



مسار

IP Version 6 and the Next Generation of Internet Protocols

ETH Zürich, Zurich, Switzerland
2012-04-30 | Jeroen Massar <jeroen@massar.ch>

Who is Jeroen?

Working for own company Massar Networking (<http://www.massar.ch>) which focuses on research and development of new protocols to enhance anonymous networking, privacy on the Internet and performance measurements.

One part of the duo running and developing SixXS (<http://www.sixxs.net>) which has been providing IPv6 connectivity to tens of thousands of users worldwide for the last 10 years using custom protocols for tunneling IPv6 over IPv4, IPv6Gate for users without tunnels and the GRH system to monitor the IPv6 routing system.

Other details: <http://jeroen.massar.ch>





IPv6

IP next generation

“Under the hood...”

ETH Zürich, Zurich, Switzerland

2012-04-30 | Jeroen Massar <jeroen@massar.ch>

IPv4 Internet History

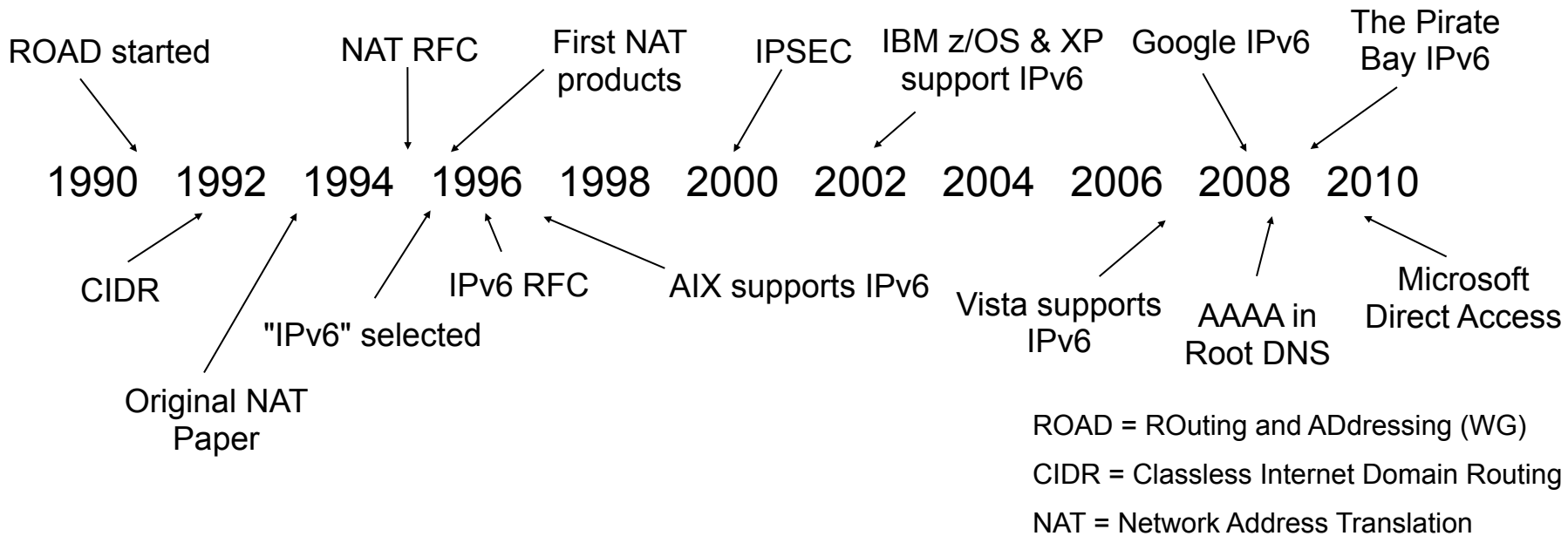
- 1969: ARPAnet commissioned by DoD
- 1970: ARPAnet hosts start using NCP:
Network Control Protocol (RFC33) - 8 bit addressing - only 256 hosts
- 1972: Telnet (RFC318)
- 1973: Ethernet outlined in PhD Thesis + FTP (RFC454)
- 1976: Queen Elizabeth sends an email
- 1982: DCA & ARPA establish TCP and IP, DoD declares TCP/IP as their standard for networks
- 1983: Cutover NCP to TCP/IP (IPv4)
Flagday on January 1st

NCP vs IPv4 vs IPv6

- NCP: 8 bits addressing - 256 hosts
 - IPv4: 32 bits addressing - (4 294 967 296) 4+ billion hosts
 - IPv6: 128bits addressing - $3.4 \cdot 10^{28}$ hosts
-
- Note that the current world population is around 7 billion.... and expected to be around 8 billion in 2025.
 - IPv6 thus gives $4.8 \cdot 10^{28}$ addresses for each of the about 7 billion people
 - As we cut up IPv6 in /64 networks though it is better to talk about networks than hosts, but then still it should be enough to last our and our children's lifetimes

IPv6 compacted history

- IPv6 (Internet Protocol version 6)
Also called IPng (Internet Protocol Next Generation)
- No IPv5? (RFC1190 : ST Datagram Mode)



Why IPv6?

- Biggest and actually only advantage: More IP addresses!
- end-to-end, IP-layer authentication & encryption possible (IPSEC)
- for billions of new users (Japan, China, India,..)
- for billions of new devices (mobile phones, cars, appliances,..)
- for always-on access (cable, xDSL, ethernet-to-the-home,..)
- for applications that are difficult, expensive, or impossible to operate through NATs (IP telephony, peer-to-peer gaming, home servers,..)
- to phase out NATs to improve the robustness, security, performance, and manageability of the Internet
- server-less plug-and-play possible
- elimination of “triangle routing” for mobile IP

Differences between IPv6 and IPv4 header

| | | | |
|---------------------|---------------|-------------|-----------|
| Ver. | Traffic Class | Flow Label | |
| Payload Length | | Next Header | Hop Limit |
| Source Address | | | |
| Destination Address | | | |

| | | | | |
|---------------------|----------|-----------------|-----------------|-----------------|
| Ver. | Hdr Len | Type of Service | Total Length | |
| Identification | | | Flg | Fragment Offset |
| Time to Live | Protocol | | Header Checksum | |
| Source Address | | | | |
| Destination Address | | | | |
| Options... | | | | |

blue fields have no equivalent in the other version

IPv6 header is twice as long (40 bytes) as IPv4 header without options (20 bytes)

Summary of changes

- Revised
 - Addresses increased 32 bits -> 128 bits
 - Time to Live -> Hop Limit
 - Protocol -> Next Header
 - Type of Service -> Traffic Class
 - No more broadcast
- Streamlined
 - Fragmentation fields moved out of base header
 - IP options moved out of base header
 - Header Checksum eliminated
 - Header Length field eliminated
 - Length field excludes IPv6 header
 - Alignment changed from 32 to 64 bits
- Extended
 - Flow Label field added

How to notate an IPv6 address

- Full form:

2001:0db8:0000:0000:0000:0000:0000:029a

- Compressed:

2001:db8::29a

- IPv4-embedded:

0:0:0:0:0:fff:192.0.2.42

or compressed:

::fff:192.0.2.42

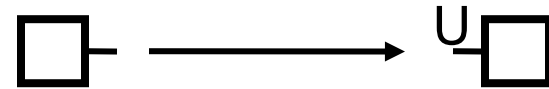
- Documentation IPv6 Prefix: 2001:db8::/32
- Documentation IPv4 Prefix: 192.0.2.0/24
- Localhost ::1

