



JANET

IPv6 Hands-on Workshop

Module 1: **Overview and Operation**

UKERNA, Lancaster University
and University of Southampton, 2006



Module Overview

- Brief IPv6 Background and Motivations
- The 10 000' view
- IPv6 Features
 - Addressing
 - Packet format
 - Node address auto-configuration
 - Protocol features
- 'Sneak Peek'

The evolution of IPv6

- IPv4 has been around since the 1970's
- In early 1990's concern was raised over potential IPv4 address shortages
 - This led to Classless InterDomain Routing (RFC1519), private IPv4 addressing (RFC1918) and Network Address Translation (RFC1631)
- Early work on TUBA ('92), SIPP ('93) and CATNIP ('94)
 - History of the original recommendation available in RFC1752
- Then the IETF began work on IPv6 in the mid 1990's
 - Led to RFC2460, the base IPv6 protocol specification
 - Plus other associated RFCs: 2461, 2462, etc
 - See: <http://www.ietf.org/html.charters/ipv6-charter.html>

Internet Engineering Task Force (IETF)

- The Internet protocol standards body
 - <http://www.ietf.org/>
- Operates via working groups in a number of broad areas (Internet, Operations, etc.)
 - Participate via three annual meetings and email discussion lists
 - Operates on the basis of “rough consensus and running code”
- Standards evolve
 - From ‘Internet Drafts’ through to Proposed Standard onto Draft Standard and then full Internet Standard (very rare)
 - Requires interoperable implementations for Draft Standard
- IETF considers the global Internet ‘well being’ (RFC3935)

Why deploy IPv6?

- It's hard to put forward a compelling business case, but:
 - If you teach computer science, hands-on IPv6 access in student workstation labs should be important to you
 - Research projects, especially EU-IST, may need to use IPv6
 - You want to enable end-to-end transparent connectivity for certain networks within or outside your site, but currently are limited by the restraints implicit in NAT (Network Address Translation for IPv4)
 - You may wish to align your strategy with that of JANET/UKERNA
 - IPv6 ships in most host and router platforms today. If you don't control/deploy it, you may find your users do before you're ready to
 - You may stimulate interest in application development by doing so
 - Future IPv4 address shortages:
 - http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_8-3/ipv4.html

IPv6 on JANET

- JANET has had a production IPv6 allocation from the European Regional Internet Registry (RIPE) since mid-1999
 - Some address allocations have already been made to universities
 - All those allocations share a common (aggregated) network prefix
- The JANET backbone is IPv6 enabled
 - It runs IPv4 and IPv6 together on the same equipment
 - Connects to other 'dual-stack' academic networks worldwide
- Many JANET services support IPv6, e.g.
 - The DNS servers for .ac.uk
 - The *www.ja.net* web site
- The issue now is deployment in the RNO's and campuses
 - Which is why you're here, we hope 😊

IPv6 technical benefits

- Very large address space: 128-bit addresses
 - 3.4×10^{38} addresses, with 'unlimited' subnet size
 - Hierarchical addressing
 - Provider-aggregatable routing model based on prefixes
- Address auto-configuration ('plug and play' to some extent)
- Restores the 'end-to-end' principle of the Internet
- No (need for) address translators (NATs)
 - See RFC2775, RFC2993, and to some extent RFC3234
- Streamlined, extensible IP headers
 - Facilitates more efficient switching/routing engines, for example
- 'Built-in' security (IPsec support for AH and ESP)

Other potential IPv6 advantages

- Improved Mobile IP
 - Mobile IPv6 (MIPv6) does not require a Foreign Agent like Mobile IPv4 does, and has in-protocol routing optimisations
 - Most IETF work on mobility now assumes IPv6, including new work on mobile networks and in the network mobility area
- Some potential for improved Quality of Service (QoS)
 - IPv6 header includes a Flow Label field
 - Currently not used – so no real QoS advantage as yet
 - But the DiffServ header is the same (8 bit field, 6 bits DSCP)
- Some IPv6-specific tricks
 - Enabled by address size, e.g. use of cryptographically generated addresses (CGAs) and IPv6 temporary (privacy) addresses

The view from 10,000 feet

- Scenario
 - You have an operational IPv4 network
 - You'd like to investigate IPv6 with a view toward some pilot or test-bed deployment, in preparation for production deployment one day
- Questions
 - What do you need to know to configure IPv6 on host and router platforms, and for some basic applications?
 - How do you get (external) connectivity?
 - How do you deploy IPv6 internally?
 - What IPv6 address space do you use, and how?
 - What about security and transition implications?
- Initially, you would most likely run IPv4 and IPv6 together

