

IPV6

Presented by:
Michael Hare
with slight review by
Dale Carder

heavily influenced by
a presentation from
Michael Sinatra, UCB

[https://webfiles.berkeley.edu/msinatra/public_html/micronet/IPv6-Micronet-notes.pdf]

96 additional bits, no magic

- * IPv4: 32-bit addresses
 - * IPv6: 128-bit addresses
 - * IPv4: 144.92.104.243
 - * IPv6: 2607:f388::a0:c9ff:fea0:110d
 - * MAC: 00:a0:c9:a0:11:0d
- * This IPv6 address was partially derived from the MAC address. The mechanism for doing this is called EUI 64. This is not a requirement for IPv6 addresses, but it does help in certain circumstances.

96 additional bits, no magic

- * IPv6 does not use subnet masks in the same way as IPv4; rather, it uses CIDR prefix length notation to designate the size of the subnet
- * 64-bit subnets are the “standard” user subnet. What does this mean? Well, it means that 64 bits of the 128 bits is used to designate the network and 64 bits is used to designate the host on the network.

Addressing Options

Stateless Autoconfig

Static Configuration

DHCPv6 [not widely supported]

There is also a privacy option, which essentially randomizes the EUI 64 portion of a host's address. This is Vista's default implementation of v6. In this scenario, someone who sees your IPv6 address cannot necessarily derive your MAC address. Your IPv6 address will change from session to session.

Different protocol, similar mistakes

- * In stateless autoconfiguration, the router makes various announcements over Ipv6 ICMP to tell hosts the subnet's /64 prefix. The host then creates a 64-bit representation of its MAC address and appends that onto the 64-bit prefix.
- * IPv6 hosts discover the default router and their neighbors using ICMP multicast
- * This discovery is not necessarily secure. See <http://www.faqs.org/rfcs/rfc3971.html>. This is not the only way to fix this; switchport ACLS [if supported] might also be useful.

Ipv6 address planning

We have our own *provider independent* (PI) address space of 2607:f388::/32.

/64 = 1 subnet [point to point links]

/60 = 16 subnets

/56 = 256 subnets

/52 = 4096 subnets

/48 = 65536 subnets

/44 = 1048576 subnets

/40 = 16777216 subnets

/36 = 268435456 subnets

/32 = 4294967296 subnets

Current Ipv6 address planning

IPv6 assumes more hierarchical addressing plan than IPv4.

This is for prefix aggregation purposes. We hope to avoid address fragmentation in the longer term.

4 bit boundaries are much easier to work with than other boundaries

PROPOSED ALLOCATION STRATEGY

a /56 is the smallest chunk to set aside for a customer

a /48 [medium] for groups that today need a v4 /18

a /44 [large] for groups that today need a v4 /16

a /40 [jumbo] for groups that today need a v4 /14 [any?]

* Real need might be obscured by current use of NAT.

* Don't forget to save the adjacent allocation for growth

Current Ipv6 address planning thoughts

- * Renumbering will always suck even if the protocols says it's going to be easy
- * Don't try aggregating by today's network topology since it will change tomorrow
- * Aggregate by usage [departmental or use]
- * Examples: Network Management, Distributed projects [vending machine net (tm)], etc.

Full disclosure; current assignments

1/8 of our space has been "carved" up.

2607:f388::/36 # Central Campus Services

today: DoIT stuff is being carved out of a single /44

future: a bunch of these /40's might end up going to 'large' departments and projects

2607:f388:1000:/36 # Customer IP Space

today: everything else

2607:f388:2000:/36 # un-allocated

2607:f388:3000:/36 # un-allocated

..

..

2607:f388:e000:/36 # un-allocated

2607:f388:f000:/36 # un-allocated

Ipv6 at UW-Madison now [8/08]

- * We already have some IPv6 test networks in place.
- * We have begun integrating IPv6 into our existing routing topology. Existing IPv4-only routers have become IPv4-IPv6 dual-stack routers.
- * What about OS support?

Enabling V6

- * We can start expecting regular IPv6 traffic very soon after we enable it on a given subnet.
- * Hosts on that subnet that are v6 enabled will request IPv6 DNS records (AAAA records) and will use the resulting address (if available) to contact a given host. If the AAAA record is not available, they will look for an A and contact the host over Ipv4.
- **Remember that the advertisement of a AAAA record for a popular service name implies a production service.**
- * That has security and management and support and performance and other implications.

Good news: It's rather impractical to scan IPv6 address space.

Blatant copy of Sinatra slide

What about other OSes?

