

Introduction to IPv6 protocol

*2nd South Eastern European 6DISS Workshop
Plovdiv, Bulgaria
27-29 June 2007*

*Athanassios Liakopoulos
(aliako@grnet.gr)*



Copy ...Rights

- *This slide set is the ownership of the 6DISS project via its partners*
- *The Powerpoint version of this material may be reused and modified only with written authorization*
- *Using part of this material must mention 6DISS courtesy*
- *PDF files are available from www.6diss.org*
- *Looking for a contact ?*
 - *Mail to : martin.potts@martel-consulting.ch*
 - *Or helpdesk@6diss.org*



Outline

- IPv6 Headers
- Addressing
- Associated features & protocols
- Enabling IPv6
- Transition mechanisms



IPv6 Header

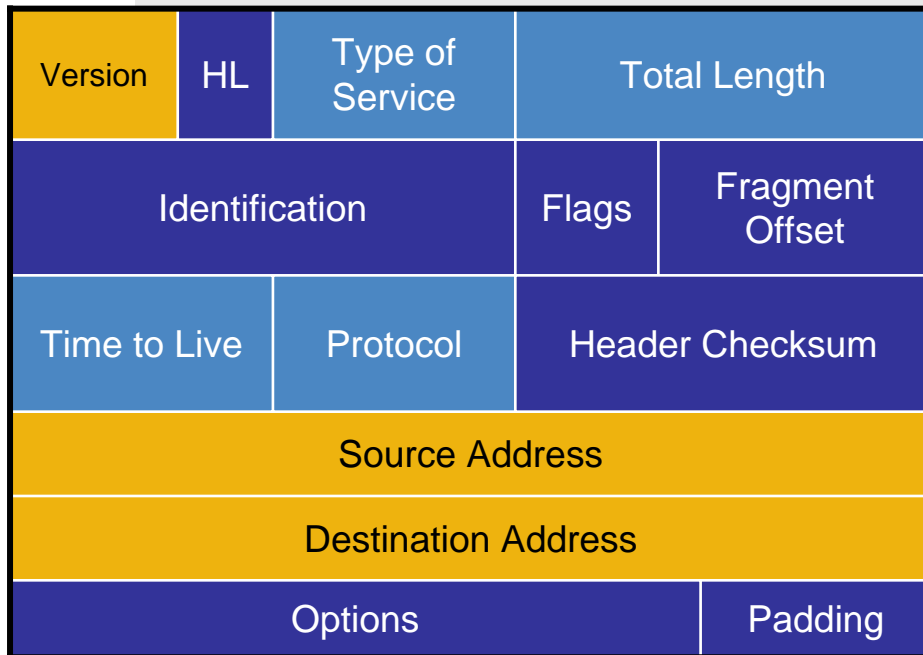
- The IPv6 header is redesigned.
- Minimize header overhead and reduce the header process for the majority of the packets.
- Less essential and optional fields are moved to extension headers

IPv6 and IPv4 headers are not *interoperable!*

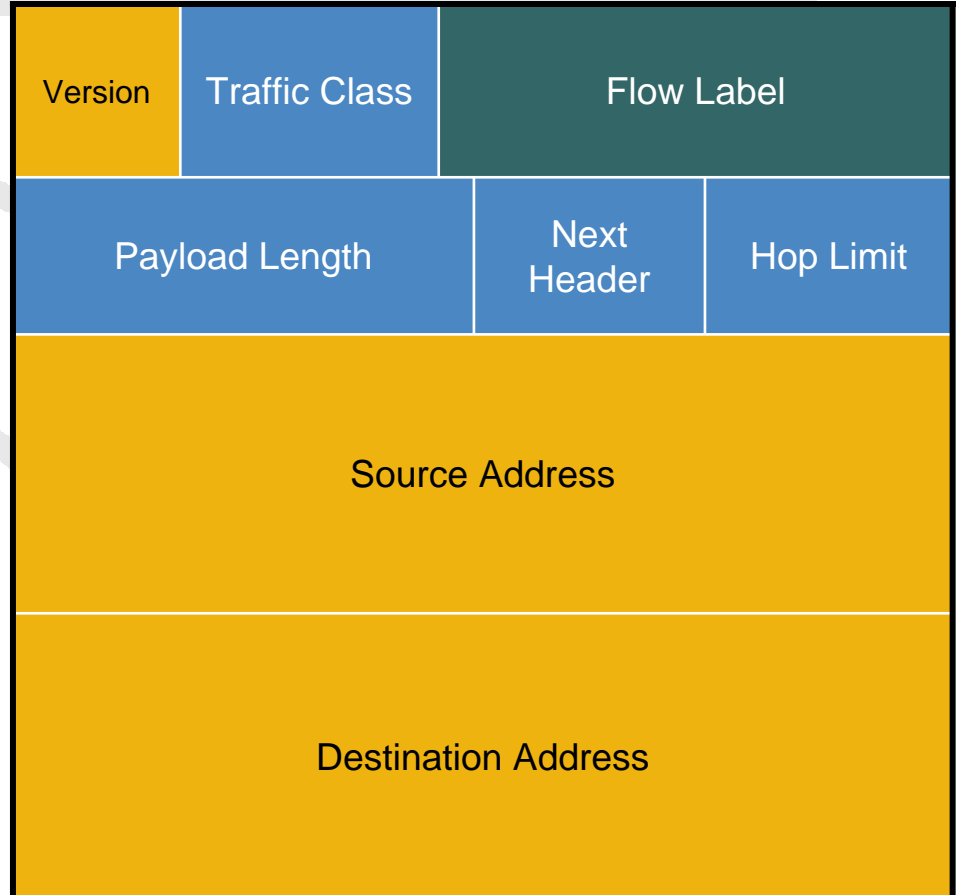


IPv4 and IPv6 Header Comparison

IPv4 Header



IPv6 Header



- Field's Name Kept from IPv4 to IPv6
- Fields Not Kept in IPv6
- Name and Position Changed in IPv6
- New Field in IPv6



IPv4 and IPv6 Header Comparison

- **Version:** a 4-bit field that contains the number 6 instead of 4

IPv6 Header

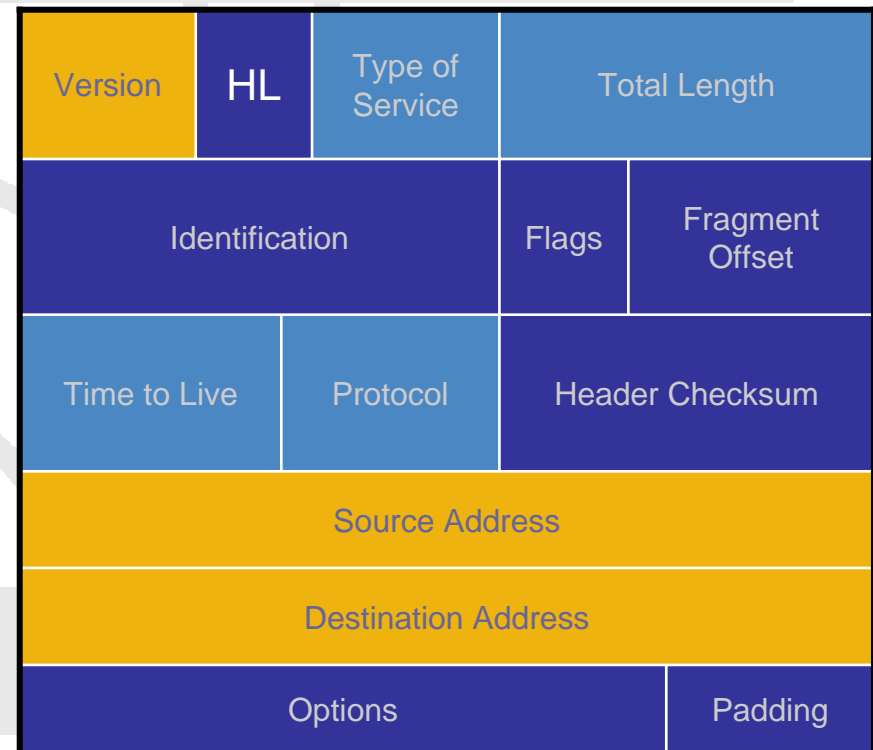


IPv4 and IPv6 Header Comparison

Fields Removed

- Header length: IPv6 has a fixed header length (40 bytes)

IPv4 Header



IPv4 and IPv6 Header Comparison

Fields Removed

- **Fragmentation:** In IPv6, routers do not do fragmentation
- If a sending host wants to do fragmentation, it will do it through extension headers

IPv4 Header

Version	HL	Type of Service	Total Length	
Identification			Flags	Fragment Offset
Time to Live	Protocol		Header Checksum	
Source Address				
Destination Address				
Options				Padding



IPv4 and IPv6 Header Comparison

Fields Removed

- **Identification:** used to identify the datagram from the source
- No fragmentation is done in IPv6 so no need for **identification**, also no need for **flags**

IPv4 Header

Version	HL	Type of Service	Total Length	
Identification			Flags	Fragment Offset
Time to Live	Protocol		Header Checksum	
Source Address				
Destination Address				
Options				Padding

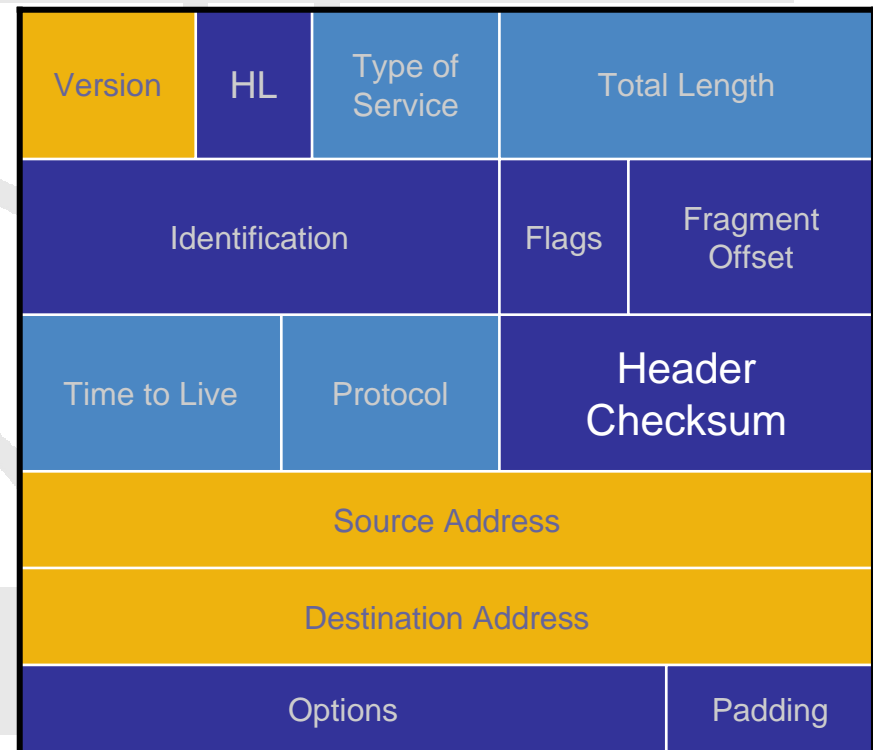


IPv4 and IPv6 Header Comparison

Fields Removed

- **Checksum** not needed because both media access and upper layer protocol (UDP and TCP) have the checksum; IP is best-effort, plus removing checksum helps expedite packet processing

