Université de Rennes

istic Informatique Électronique

Network Security

ACLs: Our list of Allies grows thin

Gwendal Patat
Univ Rennes, CNRS, IRISA
2025/2026

On the road so far...

During our layer inspections from L2 to above, we have seen a lot of possible attacks.

- Smurf attacks.
- TCP hijacking.
- UDP Flooding attacks.
- etc.

Existing mitigations for them often include **firewalls** to avoid, for instance, DoS attacks or IP spoofing.

Firewalls

- **Router-based firewall:**
  - Implements Access Control Lists (ACLs)
    - **Today's topic!**
  - Filters traffic at network layer (L3).
  - Used to control inter-network flows (e.g., LAN ↔ Internet).
- **Host-based firewall:**
  - Runs on the machine itself.
  - Filters incoming and outgoing traffic per host.
  - Can be stateful.
  - Often complements router ACLs for defense in depth.
- **Dedicated Firewalls:**
  - Specialized device on the network whose sole purpose is security.

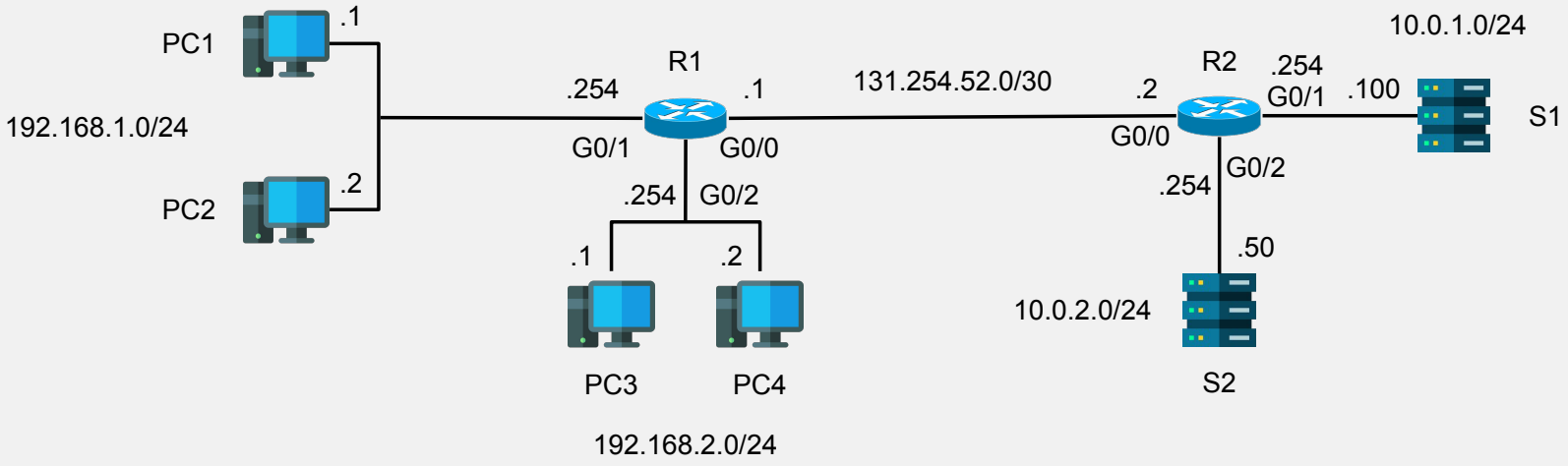
Access Control List (ACL)

Access Control List (ACL)

ACLs (Access Control Lists) function as a packet filter on network devices such as routers to permit or discard specific traffic.

ACLs can apply this filter based on source and destination IP addresses, source/destination ports, etc.

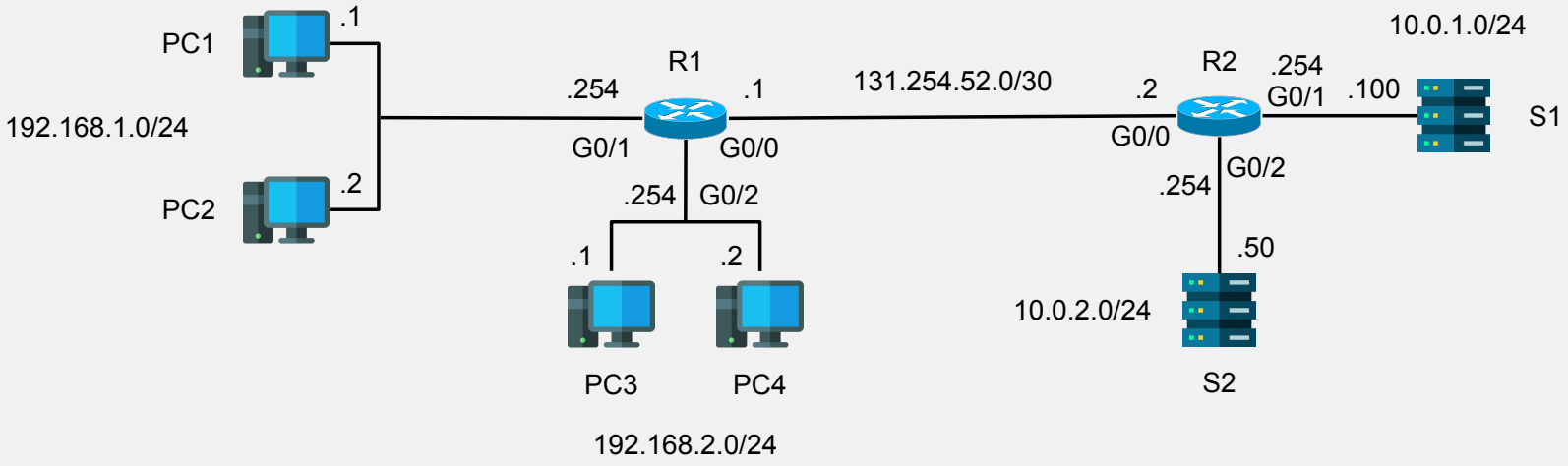
Access Control List (ACL)



Access Control List (ACL)

Network needs:

