



# RIP version 2 Access Control Lists

Jirasak Sittigorn

Internetworking Standards & Technologies

Department of Computer Engineering, Faculty of Engineering  
King Mongkut's Institute of Technology Ladkrabang

# RIP version 2

Introduction

RIPv1 Limitations

Configuring RIPv2

VLSM & CIDR

Verifying & Troubleshooting RIPv2

## Access Control Lists

Standard versus Extended IPv4 ACLs

Wildcard Masks in ACLs

Guidelines for ACL creation

Configure Standard IPv4 ACLs

Configure Extended IPv4 ACLs

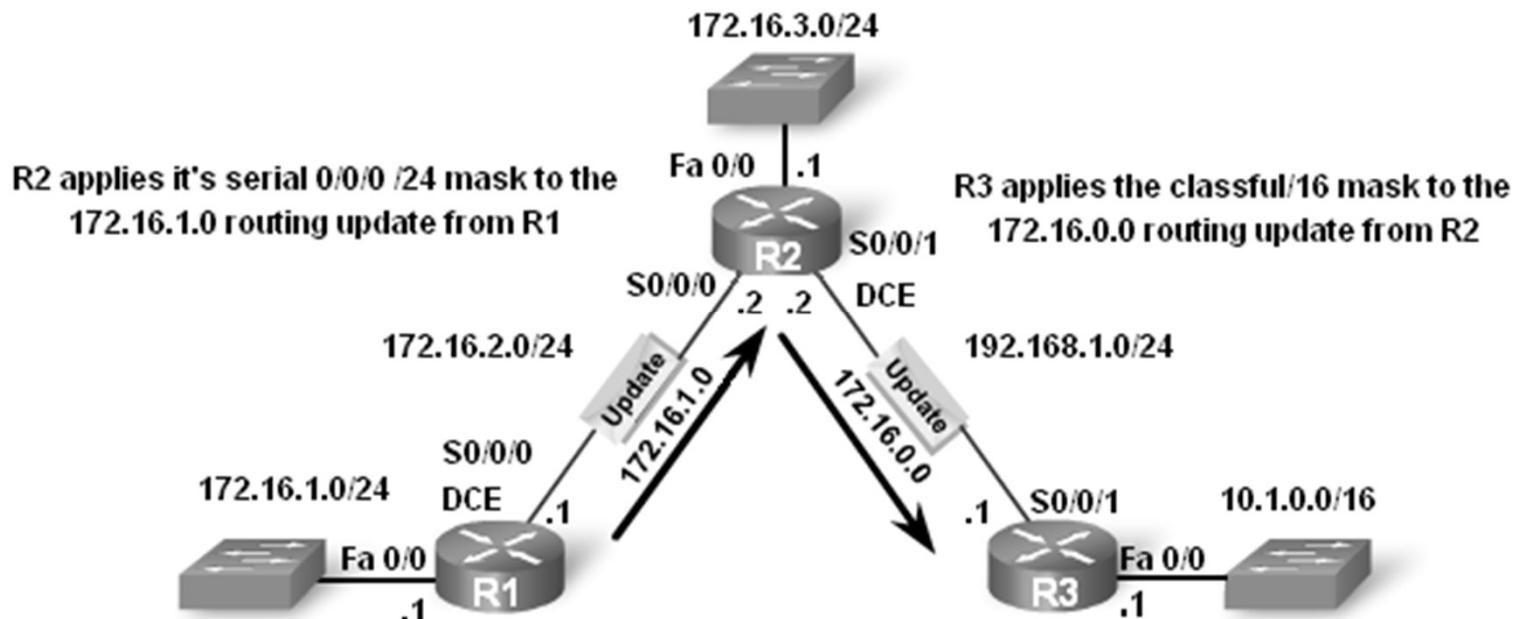
Limiting Debug / Processing Packets / Errors

# Introduction

- Difference between RIPv1 & RIPv2
  - RIPv1
    - A classful distance vector routing protocol
    - Does not support discontiguous subnets
    - Does not support VLSM
    - Does not send subnet mask in routing update
    - Routing updates are broadcast
  - RIPv2
    - A classless distance vector routing protocol that is an enhancement of RIPv1's features.
    - Next hop address is included in updates
    - Routing updates are multicast
    - The use of authentication is an option → 消息认证

# Introduction

- Classful Routing Updates
  - Recall that classful routing protocols (i.e. RIPv1) do not send subnet masks in their routing updates
    - The reason is that the Subnet mask is directly related to the network address

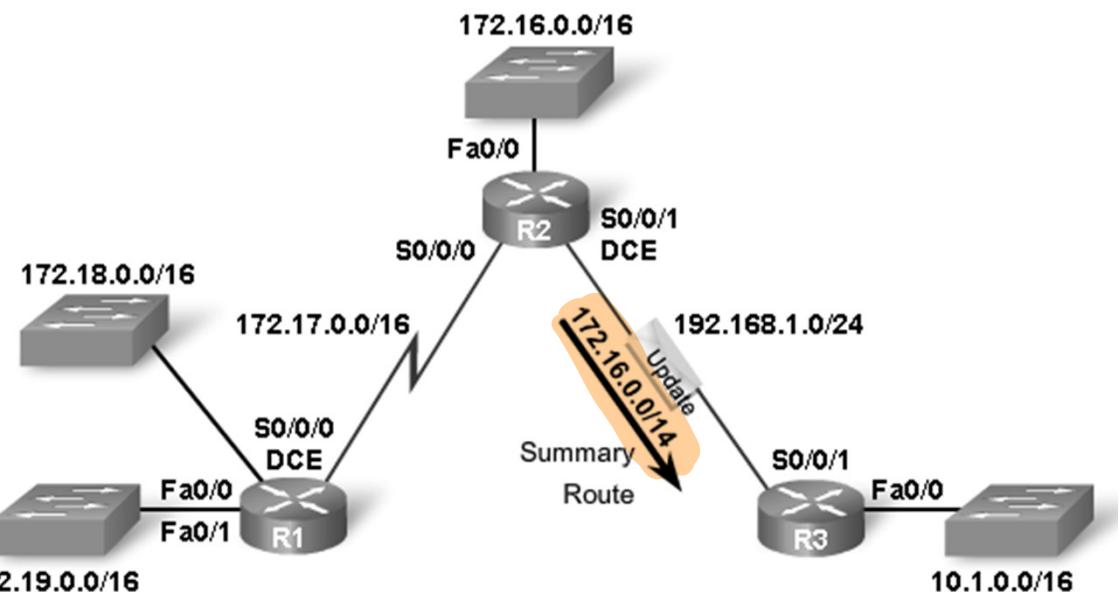


# Introduction

- Classless Routing Protocol
  - Classless Inter-domain Routing (CIDR - RFC 1517)

• Characteristics of classless routing protocols:

- Routing updates include the subnet mask
- Supports Variable Length Subnet Masking (VLSM)
- Supports Route Summarization  
(Prefix Aggregation)



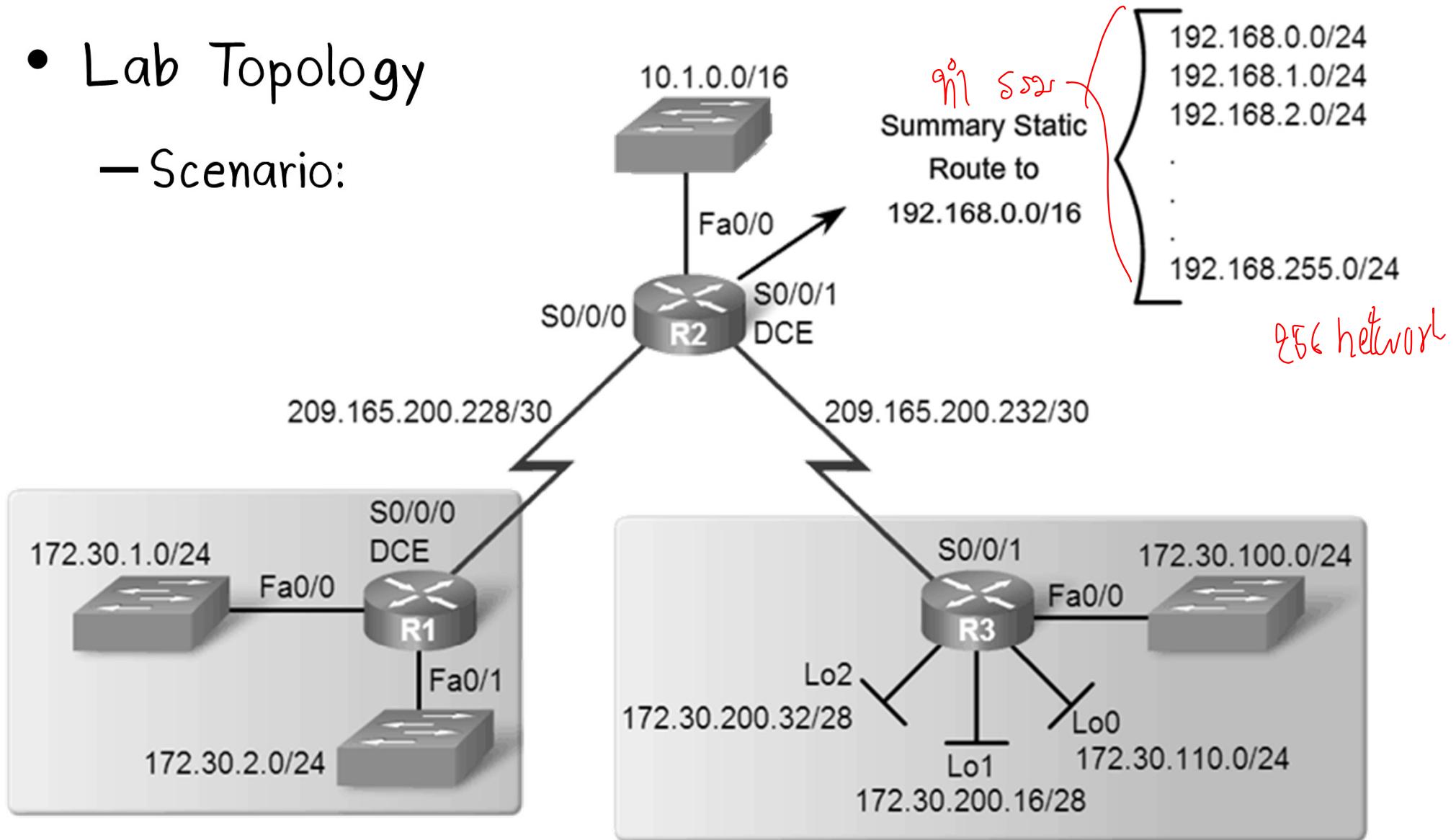
# Introduction

## • Similarities between RIPv1 & RIPv2

- Use of timers to prevent routing loops
- Use of split horizon or split horizon with poison reverse
- Use of triggered updates
- Maximum hop count of 15

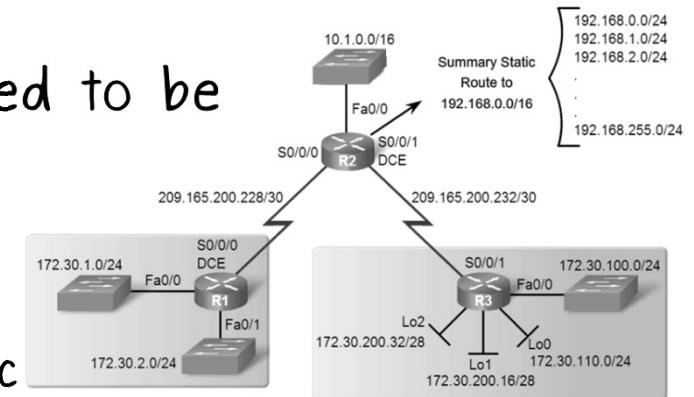
# RIPv1 Limitations

- Lab Topology
  - Scenario:



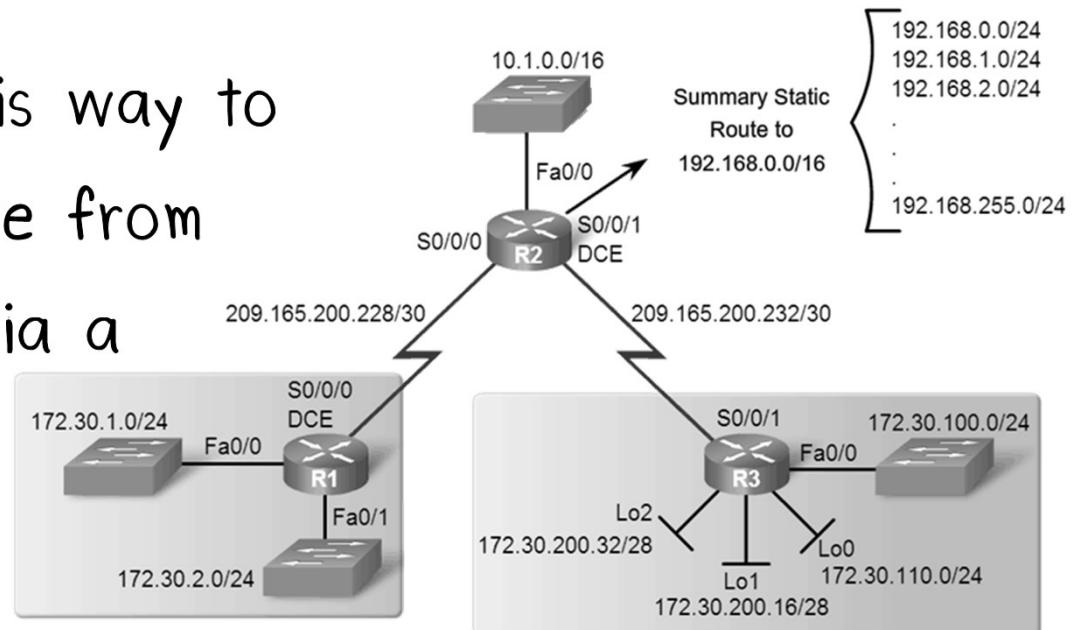
# RIPv1 Limitations

- Loopback interfaces → ping  $\text{127.0.0.1}$ 
    - These are virtual interfaces that can be pinged and added to routing table  
↳  $\text{TCP stack}$
  - Null Interfaces → ping  $\text{127.0.0.1}$ 
    - This is a virtual interface that does not need to be created or configured
      - Traffic sent to a null interface is discarded
      - Null interfaces do not send or receive traffic
  - Static routes and null interfaces
    - null interfaces will serve as the exit interface for static route
      - Example of configuring a static supernet route with a null interface
- R2 (config) #ip route 192.168.0.0 255.255.0.0 Null0**



# RIPv1 Limitations

- Route redistribution
  - Redistribution command is way to disseminate a static route from one router to another via a routing protocol
  - Example



**R2 (config-router) #redistribute static**

↴ The rip នឹង static ដីជា ផ្សេងៗ  
 តាមរបៀបនៃពារិវេល (protocol) នូវការណ៍ និង  
 static នូវ protocol ដូច

# RIP V1

```
R1(config)#router rip  
R1(config-router)# network 172.30.0.0  
R1(config-router)# network 209.165.200.0
```

```
R2(config)#ip route 192.168.0.0 255.255.0.0 null 0  
R2(config)#router rip  
R2(config-router)# network 209.165.200.0  
R2(config-router)# network 10.0.0.0  
R2(config-router)# redistribute static
```

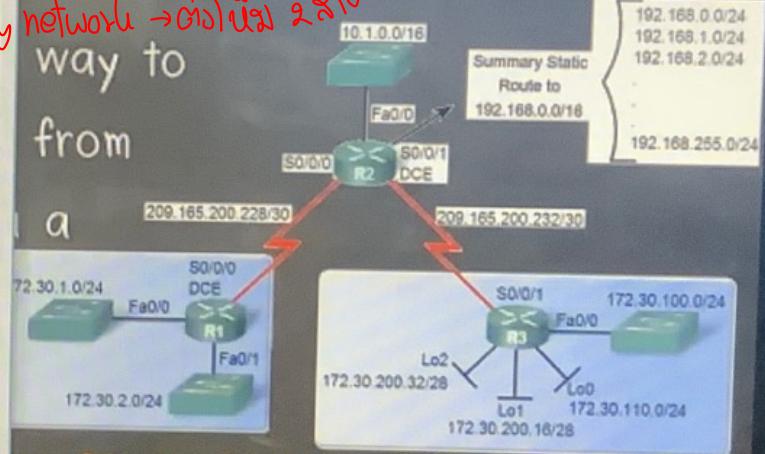
```
R3(config)#router rip  
R3(config-router)# network 172.30.0.0  
R3(config-router)# network 209.165.200.0
```

→ boundary network

→ boundary network → ចិត្តផ្លូវ 2 គេ

way to  
from

a



as (config router), "redistribute static"

r1 ping r2 ✓ នៅពេលព័ត៌មាន ip 209 → 10.1.0

រីបចកសម្រាប់ r2 នូវពេលណាន

r1 → r3 នៅពេលព័ត៌មាន r1 នឹងតែ 172.3.1.0 } នៅពេលរីបចក  
172.3.2.0 } នៅពេលរីបចក

នូវសម្រាប់ r3

# RIPv1 Limitations

- Verifying and Testing Connectivity

  - Use the following commands:

**show ip interfaces brief**

**ping**

**traceroute**

R2#show ip interface brief					
Interface	IP-Address	OK?	Method	Status	Proto
FastEthernet0/0	10.1.0.1	YES	manual	up	up
Serial0/0/0	209.165.200.229	YES	manual	up	up
FastEthernet0/1	unassigned	YES	unset	administratively down	down
Serial0/0/1	209.165.200.233	YES	manual	up	up



R1#ping 10.1.0.1

Type escape sequence to abort  
Sending 5, 100-byte ICMP Echoes to 10.1.0.1  
!!!!!

Success rate is 100 percent

✓ 100%

R1#ping 172.30.100.1

Type escape sequence to abort  
Sending 5, 100-byte ICMP Echoes to 172.30.100.1  
.....

Success rate is 0 percent

R2#ping 172.30.1.1

Type escape sequence to abort  
Sending 5, 100-byte ICMP Echoes to 172.30.1.1  
!!!!.

✓ 60%

R2#ping 172.30.100.1

Type escape sequence to abort  
Sending 5, 100-byte ICMP Echoes to 172.30.100.1  
!!!!.

Success rate is 60 percent (3/5)

R3#ping 10.1.0.1

Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echoes to 10.1.0.1  
!!!!.

✓ 100%

R3#ping 172.30.1.1

Type escape sequence to abort.  
Sending 5, 100-byte ICMP Echoes to 172.30.1.1  
.....

Success rate is 0 percent (0/5)

□ វិនិស់របស់ភ្លាមៗ, តាមលក្ខ  
អត្ថបំភាគខេត្តចាំបី suc → 60%.

