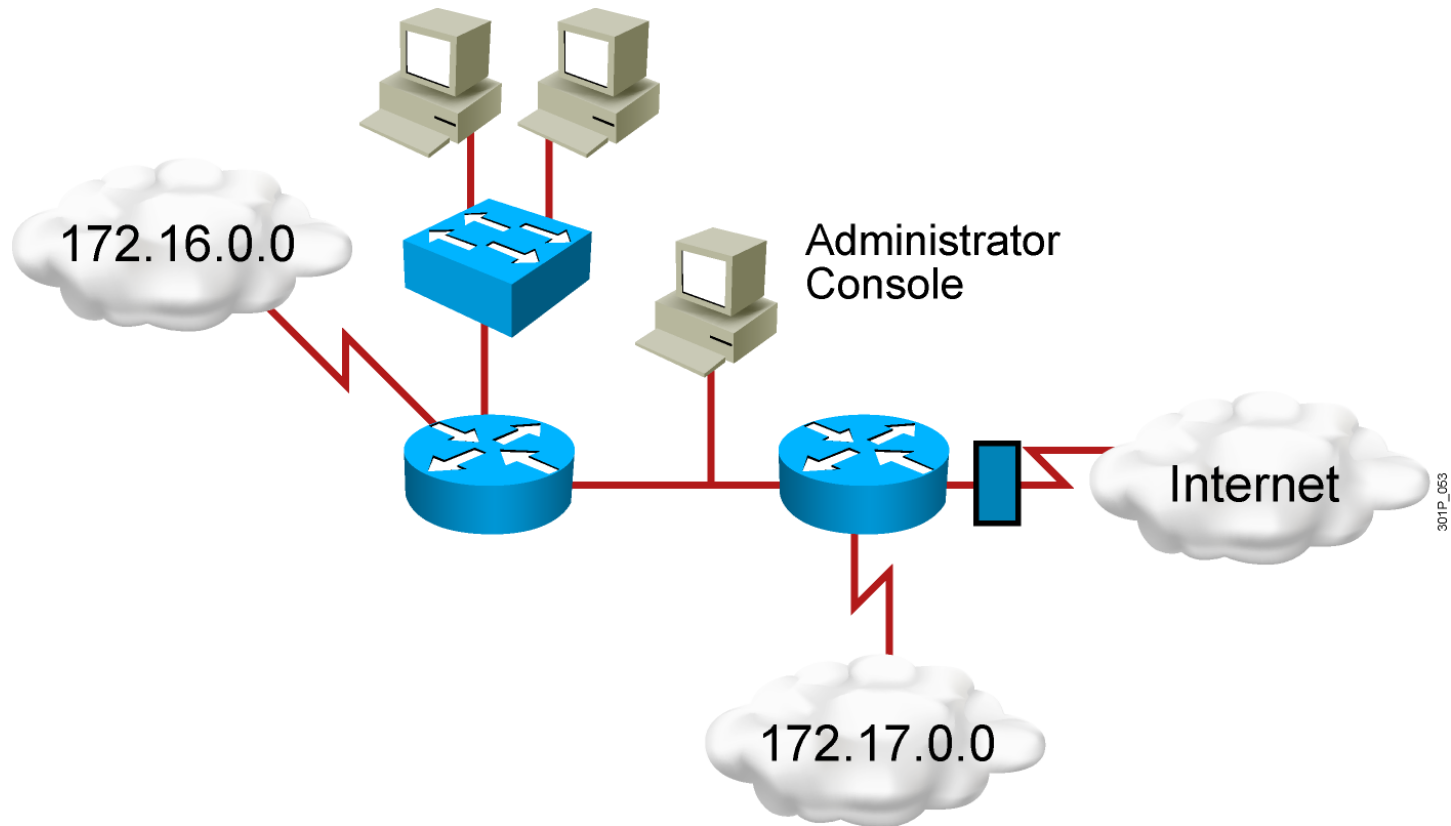




Access Control Lists

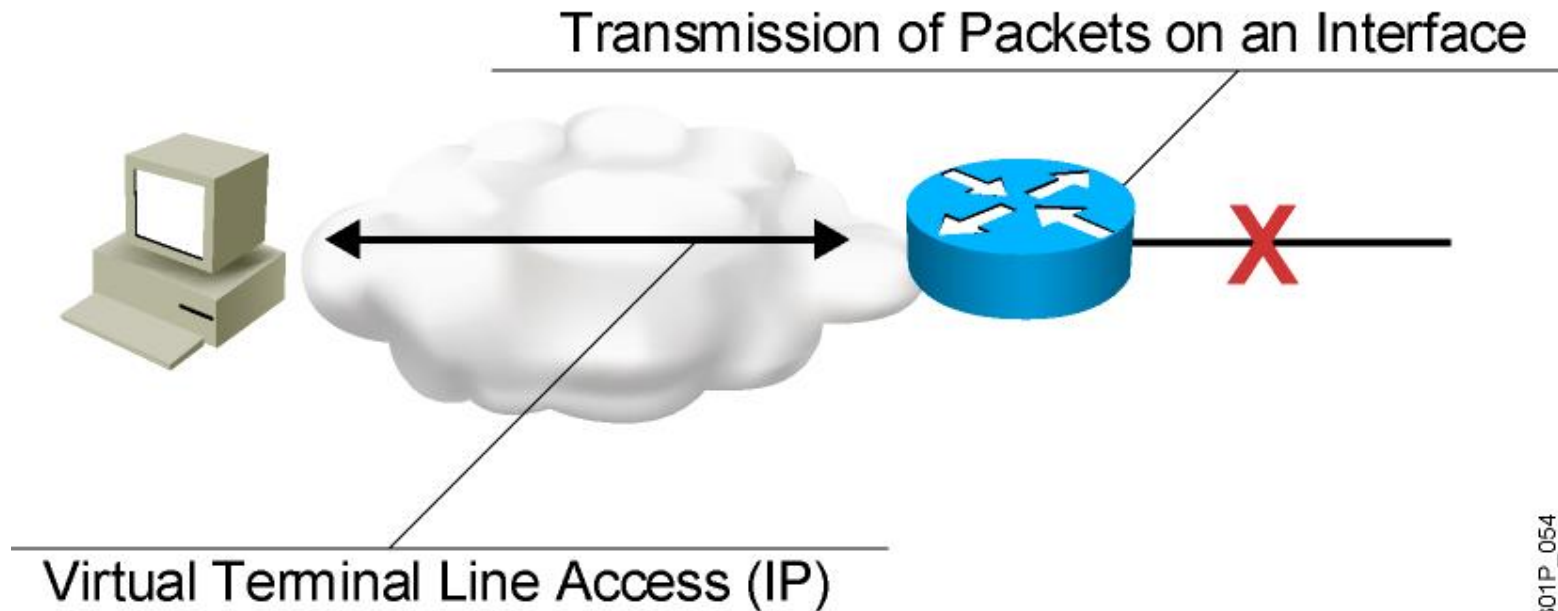
Introducing ACL Operation

Why Use ACLs?



- **Filtering:** Manage IP traffic by filtering packets passing through a router
- **Classification:** Identify traffic for special handling

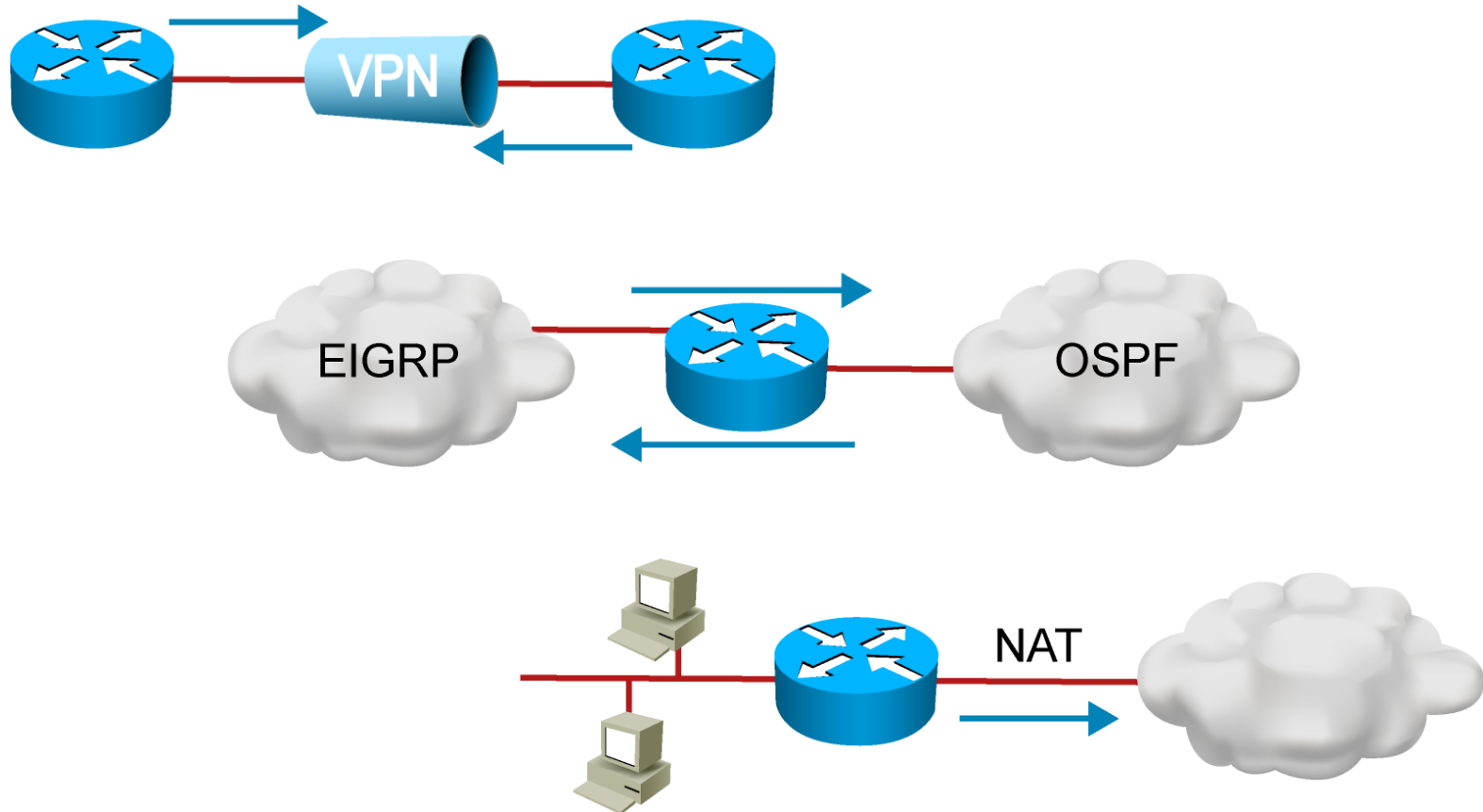
ACL Applications: Filtering



301P_054

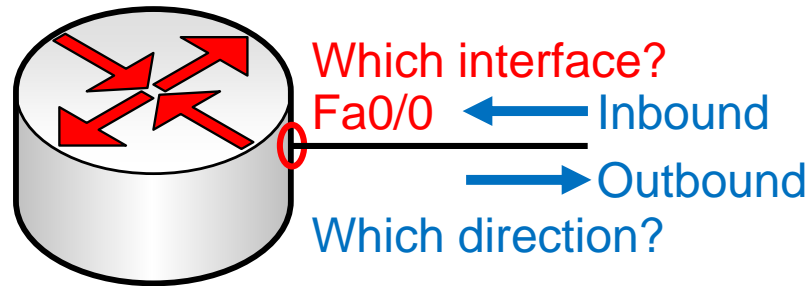
- Permit or deny packets moving through the router.
- Permit or deny vty access to or from the router.
- Without ACLs, all packets could be transmitted to all parts of your network.