The good news

- IPv6 is still basically the IP you know today with IPv4
- IPv6 implementations are generally mature
- Applications and services can, in general, be easily ported
 - Most, especially open source, applications are already IPv6 ready
 - Berkeley Sockets API well understood and a ‘simple’ extension
- UKERNA offers an IPv6 ‘Experimental’ Service
 - <http://www.ja.net/development/ipv6/experimental-service.html>
 - Any UK HE site can connect to this service
- Documentation and best practice examples exist, e.g.
 - The JANET IPv6 Technical Guide
 - <http://www.ja.net/services/publications/technical-guides/ipv6-tech-guide-for-web.pdf>
 - The 6NET Deployment Guide: <http://www.6net.org/book>

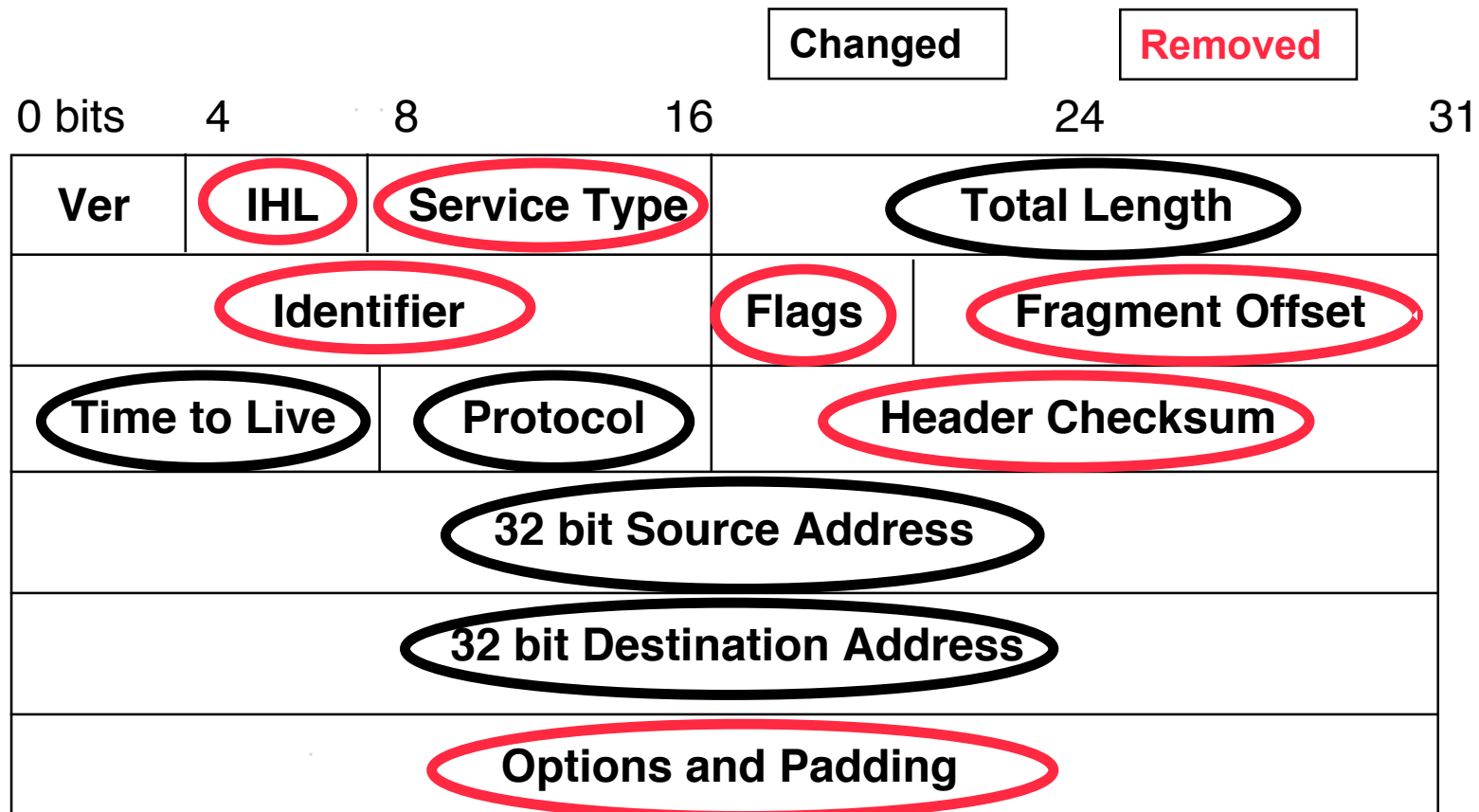
IPv6 'Basic' Features

- Topics covered:
 - IPv6 packet header, and differences to the IPv4 header
 - IPv6 address format and address architecture
 - Address management
 - IPv6 address auto-configuration
 - Stateful and stateless address configuration techniques
 - JANET's address allocation from the RIPE European registry
 - Protocol elements
 - Neighbour Discovery, Duplicate Address Detection, Path Maximum Transmission Unit (PMTU) Discovery
 - IPv6 specific 'tricks'

