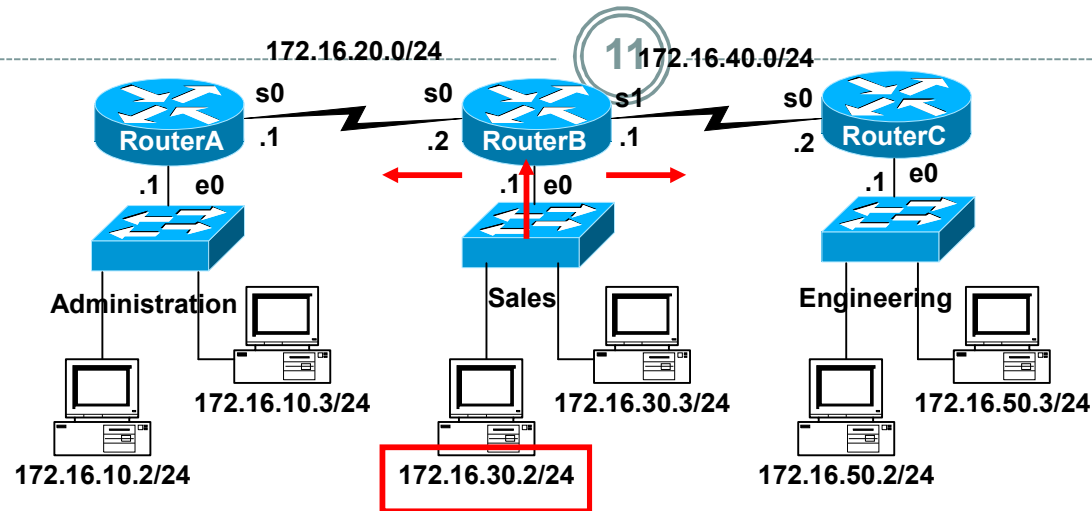
Learn by example!



- 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.

Learn by example!



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}
```

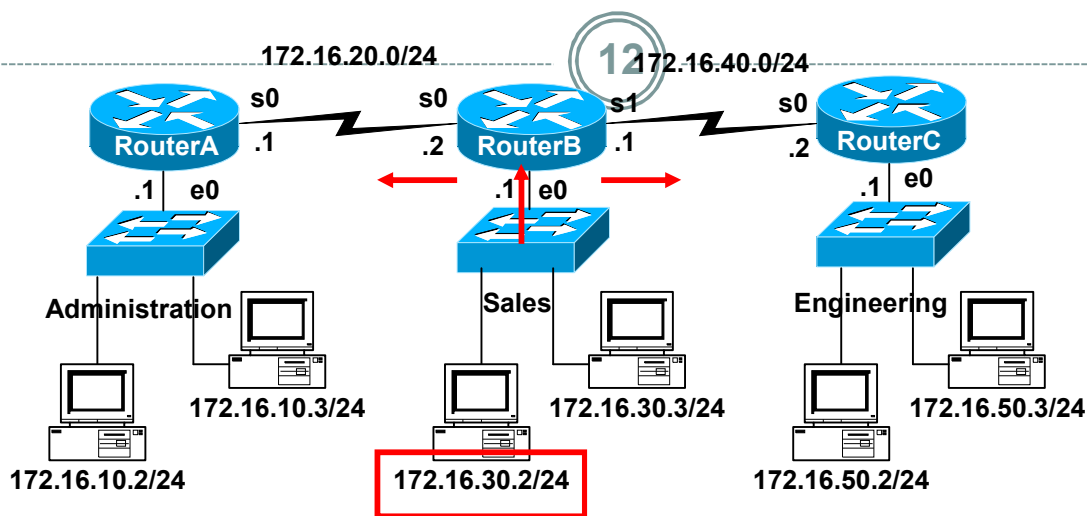
Protocol

IP (Standard IP)

Range

1-99

From Cisco Web Site



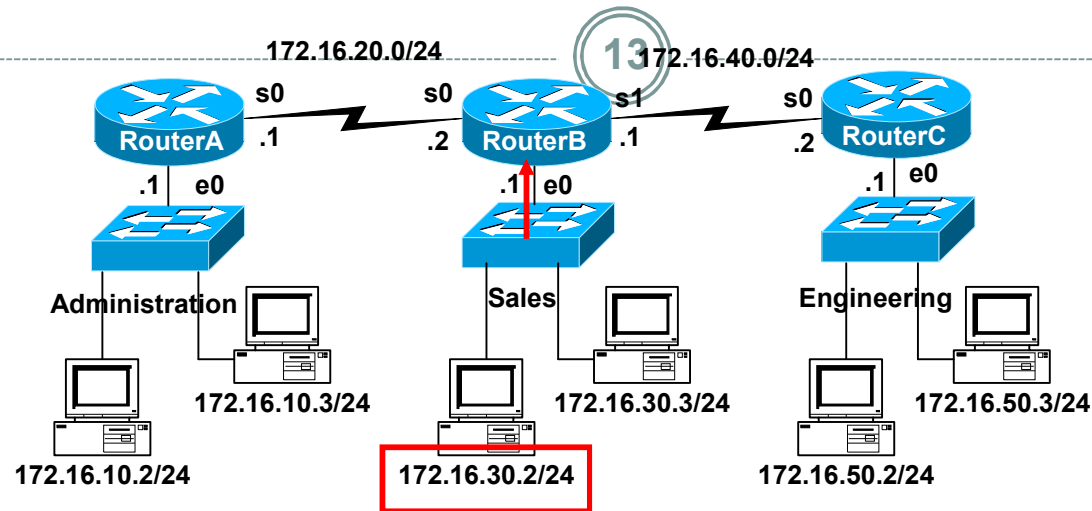
Applying ACLs

- 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.** (coming later)

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.

Learn by example!



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

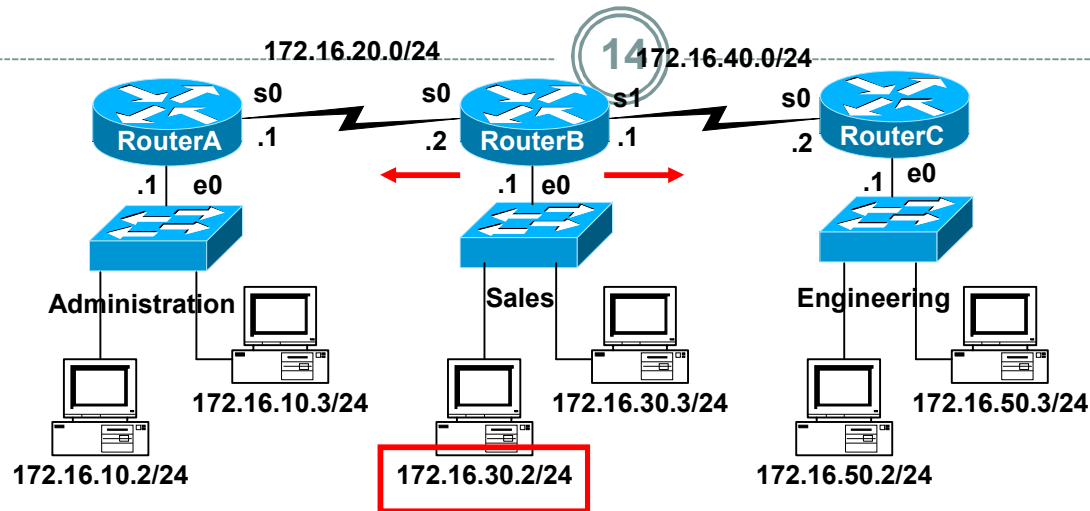
```
RouterB(config) #access-list 10 deny 0.0.0.0 255.255.255.255
```

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

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

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

Learn by example!



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
```

