ICT2203/ICT2203X

Network Security

SINGAPORE INSTITUTE OF TECHNOLOGY

Lab 4 Notes:

Network Traffic Filtering with Access Control List (ACL)

2021-2022 Trimester 3

In this lab you will experiment with access control list

SINGAPORE INSTITUTE OF TECHNOLOGY

4.1 Standard numbered ACL is used to filter network traffic based on source IP addresses.

4.2 Extended numbered ACL is more powerful as it can filter network traffic based on protocol type, source/destination IP addresses, and source/destination port numbers.

4.3 Named ACL enables ACL to be configured with meaningful name to make it easier to remember.

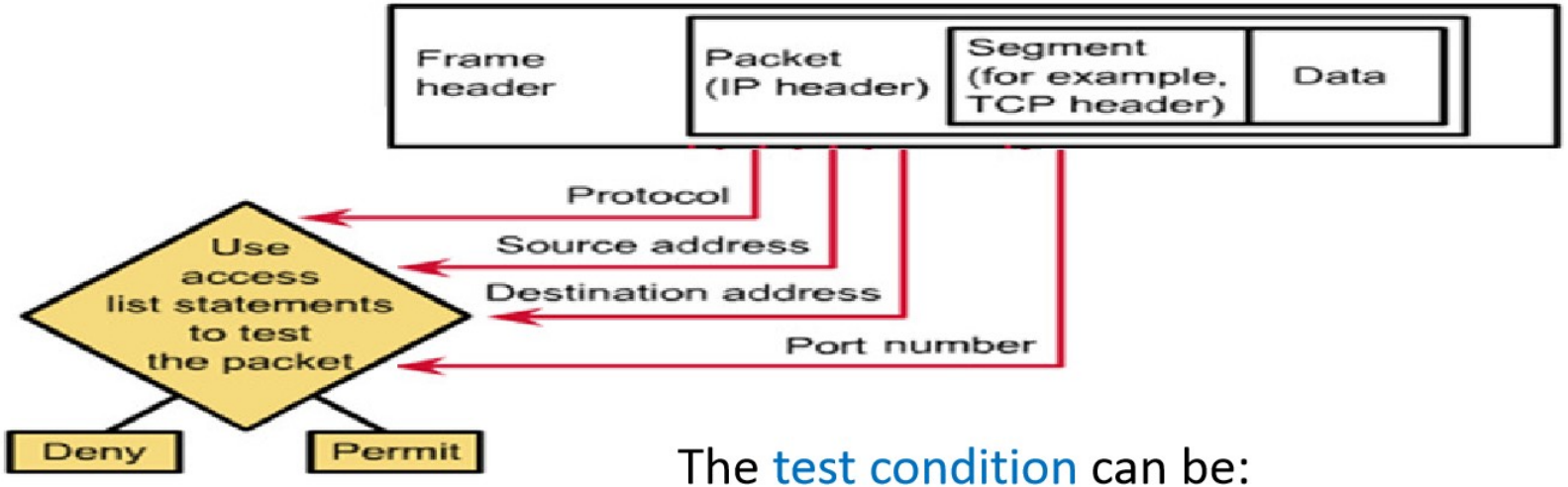
4.4 Time-based ACL enhances ACL with time range to automatically turn on/off ACL which can be useful in practice.

4.5 Object group-based ACL is a new way of writing ACL to make it more readable and manageable.

An access control list (ACL) is a sequential list of one or more ACL statement, also called access control entry (ACE) to filter network traffic.

Each ACL statement or ACE contains two components:

- Action to be taken
- When test condition is met

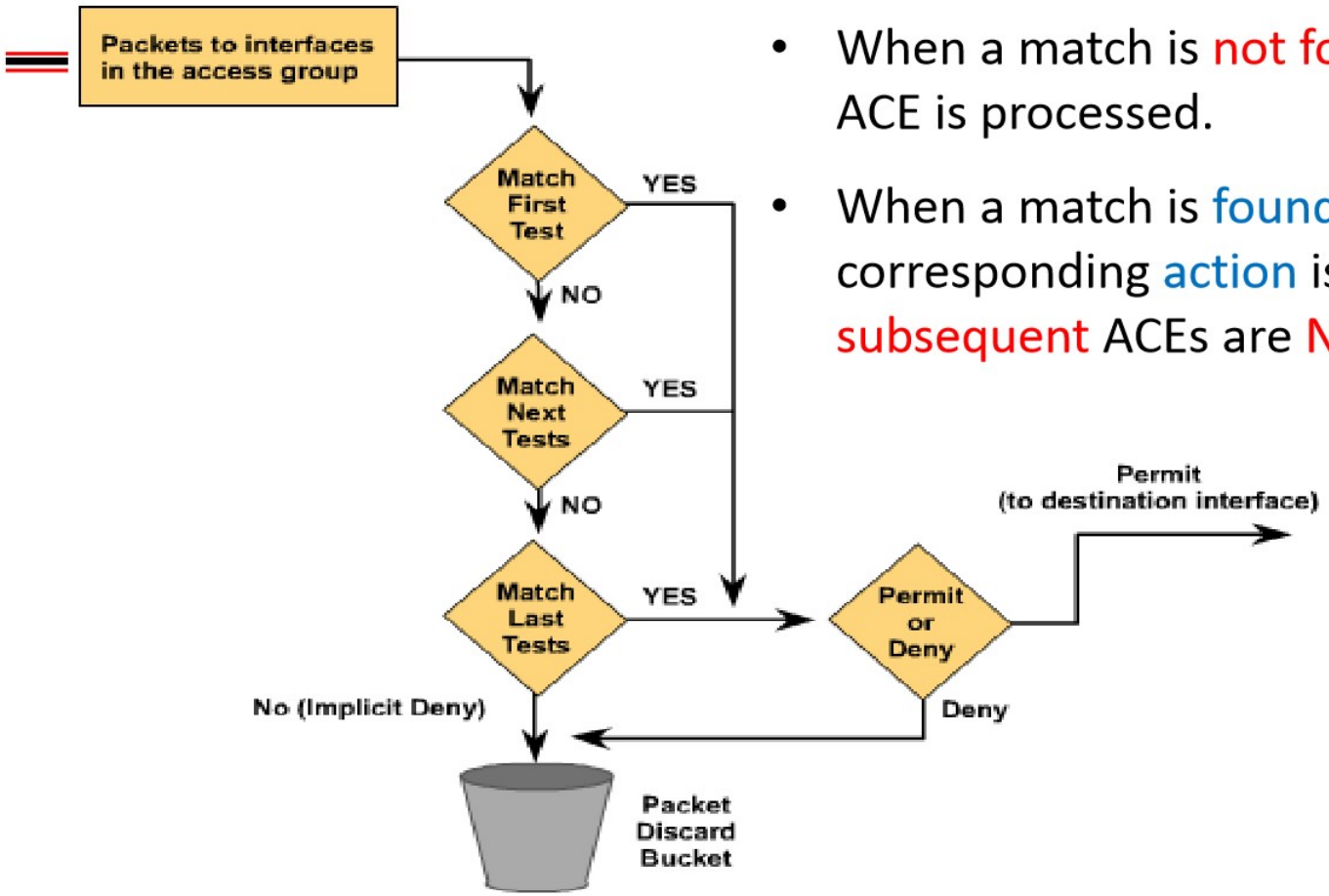


The action is either permit or deny the network traffic when condition is met.

The test condition can be:

- Standard – matches source IP address
- Extended – matches source IP address, destination IP address, protocol type, port number.

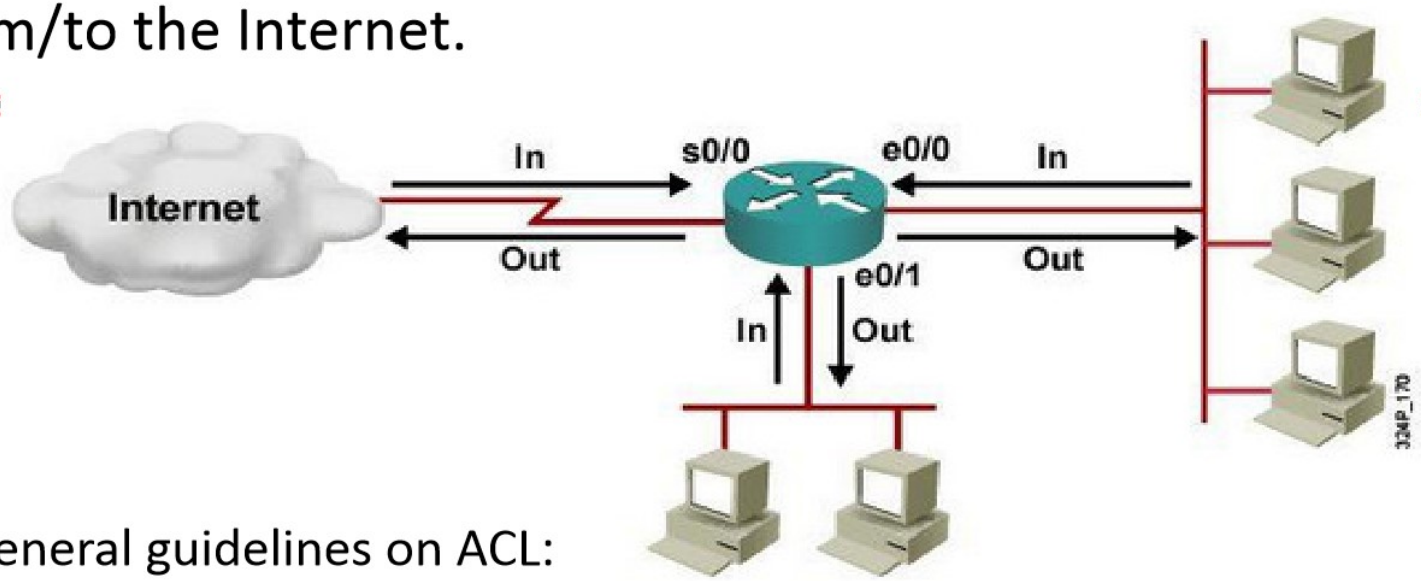
The ACEs are processed from top to bottom in order, and hence the sequence of ACEs in an ACL is important.



