# IPv6 Part 3: Addressing, NDP, SLAAC, and Static Routing

- This video wraps up IPv6 studies for the CCNA
- IPv6 will continue to be referenced and used in future videos (labs and examples)
- Many CCNA students don't feel confident about IPv6 due to limited exposure — most course content outside a few IPv6-specific lessons focuses on IPv4
- The focus of this video is **CCNA exam topic 3.3**: applying static routing concepts from IPv4 to IPv6

## Lecture Outline

- Correction about previous IPv6 videos (a mistake even Cisco makes)
- IPv6 header (brief overview)
- Neighbor Discovery Protocol (NDP) — replaces ARP in IPv6; IPv6 does not use ARP
- SLAAC (Stateless Address Auto-configuration)
- IPv6 static routing — the main topic; builds on IPv4 static routing concepts already learned

---

## IPv6 Address Representation Correction

- The correction is about how IPv6 addresses are written (represented)
- An **RFC (Request for Comments)** is a publication from the **ISOC (Internet Society)** and associated organizations like the **IETF (Internet Engineering Task Force)**
- RFCs are the official documents of Internet specifications, protocols, procedures, etc.
- Example: For in-depth study of OSPF, there are many RFCs that document how it works
- **RFC 5952** is titled "A Recommendation for IPv6 Address Text Representation"

- Before RFC 5952, IPv6 address representation was more flexible:
    - You could remove leading 0s from the quartets of the address, or leave them
    - You could replace all-0 quartets with a double colon, or choose not to
    - You could use upper-case or lower-case hexadecimal A, B, C, D, E, F
- RFC 5952 suggests standardizing IPv6 address representation so everyone writes them the same way

---

## RFC 5952 Rules for IPv6 Address Representation

- Leading 0s **MUST** be removed
    - Example: An IPv6 address must be represented with all leading 0s removed from each quartet
- The double colon **MUST** be used to shorten the **longest** string of all-0 quartets; if there is only a single all-0 quartet, do not use the double colon
    - Example: An address with three all-0 quartets on the left and two on the right must be shortened using the double colon on the left (the longer string)
- If there are two **equal-length** choices for the double colon, use the double colon to shorten the **leftmost** one
    - Example: An address with two quartets of 0s on the left and two on the right must be shortened using the double colon on the left
- Hexadecimal characters a, b, c, d, e, and f **MUST** be written using **lower-case**, not upper-case
    - Note: Cisco routers sometimes display upper-case hex characters, which is technically incorrect per RFC 5952 — but because even Cisco's devices don't follow this rule, it's not a big deal if you write them in upper-case

---

# IPv6 Header Overview

<table>
<caption>Fixed header format</caption>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><th>0</th><th>0</th><td colspan="4">Version</td><td colspan="8">Traffic Class</td><td colspan="20">Flow Label</td></tr>
<tr><th>4</th><th>32</th><td colspan="16">Payload Length</td><td colspan="8">Next Header</td><td colspan="8">Hop Limit</td></tr>
<tr><th>8</th><th>64</th><td colspan="32" rowspan="4">Source Address</td></tr>
<tr><th>12</th><th>96</th></tr>
<tr><th>16</th><th>128</th></tr>
<tr><th>20</th><th>160</th></tr>
<tr><th>24</th><th>192</th><td colspan="32" rowspan="4">Destination Address</td></tr>
<tr><th>28</th><th>224</th></tr>
<tr><th>32</th><th>256</th></tr>
<tr><th>36</th><th>288</th></tr>
</table>

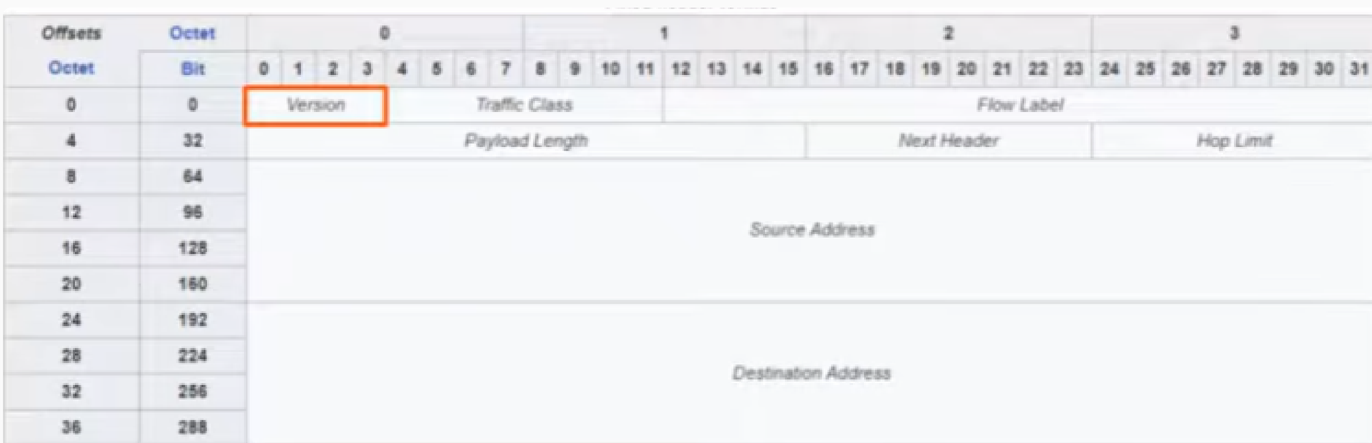
- The IPv6 header is much simpler than the IPv4 header
- The IPv6 header has a **fixed size of 40 bytes**

<table>
<caption>IPv4 header format</caption>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><th>0</th><th>0</th><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="6">DSCP</td><td colspan="2">ECN</td><td colspan="16">Total Length</td></tr>
<tr><th>4</th><th>32</th><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><th>8</th><th>64</th><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><th>12</th><th>96</th><td colspan="32">Source IP Address</td></tr>
<tr><th>16</th><th>128</th><td colspan="32">Destination IP Address</td></tr>
<tr><th>20</th><th>160</th><td colspan="32" rowspan="4">Options (if IHL > 5)</td></tr>
<tr><th>24</th><th>192</th></tr>
<tr><th>28</th><th>224</th></tr>
<tr><th>32</th><th>256</th></tr>
</table>

  - In contrast, the IPv4 header has a variable header length from 20 to 60 bytes
- Because the header is always 40 bytes, there is no 'header length' field like in IPv4
- Instead, there is a 'payload length' field indicating the length of the encapsulated Layer 4 segment
- The processing of the IPv6 header is much easier for routers, so performance is generally improved

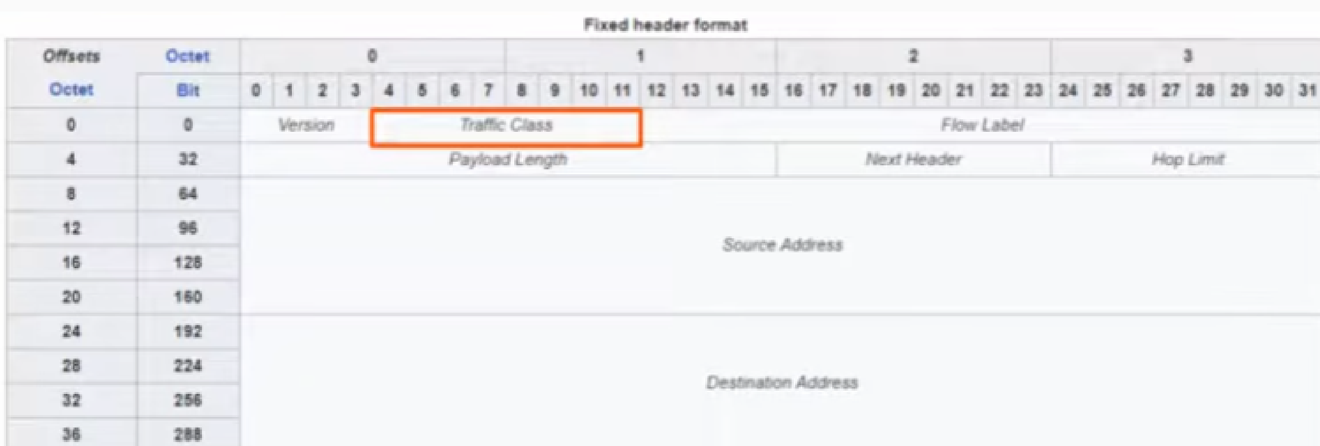
# IPv6 Header Fields

- **Version**




  - 4 bits in length
  - Indicates the version of IP used
  - Always set to 6 (binary 0110) for IPv6