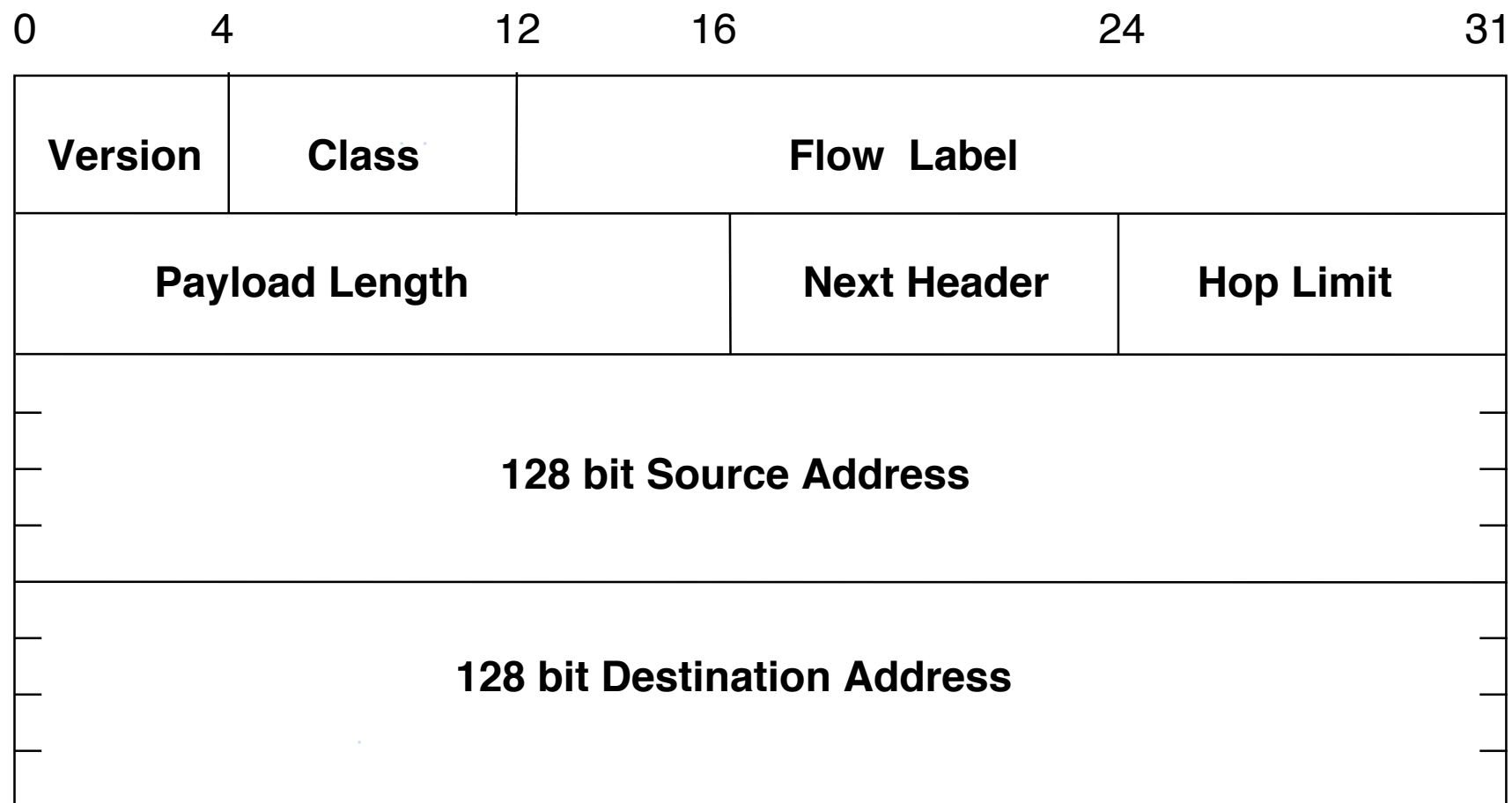
The IPv6 header

- The IPv6 packet header is streamlined compared to IPv4
 - Less fields than IPv4
 - Fixed header size
 - Should make processing more efficient
- IPv6 has concept of a chain of headers
 - One header per function, e.g.
 - Authentication header
 - Hop-by-Hop header
 - TCP header (the data!)
 - The 'next header' field links the headers together
 - Can define new headers, making the protocol extensible

The IPv4 header



The IPv6 header



Extensible header format

- Six basic header types:
 - Hop by hop (next header value = 0)
 - Routing (43)
 - Fragment (44)
 - Authentication Header (51) (see RFC2402)
 - Encapsulated Security Payload (50) (see RFC2406)
 - Destination options (60)
- Readily extensible
 - But for hardware forwarding engines there may be difficulty of adding new features, depending on whether the router needs to process the header

Sniffing on the wire

- Today you probably use standard tools to look at packets on your networks
 - tcpdump
 - Ethereal
- Both these packages support IPv6 packet analysis
- You can see these in action in the first lab session
- Note: IPv6 is Ethernet type *0x86dd*