- Okay, I already mentioned Vista and Mac OS X. They have v6 turned on by default. XP users can enable v6 with a simple command: ‘netsh ipv6 install’. Linux has very good support, as do all of the *BSD operating systems--most of them have had v6 in the main codebase for several years, so it is likely stable. Recent versions of Solaris (8, 9 and 10), AIX (5L and later), and other proprietary UNIXes have v6 capability built in.
- However, in many systems other than Vista and OS X, v6 must be explicitly enabled--it is not turned on by default.
- Bottom line: most modern OSes can easily support IPv6, and at least two will have it turned on whether we want them to or not.

Another blatant slide copy

What about applications?

- In theory, applications operate in a layered fashion, so they simply talk to the network stack using sockets or some similar API. They therefore, don't need to know anything about the underlying protocol. In practice this is hardly ever the case. Many applications do some sort of sanity checking and expect IP addresses to be in the 32-bit dotted-quad format.
- Other application protocols actually embed the IP address in the packet.
- Most of this has not been too difficult to solve. The major web browsers are all v6-capable, as is apache, many mail servers (SMTP and IMAP services), as well as clients. The best way to see what works and doesn't work is to give it a try.

Routing table bloat and obligatory state worker joke

The IPv4 routing table stands at approximately 262k routes [8/08]. This is a problem because a big routing table means big price tags for routers. Ipv6 does not solve the problem by may put a band-aid on it.

Some protocol proposals that were vying for the role of IPv6 did try to address this, but those other proposals implied a distributed traffic engineering nightmare.

The current way of dealing with this is to give ISPs and enterprises more IPv6 address space than they will ever need. In Ipv4, UW Madison advertises seven prefixes. Hopefully, before when this lifer retires, we will still be advertising a single Ipv6 prefix.

Blatant Slide Copy #3

Transition Mechanisms: IPv6 over IPv4

- **Dual Stack:** Most hosts will operate in a native dual-stack configuration. This allows them to contact either IPv4 or IPv6 hosts. It also requires the host to have an IPv4 *and* an IPv6 address.
- **Automatic Tunnels:** There are a variety of automatic tunnels that allow hosts to have IPv6 connectivity where native IPv6 is not available. We are looking at a variety of options to determine the best set of services to offer. We may offer services to users on their home ISP links.
- **Manual Tunnels:** Manually-configured tunnels work well, but the tunnel must be re-configured if the IP address on either of the hosts changes. For dynamically-addressed hosts, this can become cumbersome.
- All of the above tunnels require valid (and in many cases globally-routable) IPv4 addresses. This means we will need to continue conserving IPv4 address space even as we roll out IPv6, as we will need to use IPv4 space for many years to come.

Blatant Slide Copy #4

Transition Mechanisms: Protocol Translation

- What about the case where you want to have a bunch of hosts using IPv6 addresses *only* with little or no consumption of IPv4 addresses? Can we use IPv6 in this manner to reduce our reliance on our dwindling supply of IPv4 addresses?
- One way is to just have some IPv6 only hosts, but these hosts won't be able to access IPv4-only resources. Other mechanisms include:
 - **Application Proxies:** These allow IPv6-only hosts to be able to access common services like HTTP, POP/IMAP, SMTP, FTP, etc. via individual proxy services for each application. This method has the advantage of simplicity and the disadvantage of not providing for IPv4 connectivity beyond a known--and limited--set of applications.
 - **NAT-PT and TRT:** These provide IPv6 to IPv4 translation at the IP and transport layers, respectively. These methods provide different mechanisms to convert various IPv6 traffic to IPv4 for connecting to IPv4-only devices. Because TRT is transportlayer only, it can only handle TCP and UDP and there are other (annoying) limitations. NAT-PT has all of the annoyances of IPv4 NAT, plus additional complexities. Both NAT-PT and TRT require DNS responses for IPv4-only services to be rewritten on the fly by an Application-Level Gateway (ALG), creating further complexities.

Moving Forward

Providing Content

- DNS
- Web
- Email
- Streaming Media
- Network Management
- Helpdesk
- Other

<http://net.doit.wisc.edu/~dwcarter/ipv6.html>

DoIT needs to learn first and then provide leadership to campus.

RANT

- We may think that IPv6 stinks. We may end up abandoning it well before we run out of v6 address space. However IPv4 is running out and IPv6 is all we have to replace it. It's go time.

Moving Forward

DoIT needs to learn first and then provide leadership to campus. We also need to accept the assistance of campus folks who are willing and able to help the campus make the transition to IPv6.

- “We” should work with individual e-mail and web administrators to help them run dual-stack services.

FIN

**THIS PAGE INTENTIONALLY LEFT AT THE
END**