LISA '13
NOVEMBER 3-8, 2013
WASHINGTON, D.C.
Sponsored by USENIX
in cooperation with LOPSA
27th Large Installation System Administration Conference
usenix

Using & Migrating to IPv6

Shumon Huque
University of Pennsylvania

USENIX LISA Conference
Washington, D.C., November 4th 2013

Penn
UNIVERSITY of PENNSYLVANIA

Using & Migrating to IPv6

This tutorial is being presented at the LISA 2013 Conference held in Washington, DC, on November 4th 2013.

Feedback, critique, suggestions on these slides gladly received at <shuque @ upenn.edu>

Reminder: Please fill out the evaluation forms for this course!

Who am I?

- An I.T. Director at the University of Pennsylvania
- Have also been:
  - Programmer (C, Perl, Python, Lisp)
  - UNIX Systems Administrator
  - Network Engineer
- Education: B.S. and M.S. (Computer Science) from Penn
- Also teach a Lab course on Network Protocols at Penn's School of Engineering & Applied Science

Who am I?

- Website: http://www.huque.com/~shuque/
- Blog: http://blog.huque.com/
- Twitter: https://twitter.com/shuque
- Google Plus: https://plus.google.com/+ShumonHuque

twitter
@shuque

My IPv6 experience

- More than a decade of hands on experience
- Have been running production IPv6 network infrastructure since 2002
  - 2002: MAGPI (Mid-Atlantic GigaPoP in Philadelphia for Internet2)
  - 2005: University of Pennsylvania campus network
  - Various application services at Penn (DNS, NTP, HTTP, XMPP, LDAP, Kerberos, etc)

Course Topics *(roughly)*

```
 1. IPv6 Motivation
 2. IPv6 Addressing and Protocol
 3. IPv6 support in service providers
 4. IPv6 support in Operating Systems
 5. IPv6 support in Applications
 6. IPv6 Tunneling
 7. Address Selection
 8. IPv6 and Security
 9. Troubleshooting & debugging tools
10. Transition & Co-existence mechanisms
11. Parting advice for IPv6 deployers

Bonus Material:
12. Programming Introduction
13. Routing Protocols & other network stuff
14. Router configuration examples
```

IPv6 Motivation

World IPv6 Launch

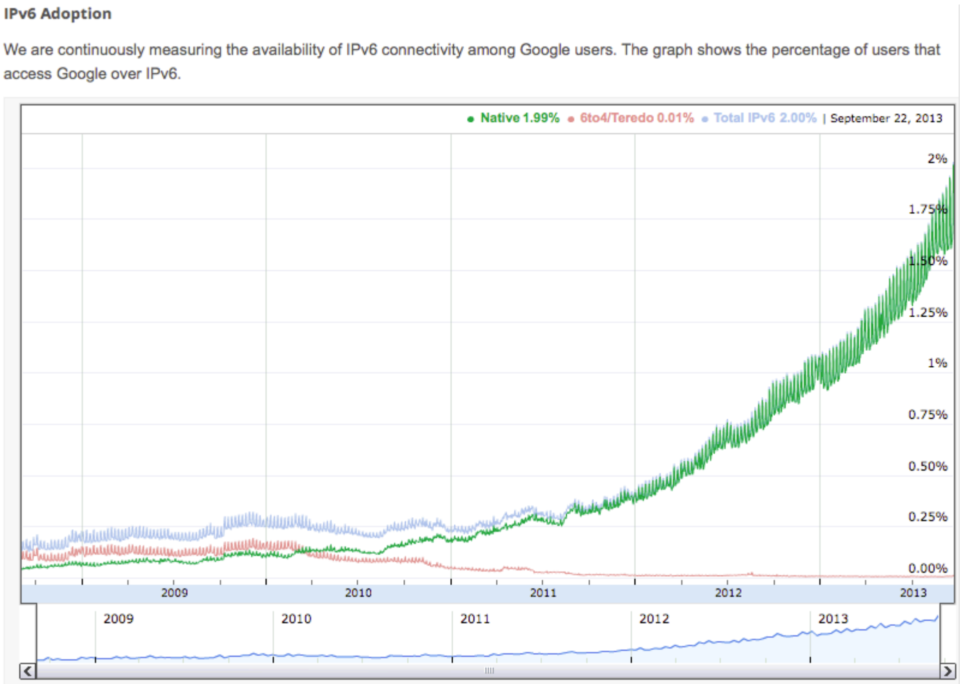
- http://www.worldipv6launch.org/

6 JUNE 2012

Major Internet service providers (ISPs), home networking equipment manufacturers, and web companies around the world are coming together to permanently enable IPv6 for their products and services by 6 June 2012.

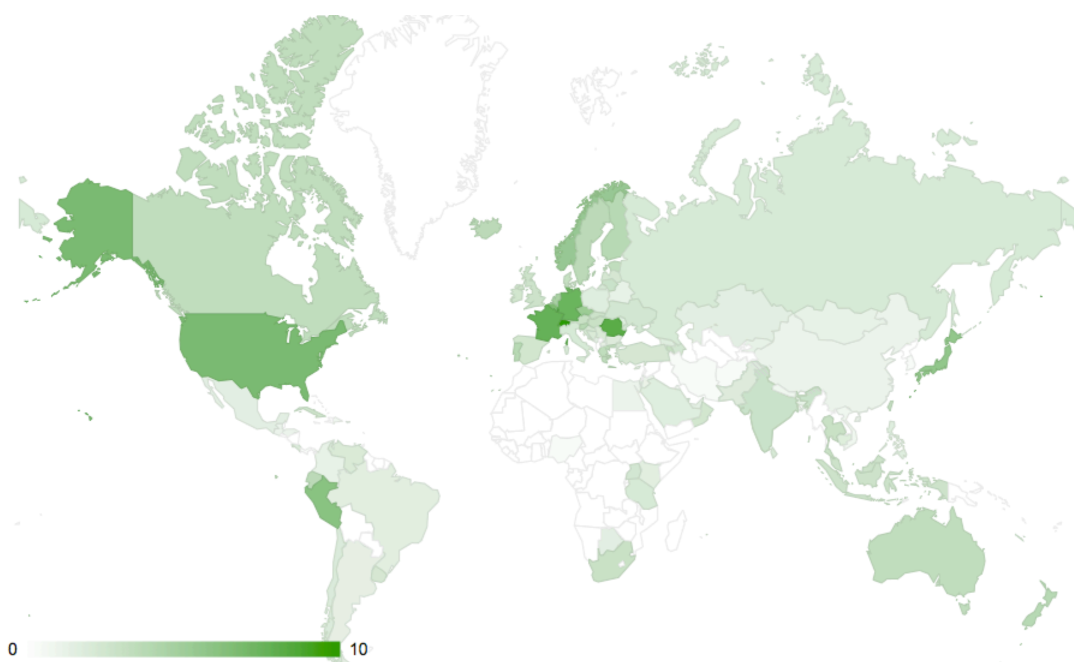
- Google, Facebook, Netflix, Yahoo!, MS Bing, ...
- ISPs: Comcast, AT&T, Free Telecom, Time Warner, ...
- CDNs: Akamai, Limelight, ...
- Some universities, corporations, government agencies, ...

http://www.google.com/intl/en/ipv6/statistics.html



[Migrating to IPv6, USENIX LISA 2013]

http://6lab.cisco.com/stats/



[Migrating to IPv6, USENIX LISA 2013]

http://www.worldipv6launch.org/measurements/

Participating Network	ASN(s)	IPv6 deployment
KDDI	2516	8.49%
ATT	6389, 7018, 7132	11.52%
Comcast	7015, 7016, 7725, 7922, 11025, 13367, 13385, 20214, 21508, 22258, 33287, 33489, 33490, 33491, 33650, 33651, 33652, 33653, 33654, 33655, 33656, 33657, 33659, 33660, 33661, 33662, 33664, 33665, 33666, 33667, 33668, 36733	7.95%
Free	12322	18.04%
Verizon Wireless	6167, 22394	35.73%
RCS & RDS	8708	13.25%
Deutsche Telekom AG	3320	5.18%
Time Warner Cable	7843, 10796, 11351, 11426, 11427, 12271, 20001	2.38%
Swisscom	3303	18.83%
SoftBank BB	17676	1.07%
Chubu Telecommunications	18126	19.06%
Telefonica del Peru	6147	3.11%
Hughes Network Systems	6621	28.00%
VOO	12392	45.32%
XS4ALL	3265	16.14%
StarHub	4657, 55430	6.68%
Forthnet	1241	5.81%
Energy Group Networks	18779	4.62%
M1 Limited	4773, 17547	4.56%
Internode	4739	4.33%
University of Minnesota	57, 217	26.44%
Indiana University	87	18.08%
Virginia Tech	1312	59.54%

[Migrati

http://www.worldipv6launch.org/measurements/

Participating Network	ASN(s)	IPv6 deployment
interscholz Internet Services GmbH & Co. KG	33843	81.22%
Sauk Valley Community College	13953	71.23%
ThaiSarn	3836	69.41%
Rensselaer Polytechnic Institute	91	61.25%
Virginia Tech	1312	59.54%
Universidad de Carabobo	27893	58.50%
Sistemas Fratec S.A.	262149	58.19%
Universidad Panamericana	13679	57.89%
Bayu Krisnawan	55694	56.64%
Dedicated Zone Inc	393237	56.55%
Google Fiber	16591	55.64%
Red Académica de Centros de Investigación y Universidades Nacionales REACCIUN	20312, 27807	52.41%
NETIS TELECOM	25592	52.17%
Gustavus Adolphus College	17234	46.64%
DreamHost	26347	46.32%

[Migrating to IPv6, USENIX LISA 2013]

IPv6: Internet Protocol v6

- Version 6: The next generation Internet Protocol
- Much larger address space: 128 bits vs 32 bits
  - (Note: not 4x larger, but 2^{96} times larger!)
- No NAT (goal: restore end-to-end architectural model)
- Scalable routing (we'll talk about multihoming later)
- Other: header simplification, NDP (a better version of ARP), auto-configuration, flow labelling, and more ..
- Note: *IPv6 is not backwards compatible with IPv4*

[Migrating to IPv6, USENIX LISA 2013]

IPv6: Internet Protocol v6

- But primary impetus is the much larger address space
- Impending exhaustion of IPv4 addresses
- But Internet continues to grow
  - Not only in terms of the number of users, but also in the number and range of devices being connected to the network
  - The "*Internet of Things*"

[Migrating to IPv6, USENIX LISA 2013]

Adverse Consequences of not deploying IPv6?

[Migrating to IPv6, USENIX LISA 2013]

IPv4 Transfer Markets

- IPv4 transfer markets (sanctioned or unsanctioned)
  - March 2011: Microsoft acquired block of 600,000 addresses from Nortel for $7.5 million ($11.25/address)
  - December 2011: Borders books sold a /16 to Cerna for $786,432 ($12.00/address)
  - Rise of brokering companies: Addrex, Kalorama, Hilco streambank, etc

[Migrating to IPv6, USENIX LISA 2013]

More NAT

- More NAT
- More layers of NAT
- Carrier Grade NATs (CGN), Large Scale NATs (LSN) ...
- Damaging impacts on applications
- Implications of large scale address sharing
  - single points of failure, performance bottleneck, easy DoS target, geolocation difficulty, impacts on ACLs, blocklists, port space rationing, resource management, NAT traversal method reliability, ALG complexity, and more ...

[Migrating to IPv6, USENIX LISA 2013]

Balkanization

- Disconnected islands of IPv4 and IPv6
- Balkanization, and resulting disruption of universal connectivity

[Migrating to IPv6, USENIX LISA 2013]

Transition vs Co-existence

- IPv4 isn't going away anytime soon, possibly not for many decades
- So, for most folks, already connected to the IPv4 Internet, ***we are not (yet) transitioning to IPv6***
- We are ***deploying IPv6 to co-exist with IPv4***
- To allow us to communicate with both the IPv4 and IPv6 Internet
- Note: some folks in the near future will move directly to IPv6, due to complete IPv4 depletion

[Migrating to IPv6, USENIX LISA 2013]

IPv6: Brief History

- Design work began by IETF in 1993, to deal with projected depletion of IPv4 addresses (then ~ 2010-2017)
- Completed in ~1999
  - RFC 1883: first version of IPv6 specification (Dec 1995)
  - RFC 2460: Internet Protocol version 6 specification (Dec 1998)
- April 1999: first RIR allocation of IPv6 address space
- By now hundreds of RFCs exist, describing various aspects of IPv6 and its applications
- IPv6 is still evolving ...

[Migrating to IPv6, USENIX LISA 2013]

IP address allocation

- IANA (Internet Assigned Numbers Authority)
  - Top level allocator of IP address blocks
  - Usually allocates to "Regional Internet Registries" (RIR)
- 5 RIRs, serving distinct geographic regions:
  - ARIN, RIPE, LACNIC, APNIC, AFRINIC
- RIRs allocate to large Internet Service Providers (ISPs), and some large organizations
- Large ISPs allocate to smaller entities (other ISPs, enterprises etc)

from <u>http://ipv4.potaroo.net</u> (Geoff Huston, APNIC)

(from September 25th 2013)

IANA Unallocated Address Pool Exhaustion:

03-Feb-2011

Depleted already!

