IPv6

The Next Generation Internet Protocol

Ing. Carlos Barcenilla / Universidad Tecnológica Nacional Facultad Regional La Plata
c.a.barcenilla@ieee.org

IPv6 Motivations

- Address space depletion.
- Router table explosion.
- Other protocol constraints.
  - Fragmentation inefficiency
  - Control (ICMP useless messages)
  - Checksums

IPv6 Technical Criteria for IPng

- Scale
- Topological flexibility
- Performance
- Robust Service
- Straightforward transition
- Media independence
- Unreliable Datagram Service
- Configuration, Administration and Operation
- Secure Operation
- Unique Naming
- Access and Documentation
- Multicast
- Extensibility
- Network Service
- Mobility
- Control Protocol
- Private Networks

IPv6 Address Space Depletion

- IPv4 Address = 32 bits.
  - Key Drivers:
    - Cellular / IP, DSL & Cable Modems, “always on” service
    - IP-Enabled devices in home and car environments
    - IP Telephony
    - Asia/Pacific region grow
  - Time frame for exhaustion: 2004-2007 (According ICANN 450)

IPv6 Address Space Depletion / Solutions

- Short term solution: Use of NAT boxes
  - Does not work for IP Telephony
  - They inhibit deployment of new services
  - They compromise the performance, robustness, security, and manageability of the Internet
  - No end-to-end IPSec
    - NATs Are Not Adequate!

- New types of applications and new types of access need unique address!

- Long term solution: IPv6 Address = 128 bits
  - 66,722,032,729,595,777,149,681 addresses per square meter of the earth’s surface
  - 54,442,035,206,815,861,340,125,082,737 per human being
Routing requires tables which have grown unmanageably large (more than 133,000 entries at the core, 15-Set-2002).

Under IPv4 a Classless Interdomain Routing is being used (CIDR).

A good use of CIDR would led to a 30% reduction in router table size (14-Set-2002).

IPv6 addressing is Classless by nature.

Expanded addressing capabilities.
- Address size: 128 bits.
- Improved scalability of multicast (scope field).
- Anycast addresses.
- No more broadcast addresses.

Header format.
- Some IPv4 fields were dropped or made optional.
- Improved support for extensions and options.

Flow labeling (QoS real-time).
- Authentication and privacy capabilities.

Address: an IPv6-layer identifier for an interface or a set of interfaces.

Link: A communication facility or medium over which nodes can communicate at the link level (e.g. Ethernet, Token Ring, Frame Relay, ATM and so on.)

Packet: an IPv6 header plus payload.

Link MTU: the maximum transmission unit (max. packet size in octets) that can be conveyed over a link.

Path MTU: The minimum link MTU of all the links in a path between a source node and a destination node.

Upper layer: a protocol layer immediately above IPv6 (e.g. TCP, UDP, ICMP, OSPF and so on.)

Motivations
- Address Space, Router Table Size, IPv4 Protocol Constraints.

Changes
- Vast Address Space, Improved Multicast, No Broadcast, Anycast.

New Header
- Bigger but Simpler, Flow Labeling, Authentication & Privacy, Extensible.

Terminology
- Node, Router, Host, Link, Neighbor, Address.
**IPv4 Header Format**

<table>
<thead>
<tr>
<th>Field</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>0-4</td>
<td>0</td>
</tr>
<tr>
<td>Internet Header Length (IHL):</td>
<td>5-15</td>
<td>0-4</td>
</tr>
<tr>
<td>Type of Service (TOS):</td>
<td>16-20</td>
<td>0</td>
</tr>
<tr>
<td>Total Length:</td>
<td>21-32</td>
<td>0-4</td>
</tr>
<tr>
<td>Identification:</td>
<td>33-47</td>
<td>0-8</td>
</tr>
<tr>
<td>Flags and Fragment Offset:</td>
<td>48-55</td>
<td>0-16</td>
</tr>
<tr>
<td>Protocol:</td>
<td>56-63</td>
<td>0-4</td>
</tr>
<tr>
<td>Header Checksum:</td>
<td>64-71</td>
<td>0-16</td>
</tr>
<tr>
<td>Source Address:</td>
<td>72-95</td>
<td>0-8</td>
</tr>
<tr>
<td>Destination Address:</td>
<td>96-127</td>
<td>0-16</td>
</tr>
<tr>
<td>Options:</td>
<td>Present in IPv4</td>
<td>Present in IPv6</td>
</tr>
</tbody>
</table>

**Summary of Header changes**

- **Size:**
  - 40-byte Fixed-Length
- **Address Size:**
  - Increased from 32 to 128 bits.
- **Removed Fields:**
  - Fragmentation options
  - Header Checksum
- **Changed Fields:**
  - Total Length ➔ Payload Length
  - TOS ➔ Traffic Class
  - TTL ➔ Hop Limit
- **New Field:**
  - Flow Label

**IPv6 Header Format**

- **Version:** 4-bit version number (6).
- **Traffic Class:** 8-bit traffic class field.
- **Flow Label:** 20-bit flow label.
- **Payload Length:** 16-bit unsigned integer.
- **Next Header:** 8-bit selector.
- **Hop Limit:** 8-bit unsigned integer.
- **Source Address:** 128-bit address.
- **Destination Address:** 128-bit address.

**Extension Headers**

- Extension headers are not examined or processed by any node along a packet’s delivery path, until the packet reaches the node (or nodes in case of multicast).

  The exception is the hop-by-hop header which carries information that must be examined and processed by every node along the path, including the source and destination nodes.

- Must be processed strictly in the order they appear in the packet.

  If a node does not recognize a Next header value, it should discard the packet and send an ICMP Parameter Problem message.

  Each extension header should occur at most once, except for the destination options header which should occur at most twice.
Extension Headers

- Hop-by-hop options.
- Routing.
- Fragment.
- Destination options.
- Authentication.
- Encapsulating security payload.

Hop-by-Hop and Destination Options Headers:

Options

- The Hop-by-Hop Options header and the Destination Options header carry a variable number of type-length-value (TLV) encoded "options".

<table>
<thead>
<tr>
<th>Option Type</th>
<th>Opt Data Len</th>
<th>Option Data</th>
</tr>
</thead>
</table>

- Option Type: 8-bit identifier of the type of option.
- Opt Data Len: 8-bit unsigned integer.
- Option Data: Variable-length field.

- The sequence of options within a header must be processed strictly in the order they appear in the header.

Hop-by-hop Options Header

- Carries additional information that must be examined by every node along a packet’s delivery path.

<table>
<thead>
<tr>
<th>Next Header</th>
<th>Hdr Ext Len</th>
<th>Options</th>
</tr>
</thead>
</table>

- Next Header: 8-bit selector.
- Hdr Ext Len: 8-bit unsigned integer (Length of the header not including the first 8 octets).
- Options: variable-length field (contains one or more TLV-encoded options, the length of the complete header must be multiple of 8 octets long).
- The only options defined in RFC2460 are Pad1 and PadN (for alignment).

Hop-by-hop Option / Router Alert Option

- Alerts transit routers to more closely examine the contents of an IP datagram.
- It is useful for situations where a datagram addressed to a particular destination contains information that may require special processing by routers along the path.

<table>
<thead>
<tr>
<th>Option Type</th>
<th>Opt Data Len</th>
<th>Value</th>
</tr>
</thead>
</table>

- Option Type: 5, means that nodes not recognizing this option type should skip over it and continue processing the header and that the option must not change en route.
- Opt Data Len: 2 bytes.
- Value:
  - 0: Datagram contains a Multicast Listener Discovery message.
  - 1: Datagram contains RSVP message.
  - 2: Datagram contains an Active Networks message.

Destination Options Header

- This header is used to carry optional information that need be examined only by a packet’s destination node(s).

<table>
<thead>
<tr>
<th>Next Header</th>
<th>Hdr Ext Len</th>
<th>Options</th>
</tr>
</thead>
</table>

- Next Header: 8-bit selector.
- Hdr Ext Len: 8-bit unsigned integer (Length of the header not including the first 8 octets).
- Options: variable-length field (contains one or more TLV-encoded options, the length of the complete header must be multiple of 8 octets long).
- The only options defined in RFC2460 are Pad1 and PadN (for alignment).
Routing Header

- Used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet’s destination.

<table>
<thead>
<tr>
<th>Next Header</th>
<th>Hdr Ext Len</th>
<th>Routing Type</th>
<th>Segments Left</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Next Header: 8-bit selector.
- Hdr Ext Len: 8-bit unsigned integer (Length of the header not including the first 8 octets).
- Routing Type: 8-bit identifier of a particular Routing header variant.
- Segments Left: 8-bit unsigned integer. Number of route segments remaining.
- Type-specific data: Variable-length field, of format determined by the Routing Type, and of length such that the complete Routing header is an integer multiple of 8 octets long.

Type 0 Routing Header

<table>
<thead>
<tr>
<th>Next Header</th>
<th>Hdr Ext Len</th>
<th>Routing Type: 0</th>
<th>Segments Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td></td>
<td>Address[1]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address[2]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address[n]</td>
<td></td>
</tr>
</tbody>
</table>

- Routing Type: 0.
- Segments Left: 8-bit unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
- Reserved: 32-bit reserved field. Initialized to zero for transmission; ignored on reception.
- Address[1..n]: Vector of 128-bit addresses, numbered 1 to n.

Routing Header Example

<table>
<thead>
<tr>
<th>Src Address</th>
<th>Dst Address</th>
<th>Hdr Ext Len</th>
<th>Routing Type</th>
<th>Segments Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>I1</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2</td>
<td>I3</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I1</td>
<td>D</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type 0 Routing Header Processing Algorithm

```plaintext
if Segments Left = 0 { process next header in the packet }
else {
  n = Hdr Ext Len / 2 (number of addresses in the Routing Header)
  Segments Left = Segments Left – 1
  i = n - Segments Left (i: index of next address to be visited)
  swap the Destination Address and Address[i]
  if Hop Limit is <= 1 { send ICMP Time Exceeded }
  else {
    Hop Limit = Hop Limit – 1
    resubmit the packet to the IPv6 module for transmission to the new destination
  }
}
```

RFC2460 contains a more detailed algorithm.

Summary / Extension Headers

- Extension Headers
  - End-to-end headers (except hop-by-hop Ext. Header.)
  - Hop-by-hop and Destination Options contain TLV options.
  - No more 40-byte limit on options (IPv4)

Fragmentation

- The original, unfragmented packet consists of two parts.

<table>
<thead>
<tr>
<th>Unfragmentable Part</th>
<th>Fragmentable Part</th>
</tr>
</thead>
</table>

- The Unfragmentable Part consists of the IPv6 header plus any extension headers that must be processed by nodes en route to the destination.
- The Fragmentable Part consists of the rest of the packet.
- The Fragmentable Part of the original packet is divided into fragments, each, except possibly the last one, being an integer multiple of 8 octets long.
- Original packet:
IP v6 Fragmentation

- Fragment packets:
  - Unfragmentable Part
  - Fragment Header
  - first fragment
  - Unfragmentable Part
  - Fragment Header
  - second fragment
  - Unfragmentable Part
  - Fragment Header
  - last fragment

- Each fragment packet is composed of:
  1. The Unfragmentable Part of the original packet.
  2. A Fragment header.
  3. The fragment itself.
  4. The lengths of the fragments must be chosen such that the resulting fragment packets fit within the path MTU.

IP v6 Fragmentation Header

- Is used by a source to send a packet larger than the path MTU to its destination. Unlike IPv4, fragmentation is only performed by source nodes.

- Next Header: 8-bit selector.
- Reserved: 8-bit reserved field.
- Fragment Offset: 13-bit unsigned integer. The offset, in 8-octet units, of the data following this header, relative to the start of the Fragmentable Part of the original packet.
- Res: 2-bit reserved field.
- M flag: 1 = more fragments; 0 = last fragment.
- Identification: 32 bits. The Identification must be different than any other fragmented packet sent recently with the same Source Address and Destination Address.

IP v6 Reassembly

- At the destination, fragment packets are reassembled into their original, unfragmented form:

    Unfragmentable Part | Fragmentable Part

- An original packet is reassembled only from fragment packets that have the same Source Address, Destination Address, and Fragment Identification.
- The Unfragmentable Part of the reassembled packet consists of all headers up to, but not including, the Fragment Header of the first fragment packet.
- The Fragmentable Part of the reassembled packet is constructed from the fragments following the Fragment headers in each of the fragment packets.

