IPv6 Tutorial

Yves Legrandgérad
Thomas Carlu
Bernard Tuy

IPv6 Addressing

Addressing scheme

- RFC 3513 (obsoletes RFC 2373)
- 128 bit long addresses
  - Allow hierarchy
  - Flexibility for network evolutions
- Use CIDR principles:
  - Prefix / prefix length
    - 2001:680:3003::/48
    - 2001:680:3003:2:a00:20ff:fe18:964c/64
  - Aggregation reduces routing table size
- Hexadecimal representation
- Interfaces have several IPv6 addresses
IPv6 Addresses

- **Loopback**: ::1
- **Link local**: FE80:....
- **Site local**: FEC0:....
- **Global**
  - Official: 2001:....
  - 6bone: 3FFE:....
  - **IPv4 mapped**
  - **IPv4 compatible**
  - 6to4: 2002:....

Unicast

Multicast

Ayncast

specific to IPv4/IPv6 integration

---

Interface Identifier

- 64 bits to be compatible with IEEE 1394 (FireWire)
- Eases auto-configuration
- IEEE defines the mechanism to create an EUI-64 from IEEE 802 MAC addresses (Ethernet, FDDI)

![Interface Identifier Diagram]

- 24 bits vendor
- 24 bits serial number
- 24 bits vendor
- 16 bits 0xFFFF
- 24 bits vendor
- 0xFFFE
- 24 bits serial number

---

Interface Identifier (2)

- Links with non global identifier (e.g, the Localtalk 8 bits node identifier) → fill first left bits with 0
- For links without identifiers, there are different ways to proceed (e.g, tunnels, PPP):
  - Choose the identifier of another interface
  - Random number
  - Manual configuration

**THEN**: Invert IEEE EUI-64 “u” bit to become an “interface identifier”
Interface Identifier (3)

- IEEE 24bits OUI can be used to identify HW:
- Interface Identifier can be used to trace a user:
  - The prefix changes, but the interface ID remains the same,
  - Psychological issue.
- Possibility to change Interface ID (RFC 3041 PS):
  - If local storage, use MD5 algorithm
  - Otherwise draw a random number

Anycast Addresses (RFC 2526)

- Anycast IDs are defined in RFC 2526
- Anycast addresses have been defined for routers only so far
  - Subnet prefix = unchanged
  - Anycast ID = highest 128 interface ID values
- 2 different scenarios:
  | 64 bits | 57 bits | 7 bits |
  +---------------------------------+-----------------+----------+
  |   subnet prefix      | 1111110111...111 | anycast ID |
  +---------------------------------+-----------------+----------+
  | interface identifier field     | 1111111111...111 | anycast ID |
  +---------------------------------+-----------------+----------+
- Anycast address of all home agent in 2001:660:3001:4002::/64

Multicast Addresses

- Flag bits: 0 R P T
  - T = 0: permanent addresses (managed by IANA)
  - T = 1: transient multicast addresses
  - P = 1 > T = 1: derived from unicast prefix (RFC3306)
  - R = 1 > P = 1 > T = 1: embedded RP addresses (I-D)

- Scope:
  - 0: Reserved
  - 1: Interface-local
  - 2: Link-local
  - 3: Subnet-local
  - 4: Admin-local
  - 5: Site-local
  - 8: Organization-local
  - E: Global
  - F: Reserved
IPv6 Addresses (continued)

48 bits | 80 bits
--- | ---
001 TLA | NLA | SLA | Interface ID

3 bits 13 bits 32 bits 16 bits 64 bits EU64

Public Topology | Private Topology
--- | ---
TLA : Top Level Aggregator => (/16)
NLA : Next Level Aggregator => (/48)
SLA : Site Level Aggregator => (/64)

RFC 3587: Aggregatable Global Unicast
(obsoletes RFC 2374)

<table>
<thead>
<tr>
<th>3</th>
<th>13</th>
<th>8</th>
<th>24</th>
<th>16</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>TLA</td>
<td>Res</td>
<td>NLA</td>
<td>SLA</td>
<td>Interface ID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>45</th>
<th>16</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Global routing prefix</td>
<td>Subnet ID</td>
<td>Interface ID</td>
</tr>
</tbody>
</table>

RFC 2471: Aggregatable Test Addresses

<table>
<thead>
<tr>
<th>3</th>
<th>13</th>
<th>x</th>
<th>32 - x</th>
<th>16</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>TLA</td>
<td>NLA</td>
<td>SLA</td>
<td>Interface ID</td>
<td></td>
</tr>
</tbody>
</table>

- Used in the 6bone
- TLA value is 0x1FFE => Prefix = 3FFE::/16
- pTLA in the NLA part assigned by ngtrans wg
  http://www.6bone.net/6bone_pTLA_list.html

58 x ::/24
- INNi vệ UNESCO/US-VA: 3FFE::0000::/24
- TELEBIT/DK: 3FFE::0100::/24
- SICS/SE: 3FFE::0200::/24
- GS/FR: 3FFE::0300::/24
- JCON/DB: 3FFE::0400::/24

56 x ::/28
- 3FFE::0000::/28

24 x ::/32
- 3FFE::1553::/32 (2003/03/28)
Production Addressing Scheme (4)

<table>
<thead>
<tr>
<th>FF</th>
<th>IANA/RIK/ER</th>
<th>Public topology</th>
<th>Site topology</th>
<th>Interface ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>45</td>
<td>16</td>
<td>64 bits</td>
<td></td>
</tr>
</tbody>
</table>

IPv6 associated Protocols

- New features specified in IPv6 Protocol (RFC 2460 DS)
- Neighbor Discovery (ND) (RFC 2461 DS)
- Auto-configuration:
  - Stateless Address Autoconfiguration (RFC 2462 DS)
  - DHCPv6: Dynamic Host Configuration Protocol for IPv6 (RFC 3315 PS)
  - Path MTU discovery (pMTU) (RFC 1981 PS)
New Protocols (2)