Projected RIR Address Pool Exhaustion Dates:

RIR	Projected Exhaustion Date	Remaining
APNIC:	**19-Apr-2011** (actual)	0.8327
RIPE NCC:	**14-Sep-2012** (actual)	0.8630
ARIN:	**06-Jan-2015**	1.7610
LACNIC:	**19-Apr-2015**	1.8963
AFRINIC:	**19-Aug-2022**	3.5704

Also see <u>https://www.arin.net/resources/request/ipv4_countdown.html</u>

View of IPv4 /8's (1st octet)

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
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

[Migrating to IPv6, USENIX LISA 2013]

[**Red** = not publicly usable]

Special use IPv4 addresses

0.0.0.0/8	Source hosts on this net
10.0.0.0/8	RFC 1918 private address space
127.0.0/8	Loopback addresses block
169.254.0.0/16	Link Local address block (rfc 3927)
172.16.0.0/12	RFC 1918 private address space
192.0.0.0/24	IANA reserved (proto assignments)
192.0.2.0/24	TEST-NET-1: documentation and example code
192.88.99.0/24	6to4 Relay anycast addresses
192.168.0.0/16	RFC 1918 private address space
192.18.0.0/15	testing
192.51.100.0/24	TEST-NET-2
203.0.113.0/24	TEST-NET-3
224.0.0.0/4	Class D: IP Multicast address range
240.0.0.0/4	Class E: Reserved address range for future use
100.64.0.0/10	Shared address for v4/v6 transition mechanisms

[Migrating to IPv6, USENIX LISA 2013]

See RFC 6890 for details

What you need to deploy IPv6

- Obtain IPv6 address space
  - from your RIR or ISP
- IPv6 connectivity (preferably native) from your ISP
- IPv6 deployment in network infrastructure, operating systems, and applications (may require upgrades)
- IT staff and customer service training

IPv6 addressing and protocol details

IPv4 addresses

- Example: 192.168.7.13
- 32 bits
- “Dotted Quad notation”
- Four 8-bit numbers (“octets”) in range 0..255, separated by dots
- 2^{32} = 4.3 billion (approximate) possible addresses
  - *(Usable number of addresses much lower though: routing & subnet hierarchies - see RFC 3194 - Host Density ratio)*

IPv6 addresses

- 128-bits (four times as large)
- 8 fields of 16 bits each (4 hex digits) separated by colons (:)
- [Hex digits are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f]
- 2^{128} possible addresses (an incomprehensibly large number)

`2001:0db8:3902:00c2:0000:0000:0000:fe04`

(2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456)

IPv6 addresses

- Zero suppression & compression for more compact format
  - Suppress (omit) leading zeros in each field
  - Replace consecutive fields of all zeros with a double colon (::) - only one sequence of zero fields can be compressed this way

```
2001:0db8:3902:00c2:0000:0000:0000:fe04

2001:db8:3902:c2::fe04
```


IPv6 canonical form

- RFC 5952: A recommendation for IPv6 Text Representation
- Same address can be represented many ways in IPv6, making it more challenging to do some tasks (searching, pattern matching, programmatic processing of the text forms, etc)
- Define a (recommended) canonical text representation
  - must suppress leading zeroes in a field
  - Use :: to compress only the longest sequence of zero fields, and only the first one if there are multiple equal length sequences
  - Compression of a single zero field is not allowed
  - a, b, c, d, e, f must be in lower case

IPv4 mapped IPv6 address

```
Uses prefix ::ffff:0:0/96
            ( 0:0:0:0:0:ffff:0:0/96 )
Example     ::ffff:192.0.2.124
```

- Used for handling IPv4 connections on an IPv6 socket
- Note slightly different text representation to make it easier to embed 32-bit IPv4 address in the IPv6 address
- See RFC 4038 for details ("Application aspects of IPv6 transition")
- Not normally seen on wire (only IPv4 packets seen)

IPv6 in URLs

- To represent literal IPv6 addresses in Uniform Resource Locators (URL), enclose the address in square braces:
  - `http://[2001:db8:ab:cd::3]:8080/index.html`
  - `ldap://[2001:db8:ab:cd::4]/`
  - `ftp://[2001:db8:ab:cd::5]/blah.txt`
- See RFC 3986 for details [URI: Generic Syntax]
- (For zone IDs, see RFC 6874)
- (This is generally only needed for debugging and diagnostic work)

IPv6 network prefixes

- Format: IPv6-Address / prefix-length

- `2001:db8::/32`
- `2001:db8:ab23::/48`   (typical org assignment)
- `2001:db8:ab23:74::/64`   (most subnets)
- `2001:db8:ab23:74::2/64`
- `2001:db8:ab23:75::1/127`   (p2p links by some)
- `2001:db8:ab23:76::a/128`   (loopback)

IPv6 DNS records

- **AAAA** ("**Quad-A**") DNS record type is used to map domain names to IPv6 addresses
- IPv4 uses the "**A**" record
- There was another record called **A6**, which didn't catch on (and now declared historic by RFC 6563)

```
www.ietf.org.  1800 IN  A     12.22.58.30
www.ietf.org.  1800 IN  AAAA  2001:1890:123a::1:1e
```

IPv6 Reverse DNS

- As in IPv4, PTR records are used for reverse DNS
- Uses "**ip6.arpa**" subtree (IPv4 uses "in-addr.arpa")
- The LHS of the PTR record ("owner name") is constructed by the following method:
  - Expand all the zeros in the IPv6 address
  - Reverse all the hex digits
  - Make each hex digit a DNS label
  - Append "ip6.arpa." to the domain name (note: the older "ip6.int" was formally deprecated in 2005, RFC 4159)

IPv6 reverse DNS example

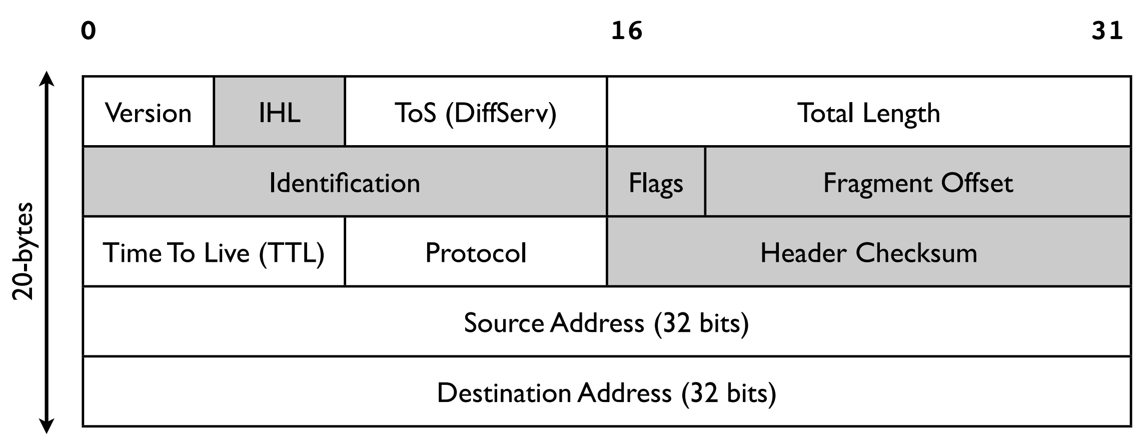
```
host1.example.com. IN AAAA 2001:db8:3902:7b2::fe04

2001:db8:3902:7b2::fe04                   (orig IPv6 address)

2001:0db8:3902:07b2:0000:0000:0000:fe04   (expand zeros)

20010db8390207b2000000000000fe04          (delete colons)

40ef0000000000002b7020938bd01002          (reverse digits)

4.0.e.f.0.0.0.0.0.0.0.0.0.0.0.0.2.b.7.0.2.0.9.3.8.b.d.
0.1.0.0.2                                 (make DNS labels)

4.0.e.f.0.0.0.0.0.0.0.0.0.0.0.0.2.b.7.0.2.0.9.3.8.b.d.
0.1.0.0.2.ip6.arpa.                       (append ip6.arpa.)

4.0.e.f.0.0.0.0.0.0.0.0.0.0.0.0.2.b.7.0.2.0.9.3.8.b.d.
0.1.0.0.2.ip6.arpa. IN PTR host1.example.com.
```

IPv6 DNS references

- RFC 3596: DNS Extensions to Support IP Version 6
- RFC 3363: Representing IPv6 Addresses in DNS
- RFC 3364: Tradeoffs in DNS Support for IPv6
- RFC 4472: Operational Considerations and Issues with IPv6 DNS

[Migrating to IPv6, USENIX LISA 2013]

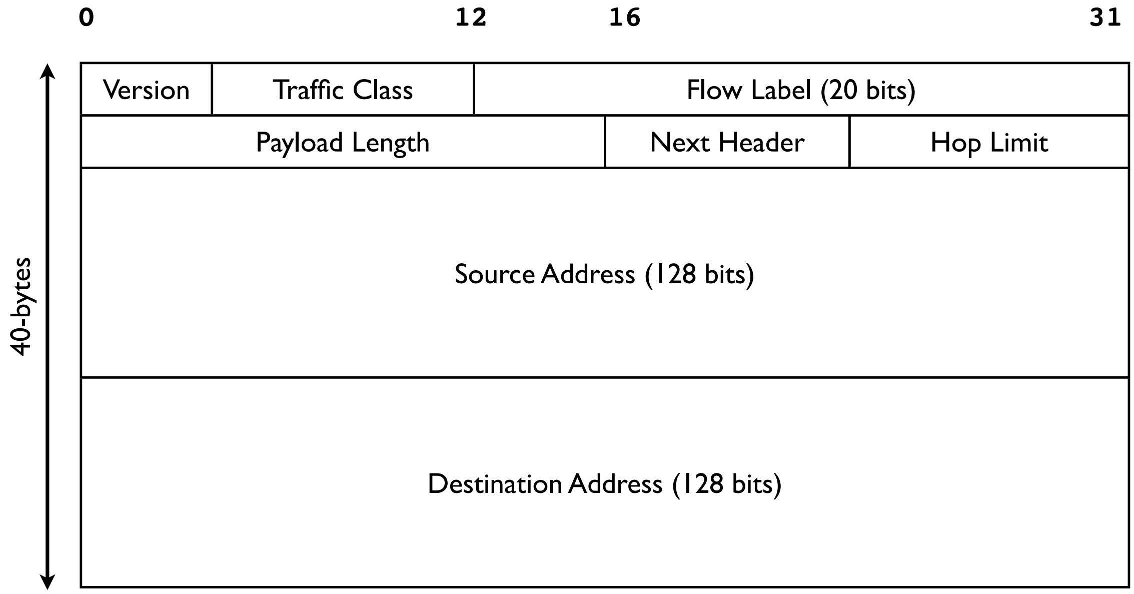
IPv4 Header



`[Followed optionally by Options and Padding]`

[Migrating to IPv6, USENIX LISA 2013]

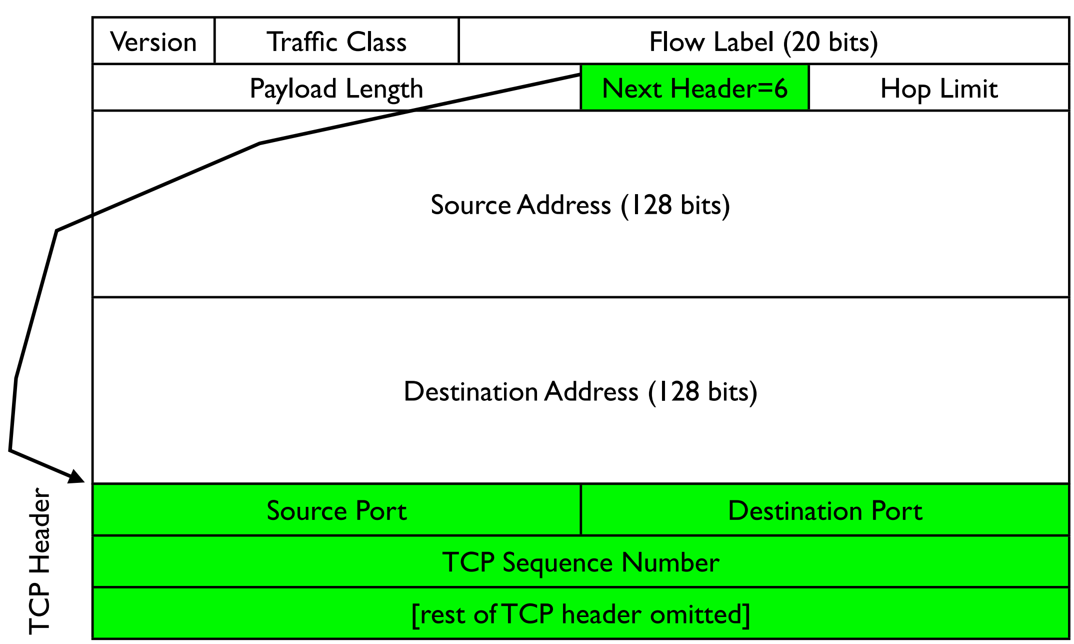
IPv6 Header



[Followed optionally by Extension Headers & Upper Layer payload]

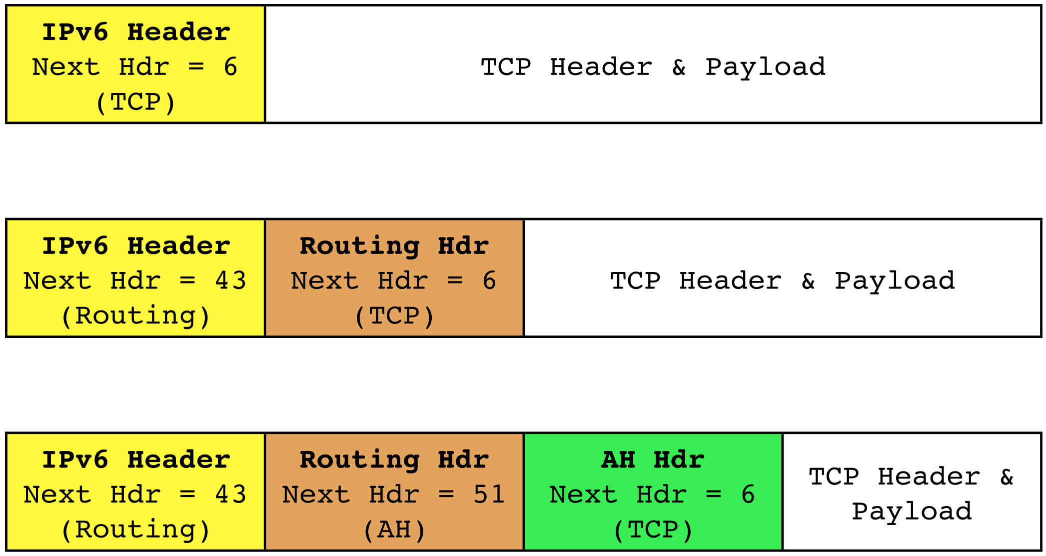
[Migrating to IPv6, USENIX LISA 2013]

Upper Layer payload



[Migrating to IPv6, USENIX LISA 2013]

Extension Headers



[Migrating to IPv6, USENIX LISA 2013]

Extension Headers

- Hop-by-Hop (must be examined by all routers along path: eg. router alert)
- Destination Options (can appear after hop-by-hop when RH used, or at end)
- Routing (Note: RFC 5095, Dec 2007, deprecated RH type 0)
- Fragment (fragments less common in v6 because of path MTU discovery)
- Authentication (IPsec AH)
- Encapsulating Security Payload (IPsec ESP)
- Others: MIPv6, HIP, SHIM6, ...