- **Traffic Class**




  - 8 bits in length
  - Used for QoS (Quality of Service) to indicate high-priority traffic
  - Example: IP phone traffic, live video calls, etc. will have a traffic class value giving them priority over other traffic
  - QoS will be covered later in the course

- **Flow Label**

<table>
<tr><th colspan="34">Fixed header format</th></tr>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Version</td><td colspan="8">Traffic Class</td><td colspan="20">Flow Label</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Payload Length</td><td colspan="8">Next Header</td><td colspan="8">Hop Limit</td></tr>
<tr><td>8</td><td>64</td><td colspan="32" rowspan="4">Source Address</td></tr>
<tr><td>12</td><td>96</td></tr>
<tr><td>16</td><td>128</td></tr>
<tr><td>20</td><td>160</td></tr>
<tr><td>24</td><td>192</td><td colspan="32" rowspan="4">Destination Address</td></tr>
<tr><td>28</td><td>224</td></tr>
<tr><td>32</td><td>256</td></tr>
<tr><td>36</td><td>288</td></tr>
</table>

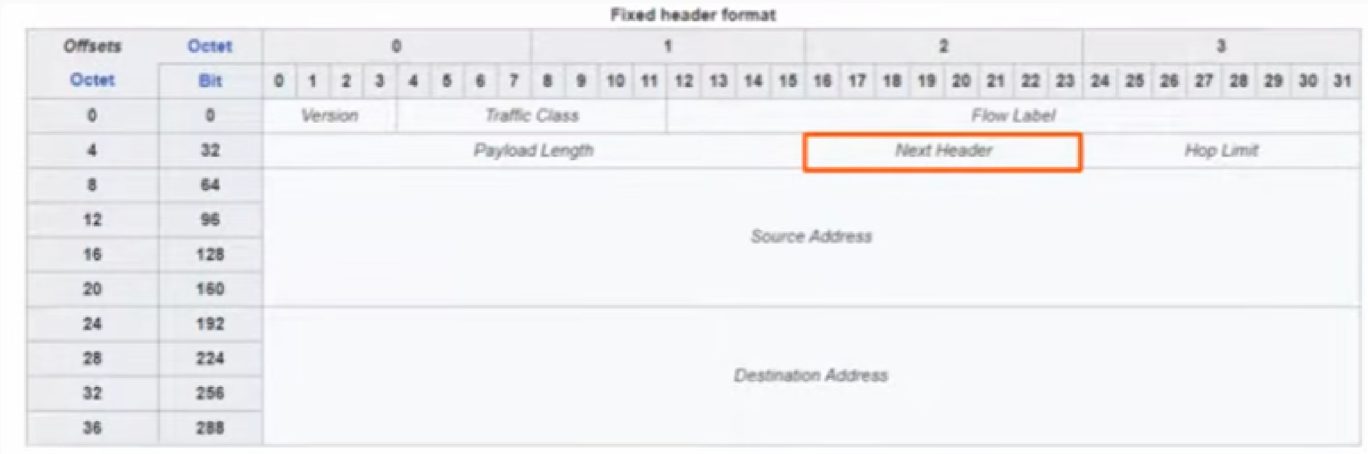
  - 20 bits in length
  - Used to identify specific traffic flows
  - Flows are communications between a specific source and destination
  - Example: The interaction between a server and a client downloading a file

- **Payload Length**

<table>
<tr><th colspan="34">Fixed header format</th></tr>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Version</td><td colspan="8">Traffic Class</td><td colspan="20">Flow Label</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Payload Length</td><td colspan="8">Next Header</td><td colspan="8">Hop Limit</td></tr>
<tr><td>8</td><td>64</td><td colspan="32" rowspan="4">Source Address</td></tr>
<tr><td>12</td><td>96</td></tr>
<tr><td>16</td><td>128</td></tr>
<tr><td>20</td><td>160</td></tr>
<tr><td>24</td><td>192</td><td colspan="32" rowspan="4">Destination Address</td></tr>
<tr><td>28</td><td>224</td></tr>
<tr><td>32</td><td>256</td></tr>
<tr><td>36</td><td>288</td></tr>
</table>

  - 16 bits in length
  - Indicates the length of the payload (the encapsulated Layer 4 segment) in bytes
  - Example: A value of 1024 means the encapsulated segment is 1024 bytes in length
  - The IPv6 header length itself is not included because it's always a fixed 40 bytes

- **Next Header**

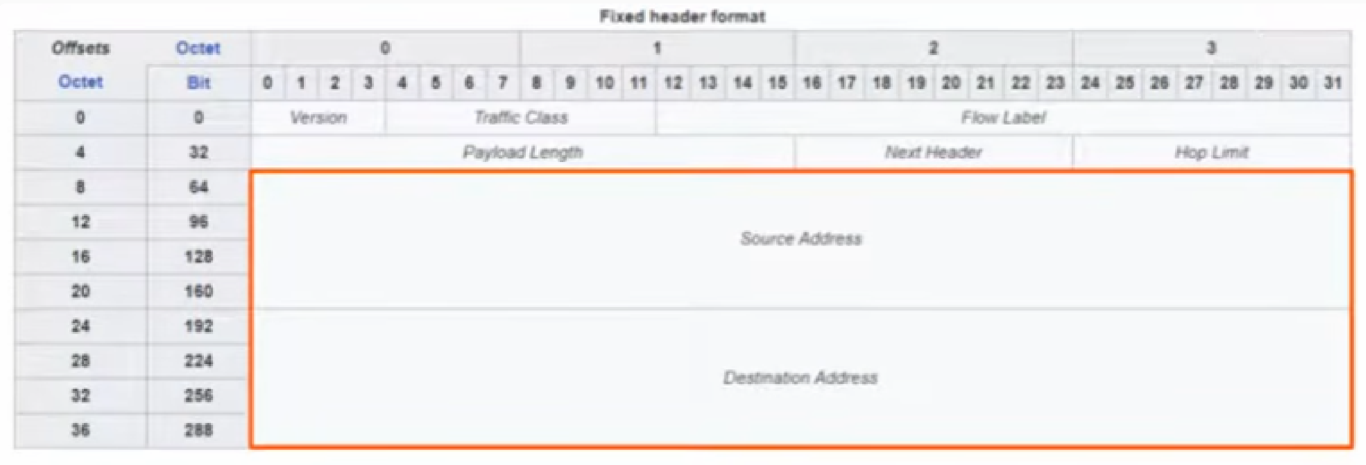



  - 8 bits in length
  - Indicates the type of the next header (the encapsulated segment's header), for example TCP or UDP
  - Has the same function as the 'protocol' field in the IPv4 header

- **Hop Limit**


  - 8 bits in length
  - The value is decremented by one by each router that forwards the packet
  - If the value reaches 0, the packet is discarded
  - Has the same function as the TTL field in the IPv4 header

- **Source Address / Destination Address**




  - **Source Address:** 128 bits in length
  - **Source Address:** Contains the IPv6 address of the packet's source
  - **Destination Address:** 128 bits in length
  - **Destination Address:** Contains the IPv6 address of the packet's intended destination

- Note: Direct questions about the IPv6 header are unlikely on the CCNA exam, but it is good foundational knowledge

---

## Solicited-Node Multicast Address

- An IPv6 solicited-node multicast address is calculated from a unicast address

`ff02:0000:0000:0000:0000:0001:ff` + `Last 6 hex digits of unicast address`

- The address begins with a fixed prefix: **ff02::1:ff**
- The last 6 hex digits of the solicited-node multicast address are taken from the last 6 hex digits of the unicast address

Example:


- **Problem 1**
  - Original IPv6 address: `2001:0db8:0000:0001:0f2a:4fff:fea3:00b1`
  - Take the last 6 hexadecimal digits: `a3:00b1`
  - Remove the leading zeros: `a3:b1`
  - Add them to the prefix `ff02::1:ff`
  - Final answer: `ff02::1:ffa3:b1`

  **Problem 2**
  - Original IPv6 address: `2001:0db8:0000:0001:0489:4eda:073a:12b8`
  - Take the last 6 hexadecimal digits: `3a:12b8`
  - Add them to the prefix `ff02::1:ff`
  - Final answer: `ff02::1:ff3a:12b8`

