YAHOO!

IPv6

CS 168, Fall 2014

Stephen Strowes, `sds@yahoo-inc.com`

`http://inst.eecs.berkeley.edu/~cs168/`

2014-11-19

Who am I?

I'm helping push on IPv6 deployment at Yahoo

This means I get to pay attention to how healthy our IPv6 traffic is...

... and how healthy our IPv4 address space is...

... and try to guide internal standards, managers, engineers, etc, in the correct direction

Outline

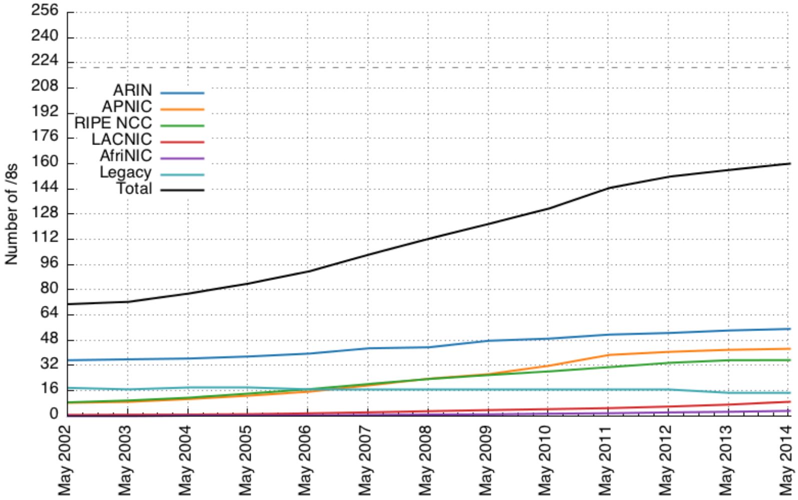
Outline

Why do we care about IPv6?

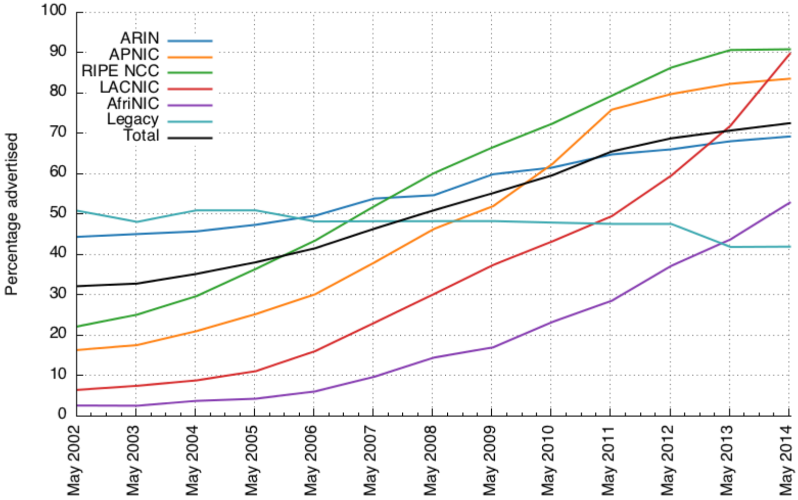
Why do we care about IPv6? Registries are out of space

- We (as in, globally) are *effectively* out of IPv4
  - APNIC ran out on the 15th of April, 2011
  - RIPE ran out on the 14th of September, 2012
  - ARIN ran out on the 23rd of April, 2014
  - LACNIC ran out on the 10th of June, 2014
- IPv6 was standardised in 1998
- IPv6 is now, at last, carrying significant volumes of traffic