↳ ពួកគេតែទាន់ដោយខ្លួន

↳ boundary net

192.168.0.0/24  
192.168.1.0/24  
192.168.2.0/24  
.  
192.168.255.0/24

# RIPv1 Limitations

- RIPv1

  - a classful routing protocol

*update with no subnet mask*

R2

Gateway of last resort is not set

```
R 172.30.0.0/16 [120/1] via 209.165.200.230, 00:00:09, Serial0/0/0
R 172.30.0.0/16 [120/1] via 209.165.200.234, 00:00:11, Serial0/0/1
C 209.165.200.0/30 is subnetted, 2 subnets
C     209.165.200.232 is directly connected, Serial0/0/1
C     209.165.200.228 is directly connected, Serial0/0/0
C 10.0.0.0/16 is subnetted, 1 subnets
C     10.1.0.0 is directly connected, FastEthernet0/0
S 192.168.0.0/16 is directly connected, Null0
```

*load balance*

  - Subnet mask are not sent in updates

*discuss here in network mgd class*

  - Summarizes networks at major network boundaries

  - if network is discontiguous and

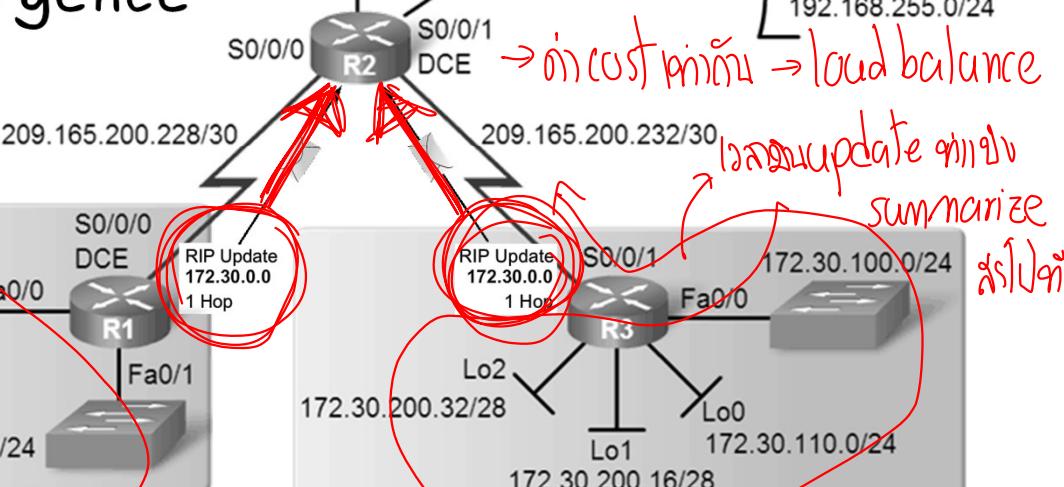
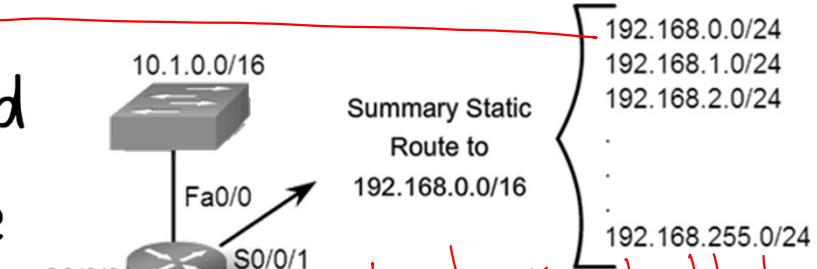
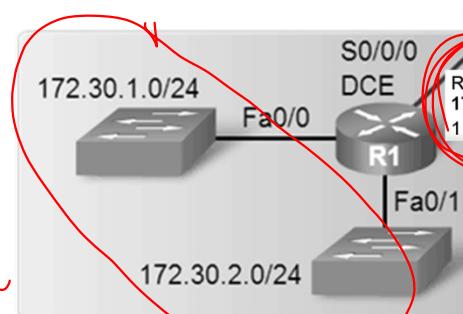
  - RIPv1 configured convergence

  - will not be reached

*No SPT*

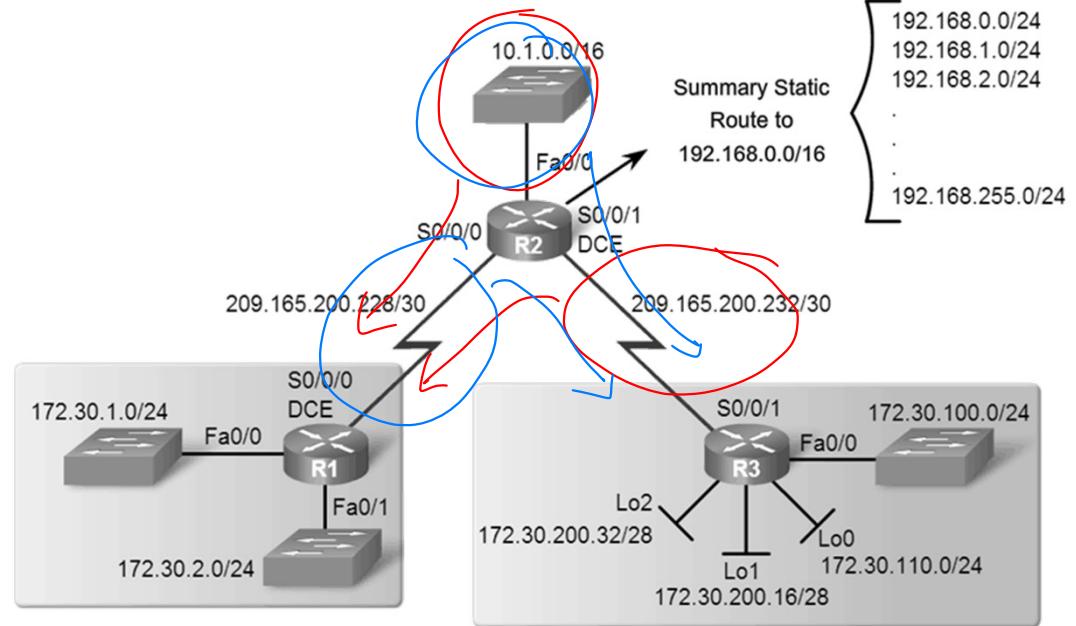
*↳ R2 update ok*  
*↳ split horizontal*

*↳ 10.1.0.0 doesn't come & check*



# RIPv1 Limitations

- Examining the routing tables
  - To examine the contents of routing updates use the **debug ip rip** command
  - If RIPv1 is configured then Subnet masks will not be included with the network address



```
R2#debug ip rip
RIP protocol debugging is on on [JUN]

RIP: received v1 update from 209.165.200.230 on Serial0/0/0
    172.30.0.0 in 1 hops
RIP: received v1 update from 209.165.200.234 on Serial0/0/1
    172.30.0.0 in 1 hops

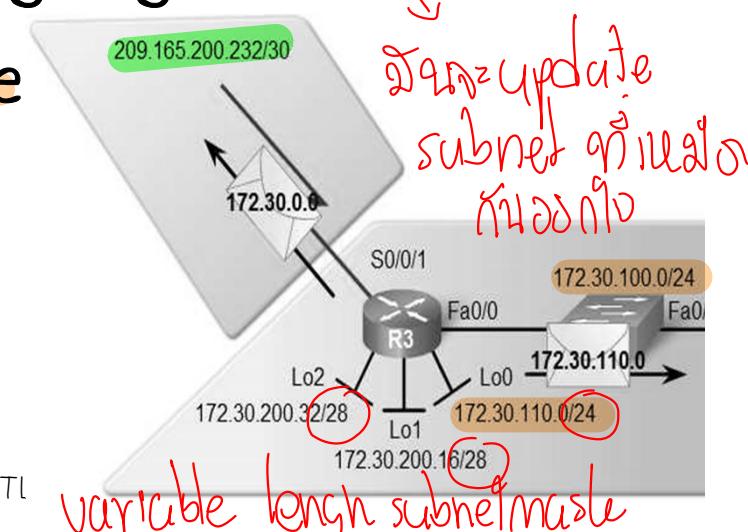
R2#
RIP: sending v1 update to 255.255.255.255 via Serial0/0/0 (209.165.200.229)
RIP: build update entries
    network 10.0.0.0 metric 1
    subnet 209.165.200.232 metric 1
RIP: sending v1 update to 255.255.255.255 via Serial0/0/1 (209.165.200.233)
RIP: build update entries
    network 10.0.0.0 metric 1
    subnet 209.165.200.228 metric 1

R2#
```

# RIPv1 Limitations

- RIPv1 does not support VLSM
  - Reason: RIPv1 does not send subnet mask in routing updates
- RIPv1 does summarize routes to the Classful boundary
  - Or uses the Subnet mask of the outgoing interface to determine which subnets to advertise

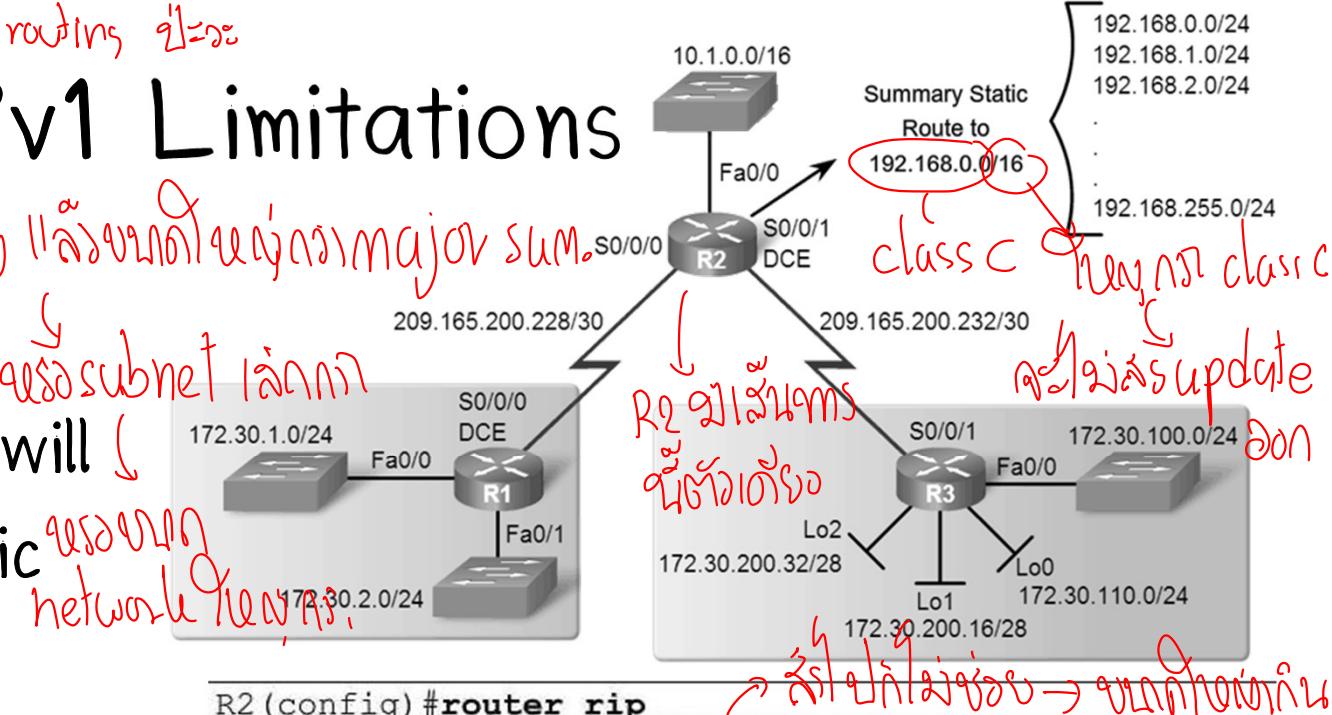
```
R3#debug ip rip
RIP protocol debugging is on
RIP: sending v1 update to 255.255.255.255 via FastEthernet0/0 (172.30.100.1)
RIP: build update entries
  network 10.0.0.0 metric 2
  subnet 172.30.110.0 metric 1
  network 209.165.200.0 metric 1
RIP: sending v1 update to 255.255.255.255 via Serial0/0/1 (209.165.200.234)
RIP: build update entries
  network 172.30.0.0 metric 1
```



classless inter domain routing չեաբ  
→ ԵԱՄՏՏ ՏԱՐԱՅԻ ԽԵՆԱԳՈՎՈՐԱԿԱՅԻ ՀԱՐԱՐԴԱՐԱՑՈՒՅԹ

# RIPv1 Limitations

- No CIDR Support
- In the diagram R2 will not include the static network 192.168.0.0/16 in its update
- Reason: Classful routing protocols do not support CIDR routes that are summarized with a smaller mask than the classful subnet mask



```
R2(config)#router rip
R2(config-router)#redistribute static
R2(config-router)#network 10.0.0.0
R2(config-router)#network 209.165.200.0
R2(config-router)#exit
R2(config)#ip route 192.168.0.0 255.255.0.0 null0
```

```
R2#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BG
(**output omitted**)
R    172.30.0.0/16 [120/1] via 209.165.200.230, 00:00:09, Serial0/0/0
                  [120/1] via 209.165.200.234, 00:00:11, Serial0/0/1
          209.165.200.0/30 is subnetted, 2 subnets
C      209.165.200.232 is directly connected, Serial0/0/1
C      209.165.200.228 is directly connected, Serial0/0/0
          10.0.0.0/16 is subnetted, 1 subnets
C          10.1.0.0 is directly connected, FastEthernet0/0
S          192.168.0.0/16 is directly connected, Null0
```

```
RIP: sending v1 update to 255.255.255.255 via Serial0/0/0 (209.165.200.229)
RIP: build update entries
      network 10.0.0.0 metric 1
      subnet 209.165.200.232 metric 1
RIP: sending v1 update to 255.255.255.255 via Serial0/0/1 (209.165.200.233)
RIP: build update entries
      network 10.0.0.0 metric 1
      subnet 209.165.200.228 metric 1
```

# Configuring RIPv2

- Comparing RIPv1 and RIPv2 Message Formats

Comparing RIPv1 and RIPv2 Message Formats

RIPv1	
Bit	0      7   8      15   16      23   24      31
	Command = 1 or 2      Version = 1      Must be zero
Route Entry {	Address family identifier (2 = IP)      Must be zero
	IP Address (Network Address)
	Must be zero
	Must be zero
	Metric (Hops)
Multiple Route Entries, up to a maximum of 25	
RIPv2	
Bit	0      7   8      15   16      23   24      31
	Command = 1 or 2      Version = 2      Must be zero
Route Entry {	Address family identifier (2 = IP)      Route Tag
	IP Address (Network Address)
	Subnet Mask
	Next Hop
	Metric (Hops)
Multiple Route Entries, up to a maximum of 25	

# Configuring RIPv2

- Enabling and Verifying RIPv2
- Configuring RIP on a Cisco router
  - By default it is running RIPv1

---

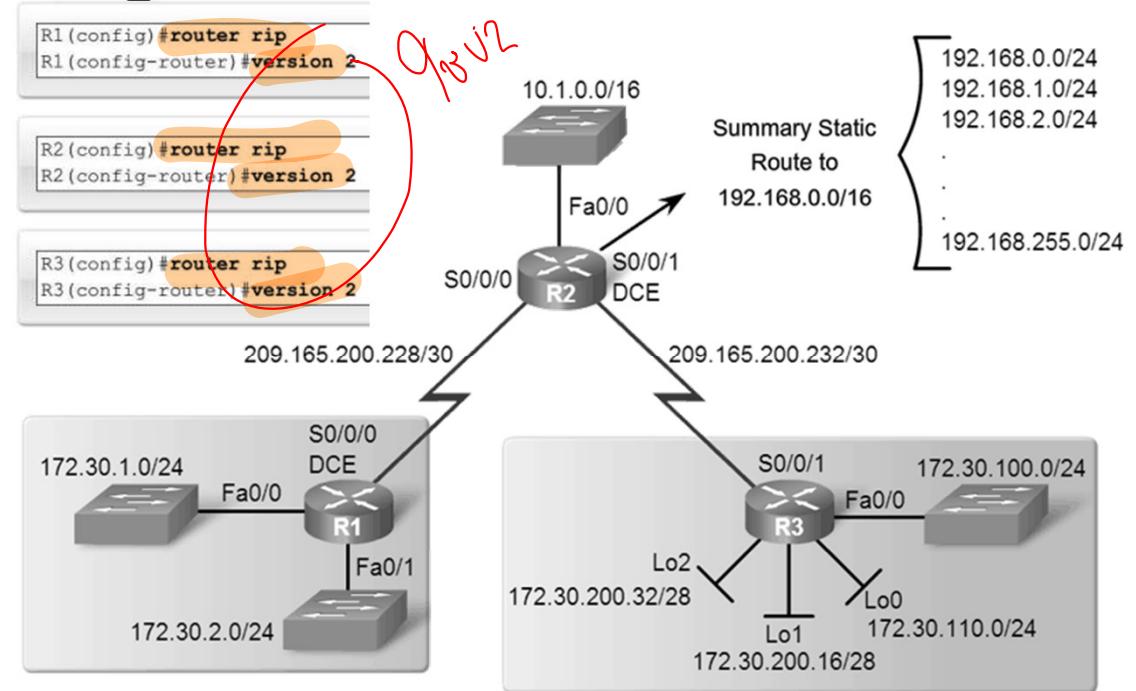
```
R2#show ip protocols
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 1 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is
  Incoming update filter list for all interfaces is
  Redistributing: static, rip
  Default version control: send version 1, receive any version
    Interface          Send   Recv   Triggered RIP  Key-chain
    Serial0/0/0         1       1 2
    Serial0/0/1         1       1 2
  Automatic network summarization is in effect
  Routing for Networks:
    10.0.0.0
    209.165.200.0
  Passive Interface(s):
    FastEthernet0/0
  Routing Information Sources:
    Gateway          Distance      Last Update
    209.165.200.234      120        00:00:03
    209.165.200.230      120        00:00:17
  Distance: (default is 120)
```

*is default*

---

# Configuring RIPv2

- Configuring RIPv2 on a Cisco router
  - Requires using the **version 2** command
  - RIPv2 ignores RIPv1 updates
- To verify RIPv2 is configured use the **show ip protocols** command



R2#**show ip protocols**

Routing Protocol is "rip"

Sending updates every 30 seconds, next due in 1 seconds  
 Invalid after 180 seconds, hold down 180, flushed after 240  
 Outgoing update filter list for all interfaces is  
 Incoming update filter list for all interfaces is  
 Redistributing: static, rip

Default version control: send version 2, receive version 2

Interface	Send	Recv	Triggered RIP	Key-chain
Serial0/0/0	2	2		<b>Only</b>
Serial0/0/1	2	2		

Automatic network summarization is in effect

Routing for Networks:

- 10.0.0.0
- 209.165.200.0

Passive Interface(s):  
 FastEthernet0/0

Routing Information Sources:

Gateway	Distance	Last Update
209.165.200.234	120	00:00:03
209.165.200.230	120	00:00:17

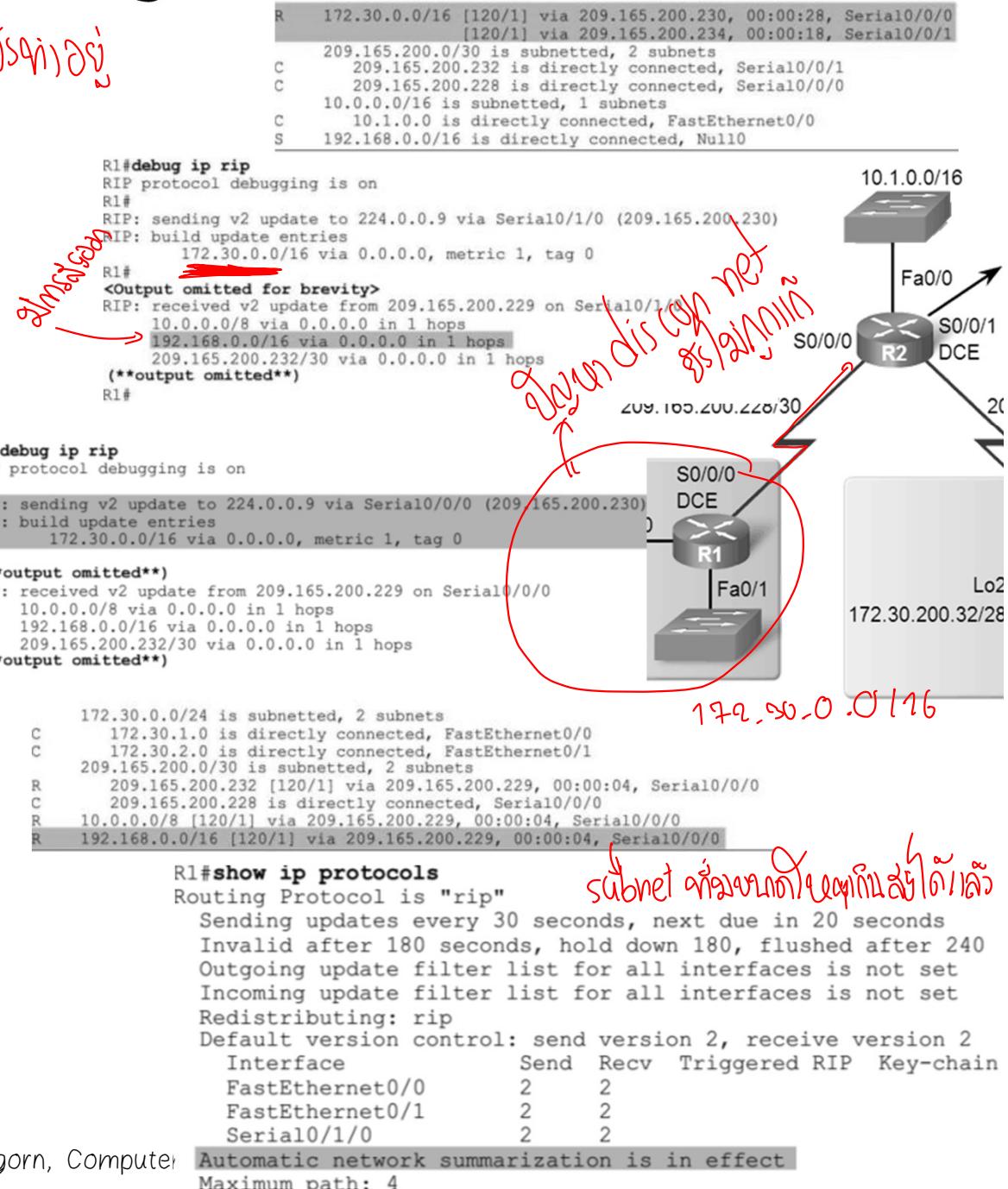
Distance: (default is 120)