  **Summary**
  - `2001:0db8:0000:0001:0f2a:4fff:fea3:00b1` → `ff02::1:ffa3:b1`
  - `2001:0db8:0000:0001:0489:4eda:073a:12b8` → `ff02::1:ff3a:12b8`
  - Note: The main trick is to ignore everything except the last 6 hex digits, then attach them to `ff02::1:ff`

---

## Solicited-Node Multicast Address

- An IPv6 solicited-node multicast address is calculated from a unicast address
- The address begins with a fixed prefix: **ff02::1:ff**
- The last 6 hex digits of the solicited-node multicast address are taken from the last 6 hex digits of the unicast address
- Example:

```
R1#sh ipv6 int g0/0
GigabitEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::EF8:22FF:FE36:8500
  No Virtual link-local address(es):
  Global unicast address(es):
    2001:DB8::EF8:22FF:FE36:8500, subnet is 2001:DB8::/64 [EUI]
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF36:8500
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND advertised reachable time is 0 (unspecified)
  ND advertised retransmit interval is 0 (unspecified)
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  ND advertised default router preference is Medium
```

  - Unicast IPv6 address: 2001:db8::**36:8500**
  - Last 6 hex digits: **36:8500**
  - Solicited-node multicast address: **ff02::1:ff36:8500**
- Routers join the FF02::1 and FF02::2 multicast groups by default
- Example: R1 also joined FF02::1:FF36:8500, a solicited-node multicast address where the last six digits (36:8500) match its global unicast address

## Neighbor Discovery Protocol (NDP)

- NDP is a protocol used with IPv6
- Not directly listed on the exam topics list, but is an essential part of IPv6 routing (equivalent to ARP's role in IPv4)
- Has various functions; one primary function is to replace ARP, which is no longer used in IPv6

## NDP's ARP-like Function

- Uses ICMPv6 and solicited-node multicast addresses to learn the MAC address of other hosts
- ARP in IPv4 uses broadcast messages for ARP requests; solicited-node multicast messages are much more efficient because they are addressed to a specific host, unlike a broadcast which is for all hosts,

## NDP Message Types (ARP Replacement)

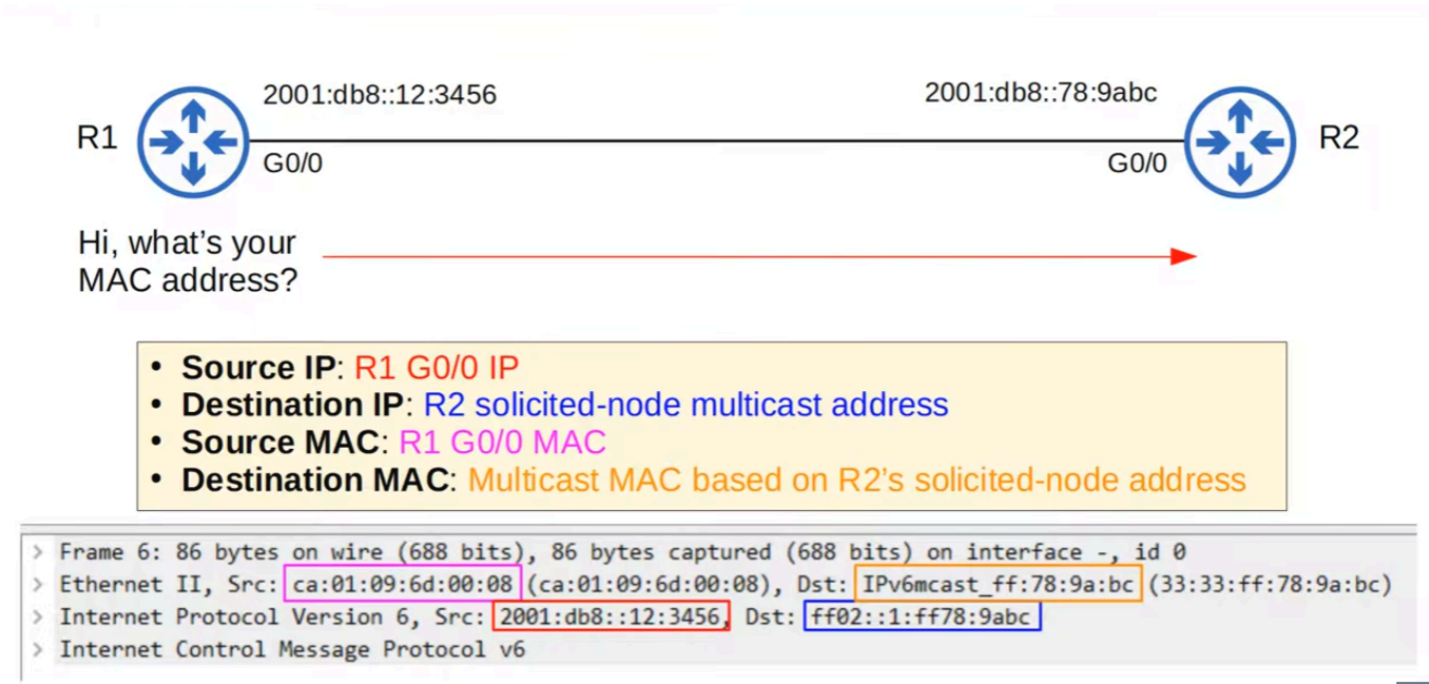
- **Neighbor Solicitation (NS)**
  - The NDP equivalent of an ARP request
  - ICMPv6 Type 135
- **Neighbor Advertisement (NA)**
  - The NDP equivalent of an ARP reply
  - ICMPv6 Type 136

---

## Neighbor Solicitation Message Function

- **Scenario**: R1 pings R2's address `2001:db8::78:9abc`
- R1 needs R2's MAC address to encapsulate the packet in an Ethernet frame
- R1 sends a **Neighbor Solicitation (NS)** message to ask for R2's MAC address
- R1 is basically saying "Hi, what's your MAC address?"

## Neighbor Solicitation vs ARP Request



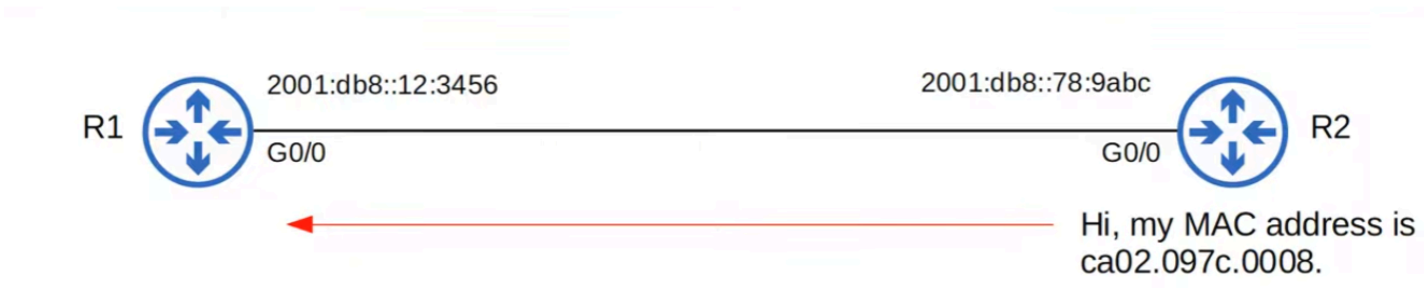

- **Source IP**: R1's IP address — same as ARP
- **Destination IP**: R2's solicited-node multicast address
  - R1 knows R2's unicast address because it was typed in the PING command
  - R1 automatically calculates the solicited-node multicast address from that unicast address
- **Source MAC**: R1's G0/0 MAC address — same as ARP
- **Destination MAC**: A multicast MAC address based on R2's solicited-node multicast address
  - Example: In Wireshark, displayed as `IPv6mcast_ff:78:9a:bc` with the real multicast MAC address shown in brackets to the right
  - Note: Multicast MAC addresses don't need to be studied for the CCNA

## Key Difference

- **ARP request**: Uses a **broadcast** destination (all hosts on the local network)
- **Neighbor Solicitation**: Uses a **multicast** destination (more efficient, targeted to a specific host)