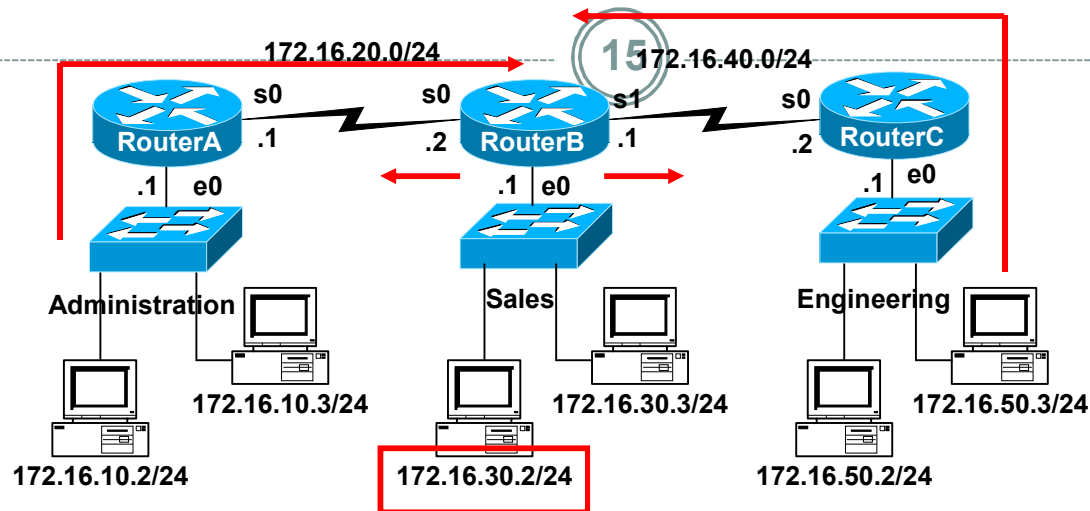
```
RouterB(config) # interface s 0
```

```
RouterB(config-if) # ip access-group 10 out
```

```
RouterB(config) # interface s 1
```

```
RouterB(config-if) # ip access-group 10 out
```

Learn by example!

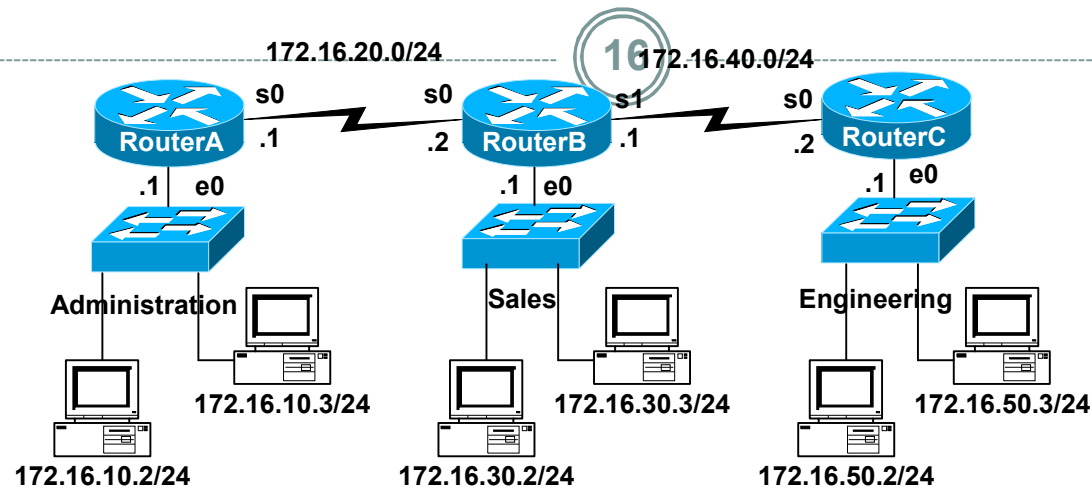


Because of the implicit deny any, this has an adverse affect of also denying packets from Administration from reaching Engineering, and denying packets from Engineering from reaching Administration.

```
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
```

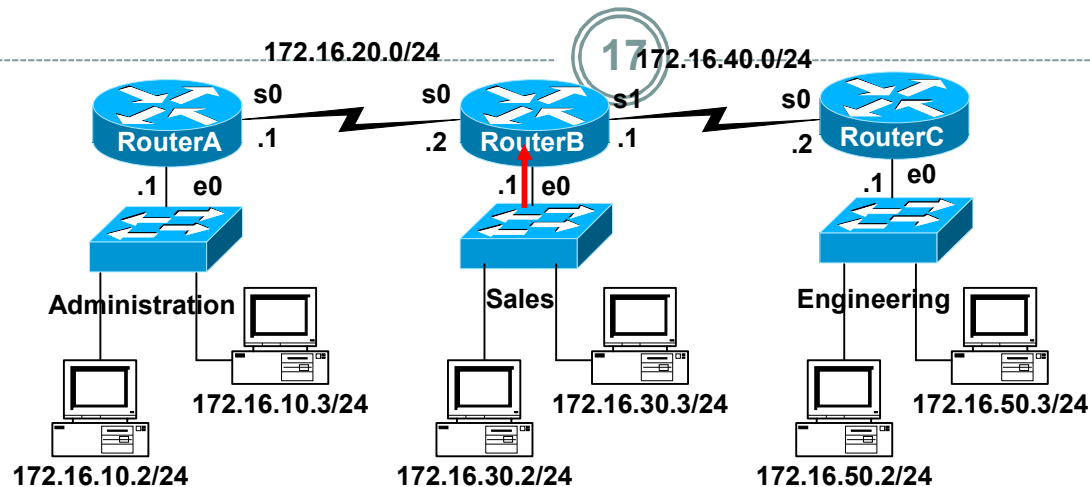
```
RouterB(config)# interface s 0  
RouterB(config-if)# ip access-group 10 out  
RouterB(config)# interface s 1  
RouterB(config-if)# ip access-group 10 out
```

Example 2



- 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.

