Tkkrlab IPv6 workshop

OpenFortress*
digital signatures

how is ipv6 different?

The basic idea is simple:

* Addresses 32 bit → 128 bit
* Learn from the past

The results are quite big:

* Internet is fully transparant again
* Usable peer-to-peer functionality
* IPv4 and IPv6 are incompatible

living side by side

IPv4 and IPv6 cannot talk to each other

- Desktops run both, side by side
- As a rule, try IPv6 first and fallback to IPv4
- Embedded apps usually make a choice
- Routers are often in the way

There are all sorts of transitioning techniques

- Tunneling: Pack IPv6 into IPv4
- Translation: NAT64, DNS64, SIPproxy64

address and prefix syntax

* 128 bit = 8 groups of 16 bits
* each 16 bit is in hexadecimal, separated by :
* up to one ”filler” with zeroes looks like ::

Example addresses:

```
* 2001:db8:123:567:102:11:16:20
* 2001:db8:0:0:0:0:12:13         = 2001:db8::12:13
* 0:0:0:0:0:0:0:1                = ::1
```

address and prefix syntax

Subnets fixate the initial `n` bits with the CIDR notation `/n`

* `2001:db8::/32` covers example addresses
* `::1/128` is `localhost`
* `ff00::/8` is for multicast
* `2000::/3` is for unicast (so, normal use)
* `fe80::/10` is for local addressing

Prefixes are used for routing, BGP can merge them.

address and prefix syntax

Routing parties process short prefixes, like `/32`

* `2001:610::/32` is handed out by SURFnet BV

Individual users get a longer prefix

* `2001:610:7a6::/48` belongs to Ecocentrum EMMA

End users can distinguish separate networks if they want

* `2001:610:7a6:7::/64` for the food store
* `2001:610:7a6:8::/64` for the plants business
* `2001:610:7a6:9::/64` for MTB Reparatie
* `2001:610:7a6:5060::/64` for telephony

address and prefix syntax

* Router interfaces advertise a /64 prefix

```
# /etc/radvd.conf
interface eth0
{
  AdvSendAdvert on;
  AdvManagedFlag off;
  prefix 2001:db8:66f:0::/64 { };
  RDNSS 2001:db8:66f::5 2001:db8:66f::6 { };
};
```

* Router sends these only rarely
* Upcoming interfaces inquire after routers

address and prefix syntax

The last 64 bits are usually determined by the host

* `PREFIX::1` is still common for routers, similarly servers
* `PREFIX:xxxx:xxff:fexx:xxxx` for autoconfiguration

Autoconfiguration?

* Router advertises `/64` prefix, router, [nameservers]
* Attach MAC address with `ff:fe` filler
* First MAC byte `^=0x02`
* Ask neighbours if address is available
* Defend address from then on

address and prefix syntax

Autoconfiguration... no DHCPv6 then?

* No need, but it is possible

* Router advertisement can set a `ManagedFlag`

* DHCPv6 can help with service location

transitioning techniques

Tunneling:

* IPv4 `proto 41`: IPv6 inside IPv4
* No NAT traversal (it is not TCP, UDP, or ICMP)
* Dependent on co-operative router

transitioning techniques

`proto` 41 used for `6in4` tunnels:

* Have a router unpack it

* Linux interface type `sit`, BSD calls it `gif`

```
# /etc/network/interfaces
iface sixxs inet6 v4tunnel
      address 2001:db8:123:456::789
      netmask 64
      local 192.0.2.12
      endpoint 192.0.2.163
      ttl 64
```

transitioning techniques

`proto 41` used for `6to4` tunnels:

* Addresses look like `2002::/16`
* Following 32 bits are an IPv4 endpoint
* In the endpoint, receive and unpack `proto 41`
* Packets from `2002::/16` can be sent to `192.88.99.1`
* `192.88.99.1` Is an *anycast* address

The `RD` variant can have different prefixes, default routers

a mistake named teredo

* Teredo is a ”specification” created by Microsoft
* Addresses look like `2001:0000::/32`
* Un*x implementation is called `miredo`
* Only intended as a last resort fallback

a mistake named teredo

One problem with Teredo is:

* Very slow initial connections
* Delays discovery if IPv6 works
* Performance degradation if IPv6 is preferred
* Makes people switch off IPv6

And if that wasn't enough:

* NAT problems reflect on Teredo connectivity
* Teredo is only suitable for client-to-server, not peer-to-peer
* Teredo makes IPv6 inherit IPv4-specific problems

beyond nat: freenet6 tunnels

Freenet6 offers free tunnels

* No subscription needed
* Dynamic IPv6 assignment
* Poor performance (500 ms roundtrips)

They use the TSP protocol

* Standardised by Freenet6 in RFC 5572
* Their software does not comply to their own standard
* Independent implementation on `public-tsp.org`

beyond nat: sixxs tunnels

SIXXS hosts very good tunnels

* Proper registration required
* Fixed IPv6 assignments
* Static or dynamic tunnels
* Tunnels are pointopoint
* /48 Subnets can be routed over tunnels

Dynamic tunnels use a protocol named AYIYA

* Widely adopted implementation is AICCU
* IPv6-in-AICCU-in-UDP-in-IPv4
* Synchronise watches (use NTP)
* NAT traversing (includes keepalives)

security issues

* Tunnels: Check addresses!

* No more NAT: Stateful firewalls!

try it out

* Attach a computer, see autoconfiguration at work
* Configure name service (DNSSEC is available)
* Use `ping6`, `traceroute6`, `dig` *`aaaa`*
* Use `ip -6` instead of `ifconfig`
* Connect hosts with `6in4` or `6to4`
* Visit `rijksoverheid.nl` over IPv6
* Remind government of their comply-or-explain policy on IPv6
* Use Google over IPv6 (manually: `ipv6.google.com`)
* Lookup IPv6 addresses in `whois.ripe.net`
* Setup a tunnel to `freenet6.net` or `sixxs.net`
* Setup a subnet over that tunnel
* Make an IPv6 phone call through `SIPproxy64`
* Advertise your routes to your neighbours using `radvd`
* Does your OS accept multi-homed IPv6?

info@openfortress.nl

http://openfortress.nl

OpenFortress*
digital signatures