Why do we need a new IP?

- Address space is almost depleted
  - IPv6 addresses are four times as wide

- Efficiency
  - Header has been simplified
  - to support faster routing.

- Ease of configuration
  - Neighbour and Router Discovery
  - Stateless Address Configuration
  - Dynamic Host Configuration

- Real-time apps

- There is no security model in IPv4
  - Authentication
  - Encrypted payload

Addresses in IPv6

IPv6 addresses are 128 bits wide, quite a lot!

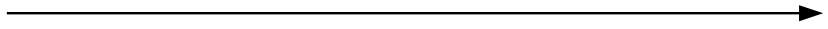
Notation: 8 groups of two bytes each in hex.
So: **`x:x:x:x:x:x:x:x`**
A single sequence of contiguous zero-bytes may be written as **`::`**
So: **`7546:0:0:0:0:0:0:1 == 7546::1`**

There is no class concept in IPv6.
Routing is always expressed as an address starting with a certain bitpattern.

`FE80/10` Only if the first 10 bits contain...

Pre-allocated ranges:

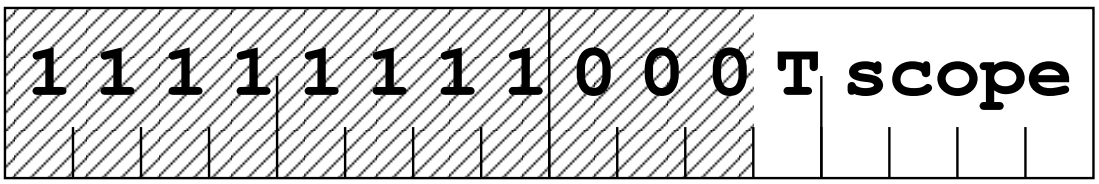
`FE80/10`	link locale addresses (**`= FE80`** ... **`FEBF`**)
`FEC0/10`	site locale addresses (**`= FEC0`** ... **`FEFF`**)
`::1`	loopback
`::`	address not established yet
`2000/3`	aggregatable global unicast address

No broadcast addresses!
There is multicast 

Multicast Addresses

Addresses starting with **FF00/8** are multicast. Messages send to such an address will be sent to all host listening on that address.

The first two bytes of a multicast look like this:



T: 0 == well known
scope: who is supposed to get the message

- 1 node-local
- 2 link-local
- 5 site-local
- 8 org-local
- E global

Examples:
FF02::1 All nodes on link
FF0E::1 All nodes on the Internet :-)
FF02::2 All routers on this link

The link local address

Intended for communication across a single link, must be formed for each interface!

Prefix **`FE80/10`**

address becomes **`FE80::token`**

often the token is linked to the
link layer (or MAC) address

Example:

`FE80::0a00:20ff:fe07:9757`

Before using the address it must
be tested for uniqueness (MAC-addresses are supposed to be unique...)
= Duplicate Address Detection (DAD)
Send Neighbor Solicitation ICMP (NS)
from unspecified host to
Solicited Node Multicast:

`::  ->  FF02::1:FF07:9757`

Multicast

Link scope

Last three bytes
of IPv6-address

Answer: Oh, Oh!

Nodes

A node is either a router OR a host
(possibly multi-homed).

Hosts are dumb.
Routers are smart.

Hosts do not require a network administrator

Hosts configuration is done
with the help of routers
(they are going to need a lot of help)

Now which of the two is the dumb one??

Router Solicitations

Hosts request information from routers
by sending a Router Solicitation to
the All Routers Multicast.

Router(s!) respond by sending
Router Advertisement(s).
They contain:

- Prefix to use with token
- Prefixes of all the hosts hooked up to this link
- I am (one of your) default router(s)

Also

- MTU to use on the link
- Timers (addresses are not forever)
- I am a Home Agent (used with mobile-IP)
- Use DHCPv6 as well

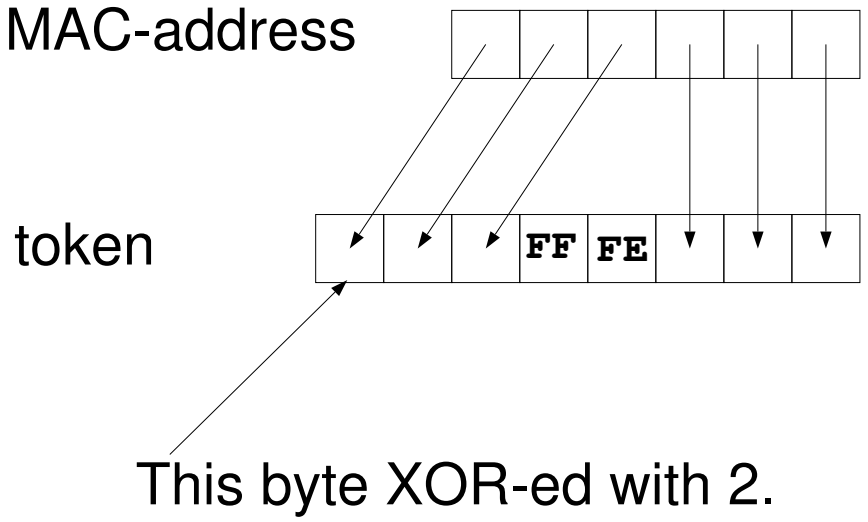
Common adresses for nodes

Every interface may (and will) have several addresses, typically of different scope:

link local:
    FE80::token
site local:
    FEC0:site_routing::token
global (from AGUA):
    3218:xxyy:yyyy:site_routing::token