- When a match is not found, the next ACE is processed.
- When a match is found, the corresponding action is taken, and subsequent ACEs are NOT processed.

- If no match is found after processing all ACEs, there is a default deny statement at the end to drop the traffic.

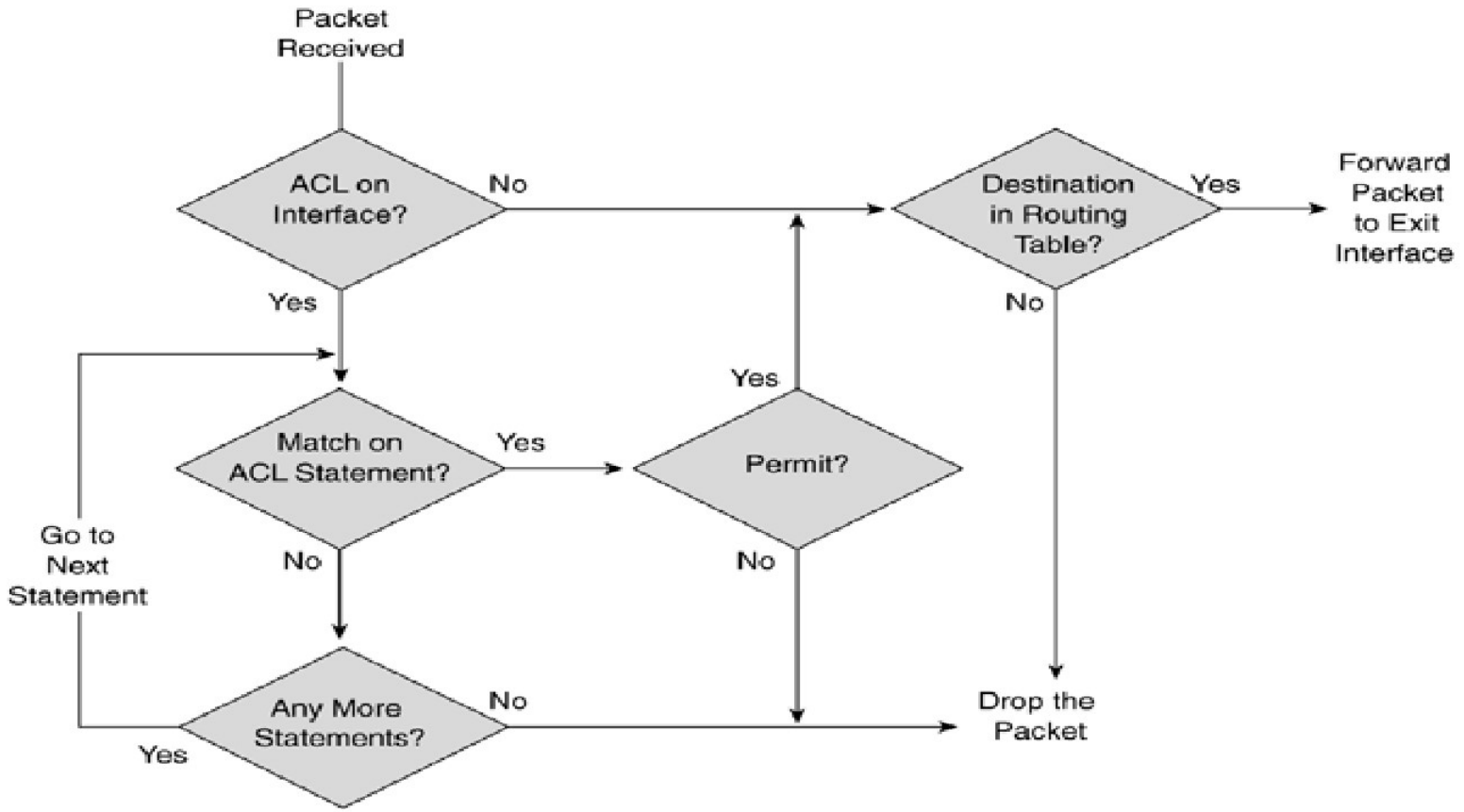
In practice, ACL are commonly applied on the switch SVIs or router interfaces to filter ingress or egress IP packets, especially from/to the Internet.



General guidelines on ACL:

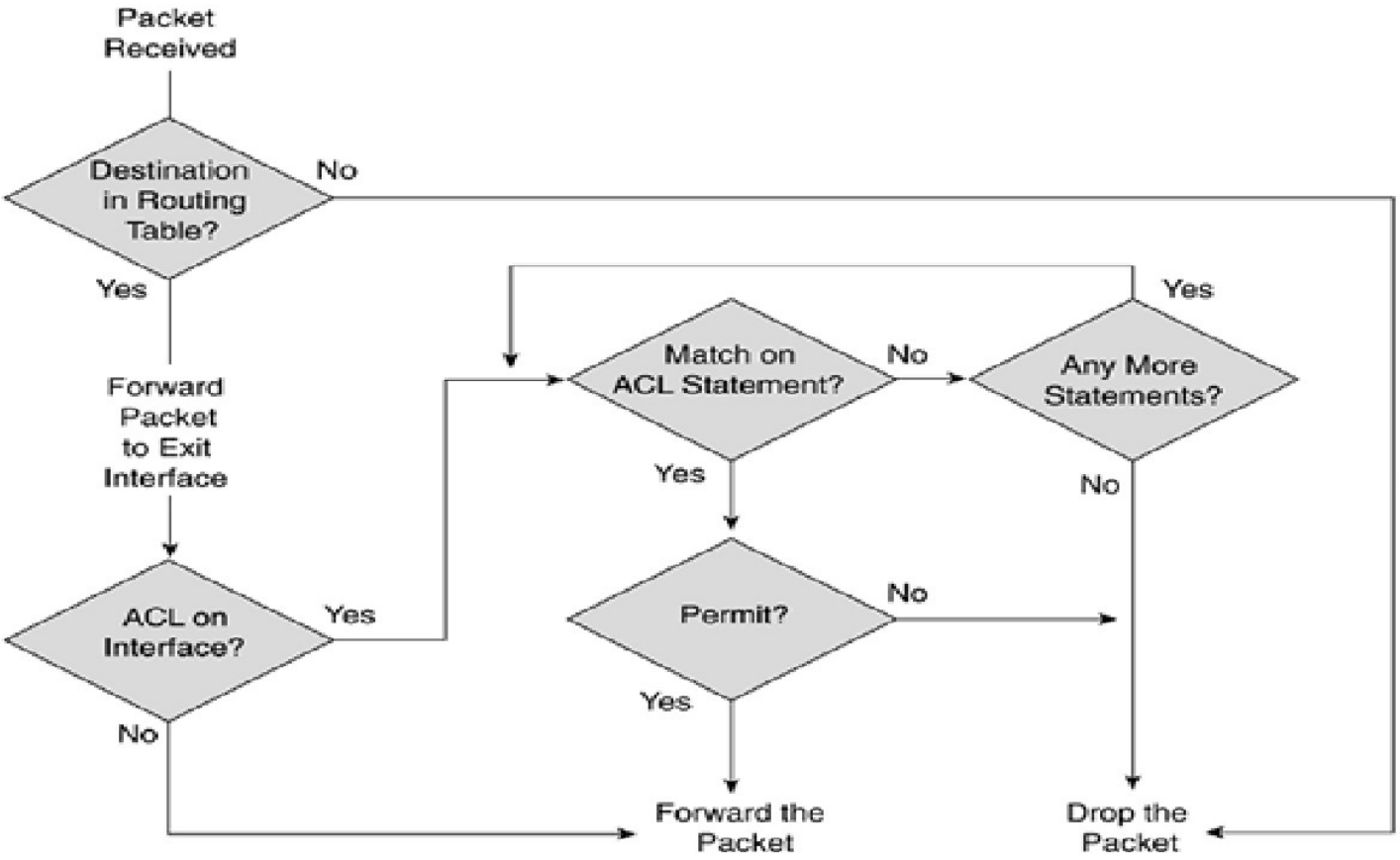
- One ACL per direction: ACL can only filter packets in one direction as configured. To filter both ingress and egress traffic, two separate ACLs must be written.
- One ACL per interface: Only one in ACL and one out ACL can be applied on each interface.
- One ACL per protocol: Separate ACLs must be written to filter IPv4 and IPv6 packets (not covered in this module)

For inbound ingress-filtering ACL, a router will ① filter all IP packets received on that interface, and ② only the permitted packets are then processed and forwarded.



