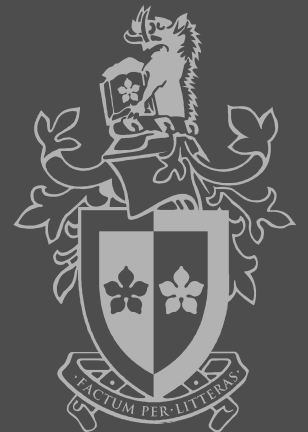


TNE20002/TNE70003

# Access Control Lists





## 3.1 Access Control Concepts

- Purpose Access Control Lists
- Packet Filtering

## 3.2 Types of Access Control Lists

- Standard ACL
- Extended ACL

## 3.3 Wildcard Mask

- Purpose
- Examples

## 3.4 Designing an ACL

- Review business rules
- ACL placement

## 3.5 Creating numbered standard ACL

- Steps to create numbered standard ACL
- Examples 1 & 2

## 3.6 Creating numbered extended ACL

- Steps to create numbered extended ACL
- Examples 1 – 3
- Named ACL



Controlling

access

to a

Resource

# Access Control Lists purpose

---



## What is an ACL

### Basic means of Access Control

An ACL is a series of IOS commands (ACEs) that control whether a router forwards or drops packets based on information found in the packet header. Note. ACLs are not configured by default on a router.

### ACL's can improve Network performance by:

- Limit types of network traffic to increase network performance.  
e.g. video traffic could be blocked if it's not permitted.
- Provide traffic flow control. ACLs can help verify routing updates are from a known source.
- ACLs provide security for network access, can block a host or network.
- Filter traffic based on traffic type such as Telnet traffic.
- Screen hosts to permit or deny access to network services such as FTP or HTTP.

# ACLs - Packet Filtering

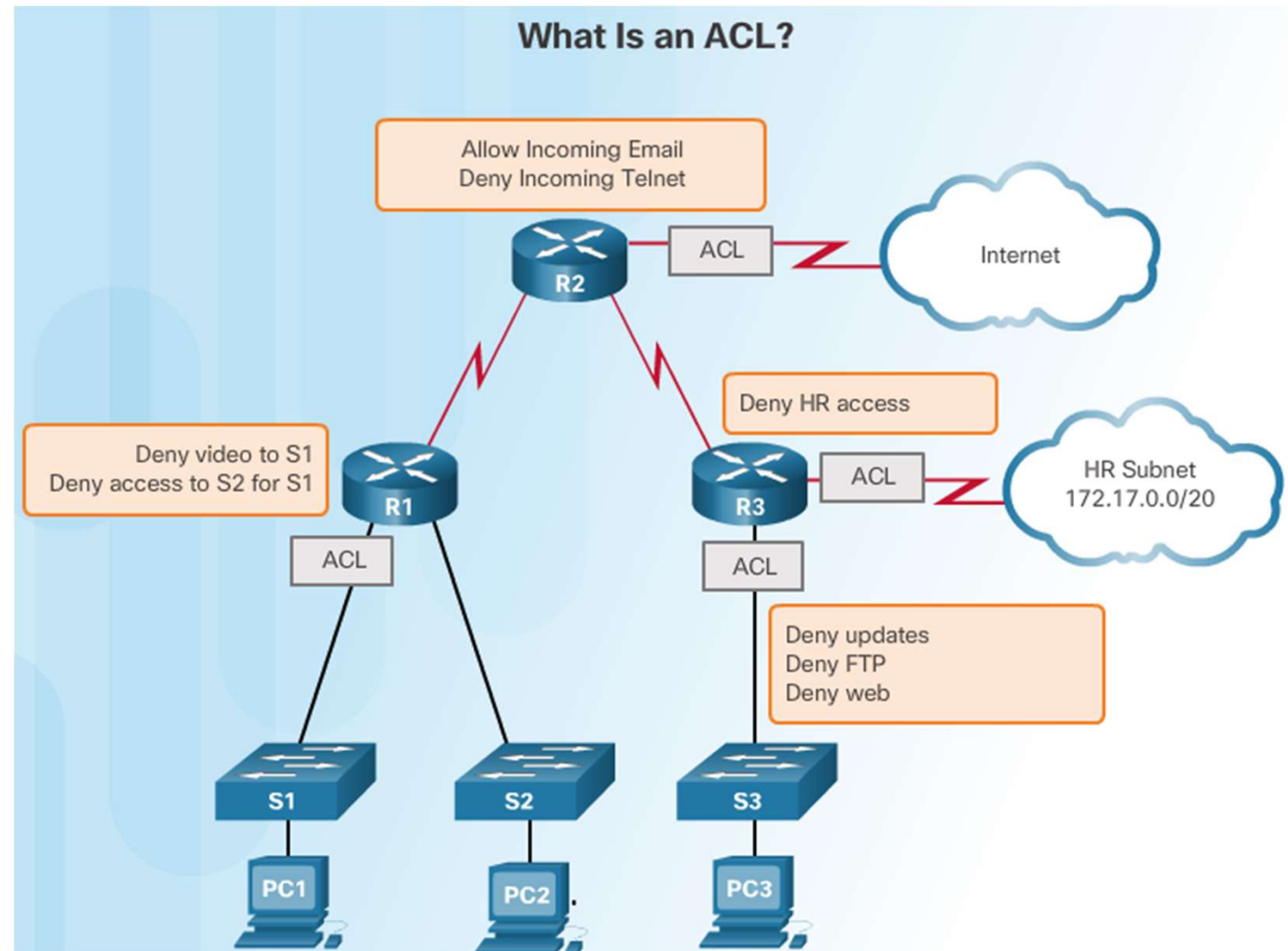


## ACL

Permit or Deny  
packets  
based on set of  
Business rules

**‘Stateless Routing’**  
Each Packet is  
treated independently

Allow Routers to  
filter many packets  
per second





Access Control Lists allow us to **filter** data traffic,  
or **control** the flow of **IP** packets by  
**permit** or **deny** packets

- **into** and **out** of an **internal** subnet or VLAN
- **between** internal subnets
- **between** **internal** subnets and **external** networks
- to a **specific host** address, **internal** or **external**
- to **specific services** (Web, Email, Database, Chat etc) **on a host server**



## ACL Packet Filtering

- ACLs are sequential lists of access control entries (ACEs) that perform permit or deny packet, based on predefined conditional matching statements.
- Packet classification starts at the top ( Rule 1 ) and proceeds down ( Rule 2 etc) until a matching pattern is identified.
- When a match is found, the appropriate action (permit or deny) is taken, and processing stops.

### Note. Implicit Deny ACE

- At the end of every ACL an implicit deny ACE, which denies all packets that did not match earlier in the ACL.

## Filtering each Packet

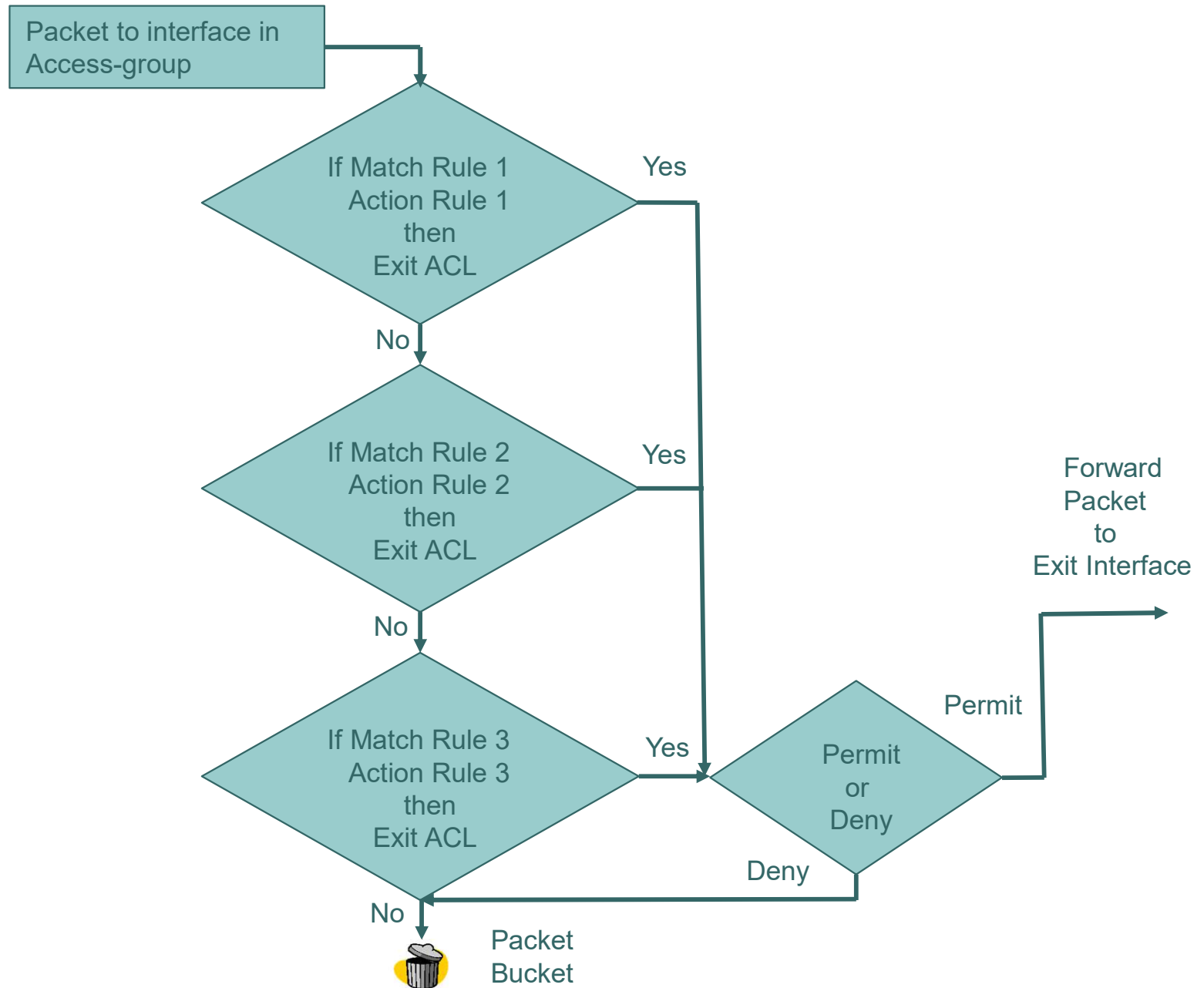
Look at each IP packet passing through an interface / subinterface or line

