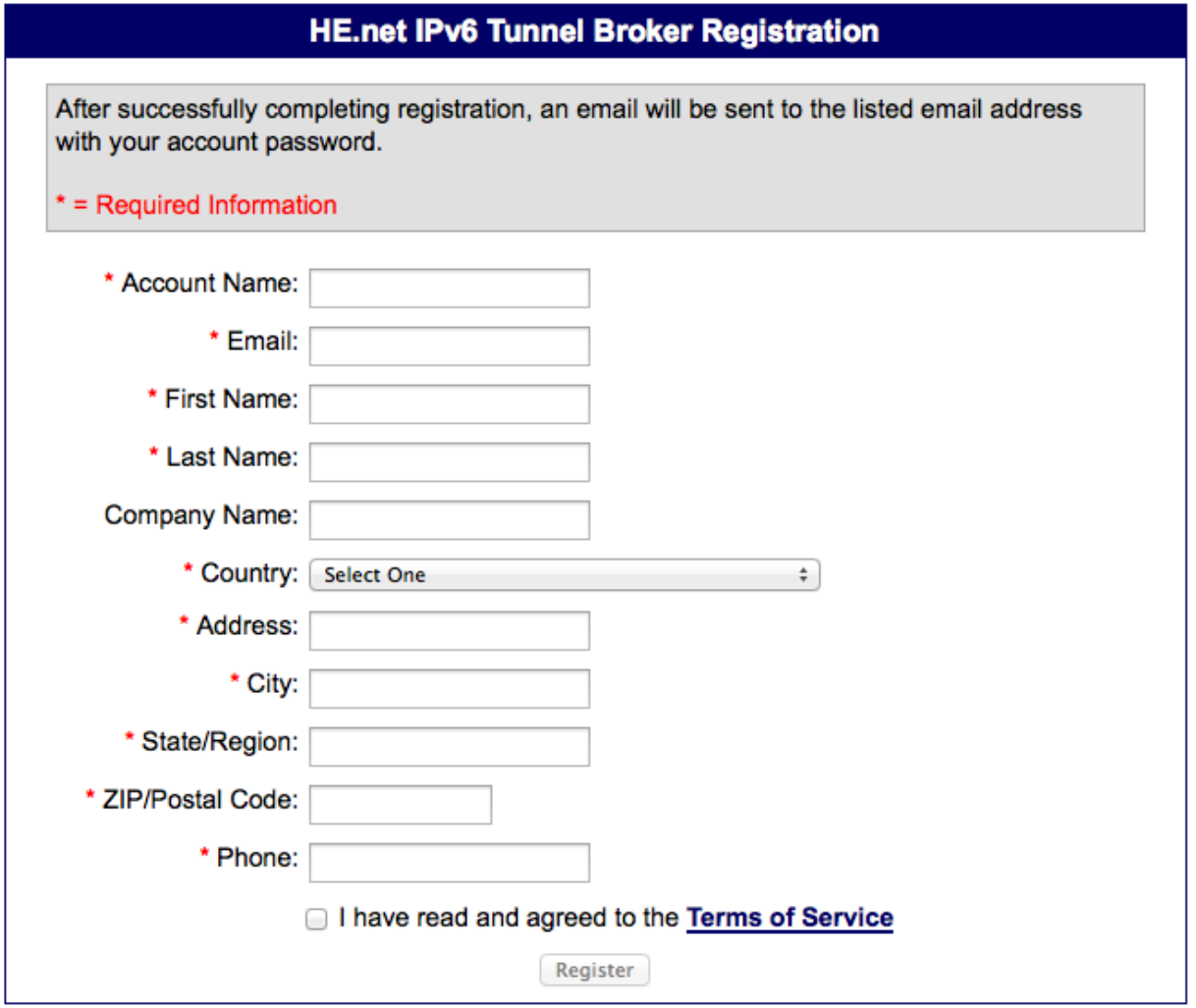
IPv6 Home Networking

Daniel Hagerty
hag@linnaean.org

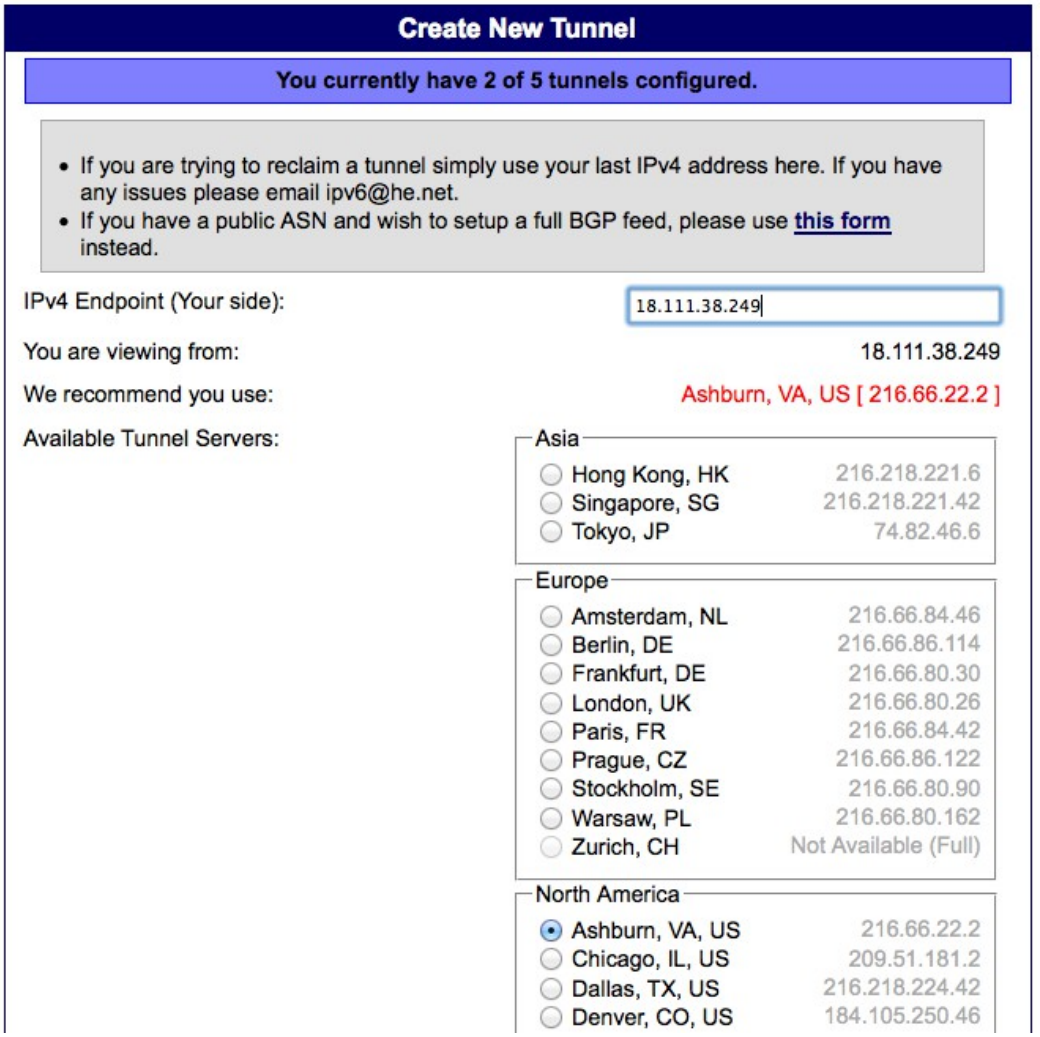
IPv6 Networks I Have Known

- Setup MIT AI Lab in 1999 with MIT SIPB as upstream towards 6bone. Look ma, I can ping6!
- Used 6to4 and Hurricane Electric to run it at home in 2007.
- Provisioned Cambridge Bandwidth Consortium's (AS10255) coreward BGP, colocation subnets, and member tunnels.
- Starting in on my employer. Next step is training.