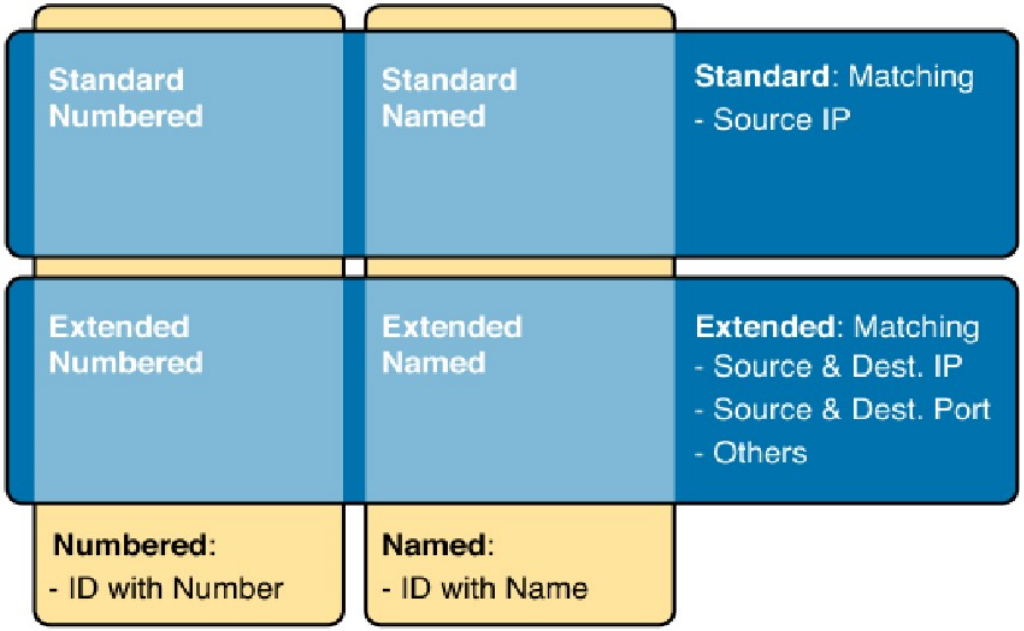
Note: Packets destined to the router itself can be filtered by appropriate inbound ACLs.

In contrast, for outbound egress-filtering ACL, ① only after forwarding decision has been made, ② IP packets are then filtered before permitted to be sent out from that interface.



Note: Packets originated from the router itself are not filtered by outbound ACLs.

In Cisco network devices, ACLs are divided into standard and extended ACLs, and may be identified by a number or name, called numbered or named ACLs respectively.

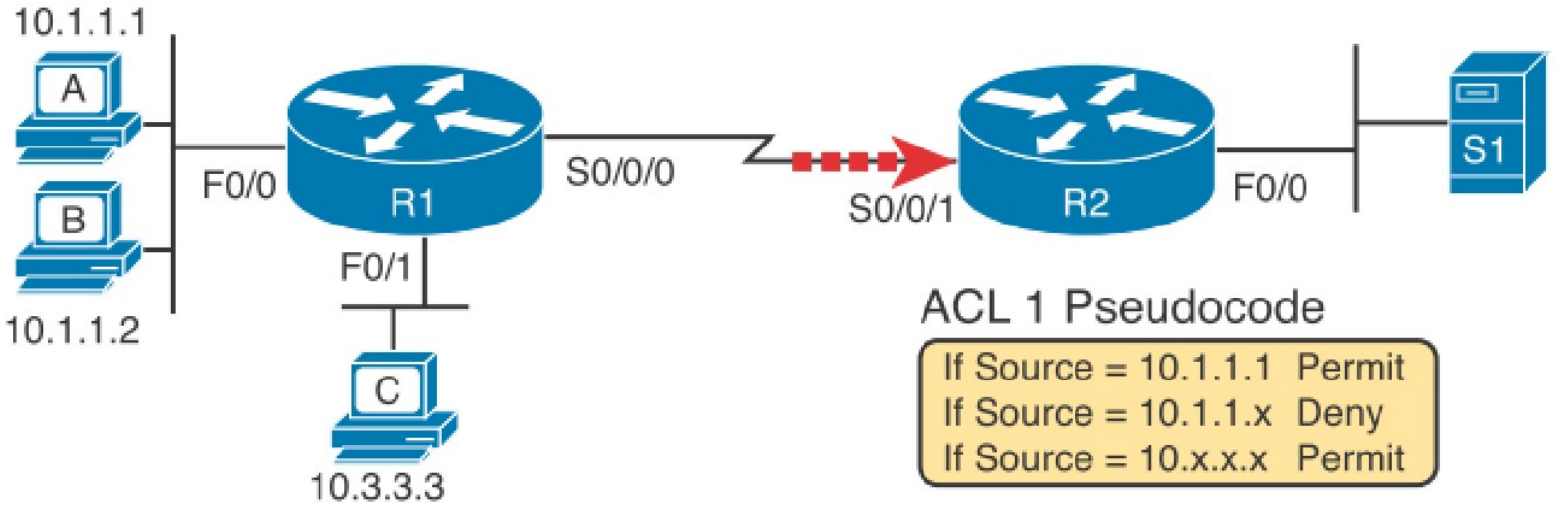


- Standard ACLs filter at layer 3 based on source IP only.
- Extended ACLs filter at layer 3 and 4 based on more attributes.

- Standard numbered ACLs is configured using any number from 1 – 99 or 1300 – 1999.
- Extended numbered ACLs is configured using any number from 100 – 199 or 2000 – 2699.
- Named ACLs is configured using any alphanumeric characters.

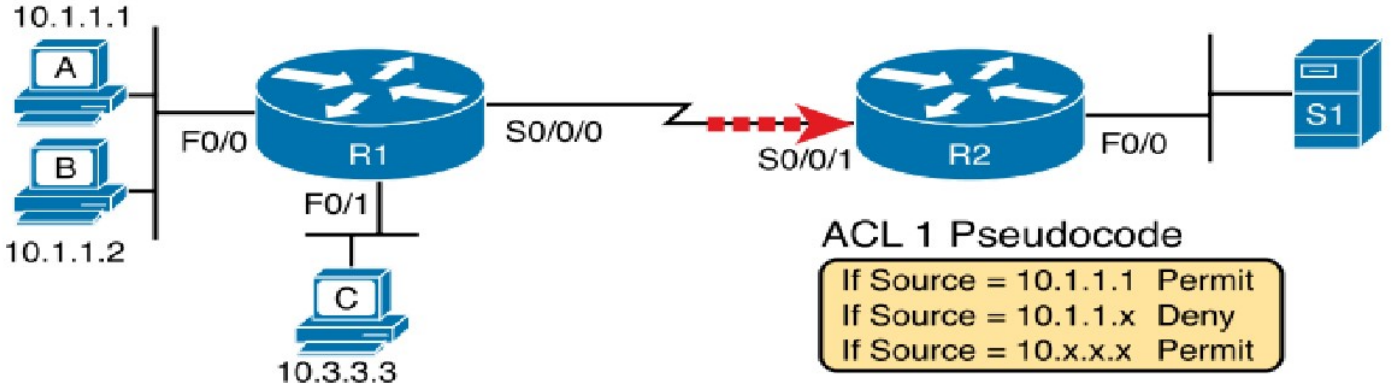
(4.1) Standard numbered ACL for controlling access to a network or end host/server.

E.g.: A company's security policy allows any users at subnet C but only one user at host A to access server S1. How can you implement this?



Typically, ACL is written with more specific ACEs (in terms of IP addresses) at the top and more general ACEs at the bottom.

An example of standard numbered ACL for implementing the required security policy in the previous slide with explanations of the configuration commands in the following slides.




```
R2(config)# access-list 1 permit 10.1.1.1
R2(config)# access-list 1 deny 10.1.1.0 0.0.0.255
R2(config)# access-list 1 permit 10.0.0.0 0.255.255.255
R2(config)# interface S0/0/1
R2(config-if)# ip access-group 1 in
```

Note: Standard ACLs can only filter based on source IP. To avoid unintentionally filtering more packets than necessary, it is recommended to be implemented near the destination; e.g. at router R2 instead of R1.

In general, numbered ACL is written by (1) specifying the related ACEs using the same access-list-number, and (2) applying the complete ACL on the required interface.

1) Each ACE is configured using an access-list command:
   - 'remark' is optional but helps you remember what the ACL statements are for
   - 'log' is optional which will generate a log message if matches

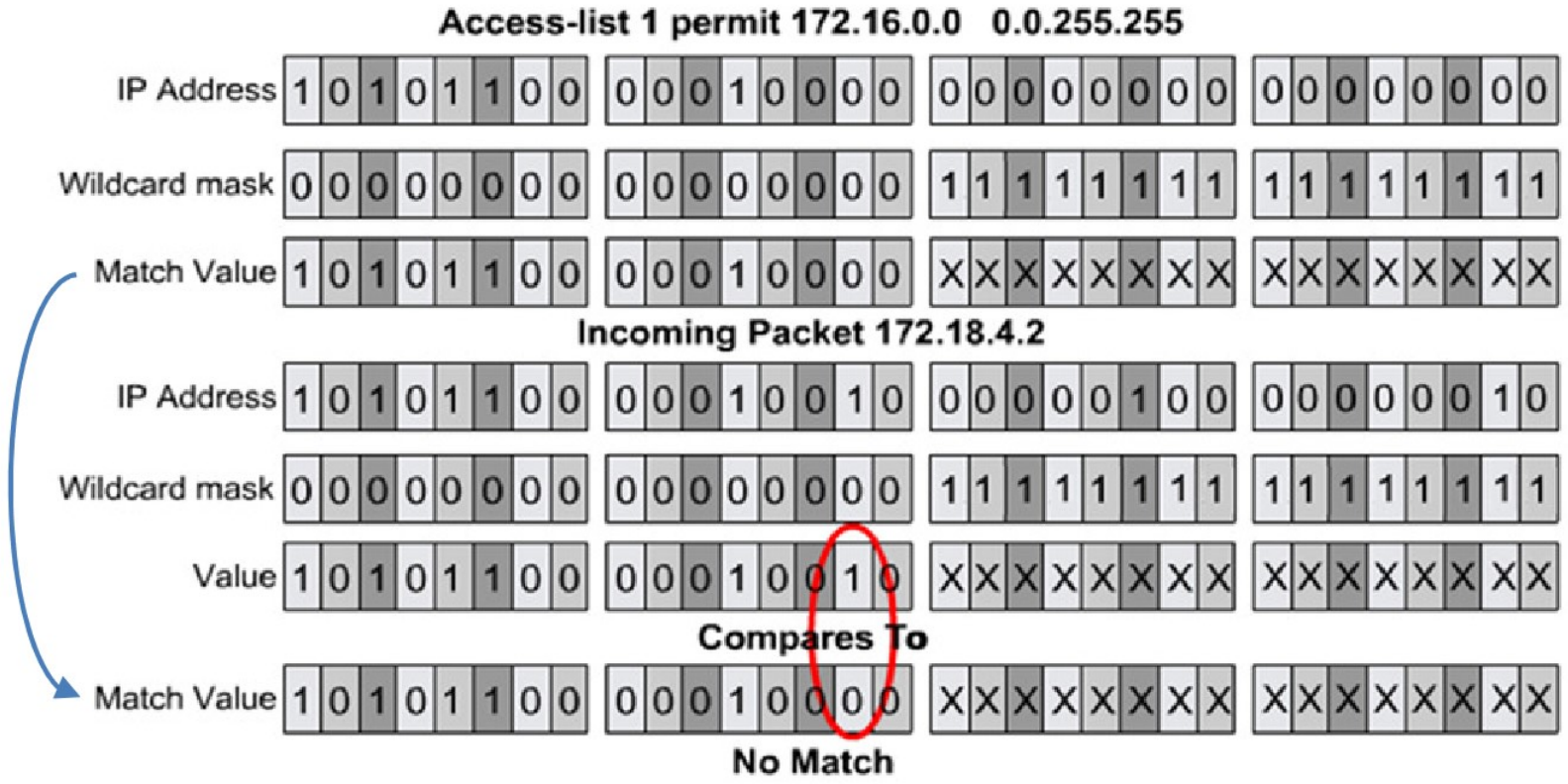
Step 1	`Router(config)#access-list access-list-number remark test` `Router(config)#access-list access-list-number {permit | deny} {test-conditions} log`

