IPv6 Tutorial

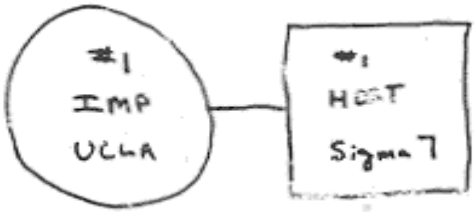
kurtis@netnod.se

About me

- Operator background
  - Started two ISPs, worked for EUnet/KPNQwest 1997-2002 as among other things responsible for network architecture
  - Currently CEO of Netnod/Autonomica
- Member of the IAB since 3 years
- Chair of Euro-IX and SOF
- WG chair in IETF and RIPE

Background

The Internet emerges

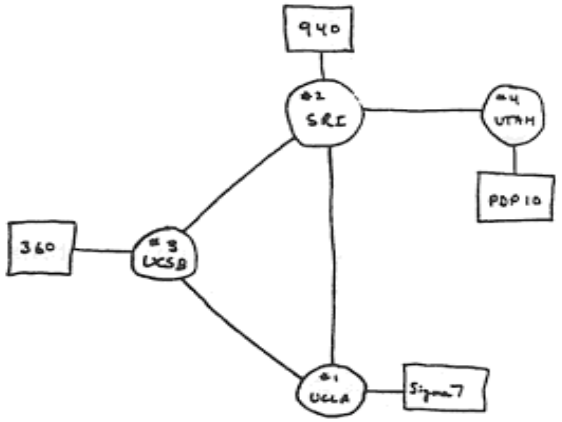


THE ARPA NETWORK

SEPT 1969

1 NODE

FIGURE 6.1 Drawing of September 1969
(Courtesy of Alex McKenzie)



THE ARPA NETWORK

DEC 1969

4 NODES

FIGURE 6.2 Drawing of 4 Node Network
(Courtesy of Alex McKenzie)

The background of IPv6

- Original IPv4 address plan was non-CIDR
  - I.e the addresses was split in A (/8), B (/16), C (/24) - which was supposed to correspond to the different needs of different users
  - The address blocks was originally handed out more or less arbitrary
- During the beginning of the 1990:s this lead to concerns that we where running out of IPv4 addresses

IPv4 Prognosis

- How much IPv4 space do we have?
- When will we run out?
- Based on http://www.potaroo.net/tools/ipv4/

The date predicted by this model where the IPv4 unallocated address pool will be exhausted is **09-Jul-2012**. A related prediction is the exhaustion of the IANA IPv4 unallocated address pool, which this model predicts will occur on **31-Mar-2011**.

Projections

- Projections are really hard for a number of factors
- This is just a summary of the issues involved
- You really should study Geoff's data first hand to make up your mind

The background of IPV6

- A first step was the introduction of CIDR
  - I.e a assigned address block could have any length
  - Described in 1993 in RFC1517 and RFC1519
  - Also required a interdomain routing protocol that could handle it - BGP v4
    - RFC1654 in July 1994
    - Implemented during 1995
- In parallel in 1992, discussions about a new addressing model and address plan started
  - The first suggestion was based on the ISO OSI model
  - Hard resistance lead to the suggestion being withdrawn

The background of IPv6

- IETF in July 1992 decided to make an estimate of the Internets future size and addressing needs
  - 2020 : 10 Billion people
  - 100 computers per person
- To leave room for errors and sparse allocations, it was decided that the needs would be
  - 10^15 computers
  - 10^12 networks

The background of IPv6

A number of proposals was developed

- TUBA
  - Would eventually be known as IPv9
- IAE
- Nimrod
- EIP
- PIP
  - Would eventually be known as IPv8
- SIP
- TP/IX
  - Would eventually be known as IPv7

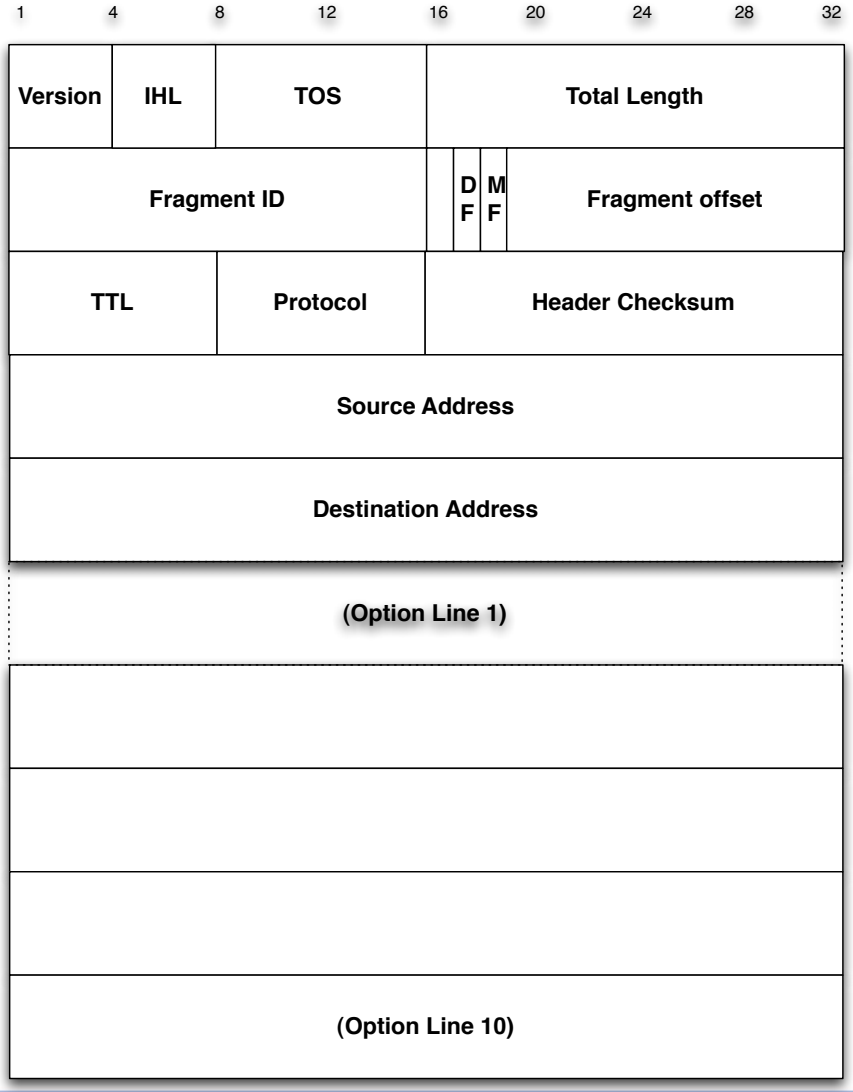
The background of IPv6

- The various proposals where developed and evolved through the merger of ideas
- Eventually three would remain
  - OSI
  - IPv7: TP/IX by Robert Ullman
    - Also waned to modify TCP
    - Was compatible with IPv4, CLNP and IPX
  - IP in IP: A mix of proposals, among others Steve Deering's SIP, PiP och IPAE
- The IPng working group in June 1994 decided to use IP in IP as base
  - However the address-size was changed to 128-bits
  - The new proposal was called IPv6

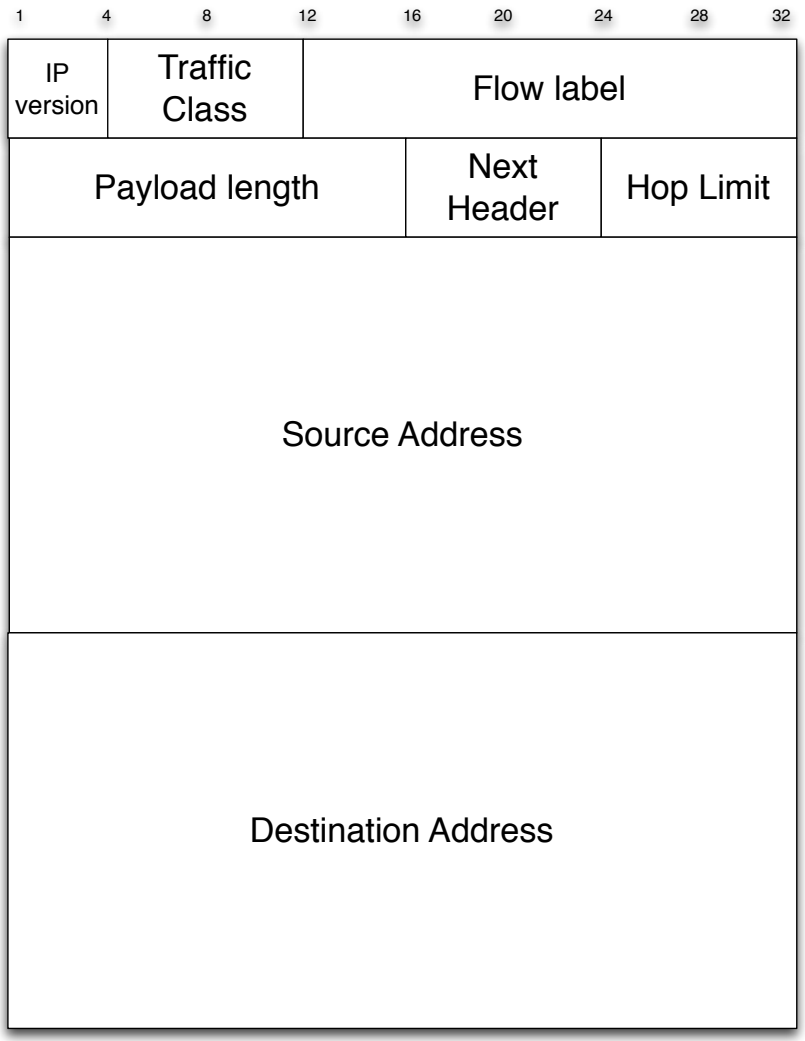
© 2006 - Netnod AB

IPv6 design and addressing

IPv4 header


1
4
8
12
16
20
24
28
32
Version
IHL
TOS
Total Length
Fragment ID
D F
M F
Fragment offset
TTL
Protocol
Header Checksum
Source Address
Destination Address
(Option Line 1)
(Option Line 10)

IPv6 header


1
4
8
12
16
20
24
28
32
IP version
Traffic Class
Flow label
Payload length
Next Header
Hop Limit
Source Address
Destination Address

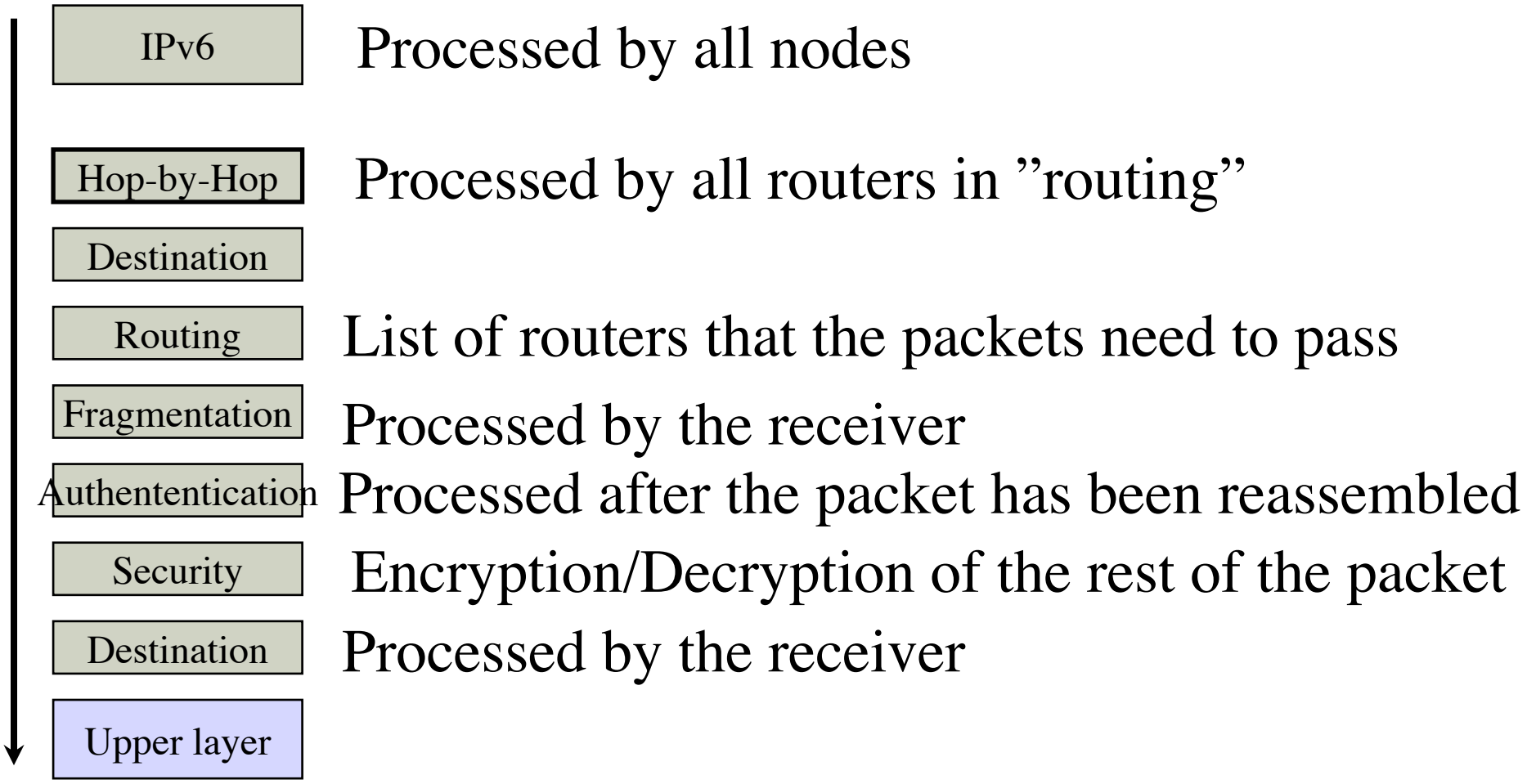
The IPv6 packet format - main differences

- All fields are of a predefined and static size
  - Faster processing
  - No "Initial Header Length" (IHL) needed
  - Instead a number of "extension headers" are defined
- Header checksum is removed
  - Speeds up processing of packets in intermediary hops, as no new checksum calculation is needed
  - Packets can in theory be misrouted due to bit errors, but the risk is extremely low
  - The payload often has it own checksums

The IPv6 packet format - main differences

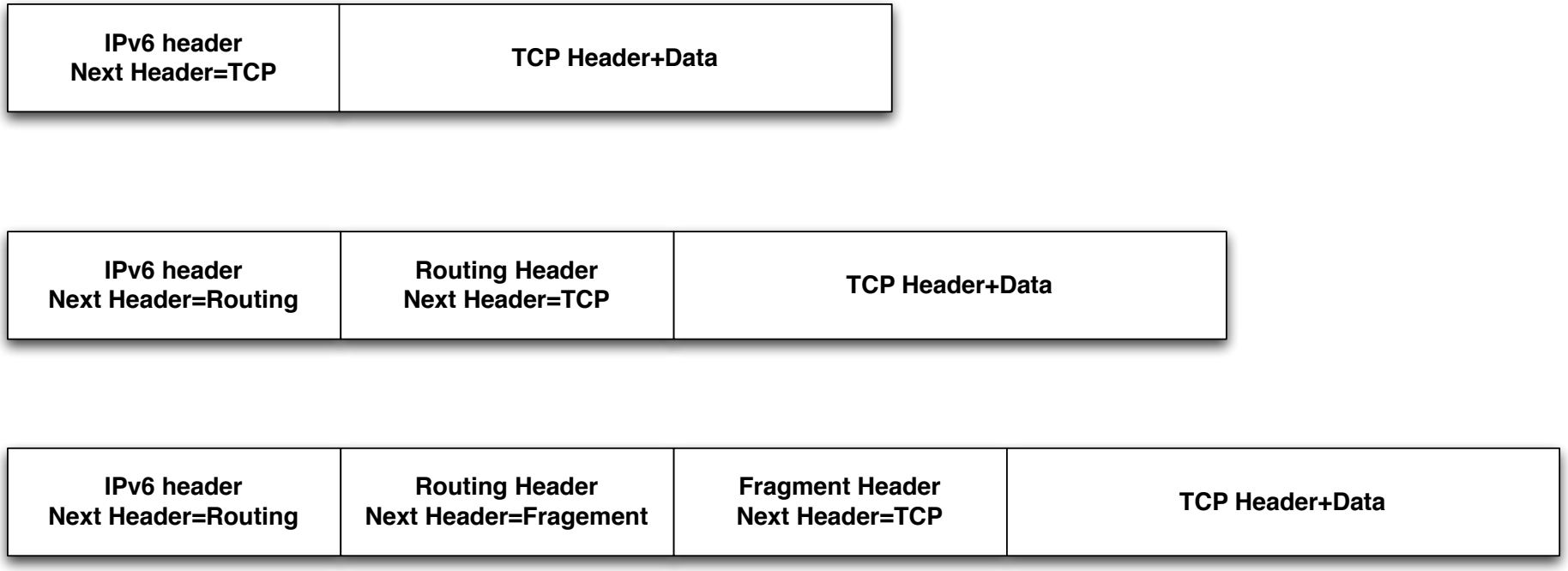
- Fragmentation is gone
  - IPv6 will only send packets after first having performed Path Maximum Transmission Unit Discovery, PMTUD
- IPv6 does not have a TOS field
- The fields have changed names
  - Packet length -> Payload length
  - Protocol type -> Next header
  - Time to live -> Hop Limit

IPv6 extension headers



IPv6 extension headers

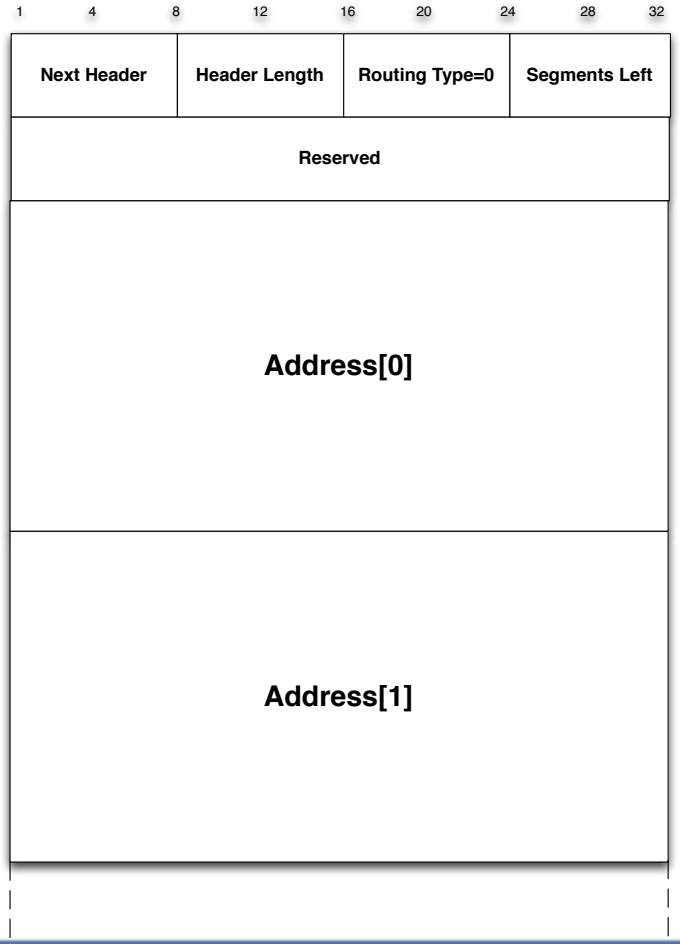
- Extension headers are a list of headers



IPv6 extension headers

- The "Next header" coding is the same as for "payload type", and the IPv4 codes are used in IPv6 as well
- Most are the same but with minor differences :
  - 0: Reserved (IPv4)/ Hop-by-Hop options (IPv6)
  - 1: ICMP (IPv4)
  - 2: IGMP (IPv4)
  - 3: ICMP (IPv6)
  - 59: No next header (IPv6)

Routing type 0 extension header



Routing type 0 extension header

- Was original intended as equivalent to source routing in IPv4
  - Although drawbacks where know the goal was to keep as much of the functionality as possible from IPV4 in the case it was being used
- However it was (re-)discovered for IPv6 that source routing is a really bad idea
- And it got depracted

IPv6 extension headers - Fragmentation

- Contrary to IPv4, no fragmentation by intermediate systems is done
  - E.g. If a packet is to large for a given link the intermediate node in IPv4 will fragment the packet
- In IPv6 PMTUD must be done before sending a packet to a destination
- An applications can however request to send a packet that is larger than the discovered
  - Fragmentation will then occur at the source host