The token is always 64-bit and is supposed to be the EUI64 identifier of the interface.

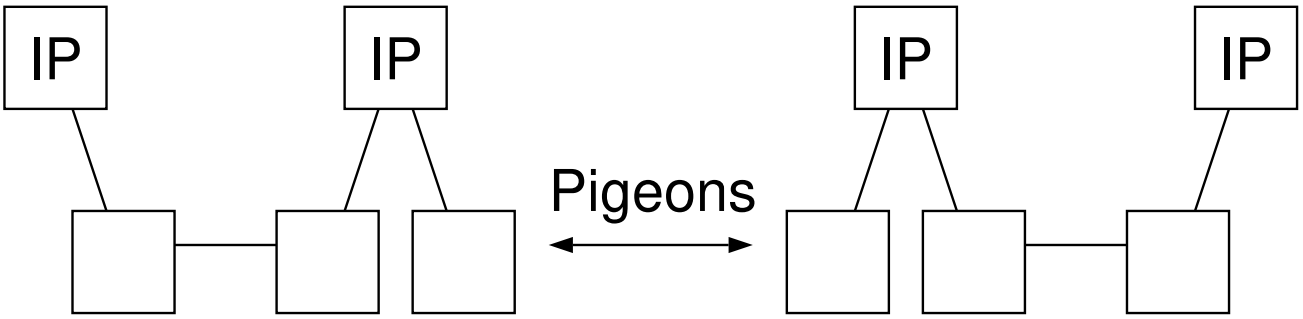
For Ethernet this identifier is constructed as follows:



Why the fuzz?

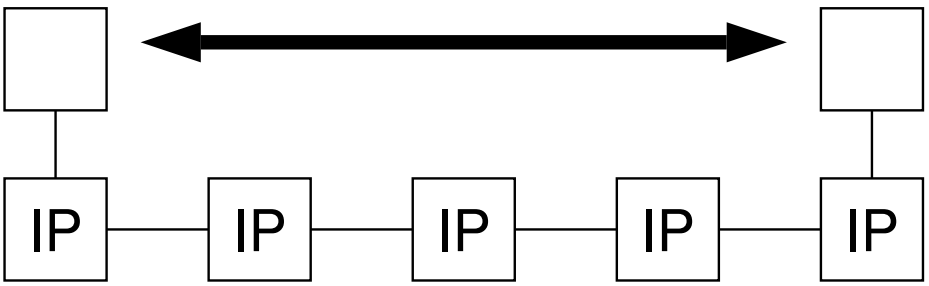
Network protocol plays critical role.

You can *add* a new DLL e.g.:



You can, gradually, introduce a new HTTP version: clients and servers using it can talk

wherever they are



However, all nodes en-route ****must**** use the same networklayer.

Why not switch on januari 1st 00?

- It (the Internet) is to d**n big!
- There are bound to be problems with vendor X's implementation.
- Vendor Y might not be ready in time.
- Users will not upgrade "just for IPv6" (what if their business critical app doesn't run under the new OS version?)
- It is a big risk for ISP's as well.
- I would like to "play" with it first...

Rolling updates

You want hosts to use both IPv4 and IPv6.

On an Ethernet apps can pick their preferred IP version.

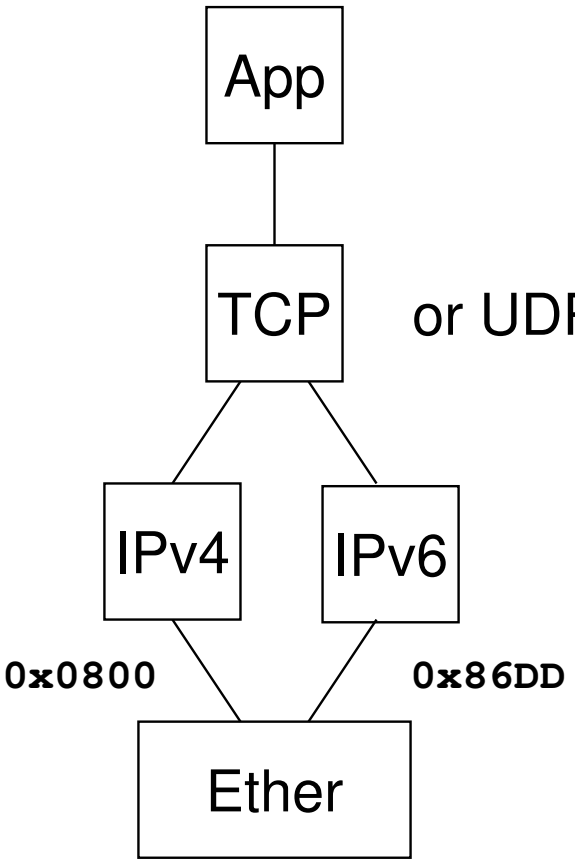
*** The transport layer stays the same ***