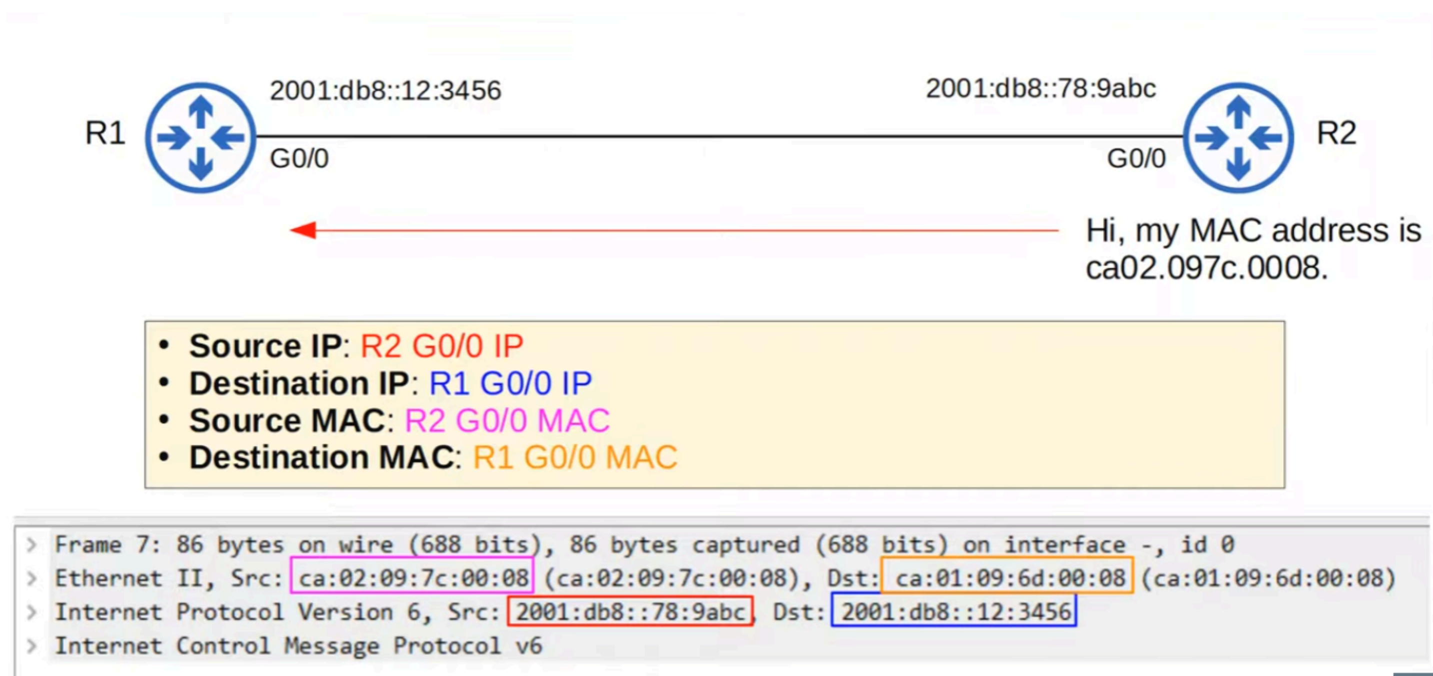
---

## Neighbor Advertisement Message Function




- R2 received the neighbor solicitation message from R1 asking for its MAC address
- R2 sends a **Neighbor Advertisement (NA)** message to tell R1 its MAC address

## Neighbor Advertisement Addresses



- **Source IP**: R2's G0/0 IP address
- **Destination IP**: R1's G0/0 IP address
    - R2 knows R1's IP address because it was the source address of the solicitation message
- **Source MAC**: R2's G0/0 MAC address
- **Destination MAC**: R1's MAC address
    - R2 knows R1's MAC address because R1 used it as the source for the solicitation message

---

# IPv6 Neighbor Table

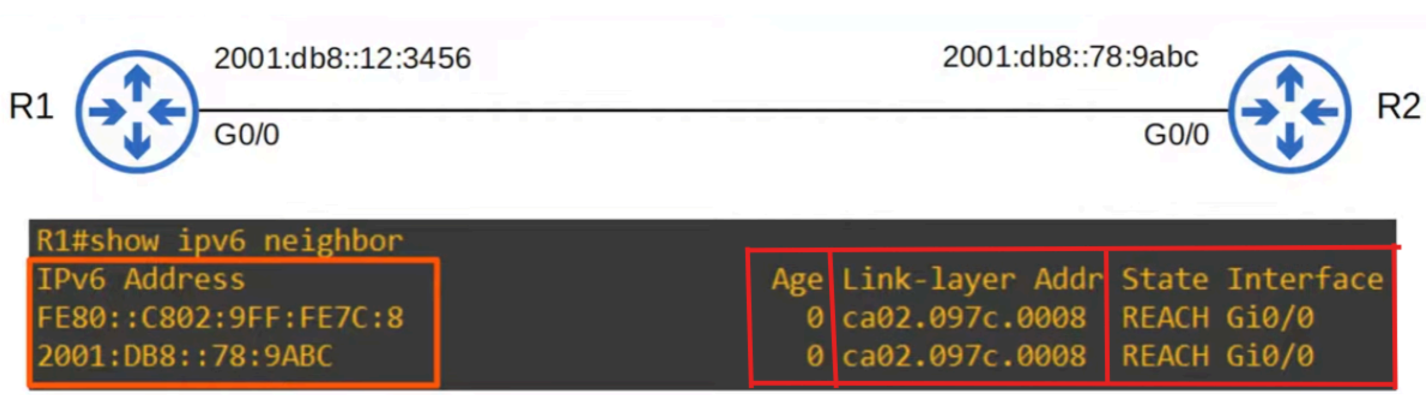
- IPv6 doesn't use ARP, so there is no ARP table; instead, devices keep an **IPv6 neighbor table**
- All devices running IPv6 will use NDP and keep a neighbor table (not just Cisco routers)

# Viewing the Neighbor Table

- Command to view the neighbor table:

```
R1# show ipv6 neighbor
```

# Neighbor Table Output



- **IPv6 address column**: Shows the IPv6 addresses of neighbors
  - Example: R1 has an entry for both R2's global unicast address and its link-local address (the link-local address was learned automatically)
- **Age column**: Indicates how long ago traffic was received from these addresses, in minutes
- **Link-layer address column**: Shows the neighbor's MAC address
- **Interface column**: Shows the interface the entry was learned on
- **State column**: Indicates the neighbor's status
  - Example: The REACH state means the neighbor is reachable

---

## NDP Router Discovery Function

- Another function of NDP allows hosts to automatically discover routers on the local network
- Two messages are used for this process

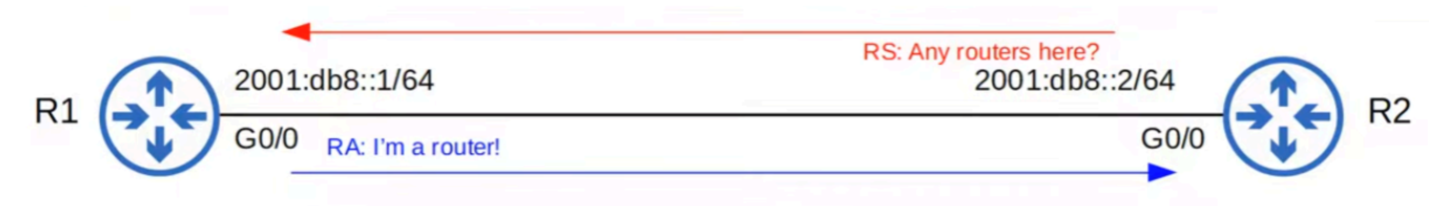
## Router Solicitation (RS)

- ICMPv6 Type 133
- Sent to multicast address **FF02::2** (the all-routers address)
- Asks routers on the local link (local network) to identify themselves
- Sent when an interface is enabled or when a host is connected to the network
- Note: Don't mix this up with Neighbor Solicitation

## Router Advertisement (RA)

- ICMPv6 Type 134
- Sent to multicast address **FF02::1** (the all-nodes address)
- Received by all hosts on the local link, not just routers
- The router announces its presence and provides information about the link (local network)
- Sent in response to RS messages; if a router receives an RS, it will send an RA
- Routers also send RAs periodically, even without receiving an RS

## Router Discovery Example




- Only routers send Router Advertisements, but all IPv6 hosts send Router Solicitations when they come online
- Example: R2's G0/0 interface is enabled → R2 automatically sends an RS asking if there are any routers on the link → R1 replies with an RA, identifying itself
- Hosts can automatically learn their default gateway from RA messages

---

## SLAAC (Stateless Address Auto-configuration)

- **SLAAC** stands for Stateless Address Auto-configuration — another way to configure IPv6 addresses
- When using SLAAC, hosts use the RS and RA messages to learn the IPv6 prefix of the local link (e.g., `2001:db8::/64`)
  - The host then uses that prefix to automatically generate an IPv6 address

