Introduction to IPv6

kurtis@inettech.se

Internet Technology Advisors

Agenda

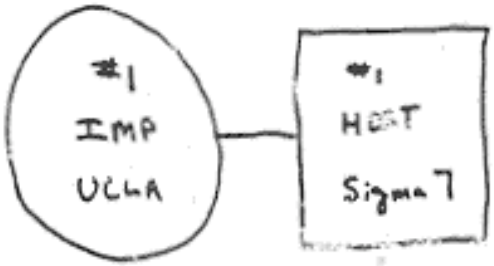
- Background
  - Background to IPv6
  - Which where the problems?
  - Which where the solutions?
  - Why was IPv6 picked as IPng?
- IPv6 design and addressing
  - What is an IPv6 address?
  - What “functions” does it contain?
  - Packet formats
  - Comparison between an IPv4 and an IPv6 packet
  - Address allocation
- Transition technologies
  - Teredo, 6to4, ISATAP, 6over4, etc..
- IPv6 Neighbour discovery
- IPv6 Stateless address autoconfiguration

Agenda cont.

- DHCPv6
- Mobile IPv6
- Routing and network design
  - Unicast, Multicast and anycast
  - Interdomain routing
  - Intradomain routing
  - Autoconfiguraiton
  - Built in security
  - QoS
- IPv6 and DNS
  - To think about
  - Configuration
- Applications
  - How are they affected?
  - What applications exist?
  - What do I need to do to port my application?
- Standardization
- Is this good?

Background

The Internet emerges

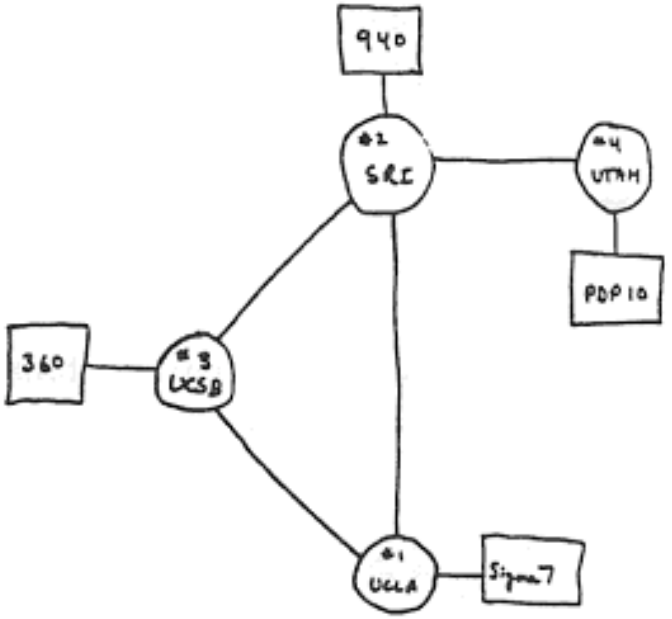


THE ARPA NETWORK

SEPT 1969

1 NODE

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



THE ARPA NETWORK

DEC 1969

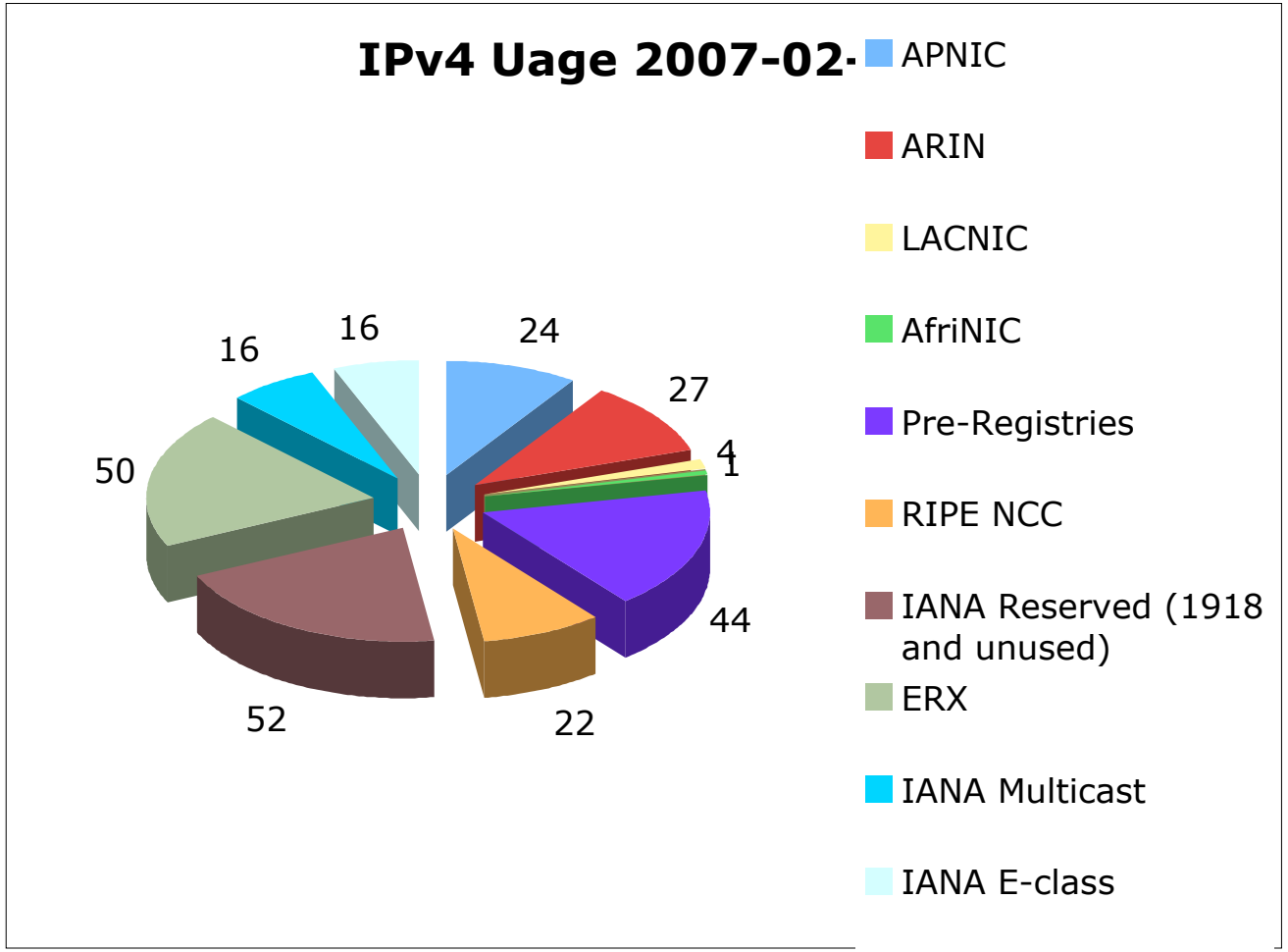
4 NODES

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

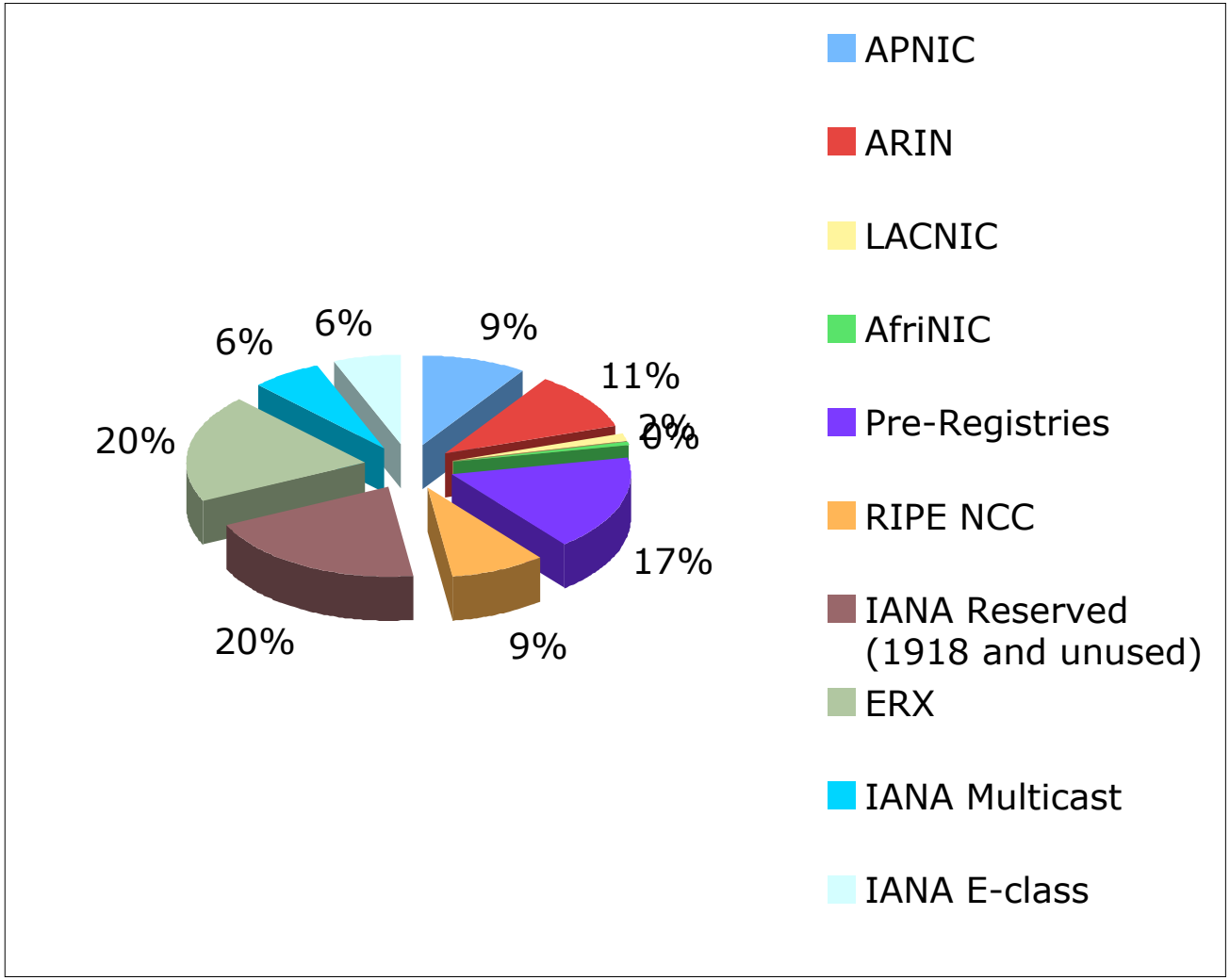
The background till IPv6

- 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

The background of IPv6



The background of IPv6

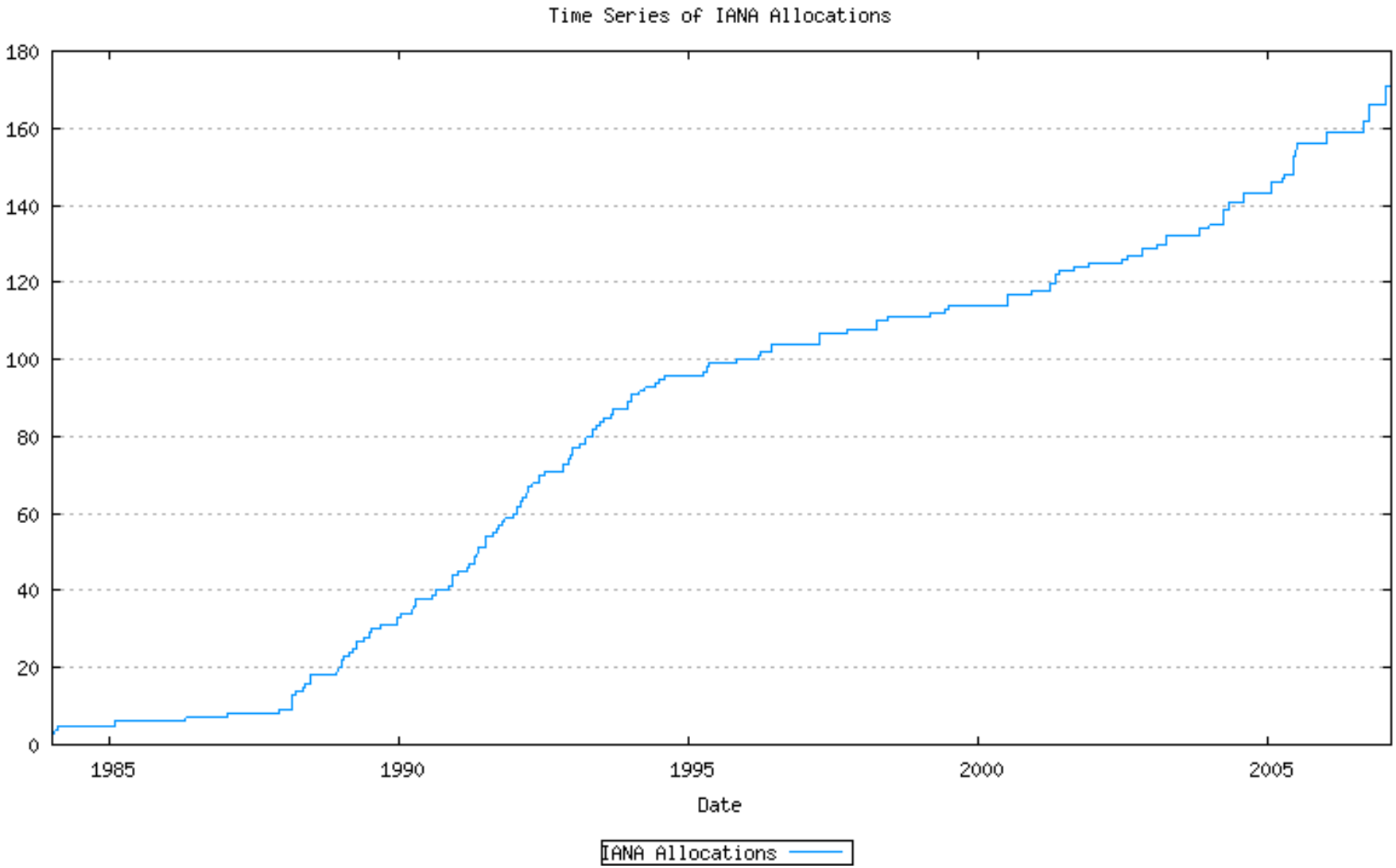


IPv4 Prognosis

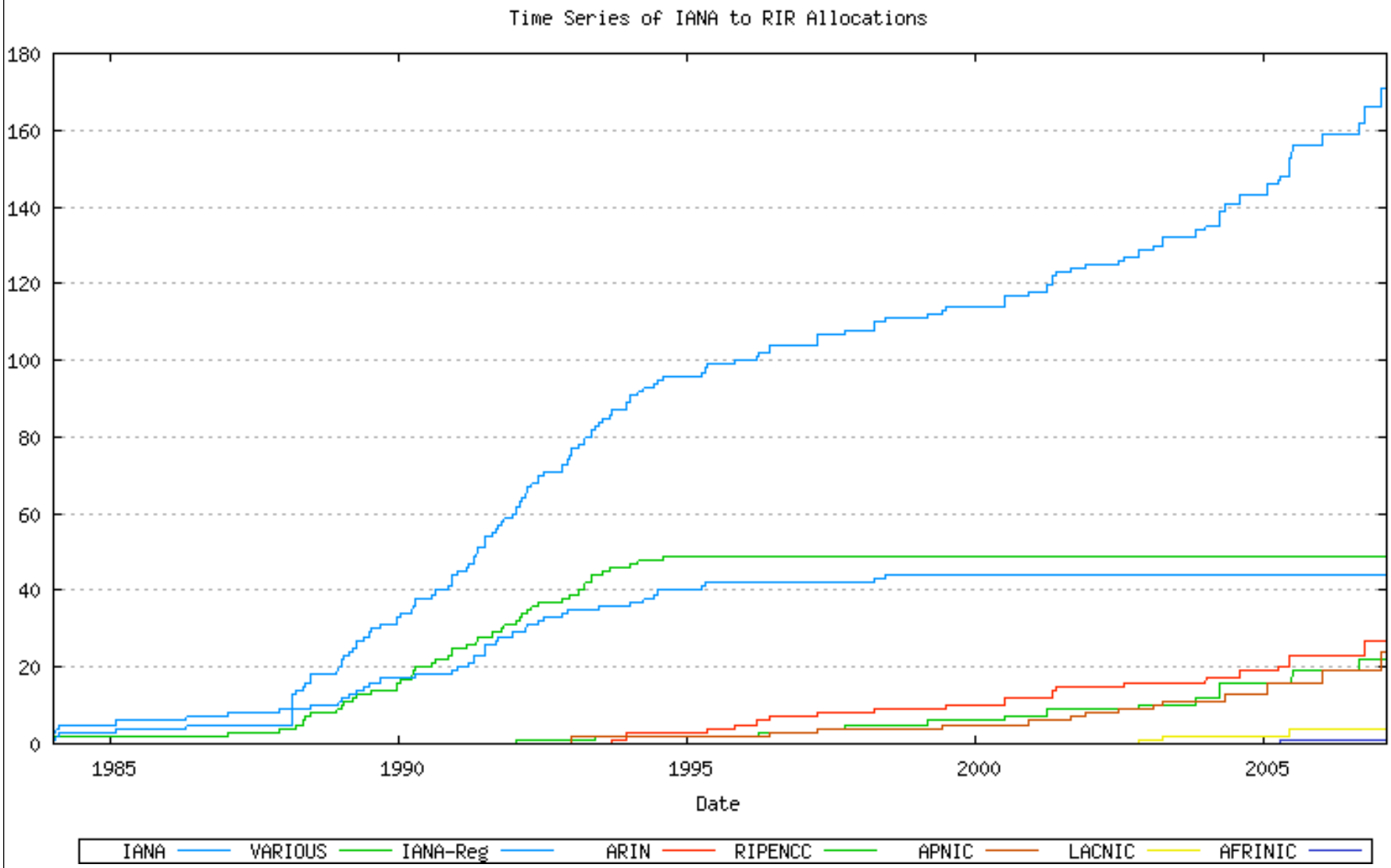
- How much IPv4 space do we have?
- When will we run out?
- Source of controvercy over how we calculate...