Why do we care about IPv6? Most of IPv4 space is already routable



Why do we care about IPv6? Most of IPv4 space is already routable

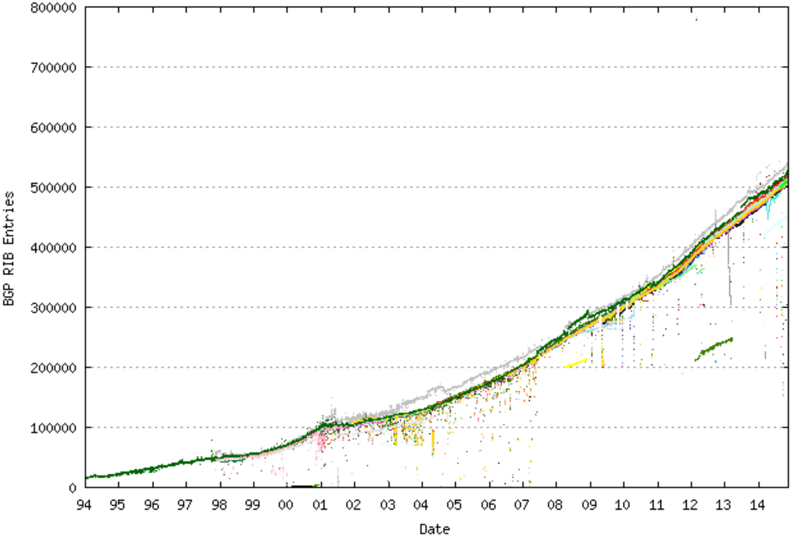


YAHOO!

Why do we care about IPv6? IPv4 starts to get expensive

- “Microsoft pays Nortel $7.5 million for 666k IPv4 addresses” (2011)
- “A first look at IPv4 transfer markets”, CoNEXT 2013 `http://dl.acm.org/citation.cfm?id=2535416`
- “Microsoft Azure’s use of non-US IPv4 address space in US regions”

Why do we care about IPv6? IPv4 BGP growth



Why do we care about IPv6? IPv4 BGP growth

"Internet Touches Half Million Routes"

`http://research.dyn.com/2014/08/internet-512k-global-routes/`

Why do we care about IPv6?

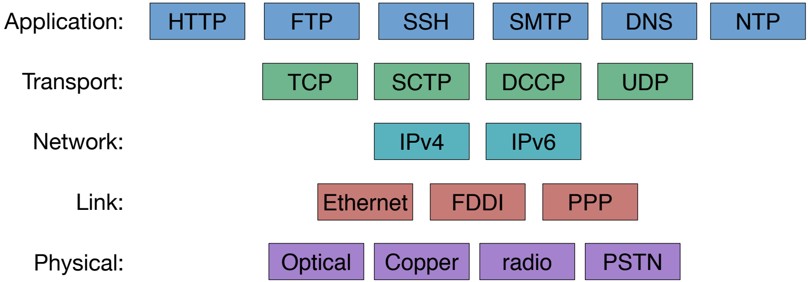
We could keep dealing with this, or...

Outline

Context: The Hourglass

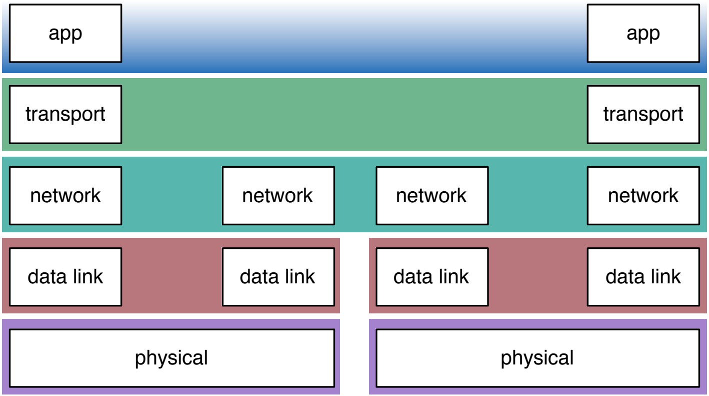
Application:	HTTP	FTP	SSH	SMTP	DNS	NTP
Transport:	TCP	SCTP	DCCP	UDP		
Network:	IPv4					
Link:	Ethernet	FDDI	PPP			
Physical:	Optical	Copper	radio	PSTN		

Context: The Hourglass



Context: The Hourglass

In other words, in this lecture we're paying attention to the network layer, and end-to-end addressability and connectivity *across networks*.



