SLIIT

Discover Your Future

IT2050 - COMPUTER NETWORKS

Lecture 9

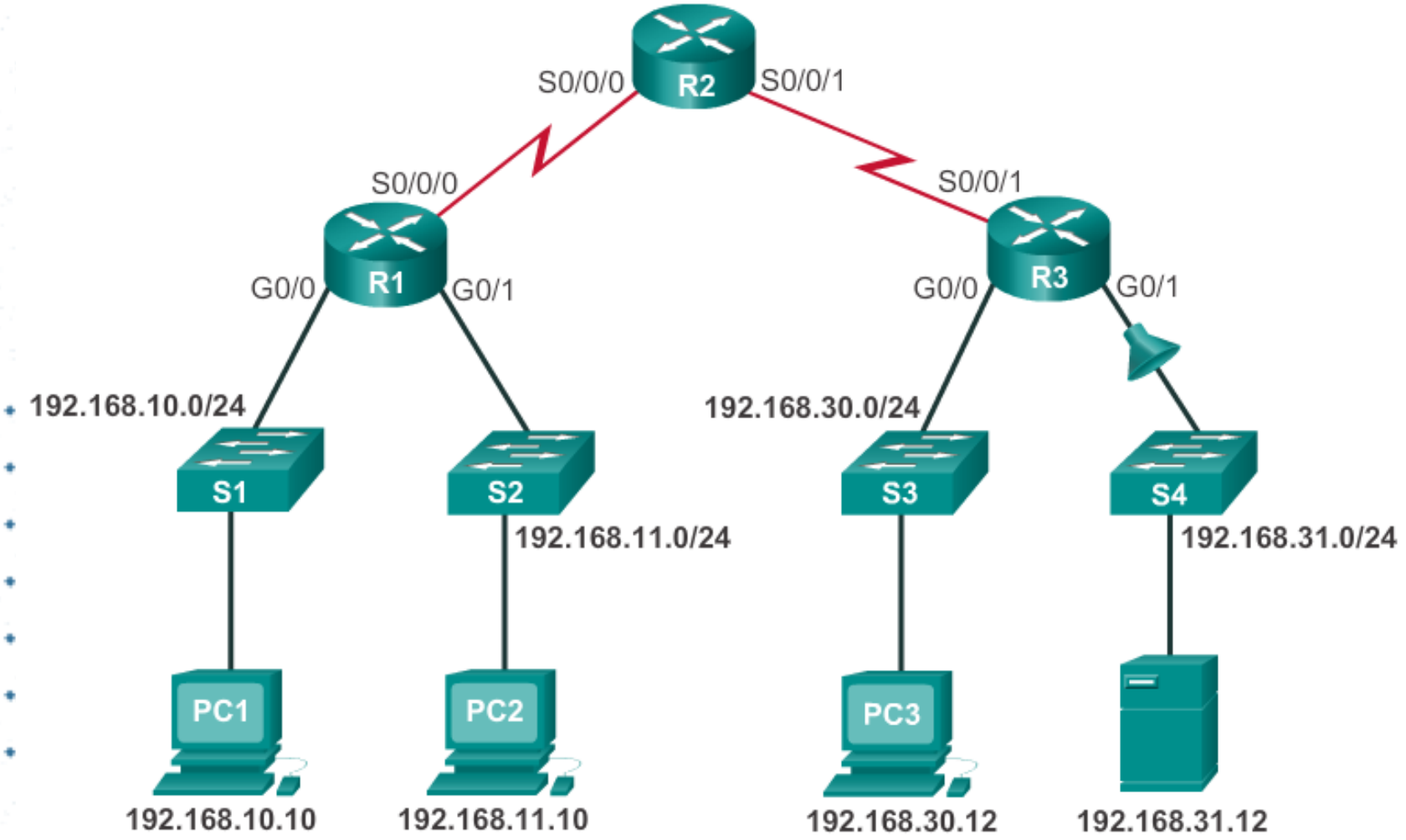
Access Control Lists

SLIIT FACULTY OF COMPUTING

Introduction

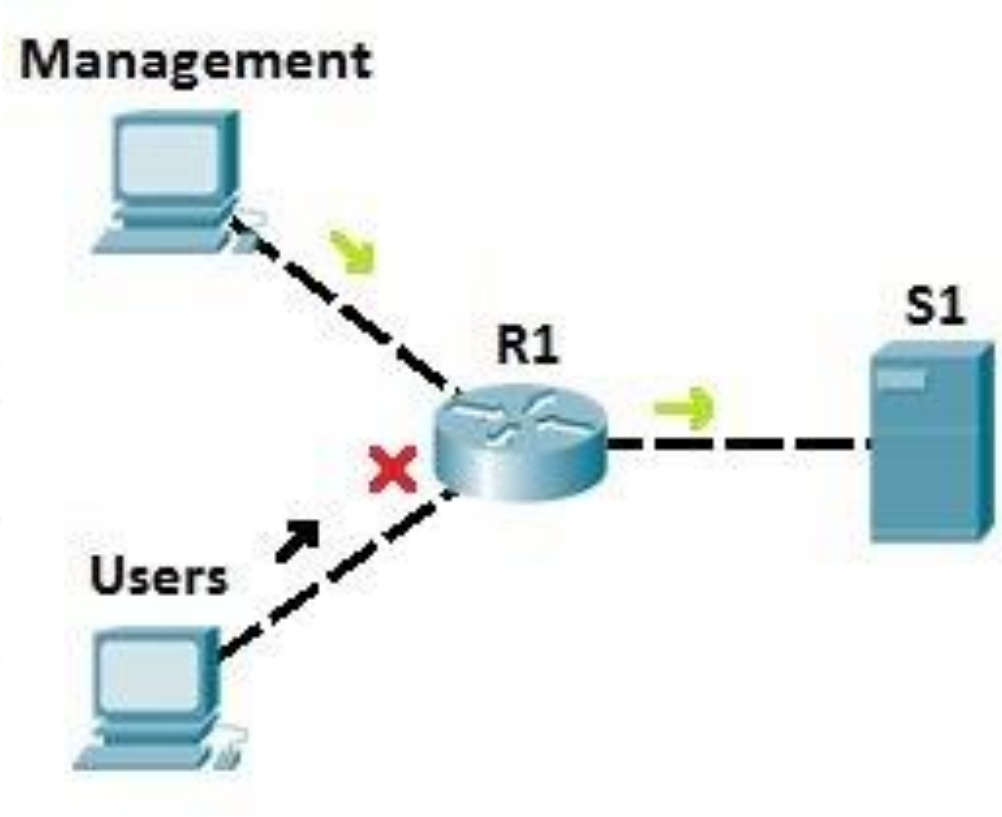
- Access Control Lists (ACLs) in routers are like security rules that control the flow of data in and out of the router.
- They work like a filter to decide which data is allowed to enter or leave the router and which should be blocked.
- ACLs are lists of conditions used to test network traffic that tries to travel across a router interface.
- These conditions tell the router what types of packets to accept or deny.
- Acceptance and denial are based on specified conditions, such as source address, destination address, protocols, and upper-layer port numbers.