- MLD (Multicast Listener Discovery) (RFC 2710 PS)
  - Multicast group management over an IPv6 link
  - Based on IGMPv2
  - MLDv2 (equivalent to IGMPv3 in IPv4)
- ICMPv6 (RFC 2463 DS) "Super" Protocol that:
  - Covers ICMP (v4) features (Error control, Administration, …)
  - Transports ND messages
  - Transports MLD messages (Queries, Reports, …)

Solicited Node Multicast Address:
Recall: IPv4

- Correspondence @IPv4 unicast – MAC made by ARP
- Request ARP broadcasted (ethernet FF-FF-FF-FF-FF-FF)

<table>
<thead>
<tr>
<th>HARD ADDRESS TYPE</th>
<th>PROTOCOL ADDRESS TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARD LEN</td>
<td>PADDR LEN</td>
</tr>
<tr>
<td>HARD HADDR (4 premiers Octets)</td>
<td>SENDER HADDR (2 derniers Octets)</td>
</tr>
<tr>
<td>HARD PADDR (2 premiers Octets)</td>
<td>SENDER PADDR (2 derniers Octets)</td>
</tr>
<tr>
<td>TARGET HADDR (2 premiers Octets)</td>
<td>TARGET HADDR(4 derniers Octets)</td>
</tr>
<tr>
<td>TARGET PADDR (les 4 Octets)</td>
<td>TARGET FADDR (3 premiers Octets)</td>
</tr>
</tbody>
</table>

Solicited Node Multicast Address:
And now IPv6…

- IPv6 uses for that the protocol NDP (Network Discovery Protocol) which uses the solicited multicast
Solicited Node Multicast Address:
A Solicited multicast Address

- Concatanation of the prefix FF02::1:FF00:0/104 with the last 24 bits of the IPv6 address

  **Example:**
  - FF02::0000:0000:0000:0001:FF00:0000/104
  - FF02::0000:0000:0000:0001:FF24:87c1

Solicited Node Multicast Address:
Multicast with ethernet

- Ethernet supports multicast (not always implemented)
- 8th bit of the MAC address at 1
- For IPv6: @MAC 33-33-xx-yy-zz-kk
- xx-yy-zz-kk are the last 32 bits of the IPv6 address

  **Example:**
  - Mc sol FF02::0000:0000:0000:0001:FF24:87c1
  - Eth 33-33-FF-24-87-c1

Solicited Node Multicast Address:
The resolution of address in detail

- A wants to send a datagram to B (A knows the IPv6 address of B)
- A builds the solicited multicast address of B
- A sends a message « neighbor sollicitation » to the solicited multicast address of B
Solicited Node Multicast Address: Solicitation message of a neighbor

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>135</td>
<td>0</td>
<td>reserved</td>
</tr>
</tbody>
</table>

- Unicast address of B

Option
(physical address of A)

Solicited Node Multicast Address: Announce of a neighbor

When the machine B receives the datagram « neighbor request »

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>136</td>
<td>0</td>
<td>reserved</td>
</tr>
</tbody>
</table>

- Unicast address of B

Option
(physical address of B)
Neighbor Discovery

- IPv6 nodes which share the same physical medium (link) use Neighbor Discovery (ND) to:
  - discover their mutual presence
  - determine link-layer addresses of their neighbors
  - find routers
  - maintain neighbors’ reachability information (NUD)
  - not directly applicable to NBMA (Non Broadcast Multi Access) networks → ND uses multicast for certain services.

Neighbor Discovery (2)

- Protocol features:
  - Router discovery
  - Prefix(es) discovery
  - Parameters discovery (link MTU, Max Hop Limit, ...)
  - Address autoconfiguration
  - Address resolution
  - Next Hop determination
  - Neighbor Unreachability Detection
  - Duplicate Address Detection
  - Redirect

Neighbor Discovery (3):
Comparison with IPv4

- It is the synthesis of:
  - ARP
  - R-Disc
  - ICMP redirect
  - ...
**Neighbor Discovery (4)**

- ND specifies 5 types of ICMP packets:
  - **Router Advertisement (RA):**
    - periodic advertisement (of the availability of a router) which contains:
      - list of prefixes used on the link
      - a possible value for Max Hop Limit (TTL of IPv4)
      - value of MTU
  - **Router Solicitation (RS):**
    - the host needs RA immediately (at boot time)

**Neighbor Discovery example: Address Resolution**

At boot time, every IPv6 node has to join 2 special multicast groups for each network interface:
- All-nodes multicast group: ff02::1
- Solicited-node multicast group: ff02::1:ffxx:xxxx (derived from the lower 24 bits of the node’s address)

**Neighbor Discovery (5)**

- **Neighbor Solicitation (NS):**
  - to determine the link-layer address of a neighbor
  - or to check its unreachability
  - also used to detect duplicate addresses (DAD)
- **Neighbor Advertisement (NA):**
  - answer to a NS packet
  - to advertise the change of physical address
- **Redirect:**
  - Used by a router to inform a host of a better route to a given destination
Path MTU discovery (RFC 1981)

- Derived from RFC 1191, (IPv4 version of the protocol)
- Path = set of links traversed by an IPv6 packet between source and destination
- link MTU = maximum length (in bytes) of a packet that can be transmitted on the link without fragmentation
- Path MTU (or pMTU) = min { link MTU } for a given path
- Path MTU Discovery = automatic pMTU discovery for a given path

Path MTU discovery(2)

- Protocol operation
  - makes assumption that pMTU = link MTU to reach a neighbor (first hop)
  - if there is an intermediate router such that link MTU < pMTU \( \Rightarrow \) it sends an ICMPv6 message: "Packet size Too Large"
  - source reduces pMTU by using information found in the ICMPv6 message