Writing IPv6 addresses

- An IPv6 address is 128-bits long
 - Contrast to the 32-bit IPv4 addresses
- Written in the form x:x:x:x:x:x:x:x
 - e.g. 2001:0db8:0001:0035:0bad:beef:0000:cafe
- But you don't have to use all characters
 - So you can write 2001:db8:d:46:bad:beef:0:cafe
- You can use :: **once** in an address in place of zeros
 - e.g. 2001:630:d0:0:0:0:0:1 can be written 2001:630:d0::1
- If you have custom UIs for your management tools, these will likely need bigger input/display fields

IPv6 literals in URLs

- Web URLs use the ':' delimiter for specifying an alternative port number to the IANA default port 80
 - e.g. `http://www.foo.com:8080/`
- How, though, do we express port numbers when using a literal IPv6 address?
 - `http://2001:630:1::1:8080/` is difficult to parse
 - Is it host `2001:630:1::1` port 8080?
 - Is it (the default) port 80 on node `2001:630:1::1:8080`?
- The solution is to bracket literals as per RFC2732
 - e.g. `http://[2001:630:1::1]:8080/`

Network Prefixes

- Networks are identified by their address prefix
- The network prefix length is expressed as a bit-length, denoted by a '/' in addresses
 - e.g. 2001:630:1::/48 means the first 48 bits are for the network prefix, the rest can be used for subnetting and for host addressing within those subnets
- Addresses comprise a network prefix part and a host part
 - An IPv6 end-user subnet should always be a /64, i.e. 64 bits of network prefix and 64 bits for the host address in the subnet
- The prefix part can be allocated from a registry to a connectivity provider, or from a provider to an enterprise
 - The prefix can then be divided up into subnets

IPv6 production address space

- Top level production address space under 2000::

- See <http://www.ripe.net/rs/ipv6/stats/>

- ISPs are allocated a default /32 network prefix, from within which they allocate prefixes to customers

- ISP typically allocates /48 prefixes to sites

- Thus universities get a /48 size prefix from JANET

- The recommended longest prefix for ISPs is /32

- The biggest allocation to date is a /19 to France Telecom

- JANET was allocated 2001:0630::

- Allocation procedure is strictly hierarchical so to maximise aggregation of routing information

IPv6 address architecture

- Defined in RFC4291
- Includes the type of addresses you'll know from IPv4
- Loopback address
 - Expressed as ::1, equivalent to 127.0.0.1 in IPv4
- Global unicast addresses
 - Production use prefixes under 2000::/8, e.g 2001:630:1:2::1
 - But also old '6bone' testbed prefixes under 3ffe::/16 (now obsolete)
 - And 6to4 transition addresses under 2002::/16 (more on this tomorrow)
- Multicast addresses
 - Fourteen different scopes available: link, site, organisation, ...

IPv6 scoped addresses

- IPv6 introduces scoped addresses
- Link-local unicast addresses (fe80::/10)
 - Used on a link (subnet), and is unique on the link
 - Used for various protocols, including Neighbour Discovery
 - e.g. fe80::230:48ff:fe23:58df
- Unique Local IPv6 Unicast Addresses (fc00::/7)
 - See RFC4193
 - Generates/offers a 'unique' /48 sized site prefix, independent of any upstream connectivity provider
 - A bit like RFC1918 IPv4 addresses, but in theory non-ambiguous
 - Assuming people use the randomised 41 bits in the prefix, not just fc00::/64

IPv6 Multicast addresses

- Multicast replaces broadcast on links
 - IPv6 uses link local multicast (with link local source addresses)
 - Nodes join a variety of multicast groups, covering the functions that broadcasts achieved in IPv4, but without the ‘noise’ to disinterested nodes
- Nodes inherently multi-addressed on a per-interface basis
 - Every interface has a fe80::/10 link local address
 - One or more global unicast addresses can be configured (manually, or by stateless address auto-configuration)
- A node may have multiple ‘privacy’ addresses (RFC3041)
- Source and destination address selection metrics apply for multi-addressed nodes (see RFC3484)

Multicast addresses

- Format is `ffxy::[groupID]`
 - `x` = 4-bit flag, including whether a well-known or a transient group
 - `y` = 4-bit scope id
- Scopes:
 - Node = 1, link = 2, site = 5, organization = 8, global = e
 - Router policy defines how far a given scope will transit a network
- Some special multicast addresses, including:
 - All nodes on a link join `ff02::1`
 - All routers on a link join `ff02::2`
 - Solicited-node multicast address `ff02::1:ffXX:XXXX`
 - For each unicast address a node responds to the solicited-node address where the X'es are the last 24 bits of their unicast address