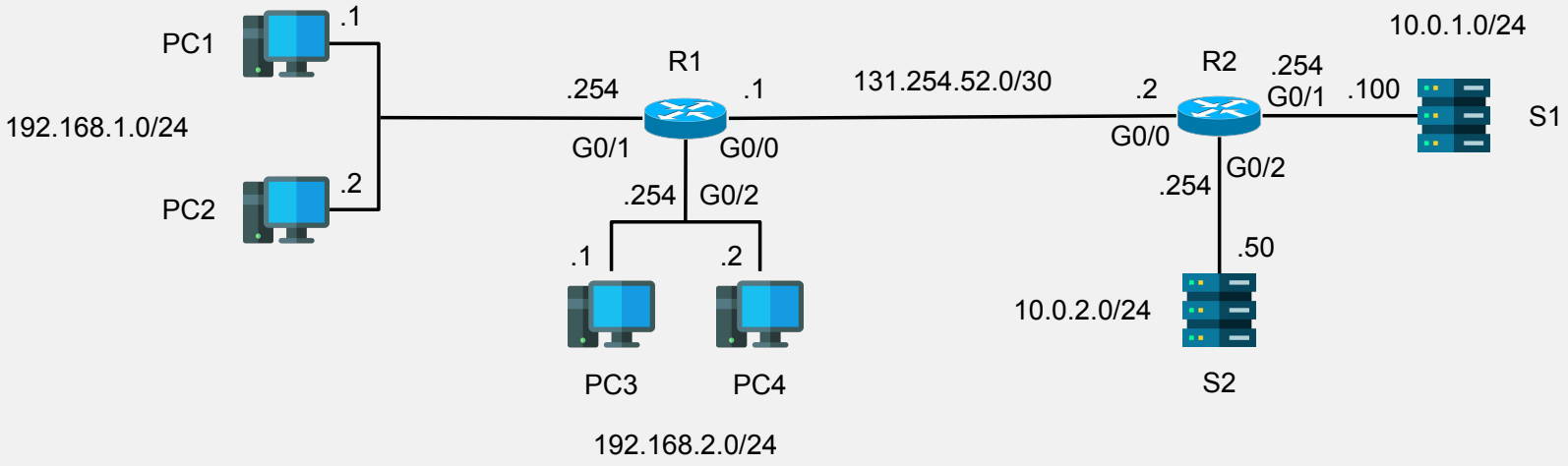
- Hosts in 192.168.1.0/24 can access the 10.0.1.0/24 network.
- Hosts in 192.168.2.0/24 cannot access the 10.0.1.0/24 network.



Access Control List (ACL)

ACLs:

- ACLs are configured **globally** on a router.
- Ordered sequence of **ACEs (Access Control Entries).**

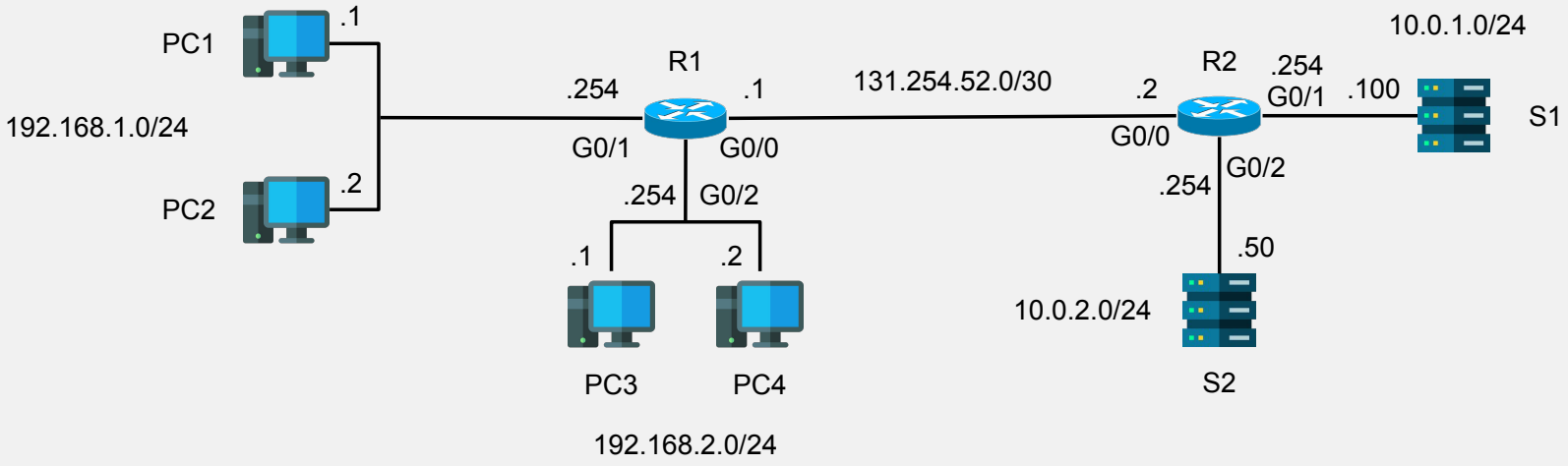


Access Control List (ACL)

ACLs:

- ACLs are configured **globally** on a router.
- Ordered sequence of **ACEs (Access Control Entries).**

ACL 1:
1. If src IP == 192.168.1.0/24: permit
2. If src IP == 192.168.2.0/24: deny
3. If src IP == any: permit

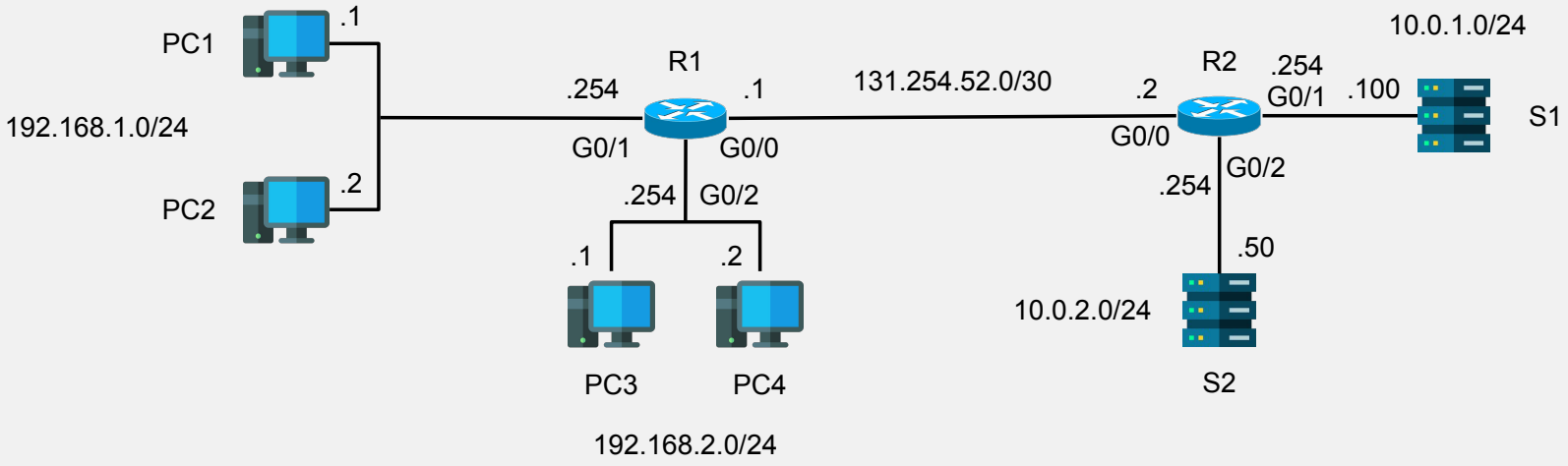


Access Control List (ACL)

ACLs are configured globally but need to be applied to interfaces:

- **Inbound**: Applied to packets coming to the interface.
- **Outbound**: Applied to packets leaving the interface.