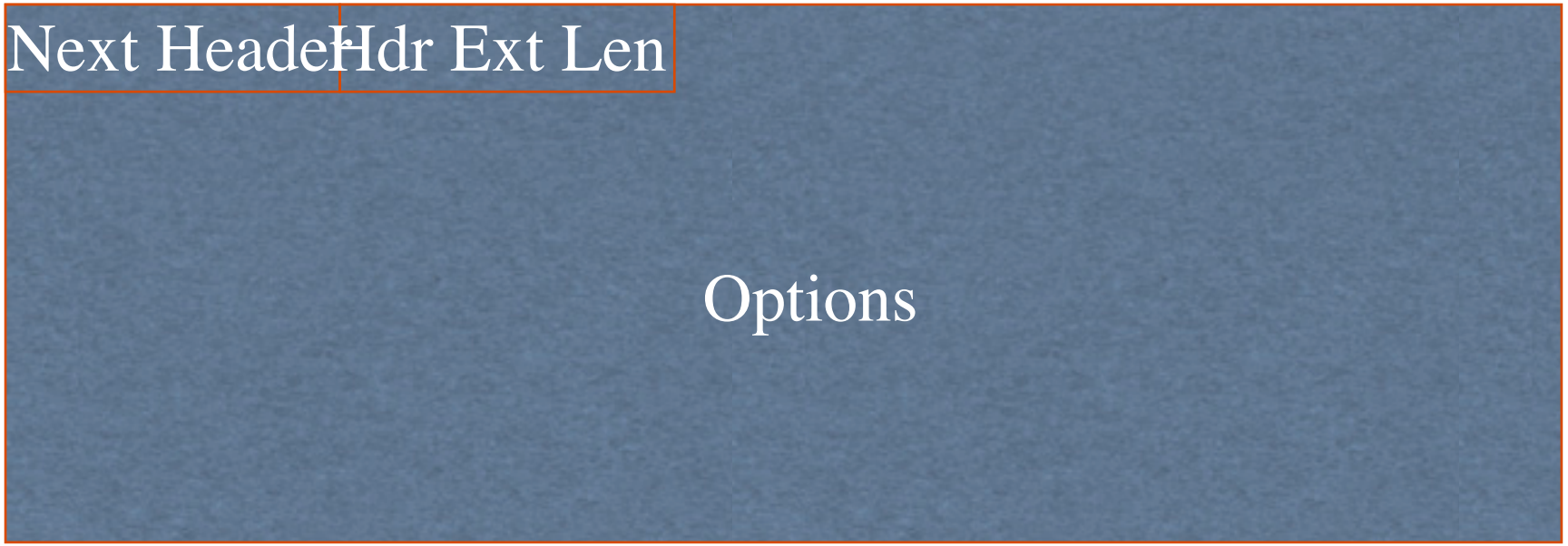
IPv6 extension headers - Fragmentation

Fragementation header

IPv6 extension headers - Destination option

- If we want to add functionality to IPv6 in the future, we could add new extension headers
  - This would however use up more of the 255 available "Next header" codes
  - It would require that both sender and receiver understands the code
  - Requires that intermediate node such as firewalls understands the code
- Instead we can use the "Destination option" extenstion header
  - "Undefined" header to be used in the future

IPv6 extension headers - Destination option



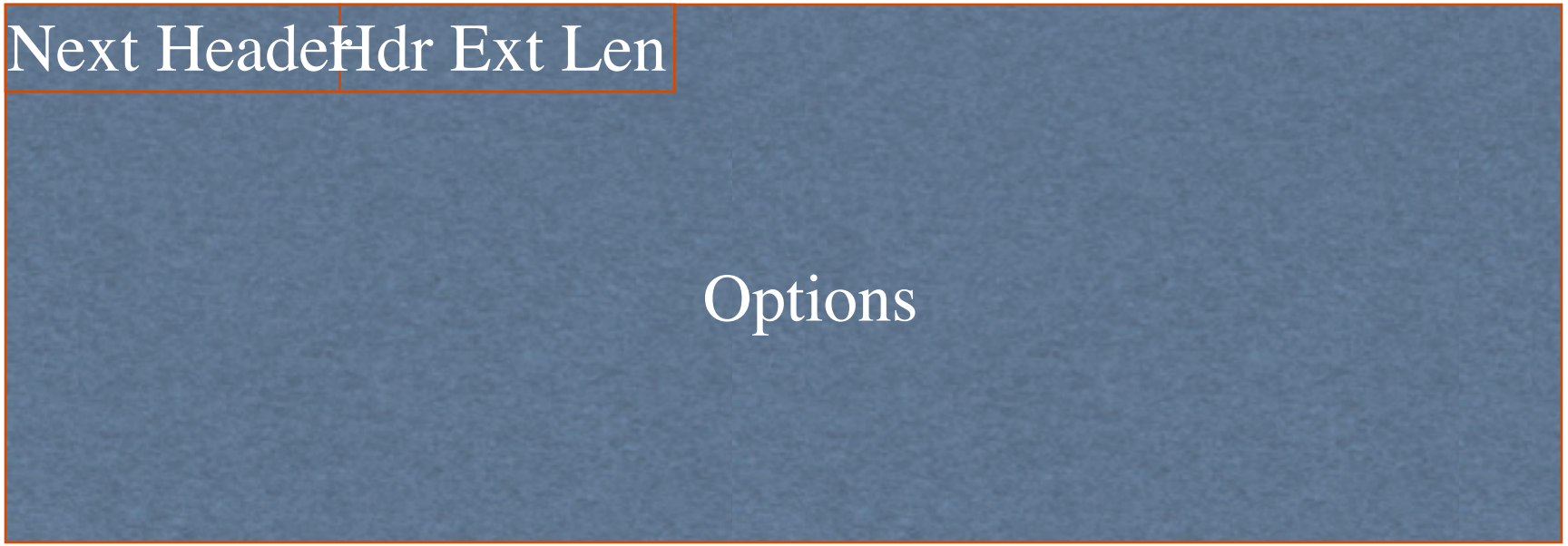
IPv6 extension headers - Destination option

- The receiver will decode the header
- If the receiver does not understand the header, or the the data (or it is not what it expected), the receiver will send a ICMP ”Unrecognized type” packet back to the sender
- Today only two “byte-padding” types are defined

IPv6 extension headers - Hop-by-Hop

- All other extension headers will be processed by the final destination
- The hop-by-hop header will be processed by intermediary nodes
- Hop-by-hop have exactly the same coding as the “Destination option” header

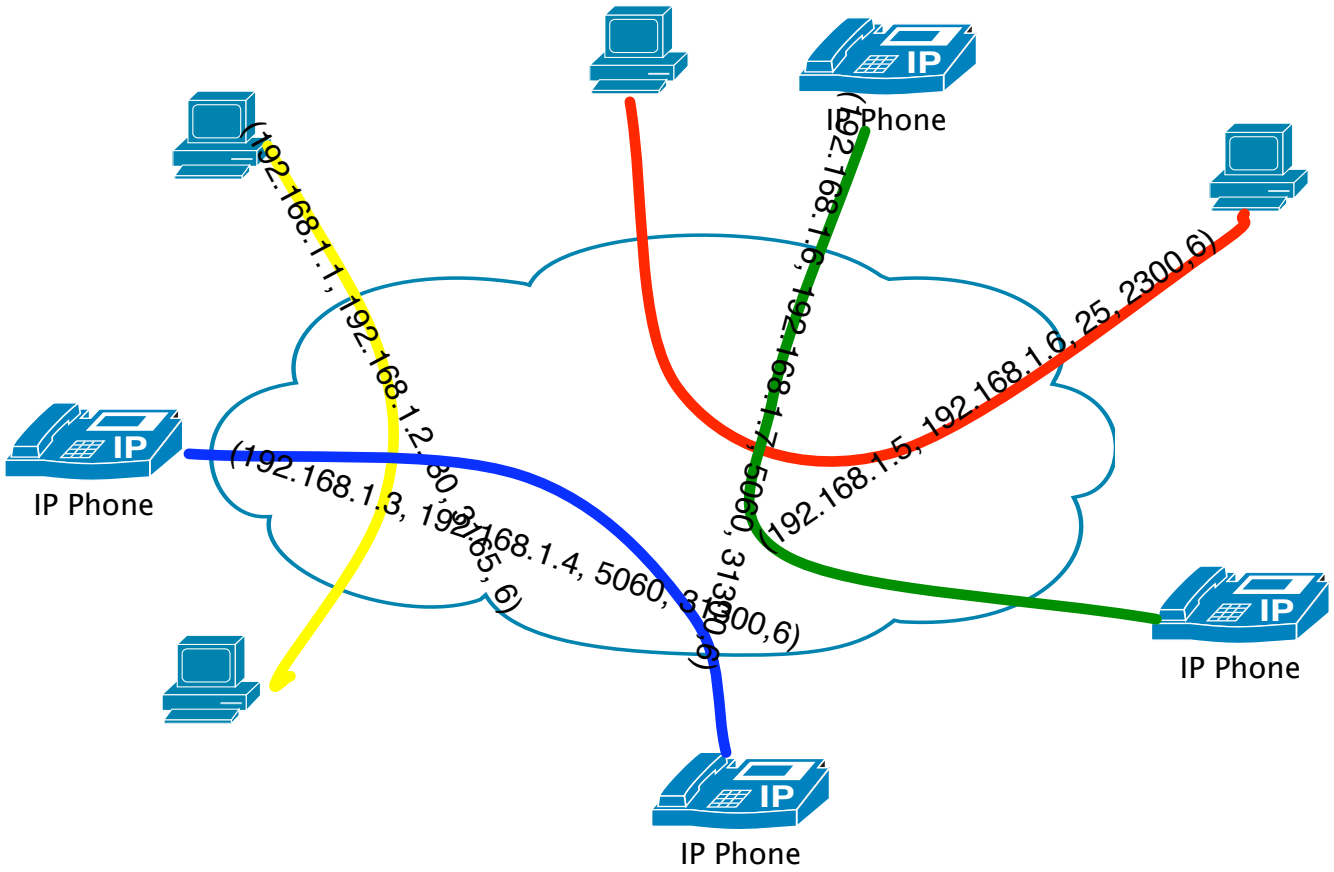
IPv6 extension headers - Hop-by-Hop



IPv6 and Flows

- A flow is a sequence of packets, sent from a single destination to a unicast, multicast or anycast destination that the source decides to label as a flow
- Traditionally classified as
  - (src, dst, src port, dst port, protocol type)
- Does not really work in IPv6
  - Some of those can be encrypted, fragmented or in anohter extension header
  - For processing of the traditional definition the implementations are also dependent on understanding the transport protocol in use
- The IPv6 flow label field specified in RFC3697

IPv6 Flows


IP Phone
IP Phone
IP Phone
IP Phone
IP Phone
(192.168.1.1, 192.168.1.2, 80, 2165, 6)
(192.168.1.3, 192.168.1.4, 5060, 3100, 6)
192.168.1.5, 192.168.1.6, 25, 2300,6)

IPv6 and Flows

- IPv6 instead uses
  - (src, dst, flow label)
- The flow label is a 20-bit field in the IPv6 header
  - A zero value indicates that a packet is not part of any particular flow
  - Nodes must assume that there is no semantic meaning of the flow lable
- Receiving nodes must not assume that a packet arriving 120s or more after last one with the same flow label is part of the same flow
  - Unless state is manged/signaled in some other way
- Flow labelling makes no assumption on packet re-ordering

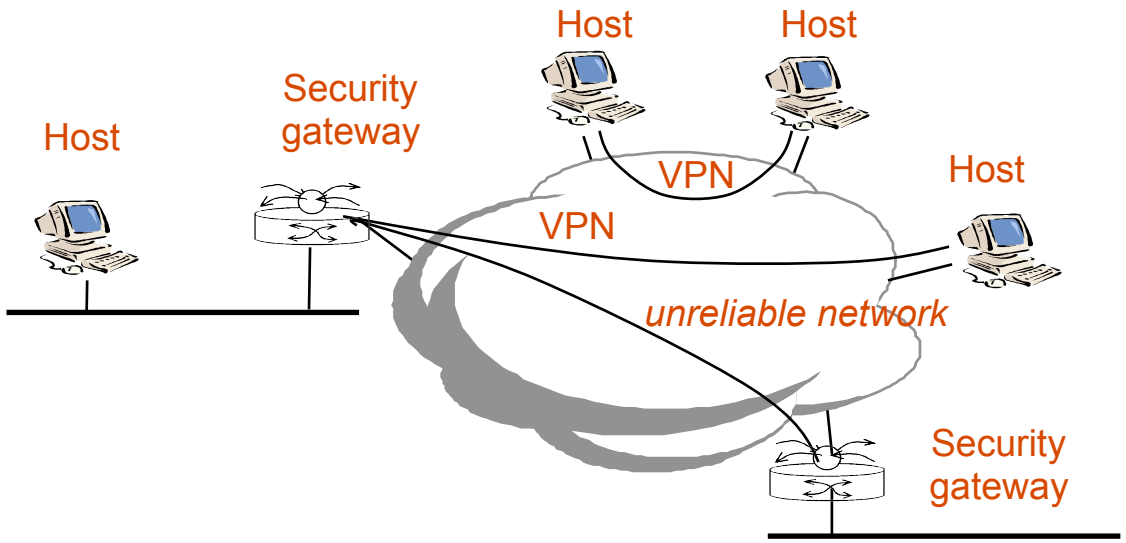
IPv6 and flows

- Source nodes must make sure that there is no unintentional reuse of flow labels
  - Should assign flow labels in sequence
- Theft of Service / Denial of Service
  - An attacker could possibly make use of resources that are reserved for a specific flow, by modifying flow labels
  - This could be taken to a denial of service attack
  - As the flow label is part of the IPv6 header, it is not protected or authenticated, which means it is not more secure than a src/dst address
    - Unless IPsec tunnel mode is used
    - Ingress filtering might give some security

IPv6 security

- In IPv4 secure (encrypted) tunnels are built using IP-Sec
- IPv6 have similar functions built in
  - Authentication header is used to verify the sender
  - The Security header to protect the content from a third party
- **But the security is the same for IPv6 and IPv4!!!!**

IPv6 security

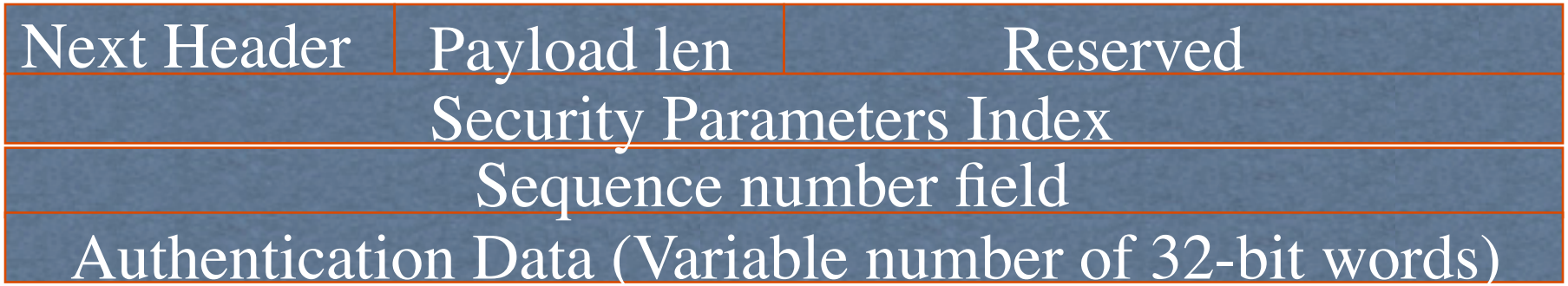


IPv6 Security - Authentication

- An extension header as the others
- Will be added after the other headers, but before the payload

- The authentication header is defined in a separate RFC, 2402
  - Used to be part of the IPv6 specification

IPv6 Security - Authentication



IPv6 Security - Authentication

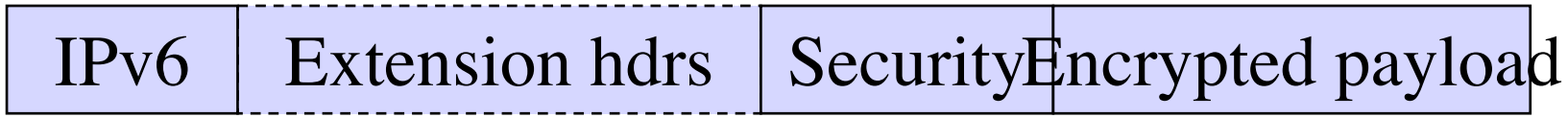
- Next Header
  - Next header coding
- Payload length
  - Length of the header and data
- Reserved
  - Reserved for future use. Must be set to zero
- Security Parameter Index (SPI)
  - Arbitrary 32-bit value that together with the destination IP and the security protocol uniquely identifies the security association

IPv6 Security - Authentication

- Sequence number
  - Unsigned 32-bit that is a monotonically increasing counter
  - Used to prevent replay attacks
  - Initialized to zero when the security association is established
  - Must never be allowed to cycle
- Authentication data
  - Variable length field
  - Contains the Integrity Check Value (ICV) for the packet
  - What authentication algorithm used for the ICV is determined by the SA (DES, MD5, SHA-1, etc)
  - ICV is calculated over
    - IP header fields that does not change in transit or the value at the end point can be predicted
    - The AH
    - Upper layer data
- What packets get an AH is determined by the SA (IPsec part of the stack)

IPv6 security - Encryption

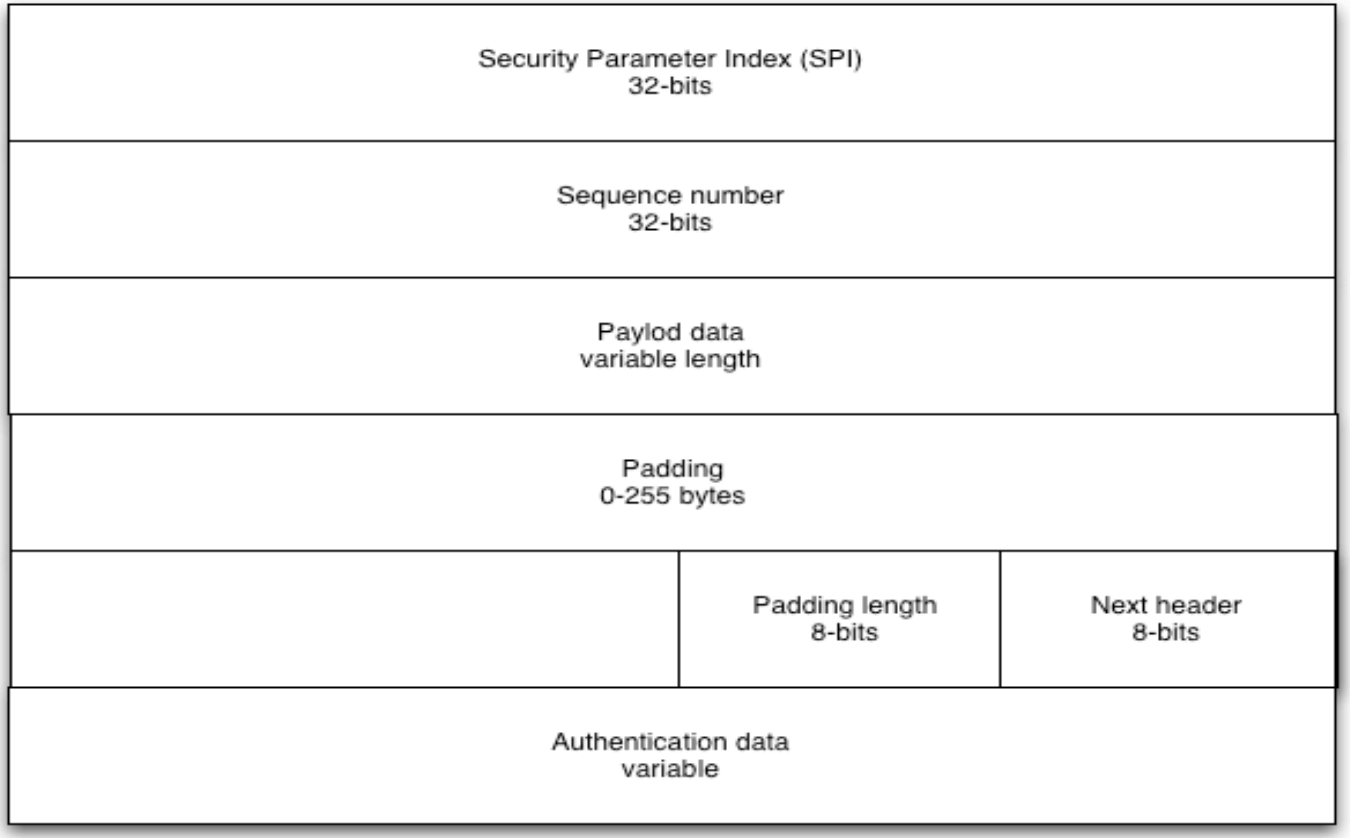
- Yet another extension header…
- …the last one to be added, and that can be read in clear text



- How encryption is done is dependent on the crypto algorithm in use
  - For more information see RFC2406

© 2006 - Netnod AB

IPv6 security - Encryption



IPv6 security - Encryption

- Security Parameter Index (SPI)
  - Arbitrary 32-bit value that together with the destination IP and the security protocol uniquely identifies the security association
- Sequence number
  - Unsigned 32-bit that is a monotonically increasing counter
  - Used to prevent replay attacks
  - Initialized to zero when the security association is established
  - Must never be allowed to cycle
- Payload data
  - The data to be encrypted
  - Which encryption algorithm that is used is part of the SA
  - If the encryption algorithm needs a Initialization Vector that is stored in the payload data field

IPv6 security - Encryption

- Padding
  - Some block chiper algorithms needs that data to be of the block size (or multiples thereof)
  - The padding length and next header fields needs to be right aligned
  - Can also be used to conceal the actual length of the message for traffic flow confidentiality
- Padding length
  - The length of the padding field
- Next header
  - Indicates the type data in the payload field
  - Either a next-header or a upper layer protocol
- Authentication data
  - Optional ICV calculated over the ESP packet
  - Same as for AH
  - Only calculated if required by the SA

IPv6 Quality of Service

- In IPv4 we have DiffServ and IntServ that make use of the TOS field
- In IPv6 there is "traffic class" that is used for the same functionality
- Standardization is on-going

IPv6 Design and addressing

- IPv6, 128-bits -> A lot of text…
- Three ways to write IPv6 addresses
- Most common is

X:X:X:X:X:X:X:X

Where X is the Hex representation of a 16-bit octet

- Example

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

1080:0:0:0:8:800:200C:417A

Note that you do not have to write the leading zeros in each group

IPv6 Design and addressing

- Due to the nature of IPv6 addressing, most addresses will have groups of zeros as well as lot's of leading zeros. To simplify could can compress a group of zeros with "::"
  - Example

