
Access Control Lists

What are ACLs?

ACLs are lists of conditions that are applied to traffic traveling across a router's interface.
These lists tell the router what types of packets to accept or deny.
Acceptance and denial can be based on specified conditions.

ACLs can be created for all routed network protocols, such as Internet Protocol (IP).

ACLs can be configured at the router to control access to a network or subnet.

Reasons to Create ACLs

The following are some of the primary reasons to create ACLs:

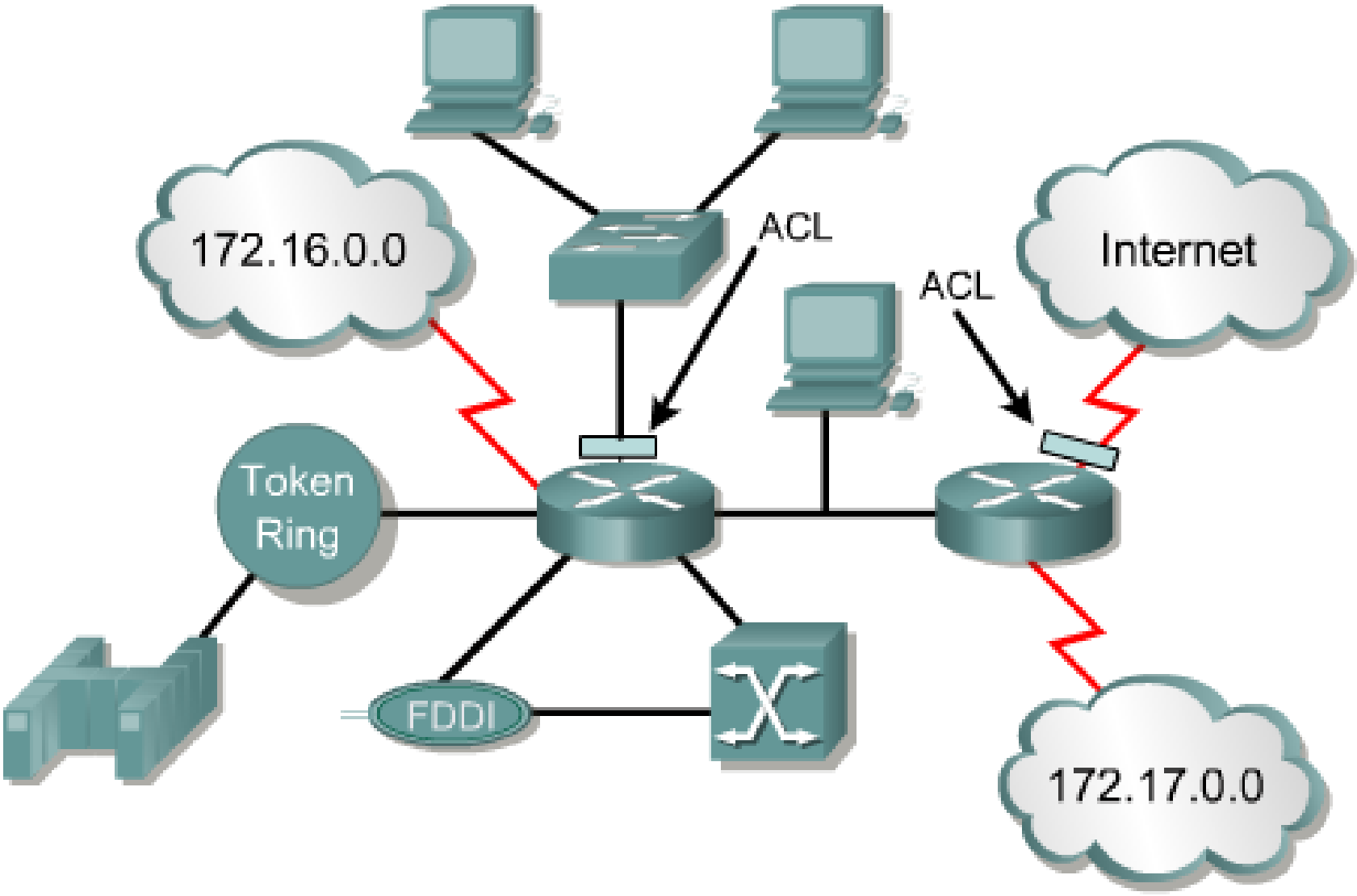
- Limit network traffic and increase network performance.
- Provide traffic flow control.
- Provide a basic level of security for network access.
- Decide which types of traffic are forwarded or blocked at the router interfaces.

•For example: Permit e-mail traffic to be routed, but block all telnet traffic.