Example 2

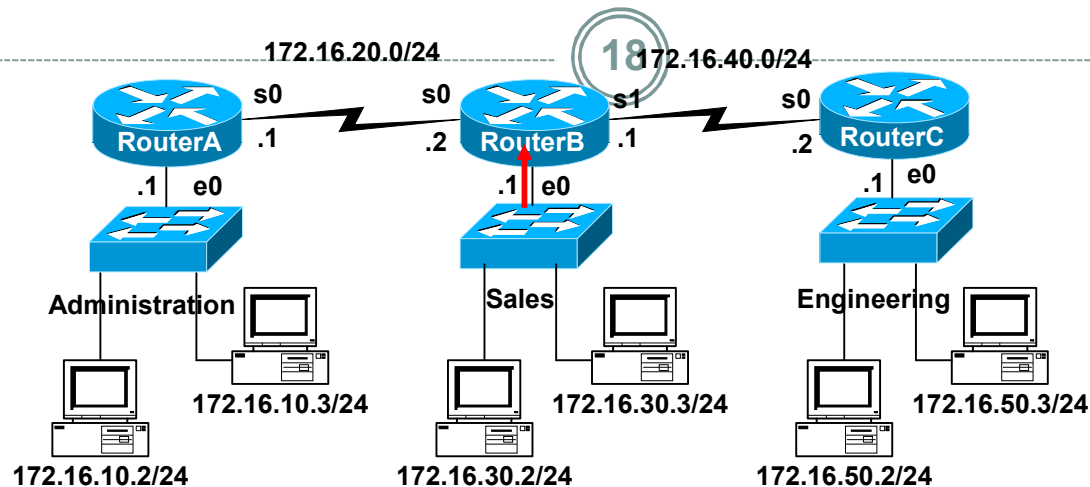


Once a condition is met, all other statements are ignored, so the implicit **deny any** only applies to not-matched packets.

```
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
```

Example 2



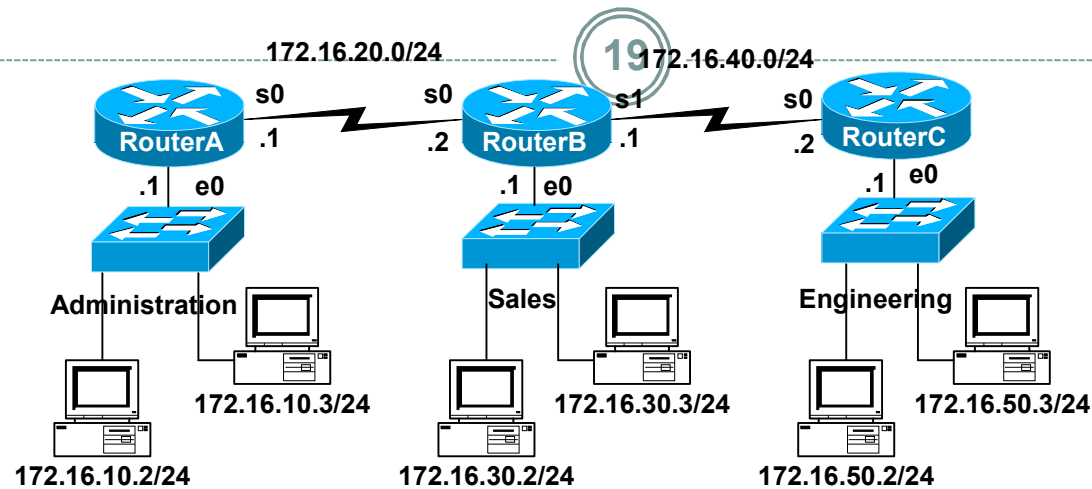
To remove an Access List, use the `no access-list` command. Removing the access-group only from the interface leaves the access list, but they are not currently being applied. Usually, best to remove it from both.

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

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

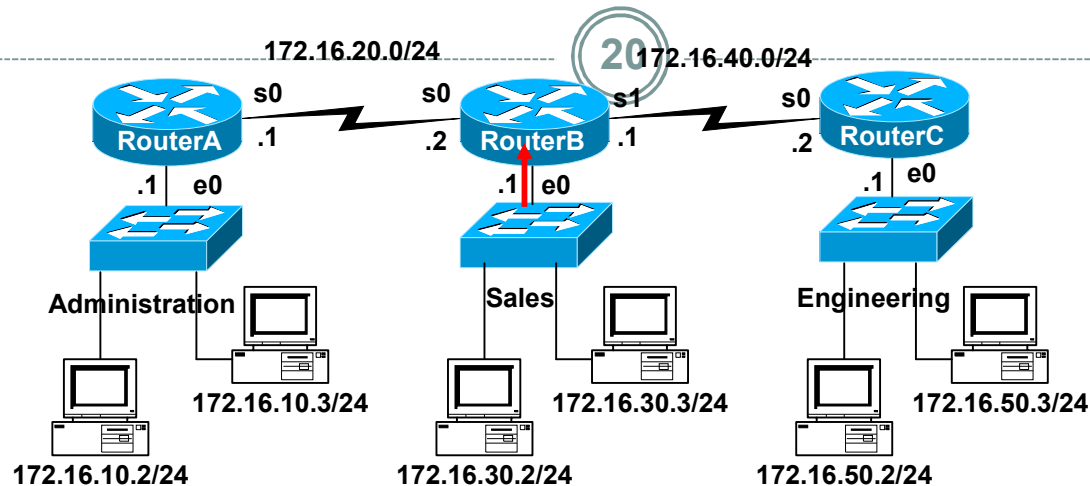
```
RouterB(config-if)# no ip access-group 10 in
```

Example 3



- 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.

Example 3



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
```

Notes from www.cisco.com

21

- 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)
```

Time for 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!