IP v6 Packet Size

- IPv6 requires that every link in the internet have an MTU of 1280 octets or greater.
- On links with MTU < 1280, link-specific fragmentation and reassembly must be used.
- From each link to which a node is directly attached, the node must be able to accept packets as large as that link’s MTU.
- It is strongly recommended that IPv6 nodes implement Path MTU Discovery, in order to discover and take advantage of path MTUs greater than 1280 octets.
- In order to send a packet larger than a path’s MTU, a node may use the IPv6 Fragment header.
- A node must be able to accept a fragmented packet that, after reassembly, is as large as 1500 octets.
- Recommended MTU: 1500 bytes (If configurable)

IP v6 Jumbograms

- IPv6 Header supports up to 65535-byte payload size.
- Bigger payloads can be carried setting the IPv6 Payload Length to zero, and adding the “Jumbogram” hop-by-hop option.

<table>
<thead>
<tr>
<th>Option Type (194)</th>
<th>Opt Data Len (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jumbo Payload Length (32-bit unsigned integer)</td>
<td></td>
</tr>
</tbody>
</table>

- The Jumbo Payload option must not be used in a packet that carries a Fragment header.
- Allows payloads between 65,536 and 4,294,967,295 octets in length.
### Maximum Upper-Layer Payload Size

When computing the maximum payload size for upper-layer data, an upper-layer protocol must take into account the larger size of the IPv6 header relative to the IPv4 header.

- For example TCP MSS:
  - IPv4: $\text{MSS} = \text{Max. Packet Size} - 40$ (20 octets for the minimum-length IPv4 header and 20 octets for the minimum-length TCP header)
  - IPv6: $\text{MSS} = \text{Max. Packet Size} - 60$ (40 octets for the minimum-length IPv6 header and 20 octets for the minimum-length TCP header)

### Summary / Fragmentation / Packet Size

- Occurs when the Packet Size > PMTU
- **Fragmentation**: Always at the Source Node!
- **Reassembly**: At the Destination Node
- Fragment Header is used.
- Minimum MTU of a link: 1280 bytes
- Path MTU Discovery recommended
- Jumbograms for payloads > 65535 bytes

### Flow Labels

- The 20-bit Flow Label in the IPv6 header may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or "real-time" service.

- This aspect of IPv6 is still experimental, and may change.

- A flow is a sequence of packets sent from a particular source to a particular destination for which the source desires special handling by the intervening routers.

- There may be multiple active flows from a source to a destination, as well as traffic that is not associated with any flow.

- There is no requirement that all, or even most, packets belong to flows.

### Traffic Classes

- The 8-bit Traffic Class field in the IPv6 header is available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets (e.g., "differentiated services").

- **General requirements**:
  - The service interface to the IPv6 service within a node must provide a means for an upper-layer protocol to supply the value of the Traffic Class bits.
  - Nodes that support a specific use of the Traffic Class bits are permitted to change the value of those bits in packets that they originate, forward, or receive.
  - An upper-layer protocol must not assume that the value of the Traffic Class bits in a received packet are the same as the value sent by the packet's source.

### Upper-Layer Checksums

- Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6. The TCP/UDP "pseudo-header" for IPv6 is:

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
<th>Upper-Layer Packet Length</th>
<th>Next Header</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- If the IPv6 packet contains a Routing Header, the Destination Address in the pseudo-header is that of the final destination.

- The Next Header in the pseudo-header identifies the upper-layer protocol (e.g., 6 for TCP, or 17 for UDP).

- The Upper-Layer Packet Length in the pseudo-header is the length of the upper-layer header and data.

- Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional.

- The IPv6 version of ICMP includes this pseudo-header in its checksum computation.
Maximum Packet Lifetime

- Unlike IPv4, IPv6 nodes are not required to enforce maximum packet lifetime.
- That is the reason the IPv4 “Time to Live” field was renamed “Hop Limit” in IPv6.

Addressing Model

- IPv6 addresses of all types are assigned to interfaces, not nodes.
- All interfaces are required to have at least one link-local unicast address.
- A single interface may be assigned multiple IPv6 addresses of any type or scope.
- A subnet prefix is associated with one link. Multiple subnet prefixes may be assigned to the same link.
- Address size has been expanded to 128 bits.
  - Address scope can be: link-local, site-local or global.

Types of addresses

- Unicast.
  - An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.

- Anycast.
  - An identifier for a set of interfaces. A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the “nearest”).

- Multicast.
  - An identifier for a set of interfaces. A packet sent to a multicast address is delivered to all interfaces identified by that address.

There are no broadcast addresses in IPv6.
IP v6
Address assigned to interfaces
Multiple addresses per interface
Multiple prefixes per link
Types
- Unicast
- Multicast
- Anycast
Scope
- Link-Local
- Site-Local
- Global

No Broadcast!

Text Representation of Addresses

Preferred form: x:x:x:x:x:x:x:x
x: hex. Values of the eight 16 bit pieces of the address.

Syntax for compress the zeros:
"::" Indicate multiple groups of 16 bit zeros.
The "::" can only appear once in an address.
Ex.: 1080::8:800:200c:417a
0:0:0:0:0:0:0:1
0:0:0:0:0:0:0::

In URLs:
http://[3FFE:38E1:100::2:4A]:8000

Text Representation of Address Prefixes

IPv6-address/prefix-length

Where:
- IPv6-address: an IPv6 address in any notation.
- Prefix-length: specifies how many of the leftmost contiguous bits of the address comprise the prefix.

Examples:
- Node address: 12AB:0:0:CD31:123:4567:89AB:CDEF
- Subnet: 12AB:0:0:CD30::/60
- Node + Subnet: 12AB:0:0:CD31:123:4567:89AB:CDEF/60

Address Type Representation

<table>
<thead>
<tr>
<th>Allocation</th>
<th>Prefix (binary)</th>
<th>Fraction of Address Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0000 0000</td>
<td>1/32</td>
</tr>
<tr>
<td>Unassigned</td>
<td>0000 0001</td>
<td>1/32</td>
</tr>
<tr>
<td>Reserved for ISAP Allocation</td>
<td>0000 0001</td>
<td>1/32</td>
</tr>
<tr>
<td>Reserved for IPv6 Allocation</td>
<td>0000 0010</td>
<td>1/128</td>
</tr>
<tr>
<td>Reserved</td>
<td>0000 0111</td>
<td>1/32</td>
</tr>
<tr>
<td>Unassigned</td>
<td>0001 0000</td>
<td>1/16</td>
</tr>
<tr>
<td>Aggregatable Global Unicast Addresses</td>
<td>001</td>
<td>1/6</td>
</tr>
<tr>
<td>Unassigned</td>
<td>001</td>
<td>1/6</td>
</tr>
<tr>
<td>Unassigned</td>
<td>001</td>
<td>1/6</td>
</tr>
<tr>
<td>Unassigned</td>
<td>010</td>
<td>1/6</td>
</tr>
<tr>
<td>Unassigned</td>
<td>100</td>
<td>1/6</td>
</tr>
<tr>
<td>Unassigned</td>
<td>101</td>
<td>1/6</td>
</tr>
<tr>
<td>Unassigned</td>
<td>110</td>
<td>1/6</td>
</tr>
<tr>
<td>Unassigned</td>
<td>111</td>
<td>1/6</td>
</tr>
<tr>
<td>Unassigned</td>
<td>1110</td>
<td>1/32</td>
</tr>
<tr>
<td>Unassigned</td>
<td>1111 0</td>
<td>1/128</td>
</tr>
<tr>
<td>Unassigned</td>
<td>1111 10</td>
<td>1/128</td>
</tr>
<tr>
<td>Unassigned</td>
<td>1111 110 0</td>
<td>1/512</td>
</tr>
<tr>
<td>Link-Local Unicast Addresses</td>
<td>1111 1110 10</td>
<td>1/1624</td>
</tr>
<tr>
<td>Site-Local Unicast Addresses</td>
<td>1111 1111 0</td>
<td>1/1624</td>
</tr>
<tr>
<td>Unicast Addresses</td>
<td>1111 1111</td>
<td>1/32</td>
</tr>
</tbody>
</table>

Unicast addresses are distinguished from multicast addresses by the value of the high-order octet of the address; a value of FF identifies a multicast address.

Anycast addresses are taken from the unicast address space, and are not syntactically distinguishable.
IP v6

**Summary / Addressing Type Representation**
- Preferred form: 3ffe:38e1:100:a001:80:0:0:36
- Compression: 3ffe:38e1:100:a001:80::36
- Mixed: 3ffe:38e1:100:a001:128.0.0.54
- URL: http://[3ffe:38e1:100:a001:80::36]:8080
- How to distinguish address types:
  - Multicast: FF::/16
  - Aggregatable Global Unicast Addresses: 2000::/3
  - Anycast: Undistinguishable from unicast

---

IP v6

**Unicast Addresses**
- Aggregatable with contiguous bitwise mask (like IPv4 CIDR).
- Forms:
  - Global aggregatable unicast address.
  - NSAP address.
  - IPX address.
  - Site-local.
  - Link-local.
  - IPv4-capable host address.

---

IP v6

**Unicast Addresses**
- At a minimum, a node may consider that unicast addresses have no internal structure.
  - 128 bits
    - node address
  - A slightly sophisticated host may additionally be aware of subnet prefix(es) for the link(s) it is attached to.
    - n bits
    - 128-n bits
      - Subnet prefix
      - Interface ID
  - Still more sophisticated hosts may be aware of other hierarchical boundaries in the unicast address.

---

IP v6

**Interface Identifiers**
- Interface identifiers in IPv6 unicast addresses:
  - Used to identify interfaces on a link.
  - Are required to be unique on that link.
  - In many cases an interface’s ID will be the same as that interface’s link layer address.
  - The same interface identifier may be used on multiple interfaces on a single node.

---

IP v6

**The Unspecified Address**
- The address 0:0:0:0:0:0:0:0 is called the unspecified address.
  - It indicates the absence of an address.
  - Ex.: In the Source Address field do any IPv6 packets sent by an initializing host before it has learnt its own address.
  - The unspecified address must not be used as the destination address of IPv6 packets or in IPv6 Routing Headers.

---

IP v6

**The Loopback Address**
- The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It may be used by a node to send an IPv6 packet to itself.
  - It may never be assigned to any physical interface.
  - Must not be used as the Source Address in IPv6 packets that are sent outside of a single node.
  - A packet containing this address must never be forwarded by an IPv6 Router.
The IP v6 transition mechanisms include a technique for hosts and routers to dynamically tunnel IP v6 packets over IP v4 routing infrastructure. IP v6 nodes that utilize this technique are assigned special IPv6 unicast addresses that carry an IPv4 address in the low-order 32-bits.

Example: ::170.210.16.2

This address is used to represent the addresses of IPv4-only nodes as IPv6 addresses.

For example, an IPv6 host would use an IPv4-mapped IPv6 address to communicate with another host that only supports IPv4.

Example: ::FFFF:170.210.16.2

Link-Local addresses are designed to be used for addressing on a single link for purposes such as auto-address configuration, neighbor discovery, or when no routers are present.

Example: FEB0::1234:5678:9ABC:DEFO

Routers must not forward any packets with link-local source or destination addresses to other links.

Site-Local addresses are designed to be used for addressing inside a site without the need for a global prefix.

Routers must not forward any packets with site-local source or destination addresses outside of the site.

Aggregatable addresses are organized into a three level hierarchy:
- Public Topology
- Site Topology
- Interface Identifier

Designed to support both provider based aggregation and exchanges. Sites will have the choice to connect to either type of aggregation point.
### Aggregatable Global Unicast Addresses Example

```
<table>
<thead>
<tr>
<th>Prefix</th>
<th>TLA ID</th>
<th>NLA ID</th>
<th>SLA ID</th>
<th>Interface ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2345:00C1:CA11:0001:1234:5678:9ABC:DEF0</td>
<td>0345 (hex) [2345::/16]</td>
<td>0001 (hex) [2345:00C0::/28]</td>
<td>0001 (hex) [2345:00C1:CA11::/48]</td>
<td>1234:5678:9ABC:DEF0 (hex)</td>
</tr>
</tbody>
</table>
```

### Summary / Unicast Addressing
- **Format**: Subnet Prefix + Interface ID
- **Unspecified Address**: ::
- **Loopback Address**: ::1
- **IPv4-Compatible**: ::w.x.y.z (ex. ::200.45.2.3)
- **IPv4-Mapped**: ::FFFF:w.x.y.z (ex. ::FFFF:200.45.2.3)
- **Scope**:
  - Link-Local
  - Site-Local
  - Aggregatable Global Unicast (FP / TLA / NLA / SLA / Interface ID)

### Anycast Addresses
- Are assigned to more than one interface (typically belonging to different nodes).
- A packet sent to an anycast address is routed to the “nearest” interface having the address, according to the routing protocols’ measure of distance.
- Are allocated from the unicast address space.
- Are syntactically indistinguishable from unicast addresses.
- Must not be used as a source address.
- May only be assigned to routers, not hosts.