2) Apply the ACL onto the required interface and specify whether to filter inbound or outbound packets

Step 2	`Router(config-if)#ip access-group access-list-number {in | out}`

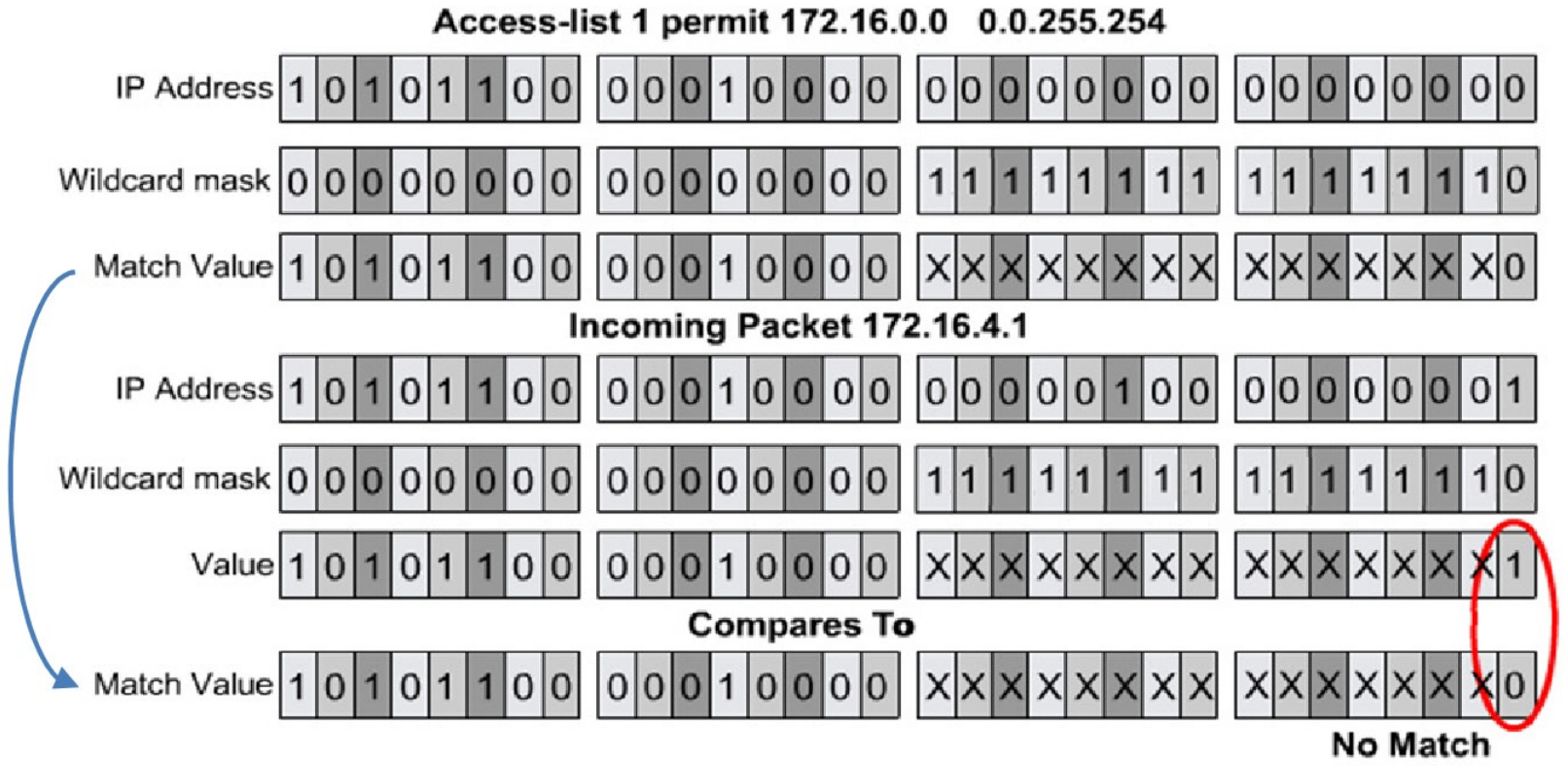
The test-conditions for standard ACL statements are source IP addresses specified with wildcard mask similar as configuring OSPF and NAT in ICT1010.

E.g.: The following ACL with wildcard mask 0.0.255.255 represents IP addresses from 172.16.0.0 – 172.16.255.255. Thus an IP packet with IP 172.18.4.2 will not match the test condition.



In fact, wildcard mask can be used to specify any bit patterns of the IP address to be matched. An example as follows:

E.g.: The following ACL with wildcard mask 0.0.255.254 denotes even-numbered IP addresses from 172.16.0.0 – 172.16.255.255. Thus an IP packet with IP 172.16.4.1 will not match.



In addition, there are two special wildcard masks for matching all IP addresses and matching only a single IP address.

Matching all addresses can be simplified using the keyword 'any':

```
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
```

Matching one single address can be simplified using 'host' keyword:

```
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
```

Note: 'host' keyword is not required for newer versions of router IOS.

To verify that the ACL is applied correctly at the interface to filter packets, use the following command:

```
R2# show ip interface s0/0/1
Serial0/0/1 is up, line protocol is up
  Internet address is 10.1.2.2/24
  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
  Multicast reserved groups joined: 224.0.0.9
  Outgoing access list is not set
  Inbound  access list is 1
! Lines omitted for brevity
```

You may also view the statistics of the number of matches of each ACL statement as follows, which can be useful for network security monitoring.

```
R2# show ip access-lists
Standard IP access list 1
    10 permit 10.1.1.1 (107 matches)
    20 deny   10.1.1.0, wildcard bits 0.0.0.255 (4 matches)
    30 permit 10.0.0.0, wildcard bits 0.255.255.255 (10 matches)
```

The number of matches in () at the end of each ACL statement is the count of IP packets that matches it so far.

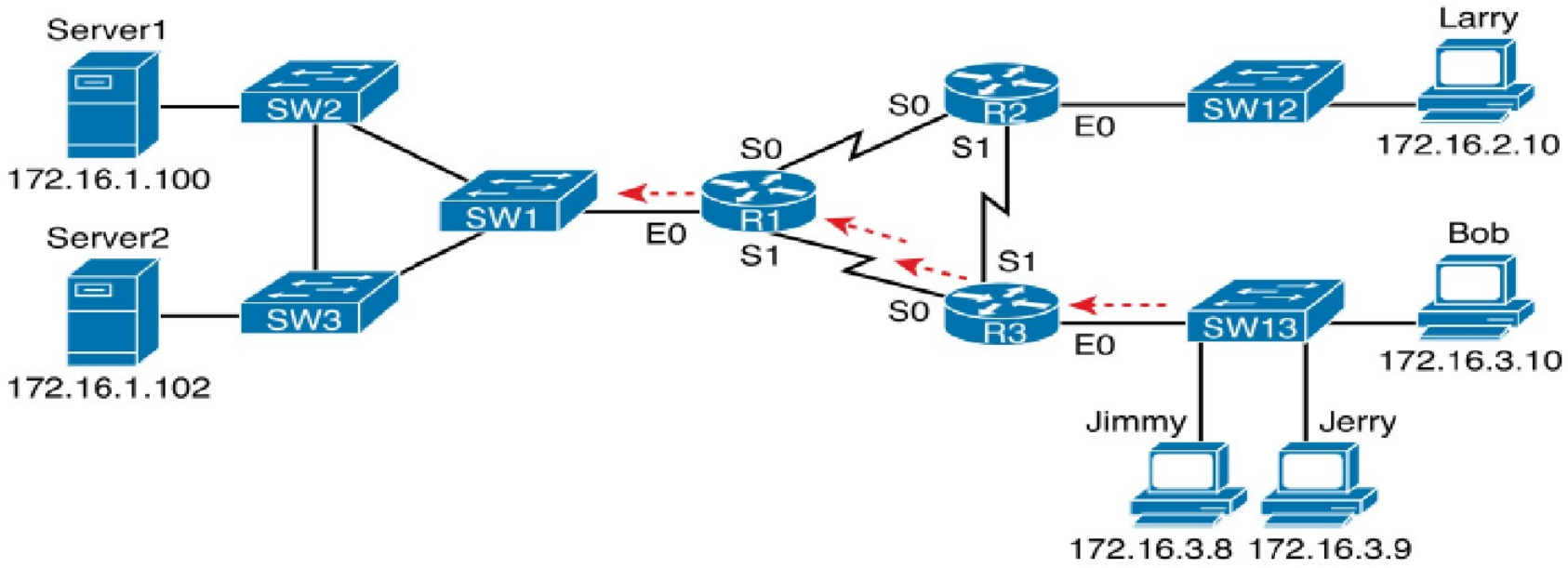
Note: To have a count of 'deny any' at end of ACL, you need to explicitly add 'deny any' to the end of ACL instead of using the default.

