Standard ACLs (Access Control Lists)

CCNA Exam Context

- ACLs are listed under Section 5: Security Fundamentals, specifically topic 5.6 ("configure and verify access control lists")
- Topic does not specify IPv4 or IPv6
- For CCNA, you only need to learn IPv4 ACLs
- IPv6 ACLs are not currently required for the exam

Course Structure for ACLs

- ACLs will be covered over two days:
  - This video (Day 34): Standard ACLs
  - Next video (Day 35): Will cover another kind of ACL

Lecture Introduction

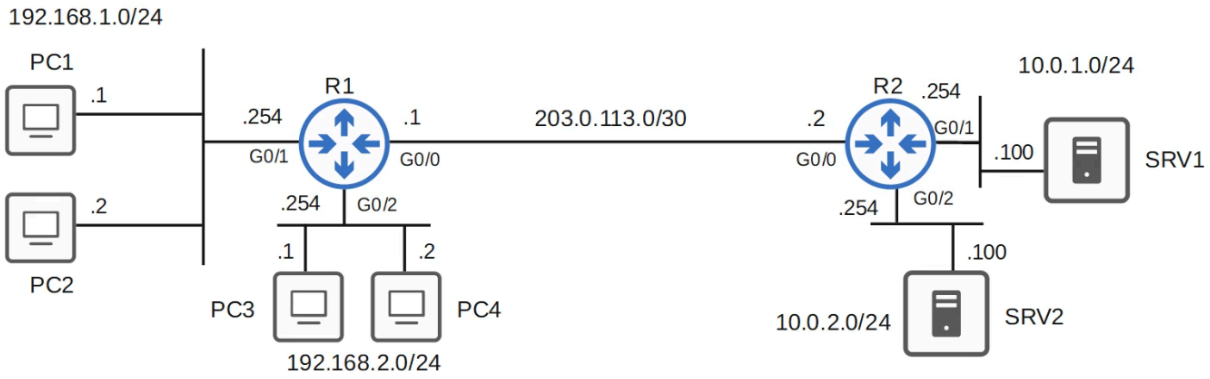
- **What are ACLs?**
- **ACL Logic**: How ACLs are processed by the router
- **Types of ACLs**: Basic types on Cisco routers
- **Configuration**:
  - Standard Numbered ACLs
  - Standard Named ACLs

What are ACLs?

- **Definition**: ACLs (Access Control Lists) control access to the network.
  - Host A should be allowed to access Server A
  - But Host B should not be allowed to access Server A
- **Primary Purpose (Security)**: ACLs function as a packet filter, instructing the router to **permit** or **discard** specific traffic.
- **Filtering Behavior**: By default, a router forwards packets if it has a route. ACLs override this; a packet may be discarded even if a route exists.
- **Filtering Criteria**: ACLs can filter traffic based on:
  - Source and destination IP addresses
  - Source and destination Layer 4 port numbers

Example Network Topology & Requirements

Network Topology Diagram:



- Two routers, R1 and R2, connected via a point-to-point link.
- **R1 connected networks**:
  - 192.168.1.0/24 (PC1, PC2)
  - 192.168.2.0/24 (PC3, PC4)

- **R2 connected networks**:
  - 10.0.1.0/24 (SRV1)
  - 10.0.2.0/24 (SRV2)

Note: In network diagrams, a switch connecting PCs to a router is often omitted for simplicity; the network segment is represented directly.

Building an ACL (Conceptual)

- **Principle**: Do not configure ACLs randomly; configure them to achieve specific network policy requirements.
- **Network Policy**:
  - Hosts in 192.168.1.0/24 should be able to access the 10.0.1.0/24 network (e.g., access files on SRV1).
  - Hosts in 192.168.2.0/24 should not be able to access the 10.0.1.0/24 network (e.g., PC3 and PC4 should not access SRV1).
  - How can we use ACL's to achieve this?

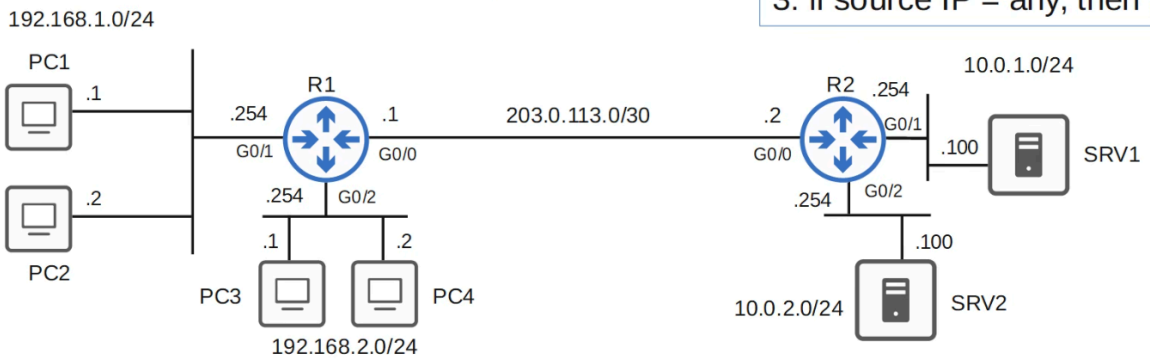
ACL Structure and Composition

- **Configuration Location**: ACLs are configured globally on the router, in global config mode.
- **Composition**: ACLs are made up of an ordered sequence of ACEs (Access Control Entries).

Example: ACL 1 Conceptual Design

- REQUIREMENT:
  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.
- ACLs are configured globally on the router. (global config mode)
- They are an ordered sequence of ACEs. (Access Control Entries)

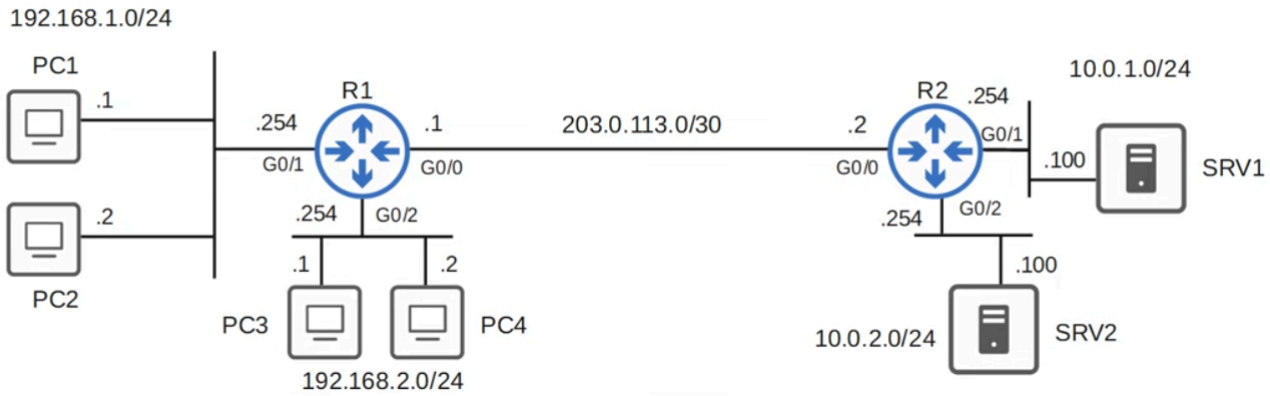
ACL 1:
1: if source IP = 192.168.1.0/24, then permit
2: if source IP = 192.168.2.0/24, then deny
3: if source IP = any, then permit



- **ACE 1**: If source IP = 192.168.1.0/24 → **permit** the packet (let the router forward it).
- **ACE 2**: If source IP = 192.168.2.0/24 → **deny** the traffic.
- **ACE 3**: All other traffic → **permit**.

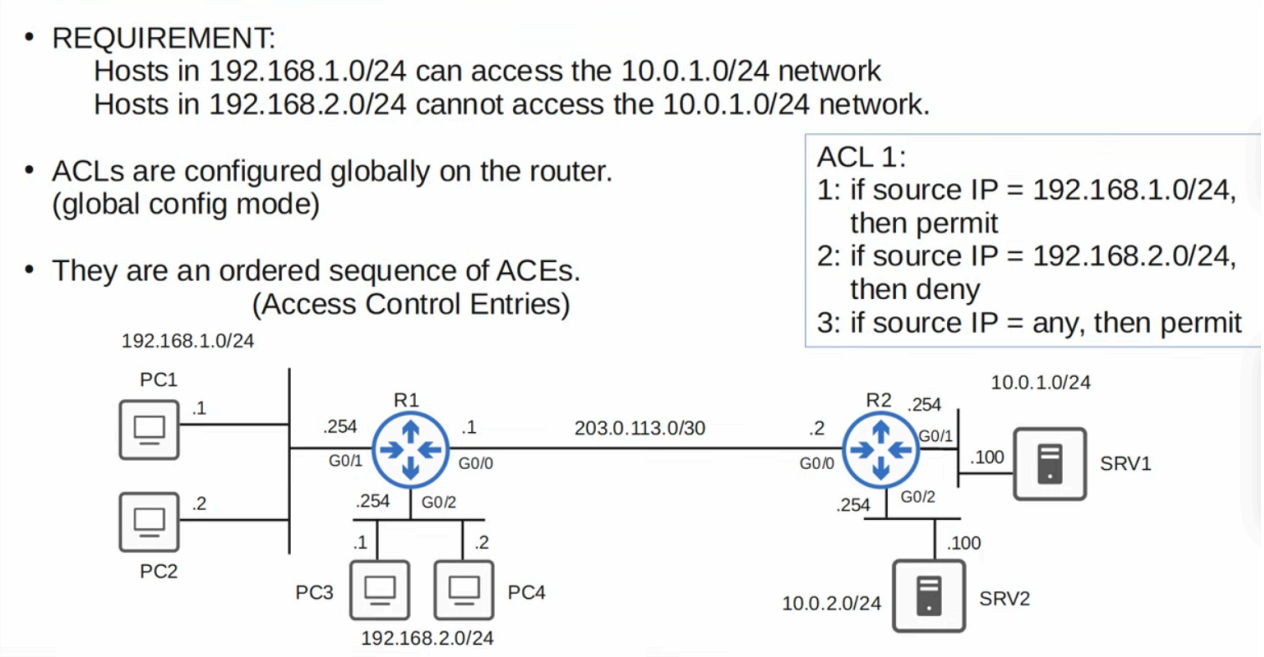
Note: The order of these entries is very important.

Applying ACL 1 to R1's G0/2 Interface (Examples)



The following examples use the previously created ACL 1 and the requirement that hosts in 192.168.1.0/24 can access 10.0.1.0/24, but hosts in 192.168.2.0/24 cannot.

- **Example: Applying ACL 1 outbound on G0/2**
  - **Result**: This **does not** fulfill the requirement.
  - **Explanation**: The ACL only filters traffic exiting G0/2.
    - When PC3 pings SRV1, the traffic enters G0/2 on R1. The ACL is not checked, and the packet is forwarded.
    - When SRV1 replies, the traffic exits G0/2 on R1. The ACL is checked.



1. **ACE 1** (Permit 192.168.1.0/24): No match. The source is SRV1, so that doesn't apply.
2. **ACE 2** (Deny 192.168.2.0/24): No match. The source is SRV1, so that doesn't apply either.
3. **ACE 3** (Permit All): **Match**. R1 reaches the last entry which says permit all other traffic, so it forwards the reply to PC3.

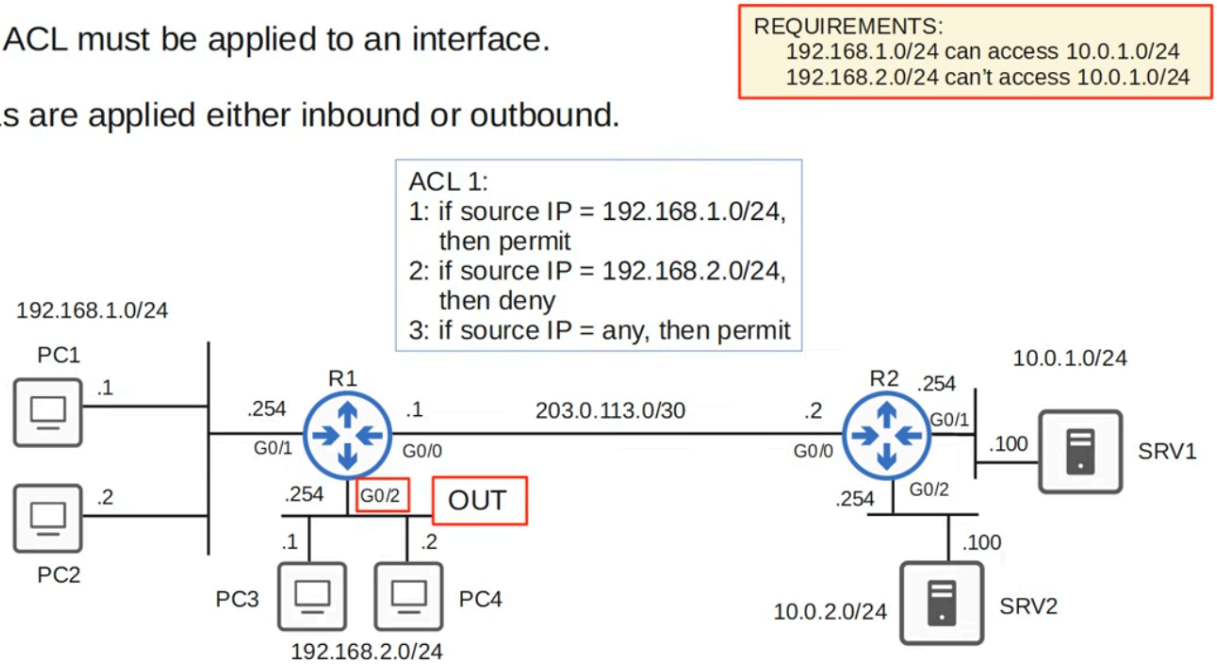
- **Conclusion**: PC3 was able to access SRV1, so the requirement failed.