ACL Applications: Classification



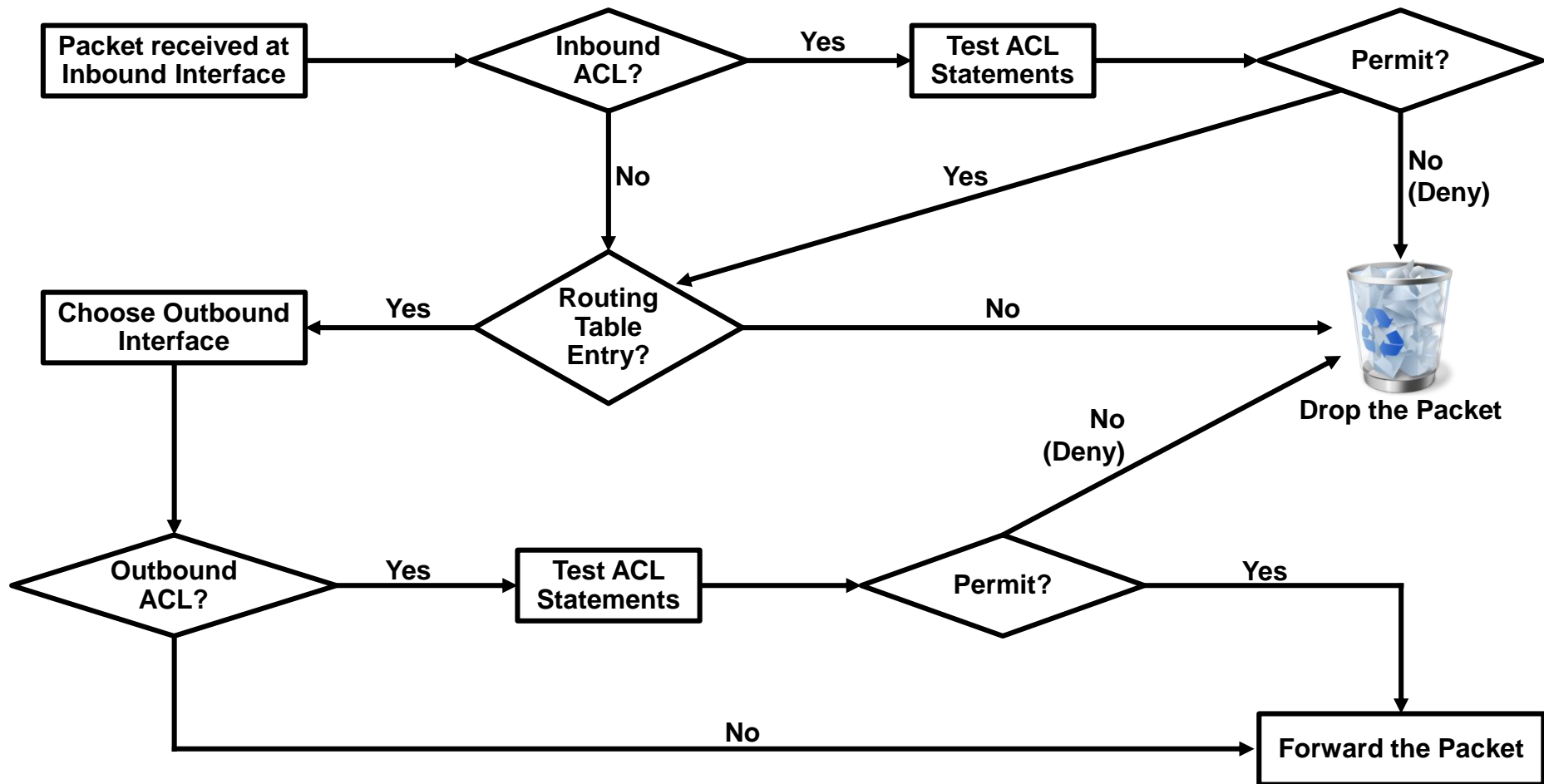
Special handling for traffic based on packet tests

ACL Operation

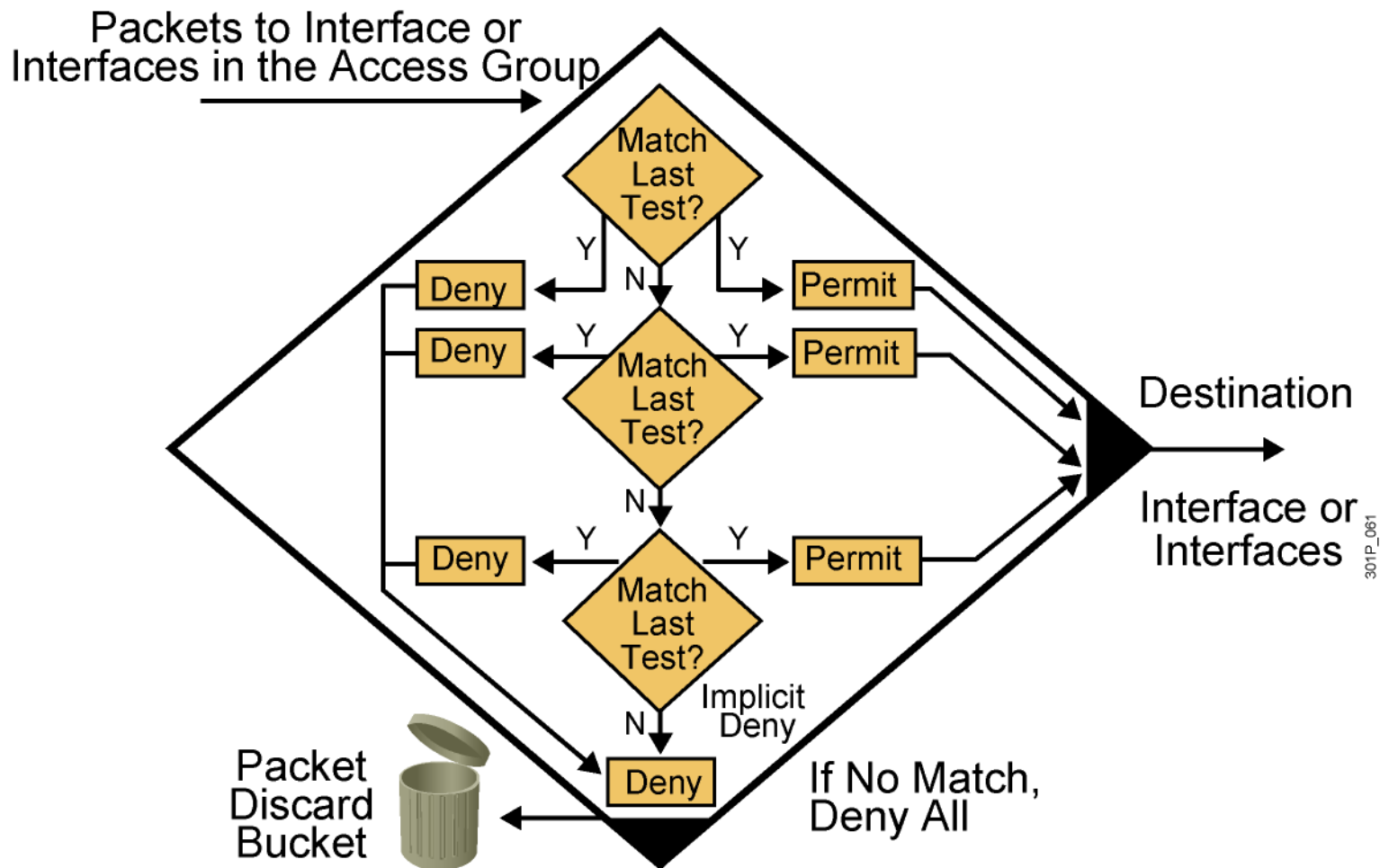


- ACL is created and then apply to an interface.
- **Inbound ACL:**
 - Filters packets coming into a specific interface and before they are routed to the outbound interface.
 - An Inbound ACL is efficient because it saves the overhead of routing lookups if the packet is denied.
- **Outbound ACL:** Filters packets after being routed, regardless of the inbound interface.

ACL Operation (Cont.)



A List of Tests: Deny or Permit



Types of ACLs

	Standard	Extended
Test conditions	Checks source address	Checks source and destination address
	Permits or denies entire protocol suite	Permits or denies specific protocols and applications
Identifier (Numbered ACL)	1-99, 1300-1999	100-199, 2000-2699
Identifier (Named ACL)	Alphanumeric string	

- Numbered standard IPv4 lists (1–99) test conditions of all IP packets for source addresses. Expanded range (1300–1999).
- Numbered extended IPv4 lists (100–199) test conditions of source and destination addresses, specific TCP/IP protocols, and ports. Expanded range (2000–2699).
- Named ACLs identify IP standard and extended ACLs with an alphanumeric string (name).