    1080:0:0:0:8:800:200C:417A

    Can be written as

    1080::8:800:200C:417A

    Note that you can only write one pair of ::

    0:0:0:0:0:0:0:0

    Written as

    ::

IPv6 Design and addressing

- Address prefixes and subnets works and are written exactly as in IPv4

```
12AB:0000:0000:CD30:0000:0000:0000:0000/60

12AB::CD30:0:0:0:0/60

12AB:0:0:CD30::/60
```

  - All are valid

IPv6 Address types

- The IPv6 address type is defined by the prefix bits as

Allocation	Prefix (binary)	Fraction of Address Space	
Unassigned (see Note 1 below)	0000 0000	1/256	
Unassigned	0000 0001	1/256	
Reserved for NSAP Allocation	0000 001	1/128	[RFC1888]
Unassigned	0000 01	1/64	
Unassigned	0000 1	1/32	
Unassigned	0001	1/16	
Global Unicast	001	1/8	[RFC2374]
Unassigned	010	1/8	
Unassigned	011	1/8	
Unassigned	100	1/8	
Unassigned	101	1/8	
Unassigned	110	1/8	
Unassigned	1110	1/16	
Unassigned	1111 0	1/32	
Unassigned	1111 10	1/64	
Unassigned	1111 110	1/128	
Unassigned	1111 1110 0	1/512	
Link-Local Unicast Addresses	1111 1110 10	1/1024	
IANA - Reserved (Formerly Site-Local)	1111 1110 11	1/1024	[RFC3879]
Multicast Addresses	1111 1111	1/256	

IPv6 addresses

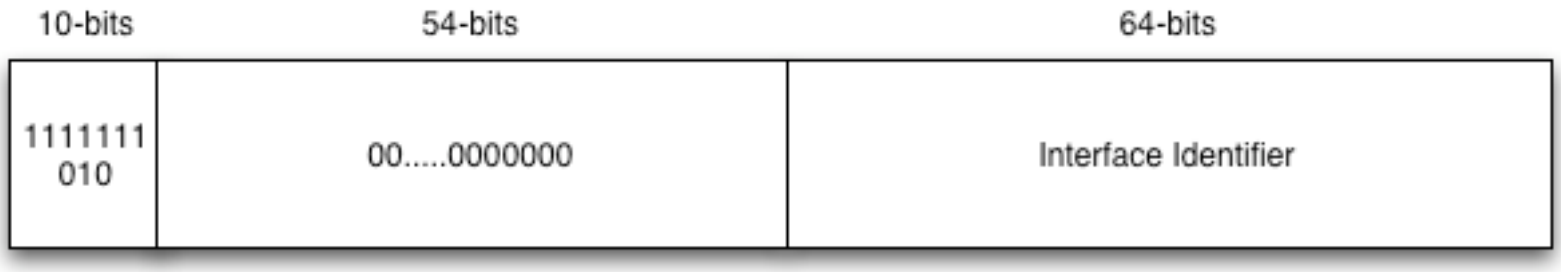
- Only IPv6 addresses of the "unspecified" type, "loopback address" and IPv6 addresses with embedded IPv4 addresses have the prefix 000
- Apart from above, IANA should only do delegation from the "001" prefix
- Leaves 85% of the IPv6 addresses for future use

Scoped addresses

- When developing IPv6, the idea that addresses should have a certain "validity", or "scope" was launched
  - That is, an area of validity/scope where the addresses are guaranteed to be unique
- Similar mechanisms are used in multicast
  - Scoped streams
- In IPv4 there is/was similar mechanisms as well
  - RFC1918
  - Link local addresses (169.254.0.0/16)

IPv6 scope

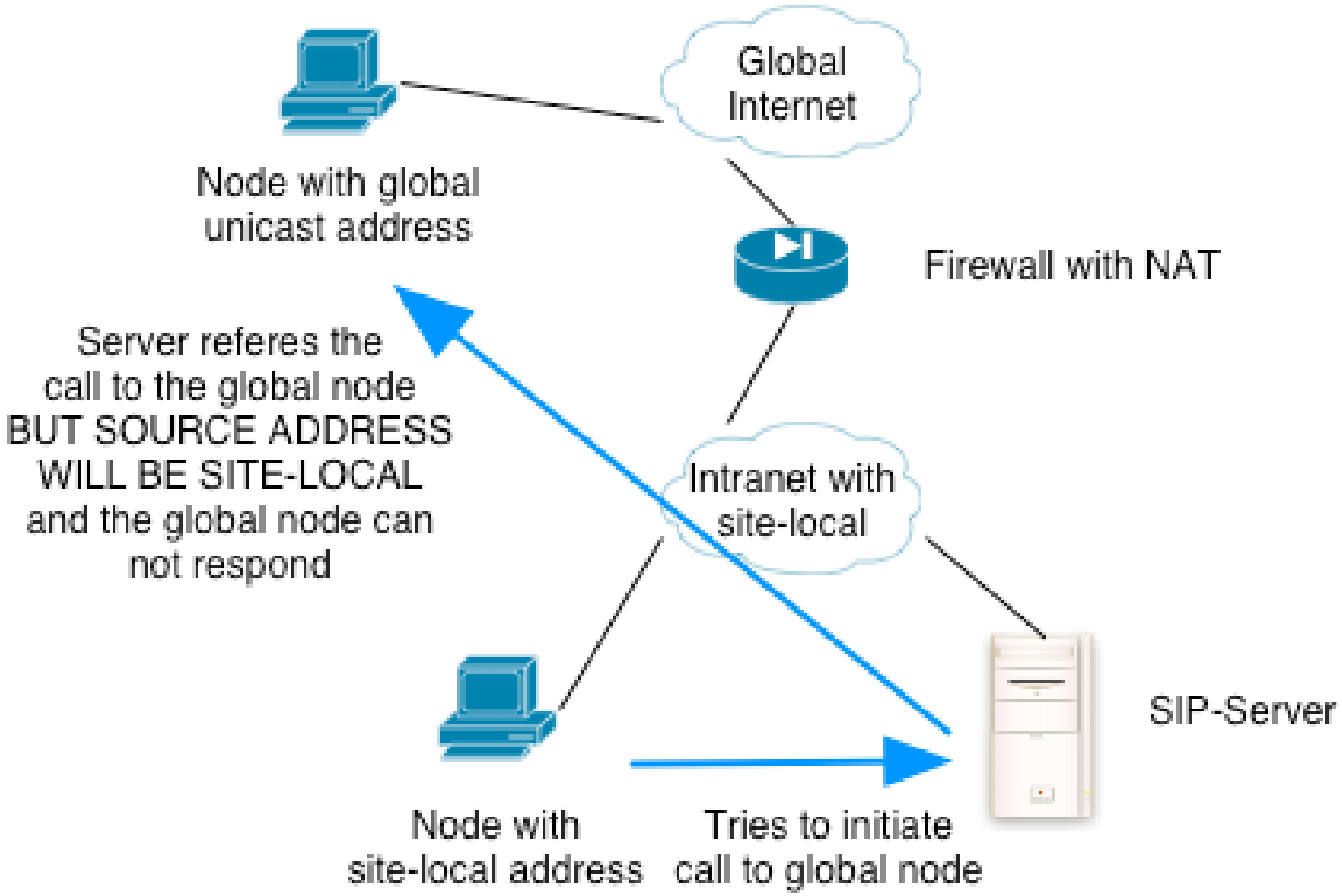
- For IPv6, three scoped where defined
- Global
  - As with all unicast addresses, these are globally unique addresses, assigned from a registry (e.g LIR)
- Link-local
  - Addresses that are unique on the link in question. These addresses are calculated and assigned by the interface itself
  - This means that a node can have an IPv6 address and communicate with others on the local network without the presence of a router or server
  - Routers are not allowed to forward packets with a link local address as source or destination address - to destinations off the link



Site-local

- Originally there was also something called “site-local”
- The idea was that these where addresses that where unique within a site or network
  - Compare with RFC1918
  - However a site or network i never specified
- But the experiences with NAT/RFC1918 deterred quite a few people
  - An the source address selection problem was hard enough, without site-local

Site-local


Global Internet
Node with global unicast address
Firewall with NAT
Server referes the call to the global node BUT SOURCE ADDRESS WILL BE SITE-LOCAL and the global node can not respond
Intranet with site-local
SIP-Server
Node with site-local address
Tries to initiate call to global node

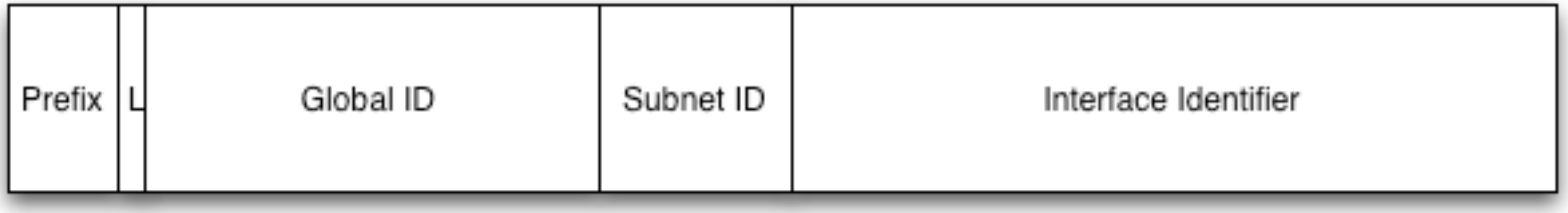
Site-local

- The debate that followed was among the hardest and longest in many years…
- It was finally decided that site-locals should be deprecated
- A number of scenarios that actually would have benefited from site-local was however still unsolved
  - "The research ship" - A network that is not connected to any other network except when the ship is in port. The ship must till be able to communicated within, independent on an external connection
  - Airplanes
- The solution that was decided on is called "Unique Local IPv6 Unicast addresses"

ULAs

- There where two versions proposed
  - Locally assigned
  - Centrally assigned
- Of these only locally assigned are standardised
  - RFC4193
- These addresses must never be routed globally
  - They will be assigned from a globally unique prefix
  - This will make it easier to filter them out
- The idea is that these addresses should be used by networks that will never be connected to the global Internet, but that still might want to interconnect between them
- If they are leaked outside their domain in either the DNS or in the global routing system, they are still unique "enough" not to cause harm
- Applications can threat them as globally unique addresses

ULAs



- Prefix
  - The globally unique prefix, proposed to be FC00::/7
- L
  - Locally/Centrally assigned prefix
- Global ID
  - The globally almost unique prefix that was generated
- Subnet ID
  - Subnet ULA block
- Interface identifier
  - EUI64 or RFC3401

ULAs

- The algorithm for generating the globally "unique" prefix is
  - Actual time through NTP
  - The local EUI64 address, alternatively anther local unique ID
  - Add the EUI64 address with the actual time to create a hash key
  - Calculate a SHA-1 has of the key
  - Use the last 40-bits of the hash as a global ID
  - Add FC00::/7 and the bit indicating if this is a locally or globally unique prefix
- The prefix will be globally "unique" if it is registered in a database
  - Fairly controversial at the moment

IPv6 Addressing - 6bone

- In order to "practice" transition an early IPv6 network was created , the 6bone
- 6bone addresses where handed out to anyone who asked
  - 3FFE::/16 was set aside for a "6bone registry"
- The 6bone was built as an overlay network of tunnels between routers and hosts
- Unfortunately there where several drawbacks with the 6bone
  - No-one applied for IPv6 addresses from the RIR/LIRs as these where harder to get
  - The tunnels often led to fairly sub-optimal routing
  - In principle the 6bone led to a slower IPv6 adoption
  - It was also in the risk of creating a new "swamp space"

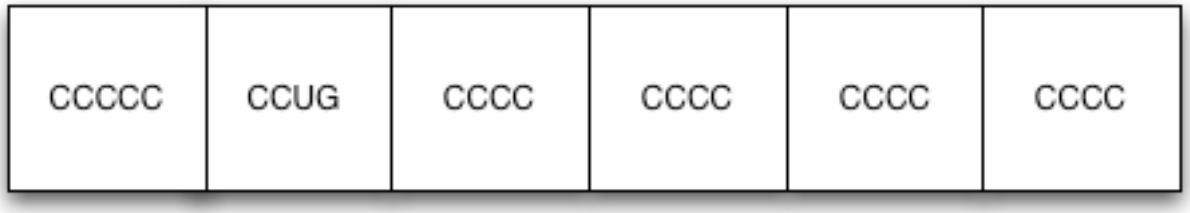
IPv6 Addressing - 6bone

- To get away from a tunnelled 6bone network, it was decided that
  - Since January 1th 2004, no new pTLAs were assigned
  - As of June 6 2006 the 6bone pTLAs are no longer valid and should not be routed
  - The 6bone prefix will then be "given back" to the IANA
  - The idea is that 3FFE::/16 should be reused
- One of the thoughts with migrating away from the 6bone is that more people will ask their ISPs for IPv6 addresses and connectivity, therefore creating a market

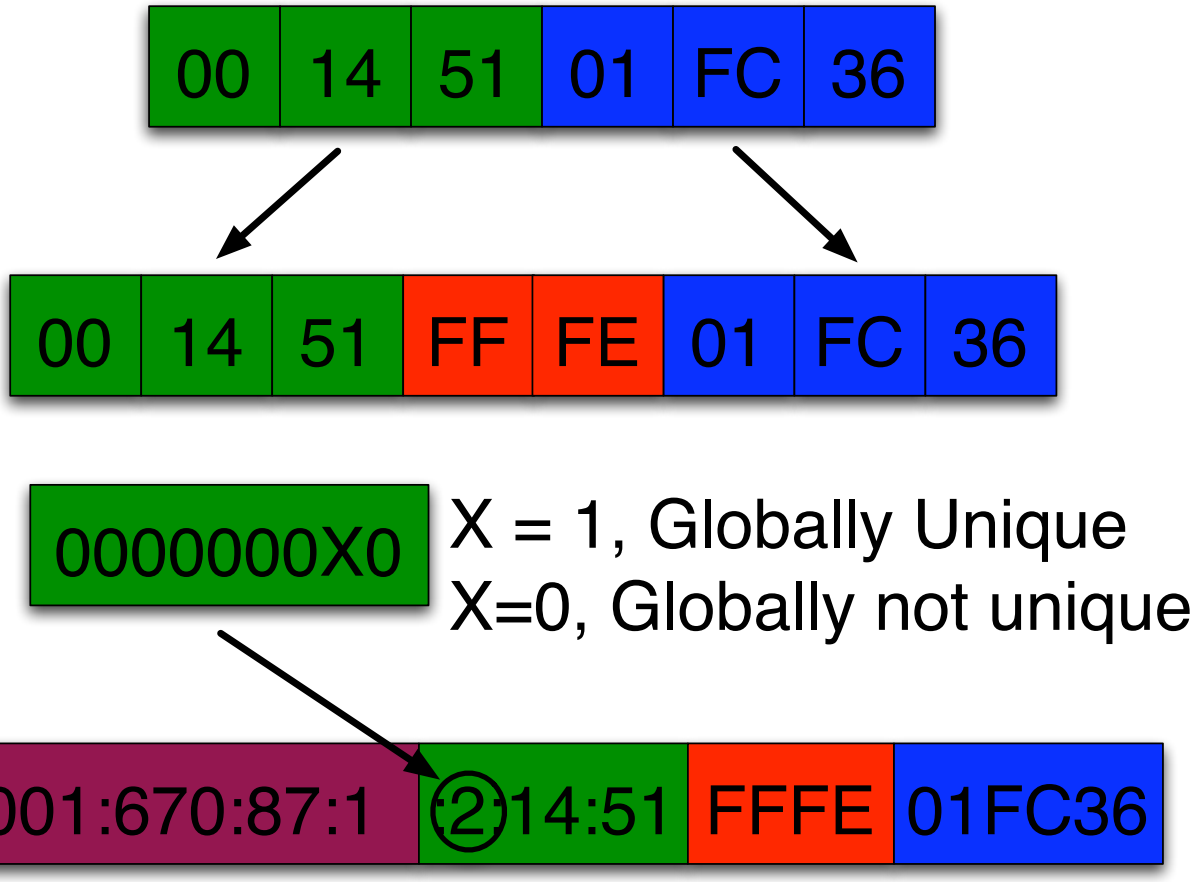
IPv6 Unicast addresses

IPv6 Unicast addresses

- For IPv6 unicast addresses, the last 64-bits should be the interface address
  - Specified in RFC3513, IPv6 Addressing Architecture
    - EUI64 coded, but as "modified EUI64"
      - Inverted 'g' bit
      - Indicates globally or locally unique EUI64 address
    - The 'U' bit will mark if this is a unique address or not
- Applies to all unicast addresses except those staring with 000



EUI64 encoding



Discussion on EUI64 coding

- A EUI64 address is unique per interface and therefore the per host
  - Some say this is a threat to integrity
  - On the other hand, a IP address in itself is fairly revealing
  - A EUI64 address can also be set to what you want

RFC4941

- More solutions to the integrity problem
  - Change MAC address periodically
    - A lot of work
  - Use DHCP
  - Generate other EUI64 addresses over time
- RFC3041 describes the latter
  - Generate a temporary interface identifier per interface
    - Can be used to generate multiple addresses
  - Changes over time
  - Describes two algorithms
    - With stable storage
    - Without stable storage

RFC4941

- Stable storage
  - Take the historical value from the last "round". If non exists, generate a (pseudo) random number
  - Calculate the MD5 value of the previous value
  - Take the 64 leftmost bits. Set bit 6 to 0
  - Use it as temporary interface ID
  - Store the rightmost 64 bits as historical value
- Without stable storage a (pseudo) random number will always be used for the first stage
- The advantage with the algorithm is that it can be hard to generate really random numbers
- A new interface ID is generated when valid lifetime for the prefix expires

IPv6 'special' addresses

- The unspecified address
  - Looks like an IPv4 deafult route, but it isn't

    0:0:0:0:0:0:0:0/128
  - Used as source address by nodes before they have their own address (For example autodiscovery)
  - Must never be forwarded
- Loopback address

  0:0:0:0:0:0:0:1/128
  - Same as in IPv4

Point to point link addressing

- RFC3513 says that all nodes should use EUI64 coded interface identifiers in the rightmost 64 bits
  - This means that each link will be a /64
- For point to point links you will most likely not want to use the EUI64 coded interface identifier for operational reasons
  - And as there is no broadcast address in IPv6, /127 seems as the natural subnet size choice for p2p links
- However, this will (at least in theory) create a number of conflicts with other IETF standards...
  - Let's look at some of them...

Point to point link addressing

- Most basically, RFC3513 says the longest subnet prefix possible is /64
  - Then again who follows standards :-)
- Second, RFC3513 defines the 70th and 71th bits as the u and g bits (universal/local)
  - If /127 is used, these bits need to be taken into account when the address is created
- Third, RFC3513 defines the subnet-router anycast address
  - In a prefix of length n bits, the 128-n last bits are all zero. And all routers on the subnet are listening on the anycast address

Point to point link addressing

- Now assume the following sequence of events (From RFC3627, /127 length considered harmful) :
  - Router A and router B are connected by a point-to-point link
  - Neither is configured
  - Router A is configured with 2001:DB8::1/127
  - Router A performs DAD for 2001:DB8::1/127, and adds the subnet-router anycast address, 2001:DB8::0/127. DAD is not performed for anycast addresses
  - Router B is now configured with 2001:DB8::0/127 as it's unicast address and performs DAD, which fails
  - Router B will not get an address
- Will be repeated at crash or reboot

Point to point link addressing

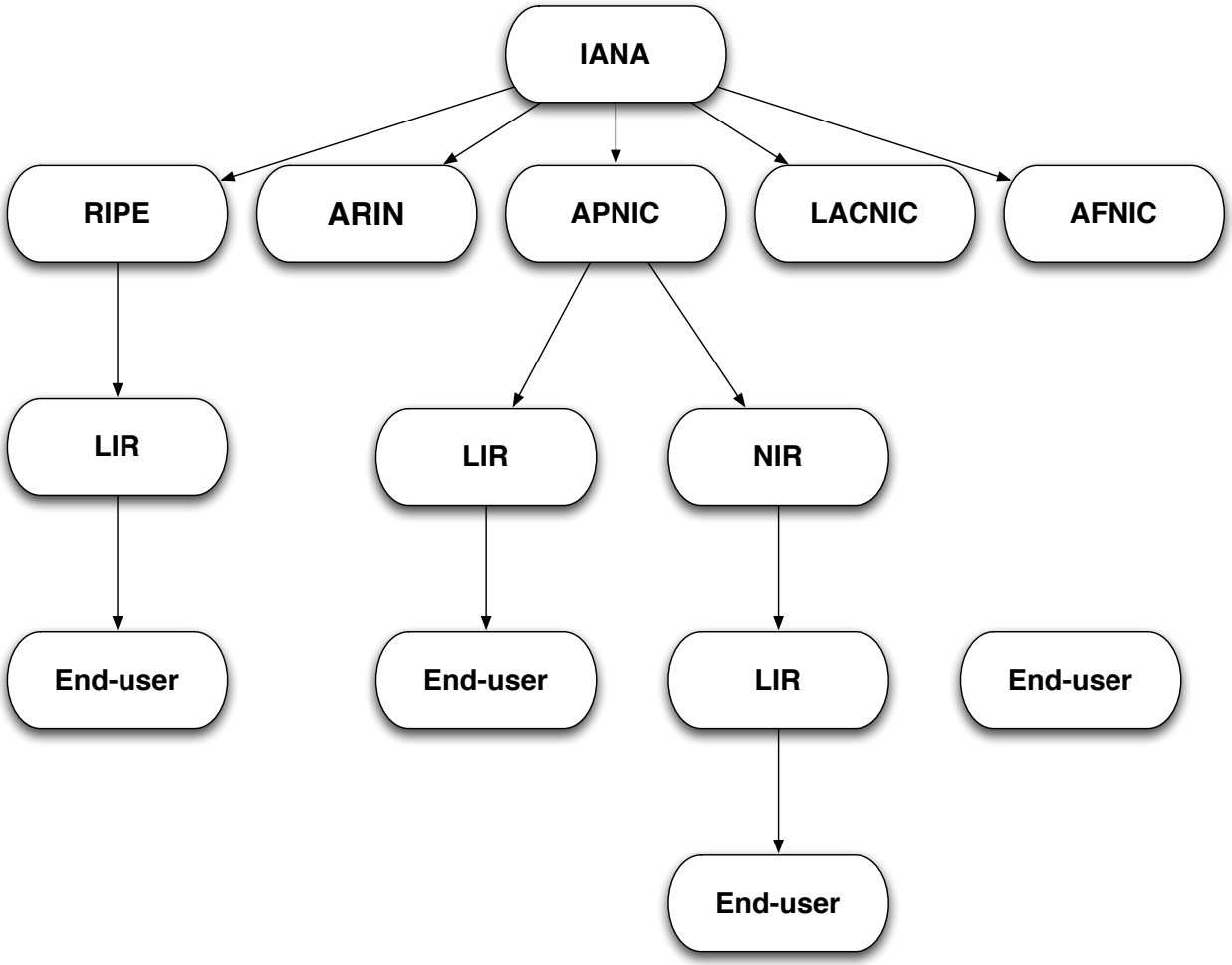
- Possible solutions
  - Use /64 for point to point links
  - Only use link-local addresses. Not operationally viable
  - Use /126. Does not have the anycast problem, but does have the u/g problem
  - Modify RFC3513. Not likely as /64 is deployed and standardised
  - RFC3627 recommends /112 to leave room for node identifiers
- There are some additional problems with Mobile IP in the use of prefixes longer than /120
  - The last 7 bits all zero have been reserved as "Mobile IPv6 Home-agents anycast address"

RFC3849 - The Documentation prefix

- To avoid the "SUN experience" the prefix 2001:DB8::/32 is reserved for use in documentation
- Should never be configured....
- Should never be announced....