### Required Anycast Address
- The Subnet-Router anycast address is predefined.

```
<table>
<thead>
<tr>
<th>subnet prefix</th>
<th>128-n bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000………………0000</td>
<td>n bits</td>
</tr>
</tbody>
</table>
```

- The “subnet prefix” in an anycast address is the prefix which identifies a specific link.
- Packets sent to the Subnet-Router anycast address will be delivered to one router on the subnet. All routers are required to support this addresses for the subnets which they have interfaces.

### Multicast Addresses
- An IPv6 multicast address is an identifier for a group of nodes. A node may belong to any number of multicast groups.

```
<table>
<thead>
<tr>
<th>8 bits</th>
<th>4 bits</th>
<th>4 bits</th>
<th>112 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111</td>
<td>flgs</td>
<td>scop</td>
<td>group ID</td>
</tr>
</tbody>
</table>
```

- 11111111 at the start of the address identifies the address as being a multicast address.
- flgs is a set of 4 flags: 0000
  - T = 0 indicates a permanently-assigned (“well-known”) multicast address.
  - T = 1 indicates a non-permanently-assigned (“transient”) multicast address.
- scop is a 4-bit multicast scope value to limit the scope of the group:
  - 1: node-local scope
  - 2: link-local scope
  - 5: site-local scope
  - 6: organization-local scope
  - E: global scope
- group ID identifies the multicast group.

### Multicast Addresses
- The “meaning” of a permanently-assigned multicast address is independent of the scope value. For example, if the “NTP servers group” is assigned a permanent multicast address with a group ID of 101 (hex), then:
  - FF01:101 means all NTP servers on the same node as the sender.
  - FF02:101 means all NTP servers on the same link as the sender.
  - FF05:101 means all NTP servers at the same site as the sender.
  - FF0E:101 means all NTP servers in the internet
- Multicast addresses must not be used as source addresses in IPv6 packets or appear in any routing header.
Pre-Defined multicast Addresses

- Reserved: FF0x:: (x: hex digit)
- All nodes:
  - FF01::1 (node-local)
  - FF02::1 (link-local)
- All routers:
  - FF01::2 (node-local)
  - FF02::2 (link-local)
  - FF05::2 (site-local)
- Solicited-node Address: FF02::1:FFxx:xxxx
  - This address is formed by taking the low-order 24 bits of the address (unicast or anycast) and appending those bits to the prefix FF02::1:FF00:0000/104
  - Example: for the IPv6 address 3FFE:3800:FFFF:BD12:3456 the solicited-node multicast address is: FF02::1:FF12:3456.

Summary / Multicast Addressing

  - t: transient / well-known
  - s: scope
  - g: group ID
- Reserved: FF0x:: (x: hex digit)
- All nodes: FF0s::1
- All routers: FF0s::2
- Solicited-node Address: FF02::1:FFxx:xxxx
  - For Neighbor Discovery (ARP Replacement)

Node Required Addresses (Host)

- Link-local address for each interface.
- Assigned Unicast Addresses.
- Loopback Addresses.
- All-nodes Multicast Addresses.
- Solicited-node Multicast Addresses for each of its assigned unicast and anycast addresses.
- Multicast Addresses of all other groups to which the host belongs.

Example:

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>tes8::250:56ff:fe8a:0</td>
<td>IPv6 link-local</td>
</tr>
<tr>
<td>f02::1:ff8a:0</td>
<td>Solicited Node Multicast</td>
</tr>
<tr>
<td>::1</td>
<td>Loopback</td>
</tr>
<tr>
<td>f02::1:1</td>
<td>All-nodes Multicast</td>
</tr>
</tbody>
</table>

Node Required Addresses (Router)

- All the host required addresses plus:
  - Subnet-router anycast addresses for the interfaces it is configured to act as a router on.
  - All other anycast addresses with which the router has been configured.
  - All-routers multicast addresses.
  - Multicast addresses of all other groups to which the router belongs.

Example:

<table>
<thead>
<tr>
<th>Address</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>fe80::260:ff:fe14:7861/102</td>
<td>IPv6 link-local/Solicited-Node Multicast</td>
</tr>
<tr>
<td>::1</td>
<td>Loopback</td>
</tr>
<tr>
<td>f02::1</td>
<td>All-nodes Multicast</td>
</tr>
<tr>
<td>f02::2</td>
<td>All-routers Multicast</td>
</tr>
</tbody>
</table>

Internet Control Message Protocol (ICMPv6)

- ICMPv6 is used by IPv6 nodes to report errors encountered in processing packets, and to perform other internet-layer functions such as diagnostics.
- ICMPv6 is an integral part of IPv6 and must be fully implemented by every IPv6 node.
- ICMPv6 messages are grouped into two classes: error messages and informational messages.
- High-order bit of the message Type:
  - 0: Error messages (Type: 0 to 127).
  - 1: Informational messages (Type: 128 to 255).

Integration of protocols in ICMPv6

ARP -> ICMPv4 -> ICMPv6 -> IGMP
ICMPv6 Messages

- ICMPv6 error messages:
  - 1 Destination Unreachable.
  - 2 Packet Too Big.
  - 3 Time Exceeded.
  - 4 Parameter Problem.

- ICMPv6 informational messages:
  - 128 Echo Request.
  - 129 Echo Reply.

ICMPv6 Message General Format

Every ICMPv6 message is preceded by an IPv6 header and zero or more IPv6 extension headers. The ICMPv6 header is identified by a Next Header value of 58 in the immediately preceding header.

<table>
<thead>
<tr>
<th>8</th>
<th>8</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

- Type: indicates the type of the message.
- Code: depends on the message type.
- Checksum: is used to detect data corruption in the ICMPv6 message and parts of the IPv6 header.

ICMPv6 Message Processing Rules

- If an ICMPv6 error message of unknown type is received, it must be passed to the upper layer.
- If an ICMPv6 informational message of unknown type is received, it must be silently discarded.
- Every ICMPv6 error message includes as much of the IPv6 offending packet as will fit without making the error message packet exceed the minimum IPv6 MTU.
- In those cases where the internet-layer protocol is required to pass an ICMPv6 message to the upper layer process, the upper-layer protocol type is extracted from the original packet and used to select the appropriate upper-layer process to handle the error.

ICMPv6 Error Message: Destination Unreachable

A Destination Unreachable message should be generated by a router or by the IPv6 layer in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion.

<table>
<thead>
<tr>
<th>8</th>
<th>8</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

- Unused: Must be initialized to zero by the sender and ignored by the receiver.

IPv6 Destination Address: Copied from the Source Address field of the invoking packet.

- Type: 1
- Code: 0 – no route to destination 1 – communication with dest. prohibited 2 – address unreachable 4 – port unreachable
- Unused: Must be initialized to zero by the sender and ignored by the receiver.

Used: As much of invoking packet as will fit without ICMPv6 packet exceeding the minimum IPv6 MTU.
**ICMPv6 Error Message:**

**Packet Too Big**

- Must be sent by a router in response to a packet that cannot be forwarded because the packet is larger than the MTU of the outgoing link. The information in this message is used as part of the Path MTU Discovery process.

<table>
<thead>
<tr>
<th>8</th>
<th>8</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>MTU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As much of invoking packet as will fit without ICMPv6 packet exceeding the minimum IPv6 MTU

- IPv6 Destination Address: Copied from the Source Address field of the invoking packet.
- Code: Set to 0 (zero) by the sender and ignored by the receiver.
- MTU: The Maximum Transmission Unit of the next hop link.

**ICMPv6 Error Message:**

**Time Exceeded Message**

- If a router receives a packet with a Hop Limit of zero, or a router decrements a packet’s Hop Limit to zero, it must discard the packet and send an ICMPv6 Time Exceeded message with Code 0 to the source of the packet. This indicates either a routing loop or too small an initial Hop Limit Value.

<table>
<thead>
<tr>
<th>8</th>
<th>8</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As much of invoking packet as will fit without ICMPv6 packet exceeding the minimum IPv6 MTU

- IPv6 Destination Address: Copied from the Source Address field of the invoking packet.
- Code: 0 – hop limit exceeded in transit 1 – fragment reassembly time exceeded
- Unused: Must be initialized to zero by the sender and ignored by the receiver.

**ICMPv6 Error Message:**

**Parameter Problem**

- If an IPv6 node finds a problem with a field in the IPv6 header or extension headers such that it cannot complete processing the packet, it must discard the packet and send an ICMPv6 Parameter Problem message to the packet’s source, indicating the type and location of the problem.

<table>
<thead>
<tr>
<th>8</th>
<th>8</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Pointer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As much of invoking packet as will fit without ICMPv6 packet exceeding the minimum IPv6 MTU

- IPv6 Destination Address: Copied from the Source Address field of the invoking packet.
- Type: 4
- Code: 0 – erroneous header field 1 – unrecognized Next Header type 2 – unrecognized IPv6 option
- Pointer: Identifies the octet offset within the invoking packet where the error was detected.

**ICMPv6 Error Message:**

**ICMPv6 Informational Message:**

**Echo Request**

- Every node must implement an ICMPv6 Echo responder function that receives Echo Requests and sends corresponding Echo Replies.

<table>
<thead>
<tr>
<th>8</th>
<th>8</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sequence Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data</td>
</tr>
</tbody>
</table>

- IPv6 Destination Address: Any legal IPv6 address.
- Type: 129
- Code: 0
- Identifier: An identifier to aid in matching Echo Replies to this Echo Request. May be zero.
- Sequence Number: A sequence number to aid in matching Echo Replies to this Echo Request. May be zero.
- Data: Zero or more octets of arbitrary data.

**ICMPv6 Informational Message:**

**Echo Reply**

- Every node must implement an ICMPv6 Echo responder function that receives Echo Requests and sends corresponding Echo Replies. The data received in the ICMPv6 Echo Request message must be returned entirely and unmodified in the ICMPv6 Echo Reply message.

<table>
<thead>
<tr>
<th>8</th>
<th>8</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Identifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sequence Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data</td>
</tr>
</tbody>
</table>

- IPv6 Destination Address: Copied from the Source Address field of the invoking Echo Request Packet.
- Type: 129
- Code: 0
- Identifier: The identifier from the invoking Echo Request message.
- Sequence Number: The sequence number from the invoking Echo Request message.
- Data: The data from the invoking Echo Request message.

**ICMPv6: Security Considerations**

- ICMP protocol packet exchanges can be authenticated using the IP Authentication Header. A node should include an Authentication Header when sending ICMP messages if a security association for use with the IP Authentication header exists for the destination address.

- Received Authentication Headers in ICMP packets must be verified for correctness and packets with incorrect authentication must be ignored and discarded.
IP v6

Summary / ICMPv6

ICMPv4 + ARP + IGMP  ↔  ICMPv6 (ND, MLD)

Error Messages:
- Destination Unreachable: Packet cannot be delivered
- Packet Too Big: Size > MTU
- Time Exceeded: Hop Limit = 0
- Parameter Problem: Field Error

Informational:
- Echo Request / Reply: Ping
- More messages for MLD, ND, Mobility

Neighbor Discovery

Nodes use Neighbor Discovery (ND) to determine the link-layer addresses for neighbors known to reside on attached links and to quickly purge cached values that become invalid.

Hosts also use Neighbor Discovery to find neighboring routers that are willing to forward packets on their behalf.

Nodes use the protocol to actively keep track of which neighbors are reachable and which are not, and to detect changed link-layer addresses.

When a router or the path to a router fails, a host actively searches for functioning alternates.

Neighbor Discovery Features

This protocol solves a set of problems related to the interaction between nodes attached to the same link:
- Router Discovery: How hosts locate routers that reside on an attached link.
- Prefix Discovery: How hosts discover the set of address prefixes that define which destinations are on-link for an attached link.
- Parameter Discovery: How a node learns such link parameters as the link MTU or such Internet parameters as the hop limit value to place in outgoing packets.
- Address Autoconfiguration: How nodes automatically configure an address for an interface.
- Address resolution: How nodes determine the link-layer address of an on-link destination (e.g., a neighbor) given only the destination's IP address.

Neighbor Discovery Messages

Neighbor Discovery defines five different ICMP packet types:
- Router Solicitation: When an interface becomes enabled, hosts may send out Router Solicitations that request routers to generate Router Advertisements immediately rather than at their next scheduled time.
- Router Advertisement: Routers advertise their presence together with various link and Internet parameters either periodically, or in response to a Router Solicitation message. Router Advertisements contain prefixes that are used for on-link determination and/or address configuration, a suggested hop limit value, etc.
Neighbor Discovery
(Additional Features)

- Neighbor Discovery also handles the following situations:
  - Link-layer address change: A node that knows its link-layer address has changed can multicast a few (unsolicited) Neighbor Advertisement packets to all nodes to quickly update cached link-layer addresses that have become invalid. The Neighbor Unreachability Detection algorithm ensures that all nodes will reliably discover the new address, though the delay may be somewhat longer.
  - Inbound load balancing: Nodes with replicated interfaces may want to load balance the reception of incoming packets across multiple network interfaces on the same link. Such nodes have multiple link-layer addresses assigned to the same interface. For example, a single network driver could represent multiple network interface cards as a single logical interface having multiple link-layer addresses.