How to notate an IPv6 address #2

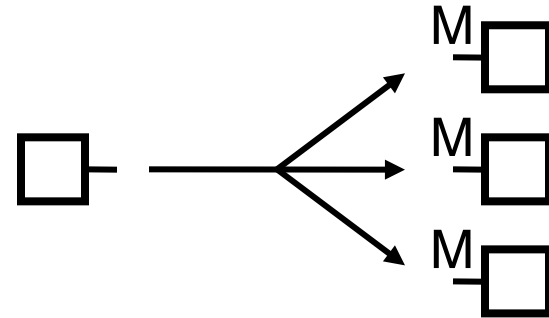
- No more netmasks, only prefixes: `2001:db8::/48`
- Scope qualifiers (%): `fe80::202:2dff:fe2a:3f78%4`
- In URL's: `http://[2001:db8::42]:80`
(square-bracket convention also used anywhere else there's a conflict with address syntax. eg IRC configuration files, SPF records)
- Forward DNS:
`kojiki.ch.example.org AAAA 2001:db8:302::202:2dff:fe2a:3f78`
- Reverse DNS:
`8.7.f.3.a.2.e.f.f.d.2.2.0.2.0.0.0.0.0.2.0.3.0.8.b.d.0.1.0.0.2.ip6.arpa.
PTR kojiki.ch.example.org.`
- `ip6.int` and bitstring notation are deprecated.
- `A6/DNAME` has been marked experimental.

Addressing types

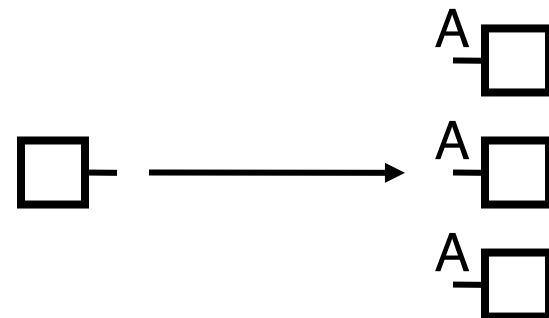
- Unicast
One on one



- Multicast
One to many



- Anycast
One to any



Address type prefixes

- Unspecified
 - used when there is no address
 - Loopback
 - Link Local Unicast
 - Multicast
 - Unicast + Anycast
 - hierarchical
 - /13 - /32 to LIR's (ISP's)
 - /48 or /56 to endusers / sites
- “Site Local” used to exist (fec0::/10)
but this has been deprecated in favor
of ULA.

0000 0000 (::/128)

0000 0001 (::1/128)

1111 1110 1000 0000 (fe80::/16)

1111 1111 (ffxx::/8)

The rest, 2000::/3, which is 1/8th of
total IPv6 space

2001::/16 = RIRs

2001::/32 = Teredo

2002::/16 = 6to4

3ffe::/16 = 6bone*

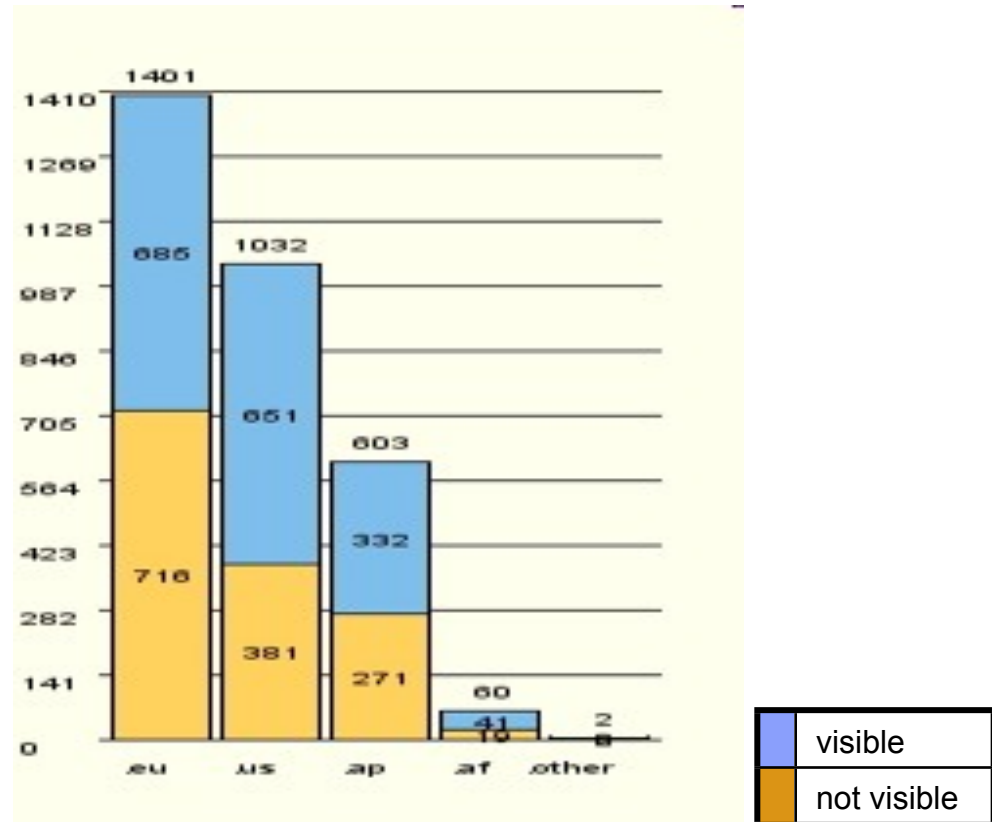
fd00::/8 = ULA

* = 6bone shut down on 6/6/6

<http://www.iana.org/assignments/ipv6-address-space>

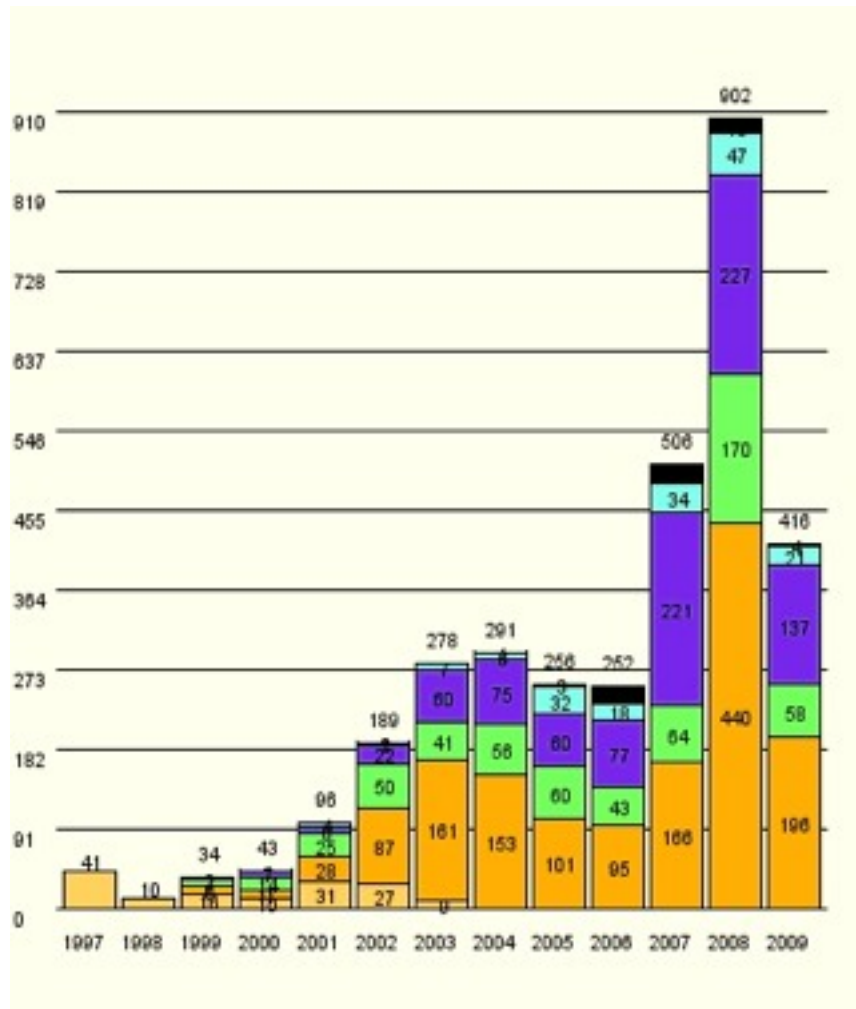
Prefixes around the world

- 1x /13 (US DoD)
- 2x /19
- 6x /20
- 4x /21
- 5x /22
- 1x /23
- 2x /24
- 1x /25
- 5x /26
- 4x /27
- 9x /28
- 4x /29
- 5x /30
- 4x /31
- 2551x /32 (most ISPs)
- ..
- 455x /48 (PI and anycast)
- 83x for Switzerland (45 visible)