- Filter against a set of rules
- **Action:**
  - **Permit** the packet to Pass
  - OR
  - **Deny** the packet
- If-then-else structure

# ACLs - Packet Filtering



## ACL Flow Chart



```
ACL Set of Rules - If then else Structure
if (Match on Rule 1) then
    ONLY action Rule 1
    then EXIT (ACL)
else if (Match on Rule 2) then
    ONLY action Rule 2
    then EXIT (ACL)
else if (Match on Rule 3) then
    ONLY action Rule 3
    then EXIT (ACL)
else if (Match on Rule 4) then
    ONLY action Rule 4
    then EXIT (ACL)
else
    Deny/Permit Packet (No Match ?)
endif
```





## ACL Operation

ACLs define the set of rules that give added control for

- packets that enter inbound interfaces,
- packets that relay through the router, and
- packets that exit outbound interfaces of the router.

ACLs can be configured to apply to inbound traffic and outbound traffic.

**Note:** ACLs do not act on packets that originate from the router itself.

- An inbound ACL filters packets before they are routed to the outbound interface. An inbound ACL is efficient because it saves the overhead of routing lookups if the packet is discarded.
- An outbound ACL filters packets after being routed, regardless of the inbound interface.

# ACLs - Packet Filtering – Inbound and Outbound



One ACL per interface / subinterface

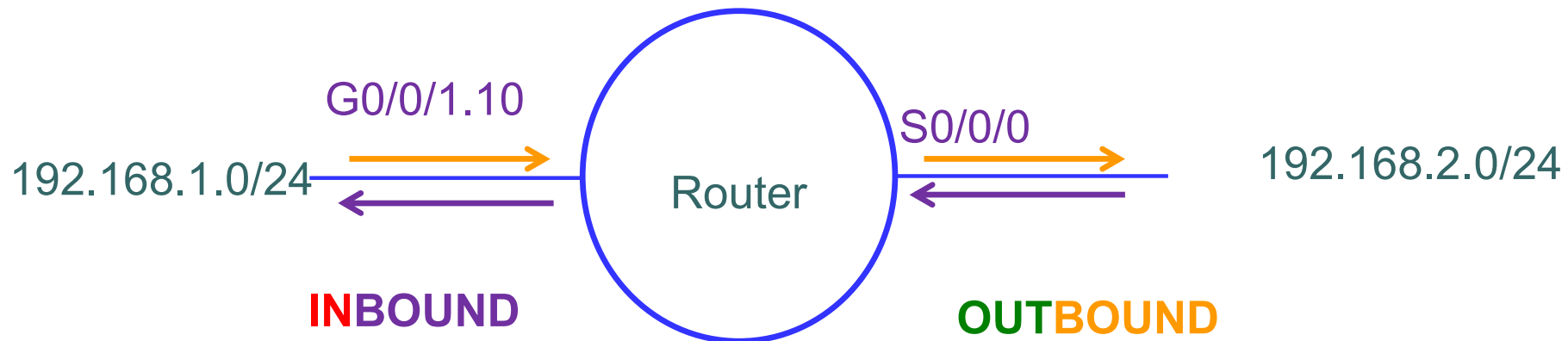
- e.g. S0/0/0, G0/0/1 or G0/0/1.10

One ACL per direction

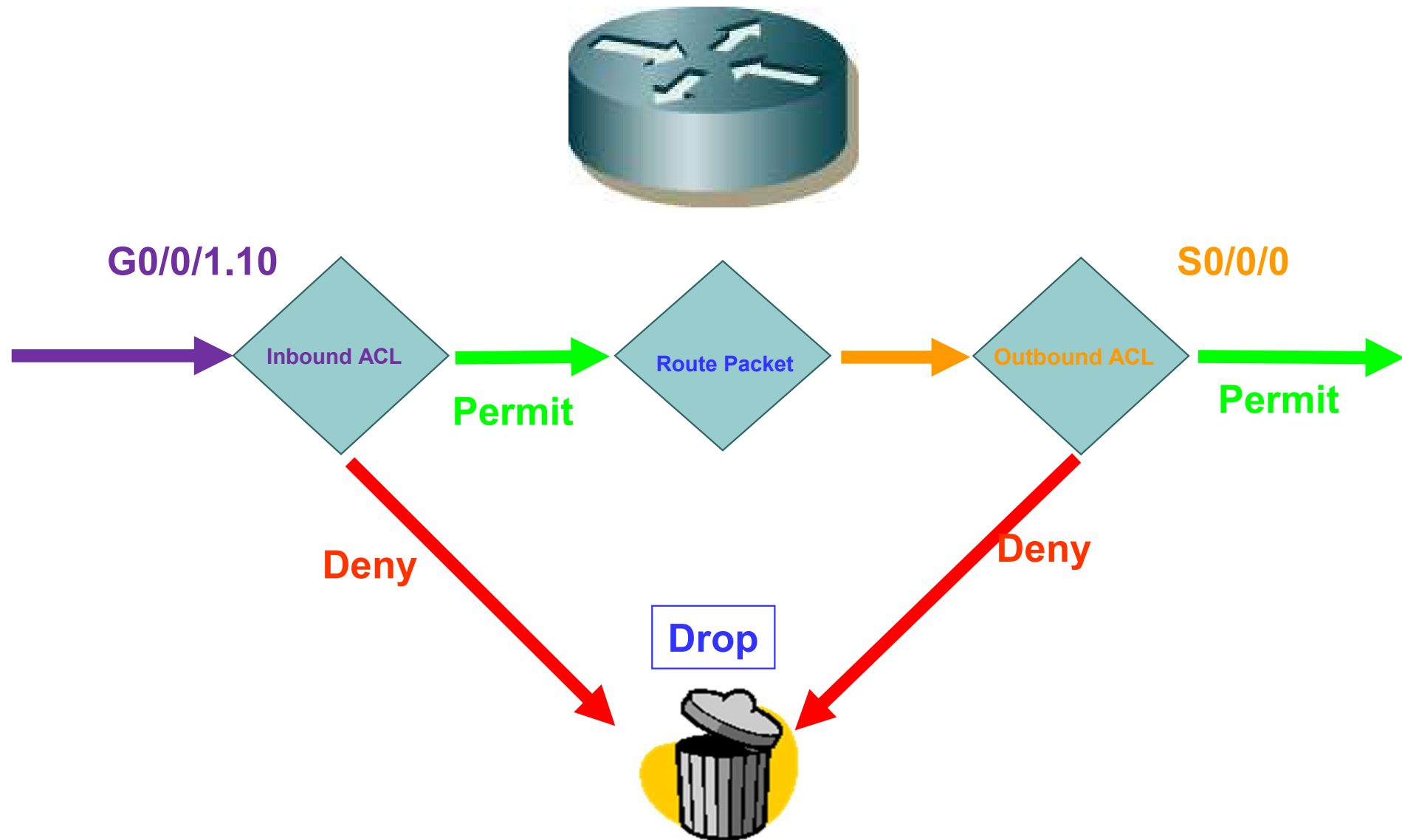
- to filter traffic in each direction:
- inbound to the router and/or outbound from the router