Comparison with IPv4

- The IPv6 Neighbor Discovery protocol corresponds to a combination of the IPv4 protocols ARP, ICMP Router Discovery, and ICMP Redirect. In IPv4 there is no generally agreed upon protocol or mechanism for Neighbor Unreachability Detection.
- Router Discovery is part of the base protocol set.
- Router advertisements and Redirects carry link-layer addresses; no additional packet exchange is needed to resolve the router’s link-layer address.
- Router advertisements carry prefixes for a link; there is no need to have a separate mechanism to configure the “netmask”.
- Router advertisements enable Address Autoconfiguration.
- Routers can advertise an MTU for hosts to use on the link.
- Address resolution multicasts are “spread” over 4 billion (2^32) multicast addresses greatly reducing address resolution related interrupts on nodes other than the target. Moreover, non-IPv6 machines should not be interrupted at all.

Router Solicitation Message Format

- Hosts send Router Solicitations in order to prompt routers to generate Router Advertisements quickly.

<table>
<thead>
<tr>
<th>8</th>
<th>2</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Checksum</td>
</tr>
<tr>
<td>Reserved …</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options …</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IPv6 Fields:
- Source Address: An IP address assigned to the sending interface, or the unspecified address if no address is assigned to the sending interface.
- Destination Address: Typically the all-routers multicast address.
- Hop Limit: 255
- Authentication Header: If a Security Association for the IP Authentication Header exists between the sender and the destination address, then the sender SHOULD include this header.

ICMP Fields:
- Type: 133
- Code: 0
- Validation Options:
  - Source link-layer address: The link-layer address of the sender, if known.

Router Advertisements

- Routers send out Router Advertisement message periodically, or in response to a Router Solicitation.

<table>
<thead>
<tr>
<th>8</th>
<th>2</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Code</td>
<td>Checksum</td>
</tr>
<tr>
<td>Cur Hop Limit</td>
<td>Reserved</td>
<td>Router Lifetime</td>
</tr>
<tr>
<td>Reachable Time</td>
<td>Retrans Timer</td>
<td></td>
</tr>
<tr>
<td>Options …</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IPv6 Fields:
- Source Address: Must be the link-local address assigned to the interface from which this message is sent.
- Destination Address: Typically the Source Address of an incoming Router Solicitation or the all-nodes multicast address.
- Hop Limit: 255
- Authentication Header: If a Security Association for the IP Authentication Header exists between the sender and the destination address, then the sender should include this header.
**Neighbor Solicitation Message Format**

- Nodes send Neighbor Solicitations to request the link-layer address of a target node while also providing their own link-layer address to the target. Neighbor Solicitations are multicast when the node needs to resolve an address and unicast when the node seeks to verify the reachability of a neighbor.

<table>
<thead>
<tr>
<th>0</th>
<th>Code</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Neighbor Advertisement Message Format**

A node sends Neighbor Advertisements in response to Neighbor Solicitations and sends unsolicited Neighbor Advertisements in order to (un)propagate new information quickly.

<table>
<thead>
<tr>
<th>3</th>
<th>5</th>
<th>8</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>0</td>
<td>Reserved</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

**Redirect Message Format**

Routers send Redirect packets to inform a host of a better first-hop node on the path to a destination. Hosts can be redirected to a better first-hop router but can also be informed by a redirect that the destination is in fact a neighbor. The latter is accomplished by sending the ICMP Target Address equal to the ICMP Destination Address.

<table>
<thead>
<tr>
<th>0</th>
<th>Code</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Message Format**

This option should be included.
Redirect Message Format

ICMP Fields:
- Type: 137
- Code: 0
- Target Address: An IP address that is a better first hop to use for the ICMP Destination Address. When the target is the actual endpoint of communication, i.e., the destination is a neighbor, the Target Address field must contain the same value as the ICMP Destination Address field. Otherwise, the target is a better first-hop router and the Target Address must be the router's link-local address so that hosts can uniquely identify routers.
- Destination Address: The IP address of the destination which is redirected to the target.
- Possible Options:
  - Target link-layer address: The link-layer address for the target. It should be included (if known). Note that on NBMA links, hosts may rely on the presence of the Target Link-Layer Address option in Redirect messages as the means for determining the link-layer addresses of neighbors. In such cases, the option must be included in Redirect messages.
  - Redirected Header: As much as possible of the IP packet that triggered the sending of the Redirect without making the redirect packet exceed 1280 octets.

Summary / Neighbor Discovery

Features
- Router Discovery
- Prefix Discovery
- Parameter Discovery
- Address Autoconfiguration
- Address resolution
- Next-hop determination
- Neighbor Unreachability Detection
- Duplicate Address Detection
- Redirect

Messages
- Router Solicitation / Router Advertisement
- Neighbor Solicitation / Neighbor Advertisement
- Redirect

Conceptual Model of a Host

Data Structures

Neighbor Cache: A set of entries about individual neighbors to which traffic has been sent recently.
- Neighbor's on-link unicast IP address (key)
- Link-layer address
- IsRouter flag
- Pointer to any queued packets waiting for address resolution to complete
- Reachability state (NUD)
- Number of unanswered probes (NUD)
- Time the next NUD event is scheduled to take place

Destination Cache: A set of entries about destinations to which traffic has been sent recently. Includes both on-link and off-link destinations. It maps a destination IP address to the IP address of the next-hop neighbor.
- Prefix List: A list of the prefixes that define a set of addresses that are on-link.
- Entries are created from information received in Router Advertisements.
- Each entry has an associated invalidation timer value used to expire prefixes when they become invalid.
- A special "infinity" timer value specifies that a prefix remains valid forever, unless a new (finite) value is received in a subsequent advertisement.

Default Router List: A list of routers to which packets may be sent.
- Entries point to entries in the Neighbor Cache.
- The algorithm for selecting a default router favors routers known to be reachable over those whose reachability is suspect.
- Each entry also has an associated invalidation timer.
**Neighbor Cache**

**Neighbor's Reachability State**

- A key piece of information in the NC is a neighbor's reachability state, which is one of five possible values.
  - **INCOMPLETE**: Address resolution is in progress and the link-layer address of the neighbor has not yet been determined.
  - **REACHABLE**: The neighbor is known to have been reachable recently (within tens of seconds ago).
  - **STALE**: The neighbor is no longer known to be reachable but until traffic is sent to the neighbor, no attempt should be made to verify its reachability.
  - **DELAY**: The neighbor is no longer known to be reachable, and traffic has recently been sent to the neighbor. Rather than probe the neighbor immediately, however, delay sending probes for a short while in order to give upper layer protocols a chance to provide reachability confirmation.
  - **PROBE**: The neighbor is no longer known to be reachable, and unicast Neighbor Solicitation probes are being sent to verify reachability.

**Host Variables**

- In addition the host maintains a number of variables, e.g.:
  - **LinkMTU**: The MTU of the link.
  - **CurHopLimit**: The default hop limit to be used when sending unicast IPv6 packets.
  - **BaseReachableTime**: A base value used for computing the random ReachableTime value.
  - **ReachableTime**: The time a neighbor is considered reachable after receiving a reachability confirmation.
  - **RetransTimer**: The time between retransmissions of Neighbor Solicitation messages to a neighbor when resolving the address or when probing the reachability of a neighbor.

**Conceptual Sending Algorithm**

**Next-Hop Determination**

- Next-hop determination at the sender for a given unicast destination operates as follows.
  - Perform a longest prefix match against the Prefix List
  - Is packet's destination on-link?
    - Yes: Next-hop address is the same as the packet's destination address
    - No: Select a router from the Default Router List*

* If the Default Router List is empty, the sender assumes that the destination is on-link.

- For efficiency reasons, next-hop determination is not performed on every packet that is sent. Instead, the results of next-hop determination computations are saved in the Destination Cache. When the sending node has a packet to send, it first examines the Destination Cache. If no entry exists for the destination, next-hop determination is invoked to create a Destination Cache entry.

**Address Resolution**

- Once the IP address of the next hop is known, the Neighbor Cache is consulted for link-layer information about that neighbor.
  - Examine the Neighbor Cache for link-layer information about that neighbor
  - Entry exists?
    - Yes: Transmit the packet
    - No: Create one entry, set its state to INCOMPLETE, initiate Address Resolution, and then queue the data packet pending completion of address resolution

- For multicast-capable interfaces Address Resolution consists of sending a Neighbor Solicitation message and waiting for a Neighbor Advertisement.

- For multicast packets the next-hop is always the (multicast) destination address and is considered to be on-link.

- Each time a Neighbor Cache entry is accessed while transmitting a unicast packet, the sender checks Neighbor Unreachability Detection related information according to the Neighbor Unreachability Detection algorithm. This unreachability check might result in the sender transmitting a unicast Neighbor Solicitation to verify that the neighbor is still reachable.
Address Resolution

Address resolution is the process through which a node determines the link-layer address of a neighbor given only its IP address.

Address resolution is performed only on addresses that are determined to be on-link and for which the sender does not know the corresponding link-layer address.

Address resolution is never performed on multicast addresses.

When a multicast-capable interface becomes enabled the node must join the all-nodes multicast address on that interface, as well as the solicited-node multicast address corresponding to each of the IP addresses assigned to the interface.

Sending Neighbor Solicitations

When a node has a unicast packet to send to a neighbor, but does not know the neighbor's link-layer address, it performs address resolution.

For multicast-capable interfaces this entails creating a Neighbor Cache entry in the INCOMPLETE state and transmitting a Neighbor Solicitation message targeted at the neighbor.

The solicitation is sent to the solicited-node multicast address corresponding to the target address.

If the solicitation is being sent to a solicited-node multicast address, the sender must include its link-layer address as a Source Link-Layer Address option.

Including the source link-layer address in a multicasting solicitation is required to give the target an address to which it can send the Neighbor Advertisement.

Reception of Neighbor Solicitations

The recipient should create or update the Neighbor Cache entry for the IP Source Address of the solicitation.

If an entry does not already exist, the node SHOULD create a new one and set its reachability state to STALE.

If an entry already exists, and the cached link-layer address differs from the one in the received Source Link-Layer option, the cached address should be replaced by the received address and the entry's reachability state must be set to STALE.

After any updates to the Neighbor Cache, the node sends a Neighbor Advertisement response.

Example

<table>
<thead>
<tr>
<th>Src MAC Addr</th>
<th>Dst MAC Addr</th>
<th>Src IPv6 Addr</th>
<th>Dst IPv6 Addr</th>
<th>ICMP TYPE</th>
<th>Dir</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00:20:50:1:100:001.20</td>
<td>0x00:20:50:1:100:000.1.20</td>
<td>3ffe:38e1:100:001:20</td>
<td>3ffe:38e1:100:000:1.20</td>
<td>neighbor sol:</td>
<td>3</td>
</tr>
<tr>
<td>0x00:20:50:1:100:000.1.20</td>
<td>0x00:20:50:1:100:001.20</td>
<td>3ffe:38e1:100:000:1.20</td>
<td>3ffe:38e1:100:001:20</td>
<td>neighbor sol:</td>
<td>2</td>
</tr>
<tr>
<td>0x00:20:50:1:100:000.1.20</td>
<td>0x00:20:50:1:100:001.20</td>
<td>3ffe:38e1:100:000:1.20</td>
<td>3ffe:38e1:100:001:20</td>
<td>neighbor sol:</td>
<td>3</td>
</tr>
<tr>
<td>0x00:20:50:1:100:000.1.20</td>
<td>0x00:20:50:1:100:001.20</td>
<td>3ffe:38e1:100:000:1.20</td>
<td>3ffe:38e1:100:001:20</td>
<td>neighbor sol:</td>
<td>4</td>
</tr>
</tbody>
</table>

Note: The example shows a sequence of packets exchanged during the address resolution process. The packets are labeled with ICMP types and directions to illustrate the flow of information.
Address Resolution: Sending Solicited Neighbor Advertisements

- A node sends a Neighbor Advertisement in response to a valid Neighbor Solicitation targeting one of the node’s assigned addresses.
- The Target Address of the advertisement is copied from the Target Address of the solicitation.
- If the solicitation’s IP Destination Address is a multicast address, the Target Link-Layer option must be included in the advertisement.
- If the node is a router, it MUST set the Router flag to one; otherwise it must set the flag to zero.
- If the source of the solicitation is the unspecified address, the node must set the Solicited flag to zero and multicast the advertisement to the all-nodes address. Otherwise, the node must set the Solicited flag to one and unicast the advertisement to the Source Address of the solicitation.