IPv4 Header

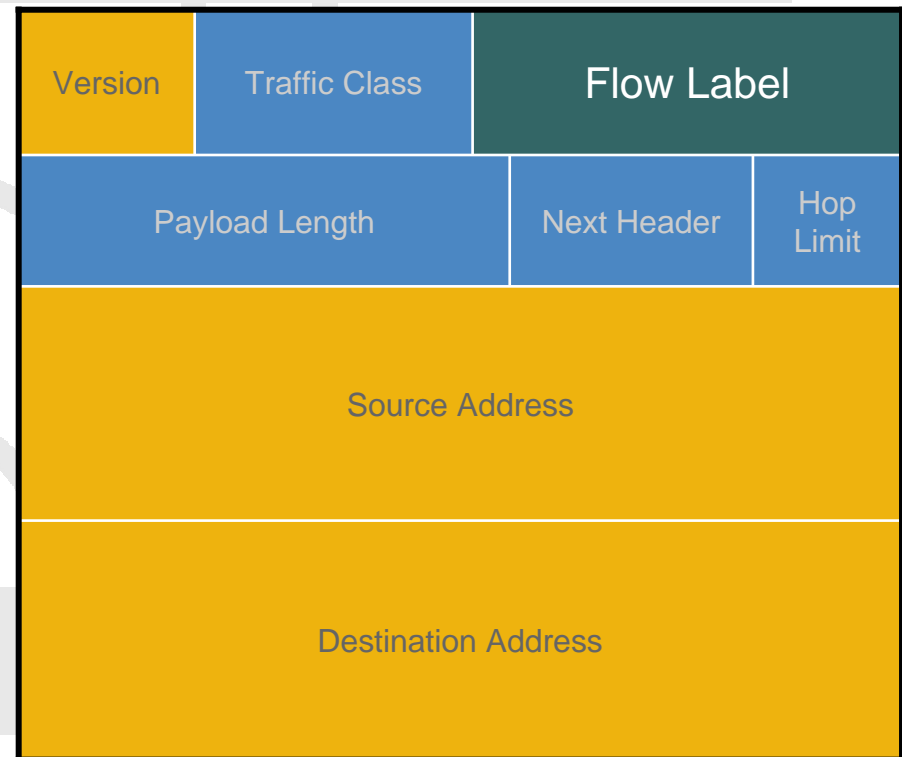


IPv4 and IPv6 Header Comparison

Fields Added

- 20-bit flow label field to identify specific flows needing special QoS
 - Each source chooses its own flow label values; routers use source address + flow label to identify distinct flows
 - Flow label value of 0 used when no special QoS requested (the common case today)
 - Fragmentation or encryption is not anymore problem, as in IPv4.
 - Without the flow label, the classifier must use transport next header value and port numbers
 - Less efficient (need to parse the option headers)
 - May be impossible (fragmentation or IPsec ESP)
- Flow Label (RFC 3697)

IPv6 Header



IPv4 and IPv6 Header Comparison

Fields Renamed

- **Traffic class:** an 8-bit field that is similar to the **TOS field** in IPv4
- It tags the packet with a traffic class that can be used in differentiated services
- Provides the *same* functionality as the ***type of service*** field in the IPv4 header.

IPv6 Header



IPv4 and IPv6 Header Comparison

Fields Renamed

- **Payload length:** this is similar to the **total length** in IPv4, except it does not include the 40-byte header

IPv6 Header



IPv4 and IPv6 Header Comparison

Fields Renamed

- **Hop limit:** like **TTL field**, decrements by one for each router

IPv6 Header



IPv4 and IPv6 Header Comparison

Fields Renamed

- **Next header:** similar to the **protocol field** in IPv4
- The value in this field tells you what type of information follows
 - E.g. TCP, UDP, extension header

IPv6 Header



IPv6 header fields #2

- Flow Label (RFC 3697)
 - A *20-bit* field, selected by the source and never modified in the network.
 - Fragmentation or encryption is not anymore problem, as in IPv4.
 - Without the flow label, the classifier must use transport next header value and port numbers
 - Less efficient (need to parse the option headers)
 - May be impossible (fragmentation or IPsec ESP)

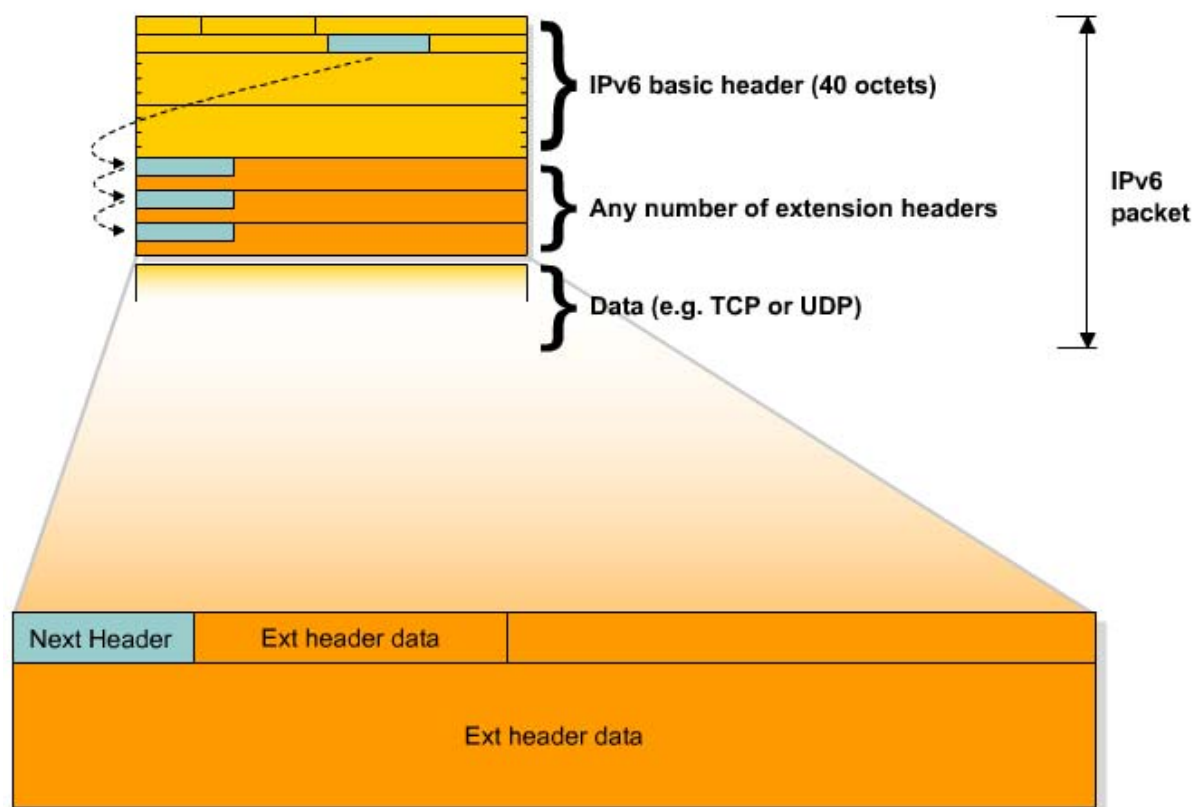


IPv6 header fields #3

- Payload Length
 - For Jumbograms (65,535 octets long) set PL to 0
- Hop Limit
- Next Header
 - Allows to use extension headers.



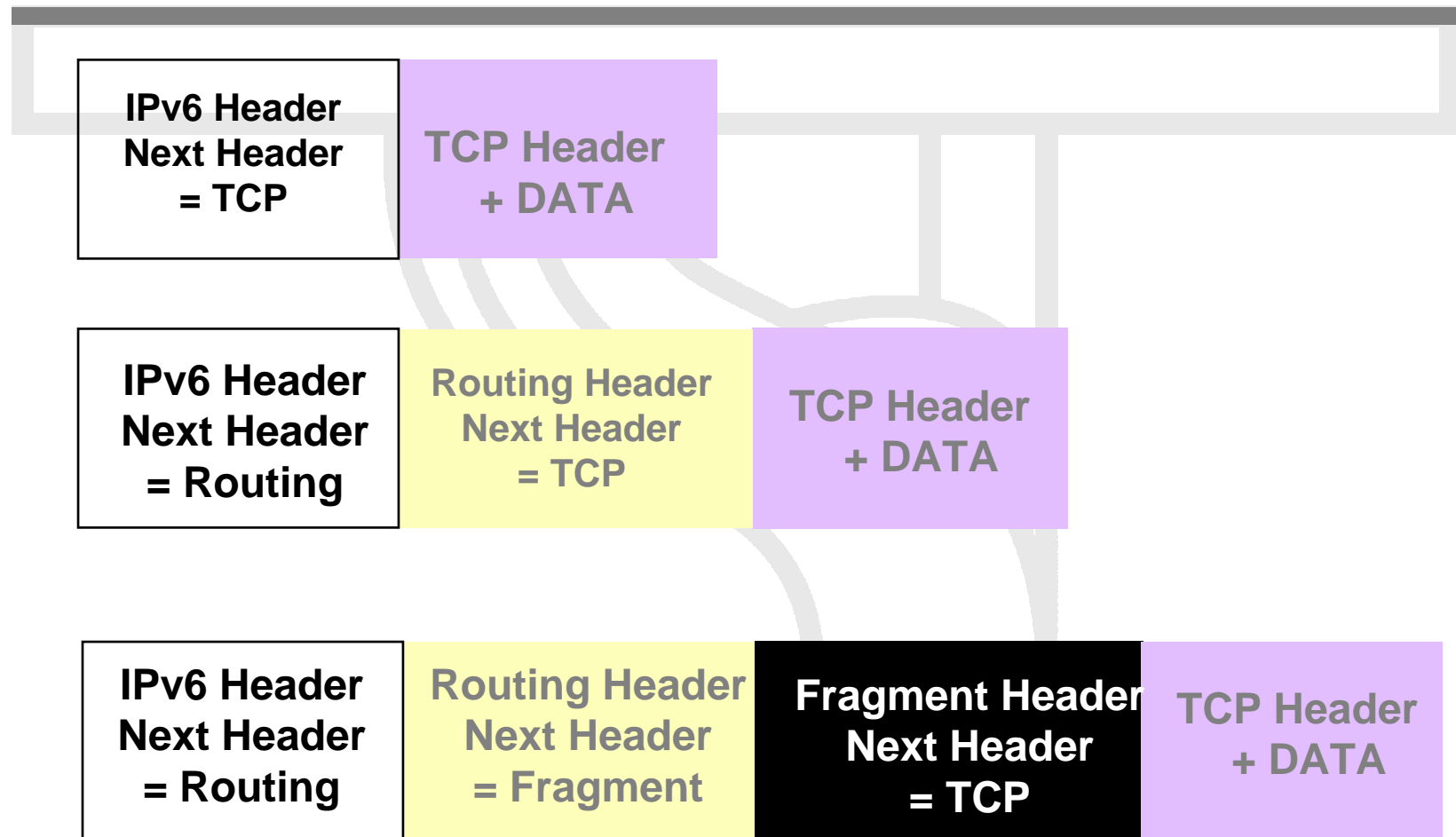
Extension Headers (RFC2460)