Aside: interface identifiers

- May sometimes need to express a specific interface
- Most OSes have now implemented the ‘%’ delimiter,
 - e.g. `fe80::230:48ff:fe23:58df%dc0`
- Useful to target a specific interface to disambiguate scoped addresses, e.g.
 - ... to send a ping on a specific link, e.g. `ff02::1%eth0`
 - ... to route certain traffic toward a next-hop router on a specific link, e.g. `fe80::203:93ff:fee9:e8e1%dc1`

Address management

- IPv6 offers several choices:
 - Managed address allocations to nodes ('Stateful')
 - Uses DHCPv6
 - Auto-configuration of node addresses ('Stateless')
 - Stateless Address Auto-configuration (SLAAC) (see RFC2462)
 - Router advertises 64-bit network prefix
 - Interface Identifiers (IID) automatically generated from MAC address
 - Manual address configuration per-node
 - Least preferred, but possible, and enables 'well-known' addresses for public-facing services, etc.

Stateful Address Allocation

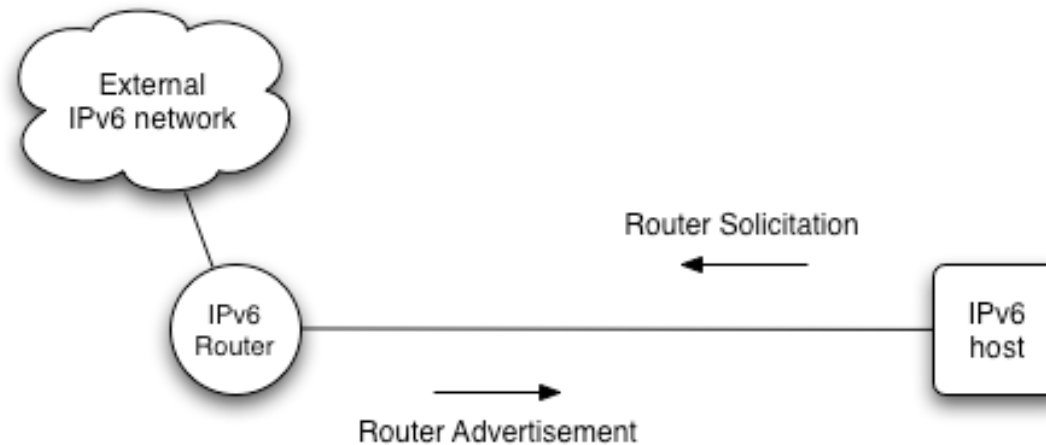
- DHCP is very common in campuses today
- For IPv6, you can use DHCPv6 (RFC 3315)
 - Implementations emerging, including an ISC project a la BIND
 - Bootstrapping a stateful configuration service is 'easy' if you already have such a database, e.g. for existing DHCPv4
- You may also need DHCPv6 even with stateless autoconfiguration
 - Because SLAAC doesn't yield DNS resolver addresses
- Look for products that let you manage DHCP for IPv4 and IPv6 in a coherent way

Stateless Address Management

- Hosts attach to a network and can determine the following information automatically:
 - The 64-bit network prefix(es) in use
 - The address of the default gateway to route traffic off-link
 - A global IPv6 address that they can use
 - PMTU of the link
 - Preferred and valid lifetimes of prefixes in use
- This is achieved by the router sending out Router Advertisements (RAs), periodically or in response to Router Solicitations sent by the host (e.g. on boot-up)
 - RAs are usually sent multicast to the all-hosts address on link

Router Advertisements

- Routers advertise a network prefix on an interface
- A host sees or solicits a Router Advertisement
 - Advertisement carries network prefix to use
 - Advertisement source address implies the default router



Auto-configuration: EUI-64

- How does the node generate a 64-bit host part of the address to use?
 - It needs a unique identifier
- The IPv6 subnet size for SLAAC is 64 bits (a /64 prefix)
 - Thus the host identifier part of the address is also 64 bits
- SLAAC combines the 64-bit network prefix with the unique host identifier to form a globally unique IPv6 address
- The host identifier is an EUI-64 address built from the MAC address
 - Take the 48 bit MC address and insert 'fffe' stuffing
- This is why the IPv6 network boundary is 'fixed' at 64 bits

IPv6 address autoconf example

- For example:
 - Host Ethernet address is 00:30:48:23:58:df
 - Network prefix is 2001:db8:1:cafe::/64
 - Address is:
 - 2001:0db8:0001:cafe:0230:48ff:fe23:58df
 - The change in the top byte of the address from '00' to '02' comes from the global bit being set in the translation from IEEE MAC-48 to EUI-64
 - The 'ff:fe' is the extra 16-bit stuffing to make 64 bits

IPv6 communication on a link

- So we've seen
 - What IPv6 addresses and packets look like
 - How address space and subnets are allocated
 - How an IPv6 node can get address configuration info
- For basic on-link communication we also need to know how an IPv6 node determines an on-link neighbour's link layer address
 - In IPv4 we use ARP
 - But IPv6 has no broadcast
 - For IPv6 we use Neighbour Discovery (ND) features

IPv6 Neighbour Discovery (ND)

- ND (RFC2461) is ICMPv6-based and uses:
 - Neighbour Solicitations (to request information)
 - Neighbour Advertisements (to offer or reply with information)
 - Router Advertisements and Solicitations are also part of ND
- The address resolution process is:
 - The node sends a solicitation message to the solicited-node multicast address of the target address
 - If present, the target responds with a unicast Neighbour Advertisement message that contains its own link layer address
- Use of the solicited-node address means nodes process less 'background' messages (i.e. only those for the solicited node multicast group that they join)

More on ND...