State Machine for the Reachability State

Address Resolution: Receipt of Neighbor Advertisements

- When a valid Neighbor Advertisement is received, the Neighbor Cache is searched for the target’s entry.
- Once the appropriate Neighbor Cache entry has been located, the specific actions taken depend on the state of the Neighbor Cache entry, the flags in the advertisement and the actual link-layer address supplied:
  - State INCOMPLETE:
    - Record the link-layer address in the Neighbor Cache entry.
    - If the advertisement’s Solicited flag is set, the state of the entry is set to REACHABLE, otherwise it is set to STALE.
    - If the advertisement’s Solicited flag is set, the state of the entry is set to STALE.
    - Note that the Override flag is ignored if the entry is in the INCOMPLETE state.
  - If the target’s Neighbor Cache entry is in any state other than INCOMPLETE when the advertisement is received, processing becomes quite a bit more complex.

Address Resolution: Sending Unsolicited Neighbor Advertisements

- A node that has multiple IP addresses assigned to an interface may multicast a separate Neighbor Advertisement for each address.
- A proxy may multicast Neighbor Advertisements when its link-layer address changes or when it is configured (by system management or other mechanisms) to proxy for an address.
- A node belonging to an anycast address may multicast unsolicited Neighbor Advertisements for the anycast address when the node’s link-layer address changes.
- Because unsolicited Neighbor Advertisements do not reliably update caches in all nodes, they should only be viewed as a performance optimization to quickly update the caches in most neighbors.

Address Resolution: Anycast Neighbor Advertisements

- From the perspective of Neighbor Discovery, anycast addresses are treated just like unicast addresses in most cases.
- Nodes that have an anycast address assigned to an interface treat them exactly as if they were unicast addresses with two exceptions.
  - Neighbor Advertisements sent in response to a Neighbor Solicitation SHOULD be delayed by a random time between 0 and MAX_ANycast_DELAY_TIME to reduce the probability of network congestion.
  - Second, the Override flag in Neighbor Advertisements should be set to 0, so that when multiple advertisements are received, the first received advertisement is used rather than the most recently received advertisement.
Address Resolution:
Proxy Neighbor Advertisements:
A router may proxy for one or more other nodes, that is, through Neighbor Advertisements indicate that it is willing to accept packets not explicitly addressed to itself. For example, a router might accept packets on behalf of a mobile node that has moved off-link.

All solicited proxy Neighbor Advertisement messages must have the Override flag set to zero. This ensures that if the node itself is present on the link its Neighbor Advertisement will take precedence of any advertisement received from a proxy.

Finally, when sending a proxy advertisement in response to a Neighbor Solicitation, the sender should delay its response by a random time between 0 and MAX ANYCAST_DELAY_TIME seconds.

Router and Prefix Discovery:
Router Discovery is used to:
- Locate neighboring routers.
- Learn prefixes.
- Learn configuration parameters related to address autoconfiguration.

Prefix Discovery is the process through which hosts learn the ranges of IP addresses that reside on-link and can be reached directly without going through a router.

Routers send Router Advertisements that indicate whether the sender is willing to be a default router.

Router Advertisements also contain Prefix Information options that list the set of prefixes that identify on-link IP addresses.

Neighbor Unreachability Detection:
Communication to or through a neighbor may fail for numerous reasons at any time, including hardware failure, hot-swap of an interface card, etc.

Thus, a node actively tracks the reachability "state" for the neighbors to which it is sending packets.

NUD is used for all paths between hosts and neighboring nodes, including host-to-host, host-to-router, and router-to-router.

When a path to a neighbor appears to be failing, the specific recovery procedure depends on how the neighbor is being used.
- If the neighbor is the ultimate destination, address resolution should be performed again.
- If the neighbor is a router, attempting to switch to another router would be appropriate.

Neighbor Unreachability Detection signals the need for next-hop determination by deleting a Neighbor Cache entry.

Reachability Confirmation:
A neighbor is considered reachable if the node has recently received a confirmation that packets sent recently to the neighbor were received by its IP layer.

Positive confirmation can be gathered in two ways:
- Hints from upper layer protocols that indicate a connection is making "forward progress".
- Receipt of a Neighbor Advertisement message that is a response to a Neighbor Solicitation message.

In TCP, for example, receipt of a (new) acknowledgement indicates that previously sent data reached the peer. It is a confirmation that the next-hop neighbor is reachable.

For off-link destinations, forward progress implies that the first-hop router is reachable.

Redirect Function:
Redirect messages are sent by routers to redirect a host to a better first-hop router for a specific destination or to inform hosts that a destination is in fact a neighbor (i.e., on-link).

A router should send a redirect message, subject to rate limiting, whenever it forwards a packet that is not explicitly addressed to itself in which:
- The Source Address field of the packet identifies a neighbor, and
- the router determines that a better first-hop node resides on the same link as the sending node for the Destination Address of the packet being forwarded, and
- the Destination Address of the packet is not a multicast address.

The transmitted redirect packet:
- In the Target Address field: the address to which subsequent packets for the destination should be sent.
- In the Destination Address field: the destination address of the invoking IP packet.
- In the options:
  - Target Link-Layer Address option: link-layer address of the target, if known.
  - Redirected Header: as much of the forwarded packet as can fit without the redirect packet exceeding 1280 octets in size.

A router must limit the rate at which Redirect messages are sent.
A router must not update its routing tables upon receipt of a Redirect.
Redirect Function

- On-link destination
- Off-link destination

IPv6 Datagram
IPv6 Src Addr = A
IPv6 Dst Addr = B
Redir Target Addr = B
Redir Dst Addr = B

IPV6 Datagram
IPv6 Src Addr = R1
IPv6 Dst Addr = A
Redir Target Addr = R2
Redir Dst Addr = B

IPV6 Datagram
IPv6 Src Addr = R1
IPv6 Dst Addr = A
Redir Target Addr = B
Redir Dst Addr = B

IPV6 Datagram
IPv6 Src Addr = R1
IPv6 Dst Addr = A
Redir Target Addr = R2
Redir Dst Addr = B

IPV6 Datagram
IPv6 Src Addr = A
IPv6 Dst Addr = B

IPV6 Datagram
IPv6 Src Addr = A
IPv6 Dst Addr = B

IPV6 Datagram
IPv6 Src Addr = A
IPv6 Dst Addr = B

Path MTU Discovery

- The Path MTU is the minimum link MTU of all the links in a path between a source node and a destination node.
- Path MTU Discovery is the process by which a node learns the PMTU of a path.
- The basic idea is that a source node initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path.
- If any of the packets sent on that path are too large to be forwarded by some node along the path, that node will discard them and return ICMPv6 Packet Too Big messages.
- Upon receipt of such a message, the source node reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message.
- The Path MTU Discovery process ends when the node's estimate of the PMTU is less than or equal to the actual PMTU.

Stateless Address Autoconfiguration

Overview

- IPv6 defines both a stateful and stateless address autoconfiguration mechanism.
- Stateless autoconfiguration requires:
  - no manual configuration of hosts
  - minimal (if any) configuration of routers
  - no additional servers
- The stateless mechanism allows a host to generate its own addresses using a combination of:
  - Locally available information (interface identifier)
  - Information advertised by routers (link prefixes)
- In the absence of routers, a host can only generate link-local addresses. Link-local addresses are sufficient for allowing communication among nodes attached to the same link.
Stateless Address Autoconfiguration

Overview
- In the stateful autoconfiguration model hosts obtain from a server:
  - Interface addresses
  - Configuration information and parameters.
  - Both.
- Stateful servers maintain a database that keeps track of which addresses have been assigned to which hosts.
- Stateless and stateful autoconfiguration complement each other.
- The stateless approach is used when a site is not particularly concerned with the exact addresses hosts use, so long as they are unique and properly routable.
- The stateful approach is used when a site requires tighter control over exact address assignments.
- Both stateful and stateless address autoconfiguration may be used simultaneously.

Stateless Address Autoconfiguration

Address Leasing
- IPv6 addresses are leased to an interface for a fixed (possibly infinite) length of time.
- Each address has an associated lifetime that indicates how long the address is bound to an interface.
- When a lifetime expires, the address may be reassigned to another interface elsewhere.
- To handle the expiration of address bindings gracefully, an address goes through two distinct phases while assigned to an interface:
  - Initially, it is "preferred" (its use in arbitrary communication is unrestricted).
  - Later, an address becomes "deprecated" in anticipation that its current interface binding will become invalid.
- A deprecated address should be used only by applications that have been using it and would have difficulty switching to another address without a service disruption.

Stateless Address Autoconfiguration

Design Goals
- Stateless autoconfiguration is designed with the following goals in mind:
  - Manual configuration of individual machines before connecting them to the network should not be required.
  - Small sites consisting of a set of machines attached to a single link should not require the presence of a stateful server or router as a prerequisite for communicating.
  - A large site with multiple networks and routers should not require the presence of a stateful address configuration server.
  - Address configuration should facilitate the graceful renumbering of a site's machines.
  - System administrators need the ability to specify whether stateless autoconfiguration, stateful autoconfiguration, or both should be used.

Stateless Address Autoconfiguration

Address States
- Valid address: A preferred or deprecated address.
- Deprecated address: An address assigned to an interface whose use is discouraged, but not forbidden. A deprecated address should no longer be used as a source address in new communications.
- Preferred address: An address assigned to an interface whose use by upper layer protocols is unrestricted.
- Invalid address: An address that is not assigned to any interface.
- Tentative address: An address whose uniqueness on a link is being verified, prior to its assignment to an interface.
- Preferred address: An address assigned to an interface whose use by upper layer protocols is unrestricted.
- Deprecation period should be long enough that most, if not all, communications using the new address at the time an address becomes invalid.

Stateless Address Autoconfiguration

Site Renumbering
- Address leasing facilitates site renumbering by providing a mechanism to time-out addresses assigned to interfaces in hosts.
- Dividing valid addresses into preferred and deprecated categories provides a way of indicating to upper layers that a valid address may become invalid shortly and that future communication using the address will fail.
- To avoid this scenario, higher layers should use a preferred address to increase the likelihood that an address will remain valid for the duration of the communication.
- The depreciation period should be long enough that most, if not all, communications using the new address at the time an address becomes invalid.
Example / Zebra RA Config

```
interface eth0
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 60
  ipv6 nd prefix-advertisement 3ffe:38e1:100:2001::/64 529200 604800 onlink autoconfig

interface eth1
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 60
  ipv6 nd prefix-advertisement 3ffe:38e1:100:7001::/64 2592000 604800 onlink autoconfig

interface eth2
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 60
  ipv6 nd prefix-advertisement 3ffe:38e1:100:4001::/64 2592000 604800 onlink autoconfig
```

Duplicate Address Detection Example: Successful Assignment

```
A few seconds later:
inet6 fe80::250:56ff:fe8a:0/10 scope link
inet6 3ffe:3800:fffb:8001:1:1/64 scope global
A's address state:
inet6 fe80::250:56ff:fe8a:0/10 scope link
inet6 3ffe:3800:fffb:8001:1/64 scope global tentative

```

Duplicate Address Detection Example: Duplicated Assignment

```
B's address state:
inet6 fe80::250:56ff:fecc:a8ec prefixlen 64 scopeid 0x1
inet6 3ffe:3800:fffb:8001::1 duplicate
```

Stateless Address Autoconfiguration Example

```
New A Unicast Address: 3ffe:3800:fffb:8001:250:56ff:fe8a:0/64
```

Multicast Listener Discovery

- Enables each IPv6 router to discover the presence of multicast listeners on its attached links.
- Discovers specifically which multicast addresses are of interest to those neighboring nodes.
- This information is then provided to whichever multicast routing protocol is being used.
- There are three types of MLD Messages:
  - Multicast Listener Query (ICMPv6 Type 130)
    - General
    - Multicast-Address-Specific Query
  - Multicast Listener Report (ICMPv6 Type 131)
  - Multicast Listener Done (ICMPv6 Type 132)
Multicast Listener Discovery Message Format

- MLD is a sub-protocol of ICMPv6, messages have the following format:

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Response Delay</th>
<th>Reserved</th>
</tr>
</thead>
</table>

Multicast Address

IP Fields:
- Source Address: Must be a link-local address assigned to the interface from which this message is sent.
- Hop Limit: 1
- Router Alert in a Hop-by-Hop Options header.

ICMPv6 Fields:
- Maximum Response Delay: Meaningful only in Query messages. Specifies the maximum allowed delay before sending a responding Report (milliseconds).
- Multicast Address: Set to a specific IPv6 multicast address in a Multicast-Address-Specific Query, Report or Done Message.