Auto-configuration

- Hosts should be plug & play
- Uses ICMPv6 messages (Neighbor Discovery)
- When booting, the host asks for network parameters:
  - prefix
  - default router
  - hop limit
  - ...
Auto-configuration (continued)

- Currently only routers have to be manually configured but work on prefix delegation is in progress (draft-ietf-ipv6-prefix-delegation-requirement-01.txt)
- Hosts can obtain automatically an IPv6 address
  - BUT this address will not be automatically registered in the DNS
  - If it is always the same: may be manually added
- NEED for DNS Dynamic Update (RFC 2136 PS and RFC 3007 PS) for IPv6

Stateless Auto-configuration

- IPv6 Stateless Address Autoconfiguration (RFC 2462 DS)
  - Does not apply to routers
- Allows a host to create a global IPv6 @ from:
  - its MAC @
  - router advertisements coming from router(s) on the link

Stateful Auto-configuration (DHCPv6)

- Dynamic Host Configuration Protocol for IPv6
  - RFC 3315
  - IPv4 version of DHCP (RFC 1541, RFC 2131)
    - based on BOOTP (RFC 951)
- Server
  - Memorises client’s state
  - Optionnally provides the client with IPv6 addresses and configuration parameters
- Client
  - Sends requests and acknowledgements in accordance with the protocol (DHCP)
Auto-configuration example

Create the link local @

RS

RA

(DNS Dynamic Update ?)

Do a DAD
Send a RS using a Multicast address (ff02::2)
Receive global prefix(es)
Do a DAD
Set default router
(DHCPv6 ?)

Router Renumbering (RFC 2894 PS)

- Allow to change/add prefixes into routers
  - End-systems will use Neighbor Discovery to automatically discover and configure with the new prefix(es)
- Several actions are sent to routers using well-known multicast groups:
  - Change prefix
  - Add prefix
- Security needs (IPsec, no replay)

Routing Protocols

- RFC 2080 (PS) & 2081 (INFO) : RIPng
- RFC 2740 (PS) : OSPF v3
- draft-ietf-isis-ipv4-05.txt : IS-IS (01/2003)
- RFC 2545 (PS) : based on MBGP
  - Multi-extension protocol for BGP

⇒ No major differences with IPv4

- MPLS and 6PE
IPv6 support in the DNS (DNSv6)

Overview

- How important is the DNS?
- DNS resource lookup
- The two approaches to the DNS
- DNS extensions for IPv6
- About the required AAAA glue in DNS parent zones
- Lookups in an IPv6-aware DNS tree
- DNS service continuity through IPv4 and IPv6 networks: what are the issues?
- DNSv6 migration: how to proceed?
- DNSv6 operational requirements & recommendations
- IPv6-capable DNS software
- Standardization process (RFC 1886 inter-operability tests & reports)
- References

How important is the DNS?

- Getting the IP address of the remote computer is necessary prior to any communication between TCP/IP applications
-Humans are unable to memorize millions of IP addresses 😓
- To a larger extent: the Domain Name System (DNS) provides applications with several types of resources (name servers, mail exchanges, reverse lookup, …) they need
- DNS design
  - hierarchy
  - distribution
  - redundancy
DNS extensions for IPv6

- **AAAA** (RFC 1886): forward lookup (‘Name → IPv6 address’):
  - Equivalent to ‘A’ record
  - Example:
    ```
    mscachev6.mic.fr. IN AAAA 2001:660:3003:2::1:1
    ```

- **PTR**: reverse lookup (‘IPv6 address → Name’):
  - Reverse tree equivalent to in-addr.arpa
  - Nibble (4 bits) boundary
  - Original tree: 1p6., int (RFC 1886), still maintained
  - New tree: 1p6.arpa (RFC 3152), under deployment
  - Example:
    ```
    $ORIGIN 2.0.0.0.3.0.0.3.0.6.6.0.1.0.0.2.ip6.{int,arpa}.
    1.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
About required AAAA glue in DNS parent zones

Example: In zone file rennes.enst-bretagne.fr

```
@ IN SOA rsm.rennes.enst-bretagne.fr. fradin.rennes.enst-bretagne.fr.
( 2002121501 ;serial
  86400 ;refresh
  3600 ;retry
  3600000 ;expire
  86400 ;negative ttl)

IN NS rsm

IN NS univers.enst-bretagne.fr.

[...]

ipv6 IN NS rhadamanthe.ipv6

IN NS ns3.nic.fr.

IN NS rsm

rhadamanthe.ipv6 IN A 192.108.119.134

[...]
```

IPv4 glue (A 192.108.119.134) is required to reach rhadamanthe over IPv4 transport
IPv6 glue (AAAA 2001:660:282:1::1) is required to reach rhadamanthe over IPv6 transport

Lookups in an IPv6-aware DNS tree

```

DNS service continuity through IPv4 and IPv6 networks: what are the issues? (1)

```

IPv4-only Network
IPv6-only Network
Name Server
Cache
Manually configured root file
```

13 IPv4-only Root Name Servers
```
DNS service continuity through IPv4 and IPv6 networks: what are the issues? (2)

DNSv6 migration: how to proceed?

- How to deploy DNSv6 without partitioning the Internet?
  - This means that name servers support native IPv6 transport

  A) Proceed top-down: (in an ideal world)
  - V6-fy part of (or all) root name servers, then TLD servers, and so on...
    - Putting AAAA glue (when required) in the parent zone IS POSSIBLE

  B) Proceed bottom-up: (in murphy’s world)
    - B1 - V6-fy your authoritative name servers
    - B2 - If AAAA glue is required in the parent zone, ask the parent to put it
    - B3 - If your parent is not convinced, try to explain to him that IPv6 is harmless
    - B4 - If still not convinced, start doing politics/lobbying until getting it!
    - B5 - If convinced (v6 delegation succeeded), the parent will do B1-B5 on his turn

  C) Mixture of A and B: (in the real world)
    - V6-fy the name servers where possible
    - Where not possible, negotiate (at least) putting AAAA glue when required
    - DO NOT BREAK THE CONTINUITY OF THE DNS SERVICE!