Test Condition

Access-list 1 permit 172.16.0.0 0.0.255.255

Test
Condition

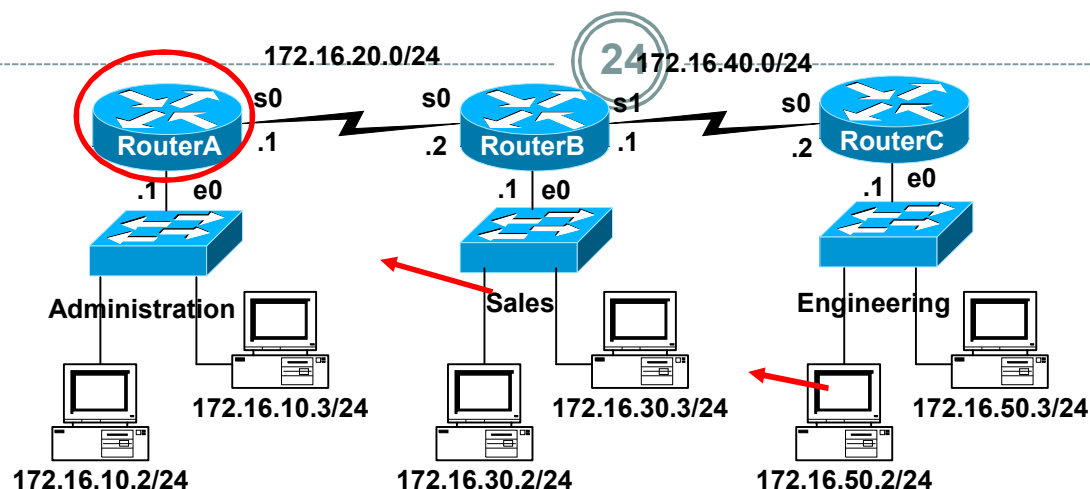
172.16.0.0	10101100.00010000	00000000.00000000
0.0.255.255	00000000.00000000	11111111.11111111
-----		-----
Must Match		No Match Necessary
<u>A Match...</u>		The packet
172.16.---	10101100.00010000	any value.any value

Resulting in the bits that must match or doesn't matter.

Matching packets will look like this.

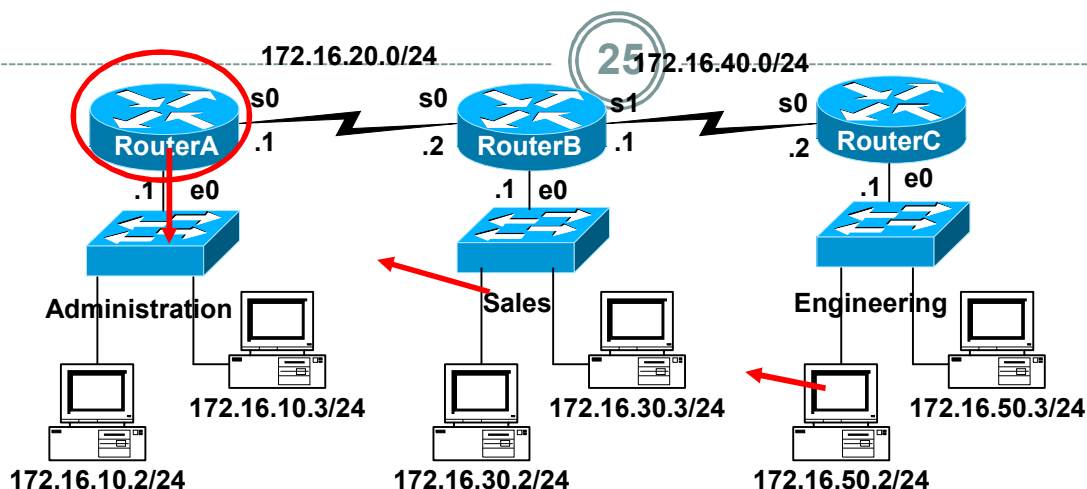
- 0 - “check the corresponding bit value.”
- 1 - “do not check (ignore) that corresponding bit value.”

Example 4 – Using Wildcard Masks



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

Example 4 – 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
```

Practice

26

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

Permit the following networks:

Network/Subnet Mask

- A. 172.16.0.0 255.255.0.0
- B. 172.16.1.0 255.255.255.0
- C. 192.168.1.0 255.255.255.0
- D. 172.16.16.0 255.255.240.0 (hmmm . . .?)
- E. 172.16.128.0 255.255.192.0 (hmmm . . .?)

Address/Wildcard Mask

Permit the following hosts:

Network/Subnet Mask

- A. 172.16.10.100
- B. 192.168.1.100
- C. All hosts

Address/Wildcard Mask

Practice – Do you see a relationship?

27

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

Permit the following networks:

<u>Network/Subnet Mask</u>	<u>Address/Wildcard Mask</u>
A. 172.16.0.0 255.255.0.0	172.16.0.0 <u>0.0.255.255</u>
B. 172.16.1.0 255.255.255.0	172.16.1.0 <u>0.0.0.255</u>
C. 192.168.1.0 255.255.255.0	192.168.1.0 <u>0.0.0.255</u>
D. 172.16.32.0 255.255.240.0	172.16.32.0 <u>0.0.15.255</u>
E. 172.16.128.0 255.255.192.0	172.16.128 <u>0.0.63.255</u>

Permit the following hosts:

<u>Network/Subnet Mask</u>	<u>Address/Wildcard Mask</u>
A. 172.16.10.100	172.16.10.100 0.0.0.0
B. 192.168.1.100	192.168.1.100 0.0.0.0
C. All hosts	0.0.0.0 <u>255.255.255.255</u>

255.255.255.255 – Subnet = Wildcard

28

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

Permit the following networks:

	255.255.255.255.	- Subnet Mask	=	Wildcard Mask
A.	255.255.255.255	- 255.255.0.0	=	0.0.255.255
B.	255.255.255.255	- 255.255.255.0	=	0.0.0.255
C.	255.255.255.255	- 255.255.255.0	=	0.0.0.255
D.	255.255.255.255	- 255.255.240.0	=	0.0.15.255
E.	255.255.255.255	- 255.255.192.0	=	0.0.63.255

Permit the following hosts: (host routes have a /32 mask)

	255.255.255.255.	- /32 Mask	=	Wildcard Mask
A.	255.255.255.255	- 255.255.255.255	=	0.0.0.0
B.	255.255.255.255	- 255.255.255.255	=	0.0.0.0

“host” option

29