Applying an ACL

- **Rule**:
  - Configuring an ACL in global config mode does not make it take effect.
  - After being created, the ACL must be **applied to an interface**.
  - **Direction**: When applying an ACL to an interface, you specify a direction:
    - **Inbound**: The router checks packets that are **entering** the interface.
    - **Outbound**: The router checks packets that are **exiting** the interface.
- **Important**: Depending on which interface the ACL is applied to and which direction is chosen, the ACL will either succeed or fail in meeting the network requirements.

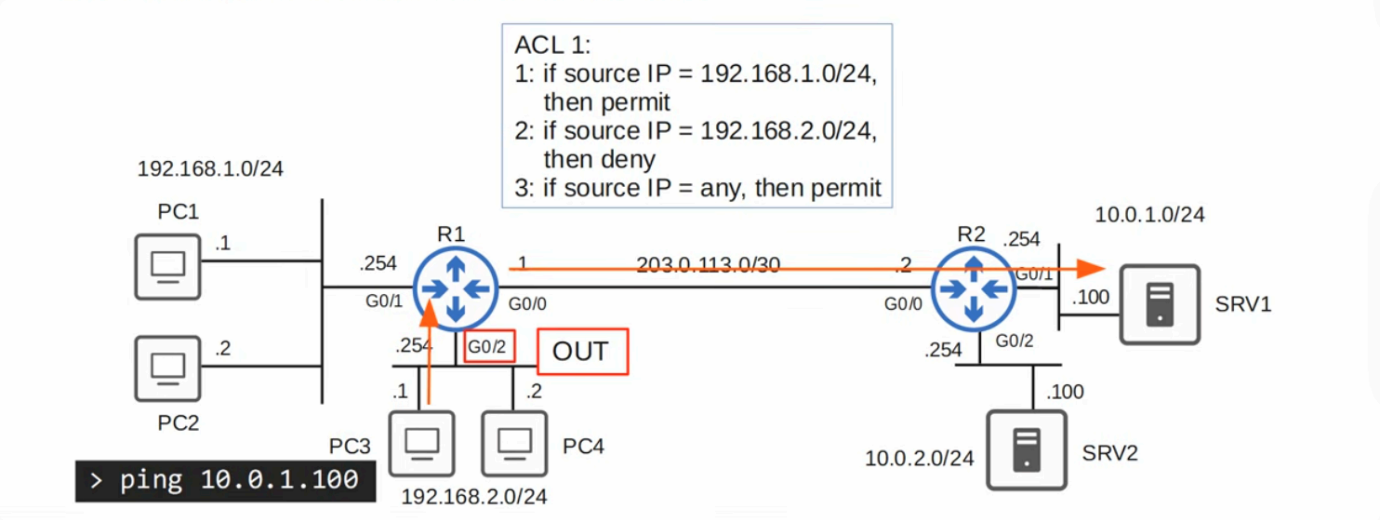
Example: Applying ACL 1 Outbound on G0/2

- Configuring an ACL in global config mode will not make the ACL take effect.
- The ACL must be applied to an interface.
- ACLs are applied either inbound or outbound.

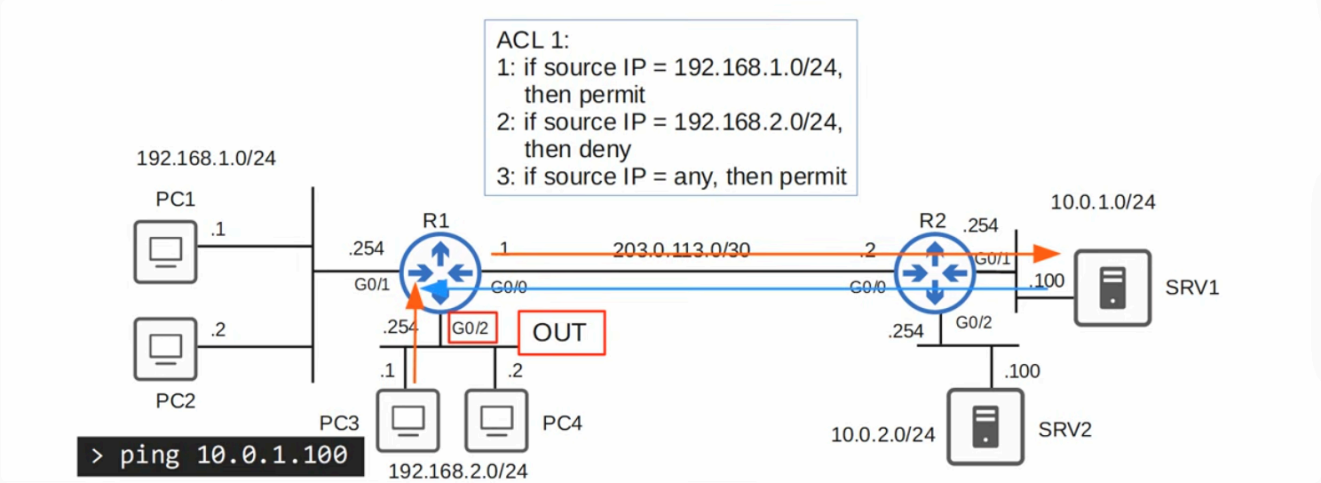


- **Scenario**: ACL 1 is applied **outbound** on R1's G0/2 interface. The requirement is to allow 192.168.1.0/24 to access 10.0.1.0/24, but block 192.168.2.0/24.
- **Result**: This **does not** fulfill the requirement.

- **Explanation**: The ACL only filters traffic exiting G0/2, not entering it.

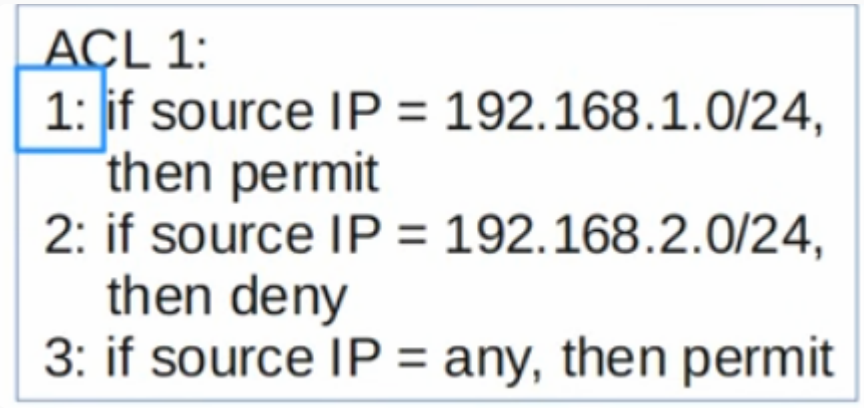


- **Step 1 (Request)**: PC3 (192.168.2.1) tries to ping SRV1 (10.0.1.100). The ping reaches R1. Because the traffic is **entering** G0/2, R1 does **not** check the ACL. R1 forwards the traffic to R2, which sends it to SRV1.

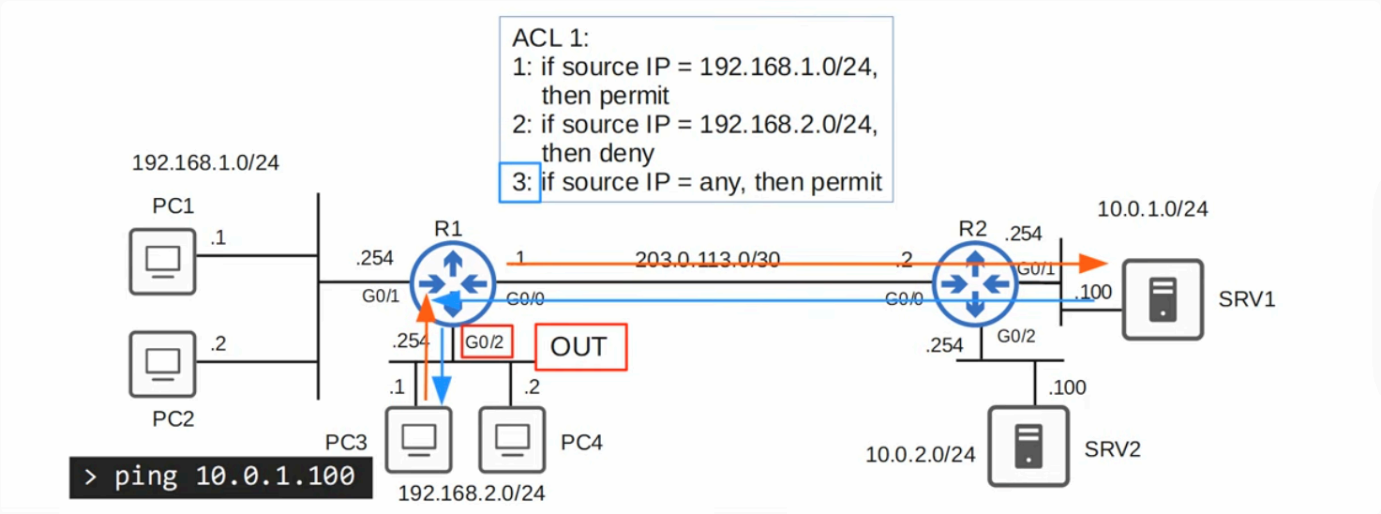


- **Step 2 (Reply)**: SRV1 sends a reply back to PC3. The reply reaches R1, which must forward it out of G0/2. Because the ACL is applied **outbound** on G0/2, R1 checks the ACL.

- **ACL Processing (Reply)**:



1. **ACE 1** (Permit 192.168.1.0/24): No match. The source is SRV1, not in this subnet.
2. **ACE 2** (Deny 192.168.2.0/24): No match. The source is SRV1, not in this subnet.
3. **ACE 3** (Permit All): **Match**. The packet is permitted.



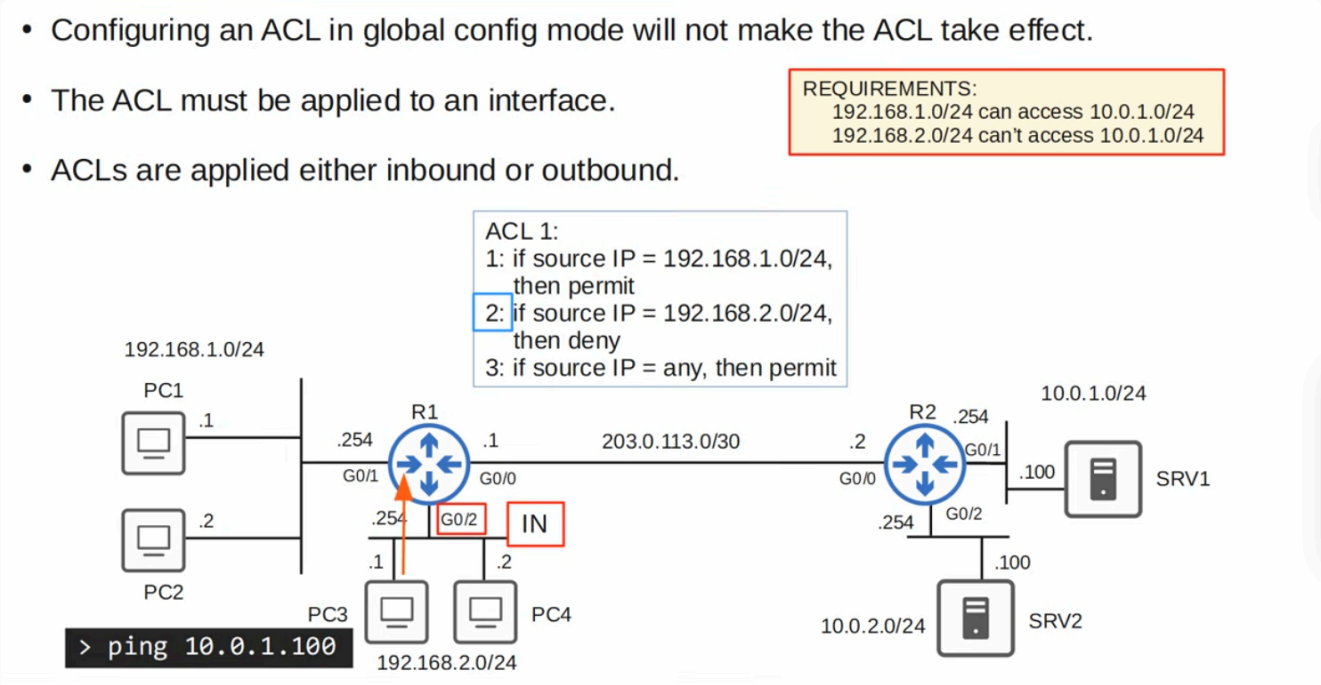
- R1 forwards the reply to PC3.