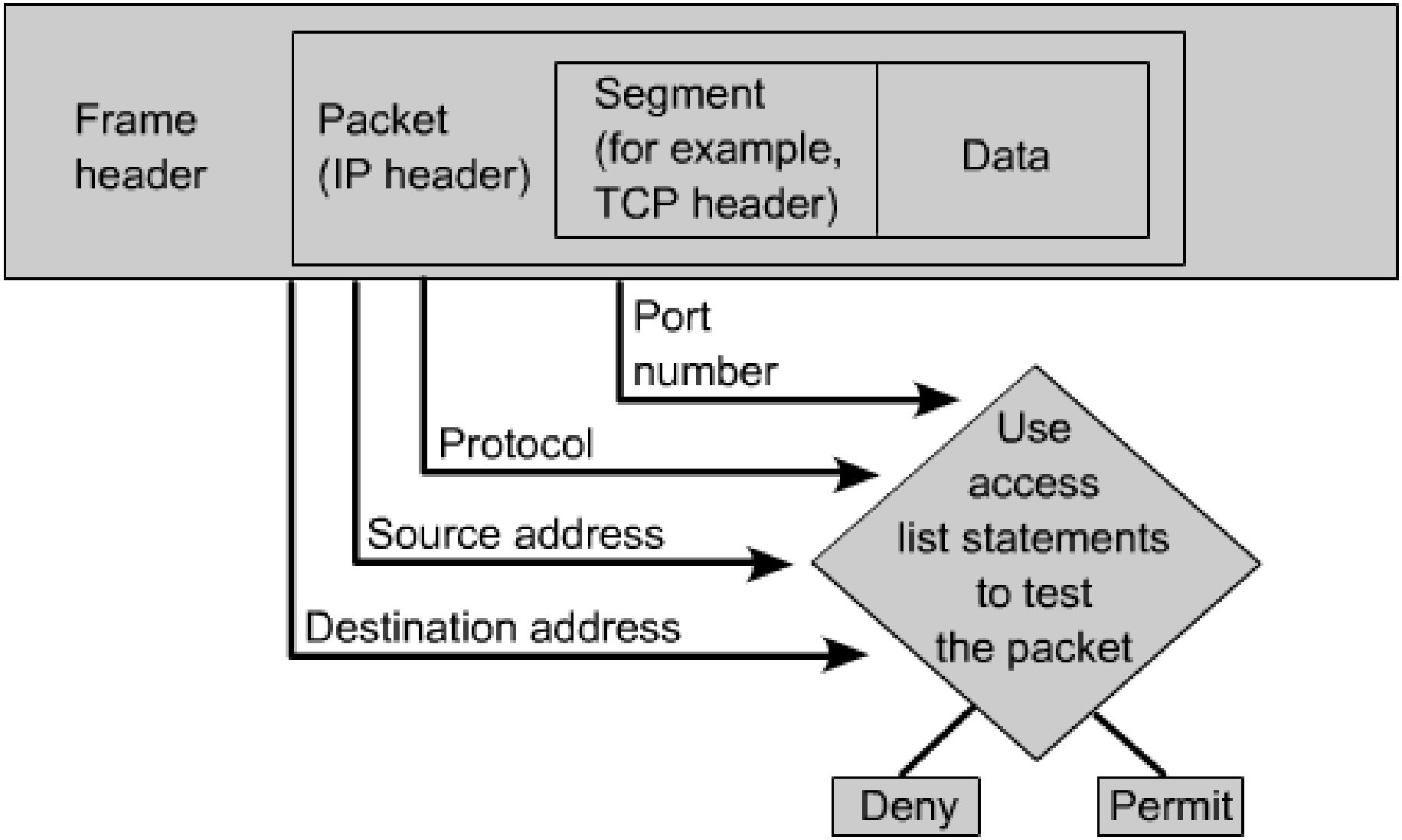
•Allow an administrator to control what areas a client can access on a network.

•If ACLs are not configured on the router, all packets passing through the router will be allowed onto all parts of the network.