One ACL per protocol

- To control IPv4 and IPv6 traffic



# Packet Filtering – ACLs – Inbound and Outbound



# Packet Filtering – ACLs – Inbound and Outbound



- Packet arrives **inbound** on **G0/0/1** interface
- Packet goes through **inbound ACL** on that interface
  - Check access rules:
    - Action = **DENY** – Drop packet
    - Action = **PERMIT** – Allow packet, stop processing ACL rules,
    - **pass packet to Router**
- **In the Router:** Routes **packet** according **match** in routing table
- Packet routes through **outbound ACL** on router determined **outbound** interface **S0/0/0**
  - Check access rules:
    - Action = **DENY** – Drop packet
    - Action = **PERMIT** – Stop processing ACL rules,
    - **allow packet to exit router**



## 3.2 Types of Access Control Lists

- Standard ACL
- Extended ACL

# Types of IPv4 ACLs



## Types of IPv4 ACLs:

### Standard ACL

filter packets based on:

- Source IP addresses ( only)

```
access-list 10 permit 192.168.30.0 0.0.0.255
```

### Extended ACL

filter packets based on:

- Protocol type / Protocol number (e.g., IP, ICMP, UDP, TCP, ...)
- Source and destination IP addresses
- Source and Destination TCP and UDP ports

```
access-list 103 permit tcp 192.168.30.0 0.0.0.255 any eq 80
```

- Two ways of referencing ACLs:

- **Numbered ACLs**

Assign a number based on protocol  
to be filtered

Standard ACL (1 to 99) , ( 1300 to 1999)

Extended ACL (100 to 199) , ( 2000 to 2699)

- **Named ACLs**

Assign a name to identify the ACL

Names can contain ALPHANumeric characters

Suggest names written in CAPITAL LETTERS

Avoid spaces and no punctuation



## Types of IPv4 ACLs:

- **Standard ACL**
  - should be located as close to the destination as possible
  - If a standard ACL was placed at the source of the traffic, it would filter traffic based on the given source address no matter where the traffic is destined.
- **Extended ACL**
  - should be located as close as possible to the source of the traffic to be filtered.
  - Denies undesirable traffic close to the source network without crossing the network infrastructure.

## Two ways of referencing ACLs:

- **Numbered ACLs**

Assign a number based on protocol  
to be filtered

Standard ACL (1 to 99) , ( 1300 to 1999)

Extended ACL (100 to 199) , ( 2000 to 2699)

- **Named ACLs**

Assign a name to identify the ACL

Names can contain **ALPHANumeric** characters

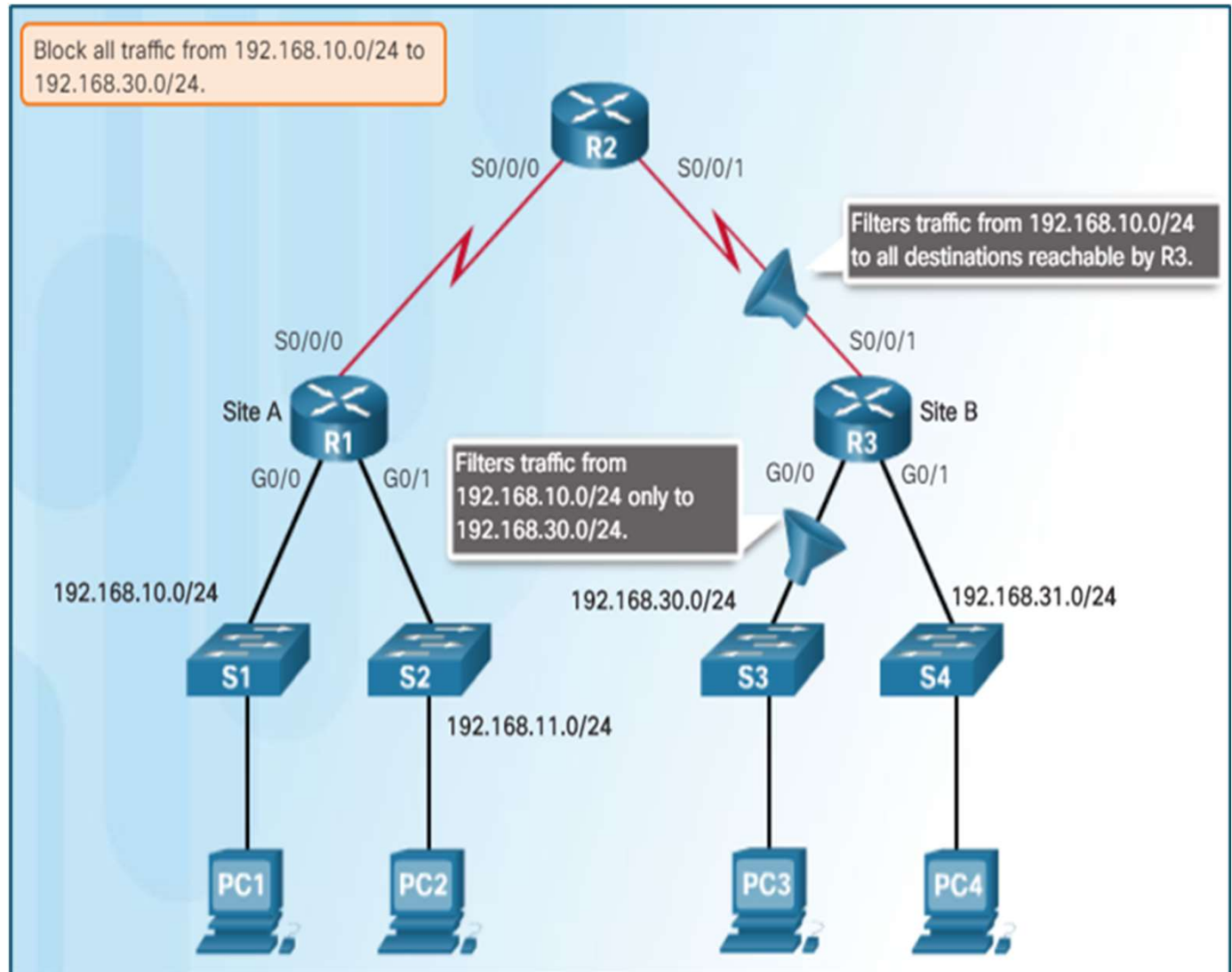
Suggest names written in CAPITAL LETTERS

Avoid spaces and no punctuation

## Standard ACL

- A standard ACL will be configured to block all traffic from 192.168.10.0/24 going to 192.168.30.0/24.
- The standard ACL should be applied closest to the destination and therefore could be applied outgoing on the R3 G0/0 interface.

Applying it incoming on the R3 S0/0/1 interface would prevent reaching 192.168.31.0/24 and therefore should not be applied to this interface.