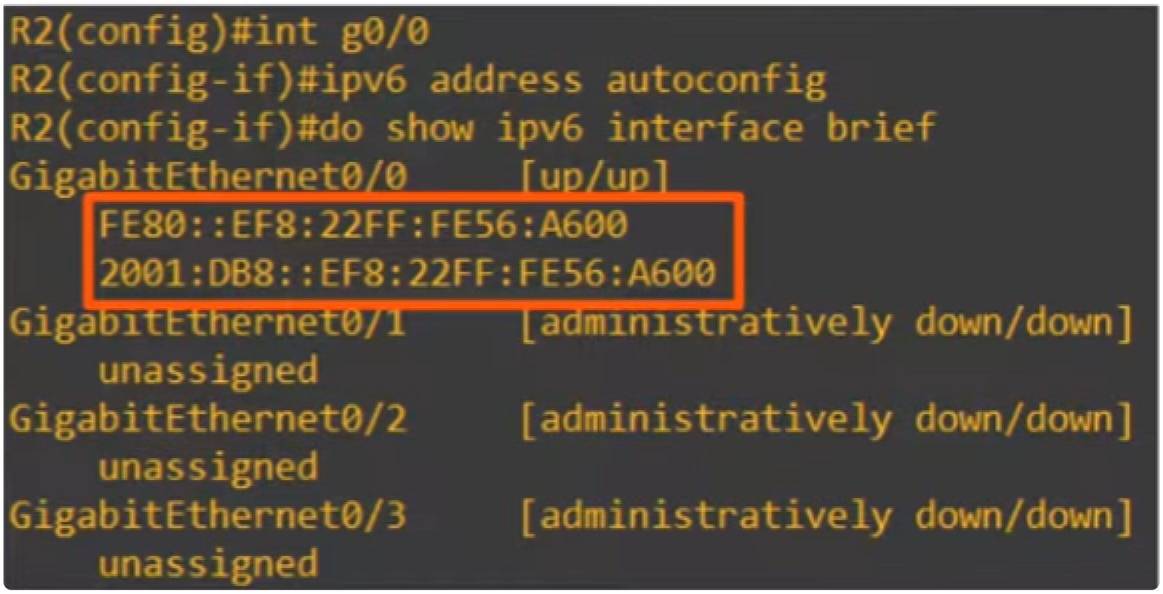
## SLAAC vs EUI-64

- When using the `ipv6 address eui-64` command, you must manually enter the prefix
- The SLAAC command, `ipv6 address autoconfig`, doesn't need the prefix
  - This is because NDP is used to learn the prefix via RS and RA messages
  - The device then uses EUI-64 to generate the interface ID, or it can be randomly generated depending on the device and maker

## SLAAC Configuration Example

- R1 is connected to R2; R1 has an IPv6 address configured, R2 does not
- The SLAAC command is applied on R2's G0/0 interface:

```
R2(config-if)# ipv6 address autoconfig
```



```
R2(config)#int g0/0
R2(config-if)#ipv6 address autoconfig
R2(config-if)#do show ipv6 interface brief
GigabitEthernet0/0      [up/up]
    FE80::EF8:22FF:FE56:A600
    2001:DB8::EF8:22FF:FE56:A600
GigabitEthernet0/1      [administratively down/down]
    unassigned
GigabitEthernet0/2      [administratively down/down]
    unassigned
GigabitEthernet0/3      [administratively down/down]
    unassigned
```

- R2 automatically generates a global unicast address and, as always, a link-local address
- Note: SLAAC is a standard function of IPv6, not unique to Cisco — end hosts like PCs can also use it (though they don't use Cisco IOS commands)

---

## Duplicate Address Detection (DAD)

- **DAD** is another function of NDP that allows hosts to check if other devices on the local link are using the same IPv6 address
- DAD is performed any time:
  - An IPv6-enabled interface initializes (e.g., via the `no shutdown` command)
  - An IPv6 address is configured on an interface (e.g., manual address or SLAAC address)
- DAD uses the same two messages already learned: **Neighbor Solicitation (NS)** and **Neighbor Advertisement (NA)**

## How DAD Works

- The host sends an NS to its own IPv6 address (its own solicited-node multicast address)
- If no reply is received → the address is unique
- If a reply (NA message) is received → another host on the network is already using that address

  > ```
  > *Oct 31 11:28:48.318: %IPV6_ND-4-DUPLICATE: Duplicate address 2001:DB8::1 on GigabitEthernet0/0
  > ```

- On a Cisco router, a duplicate address detection will generate a message indicating the conflict
- Note: What happens next depends on how the IP address was configured and the device maker — no deeper DAD knowledge is required for the CCNA

---

## IPv6 Static Routing Introduction

- You should be able to configure and verify the same kinds of static routes for IPv6 as you learned for IPv4
- IPv6 routing works the same as IPv4 routing: a packet arrives, the router looks up the destination in its routing table, and forwards based on the most specific match
- Although IPv6 routing works the same way, IPv4 and IPv6 are separate processes on the router, and the routing tables are separate
- View the IPv6 routing table with:

```
R1# show ipv6 route
```

## Enabling IPv6 Routing

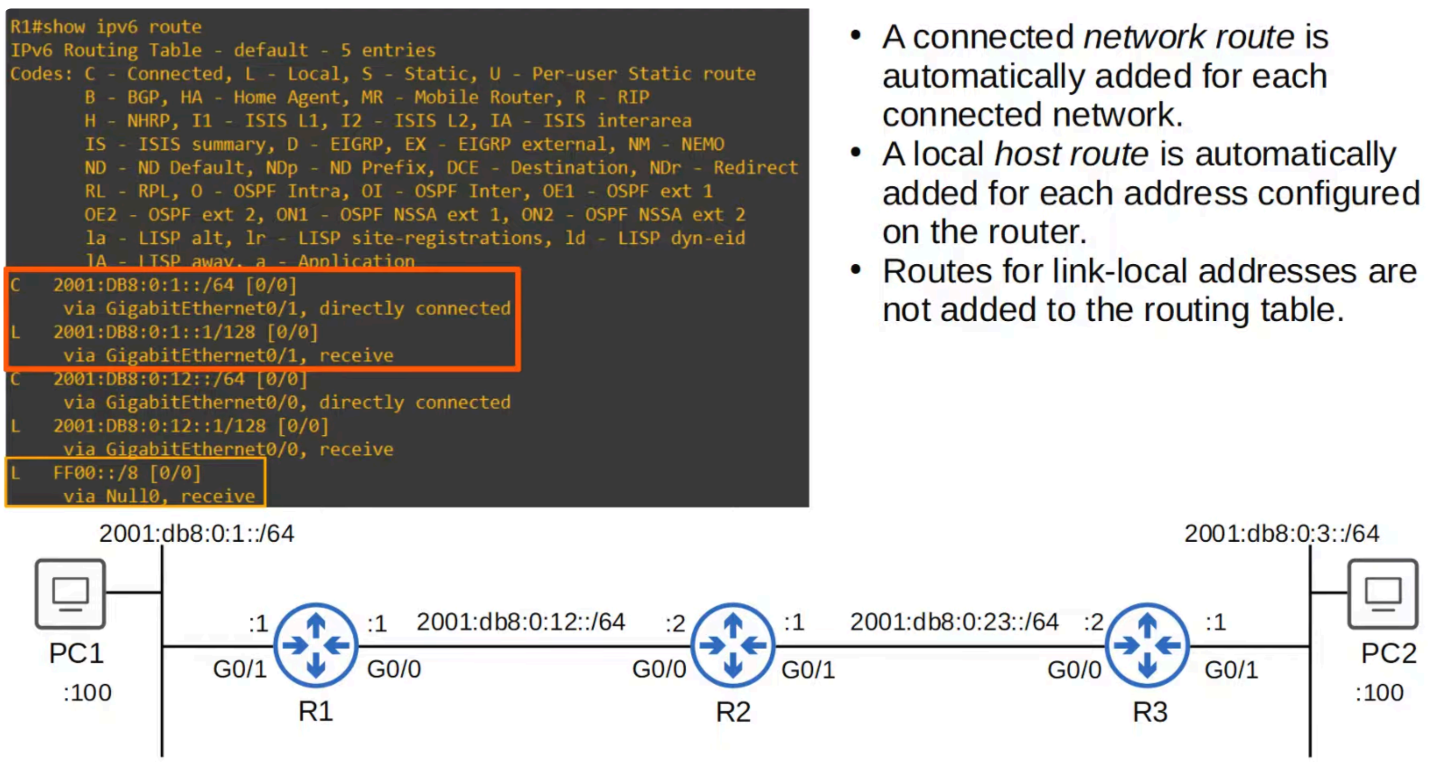
- IPv4 routing is enabled on Cisco routers by default
- **IPv6 routing is disabled by default** and must be enabled manually
- Command to enable IPv6 routing:

```
R1(config)# ipv6 unicast-routing
```