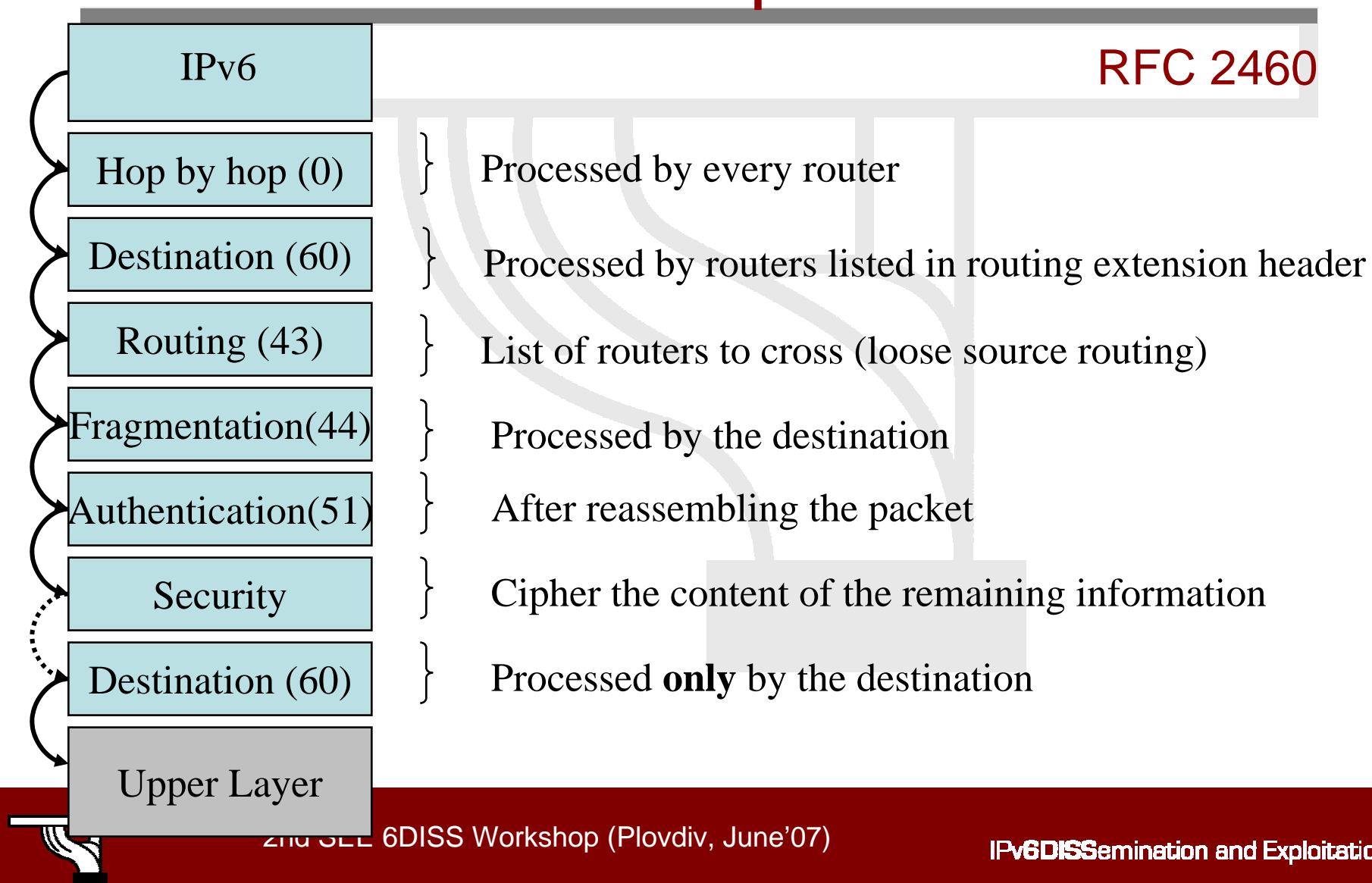
- Processed only by node identified in IPv6 Destination Address field => much lower overhead than IPv4 options
 - exception: Hop-by-Hop Options header
- Eliminated IPv4's 40-octet limit on options
 - In IPv6, limit is total packet size, or Path MTU in some cases



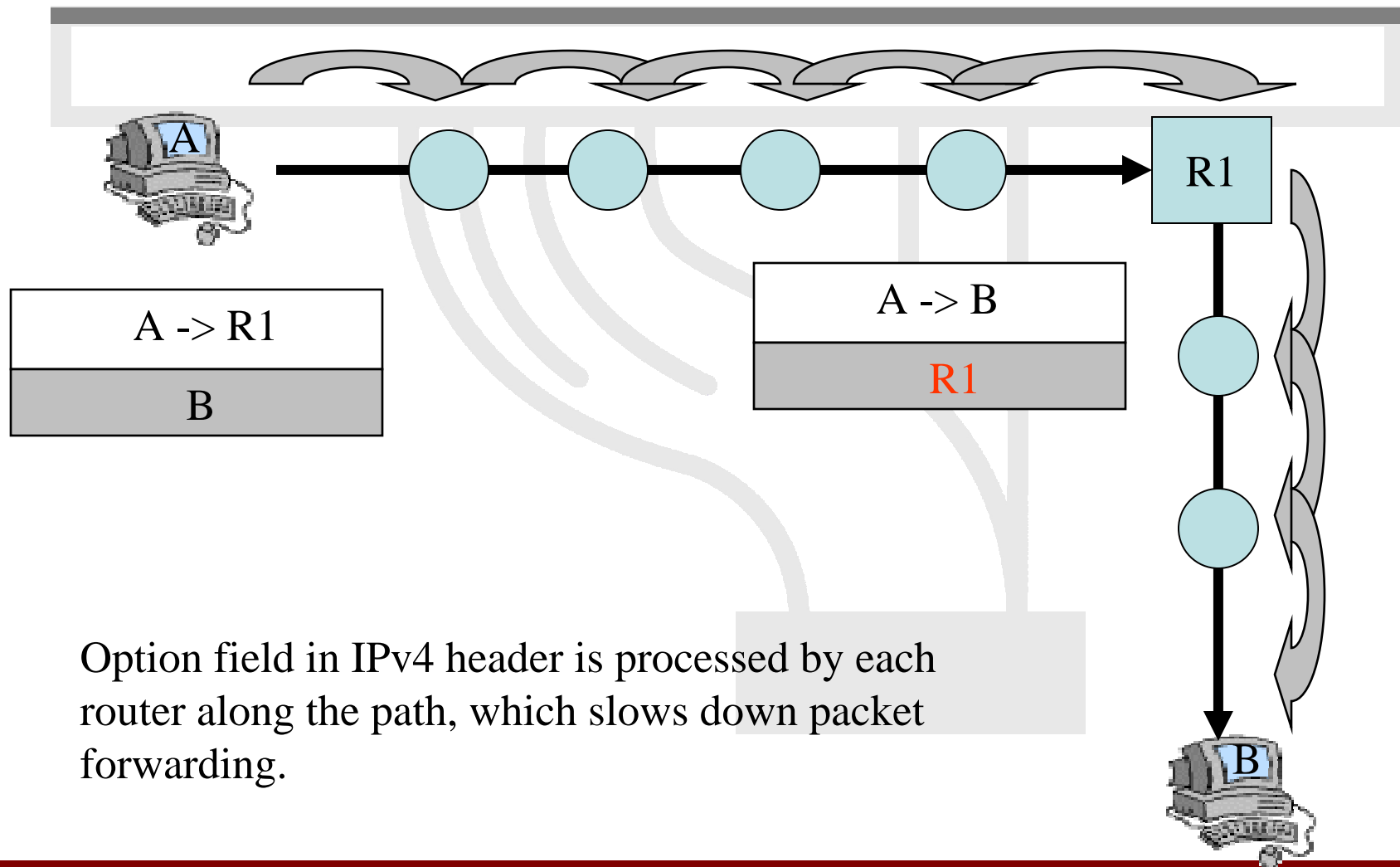
IPv6 extension headers



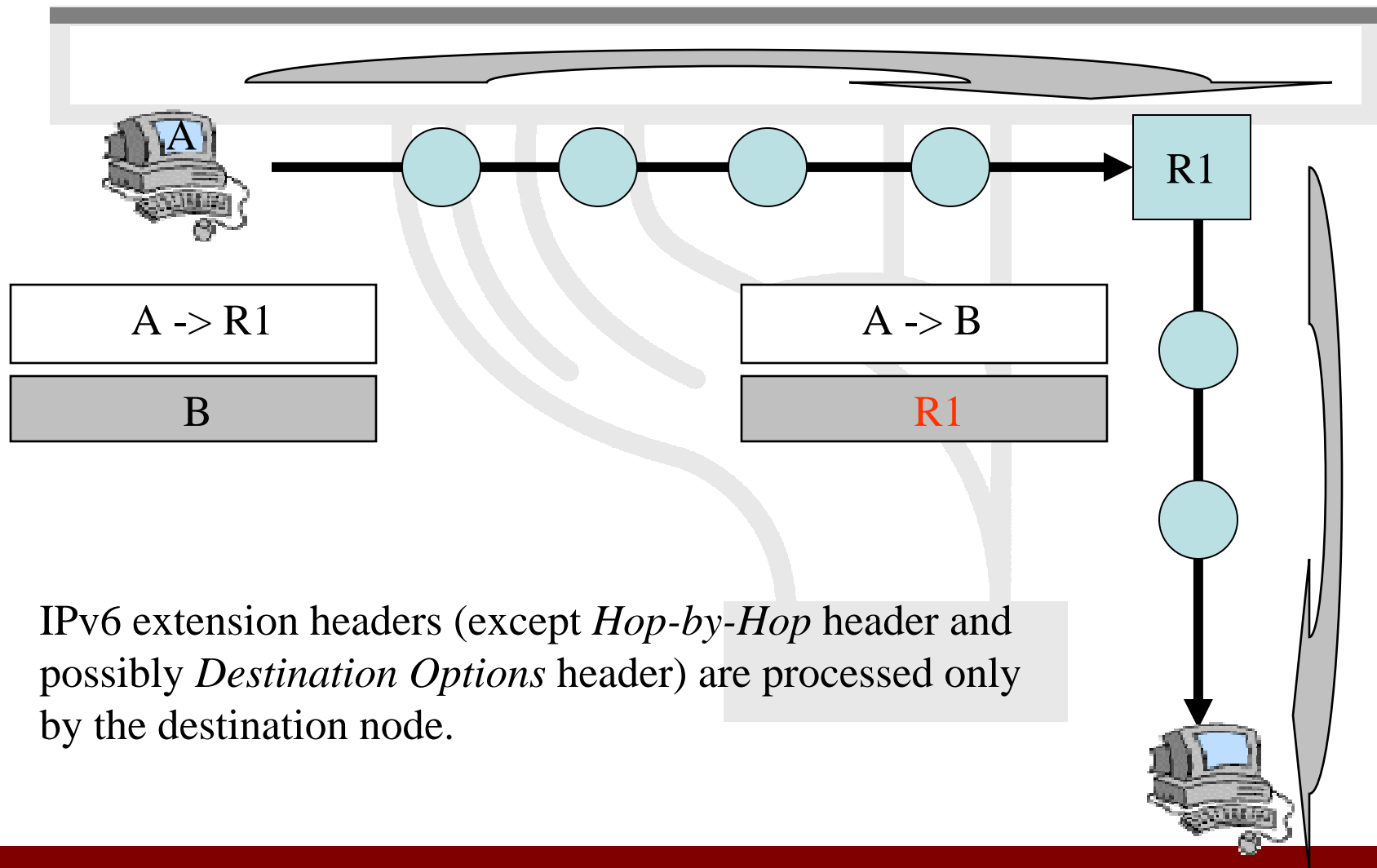
IPv6 extension headers: order is important



IPv4 header options : processing



IPv6 extension headers: processing



Outline

- IPv6 Headers
- Addressing
- Associated features & protocols
- Enabling IPv6
- Transition mechanisms



Addressing scheme

- IPv6 addressing (RFC 3513)
 - 128-bit long addresses
 - 340.282.366.920.938.463.463.374.607.431.768.211.456
 - Hexadecimal representation
 - Interfaces have several IPv6 addresses
 - CIDR principles *[address prefix / prefix length]*
 - 2001:648::/32
 - Aggregation reduces routing table size
- Global Unicast Address format (RFC 3587)
 - Allow hierarchy



Textual Address Format

- Base format (16-byte)

2001:0660:3003:0001:0000:0000:6543:210F

- Compact Format:

2001:660:3003:1::6543:210F

- Literal representation

– [2001:660:3003:2:a00:20ff:fe18:964c]



IPv6 Addresses

-
- | | | |
|----------------------|-----------|-------------------------------------|
| • Loopback | ::1 | • Unicast |
| • Link local | FE80:.... | • Multicast |
| • Site local | FEC0:.... | • Anycast |
| • Global | | |
| – Official: | 2001:.... | |
| – 6bone: | 3FFE:.... | |
| • IPv4 mapped | | } specific to IPv4/IPv6 integration |
| • 6to4 | 2002:.... | |
| • Multicast | FF... | |