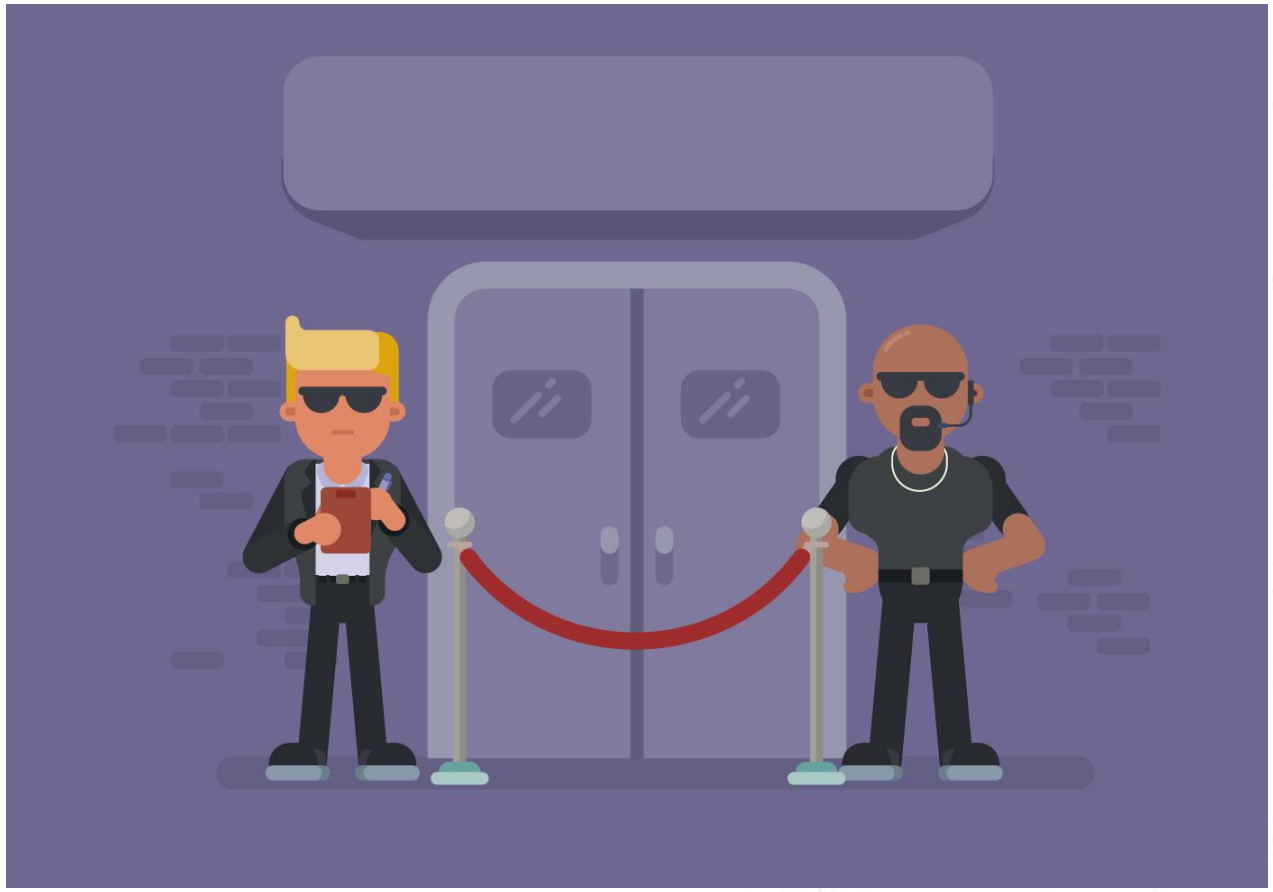
Introduction



- ACLs is a gatekeeper for a router.
- When data wants to go through the router, the ACL checks if it meets the specified conditions (allowed) or not (blocked).
- It's like a bouncer at the door of a party, deciding who can come in and who should stay out to keep the network safe and working well.

Introduction





What ACL can do ?

1. **Filter Traffic**: ACLs decide which data can go through and which gets blocked, based on set rules.
2. **Control Access**: ACLs control who can access certain things, like websites or servers. Admin can specify who is allowed and who is not allowed by specifying conditions like IP addresses or types of data.
3. **Boost Security**: ACLs help keep your network safe from bad stuff like cyber-attacks. They can block harmful traffic and only let the good stuff in.
4. **Manage Traffic:** ACLs can help direct data traffic in your network, making it flow more efficiently. You can decide how data should travel.
5. **Prioritize Traffic: ACLs** help to give priority to important things on the networks, like video calls or critical apps, so they work smoothly.

What ACL can do ?

6. **Keep an Eye on Traffic: C**an use ACLs to watch over the network traffic. This helps with troubleshooting and making sure everything is running as it should.
7. **Follow Rules:** ACLs help to enforce rules for the network. This keeps things in order and helps with security and following regulations.
8. **Separate Networks:** Can use ACLs to separate parts of the network, like keeping guest users away from sensitive data.
9. **Enhance Security:** ACLs are like bodyguards for the network. ACLs keep hackers out by blocking their attempts to enter your network without permission.
10. **VPN Tunneling:** Match Packets for VPN Tunneling

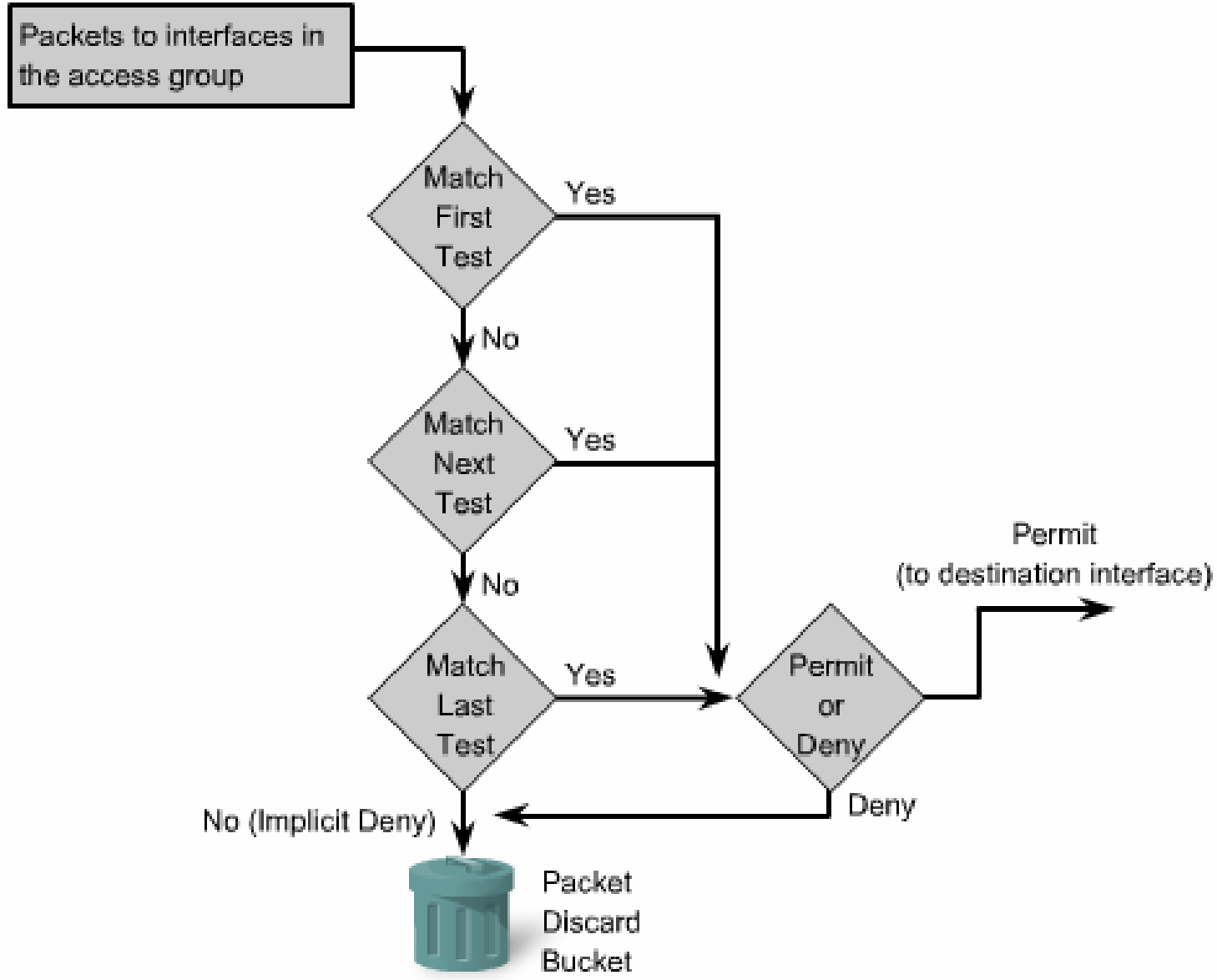
How ACL works?

- Access Control Lists (ACLs) work by creating a set of rules or conditions that determine which network traffic is allowed to pass through a network device (such as a router, switch, or firewall) and which traffic should be blocked.

How ACL works?

1. **Traffic Inspection**: When data packets arrive at a network device, such as a router, the device inspects each packet to see if it matches any of the rules defined in the ACL.
2. **Rule Matching**: Each rule in the ACL specifies certain conditions that must be met for a packet to be allowed. These conditions can include criteria like source IP addresses, destination IP addresses, protocols (e.g., TCP, UDP), and port numbers.
3. **Sequential Evaluation**: ACLs are typically evaluated in a sequential order, one rule at a time, from the top of the list to the bottom. The device checks if the incoming packet meets the conditions of each rule in the order they are listed.
4. **Decision Making**: If a packet matches the conditions of a rule, the device makes a decision based on that rule. ACLs can be configured to either permit (allow) the packet to continue its journey through the network or deny (block) it from passing further.
5. **Implicit Deny**: ACLs usually have an implicit deny statement at the end, which means that if a packet doesn't match any of the rules in the ACL, it will be denied by default. This ensures that no traffic gets through unless specifically allowed.
6. **Logging and Monitoring**: Some ACLs can be configured to log information about packets that match specific rules. This logging can help with network monitoring and troubleshooting.

How ACL works



Types of IPv4 ACLs

Standard ACLs

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

Standard ACLs filter IP packets based on the source address only.

Extended ACLs

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

Extended ACLs filter IP packets based on several attributes, including the following:

- Source and destination IP addresses
- Source and destination TCP and UDP ports
- Protocol type/ Protocol number (example: IP, ICP, UDP, TCP, etc.)

Numbering and Naming ACLs

Numbered ACL:

You assign a number based on which protocol you want filtered:

- (1 to 99) and (1300 and 1999): Standard IP ACL
- (100 to 199) and (2000 to 2699): Extended IP ACL

Named ACL:

You assign a name by providing the name of the ACL:

- Names can contain alphanumeric characters.
- It is suggested that the name be written in CAPITAL LETTERS.
- Names cannot contain spaces or punctuation.
- You can add or delete entries within the ACL.