Address allocation policies

IP address allocation flow



Address Allocation Policy

- According to the IPv6 architecture a “site”” will be give a /48 as prefix
  - Each site shall in turn under RFC3513 allocate a /64 per link
- RIRs allocates a /32 as the first allocation to an LIR
  - Larger allocations can be made with reference to the HD ration with their current customers

Today's Address Allocation Policy

- For the first allocation the following requirements needs to be met
  - LIR
  - Have plans for 200 allocations to other organizations within two years
- In principle only ISPs and large enterprises meet the criteria
  - For example NorduNET doesn't

Today's Address Allocation Policy

- Means there are
  - No PI address space, i.e portable address space
  - No other way to end-users/sites to get addresses except via their provider
- The need for redundancy is not less in IPv6 compared to IPv4

Transition technologies

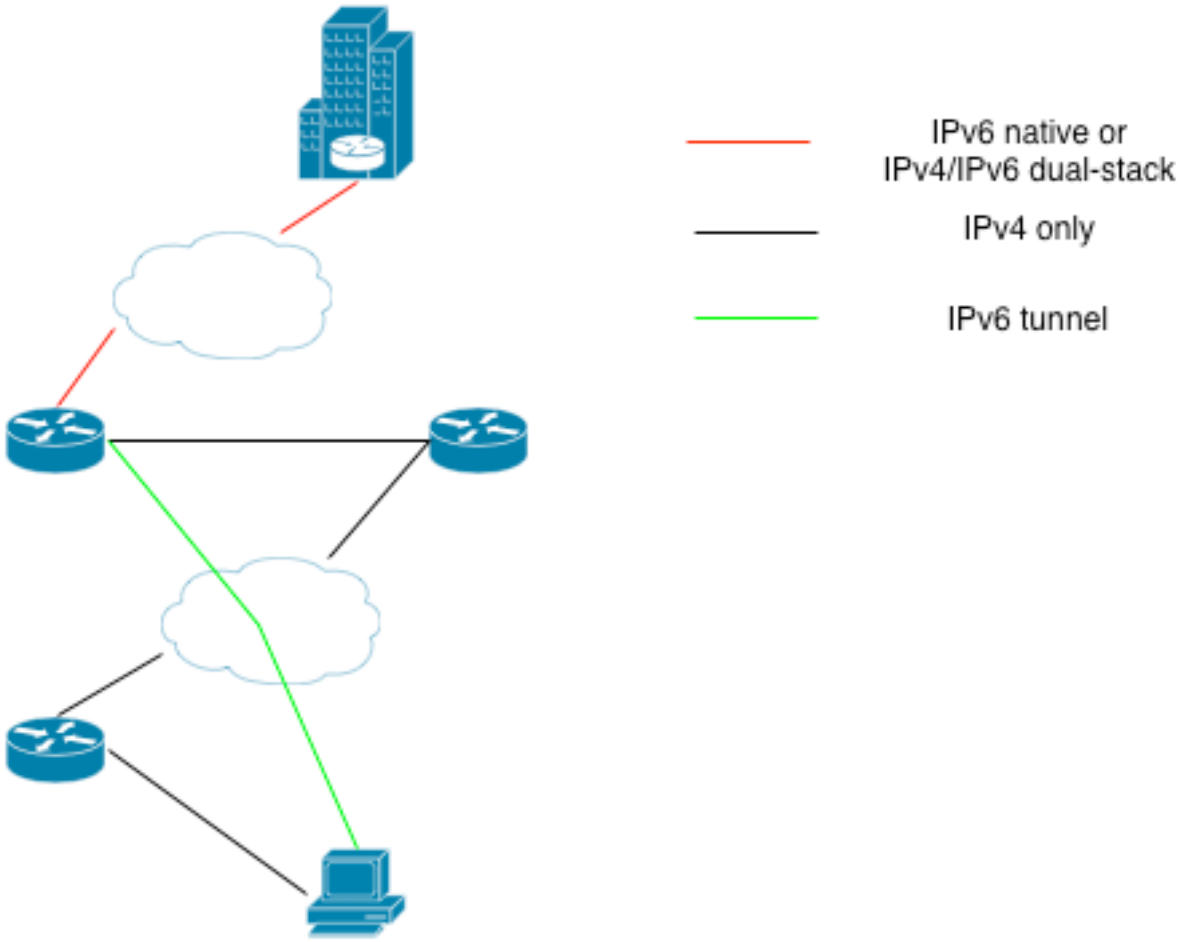
Transition technologies

- When introducing IPv6 into a network, a number of issues might occur
  - All hardware in the network might not support IPv6
  - The Internet provider might not offer/support IPv6
  - The network is behind a firewall (NAT device) that does not support IPv6
  - All applications does not support IPv6
- What problems you might encounter is very much dependent on what type of network it is

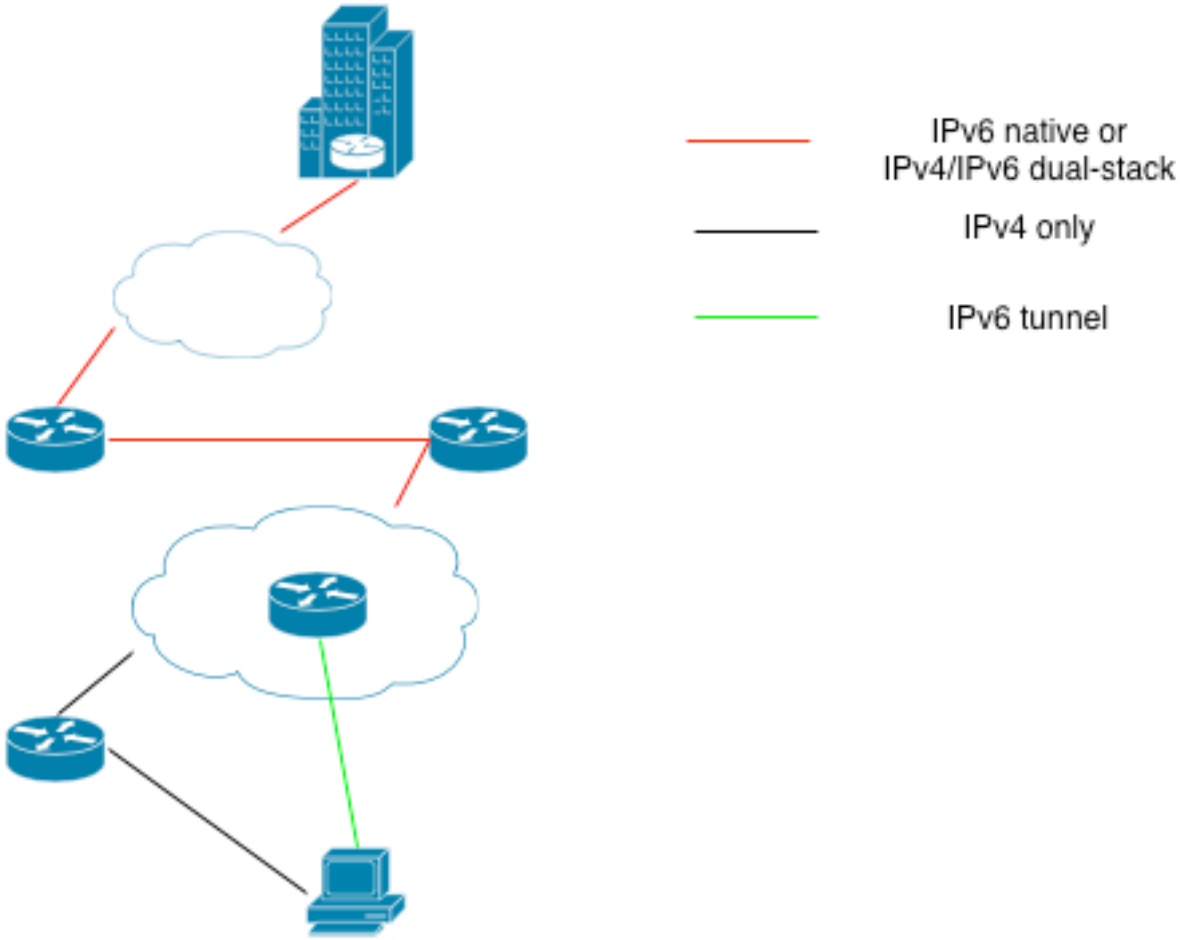
Transition technologies

- The IETF's v6ops working group decided to define a number of scenarios
  - Enterprise
  - ISP
  - Applications
  - 3GPP
  - (Campus)
- Each scenario specifies
  - Problems
  - Requirements on the solutions to the problems
- The thought was that based on this a decision could be made between the various transition technologies that where proposed
  - This has turned out much harder to do than anticipated

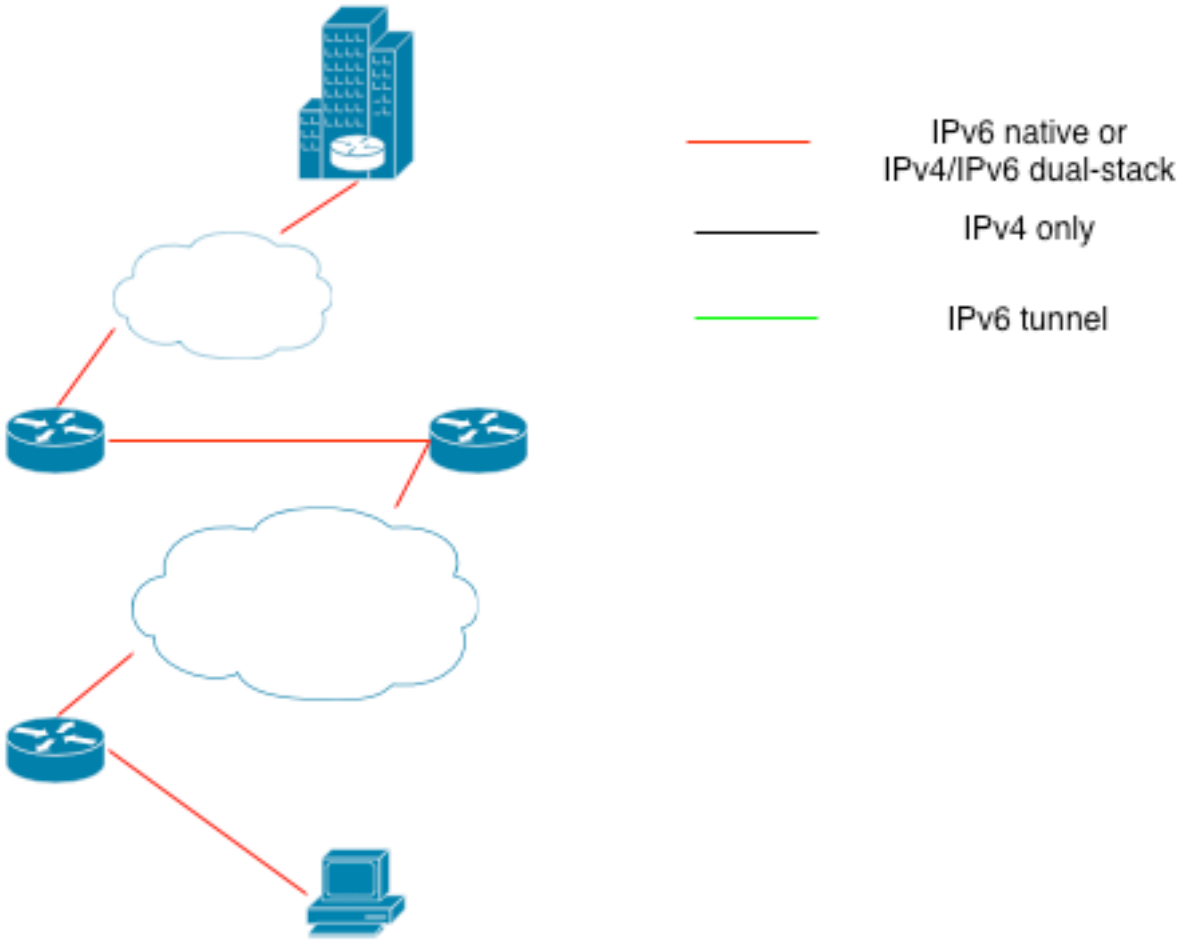
End-user I



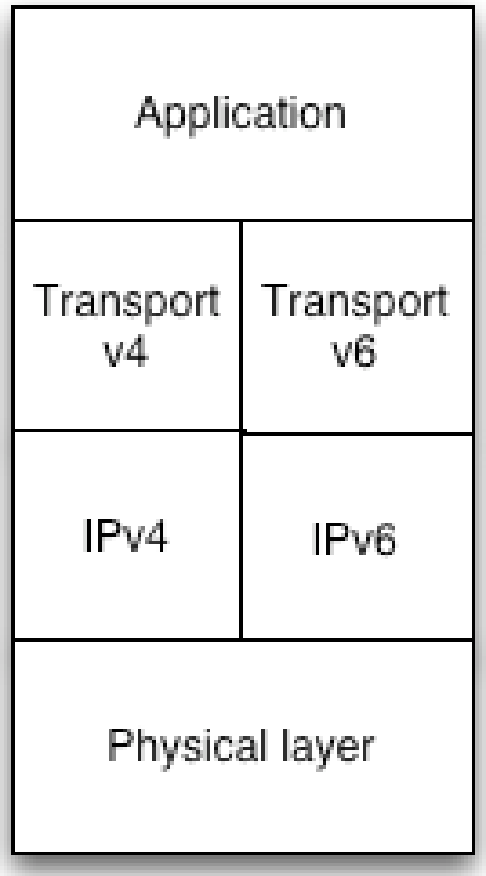
End-user 2



The “lucky” end-user



Dual-stack



www.kurtis.pp.se

TCPv4 / TCPv6

Ethernet

Dual-stack - IOS

```
interface FastEthernet0

 ip address 195.43.225.65 255.255.255.224

 ip pim sparse-mode

 speed 10

 half-duplex

 ipv6 enable

 ipv6 address 2001:670:87:1::/64

 ipv6 nd prefix-advertisement 2001:670:87:1::/64 300 300
   autoconfig

 no cdp enable
```

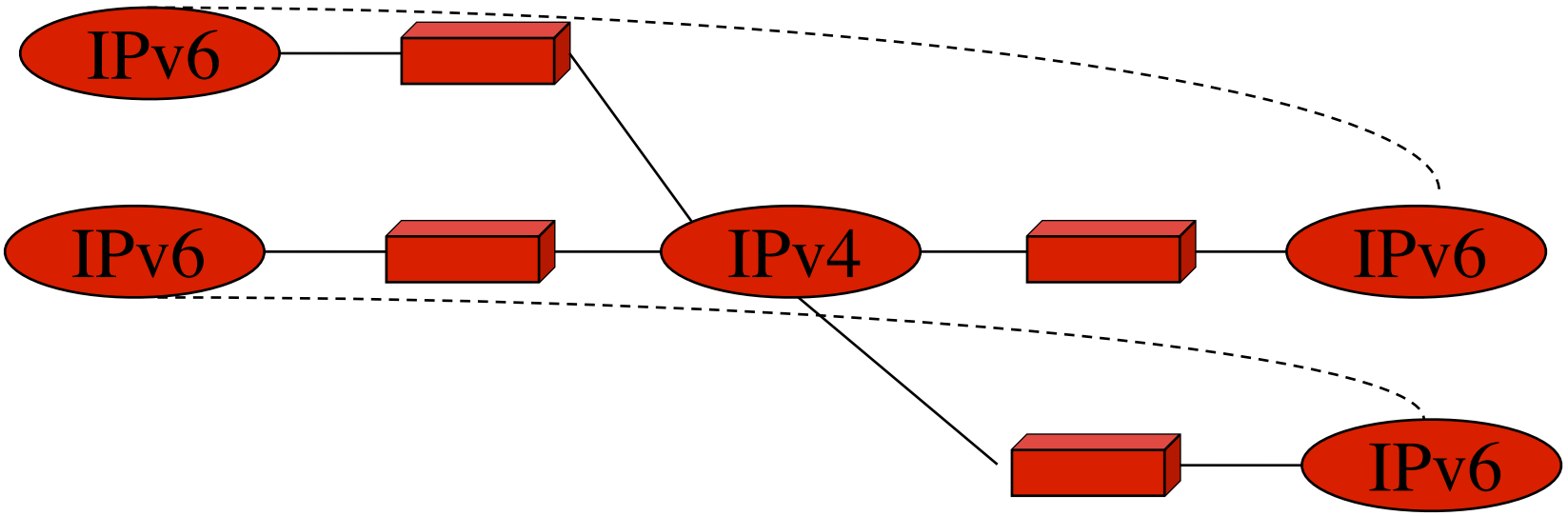
Dual -stack FreeBSD

```
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500

        options=b<RXCSUM,TXCSUM,VLAN_MTU>

        inet 194.15.141.69 netmask 0xfffffffe0 broadcast 194.15.141.95

        inet6 fe80::226:54ff:fe08:9e4c%xl0 prefixlen 64 scopeid 0x1

        inet6 2001:670:87:1:226:54ff:fe08:9e4c prefixlen 64 autoconf

        ether 00:26:54:08:9e:4c

        media: Ethernet autoselect (100baseTX <full-duplex>)

        status: active
```

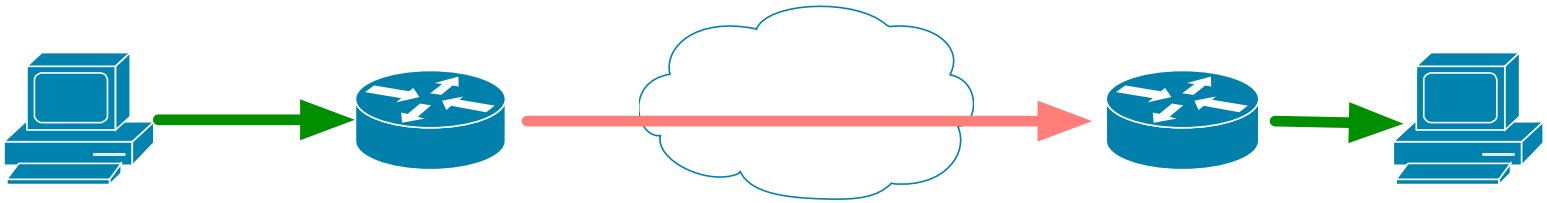
Tunnels

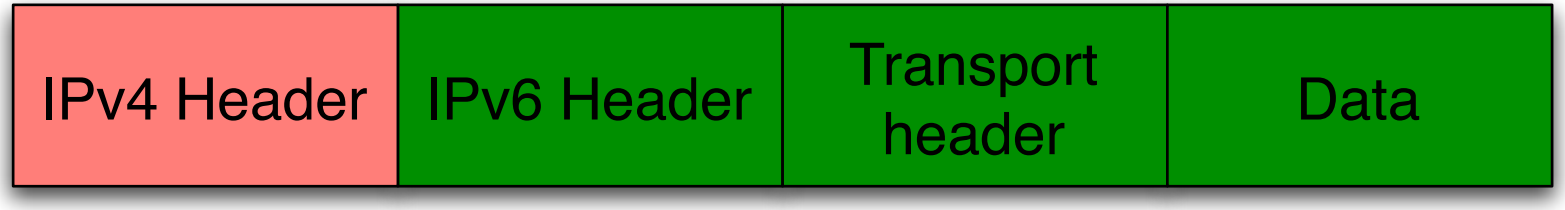
- Tunnels can be static
  - E.g. permanently configured
- Dynamic
  - ”Dial-on-demand”
  - The tunnels are established when they are needed
- Tunnels can be between
  - Host-Host
  - Host-Router
  - Router-Router