So IPv4 and IPv6 can hand off their payload to the same transport layer.

(the pseudoheader for checksums contains the source and destination addresses and therefore will be different).

Dual Stacked nodes

When we have hosts with dual stack but
the same transport layer we can visualize it like this:



If the app is IPv6 based an IPv4 address
will be mapped onto the IPv6 address spaces:

192.168.14.37 becomes **::FFFF:C0A8:0E25**
C0 A8 0E 25

This called an IPv4_MAPPED address.

On a dual stacked host you need
only change the apps.

Inetd

An IPv6 aware version of inetd will of course be provided for you.

If you start your network servers from inetd you might be able to avoid rewriting them. Provided:

- The app is not interested in the peer address
- The server is never started standalone.

(The **`sockaddr`** structure had to be changed, IPv6-addresses were simply to big)

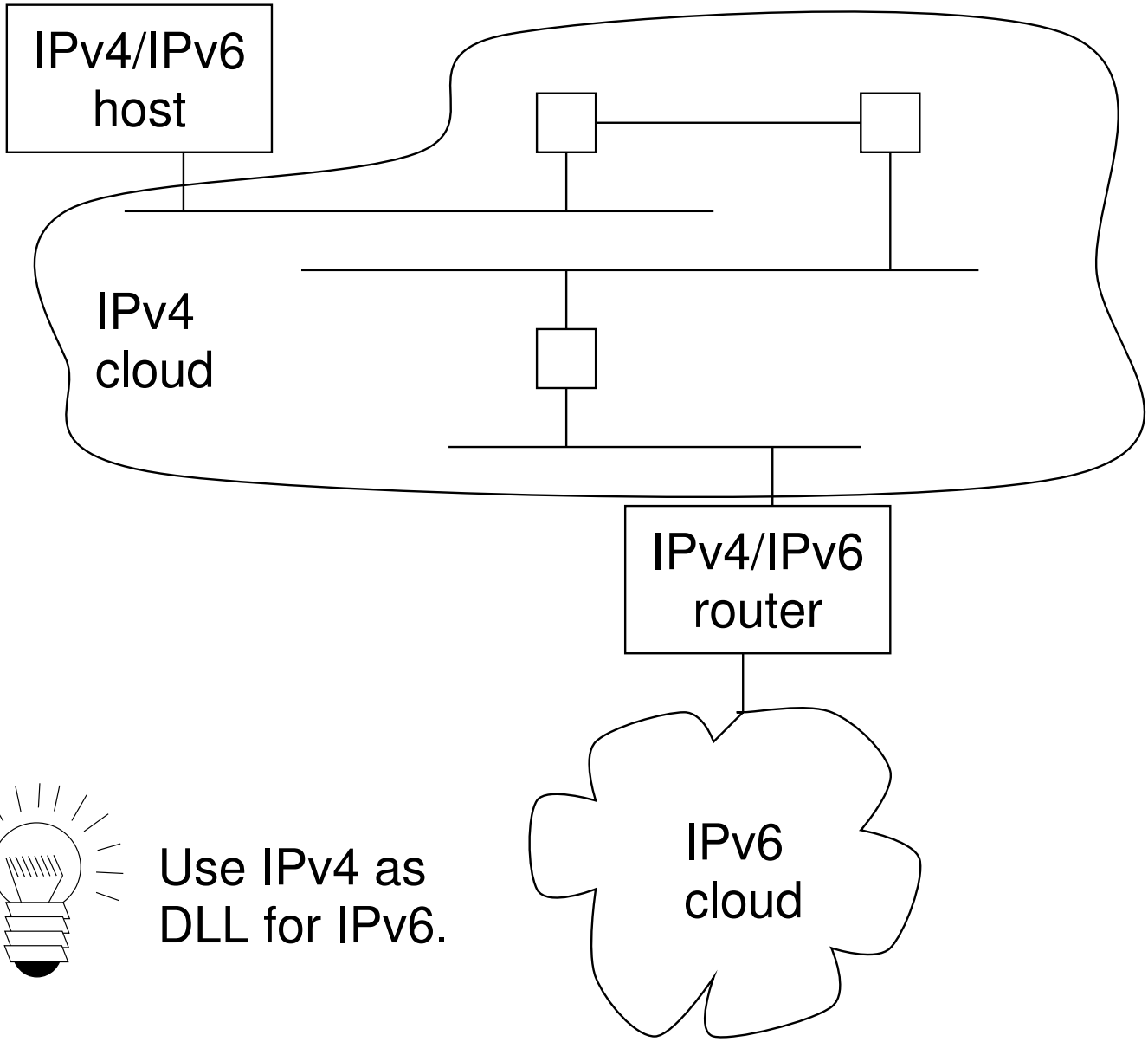
An example of an inetd.conf file (Sun):