[See also RFC 6564 - A Uniform Format for IPv6 Extension Headers]

[Migrating to IPv6, USENIX LISA 2013]

IPv6 Address Types

- **Unicast**
- **Multicast**
- **Anycast**

- Note: there is no "**broadcast**" in IPv6

[Migrating to IPv6, USENIX LISA 2013]

Unicast Address Types

- **Global** Unicast Addresses
  - Static, Stateless Address Autoconfiguration, DHCP assigned
  - Tunneled (6to4, Teredo, ISATAP, ...)
  - Others (CGA, HIP, ...)
- **Link Local** Addresses
- **Unique Local Addresses** (ULA)
- **Loopback** (::1)
- **Unspecified** (::)

Also see RFC 6890: Special Purpose IP Address Registries
and RFC 6666: IPv6 Discard Prefix

[Migrating to IPv6, USENIX LISA 2013]

Link Local Addresses

- All IPv6 network interfaces have a Link Local address
- Special address used for communication on local subnet
- Self assigned in the range fe80::/10 (actually the subset fe80::/64)
- Last 64-bits derived from MAC address (EUI-64)
- Could be the same on multiple physical interfaces
- Often written with scope-id to differentiate interface
- fe80::21b:63ff:fe94:9d73%en0

Modified EUI-64 (21b:63ff:fe94:9d73) **scope-id** (en0)

[Migrating to IPv6, USENIX LISA 2013]

Global IPv6 address form

48-bits	16-bits	64-bits
Global Routing prefix	SubnetID	Interface ID (host part)
001 + 45-bits	SubnetID	Interface ID (host part)

- Prefix 2000::/3 (address starts with bits 001)
- 45-bits: global routing prefix (IANA->RIR->LIR/ISP)
- 16-bits Subnet ID -- can number 65,536 subnets!
- 64-bits Interface ID

[Migrating to IPv6, USENIX LISA 2013]

Multicast addresses

- Multicast: an efficient one-to-many form of communication
- A special IPv6 address prefix, **`ff00::/8`**, identifies multicast group addresses
- Hosts that wish to receive multicast traffic "join" the associated multicast group
- Have scopes (link local, site, global etc)
- In IPv4, the group joining and leaving protocol is IGMP
- In IPv6, the protocol is **MLD** (Multicast Listener Discovery)

Multicast addresses

```
|   8    |  4 |  4 |                  112 bits                   |
+------ -+----+----+---------------------------------------------+
|11111111|flgs|scop|                  group ID                   |
+--------+----+----+---------------------------------------------+

   binary 11111111 at the start of the address identifies the address
   as being a multicast address.

                                 +-+-+-+-+
   flgs is a set of 4 flags:     |0|R|P|T|
                                 +-+-+-+-+
   scop is a 4-bit multicast scope value used to limit the scope of
   the multicast group.  The values are as follows:

      0  reserved
      1  Interface-Local scope
      2  Link-Local scope
      4  Admin-Local scope
      5  Site-Local scope
      8  Organization-Local scope
      E  Global scope
```

(excerpted from RFC 4291: IPv6 Addressing Architecture)

Some multicast addresses

ff02::1	All nodes on link
ff02::2	All routers on link
ff02::5	All OSPF routers
ff02::6	All OSPF DR (designated routers)
ff02::b	Mobile Agents
ff02::c	SSDP (Simple Service Discovery Protocol)
ff02::d	All PIM (Protocol Independent Multicast) routers
ff02::12	VRRP (Virtual Router Redundancy Protocol)
ff02::16	All MLDv2 capable routers
ff02::fb	mDNSv6 (Multicast DNS v6)
ff02::1:2	All DHCP relay agents and servers
ff02::1:3	LLMNR (Link Local multicast name resolution)
ff02::1:ff00:0000/104	Solicited Node Multicast Address
ff02::2:ff00:0/104	ICMP Node Information Queries (RFC 4620)

Address Configuration

- Servers: usually have statically configured IPv6 addresses (and associated DNS records)
- Client computers: can automatically configure themselves an address ("Stateless Address Autoconfiguration")
  - typically don't have associated DNS records
- Managed address allocation can be performed with DHCPv6 (Dynamic Host Configuration Protocol for IPv6)
  - DNS can be pre-populated for DHCPv6 address pools

IPv6 Subnets

- Usually fixed size: 64-bits long (p2p links are often exceptions)
- First 4 fields defined the network portion of the address
- How many hosts can such a subnet accommodate?

264 =** 18,446,744,073,709,551,616 (or approx 18.5 quintillion)

```
eg. for a subnet: 2001:db8:ab23:74::/64

start :  2001:db8:ab23:74:0000:0000:0000:0000
end   :  2001:db8:ab23:74:ffff:ffff:ffff:ffff
```


IPv6 Subnets

- IPv6 Addressing Architecture (RFC 4291) requires the host portion of the address (or the "Interface Identifier") to be 64-bits long
- To accommodate a method that allows hosts to uniquely construct that portion: Modified EUI-64 format
- Generates unique 64-bit identifier from MAC address
- This is used by Stateless Address Autoconfiguration (to be described shortly)

Neighbor Discovery

- RFC 4861
- Analog of ARP in IPv4 but provides many other capabilities
- Stateless Address Autoconfiguration (RFC 4862)
- Managed configuration indication (address configuration policy)
- Router discovery
- Subnet Prefix discovery
- Duplicate address detection (DAD)
- Neighbor unreachability detection (NUD)

Neighbor discovery messages

- Uses 5 ICMPv6 message types:

- Router Solicitation
- Router Advertisement
- Neighbor Solicitation (like ARP Request)
- Neighbor Advertisement (like ARP Response)
- Redirect

[RFC 4443: ICMPv6 Specification]

Solicited node multicast

- Neighbor discovery involves finding other hosts & routers on the local subnet, but recall there is no broadcast in IPv6
- ND uses solicited node multicast addresses, which partition hosts on a subnet into distinct groups, each corresponding to a distinct multicast addresses associated with sets of IPv6 addresses
- For every IPv6 address a host has, it joins the corresponding solicited node multicast address
- Address contains last 24 bits of the IPv6 address
- First 104 bits are the well defined prefix
  - `ff02:0:0:0:0:1:ff00::/104`

[Migrating to IPv6, USENIX LISA 2013]

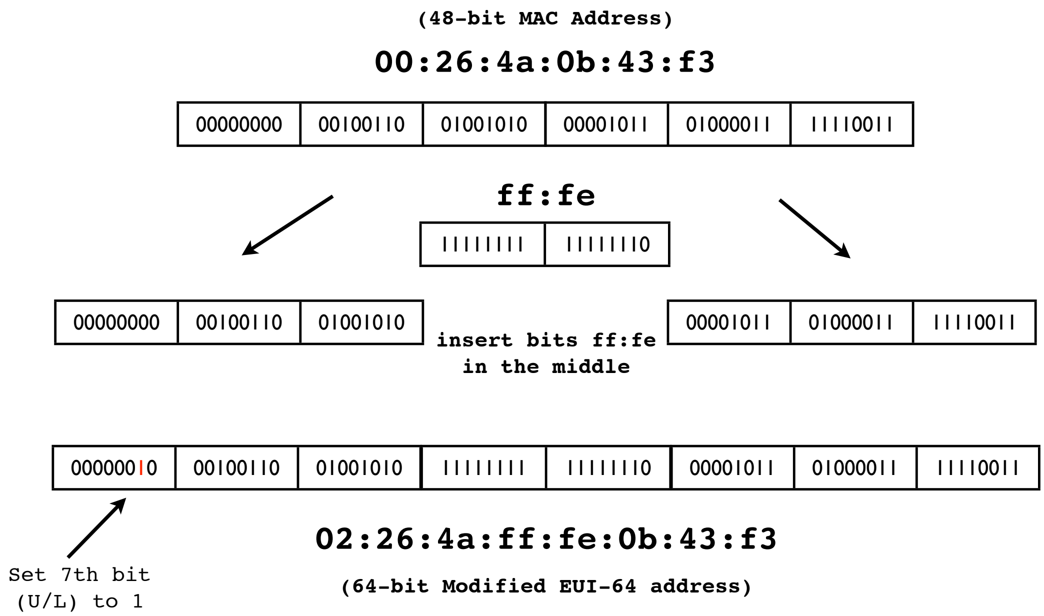
Solicited node multicast

- If target address is: 2001:db8:123::ce97:7fce
- Last 24 bits are: 97:7f:ce. Prepend ff02::1:ff00:/104
- So solicited node multicast address is: ff02::1:ff97:7fce
- If Ethernet is the link layer, the corresponding ethernet multicast address: 33-33 + last-32bits of IPv6 address = 33-33-ff-97-7f-ce

- Main takeaway: In IPv6, neighbor discovery involves host sending packet to the solicited node multicast address associated with the target (in contrast to IPv4's ARP, where we send to the broadcast address)

[Migrating to IPv6, USENIX LISA 2013]

Modified EUI-64



(details: see RFC 3513, Appendix A)

Autoconfiguration

- RFC 4862: Stateless Address Autoconfiguration (SLAAC)
- Host listens to Router Advertisements (RA) on local subnet
- Obtains 64-bit subnet prefix from RA (and perhaps other parameters)
- Computes modified EUI-64 from its MAC address and concatenates it to 64-bit subnet prefix to form IPv6 address

```
Link prefix from RA: 2001:db8:abcd:1234::/64
Host MAC address: 00:1b:63:94:9d:73
EUI-64 address:   021b:63ff:fe94:9d73
Resulting IPv6 address:
   2001:db8:abcd:1234:021b:63ff:fe94:9d73
```

Router Advertisement

0	8							16	31
Type = 134	Code = 0							Checksum	
Cur Hop Limit	M	O	H	Pref	P	R	R	Router Lifetime	
Reachable Time									
Retransmission Timer									
Options .... (Source Link Layer, MTU, Prefix Information, ...)									

```
M = managed config flag: "use stateful DHCPv6"
O = other config flag: get "other" params via stateless DHCPv6
Pref = Default Router Preference (Hi, Lo, Med) - RFC 4191
The most common option is the Prefix Information option
```

[Migrating to IPv6, USENIX LISA 2013]

RA: Prefix Info option

0	8	16			31
Type=3	Length	Prefix Length	L	A	Reserved1
Valid Lifetime					
Preferred Lifetime					
Reserved2					
Prefix					

L = "on link" prefix indicator
A = this prefix can be used for auto-configuration

[Migrating to IPv6, USENIX LISA 2013]

Router Discovery eg.

```
Router Solicitation Message ->
Src: fe80::c072:7a5f:c1b5:24d1
Dst: ff02::2 (all routers multicast)
ICMPv6 Type 133 (RS)
Option:
  Src Link Layer Addr (my MAC addr)
```

```
<- Router Advertisement Message
Src: router's link local addr
Dst: ff02::1 (all nodes or solicitor)
ICMPv6 Type 134 (RA)
Flags (M=0, O=0, pref=0)
Router Lifetime: 1800
Reachable time: 0
Retrans time: 0
Options:
  Src Link Layer Addr (my Mac)
  MTU: 1500
  Prefix Info
    prefix: 2001:db8:ab:cd::/64
    valid life: 2592000
    preferred lifetime: 604800
```

(Routers also periodically send out unsolicited router advertisements.)

[Migrating to IPv6, USENIX LISA 2013]

Neighbor Discovery eg.


```
Neighbor Solicitation Message ->
Src: A's IPv6 address
Dst: Solicited-node multicast of B
ICMPv6 Type 135 (NS)
Target: B's IPv6 address
Options:
  Src Link Layer Addr (A's MAC addr)
```

```
<- Neighbor Advertisement Message
Src: B's IPv6 address
Dst: A's IPv6 address
ICMPv6 Type 135 (NA)
Target: B's IPv6 address
Options:
  Src Link Layer Addr (B's MAC addr)
```

(Summary: A is asking: what is the link layer address associated with B's IPv6 address?)

[Migrating to IPv6, USENIX LISA 2013]

SLAAC & Privacy?

- SLAAC exposes MAC address of a host to the world
- In IPv4, MAC was exposed to local subnet only
- Does this have privacy implications?
- Remote sites may be able to track & correlate your network activities by examining a constant portion of your address
- How serious are these compared to other highly privacy invasive mechanisms already in use at higher layers?
  - think of things like web cookies that track/expose user identity, often across sites; browser fingerprinting; synthetic subdomains, ...

[Migrating to IPv6, USENIX LISA 2013]

Temporary addresses

- RFC 4941: Privacy extensions for Stateless Address Auto-configuration
- Pool of "Temporary addresses" or "Privacy addresses"
- Derived from MAC initially, ala SLAAC, but then passed through a 1-way hash algorithm
- Designed to change over time; duration configurable or based on policy; hours, days, on reboot, or different addresses for different applications or endpoints
- Cons: complicate network debugging, security/audit implications (see proposal for "*stable privacy addresses*")

[Migrating to IPv6, USENIX LISA 2013]

Temporary addresses

- On by default many modern OSes: Windows (since XP), Mac OS X, Open Suse, Ubuntu Linux, ..
- Also on in Apple iOS devices (iPhone, iPad etc)
- Android uses and prefers privacy addresses (on wifi)
- Off by default in others, but easily turned on via configuration knobs in the operating system (eg. sysctl on Linux and *BSD)

DHCPv6

- Stateless DHCPv6 (RFC 3736)
  - No IPv6 address assignment ("stateless"); assumed that SLAAC or other method will be used for address configuration
  - Other network configuration parameters are provided, eg. DNS servers, NTP servers etc
- Stateful DHCPv6 (RFC 3315)
  - Managed address allocation analogous to DHCP in IPv4
  - Easy to populate DNS & reverse DNS (compared to autoconfig)

Stateful DHCPv6

- Stateful DHCPv6 (RFC 3315) - more details
- Conceptually similar to IPv4 DHCP
- Uses RA's **M** (managed configuration) flag
- Requires DHCPv6 server, which assigns IPv6 leases to clients
- And provides other configuration info (DNS, NTP, ... etc)

Differences with IPv4 DHCP

- Uses UDP ports 546 (client) and 547 (server)
- Clients use autoconfigured link-local addresses as source
- Clients send messages to multicast group address ff02::1:2 ("all dhcp servers and relay agents group"); IPv4 uses broadcast
- **Does not assign default gateway** - use Router Advertisement
- DHCP servers can send "reconfigure" messages to clients
- Rapid Commit option (reduce exchange from 4 to 2 messages)
- DUID (DHCP Unique IDentifiers)
- Provision for temporary and non-temporary addresses

IPv4 v IPv6 DHCP messages

DHCP v4 (rfc 2131)	DHCP v6 (rfc 3315)
`C -> broadcast: DISCOVER`	`C -> multicast: SOLICIT`
`S -> C: OFFER`	`S -> C: ADVERTISE`
`C -> S: REQUEST`	`C -> S: REQUEST`
`S -> C: ACK`	`S -> C: REPLY`

[Migrating to IPv6, USENIX LISA 2013]

IPv4 v IPv6 DHCP messages

with rapid commit option

DHCP v4 (rfc 2131)	DHCP v6 (rfc 3315)
`C -> broadcast: DISCOVER`	`C -> multicast: SOLICIT`
`S -> C: OFFER`	`S -> C: REPLY`
`C -> S: REQUEST`	
`S -> C: ACK`	

[Migrating to IPv6, USENIX LISA 2013]

DHCPv6 DUID

- Clients no longer use hardware address to identify themselves
  - Issues: multiple interfaces, mobility, virtual interfaces & VMs etc
  - DUID: DHCP Unique IDentifier - use long lived unique id instead
  - Used by both clients and servers
  - Number of methods to initialize a DUID (based on link layer address, time, enterprise numbers etc): DUID-LLT/ET/LT

DHCPv6 DUID

- DUID construction methods:
  - DUID-LLT: constructed from link-layer address of one of the system interfaces (ie. from hardware address), hardware type, and timestamp
  - DUID-EN: using enterprise number of device manufacturer and an ID number
  - DUID-LL: constructed from link-layer address and hardware type
- Challenges with DUIDs:
  - when we want to obtain MACs; correlating IPv4/IPv6 addresses; persistent storage on some devices, etc

DHCPv6 Leases & Lifetimes

- Leases (bindings) as in IPv4
- Lifetimes: Offered addresses have preferred and Valid lifetimes as in stateless autoconfiguration

Stateless DHCPv6

- Triggered by "O (other config) flag" in RA messages
- INFORMATION_REQUEST message:
- To request other configuration parameters
  - C -> multicast: INFORMATION_REQUEST
  - S -> C: REPLY
- Conceptually similar to the DHCPINFORM message in IPv4

DHCPv6 options

- Used by both stateful and stateless DHCPv6
- Some common options for configuration information:
  - DNS Recursive Nameservers
  - DNS Search List
  - NTP servers
  - SIP servers
  - Prefix Delegation (RFC 3633) - eg. delegating prefix to a home router
  - and many more ...