HE Registration



HE Tunnel Creation



HE Tunnel Creation on OS-X

```
# Create a generic tunnel interface
ifconfig gif0 create

# What are the v4 tunnel endpoints (local, remote)?
ifconfig gif0 tunnel 18.111.38.249 216.66.22.2

# Assign v6 addresses
ifconfig gif0 inet6 2001:470:7:7e2::2 \
    2001:470:7:7e2::1 prefixlen 128

# Tunnel is default route to v6 internet
route -n add -inet6 default 2001:470:7:7e2::1

and from there, it just works, modulo your firewall.
These are their instructions for this OS.  YOSWV
# Firewall example for ipfw
add 02125 permit ip4 from 216.66.22.2 to \
18.111.38.249 proto ipv6 in recv re0
add 03050 permit ip4 from 18.111.38.249 to \
216.66.22.2 proto ipv6 out xmit re0
```

You probably already run it

```
$ ping6 -c 2 -w 10 ff02::1%eth0
PING ff02::1%eth0(ff02::1) 56 data bytes
64 bytes from fe80::2e0:81ff:fe80:b398: icmp_seq=1
ttl=64 time=0.040 ms
64 bytes from fe80::230:48ff:fe99:5d53: icmp_seq=1
ttl=64 time=0.070 ms (DUP!)
[...]
64 bytes from fe80::66b9:e8ff:fecd:b084:
icmp_seq=1 ttl=64 time=0.668 ms (DUP!)
64 bytes from fe80::3e07:54ff:fe62:afe: icmp_seq=1
ttl=64 time=0.670 ms (DUP!)
64 bytes from fe80::426c:8fff:fe50:67ba:
icmp_seq=1 ttl=64 time=0.812 ms (DUP!)
^C
--- ff02::1%eth0 ping statistics ---
1 packets transmitted, 1 received, +22 duplicates,
0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.040/0.279/0.812/0.207 ms
```

Saved from grief by v6

- We recently misconfigured a power control box's v4 net. Apparently unreachable.
- But our switch said it was alive and had a MAC.
- It spoke v6, as revealed by ping6 ff02::1.
- It supported logins via v6.
- Login over v6, fix typo'd address, move on.
- General reset procedure would not have been fun, given the critical hosts on the power box.

What's Changed Since 2010?

- No more google whitelist.
- Netflix, yahoo, facebook, akamai, wikipedia, etc
- Much more traffic at home
- v6 service available from work's last two ISPs
- Comcast reports 50% of its network is v6 capable, and is deploying to customers
- TimeWarner is deploying to customers
- T-Mobile has 100% coverage
- Verizon over LTE in some areas

Generic v6 algorithm

- Get IPv6 to your border. Native, tunnel.
- Enable it on the border box you control (firewall, frequently).
- But your topology doesn't have to be congruent.
- Number internal interfaces
- Either statically number internal hosts, use RA, or play with DHCPv6.
- Put AAAA records in DNS.
- Mail is a good first app, because it's a naturally robust design.

Why is Comcast eager for v6?

- RFC1918 provides 18 million addresses
- Comcast has 20 million video customers with an average of 2.5 set-top boxes per customer which need 2 IP addresses per box == 100 million addresses
- This doesn't account for the network that connects these boxes, or their VOIP offering, or their internet service.

  – Alain Durand at NANOG37
- Amazon, Google, and more have the same problem.

Programming

- Same old low level socket routines. Use PF_INET6, and struct sockaddr_in6.
- New, multi-protocol, thread-safe host/address resolution routines:

  gethostbyname → getaddrinfo

  gethostbyaddr → getnameinfo
- Your favorite languages probably have bindings.
- Where you used one IPv4 socket, you may need a socket per address family now.

getaddrinfo output

```
$ getaddrinfo --stream --service http \
www.google.com
Resolved host 'www.google.com', service '80'

socket(AF_INET , SOCK_STREAM, IPPROTO_TCP) + \
'173.194.75.147:80'
socket(AF_INET , SOCK_STREAM, IPPROTO_TCP) + \
'173.194.75.103:80'
socket(AF_INET , SOCK_STREAM, IPPROTO_TCP) + \
'173.194.75.99:80'
socket(AF_INET , SOCK_STREAM, IPPROTO_TCP) + \
'173.194.75.104:80'
socket(AF_INET , SOCK_STREAM, IPPROTO_TCP) + \
'173.194.75.106:80'
socket(AF_INET , SOCK_STREAM, IPPROTO_TCP) + \
'173.194.75.105:80'
socket(AF_INET6, SOCK_STREAM, IPPROTO_TCP) + \
'[2607:f8b0:400c:c01::6a]:80'

### Output has been wrapped
```