- Other IPv4-equivalent functionality is included in ND
- For example, there are redirects
 - To inform host of a better first hop towards a destination
- Nodes will also keep ND caches, much like ARP caches
 - The cache provides a node with neighbour reachability status
- Using ICMP is more media-independent than ARP and allows for IP security mechanisms
 - For example, secured (authenticated) Router Advertisements, as specified in SEND, RFC 3971

Duplicate Address Detection (DAD)

- IPv6 DAD assures that no two IPv6 nodes use the same IPv6 address
- A summary of the procedure (from RFC2462) is:
 - A node obtains a tentative address
 - It joins (if it hasn't already) the all nodes multicast group ff02::1
 - It joins the solicited node multicast address of the tentative address
 - ff02:0:0:0:1:ff00:<lower 24 bits>
 - It sends a neighbour solicitation on this address from the unspecified address '::'
 - The node can assume that the address is available (and thus no longer tentative) if no neighbour advertisement response is seen on ff02::1
- DAD is run whether a node uses stateless auto-configuration or DHCPv6

Path MTU discovery (RFC1981)

- Very important in IPv6
- PMTU Discovery is used by IPv6 hosts
- There is **no** fragmentation performed by routers
 - Thus no fragmentation created on a path in IPv6
 - Return ICMP “packet too big” if MTU would be exceeded on next hop at a router
 - Designed to improve efficiency
 - Routers get on with shifting packets at full-throttle, whilst nodes determine mechanistically the best MTU to use for a connection
- MTU at least 1280 bytes on all transports
- ICMP policies at firewalls are even more important with IPv6
 - ICMPv6 ‘best practise’ for filtering at borders is an on-going standards activity
 - See *draft-ietf-v6ops-icmpv6-filtering-recs-02*

IPv6-specific tricks

- Cryptographically generated addresses
 - Hash crypto data into the address host part
 - No room to do this in IPv4 addresses
- Privacy addresses
 - RFC3041: “random” host part of an address
 - We’ll look at this example in more detail
- Port scanning
 - 2^{64} hosts is a lot to scan on just one subnet
 - In IPv4 one port per second is 5 minutes (256 addresses)
 - In IPv6 a whole /64 is 500 billion years (2^{64} is a big number!)
 - See *draft-ietf-v6ops-scanning-implications-00*

IPv6 Privacy extensions

- Defined in RFC3041
- Perceived privacy issue with MAC address being embedded in an auto-configured IPv6 address
 - If device moves between networks (e.g. between WLAN hotspots), the network prefix changes but the host identifier remains the same
 - This means a device may be trackable via the last 64 bits
- Privacy extensions introduce a randomised host identifier
 - Used by a node when initiating outbound connections
 - Also useful for static hosts
 - But doesn't mitigate higher layer tracking, e.g. browser cookies
- But adds complexity to management – which addresses belong to the same host?

Some IPv4 and IPv6 differences

- Packet fragmentation only at sender (uses PMTU-D)
- No header checksum
- Unlimited subnet size
 - No need to re-jiggle subnets to conserve address space
 - Can be more resilient to port scanning
- No ARP or broadcasts
 - Now have ND and Duplicate Address Detection
- Privacy addresses
 - Host address may change with time
- Multi-addressing is the norm with IPv6
- Improved multicast (not a focus of this workshop)