IP Access List Entry Sequence Numbering

- **Requires Cisco IOS Release 12.3**
- **Allows you to edit the order of ACL statements using sequence numbers**
 - In software earlier than Cisco IOS Release 12.3, a text editor is used to create ACL statements, then the statements are copied into the router in the correct order.
- **Allows you to remove a single ACL statement from the list using a sequence number**
 - With named ACLs in software earlier than Cisco IOS Release 12.3, you must use `no {deny | permit} protocol source source-wildcard destination destination-wildcard` to remove an individual statement.
 - With numbered ACLs in software earlier than Cisco IOS Release 12.3, you must remove the entire ACL to remove a single ACL statement.

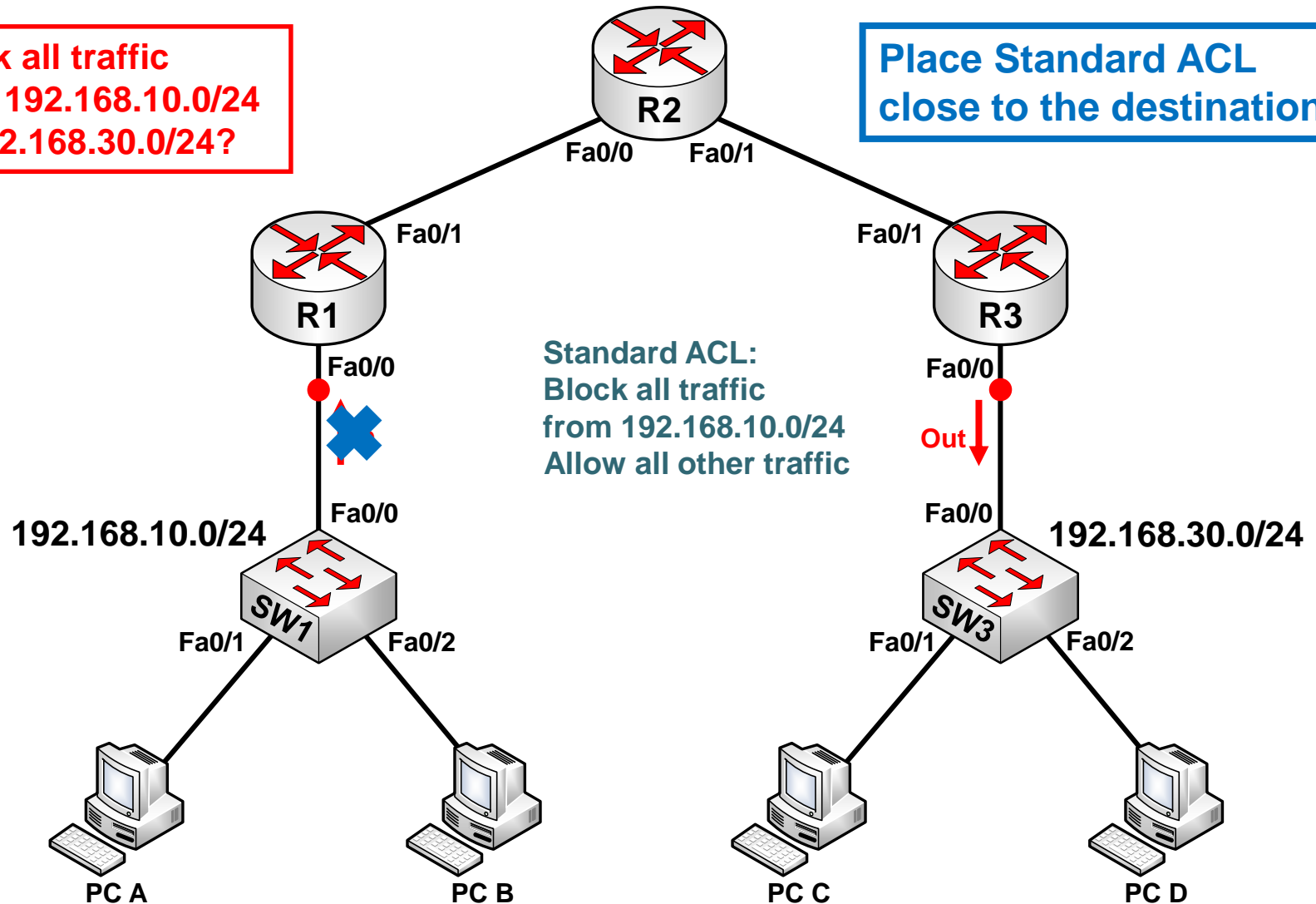
ACL Configuration Guidelines

- Standard or extended indicates what can be filtered.
- **Only one ACL per interface, per protocol, and per direction is allowed.**
- The order of ACL statements controls testing, therefore, the most specific statements go at the top of the list.
- **The last ACL test is always an implicit deny everything else statement, so every list needs at least one permit statement.**
- ACLs are created globally and then applied to interfaces for inbound or outbound traffic.
- **An ACL can only filter traffic going through the router, or traffic to and from the router (remote access only), depending on how it is applied.**

Standard ACL Placement

Block all traffic
from 192.168.10.0/24
to 192.168.30.0/24?

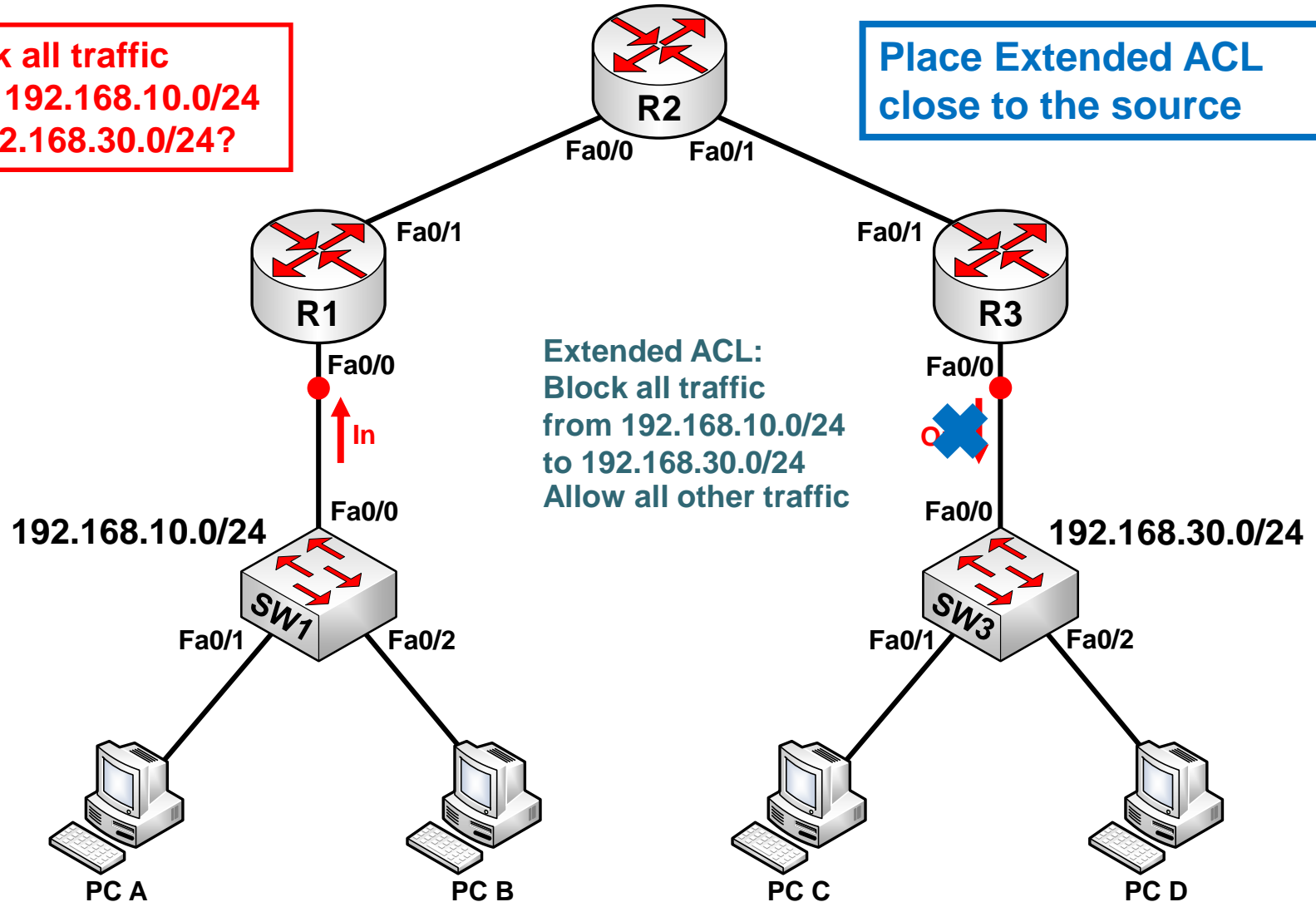
Place Standard ACL
close to the destination



Extended ACL Placement

Block all traffic
from 192.168.10.0/24
to 192.168.30.0/24?

Place Extended ACL
close to the source



Wildcard Mask vs Subnet Mask

Subnet Mask		Wildcard Mask
1 bits	Match	0 bits
0 bits	Don't care	1 bits

Example:

		Binary			
IP Address	192.168.1.0/24	11000000	10101000	00000001	00000000
Subnet Mask	255.255.255.0	11111111	11111111	11111111	00000000
Wildcard Mask	0.0.0.255	00000000	00000000	00000000	11111111

Wildcard Bits to Match IP Subnets

Example 1

Match for IP range 172.16.10.32 to 172.16.10.63

Address and wildcard mask?

172.16.10.32 0.0.0.31

Wildcard Bits to Match IP Subnets

Example 2

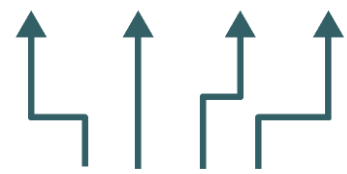
Match for IP subnets 172.30.16.0/24 to 172.30.31.0/24.

Address and wildcard mask?