- Following data based on Geoff Hustons web-site
  - http://bgp.potraoo.net

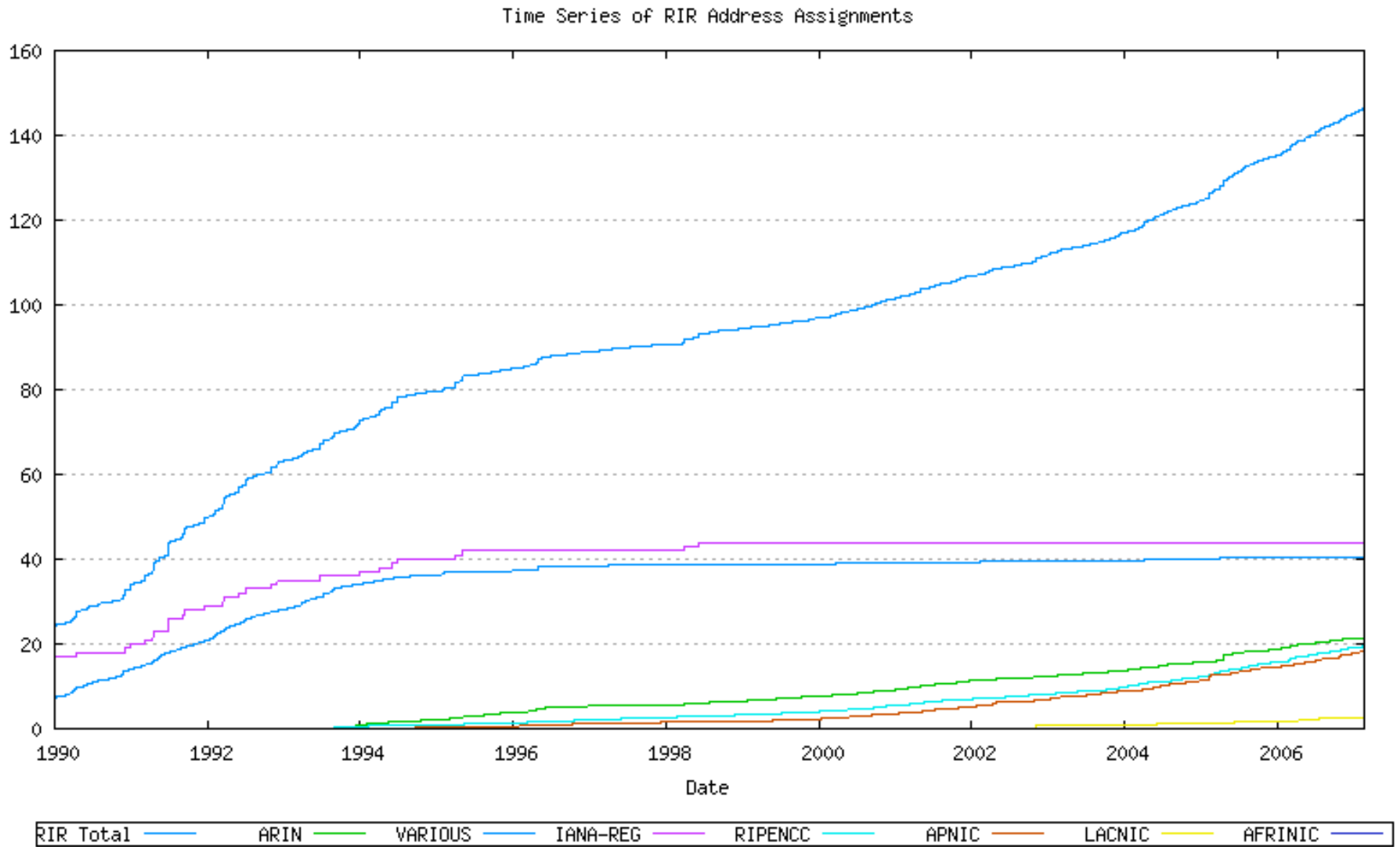
Allocated addresses - IANA



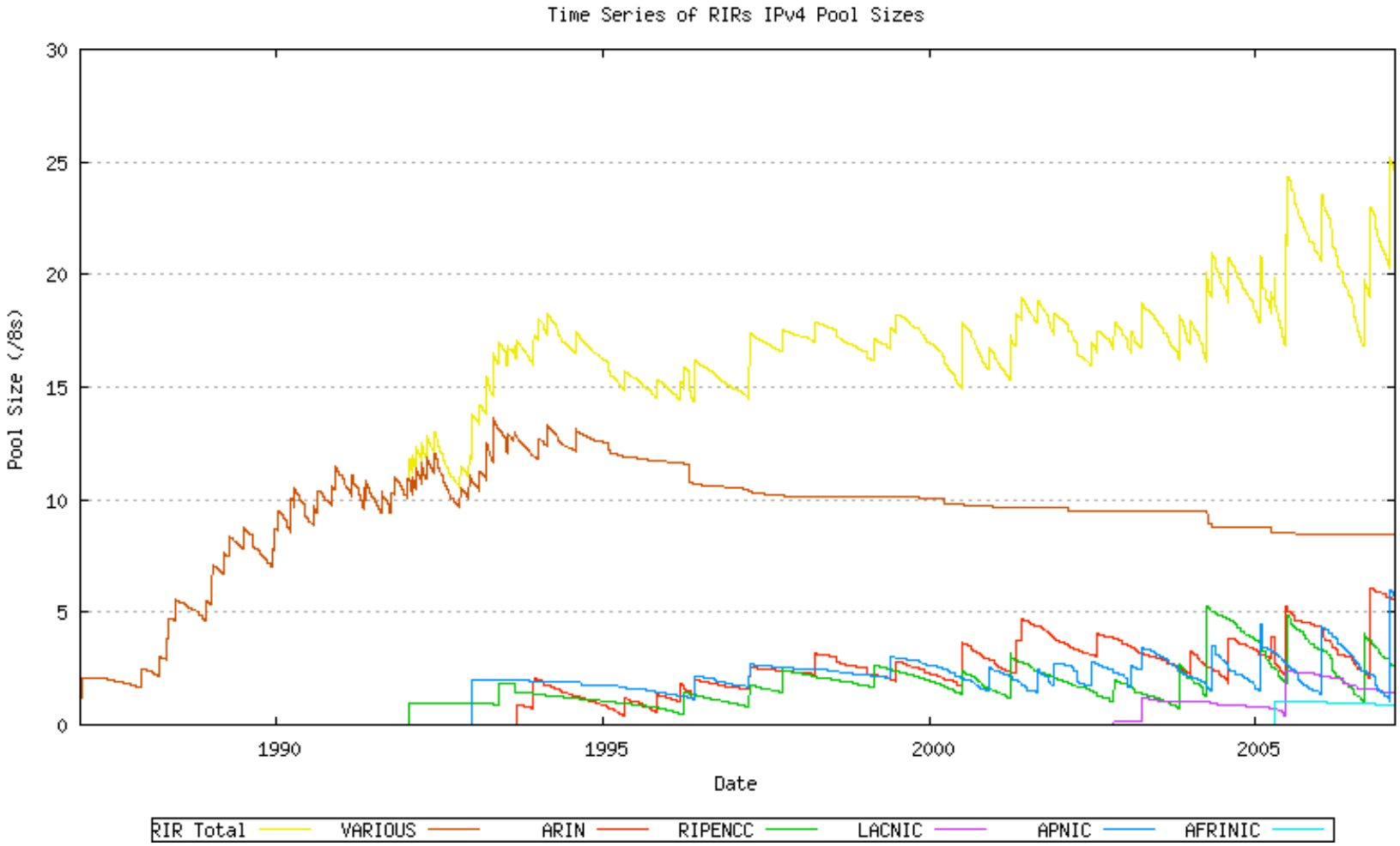
Allocated addresses - IANA to RIRs



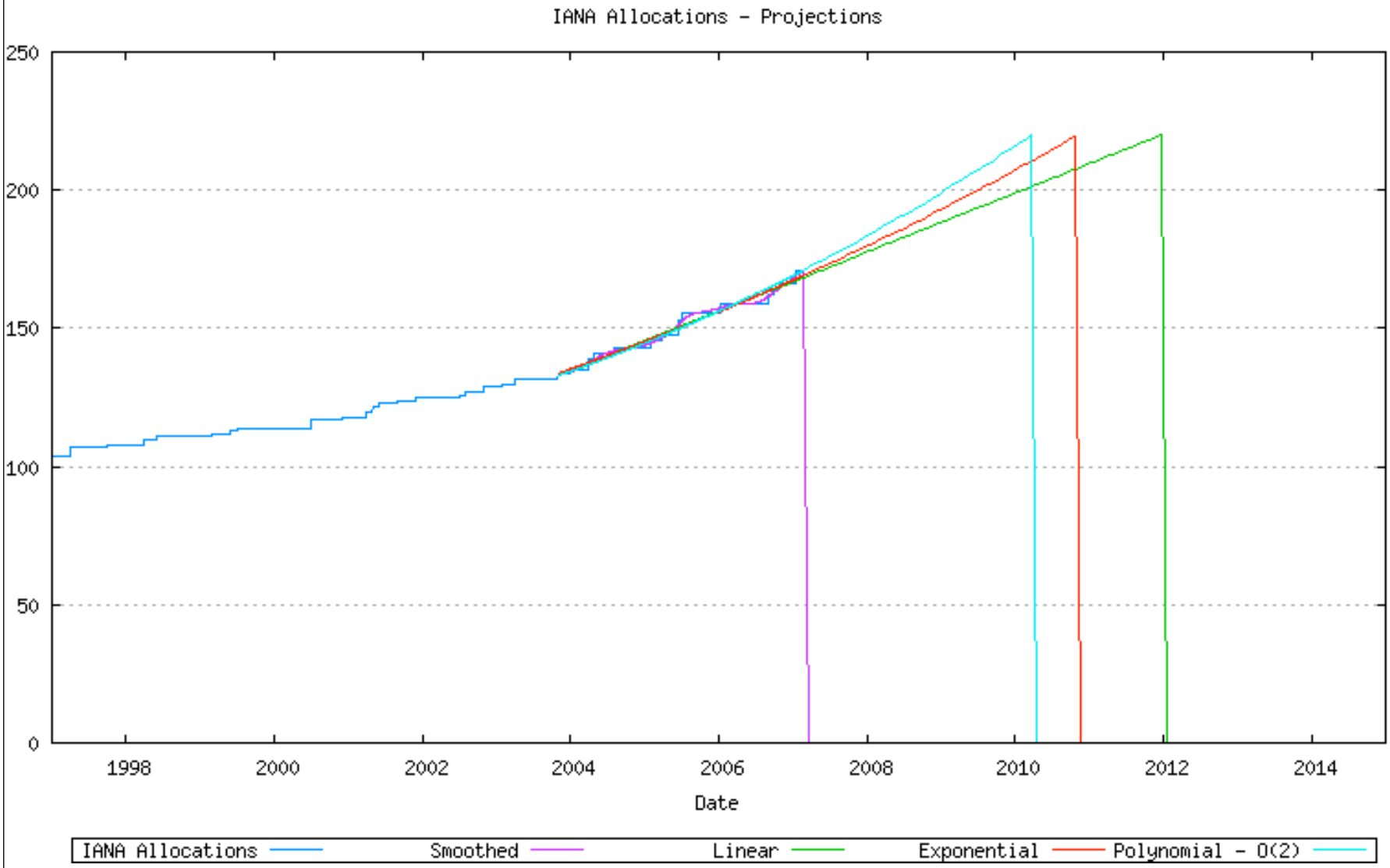
Assigned addresses - RIRs



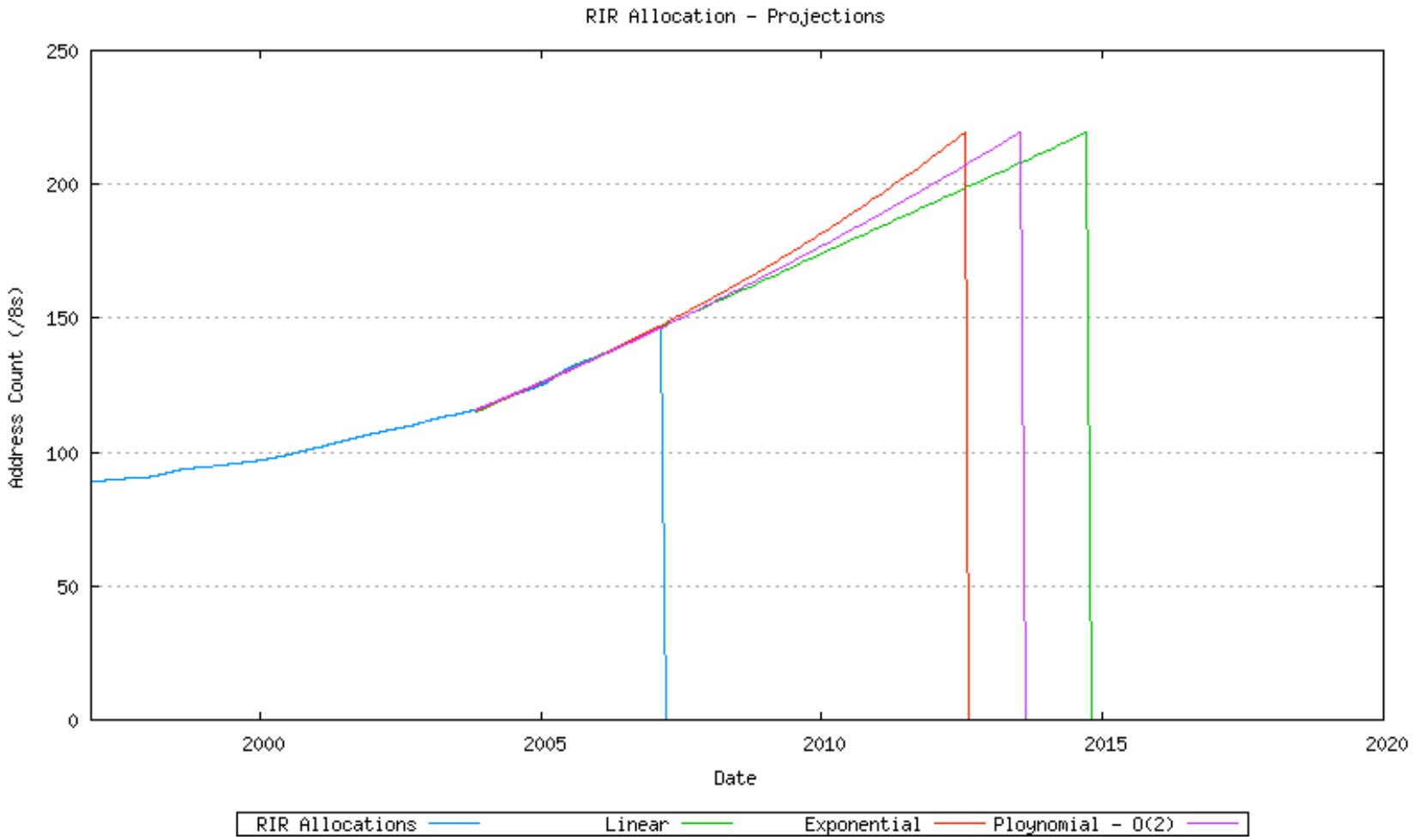
RIR Pool sizes



Predictions - Based on IANA data



Predictions exponential growth



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

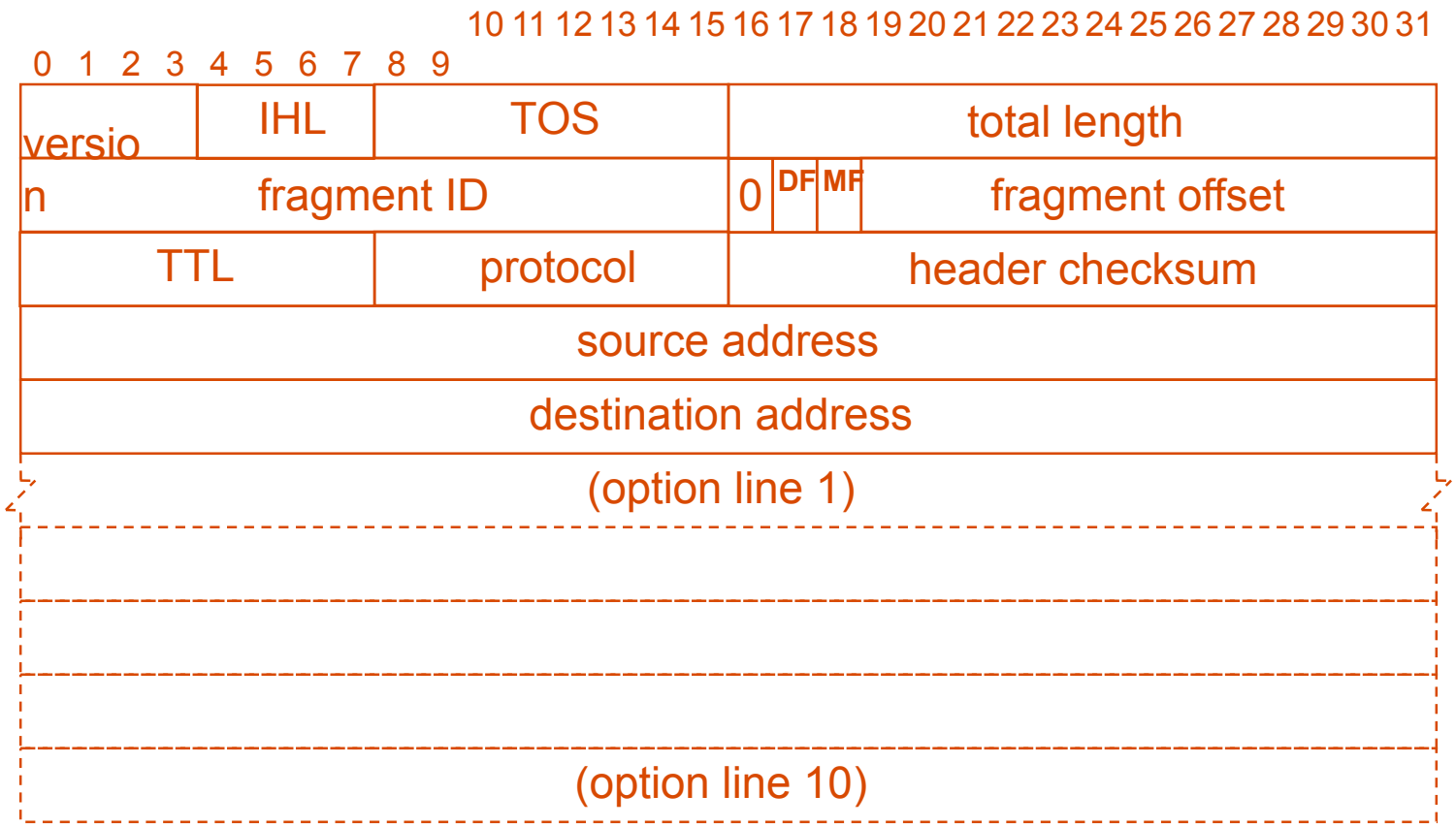
- TUBA
  - Would eventually be known as IPv9
- IAE
- Nimrod
- EIP
- PIP
  - Would eventually be known as IPv8
- SIP
- TP/IX
  - Would eventually be known as IPv7

The background of IPv6

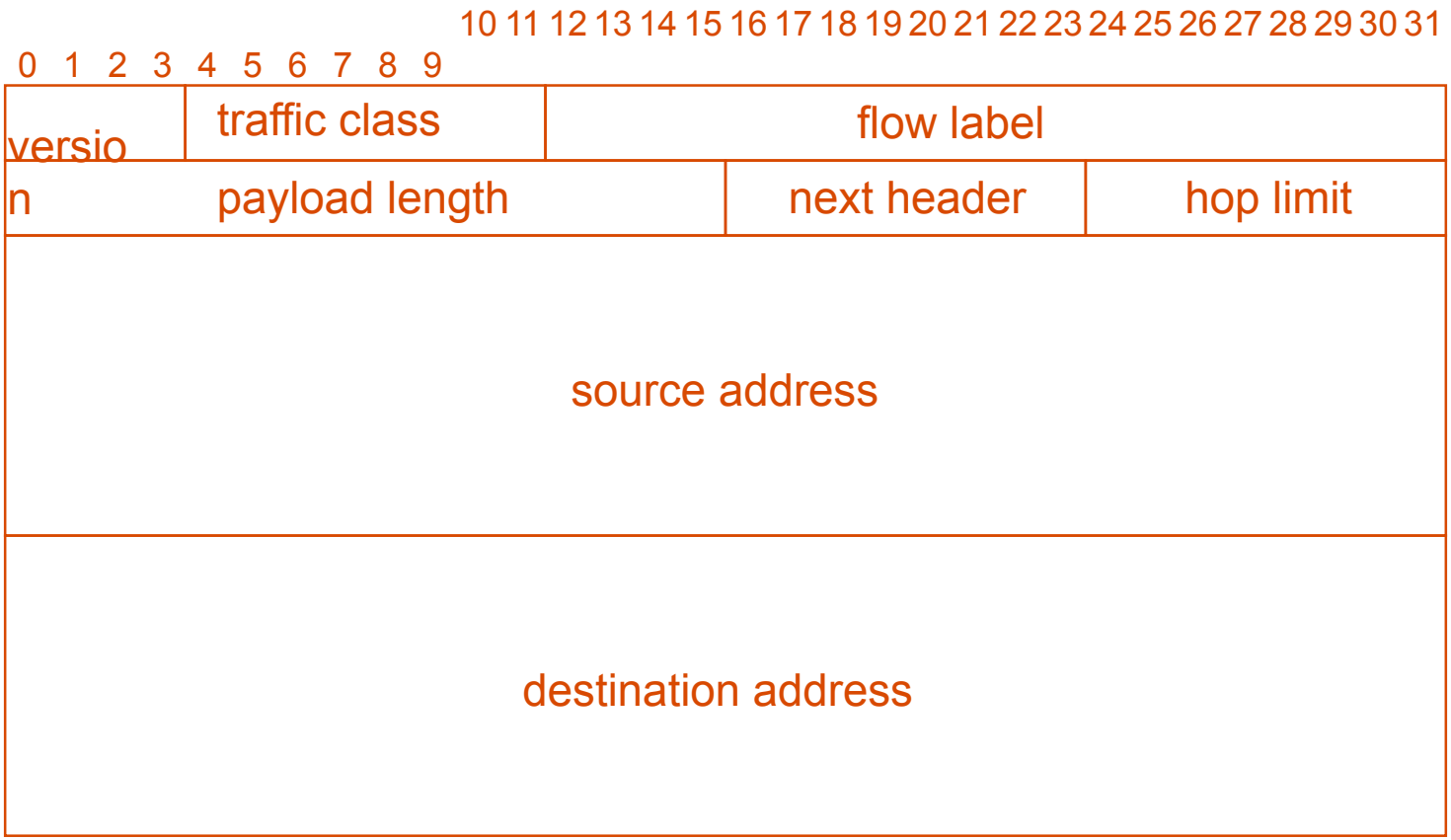
- The various proposals where developed and evolved through the merger of ideas
- Eventually three would remain
  - OSI
  - IPv7: TP/IX by Robert Ullman
    - Also waned to modify TCP
    - Was compatible with IPv4, CLNP and IPX
  - IP in IP: A mix of proposals, among others Steve Deering's SIP, PiP 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

IPv6 design and addressing

IPv4 header


10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 1 2 3 4 5 6 7 8 9
versio
n
IHL
TOS
total length
fragment ID
0
DF
MF
fragment offset
TTL
protocol
header checksum
source address
destination address
(option line 1)
(option line 10)

IPv6 header


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
versio
traffic class
flow label
n
payload length
next header
hop limit
source address
destination address

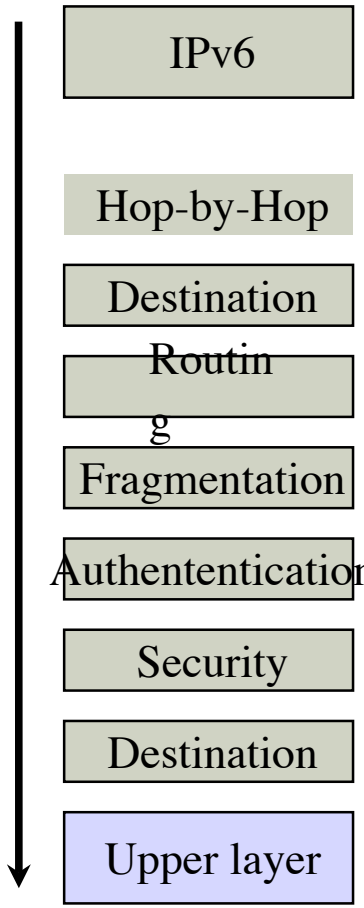
The IPv6 packet format - main differences

- 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



Processed by all nodes

Processed by all routers in ”routing”

List of routers that the packets need to pass

Processed by the receiver

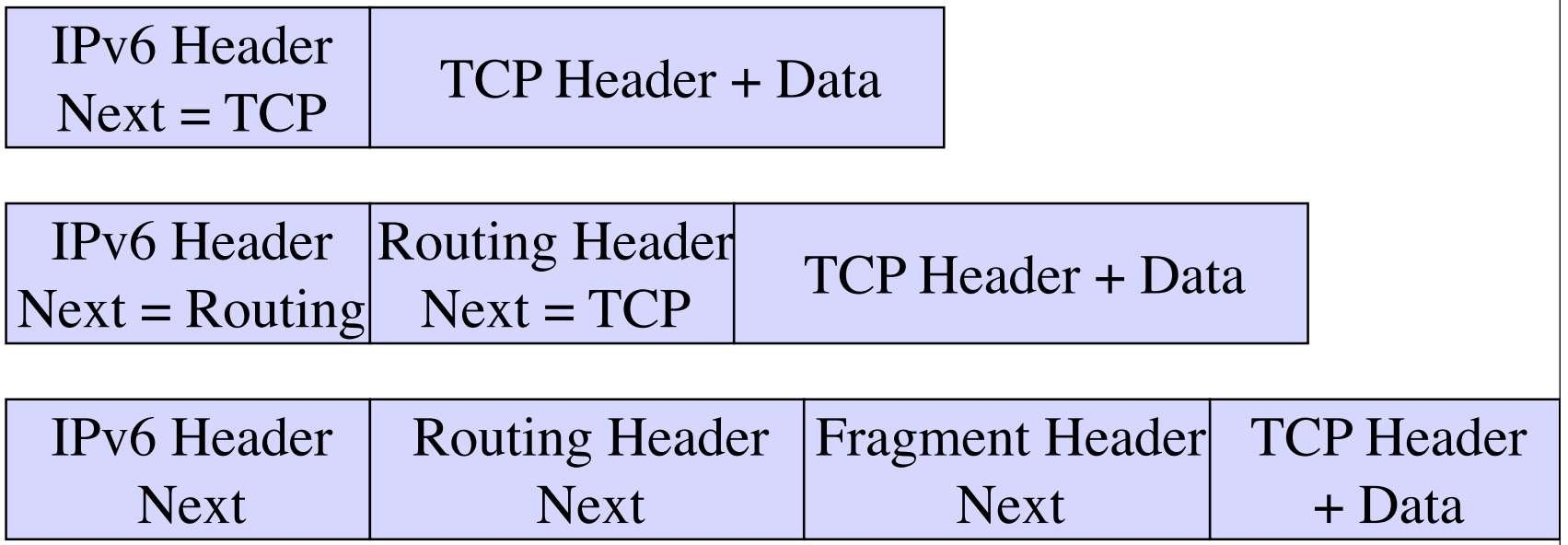
Processed after the packet has been reassembled

Encryption/Decryption of the rest of the packet

Processed by the receiver

IPv6 extension headers

- Extension headers are a list of headers

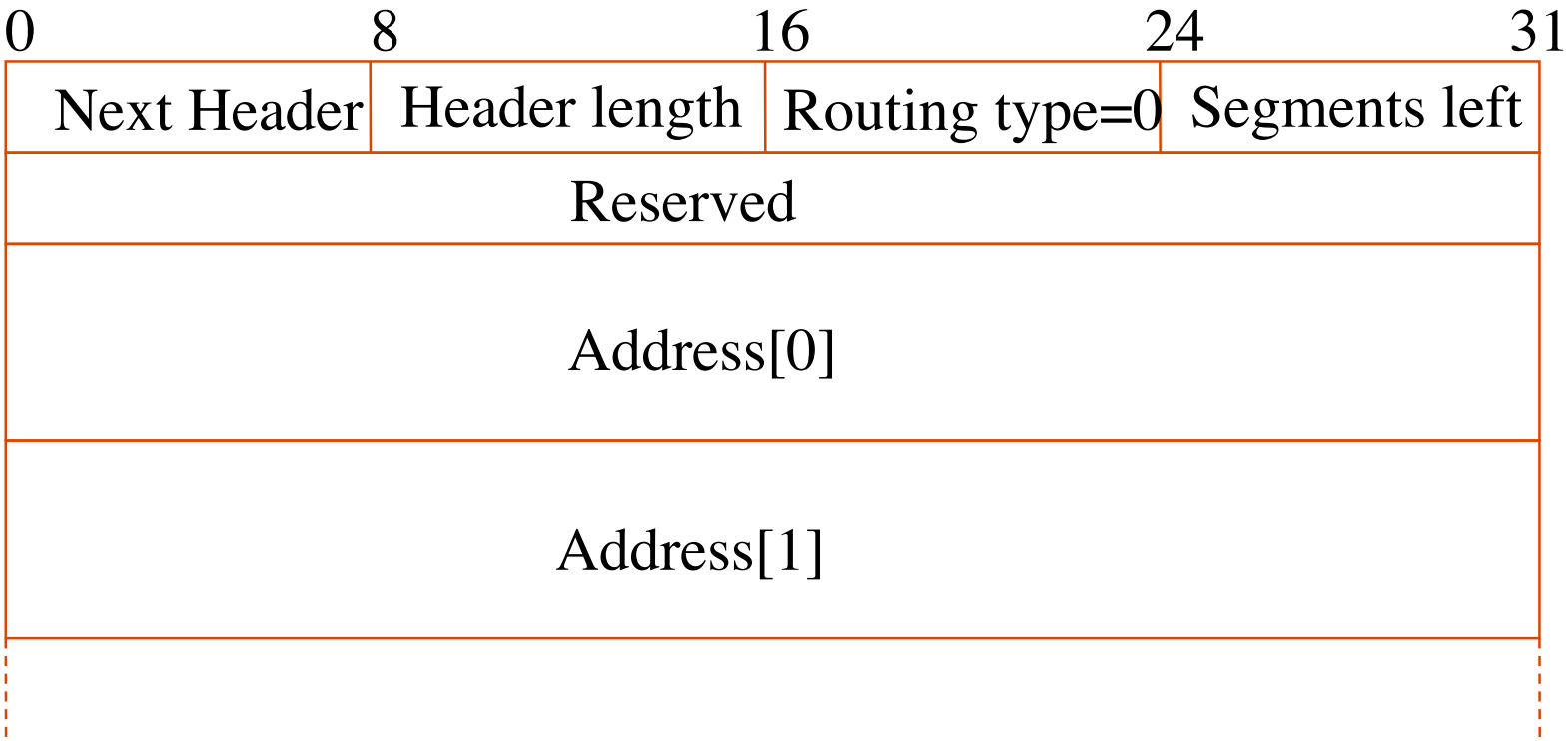


IPv6 extension headers

- 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)