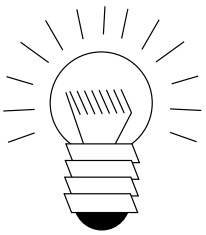
```
myecho stream tcp6 nowait\
    root /bin/echo echo IP version SIX
```

Changing your apps is the best thing to do. It is a minor coding effort and they can be made to work in the traditional IPv4-only situtation as well.

Leaving the ivory tower

By now we have a host running IPv6
and somewhere on the Internet there exists
a router with *real* IPv6 connectivity:





Use IPv4 as
DLL for IPv6.

This technique is called tunnelling.

6over4

How does an IPv6-host find a router on
any datalink layer?

He sends a router solicitation.

How do you find its datalink layer address
(here IPv4-address)?

ARP?? NO!
IPv6 uses Neighbor Solicitation ICMPs.

Both these requests are send to multicast addresses.
Therefore any multiple access network
needs a broadcasting or multicasting facility
to operate without manually having to help hosts.

Does the Internet support that?

Do you?

Automatic tunnels

When we lack multicast capability we could code the DLL(IPv4)-address in the IPv6-address (much like Decnet Phase IV)

192.168.14.37 -> **::C0A8:0E25**
C0 A8 0E 25

We call this an IPV4_COMPATIBLE address

Looks very neat but:

- Pure host-to-host (well technically...)
- Heavily based on IPv4 (so IPv4 will not go away)

It is an easy presence on the IPv6-based Internet, just add

host AAAA ::C0A8:0E25

to your nameserver (BIND 4.9.4 or up).

Configured tunnels

The third possible technique is to handconfigure tunnels.
You supply both tunnel endpoints with the IPv4 address of the other side.

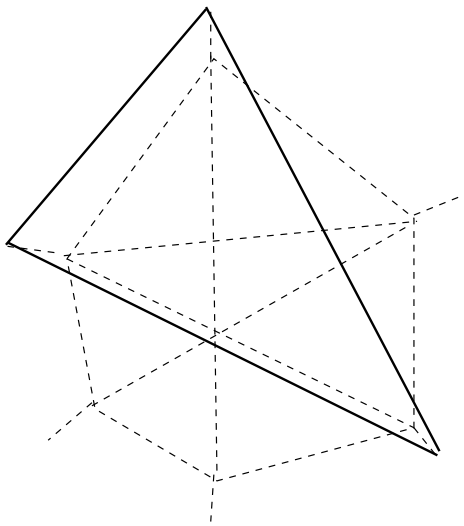
These tunnels are typically used to connect a pair of routers.

They also look very much like links in their own right.
Routers can use a routing protocol to exchange IPv6-routing information among themselves.

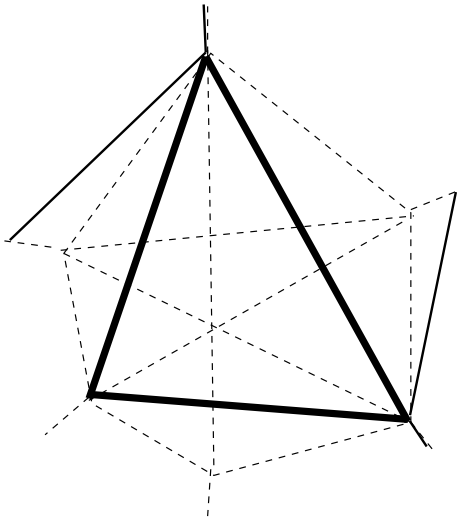
This is the approach taken in 6bone.

http://www.6bone.net/

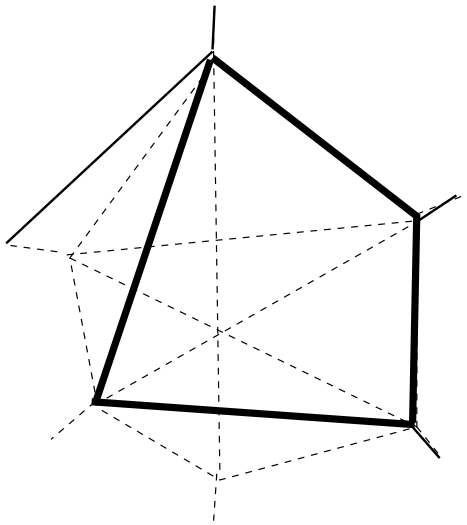
Configured tunnelling in real life



Researchers connecting their sites.