Perl Server Example

```
use IO::Socket;
use IO::Socket::IP;

my @listener_common = (
    Listen => TRUE,
    LocalPort => $port_number,
    ReuseAddr => TRUE,
    );

$listener6 = IO::Socket::IP->new(
   @listener_common,
   Domain => PF_INET6,
   V6Only => TRUE,
   ) || die "socket(PF_INET6): $!";
$listener4 = IO::Socket::IP->new(
    @listener_common,
    Domain => PF_INET,
    ) || die "socket(PF_INET): $!";

# Select loop here
```

IPv6 Ups and Downs: Ups

- It's a network protocol. When it does its job, you don't think about it.
- My wife uses it and doesn't notice.
- Can directly address house “internal” machines from IPv6 networks.
- >50% of incoming email received over v6.
- google, facebook, yahoo, wikipedia, netflix, comcast all running production IPv6.
- My house has exceeded 50% v6 traffic on some days. Youtube + Netflix.

What's new for the v4 clued

- IPv6 is mostly IPv4 with bigger addresses, however:
  - Link local addresses
  - Extensive use of multicast
  - Link scoping to help target the above
  - Stateless address auto configuration (SLAAC)
  - Router advertisements
  - Multiple addresses per interface is typical
- This isn't a complete list, but are the differences I see all the time.

Bigger Addresses

- 128 bits long, 4 times bigger than IPv4
- Represented in hex, not decimal
- Verbosely represented as 2001:0db8:b009:0000:0000:0000:0000:006a
- Some tricks to make them smaller, but the real world still gives you addresses like 2001:470:8917:8:216:cbff:feb7:ae2b

Address Compression

- You can leave off leading zeros of digit groups: “fd00::0123” and “fd00::123” are equivalent.
- You can compress a run of zeros with “::” ONCE, and the run has to be 16 bit aligned. For example, the fd00:: example above. If you have “2001:db8:0:0:0:1:0:0”, “2001:db::1:0:0” is valid, “2001:db8:0:0:0:1::” is legal, but “2001:db8::1::” is not.
- You can use IPv4 notation for the last 32 bits of an address, e.g. 2001:db8::192.0.2.255 is legal. Same address as 2001:db8::c000:2ff.

Prefixes You'll Probably See

v6 Prefix	v4 Approx Equiv	Notes
::	0.0.0.0	Unspecified/IN_ADDR_ANY
::1	127.0.0.1	Loopback
::ffff:0.0.0.0/96		v4 mapped onto v6 sockets
2001:db8::/32	192.0.2.0/24	Documentation Prefix
fc00::/7	10/8, 172.16/12, 192.168/16	Local Unicast (ULA)
fe80::/10	169.254.0.0/16	Link Local
ff00::/8	224.0.0.0/4	Multicast

Address Allocation

- Warm and breathing? You can get a /48 (2.5 ipv4 internets; 65536 64 bit subnets)
- Residential policy allows /56
- Preference for nibble-aligned delegations for operation ease. /36, /40, /44, /48, etc.
- Traditional v4-like PA/PI assignments available.
- Roughly 281 trillion /48s available
- Expect much assesment when 2000::/3 is about gone (35 trillion /48s, 2000 per person at 17e9 people). I'm guessing it will take a bit.

Routable Unicast Space

Prefix	Usage
2000::/3	Global unicast
2001:0::/32	Teredo
2001:db8::/32	Documentation Prefix
2002::/16	6to4
3ffe::/16	6bone (deprecated)

- 4000::/3 through c000::/3 are reserved, as are several other smaller holes. We have 5 tries at address allocation before we need to do IP over again.

Link Local Addresses (fe80::/10)

- Like IPv4's 169.254.0.0/16 prefix, but used extensively.
- Every single IPv6 interface has one as part of configuration.
- Link scoped, meaning the address is relative to an interface. fe80::1 on one link might be a different host than fe80::1 on another link.
- Routing protocols often use them.