172.30.16.0 0.0.15.255

Wildcard Bit Mask Abbreviations

- **172.30.16.29** **0.0.0.0** matches all of the address bits
- Abbreviate this wildcard mask using the IP address preceded by the keyword host (**host 172.30.16.29**)

172.30.16.29

Wildcard Mask: 0.0.0.0
(Matches All Bits)

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

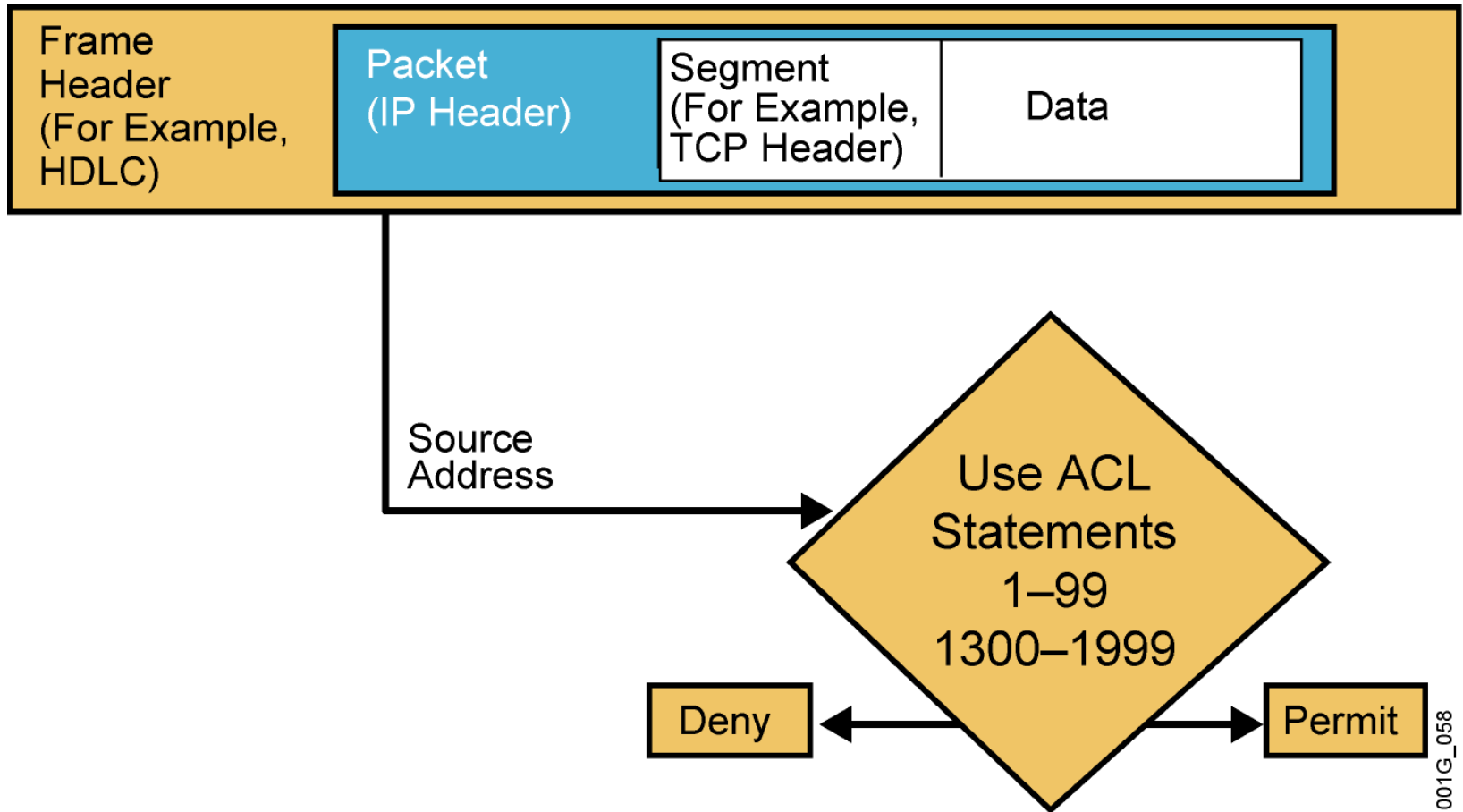
0.0.0.0

Wildcard Mask: 255.255.255.255
(Ignore All Bits)



Access Control Lists

Configuring and Troubleshooting ACLs

Testing Packets with Numbered Standard IPv4 ACLs



Numbered Standard IPv4 ACL Configuration

RouterX (config) #

```
access-list access-list-number {permit | deny | remark} source-IP  
[wildcard-mask]
```

- Uses 1 to 99, or 1300 to 1999 for the *access-list-number*.
- The first entry is assigned a sequence number of 10, and successive entries are incremented by 10.
- Default wildcard mask is 0.0.0.0 (only standard ACL).
- no access-list *access-list-number* removes the entire ACL.
- remark lets you add a description to the ACL.

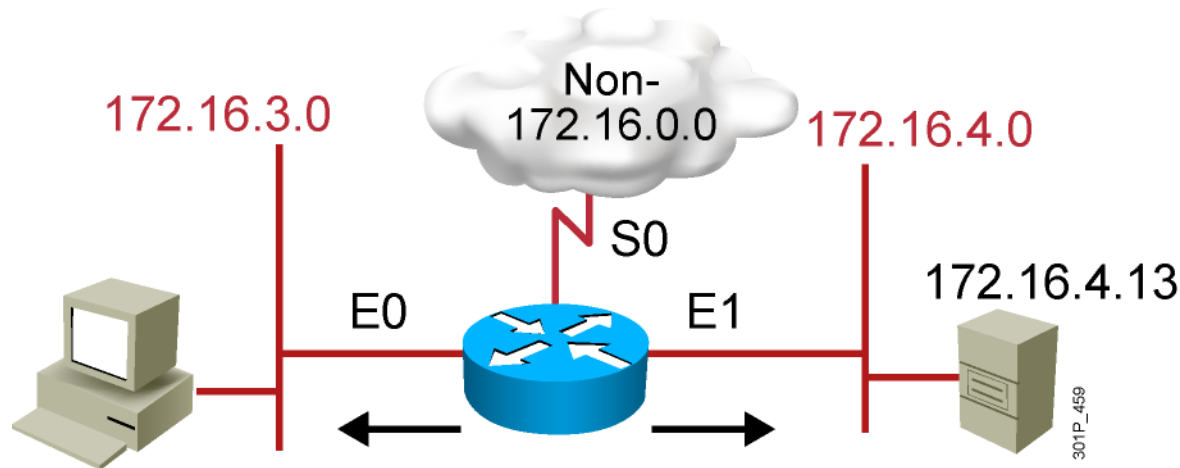
RouterX (config-if) #

```
ip access-group access-list-number {in | out}
```

- Activates the list on an interface.
- Sets inbound or outbound testing.
- no ip access-group *access-list-number* {in | out} removes the ACL from the interface.

Numbered Standard IPv4 ACL

Example 1



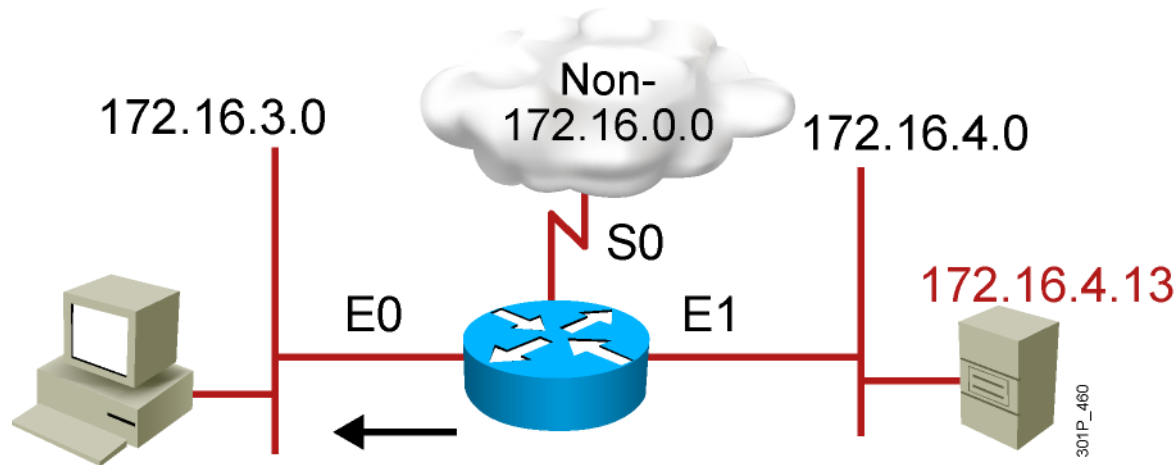
```
RouterX(config)# access-list 1 permit 172.16.0.0 0.0.255.255  
(implicit deny all - not visible in the list)  
(Or access-list 1 deny 0.0.0.0 255.255.255.255)
```

```
RouterX(config)# interface ethernet 0  
RouterX(config-if)# ip access-group 1 out  
RouterX(config)# interface ethernet 1  
RouterX(config-if)# ip access-group 1 out
```

Permit my network only

Numbered Standard IPv4 ACL

Example 2



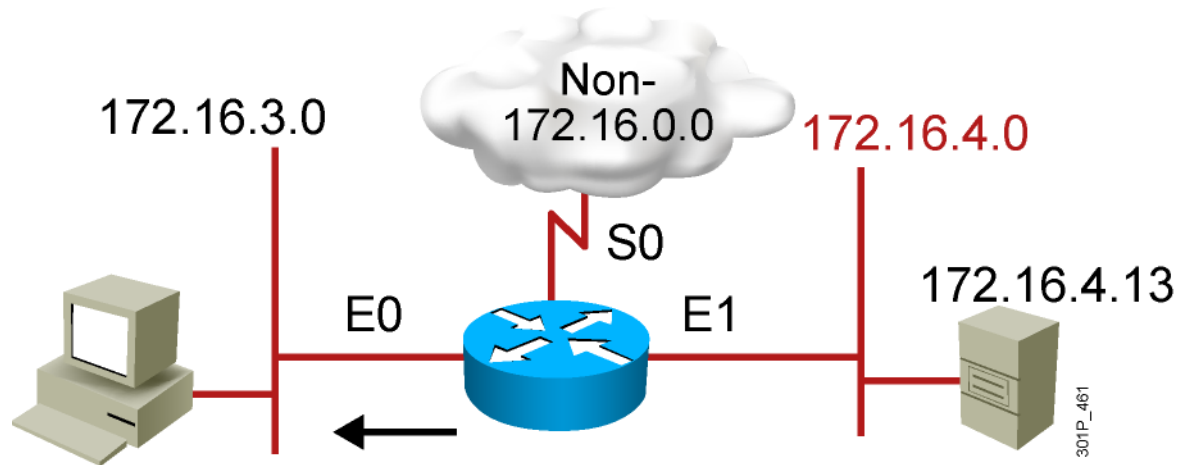
```
RouterX(config)# access-list 1 deny 172.16.4.13 0.0.0.0
                  (Or access-list 1 deny host 172.16.4.13)
RouterX(config)# access-list 1 permit 0.0.0.0 255.255.255.255
                  (Or access-list 1 permit any)
(implicit deny all)
(Or access-list 1 deny 0.0.0.0 255.255.255.255)

RouterX(config)# interface ethernet 0
RouterX(config-if)# ip access-group 1 out
```