ISPs kick in, researchers (and others) tunnel to their ISP or to an ISP willing to service them.



Some ISPs move their tunnel(s) to correspond with their IPv4 links.

They can decide now to replace the tunnel by IPv4 + IPv6 links.

Comparing tunnelling techniques

- Automatic tunnels
  - very host-to-host
  + easy to do
  - builds on IPv4

- 6over4
  + easy to do (iff you have IPv4-multicast)
  - locally sustaining IPv4
  + usefull “on the fringe”

- Configured tunnels
  - lot of work (you need to find someone to setup a tunnel to and coordinate your efforts)
  + they look like real links.
    At first they might run from research institute to research institute.
    Then they are moved to ISP to ISP, building a pseudo backbone, leaves tunnel to their ISP.
    Next the tunnels might correspond to an actual link between two ISPs.
    The link might be made IPv4/IPv6 capable.
    Much later the link might be made IPv6-only with IPv4 remnants tunnelled over IPv6, until, finally, IPv4 dies off alltogether.

Configured tunnels are real transition pushers!

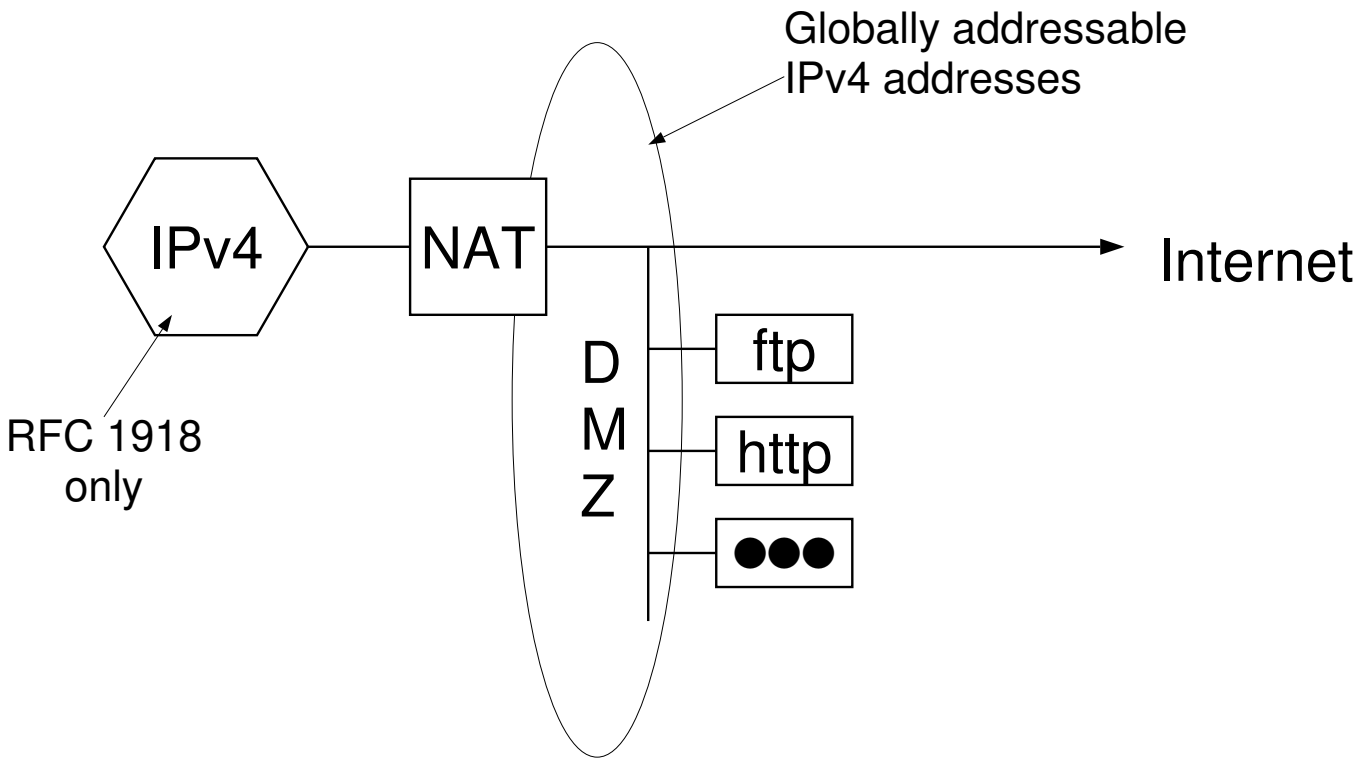
Typical sites now

Now a days sites have to cope with two important issues:

- Security
- Address depletion

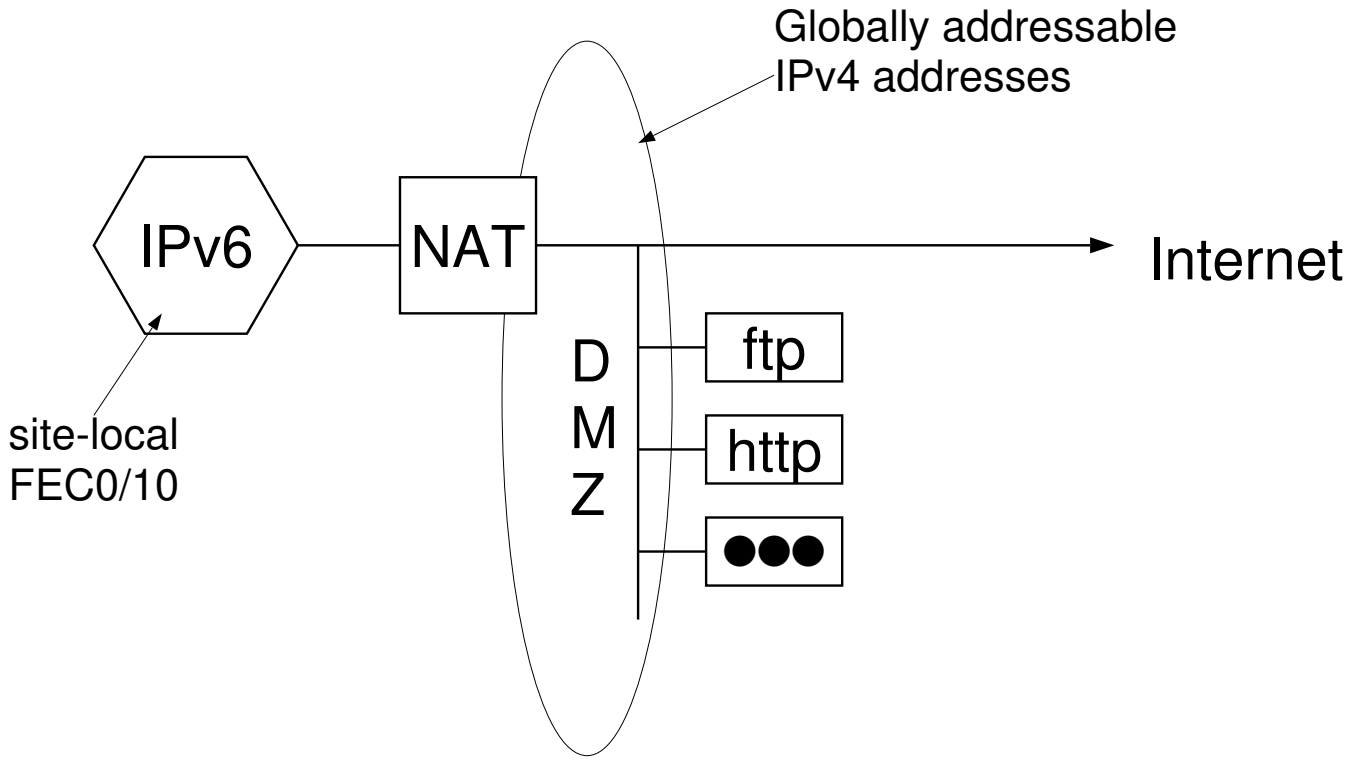
Security is solved using a firewall, address depletion with site local (RFC 1918 1597) and the use of Network Address Translation also dubbed as masquerading.

In a picture:



NAT between 4 and 6

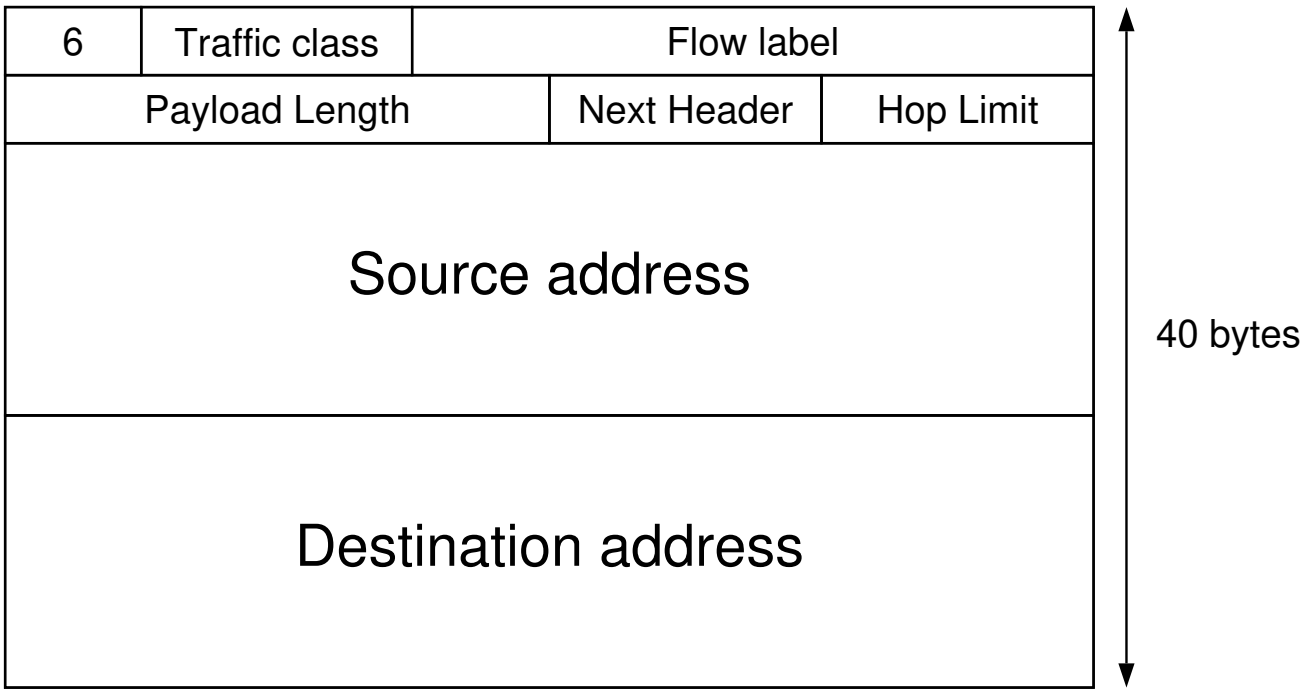
Can we replace our RFC 1918 addresses with IPv6 site-local addresses?
(They do serve the same purpose.)

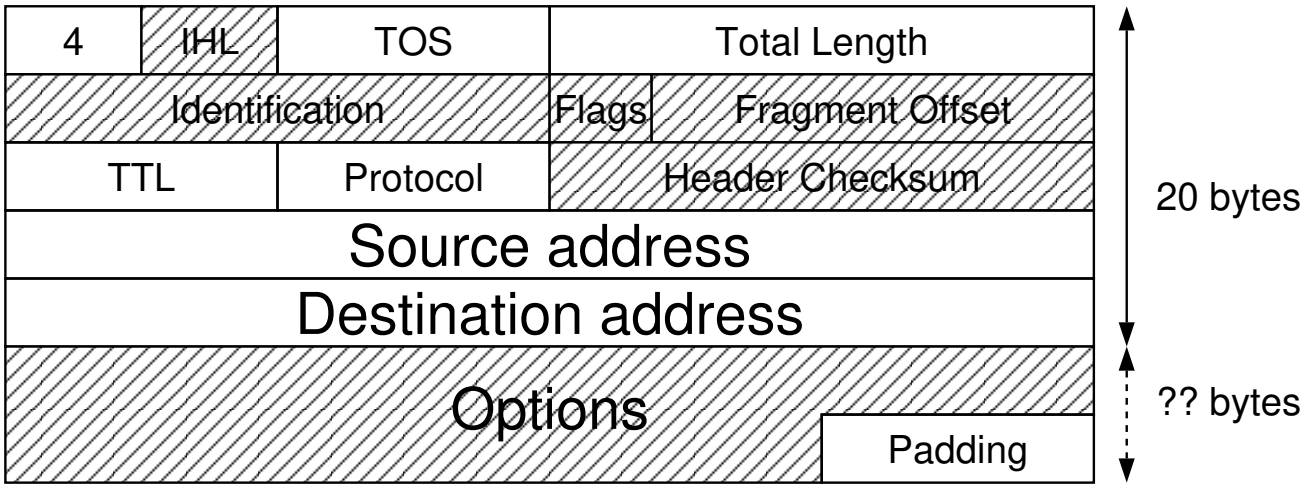


Yes we can iff we can translate IPv6 headers into IPv4 headers!

Let's have a look at the headers...

IPv6 vs IPv4 headers





- Fragmentation, separate header in IPv6
- Traffic class <-> TOS do-able
- Options (pain in the a**), separate header!

NAT 6-4 and 4-6 is possible!

Why NAT is a bad idea

In the Internet today NAT solves the problem
of address depletion allright.

It breaks the idea of peer to peer communication.

When you add authentication you are authenticating
the peer, not the NAT.
Moreover you won't even be needing a firewall
setup at all.

NAT will render authentication impossible
as it will change part of the header.