Multicast (ff00::/8)

- IPv6 does away with broadcast entirely.
- ff02::1 is the multicast equivalent of an IPv4 broadcast.
- Like link local addresses, they require link scoping.
- Propagation scoping is encoded in the 4th octet: e.g. the “2” in ff02:: addressed packets confines them to the link they were sent on (like 224.0.0.0/24 in IPv4).

Link Scoping

- You need to specify an interface for link local and multicast addresses.
- Append “%” and an interface name to the address.
- For example, “ping6 ff02::1%eth0” should get ping responses from everything in eth0's broadcast domain.
- Interface names are OS specific. Windows uses integers.

Autoconfiguration

- All hosts can use link local addresses to communicate across a single subnet with no central planning.
- The main ingredient for inventing unique addresses is the EUI-64, a 64 bit hardware identifier. Firewire uses it natively.
- Ethernet MACs can be promoted to EUI-64 by inserting “ff:fe” into the middle, after the OUI.
- Only works with /64 prefixes.

Router Advertisements (RA)

- Routers tell clients the prefixes in use and clients build themselves addresses with them.
- Clients route to the routers they see, even if it's a broken laptop somebody has been experimenting with.
- This isn't anything like DHCP.

DHCPv6

- There is one.
- It's late to the party.
- Support is spotty.
- MS Vista or higher, OS-X >= 10.7.
- Haven't played with it yet; SLAAC works, and half of my machines don't do it without effort.
- DHCP and IPv6 evolved concurrently, and didn't cross pollinate until pretty late.

DHCP Prefix Delegation

- DHCP option for delegation of address space for routed subnets downstream of the requesting DHCP client.
- It supposedly can work.
- Microsoft supports it when using ICS.
- ISC dhcpv6 supports it.
- Promising, but I'd expect some blood.

DHCPv6 vs RA

- Religion. Different constituents want different things.
- Purists hate the DHCP model and implementation. Pragmatists want the purists to suggest something that meets their needs, as RA doesn't do it yet.
- For example, DNS servers (!!) were only recently added to RA. DHCP has many, many standard options.

Unique Local Addresses (ULA)

- More or less RFC1918 for IPv6.
- 40 random bits in the prefix
- Much less likely to collide than RFC1918 addresses when used for private interconnect, mergers, etc.
- There's a “registry” where a further hint that a prefix is in use can be documented.