```
RouterB(config) #access-list 10 permit 192.168.1.100 0.0.0.0
```

```
RouterB(config) #access-list 10 permit host 192.168.1.100
```

Permit the following hosts:

	<u>Network/Subnet Mask</u>	<u>Address/Wildcard Mask</u>
A.	172.16.10.100	172.16.10.100 0.0.0.0
B.	192.168.1.100	192.168.1.100 0.0.0.0

- The **host** option substitutes for the 0.0.0.0 mask.
- This mask requires that all bits of the ACL address and the packet address match.
- The host keyword precedes the IP address.
- This option will match just one address.

172.16.10.100 0.0.0.0 *replaced by* host 172.16.10.100

192.168.1.100 0.0.0.0 *replaced by* host 192.168.1.100

Verifying Access Lists

30

```
Router#show ip interface
```

```
FastEthernet0/0 is up, line protocol is down
```

```
Internet address is 192.168.1.1/24
```

```
Broadcast address is 255.255.255.255
```

```
MTU is 1500 bytes
```

```
Helper address is not set
```

```
Directed broadcast forwarding is disabled
```

```
Outgoing access list is not set
```

```
Inbound access list is 2
```

```
Serial0/0 is down, line protocol is down
```

```
Internet address is 200.200.2.1/24
```

```
Broadcast address is 255.255.255.255
```

```
MTU is 1500 bytes
```

```
Helper address is not set
```

```
Directed broadcast forwarding is disabled
```

```
Outgoing access list is not set
```

```
Inbound access list is 101
```

Verifying Access Lists

31

```
Router#show access-lists  
Standard IP access list 2  
  deny   172.16.1.1  
  permit 172.16.1.0, wildcard bits 0.0.0.255  
  deny   172.16.0.0, wildcard bits 0.0.255.255  
  permit 172.0.0.0, wildcard bits 0.255.255.255  
Extended IP access list 101  
  permit tcp 192.168.6.0 0.0.0.255 any eq telnet  
  permit tcp 192.168.6.0 0.0.0.255 any eq ftp  
  permit tcp 192.168.0.0 0.0.0.255 any eq ftp-data  
Router#
```

Verifying Access Lists



```
Router#show running-config
Current configuration : 953 bytes
!
interface FastEthernet0/0
  ip address 192.168.1.1 255.255.255.0
  → ip access-group 2 in
!
interface Serial0/0
  ip address 200.200.2.1 255.255.255.0
  → ip access-group 101 in
!
  { access-list 2 deny    172.16.1.1
    { access-list 2 permit 172.16.1.0 0.0.0.255
    { access-list 2 deny   172.16.0.0 0.0.255.255
    { access-list 2 permit 172.0.0.0 0.255.255.255
  { access-list 101 permit tcp 192.168.6.0 0.0.0.255 any
    { eq telnet
    { access-list 101 permit tcp 192.168.6.0 0.0.0.255 any
    { eq ftp
  !
```

- Note: More than one interface can use the same access-list.

Extended Access Lists

```
Router(config)#access-list access-list-number {permit | deny}  
protocol source  
[source-mask destination destination-mask operator operand]  
[established]
```

Paramter	Description
<i>access-list-number</i>	Identifies the list using a number in the range 100 to 199.
permit deny	Indicates whether this entry allows or blocks the specified address.
<i>protocol</i>	The protocol, such as IP, TCP, UDP, ICMP, GRE, or IGRP.
source and destination	Identifies source and destination addresses.
<i>source-mask</i> and <i>destination-mask</i>	Wildcard mask; zeros indicate positions that must match, ones indicate do not care positions.
<i>operator</i> <i>operand</i>	lt, gt, eq, neq (less than, greater than, equal, not equal), and a port number.
established	Allows TCP traffic to pass if the packet uses an established connection (for example, has ACK bits set).

Extended Access Lists