Multicast Listener Discovery Message Destinations

<table>
<thead>
<tr>
<th>Message Type</th>
<th>IPv6 Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Query</td>
<td>Link-scope all-nodes (FF02::1)</td>
</tr>
<tr>
<td>Multicast-Address-Specific Query</td>
<td>The multicast address being queried</td>
</tr>
<tr>
<td>Report</td>
<td>The multicast address being reported</td>
</tr>
<tr>
<td>Done</td>
<td>Link-scope all-routers (FF02::2)</td>
</tr>
</tbody>
</table>

Multicast Listener Discovery Node Behavior

- A node may be in one of three possible states with respect to any single IPv6 multicast address on any single interface:
  - Non-Listener: when the node is not listening to the address on the interface.
  - Delaying Listener: when the node is listening to the address on the interface and has a report delay timer running for that address.
  - Idle Listener: when the node is listening to the address on the interface and does not have a report delay timer running for that address.

Multicast Listener Discovery Node State Transition Diagram

- There are seven possible actions that may be taken in response to the events:
  - Send report: for the address on the interface. The Report message is sent to the address being reported.
  - Send done: for the address on the interface. If the flag saying we were the last node to report is cleared, this action may be skipped. The Done message is sent to the link-scope all-routers address (FF02::2).
  - Set flag: since we were not the last node to send a report for this address.
  - Clear flag: since we were not the last node to send a report for this address.
  - Start timer: for the address on the interface, using a delay value chosen uniformly from the interval [0, Maximum Response Delay], where Maximum Response Delay is specified in the Query. If this is an unsolicited Report, the timer is set to a delay value chosen uniformly from the interval [0, [Unsolicited Report Interval]].
  - Reset timer: for the address on the interface to a new value, using a delay value chosen uniformly from the interval [0, Maximum Response Delay], as described in "start timer".
  - Stop timer: for the address on the interface.
A router may be in one of two possible states with respect to any single attached link:
- **Querier**: when this router is designated to transmit MLD Queries on this link.
- **Non-Querier**: when there is another router designated to transmit MLD Queries on this link.

The following three events can cause the router to change states:
- **Query timer expired**: occurs when the timer set for query transmission expires. This event is significant only when in the Querier state.
- **Query received from a router with a lower IP address**: occurs when a valid MLD Query is received from a router on the same link with a lower IPv6 Source Address.
- **Other querier present timer expired**: occurs when the timer set to note the presence of another querier with a lower IP address on the link expires. This event is significant only when in the Non-Querier state.

There are three actions that may be taken in response to the events:
- **Start general query timer**: for the attached link to [Query Interval].
- **Start other querier present timer**: for the attached link to [Other Querier Present Interval].
- **Send general query**: on the attached link. The General Query is sent to the link-scope all-nodes address (FF02::1), and has a Maximum Response Delay of [Query Response Interval].

To keep track of which multicast addresses have listeners, a router may be in one of three possible states with respect to any single IPv6 multicast address on any single attached link:
- **No Listeners Present**: when there are no nodes on the link that have sent a Report for this multicast address. This is the initial state for all multicast addresses on the router.
- **Listeners Present**: when there is a node on the link that has sent a Report for this multicast address.
- **Checking Listeners**: when the router has received a Done message but has not yet heard a Report for the identified address.

There are seven possible actions that may be taken in response to the events:
- **Start timer** for the address on the link.
- **Start timer** for the address on the link - this alternate action sets the timer to the minimum of its current value and either [Last Listener Query Interval] * [Last Listener Query Count] if this router is a Querier, or the Maximum Response Delay in the Query message * [Last Listener Query Count] if this router is a non-Querier.
- **Start retransmit timer** for the address on the link.
- **Clear retransmit timer** for the address on the link.
- **Send multicast-address-specific query** for the address on the link.
- **Notify routing +** internally notify the multicast routing protocol that there are listeners to this address on this link.
- **Notify routing -** internally notify the multicast routing protocol that there are no longer any listeners to this address on this link.
Summary / MLD

- Discover Multicast Listeners
- Discover Multicast Groups -> Inform to the MRP
- Messages: Query / Report / Done
- Node States: Non Listener / Delaying Listener / Idle Listener
- Router States: Querier / Non-Querier

IPv6 over Ethernet

- IPv6 packets are transmitted in standard Ethernet frames.
- The Ethernet header contains:
  - Destination and Source Ethernet addresses.
  - Ethernet type code (86DD hexadecimal).
- The data field contains:
  - The IPv6 header.
  - The payload.
  - Padding octets to meet the minimum frame size for the Ethernet link (if needed).
- The Maximum Transmission Unit (MTU) for Ethernet is 1500 octets. This size may be reduced by:
  - A Router Advertisement.
  - Manual configuration of each node.

IPv6 over Ethernet / Stateless Autoconfiguration

- The Interface Identifier for an Ethernet interface is based on the EUI-64 identifier derived from the interface’s built-in 48-bit IEEE 802 address.
- The Interface Identifier is then formed from the EUI-64 by complementing the “Universal/Local” (U/L) bit, which is the next-to-lowest order bit of the first octet of the EUI-64.
- For example, the Interface Identifier for an Ethernet interface whose built-in address is, in hexadecimal,
  34-56-78-9A-BC-DE
  would be
  36-56-78-FF-9A-BC-DE
- An IPv6 address prefix used for stateless autoconfiguration of an Ethernet interface must have a length of 64 bits.
IPv6 over Ethernet / Link-local Address

- The IPv6 link-local address for an Ethernet interface is formed by appending the Interface Identifier, to the prefix FE80::/64.

Example:
- EUI-48 Ethernet Address: 00:50:56:d9:88:3f
- EUI-64 Ethernet Address: 00:50:56:ff:fe:d9:88:3f
- Interface Identifier: 02:50:56:ff:fe:d9:88:3f
- Link-local Address: fe80::250:56ff:fed9:883f

IPv6 over Ethernet / Unicast address mapping

- The Neighbor Discovery Source/Target Link-layer Address option has the following form when the link layer is Ethernet.

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Ethernet Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>8 bits</td>
<td>48 bits</td>
</tr>
</tbody>
</table>

Examples:
- Type: 1 for Source Link-layer address
- Length: 1 (in units of 8 octets)
- Ethernet Address: The 48 bit Ethernet IEEE 802 address, in canonical bit order

IPv6 over Ethernet / Multicast address mapping

- An IPv6 packet with a multicast destination address DST, consisting of the sixteen octets DST[1] through DST[16], is transmitted to the Ethernet multicast address whose first two octets are the value 3333 hexadecimal and whose last four octets are the last four octets of DST.

Example:
- IPv6 Solicited-Node Multicast Address: ff02:1::1:fffd:883f
- Ethernet Link-Layer Multicast Address: 33:33:ff:d9:88:3f
- IPv6 All nodes Multicast Address: ff02:1::1
- Ethernet Link-Layer Multicast Address: 33:33:00:00:00:1

IPv6 over PPP

- The Point-to-Point Protocol (PPP) has three main components:
  - A method for encapsulating datagrams over serial links.
  - A Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection.
  - A family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols.

- The NCP for establishing and configuring the IPv6 over PPP is called the IPv6 Control Protocol (IPv6CP).

IPv6 over PPP / Link-local Address

- The interface identifier may be selected using one of the following methods:
  - If an IEEE global identifier is available anywhere on the node, then that address should be used.
  - If an IEEE global identifier is not available, then a different source of uniqueness, such as a machine serial number, should be used.
  - If a good source of uniqueness cannot be found, a random number should be generated.

The resulting link-local address is:
IP v6 over PPP

IP v6 Configuration Options

- IP v6 Configuration Options allow negotiation of desirable IP v6 parameters.
- Current values are assigned as follows:
  1: Interface-Identifier.
  - This Configuration Option provides a way to negotiate a unique 64-bit interface identifier to be used for the address autoconfiguration at the local end of the link.
  2: IPv6-Compression-Protocol.
  - This Configuration Option provides a way to negotiate the use of a specific IPv6 packet compression protocol. The IPv6-Compression-Protocol Configuration Option is used to indicate the ability to receive compressed packets. Each end of the link must separately request this option if bi-directional compression is desired. By default, compression is not enabled.

IP v6 DNS Extensions

AAA Record Type

- The AAAA record type is a new record specific to the Internet class that stores a single IPv6 address.
- A 128 bit IPv6 address is encoded in the data portion of an AAAA resource record in network byte order (high-order byte first).
- An AAAA query for a specified domain name in the Internet class returns all associated AAAA resource records in the answer section of a response.
- A type AAAA query does not perform additional section processing.
- The textual representation of the data portion of the AAAA resource record used in a master database file is the textual representation of a IPv6 address.

IP v6 DNS Extensions

A6 Resource Record

- The A6 RR contains two or three fields:
  A prefix length.
  An IPv6 address suffix.
  A name of the prefix.
  The domain name component shall not be present if the prefix length is zero.
  The address suffix component shall not be present if the prefix length is 128.
- It is suggested that an A6 record intended for use as a prefix for other A6 records have all the insignificant trailing bits in its address suffix field set to zero.
- Example (of textual representation):
  $ORIGIN example2.net.
  subnet5 A6 48 0:0:0:1:: ipv6net2.example2.net.
  ipv6net2 A6 0:6666:5555:4::
IP v6 DNS Extensions

Binary Labels

- A “Bit-String Label” may appear within domain names.
- Represents a sequence of “One-Bit Labels”.
- Enables RRs to be stored at any bit-boundary in a binary-named section of the domain name tree.
- Are intended to efficiently solve the problem of storing data and delegating authority on arbitrary boundaries (for reverse zones).

Textual Representation example:

```
[b11010000011101]
[o64072/14]
[xd074/14]
[208.116.0.0/14]
[b11101]
[o640]
[x1d]
[o64]
[o35].208.0.0.0/8
```

IP v6 DNS Extensions

IP6.APA Domain

- A new special domain is defined to look up a record given an address.
- The domain is rooted at IP6.APA.
- This new scheme for reverse lookups relies on Binary Labels.
- The inverse lookup domain name corresponding to the address 4321:0:1:2:3:4:567:89ab would be [x43210000001000200000004056789ab].IP6.APA.
- DNS address space delegation is implemented not by zone cuts and NS records, but by the new DNAME resource record.

IP v6 DNS Extensions

Non-Terminal DNS Name Redirection

- A new RR called “DNAME” provides the capability to map an entire subtree of the DNS name space to another domain.
- It's a solution to the problem of maintaining address-to-name mappings in a context of network renumbering.

Renumbering Example:

```
$ORIGIN [x20aa00bbcccc/48].ip6.arpa.
$ORIGIN [x266655550004/48].ip6.arpa.
$ORIGIN ipv6-rev.example.com.
[x123456781215675/64]    PTR    host.example.com.
```

IP v6 DNS Extensions

IP6.APA Domain Example

```
\[x234500/24\]   DNAME   IP6.ALPHA-TLA.ORG.
\[xd/4\].IP6.ALPHA-TLA.ORG.  DNAME  IP6.C.NET.
\[x0E/8\].IP6.ALPHA-TLA.ORG.  DNAME  IP6.D.NET.
\[x1CA/12\].IP6.C.NET.  DNAME  IP6.A.NET.
\[x2DA/12\].IP6.D.NET.  DNAME  IP6.A.NET.
\[xEB/8\].IP6.E.NET.    DNAME  IP6.B.NET.
\[x11/8\].IP6.A.NET.    DNAME  X.EXAMPLE.COM.
\[x22/8\].IP6.B.NET.    DNAME  X.EXAMPLE.COM.
```

IP v6 DNS Extensions

Modifications to existing Query Types

- All existing query types that perform type A additional section processing, must be redefined to perform type A, A6 and AAAA additional section processing, i.e.:
  - Name server (NS)
  - Mail exchange (MX)
  - Mailbox (MB)

These new definitions mean that a name server must add any relevant IPv4 addresses and any relevant IPv6 addresses available locally to the additional section of a response when processing any one of the above queries.

Transition Mechanisms

Dual IP Stacks

- Is the simplest mechanism for IPv4 and IPv6 coexistence.
- Node has both IPv4 and IPv6 stacks and addresses.
- DNS Resolver returns IPv6, IPv4 or both to application.
- IPv6 applications can communicate with IPv4 nodes.
Transition Mechanisms

Tunneling IPv6 in IPv4

- IPv6 encapsulated in IPv4
- Four possible configurations:
  - Router-to-Router
  - Host-to-Router
  - Host-to-Host
  - Router-to-Host
- The tunnel endpoints takes care of the encapsulation. This process is "transparent" to the other nodes.
- The manner in which endpoints addresses are determined defines:
  - Configured tunnels
  - Automatic tunnels
  - Multicast tunnels

Configuration Tunneling

- Tunnel endpoints are fixed (manually configured).
- Tunnel endpoints must be dual-stack nodes.
  - The IPv4 address is the endpoint for the tunnel.
  - Require reachable IPv4 addresses.
- The tunnels can be either unidirectional or bidirectional.
  - Bidirectional configured tunnels behave as virtual point-to-point links.