Deny a specific host

Numbered Standard IPv4 ACL

Example 3



```
RouterX(config)# access-list 1 deny 172.16.4.0 0.0.0.255
RouterX(config)# access-list 1 permit any
(implicit deny all)
(Or access-list 1 deny 0.0.0.0 255.255.255.255)

RouterX(config)# interface ethernet 0
RouterX(config-if)# ip access-group 1 out
```

Deny a specific subnet

Standard ACLs to Control vty Access

RouterX(**config-line**) #

```
access-class access-list-number {in | out}
```

- Restricts incoming or outgoing connections between a particular vty and the addresses in an ACL

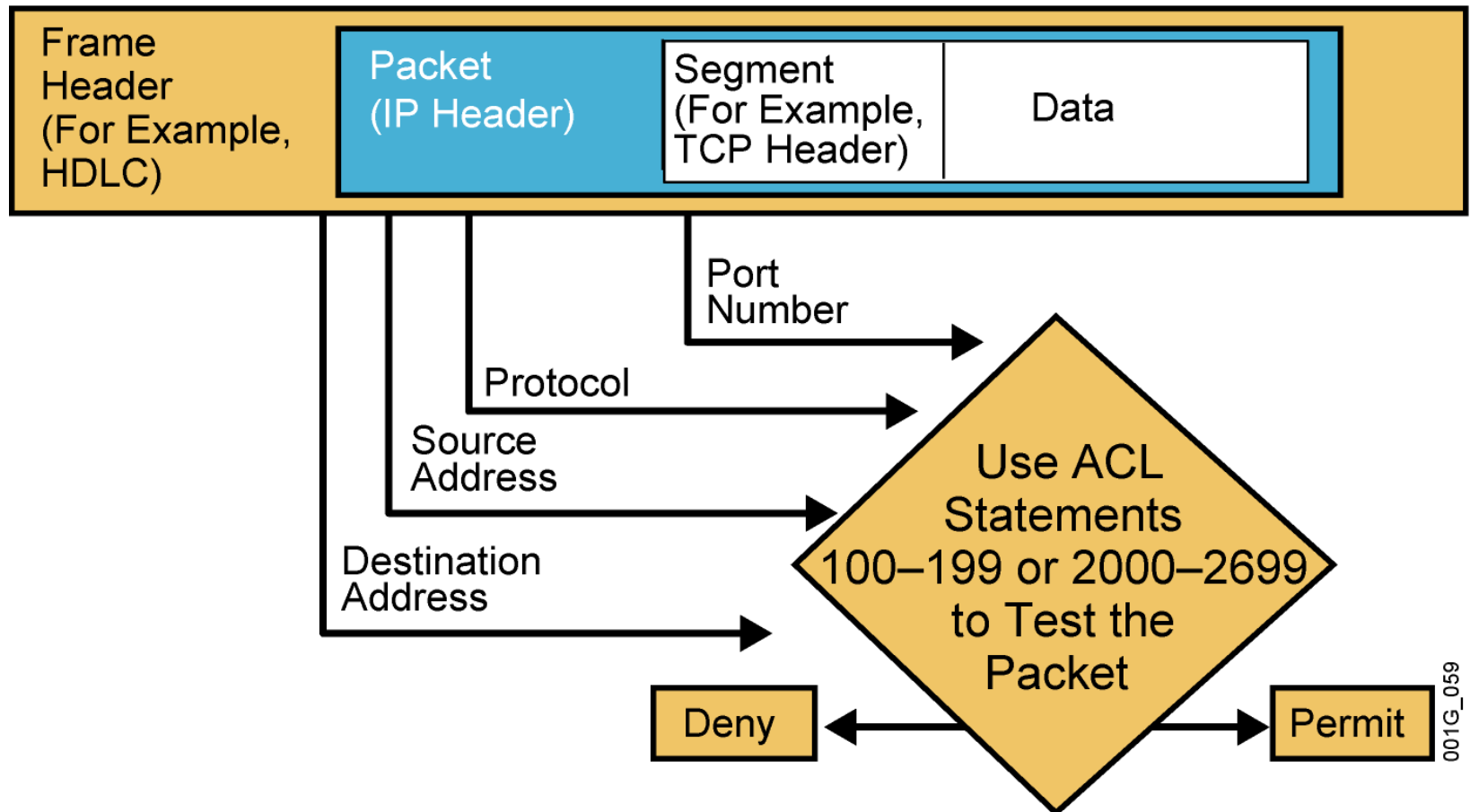
Example:

```
access-list 12 permit 192.168.1.0 0.0.0.255  
(implicit deny any)  
!  
line vty 0 4  
  access-class 12 in
```

- Permits only hosts in network 192.168.1.0 0.0.0.255 to connect to the router vty lines

Testing Packets with Numbered Extended IPv4 ACLs

An Example from a TCP/IP Packet



Numbered Extended IPv4 ACL Configuration

RouterX (config) #

```
access-list access-list-number {permit | deny}  
protocol source-IP source-wildcard [operator source-port]  
destination-IP destination-wildcard [operator destination-port]  
[established] [log]
```

Sets parameters for this list entry:

- Uses 100 to 199, or 2000 to 2699 for the *access-list-number*.
- *protocol* → IP, TCP, UDP, ICMP, ...
- *operator port (for TCP/UDP only)*:
 - lt ... → less than ...
 - gt ... → greater than ...
 - eq ... → equal ...
 - neq ... → not equal ...
 - range *low-port-number high-port-number*
→ range from *low-port-number* to *high-port-number*
 - Can use pre-defined application name instead of port number

RouterX (config-if) #

```
ip access-group access-list-number {in | out}
```

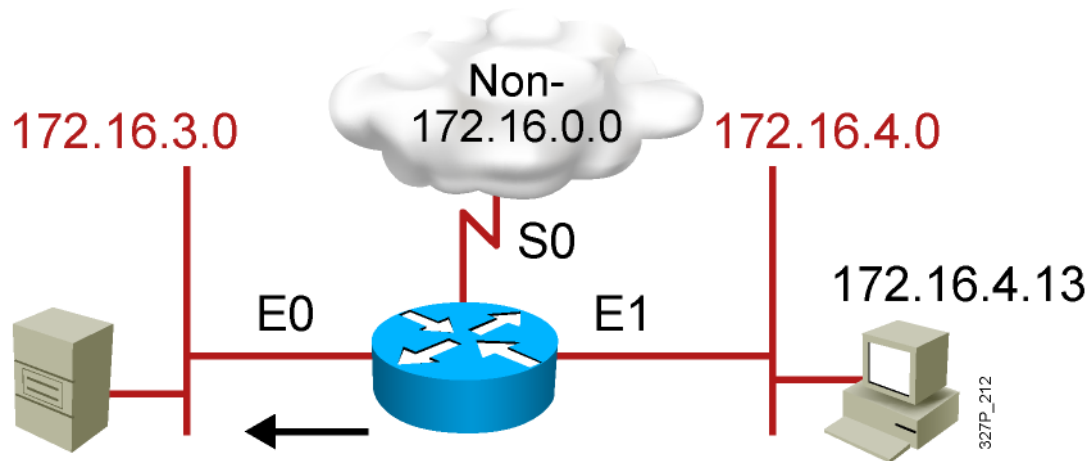
- Activates the extended list on an interface

Common Port Numbers

Protocol	Port Number	Application	Pre-defined ACL Application Name
TCP	20	FTP data	ftp-data
TCP	21	FTP control	ftp
TCP	22	SSH	
TCP	23	Telnet	telnet
TCP	25	SMTP	smtp
TCP/UDP	53	DNS	domain
UDP	67	DHCP BOOTP Server	bootps
UDP	68	DHCP BOOTP Client	bootpc
UDP	69	TFTP	tftp
TCP	80	HTTP	www

Numbered Extended IPv4 ACL

Example 1



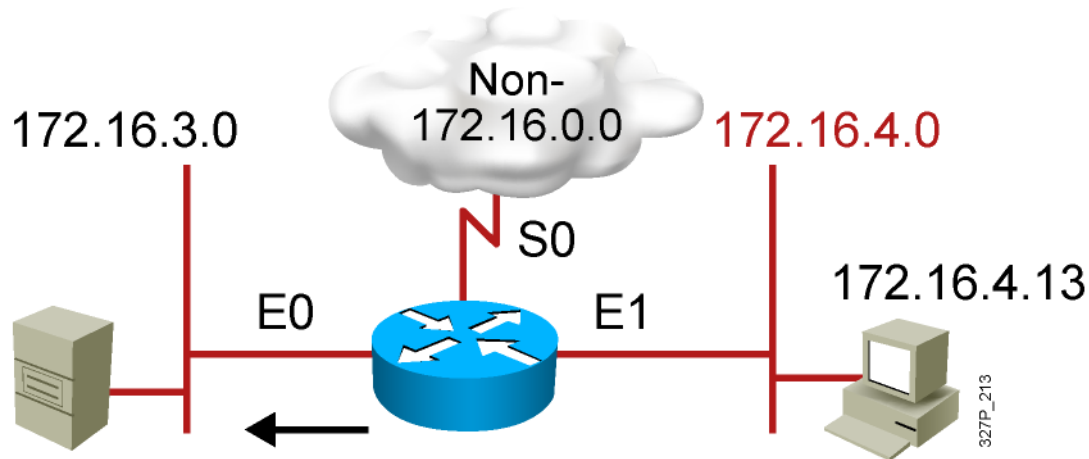
```
RouterX(config)# access-list 101 deny tcp 172.16.4.0 0.0.0.255 172.16.3.0 0.0.0.255 eq 21
RouterX(config)# access-list 101 deny tcp 172.16.4.0 0.0.0.255 172.16.3.0 0.0.0.255 eq 20
RouterX(config)# access-list 101 permit ip any any
                (implicit deny all)
                (Or access-list 101 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255)

RouterX(config)# interface ethernet 0
RouterX(config-if)# ip access-group 101 out
```