*QoS V1 នៃវា V2 ។  
 និង V2 នៃវា V1 ។*

# Configuring RIPv2

at major network → จุดต่อไป

- Auto-Summary & RIPv2
- RIPv2 will automatically summarize routes at major network boundaries and can also summarize routes with a subnet mask that is smaller than the classful subnet mask

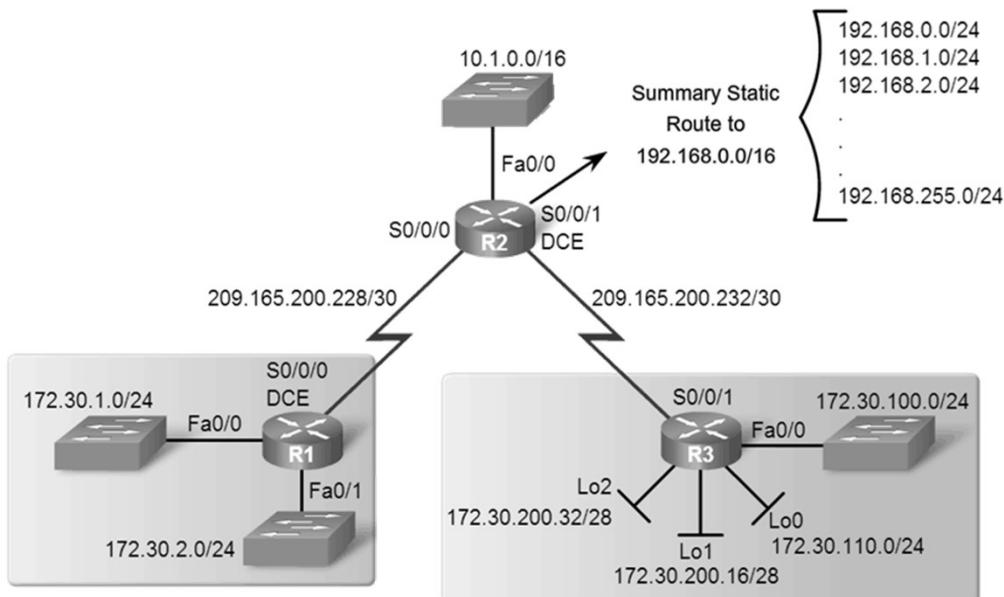


# Configuring RIPv2

- Disabling Auto-Summary in RIPv2

- To disable automatic summarization issue the **no auto-summary**

**auto-summary** command



```
R1(config)#router rip
R1(config-router)#no auto-summary
R1(config-router)#end
R1#show ip protocols
Routing Protocol is "rip"
<output omitted for brevity>
Default version control: send version 2, receive version 2
  Interface          Send  Recv  Triggered RIP  Key-chain
  FastEthernet0/0      2      2
  FastEthernet0/1      2      2
  Serial0/1/0          2      2
Automatic network summarization is not in effect
<output omitted for brevity>
```

```
R2(config)#router rip
R2(config-router)# no auto-summary
```

```
R3(config)#router rip
R3(config-router)#no auto-summary
```



RIP: sending v1 update to 255.255.255.255 via Serial0/0 (209.165.200.229)  
RIP: build update entries  
network 10.0.0.0 metric 1  
network 172.40.0.0 metric 1  
network 209.165.200.232 metric 1

RIP: sending v2 update to 224.0.0.9 via Serial0/0 (209.165.200.229)  
RIP: build update entries  
10.0.0.0/8 via 0.0.0.0, metric 1, tag 0  
172.40.0.0/16 via 0.0.0.0, metric 1, tag 0  
192.168.0.0/16 via 0.0.0.0, metric 1, tag 0  
209.165.200.232/30 via 0.0.0.0, metric 1, tag 0

RIP: sending v2 update to 224.0.0.9 via Serial0/0 (209.165.200.229)  
RIP: build update entries  
10.1.0.0/16 via 0.0.0.0, metric 1, tag 0  
172.30.100.0/24 via 0.0.0.0, metric 2, tag 0  
172.30.110.0/24 via 0.0.0.0, metric 2, tag 0  
172.30.200.16/28 via 0.0.0.0, metric 2, tag 0  
172.30.200.32/28 via 0.0.0.0, metric 2, tag 0  
172.40.0.0/16 via 0.0.0.0, metric 1, tag 0  
192.168.0.0/16 via 0.0.0.0, metric 1, tag 0  
209.165.200.232/30 via 0.0.0.0, metric 1, tag 0

มีปัญหา split horizon → ข้อห้ามการส่ง路由

ปัญหานี้ load balance

จัดการ

กำหนดค่า no auto-sum

rip2 ต้อง subsetnet  
และกำหนดค่า no-auto-sum

# VLSM & CIDR

- Verifying RIPv2 Updates **1.02.22**
- When using RIPv2 with automatic summarization turned off
  - Each subnet and mask has its own specific entry, along with the exit interface and next-hop address to reach that subnet.
- To verify information being sent by RIPv2 use the **debug ip rip** command

ឧប្បជ្ជកម្មលេខ៧

# VLSM & CIDR

R2#show ip route  
(\*\*output omitted\*\*)

Gateway of last resort is not set

```
R  172.30.0.0/16 is variably subnetted, 6 subnets, 2 masks
R    172.30.200.32/28 [120/1] via 209.165.200.234, 00:00:09, Serial0/0/1
R    172.30.200.16/28 [120/1] via 209.165.200.234, 00:00:09, Serial0/0/1
R    172.30.2.0/24 [120/1] via 209.165.200.230, 00:00:03, Serial0/0/0
R    172.30.1.0/24 [120/1] via 209.165.200.230, 00:00:03, Serial0/0/0
R    172.30.100.0/24 [120/1] via 209.165.200.234, 00:00:09, Serial0/0/1
R    172.30.110.0/24 [120/1] via 209.165.200.234, 00:00:09, Serial0/0/1

209.165.200.0/30 is subnetted, 2 subnets
C      209.165.200.232 is directly connected, Serial0/0/1
C      209.165.200.228 is directly connected, Serial0/0/0
10.0.0.0/16 is subnetted, 1 subnets
C        10.1.0.0 is directly connected, FastEthernet0/0
S  192.168.0.0/16 is directly connected, Null0
```

R2#debug ip rip  
RIP protocol debugging is on  
(\*\*output omitted\*\*)

```
R2#
RIP: received v2 update from 209.165.200.234 on Serial0/0/1
  172.30.100.0/24 via 0.0.0.0 in 1 hops
  172.30.110.0/24 via 0.0.0.0 in 1 hops
  172.30.200.16/28 via 0.0.0.0 in 1 hops
  172.30.200.32/28 via 0.0.0.0 in 1 hops

R2#
RIP: sending v2 update to 224.0.0.9 via Serial0/0/0 (209.165.200.229)
RIP: build update entries
  10.1.0.0/16 via 0.0.0.0, metric 1, tag 0
  172.30.100.0/24 via 0.0.0.0, metric 2, tag 0
  172.30.110.0/24 via 0.0.0.0, metric 2, tag 0
  172.30.200.16/28 via 0.0.0.0, metric 2, tag 0
  172.30.200.32/28 via 0.0.0.0, metric 2, tag 0
  192.168.0.0/16 via 0.0.0.0, metric 1, tag 0
  209.165.200.232/30 via 0.0.0.0, metric 1, tag 0

R2#
```

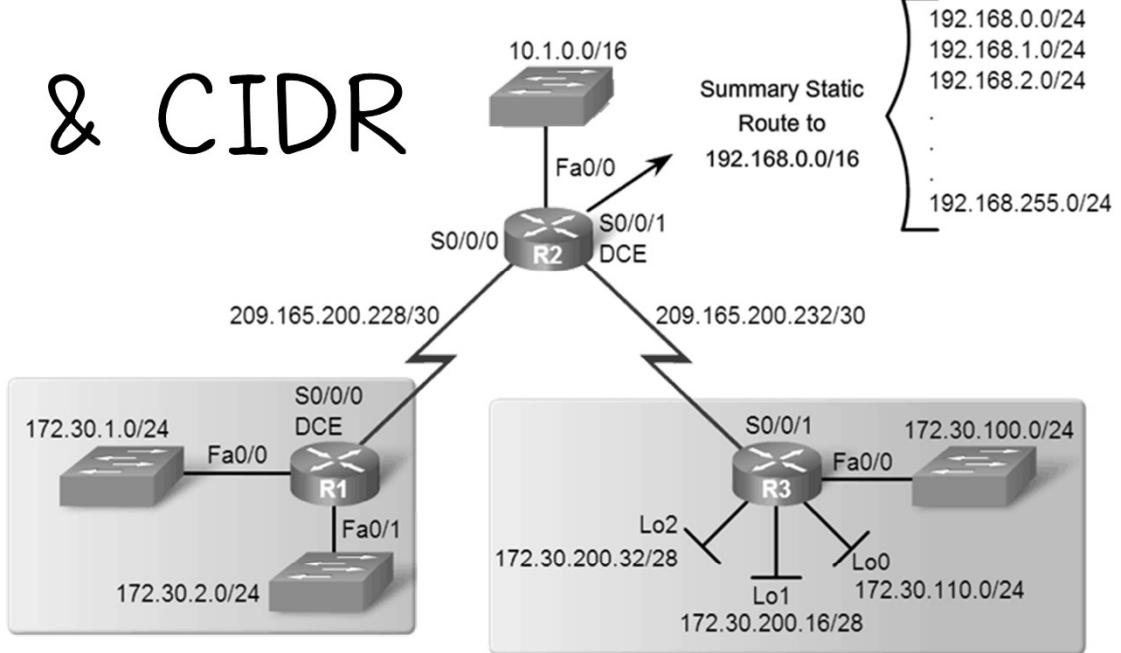
R1#show ip route  
(\*\*output omitted\*\*)

Gateway of last resort is not set

```
172.30.0.0/16 is variably subnetted, 6 subnets, 2 masks
R    172.30.200.32/28 [120/2] via 209.165.200.229, 00:00:01, Serial0/0/0
R    172.30.200.16/28 [120/2] via 209.165.200.229, 00:00:01, Serial0/0/0
C    172.30.1.0/24 is directly connected, FastEthernet0/0
C    172.30.2.0/24 is directly connected, FastEthernet0/1
R    172.30.100.0/24 [120/2] via 209.165.200.229, 00:00:01, Serial0/0/0
R    172.30.110.0/24 [120/2] via 209.165.200.229, 00:00:01, Serial0/0/0

209.165.200.0/30 is subnetted, 2 subnets
R      209.165.200.232 [120/1] via 209.165.200.229, 00:00:02, Serial0/0/0
C      209.165.200.228 is directly connected, Serial0/0/0
10.0.0.0/16 is subnetted, 1 subnets
R        10.1.0.0 [120/1] via 209.165.200.229, 00:00:02, Serial0/0/0
R  192.168.0.0/16 [120/1] via 209.165.200.229, 00:00:02, Serial0/0/0
```

orn, Computer



R3#show ip route  
(\*\*output omitted\*\*)

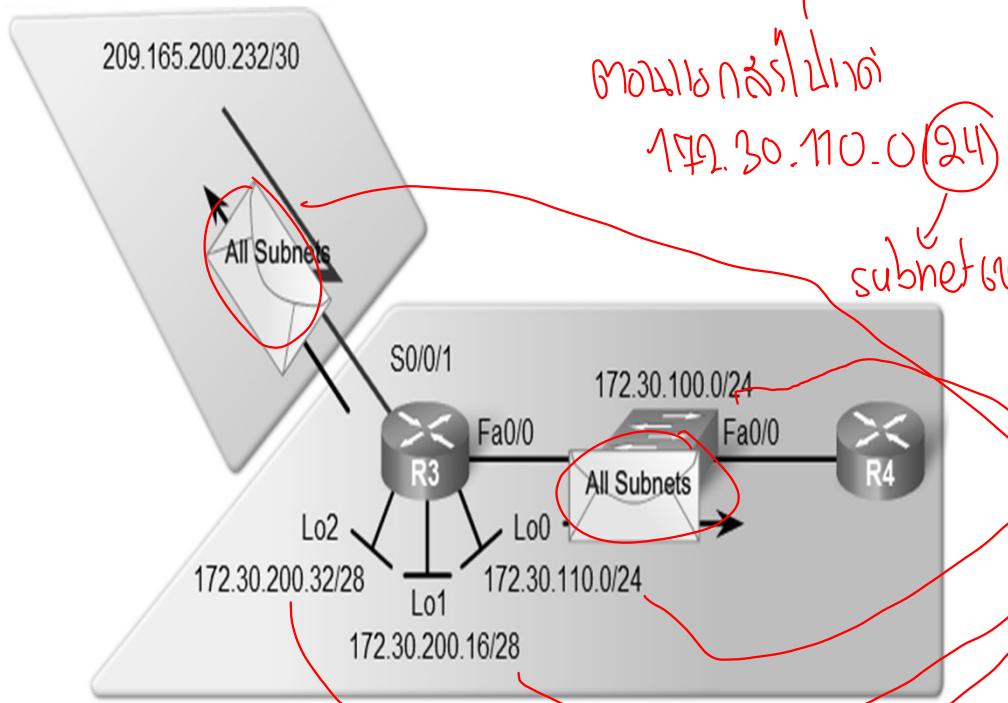
Gateway of last resort is not set

```
172.30.0.0/16 is variably subnetted, 6 subnets, 2 masks
C    172.30.200.32/28 is directly connected, Loopback2
C    172.30.200.16/28 is directly connected, Loopback1
R    172.30.2.0/24 [120/2] via 209.165.200.233, 00:00:01, Serial0/0/1
R    172.30.1.0/24 [120/2] via 209.165.200.233, 00:00:01, Serial0/0/1
C    172.30.100.0/24 is directly connected, FastEthernet0/0
C    172.30.110.0/24 is directly connected, Loopback0
209.165.200.0/30 is subnetted, 2 subnets
C      209.165.200.232 is directly connected, Serial0/0/1
R      209.165.200.228 [120/1] via 209.165.200.233, 00:00:02, Serial0/0/1
10.0.0.0/16 is subnetted, 1 subnets
R        10.1.0.0 [120/1] via 209.165.200.233, 00:00:02, Serial0/0/1
R  192.168.0.0/16 [120/1] via 209.165.200.233, 00:00:02, Serial0/0/1
```

# VLSM & CIDR

- RIPv2 and VLSM
  - Networks using a VLSM IP addressing scheme
    - Use classless routing protocols (i.e. RIPv2) to disseminate network addresses and their subnet masks

ripv2 ဆုတေသန subnet mask

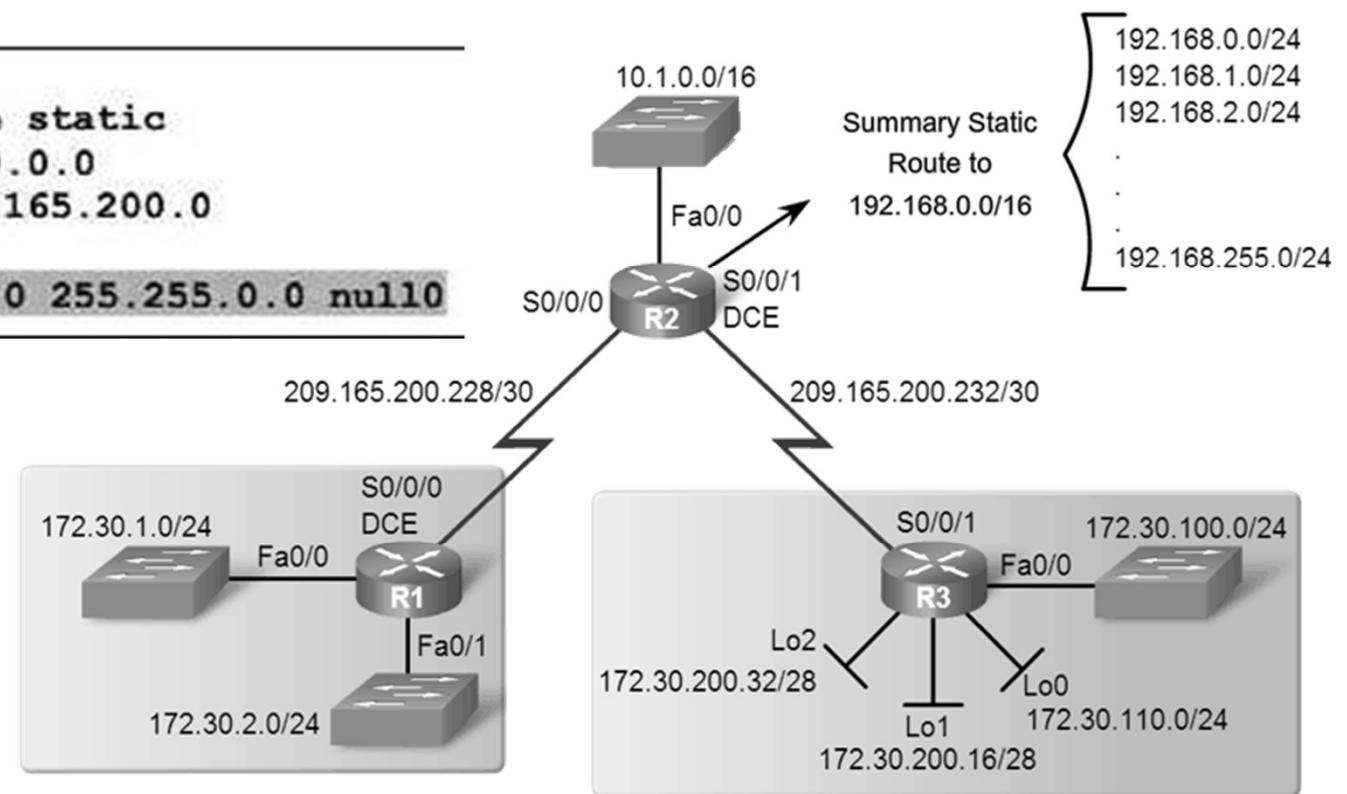


```
R3#debug ip rip
RIP protocol debugging is on
R3#
RIP: received v2 update from 209.165.200.233 on Serial0/0/1
  10.1.0.0/16 via 0.0.0.0 in 1 hops
  172.30.1.0/24 via 0.0.0.0 in 2 hops
  172.30.2.0/24 via 0.0.0.0 in 2 hops
  192.168.0.0/16 via 0.0.0.0 in 1 hops
  209.165.200.228/30 via 0.0.0.0 in 1 hops
R3#
RIP: sending v2 update to 224.0.0.9 via FastEthernet0/0 (172.30.100.1)
RIP: build update entries
  10.1.0.0/16 via 0.0.0.0, metric 2, tag 0
  172.30.1.0/24 via 0.0.0.0, metric 3, tag 0
  172.30.2.0/24 via 0.0.0.0, metric 3, tag 0
  172.30.110.0/24 via 0.0.0.0, metric 1, tag 0
  172.30.200.16/28 via 0.0.0.0, metric 1, tag 0
  172.30.200.32/28 via 0.0.0.0, metric 1, tag 0
  192.168.0.0/16 via 0.0.0.0, metric 2, tag 0
  209.165.200.228/30 via 0.0.0.0, metric 2, tag 0
  209.165.200.232/30 via 0.0.0.0, metric 1, tag 0
RIP: sending v2 update to 224.0.0.9 via Serial0/0/1 (209.165.200.234)
RIP: build update entries
  172.30.100.0/24 via 0.0.0.0, metric 1, tag 0
  172.30.110.0/24 via 0.0.0.0, metric 1, tag 0
  172.30.200.16/28 via 0.0.0.0, metric 1, tag 0
  172.30.200.32/28 via 0.0.0.0, metric 1, tag 0
```