To clear the count, use the command:

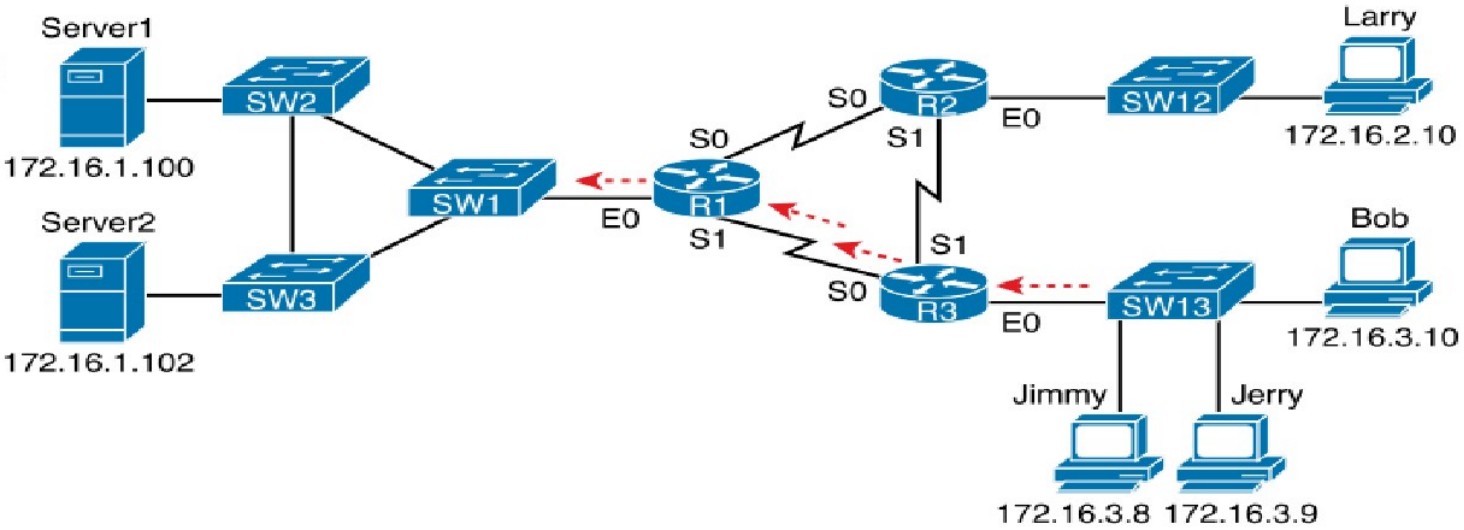
```
R2# clear access-list counters
```

(4.2) More powerful extended numbered ACL for controlling access to networks or end hosts/servers.

E.g.: Server 1 and 2 host both web servers and tftp servers. The company's security policy allows Bob to access web servers but not tftp servers. In addition, Larry is not allowed to access web server at Server 1. How can you implement this?



An example of extended numbered ACL for implementing the required security policy in the previous slide.




```
R1(config)# access-list 101 remark Stop Bob to TFTP servers, and Larry to Server1 web
R1(config)# access-list 101 deny udp host 172.16.3.10 172.16.1.0 0.0.0.255 eq tftp
R1(config)# access-list 101 deny tcp host 172.16.2.10 host 172.16.1.100 eq www
R1(config)# access-list 101 permit ip any any
R1(config)# interface Ethernet0
R1(config-if)# ip access-group 101 out
```

- The permit statement is to allow other traffic to pass through. Otherwise, the default deny any any will block all traffic.

Alternatively, two extended numbered ACLs may be written for implementing the required security policy in the previous slide as follows:

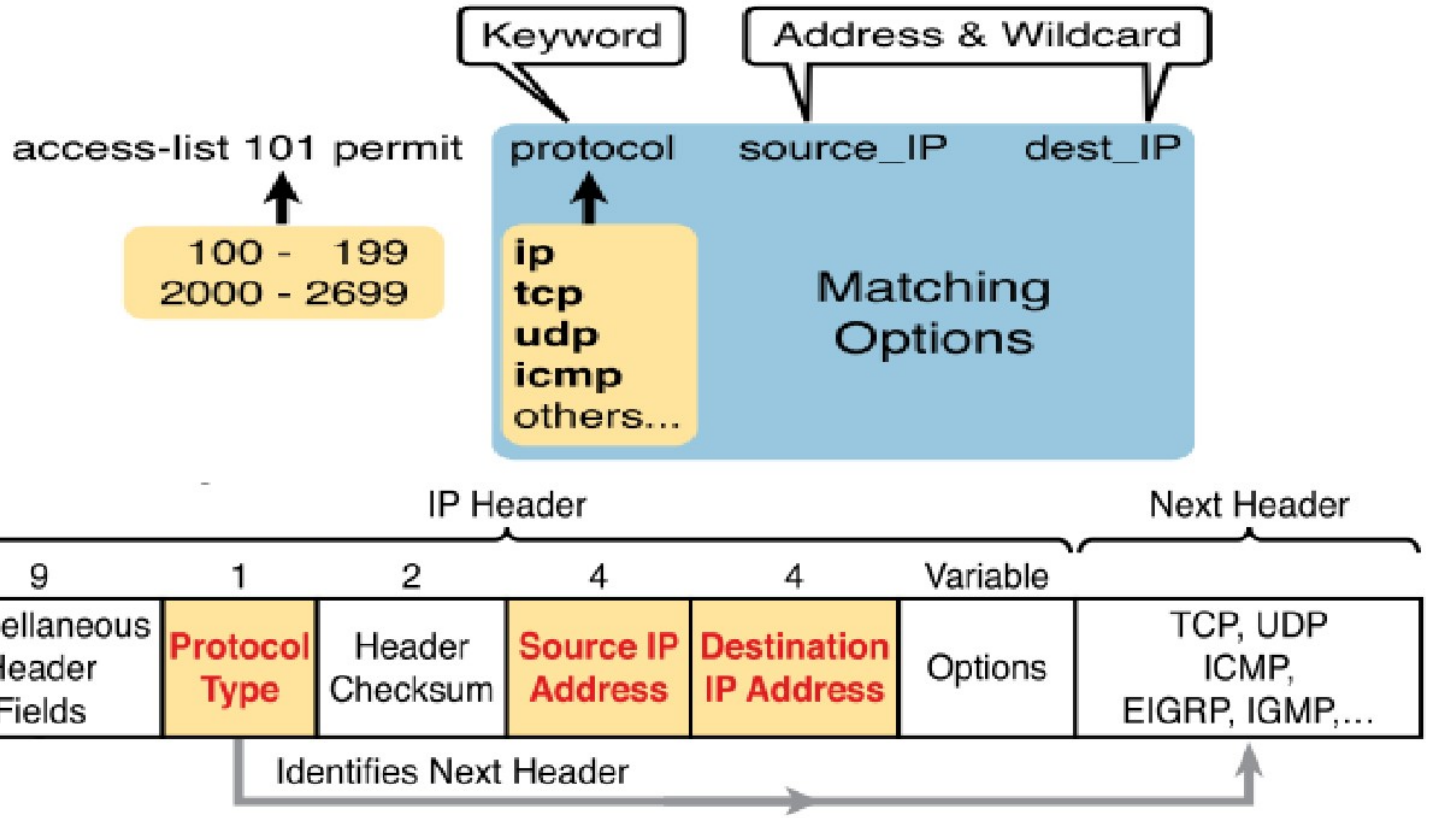
```
R2(config)# access-list 102 remark Stop Larry to Server1 web
R2(config)# access-list 102 deny tcp host 172.16.2.10 host 172.16.1.100 eq www
R2(config)# access-list 102 permit ip any any
R2(config)# interface Ethernet0
R2(config-if)# ip access-group 102 in
```

```
R3(config)# access-list 103 remark Stop Bob to TFTP servers
R3(config)# access-list 103 deny udp host 172.16.3.10 172.16.1.0 0.0.0.255 eq tftp
R3(config)# access-list 103 permit ip any any
R3(config)# interface Ethernet0
R3(config-if)# ip access-group 103 in
```

Note: Since extended ACLs can filter based on destination addresses, it is recommend to be configured near the source of packets to avoid wasting bandwidth to forward packets and then drop at the end.