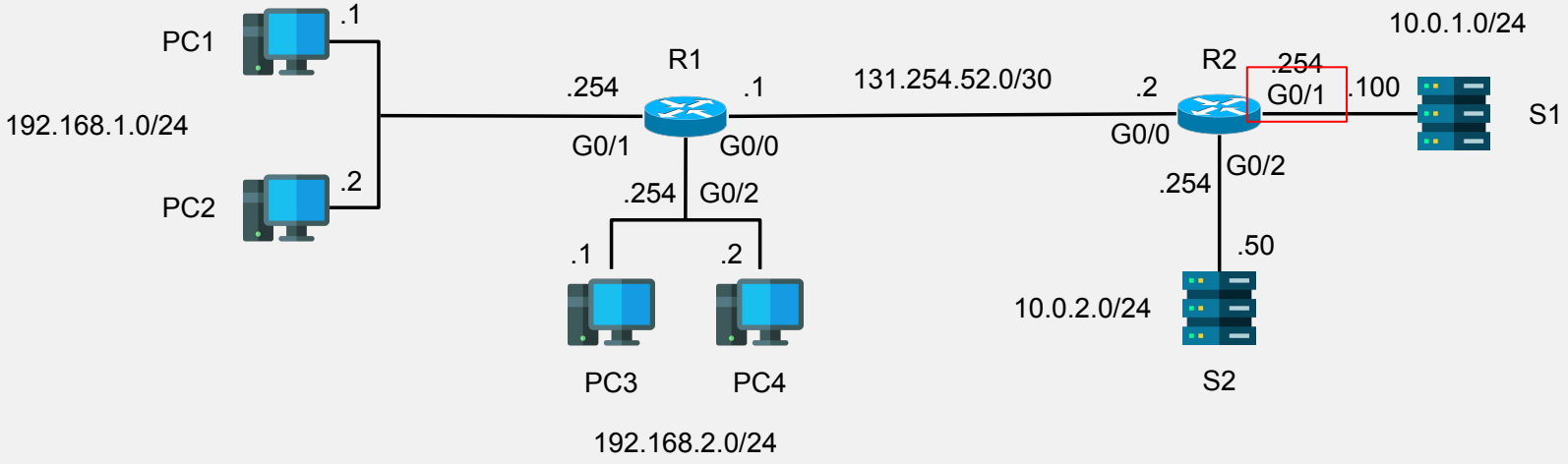
ACL 1:
1. If src IP == 192.168.1.0/24: permit
2. If src IP == 192.168.2.0/24: deny
3. If src IP == any: permit



Access Control List (ACL)

- For this example, ACL 1 outbound on R2's G0/1 is the best match.
  - Inter LAN works.
  - Access to S2 works.
  - Access to S1 is restricted.

ACL 1:
1. If src IP == 192.168.1.0/24: permit
2. If src IP == 192.168.2.0/24: deny
3. If src IP == any: permit



Access Control List (ACL)

- ACLs are defined per router and applied to interfaces.
- ACEs compose ACLs, and are evaluated in an ordered fashion.
  - Top to bottom.
  - Stop on first match, all the rest is ignored.
- They can be applied as inbound or outbound.
  - Maximum of 1 ACL applied **on an interface per direction**.

ACL Implicit Deny

- If a packet do not match an ACE in any ACL, **it will be deny**.
  - **Implicit**: if src IP == any: deny.

ACL Types

There are two types of ACLs:

- **Standard ACLs**: (ACE on source IP only)
  - Standard Numbered ACLs
  - Standard Named ACLs
- **Extended ACLs**: (ACE on src/dst IP, L4 protocol, and src/dest ports)
  - Extended Numbered ACLs
  - Extended Named ACLs

Standard ACLs

Standard Numbered ACLs

- Standard ACLs match the source IP address.
- Numbered ACLs are configured and identified by number (ACL 1, ACL 2, etc…).
  - For standard ACLs, these range from 1 to 99 and 1300 to 1999.
- In this lecture, we talk about IP ACLs, but others exist to filter packets based on ethernet types, addresses, route, etc.

```
R1(config)# access-list number {deny | permit} ip wildcard_mask
```

Wildcard mask: Inverted mask.

Standard Numbered ACLs

- Standard ACLs match the source IP address.
- Numbered ACLs are configured and identified by number (ACL 1, ACL 2, etc…).
  - For standard ACLs, these range from 1 to 99 and 1300 to 1999.
- In this lecture, we talk about IP ACLs, but others exist to filter packets based on ethernet types, addresses, route, etc.

```
R1(config)# access-list 1 permit 192.168.1.0 0.0.0.255
R1(config)# access-list 1 deny 192.168.2.0 0.0.0.255
```

Standard Numbered ACLs

- Standard ACLs match the source IP address.
- Numbered ACLs are configured and identified by number (ACL 1, ACL 2, etc…).
  - For standard ACLs, these range from 1 to 99 and 1300 to 1999.
- In this lecture, we talk about IP ACLs, but others exist to filter packets based on ethernet types, addresses, route, etc.

```
R1(config)# access-list 1 permit 192.168.1.0 0.0.0.255
R1(config)# access-list 1 deny 192.168.2.0 0.0.0.255
R1(config)# access-list 1 permit any
```

Q: What IP/mask could we used instead of any?

Standard Numbered ACLs

- Standard ACLs match the source IP address.
- Numbered ACLs are configured and identified by number (ACL 1, ACL 2, etc…).
  - For standard ACLs, these range from 1 to 99 and 1300 to 1999.
- In this lecture, we talk about IP ACLs, but others exist to filter packets based on ethernet types, addresses, route, etc.

```
R1(config)# access-list 1 permit 192.168.1.0 0.0.0.255
R1(config)# access-list 1 deny 192.168.2.0 0.0.0.255
R1(config)# access-list 1 permit any
```

Q: 0.0.0.0 255.255.255.255 (0.0.0.0/0)

Standard Numbered ACLs

Once defined, you can apply your ACL to an interface with:

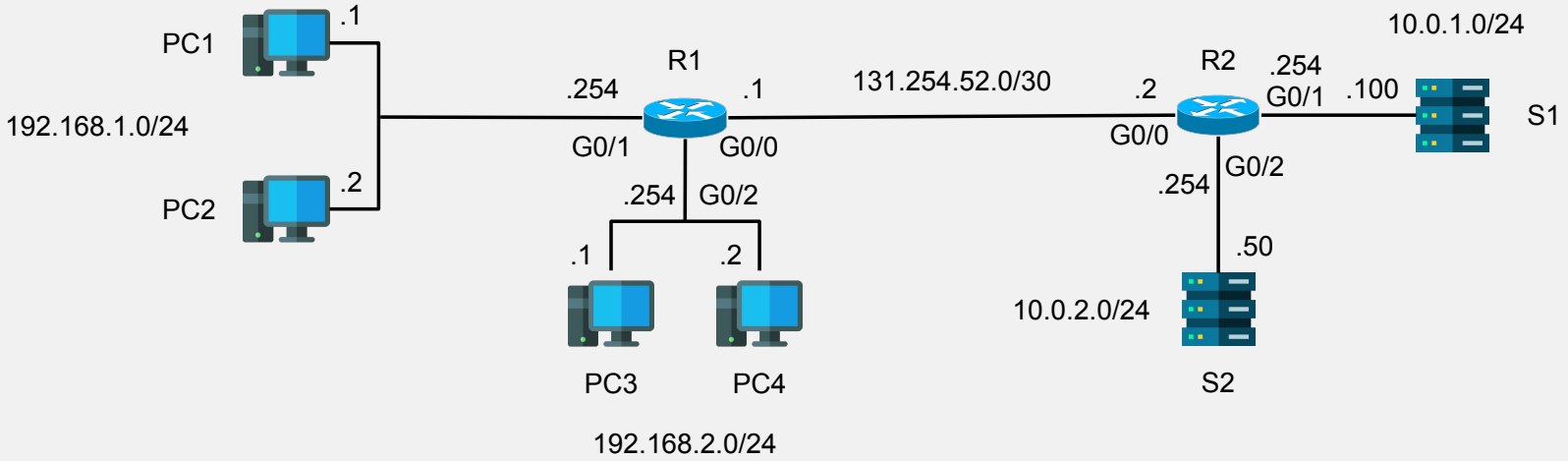
```
R1(config-if)# ip access-group number {in | out}
```

DIY Example

```
R1(config)# access-list 1 permit 192.168.1.1
R1(config)# access-list 1 deny 192.168.1.0 0.0.0.255
R1(config)# access-list 1 permit any
R1(config)# Interface G0/2
R1(config-if)# ip access-group 1 out
```

Goals:

- PC1 can access 192.168.2.0/24
- Other PCs in 192.168.1.0/24 should not access 192.168.2.0/24.



Standard ACLs Good practice

- Standard ACLs should be applied **as close to the destination** as possible to avoid blocking false positive.

Standard Named ACLs

- Standard ACLs match the source IP address.
- Named ACLs are identified by names within the router.
- Configuration is made in a similar way as interfaces or VLAN, by entering a configuration mode:

```
R1(config)# ip access-list standard acl-name
R1(config-std-nacl)# [entry-number] {deny | permit} ip wildcard-mask
R1(config-std-nacl)# 10 permit any
```

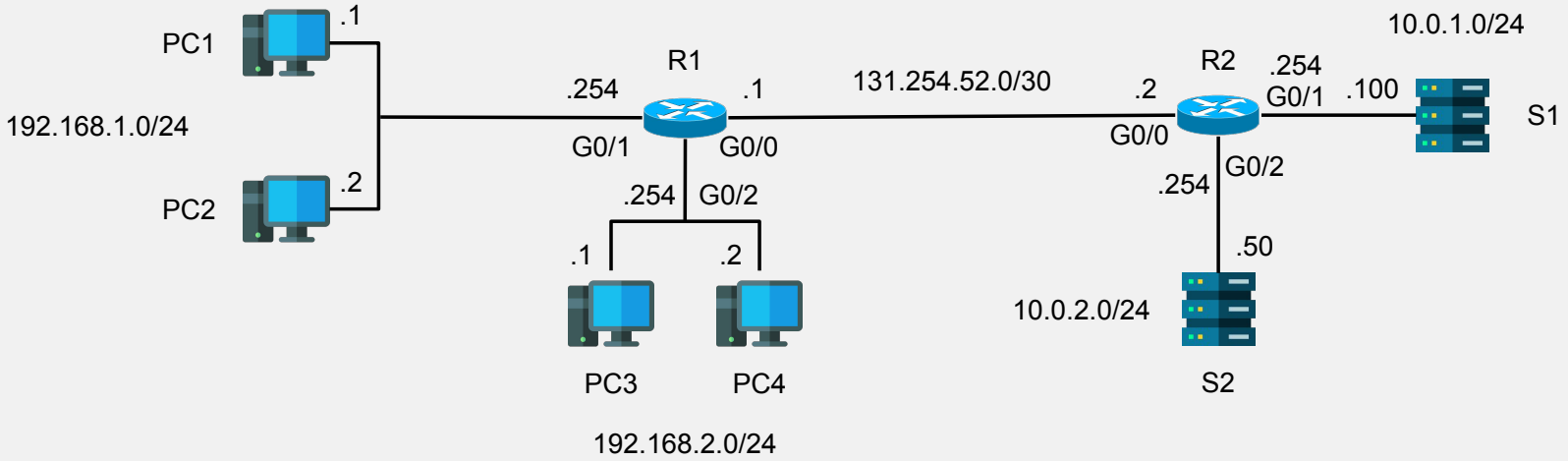
DIY Example 2

R1(config-if)# no ip access-group 1 out
R1(config)# no access-list 1

Now go to R2!

Goals:

- PCs in 192.168.1.0/24 can't access 10.0.2.0/24.
- PC3 can't access 10.0.1.0/24.
- Other PCs in 192.168.2.0/24 can access 10.0.1.0/24.
- PC1 can access 10.0.1.0/24.
- Other PC in 192.168.1.0/24 can't access 10.0.1.0/24.