# VLSM & CIDR

- CIDR uses Supernetting
  - Supernetting is a bunch of contiguous classful networks that is addressed as a single network.

```
R2 (config) #router rip
R2 (config-router) #redistribute static
R2 (config-router) #network 10.0.0.0
R2 (config-router) #network 209.165.200.0
R2 (config-router) #exit
R2 (config) #ip route 192.168.0.0 255.255.0.0 null0
```



# VLSM & CIDR

- To verify that supernets are being sent and received use the following commands

```
show ip route
debug ip rip
```

```
R1#show ip route
<output omitted>
```

Gateway of last resort is not set

```
R   172.30.0.0/16 is variably subnetted, 6 subnets, 2 masks
R     172.30.200.32/28 [120/2] via 209.165.200.229, 00:00:01, Serial0/0/0
R     172.30.200.16/28 [120/2] via 209.165.200.229, 00:00:01, Serial0/0/0
C     172.30.1.0/24 is directly connected, FastEthernet0/0
C     172.30.2.0/24 is directly connected, FastEthernet0/1
R     172.30.100.0/24 [120/2] via 209.165.200.229, 00:00:01, Serial0/0/0
R     172.30.110.0/24 [120/2] via 209.165.200.229, 00:00:01, Serial0/0/0
  209.165.200.0/30 is subnetted, 2 subnets
R       209.165.200.232 [120/1] via 209.165.200.229, 00:00:02, Serial0/0/0
C       209.165.200.228 is directly connected, Serial0/0/0
  10.0.0.0/16 is subnetted, 1 subnets
R     10.1.0.0 [120/1] via 209.165.200.229, 00:00:02, Serial0/0/0
R   192.168.0.0/16 [120/1] via 209.165.200.229, 00:00:02, Serial0/0/0
```

```
R2#debug ip rip
RIP protocol debugging is on
R2#
RIP: sending v2 update to 224.0.0.9 via Serial0/0/0 (209.165.200.229)
RIP: build update entries
  10.1.0.0/16 via 0.0.0.0, metric 1, tag 0
  172.30.100.0/24 via 0.0.0.0, metric 2, tag 0
  172.30.110.0/24 via 0.0.0.0, metric 2, tag 0
  172.30.200.16/28 via 0.0.0.0, metric 2, tag 0
  172.30.200.32/28 via 0.0.0.0, metric 2, tag 0
  192.168.0.0/16 via 0.0.0.0, metric 1, tag 0
  209.165.200.232/30 via 0.0.0.0, metric 1, tag 0
<output omitted for brevity>
R2#
```

# Verifying & Troubleshooting RIPv2

- Basic Troubleshooting steps
  - Check the status of all links
  - Check cabling
  - Check IP address & subnet mask configuration
  - Remove any unneeded configuration commands → ស្តីពីរាយការណ៍នេះ

- Commands used to verify proper operation of RIPv2

**show ip interfaces brief**

**show ip protocols**

**debug ip rip**

**show ip route**

# Verifying & Troubleshooting RIPv2

- Common RIPv2 Issues
- When trouble shooting RIPv2 examine the following issues:
  - Version
    - Check to make sure you are using version 2
  - Network statements
    - Network statements may be incorrectly typed or missing
  - Automatic summarization
    - If summarized routes are not needed then disable automatic summarization

គ្រប់គ្រងផ្លូវការណា

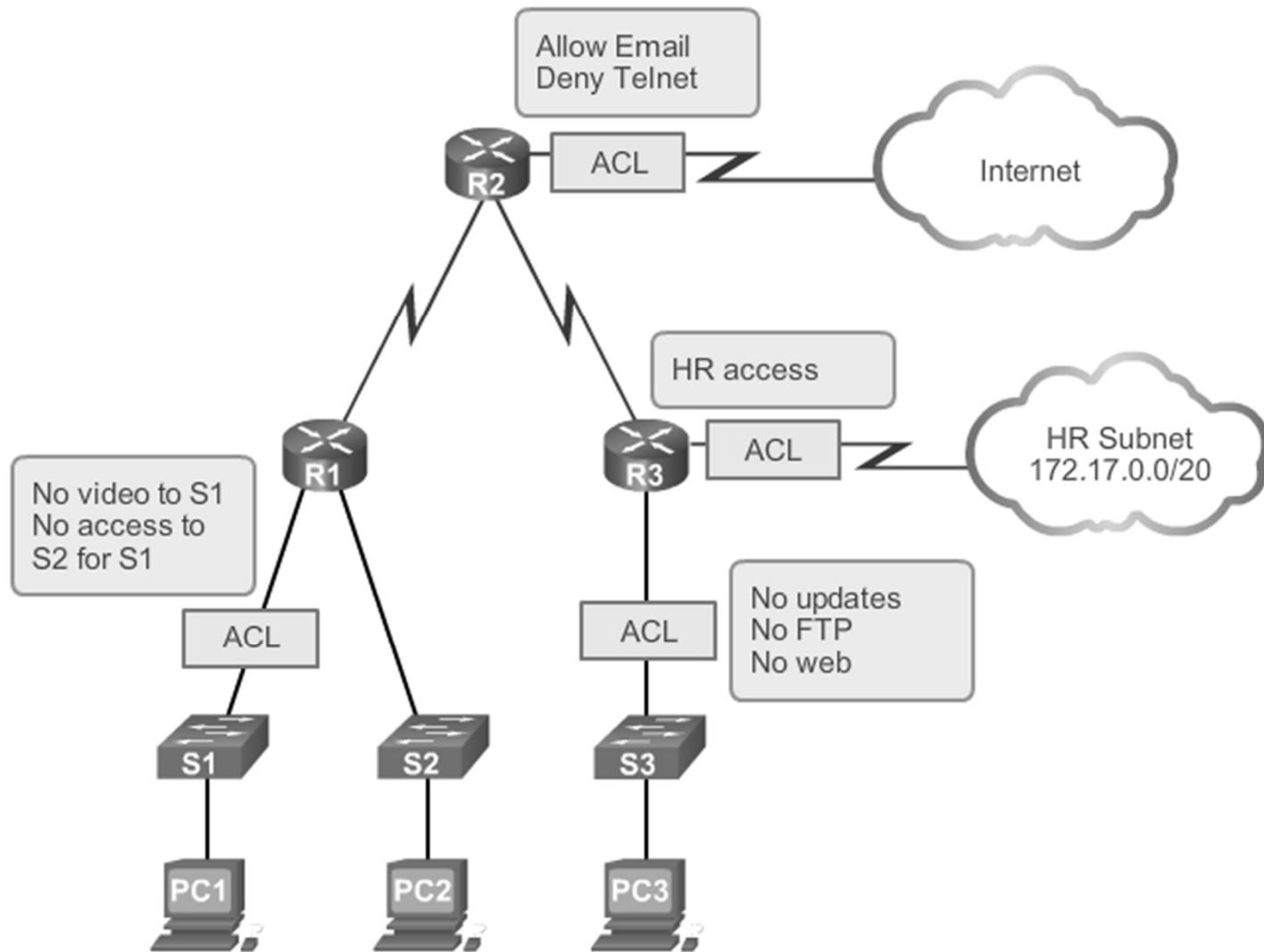
# Verifying & Troubleshooting RIPv2

msauthen

- Reasons why it's good to authenticate routing information
  - Prevent the possibility of accepting invalid routing updates
  - Contents of routing updates are encrypted
- Types of routing protocols that can use authentication
  - RIPv2
  - EIGRP
  - OSPF
  - IS-IS
  - BGP

# Access Control Lists

- What is an ACL?

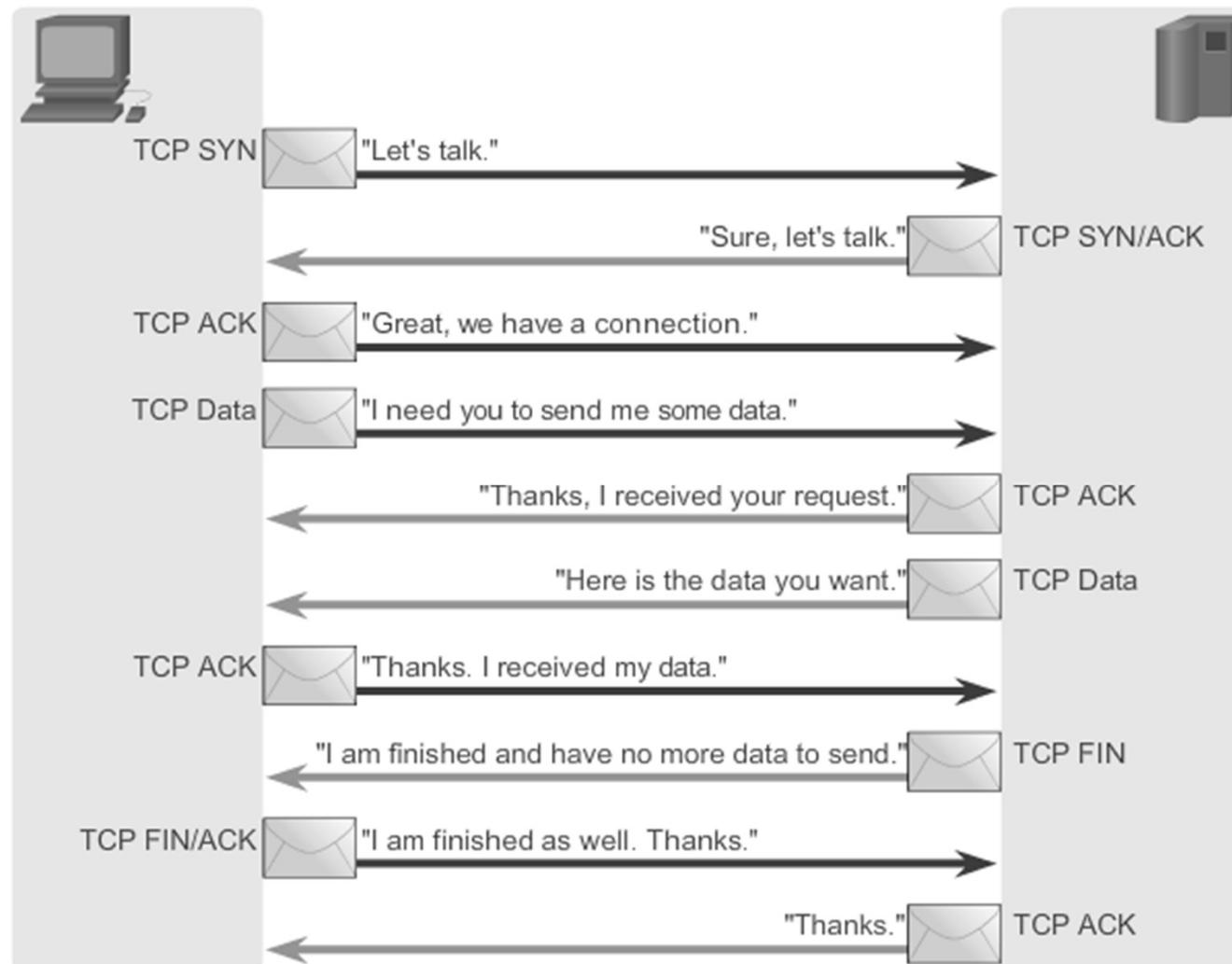


သိပ်ဆောင်ရွက်မှုတော်လောင်  
အနေဖြင့်လျော့ဝါယာတော်လောင်ပေးပို့မှု  
လုပ်မှုတော်, ၀၁၀ ရွှေ့လောင်  
စွမ်းကိုယ်စွမ်းလောင်ပေးပို့မှု

# Access Control Lists

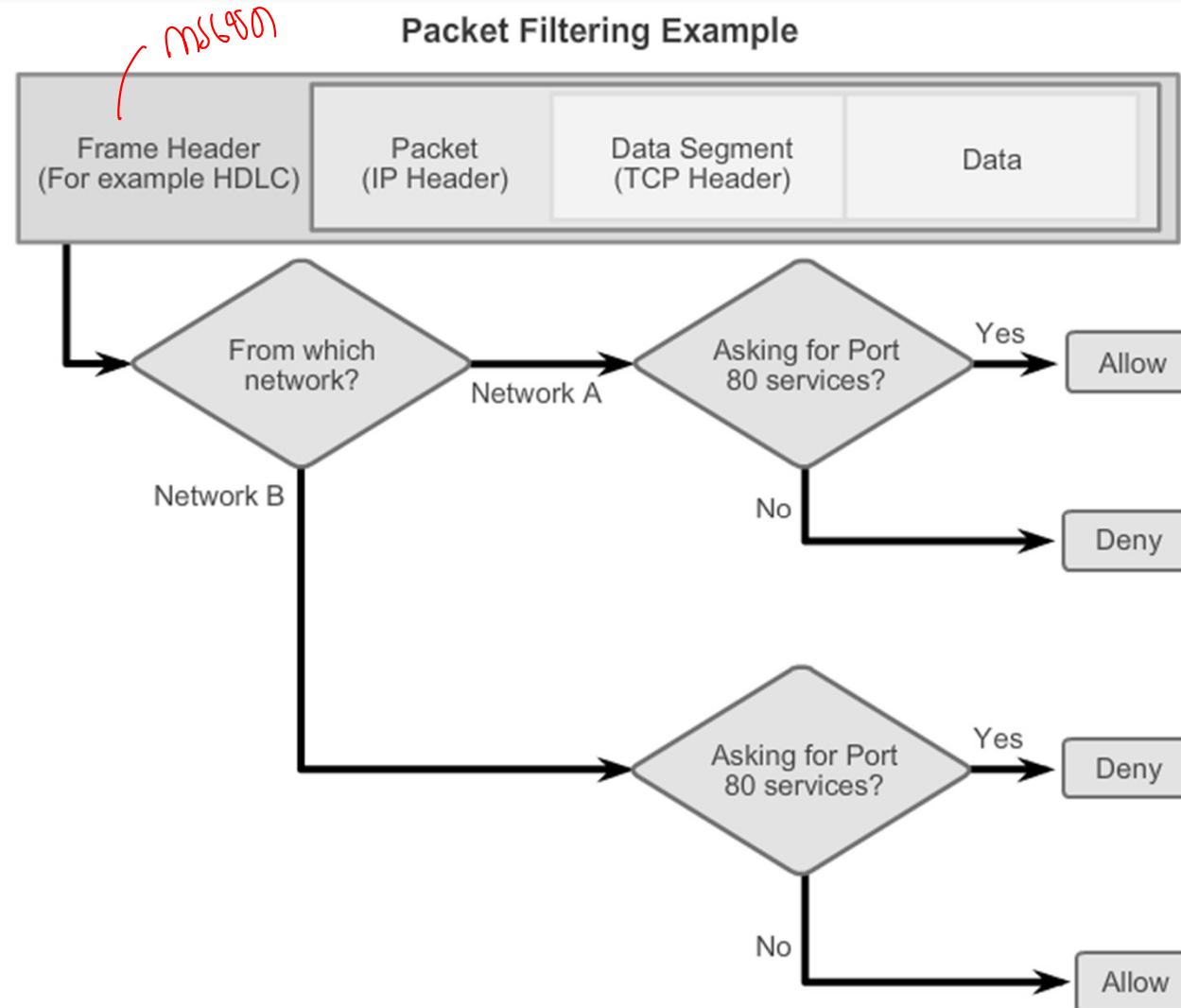
- A TCP Conversation

*Message network layer → Layer 3 tcp protocol*



# Access Control Lists

- Packet Filtering



# Access Control Lists

- ACL Operation

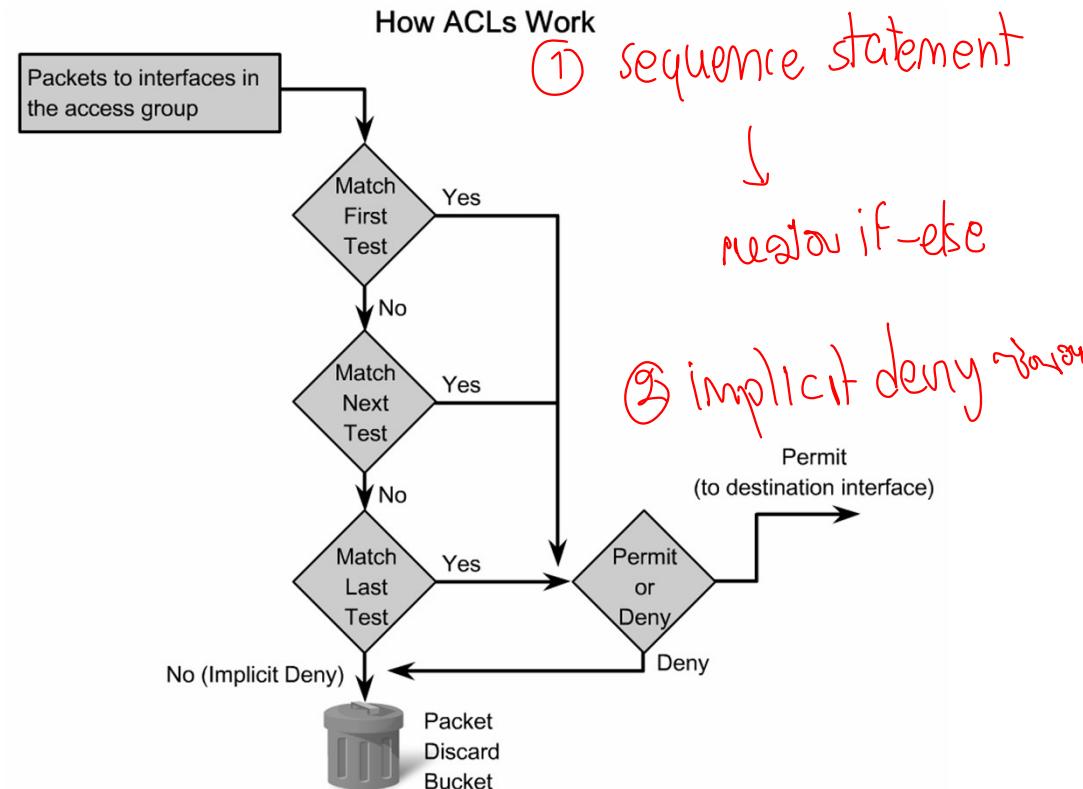
— The last statement of an ACL is always an implicit deny. This statement is automatically inserted at the end of each ACL even though it is not physically present. The implicit deny blocks all traffic. Because of this implicit deny, an ACL that does not have at least one permit statement will block all traffic.

ถ้าไม่พบ rule ให้ deny ถ้ามี rule ให้ permit



An inbound ACL filters packets coming into a specific interface and before they are routed to the outbound interface.

An outbound ACL filters packets after being routed, regardless of the inbound interface.