- **Conclusion**: PC3 was able to access SRV1, even though hosts in 192.168.2.0/24 should not be able to. The ACL was not applied correctly.

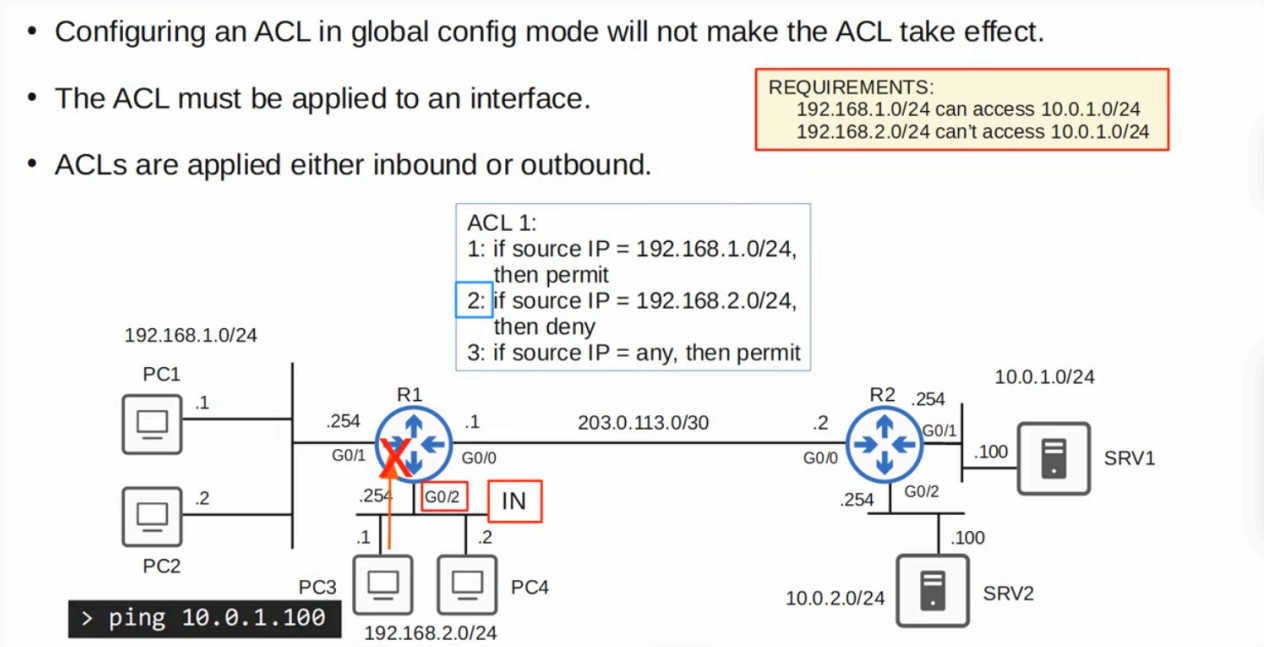
Analysis of Applying ACL 1 Inbound on G0/2

- Applying the ACL inbound on G0/2 means R1 checks the ACL for all traffic entering G0/2.

- **Scenario**: PC3 (192.168.2.1) tries to ping SRV1.



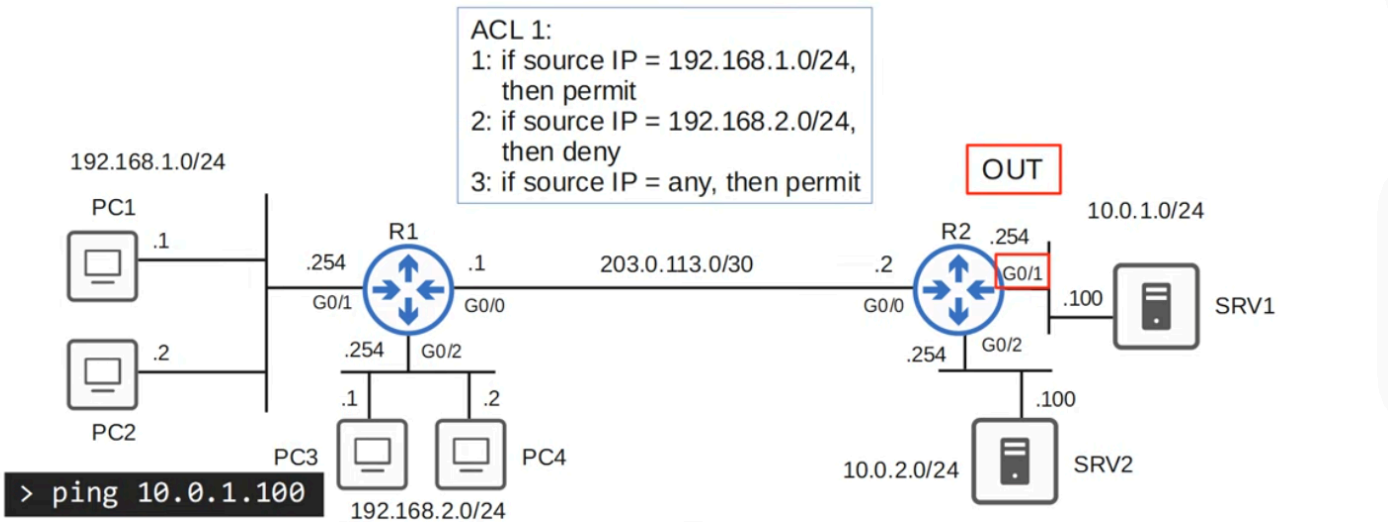
- R1 checks the packet against the ACL entries in order:
  1. **ACE 1**: Source = 192.168.1.0/24. No match (PC3 is not in this subnet).
  2. **ACE 2**: Source = 192.168.2.0/24. **Match** (PC3 is in this subnet).



- **Action**: R1 takes the action for ACE 2, which is **deny**. R1 drops the packet.

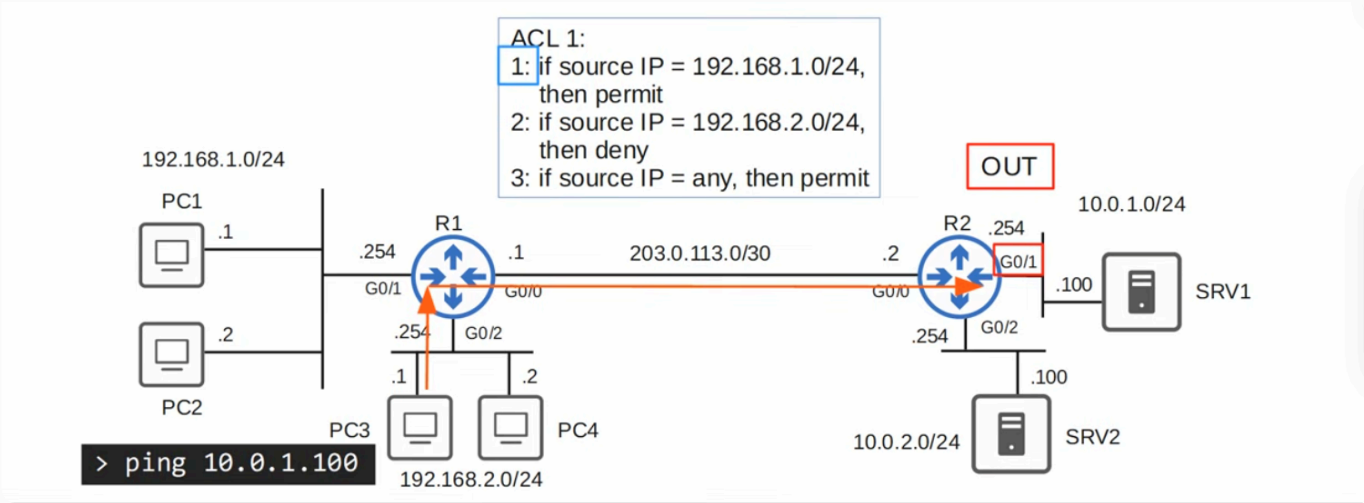
- **Key Concept**: Once a router finds a match and takes an action, it stops checking the other entries. The "permit all other traffic" entry is ignored.
- **Fulfilling the Requirements?**
  - **Yes**:
    - 192.168.1.0/24 can still access 10.0.1.0/24 (no ACL stopping them).
    - 192.168.2.0/24 is blocked from 10.0.1.0/24 (R1 dropped PC3's traffic).
  - **No (Too Restrictive)**:
    - Applying the ACL inbound on G0/2 blocks 192.168.2.0/24 from communicating with **all** other networks outside their local LAN.
    - PC3 and PC4 can only communicate with each other.
    - This is not the best way to apply the ACL.
- **Alternatives**: We could try applying it to R1's G0/0 or R2's G0/0, but we will look at the best option next.

Best ACL Placement: Outbound on R2's G0/1



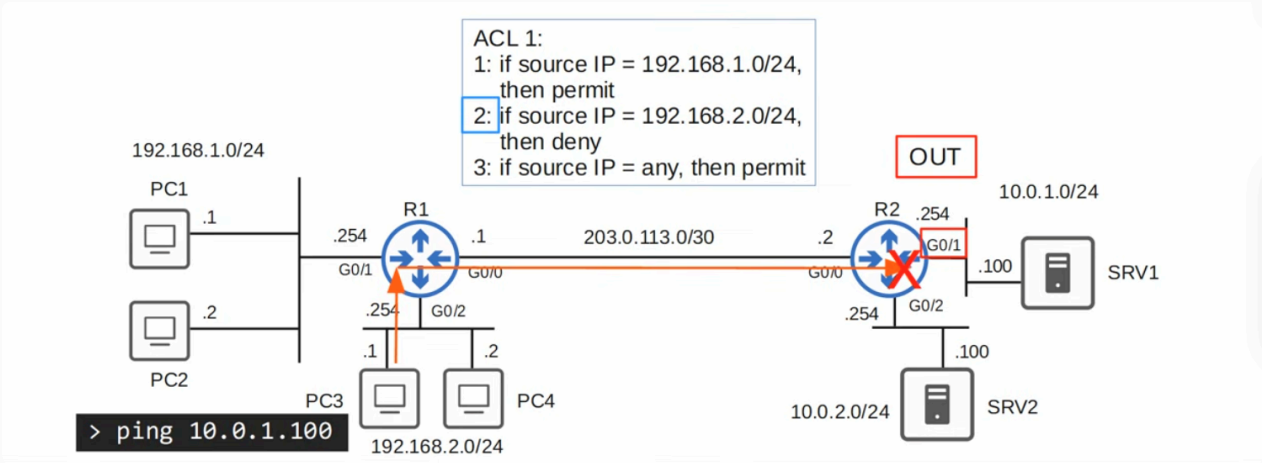
- **Optimal Location**: The best location to place this ACL is **outbound on R2's G0/1 interface**.

- **Example: PC3 tries to ping SRV1**:



  - R2 will check the ACL before forwarding the packet out of its G0/1 interface.
  - It processes the ACEs in order:

    1. **ACE 1**: If source IP = 192.168.1.0/24, then permit. The source (PC3) is not in that subnet, so R1 checks the next entry.