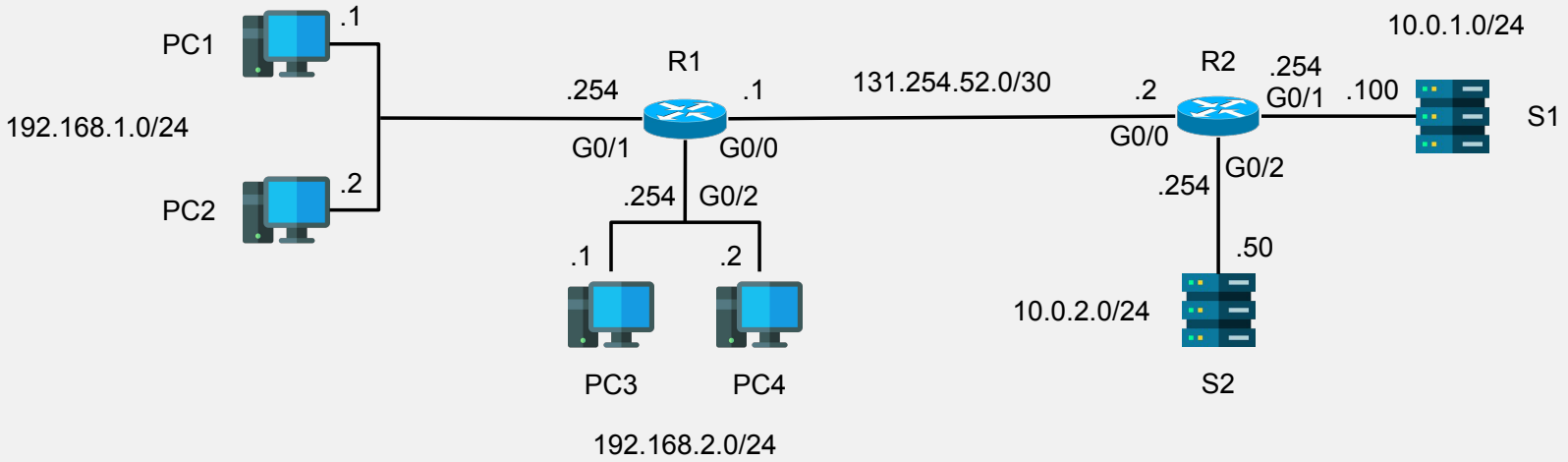


DIY Example 2

```
R2(config)# ip access-list standard to_serv2
R2(config-std-nacl)# deny 192.168.1.0 0.0.0.255
R2(config-std-nacl)# permit any
R2(config-std-nacl)# interface G0/2
R2(config-if)# ip access-group to_serv2 out
```

Goals:

- PCs in 192.168.1.0/24 can't access 10.0.2.0/24.
- PC3 can't access 10.0.1.0/24.
- Other PCs in 192.168.2.0/24 can access 10.0.1.0/24.
- PC1 can access 10.0.1.0/24.
- Other PC in 192.168.1.0/24 can't access 10.0.1.0/24.

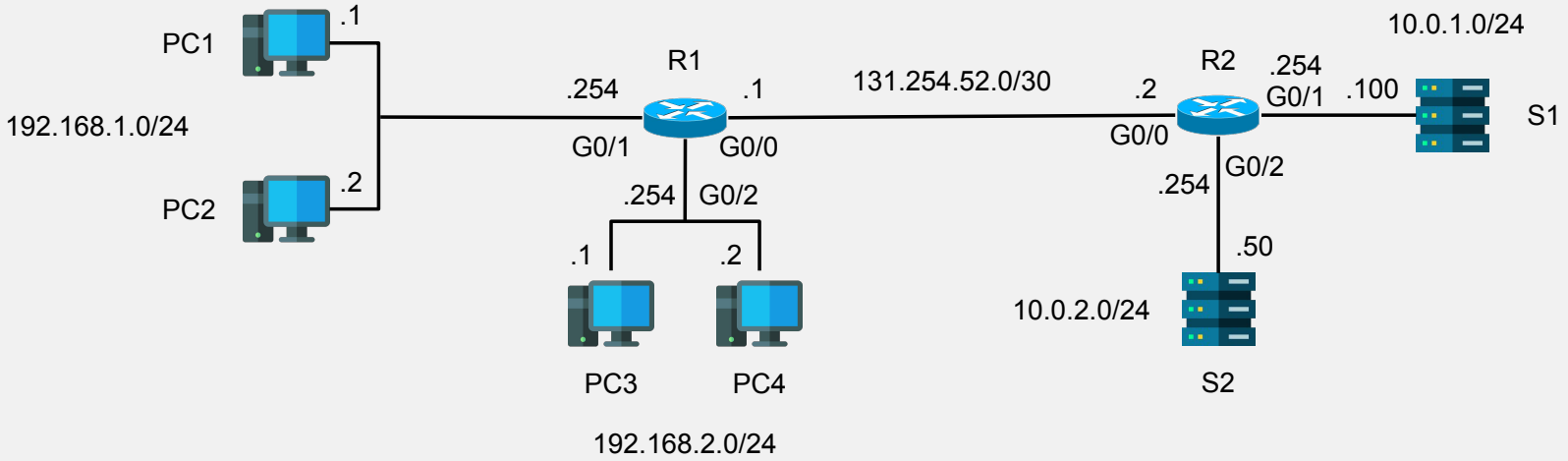


DIY Example 2

```
R2(config)# ip access-list standard to_serv1
R2(config-std-nacl)# deny 192.168.2.1
R2(config-std-nacl)# permit 192.168.2.0 0.0.0.255
R2(config-std-nacl)# permit 192.168.1.1
R2(config-std-nacl)# deny 192.168.1.0 0.0.0.255
R2(config-std-nacl)# permit any
R2(config-std-nacl)# interface G0/1
R2(config-if)# ip access-group to_serv1 out
```

Goals:

- **PCs in 192.168.1.0/24 can't access 10.0.2.0/24.**
- PC3 can't access 10.0.1.0/24.
- Other PCs in 192.168.2.0/24 can access 10.0.1.0/24.
- PC1 can access 10.0.1.0/24.
- Other PC in 192.168.1.0/24 can't access 10.0.1.0/24.

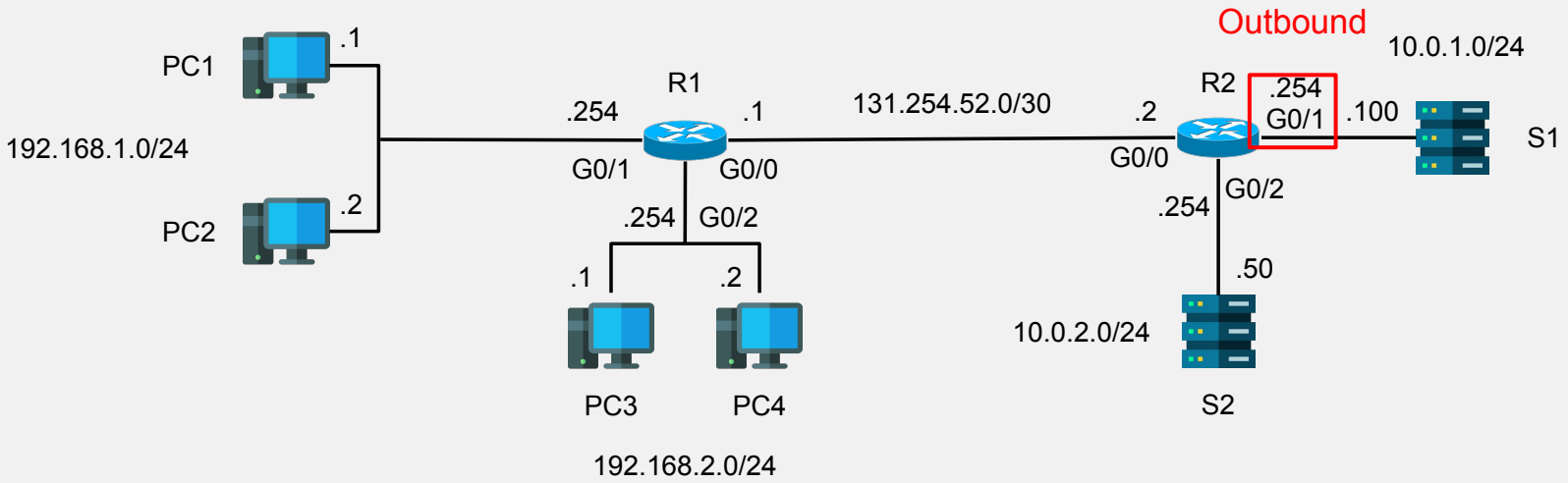


Additional Questions

```
Standard IP access list PC2_ALLOWED
        10 permit 192.168.1.2
        20 deny any
```

Goals:
- Only PC2 can access S1.