IPv6 configuration: Linux

```
% /sbin/ip -f inet6 addr show
```

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
```

```
inet6 ::1/128 scope host
```

```
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
```

```
inet6 2001:630:d0:f102:230:48ff:fe23:58df/64 scope global dynamic
```

```
valid_lft 3588sec preferred_lft 1788sec
```

```
inet6 fe80::230:48ff:fe23:58df/64 scope link
```

IPv6 configuration: Win XP

```
> netsh interface ipv6 show interface interface=4
```

Interface 4: Ethernet: Local Area Connection

uses Neighbor Discovery

uses Router Discovery

link-layer address: 00-00-cb-68-0b-2e

preferred global **2001:630:d0:112:309e:3ba9:d0df:1afc**, life 57m25s/27m25s (temporary)

deprecated global **2001:630:d0:112:cc4e:835c:7e1b:e482**, life 57m25s/0s (temporary)

deprecated global **2001:630:d0:112:f4c5:398e:b5f3:bf58**, life 57m25s/0s (temporary)

deprecated global **2001:630:d0:112:88bd:46d0:b997:6dc4**, life 57m25s/0s (temporary)

deprecated global **2001:630:d0:112:e07c:fe6b:a58a:1608**, life 57m25s/0s (temporary)

deprecated global **2001:630:d0:112:b4dc:cfc5:c6a7:3724**, life 57m25s/0s (temporary)

deprecated global **2001:630:d0:112:1ca9:c9b:849e:7869**, life 57m25s/0s (temporary)

preferred global **2001:630:d0:112:200:cbff:fe68:b2e**, life 57m25s/27m25s (public)

preferred link-local **fe80::200:cbff:fe68:b2e**, life infinite

IPv6 routing table in Linux

```
$ netstat -nr -A inet6
```

```
Kernel IPv6 routing table
```

| Destination | Next Hop | Flags | Metric | Ref | Use | Iface |
|--|--------------------------|-------|--------|-------|-----|-------|
| ::1/128 | :: | U | 0 | 1674 | 18 | lo |
| 2001:630:d0:121:202:b3ff:feb2:e248/128 | :: | U | 0 | 18938 | 2 | lo |
| 2001:630:d0:121::/64 | :: | UA | 256 | 23053 | 0 | eth0 |
| fe80::202:b3ff:feb2:e248/128 | :: | U | 0 | 200 | 0 | lo |
| fe80::/64 | :: | UA | 256 | 0 | 0 | eth0 |
| ff00::/8 | :: | UA | 256 | 0 | 0 | eth0 |
| ::/0 | fe80::280:c8ff:feb9:a8b9 | UGDA | 1024 | 779 | 0 | eth0 |

```
IPv4 :
```

```
$ netstat -nr
```

```
Kernel IP routing table
```

| Destination | Gateway | Genmask | Flags | MSS | Window | irtt | Iface |
|-------------|---------------|---------------|-------|-----|--------|------|-------|
| 152.78.71.0 | 0.0.0.0 | 255.255.255.0 | U | 0 0 | 0 | | eth0 |
| 169.254.0.0 | 0.0.0.0 | 255.255.0.0 | U | 0 0 | 0 | | eth0 |
| 0.0.0.0 | 152.78.71.254 | 0.0.0.0 | UG | 0 0 | 0 | | eth0 |

IPv4 traceroute

```
$ traceroute news.uoregon.edu
traceroute to pith.uoregon.edu (128.223.220.25), 30 hops max, 38 byte packets
 1 ug-router.core.ecs.soton.ac.uk (152.78.71.254) 0.569 ms 0.355 ms 0.391 ms
 2 nokiafw.link (192.168.250.252) 0.780 ms 0.824 ms 0.682 ms
 3 152.78.108.6 (152.78.108.6) 1.379 ms 16.549 ms 2.122 ms
 4 b54gagesw1-aa.net.soton.ac.uk (152.78.108.62) 2.412 ms 1.965 ms 3.327 ms
 5 b54hafw1-ga1.net.soton.ac.uk (152.78.109.9) 2.933 ms 2.424 ms 2.172 ms
 6 b54gagesw2-hafw.net.soton.ac.uk (152.78.0.30) 3.141 ms 4.652 ms 3.847 ms
 7 212.219.151.113 (212.219.151.113) 3.787 ms 4.439 ms 3.887 ms
 8 212.219.151.121 (212.219.151.121) 3.695 ms 4.747 ms 22.099 ms
 9 * * *
10 146.97.40.2 (146.97.40.2) 14.479 ms 15.023 ms 11.098 ms MPLS Label=38 CoS=6 TTL=1 S=0
11 cosham-bar.ja.net (146.97.40.1) 15.238 ms 16.528 ms 12.318 ms
12 po9-0.cosh-scr.ja.net (146.97.35.21) 16.052 ms 14.277 ms 5.703 ms
13 po2-0.lond-scr.ja.net (146.97.33.41) 148.677 ms 36.461 ms 11.710 ms
14 po6-0.lond-scr3.ja.net (146.97.33.30) 15.853 ms 43.865 ms 39.367 ms
15 po2-0.geant-gw3.ja.net (146.97.35.138) 40.953 ms 14.532 ms 15.618 ms
16 janet.uk1.uk.geant.net (62.40.103.149) 14.626 ms 20.634 ms 100.940 ms
17 uk.ny1.ny.geant.net (62.40.96.169) 80.445 ms 125.028 ms 94.477 ms
18 198.32.11.61 (198.32.11.61) 107.057 ms 91.693 ms 86.708 ms
19 chinng-nycmng.abilene.ucaid.edu (198.32.8.82) 125.307 ms 124.037 ms 124.741 ms
20 iplsng-chinng.abilene.ucaid.edu (198.32.8.77) 132.282 ms 137.302 ms 114.212 ms
21 kscying-iplsng.abilene.ucaid.edu (198.32.8.81) 129.751 ms 128.755 ms 125.402 ms
22 dnvrng-kscying.abilene.ucaid.edu (198.32.8.13) 151.398 ms 146.335 ms 188.029 ms
23 snvang-dnvrng.abilene.ucaid.edu (198.32.8.1) 158.912 ms 163.864 ms 166.595 ms
24 pos-1-0.core0.eug.oregon-gigapop.net (198.32.163.17) 171.245 ms 166.453 ms 176.854 ms
25 uo-0.eug.oregon-gigapop.net (198.32.163.147) 197.786 ms 202.342 ms 205.446 ms
26 ge-5-1.uonet2-gw.uoregon.edu (128.223.2.2) 174.478 ms ge-5-1.uonet1-gw.uoregon.edu (128.223.2.1) 185.444 ms 167.490 ms
27 pith.uoregon.edu (128.223.220.25) 178.671 ms 179.907 ms 167.418 ms
```

IPv6 traceroute6