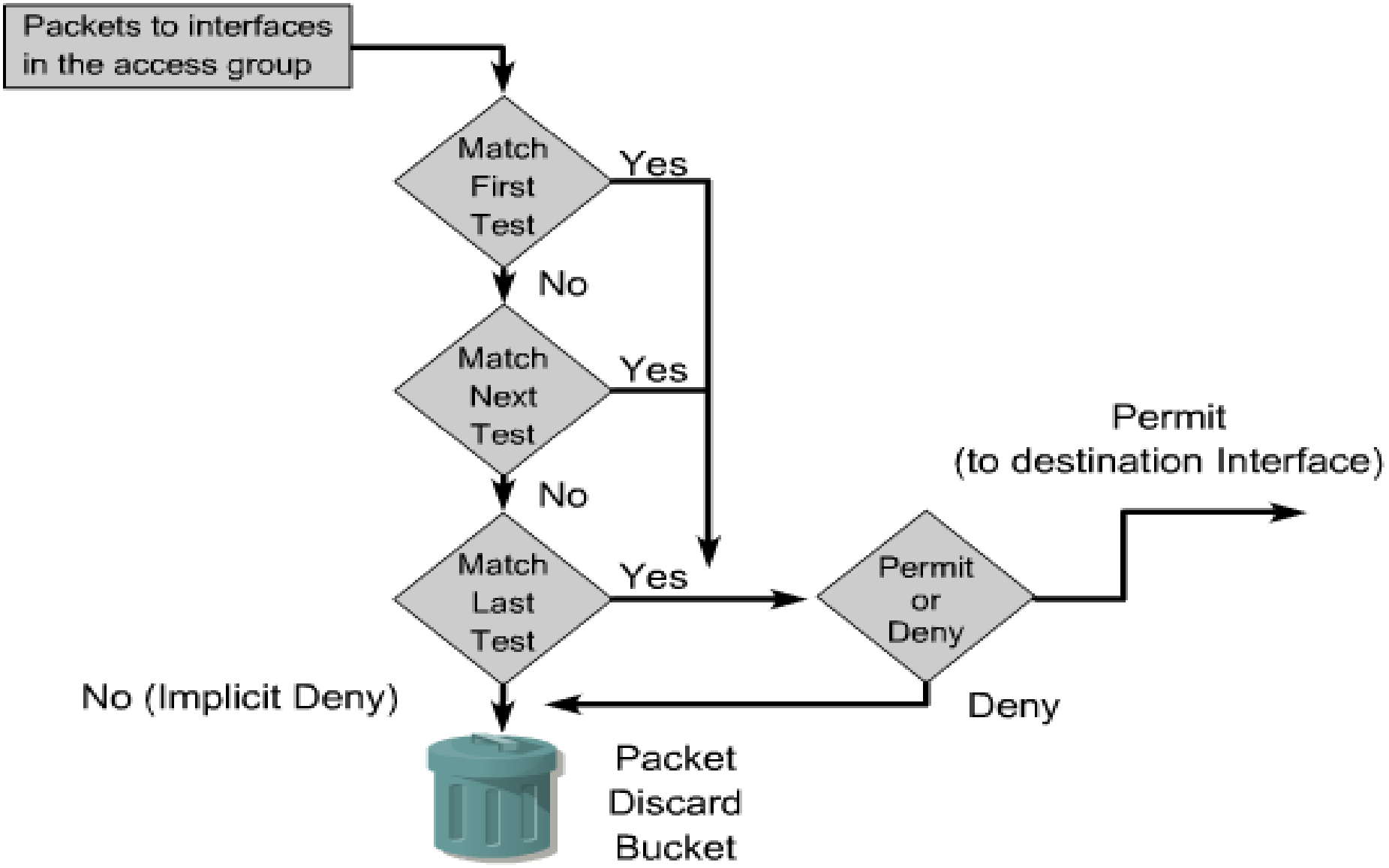
ACLs Filter Traffic Graphic



How ACLs Filter Traffic



How ACLs work.



Creating ACLs

ACLs are created in the global configuration mode.
There are many different types of ACLs including standard, extended, IPX, AppleTalk, and others.
When configuring ACLs on a router, each ACL must be uniquely identified by assigning a number to it.
This number identifies the type of access list created and must fall within the specific range of numbers that is valid for that type of list.

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

Since IP is by far the most popular routed protocol, addition ACL numbers have been added to newer router IOSs.
Standard IP: 1300-1999
Extended IP: 2000-2699

The access-list **command**

Define the ACL by using the following command:

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

The ip access-group **command**

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* { **in** | **out** }

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.

ACL Example

```
Router(config)#
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
interface ethernet 0
  ip access-group 2 in
```

Wildcard Mask Examples

5 Examples follow that demonstrate how a wildcard mask can be used to permit or deny certain IP addresses, or IP address ranges.

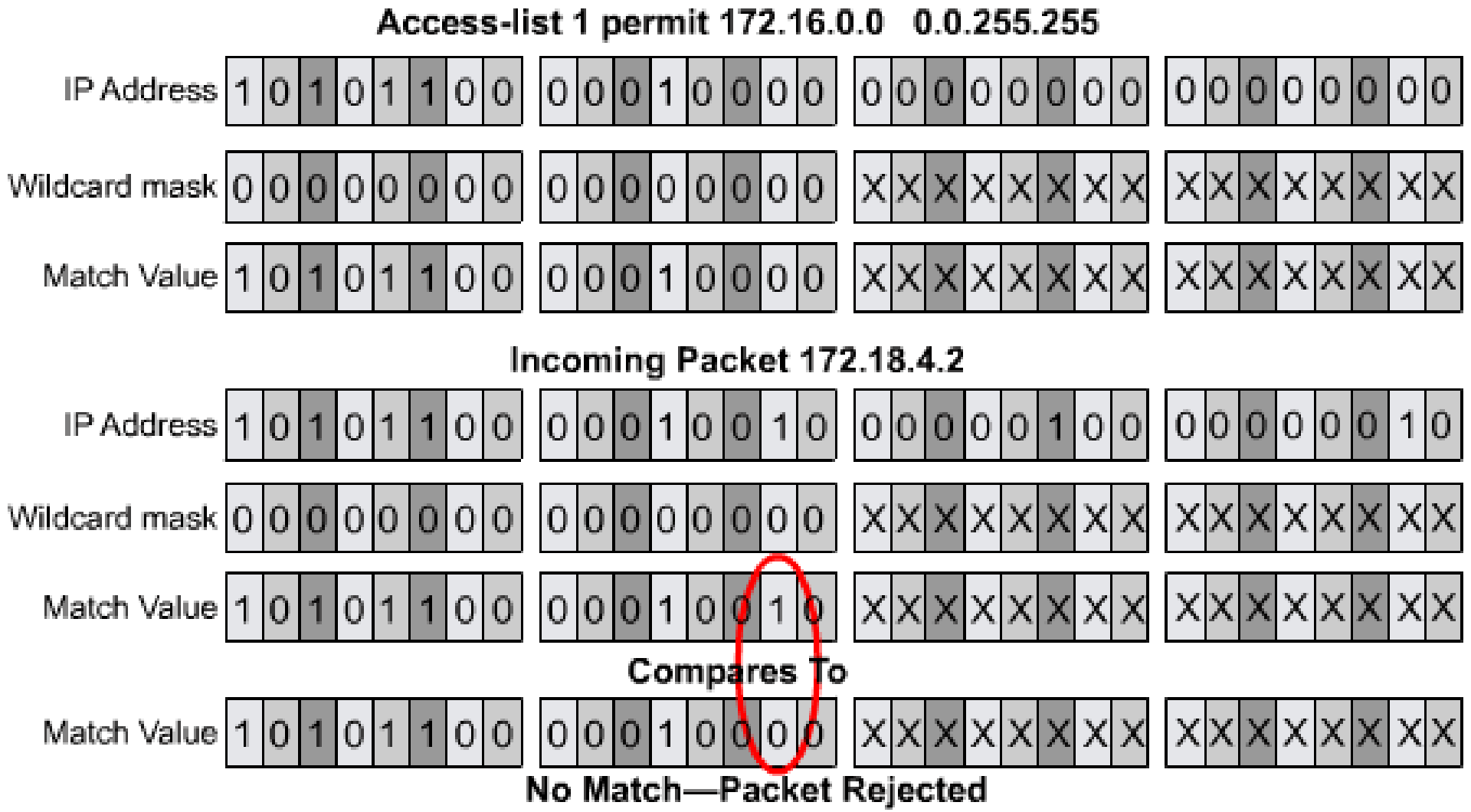
While subnet masks start with binary 1s and end with binary 0s, wildcard masks are the reverse meaning they typically start with binary 0s and end with binary 1s.

