IPv6 Only Experience at T2_US_Nebraska

HEPiX IPv6 Working Group F2F Meeting

July 5th and 6th, 2018

Garhan Attebury <garhan.attebury@cern.ch>

Motivation

“~~CAPSLOCK~~ Colons are cruise control for cool -.-”

- Almost everything dual stacked for a few years now with success… but what about pure IPv6 only?
- Necessary? Not *really*, but when has that ever been the point?
- Goal: make IPv6 only hosts do what dual stack ones can

IPv6 Only: Round 1

The low hanging fruit

- Provisioning…
- Pure IPv6 PXE support exists (in theory) in newer hardware
- Hybrid approaches exist like bootstrapping via iPXE
- Accept failure, provision with v4 as you might still need that anyway

IPv6 Only: Round 2

The marginally higher hanging fruit

- External repos without IPv6

  pkg.duosecurity.com (used to have v6 right?)
  yum.puppetlabs.com (same here, why are we going backwards?)
  repo.opensciencegrid.org (seriously people?! … ohh wait)

- Fixable with local mirrors / easy button hiera knobs in puppet
  … but building technical debt

IPv6 Only: Round 3

Software compatibility

- A few years ago the list of things not "speaking" IPv6 was high
  … much better picture these days

- Few components we use needed config tweaks / upgrading

  Ganglia / check_mk / frontier-squid

- "Weird" things like SSSD ldap_uri parsing (which is 100% fine) or autofs segfaulting because puppet can't look up HDFS namenode

- HDFS might never support IPv6
  Had to work around various checks ensuring HDFS and FUSE function

IPv6 Only: Round 4

Never forget the condor knobs

- Two critical knobs in the 8.6.x series at least:

  ```
  PREFER_IPV4 = False
  IPV4_ENABLE = False
  ```

- It … just works?

IPv6 Only: Round 5

Enter the docker

```
Handler for POST /v1.26/containers/create returned error: No such image: unlhcc/osg-wn-el6:latest
Error getting v2 registry: Get https://hcc-docker-registry.unl.edu/v2/: dial tcp 129.93.175.38:443: connect: network is unreachable
Attempting next endpoint for pull after error: Get https://registry-1.docker.io/v2/: dial tcp 34.200.90.16:443: connect: network is unreachable
```

- Again with the world not drinking the IPv6 koolaid …
  ohh wait, that's us again :(
- Could run local registry, which we do, just not correctly
- Load image manually… debt continues to build
- Side note: using NDP proxying for containers as testbed switch is only a pretend L3 switch, not a proper one

IPv6 Only: Round 6

Light at the end of the tunnel

```
May 13 09:43:46 red-c1005.unl.edu cvmfs2: (cms.cern.ch) switching proxy from http://129.93.239.137:3128 to http://131.225.205.134:3126
May 13 09:43:46 red-c1005.unl.edu cvmfs2: (cms.cern.ch) switching proxy from http://131.225.205.134:3126 to http://131.225.205.133:3126
May 13 09:43:46 red-c1005.unl.edu cvmfs2: (cms.cern.ch) switching proxy from http://131.225.205.133:3126 to http://128.142.33.31:3126
May 13 09:43:46 red-c1005.unl.edu cvmfs2: (cms.cern.ch) switching proxy from http://128.142.33.31:3126 to http://128.142.168.202:3126
May 13 09:43:46 red-c1005.unl.edu cvmfs2: (cms.cern.ch) failed to download repository manifest (6 - proxy connection problem)
```

When connecting to a proxy, by default it will try on the IPv4 address unless the proxy only has IPv6 addresses configured. The CVMFS_IPFAMILY_PREFER=[4|6] parameter can be used to select the preferred IP protocol for dual-stack proxies.

IPv6 Only: Final Round

Prognosis: not very six-ish

```
root     27815  0.1  0.1 1342040 40124 ?     Ssl  May12   0:55 /usr/bin/dockerd-current --add-runtime docker-runc=/usr/libexec/docker/docker-runc-current --default-runtime=dock
root     27826  0.0  0.0 917584 18696 ?      Ssl  May12   0:30  \_ /usr/bin/docker-containerd-current -l unix:///var/run/docker/libcontainerd/docker-containerd.sock --metrics-i
root       344  0.0  0.0 412932  4140 ?      Sl   12:56   0:00      \_ /usr/bin/docker-containerd-shim-current 9f56e063af748145071ab7be2a6f7f4f1e0d55b03e1ec4203ad0d6ac8d092035
cmsprod    362  0.9  0.0  24332  2064 ?      Ss   12:56   0:01          \_ /bin/bash ./condor_exec.exe -v std -name gfactory_instance -entry CMS_T2_US_Nebraska_Red_gw1_whole_cm
cmsprod   7151  0.3  0.0  23940  1748 ?      S    12:58   0:00              \_ /bin/bash /var/lib/condor/execute/dir_320/glide_zIZdB0/main/condor_startup.sh glidein_config
cmsprod   8112  0.0  0.0  55332  5784 ?      S    12:58   0:00                  \_ /var/lib/condor/execute/dir_320/glide_zIZdB0/main/condor/sbin/condor_master -f -pidfile /var/
```

This is looking promising…

```
05/13/18 17:58:18 (pid:7388) attempt to connect to <188.184.83.197:9685> failed: Network is unreachable (connect errno = 101).  Will keep trying for 300 total seconds (299 to go).
05/13/18 18:03:17 (pid:7388) attempt to connect to <188.184.83.197:9685> failed: Network is unreachable (connect errno = 101).
05/13/18 18:03:17 (pid:7388) CCBListener: connection to CCB server vocms0806.cern.ch:9685 failed; will try to reconnect in 60 seconds.
05/13/18 18:03:20 (pid:7388) attempt to connect to <131.225.205.232:9685> failed: Network is unreachable (connect errno = 101).  Will keep trying for 300 total seconds (297 to go).
```

… or not

vocms0806.cern.ch = IPv4 only
cmssrv258.fnal.gov = IPv4 only
gfactory-1.t2.ucsd.edu = IPv4 only

Final score

Almost there…

- Functioning worker node?
  Yes, so long as you don't actually want to run CMS jobs
- Most IPv6 issues are fixable with config tweaks or upstream/external support
- Globus (gfal-copy / globus-url-copy) work fine
  XRootD (xrdcp) works fine
- Just factory support remaining? USCMS only issue?
- **Update** the USCMS factory IPv6 support is *there* in theory, and has been tested dual stack in the past, but is "off" until some other work is finished and it can be rolled out safely (again)

(Extra slide: IPv6 testbed addressing)

(basically straight out of a Docker IPv6 page)

- Use a single /64 for testbed because “easy enough”
  2600:900:6:1105::/64

- Split into /76 nets for each physical host (4096 possible hosts)
  red-c1005 = 2600:900:6:1105:50::/76
  red-c1006 = 2600:900:6:1105:60::/76
  red-c1007 = 2600:900:6:1105:70::/76
  … etc

- DOCKER_NETWORK_OPTIONS set to unique /80 for each host. Minimum recommended is /80 to allow container IPs to end with container’s MAC
  red-c1005 = —fixed-cidr=2600:900:6:1105:51::/80
  —> container #1 = 2600:900:6:1105:51:242:ac13:4/80
  —> container #2 = 2600:900:6:1105:51:242:ac13:2/80
  … etc

- Can have 16 /80’s per host. Really just need one which can have plenty of containers within. Not implying this is “best practice” or even a good idea, but it has worked well enough for the few hosts in the testbed