DHCPv6 Other

- Other messages: RENEW, REBIND, CONFIRM, RELEASE, DECLINE, RECONFIGURE
- Relay Agents supported as in IPv4 (RELAY_FORW, RELAY_REPL)
- ServerFailover protocol
  - So far missing in v6, but development work in progress.
  - Less important for IPv6 (use multiple independent servers offering disjoint address pools), but there are some uses cases.
- Prefix delegation
- **New!** RFC 6939: Client Link-Layer Address Option

DHCPv6 with Relay Agent

Client	Relay	Server
->Solicit		
	->Relay-forw(Solicit)	
		<-Relay-repl(Advertise)
	<- Advertise	
->Request		
	->Relay-forw(Request)	
		<-Relay-repl(Reply)
	<-Reply	

[Migrating to IPv6, USENIX LISA 2013]

Other config possibilities

- New Router Advertisement options
  - RFC 6106: RA options for DNS configuration
  - Allows transmission of DNS server and related info via RA (obviating need to deliver this via other means, eg. stateless DHCPv6)
  - Very few implementations to date though ..
- In the opposing camp, there is (was?) also a proposal to extend DHCPv6 to provide default gateway options, obviating the need to use Router Advertisements
  - http://tools.ietf.org/html/draft-droms-dhc-dhcpv6-default-router-00
  - http://tools.ietf.org/html/draft-ietf-mif-dhcpv6-route-option-05

[Migrating to IPv6, USENIX LISA 2013]

Unique Local Address (ULA)

- RFC 4193, Prefix fc00::/7
- Replacement for IPv4 Private Addresses (RFC 1918)
- Note: the older *site local prefix* (fec0::/10) was deprecated
- Intended for local use within a site or group of sites
- Globally unique, but not routable on the global Internet
- Addresses some operational issues seen with IPv4 and RFC 1918 addresses

Unique Local Address (ULA)

```
| 7 bits |1|  40 bits   |  16 bits  |          64 bits           |
+--------+-+------------+-----------+----------------------------+
| Prefix |L| Global ID  | Subnet ID |        Interface ID        |
+--------+-+------------+-----------+----------------------------+
```

Where:

```
   Prefix            FC00::/7 prefix to identify Local IPv6 unicast
                     addresses.

   L                 Set to 1 if the prefix is locally assigned.
                     Set to 0 may be defined in the future.  See
                     Section 3.2 for additional information.

   Global ID         40-bit global identifier used to create a
                     globally unique prefix.

   Subnet ID         16-bit Subnet ID is an identifier of a subnet
                     within the site.
```

[RFC 4193 excerpt]

IPv6 addresses on a Mac

```
$ ifconfig -a

lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
     options=3<RXCSUM,TXCSUM>
     inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
     inet 127.0.0.1 netmask 0xff000000
     inet6 ::1 prefixlen 128
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
     ether e4:ce:8f:07:b6:13
     inet6 fe80::e6ce:8fff:fe07:b613%en1 prefixlen 64 scopeid 0x5
     inet6 2607:f470:6:3:e6ce:8fff:fe07:b613 prefixlen 64 autoconf
     inet6 2607:f470:6:3:3947:98a5:68f6:2ef1 prefixlen 64 autoconf temporary
     inet 165.123.70.49 netmask 0xffffff00 broadcast 165.123.70.255
     media: autoselect
     status: active
```


IPv6 addresses on Windows

```
C:>ipconfig

Windows IP Configuration

[...]

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :

   IPv6 Address. . . . . . . . . . . : 2607:f470:2f:1:2dde:6914:cafe:15fe
   Temporary IPv6 Address. . . . . . : 2607:f470:2f:1:806c:86ee:b372:47b2
   Link-local IPv6 Address . . . . . : fe80::2dde:6914:cafe:15fe%10
   IPv4 Address. . . . . . . . . . . : 128.91.196.91
   Subnet Mask . . . . . . . . . . . : 255.255.254.0
   Default Gateway . . . . . . . . . : fe80::216:9cff:fe6f:5dc0%10
                                       128.91.196.1
```

IPv6 addresses on Linux

```
$ ifconfig

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:544285 errors:0 dropped:0 overruns:0 frame:0
          TX packets:544285 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:355551886 (339.0 MiB)  TX bytes:355551886 (339.0 MiB)

eth0      Link encap:Ethernet  HWaddr 00:14:4F:01:31:F8
          inet addr:128.91.XXX.68  Bcast:128.91.255.255  Mask:255.255.254.0
          inet6 addr: 2607:f470:2a:1::a:2/64 Scope:Global
          inet6 addr: 2607:f470:2a:1::a:1/64 Scope:Global
          inet6 addr: 2607:f470:2a:1:214:4fff:fe01:34f7/64 Scope:Global
          inet6 addr: fe80::214:4fff:fe01:34f7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9228907 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3889095 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1686780678 (1.5 GiB)  TX bytes:1997866418 (1.8 GiB)
```


Linux RA example

Example of RA info seen on a Linux machine. This host has a static address, and 2 autoconfigured addresses, one deprecated because its preferred lifetime has expired.

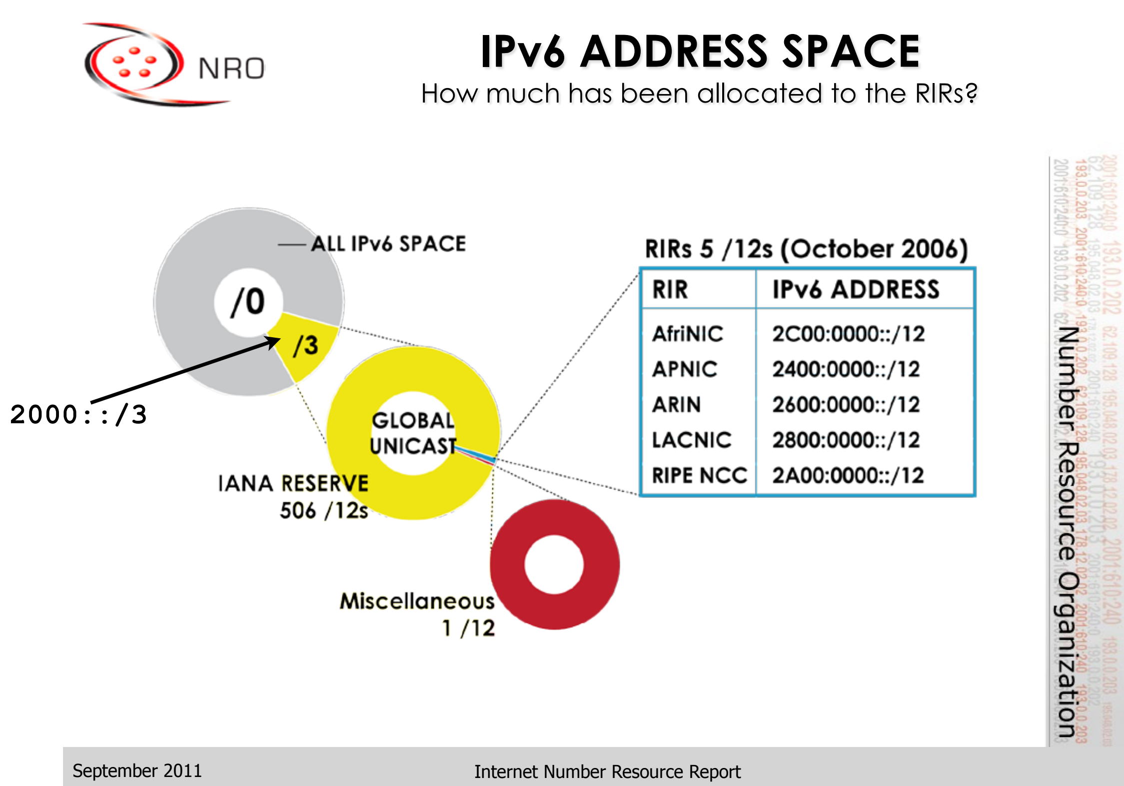
```
$ /sbin/ip -6 addr show dev eth0

eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2607:f470:1001::1:12/64 scope global
       valid_lft forever preferred_lft forever
    inet6 2607:f470:1001:0:214:4fff:fee6:b650/64 scope global
dynamic
       valid_lft 2591957sec preferred_lft 604757sec
    inet6 2001:468:1802:101:214:4fff:fee6:b650/64 scope global
deprecated dynamic
       valid_lft 6308sec preferred_lft -892sec
    inet6 fe80::214:4fff:fee6:b650/64 scope link
       valid_lft forever preferred_lft forever
```

Common IPv6 assignments

< /32	RIRs and large ISPs
/32	Typically to LIRs and ISPs. Allows 65,536 /48 assignments, or 4 billion /64 subnets
/48	Most enterprises and endsites. Allows deployment of 65,536 /64 subnets
/56	Small sites; Residential service. Allows deployment of 256 /64 subnets
/64	Residential service. Allows one /64 subnet

- See RFC 6177 for latest thinking on endsite assignments

[Migrating to IPv6, USENIX LISA 2013]



[Migrating to IPv6, USENIX LISA 2013]

PA vs PI address space

- Provider Assigned (PA)
  - Usually assigned by your ISP, and suballocated by the ISP from a larger block of addresses the ISP has
  - ISP aggregates the announcement upstream
  - Customer usually obtains one PA block from each ISP
- Provider Independent (PI)
  - Sometimes called "Portable" address space
  - Not aggregated by upstream ISPs/Peers and appears as a distinct prefix in the global Internet routing table (***scalability issues***!)
  - Needed for multihoming (pending a better scalable solution)

[Migrating to IPv6, USENIX LISA 2013]

Provider Aggregation eg.

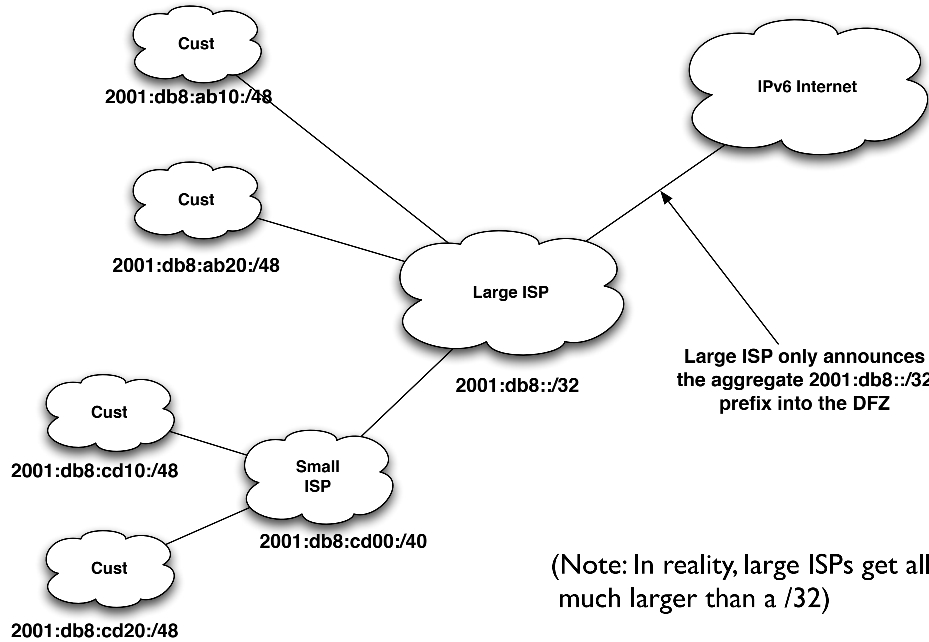
A real example ...

`2001:468::/32`	`Internet2: PI block`
`2001:468:1800::/40`	`MAGPI GigaPop: PA block`
`2001:468:1802::/48`	`University of Pennsylvania: PA block`

Internet2 suballocates the /40 block from its own PI block to MAGPI (a regional ISP), and MAGPI suballocates a /48 from that to its downstream connector UPenn. Internet2 only sends the aggregate /32 announcement to its peers (other large ISPs and organizations), and only that /32 prefix is seen in the global Internet2 routing table.

[Migrating to IPv6, USENIX LISA 2013]

Provider Aggregation eg.



(Note: In reality, large ISPs get allocations much larger than a /32)

[Migrating to IPv6, USENIX LISA 2013]

Multihoming

- Not fully solved; an area of active research & protocol design
- Initial model: everything is provider assigned and aggregatable
- Now **PI** (Provider Independent) address space common
- Future possibilities:
  - **SHIM6** - RFC 5533, 5534, 5535
  - **LISP** - Locator/Identifier Separation Protocol - see IETF wg
  - IRTF routing research group
    - **RFC 6115**: Recommendation for a Routing Architecture
    - **ILNP**: Identifier-Locator Network Protocol (RFC 6740-6748)

[Migrating to IPv6, USENIX LISA 2013]

IPv6 support in network service providers

ISPs offering IPv6

- Some: NTT/Verio, Global Crossing, Level 3, AT&T, Cogent, Cable & Wireless, Reliance, Tata Communications, TeliaSonera, Hurricane Electric, ... (growing list)
- http://www.sixxs.net/faq/connectivity/?faq=ipv6transit
- http://en.wikipedia.org/wiki/Comparison_of_IPv6_support_by_major_transit_providers
- Mixture of native and tunneled IPv6 service
- If you're a US edu, you might be able get IPv6 connectivity via the Internet2 R&E network
  - Equivalent opportunities with other national or continental RENs (JANET, SURFNet, GEANT, APAN etc)

Mobile Carriers

- An increasing number of mobile carriers now support IPv6 (e.g. on their 4G/LTE networks), at least in some parts of their networks
  - Verizon LTE, AT&T, T-Mobile
  - May need specific type of phones/hardware to take advantage of this though

Residential Service

- Not as encouraging, but ...
- Comcast is leading the charge
  - http://www.comcast6.net/
  - As of Nov'13, 75% of Comcast's broadband footprint is IPv6 enabled; 25% customers are actively using IPv6; commercial metro ethernet service IPv6 enabled too
- Who else?
  - AT&T some; Time Warner has limited trials
  - Verizon FIOS has no announced plans yet

Content Delivery Networks

- **LimeLight Networks** supports IPv6
  - claims to be first IPv6 CDN
- **Akamai** announced production IPv6 support in April 2012
  - http://www.akamai.com/ipv6
- **Cloudflare** and **Edgecast** too
- See ISOC's deploy360 page for more:
  - http://www.internetsociety.org/deploy360/resources/ipv6-and-content-delivery-networks-cdns/

Other Cloud

- Amazon Web Services
  - http://aws.amazon.com/about-aws/whats-new/2011/05/24/elb-ipv6-zoneapex-securitygroups/
  - http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/using-elb-ipv6.html
  - No internal infra support, but dualstack on outside facing possible (in some regions)
- Other Cloud providers: Cloudflare, Rackspace,

Hosting Providers

- A growing number offer native IPv6
- Linode, Dreamhost, Softlayer, and many others ...
- Larger lists at:
  - http://www.sixxs.net/wiki/IPv6_Enabled_Hosting
  - http://hosting.4or6.com/

IPv6 Support in popular Operating Systems

Operating System Support

- Most modern operating systems support IPv6 out of the box
- Microsoft Windows, Apple Mac OS X, Linux, *BSD, Solaris, Tru64 UNIX, IBM AIX, etc
- Mobile OSes like iOS, Android do also
- They generally by default, use autoconfiguration or DHCPv6 to configure IPv6 addresses

Windows

- Vista, Windows 7 onwards: IPv6 is by default ON
- Windows XP: turn it on:
    - netsh interface ipv6 install

Mac OS X

- On by default.
- In Mac OS X Lion (released July 2011), both Stateless Address Autoconfiguration and DHCPv6 are supported. In earlier versions only the former was supported.
- Details can be seen in the "Network Preferences" pane of the "Preferences" application, where it is also possible to configure a static IPv6 address and gateway.