In the examples we represent the binary 1s in the wilcard masks with Xs to focus on the specific bits being shown in each example.

You will see that while subnet masks were ANDed with ip addresses, wildcard masks are ORed with IP addresses.

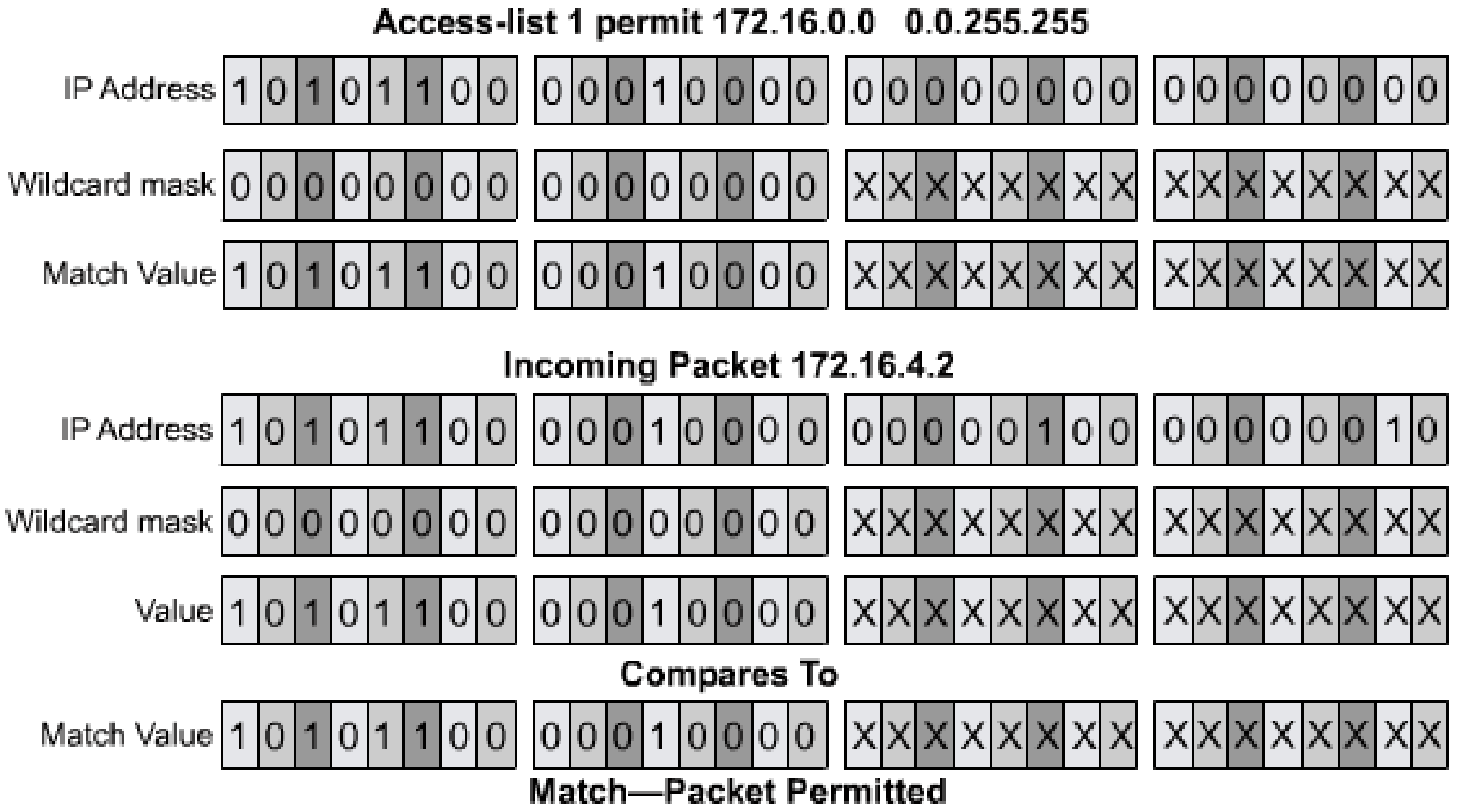
.