# Standard versus Extended IPv4 ACLs

มาตรฐาน acl

- Standard ACLs

- Checks source address

- Generally permits or denies entire protocol suite

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

ตรวจสอบ源 IP

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

deny, permit protocol ทำได้

- Extended ACLs

- Checks source and destination address

- Generally permits or denies specific protocols

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

กำหนด rule ให้ได้

กำหนด http

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, ICMP, UDP, TCP, etc.)

# Standard versus Extended IPv4 ACLs

- Numbering and Naming ACLs යුතු name කරයුතුව

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.

# Wildcard Masks in ACLs

↳ ດັວວິທີກົງ subnet mask → subnet mask ໄວ້

Octet Bit Position and Address Value for Bit

128	64	32	16	8	4	2	1
↓	↓	↓	↓	↓	↓	↓	↓
0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1
0	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1

Examples

1 mask

ໄລຍະນຸ້ມືຂຶ້ນ 0 mask

= Match All Address Bits (Match All)

= Ignore Last 6 Address Bits

= Ignore Last 4 Address Bits

= Ignore First 6 Address Bits

= Ignore All Bits in Octet

0 means to match the value of the corresponding address bit

1 means to ignore the value of the corresponding address bit

# Wildcard Masks in ACLs

- 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

ด้านขวา 0 หมายความว่า mask  
0 หมายความว่า ไม่สนใจที่นี่

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

ทุก bits เท่ากับ 1  
ip address ก็

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

mask & bit ที่ 0-255  
192.168.1.0  
XXXX XXXX 0-255

# Wildcard Masks in ACLs

- 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

0001 XXXX.XXXXX

192.168.16.0 – 192.168.31.255

Example 2

	Decimal	Binary
IP Address	192.168.1.0	11000000.10101000.00000001.00000000
Wildcard Mask	0.0.254.255	00000000.00000000.1111110.11111111
Result	192.168.1.0	11000000.10101000.00000001.00000000
All odd numbered subnets in the 192.168.0.0 major network		

.XXXX XXX1.XXXXXXXX

192.168.1.0/24 192.168.3.0/24 – 192.168.5.0/24

តាមកំណែ 192.16.64.0 - 192.16.191.255 នូវ  
0.0.0.0 wildcard mask

ចំណាំ

អង្គភាពផ្លូវ

192.16.64.0 - 192.16.191.255

ip

wildcard mask

មតិក

0000 0000 . 0000 0000 . 00111111 . 11111111  
1010 1100 . 0001 0000 . 01XXXXXX

ʃ

មិនអាចរាយ max, min ឡើង បានត្រួតពី

XXXX

XXXX

ឧបករណ៍ 192.16.64.0 - 192.16.191.255

min 192.16.0100 0000.0000 0000

max 192.16.1011 1111.1111 1111

គានមុខងារ (សេចក្តីផលនៅ)

0.0.0.255 . 255 → 11010101 តាមរបៀបនេះ → 192.16.XX )

ចិត្តបញ្ជី 2 statement

និង 2 នាម 192.16.64.0 ក្នុង 0.0.63.255 និង 192.16.128.0 ក្នុង 0.0.63.255

0011 1111

1000 0000

↙

↙

64 - 191

128 - 191

Untitled - Notepad

File Edit Format View Help

10.1.64.128-255  
10.1.66.128-255  
10.1.68.128-255  
10.1.70.128-255  
10.1.72.128-255  
10.1.74.128-255  
10.1.76.128-255  
10.1.78.128-255

ip 10.1.0100 0000.1000 0000

wildcard 0.0.00001110.0111 1111

0100 XXX0 . 0 XXX XXXX

# Wildcard Masks in ACLs

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

សែនីមុន្តែង subnet → wildcard

ឬ 255.255.255.255 តើ -

subnet ចាន់

យើរ wildcard នេះ

Example 1

255.255.255.255	① បន្ទាំង
- 255.255.255.000	② subnet mask
000.000.000.255	③ wildcard mask

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 Masks in ACLs

- 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**)

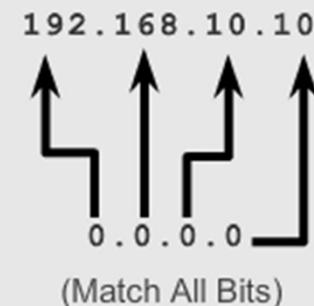
# 192.168.10.10 0.0.0.0

=

host 192.168.10.10

ກີ່ສະນັກ ດີ່ມັສຸກ ອົບ ບົດ

Wildcard Mask:



## Example 2

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

ignore ອົບ ບົດ

any

ກີ່ສະນັກ ຫາ ຖິຫາ

Wildcard Mask:



# Wildcard Masks in ACLs

- Examples Wildcard Mask Keywords

Example 1:

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

กำหนดการ์ด

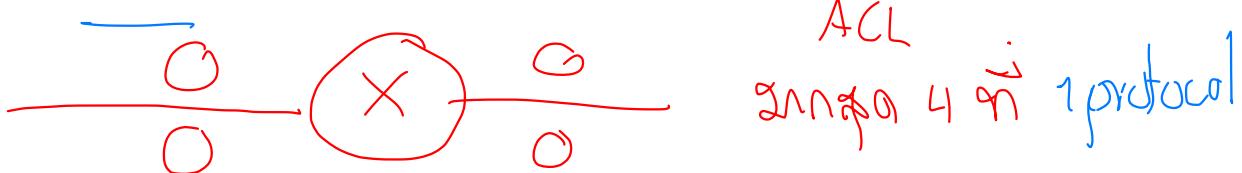
Example 2:

```
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
```

# Guidelines for ACL creation

- Use ACLs in firewall routers positioned between your internal network and an external network such as the Internet.
- Use ACLs on a router positioned between two parts of your network to control traffic entering or exiting a specific part of your internal network.
- Configure ACLs on border routers, that is routers situated at the edges of your networks.
- Configure ACLs for each network protocol configured on the border router interfaces.

# Guidelines for ACL creation



- The Three Ps
- 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.

# Guidelines for ACL creation

- ACL Best Practices

Guideline	Benefit
Base your ACLs on the security policy of the organization.	This will ensure you implement organizational security guidelines.
Prepare a description of what you want your ACLs to do.	This will help you avoid inadvertently creating potential access problems.
Use a text editor to create, edit and save ACLs.	This will help you create a library of reusable ACLs.
Test your ACLs on a development network before implementing them on a production network.	This will help you avoid costly errors.

*minimum 3 reasons*

*test now deploy*

# Guidelines for ACL creation

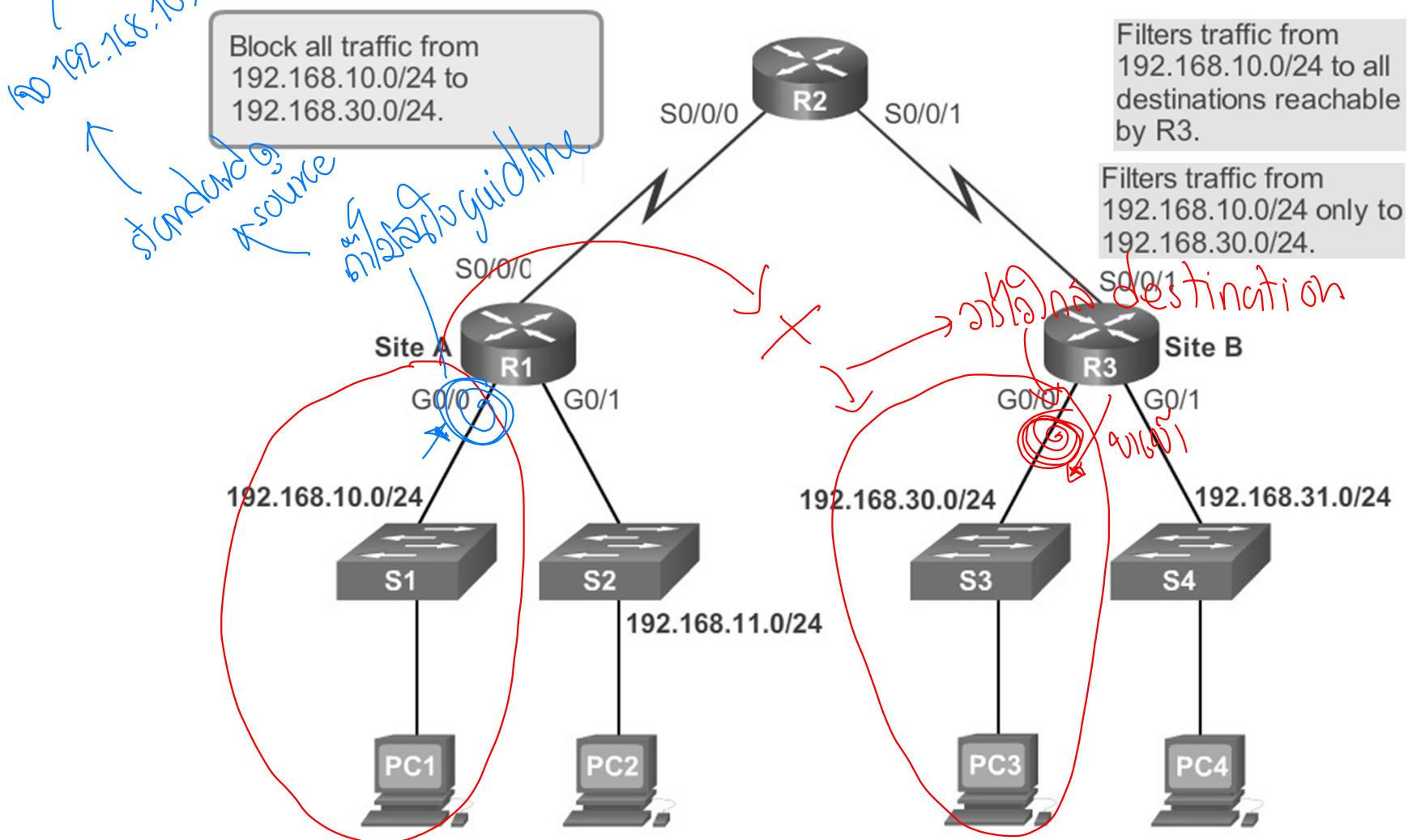
- Where to Place ACLs
  - Every ACL should be placed where it has the greatest impact on efficiency. The basic rules are:
    - Extended ACLs: Locate extended ACLs as close as possible to the source of the traffic to be filtered.
    - Standard ACLs: Because standard ACLs do not specify destination addresses, place them as close to the destination as possible.
  - Placement of the ACL and therefore the type of ACL used may also depend on: the extent of the network administrator's control, bandwidth of the networks involved, and ease of configuration.

# Guidelines for ACL creation

ນີ້ແມ່ນກຳນົດ  
ຫຼັງນັ້ນ  
ຫຼັງນັ້ນ

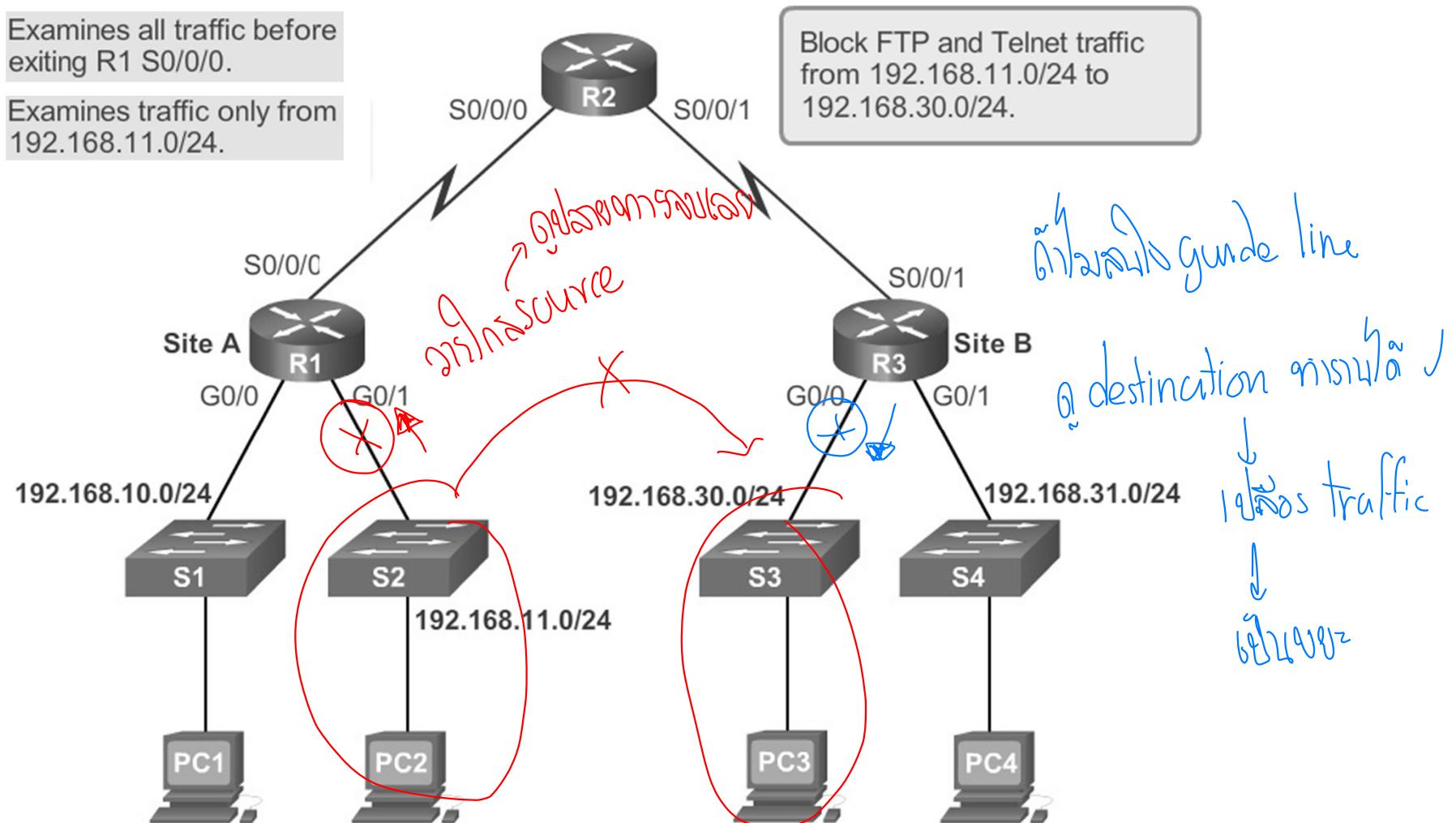
ບອກຈຳກົດສົມບໍລິຫານ

## Standard ACL Placement



# Guidelines for ACL creation

- Extended ACL Placement



# Configure ACLs

- ① • Creating ACLs Statements (Standard | Extended)
  - ↓  
ప్రయోగాను → apply to interface
- ② • Applying Standard ACLs to Interfaces

# Configure Standard IPv4 ACLs

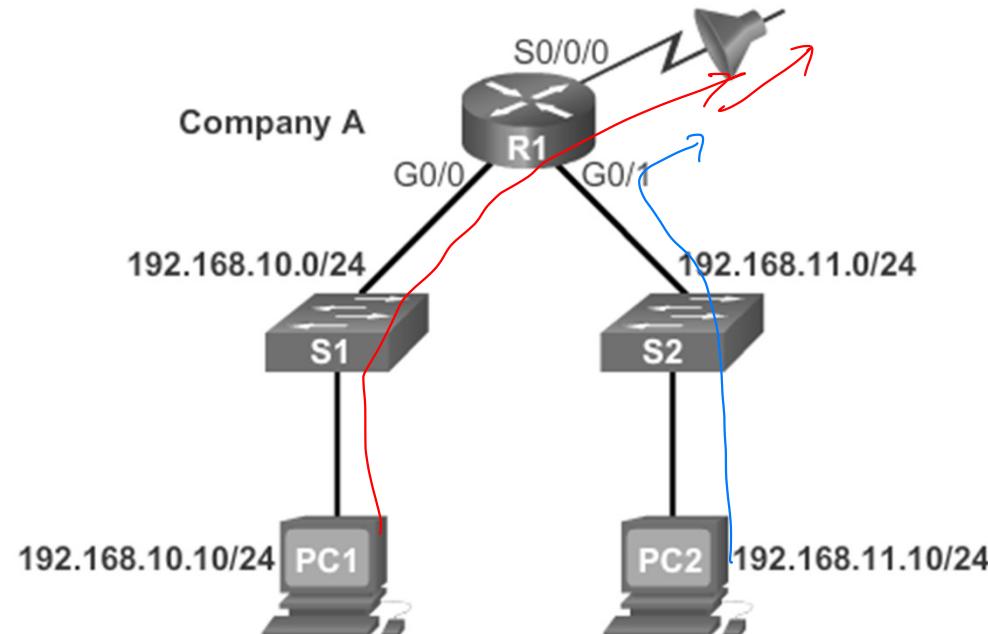
- The full syntax of the standard ACL command is as follows:

```
Router(config)# access-list access-list-number
  deny | permit | remark
  source [ source-wildcard ] [ log ]
    (1-99) standard
    logview      option        Comment
```

- To remove the ACL, the global configuration no access-list command is used. no access-list [number]
- The **remark** keyword is used for documentation and makes access lists a great deal easier to understand.

# Configure Standard IPv4 ACLs

- Entering Criteria Statements



ACL 1

```
R1 (config) #access-list 1 permit ip 192.168.10.0 0.0.0.255
```

pc1 ສ່ວນໄປໂຕ

ACL 2

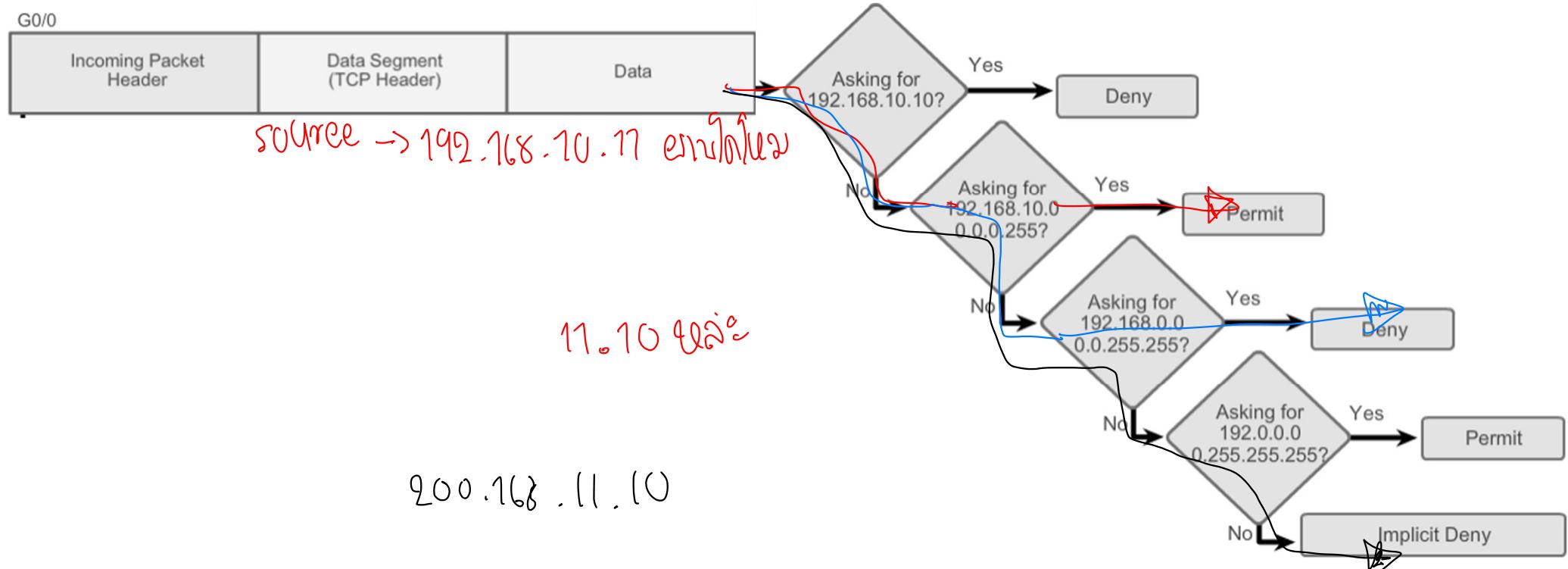
```
R1 (config) #access-list 2 permit ip 192.168.10.0 0.0.0.255
R1 (config) #access-list 2 deny any
```

pc2 ຍັງໃຫຍ່ຮັບ  
ອານຸມາດີກົດ  
ສິນເປົ້າ

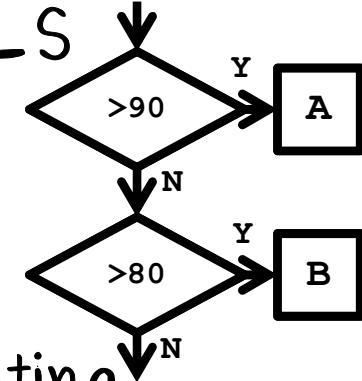
# Configure Standard IPv4 ACLs

- Example ACL

```
access-list 2 deny host 192.168.10.10 → only this
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
```



# Configure Standard IPv4 ACLs



- Internal Logic
  - Cisco IOS applies an internal logic when accepting and processing standard access list statements. As discussed previously, access list statements are processed sequentially. Therefore, the order in which statements are entered is important.

```

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)#
  
```

{ config }  
↓

ACL 3: Host statement conflicts with previous range statement.