Context for this lecture

I'll touch on:

- IPv6 addressing and address families
- IPv6 address autoconfiguration
- (briefly) naming and DNS modifications
- steps toward transition
- growth data

Context: IPv6

- Question: What does IPv6 offer that IPv4 does not?

- Primarily: a substantially larger address space
- Addresses are 128-bits wide rather than 32-bits
- $3.4 * 10^{38}$, or 340 billion billion billion billion

Context: What's different?

Fundamentally,

- addresses are larger
- packet headers are laid out differently
- address management and configuration are completely different
- some DNS behaviour changes
- some sockets code changes
- everybody now has a hard time parsing IP addresses

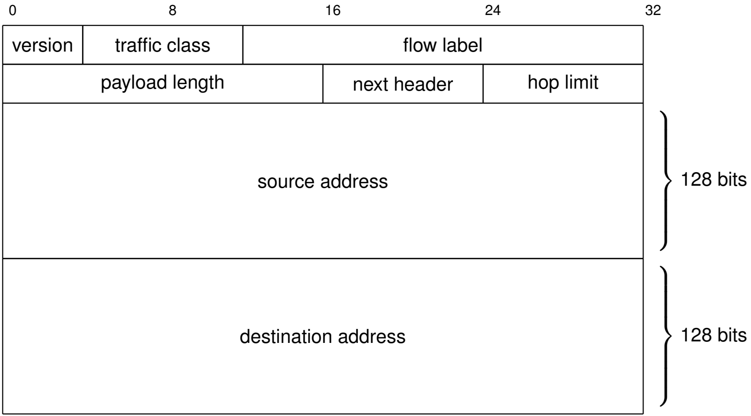
Context: What's *not* different?

IPv6 is a pretty conservative progression from IPv4.

- Routing protocols have to carry IPv6 addresses, but otherwise operate in the same fashion
- Forwarding hardware has to handle IPv6 headers, but longest-prefix/shortest-path routing is basically the same
- The forwarding plane is actually slightly simpler

The IPv6 Packet Header

The IPv6 Header



- The IPv6 header is 40 bytes long (v4 headers are 20 bytes)
- header layout is intended to be simpler

The IPv6 Header

- version: always 6
- traffic class: same as DSCP/ECN fields in IPv4
- flow label: a new field, to help the network layer identify packets belonging to the same flow
- payload length: the length (bytes) of everything *after* this header
- next header: indicates the type of the *next* header or the transport header. Same codepoints as for IPv4 'protocol' field.
- hop limit: TTL
- IPv6 source
- IPv6 destination

The IPv6 Header

- operation is intended to be simpler:
  - no *in-network* fragmentation
  - no checksums
  - optional state carried in *extension headers*

Extension Headers

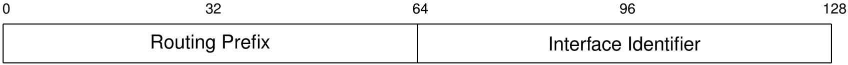
- Extension headers notionally replace IP options
- Each extension header indicates the type of the *following* header, so they can be chained
- The final ‘next header’ either indicates there is no ‘next’, or escapes into an upper-layer header (e.g., TCP)

Outline

IPv6 addressing

Basic Address Structure

IPv6 addresses are split into two primary parts:



- 64 bits is dedicated to an addressable interface (equivalent to the host, if it only has one interface)
- The network prefix allocated to a network by a registry can be up to 64-bits long
- An allocation of a /64 (i.e. a 64-bit network prefix) allows *one* subnet (it cannot be subdivided)
- A /63 allows two subnets; a /62 offers four, etc. /48s are common for older allocations (RFC 3177, obsolete).
- Longest-prefix matching operates as in IPv4.