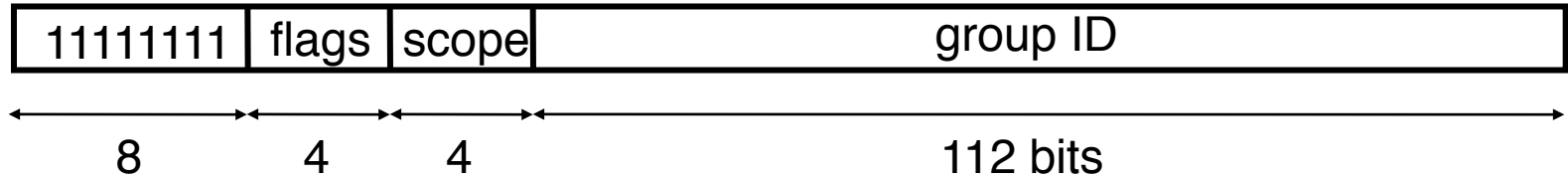


Data Source: <http://www.sixxs.net/tools/grh/growth/>

IPv6 allocation Growth



Multicast addresses



- low-order flag indicates permanent / transient group; three other flags reserved
- scope field:
 - 1 - interface-local (for multicast loopback)
 - 2 - link-local (same as unicast link-local)
 - 3 - subnet-local
 - 4 - admin-local
 - 5 - site-local (same as unicast site-local)
 - 8 - organization-local
 - B - community-local
 - E - global (same as unicast global)
- (all other values reserved)
- Per /48 automatically a globally unique Multicast prefix:
2001:db8:29a::/48 -> ff3e:30:2001:db8:29a::/96

Special Multicast Addresses

- ff02::1 = all hosts
- ff02::2 = all routers
- As such, when you want to hack into a network, the moment you are into your first host you can find out all other hosts on the network by justing pinging ff02::1. Some “secure” Operating Systems thus turn that feature off. Finding the first host is of course a lot of fun, but one can easily go through web/mail/etc access logs to find out where they are.

Subnet Anycast Address

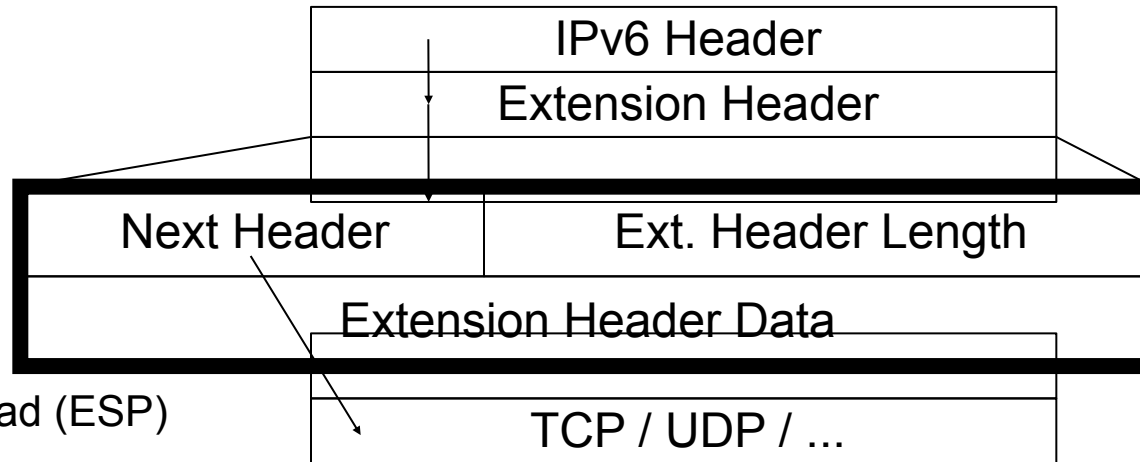
- When one has 2001:db8::/64, then 2001:db8::/128 is the subnet anycast address, which is effectively always 'localhost'.
- Problem when you get 2001:db8::/127, as then you have 2001:db8::/128 or the subnet anycast address and 2001:db8::1, but nothing else.
- This is the reason why one can't use /127's for Point-2-Point tunnels.
- General consensus: use 2x /128, or easier a /64.
- For allocations, ISPs tend to take one /64 for Point-2-Point connections, eg connections to end-users, and then their router is <prefix>::1, while the customer becomes <prefix>::2. When the customer wants more than 1 address, they route them a /64 or /48 to ::2

RFC4193 - ULA

- IPv6 ULA (Unique Local Address)
- RFC4193 Registration
- fd00::/8 ULA Locally Assigned.
It is Unique, but maybe not Unique enough as it has a chance that it is not.
- fc00::/8 ULA “Registered” – but not specified and thus can’t be used.
- Nearly 300 registrations
- Of course not guaranteed, when people don’t check this list it can’t be <http://www.sixxs.net/tools/grh/ula/> for a ‘registry’ so that when one registers there and checks before using one at least knows it is quite unique.
- IPv6 PI is available, but costs a bit of money, but is globally unique.

Header Chains

- The Next Header field describes the type of following the IP header which can point to the next header forming a chain.
- Aligned to 64 bits
- Extension headers:
 - Hop by Hop
 - Destination Options
 - Routing Header
 - Fragment Header
 - Authentication Header (AH)
 - Encapsulating Security Payload (ESP)
 - Upper Layer Header
 - TCP/UDP/ICMPv6/SCTP

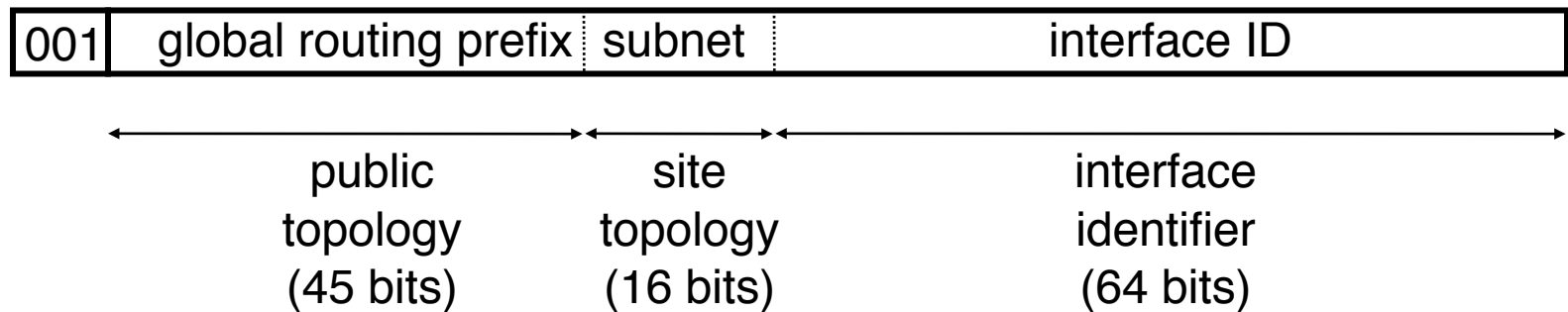


Address properties

- Multiple addresses per interface
 - Always at least one link-local address
 - One or more unicast prefixes
- Addresses can be marked 'deprecated'
 - they should not be re-used by new outgoing connections
 - old connections will keep working
- Router renumbering protocol, to allow domain-interior routers to learn of prefix introduction / withdrawal