- Deny FTP traffic from subnet 172.16.4.0 to subnet 172.16.3.0 out E0
- Permit all other traffic

Numbered Extended IPv4 ACL

Example 2



```
RouterX(config)# access-list 101 deny tcp 172.16.4.0 0.0.0.255 any eq 23
RouterX(config)# access-list 101 permit ip any any
(implicit deny all)
```

```
RouterX(config)# interface ethernet 0
RouterX(config-if)# ip access-group 101 out
```

- Deny only Telnet traffic from subnet 172.16.4.0 out E0
- Permit all other traffic

Named IP ACL Configuration

RouterX(config)#

```
ip access-list {standard | extended} name
```

- Alphanumeric name string must be unique

RouterX(config-{std-|ext-}nacl)#

```
[sequence-number] {permit | deny} {ip access list test conditions}  
{permit | deny} {ip access list test conditions}
```

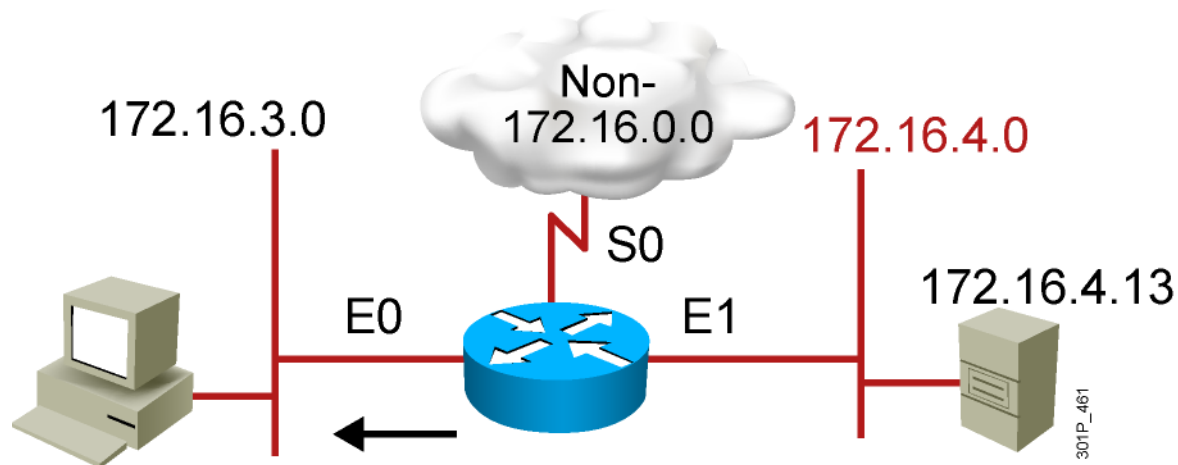
- If not configured, sequence numbers are generated automatically starting at 10 and incrementing by 10
- *no sequence number* removes the specific test from the named ACL

RouterX(config-if)#

```
ip access-group name {in | out}
```

- Activates the named IP ACL on an interface

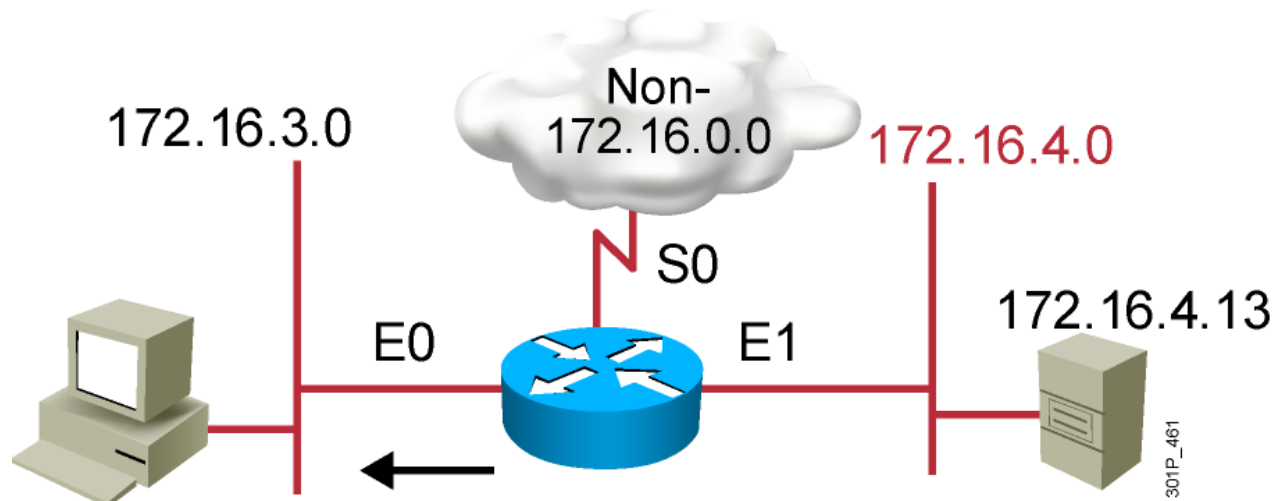
Named Standard IPv4 ACL Example



```
RouterX(config)#ip access-list standard troublemaker
RouterX(config-std-nacl)#deny host 172.16.4.13
RouterX(config-std-nacl)#permit 172.16.4.0 0.0.0.255
RouterX(config-std-nacl)#interface e0
RouterX(config-if)#ip access-group troublemaker out
```

Deny a specific host

Named Extended IPv4 ACL Example



```
RouterX(config)#ip access-list extended badgroup
RouterX(config-ext-nacl)#deny tcp 172.16.4.0 0.0.0.255 any eq 23
RouterX(config-ext-nacl)#permit ip any any
RouterX(config-ext-nacl)#interface e0
RouterX(config-if)#ip access-group badgroup out
```

Deny Telnet from a specific subnet

Commenting ACL Statements

RouterX(config) #

```
ip access-list {standard|extended} name
```

- Creates a named ACL

RouterX(config {std- | ext-}nacl) #

```
remark remark
```

- Creates a named ACL comment

Or

RouterX(config) #

```
access-list access-list-number remark remark
```

- Creates a numbered ACL comment

Monitoring ACL Statements

```
RouterX# show access-lists [access-list number | name]
```

```
RouterX# show access-lists
```

```
Standard IP access list SALES
```

```
10 deny 10.1.1.0, wildcard bits 0.0.0.255
```

```
20 permit 10.3.3.1
```

```
30 permit 10.4.4.1
```

```
40 permit 10.5.5.1
```

```
Extended IP access list ENG
```

```
10 permit tcp host 10.22.22.1 any eq telnet (25 matches)
```

```
20 permit tcp host 10.33.33.1 any eq ftp
```

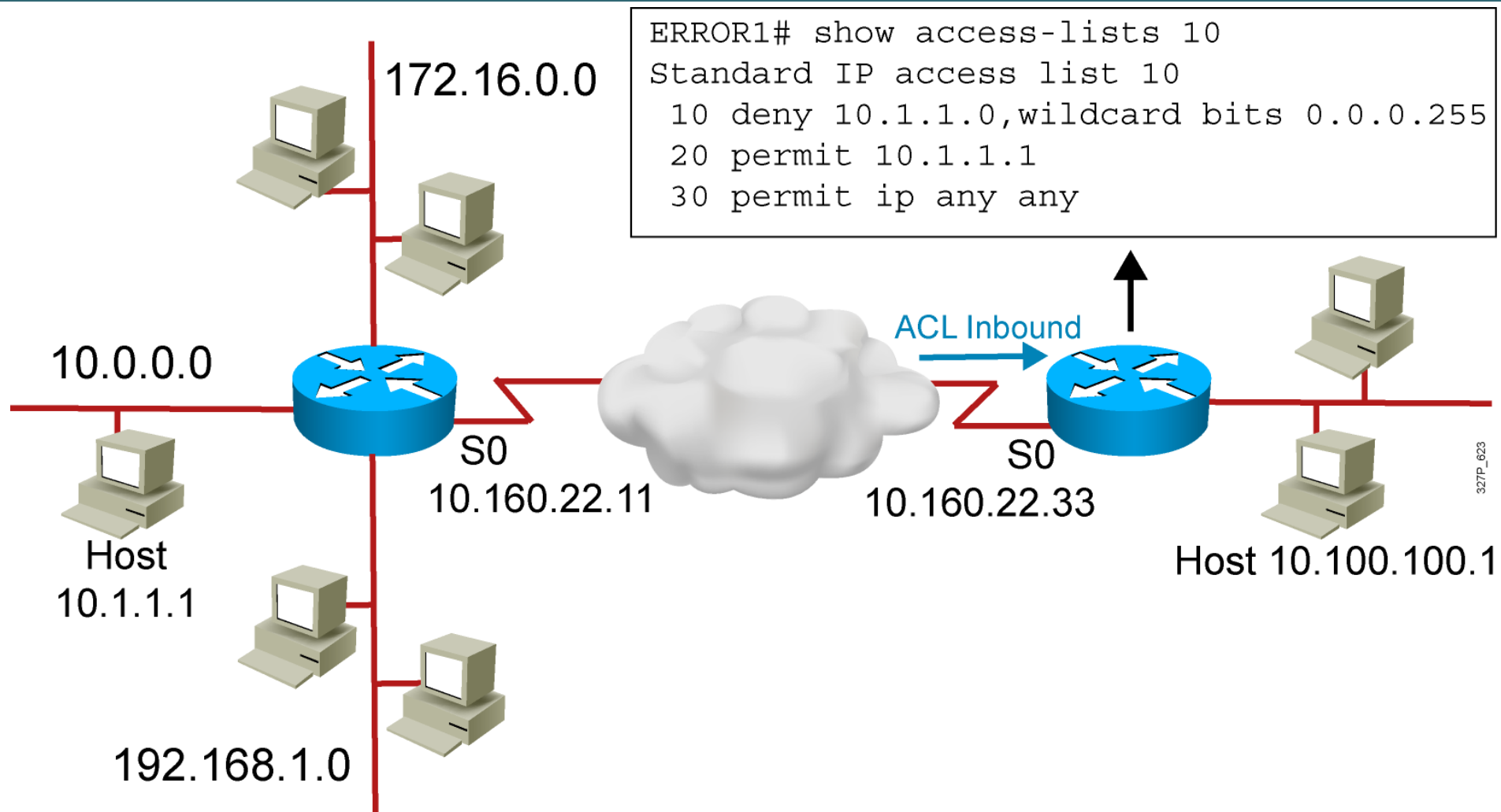
```
30 permit tcp host 10.44.44.1 any eq ftp-data
```