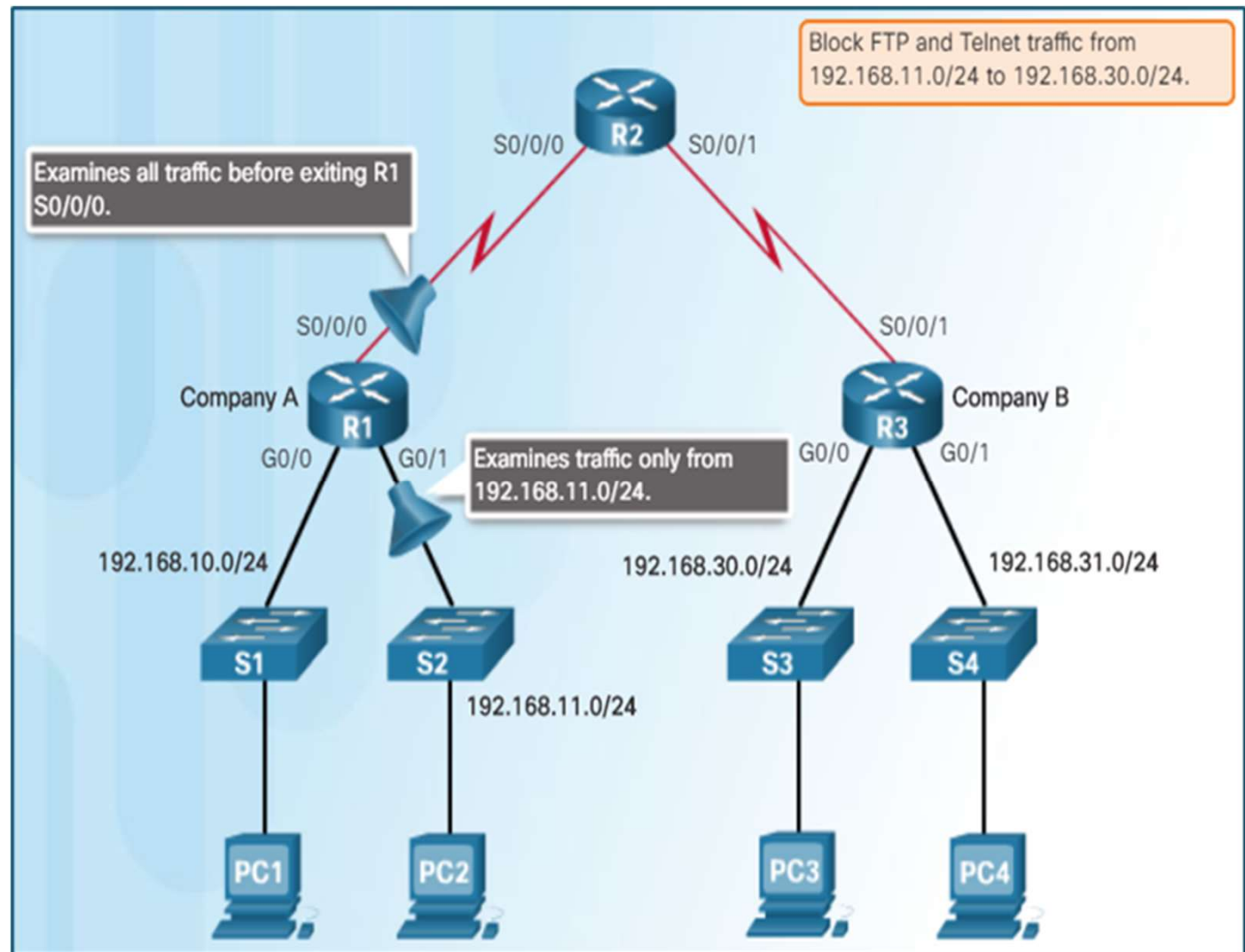




## Extended ACL

- An extended ACL will be configured to block all FTP and Telnet traffic from 192.168.11.0/24 going to 192.168.30.0/24.
- The extended ACL should be applied closest to the source and therefore could be applied incoming on the R1 G0/1 interface.

Applying it outgoing on the R1 S0/0/1 interface would prevent reaching 192.168.31.0/24 but would also needlessly process packets from 192.168.10.0/24





## 3.3 Wildcard Mask

- Purpose
- Examples

# Wildcard Masks –



## Standard ACL Statement

The full syntax of the standard ACL command is as follows.

**access-list** *ACL-#* {**deny** | **permit** | **remark**} *source* [*source-wildcard*] [**log**]

Example1      →      Source address    Wildcard Mask

**access-list 1 permit** 192.168.90.36 0.0.0.0

## Extended ACL Statement

The full syntax of the extended ACL command is as follows

**access-list** *ACL-#* {**deny** | **permit** | **remark**} *protocol* {*source source-wildcard*]  
[*operator* [*port-number* | *port-name*]]  
{*destination destination-wildcard*] [*operator* [*port-number* | *port-name*]]

Example2      →      Source address    Wildcard Mask

**access-list 125 permit ip** 192.168.90.36 0.0.0.0 192.175.63.12 0.0.0.0

Destination address    Wildcard Mask

# Wildcard Masks –



## ACL Wildcard Mask

Each statement in an ACL is called an Access Class Entry (ACE)

IPv4 ACEs require the use of **wildcard mask**

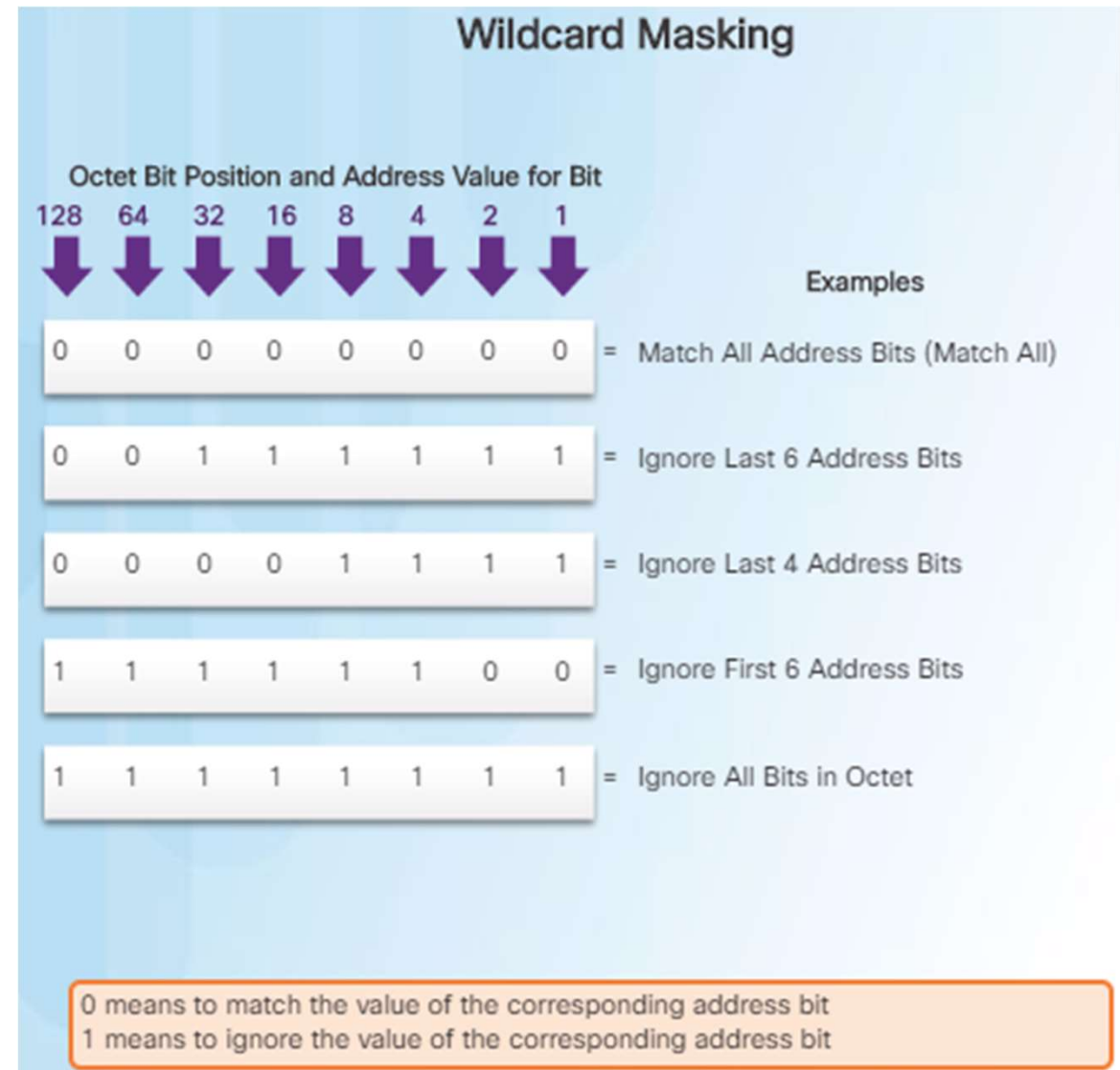
A **wildcard mask** is a string of 32 binary digits (1s and 0s) used by the router to determine which bits of the address to examine for a match.

ACLs use IPv4 address+wildcard mask combined in a **Rule** to generate a range of IP addresses to match against the source address of the packet.

If the source IPv4 address of the packet does match the **Rule** then the specified Action **Permit** or **Deny** is done.

**Wildcard masks** are often referred to as an inverse mask since unlike a **subnet mask**

**subnet mask** where a binary 1 is a match, whereas **Wildcard masks** a binary 0 is a match .



# Wildcard Masks –



## ACL Wildcard Mask examples

**Example 1:** The wildcard mask stipulates that every bit in the IPv4 192.168.1.1 address must match exactly.

**Example 2:** The wildcard mask stipulates that anything will match.

**Example 3:** The wildcard mask stipulates that any host within the 192.168.1.0/24 network will match.

### Wildcard Masks to Match IPv4 Hosts and Subnets

#### Example 1

	Decimal	Binary
IP Address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	0.0.0.0	00000000.00000000.00000000.00000000
Result	192.168.1.1	11000000.10101000.00000001.00000001

#### Example 2

	Decimal	Binary
IP Address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	255.255.255.255	11111111.11111111.11111111.11111111
Result	0.0.0.0	00000000.00000000.00000000.00000000

#### Example 3

	Decimal	Binary
IP Address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	0.0.0.255	00000000.00000000.00000000.11111111
Result	192.168.1.0	11000000.10101000.00000001.00000000

# Wildcard Masks –



## ACL Wildcard Mask examples

Wildcard Mask (Decimal Notation)	Last Octet (in Binary)	Meaning (0 - match, 1 - ignore)
<b>0.0.0.0</b>	<b>00000000</b>	Match all octets.
<b>0.0.0.63</b>	<b>00111111</b>	<ul style="list-style-type: none"><li>• Match the first three octets</li><li>• Match the two left most bits of the last octet</li><li>• Ignore the last 6 bits</li></ul>
<b>0.0.0.15</b>	<b>00001111</b>	<ul style="list-style-type: none"><li>• Match the first three octets</li><li>• Match the four left most bits of the last octet</li><li>• Ignore the last 4 bits of the last octet</li></ul>
<b>0.0.0.248</b>	<b>11111100</b>	<ul style="list-style-type: none"><li>• Match the first three octets</li><li>• Ignore the six left most bits of the last octet</li><li>• Match the last two bits</li></ul>
<b>0.0.0.255</b>	<b>11111111</b>	<ul style="list-style-type: none"><li>• Match the first three octet</li><li>• Ignore the last octet</li></ul>