DNSv6 operational requirements & recommendations

- The target today is not the transition from an IPv4-only to an IPv6-only environment
  - It is rather to get from an IPv4-only to a mixed v4-v6 environment where:
    - Some systems will remain IPv4-only
    - Some systems will be dual-stacked
    - Some systems will be IPv6-only

- How to get there?
  - Deploy DNSv6 in an incremental fashion on existing networks
  - For new large IPv6-only networks: enable IPv6-only resolvers to query the DNS for IPv4-only resources by (for example):
    - Letting them query dual-stack forwarders
    - Using some DNS ALG

- Bear in mind
  - Any DNS zone (and especially if related to an IPv6-only network) SHOULD be served by at least one IPv4 name server
  - All DNS zones (including “root”, yes, yes!) SHOULD be reachable over IPv4 and IPv6
DNS IPv6-capable software: name servers

- BIND
  - 4.9.4 or later / BIND 8 (up to 8.2.3)
    - AAAA & PTR support (contents): no IPv6 transport
  - BIND 8.2.4 (or later)
    - native IPv6 transport
  - BIND 9.0 (or later, current release: BIND-9.2.2)
    - AAAA, PTR
    - Native IPv6 transport
- Microsoft DNS server (ip6.arpa supported?)
- NSD (authoritative only, see http://www.nlnetlabs.nl)
- And many others…

DNS IPv6-capable software: resolvers

- KAME
  - AAAA and PTR
  - Support of ip6.arpa?
- BIND 9
  - AAAA, PTR (A6, DNAME)
  - Support of both ip6.int and ip6.arpa
- Linux (libc)
  - AAAA and PTR, ip6.{int,arpa}
- Microsoft Windows
  - AAAA and PTR
  - Support of ip6.arpa?

APIs

- getaddrinfo() for forward lookup
  - hostname ➔ addresses
  - Replacement of gethostbyname()
  - With AF_UNSPEC, applications become protocol-independent
- getnameinfo() for reverse lookup
  - address ➔ hostname
  - Replacement of gethostbyaddr()
Standardization process (RFC 1886 inter-operability tests & reports)

- RFC 1886: AAAA & ip6.int
- RFC 3152: ip6.arpa

RFC 1886 inter-operability tests
- Who: 6WIND, AFNIC, FT R&D and IRISA (within « G6 test » activity)
- When & where: 3 June & 4 July 2002, AFNIC and 6WIND buildings
- What was tested: support of AAAA and ip6.arpa by different name server/resolver software
- Results:
  - Successful inter-operability tests but found some minor failures

RFC 1886 inter-operability reports
- When & where: IETF 54 Yokohama (14-19 July 2002) at dnsext working group session
- Presentation:
- Results:
  - RFC 1886 currently is a Proposed Standard (PS) status
  - draft-ietf-dnsext-rfc1886bis-01.txt (0 announced on 12 September 2002)
  - Toward a Draft Standard (DS) status RFC (RFC 3596 to be published soon)

References

- DNSv6-related RFCs & Internet-Drafts
  - 1886, 2673, 2874, 3152, ...
  - draft-ietf-dnsop-ipv6-dns-issues-02.txt (Alain Durand & Johan Ihren, work in progress)
- Documentation (English & French)
  - http://www.dns.net/dnzd/ (RFC, drafts, FAQ, ...)
  - http://www.isc.org
  - http://www.nic.fr/guides/
  - http://www.nic.fr/formation/supports/
- DNS & IPv6-related IETF working groups
- Books
  - DNS and BIND, 4th edition (Paul Albitz & Cricket Lu)

Transition / Integration Mechanisms
Transition/Integration (agenda)

- Dual stack IPv4-IPv6
- Tunneling mechanisms
- Translation mechanisms
- DSTM
- Deployment strategies

Transition / Integration
Mechanisms

Dual stack IPv4/IPv6

Dual stack

- IPv4 and IPv6 running together
- 2 scenarios:
  - Existing network
  - New network
Drawbacks

- Dual stack configured for IPv4 and IPv6
- Doesn't solve the lack of IPv4 addresses
- Routers need to be configured both with IPv4 and IPv6 routing tables
- Obsoleted with RFC 2893

IPv4 Mapped addresses

- IPv6-only applications can use IPv4 transport

IPv4 Mapped Addresses

client
IPv6
3ffe:305:1002::1
IPv6
3ffe:305:1002::1
IPv4
3ffe:305:1002::1->3ffe:305:1002::2

server
IPv6
3ffe:305:1002::2
IPv6
3ffe:305:1002::2
IPv4
3ffe:305:1002::1->3ffe:305:1002::2
IPv4 Mapped Addresses (continued)

Transition / Integration Mechanisms

Tunneling

Tunnelling facility

- Configured tunnels
  - widely deployed in the 6bone
  - used to connect two sites
  - require manual configuration
- IPv4-Compatible addresses
  - no usage
- Automatic tunnelling
  - 6to4
    - Tunnel Broker
- 6over4
  - no usage
IPv6 in IPv4 configured tunnel

- Put IPv6 packet in IPv4 payload
- IPv4 protocol 41 means data = IPv6 packet
- Underlying infrastructure becomes transparent
- Makes it possible to connect to IPv6 network over an IPv4 link