ACL Wildcard Masking

Wildcard masks and subnet masks handle binary 1s and 0s differently. Here's a simplified version of how wildcard masks work:

- **Wildcard Mask Bit 0**: When there's a 0 in the wildcard mask, it means that the corresponding bit in the address needs to match.
- **Wildcard Mask Bit 1**: If there's a 1 in the wildcard mask, it tells us to ignore the corresponding bit in the address.

You can think of wildcard masks as the opposite of subnet masks. In subnet masks, 1s mean a match, and 0s mean no match.

But in wildcard masks, it's the other way around: 0s mean a match, and 1s mean no match.

Wildcard Mask Examples: Hosts / 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

Host

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

Any

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

Subnet

Wildcard Mask Examples: Match Ranges

Example 1

	Decimal	Binary
IP Address	192.168.16.0	11000000.10101000.00010000.00000000
Wildcard Mask	0.0.15.255	00000000.00000000.00001111.11111111
Result Range	192.168.16.0 to 192.168.31.255	11000000.10101000.00010000.00000000 to 11000000.10101000.00011111.11111111

192.168.16.0/24 – 192.168.31.0/24

Example 2

	Decimal	Binary
IP Address	192.168.1.0	11000000.10101000.00000001.00000000
Wildcard Mask	0.0.254.255	00000000.00000000.11111110.11111111
Result	192.168.1.0	11000000.10101000.00000001.00000000
	All odd numbered subnets in the 192.168.0.0 major network	

192.168.1.0/24,
192.168.3.0/24,
192.168.5.0/24,
192.168.7.0/24,
Etc

Wildcard Mask Cont…

Wildcard Mask	Binary Version of the Mask	Description
0.0.0.0	00000000.00000000.00000000.00000000	The entire IP address must match.
0.0.0.255	00000000.00000000.00000000.11111111	Just the first 24 bits must match.
0.0.255.255	00000000.00000000.11111111.11111111	Just the first 16 bits must match.
0.255.255.255	00000000.11111111.11111111.11111111	Just the first 8 bits must match.
255.255.255.255	11111111.11111111.11111111.11111111	Don't even bother to compare; it's automatically considered to match (0 bits need to match).
0.0.15.255	00000000.00000000.00001111.11111111	Just the first 20 bits must match.
0.0.3.255	00000000.00000000.00000011.11111111	Just the first 22 bits must match.
32.48.0.255	00100000.00110000.00000000.11111111	All bits except the 3rd, 11th, 12th, and last 8 must match.

Calculating the Wildcard Mask

Calculating wildcard masks can be challenging. One shortcut method is to subtract the subnet mask from 255.255.255.255.

Example 1

```
  255.255.255.255
- 255.255.255.000
-----------------
  000.000.000.255
```

Example 2

```
  255.255.255.255
- 255.255.255.240
-----------------
  000.000.000.015
```

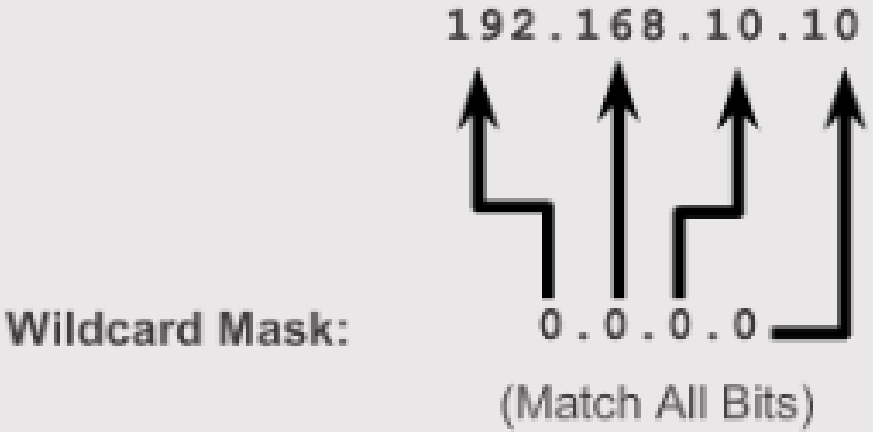
Example 3

```
  255.255.255.255
- 255.255.252.000
-----------------
  000.000.003.255
```

Wildcard Mask Keywords

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**)




```
R1(config)#access-list 1 permit 192.168.10.10 0.0.0.0
R1(config)#access-list 1 permit host 192.168.10.10
```

Wildcard Mask Keywords

Example 2

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


```
R1(config)#access-list 1 permit 0.0.0.0 255.255.255.255
R1(config)#access-list 1 permit any
```

General Guidelines for Creating ACLs

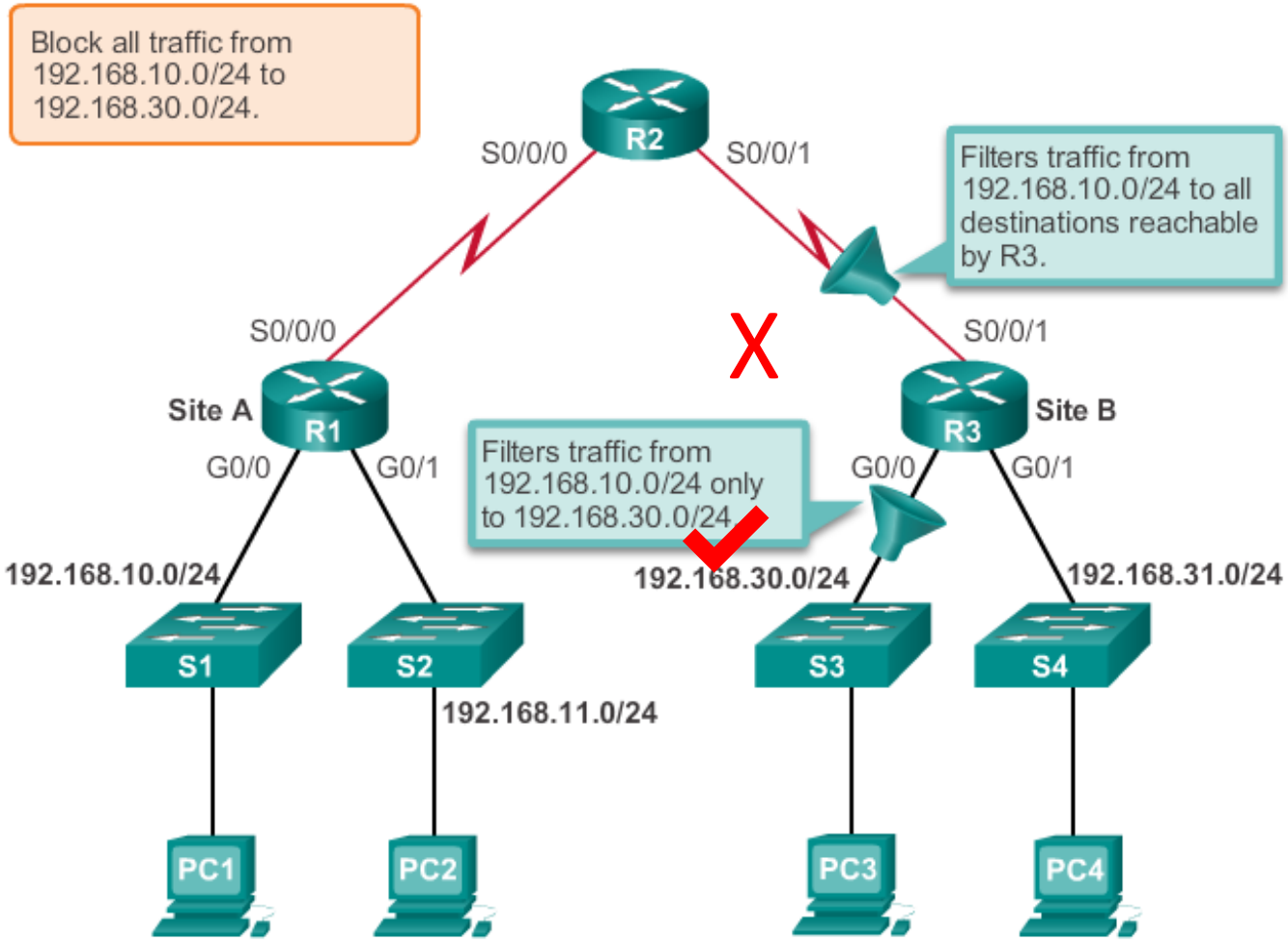
- **One ACL per protocol -** To control traffic flow on an interface, an ACL must be defined for each protocol enabled on the interface.
- **One ACL per direction -** ACLs control traffic in one direction at a time on an interface. Two separate ACLs must be created to control inbound and outbound traffic.
- **One ACL per interface -** ACLs control traffic for an interface, for example, GigabitEthernet 0/0.