Tunnels



Tunnel encapsulation





Tunnels example - Router-to-Router

Router A

```
ipv6 unicast-routing
!
interface Tunnel10
 description TO KQ FI
 no ip address
 ipv6 enable
 ipv6 address 2001:670:87:3001::1/126
 tunnel source 195.43.225.65
 tunnel destination 193.94.250.58
 tunnel mode ipv6ip
 tunnel checksum

ipv6 route ::/0 Tunnel
```

Router B

```
ipv6 unicast-routing
!
interface Tunnel10
 description TO Kurtis
 no ip address
 ipv6 enable
 ipv6 address 2001:670:87:3001::2/126
 tunnel source 193.94.250.58
 tunnel destination 195.43.225.65
 tunnel mode ipv6ip
 tunnel checksum
!
ipv6 route 2001:670:87::/48 Tunnel10
```

Tunnels example - Router-to-host

- The router side looks the same…
- Host (FreeBSD) :

```
$ ifconfig gif0 inet 194.112.11.163 195.43.225.65

$ ifconfig gif0 inet6 2001:670:87:3001::6 prefixlen 12

$ route add -inet6 2001:670:87:3001::4 -prefixlen 126 -interface gif0

$ route add -inet6 default 2001:670:87:3001::5
```

Tunnels example - Router-to-Host

```
$ ifconfig gif0

gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280

  inet 194.112.11.163 --> 195.43.225.65 netmask
   0xffffff00

  inet6 2001:670:87:3001::6 --> :: prefixlen 12
```

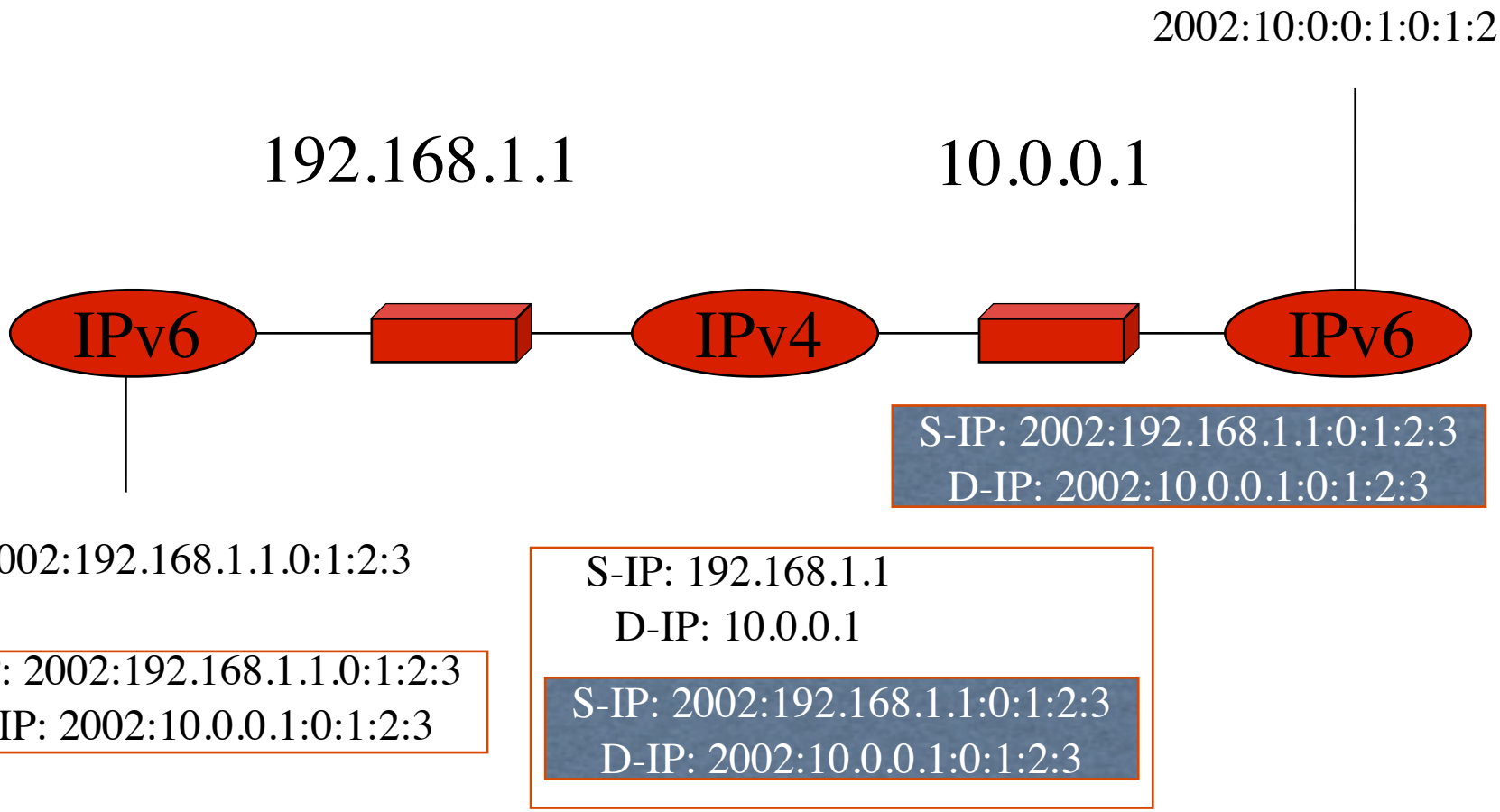
6to4

- In 6to4 an IPv4 gateway address is embedded in the IPv6 address
  - The IPv6 prefix for these addresses is 2002::/16

6to4

- 6to4
  - Base specification in RFC3056
  - Anycast prefix for the gateway described in RFC3068
- The idea behind 6to4 is to use it as a IPv6 addressing and transport mechanism for hosts (networks) with at least one global unicast IPv4 address
  - Temporary methods while IPv4 and IPv6 co-exists
  - Also works if the gateway is a NAT device
  - But not if the gateway is behind a NAT device
- The 6to4 prefix is the prefix that is formed by the use of 6to4 transport and used within the 6to4 site
- Relay router
  - The router that bridges between 6to4 and native IPv6
- Packets are sent as type IPv41
- For anycast 192.88.99.1/24 is announced as relay

6to4


2002:10:0:0:1:0:1:2:3
192.168.1.1
10.0.0.1
IPv6
IPv4
IPv6
S-IP: 2002:192.168.1.1:0:1:2:3
D-IP: 2002:10.0.0.1:0:1:2:3
2002:192.168.1.1.0:1:2:3
S-IP: 192.168.1.1
D-IP: 10.0.0.1
S-IP: 2002:192.168.1.1.0:1:2:3
D-IP: 2002:10.0.0.1:0:1:2:3
S-IP: 2002:192.168.1.1:0:1:2:3
D-IP: 2002:10.0.0.1:0:1:2:3

6to4 - Cisco IOS example

```
interface Loopback0
 ip address 192.168.30.1 255.255.255.0
 ipv6 address 2002:c0a8:1e01:1::/64 eui-64
!
interface Tunnel0
 no ip address
 ipv6 unnumbered Ethernet0
 tunnel source Loopback0
 tunnel mode ipv6ip 6to4
!
ipv6 route 2002::/16 Tunnel0
```

Thanks to Philip Smith@Cisco for the example

6to4 - FreeBSD example

In /etc/rc.conf

```
# Local IPv4 address for 6to4 tunneling interface.

# its IPv6 address will be "2002:c0a0:0001::1"

stf_interface_ipv4addr="192.168.0.1"

# RFC3068 suggests anycast IPv4 address 192.88.99.1

# for 6to4 routers, but you can use other IPv4 address

# according to the site-adminitrator configuration.

ipv6_defaultrouter="2002:c058:6301::"
```

6to4 Relay - Cisco IOS example

```
interface Loopback0
 ip address 192.168.99.1 255.255.255.0
 ipv6 address 2002:c0a8:6301:1::/64 eui-64
!
interface Tunnel0
 no ip address
 ipv6 unnumbered Ethernet0
 tunnel source Loopback0
 tunnel mode ipv6ip 6to4
!
ipv6 route 2002::/16 Tunnel0
ipv6 route ::/0 2002:c0a8:1e01::1
```

Thanks to Philip Smith@Cisco for the example

Teredo

- Many transition technologies are based on tunnelling Ipv6 packets in IPv4
  - This is a problem with NAT devices (or any middle box) as they often want to do (stateful) packet inspection
- For example 6to4 is IPv6 packets encapsulated in IPv4 with IP version number 41
  - Not supported by several types of NAT
  - Or requires manual configuration
- Teredo solves this by encapsulating IPv6 packets in a IPv4 UDP packet
  - That is both a IPv4 header and a UDP header
  - TCP and UDP is always supported by NAT

Teredo

- NAT comes in three types
- Cone NAT
  - The NAT box maintains a translation table that contains a mapping between a source address and source port to a external address and port.
  - When installed in the translation table, traffic from any global address t the external address will be translated
- Restricted NAT
  - Only accept translation for know source addresses and ports
- Symmetrical NAT
  - Only maps internal addresses and ports to specific external addresses and ports

Teredo

- Terdo client
  - The client behind the NAT device that can not obtain a IPv6 address “naturally
  - Does not have a globally reachable IPv4 address
  - Configures a Teredo interface with a Teredo address obtained by a Teredo server
- Teredo Server
  - A server that has both global IPv4 and IPv6 addresses
  - Works as a “helper” when configuring a Teredo client
  - Assists in the communication between two Teredo clients
  - Listens to UDP port 3544
- Teredo Relay
  - IPv6 router that can pass packets between Teredo clients and native IPv6 nodes
  - Listens to UDP port 3544
- IPv6 nod
  - A node that has a “real” global IPv6 address

Teredo address format

Prefix

- 32-bit Teredo prefix
- 2001::/32

- Server IPv4
  - IPv4 address for a Teredo server

Flags

- 16 bit documenting the type of address and NAT

Port

- UDP port for client side Teredo service (that is the source port when reaching the Teredo server)
- Coded by reversing bits

IPv4 address of the Teredo client NAT box

- That is the external address of the NAT device
- Coded by reversing the bits

Teredo

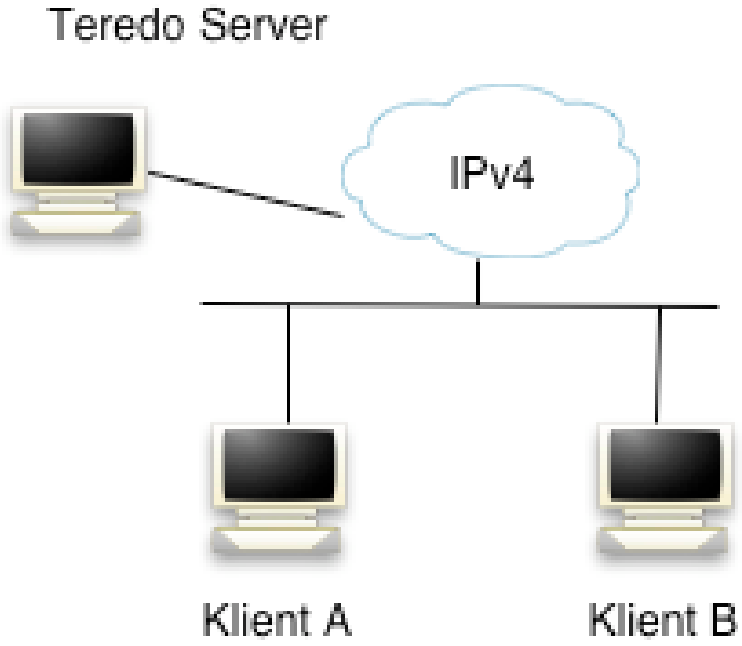
- Start sequence
  - Decide the type of NAT (Cone, restricted Cone, symmetrical)
  - If it is symmetrical NAT, Teredo fails
  - If not symmetrical NAT the set-up succeeds and the Teredo client configures a Teredo address
- When the connection is established, the Teredo client will send a number of "bubbles"
  - These are empty IPv6 packets that are used to create a mapping (state) in the NAT device

Teredo

- Windows XP SP2 and Windows XP Advance Networking pack will try and find a Teredo server by resolving teredo.ipv6.microsoft.com.
  - This cna be changed with

```
netsh interface ipv6 set teredo servername=
```

Teredo

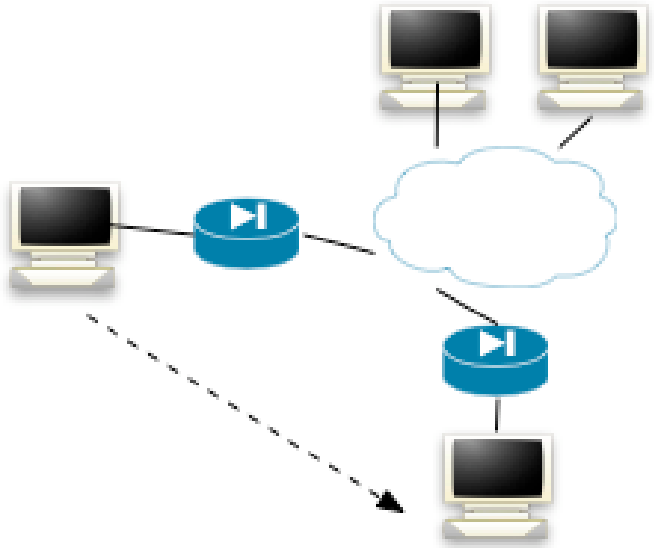


Teredo

- Communication between two routers on the same link
  - Client A sends a bubble packet to a not yet decided Teredo IPv4 Discovery address. The destination in the IPv6 header is Client B's Teredo address
  - Cleint B will answer to A's IPv4 unicast address and its Teredo port
  - Traffic will flow directly

Teredo

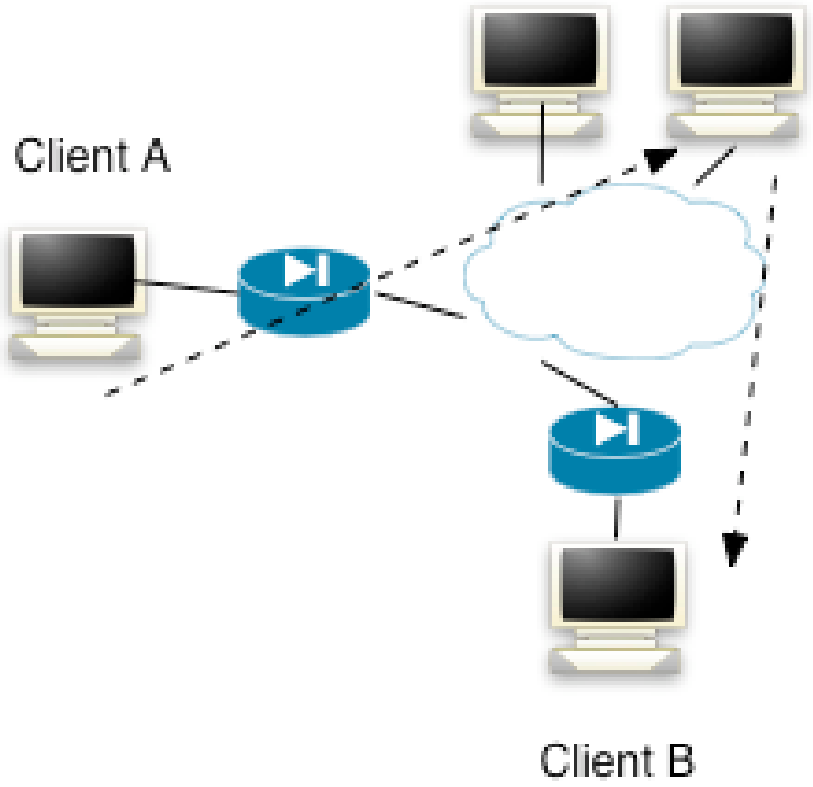
Cone NAT



As mapping exists in the NAT device, the clients can communicate directly

Teredo

Restricted NAT



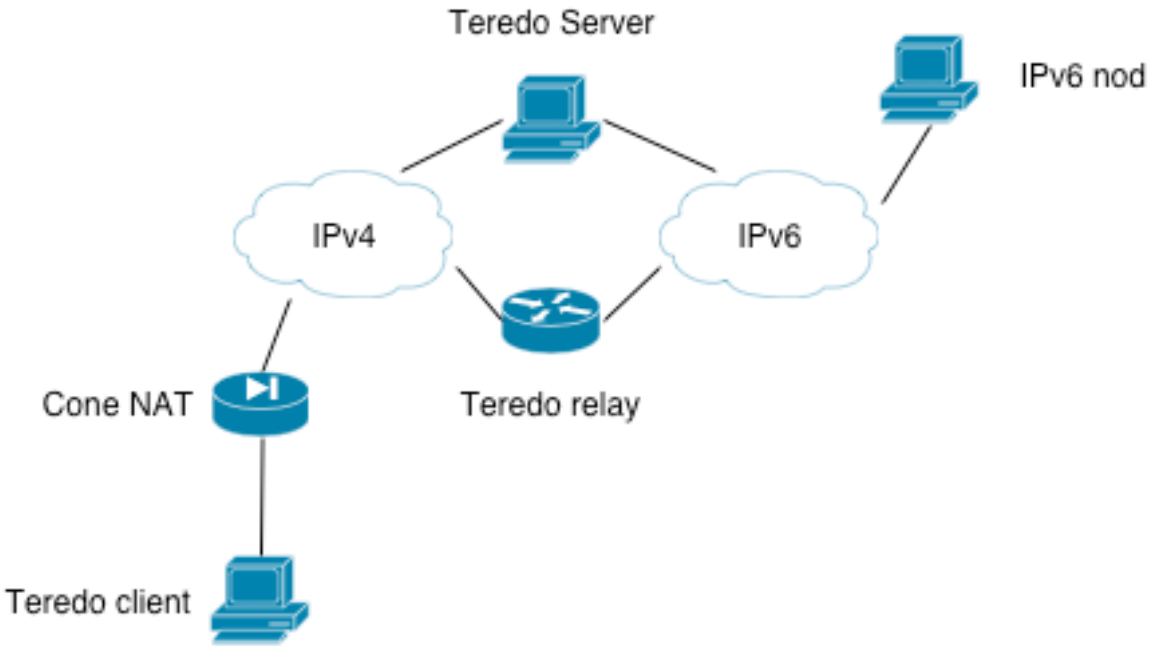
Teredo

- Client A sends a bubble packet to client B
- B's NAT will discard the packet but A's NAT has created a mapping
- Client A will send a new bubble packet to the Teredo server, that will then forward the packet to client B
- Client B will answer the packet, and thereby creating a mapping in its NAT and will be let through A's NAT
  - As A's NAT already have the mapping

Teredo

Between a IPv6 node and a Teredo

client behind a Cone NAT



Teredo

- Teredo client starts with finding a close Teredo relay by sending a ICMPv6 Echo request packet through its Teredo server
- The IPv6 node will answer with a ICMPv6 Echo Reply. By the IPv6 routing infrastructure it will choose the closest Teredo gateway
- The Teredo relay will encapsulate the packet and as it is a Cone NAT it it will be able to send it directly
- The Teredo client will get the IPv4 address of the Teredo relay

Teredo

- For restricted NAT we do the same thing, but we can not send the packets directly
  - The Teredo relay sends a bubble packet through the Teredo server
  - The Teredo client replies with a bubble packet to the Teredo relay
- Traffic will then start to flow
- Works the same way when a IPv6 node initiates the traffic

Translation mechanisms

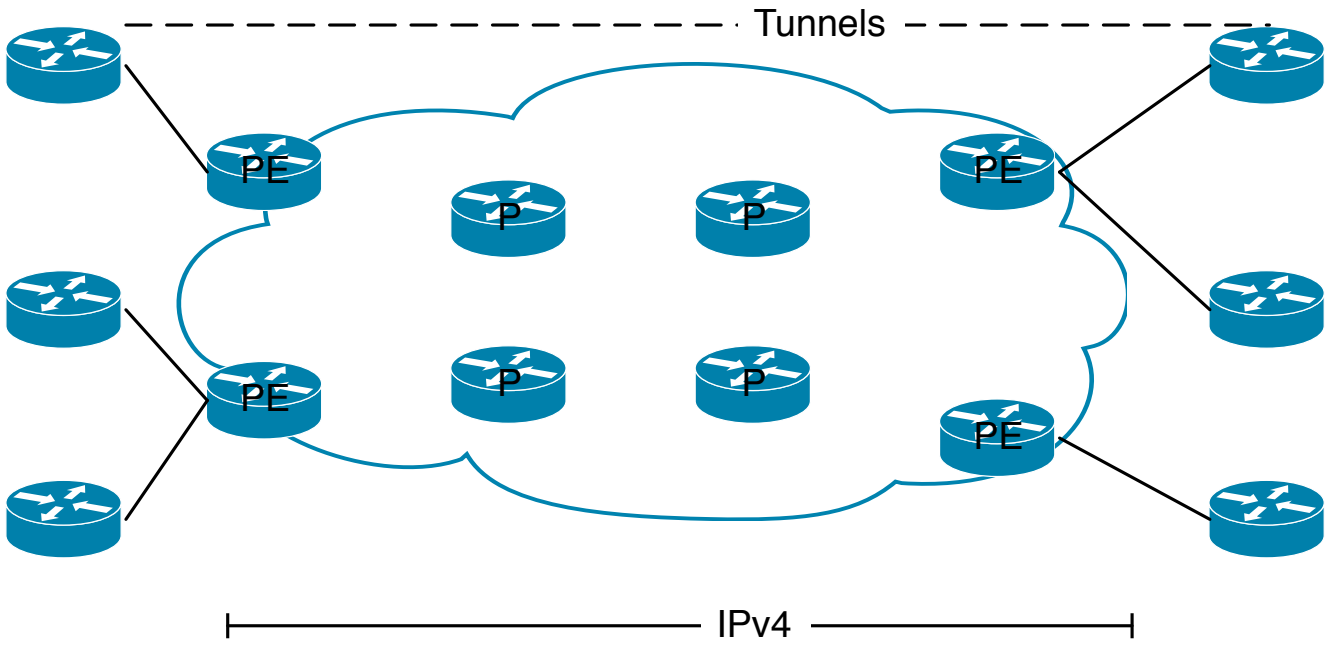
- In addition there are a set of transition technoliges based on translating between IPv4 and IPv6
- Translation
  - NAT-PT (RFC 2766 & RFC 3152)
  - TCP-UDP Relay (RFC 3142)
  - DSTM (Dual Stack TransitionMechanism)
- API
  - BIS (Bump-In-the-Stack) (RFC 2767)
  - BIA (Bump-In-the-API)
- ALG
  - SOCKS-based Gateway (RFC 3089)
  - NAT-PT (RFC 2766 & RFC 3152)