IPv6 extension header - Routing

- Same function as source routing in IPv4



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

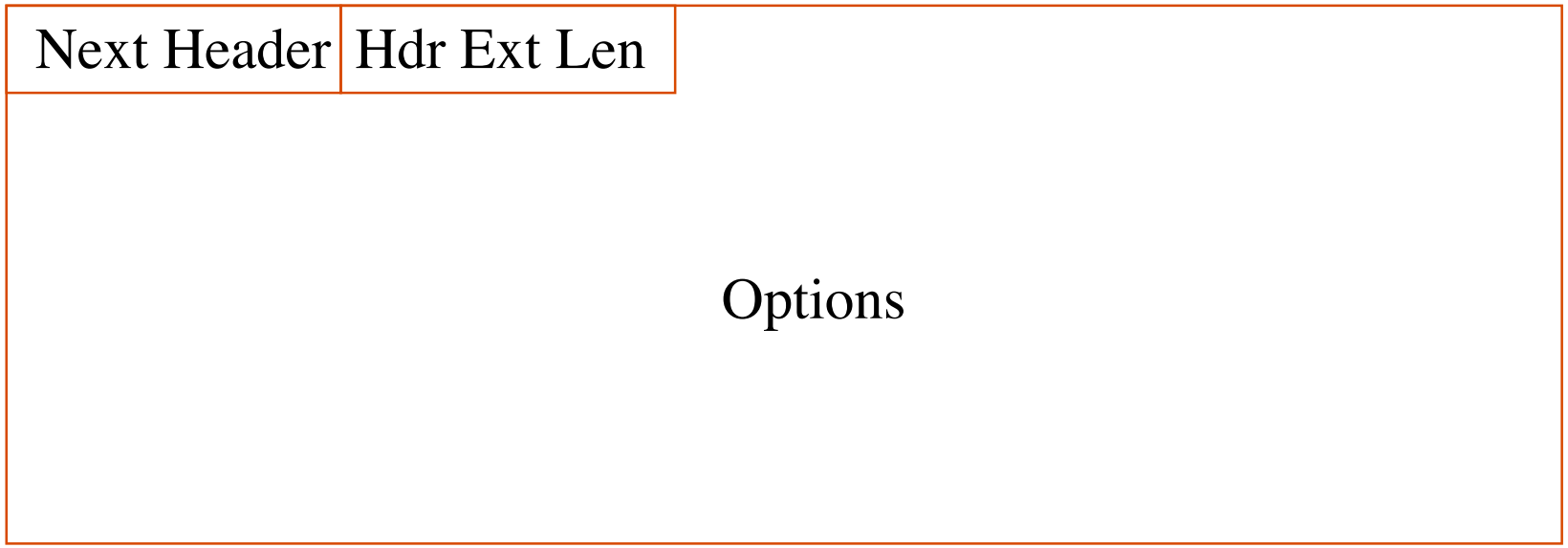
IPv6 extension headers - Fragmentation

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



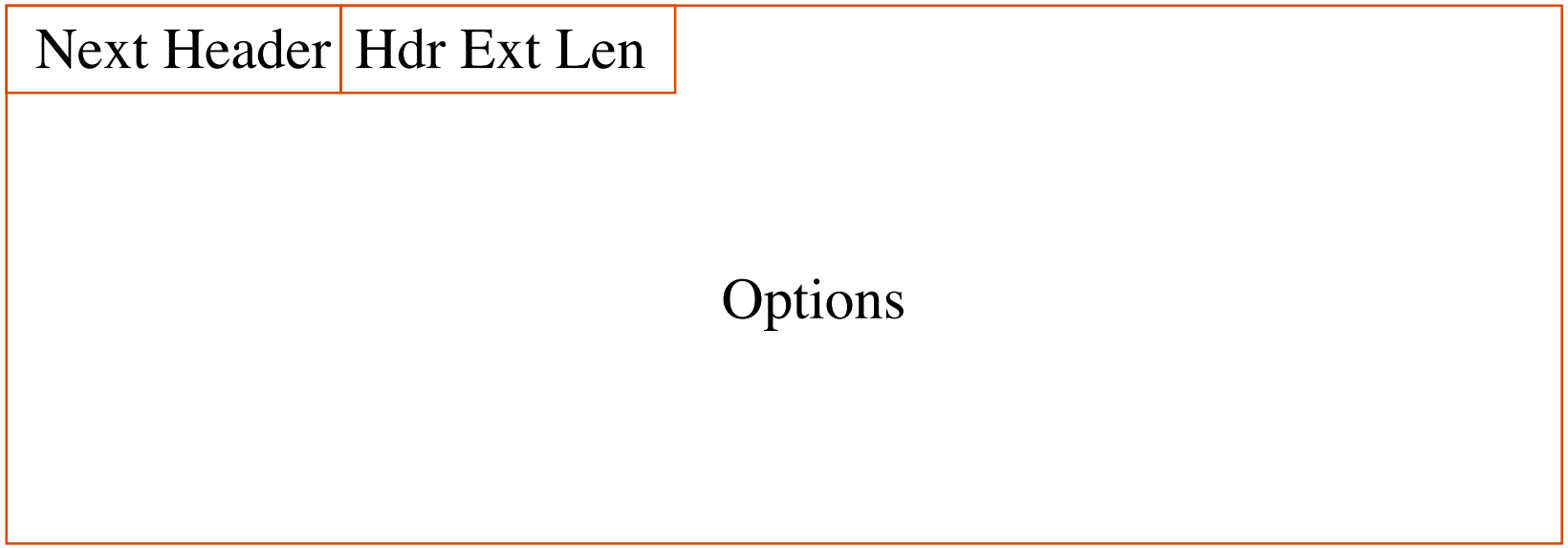
IPv6 extension headers - Destination option

- 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

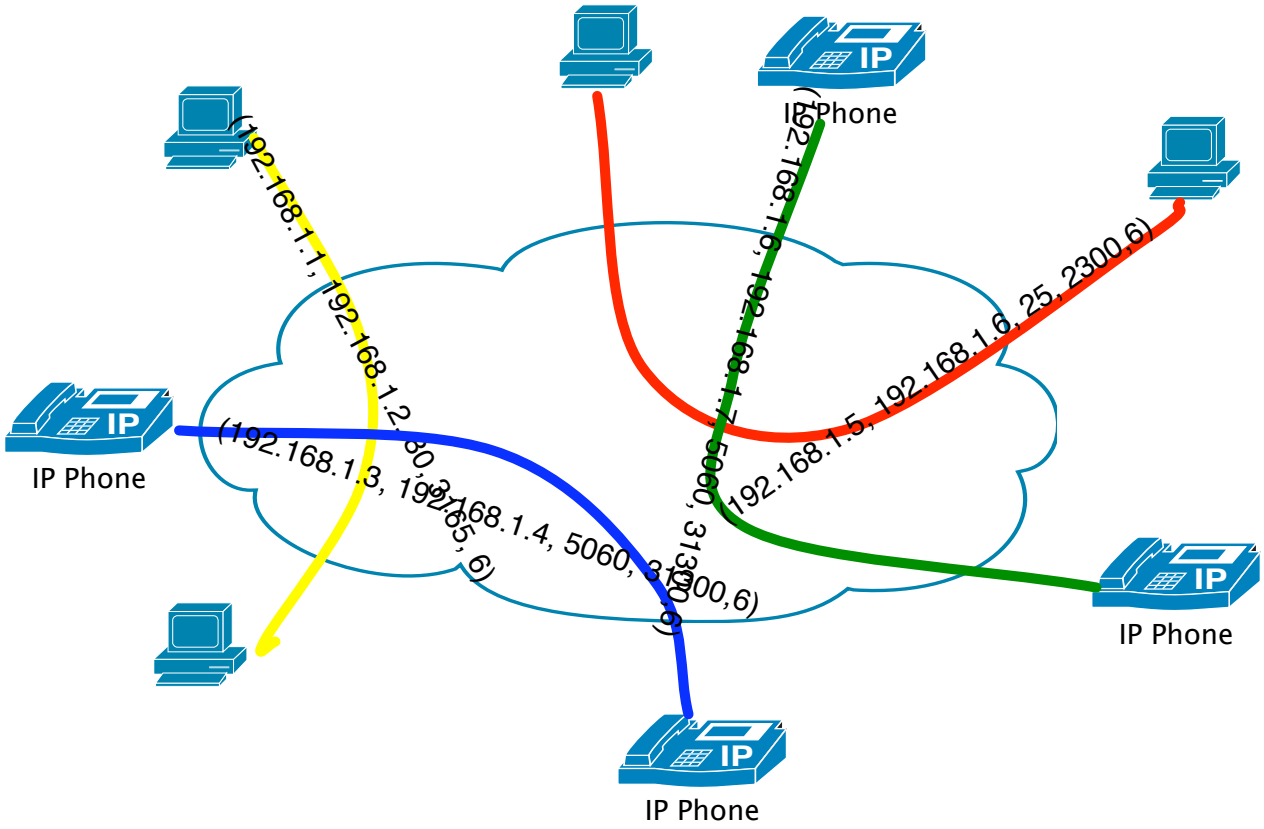
IPv6 extension headers - Hop-by-Hop



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, 3165, 6)
(192.168.1.3, 192.168.1.4, 5060, 3100, 6)
(192.168.1.5, 192.168.1.6, 25, 2300, 6)
(192.168.1.7, 192.168.1.8, 2000, 3100, 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

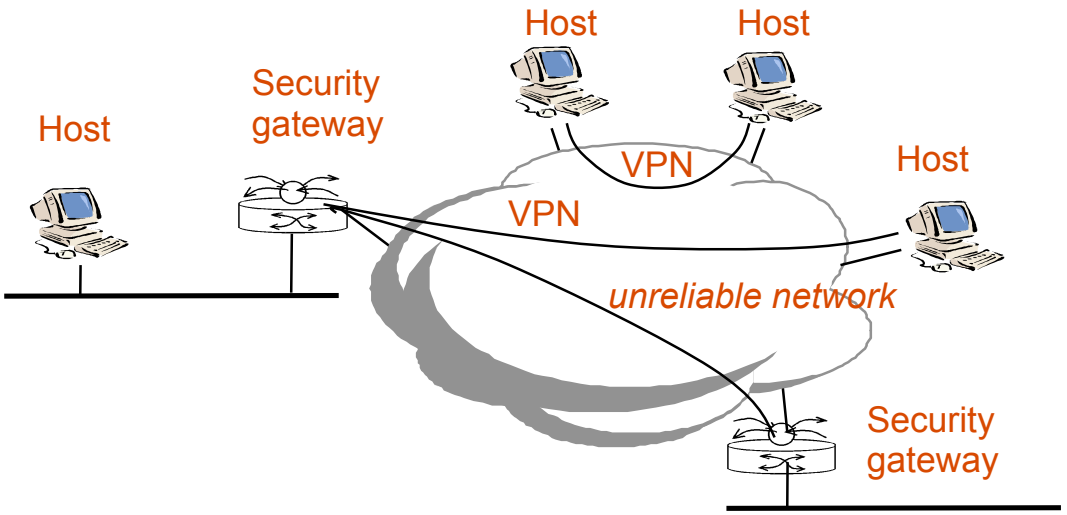
IPv6 and flows

- 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

IPv6 security

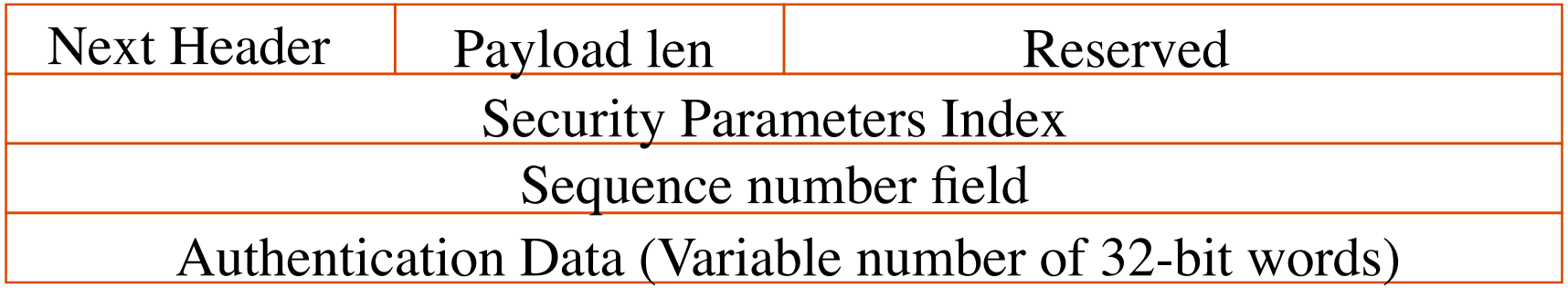


IPv6 Security - Authentication

- 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

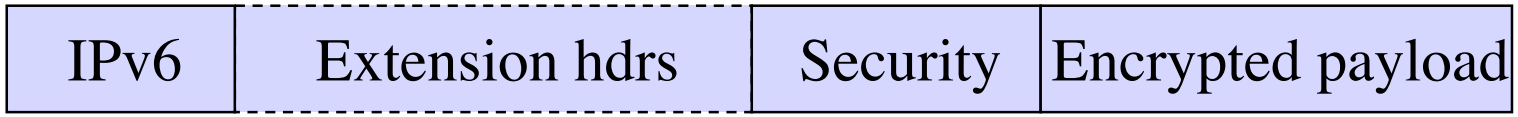
- Next Header
  - Next header coding
- Payload length
  - Length of the header and data
- Reserved
  - Reserved for future use. Must be set to zero
- Security Parameter Index (SPI)
  - Arbitrary 32-bit value that together with the destination IP and the security protocol uniquely identifies the security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

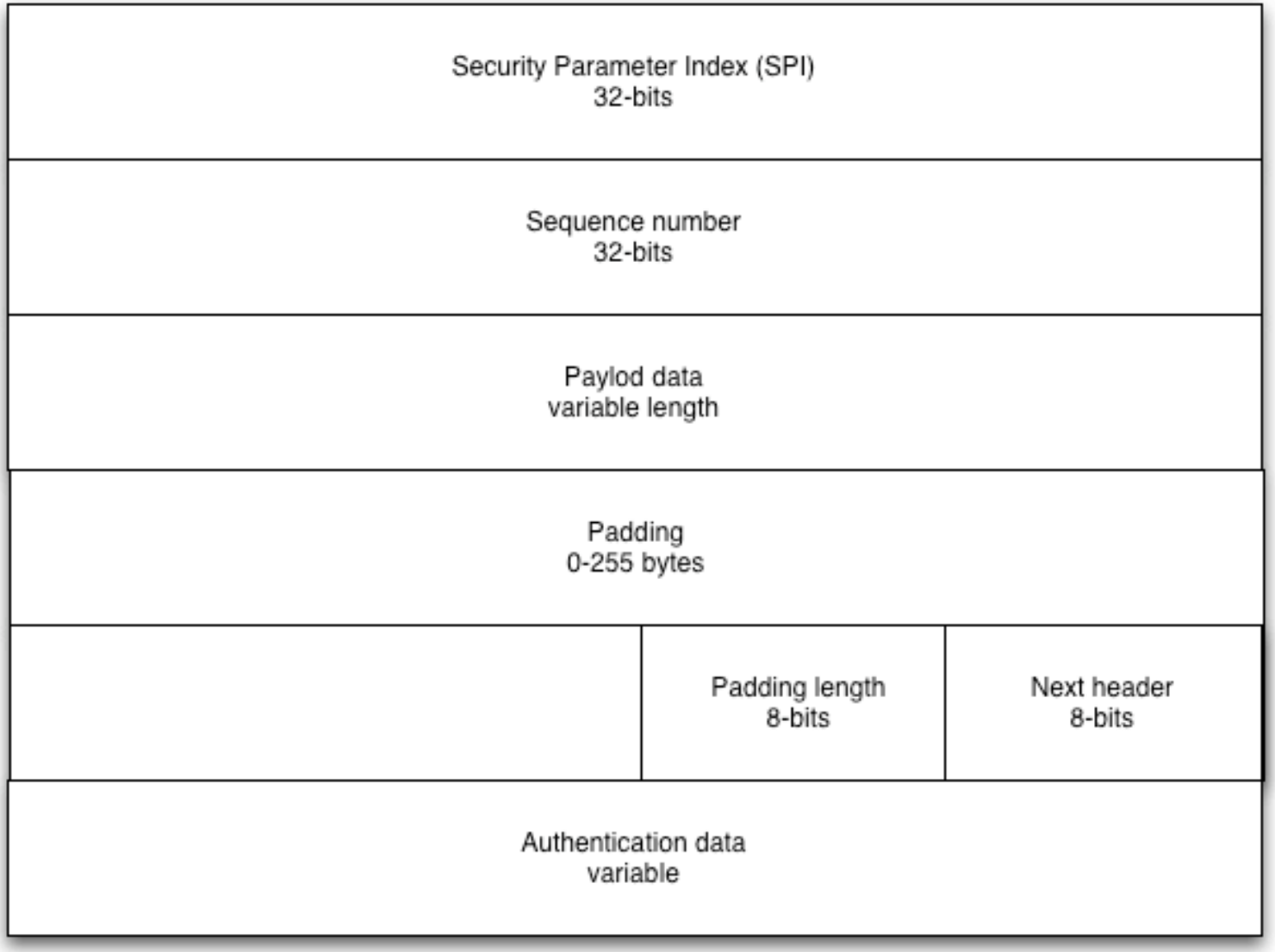
IPv6 security - Encryption

- 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

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 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

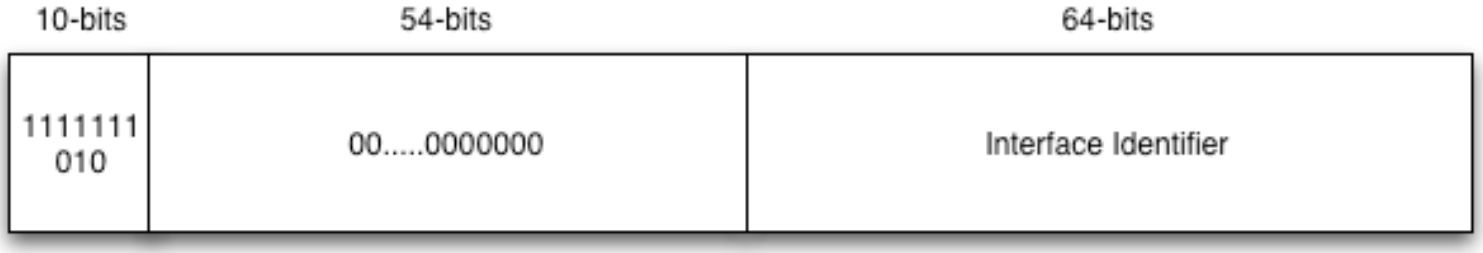
- Only IPv6 addresses of the “unspecified” type, “loopback address” and IPv6 addresses with embedded IPv4 addresses have the prefix 000
- Apart from above, IANA should only do delegation from the “001” prefix
- Leaves 85% of the IPv6 addresses for future use

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

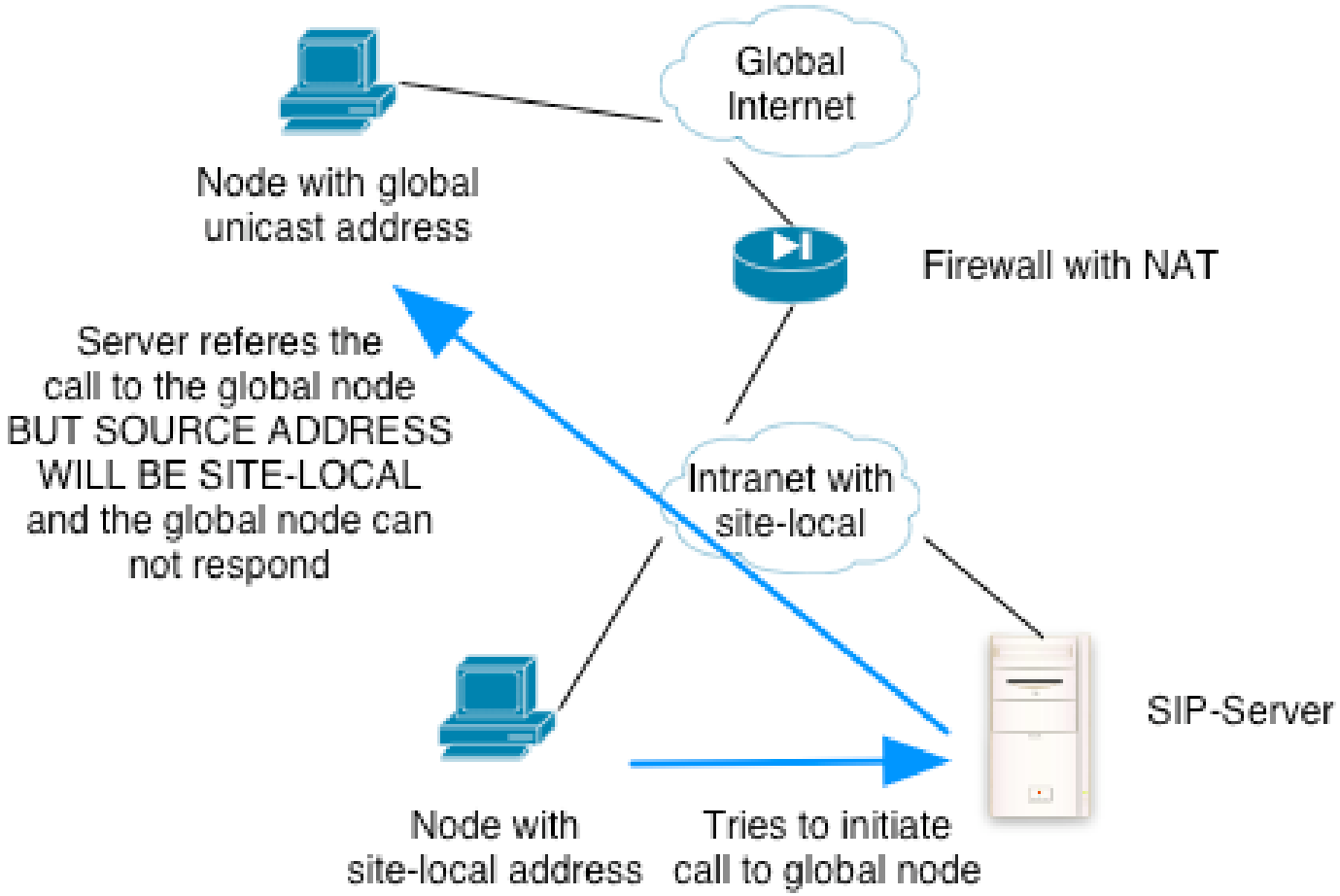
- For IPv6, three scoped where defined
- Global
  - As with all unicast addresses, these are globally unique addresses, assigned from a registry (e.g LIR)
- Link-local
  - Addresses that are unique on the link in question. These addresses are calculated and assigned by the interface itself
  - This means 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

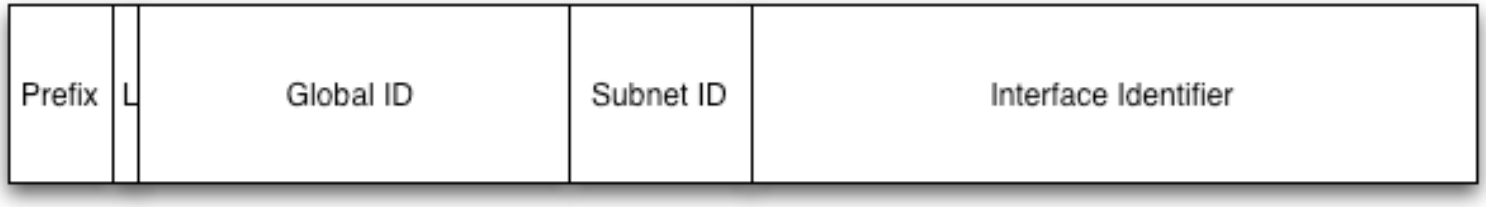
Site-local

- 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



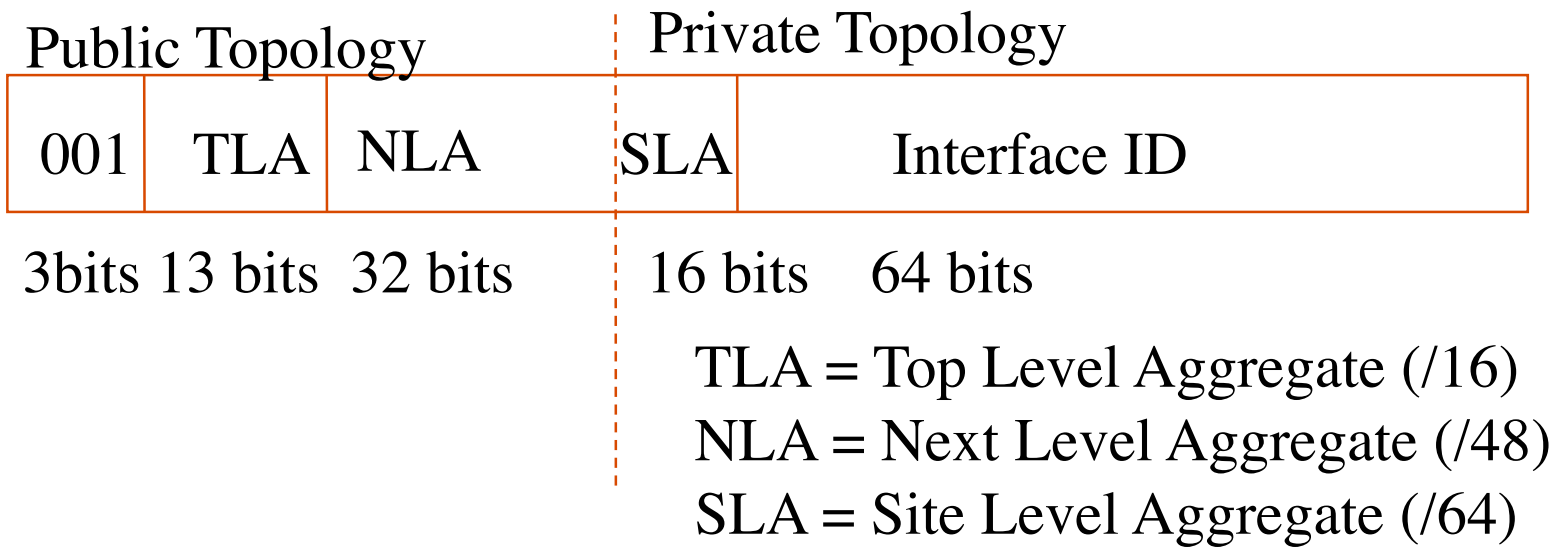
- Prefix
  - The globally unique prefix, proposed to be FC00::/7
- L
  - Locally/Centrally assigned prefix
- Global ID
  - The globally almost unique prefix that was generated
- Subnet ID
  - Subnet ULA block
- Interface identifier
  - EUI64 or RFC3401