# Wildcard Masks –



## ACL Wildcard Mask keywords

- To make wildcard masks easier to read, the keywords **host** and **any** can help identify the most common uses of wildcard masking.
  - **host** substitutes for the 0.0.0.0 mask
  - **any** substitutes for the 255.255.255.255 mask

### Example 1.

- If you would like to match the 192.169.10.10 address,

you could use **192.168.10.10 0.0.0.0** or, you can use: **host 192.168.10.10**

**Example 2.** instead of entering

**Permit 0.0.0.0 255.255.255.255**, you can use the keyword **any** by itself.

**Permit any**

## Wildcard Bit Mask Abbreviations

### Example 1

- 192.168.10.10 0.0.0.0 matches all of the address bits
- Abbreviate this wildcard mask using the IP address preceded by the keyword **host** (**host 192.168.10.10**)



### Example 2

- 0.0.0.0 255.255.255.255 ignores all address bits
- Abbreviate expression with the keyword **any**



# Wildcard Masks –



- Wildcard Mask stipulates any hosts in the 160.56.27.0 /24 network will match
- Deny all hosts in subnet 160.56.27.0 255.255.255.0
- deny 160.56.27.0 0.0.0.255 (matching rule)

Source IP Address:

10100000	00111000	00011011	00000000
----------	----------	----------	----------

Wildcard Mask:

00000000	00000000	00000000	11111111
----------	----------	----------	----------

Matching IP:

10100000	00111000	00011011	????????
----------	----------	----------	----------

Match ?:      Yes                  Yes                  Yes      /24      NO

- The Wildcard mask is often the inverse of the Subnet mask
- 0 bit in Wildcard - LOOK for an EXACT Match
- 1 bit in Wildcard – Do NOT LOOK for a Match





Wildcard mask = 32 bits

## 0 bit in Wildcard - LOOK for an EXACT Match

- this bit in the packet's source IP address
- **must match** the corresponding bit specified in **matching rule source IP address exactly**

## 1 bit in Wildcard – Do NOT LOOK for a Match

- this bit in the packet's source IP address
  - **will be ignored**
- 
- If packet source IP matches, the rule IP address and wildcard mask, then the specified action is taken



## 3.4 Designing an ACL

- Review Business Rules
- ACL Placement

# Standard Numbered ACLs



## Standard ACL Statement

The full syntax of the standard ACL command is as follows.

```
access-list ACL-# {deny | permit | remark} source [source-wildcard] [log]
```

### Example1

```
access-list 1 permit 192.168.3.0 0.0.0.255
```

The Standard ACL Matching Rule consists of

Action	Source Address	wildcard mask
permit/deny	From Host/Range/Subnet	Bits to Match

## Standard ACLs

- Filter at layer 3
- Only filter source ip address
  - if source ip address matches then deny or permit
  - the source IP address, can be:
    - a Host or Subnet address
    - range of IP addresses within a Subnet

# ACLs – The List - If then else Structure



## List of Client business rules


A client requires you to create a **standard** numbered ACL 9 to filter ip packets from different source addresses to **destination** network **192.168.2.0/24**, the rules are:

- deny host **192.168.3.140**
- deny hosts in range **192.168.3.12 – 192.168.3.15** and
- deny host **192.168.3.7**
- Permit all other ip packets from **192.168.3.0 /24**

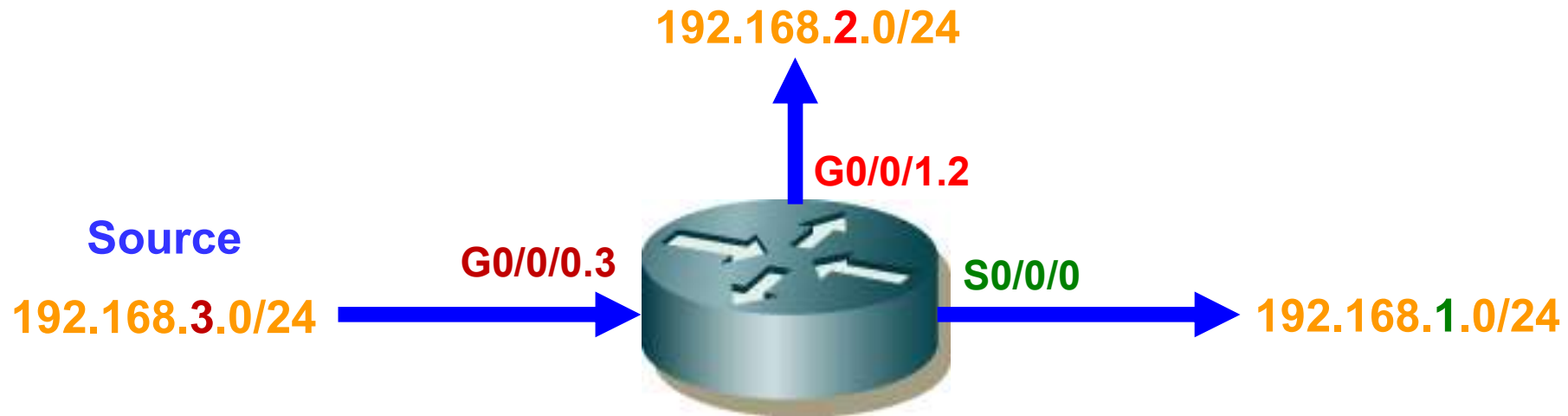
Step1. Filter packets from **source** network **192.168.3.0 /24**

## ACL Rule Set

```
if source Host IP = 192.168.3.7 (Most Specific)
    deny packet (this action taken)
else if source Host IP = 192.168.3.140
    deny packet (this action taken)
else if source is in Range of IPs between 192.168.3.12 and 15
    deny packet (this action taken)
else if source ip = any other (Match any other address) (Least Specific)
    permit packet (this action taken)
endif
```



# ACLs – The List – Check Rule Order



- ? Does the **ordering** of following **ACL** Rule Set **meet the requirements** of the **previous If then Else** structure:

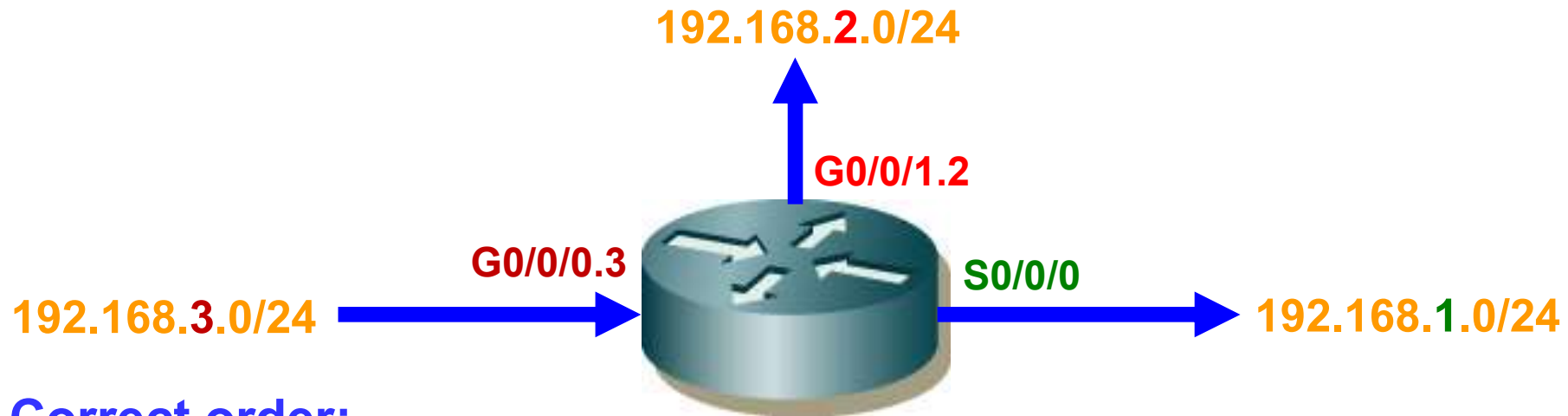
deny 192.168.3.7

**permit any**

deny 192.168.3.(12-15)