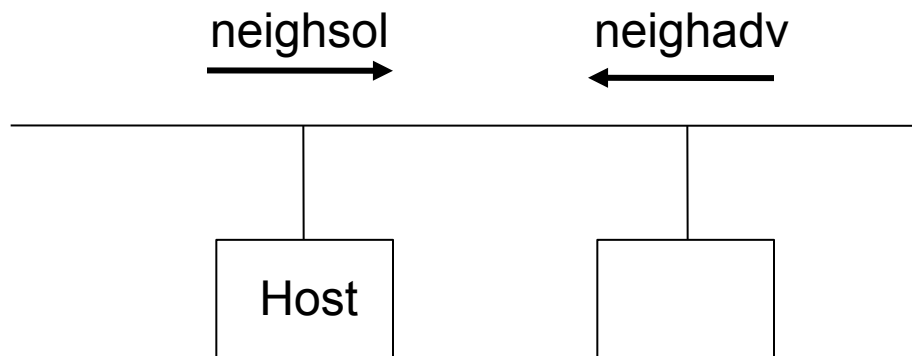
How does a host get addresses?

- Manual configuration (of interface ID or complete address)
- DHCPv6 (configures complete address + optionally services)
- Automatic derivation from 48-bit IEEE 802 address or 64-bit IEEE EUI-64 address as per:
 ::mmMM:MMff:feMM:MMMM (M = Mac address)
 split address in halve, and invert a bit in 'mm'
- Pseudo-random generation for client privacy (RFC3041)
 The latter two choices enable "serverless" or "stateless" autoconfiguration, when combined with high-order part of the address learned via Router Advertisements



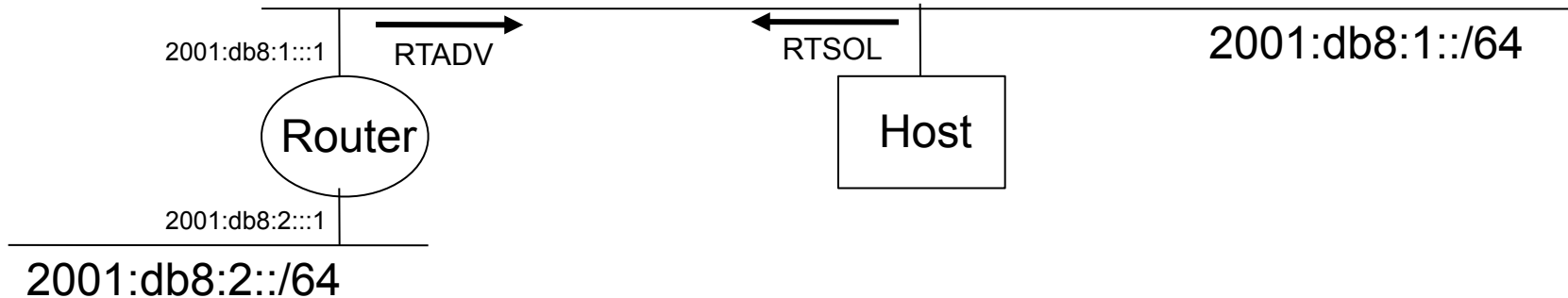
Neighbor Discovery

- Instead of ARP
- Determine the addresses that are on the same link
 - Finding routers
 - Finding directly contactable hosts
- Uses ICMPv6.
- Neighbor solicitation for requesting who are neighbors.
- Neighbor advertisement for announcing that you are a neighbor.
- Duplicate Address Detection (DAD)



Router Discovery/Advertisement

- Router announces periodically which prefix is assigned to the link
- They can learn one or more on-link prefixes
 - Lifetime is included separately for every prefix -> “easy” renumbering
 - Hosts can auto configure themselves using this prefix
- They can learn the router that they can use as a default router
- They can learn if they are required to use stateless or stateful configuration (DHCPv6 for instance)
- Additional information like MTU and hop limit can be learned.



Notes

- No fragmentation in routers
 - Hosts do fragmentation, routers only respond with ICMPv6 Packet Too Big and thus employing Path MTU Discovery.
- Smallest packet size / MTU is 1280 bytes
 - "Layer 2" does the fragmentation when the L3 packets are too big
- Checksums are done usually already by Layer 2, so why repeat doing it in IPv6 where the router has to checksum every packet it would route and thus saving on overhead and calculations.
- IPv4 and IPv6 can coexist, mixing should not be an issue.
- Recommendation to first try IPv6 AAAA's and after that the IPv4 A's when trying to contact a host from an application.

Fun notes

- RFC1888 “OSI NSAPs and IPv6”
Defines how to embed an OSI NSAP address inside IPv6
- RFC4048 “RFC 1888 Is Obsolete”
Makes RFC1888 historic

- Flow labels are simply not defined at all anywhere
- IPSEC is also available for IPv4, but the problem mostly is the uniqueness of the address space (one will find 192.168.0.0/16, 10.0.0.0/8 or 172.16.0.0/12 at home or in their business?)
- Anycast exists also in IPv4 as it is mostly a routing trick
- RH0 (Routing Header Type 0) – same kind of mistake as in IPv4, one could abuse it to let packets be sent in a loop, oops. This ‘feature’ is now disabled in most stacks and routers block these packets.



Supporting IPv6 in applications

Alice @ 192.0.2.2

Bob @ 2001:db8::4

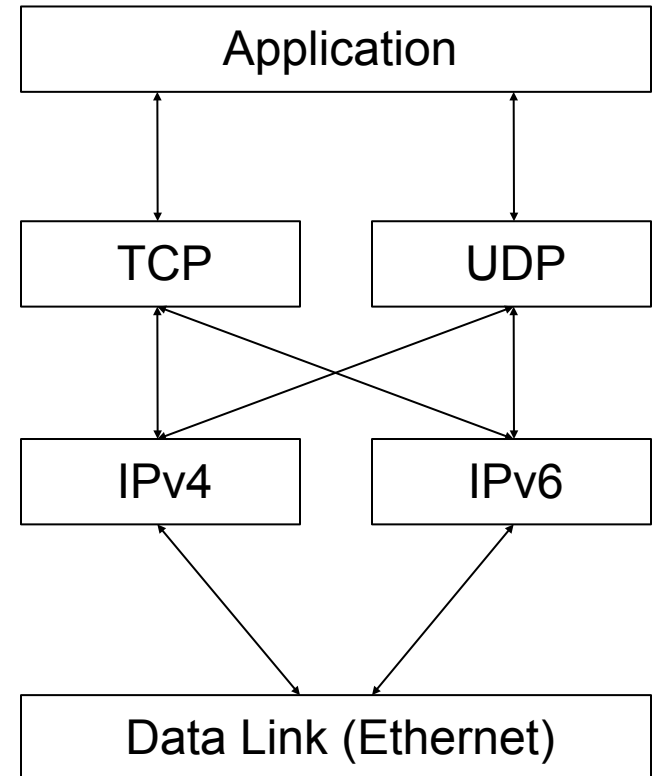
Eve @ 192.0.2.6 and 2001:db8::6

ETH Zürich, Zurich, Switzerland

2012-04-30 | Jeroen Massar <jeroen@massar.ch>

Dual stack operation

- Applications can use both IPv4 and IPv6 at the same time, protocols can coexist.
- DNS can contain both IPv4 and IPv6 addresses.
- IPv6 applications can use IPv4-mapped addresses. Though this only is available when the kernel has a mixed IPv4/IPv6 stack. Windows IPv6 implementation doesn't have this for instance.
- Best option: Address Family independency