ULAs

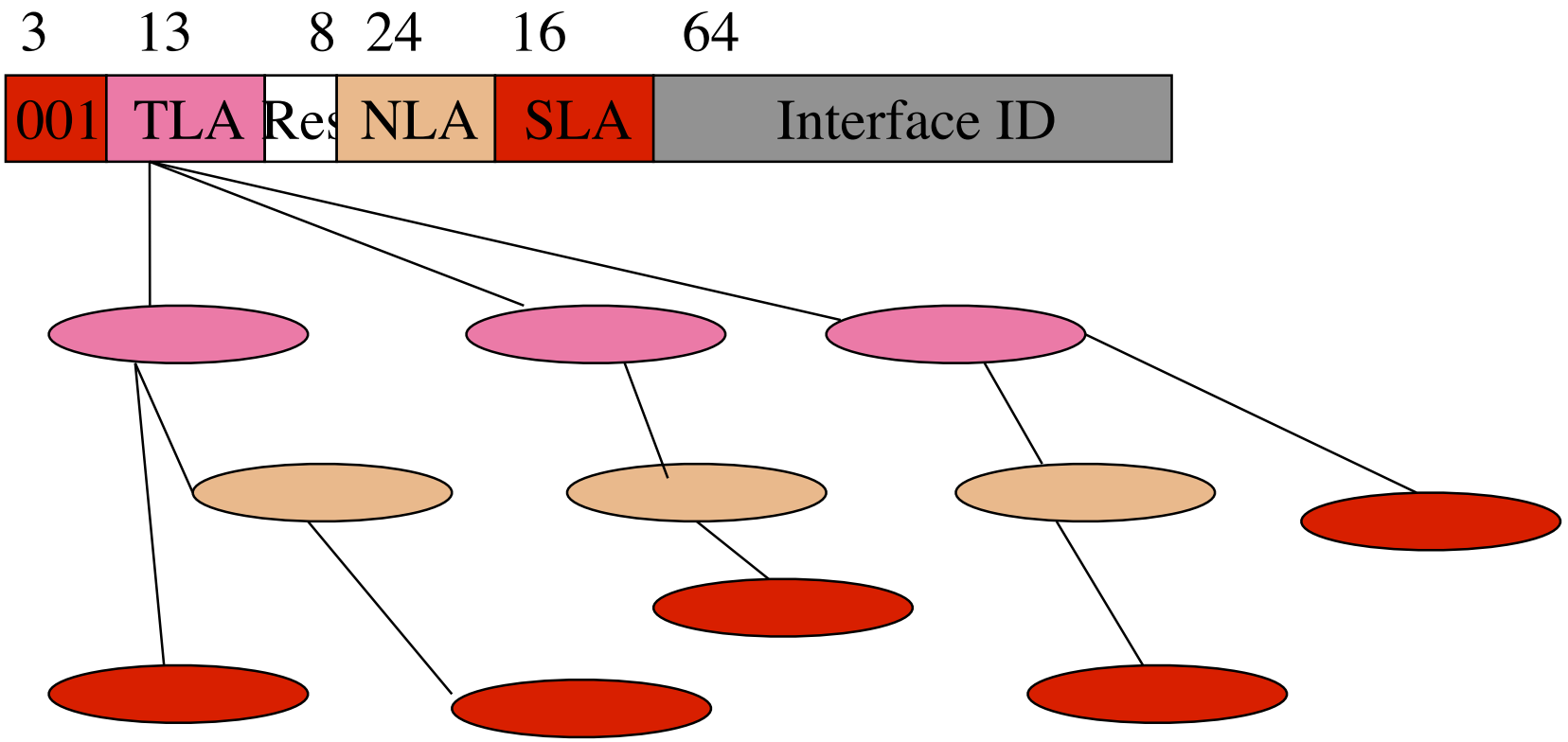
- The algorithm for generating the globally “unique” prefix is
  - Actual time through NTP
  - The local EUI64 address, alternatively anther local unique ID
  - Add the EUI64 address with the actual time to create a hash key
  - Calculate a SHA-1 has of the key
  - Use the last 40-bits of the hash as 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

- The original idea was to make IPv6 addresses “hierarchical”
  - I.e that different blocks where given to different ISPs
- The idea was that the ISPs where ordered in a hierarchy anyway



IPv6 addressing



IPv6 addressing

- In reality the Internet and the ISPs are not that hierarchical
- Smaller ISPs would never accept to be dependent on larger
- TLAs in reality came to be given to the RIRs
- TLA/NLA in principle are classfull addresses all over again
- It was decided to abandon the TLA/NLA notion
  - As for RFC3513 these do no longer exists

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”

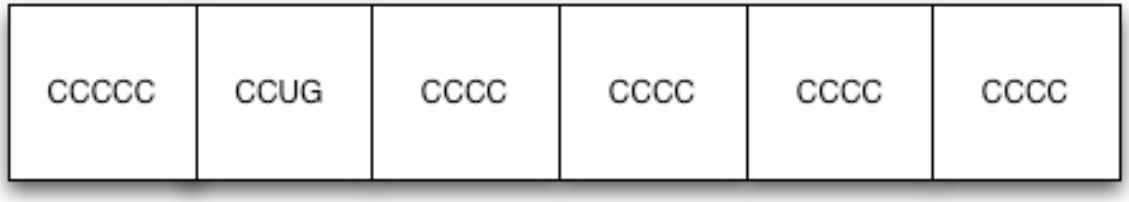
IPv6 Addressing - 6bone

- To get away from a tunnelled 6bone network, it was decided that
  - Since January 1th 2004, no new pTLAs are 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

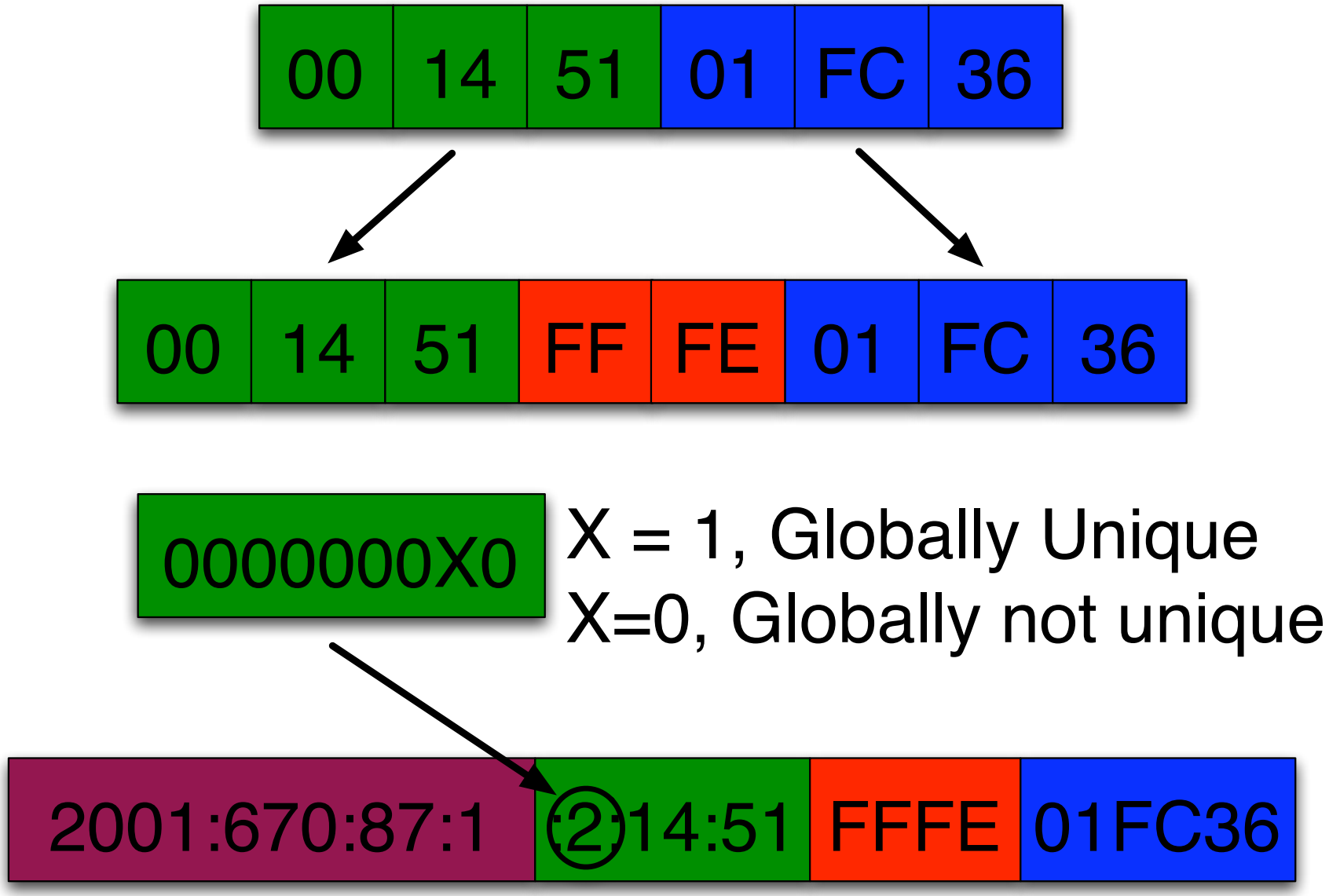
IPv6 Unicast addresses

- 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 starting 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

RFC3041

- 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

RFC3041

- 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 alloction policies

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 a LIR
  - Larger allocations can be made with reference to the HD ration with their current customers
  - HD ration is under discussion, and a change might come

HD Ratio

- H Ratio is defined in RFC1715

$$H = \frac{\log(\text{number of objects})}{\text{available bits}}$$

- Calculations on an address length of 64 was made and used as an argument for 128-bit IPv6 addresses
  - But we only use 64…
- HD Ratio is defined in RFC3194

$$HD = \frac{\log(\text{number of allocated objects})}{\log(\text{maximum number of allocatable objects})}$$

HD Ratio

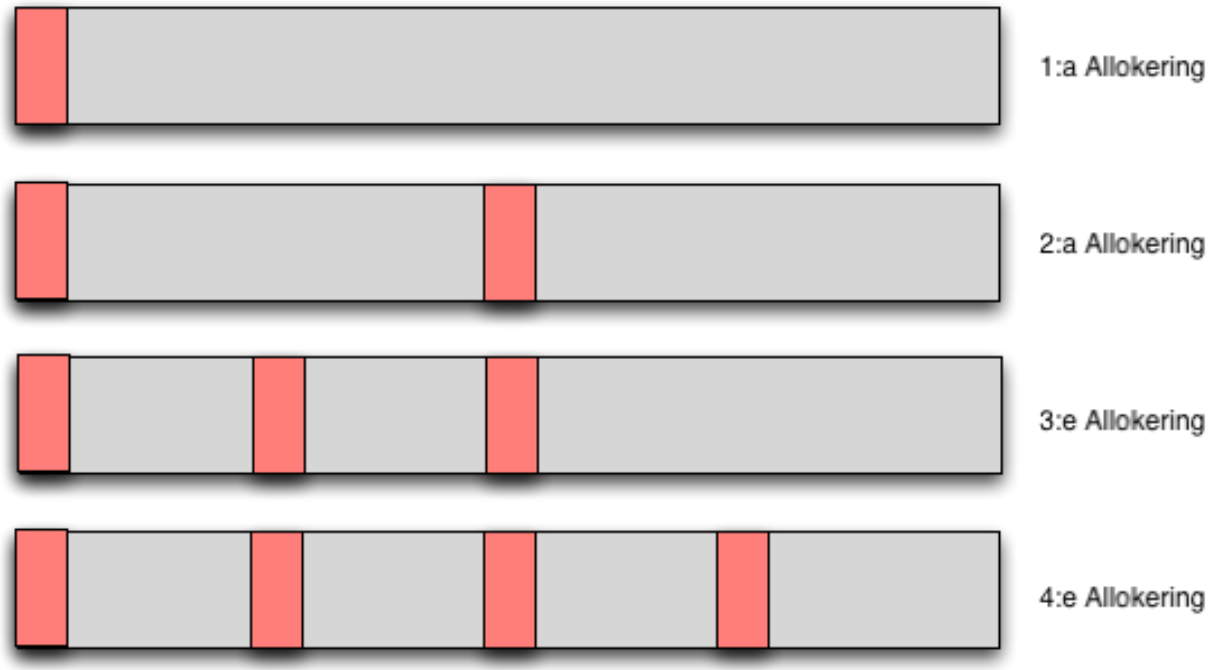
- A measurement of “usage” of address pools
  - Is built on the assumption that networks are hierarchical and allocations will have to follow that
  - There is a relationship that a given network of size X, have Y number of subnets
- Fits *fairly* well…
- …but not quite for the really large networks being built today
  - DSL, Cabel-TV etc
  - Ongoing discussions on changing this

Address Allocation Policy

- The most controversial issue is what blocks the RIRs should get from IANA
- Today the RIRs get blocks of /23 from IANA.
  - This leads to many allocations
  - Not optimally allocated
  - RIRs can not make RIR->LIR allocations optimally as the blocks are to small

Address allocation policy

- Proposal is that the allocation RIR->LIR is done using ‘binary chop’



Allocation from IANA

- Arguments currently being made for larger IANA allocations to the RIRs
  - Discussion is how large and is centring between /8 and /16
  - Most seems to be in favour of approx /12 based on studies by Geoff Houston

Allocations from IANA

- Larger blocks give a better effect of binary-chop
- If you then also weigh in previous allocation trends for a LIR in the algorithm
  - You can even further decrease the likelihood of a collision for the second allocation to the same LIR
  - Further helps to increase aggregation

Today's Address Allocation Policy

- Today IPv6 allocation is coordinated globally
  - Between all the RIRs
- 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