deny 192.168.3.140

# ACLs – The List – Correct Order



- **Correct order:**

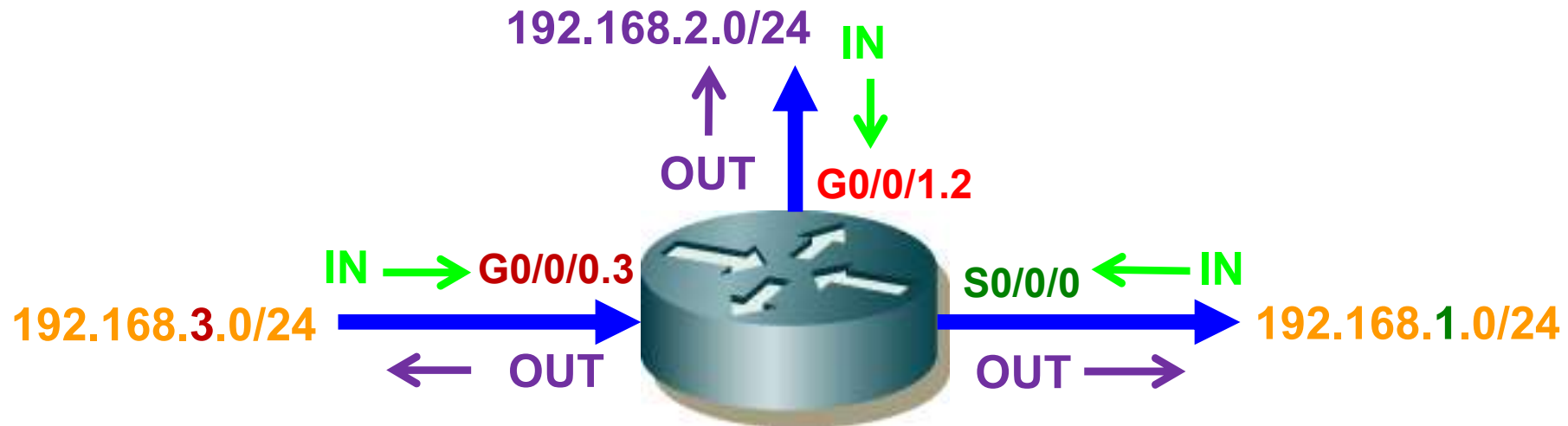
- deny 192.168.3.7
  - deny 192.168.3.140
  - deny 192.168.3.(12-15)
  - permit any**

(Most Specific)  
↓  
(Least Specific)

- ? Given diagram above on what **interface** should the ACL be **placed** and in which **direction** Inbound or Outbound.

What are we trying to achieve ?

# ACLs – The List – Where do we Place ACL ?



## Place on Subinterface G0/0/0.3

- **IN** deny/permit from 192.168.3.0 to **all** destinations
- **OUT** no effect

## Place on Interface S0/0/0

- **IN** no effect
- **OUT** deny/permit from 192.168.3.0 to **specific** destination 192.168.1.0 /24

## Place on Subinterface G0/0/1.2

- **IN** no effect
- **OUT** deny/permit from 192.168.3.0 to **specific** destination 192.168.2.0 /24

# ACL Flowchart



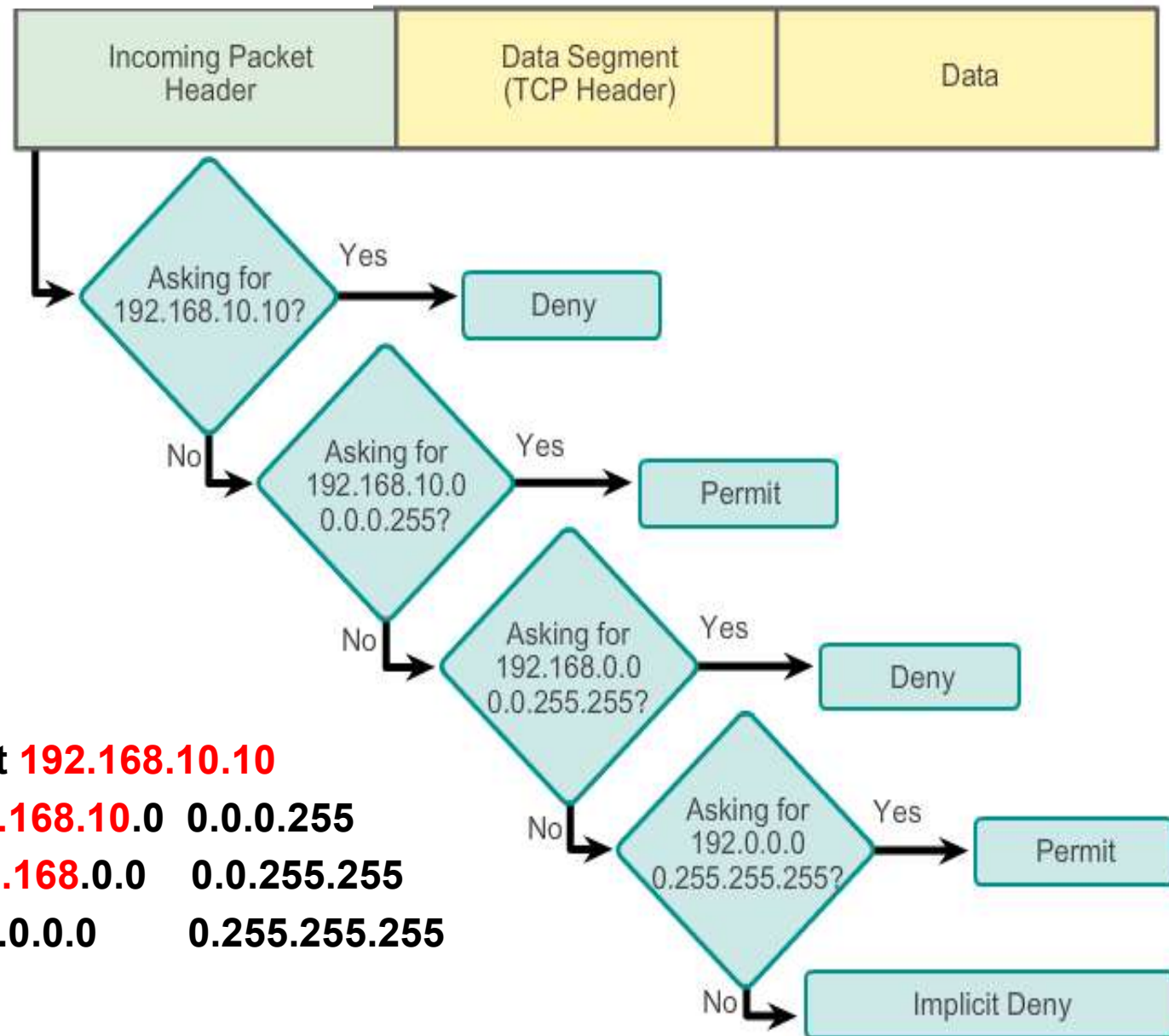
## Interface S0/0/1

**ACL Flowchart**  
Most specific  
to  
Least Specific

## The ACL

int S0/0/1  
ip access-group 2 in

```
access-list 2 deny    host 192.168.10.10
access-list 2 permit 192.168.10.0 0.0.0.255
access-list 2 deny    192.168.0.0 0.0.255.255
access-list 2 permit 192.0.0.0 0.255.255.255
```







## 3.5 Creating numbered standard ACL

- Steps to create numbered standard ACL
- Examples 1 - 2

# Standard ACLs



Standard ACL	filter on:
<ul style="list-style-type: none"><li>Source address</li></ul>	<b>ONLY, Layer 3</b>

# Creating a Standard ACL



## Two Step Process

### Defining a numbered standard ACL

**Step 1.** Define the ACL by using the command `access-list-number { deny | permit }`  
Source [source-wildcard] [log]. The ACL number can be 1–99 or 1300–1999.

**Step 2.** Apply the ACL to an interface by using the command `ip access-group {acl-number} {in|out}` under interface configuration mode.

### Example 1

Example 1 demonstrates how a numbered standard ACL is created and applied to an interface to deny traffic from 172.16.0.0 /16 subnet and from host 192.168.1.1 while allowing all other traffic coming into interface Gi0/1.

### **Example 1** *Creating and Applying a Numbered Standard ACL*

```
R1(config)# access-list 1 deny 172.16.0.0 0.0.255.255
R1(config)# access-list 1 deny host 192.168.1.1
R1(config)# access-list 1 permit any
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group 1 in
```

**Notice** that the last ACE in the ACL explicitly permits all traffic (permit any). If this ACE is not included, all traffic will be dropped because of the implicit deny (deny any) at the end of every ACL.