IPv6 Addresses

- **Loopback address representation**
 - 0:0:0:0:0:0:0:1=> ::1
 - Same as 127.0.0.1 in IPv4
 - Identifies self
- **Unspecified address representation**
 - 0:0:0:0:0:0:0:0=> ::
 - Used as a placeholder when no address available
 - (Initial DHCP request, Duplicate Address Detection DAD)
- **IPv4 mapped**
 - 0:0:0:0:0::FFFF:IPv4 = ::FFFF:IPv4
 - 0:0:0:0:0:FFFF:192.168.30.1 = ::FFFF:C0A8:1E01
- **IPv4 compatible**
 - 0:0:0:0:0:0:IPv4 = ::IPv4
 - 0:0:0:0:0:0:192.168.30.1 = ::192.168.30.1 = ::C0A8:1E01



IPv6 - Addressing Model

Addresses are assigned to interfaces

change from IPv4 model :

Interface 'expected' to have multiple addresses

Addresses have scope

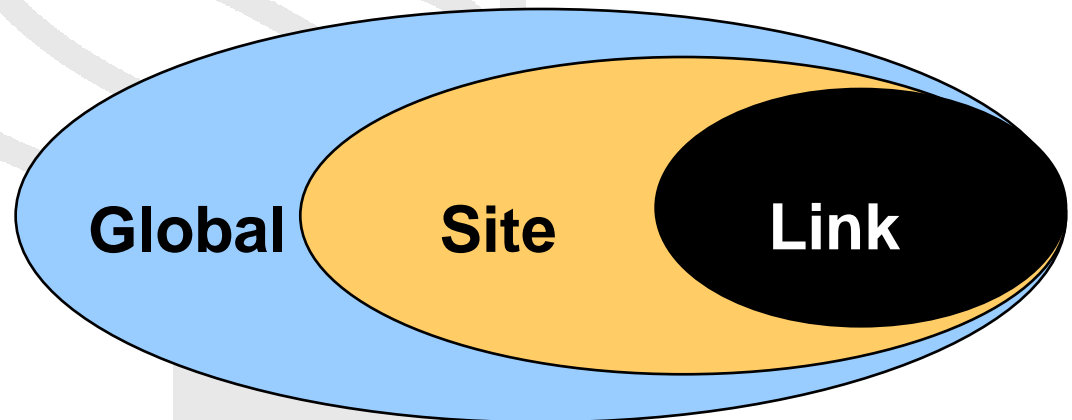
Link Local

Site Local

Global

Addresses have lifetime

Valid and Preferred lifetime

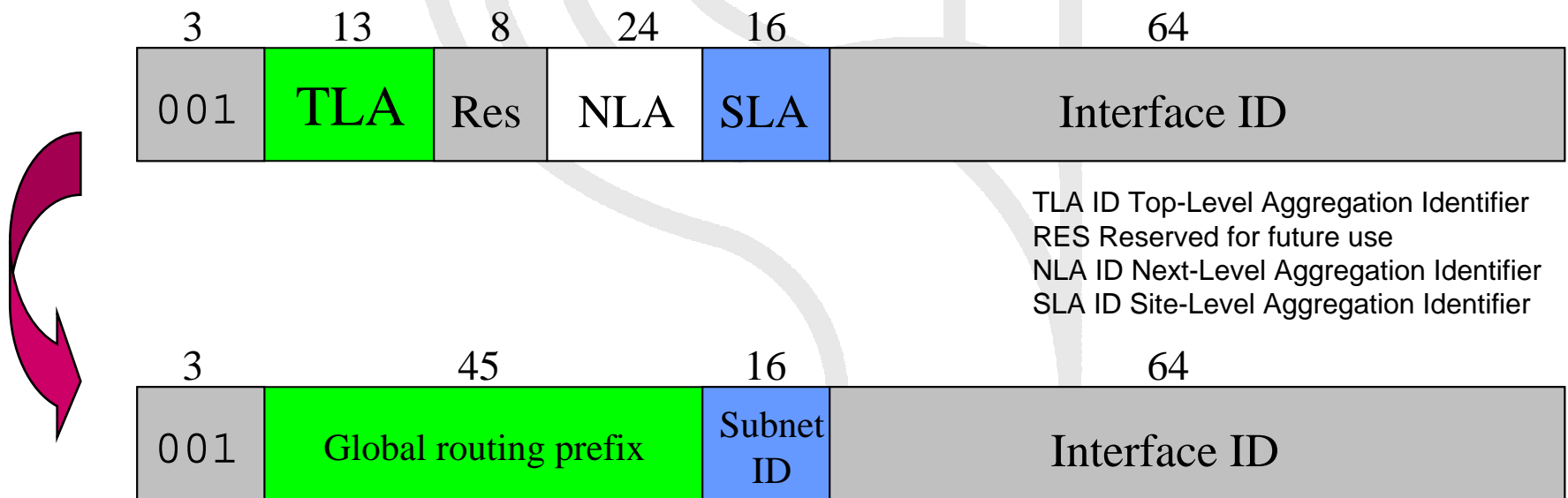


Site-Local Address Deprecated
in RFC 3879 now it is Unique
Local Address (ULA) RFC 4193



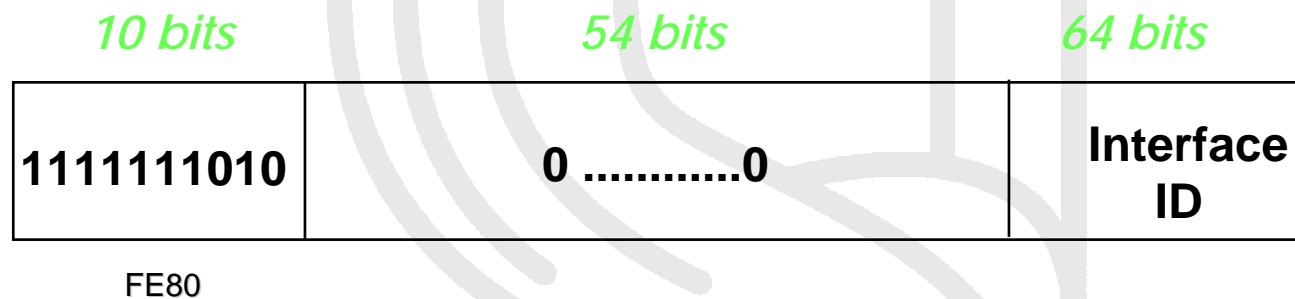
Global Unicast Address Format (RFC 3587)

(obsoletes RFC 2374)

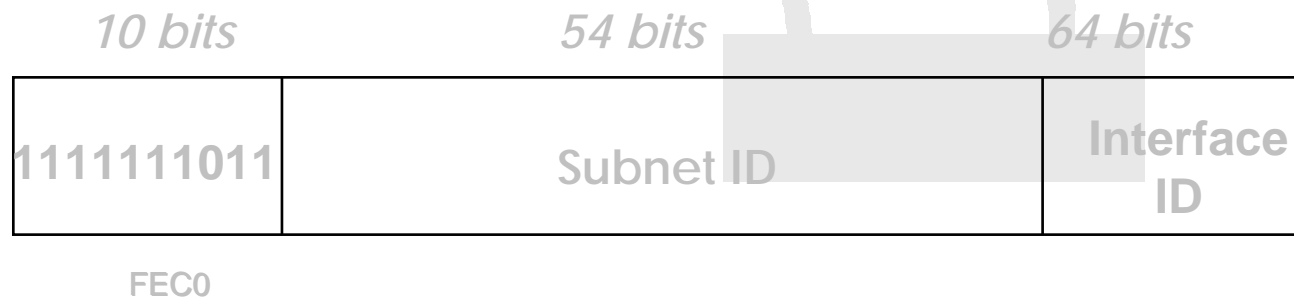


Local Addresses

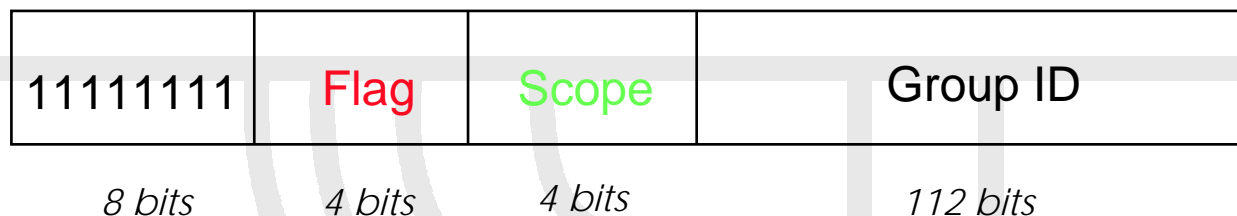
Link-local



Site-local (in the process of being deprecated)



Multicast Addresses



Flag bits: 0 **R** **P** **T**

T = 0 *permanent addresses (managed by IANA)*

T = 1 *transient multicast addresses*

- **P** = 1 *derived from unicast prefix (RFC3306)*
- **R** = 1 *embedded RP addresses (RFC 3956)*

Scope

- 0 : Reserved
- 1 : Interface-local
- 2 : Link-local
- 3 : Subnet-local
- 4 : Admin-local
- 5 : Site-local
- 8 : Organization-local
- E : Global
- F : Reserved



Anycast Addresses (RFC 3513)

- «Anycast addresses allow a packet to be **routed to one of a number** of different nodes all responding to the same address »
- «Anycast addresses are taken from the unicast address spaces (of any scope) and are not syntactically distinguishable from unicast addresses ... it may be assigned to an IPv6 router only.»
- Anycast address ...
 - ... can not be used as a source address of an IPv6 packet
 - ... must be assigned only to routers
- Reserved anycast addresses are defined in *RFC 2526*



Agenda

- Some history and facts ...
- IPv6 Headers
- Addressing
- **Associated features & protocols**
- Enabling IPv6
- Transition mechanisms



Associated features & protocols

- Neighbor Discovery (*ND*) (RFC 2461 DS)
- Auto-configuration :
 - Stateless Address Auto-configuration (RFC 2462 DS)
 - DHCPv6: Dynamic Host Configuration Protocol for IPv6 (RFC 3315 PS)
 - Path MTU (*pMTU*) discovery (RFC 1981 PS)



Associated features & protocols

- MLD (Multicast Listener Discovery) (RFC 2710)
 - Multicast group management over an IPv6 link
 - Based on IGMPv2
 - MLDv2 (equivalent to IGMPv3 in IPv4)
- ICMPv6 (RFC 2463 DS) "Super" Protocol that :
 - Covers ICMP (v4) features (error control, administration, ...)
 - Transports ND messages
 - Transports MLD messages (queries, reports, ...)



Neighbor Discovery

- IPv6 nodes which share the same physical medium (link) use Neighbor Discovery (ND) to:
 - discover their mutual presence
 - determine link-layer addresses of their neighbors
 - find routers
 - maintain neighbors' reachability information (NUD)
 - not directly applicable to NBMA (*Non Broadcast Multi Access*) networks → ND uses multicast for certain services.