- Need to specify tunnel end points
- Can give addresses on IPv6 logical link

6bone

Create a virtual topology over the IPv4 network with configured tunnels

IPv4 Compatible Addresses

- Used at the beginning for transition with IPv4
- Allows encapsulation of IPv6 packet into IPv4 packets
- Dynamic tunneling
IPv4 Compatible Addresses

- Like IPv4 addresses with 96 bits to 0
  - Used when only a few IPv6 hosts were on the Internet
  - Don’t learn how to manage an IPv6 network
- Need more sophisticated networks
  - E.g the 6bone mainly use static tunnels between routers

NOT USED ANYMORE

6to4 (RFC 3056 PS)

- Another way to build a tunneled infrastructure
- Simple configuration (no need to configure static tunnels)
- Use a special address plan
  - Prefix: 2002::/16
### 6to4: Address Allocation

- Site prefix is derived from the v4 address of the border router

<table>
<thead>
<tr>
<th>Prefix:</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002::1001:203::/48</td>
<td>192.1.2.3</td>
</tr>
<tr>
<td>2002::0102:0304::/48</td>
<td>128.1.2.3</td>
</tr>
</tbody>
</table>

### 6to4: Tunneling

- DNS (AAAA) for B?
- B = 2002:0102:0304::1

### 6to4: Interaction with the 6bone

- If one node has a 6to4 address and the other one has both a 6to4 and global IPv6 addresses
  - Select 6to4 address
- If both have 6to4 and global IPv6 addresses
  - Global IPv6 should be selected
Relays are just routers with one interface on the native IPv6 network and one on the 6to4 network.

- If the relay can be announced through an interior gateway protocol:
  - Doesn't change anything
- More complex, when an exterior protocol is used.
Tunnel brokers

- Simplify/Allow the construction of IPv4 tunnels.
- Use of a web page

CSELT IPv6 Tunnel Broker

Create Tunnel

- IPv6 Address:
- Client ID Type:
- Client Type:
- IPv6 Dynamic Tunneling Broker

CSELT IPv6 Tunnel Broker

Tunnel Info

- Server IP Address
- Server Port Address
- Server IP Address
- Client IP Address
- Client Port Address
- Client IP Address

The following table contains links to the scripts that will help you in configuring your host.
Transition / Integration Mechanisms

Translation mechanisms

Interoperability tools: Translators

- **IP level**
  - SIIT (Stateless IP/ICMP Translation)
  - NAT-PT (Network Address Translation-Protocol Translation)
  - BIS (Bump In the Stack)
- **TCP level**
  - TCP-relays
  - SOCKS
- **Application level**
  - Bump in the API
  - proxies

Cohabitation Mechanisms

- **Different approaches**
  - Application Level Gateways
  - TCP/UDP relay & SOCKv6
  - Dual Stack Transition Mechanism
  - Packets translation:
    - SIIT
    - NAT-PT
Application Level Gateways

- May be used for a large majority of common applications:
  - E-mail (POP3, IMAP, SMTP)
  - Web (proxies)
  - Printer (spoolers)
  - DNS: relay (may change the RR type)
- Reduce IPv4 traffic inside a domain

May use tunnels

For example: an old printer without an IPv6 stack

RFC 2765 PS: Stateless IP/ICMP Translation (SIIT)

- Suppress the v4 stack
- Translate the v6 header into a v4 header on some point of the network
  - Routing can direct packet to those translation points.
- Translate ICMP from both worlds
- No State in translators (≠ NAT)
V6 header contains:
- IPv4 mapped addresses
  - 0...0 FFFF IPv4
- IPv4 translated addresses
  - 0...0 FFFF 0000 IPv4
- FFFF doesn’t modify TCP/UDP checksum

Have a IPv4-translated address assigned from a pool

Network routes IPv4-mapped packets to a translation point

Have a IPv4-translated address assigned from a pool

Network routes IPv4-translated to the destination
NAT-PT (RFC 2766 PS)

- Translate addresses and headers
- A pool of routable addresses is assigned to the translator
- Outcoming session translation is easy
- Incoming translation must intercept DNS requests

NAT-PT: v6 to v4

Prefix is routed to the NAT box

May change port numbers to allow more translations

Act as DNS-ALG
Transition / Integration Mechanisms

DSTM

Dual Stack Transition Mechanism (DSTM)

What is it for?
- DSTM allows hosts in IPv6-only networks to communicate with hosts in the IPv4-only Internet.
- DSTM allows IPv4-only applications to run (without modification) over IPv6-only networks.

Assumes IPv4 and IPv6 stacks are available on host.

IPv4 stack is configured dynamically only when one or more applications need it
- A temporary IPv4 address is assigned to the host
- Needs an address allocation protocol.

All IPv4 traffic coming from the host is tunneled towards the DSTM gateway (IPv4 over IPv6).
- Needs an IPv4/IPv6 encapsulate/decapsulate gateway
- Gateway maintains an @v6 @@v4 mapping table
- Reverse route towards DSTM host MUST pass through the gateway
**DSTM: How it works (v6 → v4)**

- In A, the v4 address of C is used by the application, which sends v4 packet to the kernel
- The interface asks DSTM Server for a v4 source address
- DSTM server returns: - A temporary IPv4 address for A - IPv6 address of DSTM gateway

A creates the IPv4 packet (A \(\rightarrow\) C)
- A tunnels the v4 packet to B using IPv6 (A \(\rightarrow\) B)
- B decapsulates the v4 packet and sends it to C
- B keeps the mapping between A \(\rightarrow\) B in the routing table

**Scenario 2: v4 to v6**

- C asks for the IPv4 address of A
- Query fails, DSTM server tells A to configure its IPv4 stack
- A configures its IPv4 stack
Scenario 2: v4 to v6

- A registers to the DNS and tells to server
- Mapping table at gateway is configured
- B sends IPv4 address of A to C
- Communication can take place

DSTM: Address Allocation

- **Manual**
  - host lifetime (no DSTM server)
- **Dynamic**
  - use DHCPv6
    - DHCPv6 may not be ready soon!
  - use RPC
    - Easier, RPCv6 is ready
    - Works fine in v6 + v4 case
    - Can be secure*
  - use TSP
    - Based on XML
    - Can be secure
- Security Concerns
  - Request for IPv4 address needs authentication
  - Automatic @6 -> @4 mapping at gw, or configured by server?