Automatic Tunneling

- IPv4 tunnel endpoint address is determined from the IPv4-compatible destination IPv6 address.
  - Example: ::170.210.79.4
- Terminates on a host.
- Routing table redirects ::/96 to automatic tunneling interface.
- If two hosts have IPv4-compatible IPv6 addresses, they can communicate across an IPv4 infrastructure using automatic tunneling.
- A dual router, upon receiving an IPv6 packet destined for a host with an IPv4-compatible address, can automatically tunnel that packet to its endpoint.
### Transition Mechanisms

#### Automatic Tunneling: Router-to-Host

- **R1**: IPv6 router with IPv4-compatible address
- **IPv6**: Entry router decapsulates IPv6 packet in IPv4
- **IPv4**: IPv4 router with IPv6-compatible address
- **H1**: Destination host decapsulates IPv6 packet

#### Multicast Tunneling: 6over4

- Interconnection of isolated IPv6 domains in an IPv4 world.
- No explicit tunnels.
- The egress router must:
  - Have a dual stack
  - Have a globally routable IPv4 address
  - Have an IPv4 multicast infrastructure
  - Implement 6over4 on an external interface
- Uses IPv4 as a link layer for IPv6, that’s why IPv4 multicast is needed.

### Transition Routing

#### Terms related to transition routing architecture:

- **Border router**: A router that forwards packets across routing domain boundaries.
- **Routing domain**: A collection of routers that coordinate routing knowledge using a single protocol.
- **Routing region**: Collection of routers, interconnected by a single Internet protocol, that coordinate their routing knowledge using routing protocols from a single IP stack. A routing region may be a superset of a routing domain.
- **Reachability information**: Information describing the set of reachable destinations that can be used for packet forwarding decisions.
- **Route leaking**: Advertisement of network layer reachability information across routing boundaries.

#### Routing Example (1)

- **Region A**: IPv6/v4 routers
- **Region B**: IPv4-only routers

#### Routing Example (2)

- **Region A**: IPv6/v4 routers
- **Region B**: IPv4-only routers

#### Routing Example (3)

- **Region A**: IPv6/v4 routers
- **Region B**: IPv4-only routers
IPv6 Transition Routing Example (4)

Region A: IPv6/v4 routers
Region B: IPv4-only routers

IPv6 Transition Routing Example (5)

Region A: IPv6/v4 routers
Region B: IPv4-only routers

6to4
- Mechanism for IPv6 sites to communicate with each other over the IPv4 network without explicit tunnel setup.
- Allows communication with native IPv6 domains.
- Assigns an interim unique IPv6 address prefix to any site that currently has at least one globally unique IPv4 address.
- Not requires:
  - IPv4-compatible IPv6 addresses
  - configured tunnels
- Uses the prefix 2002::/16 to form 6to4 prefixes derived from the IPv4 Address.

6to4 – Terminology
- Requires an IPv4 network communicating both 6to4 routers.
- 6to4 address: an IPv6 address constructed using a 6to4 prefix.

6to4 – Scenario: All sites work the same
- Requires an IPv4 network communicating both 6to4 routers.
- Each site has an IPv6 prefix in the form 2002:WWXX:YYZZ::/48.
- Outgoing packets are encapsulated into IPv4 at the 6to4 router.
- Incoming packets are decapsulated and sent to the internal IPv6 network.
- Any number of 6to4 sites can interoperate with no tunnel configuration.

Transition Routing Summary

<table>
<thead>
<tr>
<th>Host A</th>
<th>Host B</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>v4-compatible address no local v6 router</td>
<td>v4-compatible address no local v6 router</td>
<td>host to host tunneling in both directions</td>
</tr>
<tr>
<td>v4-compatible address no local v6 router</td>
<td>v4-compatible address local v6 router</td>
<td>A-+B: host to host tunnel B--&gt;A: v6 forwarding plus router-&gt;host tunnel</td>
</tr>
<tr>
<td>v4-compatible address no local v6 router</td>
<td>incompatible address local v6 router</td>
<td>A-+B: host to router tunnel plus v6 forwarding B--&gt;A: v6 forwarding plus router to host tunnel</td>
</tr>
<tr>
<td>6to4 address Any IPv6 address</td>
<td>6to4</td>
<td>end to end native v6 in both directions</td>
</tr>
</tbody>
</table>

IPv4 Site A IPv4 Site B IPv6 Site A Wide Area IPv4 Network IPv6 Site B
**IPv6**

**SIIT: Stateless IP/ICMP Translation**
- Allows IPv6-only hosts to talk to IPv4 hosts.
- Header translator maps corresponding header fields of IPv4 to IPv6.
- Requires one temporary IPv4 address per host.
- Problem: if no corresponding fields/infos in both headers ➔ no translation possible.
- Conclusion: except segmentation no usage of IPv6 extension headers.

**IPv4 Network**

**IPv6/v4 Translator**

**IPv4 Network**

**IPv6**

---

**NAT-PT: Network Address Translation – Protocol Translation**
- Enables communication between pure IPv6 and IPv4 nodes.
- Combines two techniques: NAT (Network Address Translation) and SIIT Protocol Translation. Uses Stateful Translation.
- Requires at least one IPv4 address per site.
- Operation:
  - IPv6 node sends packet to NAT-PT server with special destination address.
  - NAT-PT server manages pool of IPv4 addresses, translates headers: IPv4 to IPv6. Assigns IPv4 address to IPv6 address, forwards packet to IPv4 node.
  - IPv4 node: first IPv4 address of IPv6 node has to be received from DNS. DNS server requests NAT-PT to assign and delivers reserved IPv4 address of IPv6 node.

**IPv4 Address**

**IPv6 Address**

---

**Traditional NAT-PT**
- Traditional NAT-PT would allow hosts within a V6 network to access hosts in the V4 network.
- In a traditional NAT-PT, sessions are unidirectional, outbound from the V6 network.
- This is in contrast with Bi-directional NAT-PT, which permits sessions in both inbound and outbound directions.
- There are two variations to traditional NAT-PT:
  - Basic NAT-PT: a block of V4 addresses are set aside for translating addresses of V6 hosts as they originate sessions to the V4 hosts in external domain.
  - NATPT, which stands for “Network Address Port Translation + Protocol Translation”, would allow V6 nodes to communicate with the V4 nodes transparently using a single V4 address.

**IPv4 Network**

**IPv4 Address**

**IPv6 Network**

**IPv6 Address**

---

**IPv4 Network**

**IPv6 Network**

**IPv6 Address**

---

**Bump in the Stack**
- IPv4 applications can transparently communicate via an IPv6 net (if application uses logical names and DNS service).
- Inserts 3 additional modules into IPv4 protocol stack (Dual Stack Host).
- Operation:
  - Translator maps IPv4 packets into IPv6 packets using protocol translation (SIIT).
  - Extension name resolver creates DNS requests (A-rec. + AAAA-rec.) upon appl. DNS request.
  - If DNS server replies A-rec., this is guided directly to IPv4 application.
  - If DNS server replies only AAAA-record, address mapper reserves IPv4 address.
  - Then A-rec. is derived (from reserved IPv4 address and AAAA-rec.) and given to application.
  - Address mapper manages pool of IPv4 addresses and assigned IPv6 addresses.

**IPv4 Address**

**IPv6 Address**

---

**IPSec – Network Security**
- IPSec is designed to provide interoperable, high quality, cryptographically-based security for IPv4 and IPv6.
- IPSec provides security at the IP layer, transparent to applications.
- IPSec offers services for:
  - Authentication: Authenticate the sender.
  - Confidentially: Encrypt data before transmission.
  - Data Integrity: Detect altered data in packets.
  - Anti-Replay: Detect replayed packets.
- Open standard, published by the IETF.
IPSec – Protocols

- IPSec uses two protocols to provide traffic security.
  - Authentication Header (AH)
    - Connectionless integrity
    - Data origin authentication
    - Anti-replay service (optional)
  - Encapsulating Security Payload (ESP)
    - Connectionless integrity
    - Data origin authentication
    - Anti-replay service (optional)
    - Confidentiality (encryption)
    - Limited traffic flow confidentiality
  - Two modes of use: transport or tunnel.

IPSec – Security Associations

- The concept of a “Security Association” is fundamental to IPSec.
  - A Security Association (SA) is a simplex “connection” that affords security services to the traffic carried by it.
  - A Security Association is unidirectional.
  - A Security Association is identified by a triple consisting of:
    - Security Parameter Index (SPI)
    - IP Destination Address
    - Security protocol identifier (AH or ESP).

IPSec – Security Databases

- There are two nominal databases in this model:
  - Security Policy Database
    - Specifies the policies that determine the disposition of all IP traffic inbound or outbound from an IPSec implementation.
    - An SPD must discriminate among traffic that is afforded IPSec protection and traffic that is allowed to bypass IPSec.
  - Security Association Database
    - Contains parameters that are associated with each security association.
    - Selector: a set of IP and upper layer protocol field values that is used by the SPD to map traffic to a policy, i.e., an SA.

IPSec – Basic Overview

Case 1: SA already available.

IPSec packet processing:
- Look up in the Security Association List (SA List) if a Security Association (SA) is available, i.e. if a secure transmission is possible.
  - SA stores information about Authentication and/or Encryption algorithm and symmetric, shared keys.
**IP v6**

**IPSEC – Basic Overview**

**Case 1: SA already available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- Authentication
- Encryption
- SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE
- Secure key exchange possible?

**IPSec packet processing:**
- Internet Key Exchange (IKE)
  - Create a IKE SA using public keys
  - Exchange shared keys for IP Sec

- Negotiate algorithms and keys for IP Sec

- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA

**Case 2: SA not available.**

- Host
- Internet
- TCP
- MAC
- IPSec
- SPD
- IKE SA
IPSec – Basic Overview

Case 2: SA not available.

Host → Internet → Host

IPSec  SP D
MAC

IP v6
IP Sec
–
Basic Overview

IP v6
IP Sec
–
Supported Combinations

Host to Host

Internet

SP D
MAC

IP_A H_payload (transport)
IP_ESP_payload (transport)
IP_A H_ESP_payload (transport)
IP (host)_AH_IP_payload (tunnel)
IP (host)_ESP_IP_payload (tunnel)

IP v6
IP Sec
–
Supported Combinations

Security Gateway to Security Gateway

Internet

SP D
MAC

IP (SG)_AH_IP_payload (tunnel)
IP (SG)_ESP_IP_payload (tunnel)

IP v6
IP Sec
–
Supported Combinations

Combination of Cases

Internet

Two options (tunnel)
Three options (transport)
Two options (tunnel)

IP v6
IP Sec
–
Supported Combinations

Remote Access

Internet

Two options (tunnel)
Three options (transport)
Two options (tunnel)

IP v6
IP Sec
–
Authentication header

- Authentication of data origin
- Data integrity
- Anti-replay (optional)

<table>
<thead>
<tr>
<th>SPI</th>
<th>Security Parameters Index (SPI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq. Number Field</td>
<td></td>
</tr>
<tr>
<td>Authentication Data (variable)</td>
<td></td>
</tr>
</tbody>
</table>

- SPI = 0 is forbidden, 1..255 is reserved
- Seq. Number only increases (no reset to 0) for anti-replay
**IPSec – Encapsulation Security Payload**

- Data integrity
- Data encryption
- Authentication (optional)
- Anti-replay (optional)

<table>
<thead>
<tr>
<th>SPI</th>
<th>Sequence Number Field</th>
<th>Payload Data (variable)</th>
<th>Authentication Data (variable)</th>
</tr>
</thead>
</table>

- SPI = 0 is forbidden, 1..255 is reserved
- Seq. Number only increases (no reset to 0) for anti-replay

**IPSec – AH Transport Mode**

Before applying AH

| IP Header | Optional headers | TCP, UDP, ICMP, etc. | Data |

After applying AH

| Original IP Header | Optional headers(*) | AH | Optional headers(**) | TCP, UDP, ICMP, etc. | Data |

Authenticated except for mutable fields

[**]: Dest. Opt

**IPSec – AH Tunnel Mode**

Before applying AH

| IP Header | Optional headers | TCP, UDP, ICMP, etc. | Data |

After applying AH

| New IP Header | New optional headers | AH | Original IP Header | Optional headers | TCP, UDP, ICMP, etc. | Data |

Authenticated except for mutable fields in new IP hdr

[**]: Dest. Opt

**IPSec – ESP Transport Mode**

Before applying ESP

| IP Header | Optional headers | TCP, UDP, ICMP, etc. | Data |

After applying ESP

| Original IP Header | Optional headers[*] | ESP | Optional headers(**) | TCP, UDP, ICMP, etc. | Data |

Encrypted

Authenticated

[**]: Dest. Opt

**IPSec – ESP Tunnel Mode**

Before applying ESP

| IP Header | Optional headers | TCP, UDP, ICMP, etc. | Data |

After applying ESP

| New IP Header | New optional headers | ESP | Original IP Header | Optional headers | TCP, UDP, ICMP, etc. | Data |

Encrypted

Authenticated

[**]: Dest. Opt

**IPSec – AH-ESP Transport Mode**

Before applying AH-ESP

| IP Header | Optional headers | TCP, UDP, ICMP, etc. | Data |

After applying AH-ESP

| Original IP Header | Optional headers(*) | AH | Optional headers(**) | TCP, UDP, ICMP, etc. | Data |

Encrypted

Authenticated

Authenticated except for mutable fields

[**]: Dest. Opt
**Mobility Terms**

- **Home address**: permanent address of the mobile node.
- **Home subnet prefix**: prefix corresponding to the home address.
- **Foreign subnet prefix**: prefix of the foreign link.
- **Care-of address**: address assigned to the mobile node on the foreign link.
- **Binding**: the association of the home address of a mobile node with a care-of address.