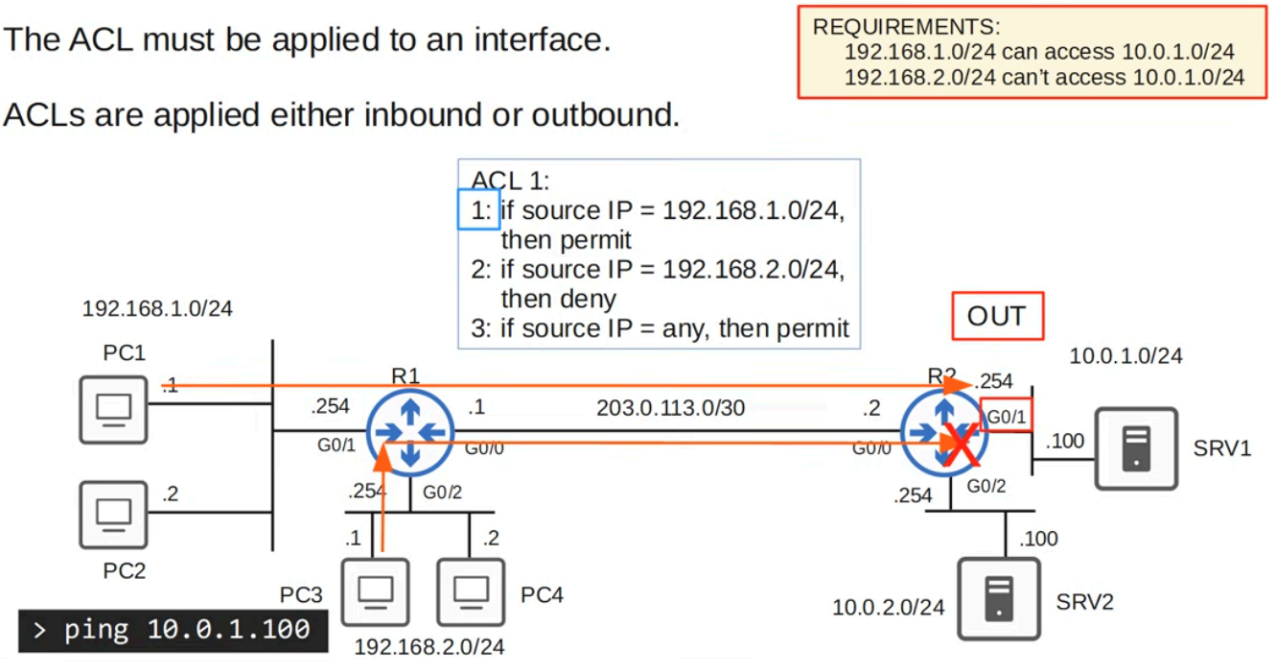
    2. **ACE 2**: If source IP = 192.168.2.0/24, then deny. The source is in that subnet (192.168.2.1).

  - **Result**: The packet is denied, and R2 will not forward it.

- **Conclusion**: This satisfies the requirement that hosts in 192.168.2.0/24 cannot access 10.0.1.0/24.

Example: PC1 Accessing SRV1 (Best Placement)

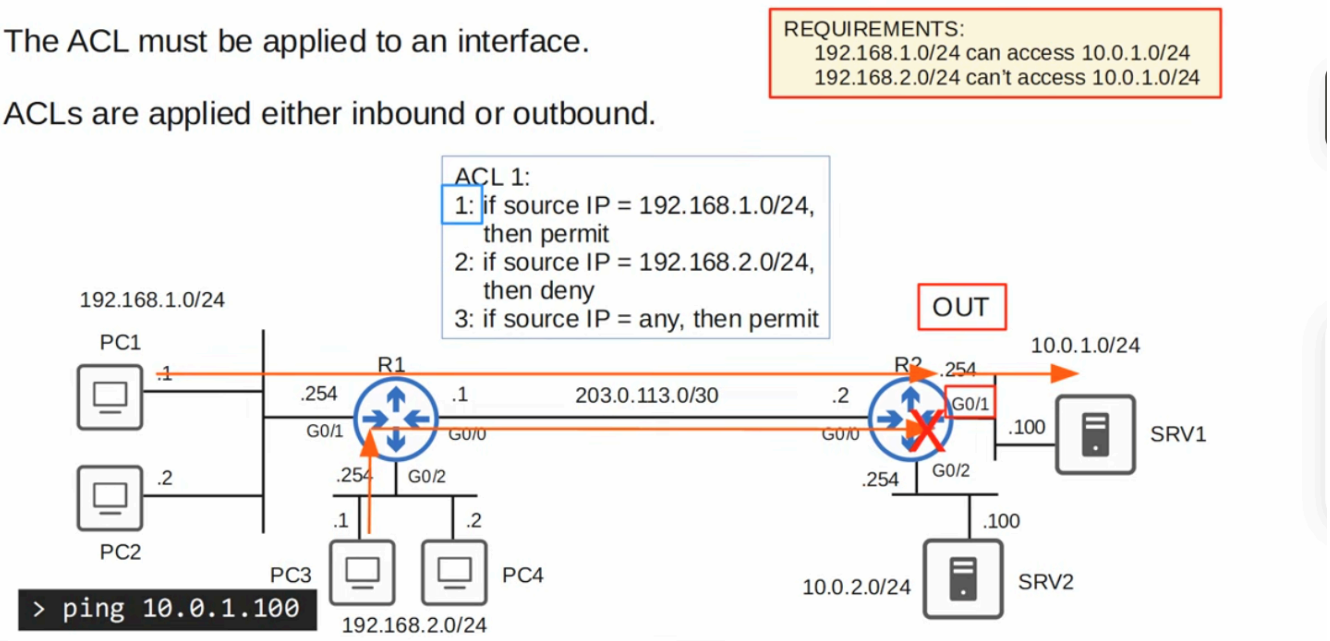
- Configuring an ACL in global config mode will not make the ACL take effect.
- The ACL must be applied to an interface.
- ACLs are applied either inbound or outbound.



- **Scenario**: PC1 (192.168.1.1), located in the 192.168.1.0/24 network, tries to ping SRV1.

- **ACL Processing**:

- Configuring an ACL in global config mode will not make the ACL take effect.
- The ACL must be applied to an interface.
- ACLs are applied either inbound or outbound.

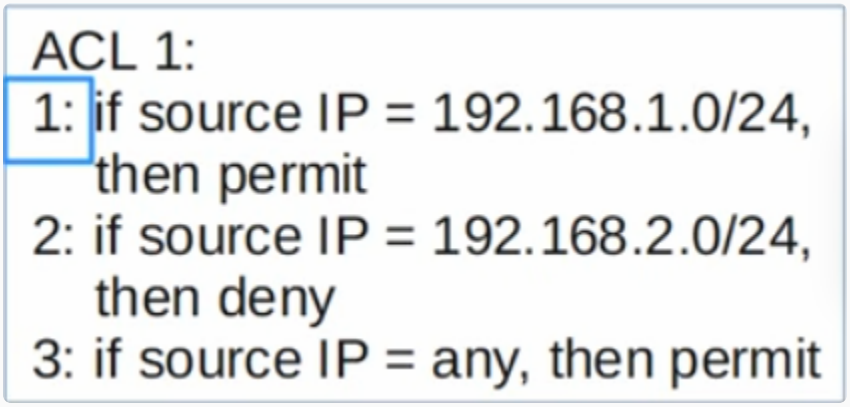


  - Before R2 forwards the packet out of its G0/1 interface, it checks the ACL.
  - The router checks the first entry: "If source IP is in 192.168.1.0/24, then permit."
  - Since the source IP is 192.168.1.1, this entry matches.
- **Result**: The packet is permitted and R2 forwards it to SRV1.
- **Conclusion**: Both requirements have been satisfied, and there is no effect on other traffic.

ACL Processing Review

- **Configuration and Application**:
  - ACLs are configured in global config mode.
  - After configuration, they must be applied to an interface.
  - When applying, you specify a direction (**inbound** or **outbound**) to tell the router whether to check packets entering or exiting the interface.

- **Composition**:
  - ACLs are made up of one or more ACEs (Access Control Entries).
- **Processing Order**:
  - When the router checks a packet against the ACL, it processes the ACEs in order, from top to bottom.



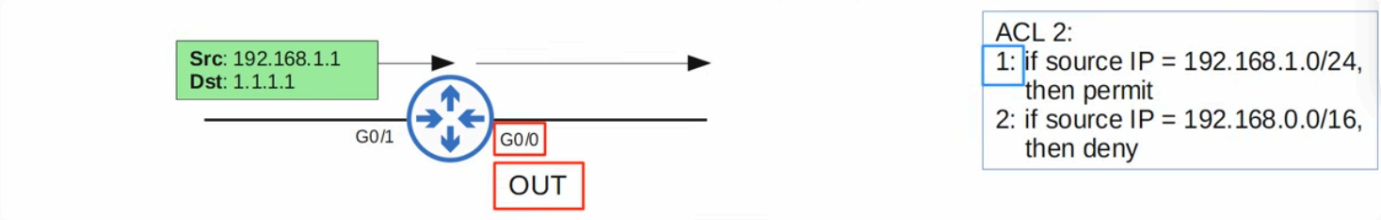
  - **Example**: In ACL 1, the router will check if the packet's source is in 192.168.1.0/24, then it will check if the packet's source is in 192.168.2.0/24, and if it doesn't match either of those, it will permit it.
- **Match Behavior**:
  - If the packet matches one of the entries in the ACL, the router takes the action and stops processing the ACL.
  - All entries below the matching entry will be ignored.

Order of ACEs Example

- **Scenario**:

  - Here we have a router and an ACL.
  - **ACE 1**: If source IP = 192.168.1.0/24, then permit.
  - **ACE 2**: If source IP = 192.168.0.0/16, then deny.
  - The ACL is applied **outbound** on the G0/0 interface.

- **Result**:

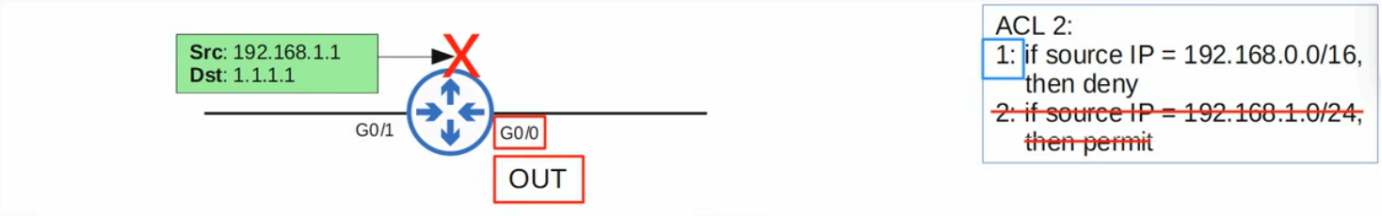


  - If a packet with a source IP of 192.168.1.1 arrives on G0/1, the router will check it against the ACL before forwarding it out of G0/0.
  - The source 192.168.1.1 matches the first entry, so the router will forward the packet as normal.
  - The second entry will simply be ignored.

Reversed Order Example

- **Scenario**:
  - Now the entries in ACL 2 are reversed.
  - **ACE 1**: Deny 192.168.0.0/16.
  - **ACE 2**: Permit 192.168.1.0/24.

- **Result**:



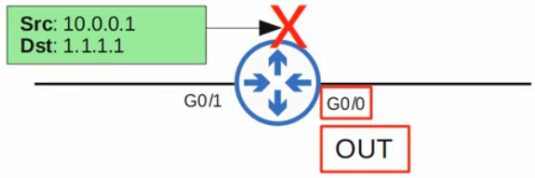
  - If the same packet (source 192.168.1.1) is received by the router, it will check ACL 2 before forwarding the packet.
  - The first entry tells the router to deny the packet (because 192.168.1.1 is inside the 192.168.0.0/16 range).
  - The packet is discarded and not forwarded out of the interface.
  - Entry 2, which tells the router to permit that packet, is ignored.
- **Key Concept**: This demonstrates how important the order of the entries in an ACL is.

Applying Multiple ACLs

- **Rule**: A maximum of one ACL can be applied to a single interface per direction.
- **Limit**:
  - One inbound ACL is allowed.
  - One outbound ACL is allowed.
  - Maximum of two ACLs per interface.
- **Replacement**: If you apply a second ACL to an interface in the same direction as another one, it will replace the previous one.

The Implicit Deny

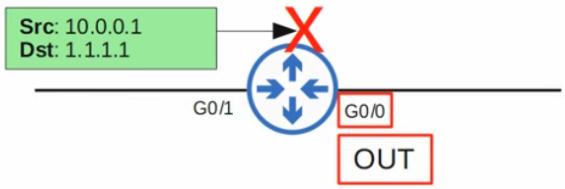
- **Scenario**:



ACL 2:
1: if source IP = 192.168.1.0/24, then permit
2: if source IP = 192.168.0.0/16, then deny
(3: if source IP = any, then deny)

  - We have the same router and the same ACL (ACE 1 permits 192.168.1.0/24; ACE 2 denies 192.168.0.0/16).
  - The router receives a packet with source IP **10.0.0.1**.
  - Before forwarding it out of G0/0, the router checks it against the ACL.

- **Processing**:



ACL 2:
1: if source IP = 192.168.1.0/24, then permit
2: if source IP = 192.168.0.0/16, then deny

  - 10.0.0.1 doesn't match the first entry (192.168.1.0/24).
  - It doesn't match the second entry (192.168.0.0/16).
  - So, what happens? The answer is, the router will **deny** the packet; it will not forward it.

- **Definition**: This is what we call the **'implicit deny'**.

  - Even though there is no entry in the ACL telling the router to deny the packet, it's like there is an invisible entry at the end: "if source IP = any, then deny".
  - This is true for all ACLs.
- **Summary**:
  - There is an implicit deny at the end of all ACLs.
  - This tells the router to deny all traffic that doesn't match any of the configured entries in the ACL.
- **Warning**: Always be aware of the implicit deny when configuring ACLs, or you might deny traffic that you didn't want to deny.

Types of ACLs

Overview: There are two main types of ACLs, and each of those has two sub-types.