Address Family (AF) independency

New functions from [RFC2553](#) and [RFC2292](#):

- `getaddrinfo()`
Getting the addresses belong to a textual identifier
Replaces:
 - `gethostbyname()`
 - `getservbyname()`
 - `inet_pton()`
- `getnameinfo()`
Getting a textual representation of an address
Replaces:
 - `getservbyaddr()`
 - `getservbyport()`
 - `inet_pton()`

Porting Considerations

- Change socket functions
- Adjust logging function so they can handle larger IP address (Don't forget to store both the hostname and the address as changing reverses is as easy as getting a new IP, having two tracking points is better)
- Increase all data member that stores IP addresses in program and in databases/configuration files
- Adjust keyboard and display interface function so they can handle larger IP addresses.
- RFC2732 states that IPv6 addresses in URIs should be delimited by square brackets [] which solves the problem where applications use the colon (:) to distinguish the port from the address (eg [2001:db8::1]:80).
- getaddrinfo() returns 0 on succes, !0 on failure unlike most other unix/posix calls.

Server code example

Daemons should listen on all possible combinations, as PF_UNSPEC, mostly 2 maximum (IPv4+IPv6)

```
int sockets[10];
int makelisten(char *server, char *service)
{
    int i = 0;
    struct addrinfo hints;
    struct addrinfo *res;
    memset(&hints, 0, sizeof(hints)); /* set-up hints structure */
    hints.ai_family = PF_UNSPEC;
    hints.ai_flags = AI_PASSIVE;
    hints.ai_socktype = SOCK_STREAM;
    if ((error = getaddrinfo(server, service, &hints, &res)))
    {
        perror(gai_strerror(error));
        return -1;
    }
    while (res)
    {
        sockets[i] = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
        if (sockets[i] == -1) continue;
        if (bind(sockets[i], res->ai_addr, res->ai_addrlen) == 0 &&
            listen(sockets[i]) == 0) { T++; continue; }
        close(sockets[i]); sockets[i] = -1;
    }
}

int socket = makelisten(NULL, "80");
```

Client code example

Client side program should try to connect to all resolved addresses,

```
int makeconnect(char *server, char *service)
{
    struct addrinfo hints;
    struct addrinfo *res;
    memset(&hints, 0, sizeof(hints)); /* set-up hints structure */
    hints.ai_family = PF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    if ((error = getaddrinfo(server, service, &hints, &res)))
    {
        perror(gai_strerror(error));
        return -1;
    }
    // Loop through all the possibilities
    while (res)
    {
        sockfd = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
        if (sockfd == -1) continue;
        if (connect(sockfd, res->ai_addr, res->ai_addrlen) == 0) return sockfd;
        close(sockfd);
    }
    return -1;
}

int socket = makeconnectio("www.massar.ch", "80");
```


RFC3484 – Address Ordering

- Orders addresses:
 - IPv6 native
 - IPv4 native
 - 6to4
 - Teredo

- Configurable
 - Linux: /etc/gai.conf
 - Solaris: ipaddrsel
 - FreeBSD: ip6addrctl
 - Windows: netsh

Use a socket per AF

```
tcp6    0    0 :::993          :::*           LISTEN        -
```

Does that listen on IPv6 only? -> Not on Linux, there it is IPv4 and IPv6, applications need to know this as they will see incoming connections on IPv4 as ::ffff:192.0.2.42

```
tcp     0    0 0.0.0.0:993    0.0.0.0:*     LISTEN        -
tcp6    0    0 :::993          :::*           LISTEN        -
```

Porting applications / coding new ones

- google(eva ipv6) or <http://gsyc.escet.urjc.es/~eva/IPv6-web/ipv6.html>
- Contains “Porting applications to IPv6 HowTo“ with great explanation and example code.
- Also serves of course as a rather good example for new programs.
- Implementing AF-independent application document by Jun-ichiro ‘itojun’ Itoh (<http://www.kame.net/newsletter/19980604/>)
- The document from the master IPv6 Samurai himself.

Security implications!

- More diverse devices connected, thus possibly also more vulnerabilities.
- Firewalls should be enabled per default for all incoming connections.
- Scanning of address space is not feasible
 - /64 per link
 - /48 or /56 per 'endsite'
- Currently only BSD has a stateful firewall, Linux since 2.6.[5|6]-USAGI. Cisco PIX has it too, but loadbalancing broken.

DOCTOR FUN



The brave new world of IPv6

4 June 2003

Copyright © 2003 David Farley, d-farley@ibiblio.org
<http://ibiblio.org/Dave/drfun.html>

This cartoon is made available on the Internet for personal viewing only. Opinions expressed herein are solely those of the author.