- Note: If IPv6 routing is disabled, the router can send and receive IPv6 traffic but will not **route** (forward) traffic between networks
- Note: If everything about your IPv6 configuration seems correct but the network isn't working, you likely forgot the `ipv6 unicast-routing` command

---

## IPv6 Routing Table (R1)




- A connected **network route** is automatically added for each connected network
  - A network route is a route to a network/subnet (not a single host)
- A local **host route** is automatically added for each address configured on the router
  - A host route is a route to a single specific host, using a /128 prefix length in IPv6 (or /32 in IPv4)

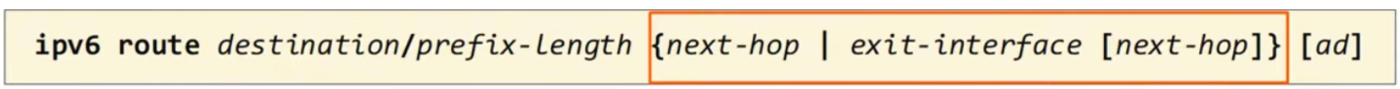
- Example: For R1's G0/1 interface, two routes are automatically added:
  - A /64 connected network route
  - A /128 local host route
- Note: Routes for **link-local addresses** are **not** added to the routing table (even though R1's G0/0 and G0/1 both have link-local addresses)

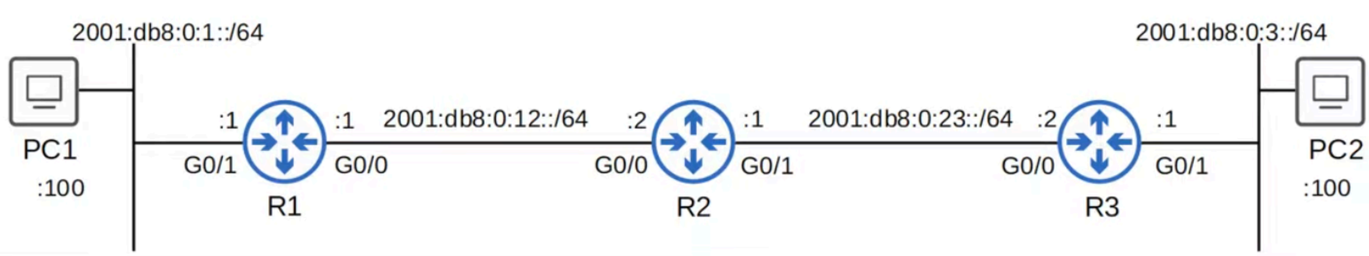
# FF00::/8 Route (Beyond CCNA Scope)

- There is an automatically configured route to FF00::/8 (the IPv6 multicast range) via Null0
- Null0 is an interface that discards traffic matching that route (discards multicast traffic)
- This route is beyond the scope of the CCNA

---

# IPv6 Static Route Command Format



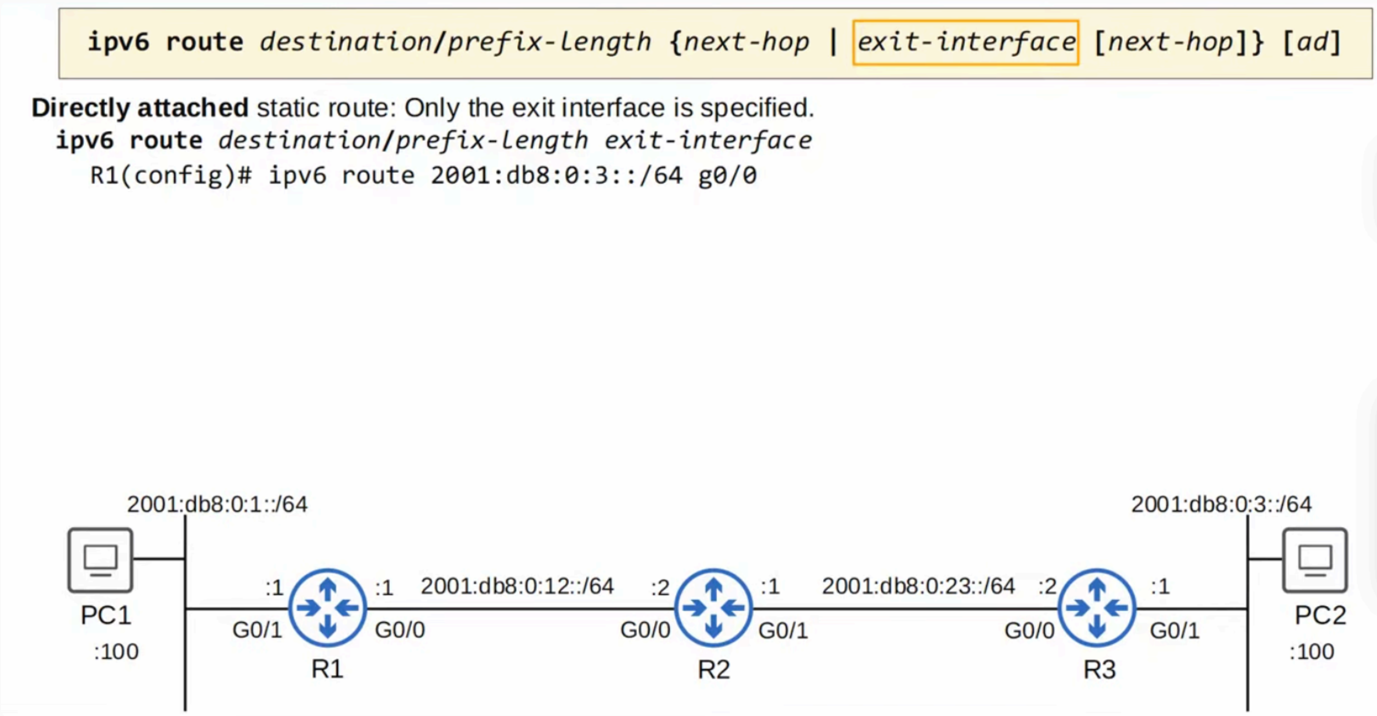




- The command begins with `ipv6 route`, followed by the destination and prefix length
- Curly brackets `{}` mean a **required choice**: you must enter either a next-hop address or an exit-interface (with an optional next-hop)

- Square brackets `[]` mean **optional**: the next-hop is optional if you specify an exit interface; AD (Administrative Distance) is also optional