Where to Place ACLs

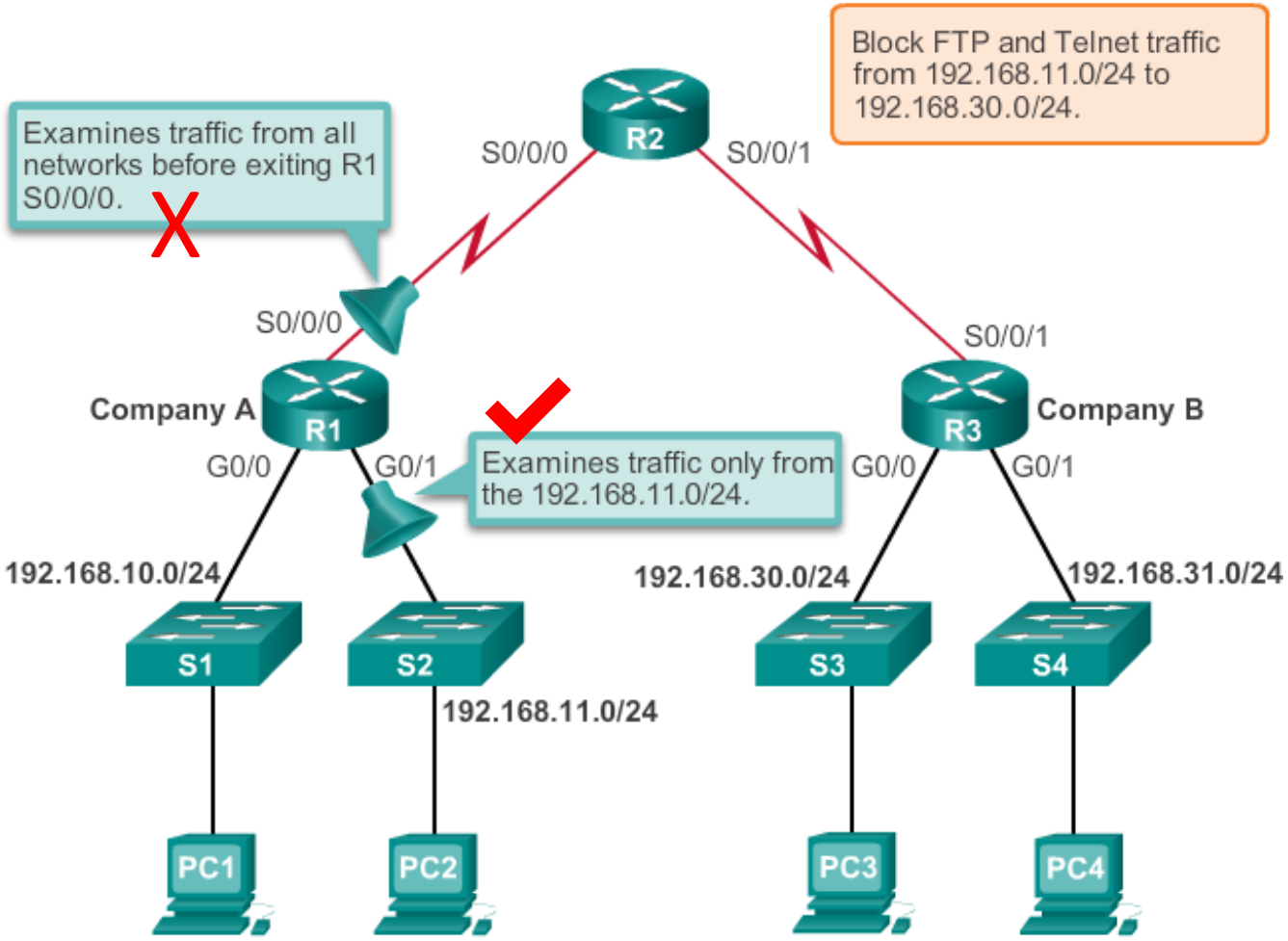
Every ACL should be placed where it has the greatest impact on efficiency. The basic rules are:

- **Extended ACLs:** Locate extended ACLs as **close as possible to the source** of the traffic to be filtered.
- **Standard ACLs:** Because standard ACLs do not specify destination addresses, place them as **close to the destination** as possible.

Where to Place ACLs – Standard



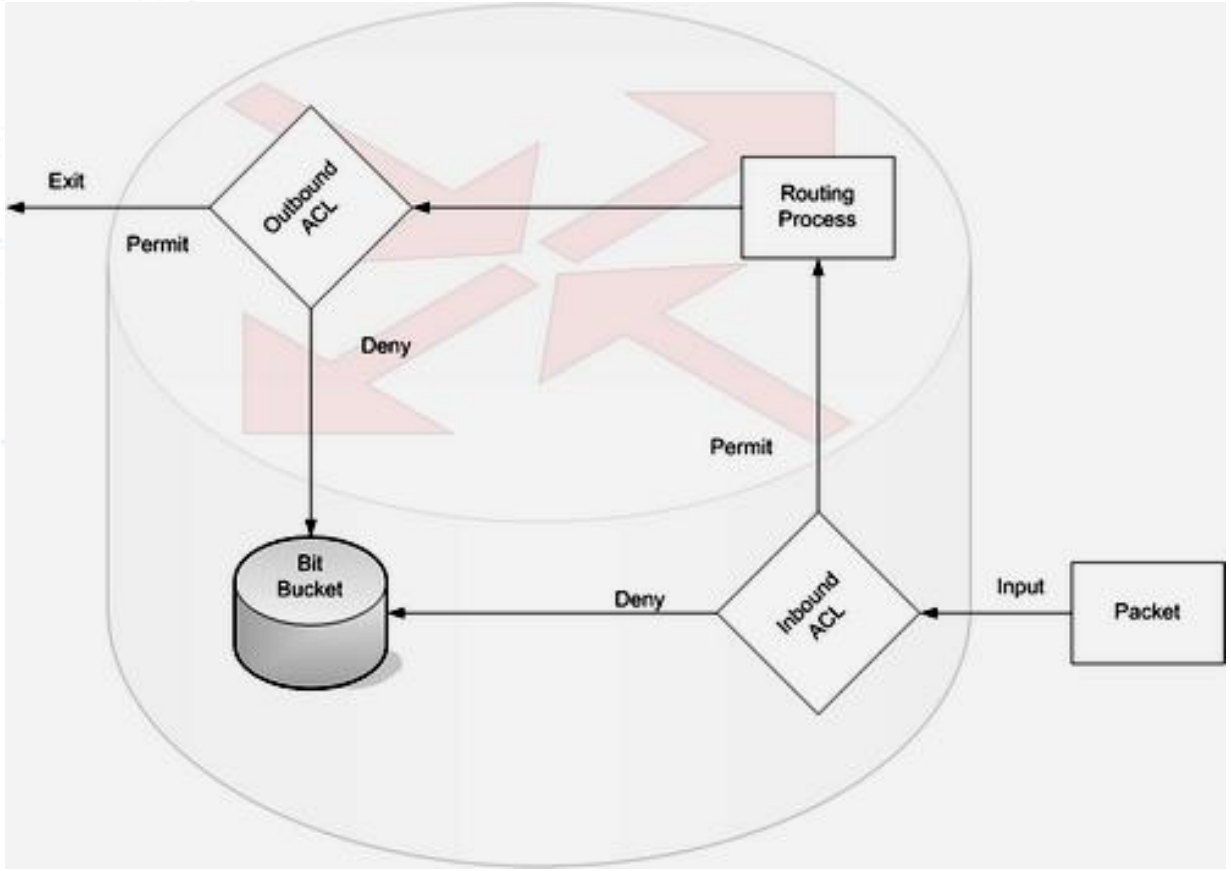
Where to Place ACLs – Extended



Apply ACL to an interface

- ACLs are configured either to apply to inbound traffic or to apply to outbound traffic
- Inbound ACLs-Incoming packets are checked with the ACLs before taking the routing decisions
- An inbound ACL is efficient because it saves the overhead of routing lookups if the packet is discarded
- If the packet is permitted by the tests, it is then processed for routing
- Outbound ACLs-Incoming packets are first process for the routing decisions and then checked with the outbound ACL

Apply ACL to an interface cont.



ACL Configuration

1. **Create ACL (Access Control List)**:
    1. ACLs are sets of rules that determine which network traffic is allowed or denied based on specific criteria like source and destination IP addresses, protocols, and ports.
    2. To create an ACL, you define rules using a unique name or number. Each rule specifies whether to permit or deny traffic and the conditions that must be met for it to match the rule.
2. **Apply ACL to an Interface**:
    1. After creating an ACL, you need to apply it to a specific network interface (e.g., a router interface) where you want the ACL to take effect.
    2. ACLs can be applied in two directions: inbound (traffic coming into the interface) or outbound (traffic leaving the interface).
    3. When applying the ACL, you specify whether it should be applied inbound or outbound and reference the ACL by its name or number.

Standard ACLs

These ACLs are called "**standard**" because they primarily focus on source IP addresses and offer basic traffic filtering capabilities.