- Standard ACLs: Match based on **Source IP address** only
  - → Standard Numbered ACLs
  - → Standard Named ACLs

- Extended ACLs: Match based on **Source/Destination IP, Source/Destination port,** etc.
  - → Extended Numbered ACLs
  - → Extended Named ACLs

- **Standard ACLs**:
  - These match based on source IP address only, so they are quite simple.
  - **Standard Numbered ACLs**: Identified with a number (e.g., ACL 1, ACL 2).
  - **Standard Named ACLs**: Identified with a name.
  - **Note**: There are differences in how you configure numbered and named ACLs, but we'll get to that later.
- **Extended ACLs**:
  - These are more complex and can match based on:
    - Source and/or destination IP address.
    - Source and/or destination port numbers.
    - As well as some other things.
  - Like standard ACLs, there are numbered and named versions of extended ACLs, too.
- **Focus of this Lecture**:
  - Today we'll focus on standard ACLs. All of the examples so far have been standard ACLs, filtering packets only based on their source IP address.
  - In Day 35 we'll cover extended ACLs.

Standard Numbered ACLs

- **Matching Criteria**: Standard ACLs match traffic based only on the source IP address of the packet.
  - The router doesn't check the destination IP, the source Layer 4 port, the destination port, etc.
  - It just looks at the source IP address of the packet and decides to forward or block it.
- **Identification**: Numbered ACLs are identified with a number (e.g., ACL 1, ACL 2).
  - This allows you to configure multiple ACLs on a single router.
  - There are also named ACLs, which will be introduced later.

Number Ranges

Protocol	Range
Standard IP	1–99 and 1300–1999
Extended IP	100–199 and 2000–2699
Ethernet type code	200–299
Ethernet address	700–799
Transparent bridging (protocol type)	200–299
Transparent bridging (vendor code)	700–799
Extended transparent bridging	1100–1199
DECnet and extended DECnet	300–399

Xerox Network Systems (XNS)	400–499
Extended XNS	500–599
AppleTalk	600–699
Source-route bridging (protocol type)	200–299
Source-route bridging (vendor code)	700–799
Internetwork Packet Exchange (IPX)	800–899
Extended IPX	900–999
IPX Service Advertising Protocol (SAP)	1000–1099

- **Standard ACL Ranges**: 1–99 and 1300–1999.
  - Originally, standard ACLs could only use 1–99 (maximum of 99 ACLs on a single router).
  - Later this was expanded to include 1300–1999.
  - You can't configure a standard ACL with the number 100, for example. The number has to be in one of these ranges.
- **Note**: There are many different types of ACLs, each with their own range of numbers. For the CCNA, just remember the standard ACL ranges.

Configuration Command

- **Basic Syntax**: `ACCESS-LIST [number] {DENY | PERMIT} [ip-address] [wildcard-mask]`
  - The number must be in the range 1–99 or 1300–1999.
  - You specify `deny` or `permit`, followed by the IP address and wildcard mask.
  - **Important**: Don't try to use a standard subnet mask when configuring ACLs; use a wildcard mask.

Configuration Examples

- **Denying a Single Host (/32)**:

  - The basic command to configure a standard numbered ACL is:
    `R1(config)# access-list number {deny | permit} ip wildcard-mask`

```
R1(config)# access-list 1 deny 1.1.1.1 0.0.0.0
R1(config)# access-list 1 deny 1.1.1.1
R1(config)# access-list 1 deny host 1.1.1.1
```

  - `ACCESS-LIST 1 DENY 1.1.1.1 0.0.0.0`: Explicitly denies host 1.1.1.1.
  - `ACCESS-LIST 1 DENY 1.1.1.1`: If you omit the wildcard mask, the router assumes a /32 mask.
  - `ACCESS-LIST 1 DENY HOST 1.1.1.1`: Uses the `HOST` keyword (older method, but still works).

- **Note**: The second and third options only work for /32 matches. To match a larger network (like a /24), you must specify the wildcard mask.

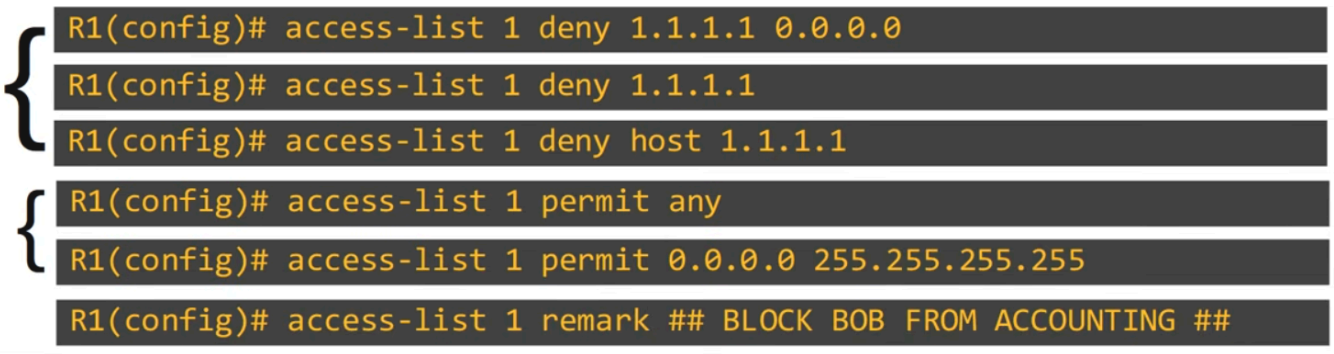
Permitting All Traffic

- The basic command to configure a standard numbered ACL is:
  `R1(config)# access-list number {deny | permit} ip wildcard-mask`

- **Using ANY**: `ACCESS-LIST 1 PERMIT ANY`
- **Using IP and Wildcard**: `ACCESS-LIST 1 PERMIT 0.0.0.0 255.255.255.255`
- **Effect**: Both commands permit all traffic from any source IP. The `ANY` keyword is simply a shorthand for `0.0.0.0 255.255.255.255`.

Adding Remarks

- The basic command to configure a standard numbered ACL is:
  `R1(config)# access-list number {deny | permit} ip wildcard-mask`



- **Purpose**: Remarks are like interface descriptions; they help you remember the purpose of the ACL but have no effect on its operation.
- **Command**: `ACCESS-LIST 1 REMARK [text]`
- **Note**: Hashtags (`#`) are not necessary but help with readability in the configuration.

Verification and Application of Standard Numbered ACLs

- **Configuration and Verification Example**:

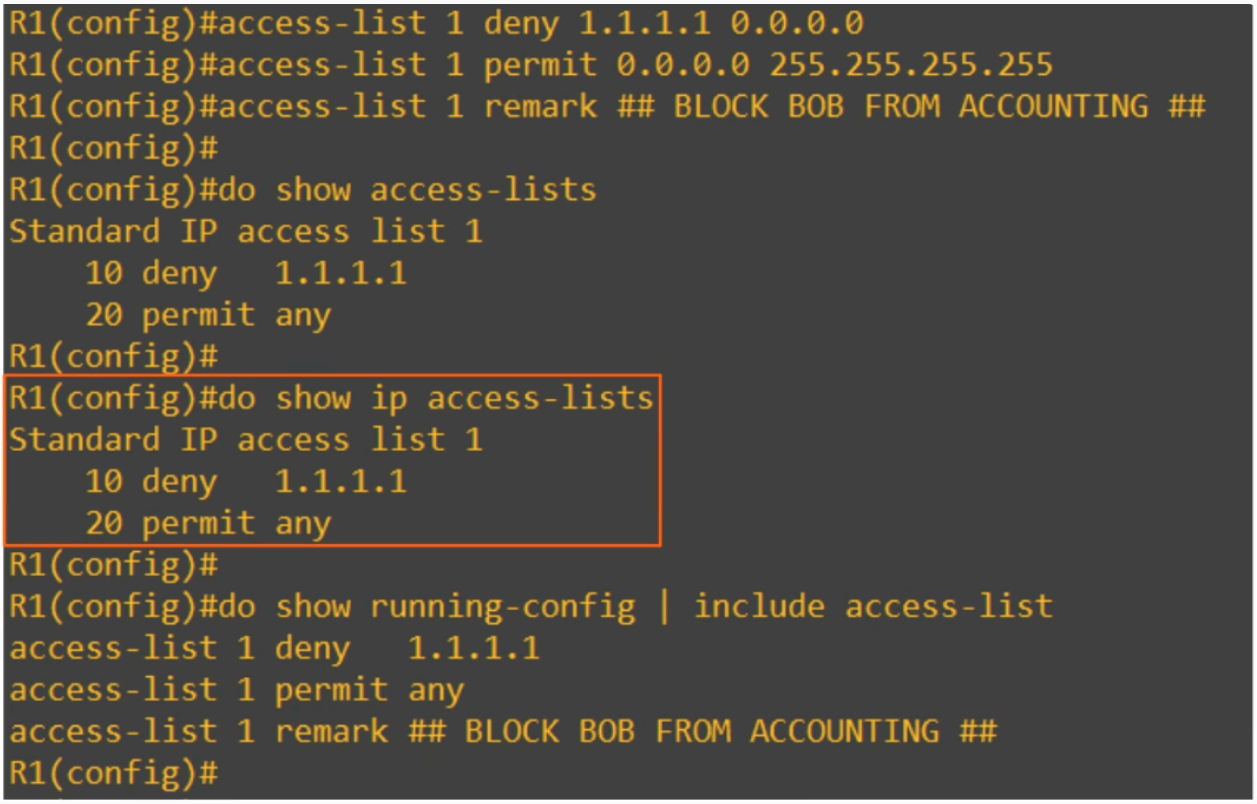
```
R1(config)#access-list 1 deny 1.1.1.1 0.0.0.0
R1(config)#access-list 1 permit 0.0.0.0 255.255.255.255
R1(config)#access-list 1 remark ## BLOCK BOB FROM ACCOUNTING ##
R1(config)#
R1(config)#do show access-lists
Standard IP access list 1
    10 deny   1.1.1.1
    20 permit any
R1(config)#
R1(config)#do show ip access-lists
Standard IP access list 1
    10 deny   1.1.1.1
    20 permit any
R1(config)#
R1(config)#do show running-config | include access-list
access-list 1 deny   1.1.1.1
access-list 1 permit any
access-list 1 remark ## BLOCK BOB FROM ACCOUNTING ##
R1(config)#
```

- The professor configured entries using the full IP address and wildcard mask (e.g., `DENY 1.1.1.1 0.0.0.0` and `PERMIT 0.0.0.0 255.255.255.255`).

```
R1(config)#access-list 1 deny 1.1.1.1 0.0.0.0
R1(config)#access-list 1 permit 0.0.0.0 255.255.255.255
R1(config)#access-list 1 remark ## BLOCK BOB FROM ACCOUNTING ##
R1(config)#
R1(config)#do show access-lists
Standard IP access list 1
    10 deny   1.1.1.1
    20 permit any
R1(config)#
R1(config)#do show ip access-lists
Standard IP access list 1
    10 deny   1.1.1.1
    20 permit any
R1(config)#
R1(config)#do show running-config | include access-list
access-list 1 deny   1.1.1.1
access-list 1 permit any
access-list 1 remark ## BLOCK BOB FROM ACCOUNTING ##
R1(config)#
```

- **Automatic Conversion**: When using `SHOW ACCESS-LISTS`, the router automatically optimized the syntax:
  - `DENY 1.1.1.1 0.0.0.0` converted to `DENY 1.1.1.1`.
  - `PERMIT 0.0.0.0 255.255.255.255` converted to `PERMIT ANY`.
- **Sequence Numbers**: Entries are assigned a number indicating their order (e.g., the first configured entry deny statement was assigned 10, the second permit statement 20).
- **Order Importance**: The order is critical. If `PERMIT ANY` were first, all traffic would be permitted, and a subsequent `DENY` entry would be ignored. Modern routers usually prevent this misconfiguration.

- **Verification Commands**:
  - `SHOW ACCESS-LISTS`: Displays all ACLs configured on the router.
    - *Note*: This command does not show remarks.



- `SHOW IP ACCESS-LISTS`: Displays only IP ACLs. The output is the same as `SHOW ACCESS-LISTS` for IP ACLs.

```
R1(config)#access-list 1 deny 1.1.1.1 0.0.0.0
R1(config)#access-list 1 permit 0.0.0.0 255.255.255.255
R1(config)#access-list 1 remark ## BLOCK BOB FROM ACCOUNTING ##
R1(config)#
R1(config)#do show access-lists
Standard IP access list 1
    10 deny   1.1.1.1
    20 permit any
R1(config)#
R1(config)#do show ip access-lists
Standard IP access list 1
    10 deny   1.1.1.1
    20 permit any
R1(config)#
R1(config)#do show running-config | include access-list
access-list 1 deny   1.1.1.1
access-list 1 permit any
access-list 1 remark ## BLOCK BOB FROM ACCOUNTING ##
R1(config)#
```

- `SHOW RUNNING-CONFIG | INCLUDE ACCESS-LIST`: Displays configuration lines containing "access-list".
    - *Note*: This command **does** display the remarks configured earlier.
    - *Note*: The router automatically applies the same optimizations (converting masks to `ANY` or removing `/32` masks) in the running configuration.