Neighbor Discovery #2

- Protocol features:
 - Router discovery
 - Prefix(es) discovery
 - Parameters discovery (link MTU, Max Hop Limit, ...)
 - Address auto-configuration
 - Address resolution
 - Next Hop determination
 - Neighbor Unreachability Detection
 - Duplicate Address Detection
 - Redirect



Neighbor Discovery #3

- ND specifies 5 types of ICMP packets :
 - **Router Advertisement (RA) :**
 - periodic advertisement (of the availability of a router) which contains:
 - list of prefixes used on the link (autoconf)
 - a possible value for Max Hop Limit (TTL of IPv4)
 - value of MTU
 - **Router Solicitation (RS) :**
 - the host needs RA immediately (at boot time)



Neighbor Discovery #4

– **Neighbor Solicitation (NS):**

- to determine the link-layer @ of a neighbor
- or to check its impeachability
- also used to detect duplicate addresses (DAD)

– **Neighbor Advertisement (NA):**

- answer to a NS packet
- to advertise the change of physical address

– **Redirect :**

- Used by a router to inform a host of a better route to a given destination



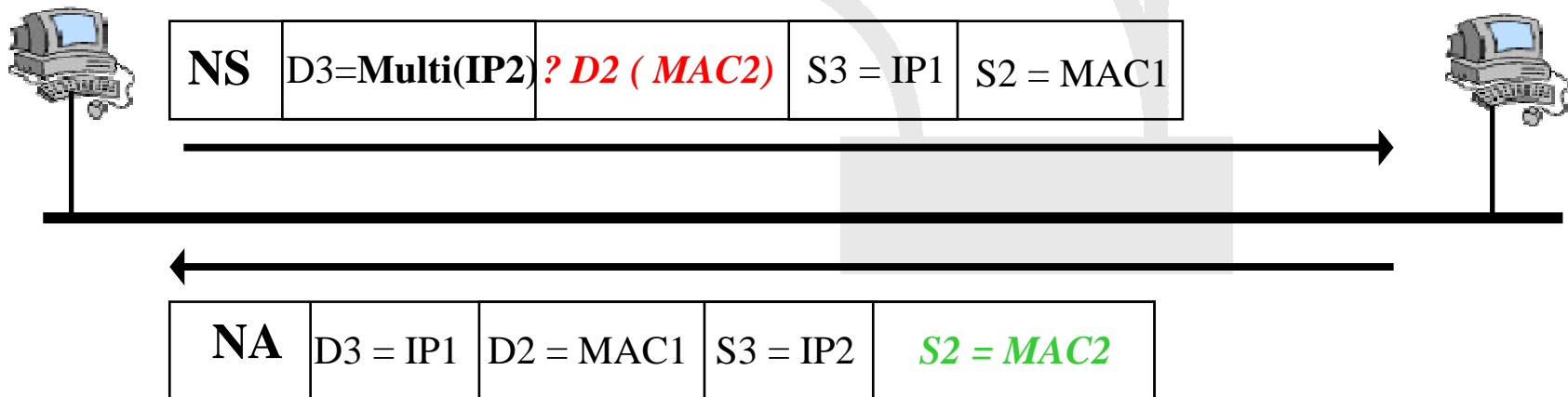
Address Resolution IPv6 with Neighbor Discovery

At boot time, every IPv6 node has to join 2 special multicast groups for each network interface:

- All-nodes multicast group: ff02::1
- Solicited-node multicast group: ff02::1:ffxx:xxxx (derived from the lower 24 bits of the node's address)

H1: IP1, MAC1

H2: IP2, MAC2



Address Resolution with Solicited Multicast Address

- **Concatenation** of the prefix FF02: : 1: FF00: 0/104 with the last 24 bits of the IPv6 address

Example:

- **Dst IPv6 @:** 2001: 0660: 010a: 4002: 4421: 21FF: FE24: 87c1
- **Sol. Mcast @:** FF02: 0000: 0000: 0000: 0000: 0001: FF24: 87c1
- **ethernet:** FF-FF-FF-24-87-c1



Stateless Autoconfiguration

- Described in RFC 2462
- Objective: IPv6 hosts should be *Plug & Play*
- Uses some of the Neighbor Discovery ICMPv6 messages
 - Route advertisement, route solicitation
 - IPv6 does not support ARP and there are no broadcast addresses.

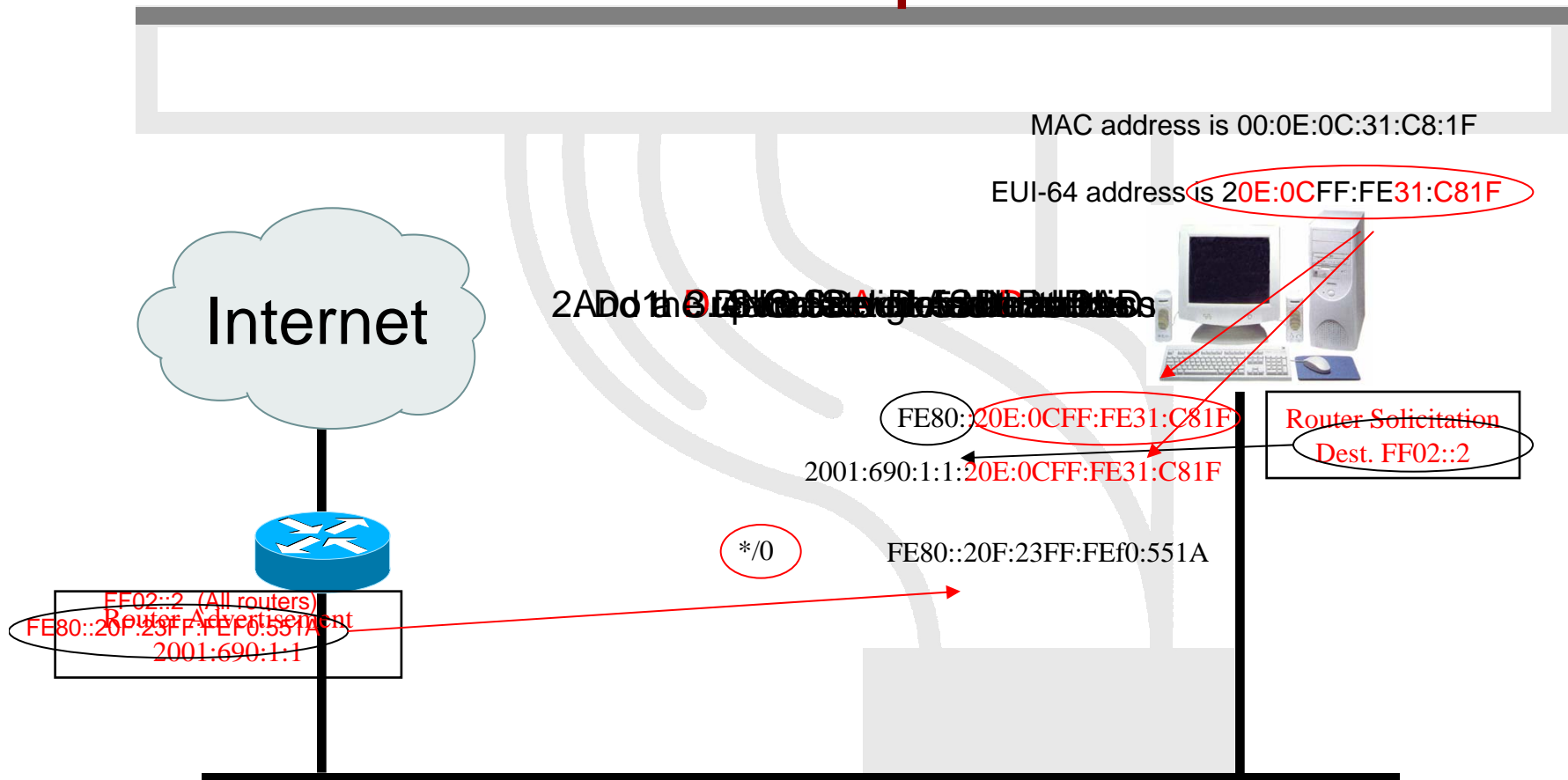


Stateless Autoconfiguration: Procedure

- When booting, a host asks/seeks for some network parameters:
 - IPv6 prefix (-es), default router address (-es), hop limit, (link local) MTU, prefix validity time, etc
- Hosts listen Router Advertisements (RA) messages
 - RA are periodically transmitted by routers or when prompt by hosts
 - If a RA doesn't carry any prefix, the hosts may only configure the default gateway address.
- A host to creates ...
 - a link local address (fe80::/10)
 - a global IPv6 address using:
 - its interface identifier (EUI-64 address)
 - link prefix obtained via Router Advertisement

