The Road To IPv6
Bumpy

Paul Saab
Infrastructure
2014/03/20
Agenda

1. Motivation
2. Running out of RFC1918
3. IPv6 @ face:bo0c
4. The good stuff
5. Where are we now?
Motivation
Motivation

- We’re out of RFC1918
  - More on this later.
- Why maintain two stacks in your entire network?
  - Much easier to handle two stacks at the edge and one internally.
- It needs to be done someday, why not now while we’re motivated?
- Stop engineers from continually writing IPv4 only code that will need to be fixed later.
- Push the industry to move faster and re-prioritize IPv6.
Running out of RFC1918
16,777,216 addresses isn’t enough
/24 for every rack – Genius!

- Math is easy
- Subnet mask is easy to remember
- Wastes a lot of space
  - 254 usable addresses
  - 80 addresses in a rack
- /25 is what we could re-number into
  - Not enough savings
- Do it if you want to get to IPv6 faster
Solutions

- re-number/re-subnet IPv4
  - Too much code assumed racks are /24
  - Too much code assumed clusters are /n (where n < 24)
  - /25 doesn’t save us much
- IPv6
  - Easier to overlay IPv6 on top of the network than re-subnet
  - Can be done without taking datacenters/clusters offline
  - Most importantly, you can test incrementally and iterate
IPv6 @ face:booc
79,228,162,514,264,337,593,543,950,336 addresses enough?
The Network

- Each rack is a /64
- Each cluster is a /n (where n < 64)
- Layer 3
  - Core should not handle Neighbor Discovery
  - fping6 –g xxx::/64 confined to a rack
  - Just Do It
The Problems - Switches

- Vendors do not QA IPv6 like they do IPv4
- Started seeing multi-second latency to hosts over IPv6
  - Occurred when host eth link goes up and down
  - Suspected Linux
- Turns out vendor batch updated to the hardware table
  - Add and Delete occurred in the same batch
  - Ended up software switching
- Set us back about 6 months
  - Thousands of racks had to be upgraded
The Problems – Switches (cont)

- Hardware ASIC has a separate ECMP table for /65 - /128 routes
  - Total of 127 entries
  - Forced us to use /64 for route all route injection
  - Required us to renumber
- Dual BGP sessions
  - Cluster switches could not support that many BGP sessions
  - Forced to run IPv4 and IPv6 over a single BGP session
- Turning on IPv6 Address Family on BGP sessions to rack switches that did not have IPv6 enabled crashed all of the rack switches. Awesome!
The Problems – Switches (cont)

- Multi-second latency returned again!
  - Issue was between the rack switch and the cluster switch
  - No rebuild was needed, just a one line configuration change

- Uneven traffic across multiple links
  - Issue between the rack switch and cluster switch
  - BGP comes up before Neighbor Discovery
  - Traffic goes only over links where ND happened before BGP
The Problems - PHP

- ip2long is the devil
  - IP addresses are not integers (or strings!)
- Inconsistent API’s to use IPv6 addresses.
  - Some functions expect a URL (must enclose with brackets for IPv6)
  - Some functions expect just an IP (no bracket)
The Problems – Strings

- Java’s InetAddress produces different zero compressed string than glibc, FreeBSD, and MacOS X
  - pick a format and normalize all input
  - regex matching (10000000 different ways to match an IPv6 address)
- "host:port".split(':'), explode(‘:’, “host:port”)
  - everyone assumes you can split on a ':' to extract a host port
  - IPv6 addresses must be enclosed in ‘[]’, adds complexity
- strcmp(ip1, ip2) == 0
  - “2a03:2880::1” != “2a03:2880:0000:0000:0000:0000:0000:0001”
The Problems – Storage

Integers

Strings

- In MySQL use VARBINARY(16)

Store all in binary format
The Problems – < glibc-2.17

- getaddrinfo(ipv6-ip-address) failed with EAI_FAMILY
  - Happens once, and continues until process is restarted
  - Single netlink socket failure inside glibc causes this
  - Not fixed until glibc-2.17
The Problems – Engineers & AF_INET