# Types of Static Routes (IPv4 and IPv6)

- **Directly Attached Static Route**

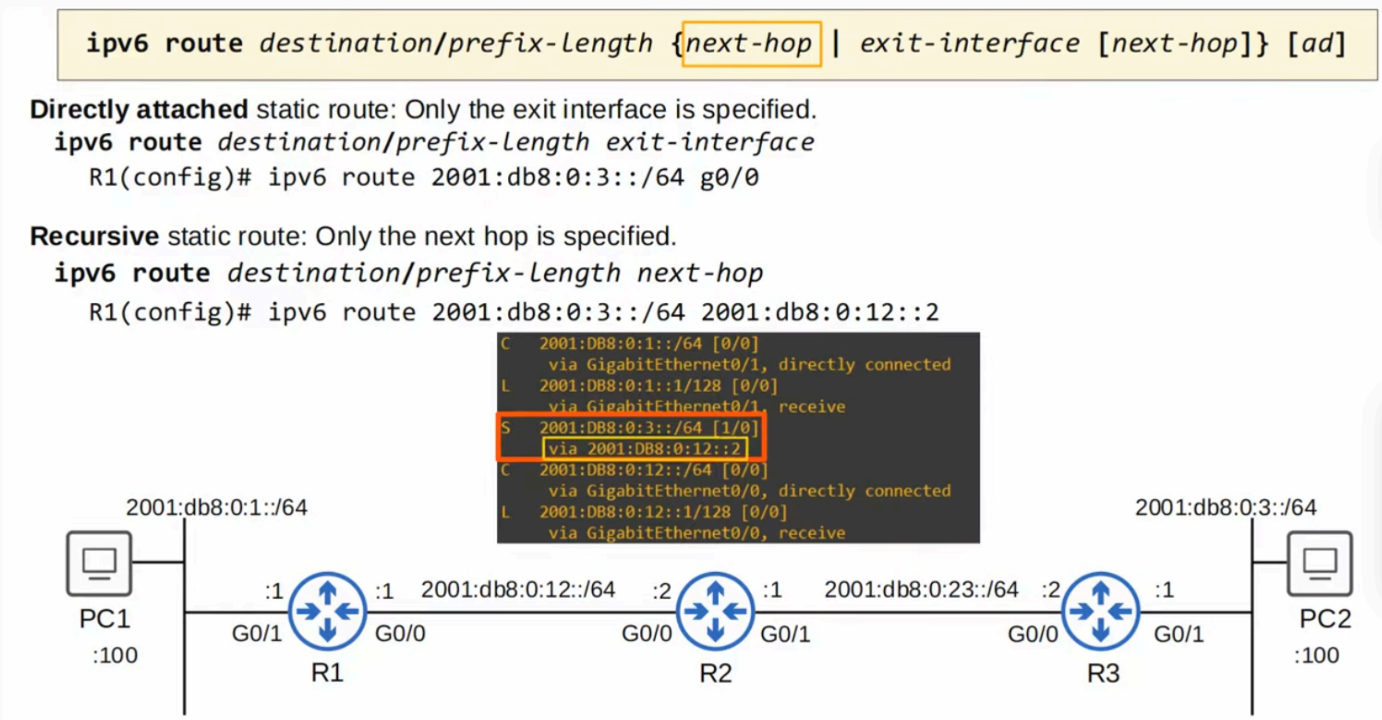



  - Specifies only the exit interface
  - Example:

```
R1(config)# ipv6 route 2001:db8:0:3::/64 g0/0
```

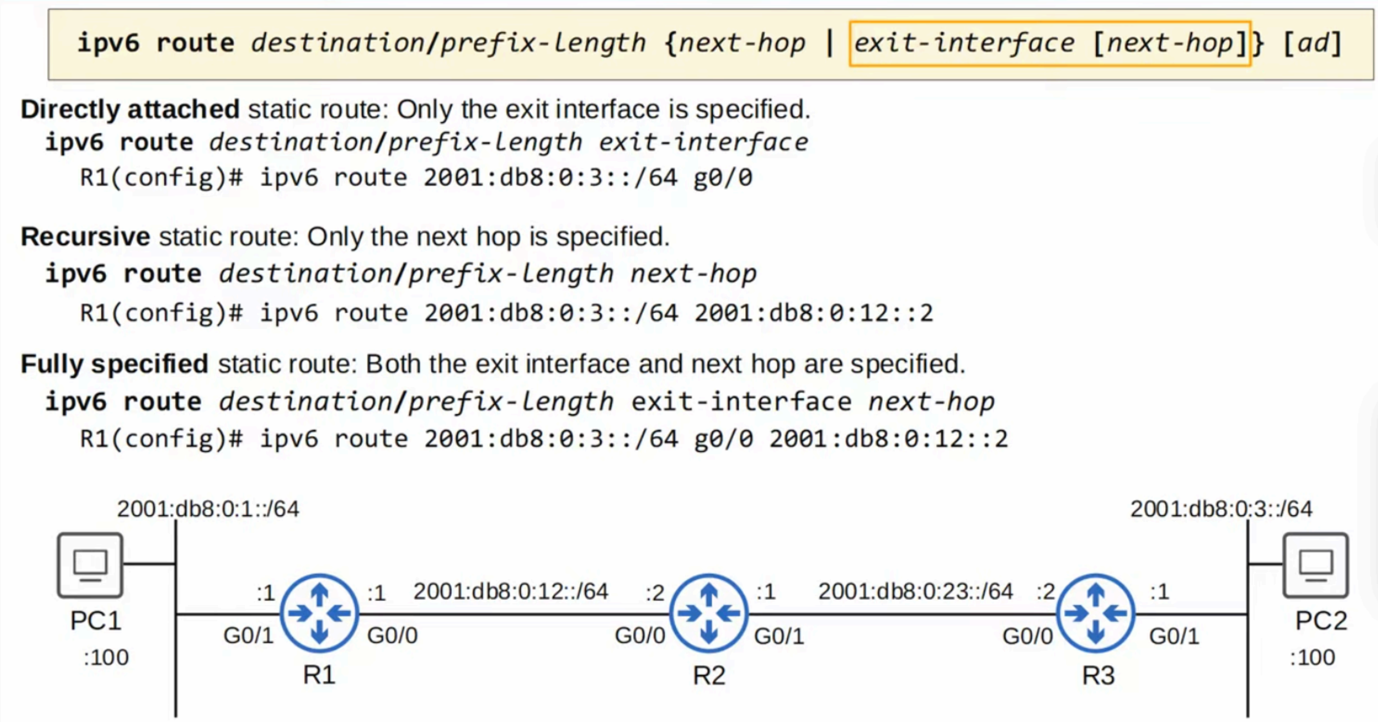
- **Recursive Static Route**
  - Specifies only the next-hop address
  - Example:

```
R1(config)# ipv6 route 2001:db8:0:3::/64 2001:db8:0:12::2
```




  - Called "recursive" because it requires a recursive lookup in the routing table: the router first looks up the destination, then must look up the next-hop address to determine the exit interface
  - Example: A packet destined for PC2 matches this route with next-hop 2001:db8:0:12::2. R1 then looks up that next-hop address, finds a connected route, and determines the exit interface is G0/0.

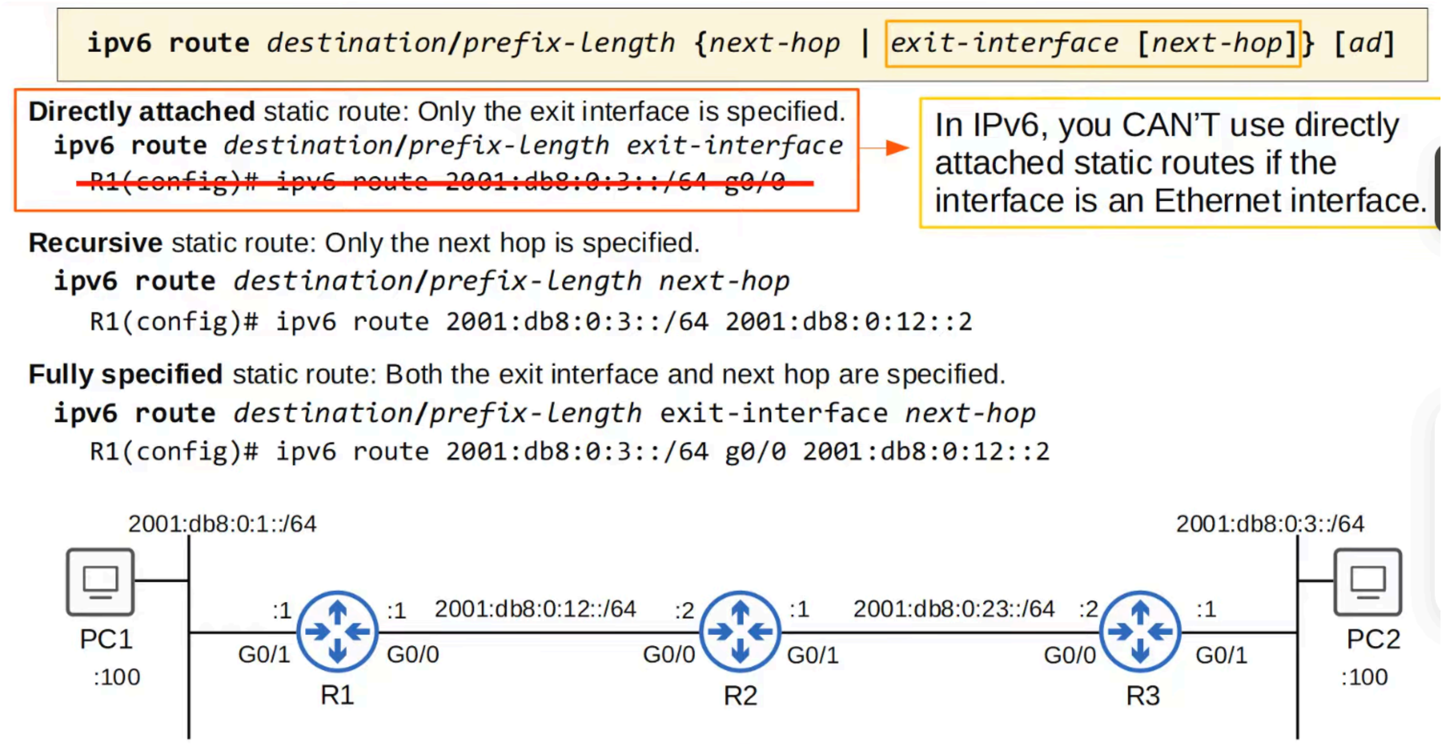
- **Fully Specified Static Route**




  - Specifies both the exit interface and the next-hop address
  - Example:

```
R1(config)# ipv6 route 2001:db8:0:3::/64 g0/0 2001:db8:0:12::2
```