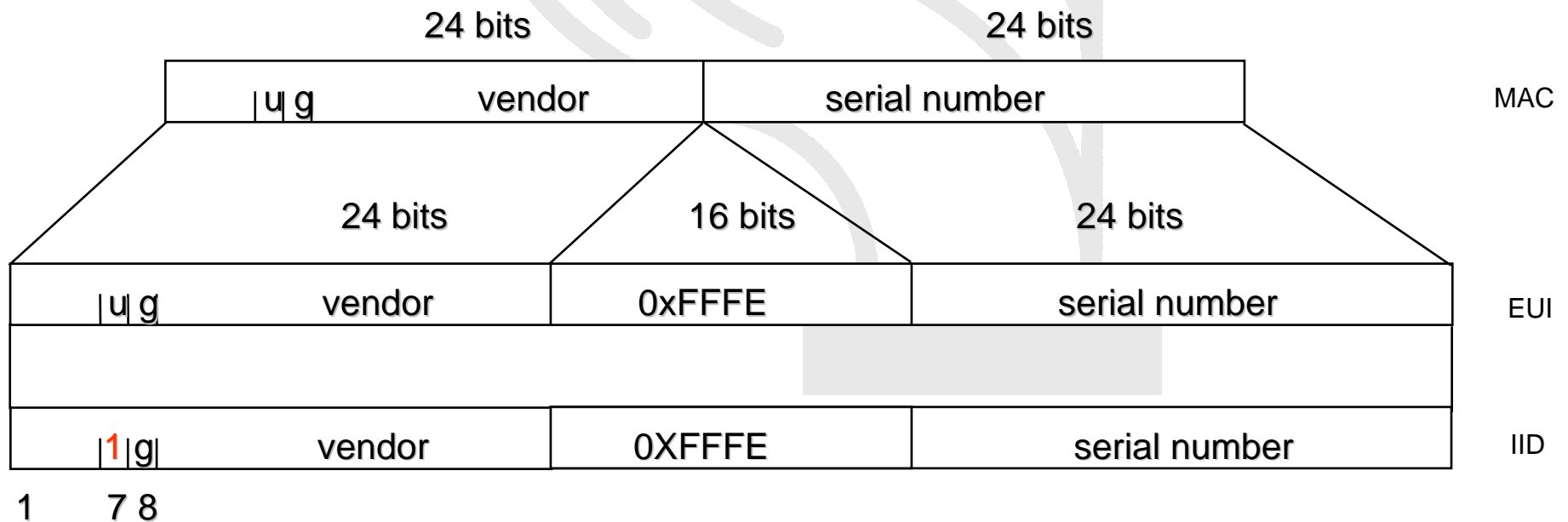


Stateless Autoconfiguration example

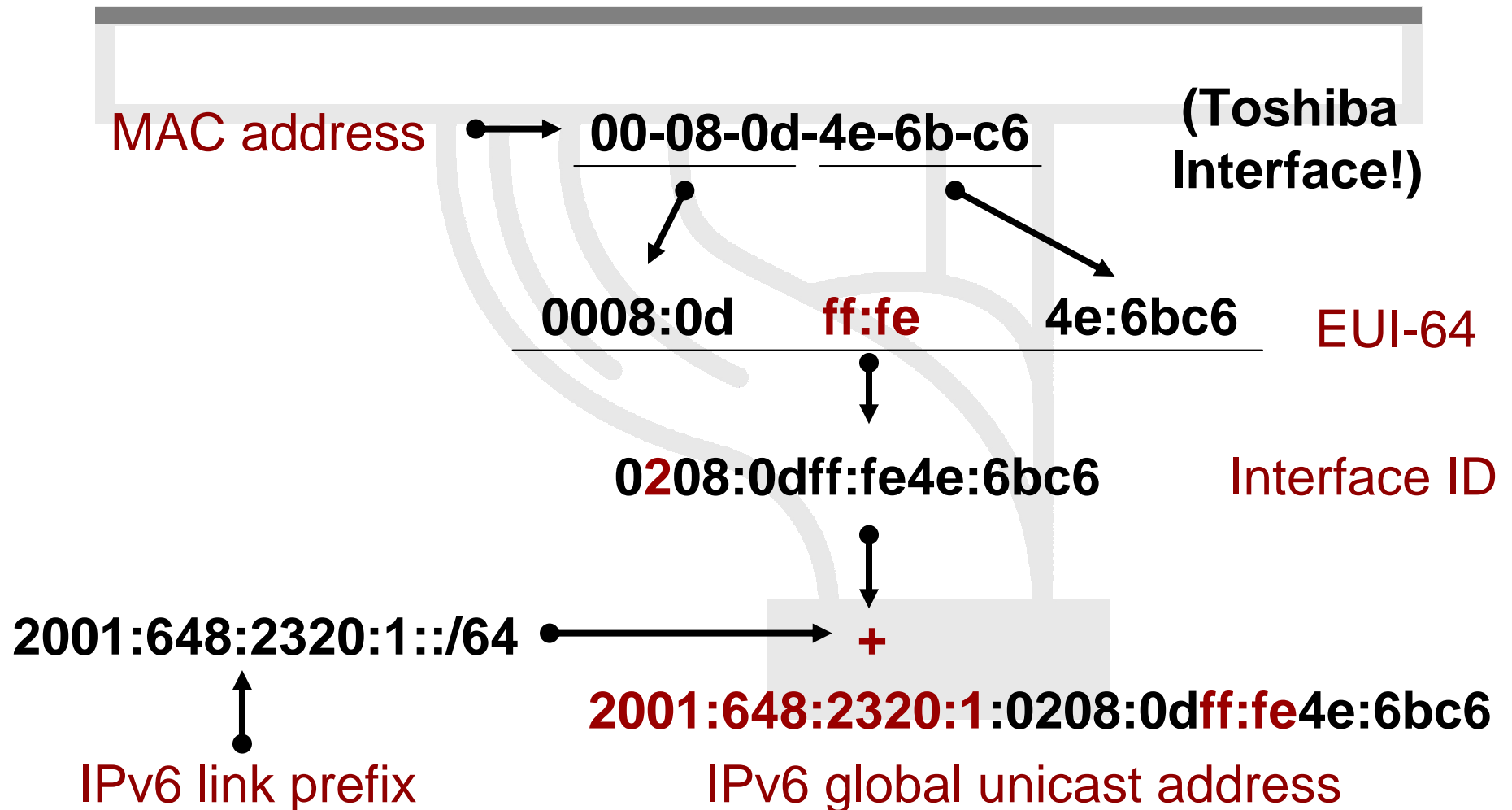


Interface Identifier

- 64 bits to be compatible with IEEE 1394 (FireWire)
- Eases auto-configuration
- IEEE defines the mechanism to create an EUI-64 from IEEE 802 MAC addresses (Ethernet, FDDI)



Interface Identifier: Example



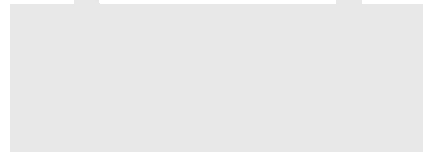
Interface Identifier: Privacy issues

- IEEE 24 bit OUI identify hardware
(<http://standards.ieee.org/regauth/oui/oui.txt>)
- Interface ID can be used to trace a user:
 - The prefix changes, but the interface ID remains the same!
- Privacy extensions (RFC 3041)
 - Possibility to change Interface ID
 - If local storage, use MD5 algorithm. Otherwise, draw a random number.
 - Creates security problem?



Stateless Autoconfiguration: Best practices

- Only routers have to be manually configured
 - Ongoing work on prefix delegation
<http://www.ietf.org/rfc/rfc3633.txt>
- Hosts get automatically an IPv6 address
 - BUT it isn't automatically registered in the DNS!
- Servers should be manually configured!



Stateful autoconfiguration

- Dynamic Host Configuration Protocol for IPv6 (DHCPv6)
 - Defined in RFC 3315
 - Stateful counterpart to IPv6 Stateless Address Autoconfiguration.
- DHCPv6 is used when:
 - no router is found
 - Or if Router advertisement message enable use of DHCP



Stateful autoconfiguration

- DHCPv6 works in a client-server model
 - **Server**
 - Responds to requests from clients
 - Optionally provides the client with:
 - IPv6 addresses
 - Other configuration parameters (DNS servers...)
 - Is listening on multicast addresses:
 - All_DHCP_Relay_Agents_and_Servers (FF02::1:2)
 - All_DHCP_Servers (FF05::1:3)
 - Memorize client's state
 - Provide means for securing access control to network resources



Stateful autoconfiguration

– Client

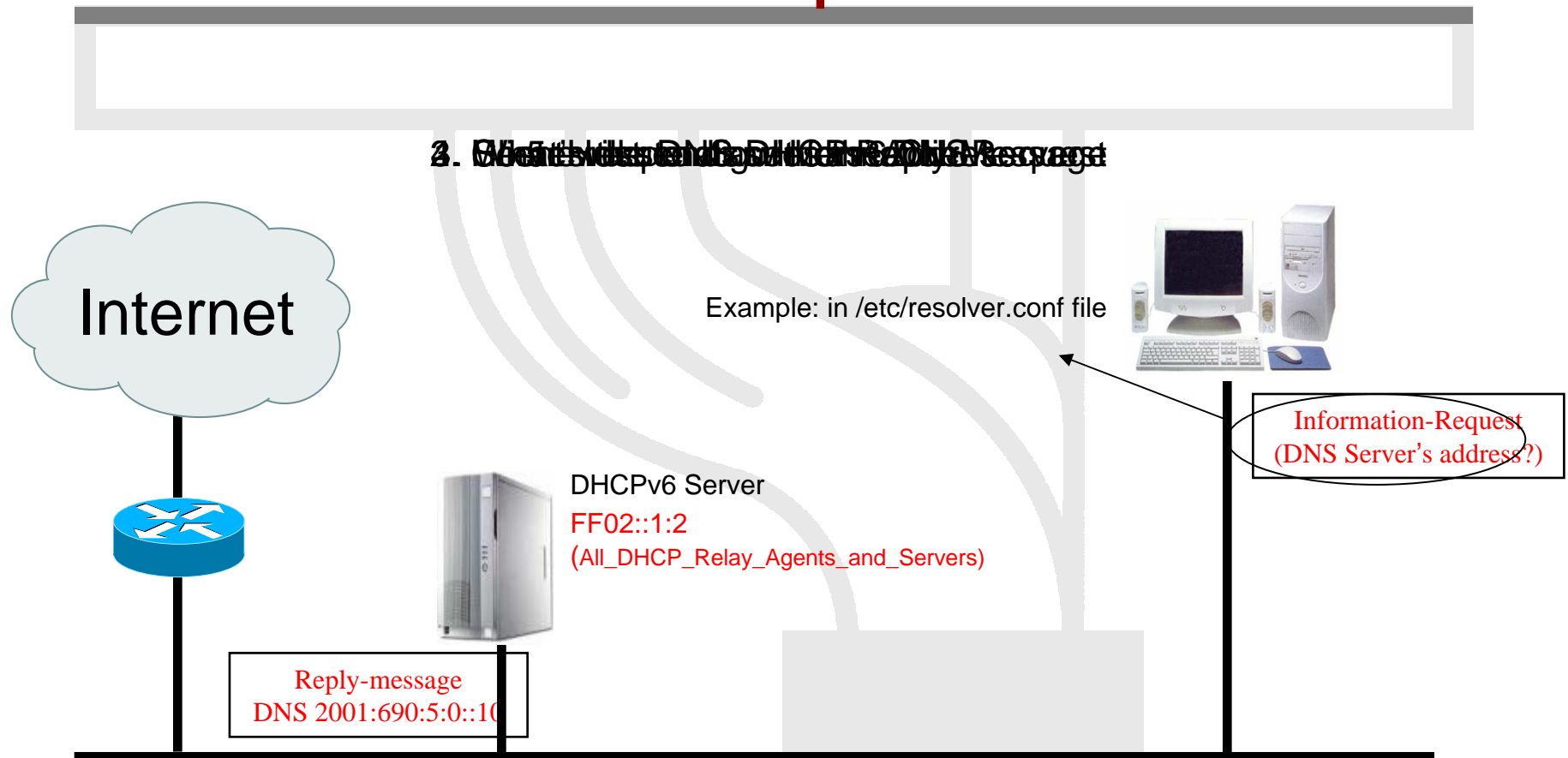
- Initiates requests on a link to obtain configuration parameters
- Use its link local address to connect the server
- Send requests to FF02::1:2 multicast address (All_DHCP_Relay_Agents_and_Servers)

– Relay agent

- Node that acts as an intermediary to deliver DHCP messages between clients and servers
- Is on the same link as the client
- Is listening on multicast addresses:
 - All_DHCP_Relay_Agents_and_Servers (FF02::1:2)



Stateful Autoconfiguration example



Path MTU discovery (RFC 1981)

- Derived from RFC 1191, (IPv4 version of the protocol)
- **Path** : set of links followed by an IPv6 packet between source and destination
- **link MTU** : maximum packet length (bytes) that can be transmitted on a given link without fragmentation
- **Path MTU** (or pMTU) = $\min \{ \text{link MTUs} \}$ for a given path
- Path MTU Discovery = automatic pMTU discovery for a given path

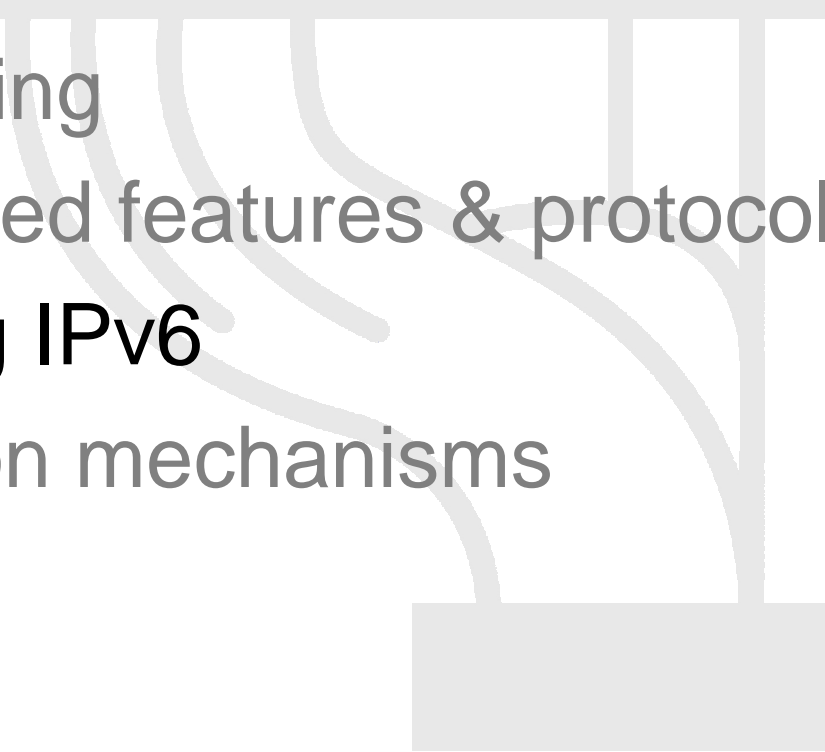


Path MTU discovery (2)

- Protocol operation
 - makes assumption that pMTU = link MTU to reach a neighbor (first hop)
 - if there is an intermediate router such that link MTU < pMTU → it sends an ICMPv6 message: "Packet size Too Large"
 - source reduces pMTU by using information found in the ICMPv6 message
- => Intermediate equipments aren't allowed to perform packet fragmentation



Outline

- IPv6 Headers
 - Addressing
 - Associated features & protocols
 - **Enabling IPv6**
 - Transition mechanisms
- 



IPv6 Support: Windows

- WinXP
 - SP0: Autoconfiguration, tunnels, ISATAP, etc. IPv6 has explicitly to be activated!
 - SP1: GUI installation, `netsh` command line interface
 - SP2: Teredo, firewall, and other additions
- Win2000
 - Only developer edition available
- Windows 95/98/ME
 - No official support