Briefly: Address Representation

IPv6 addresses represented as eight 16-bit blocks (4 hex chars) separated by colons:

- `2001:4998:000c:0a06:0000:0000:0002:4011`

But we can condense the representation by removing leading zeros in each block:

- `2001:4998:c:a06:0:0:2:4011`

And further by reducing consecutive blocks of zeros to a “`::`”:

- `2001:4998:c:a06::2:4011`

Address Families

The address space is carved, like v4, into certain categories [1]:

- host-local : localhost; **`::1`** is equivalent to **`127.0.0.1`**
- link-local : not routed: **`fe80::/10`** is equivalent to **`169.254.0.0/16`**
- site-local : not routed *globally*: **`fc00::/7`** is equivalent to **`192.168.0.0/16`** or **`10.0.0.0/8`**
- global unicast : **`2000::/3`** is basically any v4 address not reserved in some other way
- multicast : **`ff00::/8`** is equivalent to **`224.0.0.0/4`**

[1] http://www.ripe.net/lir-services/new-lir/ipv6_reference_card.pdf

The EUI-64 Interface Identifier

What is the interface identifier?

- IEEE 64-bit Extended Unique Identifier (EUI-64)[2]
- There are various techniques to derive a 64-bit value, but oftentimes we care about deriving that value from a 48-bit MAC address.

Company ID			Vendor ID		
c8	e0	eb	18	75	87

↓

c8	e0	eb	**ff**	**fe**	18	75	87

↓

ca	e0	eb	ff	fe	18	75	87

[2] http://tools.ietf.org/html/rfc2373

Outline

At this point, we have addresses and packet headers. How do hosts configure themselves to be addressable on the network?

Addresses can be configured statically (in some environments; tools such as chef can make this manageable), or dynamically. I'll cover the mechanics of dynamic assignment here.

Neighbour Discovery

Neighbour Discovery

- The Neighbour Discovery Protocol[3] specifies a set of ICMPv6 message types that allow hosts to discover other hosts or routing hardware on the network
  - neighbour solicitation
  - neighbour advertisement
  - router solicitation
  - router advertisement
  - redirect
- In short, a host can *solicit* neighbour (host) state to determine the layer-2 address of a host *or* to check whether an address is in use
- or it can solicit router state to learn more about the network configuration
- In both cases, the solicit message is sent to a well-known multicast address

[3] http://tools.ietf.org/html/rfc4861

SLAAC:
StateLess Address Auto Configuration

IPv6 Dynamic Address Assignment

We have the two halves of the IPv6 address: the network component and the host component. Those are derived in different ways.

Network (top 64 bits):

- Router Advertisements (RAs)

Interface Identifier (bottom 64 bits):

- Stateless, automatic: we have already seen the EUI-64
- Stateful, automatic: DHCPv6 (which I won't cover here)

SLAAC: overview

SLAAC is:

- ... intended to make network configuration easy without manual configuration *or even a DHCP server*
- ... an algorithm for hosts to automatically configure their network interfaces (set up addresses, learn routes) without intervention

SLAAC: overview

- When a host goes live or an interface comes up, the system wants to know more about its environment
- It *can* configure link-local addresses for its interfaces: it uses the interface identifier, the EUI-64
- It uses this to ask (solicit) router advertisements sooner than the next periodic announcements; ask the network for information

SLAAC: overview

The algorithm (assuming one interface):

1. Generate potential link-local address
2. Ask the network (multicast[4]) if that address is in use: *neighbour solicitation*
3. Assuming no responses, assign to interface

[4] `https://tools.ietf.org/html/rfc2373`

SLAAC: overview; Router Solicitation

Then,

- Once the host has a unique *link-local* address, it can send packets to anything else sharing that link substrate
- ... but the host doesn't yet know any routers, or public routes
- ... bootstrap: routers listen to a well-known multicast address

4. host asks the network (multicast) for router information: *router solicitation*
5. responses from the routers are sent directly (unicast) to the host that sent the router solicitation
6. the responses *may* indicate that the host should do more (e.g., use DHCP to get DNS information)

Router Advertisement

Without solicitation, router advertisements are generated intermittently by routing hardware.