Displays all access lists

Verifying ACLs

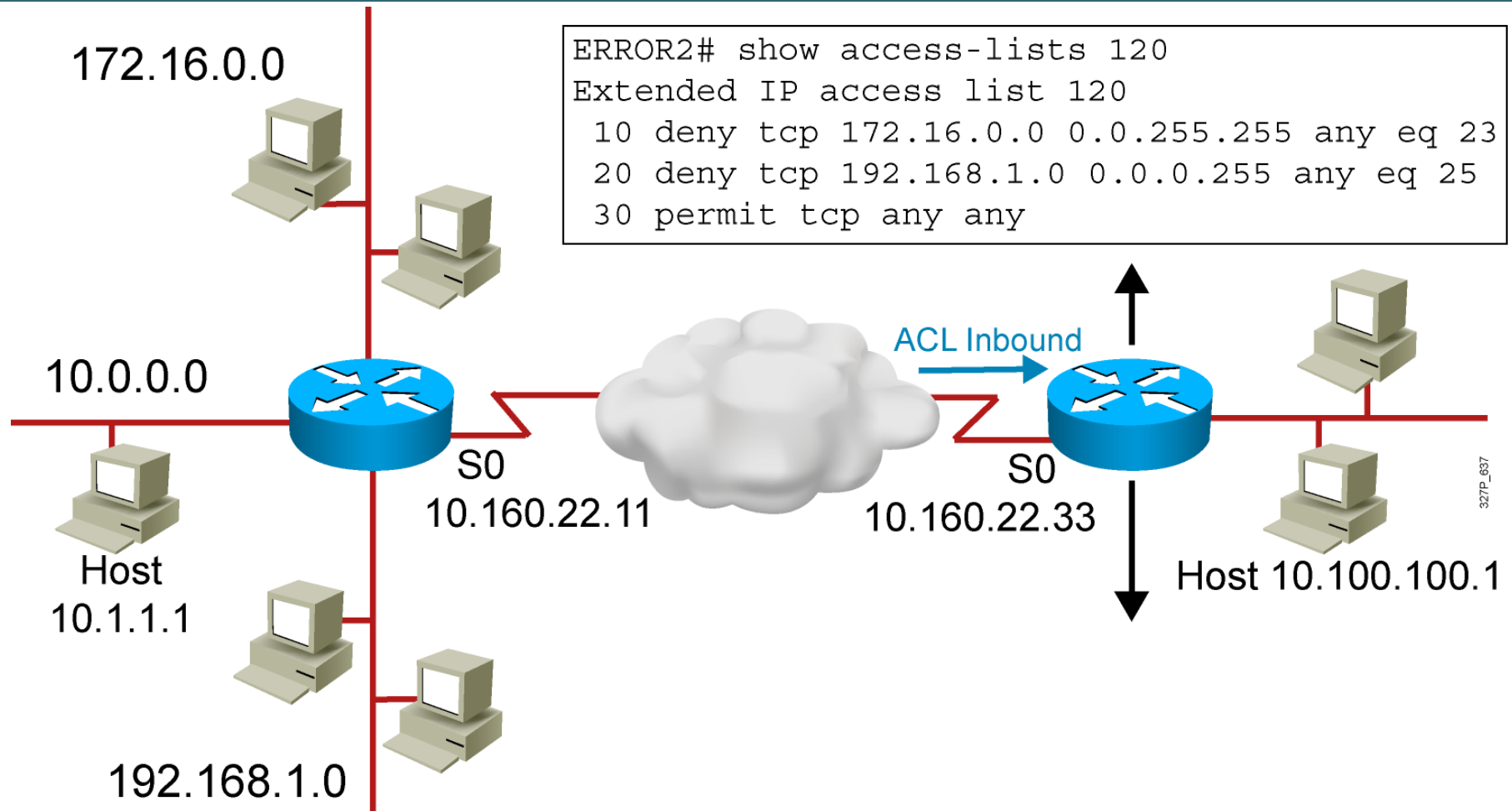
```
RouterX# show ip interfaces e0
Ethernet0 is up, line protocol is up
  Internet address is 10.1.1.11/24
  Broadcast address is 255.255.255.255
  Address determined by setup command
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound access list is 1
  Proxy ARP is enabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachable are always sent
  ICMP mask replies are never sent
  IP fast switching is enabled
  IP fast switching on the same interface is disabled
  IP Feature Fast switching turbo vector
  IP multicast fast switching is enabled
  IP multicast distributed fast switching is disabled
<text ommitted>
```

Troubleshooting Common ACL Errors



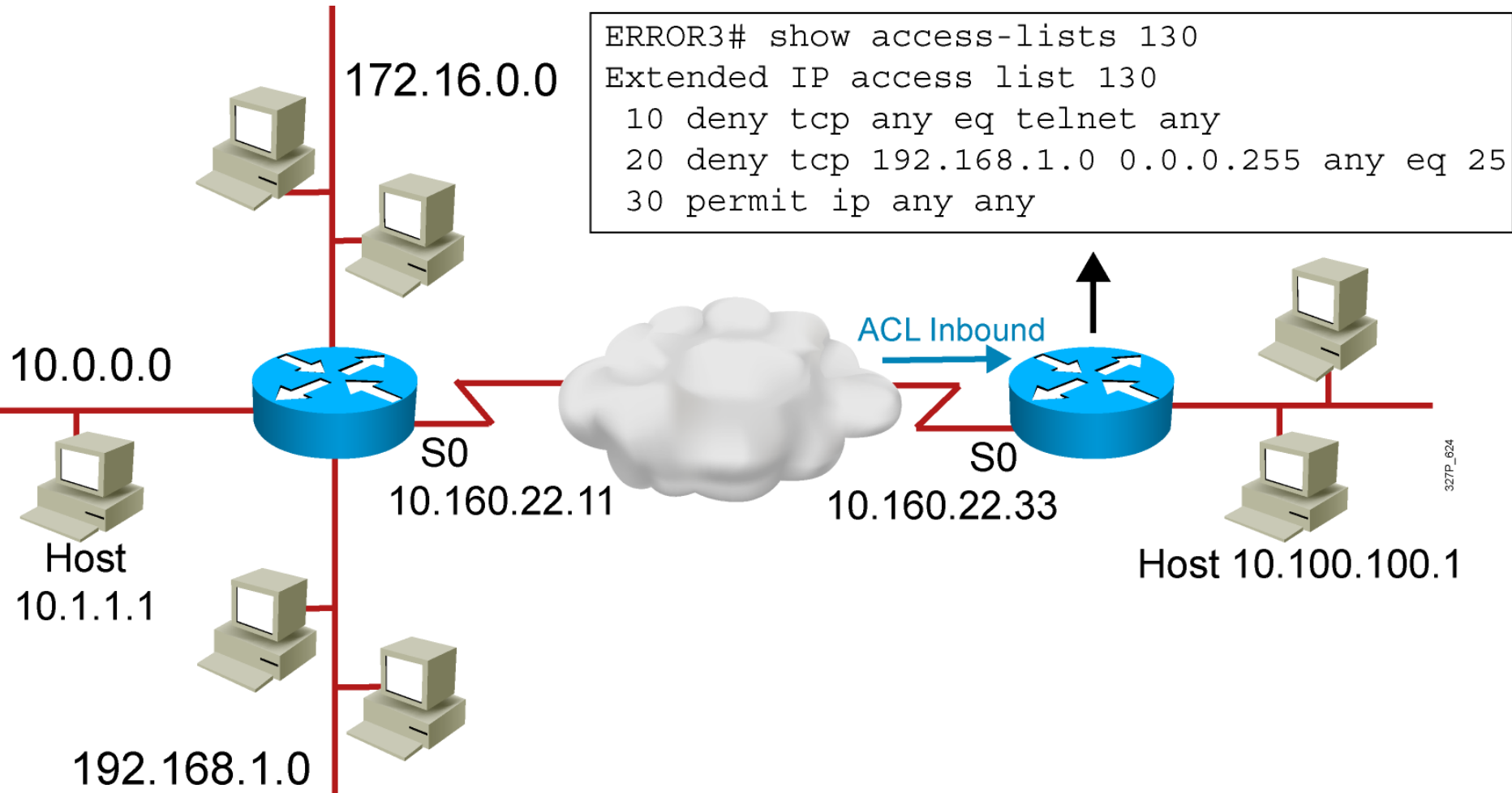
Error 1: Host 10.1.1.1 has no connectivity with 10.100.100.1.

Troubleshooting Common ACL Errors (Cont.)