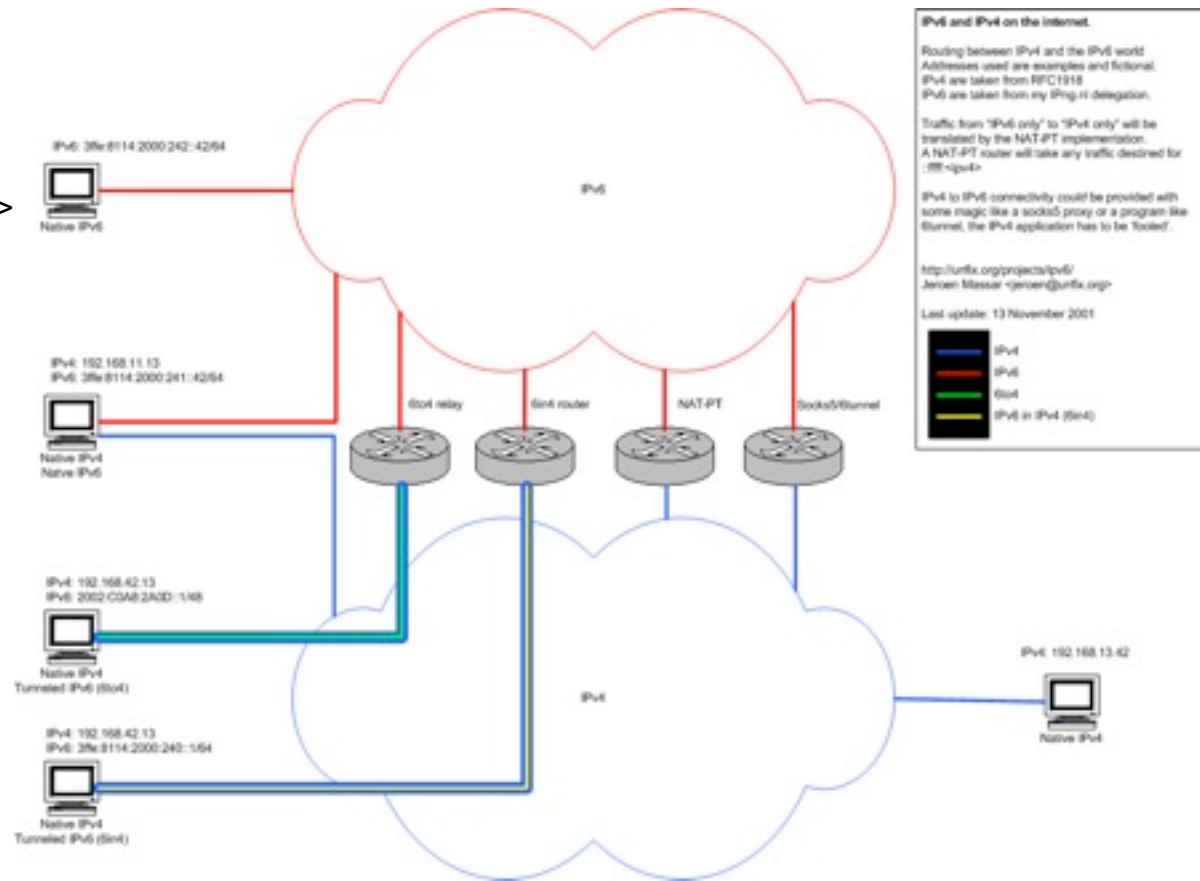
Transition to IPv6

The move to IPv6

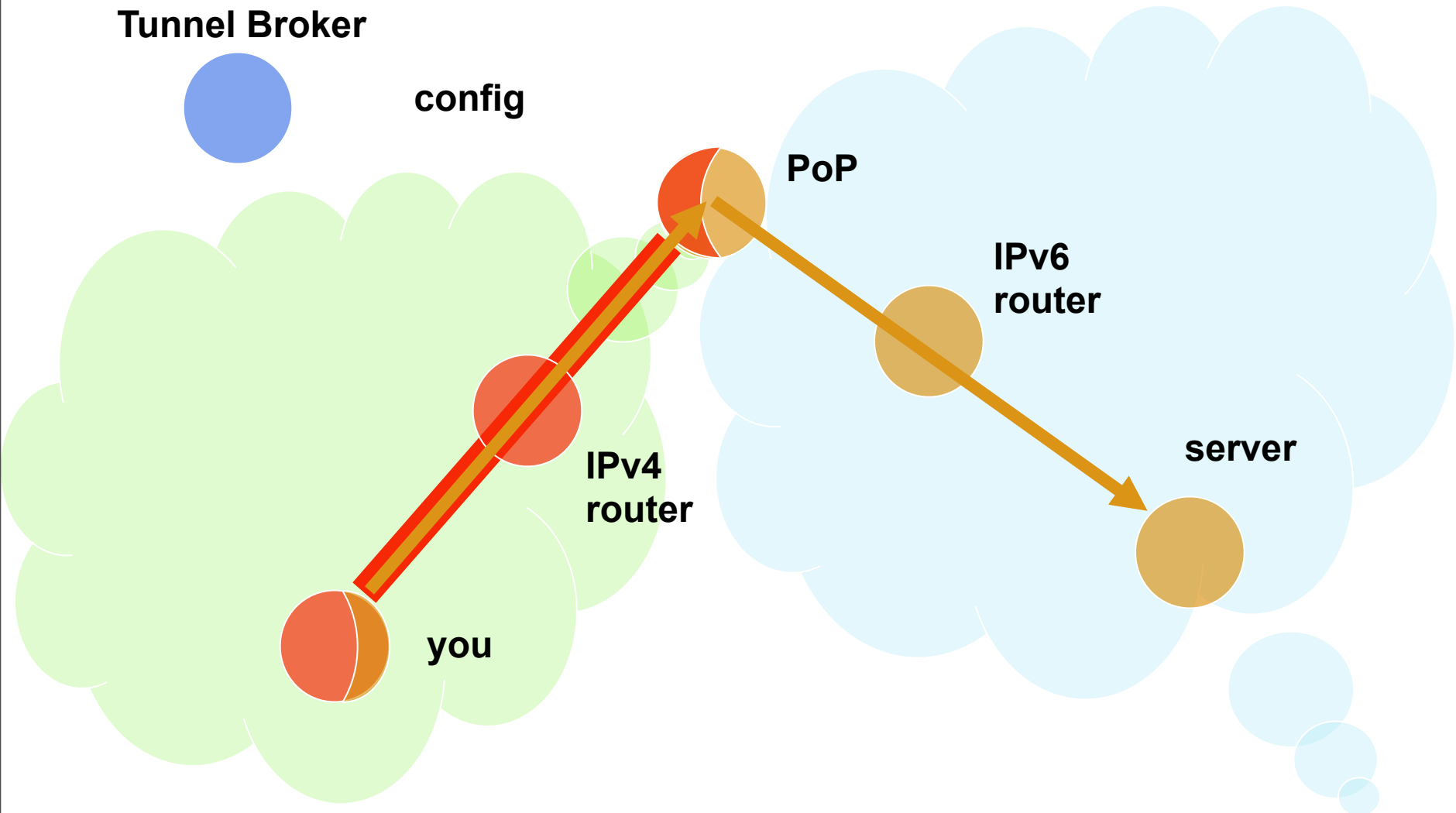
ETH Zürich, Zurich, Switzerland
2012-04-30 | Jeroen Massar <jeroen@massar.ch>

Transition Methods

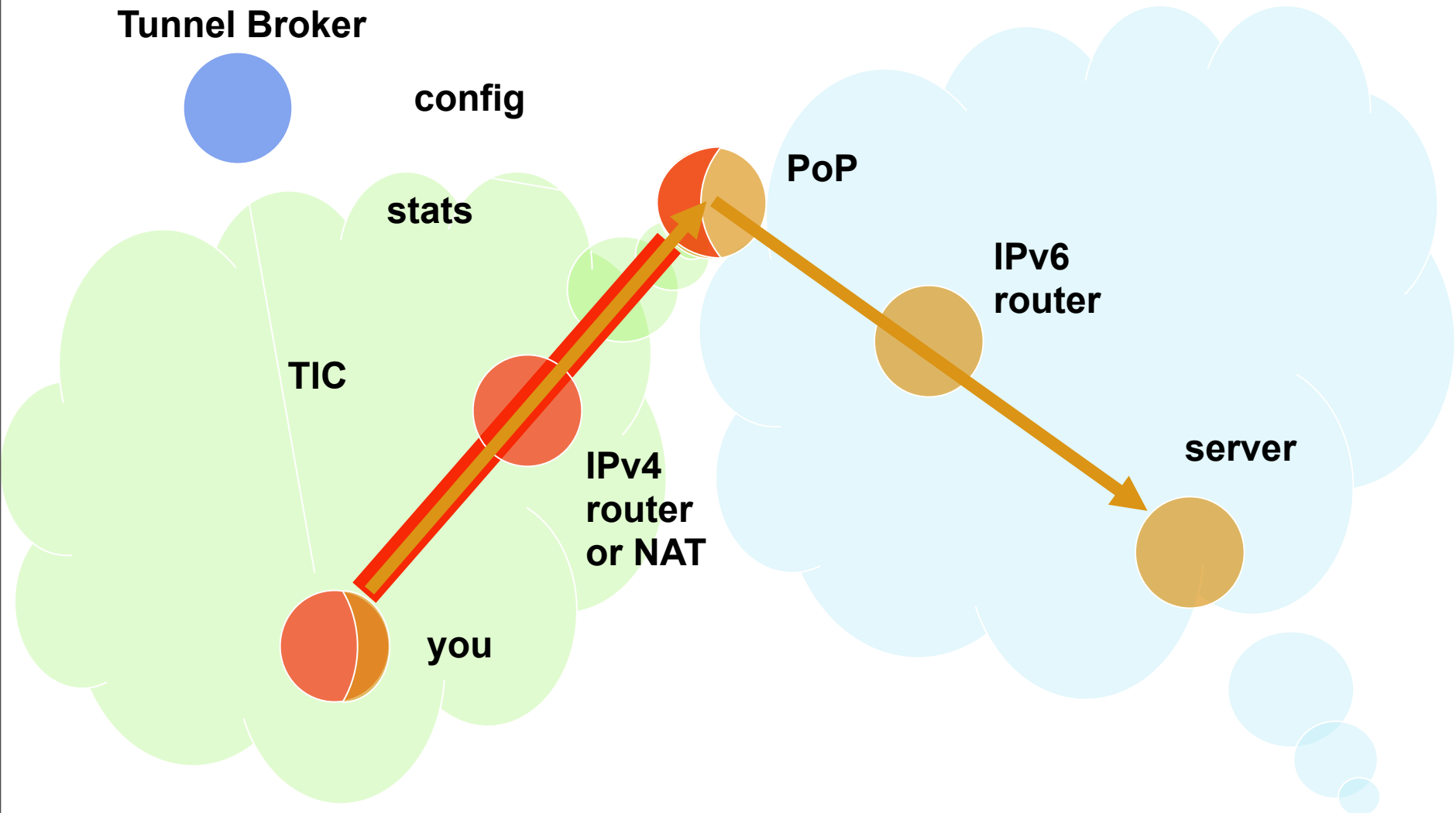
- 6to4
 - 2002:<aabb>:<ccdd>::/48
- Teredo
- ISATAP
 - <64bit prefix>:0:5efe:<aabb>:<ccdd>
- 6in4 (Proto-41/SIT)
- GRE
- tinc / openvpn / ...
- AYiYA
- IPv6 over MPLS/6PE
- DSTM (IPv6 -> IPv4)
- NAT-PT / SIIT
- BIS & BIA
- Faith/TCP-UDP Relay
- Application Proxies
(SOCKS / http / smtp/...)