Router Advertisements:

- nodes that forward traffic periodically advertise themselves to the network
- periodicity and expiry of the advertisement are configurable

Router Advertisement (RA), among other things, tells a host where to derive its network state with two flags: M(anaged) and O(ther info):

- M: “Managed Address Configuration”, which means: use DHCPv6 to find your host address (and ignore option O)
- O: Other information is available via DHCPv6, such as DNS configuration

Address Configuration: SLAAC

Question:

What problem arises from totally decentralised address configuration?

Address Configuration: SLAAC

Privacy concerns that arise from using an EUI-64:

- Privacy: SLAAC interface identifiers don't change over time, so a host can be identified across networks
- Security: embedding a MAC address into an IPv6 address will carry that vendor's ID(s)[5], a possible threat vector

[5] http://standards.ieee.org/develop/regauth/oui/public.html

Address Configuration: SLAAC Privacy Addresses

Privacy extensions for SLAAC[6]

- temporary addresses for inititating outgoing sessions
- generate one temporary address per prefix
- when they expire, they are not used for new sessions, but can continue to be used for existing sessions
- the addresses should appear random, such that they are difficult to predict
- lifetime is configurable; this OSX machine sets an 86400s timer (1 day)

[6] https://tools.ietf.org/html/rfc4941

Address Configuration: SLAAC Privacy Addresses

The algorithm:

- Assume: a stored 64-bit input value from previous iterations, or a pseudorandomly generated value

1. take that input value and append it to the EUI-64
2. compute the MD5 message digest of that value
3. set bit 6 to zero
4. compare the leftmost 64-bits against a list of reserved interface identifiers and those already assigned to an address on the local device. If the value is unacceptable, re-run using the rightmost 64 bits of the result instead of the historic input value in step 1
5. use the leftmost 64-bits as the randomised interface identifier
6. store the rightmost 64-bits as the history value to be used in the next iteration of the algorithm

DNS

DNS Additions

- The addition of an “AAAA” record to DNS to carry IPv6 bindings that hosts can query is sufficient
- Modification of DNS sort list semantics[7]

[7] `http://tools.ietf.org/html/rfc3484`

YAHOO!

Outline

Question: why has the transition taken so long?

Problem: How do you (we, us) transition from IPv4 to IPv6

- IPv4 and IPv6 are not compatible:
  - different packet formats
  - different addressing schemes
- as the Internet has grown bigger and accumulated more IPv4-only services, transition has proven ... tricky

Problem: How do you (we, us) transition from IPv4 to IPv6

- IPv4 has/had the momentum
- ... which led to CIDR
- ... and encouraged RFC1918 space and NAT
  - the details of IPv4 NAT are not worth discussion here, but in essence: your ISP hands you only one IPv4 address, and you share that across multiple devices in your household. The NAT handles all the translation between internal ("private") and external ("public") space

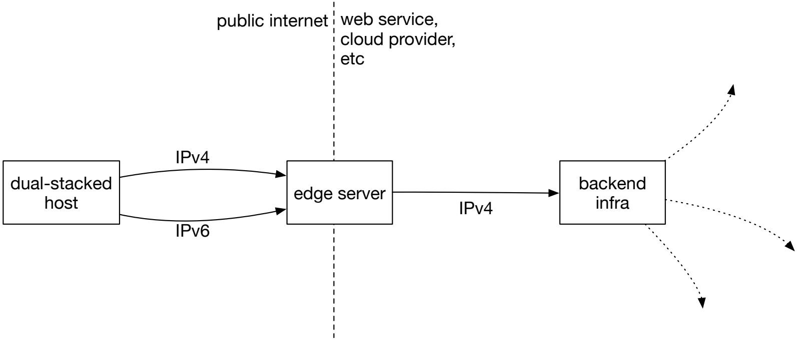
Transition tech: outline

- Tunneling
- dual-stacked services, and happy eyeballs
- DNS64 and NAT64[8]
- 464XLAT
- DNS behaviour