```
Router(config)#access-list access-list-number {permit | deny}  
protocol source  
[source-mask destination destination-mask operator operand]  
[established]
```

Protocol	Range
IP	1-99
Extended IP	100-199
AppleTalk	600-699
IPX	800-899
Extended IPX	900-999
IPX Service Advertising Protocol	1000-1099

- Extended ACLs are used more often than standard ACLs because they provide a **greater range of control**.
- Extended ACLs **check the source and destination packet addresses** as well as being able to **check for protocols and port numbers**.
- This gives **greater flexibility** to describe what the ACL will check.
- Packets can be permitted or denied access based on where the packet originated and its destination as well as protocol type and port addresses.

Extended Access Lists

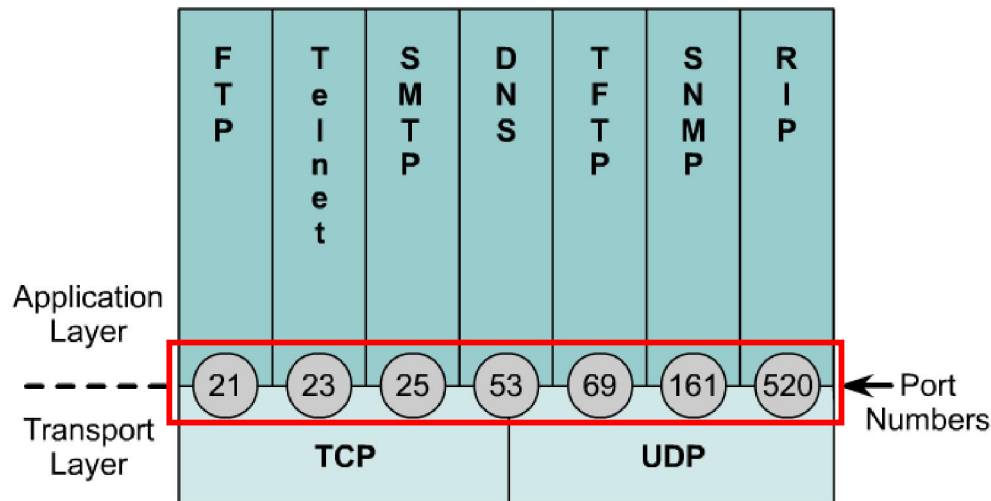
```
Router(config)#access-list access-list-number {permit | deny}  
protocol source  
[source-mask destination destination-mask operator operand]  
[established]
```

protocol

The protocol, such as IP, TCP, UDP, ICMP, GRE, or IGRP.

operator operand

lt, gt, eq, neq (less than, greater than, equal, not equal), and a port number.



- **Operator and operand** can also refer to ICMP Types and Codes or whatever the **protocol** is being checked.
- If the **operator and operand** follow the **source address** it refers to the **source port**
- If the **operator and operand** follow the **destination address** it refers to the **destination port**.

Extended Access Lists - Examples



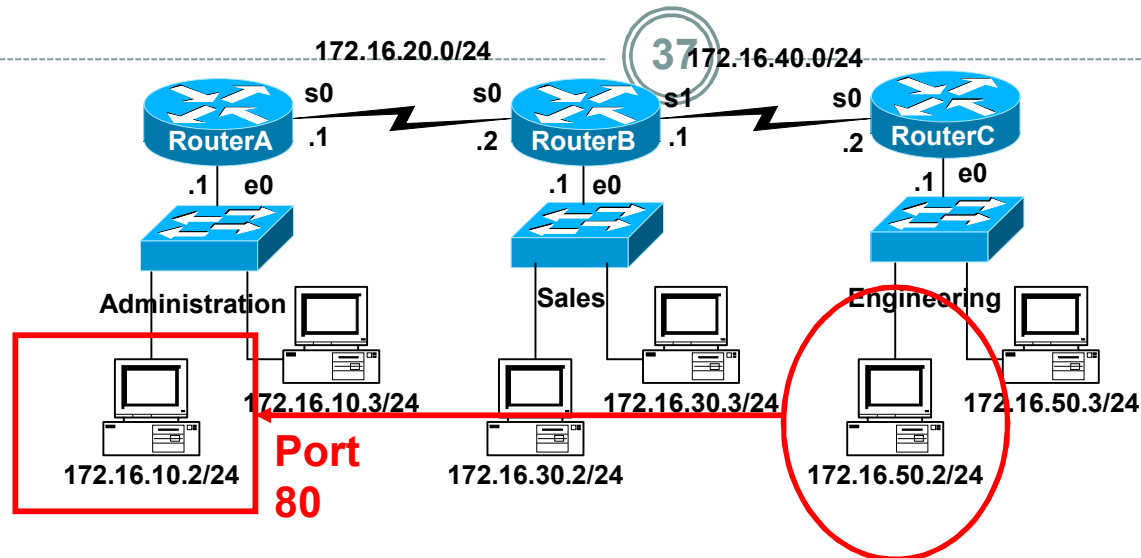
```
access-list 114 permit tcp 172.16.6.0 0.0.0.255 any eq telnet
access-list 114 permit tcp 172.16.6.0 0.0.0.255 any eq ftp
access-list 114 permit tcp 172.16.6.0 0.0.0.255 any eq ftp-data
```

port number or protocol name

- Access list number range of 100-199
- Source destination IP address
- Layer 4 protocol number
- Applied to port closest to source host

- The **ip access-group** command links an existing extended ACL to an interface.
- Remember that only one ACL per interface, per direction, per protocol is allowed. The format of the command is:
- Router(config-if) #**ip access-group** *access-list-number* {in | out}

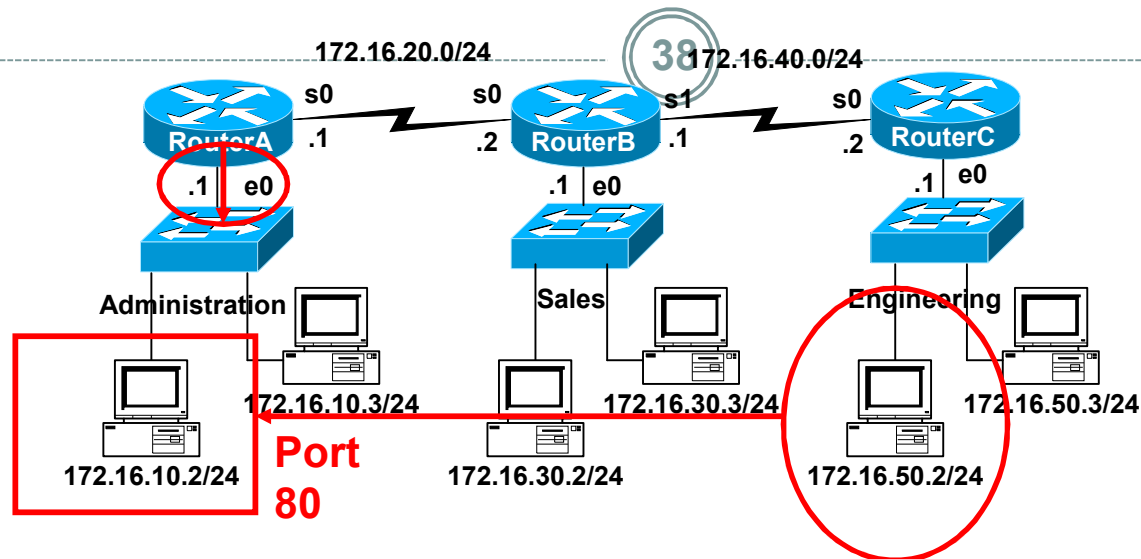
Example 1



Task

- What if we wanted Router A to permit only the Engineering workstation 172.16.50.2 to be able to access the web server in Administrative network with the IP address 172.16.10.2 and port address 80.
- All other traffic is denied.

Example 1



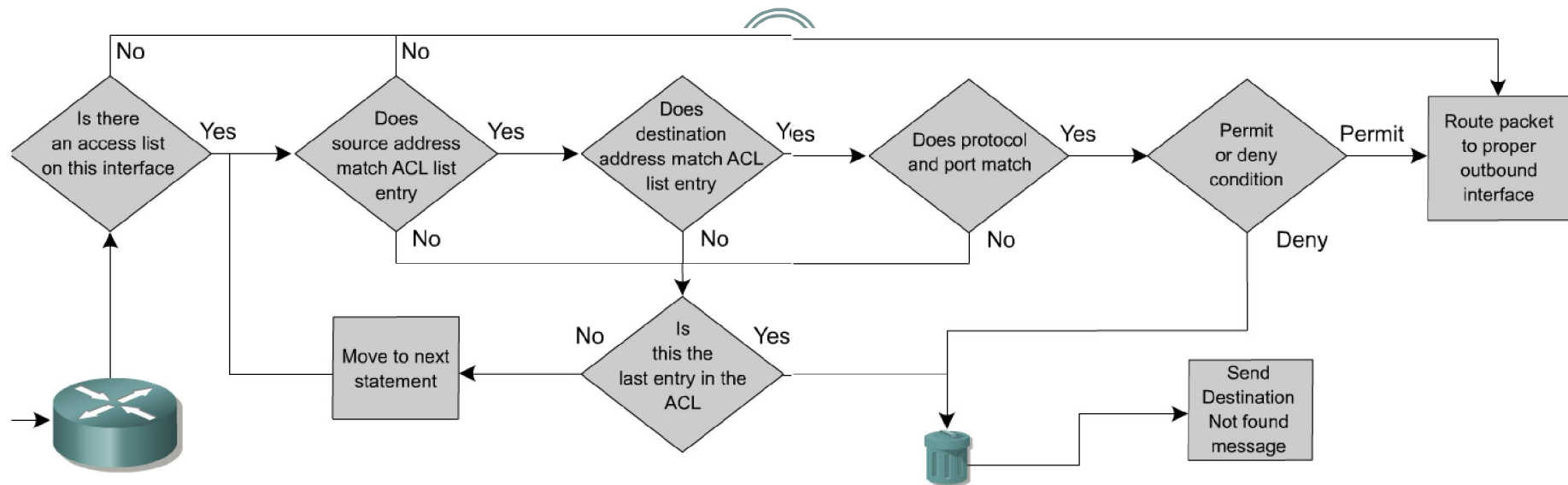
```
RouterA(config) #access-list 110 permit tcp host 172.16.50.2  
host 172.16.10.2 eq 80
```