# Configure Standard IPv4 ACLs

- Applying Standard ACLs to Interfaces
  - After a standard ACL is configured, it is linked to an interface using the **ip access-group** command in interface configuration mode:
 

*binding*

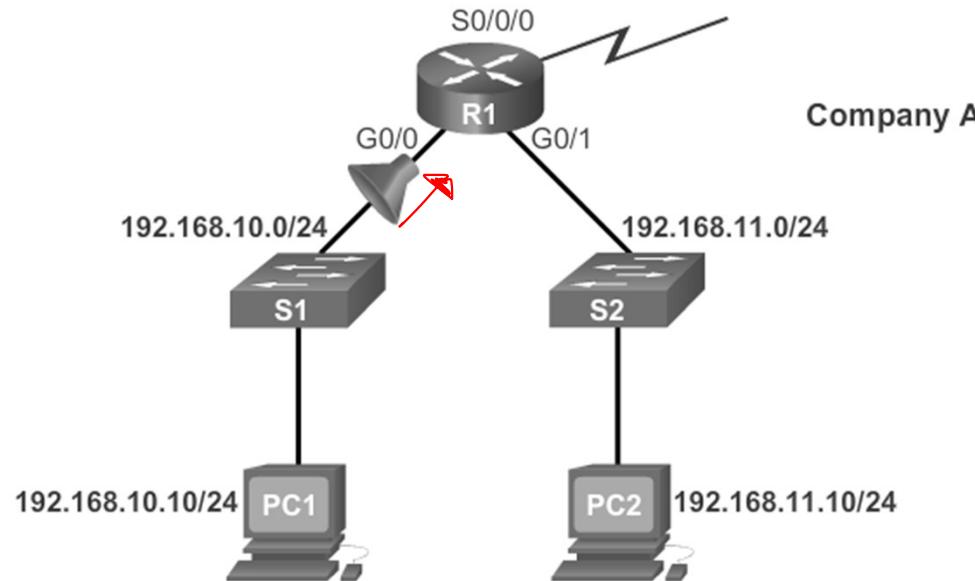


```
Router(config-if)# ip access-group
  { access-list-number | access-list-name }
  { in | out }
```
  - To remove an ACL from an interface, first enter the **no ip access-group** command on the interface, and then enter the global **no access-list** command to remove the entire ACL.

# Configure Standard IPv4 ACLs

- Applying Standard ACLs to Interfaces (Cont.)

Deny a Specific Host



```

R1(config)#no access-list 1 → გიანსებადი
R1(config)#access-list 1 deny host 192.168.10.10 } გვა
R1(config)#access-list 1 permit any
R1(config)#interface g0/0
R1(config-if)#ip access-group 1 in } implement

```

# Configure Standard IPv4 ACLs

- Creating Named Standard ACLs <sup>of 9/92</sup>

```
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 interface.

# Configure Standard IPv4 ACLs

- Commenting ACLs

Example 1: Commenting a numbered ACL

კუთხისამართი

```
R1(config)#access-list 1 remark Do not allow Guest workstation
through
R1(config)#access-list 1 deny host 192.168.10.10
R1(config)#access-list 1 remark Allow devices from all other
192.168.x.x subnets
R1(config)#access-list 1 permit 192.168.0.0 0.0.255.255
R1(config)#interface s0/0/0
R1(config-if)#ip access-group 1 out
R1(config-if)#{
```

Example 2: Commenting a named ACL

```
R1(config)#ip access-list standard NO_ACCESS
R1(config-std-nacl)#remark Do not allow access from Lab
workstation
R1(config-std-nacl)#deny host 192.168.11.10
R1(config-std-nacl)#remark Allow access from all other networks
R1(config-std-nacl)#permit any
R1(config-std-nacl)#interface G0/0
R1(config-if)#ip access-group NO_ACCESS out
R1(config-if)#{
```

# Modify IPv4 ACLs

- Editing Standard Numbered ACLs

## Editing Numbered ACLs Using a Text Editor

Configuration

```
R1(config)#access-list 1 deny host 192.168.10.99
R1(config)#access-list 1 permit 192.168.0.0 0.0.255.255
```

Step 1

```
R1#show running-config | include access-list 1
access-list 1 deny host 192.168.10.99
access-list 1 permit 192.168.0.0 0.0.255.255
```

→ copy & paste 5555

Step 2

```
<Text editor>
access-list 1 deny host 192.168.10.10
access-list 1 permit 192.168.0.0 0.0.255.255
```

Step 3

```
R1#config t
Enter configuration commands, one per line. End with
CNTL/Z.
R1(config)#no access-list 1
R1(config)#access-list 1 deny host 192.168.10.10
R1(config)#access-list 1 permit 192.168.0.0 0.0.255.255
```

Step 4

```
R1#show running-config | include access-list 1
access-list 1 deny host 192.168.10.10
access-list 1 permit 192.168.0.0 0.0.255.255
```

# Modify IPv4 ACLs

- Editing Standard Numbered ACLs (Cont.)

## Editing Numbered ACLs Using Sequence Numbers

Configuration

```
R1(config)#access-list 1 deny host 192.168.10.99
R1(config)#access-list 1 permit 192.168.0.0 0.0.255.255
```

Step 1

```
R1#show access-lists 1
Standard IP access list 1
 10 deny    192.168.10.99
 20 permit 192.168.0.0, wildcard bits 0.0.255.255
R1#
```

↳ *more detail*

Step 2

```
R1#conf t
R1(config)#ip access-list standard 1
R1(config-std-nacl)#no 10
R1(config-std-nacl)#10 deny host 192.168.10.10
R1(config-std-nacl)#end
R1#
```

Step 3

```
R1#show access-lists
Standard IP access list 1
 10 deny    192.168.10.10
 20 permit 192.168.0.0, wildcard bits 0.0.255.255
R1#
```

# Modify IPv4 ACLs

- Editing Standard Named ACLs

Adding a Line to a Named ACL

↳ detail

```
R1#show access-lists
Standard IP access list NO_ACCESS
    10 deny   192.168.11.10
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#conf t
Enter configuration commands, one per line. End with
CNTL/Z.
R1(config)#ip access-list standard NO_ACCESS
R1(config-std-nacl)#15 deny host 192.168.11.11
R1(config-std-nacl)#end
R1#show access-lists
Standard IP access list NO_ACCESS
    10 deny   192.168.11.10
    15 deny   192.168.11.11
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

**Note:** The **no sequence-number** named-ACL command is used to delete individual statements.

# Modify IPv4 ACLs

- Verifying ACLs

ANSWER

```
R1# show ip interface s0/0/0
Serial0/0/0 is up, line protocol is up
  Internet address is 10.1.1.1/30
<output omitted>
  Outgoing access list is 1
  Inbound access list is not set
<output omitted>
```

```
R1# show ip interface g0/0
GigabitEthernet0/0 is up, line protocol is up
  Internet address is 192.168.10.1/24
<output omitted>
  Outgoing access list is NO ACCESS
```

access list now name

```
R1# show access-lists
Standard IP access list 1
  10 deny  192.168.10.10
  20 permit 192.168.0.0, wildcard bits 0.0.255.255
Standard IP access list NO_ACCESS
  15 deny  192.168.11.11
  10 deny  192.168.11.10
  20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

# Modify IPv4 ACLs

- ACL Statistics

```
R1#show access-lists
Standard IP access list 1
    10 deny   192.168.10.10 (4 match(es))
    20 permit 192.168.0.0, wildcard bits 0.0.255.255
Standard IP access list NO ACCESS
    15 deny   192.168.11.11
    10 deny   192.168.11.10 (4 match(es))
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

in log misiv  
deny any is

Output after pinging PC3 from PC1.

Matches have been incremented.

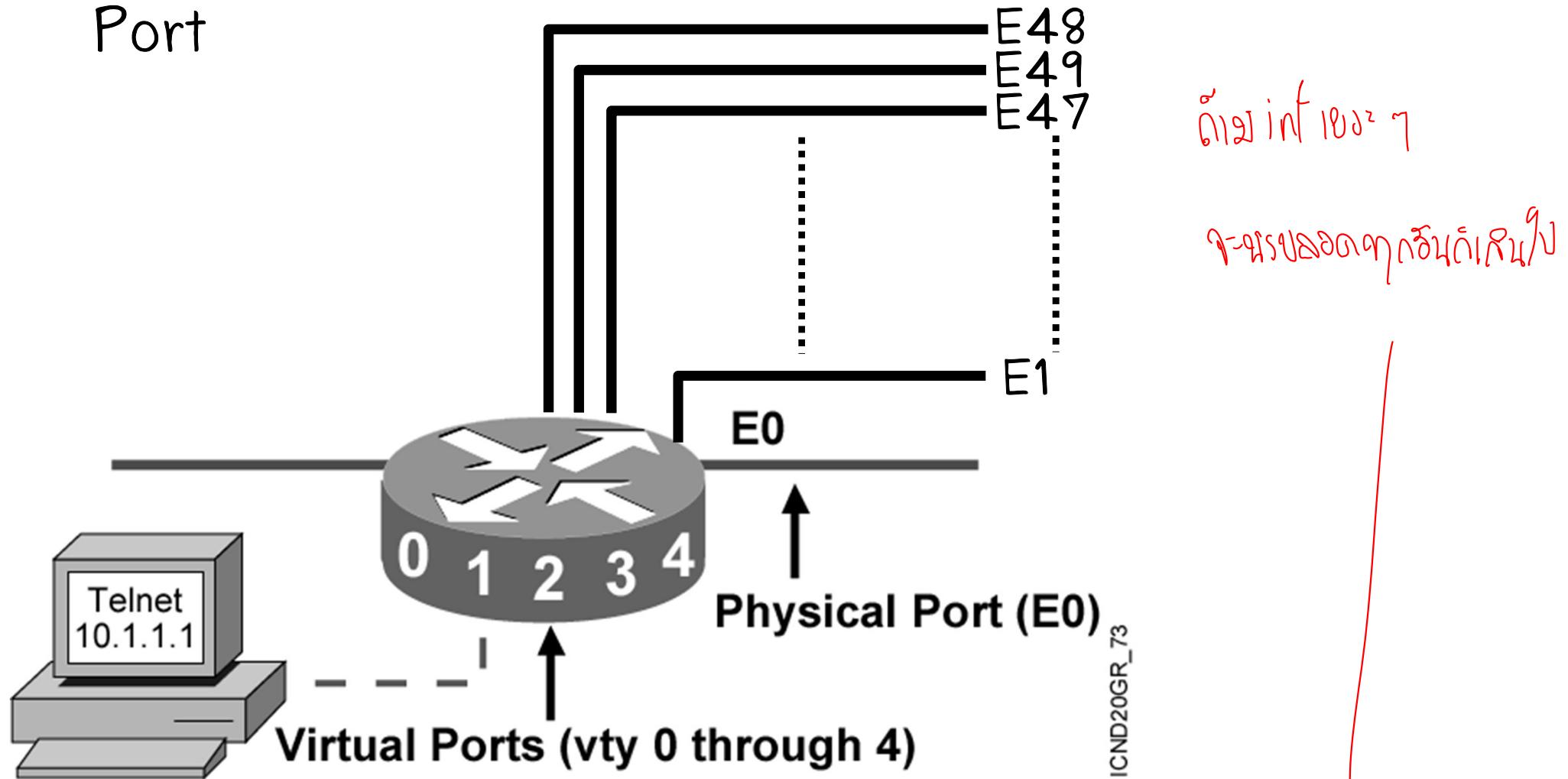
```
R1#show access-lists
Standard IP access list 1
    10 deny   192.168.10.10 (8 match(es))
    20 permit 192.168.0.0, wildcard bits 0.0.255.255
Standard IP access list NO ACCESS
    15 deny   192.168.11.11
    10 deny   192.168.11.10 (4 match(es))
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

# Modify IPv4 ACLs

- Standard ACL Sequence Numbers
  - Another part of the IOS internal logic involves the internal sequencing of standard ACL statements. Range statements that deny three networks are configured first followed by five host statements. The host statements are all valid statements because their host IP addresses are not part of the previously entered range statements.
  - The host statements are listed first by the show command, but not necessarily in the order that they were entered. The IOS puts host statements in an order using a special hashing function. The resulting order optimizes the search for a host ACL entry.

# Securing VTY ports with a Standard IPv4 ACL

- Configuring a Standard ACL to Secure a VTY Port



# Securing VTY ports with a Standard IPv4 ACL

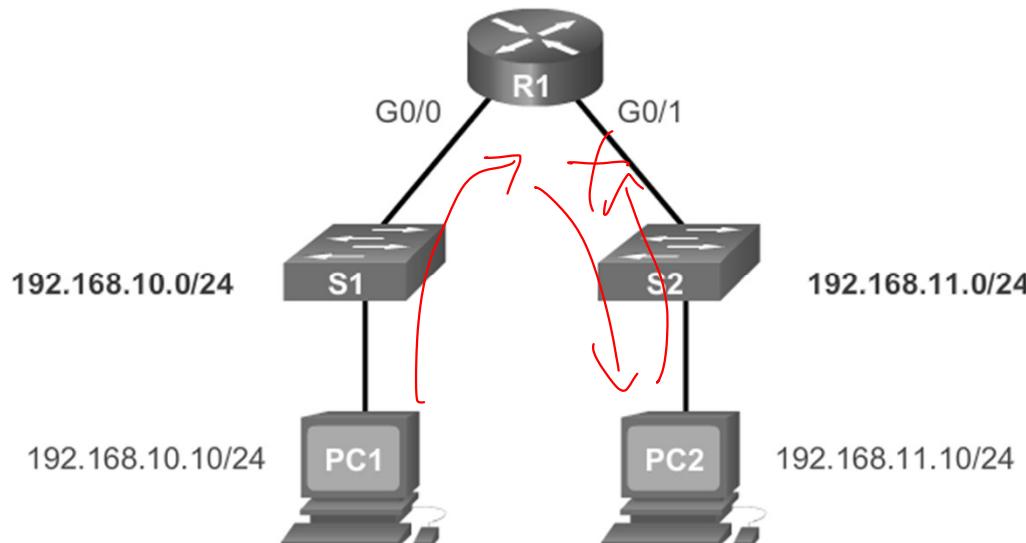
- Configuring a Standard ACL to Secure a VTY Port
  - Filtering Telnet or SSH traffic is typically considered an extended IP ACL function because it filters a higher level protocol. However, because the access-class command is used to filter incoming or outgoing Telnet/SSH sessions by source address, a standard ACL can be used.

```
Router(config-line)# access-class access-list-number { in [ vrf-also ] | out }
```

# Securing VTY ports with a Standard IPv4 ACL

- Verifying a Standard ACL used to Secure a VTY Port

```
R1#show access-lists
Standard IP access list 21
  10 permit 192.168.10.0, wildcard bits 0.0.0.255 (2 matches)
  20 deny   any (1 match)
R1#
```

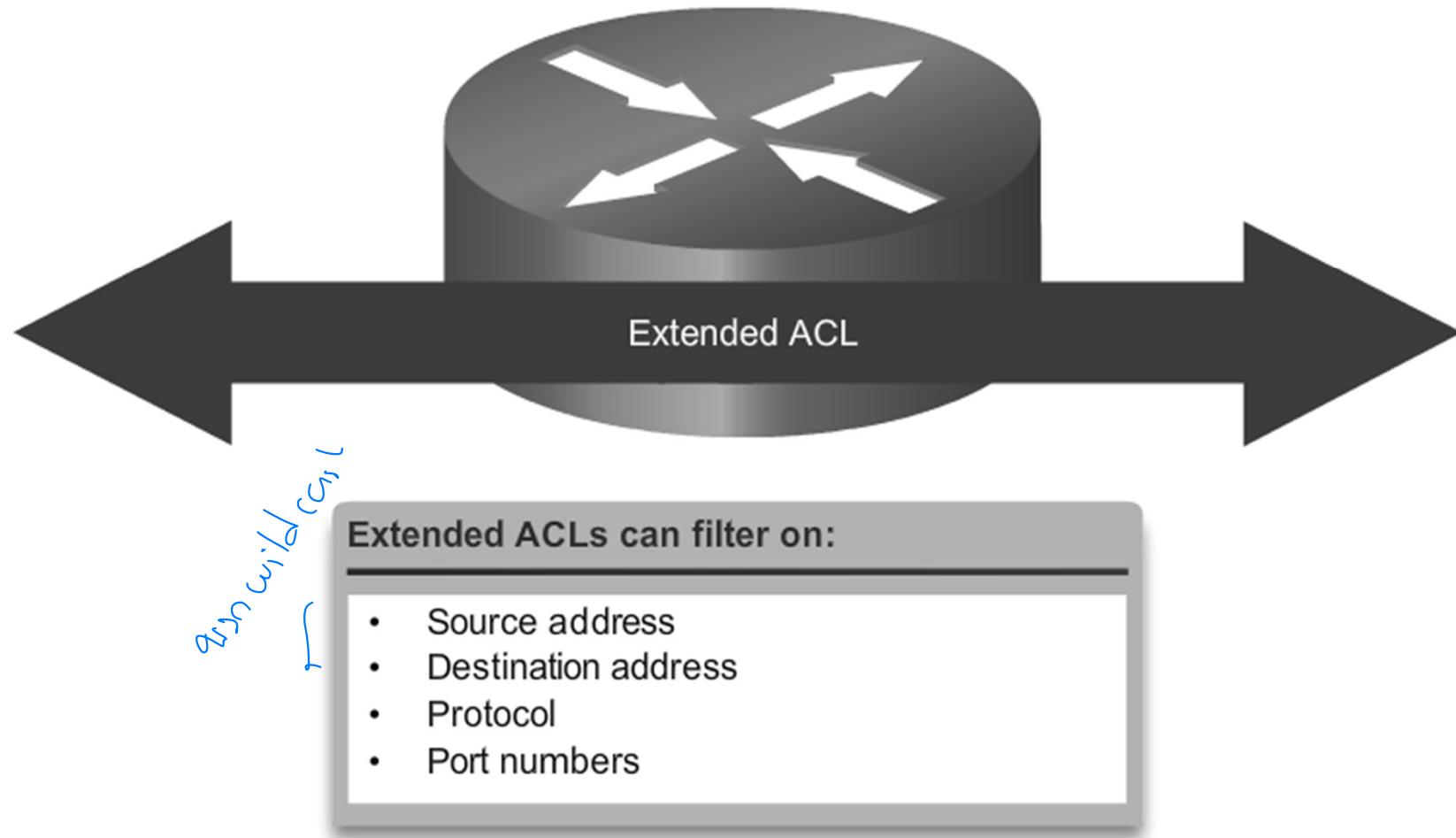


```
PC1>ssh 192.168.10.1
Login as: admin
Password: *****
R1>
```

```
PC2>ssh 192.168.11.1
ssh connect to host 192.168.11.1 port
22: Connection refused
PC2>
```

# Structure of an Extended IPv4 ACL

- Extended ACLs



# Structure of an Extended IPv4 ACL

- Extended ACLs (Cont.)

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
```

# Configure Extended IPv4 ACLs

- The procedural steps for configuring extended ACLs are the same as for standard ACLs. The extended ACL is first configured, and then it is activated on an interface. However, the command syntax and parameters are more complex to support the additional features provided by extended ACLs.