- **Applying the ACL**:

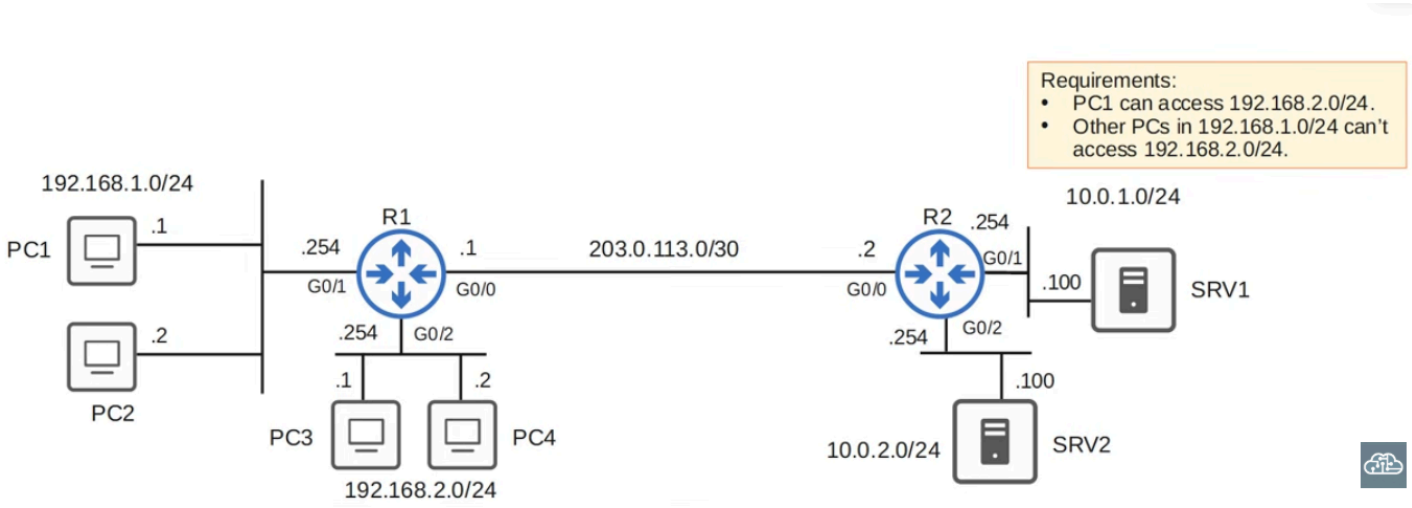
```
R1(config)#access-list 1 deny 1.1.1.1 0.0.0.0
R1(config)#access-list 1 permit 0.0.0.0 255.255.255.255
R1(config)#access-list 1 remark ## BLOCK BOB FROM ACCOUNTING ##
R1(config)#
R1(config)#do show access-lists
Standard IP access list 1
    10 deny   1.1.1.1
    20 permit any
R1(config)#
R1(config)#do show ip access-lists
Standard IP access list 1
    10 deny   1.1.1.1
    20 permit any
R1(config)#
R1(config)#do show running-config | include access-list
access-list 1 deny   1.1.1.1
access-list 1 permit any
access-list 1 remark ## BLOCK BOB FROM ACCOUNTING ##
R1(config)#
```

Apply to an interface:
R1(config-if)# **ip access-group** *number* {**in** | **out**}

  - An ACL does not take effect until it is applied to an interface.
  - **Command**: `IP ACCESS-GROUP [acl-number] {IN | OUT}`
  - **Configuration Mode**: Interface configuration mode.
  - **Important**: The command uses **ACCESS-GROUP**, not ACCESS-LIST.

Standard Numbered ACL Configuration Example

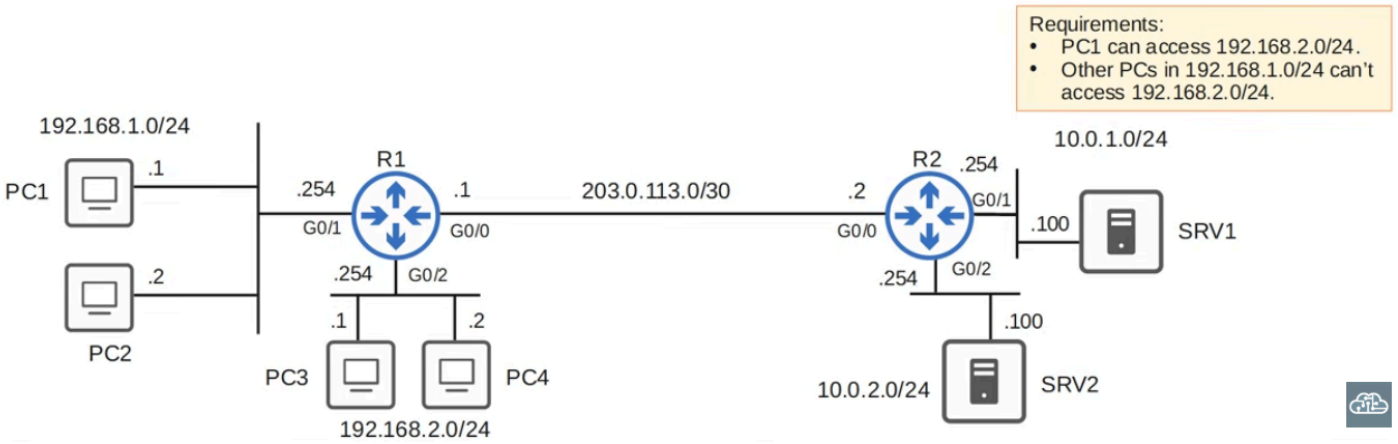
Requirements



- PC1 (192.168.1.1) should be able to access the 192.168.2.0/24 network.
- Other PCs in 192.168.1.0/24 should **not** be able to access 192.168.2.0/24.

Configuration Steps (on R1)

```
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)#
R1(config)#interface g0/2
R1(config-if)#ip access-group 1 out
R1(config-if)#
```



- First, configure ACL 1 with an entry permitting 192.168.1.1/32 (PC1).
- Next, configure an entry denying the entire 192.168.1.0/24 network.
- Finally, configure a "permit any" entry.

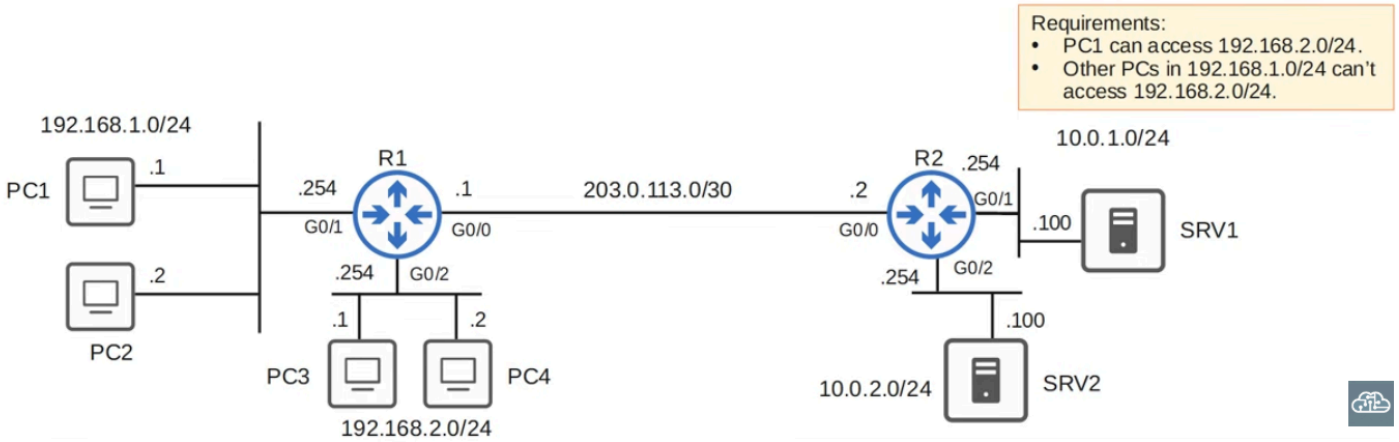
Key Concepts

- **Order is Crucial**: You must permit PC1 *before* denying the 192.168.1.0/24 network.
  - If you denied 192.168.1.0/24 first, PC1 would be blocked because ACLs are processed top-to-bottom. Once a match is found, processing stops.
- **Implicit Deny**: If you do not include the "permit any" entry at the end, the implicit deny will block all other traffic, not just the 192.168.1.0/24 network.

Application

```
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)#
R1(config)#interface g0/2
R1(config-if)#ip access-group 1 out
R1(config-if)#
```

Standard ACLs should be applied as close to the destination as possible.



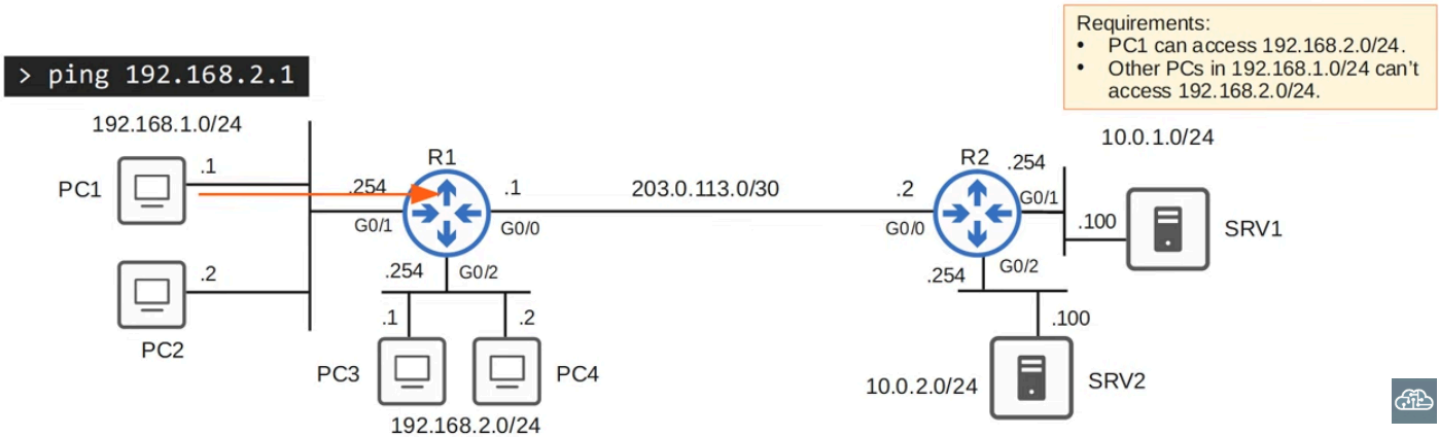
- **Command**: `IP ACCESS-GROUP 1 OUT`
- **Interface**: R1's G0/2 interface (outbound).

Placement Rule of Thumb

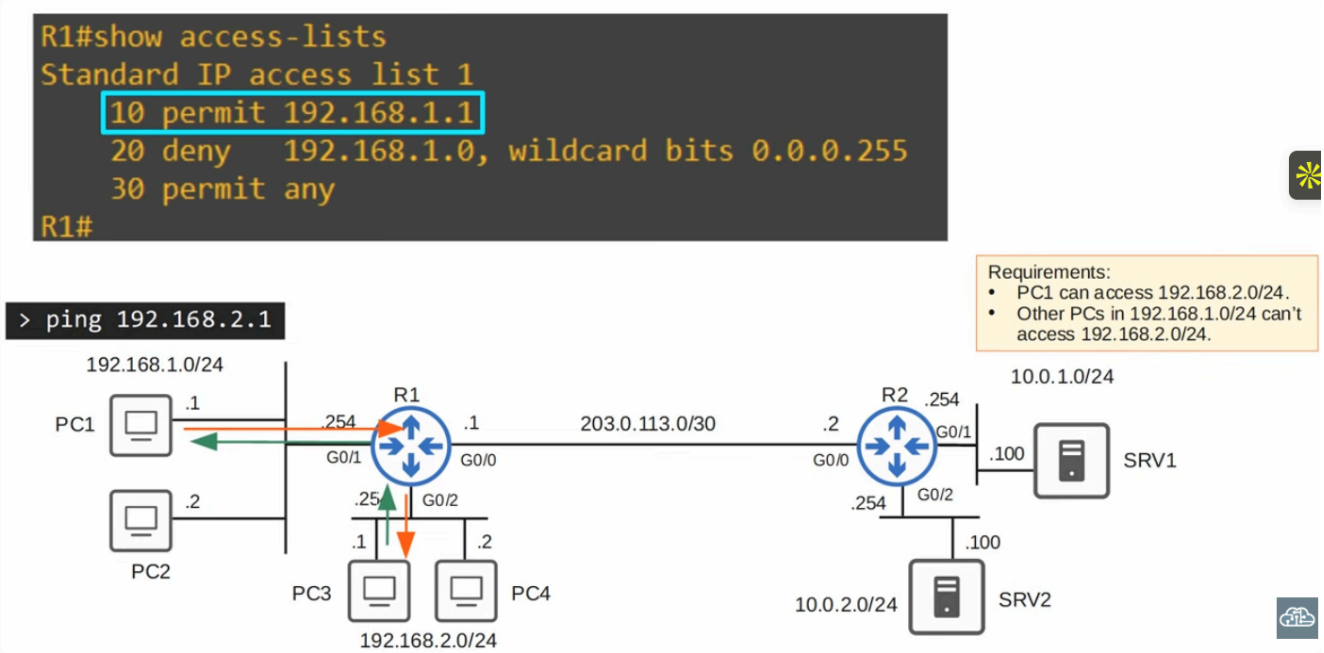
- **Standard ACLs should be applied as close to the destination as possible.**
- In this scenario, we are controlling access to the 192.168.2.0/24 network (the destination).
- If applied inbound on G0/1, it would stop 192.168.1.0/24 from accessing *anything* outside their LAN.
- By applying it outbound on G0/2, we specifically control traffic destined for the 192.168.2.0/24 network without affecting their access to other networks.