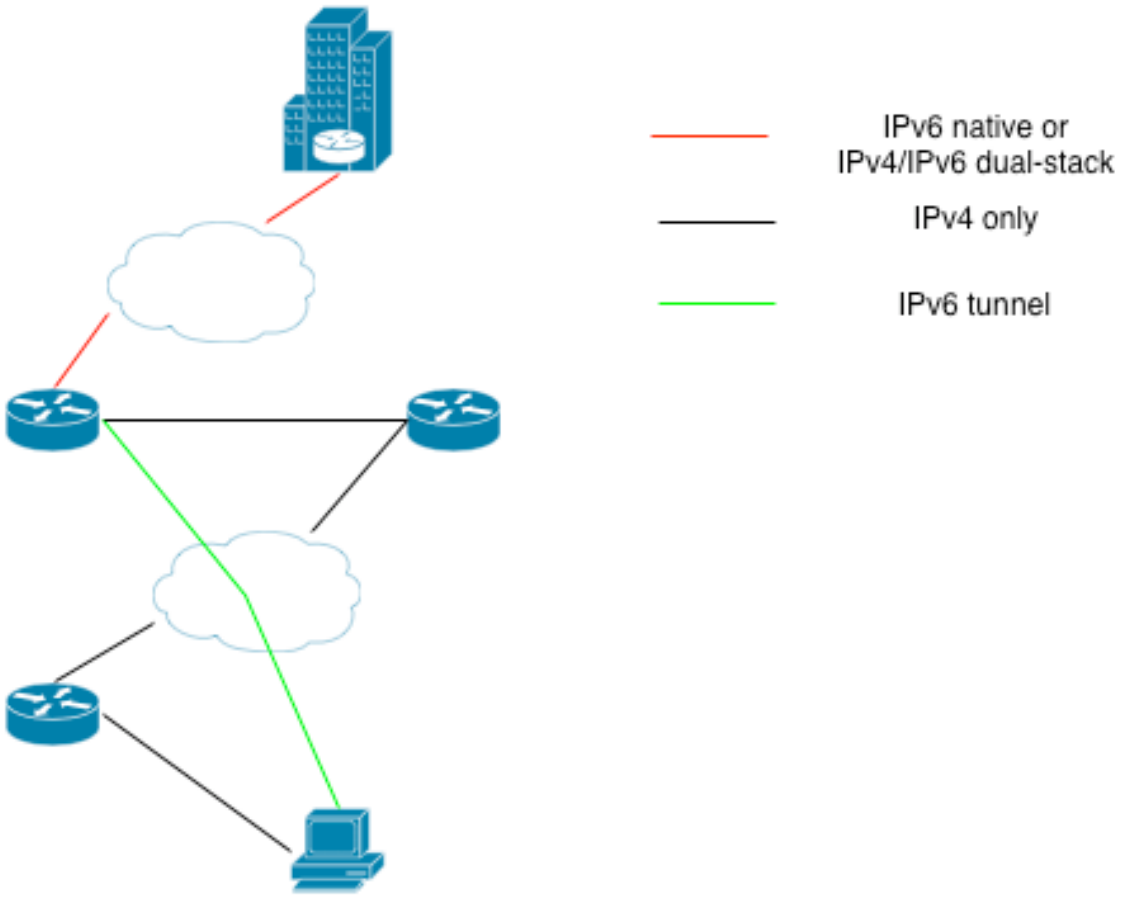
Transition technologies

- 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

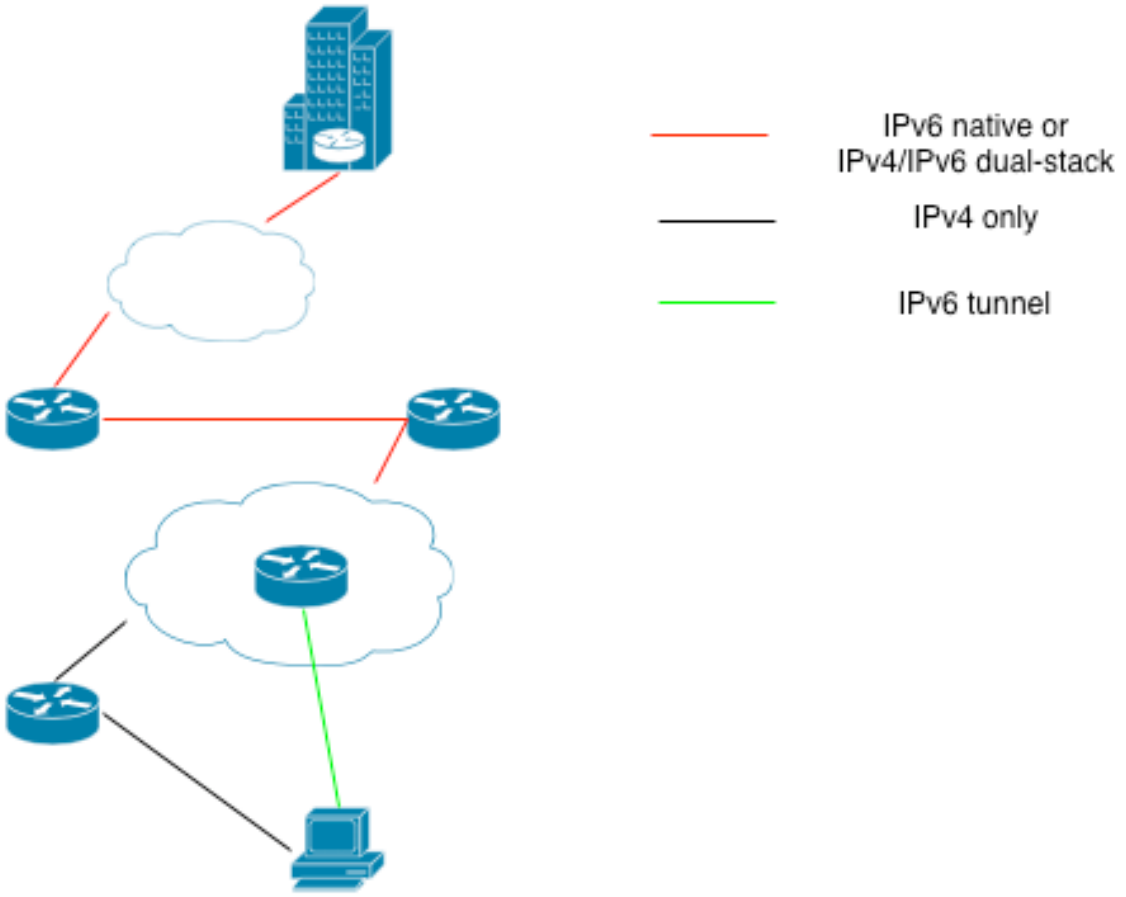
Transition technologies

- 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

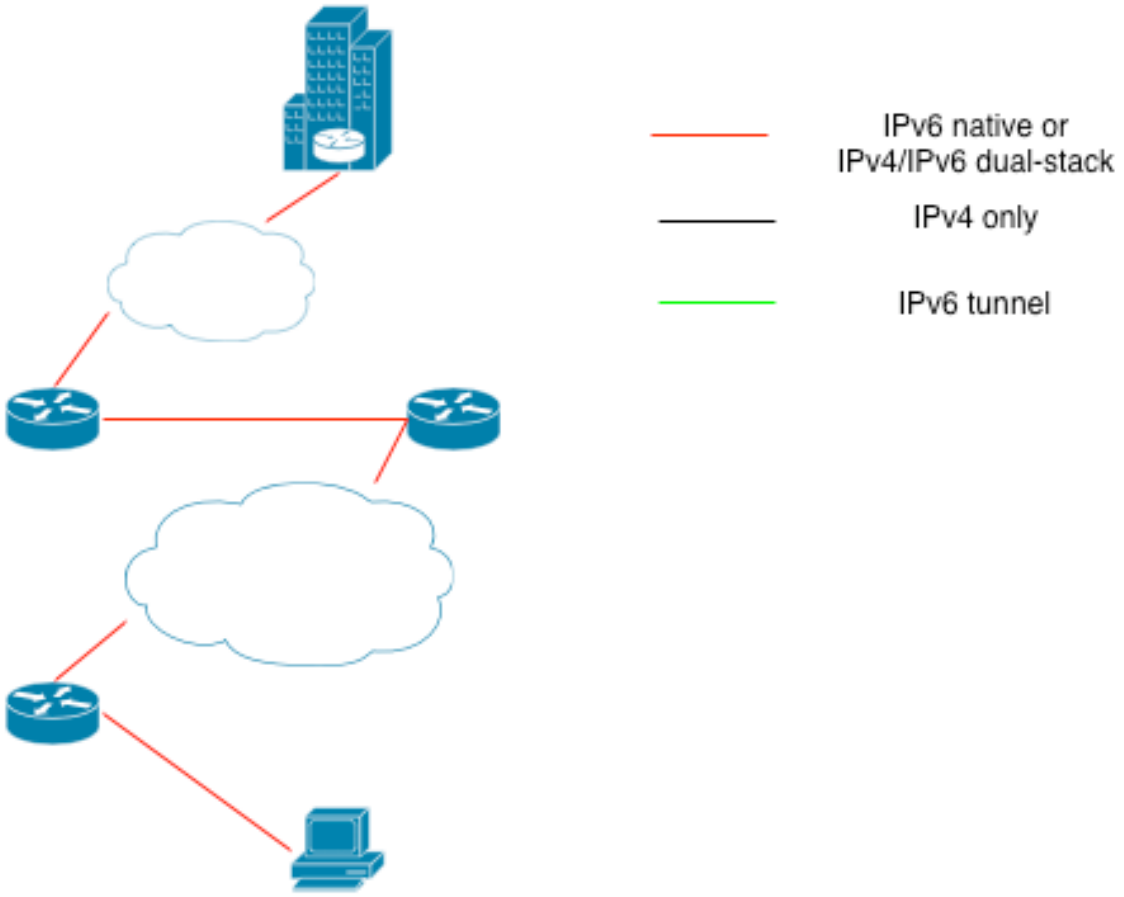
End-user 1



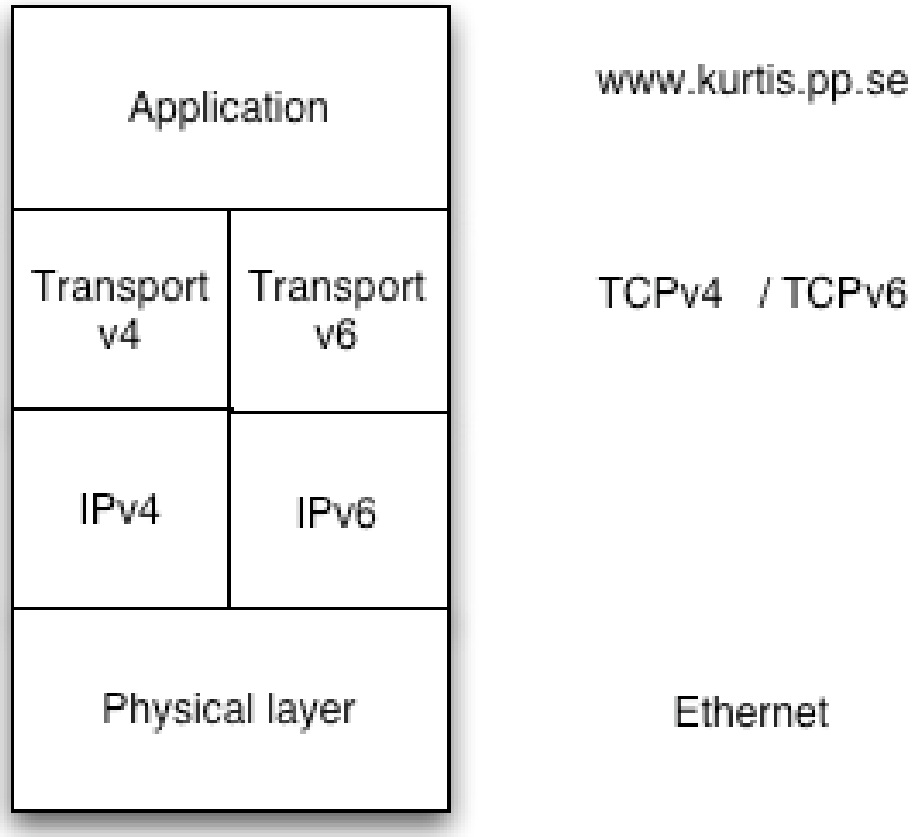
End-user 2



The “lucky” end-user



Dual-stack


Application
Transport v4
Transport v6
IPv4
IPv6
Physical layer
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
```

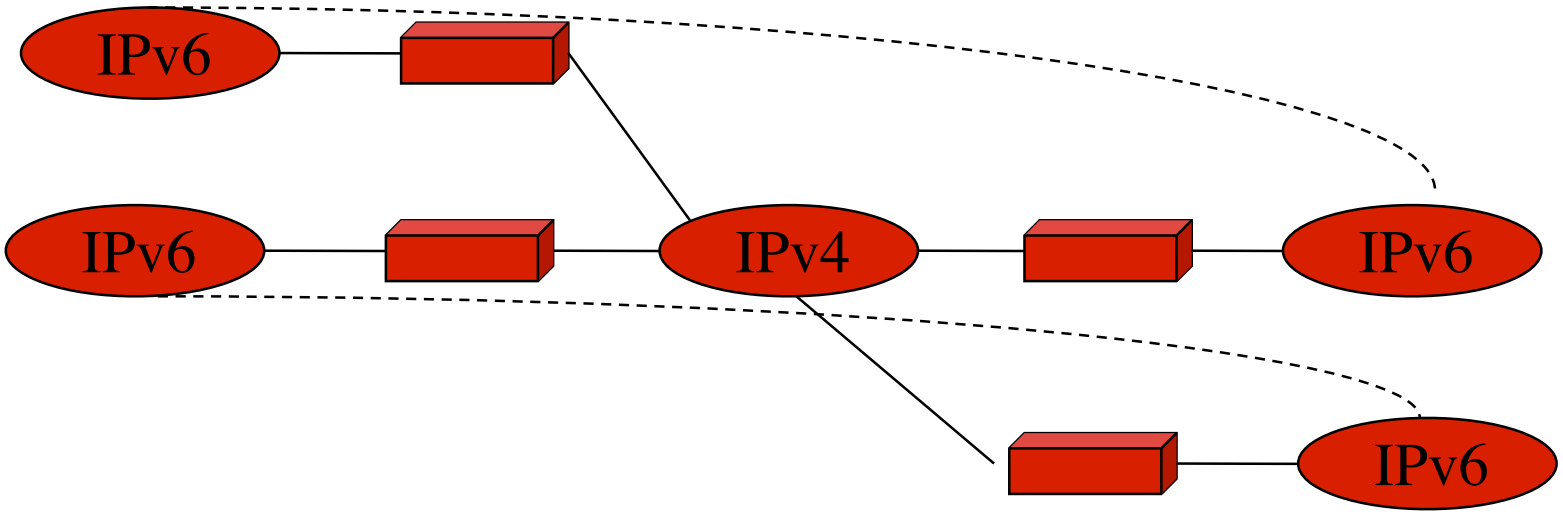
Dual -stack FreeBSD

```
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
```

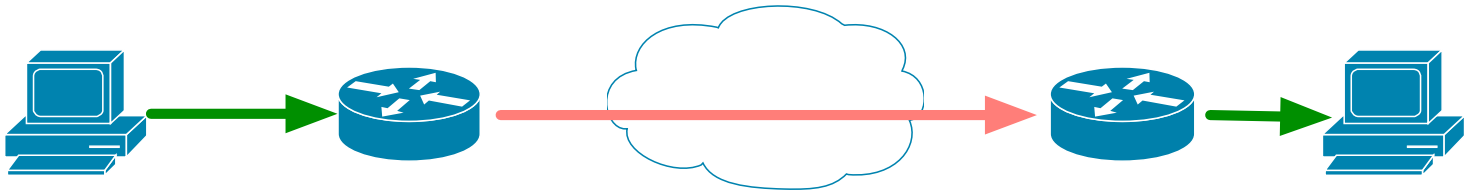
Tunnels

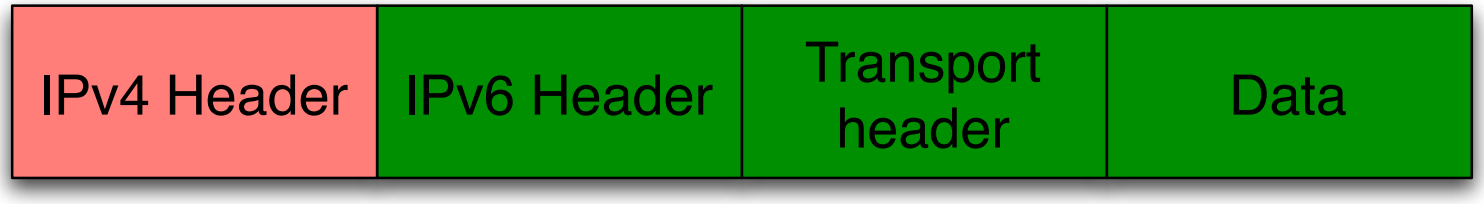
- Tunnels can be static
  - E.g. permanently configured
- Dynamic
  - ”Dial-on-demand”
  - The tunnels are established when they are needed
- Tunnels can be between
  - Host-Host
  - Host-Router
  - Router-Router

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 Tunnel10
```

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) :

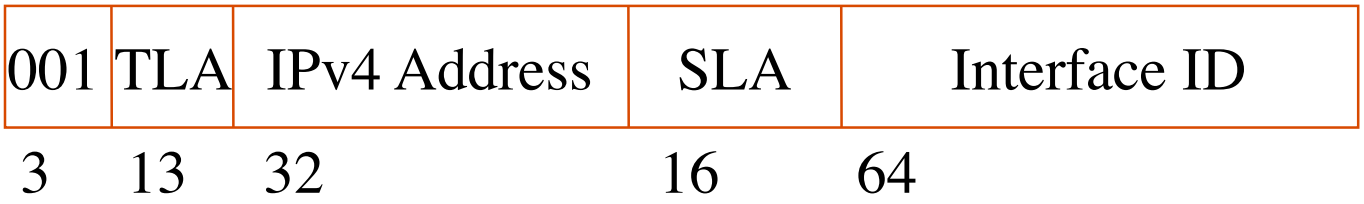
```
$ ifconfig gif0 inet 194.112.11.163 195.43.225.65

$ 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

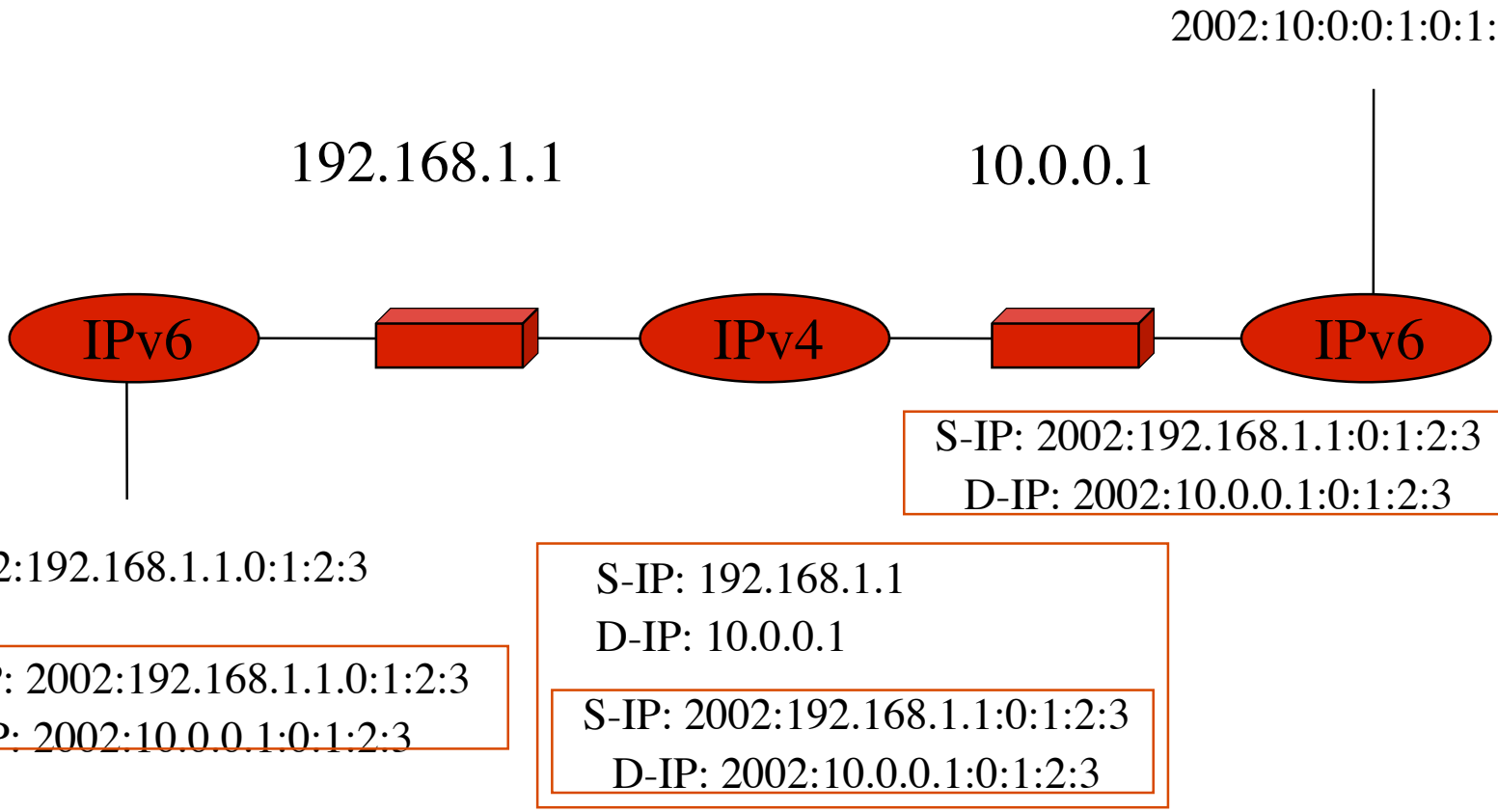
- In 6to4 an IPv4 gateway address is embedded in the IPv6 address
  - The IPv6 prefix for these addresses is 2002::/16



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

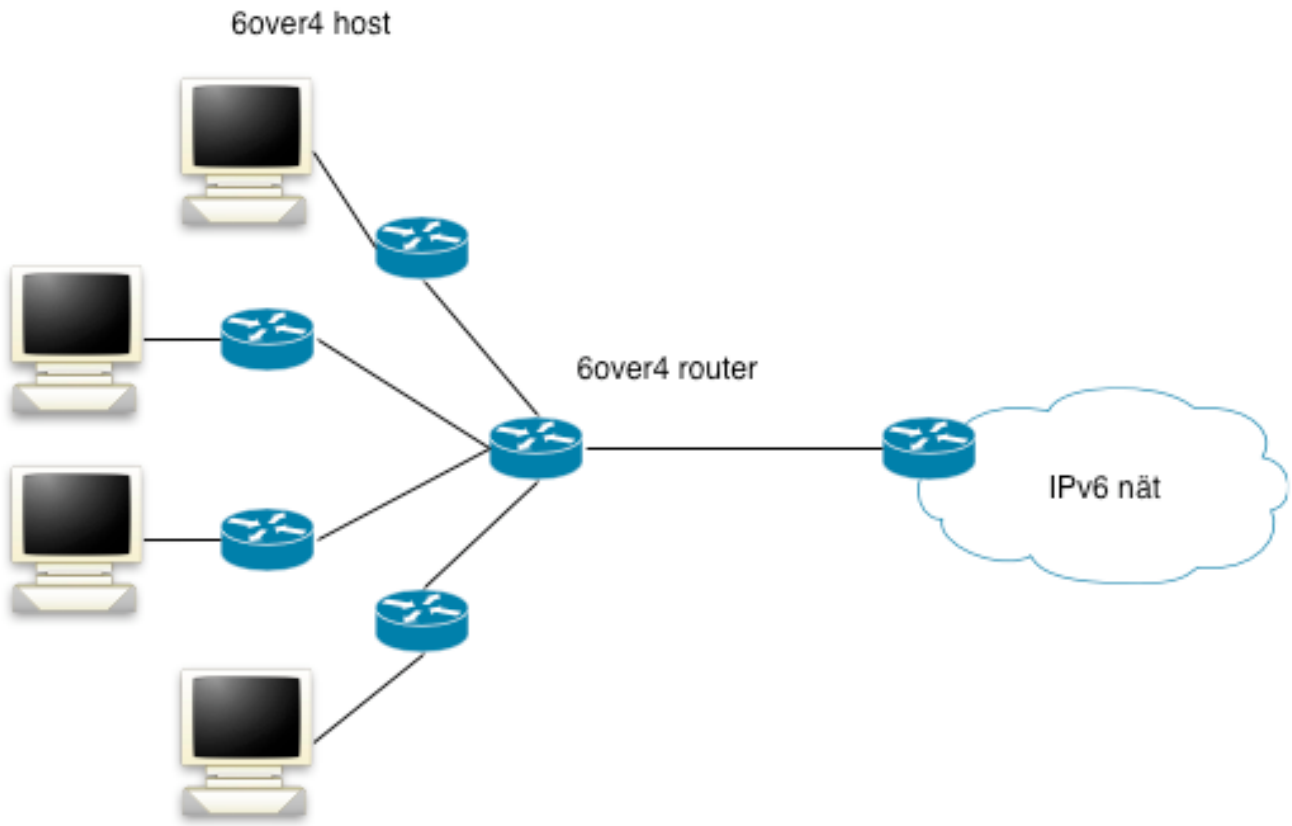
```
interface Loopback0
 ip address 192.168.99.1 255.255.255.0
 ipv6 address 2002:c0a8:6301:1::/64 eui-64
!
interface Tunnel0
 no ip address
 ipv6 unnumbered Ethernet0
 tunnel source Loopback0
 tunnel mode ipv6ip 6to4
!
ipv6 route 2002::/16 Tunnel0
ipv6 route ::/0 2002:c0a8:1e01::1
```

Thanks to Philip Smith@Cisco for the example

6over4

- RFC2529
- Uses a IPv4 network that can handle multicast as point-to-point links
  - Builds on the use of dynamic tunnels
- Demands a router connected to native IPv6 and that speaks 6over4
- Uses a scoped IPvp4 multicast group as an Ethernet link
  - Will perform ND and “autoconfiguration” over the multicast group
  - Configures a 6over4 interface
  - The host will then send packets as IPv4 to the 6over4 router

6over4



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
- Not yet specified but 3FFE:831F::/32 used for testing

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

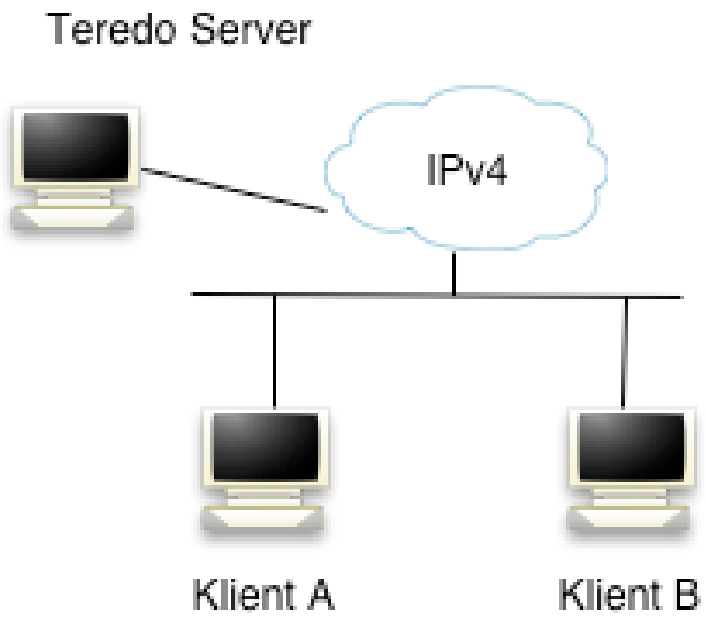
- Start sequence
  - Decide the type of NAT (Cone, restricted Cone, symmetrical)
  - If it is symmetrical NAT, Teredo fails
  - If not symmetrical NAT the set-up succeeds and the Teredo client configures a Teredo address
- When the connection is established, the Teredo client will send a number of “bubbles”
  - 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

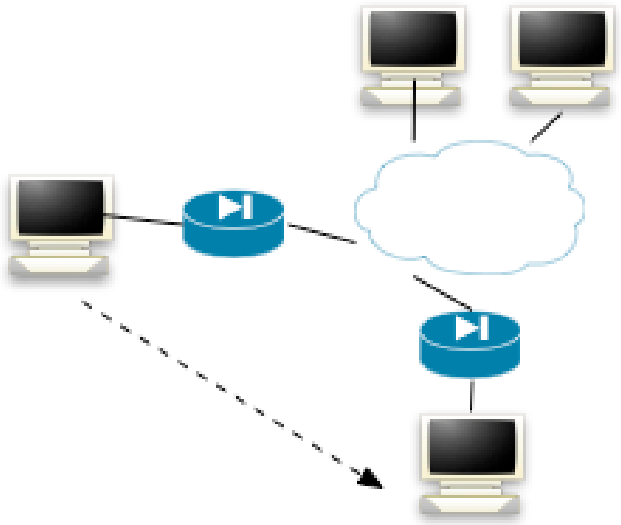


Teredo

- 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

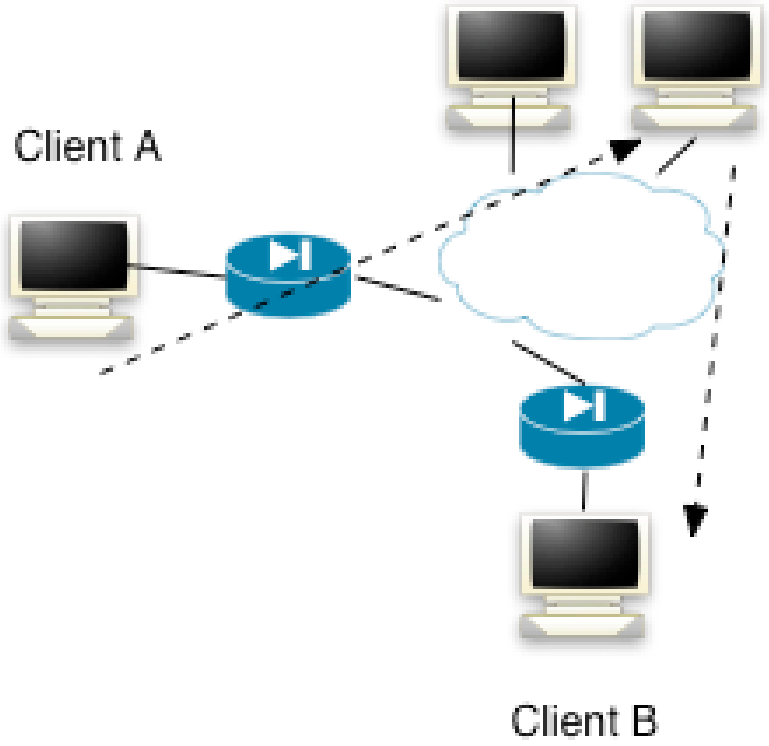
Cone NAT



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

Teredo

Restricted NAT

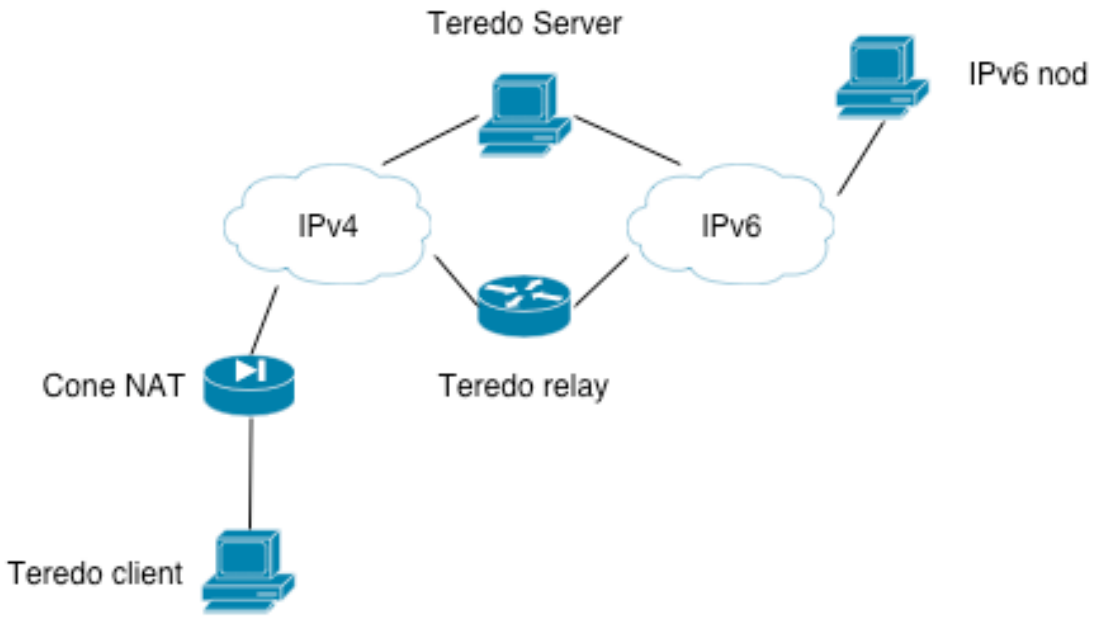


Teredo

- 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

ISATAP

- Intra Site Automatic Tunnelling Addressing Protocol
- Bridges dual-stack nodes within an IPv4 domain through automatic tunnelling
  - IPv6 in IPv4
- Uses a ISATAP aggregateable globally unique address that contains the IPv4 address
- Does not work through NAT
- Tries to solve the problem when a node is not in the same broadcast domain as an IPv6 router
  - Without multicast

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)

Transitioning Broadband Deployment in ISP networks

ISP Network Scenarios

- draft-ietf-v6ops-bb-deployment-scenarios-04.txt defines a number of scenarios
- Core/Backbone Networks
  - Access provider network
    - Broadband access and customer aggregation
    - Traffic handed over to service provider at L2 or L3
  - Service provider network
- Broadband cable networks
- Broadband xDSL networks
- Broadband Ethernet networks
- Broadband PLC networks

Access provider network

- Layer 2 network
  - Access provider operates a L2 network that aggregates to a L2 core where traffic is handed over to the service provider
  - Impact from deploying IPv6 is minimal
- Layer 3 network
  - Routers terminate customer traffic and then routes the traffic to the service provider
  - Significant impact of IPv6 deployment
  - Access network provider have three choices for IPv6 forwarding
    - Tunneling
    - Native IPv6 / Dual-stack
    - MPLS 6PE Deployment

Layer 3 access provider

- Native / Dual-stack
  - Straight forward concept
  - Requires software / hardware support
- Tunneling
  - If customers are assigned public IPv4 addresses the provider can support one of the many tunneling protocols, either dynamically or statically
    - For example the provider might want to operate it's own, private (to the customers) 6to4 gateway
  - If customers are assigned private IPv4 addresses the provider needs to make sure customers have access to tunneling technologies that supports private addresses such as Teredo
    - Provider might want to operates it's own Teredo server and relay

Broadband xDSL networks

- Point to point model
  - Each customer connected on a twisted pair to the DSLAM and uses a particular PVC
- PPP Terminated Aggregation (PTA)
  - PPP sessions between the CPE and the BRAS. PPP provides the L3 connectivity between customer and ISP
- L2TP aggregation
  - PPP session between CPE and BRAS. BRAS tunnels traffic to the ISP using L2TP
- The document maintains the current deployment scenarios
  - Based on that current models are supposed to have proven revenue streams :-)

xDSL point to point models

- Ethernet frames from CPE to the aggregation router are bridged (ATM)
- DSL modem and access infrastructure is L3 unaware
- Customer hosts, CPE, and aggregation router needs to be IPv6 capable
- If CPE is not router the aggregation router can assign a /64 and do stateless address autoconf or DHCPv6
- If CPE is a router it can configure the link with either link-local addresses or autoconf
  - Can then either use DHCP-PD or DHCPv6 relay to configure customer hosts

xDSL PTA

- PPPoA or PPPoE between CPE and BRAS. BRAS does L3 routing to ISP edge
- CPE authenticates to BRAS. BRAS provides the address and DNS-server etc either directly or based on customer profile in DHCP server
- Host, CPE, BRAS and ISP edge router needs to be upgraded
- The BRAS needs to support DHCP-PD and to relay DHCPv6 requests from hosts

L2TPv2

- The BRAS forwards the CPE initiated session to the ISP edge router
  - AAA performed by the ISP
- L2TPv2 can be run over either of IPv4 or IPv6. If IPv6 the BRAS needs to support IPv6
  - Host, CPE and edge router needs to support IPv6
- The link will be addressed by the authentication
- The edge router needs to support DHCP-PD and DHCPv6 relay

Broadband Ethernet

- Fiber To The Home (FTTH)
- Scenarios either of
  - Point to point
    - Customers are assigned a uniqe VLAN to the BRAS. VLANs are 802.1q trunked to either BRAS or edge router
  - PTA
  - L2TPv2

FTTH p2p

- L2 infrastructure is L3 unaware
- Only equipment that needs to be upgraded are host, (CPE), and edge router
- Switches might have to support MLD snooping
- Customer can configure addresses either by statelss address autoconf or DHCPv6
- If a CPE is present the edge router needs to support DHCP-PD and DHCPv6 relay

FTTH

- PTA and L2TPv3 are analogous to xDSL
  - Especially if CPE is present
- Else the DHCP-PD is not needed
- In all cases the edge-router currently needs to be the DHCP-PD server as it otherwise have no method for learning the prefixes
-

IPv6 Neighbour discovery

IPv6 Neighbour Discovery

- Described in RFC2461
  - Being updated
- 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

- RFC2462
- 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 RFC3041
- Thought to help renumbering of a network
- Problem
  - How do I find a DNS server?
  - How do I send update to the DNS server?

DHCPv6

DHCPv6

- Defined RFC3315
- DHCPv6 could be specified to also carry IPv4 addresses as replies, but that is currently not done
- UDP based
  - Client listens on UDP port 546
  - Servers and Relay agents listens on UDP 547

DHCPv6

- The DHCPv6 server will receive requests on a link-scoped multicast address. The client is unaware of the servers address
  - And there is no broadcast in IPv6
- DHCPv6 Relays can be used to send messages to servers that are off-link

Client/Server exchange with 2 messages

- If the client does not need an address but just configuration information (DNS, NTP etc) the client sends a message to All_DHCP_Relay_Agents_and_Servers multicast address
- Servers will respond with a messages
- This assumes no need for address configuration

Client/Server exchange with 2 messages

- If the server is willing to commit IPv6 address and configuration to a client without "policy verification" the exchange can be done with two messages
  - Client sends *Solicitation* message to All_DHCP_Relay_Agents_and_Server address
  - Indicates it accepts an immediate reply
  - A server willing to serve replies with *Replay* message

4 Message exchange

- If the client does not know the unicast address of a server
  - Client first find servers by sending a *Solicitation* message to All_DHCP_Relay agents_and_Servers
  - Servers reply with *Advertise* message
  - Client picks one server and sends a *Request* message
  - Server replies with *Reply* message

Valid lifetimes

- A client can request the extension of the lifetime of an address with a *Renew* message to the server
- The server replies with a *Reply* message with new lifetimes

Multicast addresses

- All_DHCP_Relay_Agents_and_Servers
  - FF02::1:2
  - Link-local scoped multicast group
  - All relay agents and servers are members
- All_DHCP_Servers
  - FF05::1:3
  - Site scoped multicast gruoup
  - Used by relay agent to communicate with servers when it
    - Want to send a message to all servers
    - Does not have the unicast address of a server

DHCPv6 Message types

- SOLICIT (1)
  - A client sends a Solicit message to locate servers.
- ADVERTISE (2)
  - A server sends an Advertise message to indicate that it is available for DHCP service, in response to a Solicit message received from a client.
- REQUEST (3)
  - A client sends a Request message to request configuration parameters, including IP addresses, from a specific server.
- CONFIRM (4)
  - A client sends a Confirm message to any available server to determine whether the addresses it was assigned are still appropriate to the link to which the client is connected.

DHCPv6 Message types

- RENEW (5)
  - A client sends a Renew message to extend the lifetimes on the addresses assigned to the client and to update other configuration parameters.
- REBIND (6)
  - A client sends a Rebind message to any available server to extend the lifetimes on the addresses assigned to the client and to update other configuration parameters; this message is sent after a client receives no response to a Renew message.
- REPLY (7)
  - Contains assigned addresses and configuration parameters.
  - Sent in response to a Solicit, Request, Renew, Rebind message received from a client. Sent containing only configuration parameters in response to an Information-request message.
  - Sent in response to a Confirm message confirming or denying that the addresses assigned to the client are appropriate to the link to which the client is connected.
  - Sent to acknowledge receipt of a Release or Decline message.

DHCPv6 Message types

- RELEASE (8)
  - A client sends a Release message to the server that assigned addresses to the client to indicate that the client will no longer use one or more of the assigned addresses.
- DECLINE (9)
  - A client sends a Decline message to a server to indicate that the client has determined that one or more addresses assigned by the server are already in use on the link to which the client is connected.
- RECONFIGURE (10)
  - A server sends a Reconfigure message to a client to inform the client that the server has new or updated configuration parameters, and that the client is to initiate a Renew/Reply or Information-request/ Reply transaction with the server in order to receive the updated information.

DHCPv6 Message types

- INFORMATION-REQUEST (11)
  - Sent to a server to request configuration parameters without the assignment of any IP addresses to the client.
- RELAY-FORW (12)
  - A relay agent sends a Relay-forward message to relay messages to servers, either directly or through another relay agent. The received message, either a client message or a Relay-forward message from another relay agent, is encapsulated in an option in the Relay-forward message.
- RELAY-REPL (13)
  - A server sends a Relay-reply message to a relay agent containing a message that the relay agent delivers to a client. The Relay-reply message may be relayed by other relay agents for delivery to the destination relay agent.
  - The server encapsulates the client message as an option in the Relay-reply message, which the relay agent extracts and relays to the client.

DHCP Unique Identifier (DUID)

- Used to uniquely identify a client/server in DHCPv6
- Must be stable over time and interface hardware changes

DHCP Unique Identifier (DUID)

- DUID-LLT
  - DHCPv6 Unique Identifier based on Link-local address and local time
- DUID-EN
  - DHCPv6 Unique Identifier based on Enterprise ID as registered with IANA
- DUID-LL
  - DHCPv6 Unique Identifier based on Link-layer address
    - Link-layer address must be guaranteed to be stable and unique

Use of DUID

- Used by the servers to create Identity Assocications
  - Per client configurations
- Clients use it to identify servers
  - A client can reach a specific DHCPv6 server either by using the unicast address or by setting the server DUID in the *Server identifier* option
  - All servers are identified by leaving the *Server identifier* option unset

DHCPv6 client Src selectio

- A DHCP

DHCPv6 and RFC3041

- DHCPv6 clients can request a temporary address
- DHCPv6 server will then assign a RFC3041 based address
- Can be used to update DNS for the assigned temporary addresses

DHCPv6 and DNS

- DHCPv6 can be used to provide stateless address autoconfiguration clients with extra configuration information, such as DNS
- DHCPv6 servers can be used to handle updates of reverse and forward DNS for stateful, stateless and RFC3041 clients

DHCPv6-PD

- RFC3633
- Defines a mechanism where a *delegating router* acts as DHCPv6 server for prefixes requested by a *requesting router*
- The *requesting router* can then delegate addresses out of that prefix

IPv6 QoS

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

Mobile IPv6

Mobile IPv6

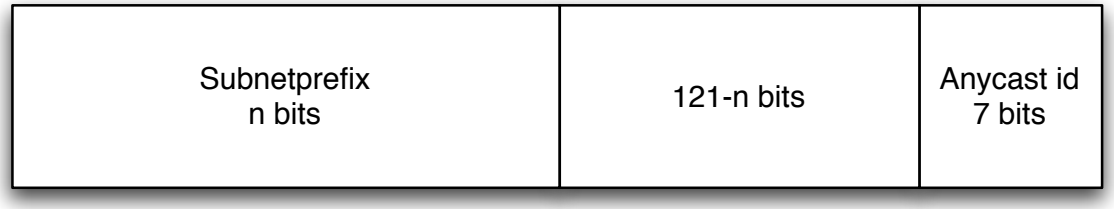
- Work to migrate Mobile-IP to support IPv6 was started
- That work showed there was quite a lot of things that needed to be done to improve Mobile IP
  - Security
  - Route optimizing
- The Mobile-IPv6 WG was given an extened mission
- HIP
  - The ultimate solution
  - Modifies the IP stack

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

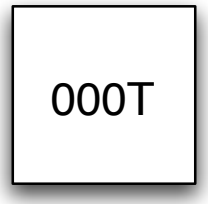
- Required anycast address
  - The subnet-router anycast address is predefined

- 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

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

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