1/21/21

```
access-list access-list-number {deny | permit | remark}  

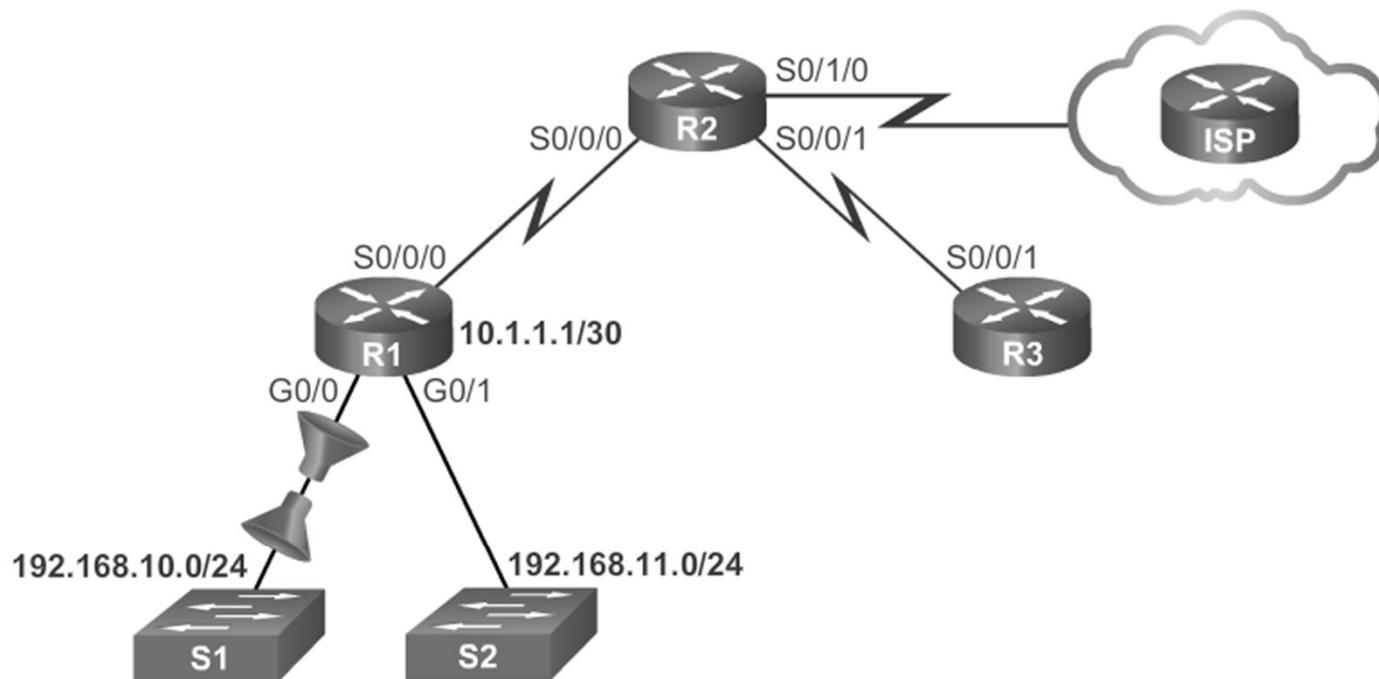
protocol source [source-wildcard] [operator operand]  

[port port-number or name] destination [destination-wildcard]  

[operator operand] [port port-number or name] [established]
```

# Configure Extended IPv4 ACLs

- Applying Extended ACLs to Interfaces



```

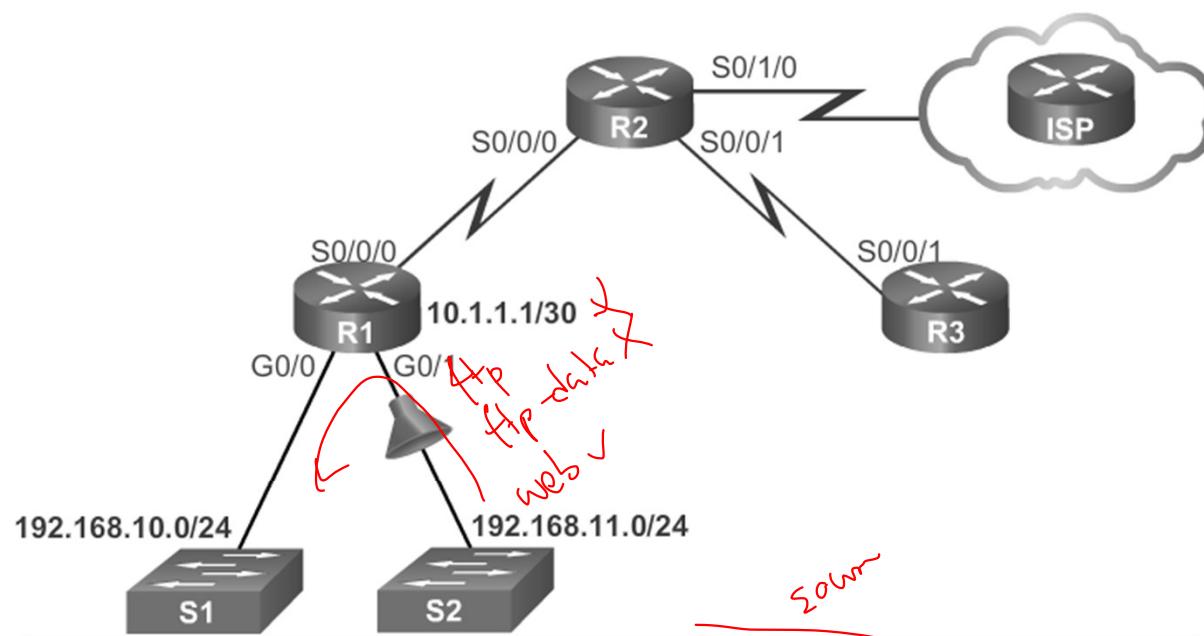
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config)#access-list 104 permit tcp any 192.168.10.0 0.0.0.255 established
R1(config)#interface g0/0
R1(config-if)#ip access-group 103 in
R1(config-if)#ip access-group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

```

# Configure Extended IPv4 ACLs

- Filtering Traffic with Extended ACLs

Extended ACL to Deny 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
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

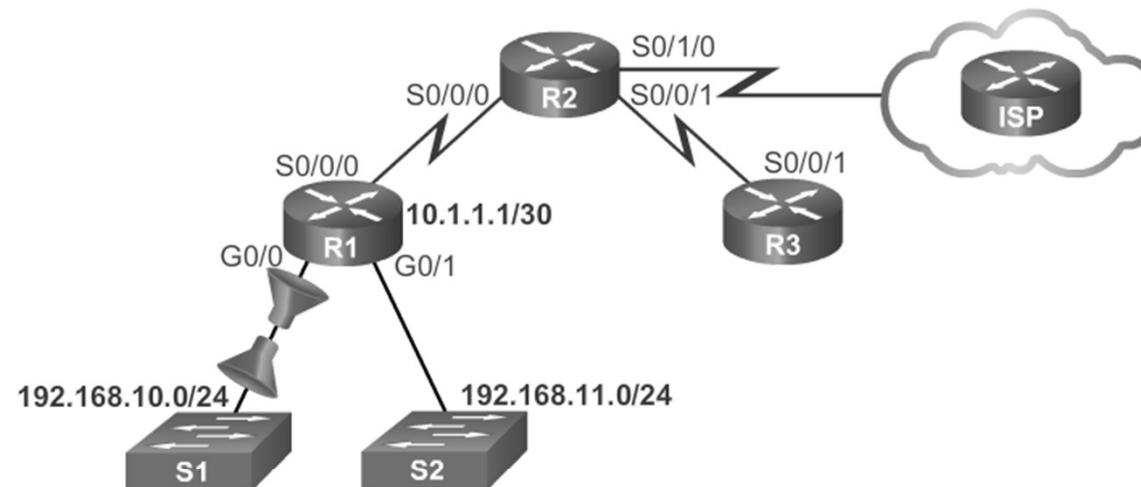
```

# Configure Extended IPv4 ACLs

- Creating ~~Named~~ Extended ACLs

JM

Creating Named Extended ACLs



```
R1(config)#ip access-list extended SURFING
R1(config-ext-nacl)#permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config-ext-nacl)#permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config-ext-nacl)#exit
R1(config)#ip access-list extended BROWSING
R1(config-ext-nacl)#permit tcp any 192.168.10.0 0.0.0.255 established
R1(config-ext-nacl)#exit
R1(config)#interface g0/0
R1(config-if)#ip access-group SURFING in
R1(config-if)#ip access-group BROWSING out
```

# Configure Extended IPv4 ACLs

- Verifying Extended ACLs



```
R1#show access-lists
Extended IP access list BROWSING
    10 permit tcp any 192.168.10.0 0.0.0.255 established
Extended IP access list SURFING
    10 permit tcp 192.168.10.0 0.0.0.255 any eq www
    20 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1#
R1#show ip interface g0/0
GigabitEthernet0/0 is up, line protocol is up
    Internet address is 192.168.10.1/24
<output omitted for brevity>
    Outgoing access list is BROWSING
    Inbound access list is SURFING
<output omitted for brevity>
```

# Configure Extended IPv4 ACLs

- Editing Extended ACLs
  - Editing an extended ACL can be accomplished using the same process as editing a standard. An extended ACL can be modified using:
    - Method 1 - **Text editor** ✓
    - Method 2 - Sequence numbers → *Method 2*

# Limiting Debug Output

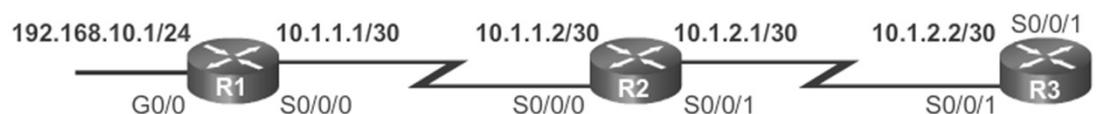
6/2

- Purpose of Limiting debug Output with ACLs
  - Debug commands are tools used to help verify and troubleshoot network operations.
  - When using some debug options, the output may display much more information than is needed or can be easily viewed.
  - In a production network, the amount of information provided by debug commands can be overwhelming and can cause network interruptions.
  - Some debug commands can be combined with an access list to limit output so that only the information needed for verification or troubleshooting a specific issue is displayed.

# Limiting Debug Output

Q10

- Configuring ACLs to Limit debug Output
  - The administrator for R2 wants to verify that traffic is being routed correctly using debug ip packet. To limit the debug output to include only the ICMP traffic between R1 and R3, ACL 101 will be applied.



```

R2(config)#ip access-list extended 101
R2(config-ext-nacl)#permit icmp host 10.1.1.1 host 10.1.2.2
R2(config-ext-nacl)#permit icmp host 10.1.2.2 host 10.1.1.1
R2(config-ext-nacl)#exit
R2(config)#interface s0/0/0
R2(config-if)#no ip route-cache
R2(config-if)#exit
R2(config)#interface s0/0/1
R2(config-if)#no ip route-cache
R2(config-if)#end
R2#
R2#debug ip packet 101
IP packet debugging is on for access list 101
R2#
  
```

# Limiting Debug Output

- Verifying ACLs that Limit debug Output



2/2

```

R2# debug ip packet 101
IP packet debugging is on for access list 101
R2#
<ping 10.1.2.2 command entered on R1>

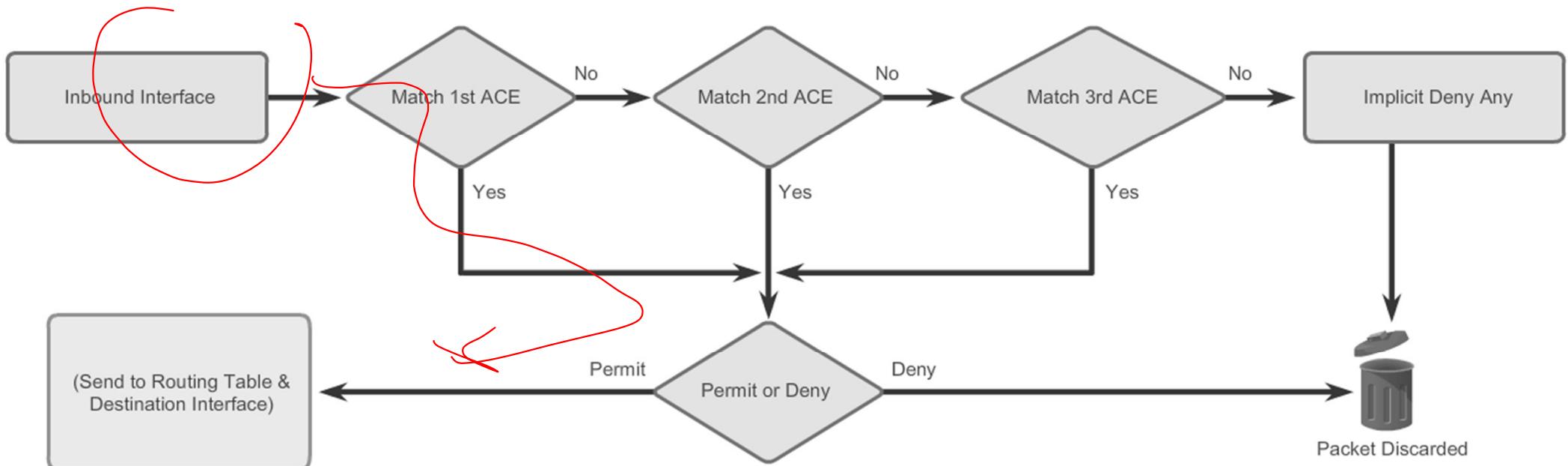
*Jan 25 20:49:26.910: IP: s=10.1.1.1 (Serial0/0/0), d=10.1.2.2
(Serial0/0/1), g=10.1.2.2, len 100, forward
*Jan 25 20:49:26.910: IP: s=10.1.1.1 (Serial0/0/0), d=10.1.2.2
(Serial0/0/1), len 100, sending full packet
*Jan 25 20:49:26.938: IP: s=10.1.2.2 (Serial0/0/1), d=10.1.1.1
(Serial0/0/0), g=10.1.1.1, len 100, forward
*Jan 25 20:49:26.938: IP: s=10.1.2.2 (Serial0/0/1), d=10.1.1.1
(Serial0/0/0), len 100, sending full packet