1. **Source IP Address Filtering**: Standard ACLs filter traffic based on the source IP addresses of incoming packets. You can specify which source IP addresses are allowed or denied access.
2. **Limited Criteria**: Standard ACLs provide limited criteria for traffic filtering. They don't consider destination IP addresses, protocols, or port numbers. This makes them less flexible compared to extended ACLs.
3. **Sequential Evaluation**: Like other ACLs, standard ACLs are evaluated in a sequential order, from the top rule to the bottom rule. The first rule that matches determines the action (permit or deny), and subsequent rules are not checked.
4. **Implicit Deny**: Standard ACLs have an implicit deny statement at the end. This means that if a packet doesn't match any of the defined rules, it will be denied by default.
5. **Numeric Range**: Standard ACLs are identified by a numeric range (1 to 99) or a name assigned to the ACL.
6. **Use Cases**:
    1. Standard ACLs are commonly used when you want to restrict access to specific source IP addresses or networks.
    2. They are often used at network boundaries to control traffic entering or exiting a network.

Standard ACLs

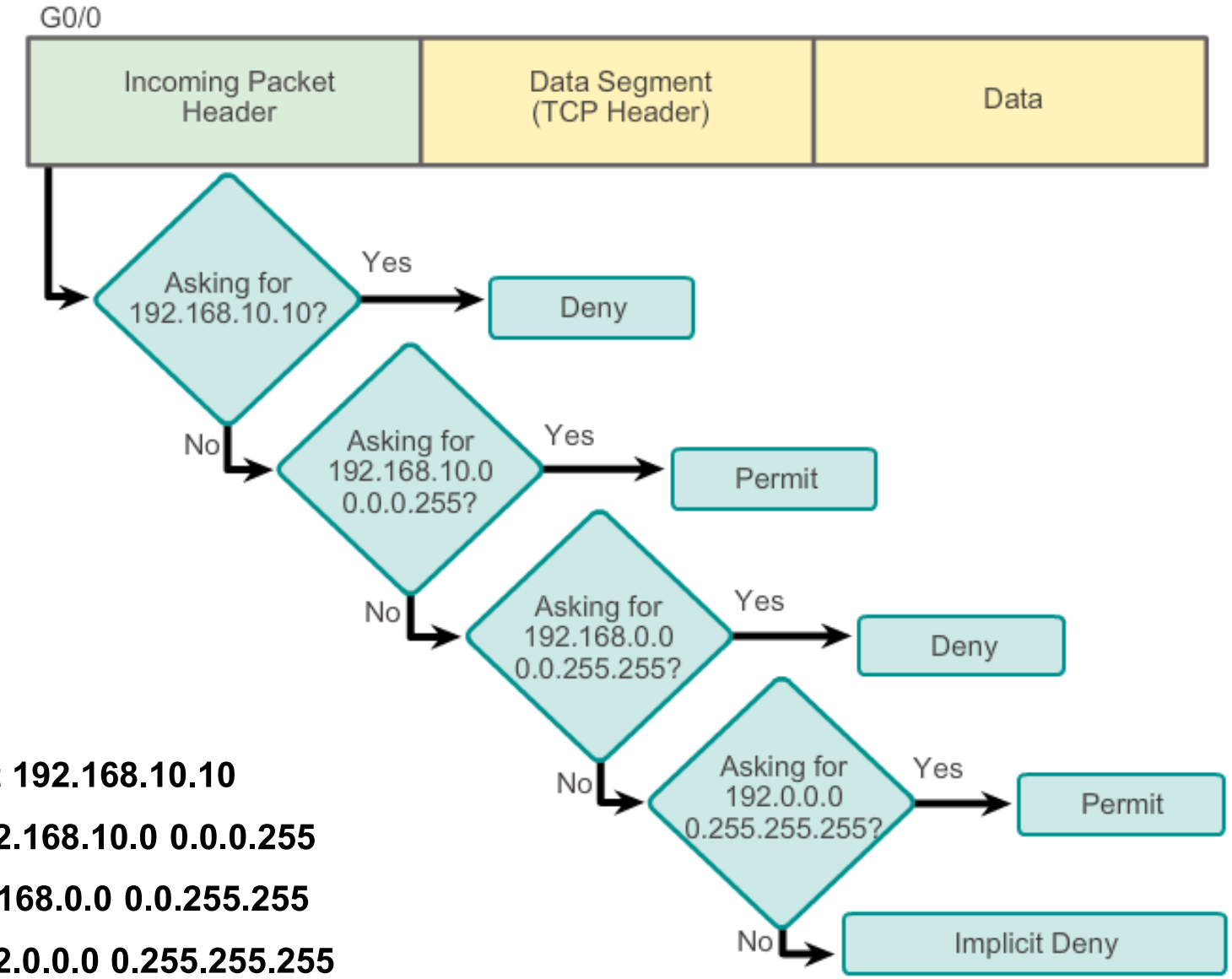
The full syntax of the standard ACL command is as follows:

Router(config)# **access-list *access-list-number*** **{permit | deny}** **{Source address}** **{wildcard mask}**

```
R1(config)#access-list 3 deny 192.168.10.0 0.0.0.255
R1(config)#access-list 3 permit host 192.168.10.10
% Access rule can't be configured at higher sequence num as
it is part of the existing rule at sequence num 10
R1(config)#
```

As discussed previously, access list statements are processed sequentially. Therefore, the order in which statements are entered is important.

Standard ACLs

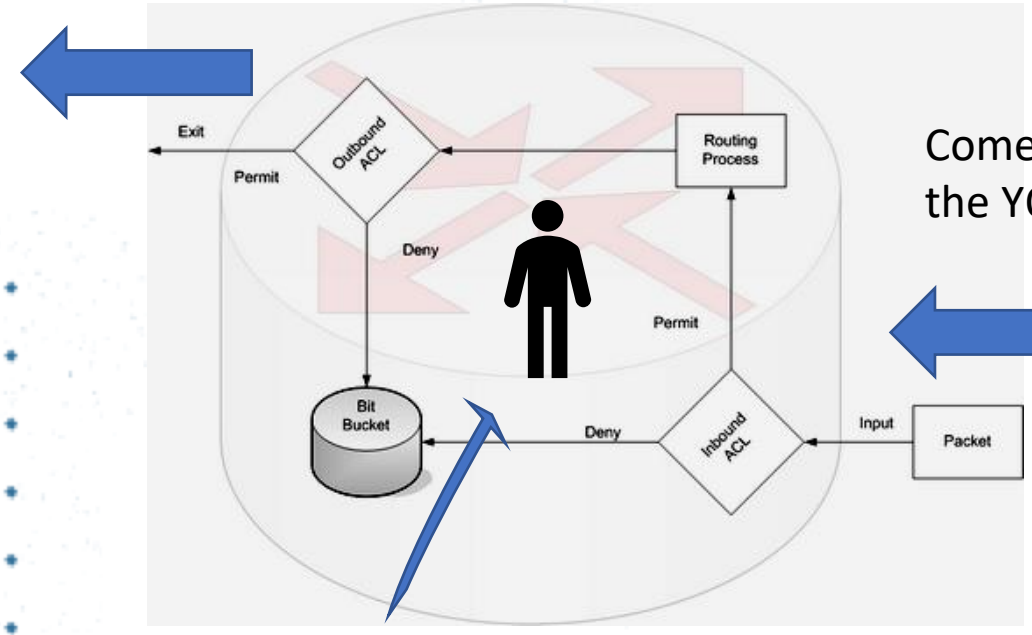


Example ACL

- **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**

Applying Standard ACL to Interfaces

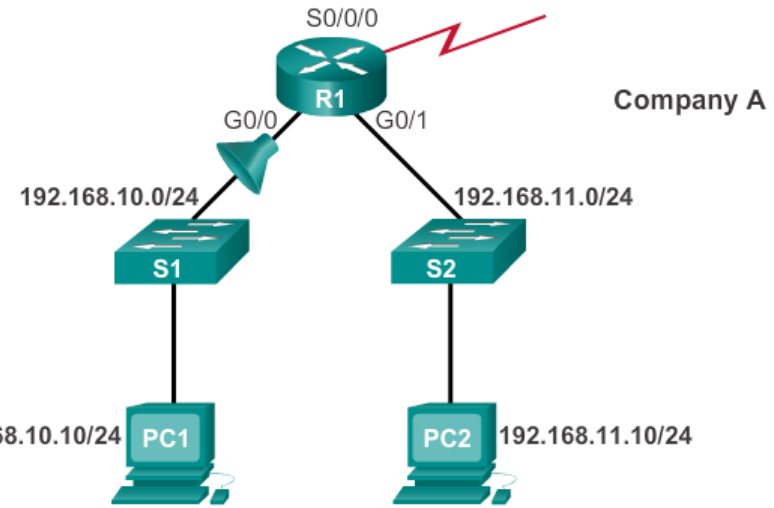
Leaving the YOU (***OUT***)



Come towards the YOU (***IN***)