IPv6 and MPLS

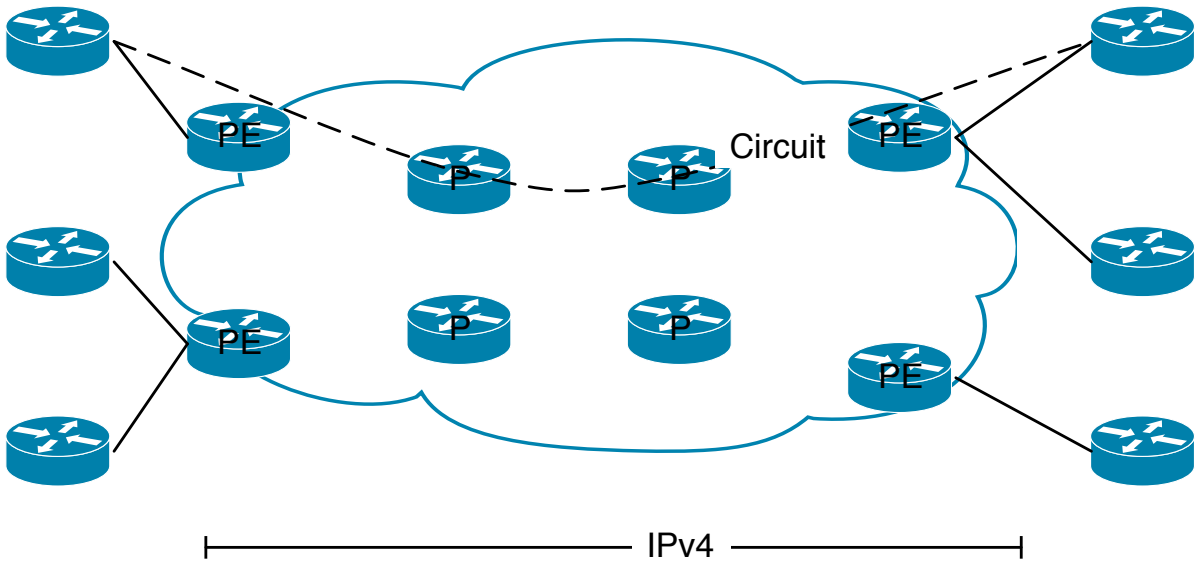
Scenarios

- Many operators already have MPLS in their networks for other reasons (services). This can be used for IPv6 migration by
  - IPv6 over tunnels
  - IPv6 over circuit MPLS
  - Native IPv6 MPLS (LDP & IGP over IPv6)
  - IPv6 Provider Edge Router (6PE) over MPLS

IPv6 over tunnels



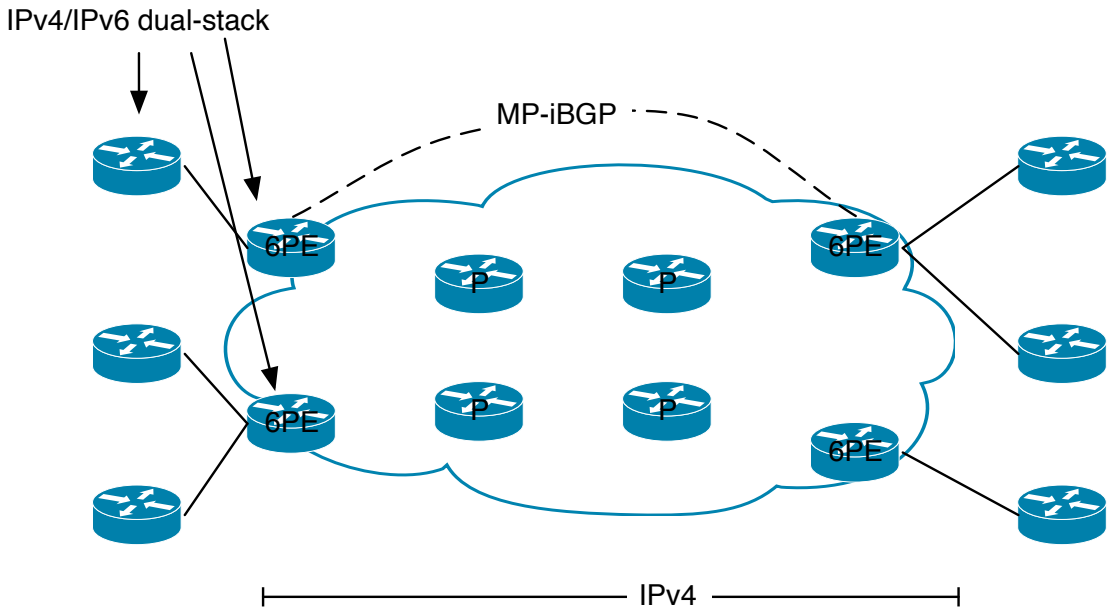
IPv6 over Circuit MPLS



Native MPLS + IPv6

- Requires upgrading of infrastructure
  - Control plane with IPv6
  - IGP/EGP
  - LDP over IPv6

6PE



6PE

- IPv4 core with MPLS as “usual”
- PE routers upgraded to dual-stack
- iBGP uses MP-BGP to distribute reachability between 6PE routers
- 6PE transports IPv6 packets to other 6PE in MPLS

What do I need to do to get started?

Checklist to get started

1. Apply for addresses from APNIC or your ISP
2. Make an address plan
3. Make an inventory of internal systems and the use of IPv4 addresses in interal system
4. In the case you have internal / internally developed systems that are dependent on IPv4 - create a porting plan
5. Make a migration calendar
6. Start to migrate!

Apply for IPv6 address space - LIR

- Fill in the request form!
- If you do not have an allocation
  - Take this opportunity to go and talk to your friendly hostmasters!

Applying for IPv6 address space from your ISP

- Depends on which ISP you use
- Most ISPs will give out /48s to end-users
  - But also /56 are becoming common
- Start with assessing your minimum needs
  - Assume a /64 per broadcast domain
  - I.e per each Ethernet/VLAN/etc
  - The assessment shall meet your needs for the coming two years
- Contact the ISPs customer service to find out what their policy is and how you order

Make an addressing plan

- In the beginning you where supposed to run EUI64 addresses everywhere
  - And router advertisement to announce it
- First of all, try and use the same binary-chop method as the RIR/LIRs do
  - But you can adjust it to your own needs
- Always allocate /64s to services
  - That is so you can later migrate them to their own LANs if you have to
  - Do not you use EUI64 address for services
  - Allocate static address for services

Make an inventory of the systems

- List workstations
- Servers and services addresses
- Analyse DNS and DHCP servers
- Verify which clients will try and use IPv6 as transport
  - Can the associated servers reply over IPv6 transport?
- Are IPv4 used in referrals?

To keep in mind...

- Start with configuring IPv6 addresses on infrastructure such as routers, switches etc
- Continue to activate IPv6 on services
- Configure IPv6 on clients
- Update DNS
  - The last can be hard to decide in which order to do
  - Will depend on how client/serves will handle IPv6

Configuration examples

-

networking equipment

Thanks to Ron Bonica@Juniper and Philip Smith@Cisco
for some of the examples!

Configuration - Cisco IOS

- IPv6 is supported in all modern IOS releases
  - If you don't have IPv6 support you will have other problems :-)
  - Licenses can be a problem though
- Activating IPv6 (Global command)
  - `router# ipv6 unicast-routing`
- To enable IPv6 CEF
  - `router# ipv6 cef`

Configuration - Cisco IOS-XR/IOS

- IPv6 on Cisco XOS
  - Enabled by default
- Interface commands (For IOS and XOS)
  - To activate IPv6 on an interface
    - router(config)# ipv6 enable
  - To configure a static IP address
    - `router(config)# ipv6 address 2001:db8:1::/64`
  - To configure the interface with an EUI64 address
    - `router(config)# ipv6 address 2001:db8:1::/64 eui-64`

Configuration - JunOS

- IPv6 enabled by default
- Interface configuration

```
interfaces {
 fe-3/0/0 {
  unit 0 {
   family inet6 {
    address 2001:db8:1::45c/64;
   }
  }
 }
}
```

Configuration - JunOS

- Dual-stack interface configuration

```
interfaces {
  fe-3/0/0 {
    unit 0 {
      family inet {
        address 10.1.1.1/24;
      }
      family inet6 {
        address 2001:db8:1::45c/64;
      }
    }
  }
}
```

Configuration - JunOS

- EUI64 interface address configuration

```
interfaces {
  fe-3/0/0 {
    unit 0 {
      family inet6 {
        address 2001:db8:1::45c/64 eui-64;
      }
    }
  }
}
```

Interface status - IOS

- Link-locals

```
Router1# conf t
Router1(config)# ipv6 unicast-routing
Router1(config)# ^Z
Router1#sh ipv6 interface
Ethernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is
FE80::A8BB:CCFF:FE00:1E00
  No global unicast address is configured
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF00:1E00
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
```

Interface status - IOS

- Interface status (EUI64 configured)

```
Router1#sh ipv6 interface eth0/0
Ethernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::A8BB:CCFF:FE00:1E00
  Global unicast address(es):
    2001:DB8::A8BB:CCFF:FE00:1E00, subnet is 2001:DB8::/64 [EUI]
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF00:1E00
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
  ND advertised reachable time is 0 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
```

Interface status - IOS-XR

```
RP/0/0/CPU0:as4byte#sh ipv6 interface gig 0/2/0/1
GigabitEthernet0/2/0/1 is Up, line protocol is Up, Vrfid is
0x60000000
  IPv6 is enabled, link-local address is fe80::204:6dff:fea2:90fd
  Global unicast address(es):
     2001:db8::204:6dff:fea2:90fd, subnet is 2001:db8::/64
  Joined group address(es): ff02::6 ff02::5 ff02::2
      ff02::1
  MTU is 1514 (1500 is available to IPv6)
  ICMP redirects are disabled
  ICMP unreachables are enabled
  ND DAD is enabled, number of DAD attempts 1
  ND reachable time is 0 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  Hosts use stateless autoconfig for addresses.
  Outgoing access list is not set
  Inbound  access list is not set
```

Interface status - JunOS

```
regress@UI-J6300-2> show interfaces fe-3/0/0
  Logical interface fe-3/0/0.0 (Index 68) (SNMP ifIndex 42)
                      . . .
    Flags: SNMP-Traps Encapsulation: ENET2
    Input packets : 70
    Output packets: 79
    Protocol inet, MTU: 1500
      Flags: None
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 1.1.1/24, Local: 1.1.1.2, Broadcast: 1.1.1.255
    Protocol inet6, MTU: 1500
      Flags: Is-Primary
      Addresses, Flags: Is-Preferred
        Destination: fe80::/64, Local: fe80::205:85ff:fec7:683c
      Addresses, Flags: Is-Default Is-Preferred Is-Primary
        Destination: 2001:db8:2:1::/64, Local: 2001:db8:2:1::2
```

Configuration examples

-

Operating systems

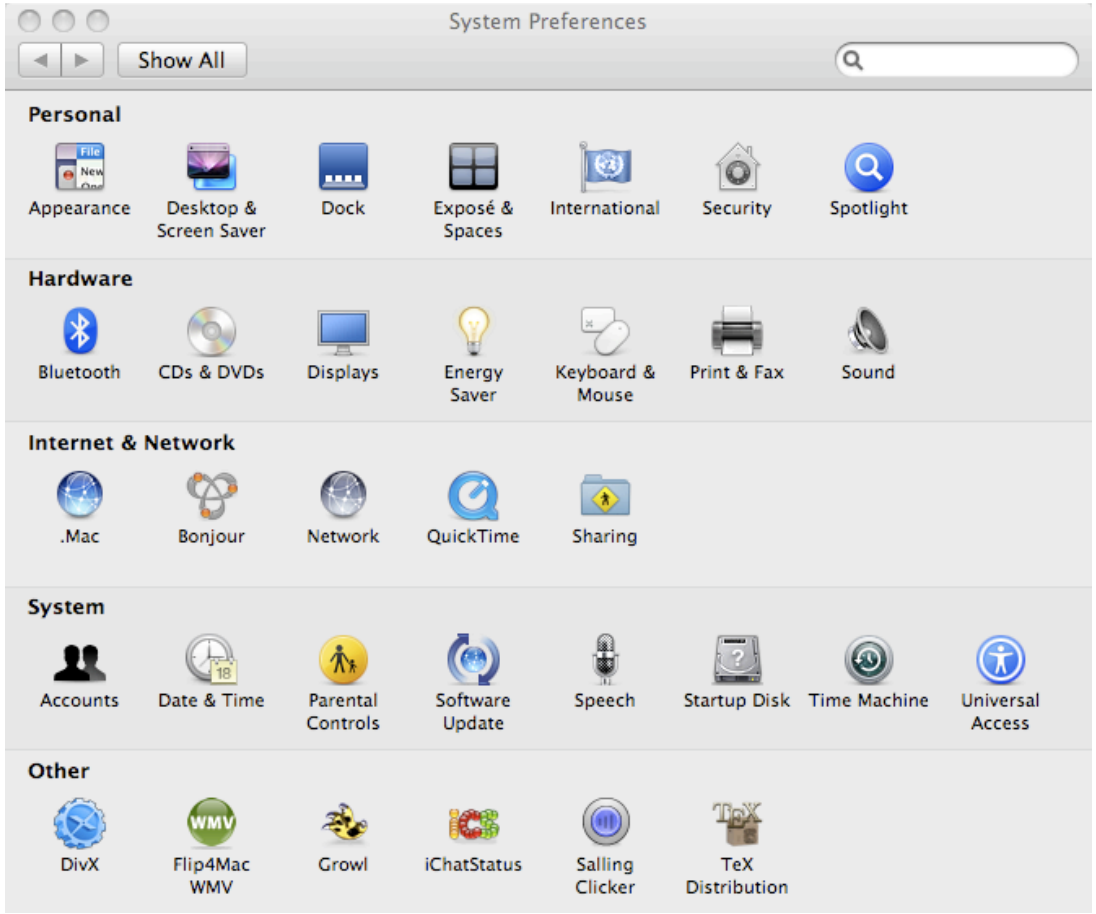
FreeBSD

- FreeBSD6.3-REL and FreeBSD-7.0-CURRENT all supports IPv6 per default
- In /etc/r.conf you only need to add

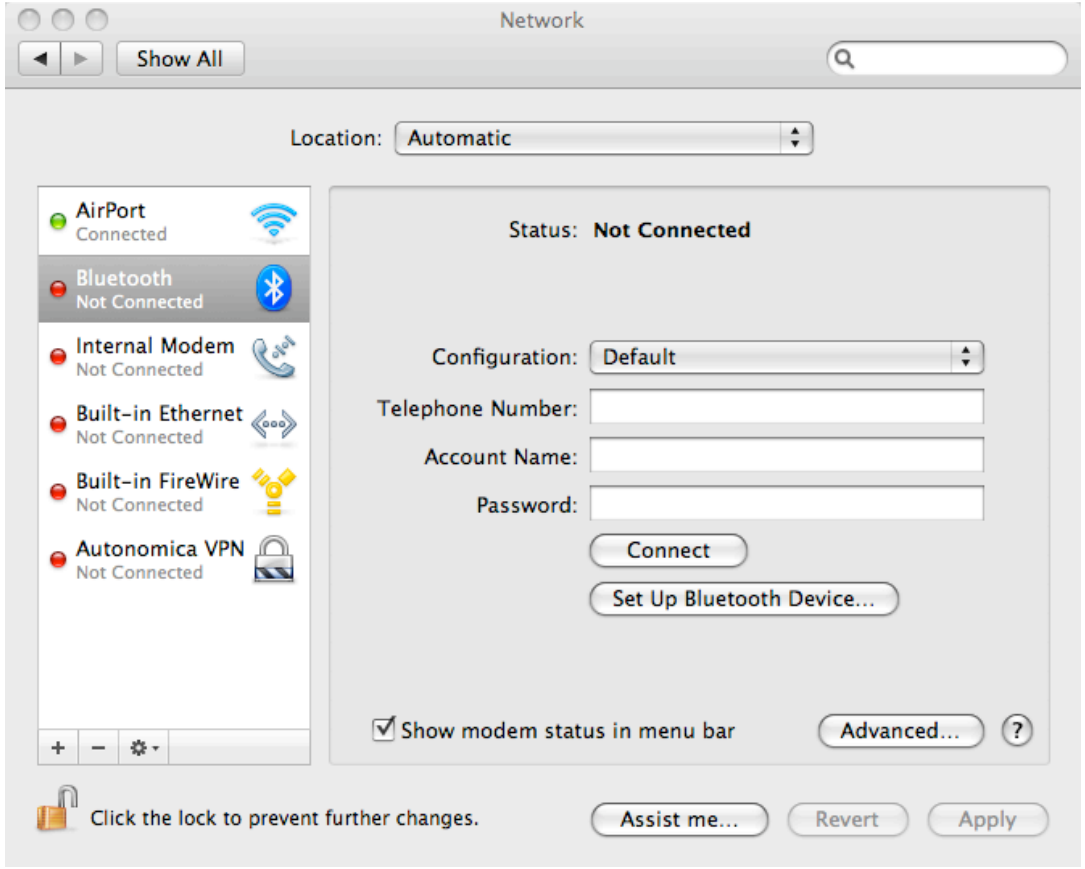
```
ipv6_enable=YES                         # Set to YES to set up for IPv6.
ipv6_ifconfig_bge0="2001:db8:1::39/64" # Sample manual assign entry
ipv6_ifconfig_bge0_alias0="2001:db8:1::40/64"
ipv6_defaultrouter="2001:db8:1::1"      # Use this for 6to4 (RFC 3068)
ipv6_ipv4mapping=NO                     # Set to NO to disable IPv4 mapped IPv6 addr
```

- Reboot and it should work!