# Creating a Standard ACL



## List Client business rules

A client requires you to create a numbered **standard** ACL 9 to filter ip packets from different networks

to **destination** network **192.168.2.0/24**, the rules are:

- Deny all ip packets from **192.168.0.0/24** network
- Deny all ip packets from **192.168.1.0/24** network but
- permit hosts in range **192.168.1.32–192.168.1.47** and
- deny host **192.168.1.40**
- Permit all other ip packets

**Logic rules** in a correct order **most** specific to **least** specific:

- deny 192.168.1.40 **(most)**
- permit 192.168.1.(32-47)
- deny 192.168.1.0/24
- deny 192.168.0.0/24
- permit any **(least)**

# Creating a Standard ACL – Use Notepad



Create ACL in Notepad – easy to edit

## Logical Rules

## Notepad

Delete previous ACL

deny 192.168.1.40

permit 192.168.1.(32 - 47)

deny 192.168.1.0/24

deny 192.168.0.0/24

permit all

```
no access-list 9
```

```
access-list 9 deny host 192.168.1.40
```

```
access-list 9 permit 192.168.1.32 0.0.0.15
```

```
access-list 9 deny 192.168.1.0 0.0.0.255
```

```
access-list 9 deny 192.168.0.0 0.0.0.255
```

```
!
```

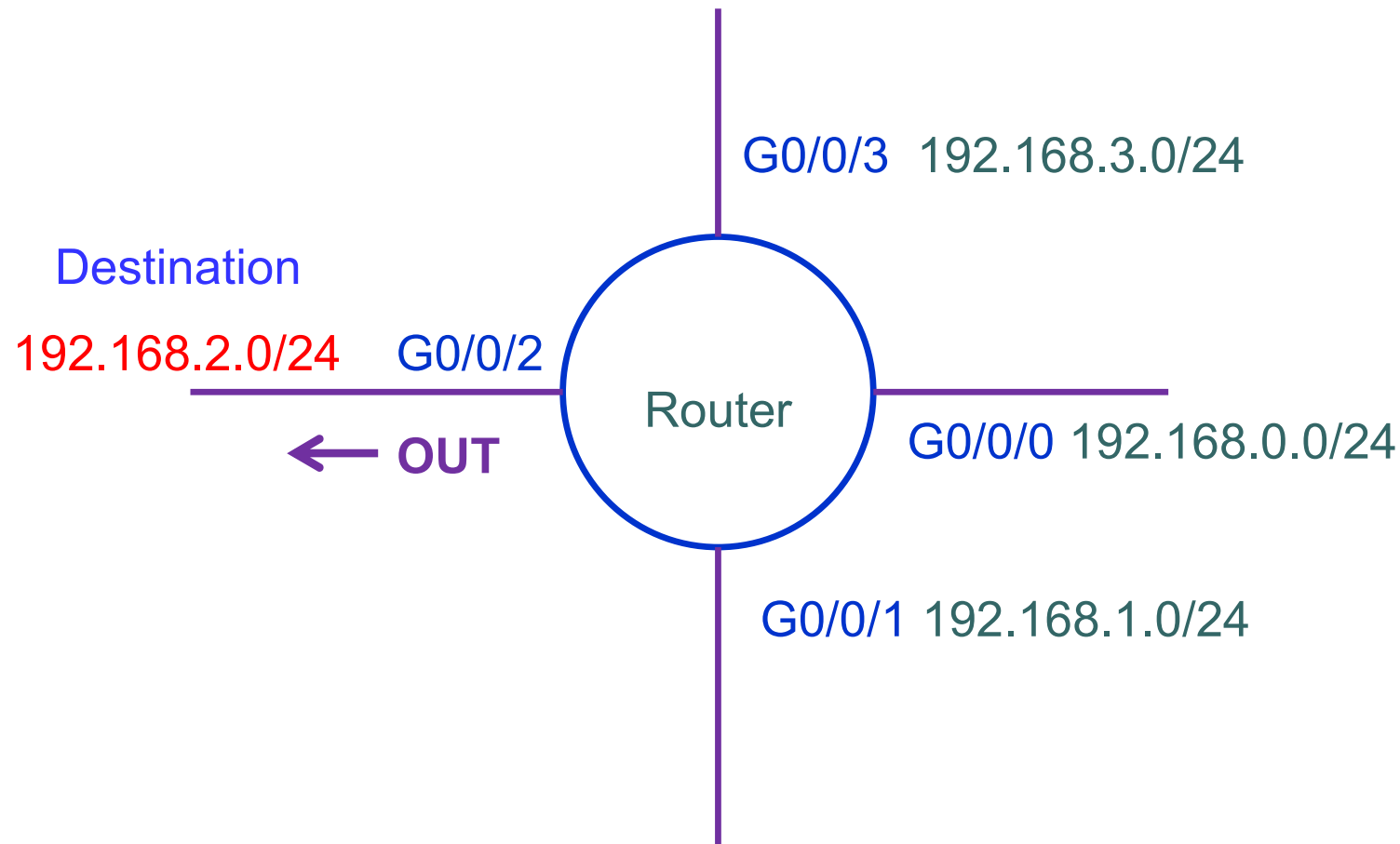
```
access-list 9 permit any
```

```
exit
```

- Paste into router - Must be in configuration mode

**Router(config)#**

# Standard ACL – Placement - Which Interface, IN or Out Bound ?



# Standard ACL – Placement - Which Interface, IN or Out Bound ?



- Place **outbound** on interface **G0/0/2** with IP address 192.168.2.1/24

```
Router(config)# int G0/0/2  
                ip access-group 9 out
```

**(out means outbound)**

- Standard ACLs generally **placed Close to the Destination**



- ACLs can be removed from an interface

```
int G0/0/2  
no ip access-group 9 out
```

- ACLs can be replaced with a different ACL

```
int G0/0/2  
ip access-group 27 out
```

- ACLs can be deleted from memory

```
no access-list 9
```

**show ip interface G 0/0/2 command**  
to verify the ACL is applied to the interface



# Verify - Standard ACL



## Commands to verify Standard ACL

Use **show access-lists** [*ACL-#* | *access-list-name*] command to view the content of a standard ACL.

Notice that some statements are out of order because Cisco IOS uses a special hashing function for standard ACLs and re-orders host ACEs so they are processed first optimizing the search for a host ACL entry.

## ACL Statistics

Use **clear access-list counters** command to clear the counter matches from previous tests .

```
R1# show access-lists
Standard IP access list NO-ACCESS
    10 deny    192.168.10.10  (20 matches)
    20 permit 192.168.10.0, wildcard bits 0.0.0.255  (64 matches)
R1# clear access-list counters NO-ACCESS
R1# show access-lists
Standard IP access list NO-ACCESS
    10 deny    192.168.10.10
    20 permit 192.168.10.0, wildcard bits 0.0.0.255
R1#
```

Use **show ip interface** command to verify the ACL on the interface.

The output includes the number or name of the access list and the direction in which the ACL was applied.



## 3.6 Creating numbered extended ACL

- Steps to create numbered extended ACL
- Examples 1 - 3
- Named ACL



## Extended ACLs can filter on:

- Source address
  - Destination address
  - Protocol
  - Port numbers
- Layer 3**
- Layer 4**

# Creating an Extended ACL



## Two Step Process

Defining a numbered extended ACL

**Step 1. Define ACL** `access-list ACL-# {deny | permit | remark} protocol { source source-wildcard} [operator [port-number | port-name]] { destination destination-wildcard} [operator [port-number | port-name]]`

The ACL number can be 100–199 or 2000–2699.

**Step 2.** Apply the ACL to an interface by using the command `ip access-group {acl-number} {in|out}` under interface configuration mode.

### Example 1

Example 1 demonstrates how a numbered extended ACL is created and applied to an interface to block all telnet & ICMP traffic and deny all IP traffic from Source host 10.1.2.2 /32 To destination host 10.1.2.1 /32 while allowing all other traffic.

Notice how Telnet's TCP port 23 is being matched with the `eq` keyword

Notice that the last ACE in the ACL explicitly permits all traffic (`permit any any`).

If this ACE is not included, all traffic will be dropped because of the implicit deny (`deny any`) at the end of every ACL.

### **Example 1** *Creating and Applying Numbered Extended ACLs*

```
R1(config)# access-list 100 deny tcp any any eq 23
R1(config)# access-list 100 deny icmp any any
R1(config)# access-list 100 deny ip host 10.1.2.2 host 10.1.2.1
R1(config)# access-list 100 permit ip any any
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group 100 in
```