# Note on Directly Attached Static Routes in IPv6




- Directly attached static routes **do not work on Ethernet interfaces** in IPv6
    - Example: The command `ipv6 route 2001:db8:0:3::/64 g0/0` will be accepted but the route will not function; the router will not send packets
- This restriction does not apply to serial interfaces
- Note: While directly attached static routes work on Ethernet interfaces in IPv4, they do not in IPv6 — you must use a recursive or fully specified static route instead

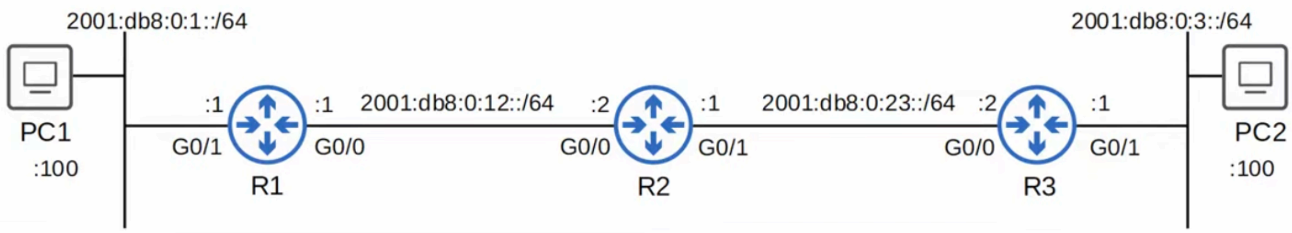
# IPv6 Static Route Examples (Exam Topics)

- **Network Route**

```
ipv6 route destination/prefix-length {next-hop | exit-interface [next-hop]} [ad]
```

**Directly attached** static route: Only the exit interface is specified.

```
ipv6 route destination/prefix-length exit-interface
  R1(config)# ipv6 route 2001:db8:0:3::/64 g0/0
```




  - A route to a specific subnet
  - Example:

```
R1(config)# ipv6 route 2001:db8:0:3::/64 2001:db8:0:12::2
```

  - This tells R1: if you receive a packet with a destination in the 2001:db8:0:3::/64 network (connected to R3), send the packet to R2 at 2001:db8:0:12::2

- ### Host Route

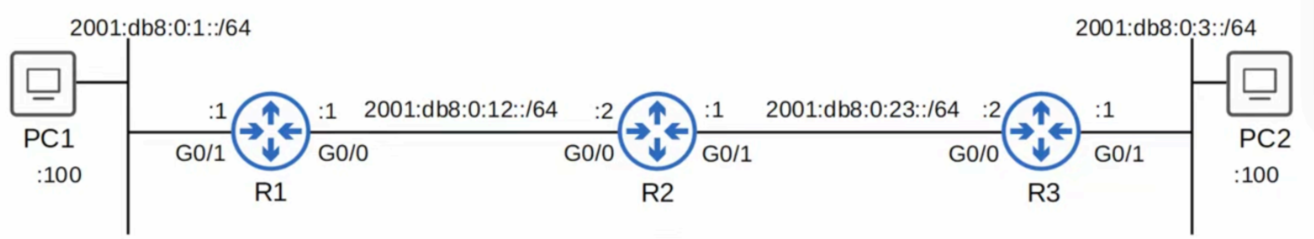
```
ipv6 route destination/prefix-length {next-hop | exit-interface [next-hop]} [ad]
```

**Network route:**

```
R1(config)# ipv6 route 2001:db8:0:3::/64 2001:db8:0:12::2
```

**Host route:**

```
R2(config)# ipv6 route 2001:db8:0:1::100/128 2001:db8:0:12::1
R2(config)# ipv6 route 2001:db8:0:3::100/128 2001:db8:0:23::2
```




  - A route to a single specific host (uses /128 prefix length)
  - Example — host routes on R2 to PC1 and PC2:

```
R2(config)# ipv6 route 2001:db8:0:1::1/128 [next-hop]
R2(config)# ipv6 route 2001:db8:0:2::1/128 [next-hop]
```

  - Note: Host routes are not normally used in this scenario; network routes would be preferred, but this demonstrates how to configure them

- **Default Route**

```
ipv6 route destination/prefix-length {next-hop | exit-interface [next-hop]} [ad]
```

**Network route:**

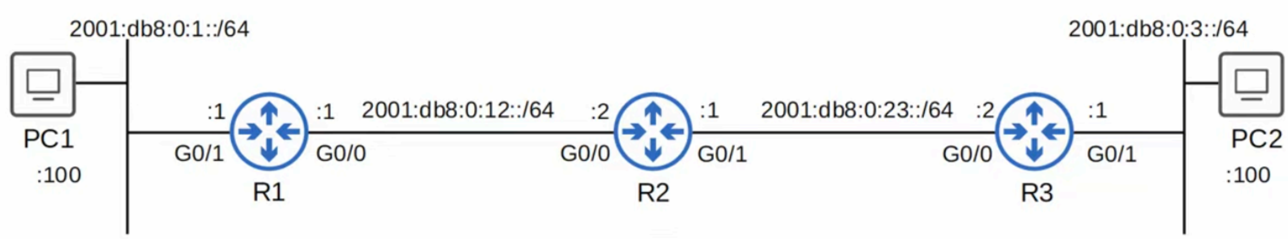
```
R1(config)# ipv6 route 2001:db8:0:3::/64 2001:db8:0:12::2
```

**Host route:**

```
R2(config)# ipv6 route 2001:db8:0:1::100/128 2001:db8:0:12::1
R2(config)# ipv6 route 2001:db8:0:3::100/128 2001:db8:0:23::2
```

**Default route:**

```
R3(config)# ipv6 route ::/0 2001:db8:0:23::1
```




  - Uses `::/0` (equivalent to `0.0.0.0/0` in IPv4)
  - Example:

```
R3(config)# ipv6 route ::/0 [next-hop]
```

- All four of the above routes are **recursive** — they all specify only the next hop

# Floating Static Routes

```
ipv6 route destination/prefix-length {next-hop | exit-interface [next-hop]} [ad]
```

- Created by raising the **Administrative Distance (AD)** to make a static route a backup route
- Example: If the main route is learned via OSPF (AD 110), set the static route's AD to higher than 110

- Example: If the main route is learned via EIGRP (AD 90), set the static route's AD to higher than 90
- Note: Always set the AD to higher than the main route

## Link-Local Address as Next-Hop


- On R1, attempted to configure a route to R3’s network using R2’s link-local address as a next hop

- Received an error message and the command did not work: **'Interface has to be specified for a link-local nexthop'**


- To use a link-local address as a next-hop, you must specify both the next hop address and the exit interface
- This makes it a **fully specified static route**
    - Example:

```
R1(config)# ipv6 route 2001:db8:0:3::/64 g0/0 fe80::2
```

- The reason: With a link-local next-hop address, the router cannot determine on its own which interface that next-hop is connected to

## Lecture Review

- Basic rules for IPv6 address representation (per RFC 5952) — not a major test topic, but good to know
- IPv6 header: fixed size of 40 bytes, much simpler than the IPv4 header — unlikely to be directly tested, but fundamental knowledge
- **NDP (Neighbor Discovery Protocol)**: serves multiple functions
  - Replaces ARP using **Neighbor Solicitation (NS)** and **Neighbor Advertisement (NA)** messages
  - Automatic router discovery using **Router Solicitation (RS)** and **Router Advertisement (RA)** messages
- **SLAAC (Stateless Address Auto-configuration)**: uses RS/RA messages to learn the network prefix and automatically configure an IPv6 address
- **IPv6 static routes**: same types as IPv4
  - Three structural types: directly attached, recursive, fully specified
  - Four functional types: network, host, default, floating