[8] https://tools.ietf.org/html/rfc6146

Dual-Stack Services: Common Deployment

It's common for web services to play conservatively: dual-stack your edge services (e.g., load balancers), leaving some legacy infrastructure for later:



Dual-Stack Services: Common Deployment

Aim is to reduce the pain:

- You can dual-stack the edge hosts, and carry state in, say, HTTP headers indicating the user's IP address (common over v4 anyway)
- You can dual-stack the backend opportunistically, over a longer period of time
- You use DNS to enable/disable the v6 side last (if there is no AAAA record in DNS, no real users will connect to the IPv6 infrastructure

Happy Eyeballs

- The introduction of IPv6 carried with it an obligation that applications attempt to use IPv6 before falling back to IPv4.
- What happens though if you try to connect to a host which doesn't exist?[9]
- But the presence of IPv6 modifies the behaviour of DNS responses and response preference[10]

[9] https://tools.ietf.org/html/rfc5461
[10] https://tools.ietf.org/html/rfc3484

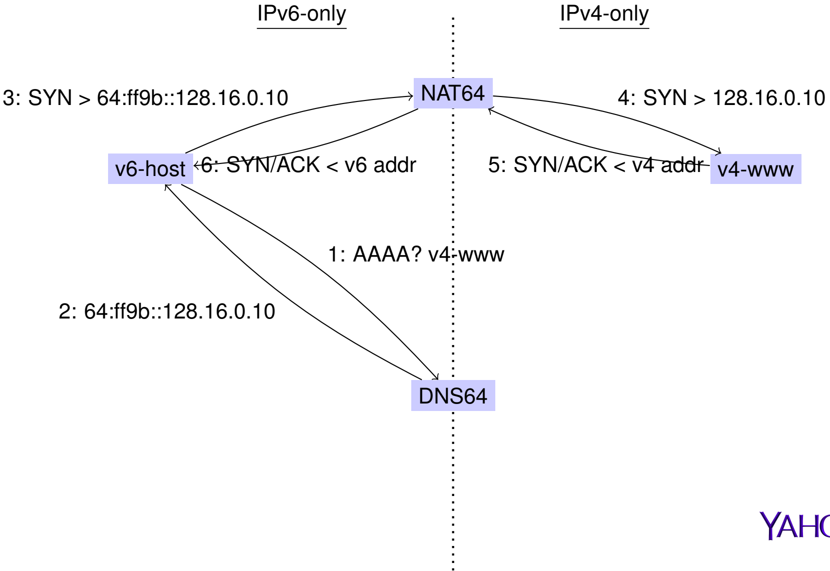
Happy Eyeballs

- Happy Eyeballs[11] was the proposed solution
  - the eyeballs in question are yours, or mine, or whoever is sitting in front of their browser getting mad that things are unresponsive
- Modifies application behaviour

[11] https://tools.ietf.org/html/rfc5461

DNS64 & NAT64

DNS64 & NAT64


IPv6-only
IPv4-only
NAT64
3: SYN > 64:ff9b::128.16.0.10
4: SYN > 128.16.0.10
v6-host
6: SYN/ACK < v6 addr
5: SYN/ACK < v4 addr
v4-www
1: AAAA? v4-www
2: 64:ff9b::128.16.0.10
DNS64

464XLAT

464XLAT

Problem: IPv6-only to the host, but an IPv4-only app trying to access an IPv4-only service

- Some *applications* do not understand IPv6, so having an IPv6 address doesn't help
- 464XLAT[12] solves this problem
- In essence, DNS64 + NAT64 + a shim layer on the host itself to offer IPv4 addresses to apps

[12] `https://tools.ietf.org/html/rfc6877`

Outline

Where are we now?

Where are we now?

- Places we see deployment
- Who’s pushing forward?