DSTM: Application

- DSTM is a useful tool when support for IPv4 addressing and routing is to be turned off inside a network.
  - No IPv4 addresses… No address exhaustion problem
  - No IPv4 routing (only IPv6)… easier to manage
  - DSTM assures IPv4 communication with the external world.

- DSTM is to be used ONLY when no other means of communication is possible.
  - ALGs may be a better solution for several services
  - ALGs reduce the need of IPv4 addresses.
DSTM: Application

IPv4 Internet

DSTM: Deployment

- DSTM may be deployed in several phases:
  - If IPv4 address space is not a problem, static tunnels may be set up from DSTM nodes to the DSTM gateway. No dynamic allocation.
  - If address space is a problem, a dynamic address allocation mechanism may be set up (TSP, RPC, DHCPv6).
  - If address space is a big problem, address allocation may also involve port numbers.

Application: The VPN scenario

- Giving IPv4 addresses to visitors can become expensive:
  - Visited Network offers IPv6 connectivity only
  - Home network offers connection to the v4 world via DSTM
    - to Corporate Intranet
    - to Global Internet

- Home network offers IPv6 connectivity to the v6 world via DSTM

- Giving IPv4 addresses to visitors can become expensive:
  - Visited Network offers IPv6 connectivity only
  - Home network offers connection to the v4 world via DSTM
    - to Corporate Intranet
    - to Global Internet

- Giving IPv4 addresses to visitors can become expensive:
  - Visited Network offers IPv6 connectivity only
  - Home network offers connection to the v4 world via DSTM
    - to Corporate Intranet
    - to Global Internet
**DSTM vs. NAT-PT**

- NAT-PT has the same problems as classic NAT:
  - Translation is sometimes complex (e.g. FTP)
  - NAT box may need to be configured for every new application.
  - NAT-PT supposes v6fied applications
    - This is not the case!
    - In DSTM, applications can send IPv4 packets to the kernel.

**DSTM: Implementations**

- **BSD « INRIA »**
  - DSTM gateway
  - DSTM server (RPC)
  - Client: manual conf, dynamic conf
- **BSD Kame**:
  - Gateway/Server on the same host
  - Based on RPC (dynamic conf)
  - Compatible with Linux implementation

- **Linux**:
  - Dynamic configuration using RPC
  - 4over6 interface
  - Same capabilities as BSD version

- **Windows**:
  - Prototype from isoft (Korea)
  - 4over6 interface for windows client
  - Uses DHCPv6
  - Server runs over Linux
  - Needs external TEP

[http://www.ipv6.rennes.enst-bretagne.fr/dstm/]
Deployment/migration strategies

Deployment strategies

- Technical factors
  - IPv6 availability (connectivity)
  - Native IPv6 applications/services availability
  - Avoid blocking situations (chicken and egg problem)

- Psychological factors
  - skills to configure IPv6
  - risk to modify something that works

- Deploy only one version of IP (either v4 or v6) on a given area of the network
  - To manage both routing plans

Case study: phase 0
IPv4 site
Case study: phase 1
hybrid stack servers & routers

Case study: phase 2
hybrid stack clients

Case study: phase 3
Connection to the 6bone
Case study: phase 4
IPv6 only hosts

Case study: phase 5
IPv6 only hosts to IPv4 server

Case study: Phase 6
No IPv6 ISP available - IPv6 site
Case study: phase 7
IPv6 site / IPv6 Internet

IPv4 Internet

IPv6 Internet

NFS

client

IPv6 site

Exit router to IPv6 ISP

NAT-PT or proxies

Equipment Configuration

- CISCO
- JUNIPER
- 6WIND
- FreeBSD
- Debian
- Microsoft (Windows XP)
- Zebra
Enable IPv6 on an interface
interface xxxxx
ipv6 enable

Configure an address
interface xxxxx
ipv6 address X::X::X::X/0-128 (general address)
ipv6 address X::X::X (link-local address)
ipv6 address autoconfig (autoconfiguration)

Configure an IPv6 in IPv4 tunnel
interface tunnel x
tunnel source interface
tunnel destination X.X.X.X
ipv6 address X::X::X::X/0-128
tunnel mode ipv6 (for direct tunneling)
tunnel mode gre ip (for gre encapsulation)

Configure an IPv6 in IPv6 tunnel
interface tunnel x
tunnel source interface
tunnel destination X::X::X::X
ipv6 address X::X::X::X/0-128
tunnel mode ipv6 (for direct tunneling)
tunnel mode gre ipv6 (for gre encapsulation)
Enable IPv6 routing
ipv6 unicast-routing

Configure static routes
ipv6 route prefix/prefixlen next_hop
Ex: ipv6 route ::/0 2001:680:10a::1

BGP configuration
router bgp xxxx
neighbor X:X:X:X::X remote-as ...
neighbor X:X:X:X::X ...
address-family ipv6
neighbor X:X:X:X::X activate
neighbor X:X:X:X::X ...
exit address-family