```
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 onof
 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
```

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

IS-IS

IS-IS

- draft-ietf-isis-ipv6-06.txt
- 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 configuration example

```
!
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
```

BGP

BGP and IPv6

- BGP will behave exactly as for IPv4
  - IPv6 will make use of multiprotocol support
- NEXT_HOP and NLRI are expressed as IPv6 addresses and prefix

- Cisco IOS specific
  - For IPv6 different address-families where introduced into the configuration
  - And some commands got renamed
    - `show ip bgp` became `show bgp`

BGP and IPv6

```
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
 !
```

BGP and IPv6

```
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
!
```

Peer Groups

- With many iBGP peers, the BGP process will take longer as the same updates are calculated multiple times
  - The CPU of the router will re-calculate the updates sequentially per peer
- With many peers the configuration will become quite long
  - Same route-map statements and filters are listed for multiple peers
  - Risk for errors increase
- Peer-groups let you group your peers and give all members of a peer-group the same set of configuration
- Outgoing UPDATE messages are generated once per peer-group
- Works with iBGP and eBGP peers

Peer Groups

```
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
```

Peer Group

```
router bgp 1
 neighbor netnod peer-group
 neighbor netnod send-community
 neighbor netnod version 4
 neighbor netnod soft-reconfiguration inbound
 neighbor netnod route-map set-peer-locpref in
 neighbor netnod update-source Loopback10
!
 neighbor 2001:db8::1 remote-as 3220
 neighbor 2001:db8::1 peer-group netnod
 neighbor 2001:db8::1 description Peering with Kurtis
 neighbor 2001:db8::1 remote-as 1257
 neighbor 2001:db8::1 peer-group netnod
 neighbor 2001:db8::1 description other Netnod peering
```

Prefix-list

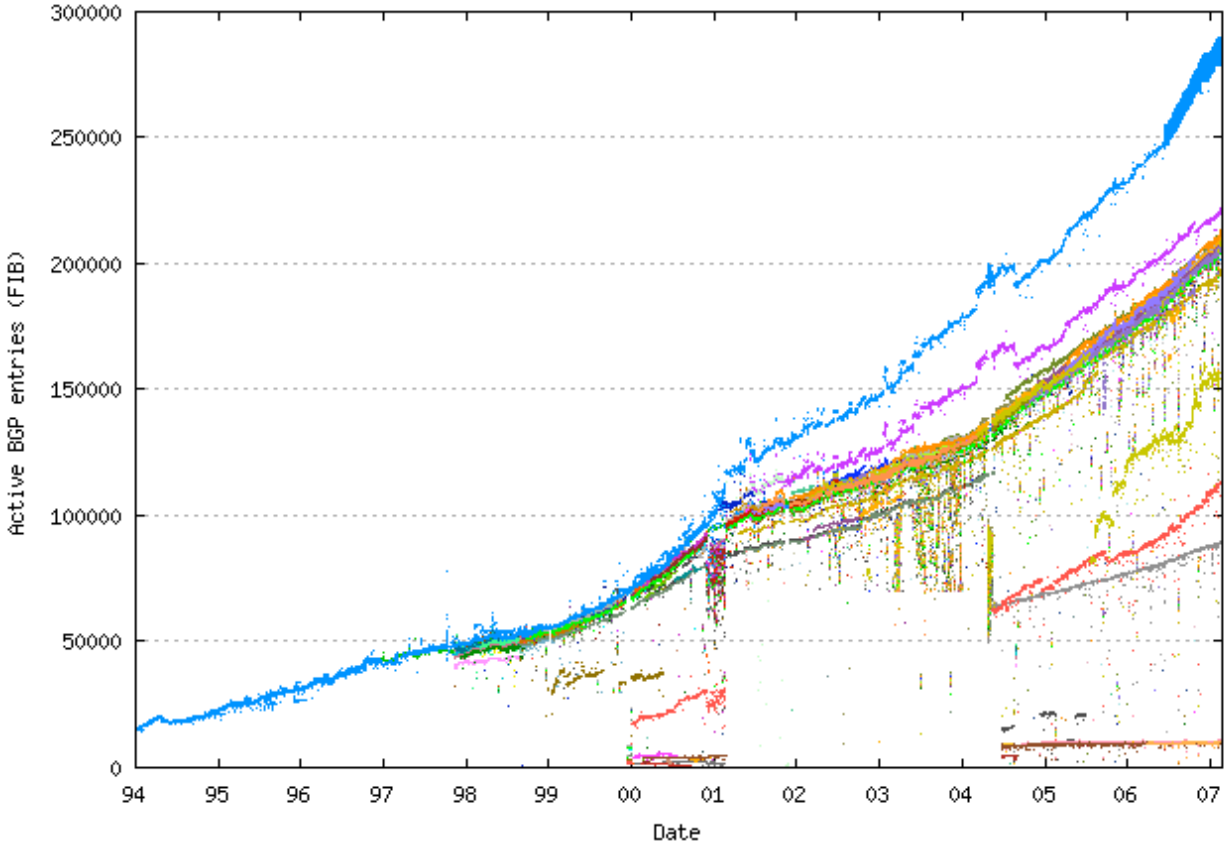
```
router bgp 1
 neighbour 2001:db8:1::1 remote-as 2
 neighbour 2001:db8:1::1 prefix-list infilter in
 neighbour 2001:db8:1::1 prefix-list outfilter out
 neighbour 2001:db8:1::1 remote-as 3
 neighbour 2001:db8:1::1 prefix-list infilter in
 neighbour 2001:db8:1::1 prefix-list outfilter out
!
ipv6 prefix-list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

IPv6 and multihoming

IPv6 and routing

- When developing IPv6, an active choice was made not to address the scalability problem of the global routing table
  - The global default free table has occasionally grown faster than Moores law
  - This *MIGHT* lead to problems in the future, especially with visions of multihomed DSL customers
    - And why would we have an architecture that did not allow that
  - fixed address length was also chosen
- Later we realized that
  - Exponential growth does not scale
  - We need to address this in IPv6
- This lead to the creation of the multi6 working group in the IETF

IPv6 and routing



- Thanks to Dave Meyer for the following slides

Workshop Conclusions

- As one might expect, much of the discussion centered around a general concern about the growth of the Default Free Zone (DFZ) RIB/FIB and the overall scalability of the routing system
- In particular, the concern was (generically) that DFZ RIB/FIB growth, driven by multi-homing, traffic engineering, and the cost of renumbering is problematic and not scalable in any event
  - However, a sufficiently precise definition of “scalability” has turned out to be illusive
  - **There has been no subsequent consensus on whether there is are scalability issues with the routing/addressing system or not**
- BTW, strawman scalability defn for the DFZ RIB:
  - DFZ RIB size scales sub-linearly with number of end sites
  - UPDATE rate scales sub-linearly with number of end sites
  - Routing converges faster than it currently does
    - And with a upper bound better than O(n!)
  - And all while providing (for example) for multihoming, some form of TE, and provider independence

Workshop Conclusions, cont.

- The size of the internal RIB/FIB and its effect on convergence was also a point of concern
  - Various copies of the DFZ RIB + VPN + ...
  - And what about multiple RIBs at AFBRs?
  - Or multiple peers/router, say, like 15?
    - 15 * 300K ~ O(4.5 x 10^6) RIB entries
  - Are RIBs of that size a problem, and if so, along which dimensions?
    - **No subsequent consensus on this point**
- Overloading of the IP address with both locator and identifier properties was a concern
  - However, there was a lack of clarity as to whether this was a problem (overloading) or a solution (loc/id split)
  - And can we maintain some sort of TE capability?
- BTW, what are the issues with TE?

What are the Issues with TE?

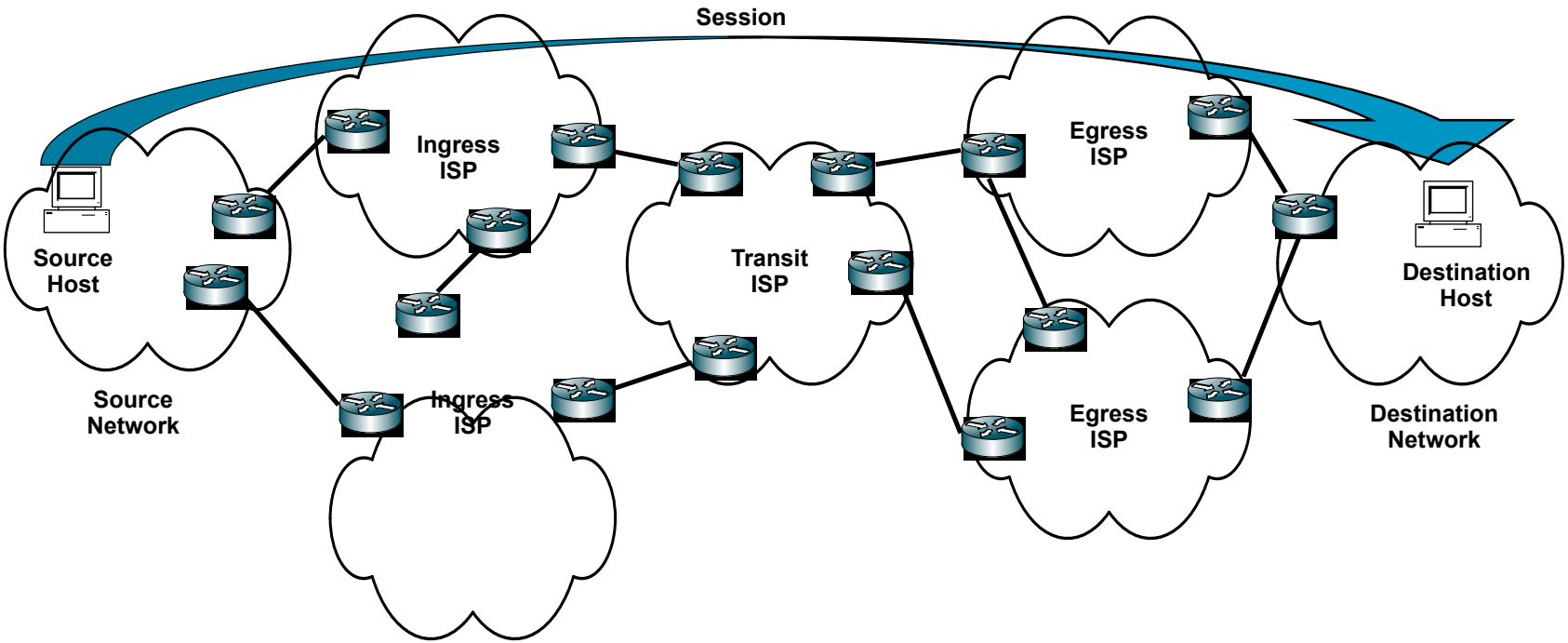
- First, TE is (very roughly) the practice of moving traffic to where the network has capacity for that traffic, or moving traffic as required by policy
  - Contrast capacity planning
- 10K View: How can TE be accomplished in a scalable manner
  - In particular, without more specifics?
- Note that it is unclear as to whether there is a way to do TE, as currently practiced, without more specifics
- So, how is TE currently practiced?
  - Lets take a look

Overview of Current TE Practices

- First, credit to Jason Schiller @ VZ for much of the framework presented here

- TE seems to be the “hard” problem
  - Technologies exist (e.g., Locator/ID split) can promise to solve many of the problems in the multihoming/mobiity space
    - In theory, anyway…
- Now, to understand current TE practices, we need to consider each of following roles/perspectives:
  - Source Host
  - Destination Host
  - Source Network
  - Destination Network
  - Ingress ISP
  - Egress ISP
  - Transit ISP

TE Context


Session
Ingress ISP
Egress ISP
Transit ISP
Source Host
Destination Host
Source Network
Ingress ISP
Egress ISP
Destination Network

TE Context, in words

- Source Host is the source of the session
- Destination Host is the destination of the session
- Source Network is the administrative network containing the source host
- Destination Network is the administrative network containing the destination host
- Ingress ISP is the ISP that is directly connected to the source network
- Egress ISP is the ISP that is directly connected to the destination network
- Transit ISP is an ISP in the path of the session that is neither connected to the source network nor connected to the destination network

Source Host to Destination Host Requirements

- Hosts need to be able to build e2e sessions
- Session *should* survive transitions to different upstream/downstream ISPs
- Host should be able to send/receive traffic on any of its interfaces with any of the valid addresses of its interfaces
- Destination host may be a content provider server with many concurrent sessions, and should not carry a burdensome amount of state

Source Host to Destination Host Current IPv4 Implementation

- A session is built between source and destination IP address

  IP address is both end system ID and Locator

- A session must survive transitions to different ISPs since the destination network is reachable through all upstream ISPs

  Causes the DFZ to carry all multi-homed prefixes

- A host may have only one IP address that all upstream ISPs route

  Causes the DFZ to carry all multi-homed prefixes

- Source and destination hosts need not be aware of network topology

  Causes the DFZ to carry all multi-homed prefixes

Source Network Requirements

- Source network should be able to control routing policy for its outbound links, including

  Routing policy should be configured at the network level

  - Limited points of maintenance
  - Administrative responsibility for TE consistent with network router configuration responsibility

  Supporting various configurations (combinations of):

  - Primary/backup
  - (weighted) load sharing
  - Best path
  - And perhaps dial traffic

- Sessions must survive transitions to different upstream/ downstream ISPs

Source Network Current IPv4 Implementation

- A source network controls routing policy for its outbound links via:
  - BGP external routes learned/manipulated or IGP metrics
    - IGP metrics can have various attributes, including:
    - Primary/backup – learning only a default route from upstreams, setting different local pref (MED if all links are to a single AS) via policy
    - (weighted) load shared – learning only a default route setting equal metrics
      - Some of the network IGP closer to one exit point
      - A subset of routes from one or more upstreams is made less preferred
    - Best – Learning full routes from all upstreams
      - Some of the network IGP closer to one exit point
      - A subset of routes from one or more upstreams is made less preferred
      - Providers may offer different sets of routes

- Sessions survive transitions to different upstream/downstream ISPs since the destination network is reachable through all upstream ISPs

  Causes the DFZ to carry all multi-homed prefixes

Destination Network Requirements

- Destination network should be able to control routing policy for its inbound links, typically via
  - Routing policy configured at the network level
    - Limited points of maintenance
      - Administrative responsibility for TE consistent with network router configuration responsibility
  - Destination networks must also support various configurations (and combinations thereof), including
    - Primary/backup
    - (weighted) load shared
    - Best
    - Dial traffic up or down in each case
- Session *should* survive transitions to different upstream/ downstream ISPs

Destination Network Current IPv4 Implementation

- Destination network controls routing policy for its inbound links
  - BGP external routes advertised/manipulated, including
    - Primary/backup – Advertise the same prefix to all upstreams with BGP community to lower the local pref (high MED w/ single upstream AS) on backup links
    - Load shared – Advertise the same prefix to multiple peering points on the same upstream AS. Or advertise different sets of routes to each upstream.
      - Balance traffic by changing the amount of IP space advertised to each upstream
    - Best – Advertise all routes to all upstreams
      - Shift traffic by making the best path a little less best (using AS prepending)
      - Shift traffic by withholding (deprefing) some destinations from one or more upstreams
- Sessions survive transitions to different upstream or downstream ISPs as the destination network is reachable through all upstream ISPs
  - ***Causes the DFZ to carry all multi-homed prefixes***

Ingress ISP Requirements

- Allows customers to send transit traffic
- Support customer's outbound policy
- Note that the ingress ISP has little or no control if customer prefers a different upstream ISP for outbound traffic

Ingress ISP Current IPv4 Implementation

- May send different sets of routes to the customer to help support customer outbound policy

  Full routes

  Full routes with MEDs

  ISP aggregates only

  ISP aggregates and routes from directly connected BGP customers

Egress ISP Requirements

- Supports customer's inbound TE preference
- Prefers to deliver traffic directly to customer if no customer inbound TE is configured
- Will deliver traffic to another Egress ISP (or transit ISP) if inbound customer TE policy desires

Egress ISP Current IPv4 Implementation

- Ways to implement a customer's inbound TE preference:

- Local Preference (Primary and/or Backup)
  - Support BGP communities to set higher LOCAL_PREF
  - Support BGP communities to set lower LOCAL_PREF
  - Set LOCAL_PREF on peer routes to default

- AS prepend
  - Allow customer to prepend their own AS
  - Support a BGP community to prepend the ISP's AS towards peers (various amounts of prepends)
  - Prepend only to a particular peer

Egress ISP Current IPv4 Implementation, cont.