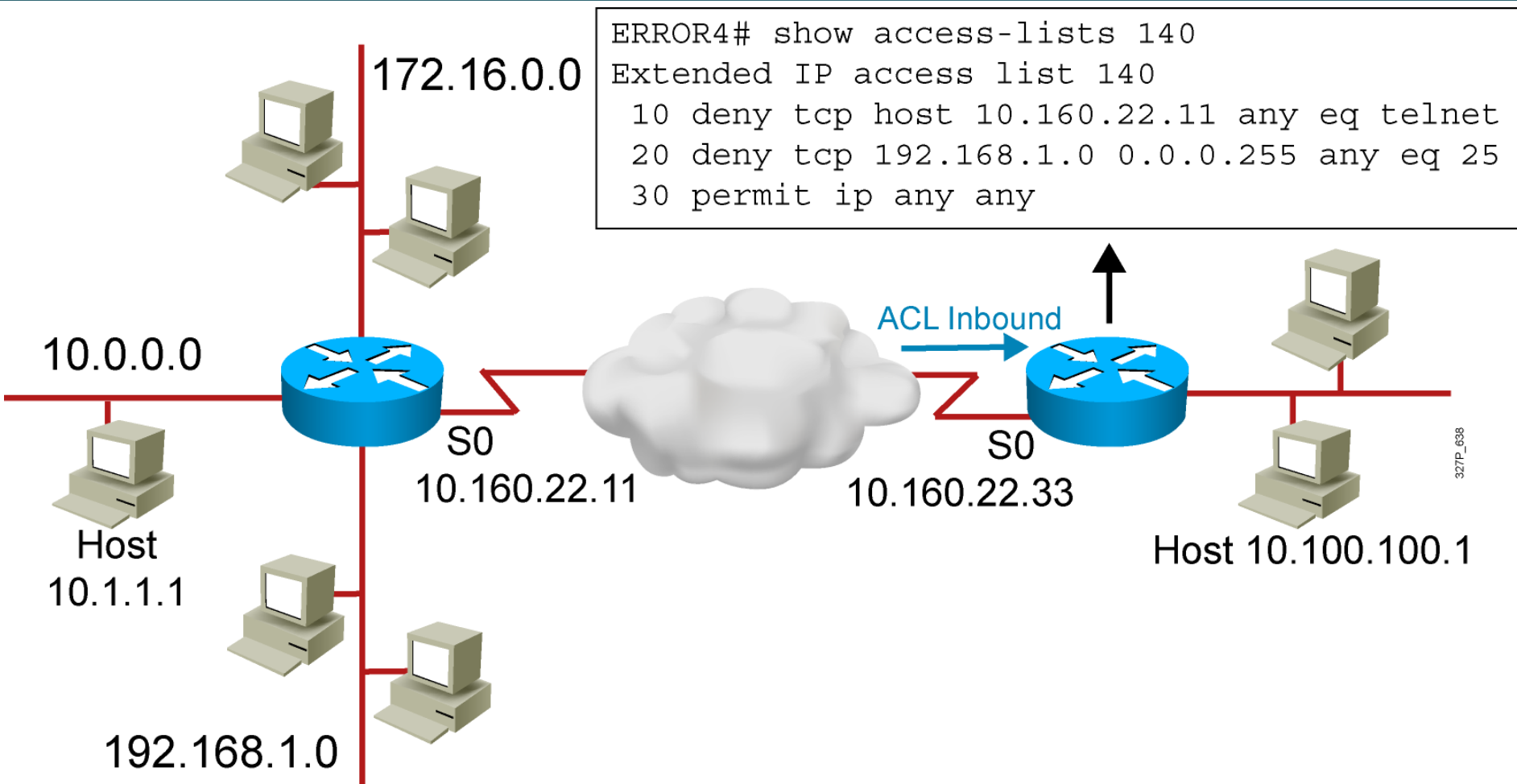
Error 2: The 192.168.1.0 network cannot use TFTP to connect to 10.100.100.1.

Troubleshooting Common ACL Errors (Cont.)



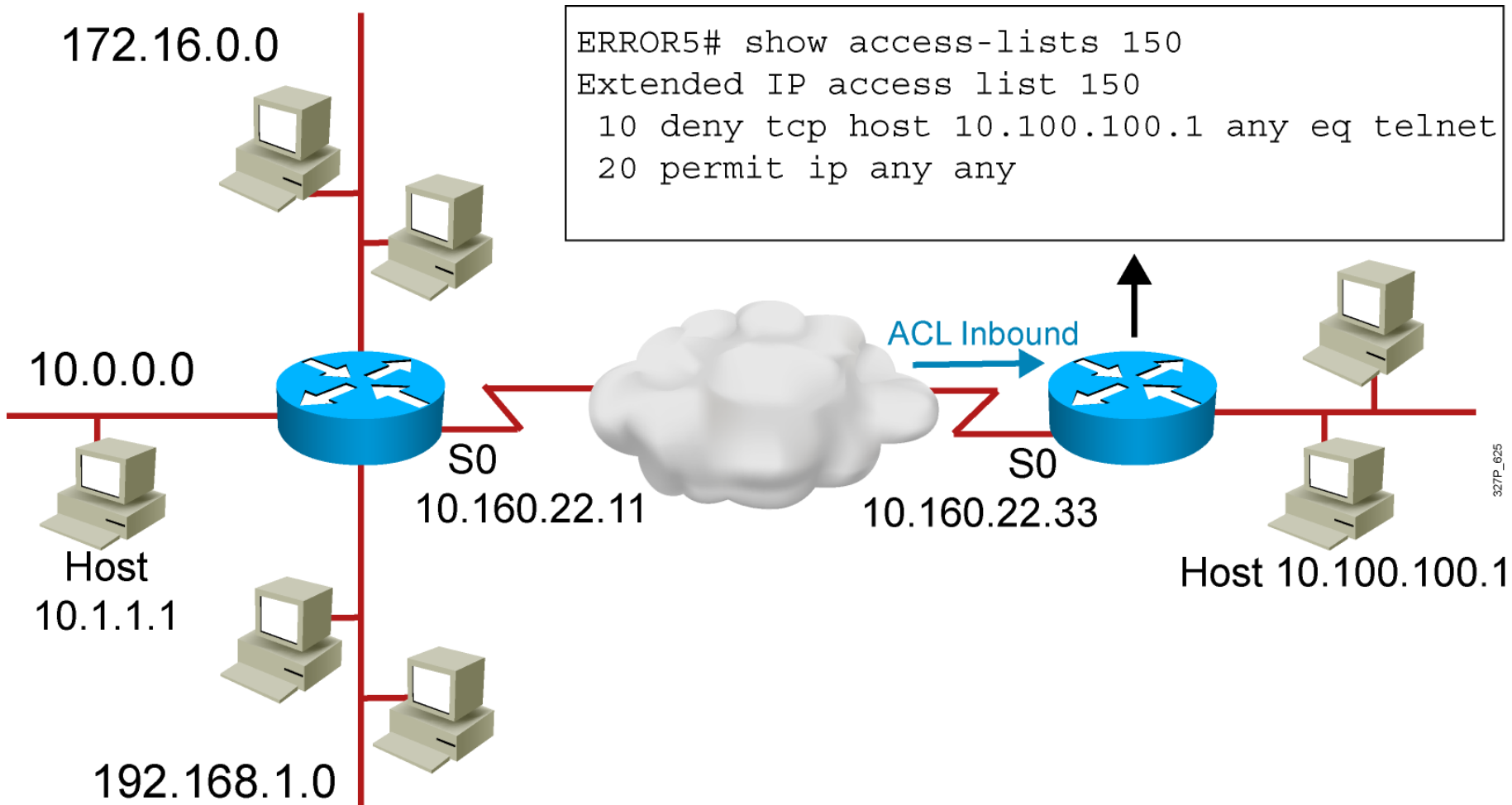
Error 3: 172.16.0.0 network can use Telnet to connect to 10.100.100.1, but this connection should not be allowed.

Troubleshooting Common ACL Errors (Cont.)



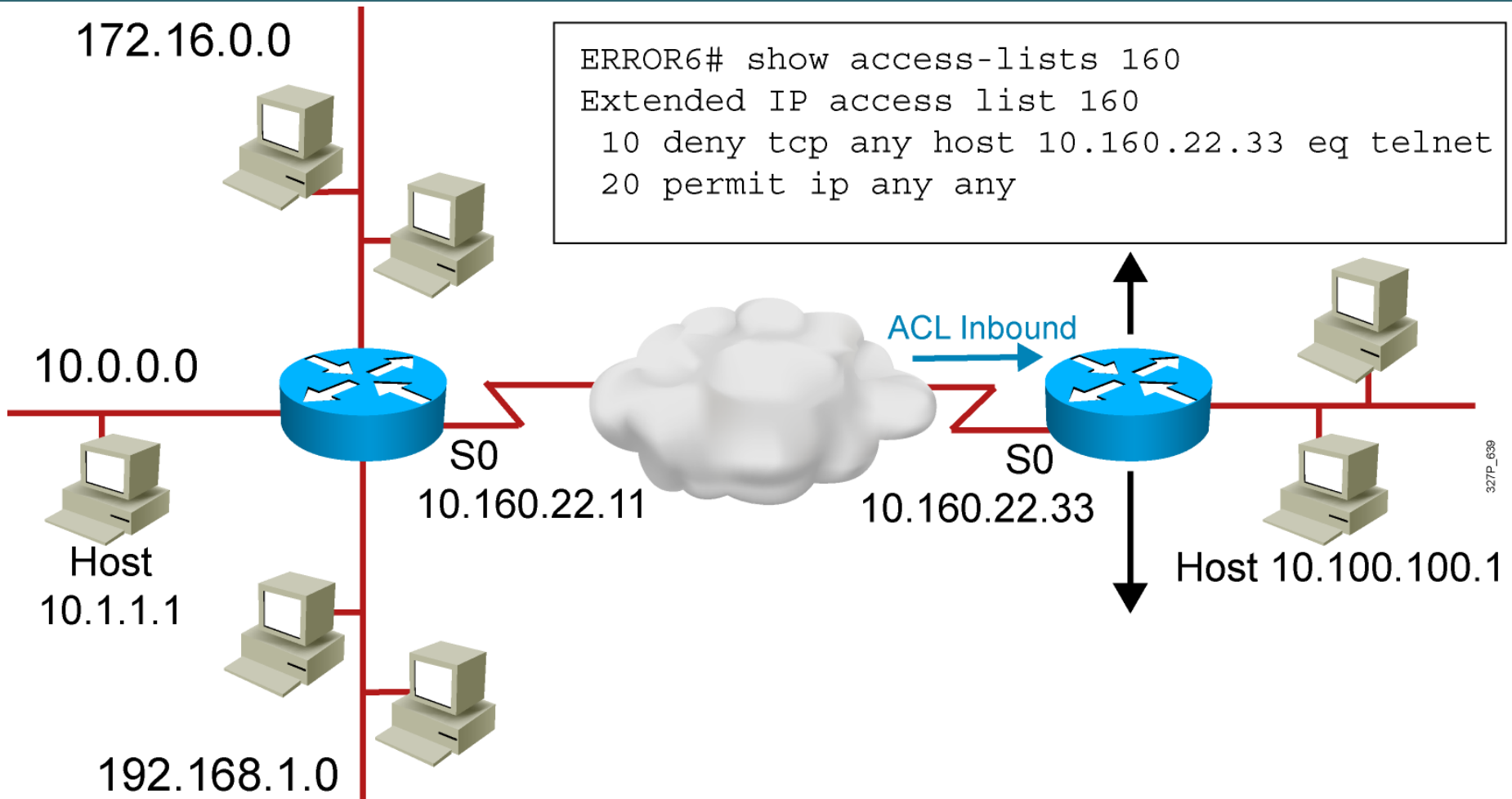
Error 4: Host 10.1.1.1 can use Telnet to connect to 10.100.100.1, but this connection should not be allowed.

Troubleshooting Common ACL Errors (Cont.)



Error 5: Host 10.100.100.1 can use Telnet to connect to 10.1.1.1, but this connection should not be allowed.

Troubleshooting Common ACL Errors (Cont.)



Error 6: Host 10.1.1.1 can use Telnet to connect into router B, but this connection should not be allowed.