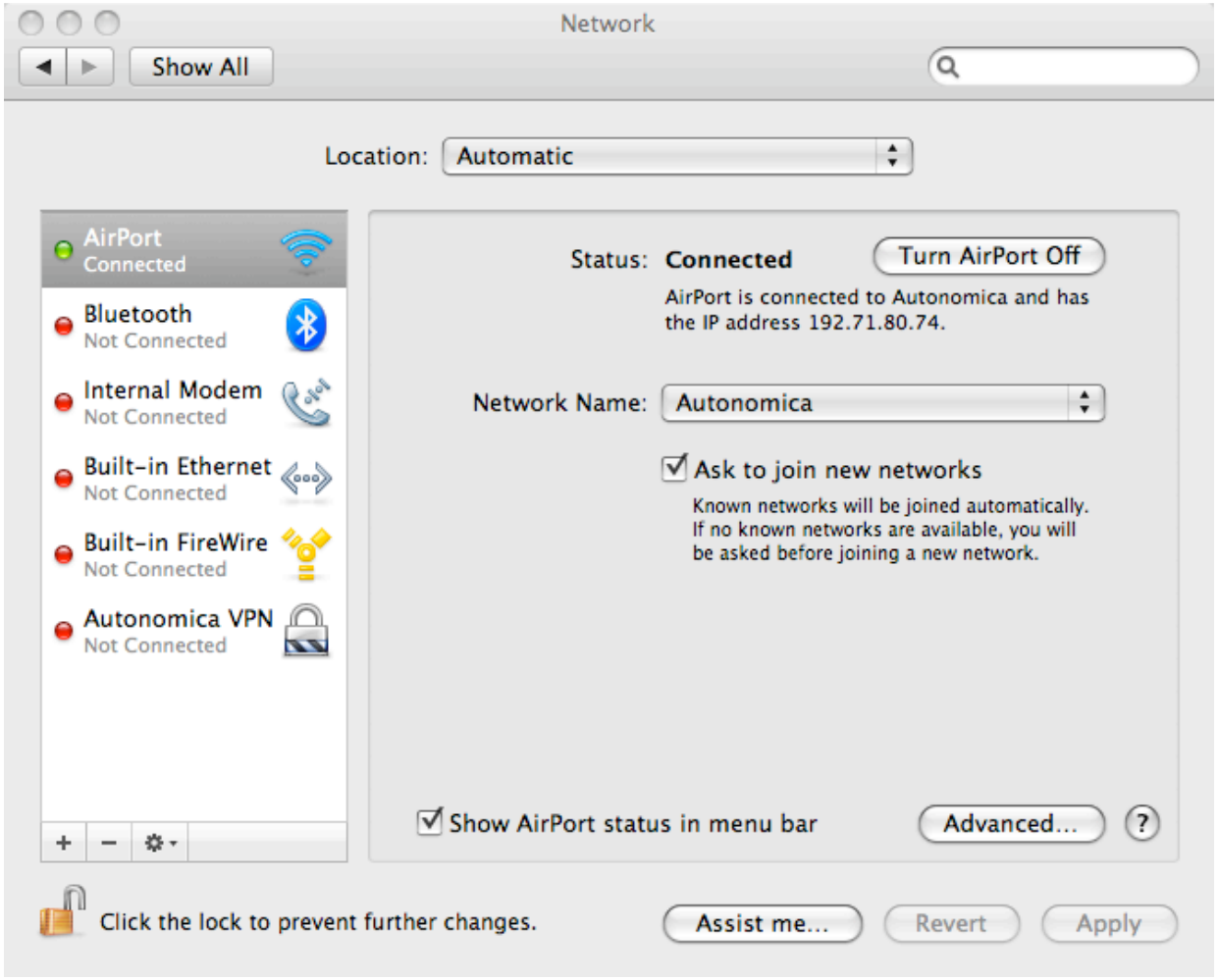
OS X


System Preferences
Show All
Personal
Appearance
Desktop & Screen Saver
Dock
Exposé & Spaces
International
Security
Spotlight
Hardware
Bluetooth
CDs & DVDs
Displays
Energy Saver
Keyboard & Mouse
Print & Fax
Sound
Internet & Network
.Mac
Bonjour
Network
QuickTime
Sharing
System
Accounts
Date & Time
Parental Controls
Software Update
Speech
Startup Disk
Time Machine
Universal Access
Other
DivX
Flip4Mac WMV
Growl
iChatStatus
Salling Clicker
TeX Distribution

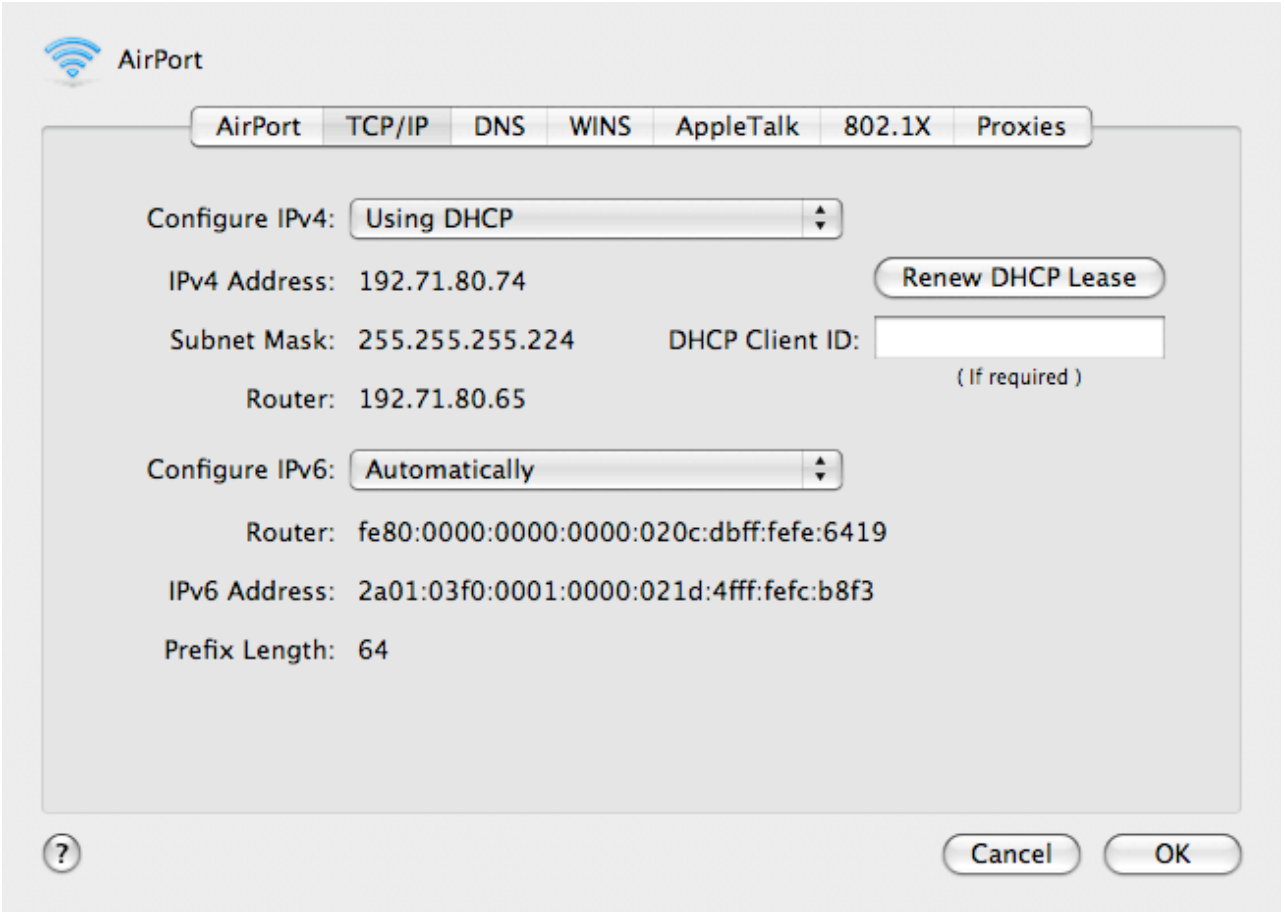
OS X


Network
Show All
Location: Automatic
AirPort
Connected
Bluetooth
Not Connected
Internal Modem
Not Connected
Built-in Ethernet
Not Connected
Built-in FireWire
Not Connected
Autonomica VPN
Not Connected
Status: Not Connected
Configuration: Default
Telephone Number:
Account Name:
Password:
Connect
Set Up Bluetooth Device...
Show modem status in menu bar
Advanced...
Click the lock to prevent further changes.
Assist me...
Revert
Apply

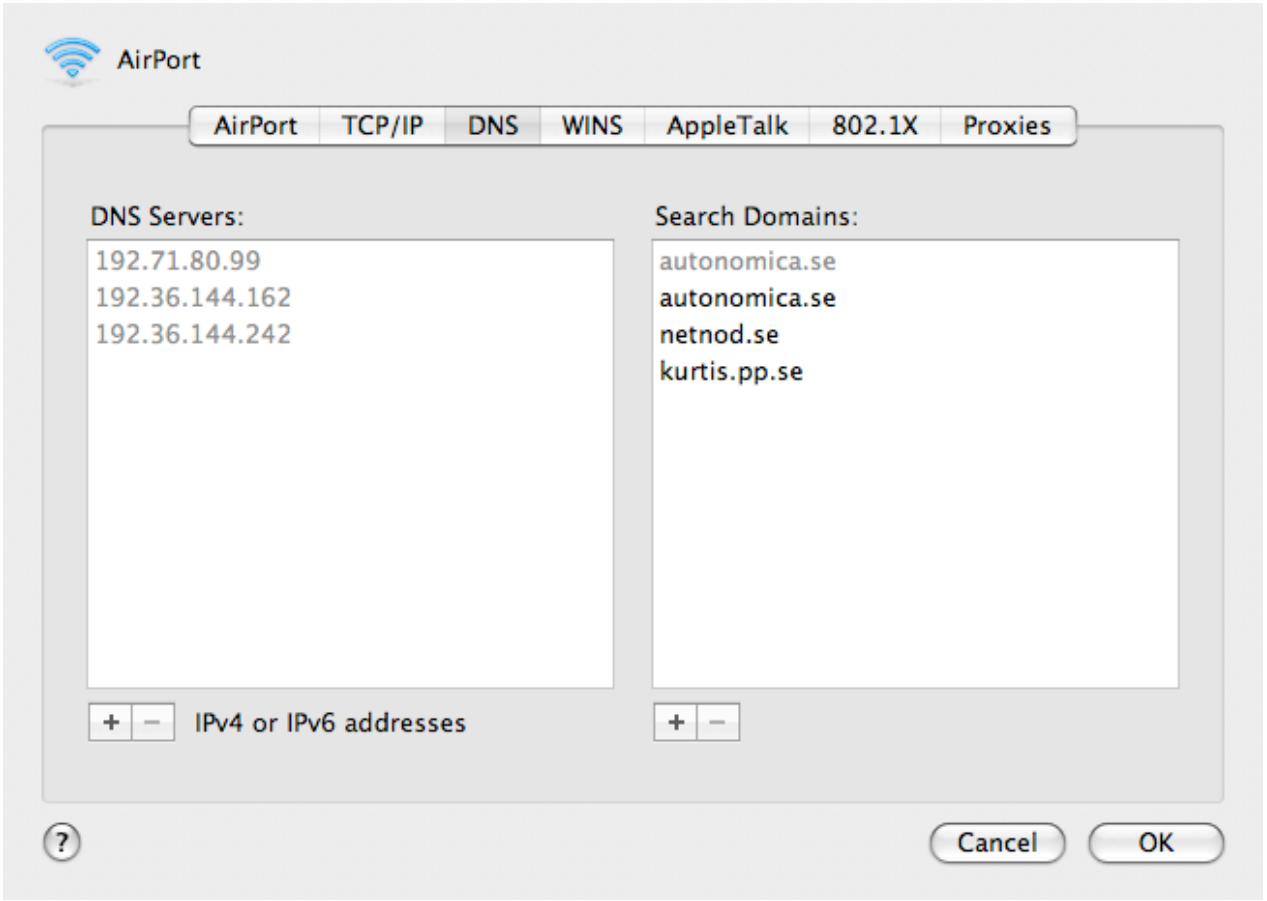
OS X


Network
Show All
Location: Automatic
AirPort
Connected
Bluetooth
Not Connected
Internal Modem
Not Connected
Built-in Ethernet
Not Connected
Built-in FireWire
Not Connected
Autonomica VPN
Not Connected
Status: Connected
Turn AirPort Off
AirPort is connected to Autonomica and has the IP address 192.71.80.74.
Network Name: Autonomica
Ask to join new networks
Known networks will be joined automatically. If no known networks are available, you will be asked before joining a new network.
Show AirPort status in menu bar
Advanced...
Click the lock to prevent further changes.
Assist me...
Revert
Apply

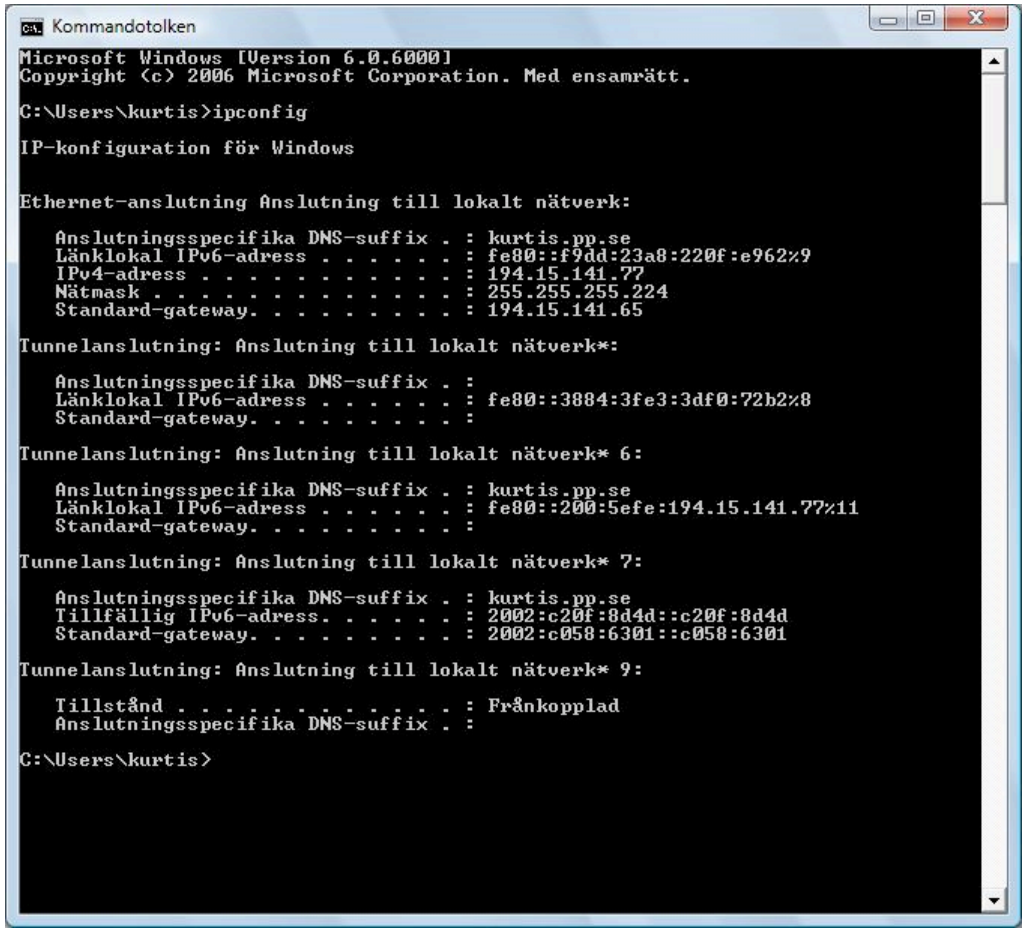
OS X


AirPort
AirPort
TCP/IP
DNS
WINS
AppleTalk
802.1X
Proxies
Configure IPv4:
Using DHCP
IPv4 Address:
192.71.80.74
Renew DHCP Lease
Subnet Mask:
255.255.255.224
DHCP Client ID:
(If required)
Router:
192.71.80.65
Configure IPv6:
Automatically
Router:
fe80:0000:0000:0000:020c:dbff:fefe:6419
IPv6 Address:
2a01:03f0:0001:0000:021d:4fff:fefc:b8f3
Prefix Length:
64
?
Cancel
OK

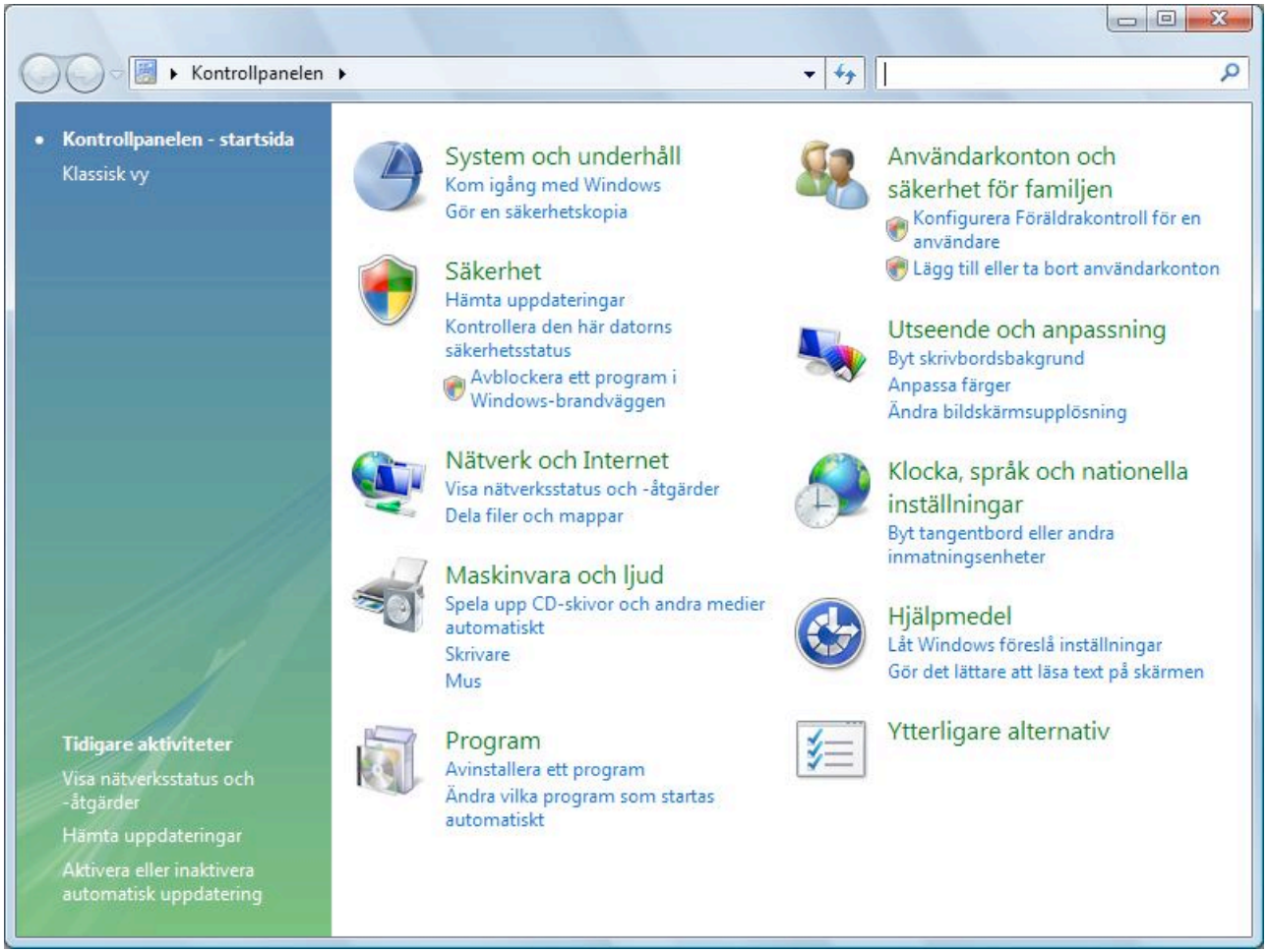
OS X


AirPort
AirPort
TCP/IP
DNS
WINS
AppleTalk
802.1X
Proxies
DNS Servers:
192.71.80.99
192.36.144.162
192.36.144.242
IPv4 or IPv6 addresses
Search Domains:
autonomica.se
autonomica.se
netnod.se
kurtis.pp.se
Cancel
OK

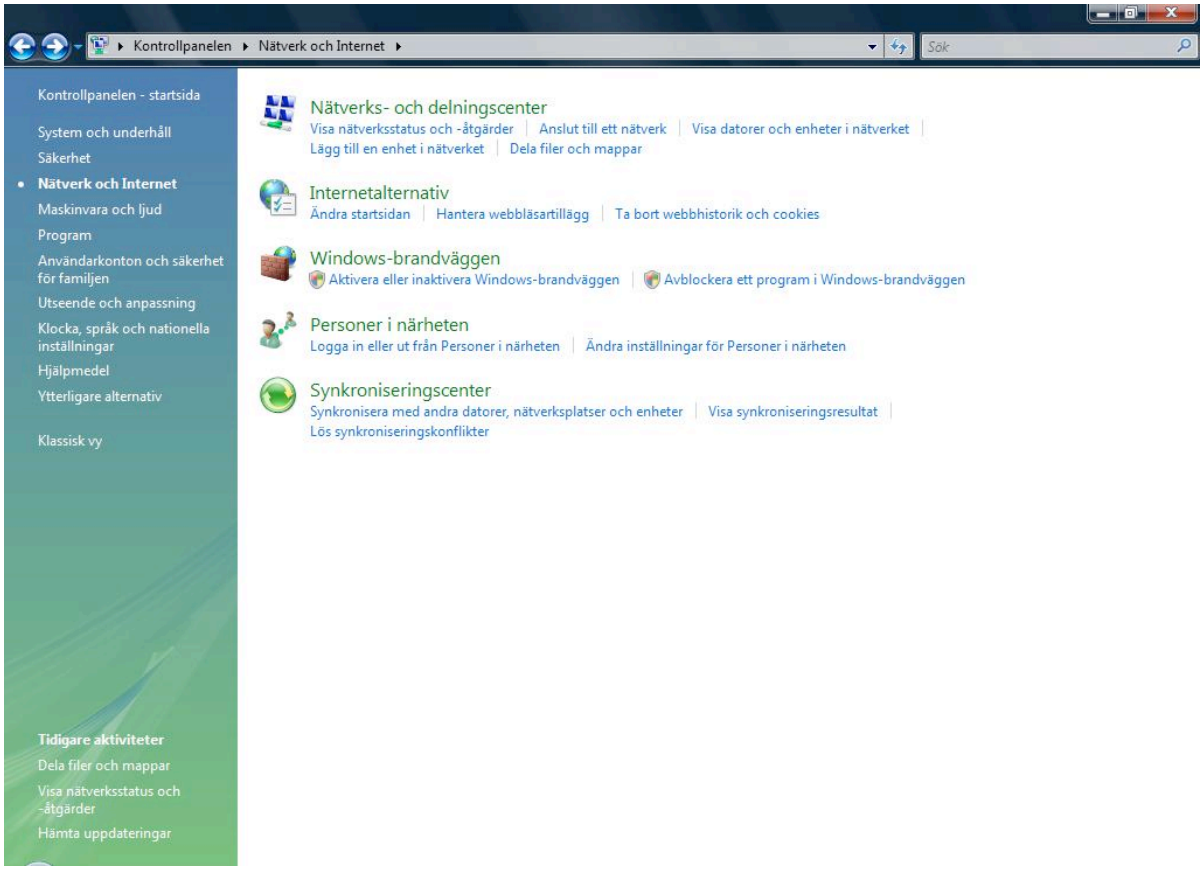
Windows Vista



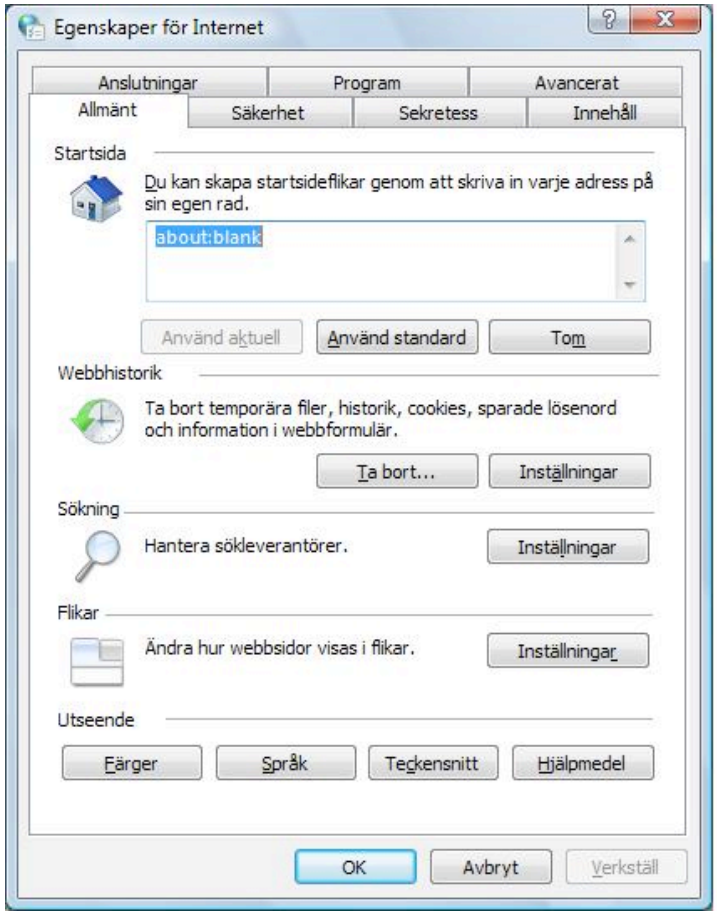
Windows Vista



Windows Vista



Windows Vista



Configuration examples

-

Applications

Bind9

- IPv6 is supported from Bind8
  - You should anyway be running Bind9.4.2
- In most distributions IPv6 is compiled in by default
  - If not, you do
  - `./configure --enable-ipv6 --with-openssl`
- Then all you need in your named.conf is
  - `listen-on-v6 { any; };`
- Your named is now ready to reply to IPv6 queries
- Remains is to build zone files....