Question:
- On which interface and direction?



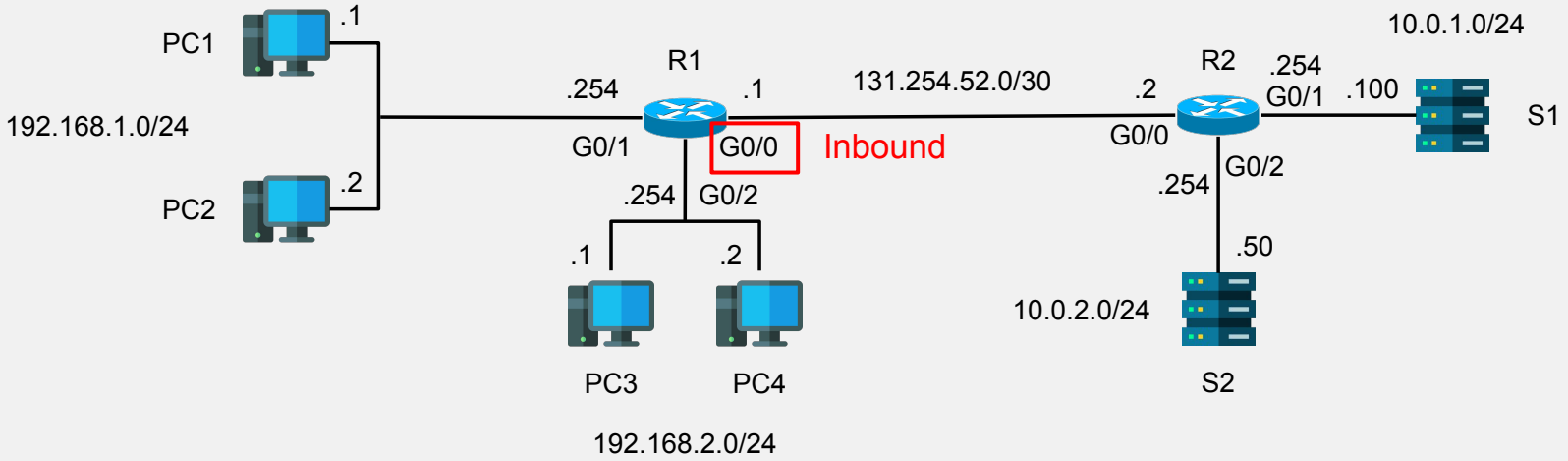
Additional Questions

```
Standard IP access list to_serv2
10      permit 192.168.1.1
20      permit 192.168.1.2
30      deny 192.168.2.1
40      deny 192.168.1.0 0.0.0.255
50      permit 192.168.2.0 0.0.0.255
60      permit 192.168.0.0 0.0.0.255
70      permit any
```

Question:

- Which PCs can access S2?

Answer: ALL PCs



Extended ACLs

Extended ACLs

- Function in the same fashion as Standard ACLs but are much more specific.
- Can be numbered or named.
  - For extended numbered: ranges 100 - 199, 2000-2699.
- Can be used to match entry based on src/dst IPs, layer 4 protocol, and src/dst port numbers.

```
R1(config)# access-list number {deny | permit} protocol src-ip dst-ip      //numbered
R1(config)# ip access-list extended {name | number}                        //named
R1(config-ext-nacl)# [seg-num] {deny | permit} protocol src-ip dst-ip
```

IP and Protocol matching in Extended ACLs

- In extended ACLs, specific keywords are used:
  - **host**: specific host IPs need to be flagged as is.
  - **any**: any IP addresses.
  - **ip**: match every protocol.

```
R1(config-ext-nacl)# deny tcp any 10.0.2.0 0.0.0.255 // any tcp to 10.0.2.0/24
R1(config-ext-nacl)# deny udp host 192.168.1.1 any // any udp from 192.168.1.1
R1(config-ext-nacl)# permit ip any any                // ??
```

Matching Port numbers

- Port number can be compared in Extended ACLs using keywords:
  - eq, gt, lt, neq, range…

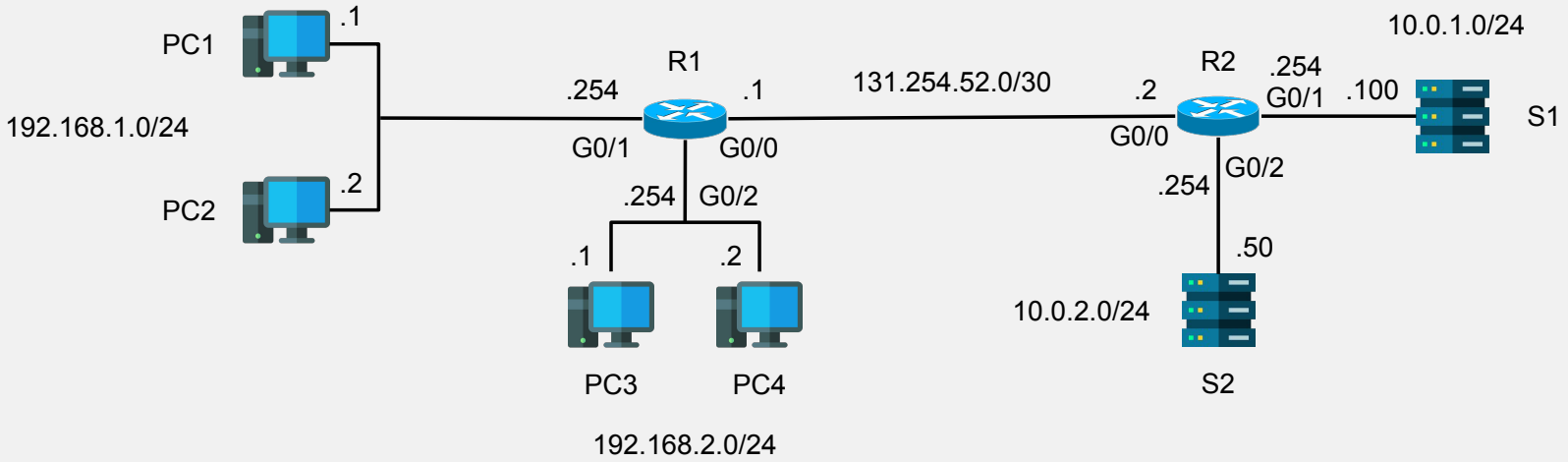
```
R1(config-ext-nacl)# permit tcp host 192.168.1.1 gt 1023 host 10.0.2.100 eq 80 // tcp from above reserved ports to 10.0.2.100 on port 80.

// Practice: Allow traffic for DNS request to server 10.0.1.100
R1(config-ext-nacl)#
R1(config-ext-nacl)# permit udp any host 10.0.1.100 eq 53
R1(config-ext-nacl)# permit tcp any host 10.0.1.100 eq 53
```