Linux

- Most modern versions have IPv6 turned on by default
- Actual details vary, from distribution to to distribution
- RedHat/Fedora/CentOS etc:
  - File: /etc/sysconfig/network:
    - NETWORKING_IPV6=yes
- Many more details in http://www.bieringer.de/linux/IPv6/

Linux: static address

(This example is for Redhat/CentOS/Fedora etc ...)

/etc/sysconfig/network:

```
NETWORKING_IPV6=yes
IPV6_AUTOCONF=no
IPV6_DEFAULTGW=fe80::4
IPV6_DEFAULTDEV=eth0
```

/etc/sysconfig/network-scripts/ifcfg-eth0:

```
IPV6INIT=yes
IPV6ADDR=2001:db8:ab:cd::4/64
```

Manually adding, deleting IPv6 addresses on an interface:

```
ifconfig eth0 add inet6 2001:db8:ab:cd::4/64
ifconfig eth0 del inet6 2001:db8:ab:cd::4/64
```


FreeBSD

Again, IPv6 is on by default, and uses autoconfig.

Static IPv6 address configuration example:
/etc/rc.conf:

```
ipv6_enable="yes"
ipv6_network_interfaces="auto"
ipv6_ifconfig_fxp0="2001:db8:ab:cd::4/64"
```

However, a regression in 9.x: IPv6 not automatically preferred. To fix, put in /etc/rc.conf:
ip6addrctl_policy="ipv6_prefer"

For more details, see:
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-ipv6.html

Solaris

```
Again, IPv6 is on by default.

Interface address configuration file:
/etc/hostname6.<interfacename> eg./etc/hostname6.e1000g0

Some possible contents of this file:

<empty file>          # use stateless autoconfiguration

token ::2:2/64        # Defines the 64-bit IID; network
                      # prefix is derived from RA

addif inet6 2001:db8:ab::1 up    # full static address
```


Apple iOS

- Apple's iOS (iPhone, iPad, etc) supports IPv6 (on Wi-fi interface, and possibly LTE/4G interface)
- Supports both stateless address autoconfiguration, temporary addresses, and DHCPv6.
- Configured automatically; no configuration knobs are provided in the UI.

Android

- Google's Android mobile operating system supports IPv6

IPv6 Support in Application Software

Application Services

- Recall: IPv6 is not backwards compatible with IPv4
- Applications need to be modified to support IPv6
- Many open source & commercial applications already do!
- Don't forget to consider home grown, and locally developed applications also!

IPv6 ready applications

- Webservers: Apache, IIS
- E-mail: Sendmail, Postfix, UW IMAP, Cyrus, MS Exchange, Exim, Qmail, Dovecot, Courier
- DNS: BIND, NSD, PowerDNS, Microsoft DNS
- LDAP: OpenLDAP, Active Directory
- Kerberos: MIT, Heimdal, Active Directory
- More comprehensive lists:
  - http://www.ipv6-to-standard.org/
  - http://www.deepspace6.net/docs/ipv6_status_page_apps.html

IPv6 client software

- Browsers: Firefox, Internet Explorer, Safari, Chrome, Opera
- E-mail: Apple Mail, Thunderbird, MS Outlook
- [others to be added ...]

A few configuration examples ...

Not exhaustive by any means. I'm just showing quick configuration examples of some popular UNIX based software applications.

DNS: ISC BIND

named.conf

only use one
listen-on statement

```
# tell nameserver to listen on IPv6 socket
options {
    listen-on-v6 { any; };
    listen-on-v6 { ::1; 2001:db8:ab:cd::3; };
    ...
};

# example of IPv6 prefixes in an ACL
acl trusted {
    127.0.0.0/8;
    192.168.2.0/24;
    ::1/128;
    2001:db8:ab::/48;
};
```


DNS: ISC BIND

named.conf

```
# an IPv6 reverse zone for 2001:db8:abcd::/48
zone "d.c.b.a.8.b.d.0.1.0.0.2.ip6.arpa" {
    type master;
    file "abcd-reverse.zone";
}
```

DHCP: ISC DHCP

```
option dhcp6.name-servers 2001:db8:1:1::200;
option dhcp6.domain-search "example.com";

# dynamic pool of addresses
subnet6 2001:db8:1:1::0/64 {
        range6 2001:db8:1:1::10 2001:db8:1:1::200;
}

# fixed address assignment to specific client
host ws1 {
        host-identifier option dhcp6.client-id 00:01:00:06:4d:
57:4b:d1:00:03:3a:d5:c7:04;
        fixed-address6 2001:db8:1:1::701;
}

# Dynamic address, and specifically assigned option (DNS)
host ws2 {
        host-identifier option dhcp6.client-id 00:01:00:01:14:ed:
66:c1:08:00:27:94:08:40;
        option dhcp6.name-servers 2001:db8:1:1::201;
}
```

Note: DUID, not MAC addr

Apache Web server

httpd.conf

```
Listen [::]:80
Listen [::]:443
Listen [2001:db8:ab:cd::2]:80


# A dual stack virtualhost stanza example
<VirtualHost 192.68.24.2:80 [2001:db8:ab:cd::2]:80>
   ...
   ...
</VirtualHost>
```

Web browsers

- Most IPv6 capable web browsers today (eg. Firefox, IE, Chrome, Safari, Opera, etc) are by default enabled for IPv6 operation. No special configuration is needed.
- (Note: in older versions of Firefox, IPv6 was disabled by default, and you needed to go to "about:config" and change the value of "network.dns.disableIPv6" from "true" to "false")

SMTP: Sendmail

sendmail.mc:

```
DAEMON_OPTIONS(`Port=smtp, Addr::, Name=MTA-v6,
Family=inet6')

// Then regenerate sendmail.cf and restart sendmail
```

SMTP: Postfix

Postfix 2.2 onwards supports IPv6. As of this writing, by default it uses IPv4 only; IPv6 has to be turned on explicitly.

```
main.cf:

# Enable IPv4 and IPv6 if supported
# choices are: ipv4, ipv6, all
inet_protocols = all

mynetworks = 192.168.0.0/16, [2001:db8:abcd::]/48
```

Many more details can be found at:
http://www.postfix.org/IPV6_README.html

IMAP: UW IMAP

University of Washington's IMAP server software supports IPv6, but if you compile from source, you may need to specify IP=6 in your "make" command.

Check your Linux/BSD/UNIX distribution though. They have already built UW imapd with IPv6 support. This is true in recent versions of Fedora Linux for example.

IMAP: Cyrus

IPv6 support can be enabled by service in cyrus.conf:

```
proto = tcp # tcp, udp, tcp4, udp4, tcp6, udp6
```


LDAP: OpenLDAP

OpenLDAP version 2.x onwards supports IPv6.

slapd supports a “-6” command line option for IPv6 only operation.

Kerberos

MIT Kerberos
MIT Kerberos has had support for IPv6 in the KDC for many releases.

More complete support is in the latest release (v.1.9), where the Kerberos administration server (kadmind) and propagation server (kpropd) also support IPv6, and IPv6 addresses can be directly specified in the configuration files if needed.

For details, see http://k5wiki.kerberos.org/wiki/IPv6

Heimdal
Heimdal also supports IPv6.

tcp wrappers

tcp wrappers is a popular access control facility for internet services on UNIX platforms.

Use the syntax [IPv6prefix]/prefixlength in the tcp wrappers configuration files **/etc/hosts.allow** and **/etc/hosts.deny**. IPv4 and IPv6 prefixes can be mixed on the same lines, eg.

```
sshd:   192.168.0.0/255.255.0.0 [2001:db8:ab::]/48
imapd:  192.168.4.0/255.255.255.0 [2001:db8:ab:cd::]/64
```

IPv6 Tunneling

Automatic Tunneling

- Even without IPv6 deployed in your network, computers may be using IPv6
- Via automatic tunneling mechanisms. Two popular ones are **6to4** and **Teredo**
- These work by **encapsulating** IPv6 packets inside IPv4 packets and sending them to a relay router that is connected to both the IPv4 and IPv6 Internet
- **<u>Tunnels sometimes cause connectivity and performance problems. Native IPv6 deployment usually fixes all of them</u>**

6to4

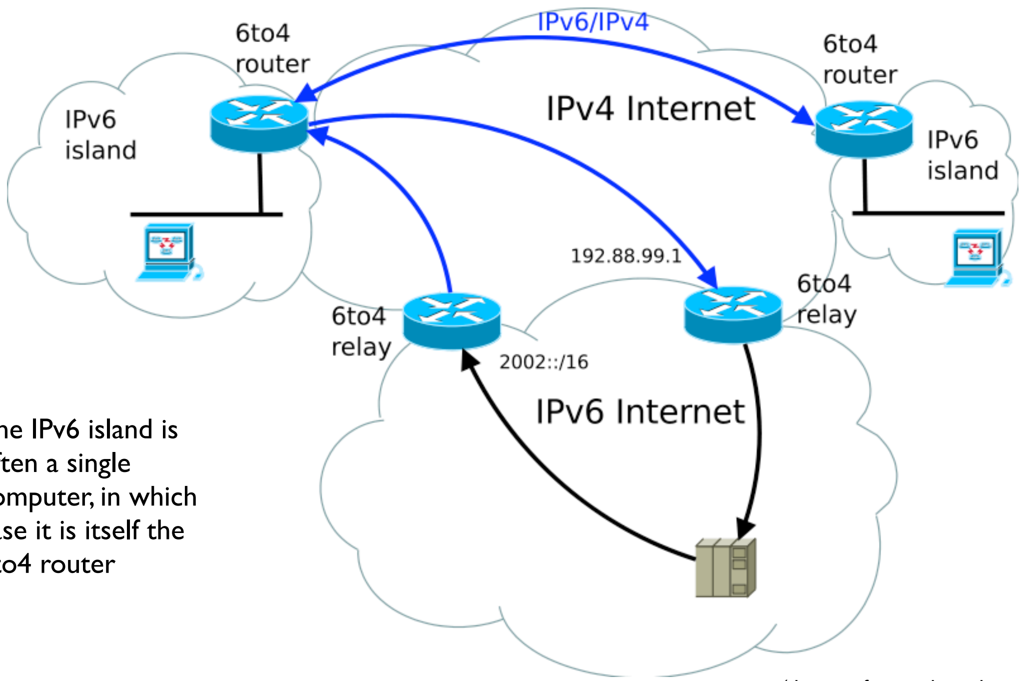
- A transition method for IPv6 capable hosts or networks that don't have native IPv6 network connectivity to use tunneling to communicate with other IPv6 islands and/or the IPv6 Internet
- Does not involve explicit setup of the tunnels.
- 6to4 hosts and networks are numbered in the **2002::/16** prefix
- **6to4 routers** sit at the edge of an IPv6 site and the IPv4 Internet
- The most common deployment model of 6to4 involves using 6to4 anycast addresses to reach **6to4 relay routers**
  - **192.88.99.1** and **2002:c058:6301::**

6to4

- Site constructs a /48 IPv6 prefix by concatenating 6to4 router's IPv4 address to 2002::/16, and tunnels IPv6 packets from the 6to4 router to a 6to4 relay router that is connected to both the IPv4 and IPv6 Internet.
- A site could be a single computer, in which case it is itself the 6to4 router

References:
RFC 3056: Connection of IPv6 domains via IPv4 clouds
RFC 3068: An anycast prefix for 6to4 relay routers
RFC 6343: Advisory Guidelines for 6to4 deployment

6to4 Diagram



The IPv6 island is often a single computer, in which case it is itself the 6to4 router

(diagram from wikimedia commons)

6to4 Addressing example

```
Example of a single computer acting as a 6to4 router.
IPv4 address: 203.0.113.5 (in hex: cb007105)
6to4 network prefix is: 2002:cb00:7105::/48   (2002::/16 + 32-bit IPv4)
Configure my IPv6 address as (subnet 1, interface-id 1)
My IPv6 address: 2002:cb00:7105:1::1
6to4 relay anycast IPv4 address: 192.88.99.1
6to4 relay anycast IPv6 address: 2002:c058:6301::

To send a packet to 2001:db8:ab:cd::3, the computer encapsulates the IPv6
packet inside an IPv4 packet that is sent to the 6to4 relay IPv4 address:

 IPv4 src = 203.0.113.5             IPv4 dst = 192.88.99.1
 IPv6 src = 2002:cb00:7104:1::1     IPv6 dst = 2001:db8:ab:cd::3

The relay router decapsulates the IPv6 packet and forwards it natively to
the IPv6 destination.

Return IPv6 traffic is directly natively to a (probably different) 6to4
relay router, which derives the destinations's IPv4 address from the 6to4
address, and encapsulates the IPv6 packet in an IPv4 header directed to
the 6to4 host's IPv4 address.
```

6to4 Addressing example

6to4 host to relay

IPv4 Header	IPv6 Header	IPv6 Payload
Src=203.0.113.5 Dst=192.88.99.1 Proto=41 (IPv6 encap)	Src=2002:cb00:7105:1::1 Dst=2001:db8:ab:cd::3	TCP, UDP, etc

relay to v6 host

IPv6 Header	IPv6 Payload
Src=2002:???? Dst=192.88.99.1	TCP, UDP, etc

v6 host to relay

IPv6 Header	IPv6 Payload
Src=2001:db8:ab:cd::3 Dst=2002:cb00:7105:1::1	TCP, UDP, etc

relay to 6to4 host

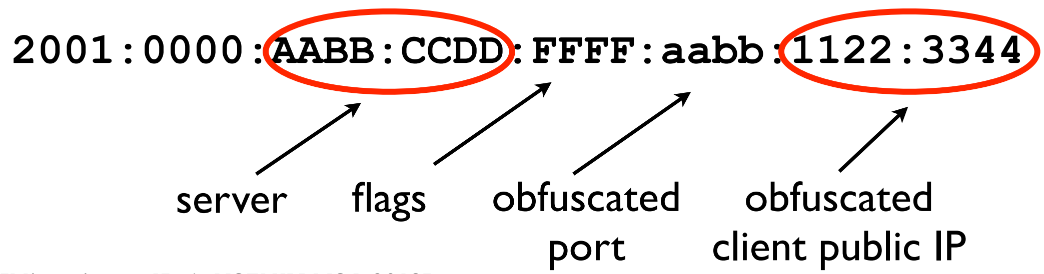
IPv4 Header	IPv6 Header	IPv6 Payload
Src=192.88.99.1 Dst=203.0.113.5 Proto=41 (IPv6 encap)	Src=2001:db8:ab:cd::3 Dst=2002:cb00:7105:1::1	TCP, UDP, etc