Wildcard Mask Example #1



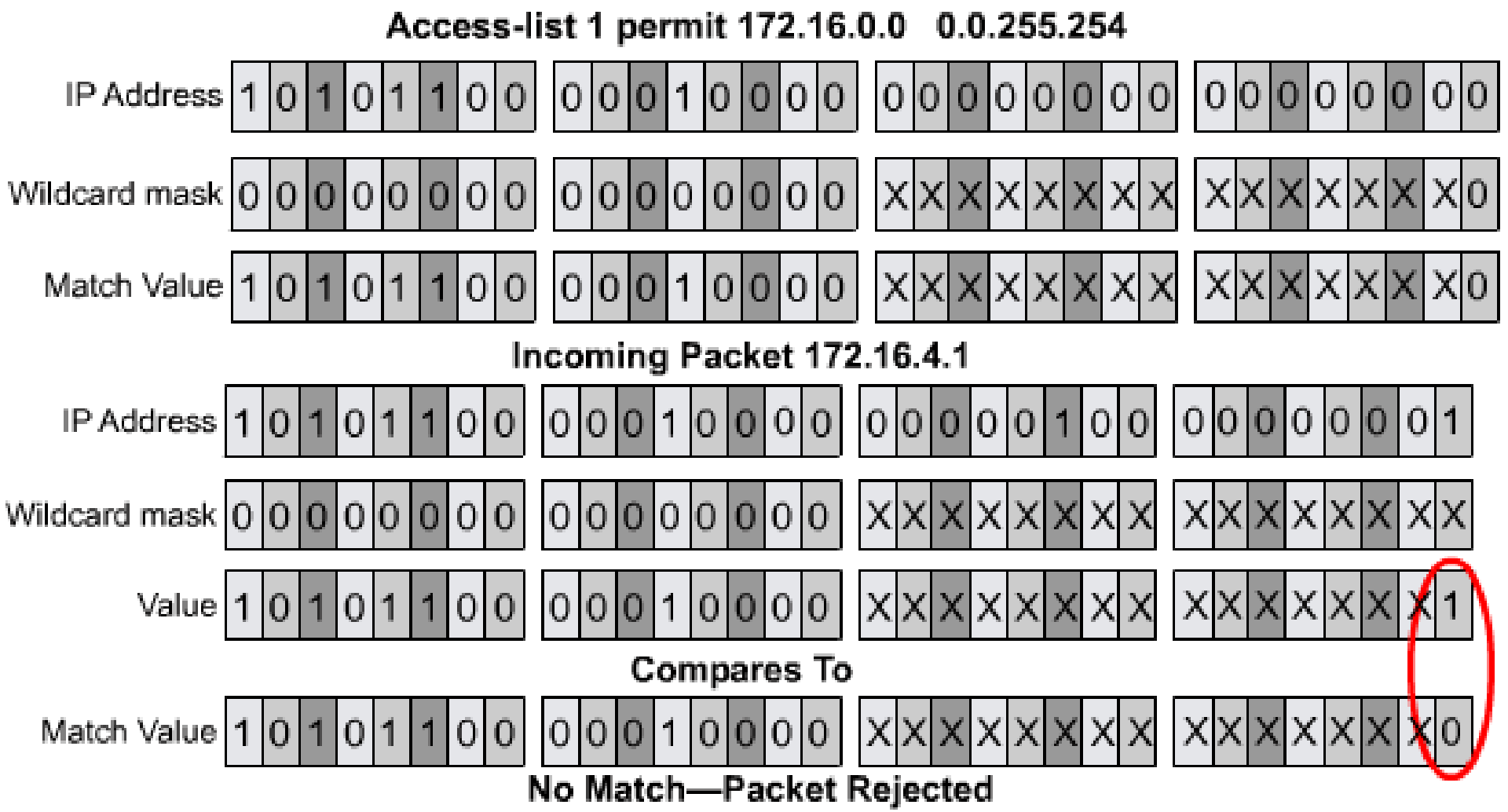
In this case, the two values do not match. In the comparison the second bit in the second octet of the two match values are different. This causes the packet to be rejected since it doesn't match.

Wildcard Mask Example #2



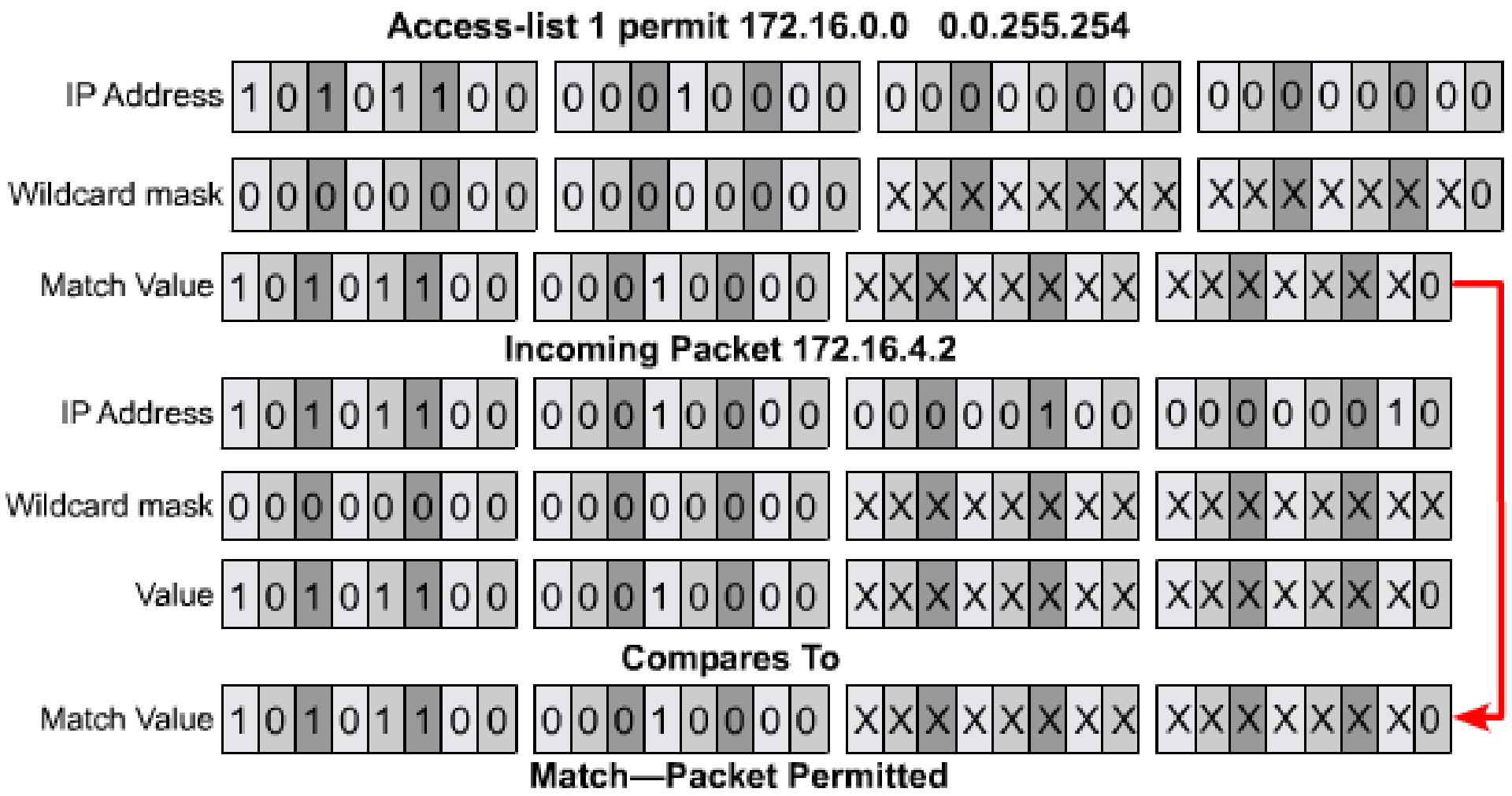
In this case, the two values match and the packet is permitted.