- GRRRRRRRRRRRRR

- Engineers have been trained to write IPv4 only code
  - Must educate the usage of getaddrinfo(3)
  - Teach engineers about how to use the hints to getaddrinfo(3)
    - AF_UNSPEC
    - AI_ADDRCONFIG | AI_PASSIVE

- New code constantly being written IPv4 only

- Solution
  - Take away IPv4 on development systems in 2014
The Problems – SLAAC vs Static Assignment

- SLAAC
  - Great idea
  - Terrible for datacenter deployment
  - NIC changes, IP address changes

- Static Assignment
  - Avoid encoding IPv4 address in the IPv6 address
  - But it makes mapping back and forth easy!
  - What happens when you stop using IPv4?
  - Take the opportunity to have a clean slate with no dependencies
The Problems – Linux

- Routing table
  - Max size defaults to 4096
  - Runs garbage collection when there are more than 512 entries
  - *ALL* connections are cached in the routing table
    - Default TTL is 30 seconds
    - Lots of churn happens
  - ip -6 route show can take forever or even duplicate output
  - /proc/net/ipv6_route returns ENOMEM with 1000s of connections (netstat)
The Problems – Linux

- non-eth0 addresses unusable on network restart
- options ipv6 disable=1
  - Requires a reboot to enable IPv6
  - blacklist ipv6 allows you to load IPv6 on a running system
- options ipv6 autoconf=0
  - SLAAC is terrible for datacenter deployments
  - Do not want multiple addresses on eth0
The Problems – AAAA records

- Can break applications which were not expecting an IPv6 address
- IPv4 hosts can “fallback” to IPv6 if IPv4 fails to connect
  - Get back EAFNOSUPPORT
  - Engineers complain
  - `getaddrinfo(3)` returns a list of addresses that applications walk connecting to until one succeeds
- No need with adequate service discovery
- Turn on selectively
The Problems – Applications

- MySQL 5.6 is required for IPv6 client and server
- Curl
  - Very hostile to the format of the IPv6 address
  - Wants everything bracket enclosed
  - Many IPv6 bugs that only recently were fixed
- Understand operational behavior of app on IPv4
  - Engineers don’t monitor under IPv4
    - All of a sudden they are interested in monitoring when turning on IPv6
  - Busted code is agnostic to IP protocols
The good stuff
It wasn’t all bad
The Good

- We were able to get rid of a lot of technical debt
- IPAddress class
  - Death to strings and integers
- Rollout of traffic
  - Most services were able to slowly roll out IPv6 from 0-100%
  - Instantaneous rollback if needed
  - Problems may not show up at 1%, 5% or 10%, but they do at 100%
- Iterate Iterate Iterate
  - Don’t make IPv6 an all or nothing proposition. You will fail.
The Good – Neteng @ Facebook

- Backbone was upgraded a couple of years ago
- Clusters were converted to Layer 3
- IPv6 native to all cluster and rack switches after World IPv6 Day
- The real heroes
The Good

- APIs to detect if host supported IPv6 and it had *working* IPv6
  - Not all hosts had working IPv6 until recently
- IPv6 became a native component of our service discovery framework
  - All services to be dual stacked
  - ip:port no longer a reasonable way to identify a service
- Thrift already supported IPv6
  - Most of our non-memcache traffic is thrift
  - Initially supported IPv6 with V4MAPPED
  - Separate AF_INET and AF_INET6 sockets today
The Good

- Automation built to handle rack switch upgrades
  - It could never be done
  - Empowered engineers to do their own maintenance
  - We finished it
Where are we now?
Where are we now?

- 100% of our hosts we care about respond on IPv6
  - Hosts that are not IPv6 ready are going away
- 75% of our internal traffic is now IPv6
  - 100% Q3 2014 (or earlier)
- 98% of traffic in & out of HHVM is IPv6
- 100% of our memcache traffic is IPv6
- 100% IPv6 only (no RFC1918) in 2-3 years
Where are we now? (cont)

- New IPv6 traffic showing up daily
  - Engineers asking if they can start writing IPv6-only code today
- Latency and other metrics show IPv6 to be the same as IPv4
- Plans for first IPv6 only cluster (no RFC1918) by end of 2014
- We will not remove RFC1918 from existing clusters