Verification of ACL 1 on R1

```
R1#show access-lists
Standard IP access list 1
    10 permit 192.168.1.1
    20 deny   192.168.1.0, wildcard bits 0.0.0.255
    30 permit any
R1#
```

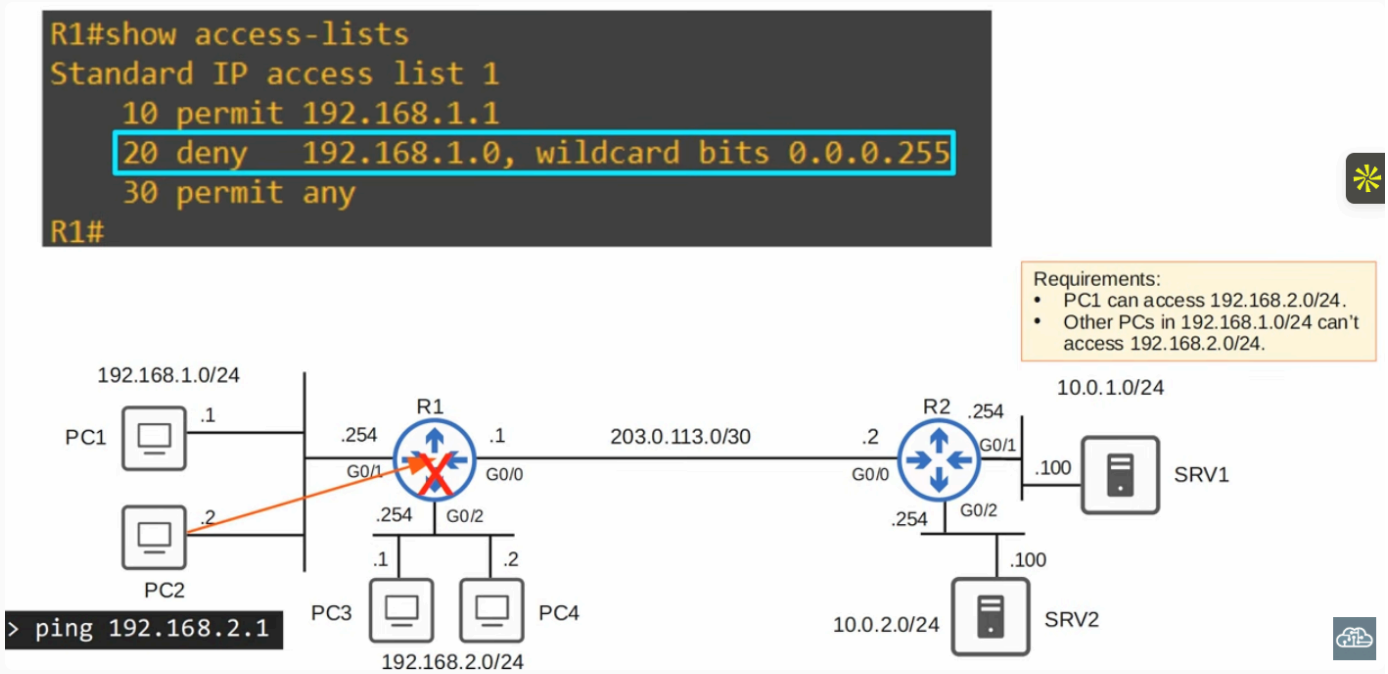


- **Scenario 1: PC1 tries to ping PC3**
  - The ping is received by R1 on G0/1. R1 does **not** check the ACL here because it is not applied to that interface.
  - R1 checks the routing table and determines the packet should be forwarded out of G0/2.
  - Because ACL 1 is applied **outbound** on G0/2, R1 checks the ACL before forwarding.



  - R1 checks the top entry (Entry 10): `permit source IP 192.168.1.1`.
    - The source of the ping is 192.168.1.1 (PC1), so this is a **match**.
  - **Action**: The packet is **permitted** and forwarded to PC3.
  - **Note**: PC3 will be able to reply because there is no ACL blocking the return path from PC3 to PC1.

- **Scenario 2: PC2 tries to ping PC3**



  - R1 receives the ping on G0/1 but does not check the ACL.
  - R1 decides to forward the packet out of G0/2, so it checks ACL 1.
  - It checks the entries in order:
    1. **Entry 10** (`permit 192.168.1.1/32`): No match (source is 192.168.1.2).
    2. **Entry 20** (`deny 192.168.1.0/24`): **Match** (192.168.1.2 is in this subnet).
  - **Action**: R1 takes the action for the matching entry, which is **deny**. It will not forward the ping to PC3.

Standard Named ACLs

- **Definition**: Standard named ACLs are still standard ACLs (they match based on source IP address only), but they are identified by a name instead of a number (e.g., "BLOCK_BOB").
- **Configuration Mode**: You configure them in "standard named ACL config mode," which is different from numbered ACLs.

- **Entering Configuration Mode**:

  - Standard ACLs match traffic based only on the source IP address of the packet.
  - Named ACLs are identified with a name (ie. 'BLOCK_BOB')
  - Standard named ACLs are configured by entering 'standard named ACL config mode', and then configuring each entry within that config mode.

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

  - Use `IP ACCESS-LIST STANDARD [name]` (note the `IP` prefix, unlike the `ACCESS-LIST` command for numbered ACLs).

- **Configuring Entries**:
  - After entering the mode, you configure `deny` and `permit` entries.
  - You can manually specify an entry number before the command to control the order.
  - If you do not specify a number, entries are automatically numbered 10, 20, 30, etc.

Standard Named ACL Configuration Example

- **Configuration Steps**:

```
R1(config)#ip access-list standard BLOCK_BOB
R1(config-std-nacl)#5 deny 1.1.1.1
R1(config-std-nacl)#10 permit any
R1(config-std-nacl)#remark ## CONFIGURED NOV 21 2020 ##
R1(config-std-nacl)#interface g0/0
R1(config-if)#ip access-group BLOCK_BOB in
```

1. Create the ACL and enter config mode: `ip access-list standard BLOCK_BOB`
2. Configure the deny entry with a specific sequence number: `5 deny 1.1.1.1 0.0.0.0`
3. Configure the permit entry: `10 permit any`
4. Configure a remark (optional): `remark # Block Bob's IP address`

- **Applying the ACL**:
  - **Command**: `IP ACCESS-GROUP [name] {IN | OUT}`
  - **Mode**: Interface configuration mode.
  - **Note**: The syntax is exactly the same as for numbered ACLs, except you use the ACL name instead of the number.

Verification of Standard Named ACLs

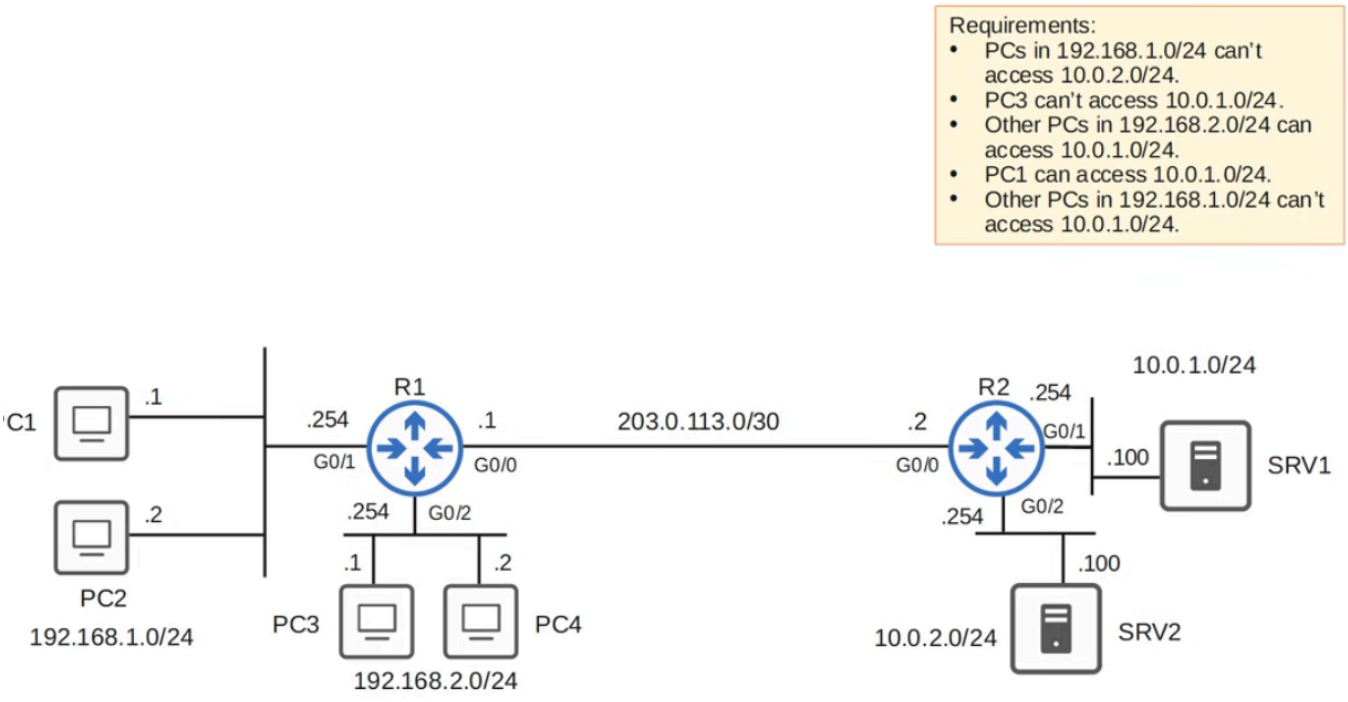
```
R1#show access-lists
Standard IP access list BLOCK_BOB
    5 deny   1.1.1.1
    10 permit any
R1#
R1#show running-config | section access-list
ip access-list standard BLOCK_BOB
 deny   1.1.1.1
 permit any
 remark ## CONFIGURED NOV 21 2020 ##
R1#
```

- **Command**: `SHOW ACCESS-LISTS`
  - The ACL is shown with the entry numbers manually configured (e.g., 5 and 10).

- **Command**: `SHOW RUNNING-CONFIG | SECTION ACCESS-LIST`
  - This command displays just the ACL section of the running config.
  - **Important Distinction**: If you use `INCLUDE ACCESS-LIST` (as we did for numbered ACLs), it will display the ACL's name line, but it will **not** display the individual entries (deny/permit). This is because those specific lines of config do not contain the word "ACCESS-LIST".
  - **Note**: When using `SECTION`, you can view the whole ACL including the remark, but interestingly the entry sequence numbers are not displayed in the running configuration output.