```
RouterA(config) #inter e 0
```

```
RouterA(config-if) #ip access-group 110 out
```

- Why is better to place the ACL on RouterA instead of RouterC?
- Question: deny http from 172.168.50.2 and permit all

Inbound Extended Access Lists



Inbound Access Lists

```
RouterA(config) # interface e 0
```

```
RouterA(config-if) #ip access-group 110 in
```

- With **inbound** Access Lists the IOS checks the packets **before** it is sent to the Routing Table Process.
- With **outbound** Access Lists, the IOS checks the packets **after** it is sent to the Routing Table Process.
 - This is because the output interface is not known until the forwarding decision is made.

Notes from www.cisco.com

- In the following example, the last entry is sufficient.
- You do not need the first three entries because TCP includes Telnet, and IP includes TCP, User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP).

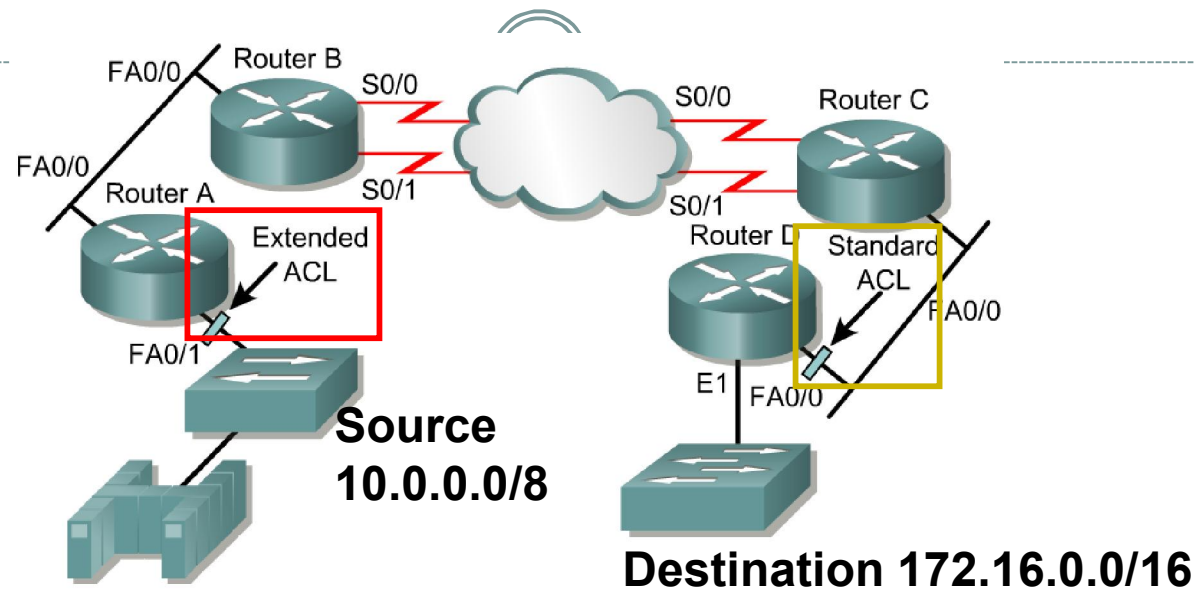
```
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq  
telnet
```