# ACL Decision Path –



## Example2. ACL filters Web traffic

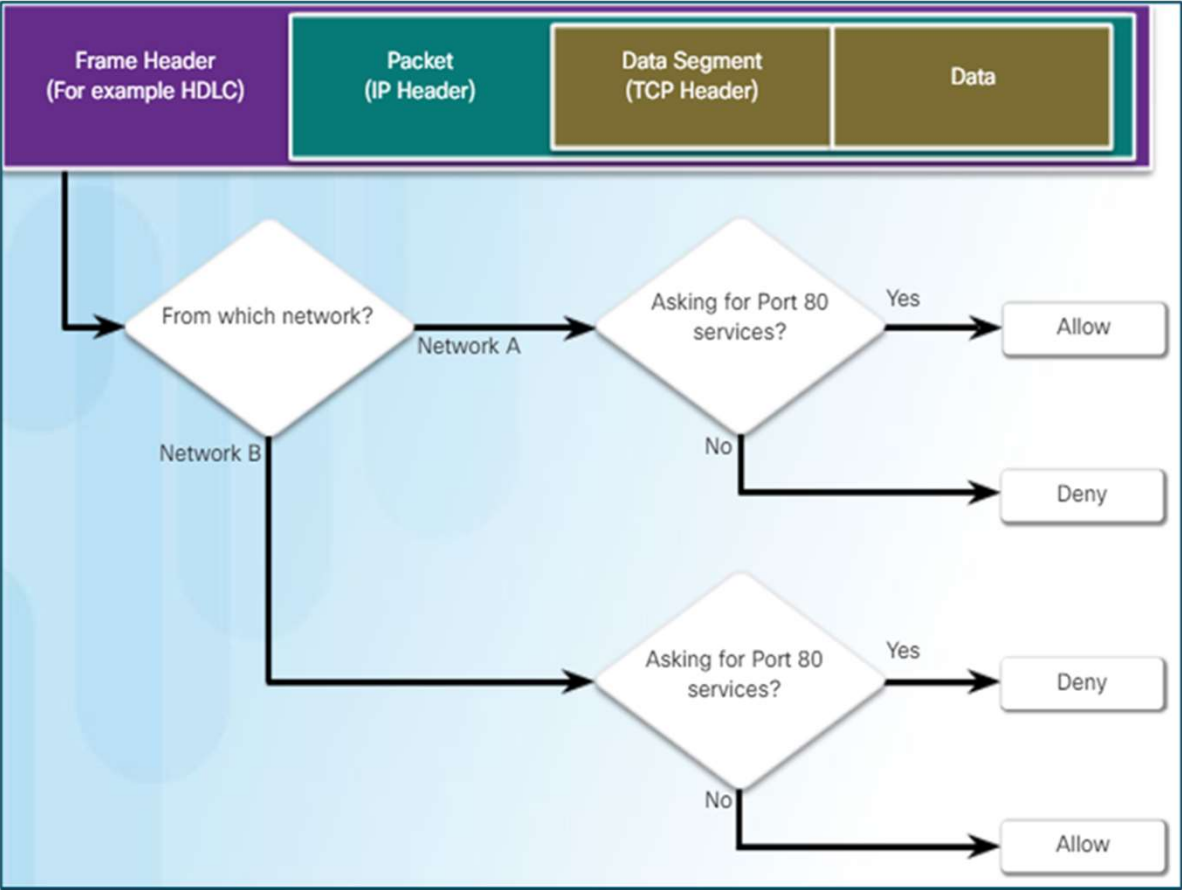
To understand how an ACL operates, refer to the decision path used to filter web traffic

### Example2 :

An ACL has been configured to:

Permit web access to users from Network A  
but deny all other services to Network A users.

Deny HTTP access to users from Network B,  
but permit network B users to have all other access.





- The logic and implementation is similar Standard ACLs
- with
- **More control**: Matching rules have **more detail**
- **What can we match against:**
  - **Layer 3 IP Addresses**
    - **Source** IP address
    - **Destination** IP address
  - **Layer 3 and 4 Protocols**
    - ip, tcp, udp, icmp etc
  - **Layer 4 Applications**
    - **Source** Port
    - **Destination** Port

# Extended ACL – Format



deny|permit *protocol* *source ip* *source wildcard*  
[*operator* operand] *destination ip* *destination wildcard*  
[*operator* operand]

## *protocol*

- *ip* – Matches *all protocols* (includes IP,TCP, UDP, ICMP etc.)
- *icmp* – Matches ICMP protocol
- *tcp* – Matches TCP protocol
- *udp* – Matches UDP protocol

## *Source ip* *source wildcard*

- Matches packet source IP Address

## *Destination ip* *destination wildcard*

- Matches packet destination IP Address



## operator

- **eq, neq** – Port number equal or not equal to specified **operand**
- **lt, gt** – Port number less or greater than specified **operand**

## and

## operand

- Integer port number eg **80, 20, 23**
- Text representation of service name eg. **http, ftp, telnet**





- Extended ACLs can filter traffic by examining port numbers
- Common TCP UDP ports include:

Port Number	Protocol	Application	Acronym
20	TCP	File Transfer Protocol (data)	FTP
21	TCP	File Transfer Protocol (control)	FTP
22	TCP	Secure Shell	SSH
23	TCP	Telnet	–
25	TCP	Simple Mail Transfer Protocol	SMTP
53	UDP, TCP	Domain Name Service	DNS
67	UDP	Dynamic Host Configuration Protocol (server)	DHCP
68	UDP	Dynamic Host Configuration Protocol (client)	DHCP
69	UDP	Trivial File Transfer Protocol	TFTP
80	TCP	Hypertext Transfer Protocol	HTTP
110	TCP	Post Office Protocol version 3	POP3
143	TCP	Internet Message Access Protocol	IMAP
161	UDP	Simple Network Management Protocol	SNMP
443	TCP	Hypertext Transfer Protocol Secure	HTTPS



## Numbered Extended ACL

```
access-list 127 permit tcp any host 136.186.1.10 eq www
```

Placing ACL on subinterface:

```
int G0/0/0  
no shutdown
```

```
int G0/0/0.127  
ip access-group 127 in
```



## Example 3.

### Client Requirements

- Create an ACL to meet following client requirements:
  1. Deny all PCs in 192.168.0.0/24 **FTP** access to host 50.50.50.10  
except for PC 192.168.0.50
  2. All PCs in 192.168.0.0/24 should not have **TELNET** access to network 200.200.200.0/24
  3. All PCs in 192.168.0.0/24 should have access to the Internet  
except for PCs 192.168.0.192 -199

- | Requirement 1       | Destination    | Source              |
|---------------------|----------------|---------------------|
| • Permit <b>FTP</b> | to 50.50.50.10 | from 192.168.0.50   |
| • Deny <b>FTP</b>   | to 50.50.50.10 | from 192.168.0.0/24 |
- | Requirement 2 | Destination                | Source              |
|---------------|----------------------------|---------------------|
| • Deny TELNET | access to 200.200.200.0/24 | from 192.168.0.0/24 |
- | Requirement 3     | Destination       | Source                 |
|-------------------|-------------------|------------------------|
| • Deny ip (all)   | to internet (any) | from 192.168.0.192-199 |
| • Permit ip (all) | to internet (any) | from 192.168.0.0/24    |

# Editing the ACL – Use Notepad



**no access-list 101** (deletes previous ACL)

**Rem** ! Permit ftp to 50.50.50.10 from 192.168.0.50

! Deny ftp to 50.50.50.10 from 192.168.0.0/24

**access-list 101** permit tcp host 192.168.0.50 host 50.50.50.10 eq ftp

**access-list 101** deny tcp 192.168.0.0 0.0.0.255 host 50.50.50.10 eq ftp

**Rem** ! Deny telnet access to 200.200.200.0/24 from 192.168.0.0/24

**access-list 101** deny tcp 192.168.0.0 0.0.0.255 200.200.200.0  
0.0.0.255 eq telnet

**Rem** ! Deny ip (all) to internet (any) from 192.168.0.192-199

**access-list 101** deny ip 192.168.0.192 0.0.0.7 any

**Rem** ! Permit ip other PCs from 192.168.0.0/24 access to internet,  
is covered by the last rule

**Rem** ! This should be the last rule

**access-list 101** permit ip any any



## Numbered IPv4 ACLs

- Standard (1-99) (1300-1999)
- Extended (100-199) (2000-2699)

## Named ACLs

- Cisco IOS allows naming ACLs with text-based names
- Difficult to remember purpose of Numbered ACL,  
Named ACL is better for documentation

# Creating a Named ACL



## Three Step Process

### Defining a Named ACL

Step 1. Define the ACL by using the command `ip access-list standard|extended {acl-number | acl-name}`. Entering this command places the CLI in ACL configuration mode.

Step 2. Configure the specific ACE in ACL configuration mode by using the command `[sequence] {permit | deny} source source-wildcard..`

Step 3. Apply the ACL to an interface by using the command `ip access-group { acl-number | acl-name } {in|out}` under interface configuration mode.

### Example 4 *Standard and Extended Named ACLs*

#### Named Standard ACL

```
R1(config)# ip access-list standard STANDARD_ACL
R1(config-std-nacl)# deny 172.16.0.0 0.0.255.255
R1(config-std-nacl)# deny host 192.168.1.1
R1(config-ext-nacl)# permit any
R1(config-ext-nacl)# exit
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group STANDARD_ACL in
```

#### Numbered Standard ACL

```
R1(config)# access-list 1 deny 172.16.0.0 0.0.255.255
R1(config)# access-list 1 deny host 192.168.1.1
R1(config)# access-list 1 permit any
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group 1 in
```

#### Named Extended ACL

```
R1(config)# ip access-list extended EXTENDED_ACL
R1(config-ext-nacl)# deny tcp any any eq 23
R1(config-ext-nacl)# deny icmp any any
R1(config-ext-nacl)# deny ip host 10.1.2.2 host 10.1.2.1
R1(config-ext-nacl)# permit ip any any
R1(config-ext-nacl)# exit
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group EXTENDED_ACL in
```

#### Numbered Extended ACL

```
R1(config)# access-list 100 deny tcp any any eq 23
R1(config)# access-list 100 deny icmp any any
R1(config)# access-list 100 deny ip host 10.1.2.2 host 10.1.2.1
R1(config)# access-list 100 permit ip any any
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group 100 in
```