RFC3053 – IPv6 Tunnel Broker



SixXS Tunnel Broker



Protocol 41

- Protocol 41 = IPv6
- It specifies how to put an IPv6 packet inside IPv4.
- Protocol 41 is static only.
- Protocol 41 doesn't cross NATs.

| |
|-------------------------------------|
| Version: IPv4 |
| Src/Dst: 192.0.2.111 -> 192.0.2.222 |
| Protocol: 41 (IPv6) |

| |
|---|
| Version: IPv6 |
| Src/Dst: 2001:db8::111 -> 2001:db8::222 |
| Protocol: 6 (TCP) |

| |
|-----|
| TCP |
|-----|

6to4

- RFC3056 (6to4) & RFC3068 (6to4 Anycast)
- Uses protocol 41 / static IPv6 tunnels
- Anycast in IPv4 (192.88.99.1/24) though 'private' 6to4 relays exist
- Anycast also in IPv6 as 2002::/16 is only allowed to be announced in BGP as the /16

- Due to the double anycast, it is very hard to determine the flow of packets as they can be 'pulled' all over the place.

6rd

- draft-despres-6rd
- IPv6 Rapid Deployment (RD also from Remi Depres who came up with it)
- Deployed at Free.fr to their 4M user base
- Instead of 2002::/16, uses ISP prefix
 - Eg ISP has existing customers in 192.0.2.0/24 then only 8 bits needed for customers, thus 2001:db8:1100::/40 can be used. The IPv4 address 192.0.2.204 (204 => 0xcc) would then get 2001:db8:11cc::/48.
- Works for Free as they have their own CPE (Cable/DSL modems) and they 'just' had to upgrade all of them with this new custom code.

Proto41/Heartbeat

- draft-massar-heartbeat
- Proto-41 is static. The moment the user unplugs / forced DHCP change, another user can get that IPv4 address. That user then gets proto-41 packets and the firewall tool beeps with warnings, which sometimes results in abuse reports because we are attacking them.
- Allows one to move around proto-41 tunnels automatically or enable/disable them on the fly.

| |
|---|
| Version: IPv4 |
| Src/Dst: 192.0.2.111 -> 192.0.2.222 |
| Protocol: 41 (IPv6) |
| Version: IPv6 |
| Src/Dst: 2001:db8::111 -> 2001:db8::222 |
| Protocol: 6 (TCP) |
| TCP |

| |
|-------------------------------------|
| Version: IPv4 |
| Src/Dst: 192.0.2.111 -> 192.0.2.222 |
| Protocol: 17 (UDP) |
| UDP |
| HEARTBEAT |

AVM Fritz!Box has support for this protocol and of course AICCU and several other mini-clients.

AYIYA – Anything in Anything

- draft-massar-ayiya
- Proto-41 tunnels can't cross NATs.
- Proto-41 tunnels are not authenticated (read: one can spoof them easily)
- Heartbeat runs next-to the proto-41 tunnel. Heartbeat might work, proto-41 might not.
- AYIYA solves these issues by tunneling IPv6 inside IPv4/UDP and signing these packets. (other options: over HTTP(s), DNS or QoS)

| |
|--|
| Version: IPv4 |
| Src/Dst: 192.0.2.111 -> 192.0.2.222 |
| Protocol: 17 (UDP) |
| AYIYA (id = 2001:db8::111, hash, nxt=41) |
| Version: IPv6 |
| Src/Dst: 2001:db8::111 -> 2001:db8::222 |
| Protocol: 6 (TCP) |
| TCP |

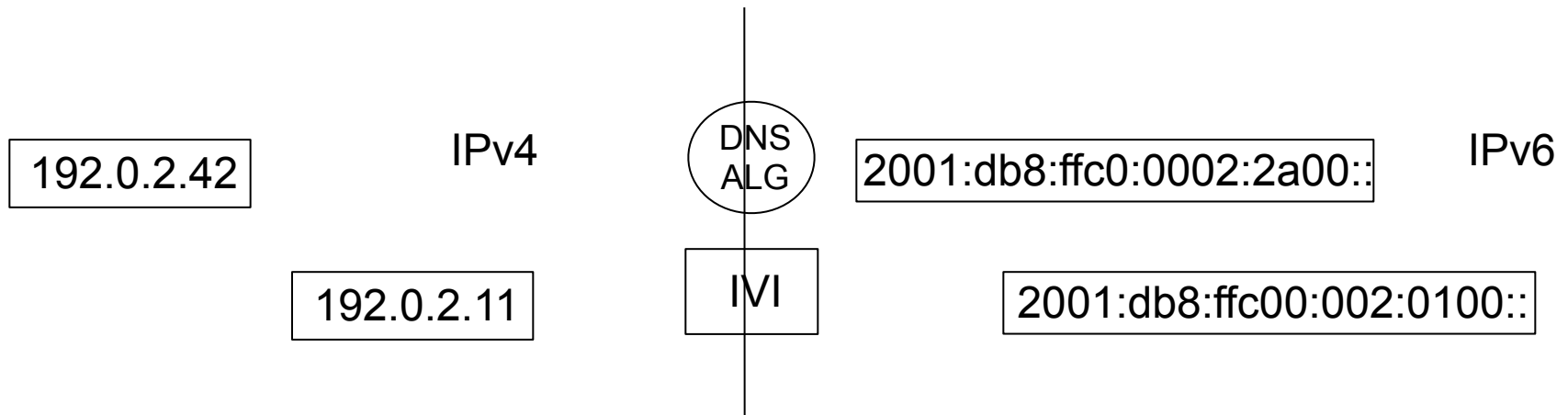
Teredo

- RFC4830 by Microsofts Christian Huitema
- Proto-41 tunnels and thus 6to4 can't cross NATs (one can messily do it with some hacks though).
- Meant to give IPv6 connectivity everywhere, as that is what Windows 2008 and up require. Also good for Xbox as it avoids having to solve NAT over and over again.