- Route Containment
  - Support BGP community to limit route export
    - Don’t send to peers (or a particular peer)
    - Don’t send outside the country or continent
    - …
- Route Advertisements
  - Allow customers to make multiple more specific route advertisements to draw specific traffic to specific links

Transit ISP Requirements

- Does not have customer connected to the network
- Must deliver traffic closer to the destination
  - For loop avoidance, if nothing else
- Must honor cumulative TE preference of all transit ASes and destination AS when delivering traffic
- May choose between two or more equally good paths
  - This TE preference is passed along to other ASes that transit it to reach the destination
- May poison the TE to a particular path (usually ISPs with one or more transit provider)

Transit ISP Current IPv4 Implementation

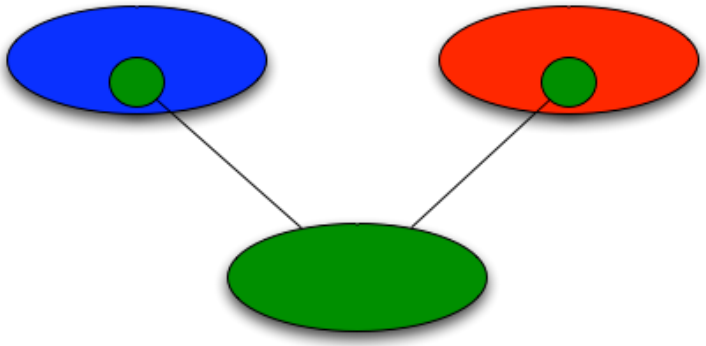
- All things being equal, for destinations not directly connected, the transit ISP needs to:
  - Choose shortest exit (IGP distance)
  - Choose most preferred exit (policy determined)
- Typically, ISPs IGP distance a particular over loaded peering point
  - Traffic to multi-homed destinations may use the other peer if that path becomes shorter
  - Some traffic to non-multi-homed destinations will exit at a different point to the same peer

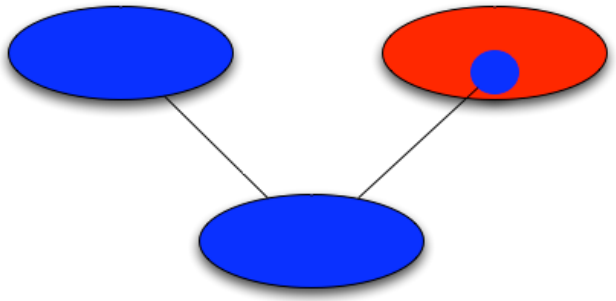
So, what architecture supports all of this?

- IPv4 multi-homing works because one or more specific routes per site are leaked to the DFZ
- IPv6 was supposed to solve this by only routing PA aggregates
  - Only routing PA aggregates isn't the answer, since
    - Doesn't solve provider independence
    - Doesn't support current multi-homing capabilities
- So the question is: Is there an architecture which satisfies all of these requirements (including multi-homing, mobility, and PI) without additional routes in the DFZ
  - That is, can similar multi-homing and TE preferences be supported in a scalable fashion (without more specifics?)

Multihoming Today

- AS + PI
  - Get an AS
  - Get PI space
  - Advertise and use BGP
- More specific PA
  - Advertise the more specific route from the ISP

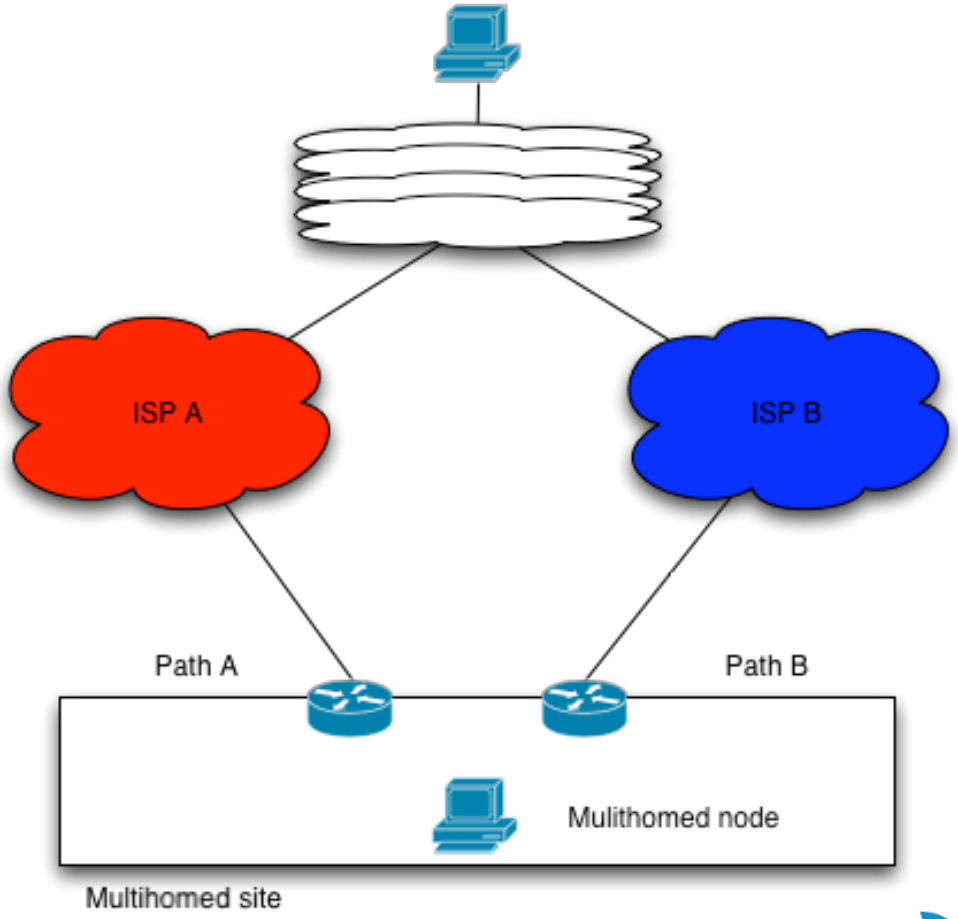




Effects of Multihoming

- Leads to "uncontrolled" growth of the routing table
  - Can lead to problems in the future
- But this is far from the only driving factor to todays problems
  - Traffic engineering using more specifics
- Would be better if each end-user/site could get a block from each provider
  - And be able to use both prefixes
  - Today this does not work due to inbound-filtering at the ISPs

Multihoming



What are the requirements for a solution?

- RFC3582 says the goals are
  - Redundancy
  - Load sharing
  - Traffic Engineering
  - Policy
  - Simplicity
  - Transport-layer survivability
  - DNS compatibility
  - Filtering
  - Scalability
  - Backward compatibility

- We should also think about
  - Interaction with routing
  - Aspects of id/loc separation if used
  - Packet rewrite in the path
  - Names, endpoints, DNS

Current Plan

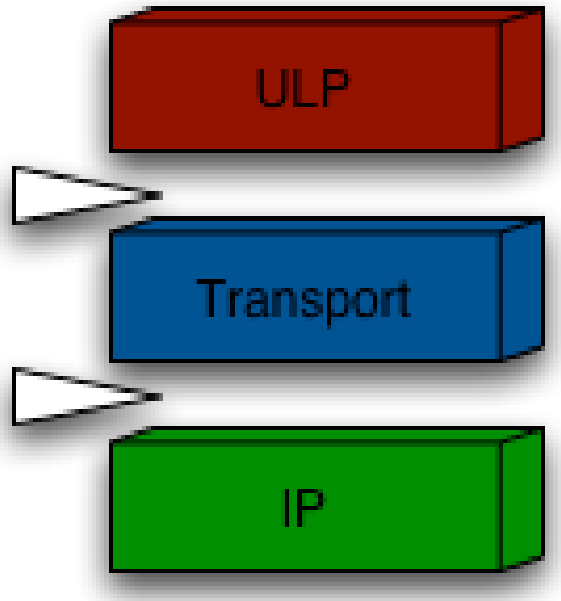
- This is not exactly IP as we know it…
- The multi6 WG decided that the way forward is a shim layer that let's you separate WHO you are from WHERE you are
  - Commonly called "identifier/locator split"
- Work moved from the multi6 WG to the shim6 WG

Possible Solutions where

- Crate a new protocol element
  - Identity
- Change the transport layer
  - Modified protocol element
- Change the interaction host / site-exit router
  - Modifying the forwarding plane

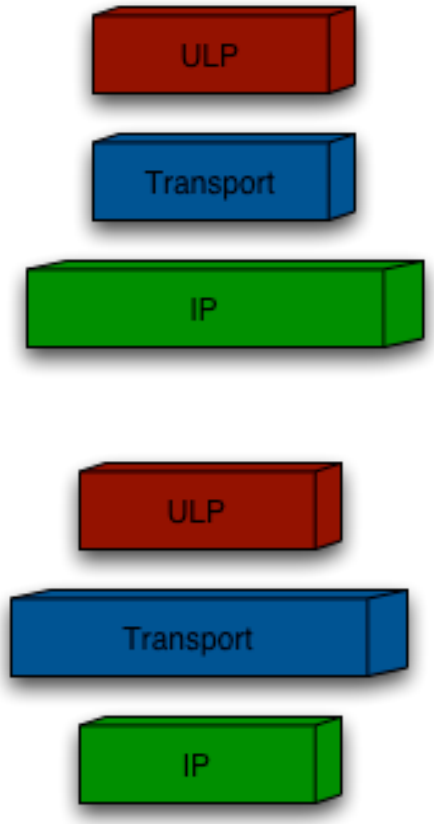
New Protocol Element

- Handles the presentation of identity to the ULP
- Handles the binding between locators and identity
- Handles mapping of sessions between locators / identity bindings



Modified Stack Element

- Modified IP to handle IP-in-IP like structures and bindings
  - For example MIPv6
- Modify transport level to handle the binding of sessions against locators
  - For example HIP, SCTP



The SHIM6 Solution

- host-based solution (rather than host and router)
- network layer (rather than transport)
- discoverable negotiated capability
- no new identifier space

The SHIM6 Approach

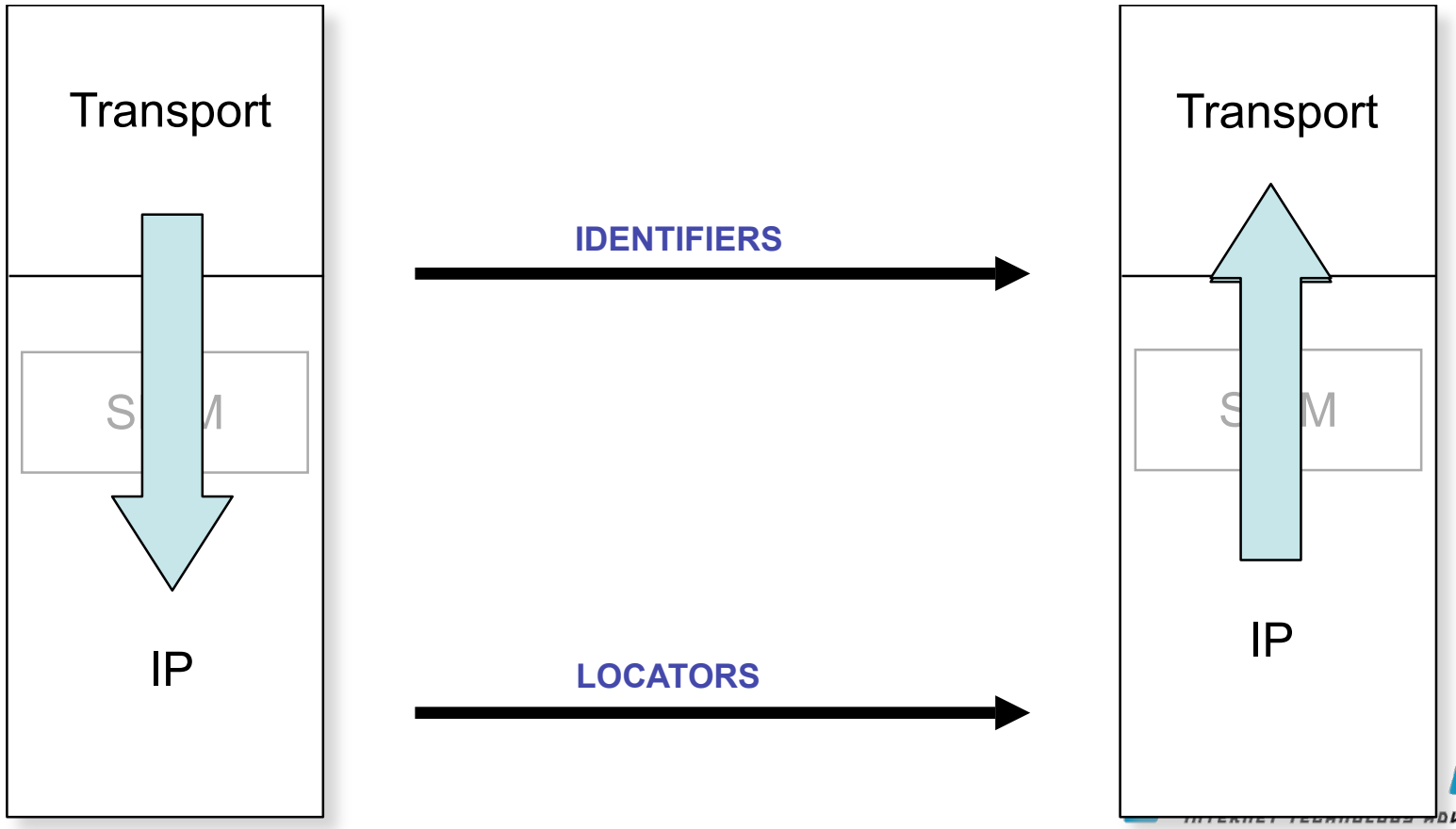
- a functional module at layer 3 (IP)
- the initial locator is the upper layer identifier (RFC3484 selection)
- subsequent negotiation to enable the Shim6 module for an upper layer identifier pair
- the Shim6 module translates upper layer identifiers into the currently active forwarding layer locators
- the upper layer identifier pair plus a context value forms the shared shim6 state identifier
- an IPv6 end-to-end header is used to signal SHIM6 context

shim6 - protocol

- Current thinking is that the base header will look remarkably like a HIP header
  - but it is *NOT!*
- Some issues are still TBD but we have come a far way....
  - At least four implementations are being written
- But it might not be what we are looking for...
  - More in a bit...

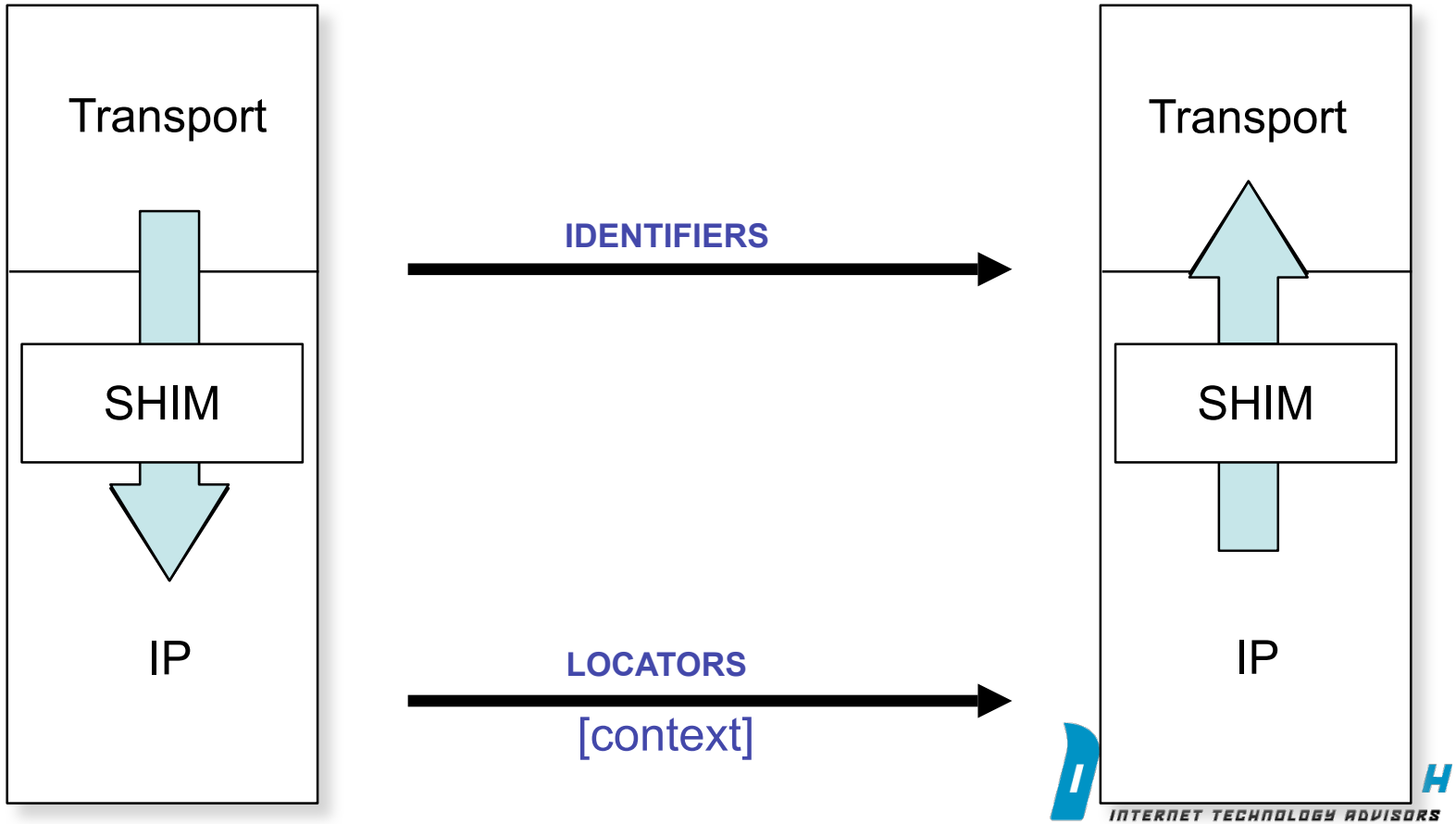
Shim6 Initial contact

No SHIM state active
Locator Selection using RFC3484
Locators and Identifiers are Equivalent



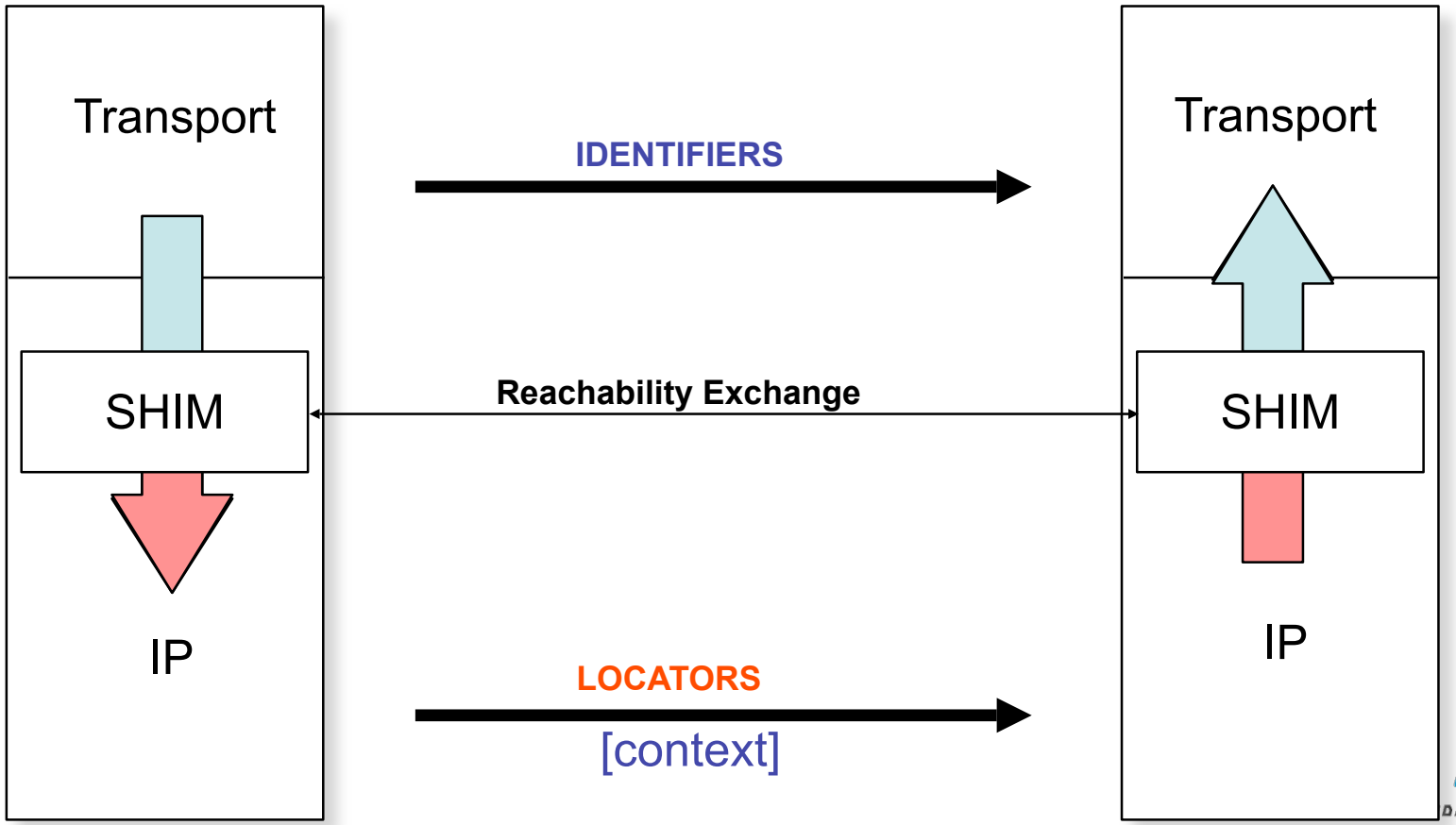
Shim6 Activation

SHIM active
Current Locator Sets exchanged
Locators and Identifiers are Equivalent