DIY Example 3

```
R1(config)# ip access-list extended http_serv1
R1(config-ext-nacl)# deny tcp 192.168.1.0 0.0.0.255 host 10.0.1.100 eq 80
R1(config-ext-nacl)# permit ip any any
// Where?
R1(config-ext-nacl)# interface G0/1
R1(config-if)# ip access-group http_serv1 in
```

Goals:

- PCs in 192.168.1.0/24 can't access S1 with HTTP.

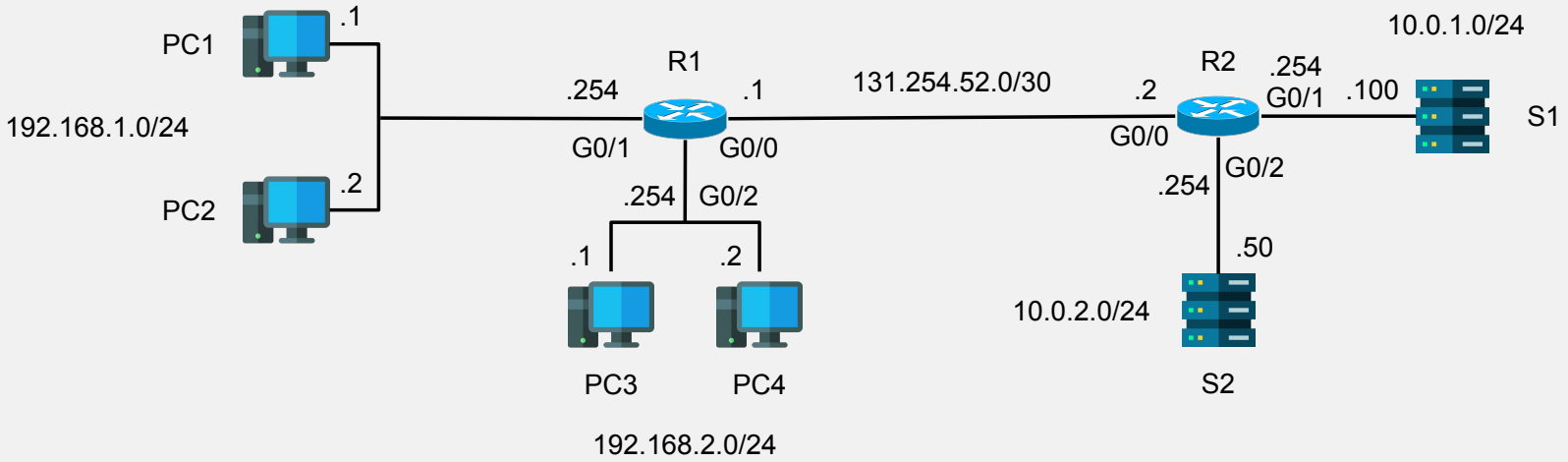


DIY Example 3

```
R1(config)# ip access-list extended no_serv2
R1(config-ext-nacl)# deny ip 192.168.2.0 0.0.0.255 10.0.2.0 0.0.0.255
R1(config-ext-nacl)# permit ip any any
// Where?
R1(config-ext-nacl)# interface G0/2
R1(config-if)# ip access-group no_serv2 in
```

Goals:

- PCs in 192.168.1.0/24 can't access S1 with HTTP.
- PCs in 192.168.2.0/24 can't access 10.0.2.0/24.

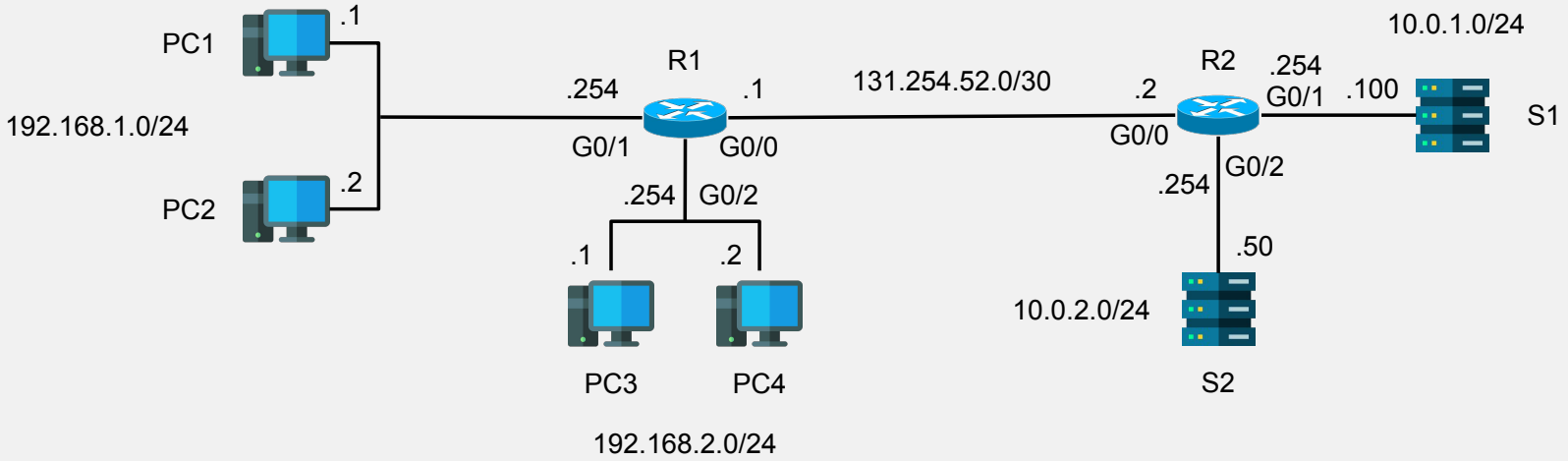


DIY Example 3

```
R1(config)# ip access-list extended no_ping
R1(config-ext-nacl)# deny icmp 192.168.1.0 0.0.0.255 host 10.0.1.100
R1(config-ext-nacl)# deny icmp 192.168.1.0 0.0.0.255 host 10.0.2.100
R1(config-ext-nacl)# deny icmp 192.168.2.0 0.0.0.255 host 10.0.1.100
R1(config-ext-nacl)# permit ip any any
// Where?
R1(config-ext-nacl)# interface G0/0
R1(config-if)# ip access-group no_ping out
```

Goals:

- PCs in 192.168.1.0/24 can't access S1 with HTTP.
- PCs in 192.168.2.0/24 can't access 10.0.2.0/24.
- No PCs should ping any servers.