Wildcard Mask Example #3



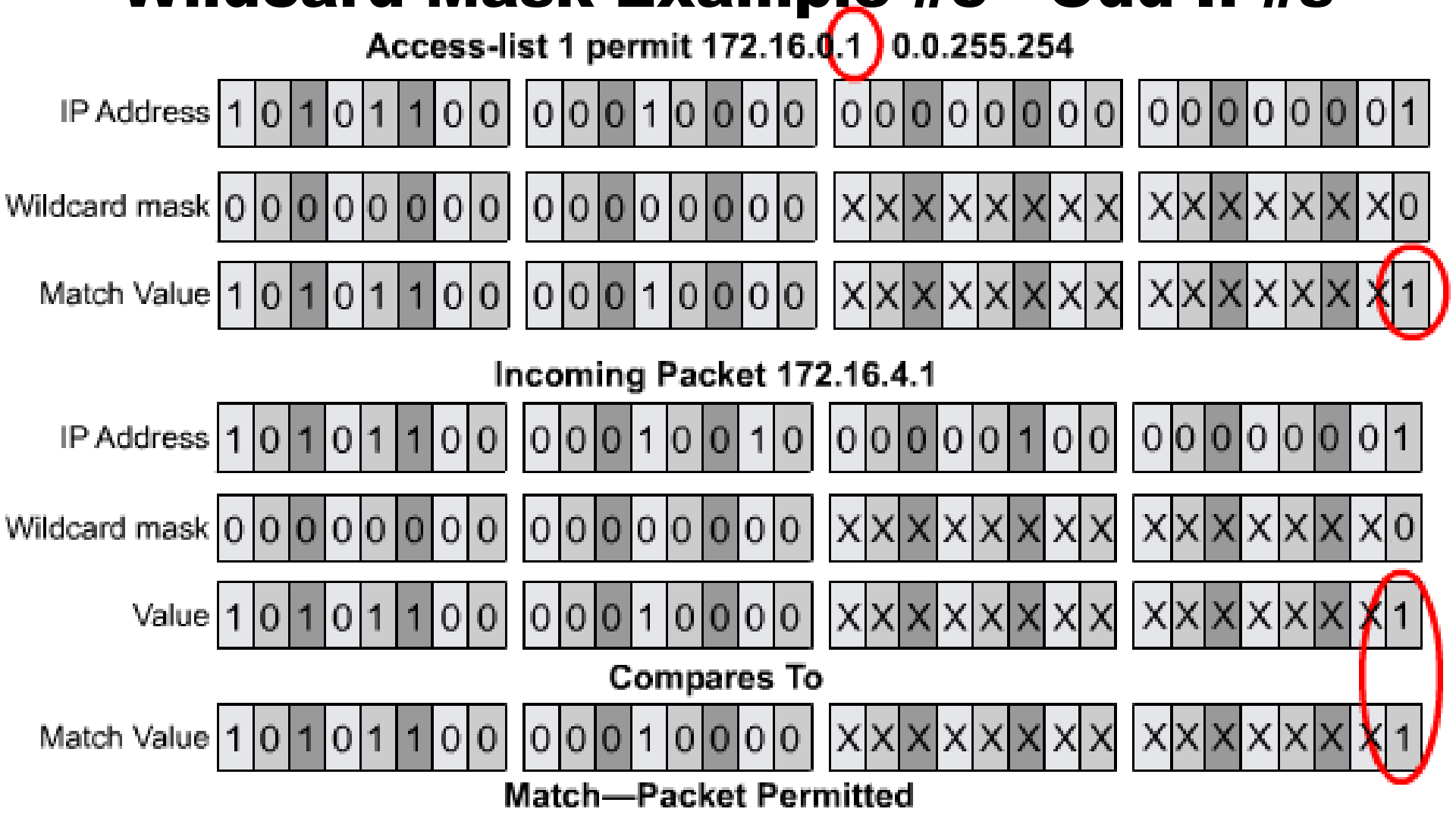
For this comparison the left most 16 bits match, but the rightmost bit does not. This causes the packet to be rejected. Remember that the match comparison must be an exact match.

