Introduction to IPv6
Sep 3 2019
Let’s begin!

Firsts, just few basic things about IPv6
IPv6

1998 – Defined by RFC 2460 (now RFC 8200)

- 128-bit addressing
- Simplified packet header
- Extension headers
- Data flow identification (QoS)
- Inclusion of IPSEC mechanisms in the protocol
- Packet fragmentation and reassembly performed at the source and destination
- Does not require the use of NAT: allows end-to-end connections
- Mechanisms to simplify network configuration
- ....
IPv6 addresses

- An IPv4 address is formed by 32 bits.
  \[2^{32} = 4.294.967.296\]
- An IPv6 address is formed by 128 bits.
  \[2^{128} = 340.282.366.920.938.463.463.374.607.431.768.211.456\]
  \(~ 5.6 \times 10^{28}\) IP addresses for each human being.
  \(~ 7.9 \times 10^{28}\) times the number of IPv4 addresses.
The IPv6 addresses representation divides an address in 8 groups of 16 bits, separated by “:”, represented by hexadecimal digits.

2001:0DB8:AD1F:25E2:CADE:CAFE:F0CA:84C1

When representing an IPv6 address it is allowed:
- To use lowercase or uppercase letters
- Omit leading zeros and ...
- Represent groups of contiguous zeros by “::”.

Example: 2001:0DB8:0000:0000:130F:0000:0000:140B
         2001:db8:0:0:130f::140b

Not valid format: 2001:db8::130f::140b (generates ambiguity)
Addresses

- Prefix representation
  - Like CIDR (IPv4)
    - “IPv6 address/prefix size”

- Example:
  - Prefix 2001:db8:3003:2::/64
  - Global Prefix 2001:db8::/32
  - Subnet ID 3003:2

- URL
  - http://[2001:12ff:0:4::22]/index.html
  - http://[2001:12ff:0:4::22]:8080
Addresses

Three types of addresses have been defined for IPv6:

- **Unicast** → Individual Identification
- **Anycast** → Selective Identification
- **Multicast** → Group Identification

Broadcast addresses don’t exist anymore.

IPv6 is a power user of anycast 😊

Please note this one!
Doing well?. Nice 😊

Now move on to the famous—and very important—IPv6 header
Doing well?. Nice 😊

Now move on to the famous –and very important- IPv6 header

Ok, not that fast !!
.... let’s recap the IPv4 header, good idea
IPv4 Header

<table>
<thead>
<tr>
<th>Versão (Version)</th>
<th>Tamanho do Cabeçalho (IHL)</th>
<th>Tipo de Serviço (ToS)</th>
<th>Tamanho Total (Total Length)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flags</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deslocamento do Fragmento (Fragment Offset)</td>
<td></td>
</tr>
<tr>
<td>Identificação (Identification)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tempo de Vida (TTL)</td>
<td>Protocolo (Protocol)</td>
<td>Soma de verificação do Cabeçalho (Checksum)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It's integrated by 12 fixed fields. It may contain options or not, therefore it's size can vary from 20 to 60 Bytes.
IPv6 Header

- Simpler
  - 40 Bytes (fixed size)
  - Only twice bigger than the older version
- More flexible
  - Extensions through additional headers
- More efficient
  - Header overhead is minimized
  - Packet processing cost is reduced
IPv6 Header

- Six fields from the IPv4 header were removed
### IPv6 Header

- Six fields from the IPv4 header were removed
- Four fields had its name and its location changed

<table>
<thead>
<tr>
<th>IPv4 Header Field</th>
<th>IPv6 Header Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (Version)</td>
<td>Version (Version)</td>
</tr>
<tr>
<td>Type of Service (ToS)</td>
<td>Traffic Class (Class of Service)</td>
</tr>
<tr>
<td>Total Length</td>
<td>Flow Label (Flow Label)</td>
</tr>
<tr>
<td>Identification</td>
<td>Payload Length</td>
</tr>
<tr>
<td>Flags</td>
<td>Next Header (Protocol)</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>Hop Limit</td>
</tr>
</tbody>
</table>