Postfix

- In main.cf

```
# IPv6 configuration

#

inet_protocols = all

smtp_bind_address6 = 2001:670:87:2:20e:a6ff:fe3c:6a8b
```

Courier IMAP

- In imapd

```
# DEFAULT SETTING from /usr/local/etc/courier-imap/imapd.dist:

#

#ADDRESS=0

#



ADDRESS=0
```

Apache2

- In httpd.conf

```
Listen * 80
```

- or

```
Listen [2001:670:87:1:226:54ff:fe08:9e4c]:80
```

IPv6 Neighbour discovery

IPv6 Neighbour Discovery

- Described in RFC4861
- Used for
  - Finding link-layer addresses of nodes on the same link
    - Delete state that is no longer in valid
  - Find routers that are willing to forward packets
  - Used to verify reachability on a link

IPv6 Neighbour Discovery

- Router Advertisement
  - Routers announce themselves to a multicast group (if the link layer supports it)
  - Nodes then listen to multicast announcements
  - Includes information on how nodes are expected to obtain an address
- Neighbour solicitation
  - Nodes ask others to send their link-layer address
  - Sent to a multicast group
  - Also used for Duplicate Address Detection, DAD
- IPv6 ND corresponds to IPv4
  - ARP
  - ICMP redirect
  - ICMP router discovery

IPv6 Neighbour Discovery

- Router solicitation
  - Sent by nodes to the routers to force more frequent Router Advertisement
- Optimistic DAD
  - A node can end up having to wait for quite a while until DAD completes
  - The likelihood of a MAC address collision is low
  - Optimistic DAD makes the assumption that you do not have to wait for DAD to complete before you can start sending traffic
    - If it afterwards turns out there was a collision, lets handle that then

IPv6 Stateless address autoconfiguration

Stateless address autoconfiguration

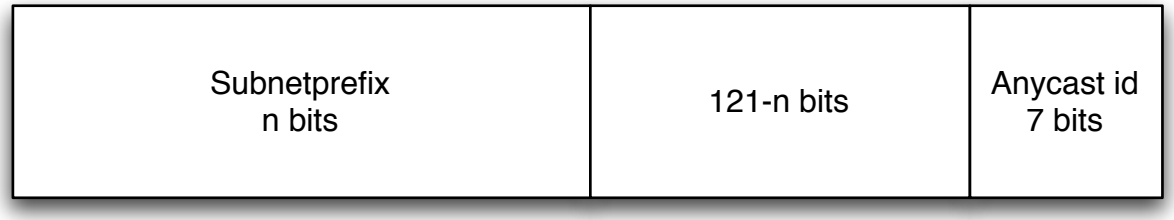
- RFC4862
- For autoconfiguration of IPv6 there are two options
  - Stateful (DHCPv6)
  - Stateless (via RA)
- For stateless autoconfiguration, this is done by combining the address prefix advertised in the RA with the Interface I-D
  - EUI64 or RFC4941
- Thought to help renumbering of a network
- Problem
  - How do I find a DNS server?
  - How do I send update to the DNS server?

Routing and network design

IPv6 Anycast addresses

- An anycast address is an address that is in use in various different hosts
- Through the routing system, the "network" will choose the closest announcement of the anycast address (host/interface)
- For IPv6 the same thing applies
  - But the anycast address can also be used for a link
  - The problem then is that DAD needs to understand it
  - Technology/Draft/Usefulness is under discussion

IPv6 Anycast address



- The subnet prefix is the same as for the rest of the subnet
- In the example above, the leftmost 121 bits are said to be a toplogy region identified by prefix p
  - Within the topology p the anycast address must be maintained as a separate routing entity
  - Outside p it might be aggregated
- Any anycast address might not be used as the source address of an IPv6 packet
- An anycast address must not be assigned to an IPv6 host

IPv6 Anycast

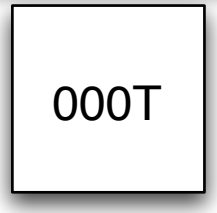
- Required anycast address
  - The subnet-router anycast address is predefined

- Subnet prefix identifies a given link
- The anycast address is syntactically the same as an unicast address, with the interface bits set to 0
- All routers are required to support the subnet-router addres
  - Packets will be delivered to one router on the link

IPv6 multicast

- An IPv6 multicast address identifies a group of interfaces
- An interface may belong to any number of groups
- The multicast address format is

- Where flgs are



IPv6 multicast

- Flags are defined as
  - T=0 IANA assgined, well-known, multicast address
  - T=1 is a non-permanent (transient) multicast address
- Scope is used to limit the scope of a particular multicast group
  - 0 reserved
  - 1 interface-local scope
  - 2 link-local scope
  - 3 reserved
  - 4 admin-local scope
  - 5 site-local scope
  - 6 (unassigned)
  - 7 (unassigned)
  - 8 organization-local scope
  - 9-D (unassigned)
  - E global scope
  - F reserved

IPv6 multicast

- *interface-local* is only valid on a single interface and is only useful for loopback tests of multicast
- *link-local* and *site-local* have the same meaning as for unicast address space
- *admin-local* has an administratively defined scope, and is the smallest non-topolgical scope
- organisation-local is intended to span multiple sites belonging to a single organisation
- IANA assigned, well-known, multicast addresses have the same meaning independent of scope
- IPv6 packet must not have a multicast address as a source address

IPv6 multicast

- Pre-defined multicast addresses
  - Reserved Multicast Addresses: FF00:0:0:0:0:0:0:0
    FF01:0:0:0:0:0:0:0
    FF02:0:0:0:0:0:0:0
    FF03:0:0:0:0:0:0:0
    FF04:0:0:0:0:0:0:0
    FF05:0:0:0:0:0:0:0
    FF06:0:0:0:0:0:0:0
    FF07:0:0:0:0:0:0:0
    FF08:0:0:0:0:0:0:0
    FF09:0:0:0:0:0:0:0
    FF0A:0:0:0:0:0:0:0
    FF0B:0:0:0:0:0:0:0
    FF0C:0:0:0:0:0:0:0
    FF0D:0:0:0:0:0:0:0
    FF0E:0:0:0:0:0:0:0
    FF0F:0:0:0:0:0:0:0
  - All nodes addresses : FF01:0:0:0:0:0:0:1
    FF02:0:0:0:0:0:0:1
  - All routers addresses : FF01:0:0:0:0:0:0:2
    FF02:0:0:0:0:0:0:2
    FF05:0:0:0:0:0:0:2
  - Solicited-node address : FF02:0:0:0:0:1:FFXX:XXXX
  - XX:XXXX are the low order 24 bits of an unicast interface address

Static routing

Static routing

- Cisco IOS syntax
  - **`ipv6 route`** ***`ipv6-prefix/prefix-length`*** {***`ipv6-`*** address | interface-type interface-number} [admin-distance]

```
ipv6 route 2001:db8:0::/48 2001:db8:FFEE::1

ipv6 route 2001:db8:0::/64 Serial0/0
```

Static routing

- Cisco IOS-XR syntax

```
router static
 address-family ipv6 unicast
  ipv6-prefix/prefix-length {ipv6-address |
  interface-type interface-number} [admin-
  distance]
```

- Example

```
router static
 address-family ipv6 unicast
  2001:db8::/64 2001:db8:0:CC00::1  110
```

Static routing

- JunOS syntax

```
[edit routing-options rib inet6.0 ]
    static {
        defaults {
            static-options;
        }
        rib-group group-name;
        route destination-prefix {
            next-hop;
            qualified-next-hop address {
                metric metric;
                preference preference;
            }
            static-options;
        }
    }
```

Static routing

- Example

```
[edit routing-options]
    rib inet6.0 {
        static {
            route 2001:db8::/64 {
                next-hop 2001:db8:0:cc00::1;
                metric 110;
            }
        }
    }
```

OSPF

OSPF

- Defined in RFC2740
  - Called OSPFv3
  - Based on OSPFv2 with many similarities
- Runs over IPv6
- Flooding, Designated router election, SPF, etc
  - Unchanged
- OSPFv3 will work per link instead of per subnet
- New LSAs for IPv6
- No longer support for authentication
  - Instead makes use of IPv6 built in encryption and authentication

Router-ids

- Keep in mind that the router-id is a 32 bit value
  - If you are deploying dual-stack, which is most likely, you will not have an issue
  - Otherwise you will have to assign router-ids

OSPF new LSA types

- Link LSA
  - A link LSA per link
  - Link local scope flooding on the link with which they are associated
  - Provide router link local address
  - List all IPv6 prefixes attached to the link
  - Assert a collection of option bit for the Router-LSA
- Inter-Area prefix LSA
  - Describes the destination outside the area but still in the AS
  - Summary is created for one area, which is flooded out in all other areas
  - Originated by an ABR
  - Only intra-area routes are advertised into the backbone
  - Link State ID simply serves to distinguish inter-area-prefix-LSAsoriginated by the same router
  - Link-local addresses must never be advertised in inter-area-prefix-LSAs

OSPF commands in IOS

- Entering router mode
  - [no] ipv6 router ospf <process ID>
- Enteringinterfacemode
  - [no] ipv6 ospf<process ID> area <areaID>
- Exec mode
  - [no] show ipv6 ospf[<process ID>]
  - clear ipv6 ospf[<process ID>]
- Configuring area range
  - [no] area <areaID> range <prefix>/<prefix length>
- Showing new LSA
  - show ipv6 ospf[<process ID>] database link
  - show ipv6 ospf[<process ID>] database prefix

Thanks to Philip Smith@Cisco for the command summary

OSPFv3 IOS

```
interface GigabitEthernet2/1
 description Local LAN
 ip address 10.0.0.1 255.255.255.0
 no ip directed-broadcast
 no ip proxy-arp
 duplex auto
 flowcontrol auto
 ipv6 address 2001:698:9:22::/64 eui-64
 ipv6 enable
 ipv6 ospf 8674 area 0
!
ipv6 router ospf 8674
 log-adjacency-changes
!
! The following are hosting interfaces.
 passive-interface GigabitEthernet2/2
 passive-interface GigabitEthernet2/2.10
 passive-interface GigabitEthernet2/2.20
 passive-interface GigabitEthernet2/2.30
```

OSPFv3 IOS-XR

```
interface POS1/1
 ipv6 address 2001:db8:FFFF:1::1/64
!
interface POS2/0
 ipv6 address 2001:db8:1:1::2/64
!
router ospfv3 ISP-BB
 address-family ipv6 unicast
 area 0
  interface POS1/1
 area 1
  interface POS2/0
```

OPSFv3 JunOS

```
interfaces {
  fe-3/0/0 {
    unit 0 {
      family inet6 {
        address 2001:db8:1:1::1/64;
      }
    }
  }
}
routing-options {
  router-id 10.1.1.104;
}
protocols {
  ospf3 {
    area 0.0.0.1 {
      interface fe-3/0/0.0 {
        metric 100;
      }
    }
  }
}
```

IS-IS

IS-IS

- RFC5308
- Two new TLVs defined
  - Reachability
  - Interface address
- Important to remember that IS-IS does not use IP for communication
  - If routers can form adjacencies they will!

ISIS - IOS

```
!
interface Ethernet1
 ip address 10.1.1.1 255.255.255.0
 ipv6 address 2001:0001::45c/64
 ip router isis
 ipv6 router isis
!
router isis
 address-familiy ipv6
  redistribute static
  exit-address-familiy
 net 42.0001.0000.0000.072c.00
 redistribute static
```

ISIS - IOS XR

```
interface Ethernet 1
 ip address 10.1.1.1 255.255.255.0
 ipv6 address 2001:db8:1::1/64
!
interface Ethernet 2
 ip address 10.2.1.1 255.255.255.0
 ipv6 address 2001:db8:2::1/64
!
router isis ISP-BB
 net 42.0001.0000.0000.072c.00
 address-family ipv4 unicast
  redistribute static
 address-family ipv6 unicast
  redistribute static
  single-topology
 interface Ethernet 1
  address-family ipv4 unicast
 interface Ethernet 2
  address family ipv6 unicast
```

ISIS - JunOS

```
interfaces {
  fe-3/0/0 {
    unit 0 {
      family inet {
        address 10.1.1.1/24;
      }
      family iso;
      family inet6 {
        address
2001:db8:1::1/64;
      }
    }
  }

(Continued -->)
```

```
fe-1/0/0 {
  unit 0 {
    family inet {
     address 10.2.1.1/24;
    }
    family iso;
    family inet6 {
     address 2001:db8:2::1/64;
    }
   }
 }
 lo0 {
  unit 0 {
    family inet {
     address 10.1.1.103/32;
    }
    family inet6;
(Continued…)
```

ISIS- JunOS

```
family iso {
    address 42.0001.0000.0000.072c.00;
   }
  }
 }
}
protocols {
 isis {
  export redistribute-static;
  interface fe-1/0/0.0;
  interface fe-3/0/0.0;
  interface lo0.0;
 }
}
policy-options {
 policy-statement redistribute-static
{
  term 1 {
   from protocol static;
   then accept;
  }
 }
}
```

BGP

BGP and IPv6

- BGP it self will behave exactly as for IPv4
  - IPv6 will make use of multiprotocol support
  - RFC2545 - *Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing*
- NEXT_HOP and NLRI are expressed as IPv6 addresses and prefix
- RFC2545
  - *Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing*

BGP and IPv6

- BGP uses it's own routing table
  - So no information shared between IPv4 and IPv6
- However, BGP uses TCP for transport
  - So the exchange of information can be done over either of IPv4 and IPv6 transport
  - Only share peering sessions if the topology is the same
- Same router-id concerns as before

eBGP configuration - IOS

```
router bgp 3220
 no synchronization
 bgp log-neighbor-changes
 neighbor ipv6-peer peer-group
 neighbor ipv6-peer advertisement-interval 0
 neighbor ipv6-peer soft-reconfiguration inbound
 neighbor 2001:670:87:3001:: remote-as 790
 no neighbor 2001:670:87:3001:: activate
 neighbor 2001:670:87:3001::A remote-as 1257
 no neighbor 2001:670:87:3001::A activate
 neighbor 2001:670:87:3001::12 remote-as 1654
 no neighbor 2001:670:87:3001::12 activate
 no auto-summary
 !
```

eBGP configuration - IOS

```
address-family ipv6
 neighbor ipv6-peer activate
 neighbor ipv6-peer route-map v6-peer-out out
 neighbor 2001:670:87:3001:: peer-group ipv6-peer
 neighbor 2001:670:87:3001::A peer-group ipv6-peer
 neighbor 2001:670:87:3001::12 peer-group ipv6-peer
 network 2001:670:87::/48
 exit-address-family
!
ipv6 route ::/0 Tunnel10
!
ipv6 prefix-list ipv6-prefix seq 5 permit 2001:670:87::/48
route-map v6-peer-out permit 10
 match ipv6 address prefix-list ipv6-prefix
!
```

iBGP configuration - IOS

```
router bgp 1
 neighbor ibgp-peer peer-group
 neighbor ibgp-peer version 4
 neighbor ibgp-peer remote-as 1
 neighbor ibgp-peer update-source Loopback10
 neighbor ibgp-peer send-community
 neighbor ibgp-peer prefix-filter announce out
 neighbor ibgp-peer soft-reconfiguration inbound
!
 neighbor 2001:db8::1 peer-group ibgp-peer
 neighbor 2001:db8::1 prefix-list kurtis in
```

Prefix-lists - IOS

```
router bgp 1
 neighbour 2001:db8:1::1 remote-as 2
 neighbour 2001:db8:1::1 prefix-list infilter in
 neighbour 2001:db8:1::1 prefix-list outfilter out
 neighbour 2001:db8:1::1 remote-as 3
 neighbour 2001:db8:1::1 prefix-list infilter in
 neighbour 2001:db8:1::1 prefix-list outfilter out
!
ipv6 prefix-list infilter deny 2001:db8:1::/48
ipv6 prefix-list infilter permit ::/0 le 32
ipv6 prefix-list outfilter permit 2001:db8:2::/48
```

BGP - Cisco IOS-XR

```
router bgp 1
 bgp router-id 10.1.1.4
 !
 address-family ipv6 unicast
 network 2001:db8:2::/48
 !
 neighbor 2001:db8:0:2::2
  remote-as 2
  route-policy all-v6-in in
  route-policy my-v6-out out
!
! all-v6-in <snipped>
route-policy my-v6-out
 if destination in my-v6 then pass
 endif
end-policy
!
prefix-set my-v6
 2001:db8:2::/48
end-set
!
```

BGP configuration - JunOS

```
interfaces {
  fe-3/0/0 {
    unit 0 {
      family inet6 {
        address 2001:db8:0:2::1/64;
      }
    }
  }
}
routing-options {
  rib inet6.0 {
    static {
      route 2001:db8:2::/48 discard;
    }
  }
  router-id 10.1.1.103;
}
(Continued -->)
```

```
protocols {
    bgp {
        local-as 1;
        group as2 {
            export export-static;
            peer-as 2;
            neighbor
2001:db8:0:2::2;
        }
    }
}
policy-options {
    policy-statement export-static {
        term 1 {
            from protocol static;
            then accept;
        }
    }
}
```

BGP - IOS IPv4 and IPv6

```
router bgp 10
 no bgp default ipv4-unicast
 neighbor 2001:db8:1:1019::1 remote-as 20
 neighbor 172.16.1.2 remote-as 30
!
 address-family ipv4
 neighbor 172.16.1.2 activate
 neighbor 172.16.1.2 prefix-list ipv4-ebgp in
 neighbor 172.16.1.2 prefix-list v4out out
 network 172.16.0.0
 exit-address-family
!
 address-family ipv6
 neighbor 2001:db8:1:1019::1 activate
 neighbor 2001:db8:1:1019::1 prefix-list ipv6-ebgp in
 neighbor 2001:db8:1:1019::1 prefix-list v6out out
 network 2001:db8::/32
 exit-address-family
!
! Continued -->
```

BGP - IOS IPv4 and IPv6

```
ip prefix-list ipv4-ebgp permit 0.0.0.0/0 le 32
!
ip prefix-list v4out permit 172.16.0.0/16
!
ipv6 prefix-list ipv6-ebgp permit ::/0 le 128
!
ipv6 prefix-list v6out permit 2001:db8::/32!
```

BGP - IOS-XR IPv4 and IPV6

```
router bgp 10
 bgp router-id 10.1.1.4
 !
 address-family ipv4 unicast
  network 172.16.0.0
 !
 address-family ipv6 unicast
  network 2001:db8::/32
 !
 neighbor 2001:db8:1:1019::1
  remote-as 20
  address-family ipv6 unicast
   route-policy ipv6-ebgp in
   route-policy v6out out
 !
 neighbor 172.16.1.2
  remote-as 30
  address-family ipv4 unicast
   route-policy ipv4-ebgp in
   route-policy v4out out
! Continued -->
```

```
route-policy ipv6-ebgp
 if destination in full-v6
then
pass
 endif
end-policy
!
prefix-set full-v6
 ::/0 le 128
!
route-policy v6out
 if destination in v6out then
pass
 endif
end-policy
!
prefix-set v6out
 2001:db8::/32
end-set
! Continued -->
```

BGP - IOS-XR IPv4 and IPV6

```
route-policy ipv4-ebgp
 if destination in full-v4 then
pass
 endif
end-policy
!
prefix-set full-v4
 0.0.0.0/0 le 32
end-set
!
route-policy v4out
 if destination in v4out then
pass
 endif
end-policy
!
prefix-set v4out
 172.16.0.0/16
end-set
```

BGP - JunOS IPv4 and IPV6

```
interfaces {
  fe-3/0/0 {
    unit 0 {
      family inet {
        address 10.1.1.1/24;
      }
      family inet6 {
       address 2001:db8:1::45c/64;
      }
    }
  }
}
routing-options {
  rib inet6.0 {
    static {
      route 2001:db8::/32 discard;
    }
  }
  router-id 10.1.1.103;
}
protocols {
  bgp {
    local-as 10;
    group as20 {
      export export-static;
      peer-as 20;
      neighbor 10.1.1.2;
    }
    group as30 {
      export export-static;
      peer-as 30;
      neighbor 2001:db8:1:1019::1;
    }
  }
}
policy-options {
  policy-statement export-static {
    term 1 {
      from protocol static;
      then accept;
    }
  }
}
```

Thank you!