Consider you living inside the router

Deny a Specific Host




```
R1(config)#access-list 1 deny host 192.168.10.10
R1(config)#access-list 1 permit any
R1(config)#interface g0/0
R1(config-if)#ip access-group 1 in
```

Extended ACLs

Unlike standard ACLs, which primarily focus on source IP addresses, extended ACLs offer a wider range of criteria for filtering traffic. Here are some key points about extended ACLs:

1. **Criteria for Filtering**:
   - **Source and destination IP addresses:** You can specify both the source and destination IP addresses in a rule.
   - **Protocol types** (e.g., TCP, UDP, ICMP): You can filter traffic based on the specific network protocols used.
   - **Port numbers**: You can control traffic based on source and destination port numbers, allowing for fine-grained control over applications and services.
2. **More Granularity**: Extended ACLs provide finer granularity in traffic filtering compared to standard ACLs. You can create rules that consider both source and destination addresses and specific application ports.
3. **Sequential Evaluation**: Similar to standard ACLs, extended ACLs are evaluated in a sequential order, from the top rule to the bottom rule. The first rule that matches determines the action (permit or deny).
4. **Implicit Deny**: Extended ACLs, like standard ACLs, have an implicit deny statement at the end. If a packet doesn't match any of the defined rules, it will be denied by default.
5. **Named or Numeric**: Extended ACLs can be identified by either a numeric range (e.g., ACL 100 to 199) or a name assigned to the ACL.
6. **Use Cases**:
   1. Extended ACLs are suitable for a wide range of network access control scenarios, including permitting or denying specific services, applications, or communication between specific hosts.
   2. They are often used for more complex security policies within a network.

Extended ACLs

The full syntax of the extended ACL command is as follows:

```
Router(config)# access-list access-list-number
                    {permit | deny} {protocol}
            {Source address} {wildcard mask}
        {destination address} {wildcard mask}
                    {eq | lt | gt} {port number}
```

As discussed previously, access list statements are processed sequentially. Therefore, the order in which statements are entered is important.

Using Port Numbers

```
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 23
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 21
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 20
```

Using Keywords

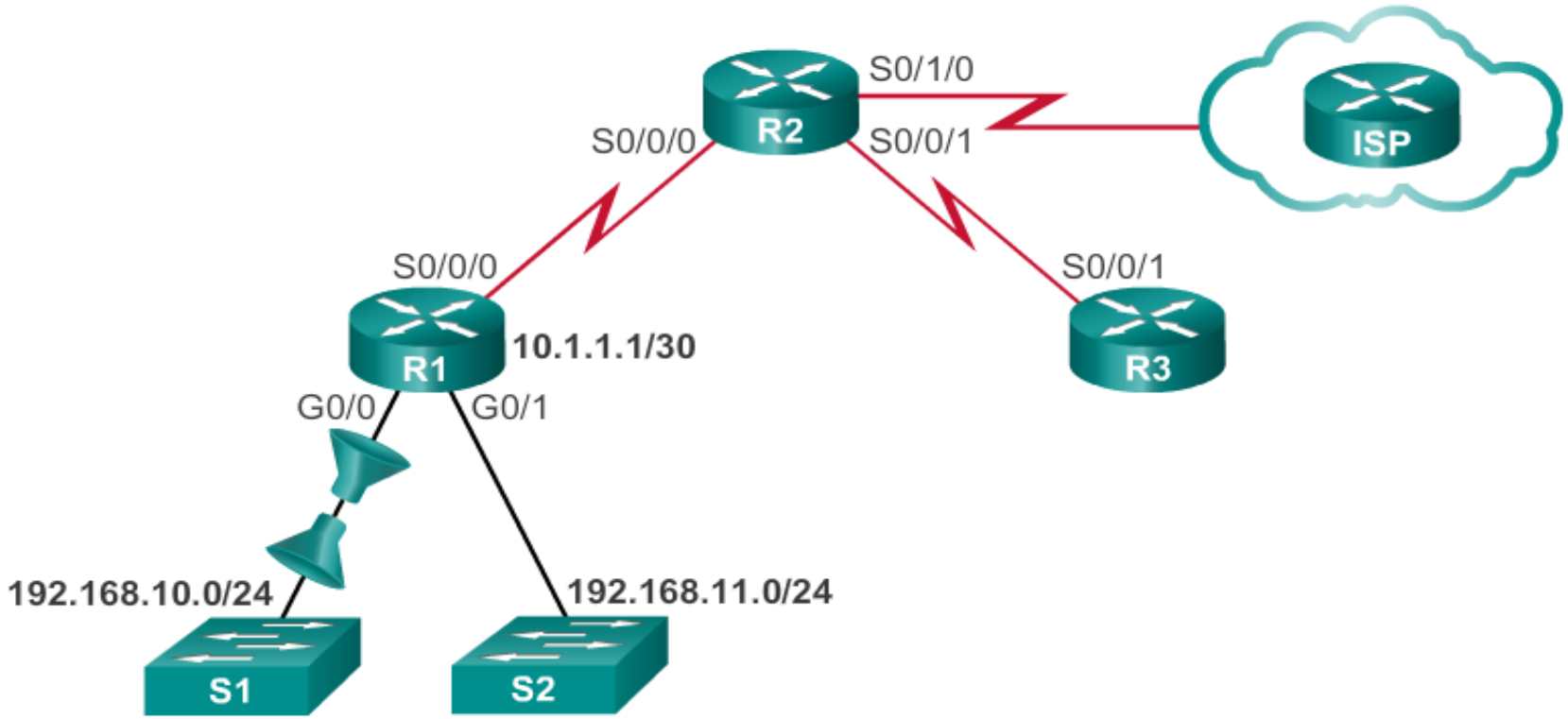
```
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq telnet
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq ftp
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq ftp-data
```

Extended ACLs

Extended ACLs can filter on:

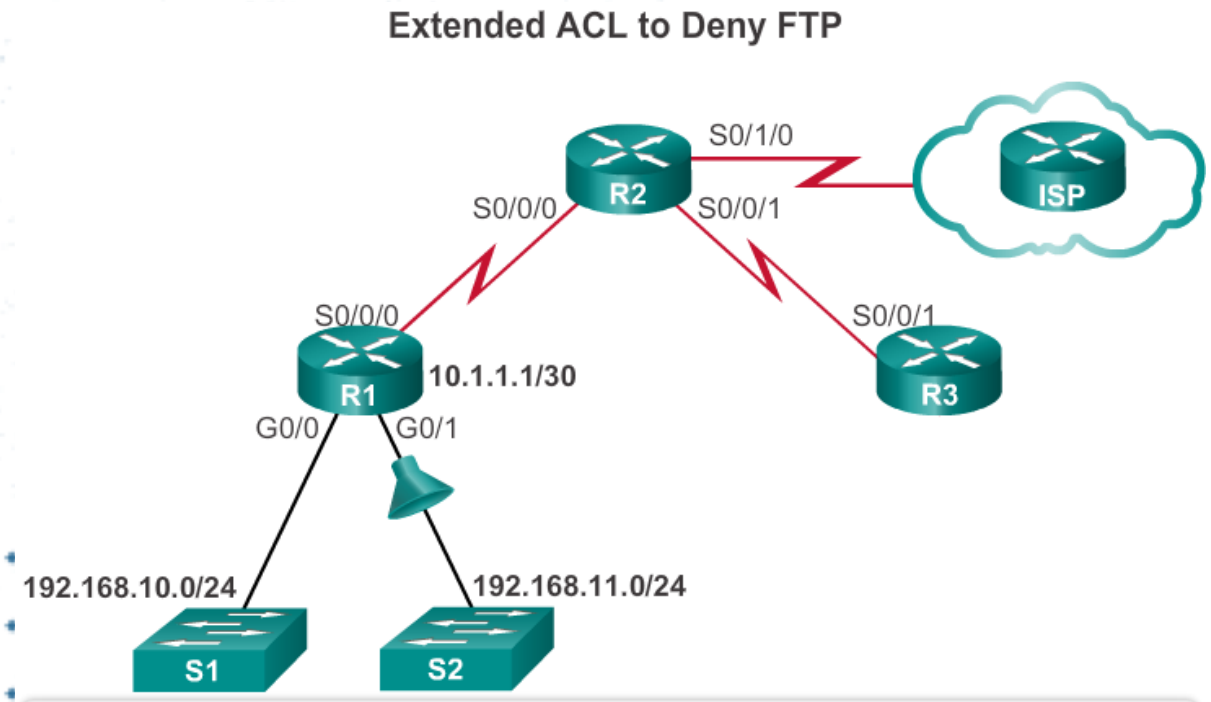
- Source address
- Destination address
- Protocol
- Port numbers

Applying Extended ACL to Interfaces




```
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config)#access-list 104 permit tcp any 192.168.10.0 0.0.0.255
R1(config)#interface g0/0
R1(config-if)#ip access-group 103 in
R1(config-if)#ip access-group 104 out
```

Filtering traffic using Extended ACLs




```
R1(config)#access-list 101 deny tcp 192.168.11.0 0.0.0.255 192.168.10.0
0.0.0.255 eq ftp
R1(config)#access-list 101 deny tcp 192.168.11.0 0.0.0.255 192.168.10.0
0.0.0.255 eq ftp-data
R1(config)#access-list 101 permit ip any any
R1(config)#interface g0/1
R1(config-if)#ip access-group 101 in
```

The FTP traffic generating from 192.168.11.0/24 network will be deny accessing all the other networks

Access-list 101 permit ip any any

is added at the end of the access list to explicitly allow all IP traffic from any source to any destination. This is done to override the implicit deny statement that is present at the end of every access list.