ACLs
ipv6 prefix-list bgp-in-6net seq 5 deny ::/0
--> Means filter ::/0 exactly
ipv6 prefix-list bgp-in-6net seq 10 deny 3FFE::/24 le 28
ipv6 prefix-list bgp-in-6net seq 15 deny 2001:660::/35 le 41
ipv6 prefix-list bgp-in-6net seq 20 permit 2002::/16
ipv6 prefix-list bgp-in-6net seq 25 permit 3FFE::/17 ge 24 le 32
ipv6 prefix-list bgp-in-6net seq 30 permit 3FFE:8000::/17 ge 28 le 28
 --> Means every prefix matching 3FFE:8000::/17 with length 28
ipv6 prefix-list bgp-in-6net seq 35 permit 2001::/16 ge 32 le 32
ipv6 prefix-list bgp-in-6net seq 40 permit 2001::/16 ge 32 le 35
 --> Means every 2001::/16 derived prefix, with length between 32 and 35
Juniper (1)

- Interface configuration

```plaintext
interfaces {
  name_of_interface {
    unit x {
      family inet {
        address X.X.X.X/prefixlength;
      }
      family iso {
        address Y.Y.Y.Y;
      }
      family inet6 {
        address Z:Z:Z:Z::Z/prefixlength;
      }
    }
  }
}
```

- Cannot autoconfigure the router interfaces

Juniper (2)

- Router advertisements (stateless autoconf)

```plaintext
protocols {
  router-advertisement {
    interface interface-name {
      prefix IPv6_prefix::/prefix_length;
    }
  }
}
```

- Configure tunnel (with Tunnel PIC)

```plaintext
interface {
  ip-x/x/x {
    tunnel {
      source ipv4_source_address;
      destination ipv4_destination_address;
      family inet {
        address ipv6_address_in_tunnel/prefixlength;
      }
    }
  }
}
```

Juniper (3)

- Static routes

```plaintext
routing-options {
  rib inet 0 {
    static {
      route IPv6_prefix next-hop IPv6_address;
    }
  }
  rib inet 0 {
    static {
      route IPv6_prefix discard; -- Useful to originate a network
    }
  }
}
```
Juniper (4)

**BGP configuration**

```plaintext
protocols {
  bgp {
    local-as local_AD_number;
    group EBGP_peers {
      type external;
      family inet6 {
        neighbor neighbor_IPv6_address;
        peer-as distant_AD_number;
        import in-PS;
        export out-PS;
      }
    }
  }
}
```

Juniper (5)

**Policy statements**

```plaintext
policy-statement in-PS {
  term from_outside_accept {
    from {
      route-filter 2002::/16 exact;
      route-filter 3FFE::/17 prefix-length-range /24-/24;
      route-filter 3FFE:8000::/17 prefix-length-range /28-/28;
      route-filter 3FFE:4000::/18 prefix-length-range /32-/32;
      route-filter 2000::/3 prefix-length-range /16-/16;
      route-filter 2001::/16 prefix-length-range /29-/35;
    }
    then {
      accept;
    }
    then reject;
  }
}
```

6WIND

**Interface Configuration**

- **Enter Ethernet Private Interface Context**
  ```plaintext
  hurricanes(mycfg0) eth0_0
  hurricanes(mycfg0-eth0_0)
  ```

- **Set IP Address**
  ```plaintext
  hurricanes(mycfg0-eth0_0) ipaddress 10.0.0.10/24
  hurricanes(mycfg0-eth0_0) ipaddress 3ffe:10::beef/48
  ```

- **Advertise an IPv6 prefix**
  ```plaintext
  hurricanes(mycfg0-eth0_0) prefix 3ffe:10::beef:48/64
  ```
Migration configuration

- Enter Migration Context
  
  ```
  hurricane{myconfig} mig
  hurricane{myconfig-mig}
  ```

- Create 6in4 interface
  
  ```
  hurricane{myconfig-mig} 6in4 0 1.1.1.10 1.1.1.20 3ffe:1::10 3ffe:1::20
  ```

- Create 4in6 interface
  
  ```
  hurricane{myconfig-mig} 4in6 0 3ffe:1::10 3ffe:1::20 1.1.1.10 1.1.1.20
  ```

- Create 6to4 interface
  
  ```
  hurricane{myconfig-mig} 6to4 1.1.1.10
  ```

Create ISATAP interface

```
hurricane{myconfig-mig} isatap_router 0 10.0.0.10
hurricane{myconfig-mig} isatap_prefix 0 2002:101:10a::/64
```

Create DSTM interface

```
hurricane{myconfig-mig} dstm eth0_0
```

Static Routing Configuration

- Enter Routing Context
  
  ```
  hurricane{myconfig-rtg}
  ```

- Set IP Default Route
  
  ```
  ipv4_defaultroute 1.1.1.20
  ipv6_defaultroute 3ffe:1::20
  ```

- Set static route
  
  ```
  route 30.0.0.0/24 3.3.3.30
  route 3ffe:30::/48 3ffe:3::30
  ```
6WIND (5)

Dynamic Routing Configuration RIP
- Enter Dynamic Routing Context
  `hurricane{myconfig-rtg} dynamic`
  `hurricane{myconfig-rtg-dynamic}`
- Activate RIP Routing Process
  `hurricane{myconfig-rtg-dynamic} router rip`
  `hurricane{myconfig-rtg-dynamic-router-rip} network 1.1.1.0/24`
  `hurricane{myconfig-rtg-dynamic-router-rip} network 3.3.3.0/24`
  `hurricane{myconfig-rtg-dynamic-router-rip} redistribute connected`

6WIND (6)

Dynamic Routing Configuration BGP4+
- Enter Dynamic Routing Context
  `hurricane{myconfig-rtg} dynamic`
- Activate BGP4+ Routing Process
  `hurricane{myconfig-rtg-dynamic} router bgp 10`
  `hurricane{myconfig-rtg-dynamic-router-bgp} neighbor 3ffe:1::20 remote-as 20`
  `hurricane{myconfig-rtg-dynamic-router-bgp} neighbor 3ffe:3::30 remote-as 30`
  `hurricane{myconfig-rtg-dynamic-router-bgp} address-family ipv6`
  `hurricane{myconfig-rtg-dynamic-router-bgp-v6} neighbor 3ffe:1::20 activate`
  `hurricane{myconfig-rtg-dynamic-router-bgp-v6} neighbor 3ffe:3::30 activate`
  `hurricane{myconfig-rtg-dynamic-router-bgp-v6} redistribute connected`