6to4 Issues

- 6to4 can fail or perform poorly due to a variety of reasons:
  - Inbound/outbound blackholes (routers or firewalls filtering protocol 41, ICMP etc)
  - Lack of working return 6to4 relay
  - Circuitous/Asymmetric path with large round trip time
  - PMTU failures due to encapsulation overhead etc
- Privacy concerns with 3rd party relay routers
- See RFC 6343: Advisory Guidelines for 6to4 Deployment

Teredo

- Encapsulates IPv6 in UDP in IPv4 (see RFC 4380 for details)
- Works through NATs
- Special IPv6 prefix 2001::/32 (ie. 2001:0000::/32)
- Uses Teredo Servers and Teredo Relays



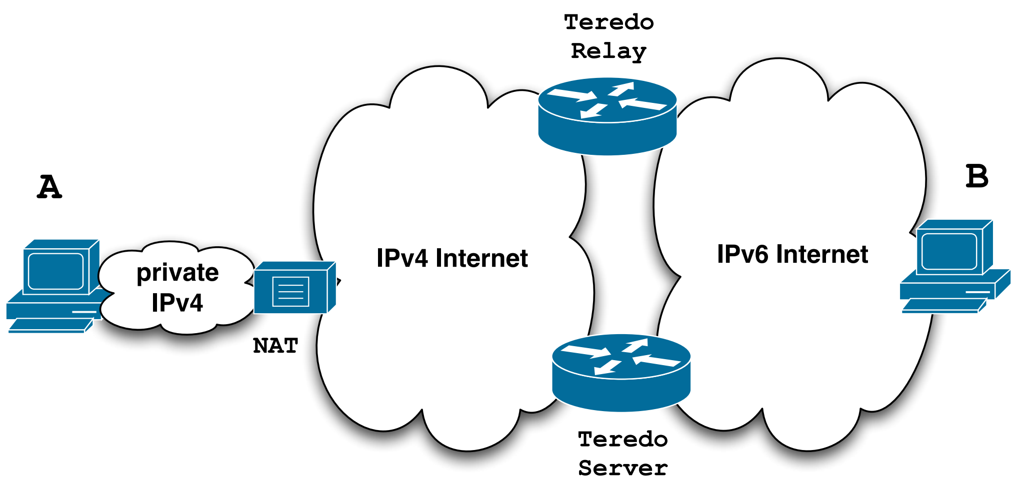
[Migrating to IPv6, USENIX LISA 2013]

Teredo

- **Teredo Servers** are used for initialization, testing type of NAT, determining client's externally routable address, and for periodically maintaining state in NATs/firewalls
- **Teredo Relays** are used for relaying tunneled traffic to and from the IPv6 Internet

[Migrating to IPv6, USENIX LISA 2013]

Teredo Diagram



Teredo Issues

- Cannot work through some types of NAT (eg. Symmetric)
- NAT detection and traversal mechanisms employed have a significant impact on network performance
- Possible issues with inoperable Teredo servers and relays
- Privacy concerns with 3rd party servers and relays
- Security concerns have been expressed:
  - http://tools.ietf.org/html/draft-ietf-v6ops-teredo-security-concerns-02

Teredo farewall

- Teredo is going away (teredo.ipv6.microsoft.com)
- In preparation for shutting down Teredo in the near future, a sunsetting experiment was performed (Microsoft), July 15th-19th 2013:
  - http://www.ietf.org/proceedings/87/slides/slides-87-v6ops-5.pdf
- There was negligible effect on IPv6 traffic
- Final date for decommission to be set
- (But .. recent XBox news)

Identifying tunneled traffic

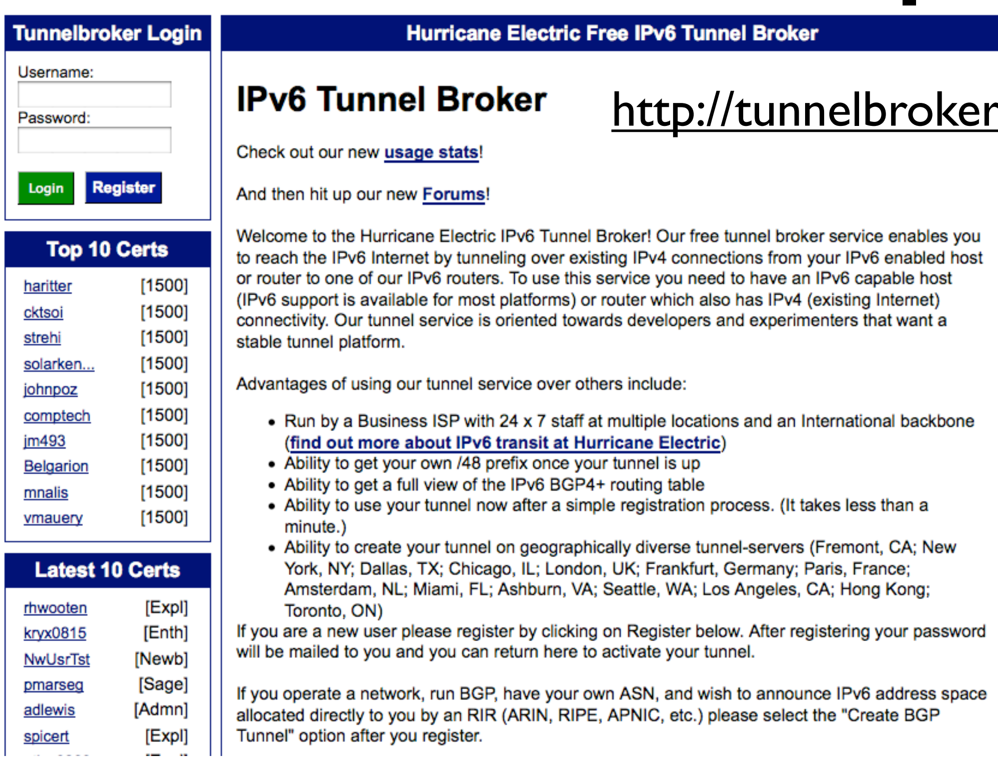
- 6to4 uses well known prefix 2002::/16
- Teredo uses 2001::/32
- Both use value 41 (IPv6 encapsulation) in the IPv4 protocol field
- 6to4 encapsulates IPv6 packets directly in IPv4
- Teredo is encapsulated in UDP inside IPv4
- 6to4 commonly uses well-known anycast relay routers (192.88.99.0/24)
- There are also public Teredo servers and relays
- *Note: blindly blocking tunneled traffic may cause more harm than good*

Managed tunnels

- Statically configured, managed, IPv6 in IPv4 tunnels usually provide more predictable and more reliable service. A few managed tunnel providers

- Hurricane Electric: www.tunnelbroker.net
- Freenet6: www.hexago.com
- Consulintel: tb.consulintel.euro6ix.org
- Sixxs: www.sixxs.net

HE tunnelbroker example



http://tunnelbroker.net/

Managed tunnel example

your ipv4 addr. if behind NAT, it's the externally visible address

```
Tunnel providers endpoint:
    IPv4: 203.0.113.1
    IPv6: 2001:db8:ab:cd::1
Your endpoint:
    IPv4: 192.0.5.2
    IPv6: 2001:db8:ab:cd::2
Prefix assignment: 2001:db8:ab:de::/64 (or /56, /48 etc)
```

```
ifconfig gif0 tunnel 192.0.5.2 203.0.113.1
ifconfig gif0 inet6 2001:db8:ab:cd::2 2001:db8:ab:cd::1 prefixlen 128
route -n add -inet6 default 2001:db8:ab:cd::1

$ ifconfig gif0
gif0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1280
        tunnel inet 192.0.5.2 --> 203.0.113.1
        inet6 fe80::ca2b:a4ff:fe16:52c7%gif0 prefixlen 64 scopeid 0x2
        inet6 2001:db8:ab:cd::2 --> 2001:db8:ab:cd::1 prefixlen 128
```


Address Selection

DualStack Address Selection

- I'm a dual stack (IPv4/IPv6) client
- I lookup "www.example.com" eg. using **getaddrinfo**()
  - Performs both A and AAAA DNS queries and may return a list of various IPv4 and IPv6 addresses
  - Which should I try connecting to? In what order?

DualStack Address Selection

- RFC 6724: Default Address Selection Algorithm
  - Updated from the original RFC 3484
- Many rules, but one effect is to generally prefer IPv6 over IPv4

Type	Prefix	Precedence	Label
Loopback	::1/128	50	0
IPv6	::/0	40	1
IPv4	::ffff:0:0/96	35	4
6to4	2001::/16	30	2
Teredo	2001::/32	5	5
ULA	fc00::/7	3	13
Site Local	fec0::/10	1	11
6Bone	3ffe::/16	1	12

Happy Eyeballs

- RFC 6555, 6556: Happy Eyeballs: Success with Dual Stack Hosts
  - Parallel connections to v4 & v6 destinations, but give v6 a small headstart or pref. Use first connection that succeeds & cache results; tunable knobs
- Apple Mac OS X Lion:
  - Not quite Happy Eyeballs: no preference for IPv6 over IPv4; use what seems to work best, leading to more non-deterministic behavior
- Windows: http://blogs.msdn.com/b/b8/archive/2012/06/05/with-ipv6-in-windows-8.aspx
- Survey of what various OS and apps used to do/currently do (G. Huston, RIPE64): https://ripe64.ripe.net/presentations/78-2012-04-16-ripe64.pdf
- Traditional resolver vs "Connect-by-Name" APIs

Migration strategies for IPv6 services

- DualStack migration is the ideal, but has possible issues if broken IPv6 client connectivity is widespread
- An overview of some alternate strategies given here:
  - RFC 6589: Considerations for Transitioning content to IPv6
  - DNS Resolver Whitelisting; Resolver Blacklisting; IPv6 specific service names, etc

IPv6 and Security

IPv6 Security issues

- IPsec myth (IPv6 is automatically more secure because of IPsec)
- Code and implementations may not be as well tested in production and at scale, leading to bugs and possible security issues
- Lack of maturity of IPv6 support in (some) firewalls, VPNs, IDS, IPS
- Lack of DNS Block Lists, geolocation, reputation services
- Defensive (or offensive) network scanning: see RFC 5157
- State of support of PCI and other regulatory requirements

IPv6 Security issues

- How to correlate network addresses with users, in the face of auto-configuration, temporary addresses, larger address space per subnet
- Local subnet attacks - these are not qualitatively different from what we have in IPv4 today. See RFC 3756 for IPv6 ND based threats.
- Potential covert channel concerns
- Network scanning and router ND queue saturation (DoS)
  - See RFC 6583: Operational problems with neighbor discovery
- Good general discussion of issues and available solutions:
  - https://wikispaces.psu.edu/display/ipv6/IPv6+security

IPv6 Security issues

- Operational security considerations for IPv6 Networks:
  - http://tools.ietf.org/html/draft-ietf-opsec-v6-00
- Security concerns with native and tunneled traffic:
  - http://tools.ietf.org/html/draft-ietf-opsec-ipv6-implications-on-ipv4-nets-00
- Security implications of IPv6 fragmentation and ND:
  - http://tools.ietf.org/html/draft-ietf-6man-nd-extension-headers-01

ICMPv6 filtering

- ICMPv6 is critical to the operation of IPv6 networks
- Used for many functions: Neighbor discovery, router discovery, Path MTU discovery, multicast group membership management (MLD), Mobile IPv6, and more
- Don't blindly block ICMPv6
- RFC 4890: Recommendations for Filtering ICMPv6 Messages in Firewalls

Rogue RA issue

- Frequently observed phenomenon at some sites
- Most incidents appear to be unintentional misconfiguration rather than malicious
- Appears to be associated with Internet Connection Sharing features in some operating systems
- RFC 6104: Rogue RA problem statement
- Defenses: ACLs, RAGuard (RFC 6105), tweak default router preferences (RFC 4191)
- SeND (cryptographic protocol - challenging to deploy)

Rogue RA vs Rogue DHCP

- IPv4 has to deal with rogue DHCP servers
- Is the situation worse or better with IPv6?
- IPv6 has to deal with both rogue RA and rogue DHCP
- RAs can impact a larger number of hosts faster
- DHCP clients generally have to wait for lease timers to expire
- But, recovery/mitigation can be faster with RA

IPv6 Firewalls

- Stateful Firewalls
- Network vs host based firewalls
- RFC 6092: simple security in IPv6 residential CPE
  - by default block unsolicited incoming except IPsec
- Advanced security CPE?
  - http://tools.ietf.org/html/draft-vyncke-advanced-ipv6-security-02

IPv6 Firewalls

- Status of open source and commercial firewall implementations (Sep 2009, European Conference on Applied IPv6):
  - www.guug.de/veranstaltungen/ecai6-2007/slides/2007-ECAI6-Status-IPv6-Firewalling-PeterBieringer-Talk.pdf
- Survey of IPv6 Availability on Commercial Firewalls (ICANN, March 2010)
  - http://www.icann.org/en/announcements/announcement-2-01mar10-en.htm
- NSA Firewall Design Considerations (July 2010)
  - www.nsa.gov/ia/_files/ipv6/I733-041R-2007.pdf

IPv6 Firewalls

- Inability to deal with long extension header chains (by some firewalls and packet filtering routers)
  - http://tools.ietf.org/html/draft-wkumari-long-headers-01
- Effort to even deprecate fragments!
  - http://tools.ietf.org/html/draft-bonica-6man-frag-deprecate-02

~~Attack~~ Tools
Vulnerability Assessment

- THC-IPv6: http://freeworld.thc.org/thc-ipv6/
- IPv6 Toolkit (SI Networks) http://www.si6networks.com/tools/
- scapy - packet manipulation tool
  - http://www.secdev.org/conf/scapy-IPv6_HITB06.pdf
- Note: attacks using IPv6 are already going on today; even on networks that haven't yet deployed IPv6
  - http://tools.ietf.org/html/draft-gont-opsec-ipv6-implications-on-ipv4-nets-00
  - RFC 6169: Security concerns with IPv6 tunneling

Attacks are happening

- IPv6 DDoS attacks observed on the Internet
  - 2012-02-22 Arbor: IPv6 sees first DDoS attacks
  - http://www.h-online.com/security/news/item/Report-IPv6-sees-first-DDoS-attacks-1440502.html
  - http://www.zdnet.com/blog/networking/first-ipv6-distributed-denial-of-service-internet-attacks-seen/2039
- Various forms of IPv6 malware
  - Using IPv6 as covert channel to communicate with botnet controller
  - including one that advertises a host as an IPv6 router and uses v4-v6 transition mechanisms to hijack both IPv4 and IPv6 traffic through it!