Where are we now? IPv6 readiness according to Yahoo data

What we measure:

- we measure requests at our CDN, and store broad aggregates
- (per day, by-*ISP* or by-country, proportion of requests, and the significance of the measurement)
- we contribute our measurements along with Google, Facebook, and Akamai, to the Internet Society: **`http://www.worldipv6launch.org/measurements/`**

YAHOO!

Where are we now? ISP activity

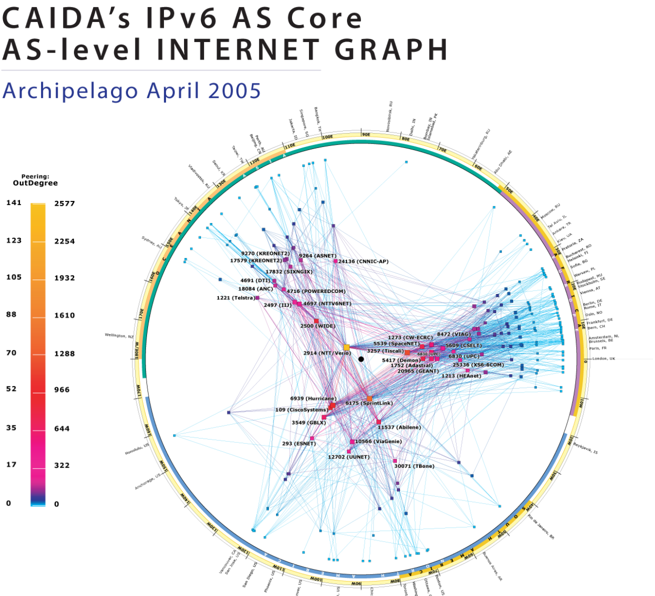
- Comcast
- T-Mobile US
- Verizon

Where are we now? Other stats

Other measurements:

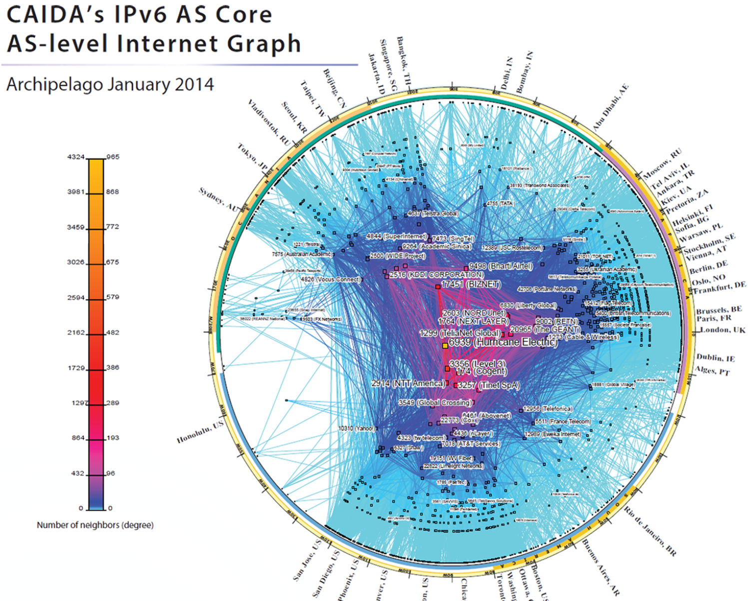
- **http://www.stateoftheinternet.com/trends-visualizations-ipv6-adoption-ipv4-exhaustion-global-html**
- **http://www.google.com/intl/en/ipv6/statistics.html**
- Lars Eggert has an ongoing measurement which looks at IPv6 readiness of top sites according to DNS: **https://eggert.org/meter/ipv6.html**

Where are we now? BGP Connectivity



YAHOO!

Where are we now? BGP Connectivity



Wrap-up

Broadly, I've covered:

- IPv4 context
- IPv6 architecture: packet headers, host addressing, configuration
- Some transition technologies
- Context for current growth

Questions?