IPv6 install: Windows

- WinXP
 - Execute “`ipv6 install`” at a command prompt (SP0)
 - Add ‘*Microsoft IPv6 Developer Edition*’ component as a new protocol in the Network Connections Control Panel pane (SP1)
 - Add ‘Microsoft TCP/IP version 6’ as a new protocol in the Network Connections Control Panel pane (SP2)



IPv6 commands: WinXP

- Command line interface (netsh):

```
c:\>netsh interface ipv6
```

- Well known (IPv4/6) commands

```
ipconfig, netstat, ping6, tracert6, pathping
```

- IPv6 depreciated command:

```
c:\>ipv6 ?
```

```
usage: ipv6 [-p] [-v] if [ifindex]
```

```
ipv6 [-p] ifcr v6v4 v4src v4dst [nd] [pmlid]
```

```
ipv6 [-p] ifcr 6over4 v4src
```

```
...
```



IPv6 commands: WinXP

- Set an IPv6 address (netsh):

```
set address [interface=]<string>  
[address=]<IPv6 address>  
[[type=]unicast|anycast]  
[[validlifetime=]<integer>|infinite]  
[[preferredlifetime=]<integer>|infinite]  
[[store=]active|persistent]
```

- Delete an IPv6 address (netsh):

```
delete address [interface=]<string>  
[address=]<IPv6 address>  
[[store=]active|persistent]
```



IPv6 commands: WinXP

- Add an route:

```
add route [prefix=]<IPv6 address>/<integer>  
[interface=]<string>  
    [[nexthop=]<IPv6 address>  
[[siteprefixlength=]<integer>  
    [[metric=]<integer>] [[publish=]no|age|yes]  
    [[validlifetime=]<integer>|infinite]  
    [[preferredlifetime=]<integer>|infinite]  
    [[store=]active|persistent]
```

- Show route:

```
show routes [[level=]normal|verbose]  
    [[store=]active|persistent]
```



IPv6 commands: WinXP

- Reset configuration (netsh):

```
renew [[interface=]<string>]
```

- Show DNS server (netsh):

```
show dns [[interface=]string]
```



IPv6 Support: Linux distributions

- Redhat (6.2+), Fedora 1&2, SuSE (7.3+), Debian (2.2+), Mandrake (8.0+), Scientific Linux (3.0+), *BSD, etc
 - Look for IPv6 support at kernel!
- USAGI
 - Collaboration between WIDE, KAME and TAHI in order to improve kernel



IPv6 install: Linux

- Load IPv6 module

```
# modprobe ipv6
```

- Search loaded module

```
# lsmod -w "ipv6"
```

- Check if IPv6 is installed

```
# test -f /proc/net/if_inet6 && echo "OK"
```

- Well known (IPv4/6) commands

```
ifconfig, netstat, ping6, traceroute6
```

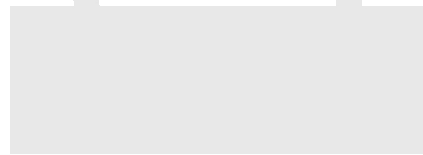
- Route management

```
route -A inet6
```



IPv6 install: Router

- Enable IPv6 routing
- Activate IPv6 at the router interfaces
- Configure IPv6 addresses
- Configure route advertisements



Basic configuration

The diagram illustrates the configuration of an IPv6 tunnel. It consists of a list of configuration commands on the left and their corresponding functions on the right, connected by arrows. A light gray background image of a network diagram is visible behind the text.

<code>ipv6 unicast-routing</code>	←	Enable IPv6 routing
...		
<code>interface Tunnel1</code>		
<code>description Tunnel example configuration</code>		
<code>no ip address</code>		
<code>ipv6 address 2001:648:2320::1/64</code>	←	Interface IPv6 address
<code>ipv6 enable</code>	←	Enable IPv6
<code>tunnel source Loopback2</code>		
<code>tunnel destination 195.251.29.1</code>		
<code>tunnel mode ipv6ip</code>	←	Tunnel mode
...		
<code>ipv6 route 2001:648:2320::/48 Tunnel1</code>	←	Forward all the traffic to /48 prefix to tunnel
...		

Example of tunnel configuration discussed in the previous session



Basic configuration

interface FastEthernet0/0.10
description VLAN 10 - INTERNAL network with user PCs
ipv6 address 2001:648:2320:1::/64 eui-64
ipv6 enable
ipv6 traffic-filter INTERNAL-IN-IPV6 in
ipv6 traffic-filter INTERNAL-OUT-IPV6 out
ipv6 nd ra-interval 30
ipv6 nd prefix 2001:648:2320:1::/64
!
ipv6 access-list INTERNAL-OUT-IPV6
permit tcp any eq www 2001:648:2320:1::/64 log
permit tcp any 2001:648:2320:1::/64 eq www log
...
permit icmp any any
...
deny ipv6 any any log

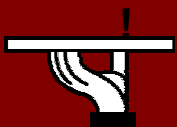
Enable IPv6 routing

IPv6 ACLs

Route advertisements

Example of ACLs

Allow ICMP



Intra-domain routing protocol

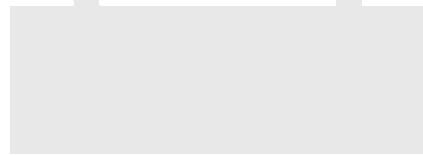
- RIPng, EIGRP
 - distance vector
- OSPFv3, IS-IS
 - link state protocols
 - two level hierarchy
 - faster converge than distance vector counterparts

IPv6 and IPv4 routing protocols
may not be the same



Border Gateway Protocol (BGP)

- BGP is an inter-domain routing protocol
 - Multi-protocol extensions for BGP-4 support IPv6 address prefixes
- Each domain is assigned an Autonomous System (AS) number
- Routing decisions as based on the AS path length



BGP information exchange

- Routers in peering domains exchange routing information
 - external BGP (eBGP)
 - advertise prefixes, filter incoming prefixes
- Routers in the same domain select the best routes, i.e. the next-hop, for all known external prefixes
 - internal BGP (iBGP)
 - inject best routes to the intra-domain routing table



BGP configuration (1/3)

- Completely separated configuration sections for IPv4 (*unicast & multicast*) and IPv6 (*unicast and multicast*)
 - Similar configuration steps for both protocols
- BGP configurations starts with the “router bgp <AS>”



BGP configuration (2/3)

```
router bgp 31065
```

AS number

```
...
```

```
neighbor 2001:5A0:400::29 remote-as 6453
```

```
neighbor 2001:5A0:400::29 peer-group mcit
```

```
neighbor 2001:5A0:400::29 description teleglobe
```

```
neighbor 3FFE:2900:300:1A::1 remote-as 6175
```

```
neighbor 3FFE:2900:300:1A::1 peer-group mcit
```

```
neighbor 3FFE:2900:300:1A::1 description Sprint
```

```
...
```

```
!
```

```
address-family ipv6
```

```
neighbor 2001:5A0:400::29 activate
```

```
neighbor 3FFE:2900:300:1A::1 activate
```

```
network 2001:4300::/32
```

```
network 2001:4300:1::/48
```

```
network 2001:4300:1:1:1:1:0/112
```

```
exit-address-family
```

```
!
```

BGP neighbours

IPv6 address family configuration

Advertised prefixes



BGP configuration (3/3)

```
router bgp 31065
```

```
...
```

```
neighbor 2001:5A0:400::29 remote-as 6453
```

```
neighbor 2001:5A0:400::29 peer-group mcit
```

```
neighbor 2001:5A0:400::29 description teleglobe
```

```
neighbor 3FFE:2900:300:1A::1 remote-as 6175
```

```
neighbor 3FFE:2900:300:1A::1 peer-group mcit
```

```
neighbor 3FFE:2900:300:1A::1 description Sprint
```

```
...
```

```
!
```

```
...
```

```
ipv6 prefix-list allip seq 5 deny 2001:4300::/32
```

```
ipv6 prefix-list allip seq 10 permit ::/0 le 128
```

```
route-map R-out permit 10
```

```
match ipv6 address prefix-list allip
```

```
set community no-export
```

```
!
```

Participate to a group

Filter prefixes for a specific group



BGP troubleshooting (1/2)

- Follow similar steps as in IPv4!

```
ipv6-router#show bgp ipv6 summary
```

```
...
```

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
2001:5A0:400::29									
	4	6453	18829	71205	97825	0	0	23:23:53	375
3FFE:2900:300:1A::1									
	4	6175	13673	60781	97825	0	0	23:23:36	605