<output omitted>
  
```

# Processing Packets with ACLs

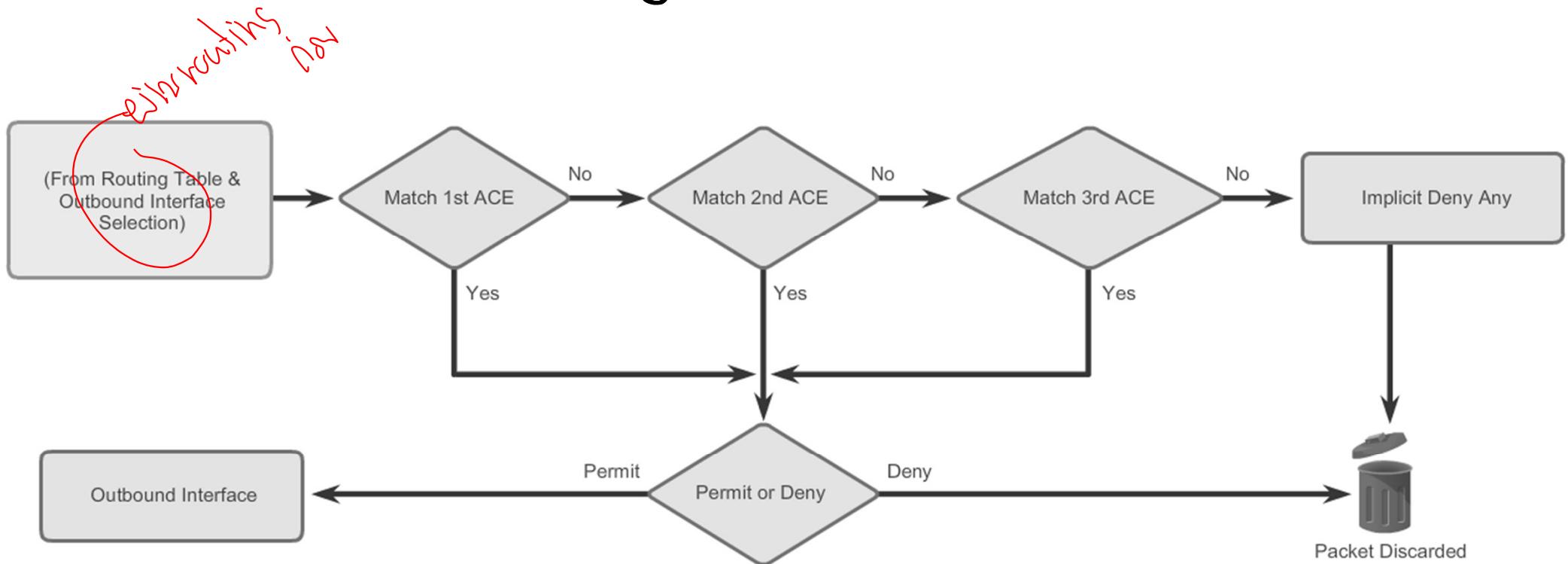
- Inbound ACL Logic

notes

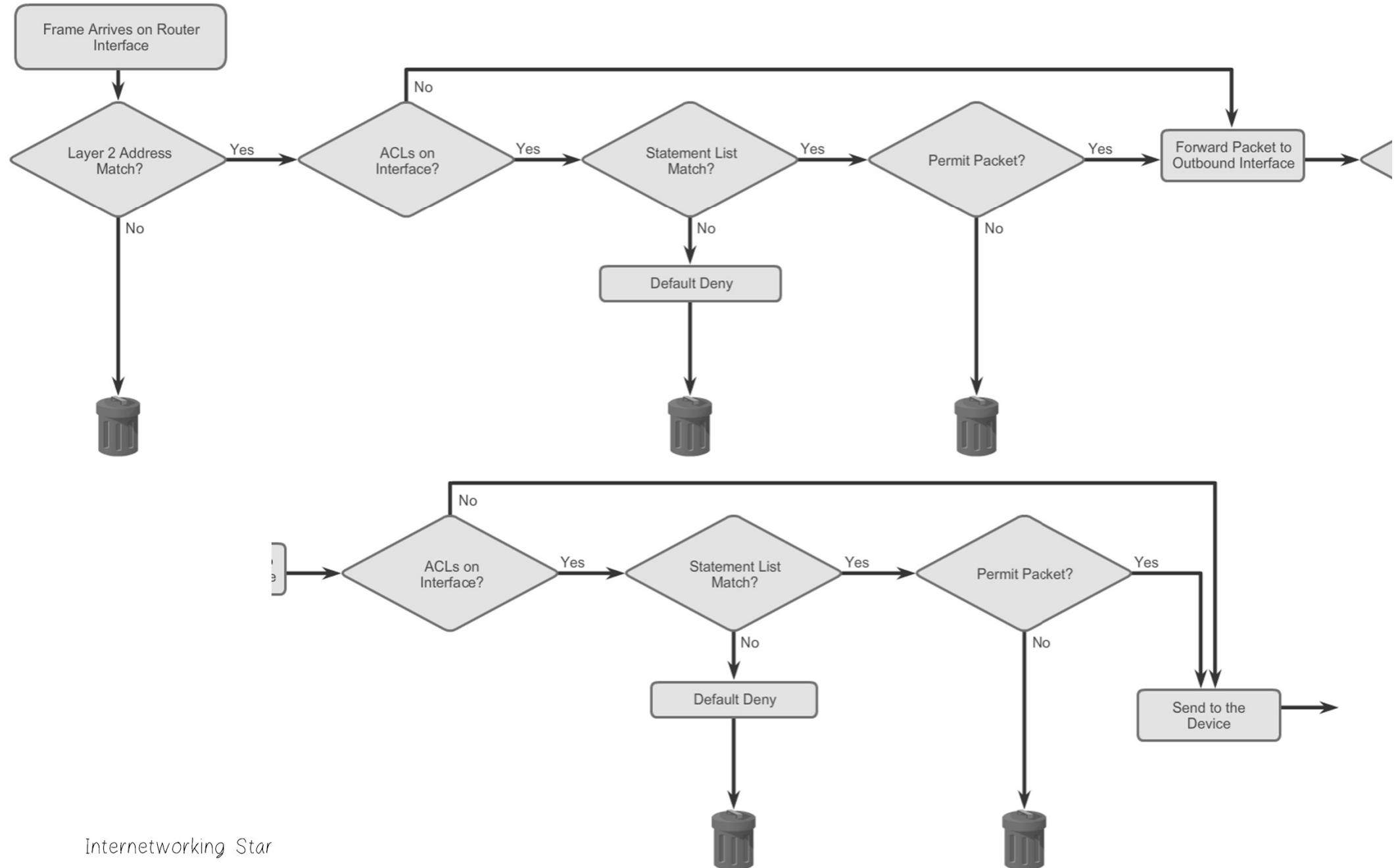


# Processing Packets with ACLs

- Outbound ACL Logic

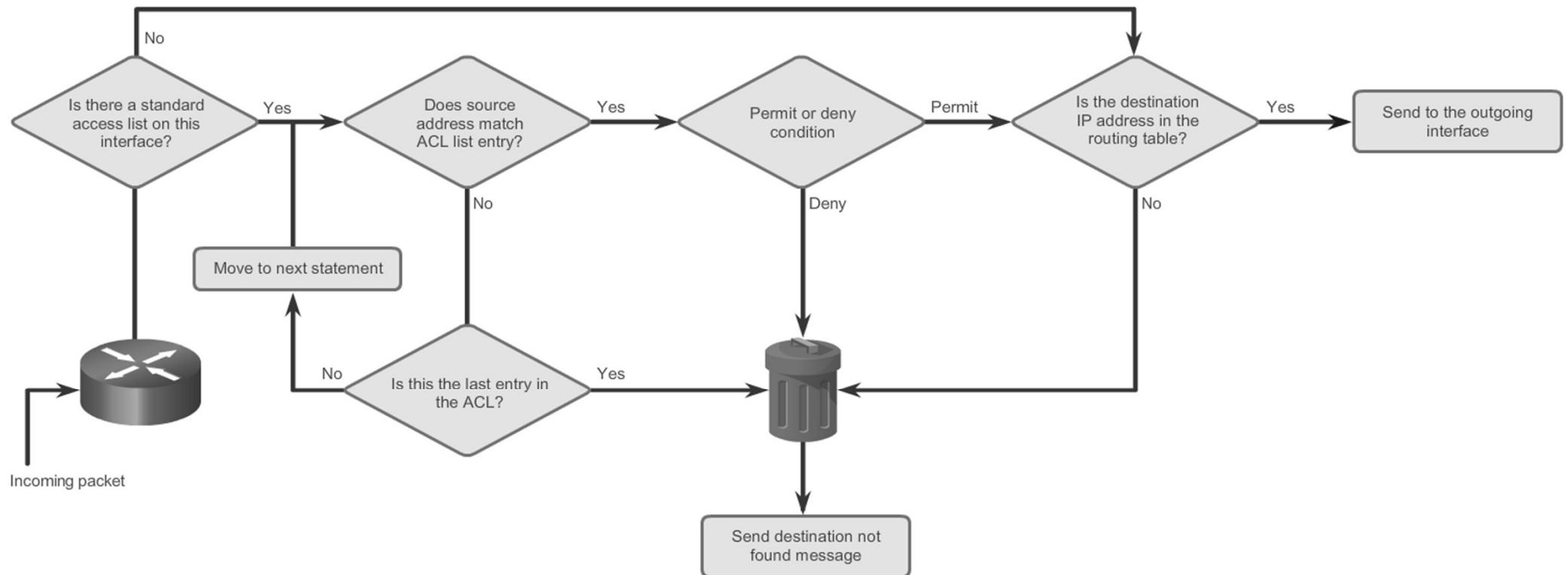


# Processing Packets with ACLs



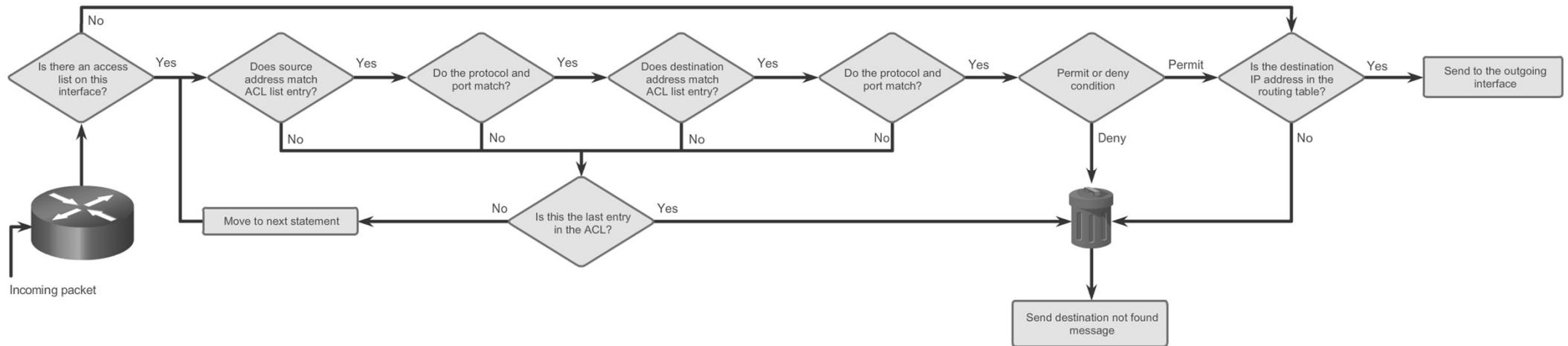
# Processing Packets with ACLs

- Standard ACL Decision Process



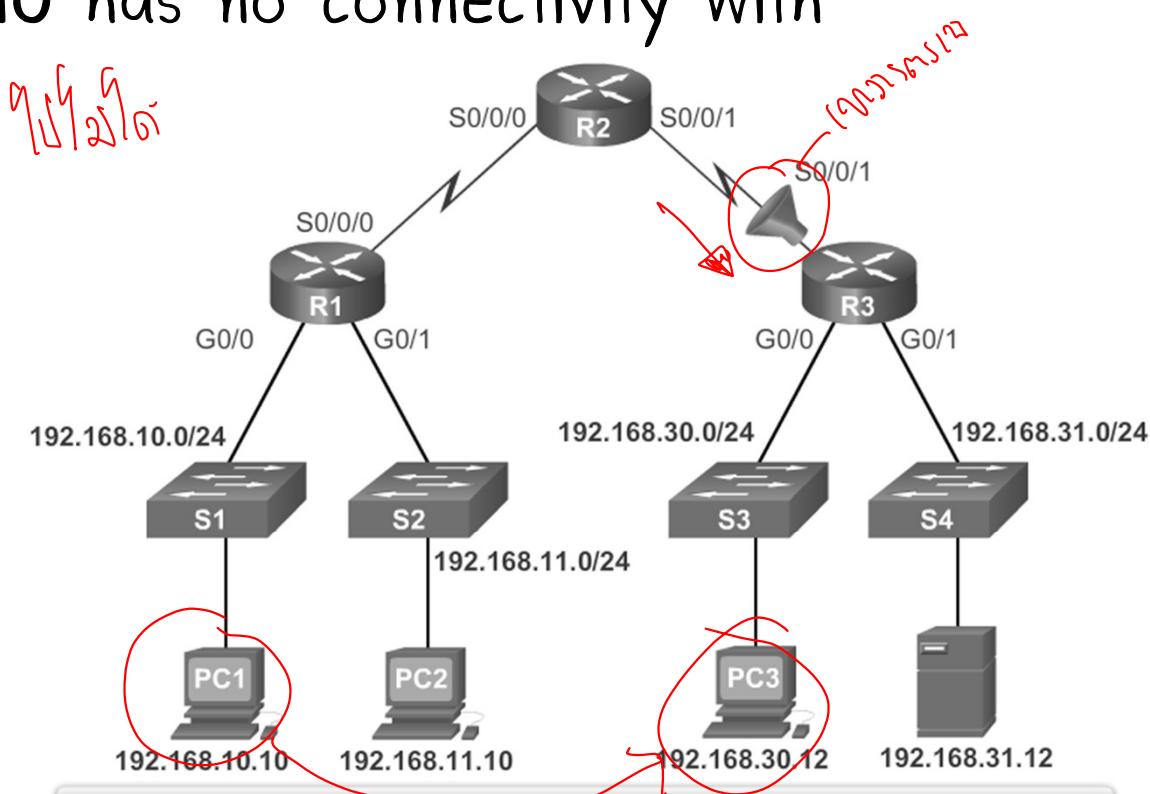
# Processing Packets with ACLs

- Extended ACL Decision Process



# Common ACLs Errors

- Troubleshooting Common ACL Errors - Example 1
  - Host 192.168.10.10 has no connectivity with 192.168.30.12.

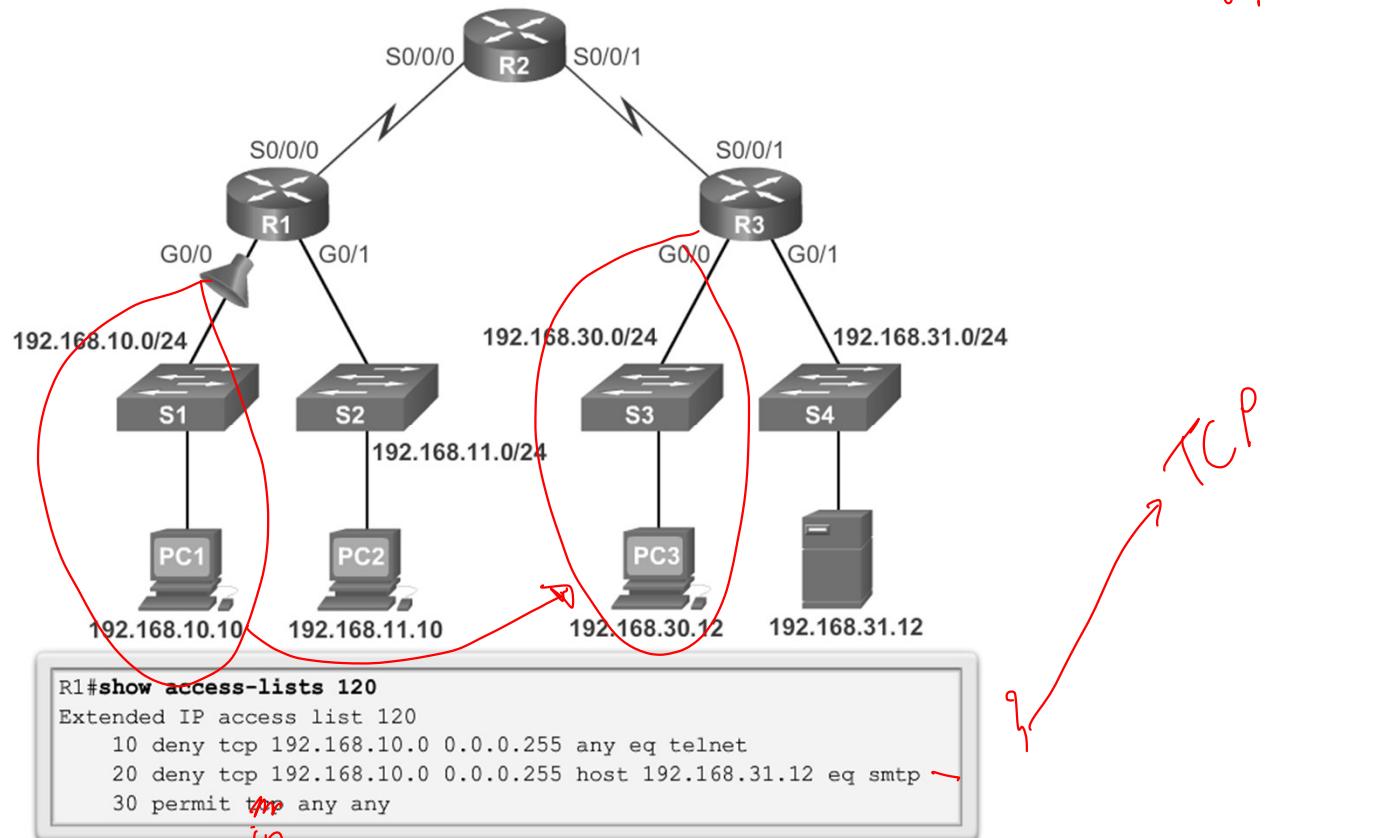


```
R3#show access-lists
Extended IP access list 110
 10 deny tcp 192.168.10.0 0.0.0.255 any (12 match(es))
 20 permit tcp 192.168.10.0 0.0.0.255 any eq telnet
 30 permit ip any any
```

กูรู้แล้ว

# Common ACLs Errors

- Troubleshooting Common ACL Errors - Example 2
  - The 192.168.10.0 /24 network cannot use **TFTP** to connect to the 192.168.30.0 /24 network.



# Common ACLs Errors

- Troubleshooting Common ACL Errors - Example 3
  - The 192.168.11.0 /24 network can use Telnet to connect to 192.168.30.0 /24, but according to company policy, this connection should not be allowed.

**R1#show access-lists 130**

```

Extended IP access list 130
 10 deny tcp any eq telnet any
 20 deny tcp 192.168.11.0 0.0.0.255 host 192.168.31.12 eq smtp
 30 permit tcp any any (12 match(es))
  
```

Annotations on the CLI output:

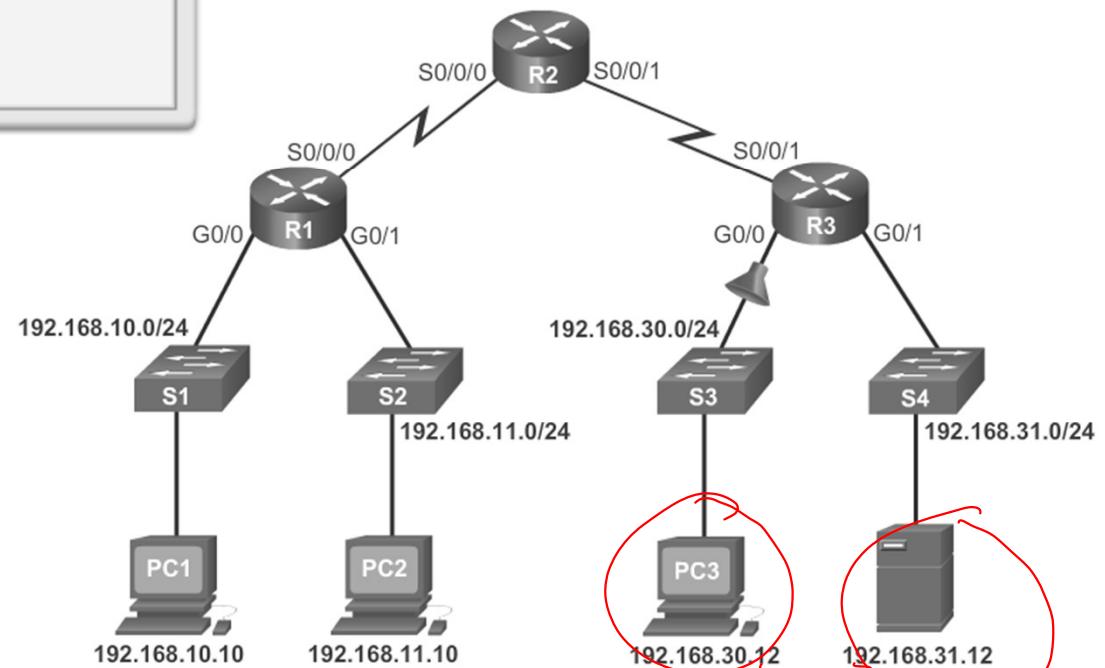
- Red circle around "any" in "deny tcp any eq telnet any": *source*
- Red circle around "eq telnet": *top any any eq telnet*
- Red circle around "host 192.168.31.12": *destination 00*
- Red circle around "any" in "permit tcp any any": *port any source 192.168.11.0/24*

# Common ACLs Errors

- Troubleshooting Common ACL Errors - Example 4
  - Host 192.168.30.12 is able to Telnet to connect to 192.168.31.12, but company policy states that this connection should not be allowed.

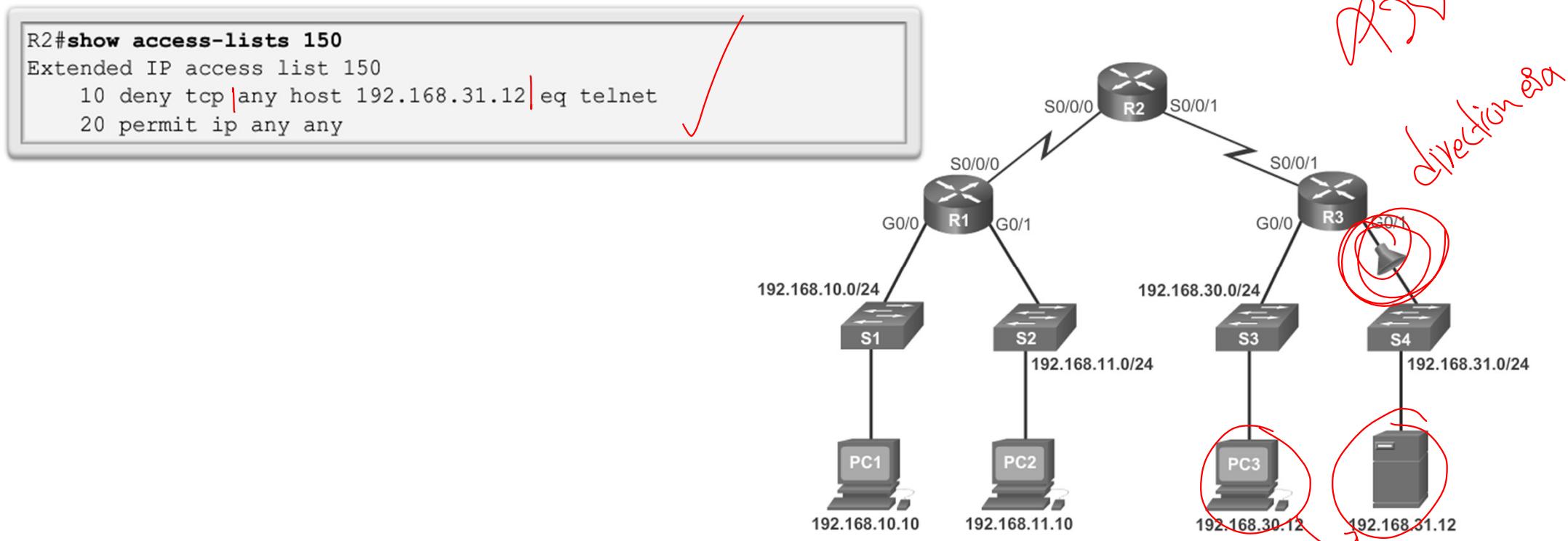
```
R3#show access-lists 140
Extended IP access list 140
 10 deny tcp host 192.168.30.11 any eq telnet
 20 permit ip any any (5 match(es))
```

^112



# Common ACLs Errors

- Troubleshooting Common ACL Errors - Example 5
  - 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.



# Questions and Answers