FreeBSD
- Enable IPv6
  `ipv6_enable=“YES“` in rc.conf file
- Autoconfiguration is automatically done while the gateway function is off
- Enable IPv6 forwarding
  `ipv6_gateway_enable=“YES“` in rc.conf file
- Add an IPv6 address on an interface
  `ifconfig interface inet6 X::X::X::X prefixlen 64`
FreeBSD (2)

- Configure an IPv6 in IPv4 tunnel
  ifconfig gif create
  ifconfig gif inet6 @IPv6_source @IPv6_dest prefixlen 128
  ifconfig gif up

- Configure an IPv6 in IPv6 tunnel
  ifconfig gif create
  ifconfig gif inet6 @IPv6_source @IPv6_dest prefixlen 128
  gifconfig gif inet @IPv4_source @IPv4_dest

FreeBSD (3)

- Configure a static route
  - Default route
    route add -inet6 default fe80::X:X:X:X%interface
    route add -inet6 default X.X.X.X:: (if global address)
  - Others
    route add -inet6 X.X.X.X:: -prefixlen YY X.X.X.X::
    route add -inet6 X.X.X.X:: -prefixlen YY fe80::X:X:X:X%interface

  %interface notation
  If link-local address, need to specify on which interface the address is available

FreeBSD (4)

- RIPng: route6d daemon
  route6d
  -L IPv6_prefix,interface (receives only prefixes derived from IPv6_prefix on interface interface)
FreeBSD (5)

- BGB: bgpd daemon
- Better to use Zebra BGP daemon

Debian

- Main URL:
  http://people.debian.org/~csmall/ipv6/

- Enable IPv6
  - Put "ipv6" in "/etc/modules"
  - Edit "/etc/network/interfaces":
    ```
    iface eth0 inet6 static
    address 2001:XXXX:YYYY:ZZZZ::1
    netmask 64
    ```

Debian (2)

- Tunnel configuration
  - Edit "/etc/network/interfaces":
    ```
    iface tun0 inet6 v4tunnel
    endpoint A.B.C.D
    address 2001:XXXX:1:YYYY::2
    gateway 2001:XXXX:1:YYYY::1
    netmask 64
    ```
Debian (3)

- RA configuration on a Debian router
  - Add in "/etc/radvd.conf":
    
    ```
    interface eth0 {
        AdvSendAdvert on;
        AdvLinkMTU 1472;
        prefix 2001:XXXX:YYYY:ZZZZ:/64 {
            AdvOnLink on;
            AdvPreferredLifetime 3600;
            AdvValidLifetime 7200;
        };
    }
    ```

Microsoft (Windows XP)

- Enable IPv6
  - `ipv6 install` in a dos window
- Autoconfiguration is automatically done
- Display IPv6 interfaces
  - `ipv6 if`
- Display IPv6 routes
  - `ipv6 rt`

Microsoft (Windows XP) (2)

- Add a static route
  - `ipv6 rtu prefix ifindex[/adresse] [life valid[/pref]] [préférence P] [publish] [age] [spl Taille_préfixe_site]`
- Anonymous addresses
  - `ipv6 gpu UseAnonymousAddresses no`
- « User-friendly » IPv6 configuration
  - `netsh in a dos window`
  - > `interface ipv6`
Zebra

- Cisco like commands
- BGP, RIPng, OSPF available

And once deployed...

Home usage

- Easy configuration
  - Plug and play
  - Compatible with IEEE 1394
- Some network games send IPv4 addresses:
  - NAT doesn't work
- Advanced users wish to create servers
  - Paging, Web servers, IP telephony,...
  - Remote control
Home usage

Mobile Telephony

- Not IP telephony
- Huge number of addresses
- Can use mobile IP
  - Interaction between L2 and L3 mobility not discussed here
- End-to-End connectivity: necessary condition, fulfilled by IPv6 global addressing
- Need for services regardless of IP version...
- Robust Header Compression
  - Include RTP/UDP/IPv6
  - IPv6 header is easier to compress

Mobile telephony

- Some Terminal:
  - Dual stack
- Limited number of applications
  - E-mail
  - Web/WAP browser
  - ...
- Can be implemented for both stacks
- Mobile PC can also be connected
Mobile Telephony

POP/IMAP/SMTP

HTTP/WSP

Mobile telephony

POP/IMAP/SMTP

HTTP/WSP

Conclusion

- Complexity will increase in the IPv4 world
  - New applications
  - New paradigms
  - End of end-to-end
- Toward a layer-7 network
  - More costs
  - Difficulty to introduce new applications
Conclusion (2)

![Graph showing complexity over time for IPv4 and IPv6]

- IPv4
- IPv6

Conclusion (3)

- IPv6 migration might be triggered by:
  - Research projects (6Bone, Renater 2 pilot, ...)
  - Developing countries (lack of IPv4 address blocks)
  - IPv6 Product availability

- Smooth migration area by area:
  - Interoperability between v4 and v6 areas must be maintained for some applications and equipment
  - Different approaches to maintain interoperability
  - Complexity will decrease with time

To go on ...

- [http://www.6bone.net/](http://www.6bone.net/)
- [http://www.ripe.net/](http://www.ripe.net/)
- [IPv6 wg](http://www.ipv6forum.com/)
- [http://www.g6.asso.fr/](http://www.g6.asso.fr/)
Bibliography … in French!