```
$ traceroute6 news.uoregon.edu
traceroute to pith.uoregon.edu (2001:468:d01:dc::80df:dc19) from 2001:630:d0:121:202:b3ff:feb2:e248, 30 hops max, 16 byte packets
 1 ug-router.6core.ecs.soton.ac.uk (2001:630:d0:121::1) 0.583 ms 0.384 ms 0.382 ms
 2 zaphod.6core.ecs.soton.ac.uk (2001:630:d0:101::1) 0.921 ms 0.761 ms 0.735 ms
 3 ford.6core.ecs.soton.ac.uk (2001:630:d0:100::1) 1.048 ms 0.852 ms 0.796 ms
 4 2001:630:c1:100::1 (2001:630:c1:100::1) 1.061 ms 1.038 ms 1.041 ms
 5 2001:630:c1:10::1 (2001:630:c1:10::1) 1.868 ms 1.618 ms 2.212 ms
 6 ***
 7 2001:630:c1::1 (2001:630:c1::1) 3.402 ms 2.14 ms 2.935 ms
 8 2001:630:c1::1 (2001:630:c1::1) 2.847 ms 2.6 ms 2.441 ms
 9 po9-0.cosh-scr.ja.net (2001:630:0:10::85) 126.799 ms 200.708 ms 3.097 ms
10 po2-0.lond-scr.ja.net (2001:630:0:10::29) 4.48 ms 5.773 ms 4.968 ms
11 po6-0.lond-scr3.ja.net (2001:630:0:10::36) 4.903 ms 5.262 ms 5.004 ms
12 2001:630:0:10::166 (2001:630:0:10::166) 4.249 ms 5.591 ms 5.472 ms
13 janet.uk1.uk.geant.net (2001:798:2028:10aa::1) 5.147 ms 5.116 ms 5.048 ms
14 uk.ny1.ny.geant.net (2001:798:20cc:1c01:2801::1) 73.984 ms 74.907 ms 73.142 ms
15 nycmng-esnet.abilene.ucaid.edu (2001:468:ff:15c3::1) 81.663 ms 74.415 ms 73.302 ms
16 chinng-nycmng.abilene.ucaid.edu (2001:468:ff:f15::1) 104.818 ms 104.435 ms 104.268 ms
17 iplsng-chinng.abilene.ucaid.edu (2001:468:ff:f12::2) 107.953 ms 107.671 ms 123.443 ms
18 kscying-iplsng.abilene.ucaid.edu (2001:468:ff:1213::2) 118.194 ms 117.178 ms 117.876 ms
19 dnvrng-kscying.abilene.ucaid.edu (2001:468:ff:1013::1) 135.564 ms 133.874 ms 128.349 ms
20 snvang-dnvrng.abilene.ucaid.edu (2001:468:ff:1017::2) 152.599 ms 153.794 ms 152.529 ms
21 oregon-snvang.abilene.ucaid.edu (2001:468:ff:174d::2) 164.727 ms 165.832 ms 165.028 ms
22 2001:468:d00:a390::3 (2001:468:d00:a390::3) 164.499 ms 166.777 ms 165.217 ms
23 ge-5-1.uonet1-gw.uoregon.edu (2001:468:d01:2::1) 164.502 ms 165.508 ms 166.048 ms
24 pith.ipv6.uoregon.edu (2001:468:d01:dc::80df:dc19) 165.853 ms 164.5 ms 166.116 ms
```

Summary

- A whistle-stop tour and a whirlwind introduction
 - Brief IPv6 Background and Motivations
 - The 10 000' view
 - IPv6 Basics
 - Addressing
 - Packet format
 - Node address auto-configuration
 - Protocol features
 - 'Sneak Peek'
- Next up: first hands-on lab...