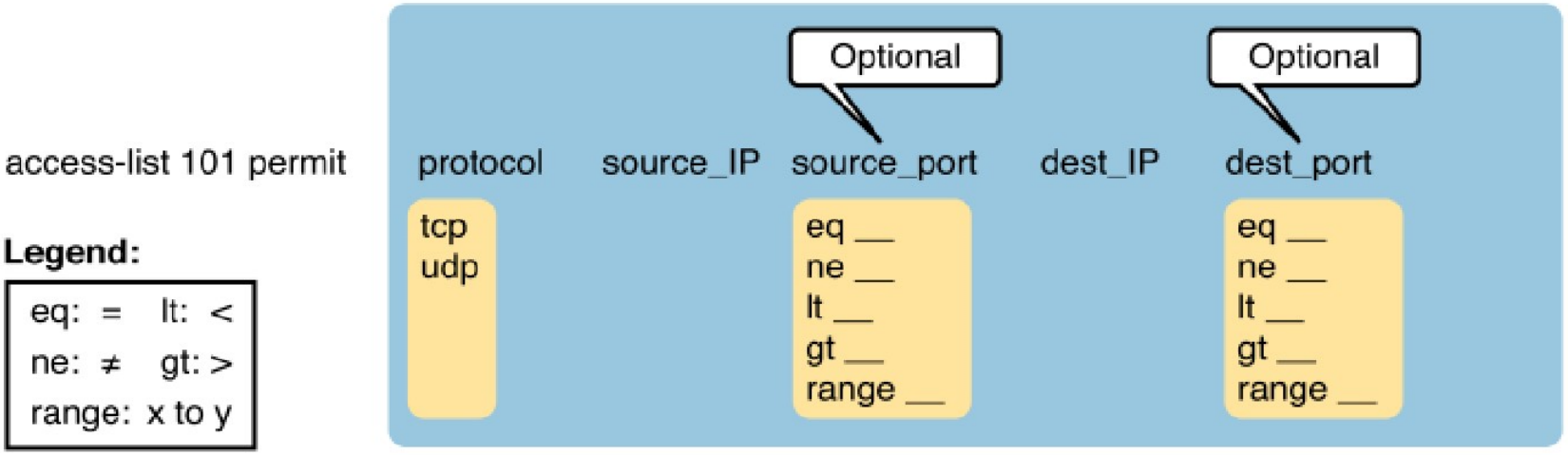
In general, extended ACLs are similar as standard ACLs except that the test-conditions are more complex but useful because they can filter based on more data fields.

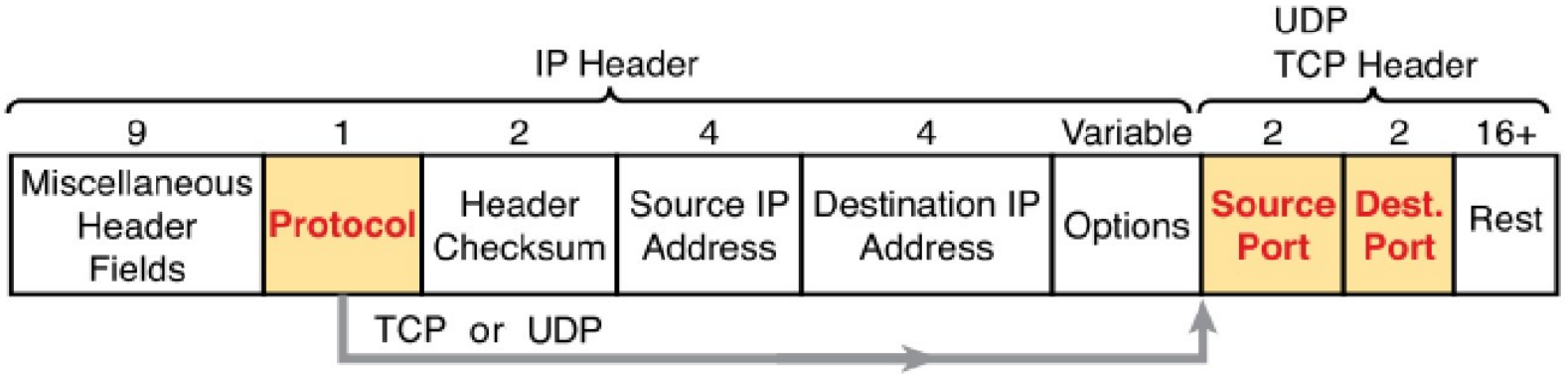
At a minimum, the test-conditions must specify protocol type, source IP address and destination IP address:



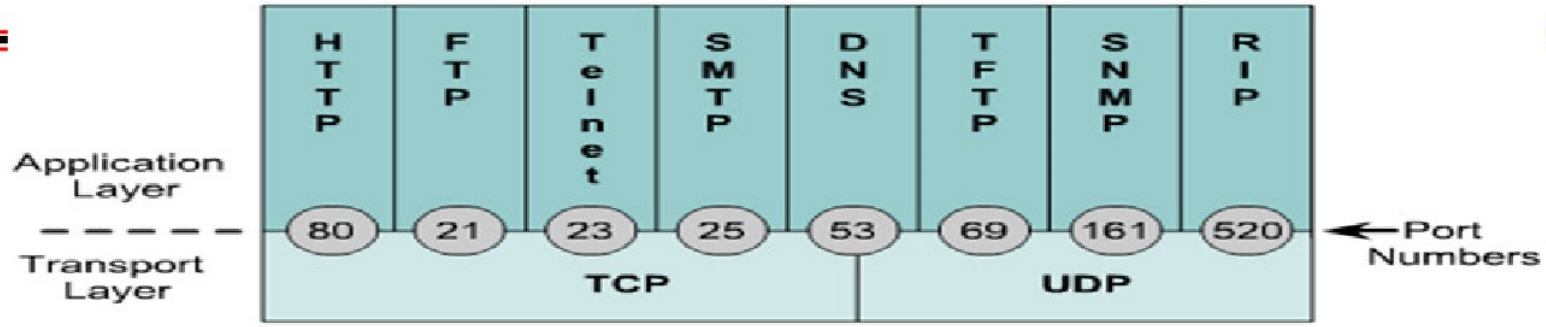
For tcp and udp protocol type, the test-conditions can further specify to filter based on layer 4 source port numbers and/or destination port numbers if need to.

Note the use of keywords like 'eq', etc before port numbers:





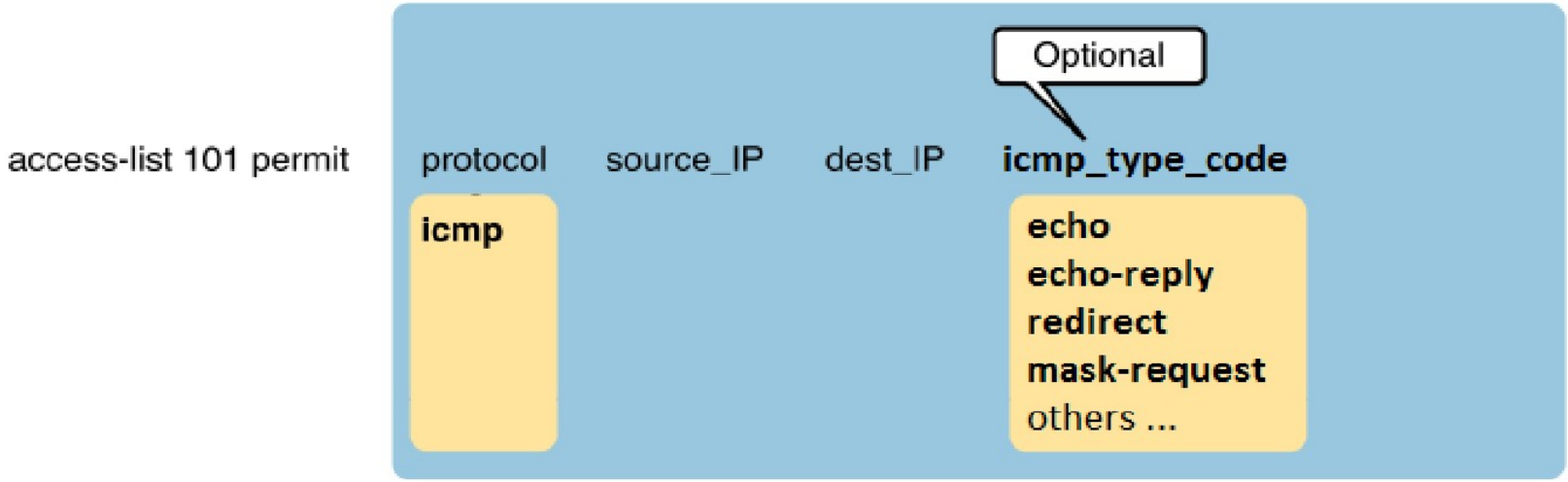
The port numbers may be specified directly using numbers, or names representing well-known port numbers. Some examples as follows:

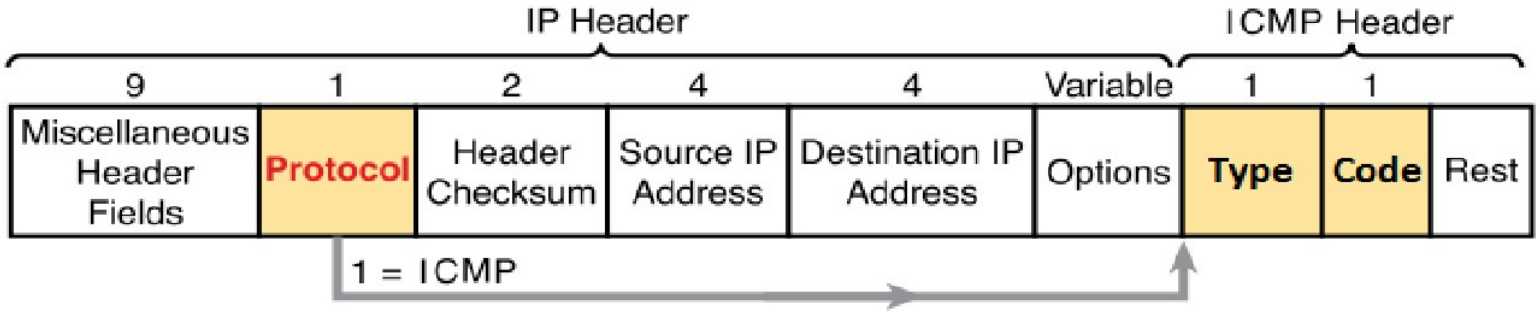


Port Number(s)	Protocol	Application	access-list Command Keyword
20	TCP	FTP data	ftp-data
21	TCP	FTP control	ftp
22	TCP	SSH	—
23	TCP	Telnet	telnet
25	TCP	SMTP	smtp
53	UDP, TCP	DNS	domain
67, 68	UDP	DHCP	nameserver
69	UDP	TFTP	tftp
80	TCP	HTTP (WWW)	www
110	TCP	POP3	pop3
161	UDP	SNMP	snmp
443	TCP	SSL	—

For icmp protocol type, the test-conditions may include optional special keywords representing different types of ICMP messages.