```
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1
```

```
access-list 101 permit udp host 10.1.1.2 host 172.16.1.1
```

```
access-list 101 permit ip 10.1.1.0 0.0.0.255 172.16.1.0  
0.0.0.255
```

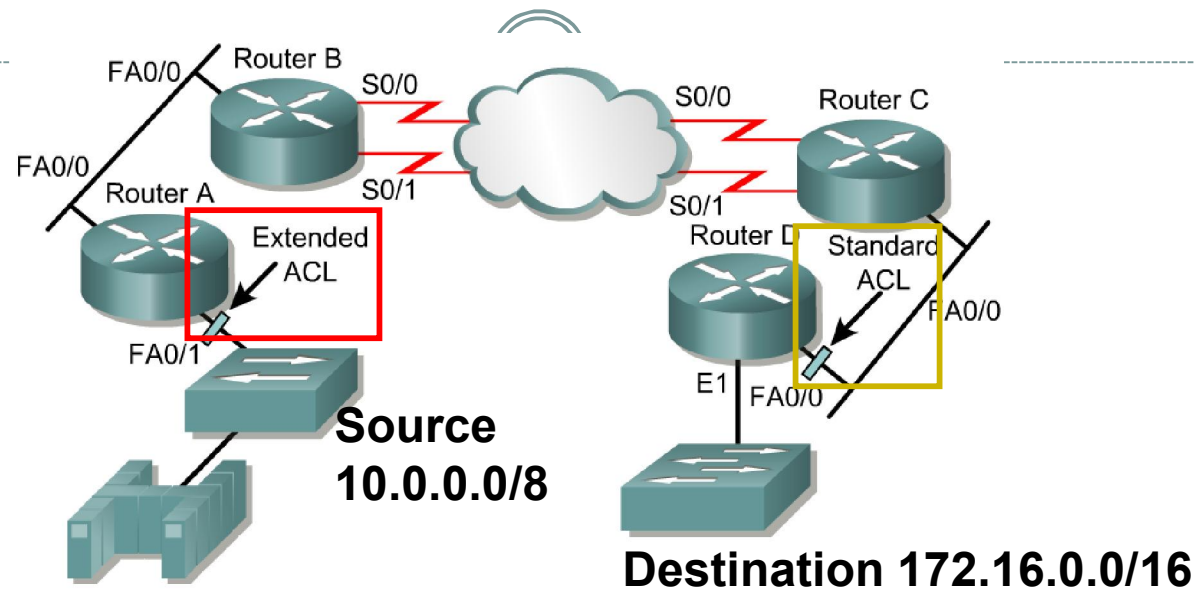

Placing ACLs



The general rule:

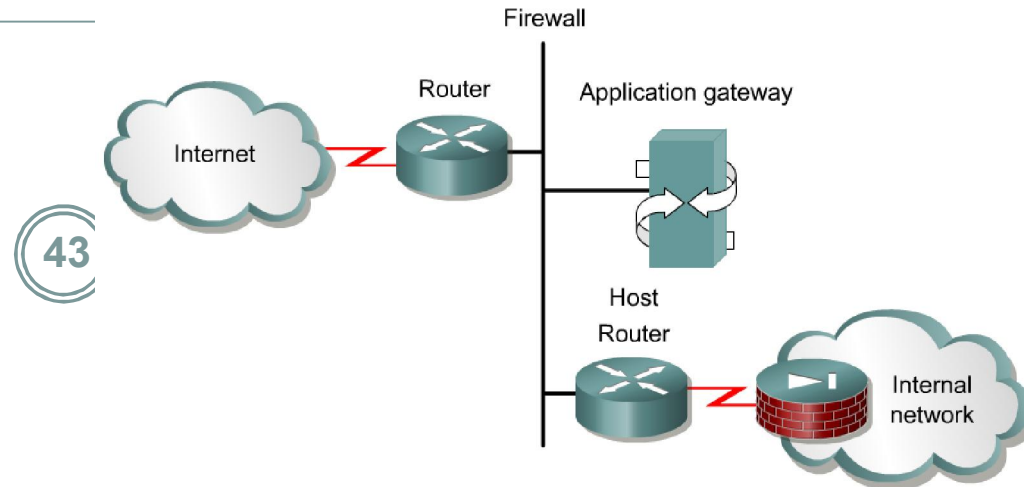
- **Standard ACLs** do not specify destination addresses, so they should be placed as close to the destination as possible.
- Put the **extended ACLs** as close as possible to the source of the traffic denied.

Placing ACLs

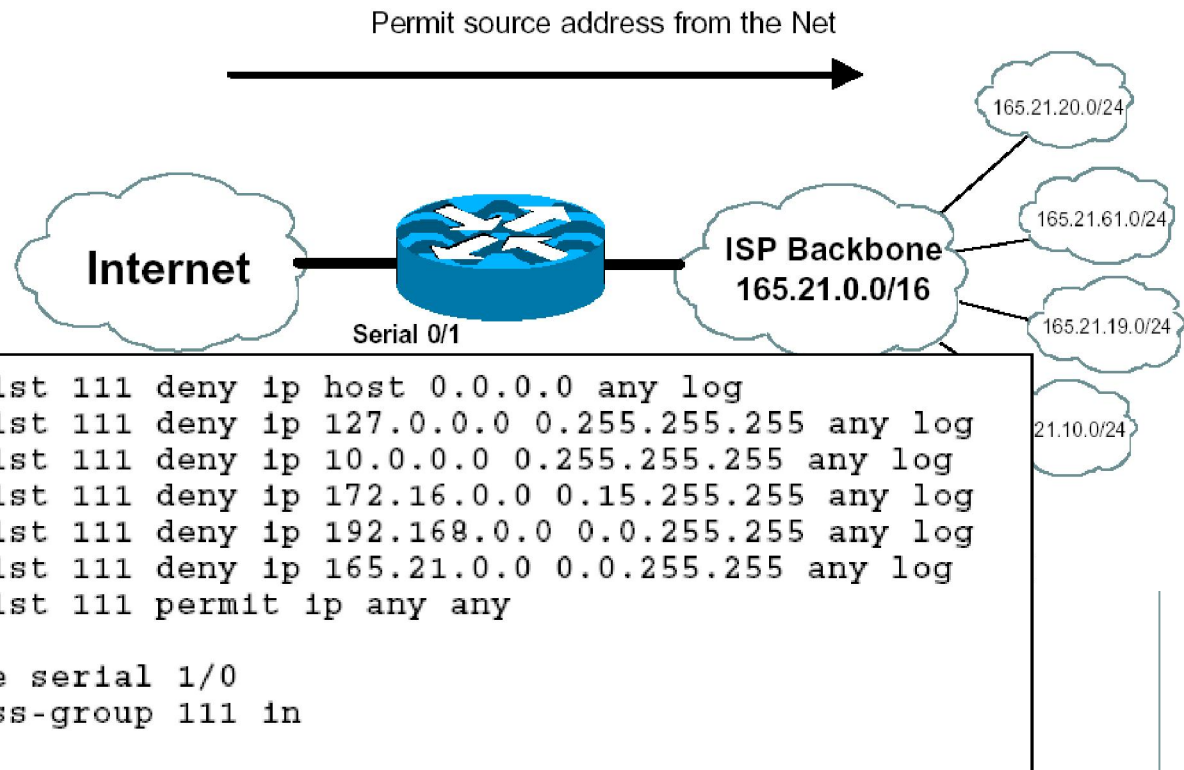


- If the ACLs are placed in the proper location, not only can traffic be filtered, but it can make the whole network more efficient.
- If traffic is going to be filtered, the ACL should be placed where it has the greatest impact on increasing efficiency.

Firewalls



- A firewall is an architectural structure that exists between the user and the outside world to protect the internal network from intruders.
- In most circumstances, intruders come from the global Internet and the thousands of remote networks that it interconnects.
- Typically, a network firewall consists of several different machines that work together to prevent unwanted and illegal access.
- ACLs should be used in firewall routers, which are often positioned between the internal network and an external network, such as the Internet.
- The firewall router provides a point of isolation so that the rest of the internal network structure is not affected.
- ACLs can be used on a router positioned between the two parts of the network to control traffic entering or exiting a specific part of the internal network.



- ISPs use ACLs to deny RFC 1918 addresses into their networks as these are non-routable Internet addresses.
- IP packets coming into your network should never have a source addresses that belong to your network. (This should be applied on all network entrance routers.)
- There are several other simple access lists which should be added to network entrance routers.