**IPv4 Header**

- Identification
- Flags
- Fragment Offset
- Protocol
- Total Length
- Time to Live (TTL)

**IPv6 Header**

- Version
- Traffic Class
- Flow Label
- Payload Length
- Next Header
- Hop Limit

*Endereço de Origem (Source Address)*

*Endereço de Destino (Destination Address)*
IPv6 Header

- Six fields from the IPv4 header were removed
- Four fields had it’s name and it’s location changed
- The Flow Label field was enlarged
IPv6 Header

- Six fields from the IPv4 header were removed
- Four fields had its name and its location changed
- The Flow Label field was enlarged
- Three fields were unchanged
### IPv6 Header

<table>
<thead>
<tr>
<th>Versão (Version)</th>
<th>Classe de Tráfego (Traffic Class)</th>
<th>Identificador de Fluxo (Flow Label)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tamanho dos Dados (Payload Length)</td>
<td>Próximo Cabeçalho (Next Header)</td>
<td>Limite de Encaminhamento (Hop Limit)</td>
</tr>
</tbody>
</table>

**Endereço de Origem (Source Address)**

**Endereço de Destino (Destination Address)**
Ok, move on and now talk about the **core** of IPv6
Ok, move on and now talk about the **core** of IPv6

If you don’t understand the following there is no chance you can deploy & troubleshoot IPv6
Neighbor Discovery

- Defined in RFC 4861.
- It assumes IPv4 functionalities such as ARP, ICMP Router Discovery and ICMP.
- It adds new methods and mechanisms that didn’t exist in IPv4.
- Optimize some of the processes for network configuration:
  - Determining MAC addresses of hosts in the network.
  - Finding neighboring routers.
  - Determining prefixes and other information for network configuration.
  - Detecting overlapping addressing.
  - Determining routers accessibility.
  - Address autoconfiguration.
Neighbor Discovery → ND → NDP

It uses five types of ICMPv6 messages:

- Router Solicitation (RS) – ICMPv6 type 133;
- Router Advertisement (RA) – ICMPv6 type 134;
- Neighbor Solicitation (NS) – ICMPv6 135;
- Neighbor Advertisement (NA) – ICMPv6 type 136;
- Redirect – ICMPv6 Type 137

A value of 255 is set to the Hop Limit field.

It can contain options:

- Source link-layer address.
- Target link-layer address.
- Prefix Information.
- Redirected header.
- MTU
Neighbor Discovery

Link layer address discovering
It determines the link layer addressing of neighbors on the same link.
Replaces the ARP protocol.
It uses multicast address “solicited-node” instead of broadcast.
The host sends a NS message informing his MAC address and requests that of the neighbor.
Neighbor Discovery

Link layer address discovering

It determines the link layer addressing of neighbors on the same link.

Replaces the ARP protocol.

It uses multicast address “solicited-node” instead of broadcast.

The host sends a NS message informing his MAC address and requests that of the neighbor.

The neighbor answers by sending a NA message informing his MAC address.
Neighbor Discovery

- **Router and neighbor discovering**
  - Finds neighboring routers on the same link.
  - Determines prefixes and parameters related to the address auto configuration.
  - In IPv4, this function is held by the ARP request.
  - Routers send RA messages to the multicast address “all-nodes”
Neighbor Discovery

Redirect
- It sends Redirect messages.
- It redirects a host to a more appropriate first hop router.
- It informs the originator host that the destination is on the same link.
- This mechanism is the same as the existing one in IPv4.

![Neighbor Discovery Diagram]

Notes:
- Paquete IPv6
- Roteador A
- Roteador B
- Computador C
- Paquete IPv6
Neighbor Discovery

Overlapping addressing detection (Duplicate Address Detection)

Verifies that the address on the link is unique. This mechanism should be held before assigning a Unicast address to an interface. It consists in sending a NS message on behalf of the host with its own address on the “Target address” field. If it gets a NA as an answer, this means that the address is currently in used.
Questions?

Comment?

Alejandro Acosta
alejandro @ lacnic.net
@ITandNetworking