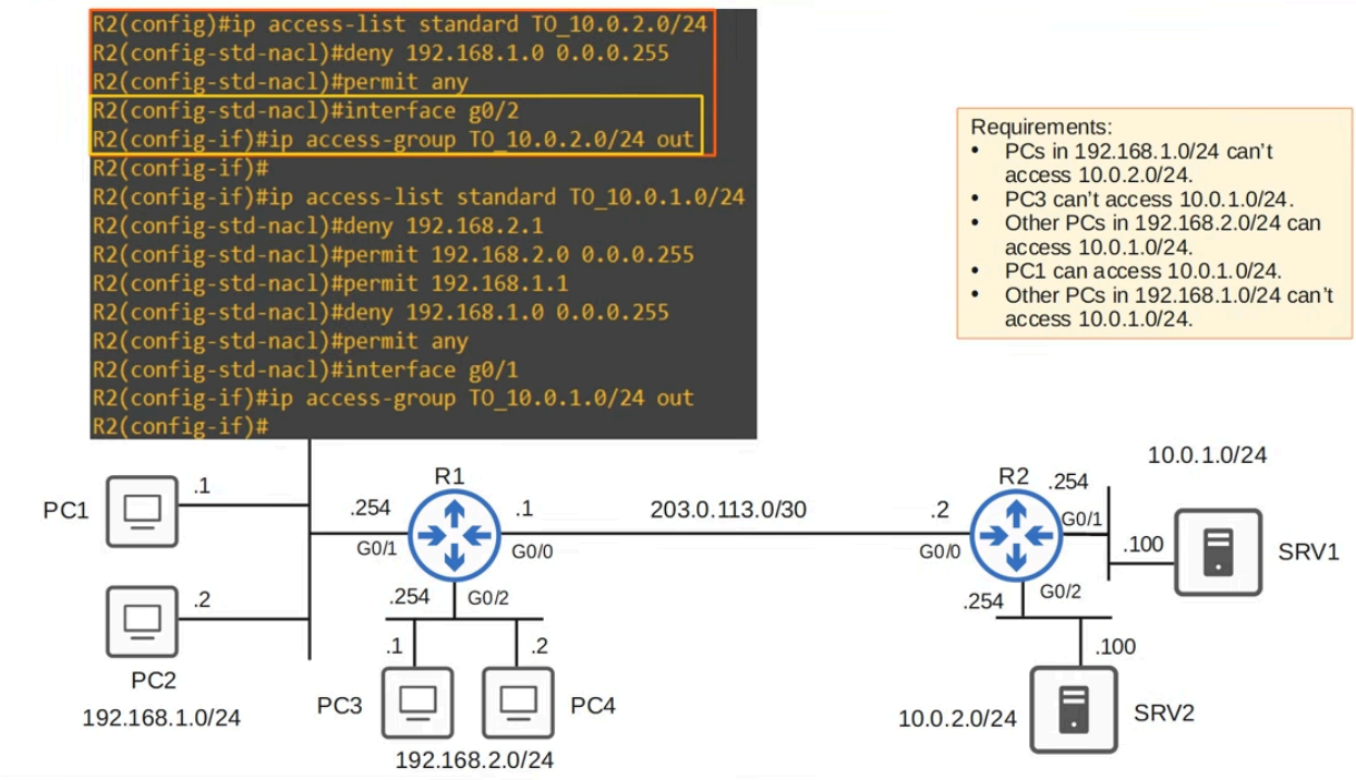
Standard Named ACL Configuration Example (on R2)

Requirements



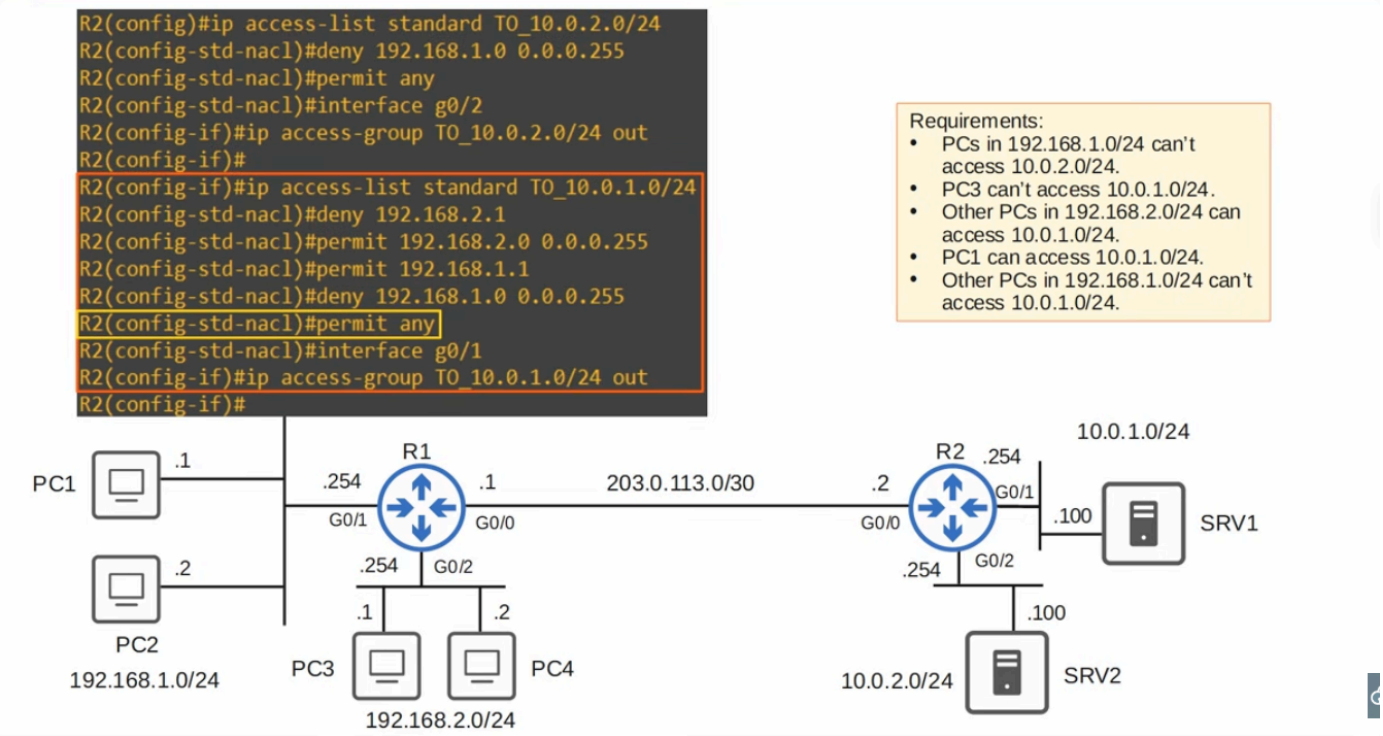
- PCs in 192.168.1.0/24 cannot access 10.0.2.0/24.
- PC3 cannot access 10.0.1.0/24, but other PCs in 192.168.2.0/24 can.
- PC1 can access 10.0.1.0/24, but other PCs in 192.168.1.0/24 cannot.

Configuration Steps



- Two ACLs are required to fulfill these requirements.
- **ACL 1:** `TO_10.0.2.0/24`
  - **Purpose**: Controls access to the 10.0.2.0/24 network.
  - **Entries**:
    1. Deny 192.168.1.0/24.
    2. Permit all other traffic (permit any).
  - **Application**: Applied **outbound** on R2's G0/2 interface.
  - **Effect**: PC1 and PC2 are blocked from accessing SRV2, but PC3 and PC4 can access it.

- **ACL 2:** `TO_10.0.1.0/24`



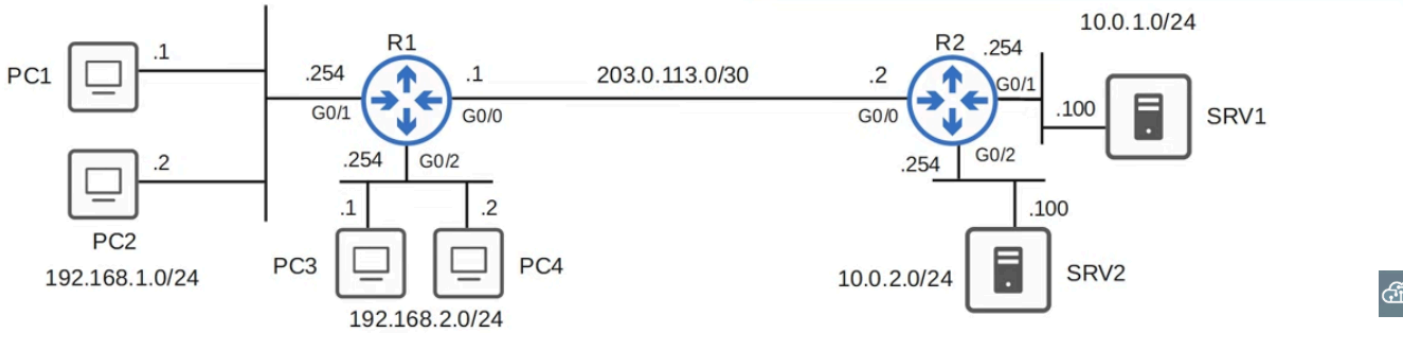
  - **Purpose**: Controls access to the 10.0.1.0/24 network.
  - **Entries**:
    1. Deny PC3 (192.168.2.1).
    2. Permit the rest of the 192.168.2.0/24 network.
    3. Permit PC1 (192.168.1.1).
    4. Deny the rest of the 192.168.1.0/24 network.
    5. Permit all other traffic (permit any).
  - **Application**: Applied **outbound** on R2's G0/1 interface.
  - **Effect**: Only PC1 and PCs in 192.168.2.0/24 (except PC3) can access SRV1.
- **Note**: ACL configuration can be flexible, so other configurations that fulfill the requirements would also be correct.

Router Optimization of ACEs

```
R2#show ip access-lists
Standard IP access list TO_10.0.1.0/24
    30 permit 192.168.1.1
    10 deny   192.168.2.1
    20 permit 192.168.2.0, wildcard bits 0.0.0.255
    40 deny   192.168.1.0, wildcard bits 0.0.0.255
    50 permit any
Standard IP access list TO_10.0.2.0/24
    10 deny   192.168.1.0, wildcard bits 0.0.0.255
    20 permit any
R2#
```

```
R2(config-if)#ip access-list standard TO_10.0.1.0/24
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
```

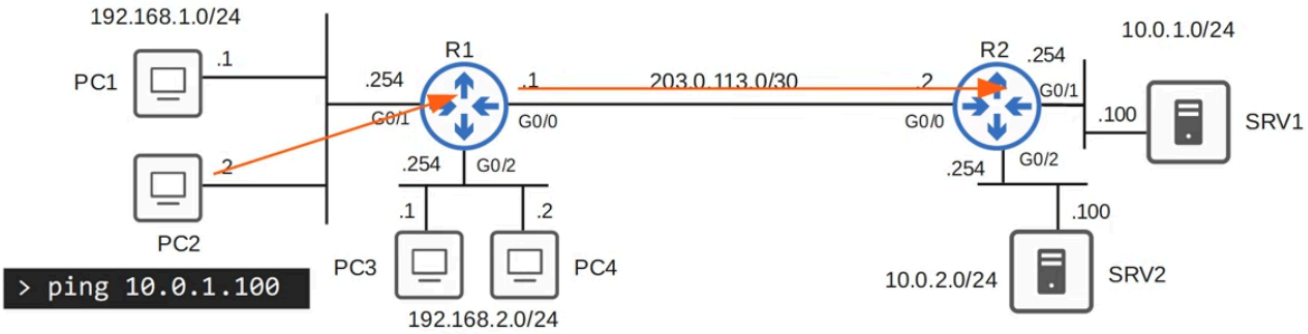
- The router may re-order the /32 entries.
- This improves the efficiency of processing the ACL.
- It **does not** change the effect of the ACL.
- This applies to both standard named and standard numbered ACLs.



- **Verification Output**: When using `SHOW IP ACCESS-LISTS`, notice the order of sequence numbers in the `TO_10.0.1.0/24` ACL (30, 10, 20, 40, 50).
  - The sequence numbers match the order the professor configured them, but the actual order displayed is different.
- **Explanation**: This is an advanced internal operation of Cisco IOS, not required for the CCNA exam.
  - The router may re-order `/32` entries (entries matching a single specific host) to improve processing efficiency.
  - This re-ordering does not change the overall effect or outcome of the ACL.
- **Scope**: This applies to both standard named and standard numbered ACLs.
- **Packet Tracer Note**: In Packet Tracer, this re-ordering does not occur; entries are displayed in the order of their sequence numbers.

ACL Processing Example (PC2 to SRV1)

```
R2#show ip access-lists
Standard IP access list TO_10.0.1.0/24
    30 permit 192.168.1.1
    10 deny   192.168.2.1
    20 permit 192.168.2.0, wildcard bits 0.0.0.255
    40 deny   192.168.1.0, wildcard bits 0.0.0.255
    50 permit any
Standard IP access list TO_10.0.2.0/24
    10 deny   192.168.1.0, wildcard bits 0.0.0.255
    20 permit any
R2#
```



- **Scenario**: PC2 wants to access SRV1, so it sends a ping to test connectivity.
- **Processing**:
  - The ping reaches R2, which is directly connected to SRV1's network.
  - However, the `TO_10.0.1.0/24` ACL is applied **outbound** on G0/1, so R2 will check the packet against that ACL before forwarding it.
  - The source is 192.168.1.2.
  - R2 checks the ACL entries:
    1. **Entry 30** (Deny 192.168.2.1): No match.
    2. **Entry 10** (Permit 192.168.2.0/24): No match.
    3. **Entry 20** (Permit 192.168.1.1): No match.
    4. **Entry 40** (Deny 192.168.1.0/24): **Match** (192.168.1.2 is in this subnet).
- **Result**: The packet is denied and not forwarded to SRV1.

Review and Summary

- **What are ACLs?**: They are used to identify and control traffic in the network.
- **ACL Logic**:
  - Entries in an ACL are processed from top to bottom.
  - Once a matching entry is found, the action is taken, and the remaining entries are not processed.
- **ACL Types**: There are two main types: **Standard ACLs** and **Extended ACLs**.
  - Each type can be configured as **Numbered** or **Named** ACLs.
- **Standard ACLs**:
  - These are simple and match traffic based only on the source IP address of the packet.
- **Configuration Methods**:
  - **Standard Numbered ACLs**: Configured in global config mode using the `ACCESS-LIST` command.
  - **Standard Named ACLs**: Use the `IP ACCESS-LIST` command to enter standard named ACL config mode, and then configure the entries.