Without this line, the 192.168.11.0/24 network would be subject to the implicit deny rule. Therefore, adding above ensures that the 192.168.11.0/24 network can access any network but ftp traffic.

Access List Configuration Guidelines

- Access list numbers indicate which protocol is filtered.
- One access list per interface, per protocol, per direction is allowed.
- The order of access list statements controls testing.
- The most restrictive statements should be at the top of list.

Access List Configuration Guidelines Contd….

- There is an implicit deny any as the last access list test—every list should have at least one permit statement.
- Access lists should be created before to interfaces being applied.
- Access lists filter traffic going through the router; they do not apply to traffic originated from the router.

Creating Named Standard ACLs

```
Router(config)#ip access-list [standard | extended ] name
```

Alphanumeric name string must be unique and cannot begin with a number.

```
Router(config-std-nacl)#[permit | deny | remark] {source
[source- wildcard]} [log]
```

```
Router(config-if)#ip access-group name [in | out]
```

Activates the named IP ACL on an inteface.

Creating Named Standard ACLs

```
R1(config)#ip access-list standard NO_ACCESS
R1(config-std-nacl)#deny host 192.168.11.10
R1(config-std-nacl)#permit 192.168.11.0 0.0.0.255
R1(config-std-nacl)#interface Fa0/0
R1(config-if)#ip access-group NO_ACCESS out
```

Advantages of Named ACLs

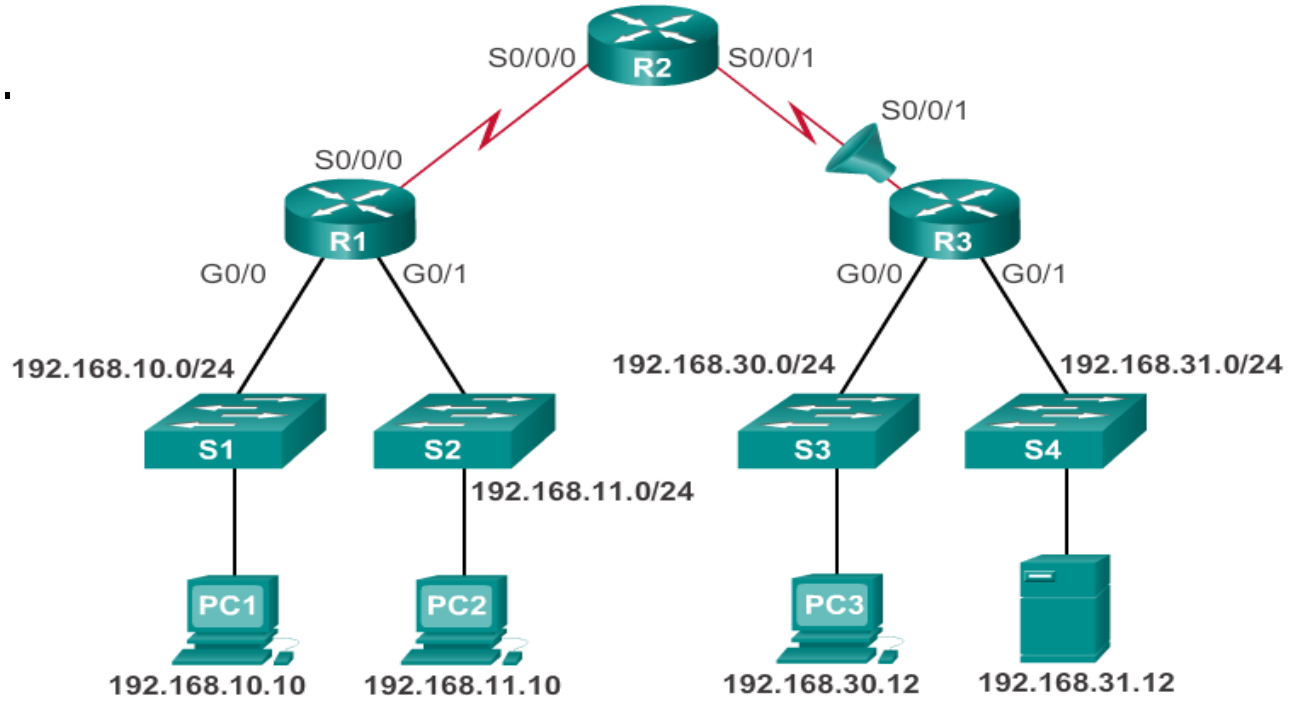
- it is easier to understand the function of ACL because you have used the function of ACL as its name
- It is easier to edit because Named ACLs allow you to delete individual entries in a specific ACL
- Can use sequence numbers to insert statements anywhere in the named ACL

Other types of ACLs

- Dynamic ACLs
- Time based ACLs
- Reflexive ACLs
- Turbo ACLs

Troubleshooting Common ACL Errors

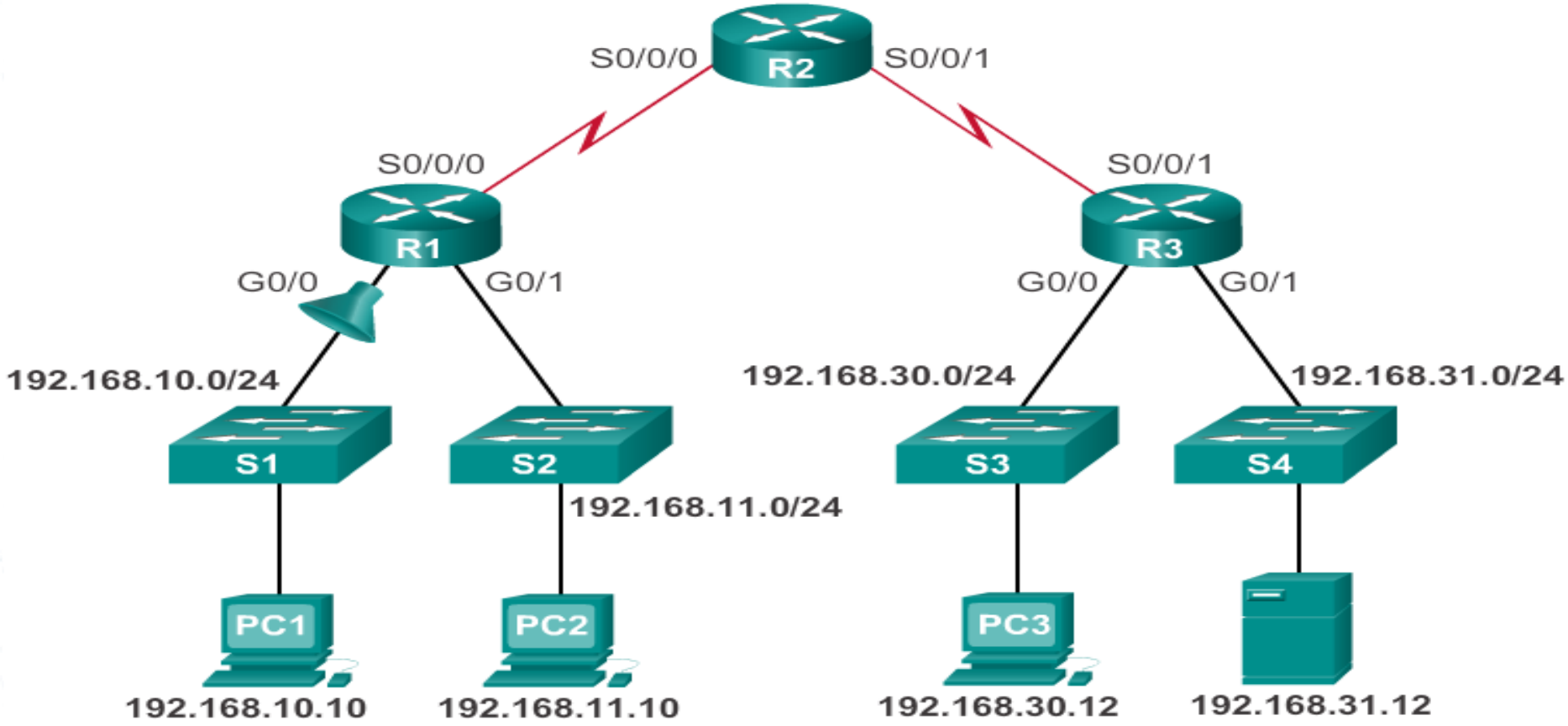
Host 192.168.10.10 has no connectivity with 192.168.30.




```
R3#show access-lists
Extended IP access list 110
    10 deny tcp 192.168.10.0 0.0.0.255 any (12 match(es))
    20 permit tcp 192.168.10.0 0.0.0.255 any eq telnet
    30 permit ip any any
```

Troubleshooting Common ACL Errors

ACL should prevent 192.168.10.0 /24 network to use SMTP to connect 192.168.30.0 /24 network.