BGP troubleshooting (2/2)

```
ipv6-router#show bgp ipv6
```

```
BGP table version is 97838, local router ID is 81.21.98.140
```

```
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,  
r RIB-failure Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 2001:200::/32	3FFE:2900:300:1A::1	81	0	6175	2497 2500 i
*> 2001:200:12D::/48	2001:470:1F00:FFFF::F5E	0	6939	6939	10566 378 6 17832 9270 7660 18083 i
...					

Check also BGP neighbour status,
received/transmitted routes, specific prefixes



Outline

- IPv6 Header
- Addressing
- Associated features & protocols
- Enabling IPv6
- **Transition mechanisms**



Transition to IPv6

- Transition to IPv6 (and coexistence with IPv4) will last for years!
 - The primary objective is the smooth migration to the IPv6 protocol! In other words, how IPv4-, IPv4/6- and IPv6-enabled systems communicate?
- There are three transition approaches
 - Dual-stack: IPv6 and IPv4 co-exist in the network devices, e.g. router or end-system
 - Tunneling: IPv6(/4) traffic is encapsulated into IPv4(/6) packets.
 - Translation: Rewrite packet headers!

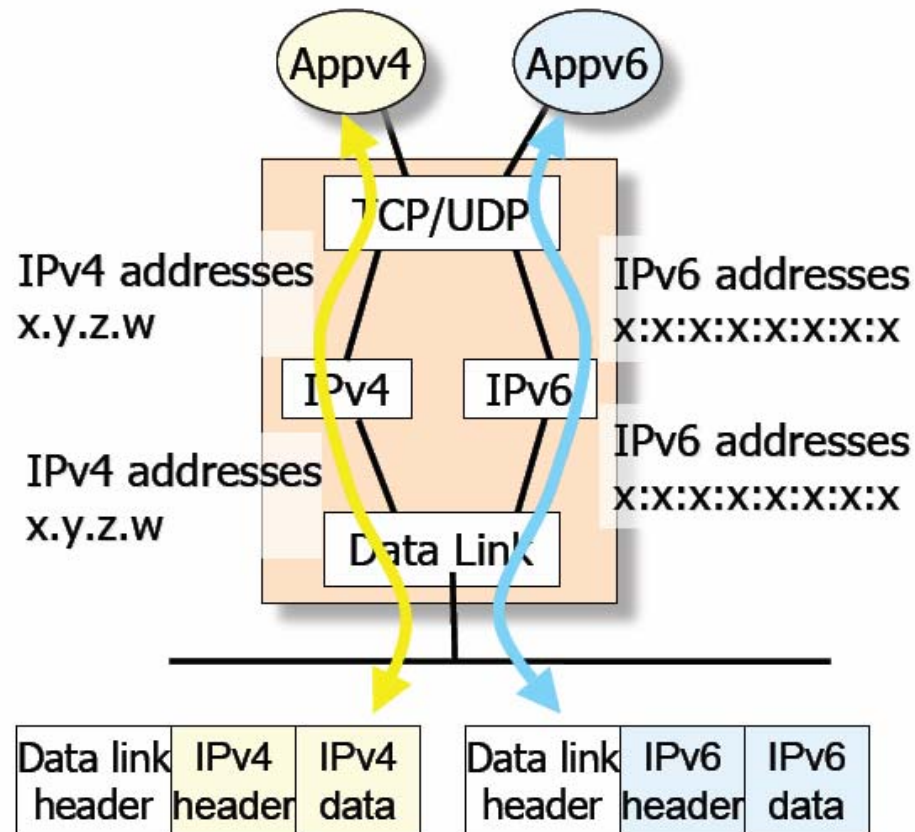


Dual Stack

- End-systems and routers support both protocols
 - Routers has to support two routing tables
 - Security has to be applied into both protocols
- Applications chooses which protocol to use
 - IPv6 is usually selected first.
 - If there is an “IPv6 connectivity problem”, IPv4 protocols is used after a while!

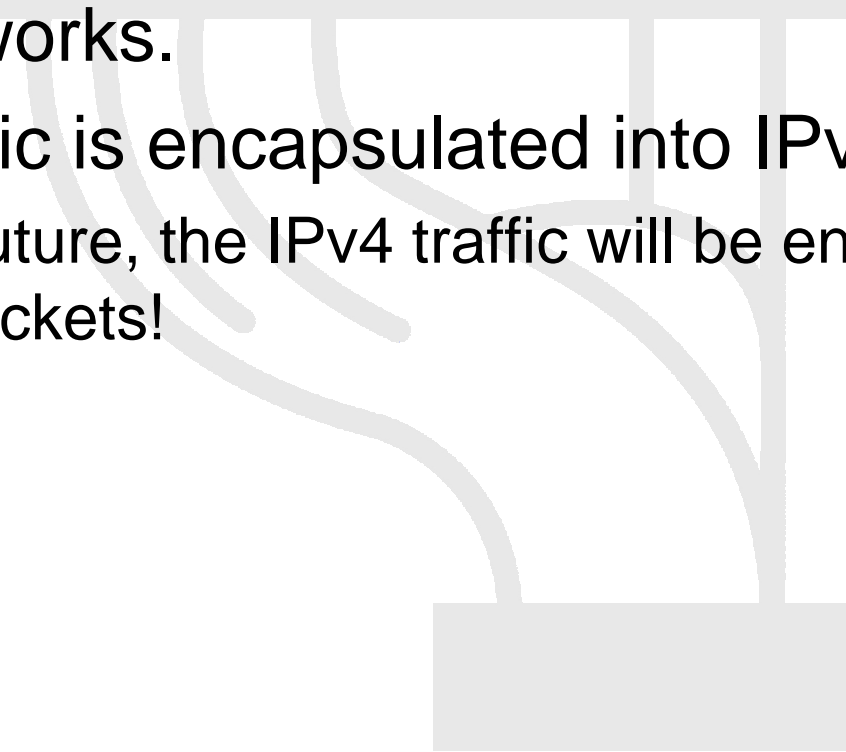


Typical dual stack implementation



Tunnelling (1/2)

- Tunnelling allows IPv6 connectivity through IPv4-only networks.
- IPv6 traffic is encapsulated into IPv4 packets
 - In the future, the IPv4 traffic will be encapsulated into IPv6 packets!

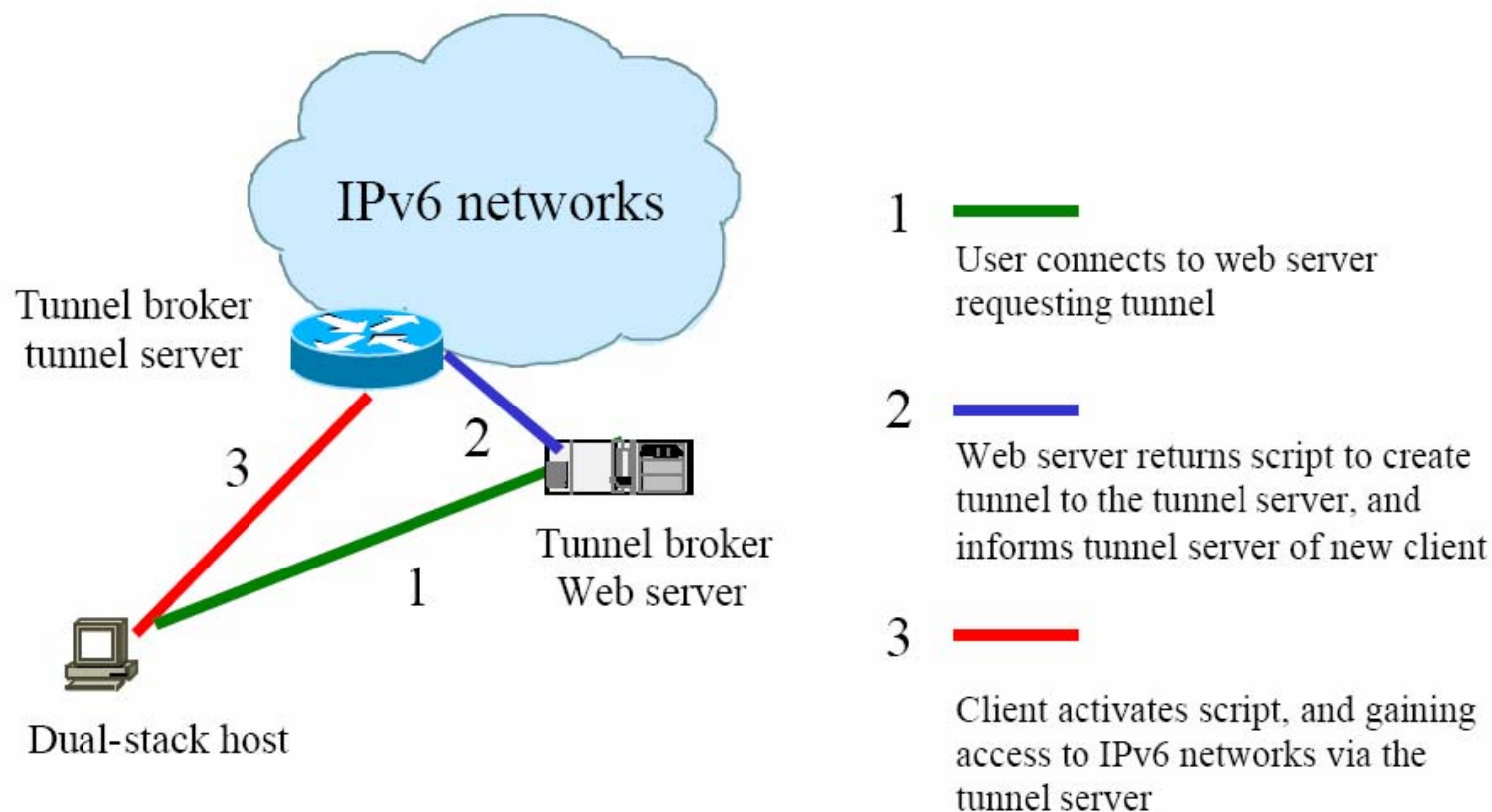


Tunnelling (2/2)

- IPv6 in IPv4: Encapsulate packets, manual install of configuration in both end-systems, protocol 41 has to be allowed
- TunnelBroker (RFC 3053): Semi manual installation of tunnels using a server.
- 6to4 (RFC 3056): Automatic tunnels, 2002::/16 addresses.
- Torpedo: Encapsulate to UDP packets, operates behind NATs, uses port 3544
- Intra-Site Automatic Tunnel Addressing Protocol (ISATAP): Create IDs using `::0:5efe:W.X.Y.Z`
- 6over4 (RFC 2529): “Virtual Ethernet”, multicast is required.



Tunnel Broker

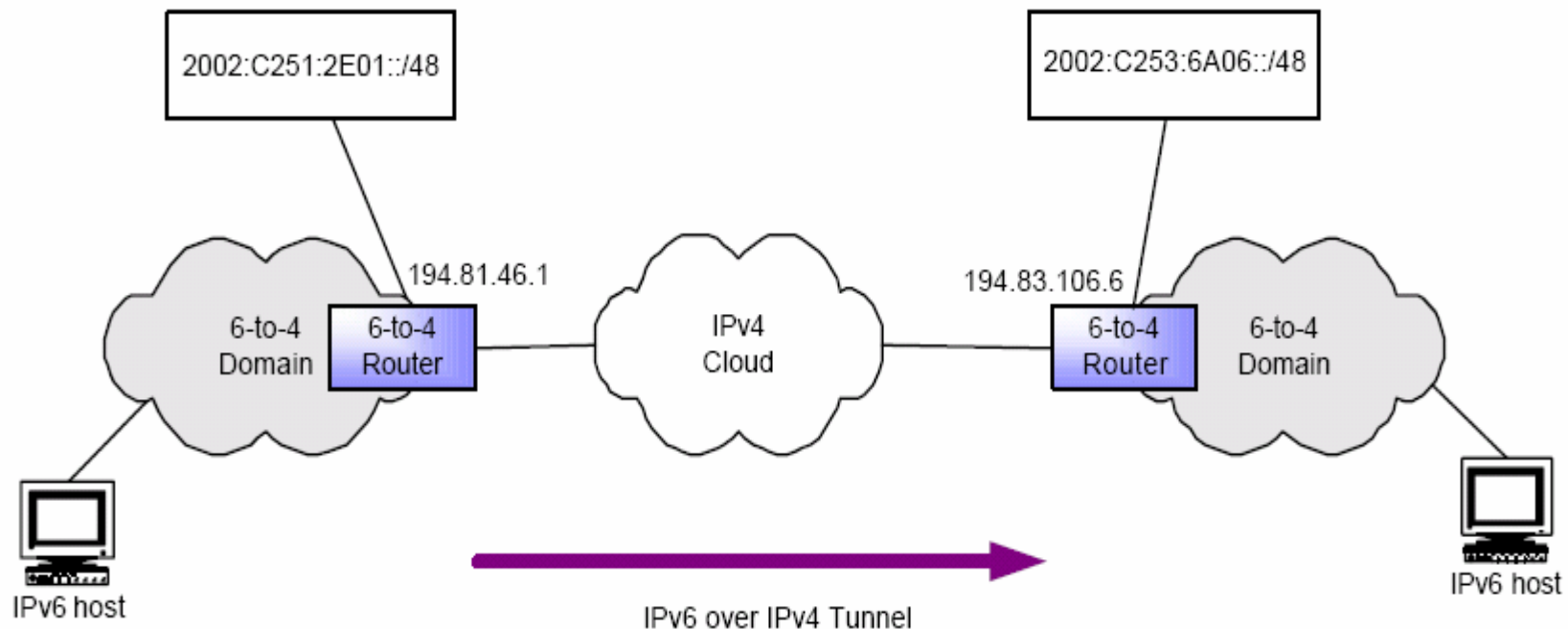


Components: Client, Tunnel Broker, Tunnel Server



6 to 4 automatic tunnelling

Reserved 6to4 TLA-ID: **2002::/16**
IPv4 address: **194.81.46.1 = c251:2e01**
Resulting 6to4 prefix: **2002:c251:2e01::/48**



Translation

- Allows IPv6-only hosts to communicate with IPv4-only hosts
 - Applied as close as possible at the edge of the network.
 - Addressing can be challenging. How to map IPv6 addresses to IPv4 addresses?
- Common mechanisms
 - Network Address Translation with Protocol Translation (NAT-PT) / Network Address Port Translation + Packet Translation (NAPT-PT) (RFC2766): An intermediate router convert IPv4 headers into IPv6 headers.
 - ALGs: Provide “proxing” at application layer, especially when it contains IP addresses.
 - Other: Bump in the Stack (RFC2767), Bump in the API (BIA), SOCKS (RFC1928), etc

Translation methods should be considered as a last resort solution!



Simplified Transition Methodology Strategy

- Get an IPv6 address space allocation
- Deploy an IPv6-enabled router
- Arrange external IPv6 connectivity
- Enable internal routing
- Deploy security measures and enable basic networking services
- Enable IPv6 to end-systems
- Active IPv6 to applications



Questions?

Thanks for your attention!

Contact

Athanassios Liakopoulos (aliako@grnet.gr)