Troubleshooting tools

Troubleshooting Tools

- ifconfig
- tcpdump, wireshark, tshark
- ndp, ip -6, route, netstat, ...
- ping, ping6
- traceroute, traceroute6, tracert, tracepath6
- ndisc6 (ndisc6, rdisc6, tcptraceroute6, rdnssd)
- scamper - great for detecting PMTU blackholes in the network
- scapy - scriptable packet injection tool

Neighbor cache

```
MacOSX$ ndp -an
Neighbor                        Linklayer Address  Netif Expire    St Flgs
Prbs
2607:f470:2f:1:215:4fff:fe01:33f8 0:15:4f:1:32:e8     en0 23h59m58s S
2607:f470:2f:1:218:f2ff:fe09:458c 0:18:f2:9:45:8c     en0 permanent R
fe80::1%lo0                     (incomplete)         lo0 permanent R
fe80::214:dfff:fe01:32f8%en0    0:14:4f:1:32:f9      en0 17h48m51s S
fe80::216:9cff:fe7f:53c0%en0    0:1e:9c:6f:53:c0     en0 17s       R  R
fe80::219:f2ff:fe09:458c%en0    0:1d:f2:9a:44:7c      en0 permanent R
```

```
Fedora-Linux$ ip -6 neigh show
fe80::216:9cff:fe6f:5dc0 dev eth0 lladdr 00:17:9c:6e:5d:c0 router STALE
2607:f470:2e:1:217:f2ff:fd09:458c dev eth0 lladdr 00:17:f2:09:4d:83 REACHABLE
fe80::21b:c000:1e83:b800 dev eth1 lladdr 00:1b:c0:84:b8:00 router STALE
```

```
Windows$ netsh interface show neighbors
```


netstat (mac)

```
MacOSX$ netstat -rn -f inet6

Destination                            Gateway                           Flags
Netif Expire
default                                fe80::216:9cff:fe6d:5ec1%en0     UGSc   en0
::1                                    ::1                              UH     lo0
2607:f470:2f:1::/64                     link#4                           UC     en0
2607:f470:2f:1:217:f2ff:fe09:457c       0:17:fd:9:45:8c                  UHL    lo0
fe80::%lo0/64                          fe80::1%lo0                      Uc     lo0
fe80::1%lo0                            link#1                           UHL    lo0
fe80::%en0/64                          link#4                           UC     en0
fe80::217:f2df:fe09:458c%en0           0:17:fd:9:45:8c                  UHL    lo0
ff02::/32                              ::1                              UmC    lo0
ff02::/32                              link#4                           UmC    en0
ff02::fb                               link#4                           UHmLW  en0
```

netstat (linux)

```
Linux$ netstat --protocol=inet6 -rn
Kernel IPv6 routing table
Destination                                 Next Hop
Flags Metric Ref    Use Iface
2001:468:1800:501::/64                      ::
UA    256    1462       0 eth1
2607:f470:2f:1:218:f2ff:fea9:358c/128        2607:f470:2d:1:217:f2ff:fea9:4d8c
UAC   0      8        1 eth0
2607:f470:2f:1::/64                          ::
UA    256    3591       0 eth0
fe80::/64                                   ::
U     256    0        0 eth0
fe80::/64                                   ::
U     256    0        0 eth1
::/0                                        fe80::216:9cff:fe6f:5ec0
UGDA  1024   11266       0 eth0
::/0                                        fe80::21b:c000:1e83:bc00
UGDA  1024   1        0 eth1
::1/128                                     ::
U     0      14192       1 lo
[ ... rest deleted ... ]


Also see: ip -6 route show, route -A inet6
```


IPv4 IPv6 Transition & Co-existence mechanisms

Transition/Co-existence

- The original IPv4-IPv6 transition and co-existence plan was based on the "Dual Stack" model.
- Since the dual stack transition has unfortunately failed to occur in a timely fashion, more drastic mechanisms are being developed and deployed.

Transition/Co-existence

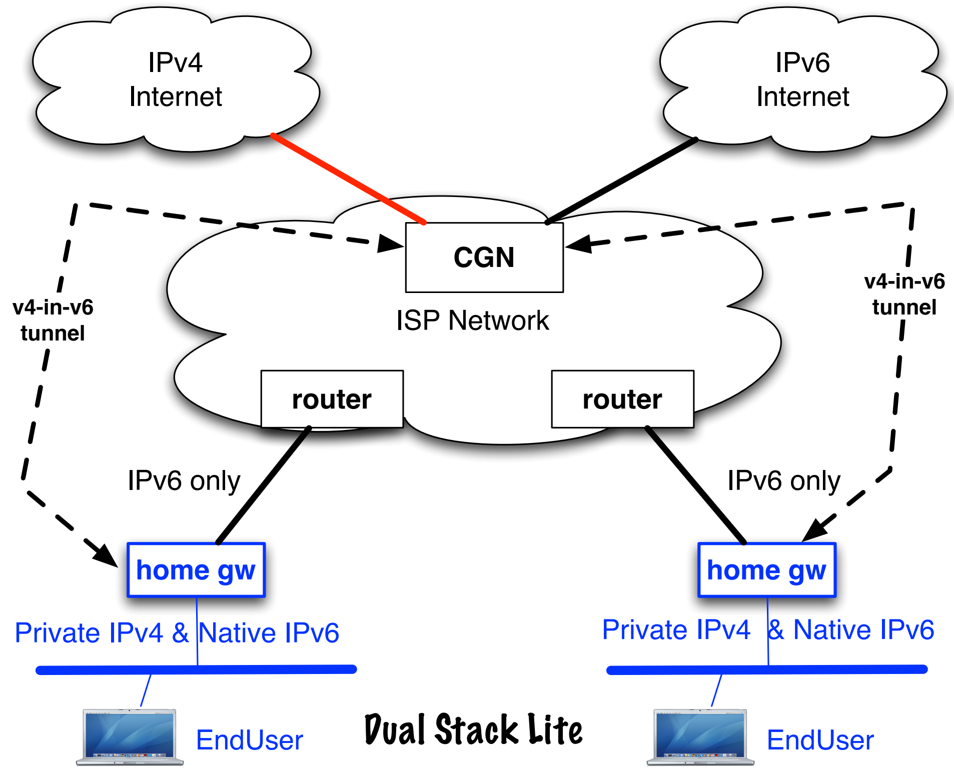
- 6RD
- NAT64 and DNS64 (Note: NAT-PT deprecated by RFC 4966)
- IVI
- Dual Stack Lite (DS Lite)
- A+P
- MAP
- 464XLAT

6rd (rapid deployment)

- RFC 5569: IPv6 Rapid Deployment on IPv4 infrastructures
- Essentially, a modified version of 6to4 Tunneling
- Managed by an ISP, using the ISP's prefix, rather than 2002://16
- More predictable and reliable performance
- Easier to support by the ISP
- Major deployment by Free Telecom in France

DualStack Lite

- Combines **Native IPv6** and **tunneled IPv4** + **centralized IPv4 NAT**
- No IP protocol family translation. Clients expected to be dualstack.
- CPE doesn't perform NAT function
- Share IPv4 addresses among **multiple** customers with a **"Carrier Grade NAT" (CGN)**
- Alternative to cascading NATs (NAT444 etc) for some ISPs
- Implications of address sharing
- http://www.isc.org/software/aftr



[Migrating to IPv6, USENIX LISA 2013]

A+P (Address + Port)

- RFC 6346: The Address plus Port (A+P) Approach to the IPv4 Address Shortage (status: experimental)
- Similar in goals to Dual-Stack Lite, but absent some of the more nasty scalability limitations of carrier grade NATs
- Replace centralized CGN with an A+P gateway (non NAT)
- Return IPv4 NAT function to CPE, but constrain its port mapping to a subset of the 16-bit port space
- With the other bits identifying the CPE to the ISP network (ie. use a shared IPv4 address plus some port bits to identify the CPE)
- Tunnel CPE traffic over IPv6 to A+P gateway

[Migrating to IPv6, USENIX LISA 2013]

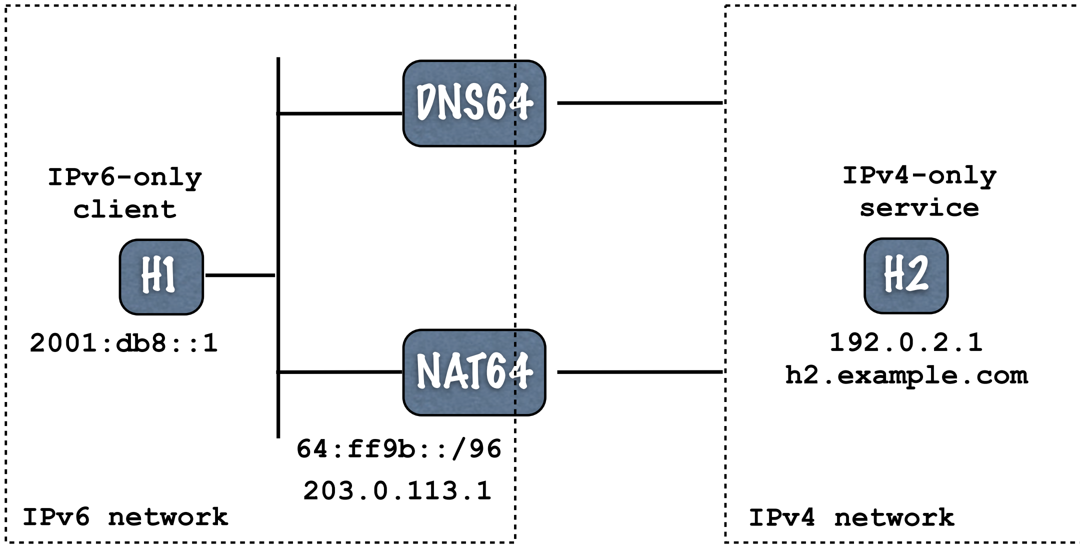
MPLS and 6PE

- RFC 4659
- A possible transition mechanism for an ISP that hasn't fully deployed IPv6 in its core network
- Run IPv6 capable Provider Edge (PE) routers to peer natively with IPv6 customers and external peers
- Use BGP/MPLS VPN to forward traffic using MPLS over interior network that has IPv4 only core routers
- (IPv6-only MPLS has some gaps)

NAT64, DNS64

- RFC 6052, 6144, 6145, 6146, 6147
- 6052: IPv6 addressing of IPv4/IPv6 translators
- 6145: IP/ICMP stateless translation
- NAT64: Stateful Network address and protocol translation ***from IPv6 clients to IPv4 servers*** (RFC 6146)
- Well known prefix: **64:ff9b::/96**
- DNS64: DNS extensions for NAT from IPv6 clients to IPv4 servers
  - synthesizes AAAA from A DNS records
- An open source implementation: http://ecdysis.viagenie.ca/

NAT64, DNS64




```
64:ff9b::192.0.2.1 <-- 192.0.2.1
h2.example.com IN AAAA 64:ff9b::192.0.2.1
```

[Migrating to IPv6, USENIX LISA 2013]

NAT64, DNS64

```
IPv6-only client, H1 is attempting to reach IPv4-only webserver H2
(h2.example.com)

H1: perform DNS lookup of "h2.example.com" AAAA
DNS64: lookup h2.example.com, see that it only has an A record,
  convert the IPv4 address to 64:ff9b::192.0.2.1 and return that
  in the AAAA DNS answer
H1: send packet to 64:ffb9::192.0.2.1, port 80 from source
  2001:db8::1, port 1500. Packet gets routed to NAT64 device
NAT64: select an unused port, say 2000, on its IPv4 address,
  203.0.113.1 and create the NAT mapping between H1's source
  IPv6 address and port (2001:db8::1, 1500) and the NAT64's
  IPv4 address and selected port (203.0.113.1, 2000)
  Translate IPv6 header into IPv4 header (using RFC 6145 algorithm)
  Send translated packet with source 203.0.113.1, 2000 to
  destination 192.0.2.1, 80 (H2)

Return traffic:
H2: send packet from source 192.0.2.1, port 80 to 203.0.113.1, port 2000
NAT64: Receives packet, look for mapping entry for 203.0.113.1, port 2000
  Finds (2001:db8::1, 1500 <-> 203.0.113.1, 2000)
  Translate IPv4 header to IPv6 header
  Send packet to H1 using source 64:ff9b::192.02.1, 80 and
  destination 2001:db8::1, 1500
```

[Migrating to IPv6, USENIX LISA 2013]

IVI: RFC 6219

- IVI
- IV = 4, VI = 6, so IVI is IPv4 IPv6 transition
- Published as informational RFC 6219 (Not an IETF standard)
- Deployed in China's Research & Education Network, CERNET
- Working translator code for Linux

464XLAT

- 464XLAT: Combination of Stateful and Stateless Translation

```
  [Trying to become an official wg document in softwires wg (2012-02)]

                               ----
                              | v6 |
                               ----
                                 |
 ----      |                 .---+---.                    .------.
| v6 |-----+                /         \                  /        \
 ----      |    ------     /   IPv6    \     ------     /   IPv4   \
           +---| CLAT |---+  Internet   +---| PLAT |---+  Internet  |
 -------   |    ------     \           /     ------     \           /
|v4p/v6 |--+                `---------'                  `----+----'
 -------   |                                                  |
 -----     |                                                -----
| v4p |----+                                               | v4g |
 -----     |                                                -----

       <- v4p -> XLAT <--------- v6 --------> XLAT <- v4g ->

  v6  : Global IPv6; v4p : Private IPv4; v4g : Global IPv4
```

NAT444/CGN/LSN Issues

- NAT a single point of failure; easy DoS target etc
- Issues with address sharing across multiple customers (see RFC6269)
- Broken applications; ALGs complex, NAT traversal methods don't always work reliably
- Network management, troubleshooting, auditing is more difficult
- Broken location aware services
- Poor performance and/or reliability (overloaded/malfunctioning middleboxes)
- RFC 7021: Assessing the impact of CGN on network applications
- DNSSEC issues (with DNS64 AAAA record synthesis)

NPTv6 (formerly NAT66)

- Technically no NAT in IPv6, but ...
- RFC 6296: NPTv6: **IPv6 to IPv6 Network Prefix Translation** (status: **Experimental**)
- Works very differently from v4-v4 NAT/NAPT
  - Stateless 1:1 ***prefix*** translation
  - goal: address independence & easier multihoming (see related draft on multi-homing with NPTv6)
- Alleged security properties of NAT can be better achieved with stateful firewalls -- see RFC 4864 (Local network protection for IPv6) for details

Multicast Transition

- Multicast IPv4-IPv6 transition mechanisms - new work in the IETF
- (Really we should just deploy IPv6 quickly and stop spending endless cycles inventing new transition mechanisms!)