Note that there is no keyword like 'eq' for icmp protocol_type_code as compared to that for tcp and udp protocol types.





(4.3) Named ACLs provide the same functionalities as numbered ACLs except that names are used to identify ACLs which are easier to remember than numbers.

To configure standard or extended named ACLs, just convert the equivalent standard or extended numbered ACLs as follows:

Numbered ACL

access-list 1 **permit ...**
access-list 1 **deny ...**
access-list 1 **remark ...**

Named ACL

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

permit ...
deny ...
remark ...

An example of configuring extended named ACL:

```
Router(config)# ip access-list extended barney
Router(config-ext-nacl)# permit tcp host 10.1.1.2 eq www any
Router(config-ext-nacl)# deny udp host 10.1.1.1 10.1.2.0 0.0.0.255
Router(config-ext-nacl)# deny ip 10.1.3.0 0.0.0.255 10.1.2.0 0.0.0.255
Router(config-ext-nacl)# deny ip 10.1.2.0 0.0.0.255 10.2.3.0 0.0.0.255
Router(config-ext-nacl)# permit ip any any
Router(config-ext-nacl)# interface serial1
Router(config-if)# ip access-group barney out
```

In newer versions of Cisco IOS, numbered ACLs are also able to be configured in the same way as named ACLs.

Just use number instead of name to configure numbered ACL using the same style as named ACL:

```
! Step 1: The Standard Numbered IP ACL is configured.
R1# configure terminal
Enter configuration commands, one per line.  End with Ctrl-Z.
R1(config)# ip access-list standard 24
R1(config-std-nacl)# permit 10.1.1.0 0.0.0.255
R1(config-std-nacl)# permit 10.1.2.0 0.0.0.255
R1(config-std-nacl)# permit 10.1.3.0 0.0.0.255
```

Note that sequence numbers are automatically inserted by IOS:

```
! Step 2: Displaying the ACL's contents, without leaving configuration mode.
R1(config-std-nacl)# do show ip access-list 24
Standard IP access list 24
    10 permit 10.1.1.0, wildcard bits 0.0.0.255
    20 permit 10.1.2.0, wildcard bits 0.0.0.255
    30 permit 10.1.3.0, wildcard bits 0.0.0.255
```

With sequence numbers being inserted automatically, both numbered and named ACLs can be edited easily as follows:

To delete ACL statement, use no sequence-number command:

```
! Step 3: Still in ACL 24 configuration mode, the line with
 sequence number 20 is deleted.
R1(config-std-nacl)# no 20
```

To insert new ACL statement, configure it starting with a sequence-number to indicate the position:

```
! Step 4: Inserting a new first line in the ACL.
R1(config-std-nacl)# 5 deny 10.1.1.1
```

Verify the resultant ACL:

```
! Step 5: Displaying the ACL's contents with the new statement
R1(config-std-nacl)# do show ip access-list 24
Standard IP access list 24
     5 deny   10.1.1.1
    10 permit 10.1.1.0, wildcard bits 0.0.0.255
    30 permit 10.1.3.0, wildcard bits 0.0.0.255
```

In addition, the automatic numbering of the ACL statements can be resequenced as follows, e.g. if you need to insert more ACL statements in between:

Resequence numbering of ACL statements in ACL list:

```
! Step 6: resequence the ACL's contents
R1(config)# ip access-list resequence 24 10 20
```

starting sequence number

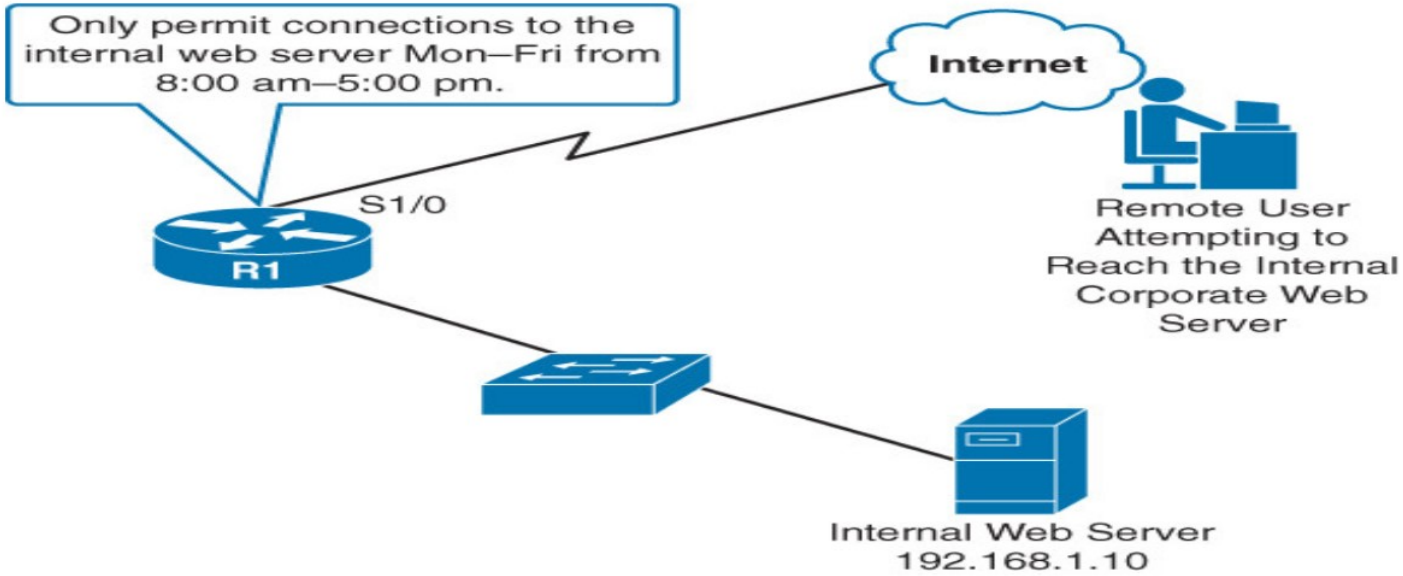
increment step

Verify the resultant ACL again:

```
! Step 7: Displaying the ACL's contents one last time
R1(config)# do show ip access-list 24
Standard IP access list 24
    10 deny   10.1.1.1
    30 permit 10.1.1.0, wildcard bits 0.0.0.255
    50 permit 10.1.3.0, wildcard bits 0.0.0.255
```

(4.4) Time-based ACLs are basically ACLs enhanced with time-range which will only become operational during certain time of the day or week.

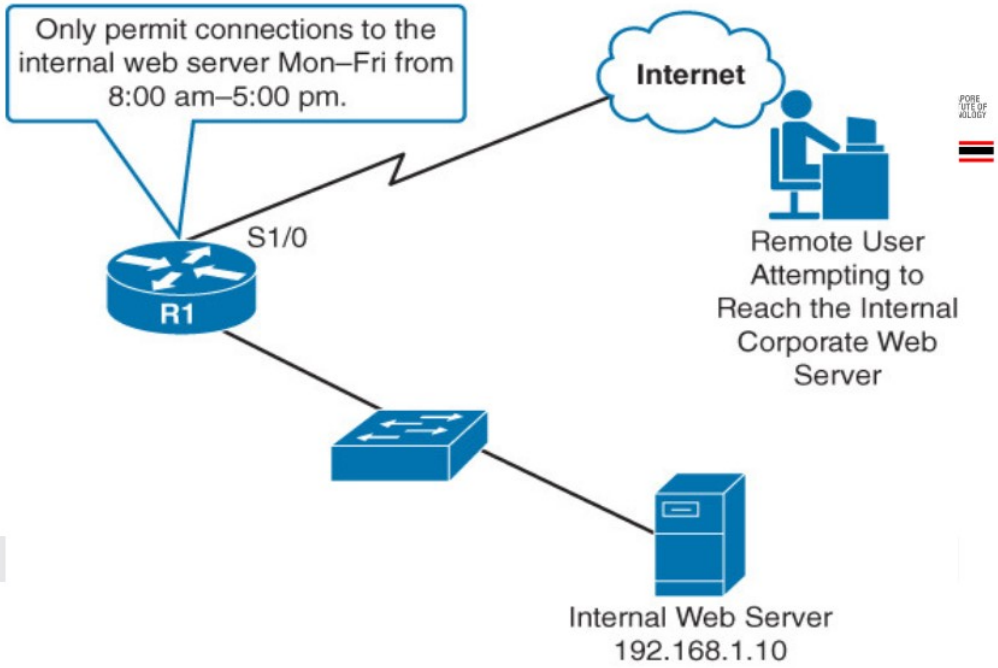
For example, a company may wish to only allow access to its internal web server during office hours. If without time-based ACL, it will be tedious and error-prone to manually remove the ACL during office hours, and add in after office hours.



Time-range is referencing the router's internal clock, which for accuracy can be synchronized using NTP (to be covered in later labs).