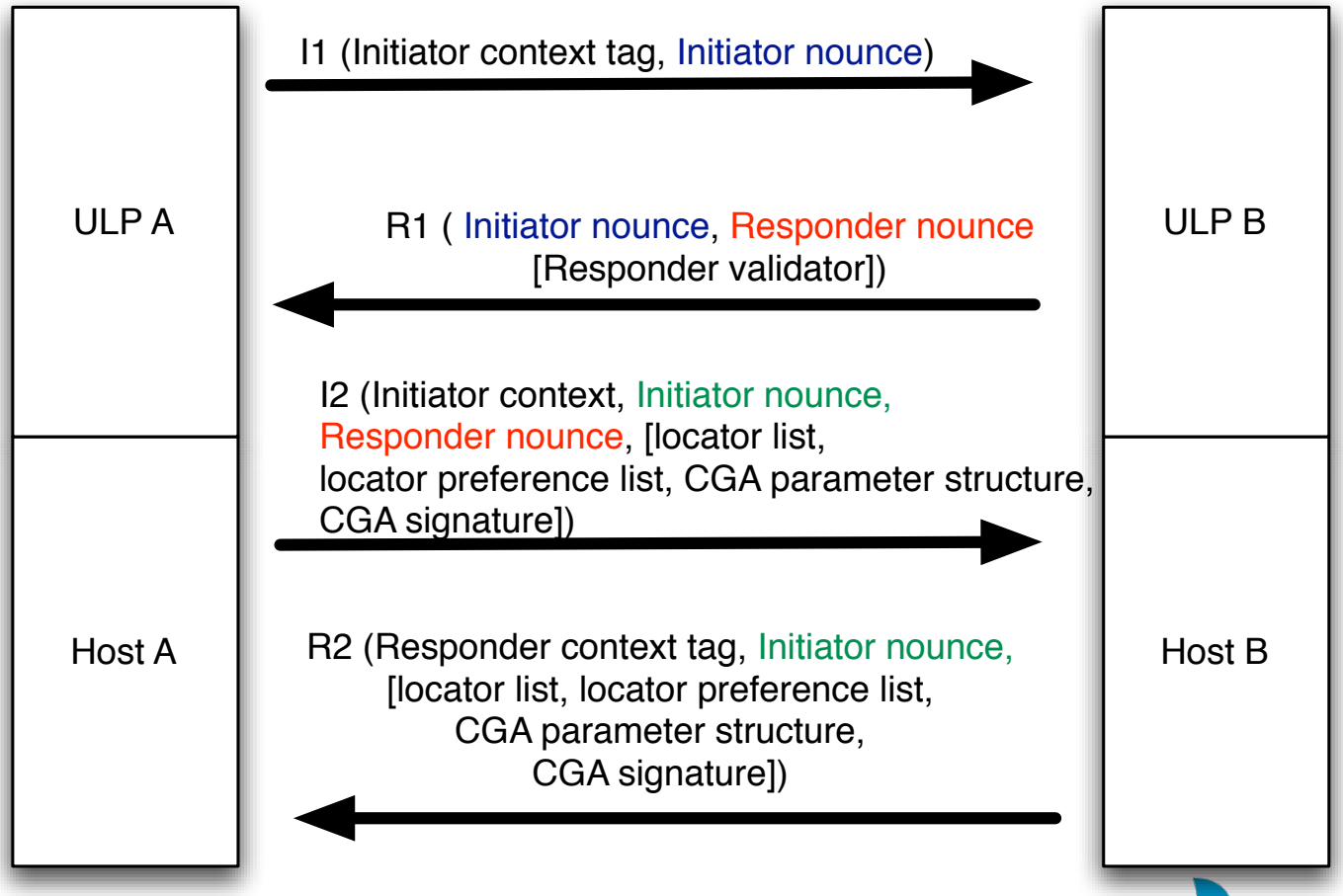


SHIM6 Locator Failure and Recovery

Detect locator failure
Explore for functioning locator pair
Use new locator pair – preserve identifier pair



Shim6 - protocol



SHIM6 Control Elements

- initial handshake (4-way) and locator set exchange
- locator list updates
- explicit locator switch request
- keepalive
- reachability probe exchange
- No-Context error exchange

SHIM6 WG Approach

- base protocol specification
  - protocol exchange and packet formats
  - address specification: CGA and HBA
  - functional decomposition

- refinements
  - upper layer signalling
  - traffic engineering hooks
  - contactless shim6
  - failure detection refinements
  - ingress filtering / source address path selection

However...

shim6 and operators

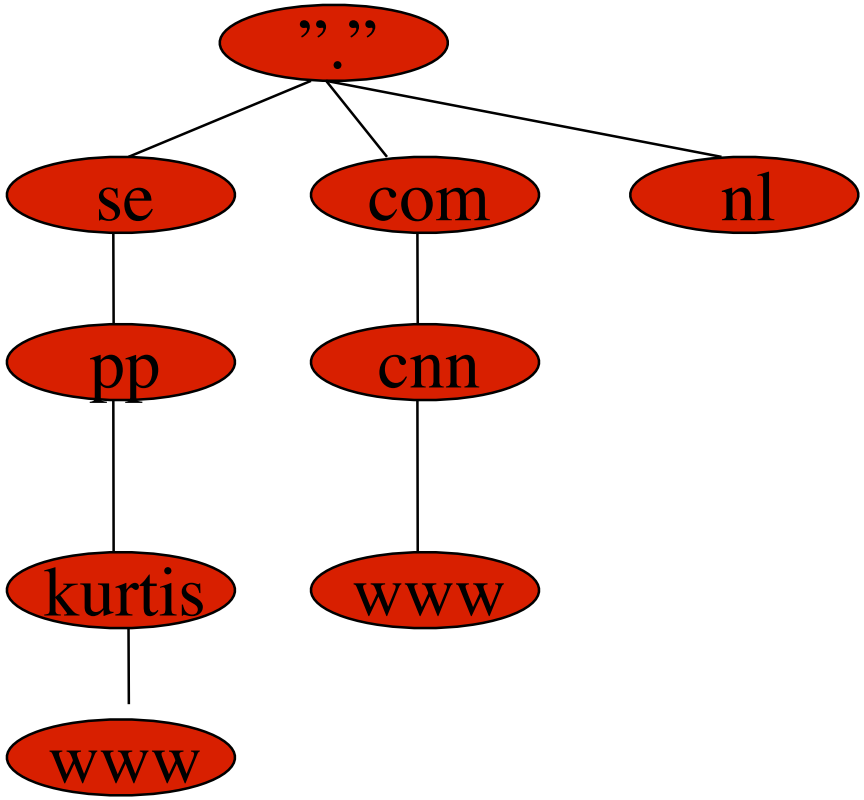
- shim6 has gotten a very mixed reception among operators
  - Some European networks does believe that shim6 would add value for them and open options for new services
    - Mostly ISPs with large residential customer base
  - Other operators believe shim6 takes away traffic predictability and TE capabilities

The 8+8/GSE/ESD split

- An old proposal since 1996
- Locators are added to addresses at provider ingress
- Have always had problems with secure bindings between id and loc
- However work in this area has picked up
- The IAB held a workshop in Amsterdam in Oct 2007
  - This effort is still on-going

IPv6 and DNS

DNS basics

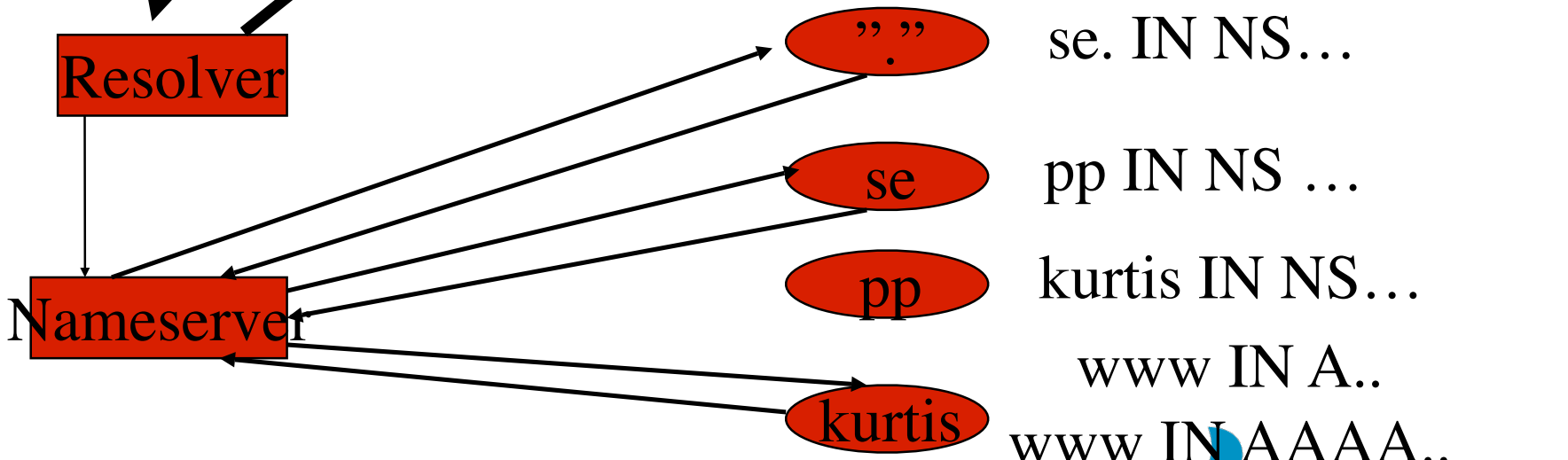


DNS grunder

Vilken adress har www.kurtis.pp.se?

```
$ nslookup
www.kurtis.pp.se
```

```
www.kurtis.pp.se  internet address = 195.43.225.69
www.kurtis.pp.se  IPv6 address = 2001:670:87:3001::6
```



New RRs for IPv6

- AAAA is used as A, that is for forward lookups
  - Example

```
www       A      195.43.225.69
          AAAA   2001:670:87:3001::6
```

New RRs for IPv6

- PTR is used just as normal for reverse lookups
  - Example `2001:670:87:3001::6`

In the zone
1.0.0.3.7.8.0.0.0.7.6.0.1.0.0.2.ip6.arpa.

```
0.0.0.0.0.0.0.6.0.0.0.0.0.0.0.0
  .
  IN   PTR   www.kurtis.pp.se.
```

IPv6 and DNS

- Reverse zones
  - in-addr.arpa. for IPv4
  - ip6.int.
    - Old zone for reverse delegations that should no longer be used
  - ip6.arpa.
    - The zone that should be used
- Delegation of reverse zones are as for IPv4
  - Your provider will register the delegation with RIPE

Applications

Applications

- To migrate applications are often a lot easier that thought
- Parts of the socket API is changed
  - And certain address agnostic calls have been introduced
- Names have instead been made more important

Applications

- IPv4 socket struct

```
struct sockaddr {

   sa_family_t sa_family;        /* Address family */

   char sa_data[14];             /* protocol-specific address */

};

typedef uint32_t in_addr_t;struct in_addr

{  in_addr_t s_addr;                      /* IPv4 address */};

   struct sockaddr_in {

        sa_family_t sin_family;  /* AF_INET */

        in_port_t sin_port;      /* Port number.  */

        struct in_addr sin_addr; /* Internet address.  */
                        /* Pad to size of `struct sockaddr'.  */
   unsigned char sin_zero[sizeof (struct sockaddr) -
   sizeof (sa_family_t) - sizeof (in_port_t) - sizeof (struct in_addr)];

};
```

Applications

```
struct in6_addr {
    union {
          uint8_t u6_addr8[16];
          uint16_t u6_addr16[8];
          uint32_t u6_addr32[4];
          }
    in6_u;
    #define s6_addr in6_u.u6_addr8
    #define s6_addr16 n6_u.u6_addr16
    #define s6_addr32 in6_u.u6_addr32
};
struct sockaddr_in6 {
    sa_family_t sin6_family;    /* AF_INET6 */
    in_port_t sin6_port;        /* Transport layer port # */
    uint32_t sin6_flowinfo;     /* IPv6 flow information */
```

Applications

- IPv4 code

```
socket(PF_INET, SOCK_STREAM, 0);  /* TCP socket */
socket(PF_INET, SOCK_DGRAM,  0);  /* UDP socket */
```

- IPv6 code

```
socket(PF_INET6, SOCK_STREAM, 0); /* TCP socket */
socket(PF_INET6, SOCK_DGRAM,  0); /* UDP socket */
```

Applications

- IPv4 code

```
struct sockaddr_in addr;

socklen_t          addrlen = sizeof(addr);

/*    fill addr structure using an IPv4 address before calling socket
    function*/

bind(sockfd,(struct sockaddr *)&addr, addrlen);
```

- IPv6 code

```
struct sockaddr_in6 addr;

socklen_t           addrlen = sizeof(addr);

/*    fill addr structure using an IPv6 address before calling socket
    function*/

bind(sockfd,(struct sockaddr *)&addr, addrlen);
```

- Portable code

```
struct sockaddr_storage addr;

socklen_t               addrlen;

/*    fill addr structure using an IPv4/IPv6 address and fill addrlen before
    calling socket function*/

bind(sockfd,(struct sockaddr *)&addr, addrlen);
```

Applications

- IPv4 code

```
struct sockaddr_in addr;
socklen_t          addrlen = sizeof(addr);
accept(sockfd,(struct sockaddr *)&addr, &addrlen);
/*    addr structure contains an IPv4 address*/
```

- IPv6 code

```
struct sockaddr_in6 addr;
socklen_t           addrlen = sizeof(addr);
accept(sockfd,(struct sockaddr *)&addr, &addrlen);
/*    addr structure contains an IPv4 address*/
```

- Portable code

```
struct sockaddr_storage addr;
socklen_t               addrlen = sizeof(addr);
accept(sockfd,(struct sockaddr *)&addr, &addrlen);
/*    addr structure contains an IPv4/IPv6 address   addrlen contains the
   size of the addr structure returned */
```

Applications

- Conversion of presentation

```
int inet_pton(int family, const char *src, void *dst);

/*   From IPv4/IPv6 binary to presentation*/

const char *inet_ntop(int family, const void *src,
   char *dst, size_t cnt);
```

- IPv4 example

```
struct sockaddr_in addr;char *straddr;memset(&addr, 0, sizeof(addr));

addr.sin_family = AF_INET;        // familiy

addr.sin_port = htons(MYPORT);    // port, network byte order

/*   from text to binary representation*/

inet_aton("138.4.2.10", &(addr.sin_addr));

/*   from binary to text representation*/

straddr = inet_ntoa(addr.sin_addr);
```

Applications

```
struct sockaddr_in6 addr;

char straddr[INET6_ADDRSTRLEN];

memset(&addr, 0, sizeof(addr));

addr.sin6_family = AF_INET6;          // family

addr.sin6_port = htons(MYPORT);     // port, networt byte
   order

/*   from presentation to binary representation*/

inet_pton(AF_INET6, "2001:720:1500:1::a100",            &
   (addr.sin6_addr));

/*   from binary representation to presentation*/

inet_ntop(AF_INET6, &addr.sin6_addr, straddr,
   sizeof(straddr));
```

Standardization

Standardization

- IPv6 Standardized by the IETF
- Three working groups handles IPv6
  - V6ops
    - Operational issues
    - Migration from v4 to v6 - to some extent
  - IPv6
    - Protocol development
    - Architecture
  - softwires
    - Develop tunnelling protocols for the transition scenarios
  - Also some work done by the IAB
    - Address architecture and address allocations policy

v6ops

- Standardization
  - IPv4 Address surveys
    - Survey of which RFCs that contains dependencies on IPv4 addresses and 32-bit address formats, or IPv4 header format
    - RFC3793, RFC3796, RFCRFC3795, RFC3794, RFC3792, RFC3791, RFC3790, RFC3789
  - Transitions scenarios for
    - 3GPP networks RFC3574, RFC4215
    - Unmanaged network RFC3570, RFC3904
  - Security considerations for 6to4, RFC3964
  - Application aspects of IPv6 transition, RFC4038

v6ops

- Scenarios and analysis for introducing IPv6 in ISP networks, RFC4029
- IPv6 Enterprise network scenarios, RFC4057
- Renumbering without a flag day, RFC4192
- Basic transition mechanisms for IPv6 hosts and routers, RFC4213
- Using VLANs for IPv4/IPv6 co-existence in entrprise networks, RFC4554
- ISP IPv6 deployment scenarios in broadband access networks RFC4779

v6ops

- Work in progress
  - IPv6 Neighbor Discovery On-Link assumption Considered Harmful
  - Reasons to move NAT-PT to Experimental
  - Enterprise network analysis
  - Local network protection for IPv6
  - IPv6 Transition/Co-existence security considerations
  - Using IPs-Sec to secure IPv6-in-IPv4 tunnels
  - Recommendations for filtering ICMPv6 in firewalls

v6ops

- Work in progress cont..
  - IPv6 deployment in 802.16 networks
  - IPv6 unicast address assignment considerations
  - IPv6 implementations for network scanning
  - IPv6 campus transition scenario
  - Requirements for distributing RFC3484 policy
  - Problems with muliple prefixes and RFC3484

IPv6

- Standardized
  - An Architecture for IPv6 address allocation RFC1887
  - DNS Extensions to support IPV6 RFC1886
  - Path MTU discovery for IPV6 RFC1981
  - OSI NSAPs and IPV6 RFC1888
  - IPV6 Multicast address assignments RFC2375

IPv6

- Standardized
  - Neighbour Discovery for IPv6 RFC2461
  - IPv6 stateless address autoconfiguration RC2462
  - ICMPv6 RFC4443
  - IPv6 over…
  - Default address selection for IPv6 RFC3484
  - Basic socket interface extensions for IPv6 RFC3493
  - IPV6 addressing architecture RC3513
  - IPv6 Global unicast address format RFC3587
  - Deprecating Site Local addresses RFC3879

IPv6

- Standardized
  - Default router preference and more specific routes RFC4191
    - Modifying of router advertisement o support multiple exits
  - IPv6 host to router loadsharing RFC4311
  - Linked scoped IPv6 multicast addresses RFC4489
  - IPv6 MIB RFC4293
  - IPv6 node requirements RFC4294
    - List requirements that a IPv6 host/router needs to meet
  - IPv6 scoped address architecture RFC4007
  - Optimistic DAD RFC4429

IPv6

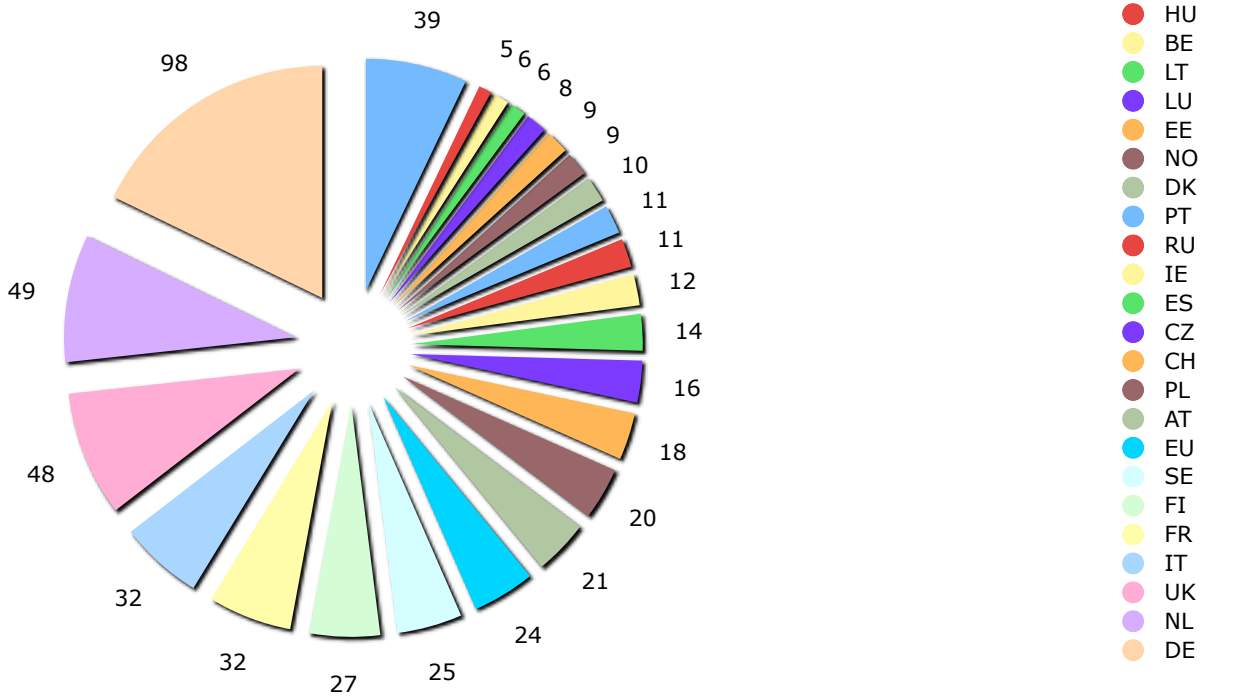
- Work in progress
  - Privacy extensions for stateless address autoconfiguration
    - Updating of RFC3041
  - IPv6 over PPP
    - Updating RFC2472
  - IPv6 Neighbour discovery
    - Updating of RFC2461
  - Stateless address autoconfiguraiton
    - Updating of RFC2462

Is this any good?

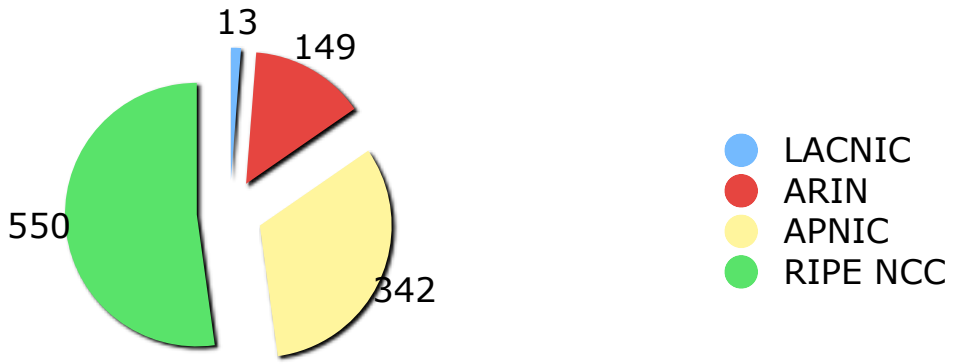
Is this any good?

- It is now more than 10 years ago since the decision to adopt IPv6 as the IPng protocol was made
- It is still not as widely used as IPv4
- When IPv6 was designed and created, it was under the assumption that we would one day turn off IPv6
  - Today most talk about keeping both
- Many of them who created competing proposals to IPv6 now mean that IPv6 is a failure
- However the expectations have in most cases been unrealistic
  - But the presentations of the benefits have not helped either...

Where are we today?



Where are we today?



Where are we today?

```
BGP router identifier 194.15.141.33, local AS number 3220
BGP table version is 40897, main routing table version 40897
755 network entries and 1359 paths using 198263 bytes of memory
1170 BGP path attribute entries using 70260 bytes of memory
1144 BGP AS-PATH entries using 34616 bytes of memory
96 BGP community entries using 3788 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP activity 3479/1237972 prefixes, 8561/7202 paths, scan
interval 60 secs
```

What does the path forward look like?

- Migration or systems
  - TLD name-servers have started
  - The root-servers are running tests
    - i.root-servers.net : 2001:7fe::53
  - Most OS:es, mail-clients and servers support v6
  - Web-browsers have turned out to be more of a problem
- Education and understanding of the differences in the systems are needed
  - Most of the time it is “only” configuration

What does the path forward look like?

- End-users connections must be migrated
  - The systems at the end-users as well as their networks need to be migrated
- As we have seen there are several methods to do so, but key components are
  - Teredo (Windows XP SP2)
  - New code to all DSL modems
  - IPv6 in middleboxes/Firewalls

What does the path forward look like?

- Previously it was assumed the the core networks would be the last bit to migrate
  - Today it seems as if the first users to migrate are the end-users
  - But their infrastructure might well be the last bit to be migrated
- It would be easier to migrate the access network first, but that requires IPv6 support in products, investments also from the end-users
- The transition scenarios documents from the v6ops WG are really good, and worth to read
  - They do give a lot of tips and advice

How do I get IPv6 today?

- Ask your provider!
  - They might give away a tunnel for testing
  - It might even be a commercial service
- Get a tunnel from the SixXS project
  - http://www.sixxs.net

When can we turn off IPv4?

- This most likely never happen
- IPv6 only networks are however being built and used
  - For specific purposes
  - Can still reach IPv4 if needed
- For the future it will be dual-stack systems
  - At least within a given network

Is IPv6 being deployed?

- Yes, mostly in Asia
  - Not only due to shortage of address space
  - New products are also being developed around IPv6
  - Seen as a growth factor
- 3GPP/3GPP2 also looks at IPv6 for product development
  - Pushing for standards
- American military are one of the more driven users
  - Have very interesting hands-on initiatives
  - To some extent driven by the same motivations as in Asia

Is any IPv6 being deployed?

- Several large allocations recently
  - TeliaSonera, DTAG, etc
- Several large GSM/3G networks are in test phases

What stops more from adopting?

- Lack of need
  - Costs for upgrades
  - A “mature market only” problem
- No real shortage of addresses yet
  - NAT has given us some breathing room
  - But NAT has also created several new problems
  - NAT can’t take us much further, and it is causing real problems

What stops more from adopting?

- The providers sees little or no market (and not extra revenues)
  - They can not charge extra for IPv6 services
  - Investments
  - Some have started though
- Protocols and functionality around IPv6 are still fairly moving targets
  - Although IPv6 it self is fairly stable
  - Applications might have to be rewritten
  - Internal corporate software

What stops more from adopting?

- Multihoming
  - The need of redundancy is still as large
  - The lack of a solution is probably seen as one of the larger barriers
  - Even if shim6 was done today, it would take years to implement and deploy

Conclusions

- IPv6 is here
  - It happens, but it happens slowly
- Everyone can today have IPv6 on their computers, if the really wanted to
  - Even with no help from your ISP
- The question is “When are *YOU* going to migrate to IPv6?”
  - Probably depends on need and cost

Conclusions

- IPv6 supporting protocols and features are still being developed
  - Several of these in them selves might be a reason for the migration
- Applications are fairly easy to migrate

Kurt Erik Lindqvist

Internet Technology Advisors

kurtis@inettech.se

http://www.inettech.se