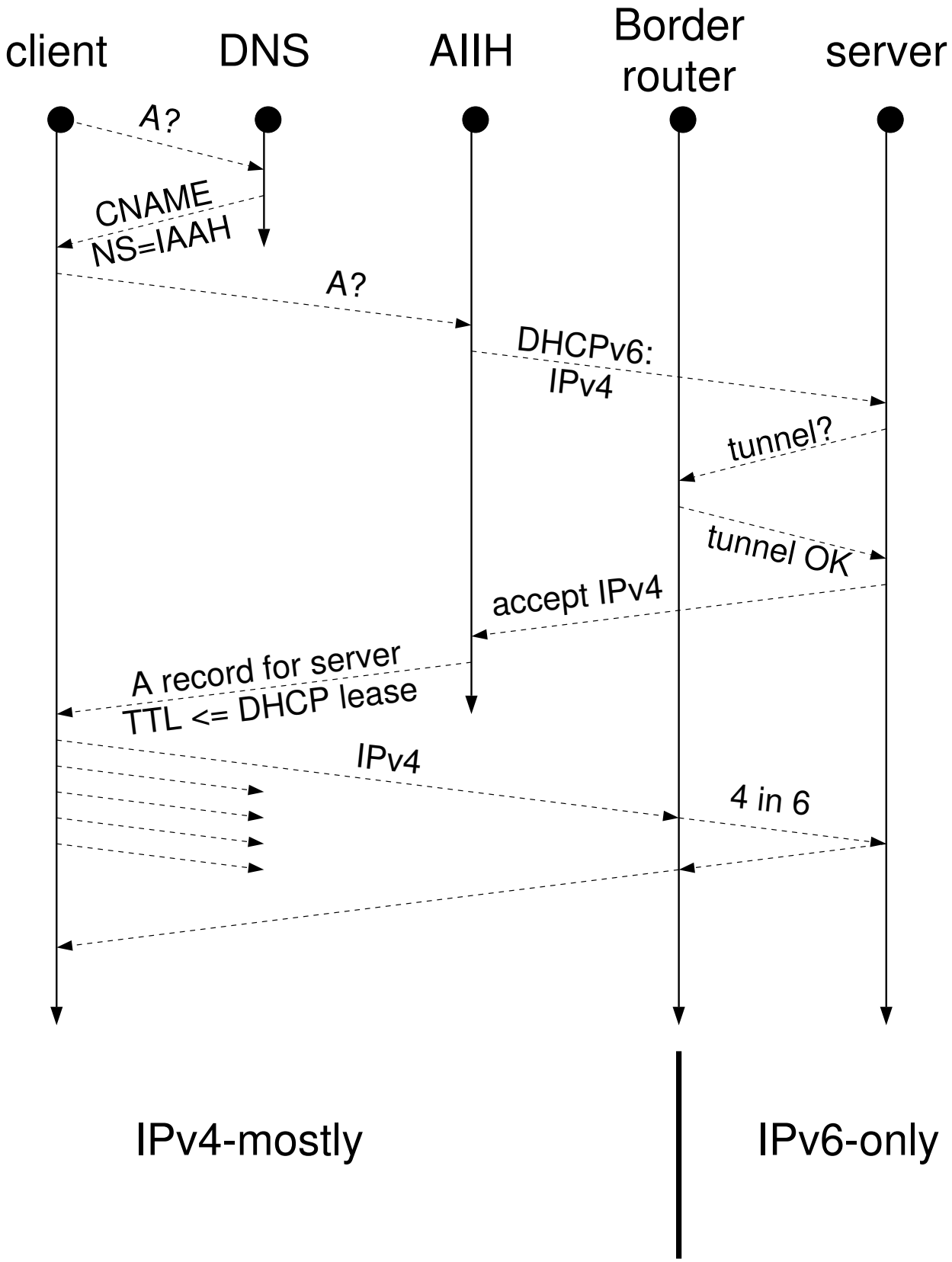
We really don't need NAT for IPv6-IPv6,
every site can get AGUAs for 64k BIG links.

So how do we deal with the rest?

For clients this is relatively easy:
when a process wants to talk to an IPv4-service
the host can acquire an IPv4-address ''just in time''.
(using DHCP or the like).

The other way around is far more difficult.
In NAT this is usually solved with port redirection
involving (tedious) hand configuration.

AIIH


client
DNS
AIIH
Border router
server
A?
CNAME
NS=IAAH
A?
DHCPv6:
IPv4
tunnel?
tunnel OK
accept IPv4
A record for server
TTL <= DHCP lease
IPv4
4 in 6
IPv4-mostly
IPv6-only

DNS

New records have been added for IPv6:

```
host AAAA 12:0034:56:78:9a:bc:de:f0
```

reverse lookup by the nibble:

```
0.f.0.0.e.d.0.0.c.b.0.0.a.9.0.0.\
    8.7.0.0.6.5.0.0.4.3.0.0.2.1.0.0\
        .ip6.int. PTR host.dom.
```

New developments in DNS:

```
host  A6  ::token       64 link1
link1 A6  0:0:0:0001:: 48 isp
isp   CNAME   NET.ISP.NET.

NET.ISP.NET. A6 3225:0400:8765:: 0
```

will yield:

```
3225:0400:8765:0001:token
```

moving a host to another link is easy,
so is changing providers and more!

Also on the roll: reverse lookup by the bit.