| |
|---|
| Version: IPv4 |
| Src/Dst: 192.0.2.111 -> 192.0.2.222 |
| Protocol: 17 (UDP) |
| Teredo |
| Version: IPv6 |
| Src/Dst: 2001:db8::111 -> 2001:db8::222 |
| Protocol: 6 (TCP) |
| TCP |

Solving IPv4 to IPv6 and back: IVI

- Prefix-specific and Stateless Address Mapping (IVI) for IPv4/IPv6 Coexistence and Transition (draft-xli-behave-ivi)
- Xing Li, Maoke Chen, Congxiao Bao, Hong Zhang and Jianping Wu
- Name trick: Roman Numerals: IV = 4, VI = 6 => IVI = 46
- Allows IPv4 to connect to IPv6 and IPv6 to connect to IPv4
- Semi-refresh/combination of NAT-PT and SIIT
- Per-ISP prefix (eg 2001:db8:ffaa:bbcc:dd::)
- DNS ALG which fakes AAAA or A record when needed



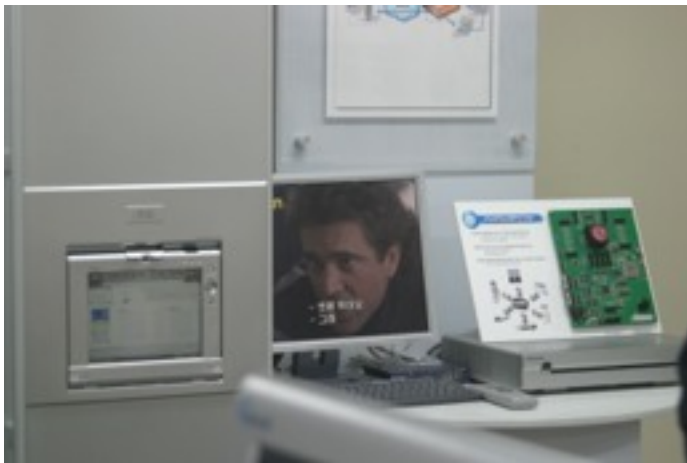
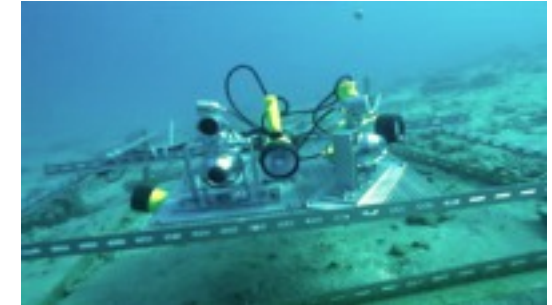
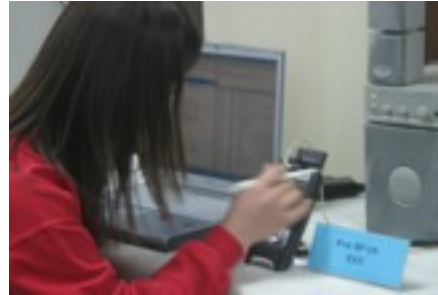


The future with IPv6

The toys to come

ETH Zürich, Zurich, Switzerland
2012-04-30 | Jeroen Massar <jeroen@massar.ch>

IPv6 Toys: Home automation, fridges, sensors, etc



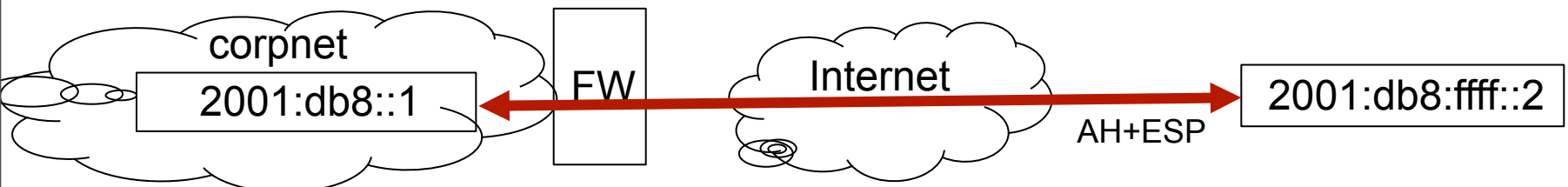
For more:
google(IPv6 toys)
google(IPv6 cool)

IPv6 Toys: \$ telnet plant



Microsoft Direct Access

- How to do 'firewalling' without having to configure prefixes.
- You have your host anywhere, just have IPv6, be that native, proto-41, Teredo, 6to4 or through a Tunnel Broker, could be remote or in the office.
- Sign every packet you sent/receive from your client to the server with an IPSEC-AH.
- Firewall in the middle allows any correctly signed and validated packet, of course only accepting keys that it knows.
- Use IPSEC-ESP to secure packets so that nobody can snoop them.
- Key distribution happens using Active Directory.
- The Direct Access tool allows one to directly access any resources one has at the workplace from any other location on the (inter-)net.
- Same method can be applied to any kind of connection of course.



Problems with IPv6 (and thus fun research for us)

- Broken IPv6 implementations, but:
 - <http://v6fix.net>
 - “Test” function in AICCU (<http://www.sixxs.net/tools/aiccu/>) to see what breaks and where (coming soon)
- IPv4 will run out before IPv6 is deployed everywhere
 - Carrier Grade NAT (CGN)
 - Tunneling
 - Accessing IPv4 from IPv6 and IPv6 from IPv4
- Routing Table Growth / handling of large amount of prefixes
 - This was already an issue partially in IPv4 but will become a larger problem with IPv6; especially when everybody will want to do IPv6 PI, which is now available in ARIN and RIPE regions.
 - With IPv6:
 - Every ISP gets at least a /32
 - Every End-site gets a /48 (or /56 for endusers in some regions)
 - A single /32 contains 65536 /48's; with 10.000 ISPs, that would be $10.000 * 65536 = 655.360.000$ routes when de-aggregated: Way more memory in all routers needed, faster CPUs (due to SPF recalculations) etc

The IPv6 Internet Today

- Google has a 'trusted tester program' (<http://www.google.com/ipv6/>)
- Wikipedia has a similar signup-first program
=> SixXS is enrolled in both
- PirateBay has native IPv6 (<http://ipv6.piratebay.org>) also for their trackers.
- Various Torrent clients support IPv6 and some (uTorrent) can enable Teredo for you so that it automatically works.
- <http://ipv6gate.sixxs.net> and <http://ipv4gate.sixxs.net> for getting to IPv6 content from IPv4 and vice versa.
- 6 Native IPv6 end-user ISPs in Switzerland
see <http://www.sixxs.net/faq/connectivity/?faq=native> for all of them
- Your ISP doesn't do IPv6? Get a tunnel, eg <http://www.sixxs.net> ☺

IPv6 in 2011

- June 8 2011, World IPv6 Day

Akamai, Comcast, AT&T, Google, Time Warner Cable, Yahoo!, Microsoft and others tests for 24 hours how many clients they would lose with enabling IPv6 on their frontend properties



IPv6 in 2012

- 6 June 2012: World IPv6 Launch
Google, Facebook, Yahoo!, Akamai, Limelight, Microsoft, AT&T, D-Link, Cisco and lots of others will enable IPv6 per default on their properties





The End

Comments or questions?

ETH Zürich, Zurich, Switzerland
2012-04-30 | Jeroen Massar <jeroen@massar.ch>

References

- IETF – The Internet Engineering Task Force
<http://www.ietf.org>
- IRTF – The Internet Research Task Force
<http://www.irtf.org>
- IVI
<http://www.ivi2.org>
- Patrick Fältström's IETF photo site
- SixXS - IPv6 Deployment & Tunnel Broker
<http://www.sixxs.net>