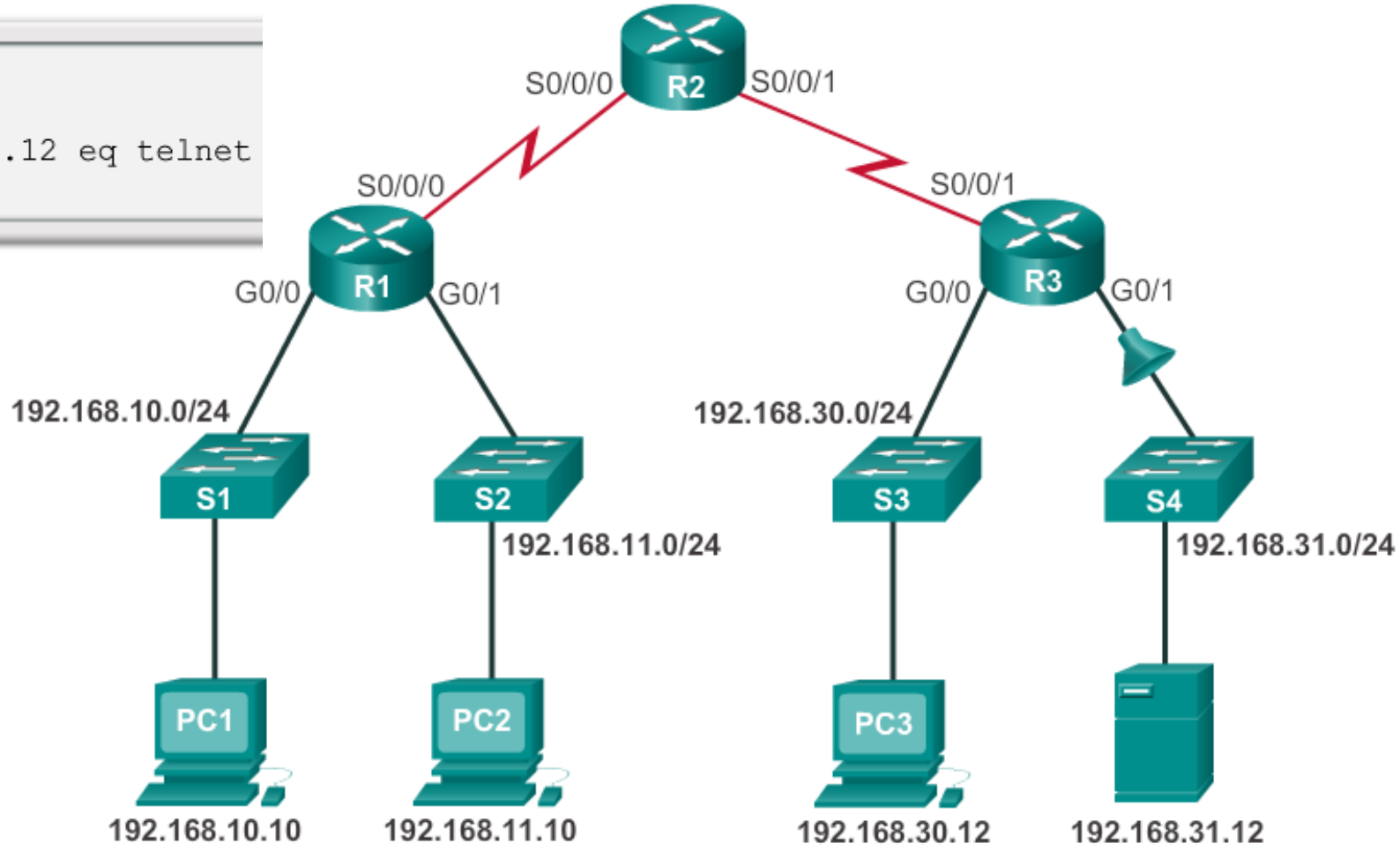



```
R1#show access-lists 120
Extended IP access list 120
    10 deny tcp 192.168.10.0 0.0.0.255 any eq telnet
    20 deny tcp 192.168.10.0 0.0.0.255 host 192.168.31.12 eq smtp
    30 permit tcp any any
```

Troubleshooting Common ACL Errors

Host 192.168.30.12 can use Telnet to connect to 192.168.31.12, but according to the security policy, this connection should not be allowed.

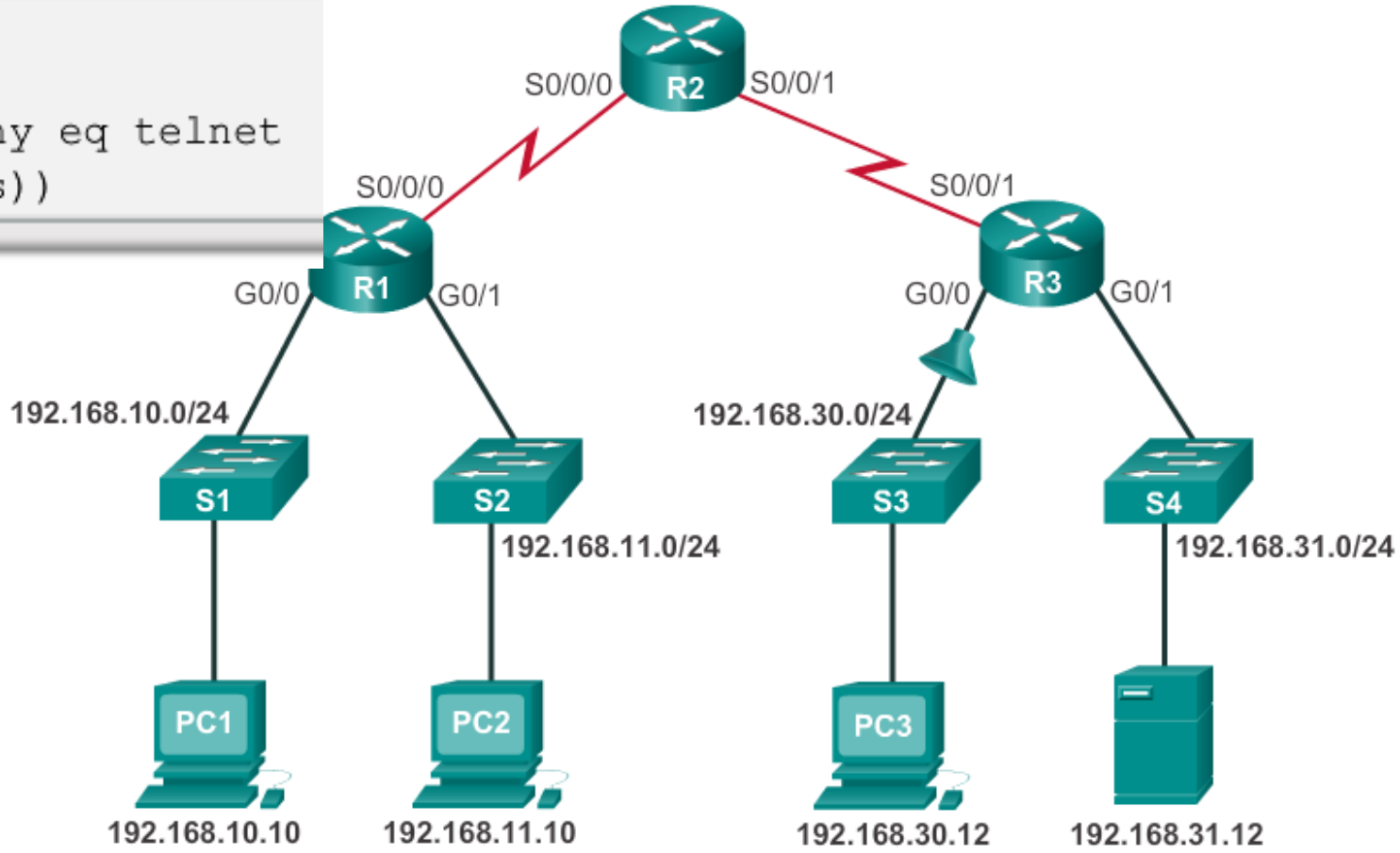
```
R2#show access-lists 150
Extended IP access list 150
    10 deny tcp any host 192.168.31.12 eq telnet
    20 permit ip any any
```



Troubleshooting Common ACL Errors

Host 192.168.30.12 use Telnet to connect to 192.168.31.12, but company policy states that this connection should not be allowed.

```
R3#show access-lists 140
Extended IP access list 140
    10 deny tcp host 192.168.30.1 any eq telnet
    20 permit ip any any (5 match(es))
```



Thank you!