Parting advice for IPv6 deployers

Preparing for IPv6

- Start early; Get management buy-in
- Develop a deployment plan
- Training for your staff and users
- Ordering/updating hardware & software
- Installing/testing/debugging hardware and software

http://www.ripe.net/ripe/docs/ripe-554

https://spaces.internet2.edu/display/~benchoff@vt.edu/IPv6+Information+Technology+Acquisition

Programming

Socket API extensions

- RFC 3493: Basic Socket Interface Extensions for IPv6
- New socket address structures, new address conversion functions, new socket options
- Also see RFC 4038: Application Aspects of IPv6 Transition
- RFC 5014: IPv6 Socket API for Source Address Selection

Socket API extensions

IPv6 address structure:

```
struct in6_addr {
    uint8_t             s6_addr[16];    /* IPv6 address */
};


struct sockaddr_in6 {
    unsigned short int  sin6_family;    /* AF_INET6 */
    in_port_t           sin6_port;      /* L4 port */
    uint32_t            sin6_flowinfo;  /* flow info */
    struct in6_addr     sin6_addr;      /* IPv6 address */
    uint32_t            sin6_scope_id;  /* scope id */
};
```

The flowinfo and scope_id fields are new. in6_addr is actually defined in terms of unions for alignment purposes.

Socket API extensions

New versions of functions that translate names to/from addresses and between numeric and textual address forms. Take an address family arg(AF_INET, AF_INET6, AF_UNSPEC)

IPv4	IPv4 & IPv6
gethostbyname()	getaddrinfo()
gethostbyaddr()	getnameinfo()
inet_ntoa()	inet_ntop()
inet_addr()	inet_pton()

[Migrating to IPv6, USENIX LISA 2013]

Socket API extensions

Note: if IP address family is unspecified, getaddrinfo() on most platforms returns its list of addresses sorted in the order dictated by the default address selection algorithm. But note the presence of newer "Happy Eyeballs" style algorithms.

```
$ python
>>> import socket
addrinfo_list = socket.getaddrinfo("www.ucla.edu", 80,
                 socket.AF_UNSPEC, socket.SOCK_STREAM)
>>> for (fam, stype, proto, canon, saddr) in addrinfo_list:
...     print saddr[0]
...
2607:f010:3fe:201:0:ff:fe01:32
2607:f010:3fe:101:0:ff:fe01:32
169.232.33.224
169.232.55.224
```

Returns 4 addresses, with the IPv6 addresses first.

Client applications (normally) should implement code to loop through the various addresses returned by getaddrinfo() until they succeed in establishing a connection.

[Migrating to IPv6, USENIX LISA 2013]

Socket API extensions

```
Replace:
        socket(AF_INET, SOCK_STREAM, 0)      # TCP socket
        socket(AF_INET, SOCK_DGRAM, 0)       # UDP socket
with:
        socket(AF_INET6, SOCK_STREAM, 0)
        socket(AF_INET6, SOCK_DGRAM, 0)
```

sockaddr_in6 structures will be used. These structures are passed as opaque pointers (sockaddr) in socket functions. And other functions like bind(), connect(), sendmsg(), sendto(), accept(), recvfrom(), recvmsg(), getpeername(), getsockname(), etc can mostly be used unchanged.

Socket options

New socket options that can be used by the **setsockopt**() and **getsockopt**() functions:

```
IPV6_UNICAST_HOPS       #set unicast hoplimit (TTL)
IPV6_MULTICAST_IF       #set outgoing interface for multicast
IPV6_MULTICAST_HOPS     #set hoplimit for outgoing multicast
IPV6_MULTICAST_LOOP     #loop back multicast to myself
IPV6_JOIN_GROUP         #join multicast group on interface
IPV6_LEAVE_GROUP        #leave multicast group
IPV6_V6ONLY             #restrict socket to IPv6 only
```

The “IPPROTO_IPV6” level constant must be used. Example:

```
  int hoplimit = 20;
  if (setsockopt(s, IPPROTO_IPV6, IPV6_UNICAST_HOPS,
               (char *) &hoplimit, sizeof(hoplimit)) == -1)
     perror("setsockopt IPV6_UNICAST_HOPS");
```

IPv4 compatibility

IPv6 applications can interoperate with IPv4 nodes using the **IPv4-mapped IPv6 address format, ::ffff:0:0/96** where the IPv4 address is encoded in the last 32 bits, eg:

```
        ::ffff:192.168.1.2
```

Applications can use IPv6 sockets to communicate with IPv4 systems by encoding their IPv4 addresses in this format. When IPv6 sockets receive packets from IPv4 nodes, socket functions that return peer addresses will automatically represent them as IPv4-mapped IPv6 addresses.

To restrict a socket to IPv6 packets only, set the IPV6_V6ONLY socket option via:

```
        setsockopt(s, IPPROTO_IPV6, IPV6_V6ONLY, ...)
```


Advanced extensions

- RFC 3542: Advanced Sockets API for IPv6
- Defines additional functions that deal with more detailed IPv6 information, such as access to variety of IPv6 and ICMPv6 header fields, extension headers, send & receive interfaces, "raw" sockets, path MTU, etc.
- "Ancillary Data" framework to exchange additional information between kernel and application

A small example program

Small demonstration client & server program written in Python. C and perl code are similar. I chose Python for this because it is more compact, readable and resembles pseudo-code.

It's a TCP echo server and client. The server simply echos back whatever the client writes to it. The server can handle both IPv6 and IPv4 connections. The client uses getaddrinfo to obtain all the addresses (IPv4 & IPv6) associated with the server name and tries them in order until one succeeds in connecting.

The server is started with a specified port number:
./echoserver 8080

The client is started with the server name, port & a string:
./echoclient server.example.com 8080 Hello

echoserver

```
#!/usr/bin/env python
import sys, socket

try:
    PORT = int(sys.argv[1])
except:
    print "Usage: echo6server <port>"
    sys.exit(1)

s = socket.socket(socket.AF_INET6, socket.SOCK_STREAM, socket.IPPROTO_TCP)
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(('', PORT))
s.listen(2)

print "Listening on port %d" % PORT
while True:
    conn, addr = s.accept()
    print 'Connection on: ', addr
    data = conn.recv(1024)
    conn.send(data)
    conn.close()
```

echoclient

```
#!/usr/bin/env python

import os, sys, socket, time
try:
    HOST, PORT, MSG = sys.argv[1:]
    PORT = int(PORT)
except:
    print "Usage: echo6client <host> <port> <message>"; sys.exit(1)

ai_list = socket.getaddrinfo(HOST, PORT, socket.AF_UNSPEC, socket.SOCK_STREAM)
for ai in ai_list:
    family, socktype, proto, canonname, sockaddr = ai
    addr, port = sockaddr[0:2]
    try:
        s = socket.socket(family, socktype, proto)
    except socket.error, diag:
        continue
    try:
        s.connect(sockaddr)
        s.send(MSG)
        data = s.recv(1024)
        print 'Received: %s' % data
        s.close()
    except socket.error, diag:
        s.close()
        continue
    break
```


Address Plans

IPv6 Address Plans

- Long subject, but a few pointers ...
- RFC 3531: A flexible method for managing the assignment of bits of an IPv6 address block
- SURFnet: Preparing an IPv6 address plan
  - http://www.surfnet.nl/Documents/rapport_201309_IPv6_numplan_EN.pdf
- http://www.internetsociety.org/deploy360/resources/ipv6-address-planning-guidelines-for-ipv6-address-allocation/
- https://www.usenix.org/sites/default/files/conference/protected-files/delong_lisa12_slides.pdf

Routing Protocols

IPv6 Routing

- Interior Routing (IGP):
  - **OSPF** version 3 (RFC 5340) - brand new version of protocol
  - **Integrated IS-IS** (RFC 5308) - 2 new TLVs
  - Other options: **RIPng** (seldom used in most real networks), **EIGRP** (cisco proprietary)
- Exterior Routing (EGP):
  - **BGP-4** with Multi-protocol extensions
    - MP-(UN)REACH-NLRI attributes that support IPv6 prefixes

IPv6 Multicast Routing

- PIM (usually PIM-SM: PIM Sparse Mode)
- BGP-4 Multi-protocol Extensions
- No MSDP (Multicast Source Discovery Protocol) exists
  - Static Rendezvous Points shared across domains
  - "Embedded RP" (RFC 3956)
  - Or if possible use Source Specific Multicast (SSM) and obviate the need for source discovery

A few brief router configuration examples

We'll show examples of configuring two of the more popular router platforms: Cisco IOS and Juniper JunOS.

Note: These examples work on most recent versions of IOS and JunOS as of the time of this writing. Occasionally router configuration commands and syntax change between operating system releases, so please confirm against your relevant documentation before trying these.

Cisco IOS: OSPFv3

```
ipv6 unicast-routing

interface Loopback0
 ipv6 address 2001:db8:ab:1::1
 ipv6 ospf 2 area 0

interface FastEthernet0/0
 ipv6 address 2001:db8:ab:2::1
 ipv6 ospf 2 area 0
 ipv6 ospf cost 10

ipv6 router ospf 2
```


Cisco IOS: IS-IS

```
ipv6 unicast-routing

interface Loopback0
 ipv6 address 2001:db8:ab:1::1

interface FastEthernet0/0
 ipv6 address 2001:db8:ab:2::1
 ipv6 router isis

router isis
 net 49.0001.1921.6805.2001.00
 is-type level-2-only
 metric-style wide
 metric 1000
 passive-interface Loopback0
```

Cisco IOS: BGP

```
router bgp 65000
 no synchronization
 neighbor 2001:DB8:5:28::2 remote-as 1111
 no neighbor 2001:DB8:5:28::2 activate
 no auto-summary
 address-family ipv6
  neighbor 2001:DB8:5:28::2 activate
  neighbor 2001:DB8:5:28::2 soft-reconfiguration inbound
  aggregate-address 2001:DB8:5:E160::/61 summary-only
  redistribute connected
  redistribute static
  redistribute isis level-2
  no synchronization
 exit-address-family
```


Cisco IOS: autoconfig

```
(config-if)#ipv6 nd ?
  dad                  Duplicate Address Detection
  managed-config-flag  Hosts should use DHCP for address config
  ns-interval          Set advertised NS retransmission interval
  nud                  Neighbor Unreachability Detection
  other-config-flag    Hosts should use DHCP for non-address config
  prefix               Configure IPv6 Routing Prefix Advertisement
  ra                   Router Advertisement control
  reachable-time       Set advertised reachability time
  router-preference    Set default router preference value
```

```
interface FastEthernet0/0
 ipv6 address 2001:DB8:AB:2::1/64
 ipv6 nd ra interval 300
 ipv6 nd prefix default 3600 1800    #valid,preferred lifetimes
 ipv6 nd ra lifetime 1800
 ipv6 nd other-config-flag           #other config via stateless dhcp
 no ipv6 redirects
```

Cisco IOS: dhcpv6

```
interface FastEthernet0/0
 ipv6 address 2001:DB8:AB:2::1/64
 ipv6 nd ra interval 300
 ipv6 nd prefix default 3600 1800 no-autoconfig
 ipv6 nd ra lifetime 1800
 ipv6 nd managed-config-flag
 ipv6 nd other-config-flag
 ipv6 dhcp relay destination 2001:DB8:CD:3::3
```


JunOS: OSPFv3

```
routing-options {
    router-id 192.168.1.1
}

protocols {
    ospf3 {
        area 0.0.0.0 {
            interface lo0.0 {
                passive;
            }
            interface ge-0/0/0.0;
            interface ge-1/1/3.0;
        }
    }
}
```

JunOS: IS-IS

```
[edit interfaces]
    ge-0/0/0 {
        unit 0 {
            family iso;
            family inet6 {
                address 2001:db8:1800:0501::1/64;
            }
    lo0 {
        unit 0 {
            family iso {
                address 49.0001.1921.6804.2009.00;
            }
            family inet6 {
                address 2001:db8:1800:0500::1/128;
            }
        }
    }
```


JunOS: IS-IS

```
[edit protocols isis]
    isis {
        reference-bandwidth 1000g;
        level 2 {
            wide-metrics-only;
        }
        interface ge-0/0/0.0 {
            level 1 disable;
            level 2 passive;
        }
        interface all {
            level 1 disable;
        }
        interface lo0.0 {
            level 1 disable;
            level 2 passive;
        }
    }
```

JunOS: BGP

```
[edit protocols]

    bgp {
        group ISP1_PEERING {
            type external;
            description "External BGP peering with ISP1";
            family inet6 {
                unicast;
                multicast;
            }
            export OUTBOUND-ISP1;       # filters routes we
                                        # send to ISP1
            peer-as 65001;
            neighbor 2001:db8:cd:2::1;
    }
```


JunOS: autoconfig

```
[edit protocols]

router-advertisement {
    interface ge-0/0/0.5 {
        other-stateful-configuration;
        prefix 2001:db8:1800:505::/64 {
            valid-lifetime 3600;
            preferred-lifetime 1800;
        }
    }
}

# the “other-stateful-configuration” option is to instruct
# autoconfigured clients to obtain non-address parameters
# (eg. dns, ntp, etc settings) via stateless DHCPv6.
```

JunOS: stateful dhcpv6

```
[edit protocols]

router-advertisement {
    interface ge-0/0/0.5 {
        managed-configuration;
        other-stateful-configuration;
        prefix 2001:db8:1800:505::/64 {
            no-autonomous;
        }
    }
}
```


JunOS: stateful dhcpv6

```
[edit forwarding-options dhcp-relay]

server-group {
    servers1 {
        2001:db8:1802:9405::7;
    }
}
group group1 {
    active-server-group servers1;
    interface ge-0/0/0.5;
    interface ge-0/0/0.6;
}
```

Other Network things

Traffic characterization

- Netflow version 9 supports IPv6 flow export, and is implemented by both Cisco and Juniper routers
- IPFIX: IETF's standardized flow export protocol (based on Netflow v9)

Network Management

- SNMP (Simple Network Management Protocol) over IPv6 - transport mappings defined in RFC 3419 for both IPv4 and IPv6
- Some older IPv6-specific SNMP MIBs exist
- Newer Unified SNMP MIBs: support both IPv4 and IPv6
  - RFC 4001, 4292, 4293, 4022, 4113
- Not all network equipment vendors support the newer unified MIBs yet
- NETCONF (RFC 6241) - XML based network configuration protocol also can run over IPv6

References

References

- http://www.internetsociety.org/deploy360/ipv6/
- http://www.getipv6.info/index.php/Main_Page
- http://www.ietf.org/ (hundreds of protocol specs!)
- http://ipv6.com/
- https://www.arin.net/resources/request/ipv4_depletion.html
- https://www.arin.net/knowledge/v4-v6.html
- "Migrating to IPv6: A practical guide .." - M. Blanchet (2006)

Mailing lists

- http://lists.cluenet.de/mailman/listinfo/ipv6-ops
- https://www.ietf.org/mailman/listinfo/v6ops
- https://www.ietf.org/mailman/listinfo/ipv6

LISA '13 — 27th Large Installation System Administration Conference
November 3–8, 2013 • Washington, D.C.

Attention Tutorial Attendees!
Please don't forget to fill out your Tutorial Surveys.

Your feedback is very important to us
and helps us shape the future
of the LISA training program.

Please visit www.usenix.org/lisa13/training/survey
and fill out the appropriate surveys.

Thanks for your help!

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

IPv6

Questions?

twitter @shuque

Shumon Huque
shuque -@- upenn.edu

Please fill out the evaluation form for this course. And say good things if you liked it!