IPv6 info in DNS

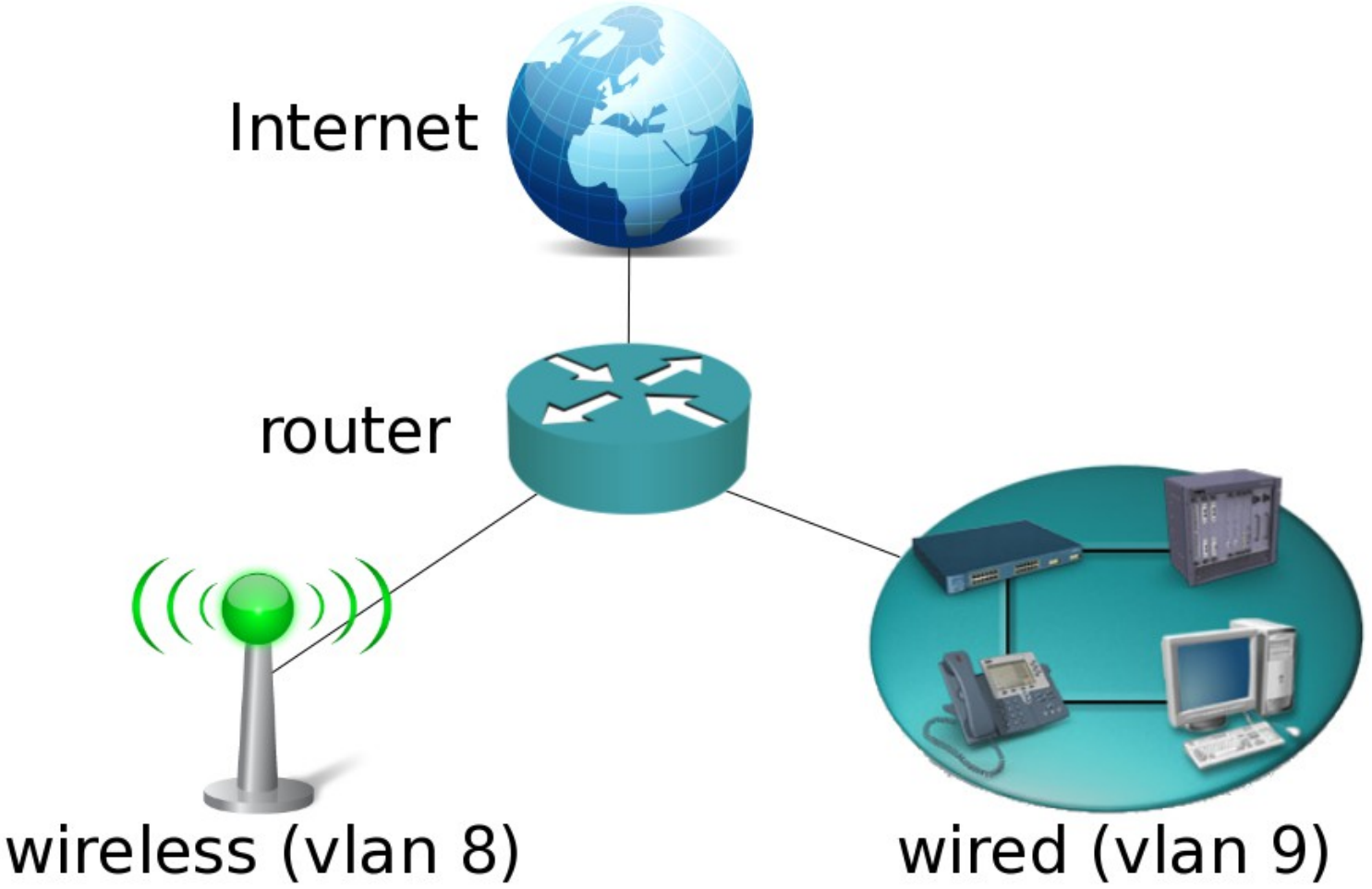
- Works roughly the same as it did in v4: there's an address record for forward, and a PTR record for reverses.
- Reverses are split by each hex digit. Use host!
- Forward:

```
perdition IN AAAA 2001:470:8917:1::1
```

- Reverse:

```
$ORIGIN 7.1.9.8.0.7.4.0.1.0.0.2.ip6.arpa.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0 IN PTR
   perdition.linnaean.org.
```

Home Network



Home Network

- Router advertisements required configuring addresses on the subnets, and starting a daemon with no options. There was a surprise, in that all v6 hosts IMMEDIATELY used it.
- Initially, I didn't expose my v6 DNS to the public, but eventually I exposed individual services over the course of a couple of months. Nothing terribly exciting.

IPv6 Connection Methods

- Native; mostly stop here if you can get it. Cogent may be an exception.
- Static tunnel providers like Hurricane or Sixxs
- 6to4 (deprecated)
- Teredo; single host only.

Static Tunnel Providers

- Hurricane Electric's tunnelbroker.net
  - Simple IP Protocol 41 tunnels.
  - They hand out /48s with a click.
  - Will speak BGP, in ways that are real multi-homing.
- Sixxs
  - Requires tunneling software. Very, very widely ported.
  - Can traverse most NATs.
  - Some POPs only offer /64 prefixes (1 subnet).
  - More bureaucratic.

Teredo

- Works through typical NATs that will pass UDP traffic with the help of a “teredo server”.
- Only provides a single /128 address
- Can directly reach other teredo users over v4.

Happy Eyeballs (RFC6555)

- User friendliness algorithm, typically in browsers (Chrome, FF)
- Tries V4,V6 connections in parallel, and uses whatever finishes first.
- Good: user requests satisfied ASAP.
- Bad: not determinstic when you need to debug.

NAT

- Zealot's heads explode at the thought.
- You can get it if you need it.
- There are clever applications of multi-addressing with both globally unique and ULA addresses that reduce need of it.
- Applications like ghetto-multihoming and reducing renumbering pain from ISP changes are still lacking in a NAT free world. Multi-addressing is not a panacea.