Wildcard Mask Example #4 - Even IPs



With an even address the last bit position will always be zero. Since the composite value and the match value are the same the packet is accepted by the router. A question, how would the ACL statement above be changed to allow only the odd hosts in the address range of 172.16.0.0 to 172.16.255.255.

Wildcard Mask Example #5 - Odd IP#s



You permit only the odd addresses by just changing the IP address in the ACL statement to an odd number. This sets the right most bit to a one and only odd numbers will have that position be a one.

The any **and** host **Keywords**

```
Router(config)#access-list 1 permit 0.0.0.0 255.255.255.255
```

Can be written as:

```
Router(config)#access-list 1 permit any
```

```
Router(config)#access-list 1 permit 172.30.16.29 0.0.0.0
```

Can be written as:

```
Router(config)#access-list 1 permit host 172.30.16.29
```

This is the format of the any and host optional keywords in an ACL statement.

Verifying ACLs

There are many **show** commands that will verify the content and placement of ACLs on the router.

The **show ip interface** command displays IP interface information and indicates whether any ACLs are set.

The **show access-lists** command displays the contents of all ACLs on the router.

show access-list 1 shows just access-list 1.

The **show running-config** command will also reveal the access lists on a router and the interface assignment information.

Standard ACLs

Standard ACLs check the source address of IP packets that are routed.

The comparison will result in either permit or deny access for an entire protocol suite, based on the network, subnet, and host addresses.

The standard version of the **access-list** global configuration command is used to define a standard ACL with a number in the range of 1 to 99 (also from 1300 to 1999 in recent IOS).

If there is no wildcard mask. the default mask is used, which is 0.0.0.0.
(This only works with Standard ACLs and is the same thing as using **host**.)

The full syntax of the standard ACL command is:

Router(config)#**access-list** *access-list-number* {deny | permit} source [*source-wildcard*] [**log**]

The no form of this command is used to remove a standard ACL. This is the syntax:
Router(config)#**no access-list** *access-list-number*

Standard ACLs: Permitting/Denying Hosts, Networks and Subnets

Permitting a Single Host

Router(config)# **access-list 1 permit 200.100.50.23 0.0.0.0**
or
Router(config)# **access-list 1 permit host 200.100.50.23**
or
Router(config)# **access-list 1 permit 200.100.50.23**

(The implicit “deny any” ensures that everyone else is denied.)

Router(config)# **int e0**
Router(config-if)# **ip access-group 1 in**
or
Router(config-if)# **ip access-group 1 out**

Denying a Single Host

Router(config)# **access-list 1 deny 200.100.50.23 0.0.0.0**
Router(config)# **access-list 1 permit 0.0.0.0 255.255.255.255**
or
Router(config)# **access-list 1 deny host 200.100.50.23**
Router(config)# **access-list 1 permit any**

(The implicit “deny any” is still present, but totally irrelevant.)

Router(config)# **int e0**
Router(config-if)# **ip access-group 1 in**
or
Router(config-if)# **ip access-group 1 out**

Permitting a Single Network

Class C

Router(config)# **access-list 1 permit 200.100.50.0 0.0.0.255**

or

Class B

Router(config)# **access-list 1 permit 150.75.0.0 0.0.255.255**

or

Class A

Router(config)# **access-list 1 permit 13.0.0.0 0.255.255.255**

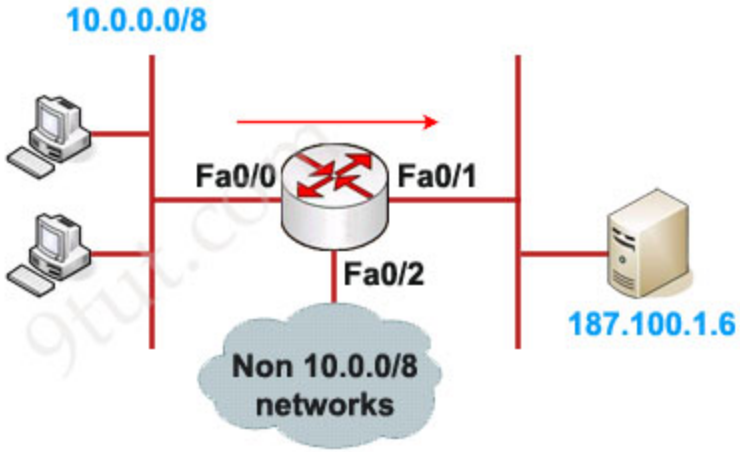
(The implicit “deny any” ensures that everyone else is denied.)

Router(config)# **int e0**

Router(config-if)# **ip access-group 1 in**

or

Router(config-if)# **ip access-group 1 out**



Configuration:

In this example we will define a standard access list that will only allow network 10.0.0.0/8 to access the server (located on the Fa0/1 interface)

Define which source is allowed to pass:

Router(config)#**access-list 1 permit 10.0.0.0 0.255.255.255**

(there is always an implicit deny all other traffic at the end of each ACL so we don't need to define forbidden traffic)

Apply this ACL to an interface:

Router(config)#**interface Fa0/1**

Router(config-if)#**ip access-group 1 out**

Denying a Single Network

Class C

Router(config)# **access-list 1 deny 200.100.50.0 0.0.0.255**

Router(config)# **access-list 1 permit any**

or

Class B

Router(config)# **access-list 1 deny 150.75.0.0 0.0.255.255**

Router(config)# **access-list 1 permit any**

or

Class A

Router(config)# **access-list 1 deny 13.0.0.0 0.255.255.255**

Router(config)# **access-list 1 permit any**

(The implicit “deny any” is still present, but totally irrelevant.)