Extended ACLs Good practice

- Unlike Standard ACLs, Extended ACLs should be applied **as close to the source** as possible to block specific traffic as soon as possible.

Additional Questions

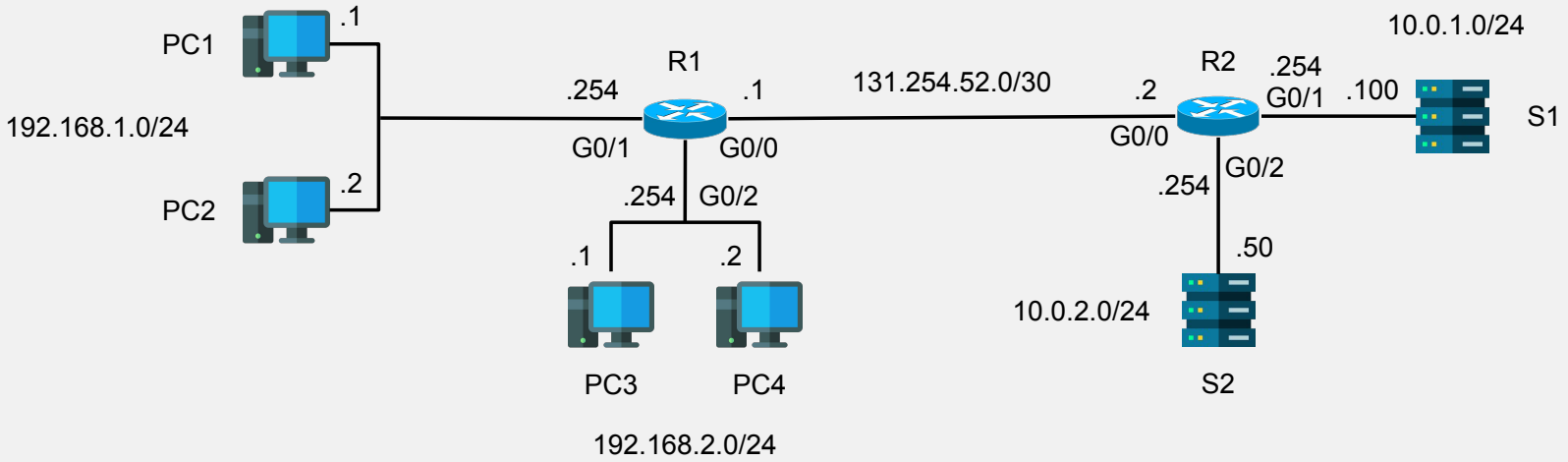
```
R1(config)# ip access-list extended too_late_to_find_a_proper_name
R1(config-ext-nacl)# permit udp 192.168.1.0 0.0.0.255 10.0.2.0 0.0.0.255 eq 80
R1(config-ext-nacl)# permit udp 192.168.1.0 0.0.0.255 10.0.2.0 0.0.0.255 eq
443
R1(config-ext-nacl)# deny ip any any
R1(config-ext-nacl)# interface G0/1
R1(config-if)# ip access-group too_late_to_find_a_proper_name out
```

Goals:

- Allow 192.168.1.0/24 to access S1 specifically with HTTP and HTTPs.
- Nothing else

Question:

- What should be changed?



Additional Questions

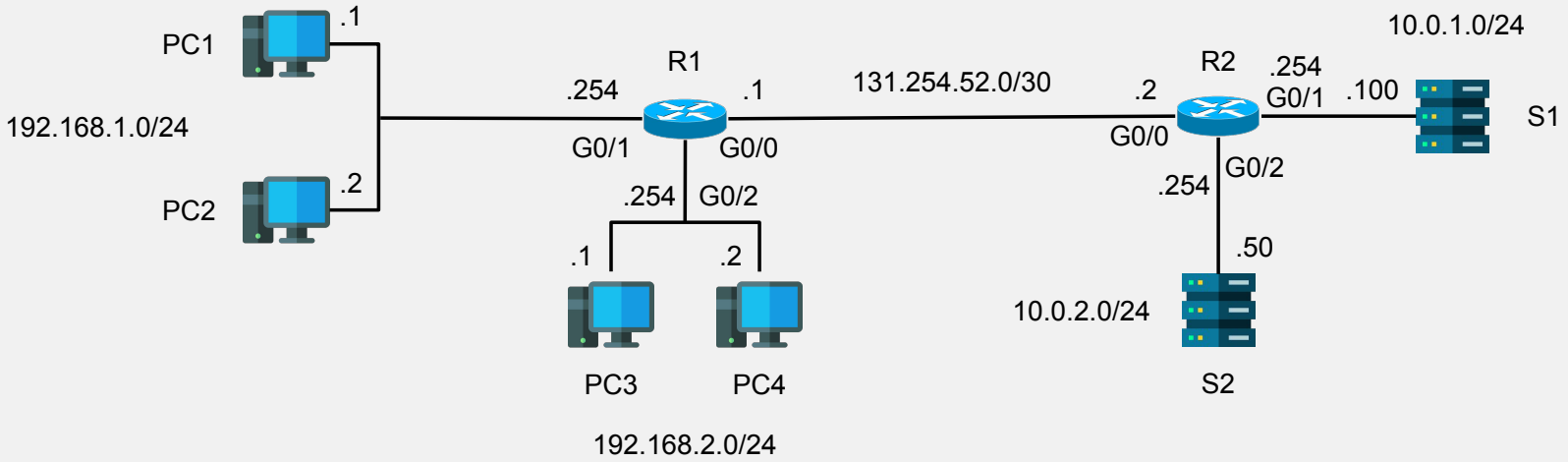
```
R1(config)# ip access-list extended too_late_to_find_a_proper_name
R1(config-ext-nacl)# permit tcp 192.168.1.0 0.0.0.255 host 10.0.1.100 eq 80
R1(config-ext-nacl)# permit tcp 192.168.1.0 0.0.0.255 host 10.0.1.100 eq 443
R1(config-ext-nacl)# deny ip any any
R1(config-ext-nacl)# interface G0/1
R1(config-if)# ip access-group too_late_to_find_a_proper_name in
```

Goals:

- Allow 192.168.1.0/24 to access S1 specifically with HTTP and HTTPs.
- Nothing else

Question:

- What should be changed?



Resources and Acknowledgements

- Cisco documentations.
- Jeremy McDowell's Cisco CCNA lectures.