RFC3484

- Introduces controls for controlling source and destination address selection, both v4 and v6
- Hackable in linux as /etc/gai.conf; FreeBSD with ip6addrctl
- Implemented on Windows, but the tables are not mutable
- Address selection is very important when there are many choices

Privacy Addresses (RFC4941)

- Generates random, throw away 64 bit interface identifiers in addition to “the” interface address.
- Can't hide your subnet, obviously.
- Default on: >= Windows Vista, >= OS-X 10.7
- Available on: XP, OS-X < 10.7, Linux, FreeBSD

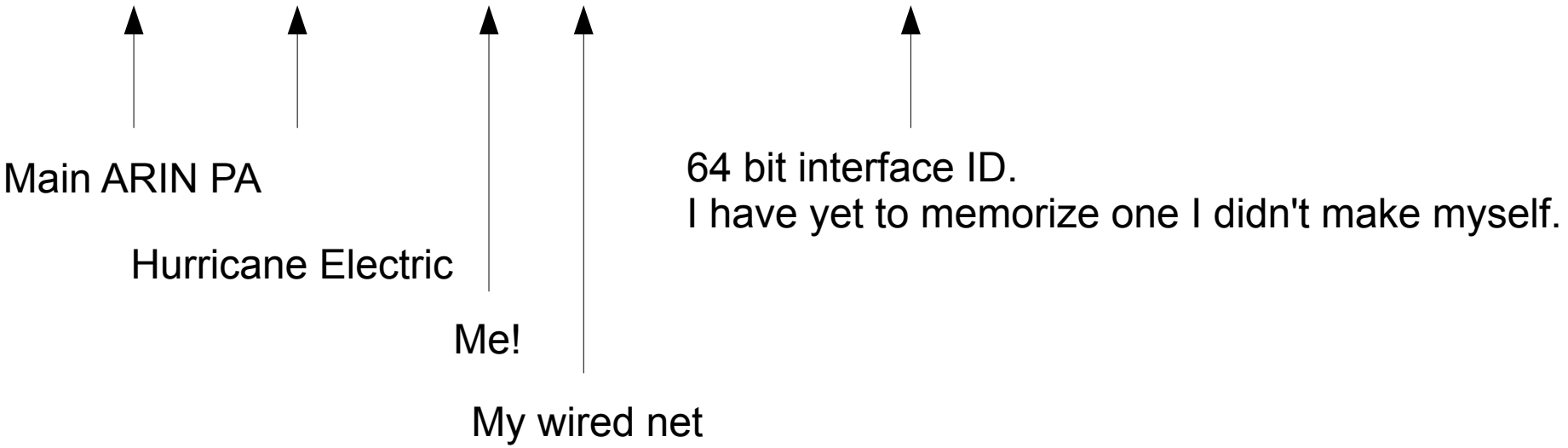
Secure Neighbor Discovery (RFC3971)

- NDP (and v4 arp!) are easy to attack on shared networks, even if switched. I've done so against arp, in anger even (comcast dhcp fail).
- 64 bits is enough to do public-key crypto.
- Protects against the comcast fail I once had, as the subnet router would have ceased listening to the other guy.
- Not exactly common yet, if ever.

De-facto address structures

- The address family is flat, like v4 w/ CIDR.
- But in practice, there's structure that makes things slightly easier to remember.

2001:470:8917:9:217:f2ff:fe0a:c4e2



De Jure address structure, Teredo

```
$ teredo-decode 2001:0:53aa:64c:2046:0674:a7e0:4a3e
addr = 2001:0:53aa:64c:2046:0674:a7e0:4a3e
server = 83.170.6.76
client = 88.31.181.193
port = 63883
flags =
flags_random = 8262 (0x2046)

2001:0 == Teredo
53aa:064c == Teredo Server, 83.170.6.76
2046 == 12 bit nonce + flags; no flags here
0674 == Obfuscated port number (port ^ 0xffff)
a7e0:4a3e == Obfuscated client IPv4 address
             (address ^ 0xffffffff)
```

Neighbor Discovery

- In v6, the arp equivalent is an ICMP protocol, making clever use of link local and multicast addresses in a way that would be circular on v4.
- You don't really notice.
- ...Unless you're writing firewall rules and forget to allow it.