Permitting a Class C Subnet

Network Address/Subnet Mask: 200.100.50.0/28

Desired Subnet: 3rd

Process:

32-28=4 2^4 = 16

1st Usable Subnet address range it 200.100.50.16-31

2nd Usable Subnet address range it 200.100.50.32-47

3rd Usable Subnet address range it 200.100.50.48-63

Subnet Mask is 255.255.255.240 Inverse Mask is 0.0.0.15

or subtract 200.100.50.48 from 200.100.50.63 to get 0.0.0.15

Router(config)# **access-list 1 permit 200.100.50.48 0.0.0.15**

(The implicit “deny any” ensures that everyone else is denied.)

Denying a Class C Subnet

Network Address/Subnet Mask: 192.68.72.0/27
Undesired Subnet: 2nd

Process:
32-27=5 2^5=32
1st Usable Subnet address range it 192.68.72.32-63
2nd Usable Subnet address range it 192.68.72.64-95

Subnet Mask is 255.255.255.224 Inverse Mask is 0.0.0.31
or subtract 192.68.72.64 from 192.68.72.95 to get 0.0.0.31

Router(config)# **access-list 1 deny 192.68.72.64 0.0.0.31**
Router(config)# **access-list 1 permit any**

(The implicit “deny any” is still present, but totally irrelevant.)

Permitting a Class B Subnet

Network Address/Subnet Mask: 150.75.0.0/24
Desired Subnet: 129th

Process:
Since exactly 8 bits are borrowed the 3rd octet will denote the subnet number.
129th Usable Subnet address range it 150.75.129.0-255

Subnet Mask is 255.255.255.0 Inverse Mask is 0.0.0.255
or subtract 150.75.129.0 from 150.75.129.255 to get 0.0.0.255

Router(config)# **access-list 1 permit 150.75.129.0 0.0.0.255**

(The implicit “deny any” ensures that everyone else is denied.)

Denying a Class B Subnet

Network Address/Subnet Mask: 160.88.0.0/22
Undesired Subnet: 50th

Process:
32-22=10 (more than 1 octet) 10-8=2 2^2=4
1st Usable Subnet address range it 160.88.4.0-160.88.7.255
2nd Usable Subnet address range it 160.88.8.0-160.88.11.255

50 * 4 = 200 50th subnet is 160.88.200.0-160.88.203.255

Subnet Mask is 255.255.252.0 Inverse Mask is 0.0.3.255
or subtract 160.88.200.0 from 160.88.203.255 to get 0.0.3.255

Router(config)# **access-list 1 deny 160.88.200.0 0.0.3.255**
Router(config)# **access-list 1 permit any**

Permitting a Class A Subnet

Network Address/Subnet Mask: 111.0.0.0/12

Desired Subnet: 13th

Process:

32-12=20 20-16=4 2^4=16

1st Usable Subnet address range is 111.16.0.0-111.31.255.255

13*16=208

13th Usable Subnet address range is 111.208.0.0-111.223.255.255

Subnet Mask is 255.240.0.0 Inverse Mask is 0.15.255.255

or subtract 111.208.0.0 from 111.223.255.255 to get 0.15.255.255

Router(config)# **access-list 1 permit 111.208.0.0 0.15.255.255**

(The implicit “deny any” ensures that everyone else is denied.)

Denying a Class A Subnet

Network Address/Subnet Mask: 40.0.0.0/24
Undesired Subnet: 500th

Process:
Since exactly 16 bits were borrowed the 2nd and 3rd octet will denote the subnet.

1st Usable Subnet address range is 40.0.1.0-40.0.1.255
255th Usable Subnet address range is 40.0.255.0-40.0.255.255
256th Usable Subnet address range is 40.1.0.0-40.1.0.255
300th Usable Subnet address range is 40.1.44.0-40.1.44.255
500th Usable Subnet address range is 40.1.244.0-40.1.244.255

Router(config)# **access-list 1 deny 40.1.244.0 0 0.0.0.255**
Router(config)# **access-list 1 permit any**

Standard ACLs: Permitting/Denying Ranges of Addresses.

Permit 200.100.50.1,5,13,29,42,77

access-list 1 permit host 200.100.50.1

access-list 1 permit host 200.100.50.5

access-list 1 permit host 200.100.50.13

access-list 1 permit host 200.100.50.29

access-list 1 permit host 200.100.50.42

access-list 1 permit host 200.100.50.77

Sometimes a group of addresses has no pattern and the best way to deal with them is individually.

(The implicit “deny any” ensures that everyone else is denied.)

Permit 200.100.50.16-127

access-list 1 deny 200.100.50.0 0.0.0.15 *(0-15)*

access-list 1 permit 200.100.50.0 0.0.0.127 *(0-127)*

First we make sure that addresses 0-15 are denied.

Then we can permit any address in the range 0-127.

Since only the first matching statement in an ACL is applied an address in the range of 0-15 will be denied by the first statement before it has a chance to be permitted by the second.

(The implicit “deny any” ensures that everyone else is denied.)