---

**Mobility – Home Binding Procedure**

- The mobile node obtains its care-of address.
- The mobile node sends a Binding Update message to the home agent.
- The home agent replies by returning a Binding Acknowledgement message.
- The home agent intercepts packets destined to the mobile node and tunnels them to the care-of address. The mobile node reverse tunnels traffic destined to the correspondent node.

---

**Mobility – Communication without binding**

- **Sending Packets**
  - The packet is sent to the home agent using IPv6 encapsulation.
  - The home agent decapsulates the tunneled packet and forwards it towards the correspondent node.
- **Receiving Packets**
  - The correspondent node sends the packet to the home network.
  - The home agent intercepts the packet.
  - The home agent encapsulates the packet using IPv6 encapsulation and sends it to the Mobile node care-of address.

---

**Mobility – New IPv6 Protocol: Mobility Header**

- Four messages to perform the Return Routability Procedure
  - Home Test Init (HoTI)
  - Home Test (HoT)
  - Care-of Test Init (CoTI)
  - Care-of Test (CoT)
- Four messages to manage the bindings
  - Binding Update
  - Binding Acknowledgement
  - Binding Refresh Request
  - Binding Error

---

**Mobility – New ICMP Messages**

- Messages use in the dynamic home agent address discovery mechanism.
  - Home Agent Address Discovery Request
  - Home Agent Address Discovery Reply
- Messages used for network renumbering and address configuration on the mobile node
  - Mobile Prefix Solicitation
  - Mobile Prefix Advertisement
Mobile IPv6 defines a new IPv6 destination Option, the **Home Address** destination option.

This option is used in a packet sent by a mobile node while away from home, to inform the recipient of the mobile node’s home address.

Mobile IPv6 defines a new IPv6 destination Option, the **Home Address** destination option.

This option is used in a packet sent by a mobile node while away from home, to inform the recipient of the mobile node’s home address.

**Binding Cache**
- Maintained by each IPv6 node.
- A separate Binding Cache maintained for each of the node’s IPv6 addresses.
- When sending a packet, the BC is searched before the Destination Cache.
- Entries marked as “home registration” or “correspondent registration”

**Binding Update List**
- Maintained by each mobile node.
- Records information for each BU sent by the node.
- Includes bindings sent to:
  - Correspondent Nodes
  - Home Agent
  - Home Agent on a previous foreign link

**Home Agents List**
- Maintained by each home agent and each mobile node.
- Records information about each home agent from which this node has received a Router Advertisement win which the Home Agent bit is set.
- This list is similar to the Default Router List (Neighbor Discovery).
- Used by a Home Agent in the dynamic home agent discovery mechanism.
- Enables a node to notify a home agent on its previous foreign link.

**Node Keys**
- Each correspondent node has a secret key, Kcn.
- The node uses this key to verify the cookies.
- This key does not need to be shared with any other entity.
- A correspondent node generates Kcn each time it boots.
- Kcn consists of 20 octets.
Mobility – Nonces
- Each correspondent generates nonces at regular intervals.
- Generated using a random number generator.
- Each nonce is identified by a nonce index.
- A correspondent node may use the same Kcn and nonce with all the mobiles it is in communication with.
- Nonce is an octet string of any length. Recommended length is 64-bit.

Mobility – Cookies
- Cookies sent to the correspondent node.
- Generated Randomly.
- Used to verify that the response matches the request.
- HoT and CoT cookies
- Cookies sent to the mobile node. Produced cryptographically from nonces.
- Home Cookie
- Care-of Cookie

Mobility – Cryptographic Functions
- \( \text{MAC}_K(m) \)
  - Computed on message \( m \) with key \( K \).
  - HMAC SHA1.
- \( \text{Hash}(m) \)
  - Hash of message \( m \).
  - SHA1.

Mobility – Return Routability Procedure
- **Home Test Init (HoTI)**
  - Source: home address
  - Destination: correspondent
  - Parameters:
    - HoT cookie
- **Care-of Test Init (CoTI)**
  - Source: care-of address
  - Destination: correspondent
  - Parameters:
    - CoT cookie
- **Session Key**
  - \( K_{bu} = \text{hash}(\text{home cookie} \mid \text{care-of cookie}) \)
- **Generate home cookie**
  - \( \text{First64}(\text{MAC}_{Kcn}(\text{home address} \mid \text{home nonce})) \)
- **Care-of Test**
  - Source: correspondent
  - Destination: care-of address
  - Parameters:
    - CoT cookie
    - care-of cookie
    - care-of nonce index
- **Generate care-of cookie**
  - \( \text{First64}(\text{MAC}_{Kcn}(\text{care-of address} \mid \text{care-of nonce})) \)
- **Home Test**
  - Source: correspondent
  - Destination: home address
  - Parameters:
    - HoT cookie
    - home cookie
    - home nonce index
- **Generate home cookie**
  - \( \text{First64}(\text{MAC}_{Kcn}(\text{home address} \mid \text{home nonce})) \)

Mobility – Binding to a Correspondent Node
- **Binding Update**
  - Source: care-of address
  - Destination: correspondent
  - Parameters:
    - home address
    - MAC_Kbu(care-of address \mid correspondent node address \mid BU)
    - home nonce index
    - care-of nonce index
    - sequence number
    - (more fields, not security related)
- **Binding Acknowledgement**
  - Source: correspondent
  - Destination: care-of address
  - Parameters:
    - sequence number
    - MAC_Kbu(care-of address \mid correspondent node address \mid BA)

Mobility – Route Optimization
- **Sending Packets**
  - The packet is sent to the correspondent using the Home Address Destination Option. Source Address: care-of address.
  - The correspondent node swaps the IPv6 Source address and the Home Address Destination Option.
- **Receiving Packets**
  - The correspondent node sends the packet using the Routing header. IPv6 Dst. Address = care-of home address set in the routing header.
  - The mobile node receives the packet, swaps the Dst. Address and the Filt header. Resubmits the packet for IP processing.
Socket Interface Extensions for IPv6

**Motivation**
- While IPv4 addresses are 32 bits long, IPv6 interfaces are identified by 128-bit addresses.
- The socket interface makes the size of an IP address quite visible to an application.
- Those parts of the API that expose the addresses must be changed.
- IPv6 also introduces new features which must be made visible to applications via the API, e.g.:
  - Traffic class
  - Flow Label

**Design Considerations**
- The API changes should:
  - Provide both source and binary compatibility for programs written to the original API.
  - Be as small as possible in order to simplify the task of converting existing IPv4 applications to IPv6.
  - Be able to use this API to interoperate with both IPv6 and IPv4 hosts. Applications should not need to know which type of host they are communicating with.

**What Needs to be Changed**
- Core socket functions
  - These functions need not change for IPv6.
- Address data structures
  - A new IPv6-specific address data structure is needed.
- Name-to-address translation functions
  - New functions are defined to support IPv4 and IPv6.
  - The POSIX 1003.1g draft specifies a new nodename-to-address translation function which is protocol independent.
- Address conversion functions
  - New functions that convert both IPv4 and IPv6 addresses.
- Miscellaneous features
  - New interfaces to support the IPv6 traffic class, flow label, and hop limit header fields.
  - New socket options are needed to control the sending and receiving of IPv6 multicast packets.

**IPv6 Address Family and Protocol Family**
- New address family name: AF_INET6
  - Defined in `<sys/socket.h>`
  - New sockaddr_in6 data structure.
- New protocol family name: PF_INET6
  - Defined in `<sys/socket.h>`
  - Used in the first argument to the socket() function.

**IPv6 Address Structure**
- A new in6_addr structure holds a single IPv6 address:

  ```c
  struct in6_addr {
    uint8_t s6_addr[16]; /* IPv6 address */
  };
  ```

- IPv4

  ```c
  struct in_addr {
    u_long s_addr;
  }
  ```
Socket Interface Extensions for IPv6

The Socket Functions

Applications call the socket() function to create a socket descriptor that represents a communication endpoint.

```
IPv4
s = socket(PF_INET, SOCK_STREAM, 0); /* TCP Socket */
```

```
IPv6
s = socket(PF_INET6, SOCK_STREAM, 0); /* TCP Socket */
```

Once the application has created a PF_INET6 socket, it must use the sockaddr_in6 address structure when passing addresses in to the system.

```
bind()
connect()
sendmsg()
sendto()
```

The system will use the sockaddr_in6 address structure to return addresses to applications that are using PF_INET6 sockets.

The functions that return an address from the system to an application are:

```
accept()
recvfrom()
recvmsg()
getpeername()
getsockname()
```

No changes to the syntax of the socket functions are needed to support IPv6.

Socket Interface Extensions for IPv6

Compatibility with IPv4 Nodes

IPv6 applications are able to interoperate with IPv4 applications.

- Uses the IPv4-mapped IPv6 address format.
- IPv4-mapped addresses are written as follows: `::FFFF:<IPv4-address>`

Applications may use PF_INET6 sockets to:

- open TCP connections to IPv4 nodes
- send UDP packets to IPv4 nodes

Encoding the destination's IPv4 address as an IPv4-mapped IPv6 address.

When applications use PF_INET6 sockets to:

- accept TCP connections from IPv4 nodes
- receive UDP packets from IPv4 nodes

The system returns the peer’s address using a sockaddr_in6 structure encoded this way.

Socket Interface Extensions for IPv6

IPv6 Wildcard Address

While the bind() function allows applications to select the source IP address of UDP packets and TCP connections, applications often want the system to select the source address for them.

- With IPv4, one specifies the address as the symbolic constant INADDR_ANY.

In IPv6 a symbolic constant can be used to initialize an IPv6 address variable, but cannot be used in an assignment.

Systems provide the wildcard in two forms:

```
extern const struct in6_addr in6addr_any;
```

```
struct in6_addr anyaddr = IN6ADDR_ANY_INIT;
```

(can be used ONLY at declaration time)

Applications use in6addr_any similarly to the way they use INADDR_ANY in IPv4.

Socket Interface Extensions for IPv6

IPv6 Loopback Address

Applications may need to send UDP packets to, or originate TCP connections to, services residing on the local node.

In IPv4, they can do this by using the constant IPv4 address INADDR_LOOPBACK

The IPv6 loopback address is provided in two forms:

```
extern const struct in6_addr in6addr_loopback;
```

```
struct in6_addr loopbackaddr = IN6ADDR_LOOPBACK_INIT;
```

(can be used ONLY at declaration time)

Socket Interface Extensions for IPv6

Unicast Hop Limit

A new setsockopt() option controls the hop limit used in outgoing unicast IPv6 packets.

The name of this option is IPV6_UNICAST_HOPS, and it is used at the IPPROTO_IPV6 layer.

Example:

```
int hoplimit = 10;
if (setsockopt(s, IPPROTO_IPV6, IPV6_UNICAST_HOPS, (char *)&hoplimit, sizeof(hoplimit)) == -1)
    perror("setsockopt IPV6_UNICAST_HOPS");
```
IPv6 applications may send UDP multicast packets by simply specifying an IPv6 multicast address in the address argument of the sendto() function.

Three socket options at the IPPROTO_IPV6 layer control some of the parameters for sending multicast packets:

- **IPV6_MULTICAST_IF**: Set the interface to use for outgoing multicast packets. The argument is the index of the interface to use.
- **IPV6_MULTICAST_HOPS**: Set the hop limit to use for outgoing multicast packets.
- **IPV6_MULTICAST_LOOP**: If a multicast datagram is sent to a group to which the sending host itself belongs: 1: loop back a copy, 0: don’t loop back a copy.
- **IPV6_JOIN_GROUP**: Join a multicast group on a specified local interface.
- **IPV6_LEAVE_GROUP**: Leave a multicast group on a specified interface.