An example of configuring time-based ACL.

```
R1# conf term
R1(config)# time-range WEEKDAYS
R1(config-time-range)# periodic ?
  Friday      Friday
  Monday      Monday
  Saturday    Saturday
  Sunday      Sunday
  Thursday    Thursday
  Tuesday     Tuesday
  Wednesday   Wednesday
  daily       Every day of the week
  weekdays    Monday thru Friday
  weekend     Saturday and Sunday
```

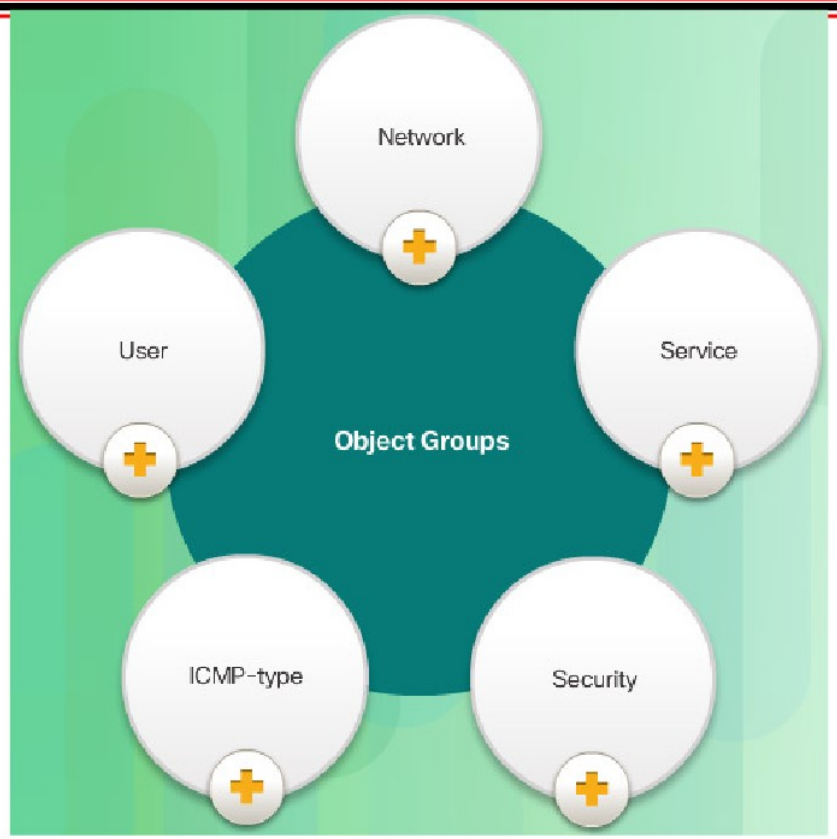



```
R1(config-time-range)# periodic weekdays 8:00 to 17:00
R1(config-time-range)# exit
R1(config)# access-list 100 permit tcp any host 192.168.1.10 eq 80 time-range
  WEEKDAYS

... OUTPUT OMITTED FOR OTHER PERMIT ACL STATEMENTS NOT RELEVANT TO THIS EXAMPLE ...

R1(config)# interface serial 1/0
R1(config-if)# ip access-group 100 in
R1(config-if)# end
```

(4.5) To be more readable, a new way of writing ACL called object group-based ACL has been introduced in Cisco routers and ASA firewalls.

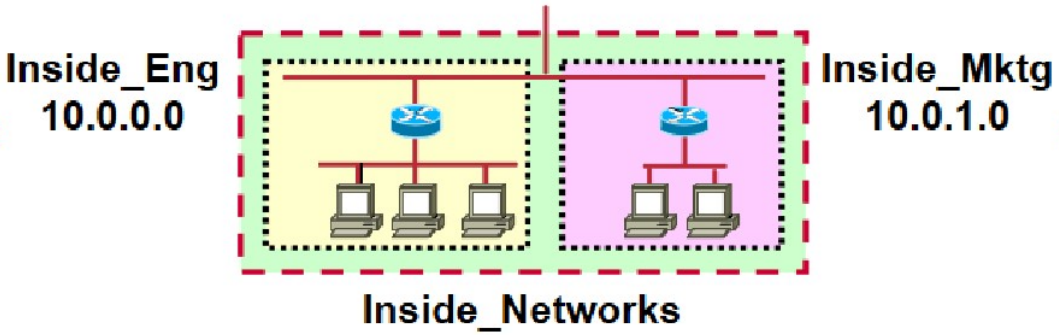


First, configure related objects into object groups:

- Network
  - host IP
  - IP ranges
  - subnet
  - etc
- Service
  - protocol
  - port number
  - etc
  - e.g. tcp www

Note: User, ICMP-type and Security object groups are less commonly used and are only available in ASA firewalls.

An example of configuring network object group.

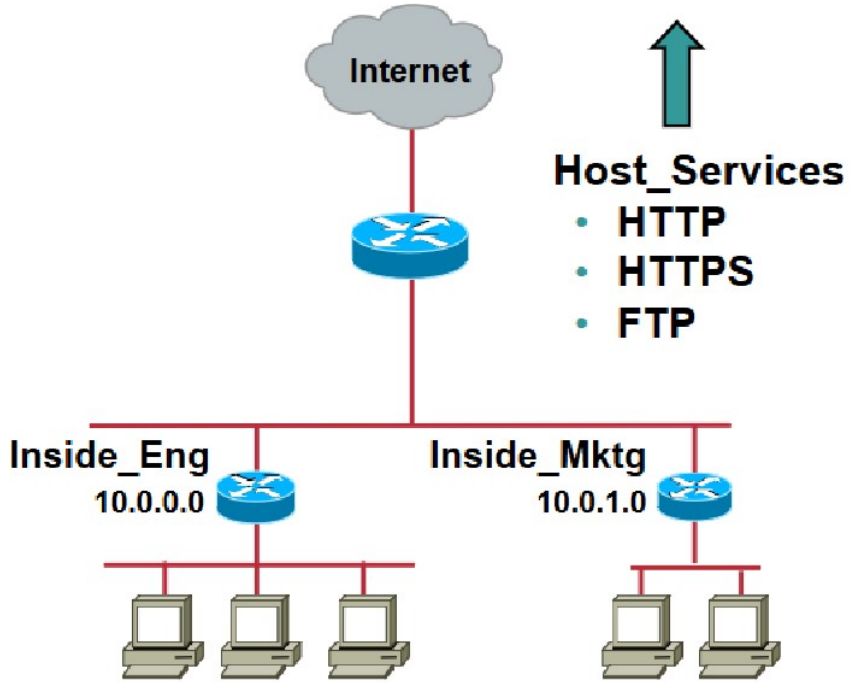



```
router(config)# object-group network Inside_Eng
router(config-network-group)# 10.0.0.0 /24
router(config-network-group)# exit
router(config)# object-group network Inside_Mktg
router(config-network-group)# host 10.0.1.1
router(config-network-group)# host 10.0.1.2
router(config-network-group)# exit
router(config)# object-group network Inside_Networks
router(config-network-group)# group-object Inside_Eng
router(config-network-group)# group-object Inside_Mktg
```

Notes:

- Object group may be nested using the group-object command
- Configuration commands on ASA firewalls are slightly different

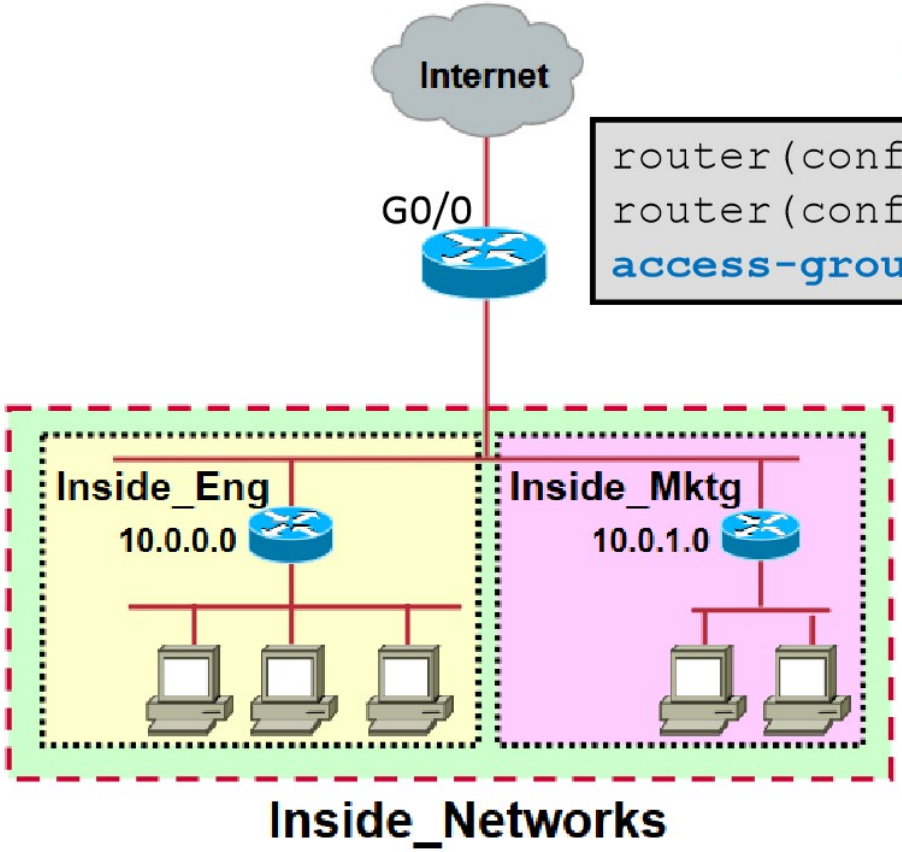
An example of configuring service object group.




```
router(config)# object-group service Host_Services
router(config-service-group)# tcp www
router(config-service-group)# tcp 443
router(config-service-group)# tcp ftp
```

Note: Configuration commands on ASA firewalls are slightly different, use ? for help.

Finally, configure object group-based ACL based on the configured network and service object groups as follows:




```
router(config)# int G0/0
router(config-if)# ip
access-group 100 out
```

```
router(config)# access-list 100 permit object-group
Host_Services object-group Inside_Networks any
```