Deployment of 464XLAT (RFC6877) alongside IPv6-only CPU resources at WLCG sites

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 082029 doi:10.1088/1742-6596/898/8/082029

Deployment of 464XLAT (RFC6877) alongside IPv6-only CPU resources at WLCG sites

T S Froy, D P Traynor, C J Walker

Queen Mary University of London, Mile End Road, E1 4NS, UK

E-mail: t.froy@qmul.ac.uk

Abstract. IPv4 is now officially deprecated by the IETF. A significant amount of effort has already been expended by the HEPiX IPv6 Working Group on testing dual-stacked hosts and IPv6-only CPU resources. Dual-stack adds complexity and administrative overhead to sites that may already be starved of resource. This has resulted in a very slow uptake of IPv6 from WLCG sites. 464XLAT (RFC6877) is intended for IPv6 single-stack environments that require the ability to communicate with IPv4-only endpoints. This paper will present a deployment strategy for 464XLAT, operational experiences of using 464XLAT in production at a WLCG site and important information to consider prior to deploying 464XLAT.

1. Introduction

The Queen Mary WLCG tier two site operated a predominantly IPv4-only network from the initial site deployment until early 2013 when IPv6 connectivity was provided by the central Networks team at Queen Mary; the site team have fervently worked to enable all public-facing services to be accessible over IPv6. This goal has been met as of January 2017.

As those who operate WLCG infrastructure are aware, the story does not end there; a significant number of hosts which are not public-facing, such as numerous compute nodes, are still constrained with being able to communicate only with the IPv4 Internet. In this paper we describe the deployment of an open source IPv4/IPv6 translation layer alongside IPv6-only compute nodes. This allows us to move towards single stack compute nodes - simplifying the administration of the cluster.

2. Software Choice

It was a very easy decision to choose the Jool SIIT/NAT64 [1] implementation; the code is distributed under an open-source license, local knowledge and significant experience existed within the site team, immediate logging of translated sessions plus with the translation logic implemented entirely in kernelspace there is minimal overhead on throughput/latency for translated traffic.

The site team also required DNS resolver software that fully implemented DNS64 so, again, due to the open-source license, local knowledge and experience within the team, the decision to deploy PowerDNS Recursor 4.x [2] was made.

3. How It All Fits Together

The DNS64 flow diagram presented here provides an overview as to how hostname resolution works in this environment.

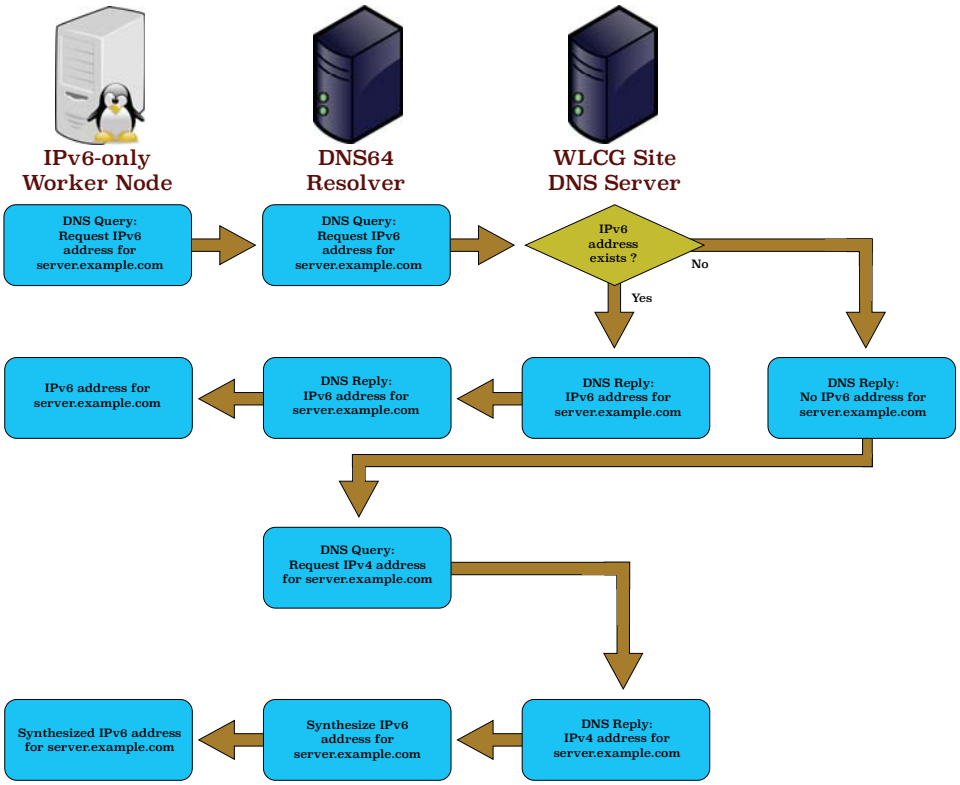

Figure 1. DNS64 Flow Diagram

As DNS64 is effectively forging answers to clients, it must be recognized that if a client is performing DNSSEC validation and the A record returned by the WLCG Site DNS Server is DNSSEC-signed, the client will reject the synthesized AAAA record as bogus.

It is highly recommended that the DNS64 Resolver performs DNSSEC validation (as the original A record can be verified at this stage) and clients are configured not to validate themselves but will instead trust the Authenticated Data (AD) bit set in the synthesized DNS response.

In the event of the remote endpoint having native IPv6 connectivity with an AAAA record published in DNS, PowerDNS Recursor simply passes on the AAAA record as-received to the IPv6-only compute node exactly how a regular DNS resolver would behave. As a result of this, the IPv6-only compute node will connect directly to the resulting IPv6 address using native IPv6 connectivity.

Alternatively, if the remote endpoint does not have an AAAA record but does have an A record, PowerDNS Recursor will take the /96 prefix that we defined in dns64.lua, append the 32-bit IPv4 address to the end of it and synthesize a 128-bit IPv6 address which is supplied to the IPv6-only compute node as a response to its query for an AAAA record.

Due to the AAAA record resulting in an IPv6 address which falls within the /96 prefix routed to the Jool Translator host, any connections made by the IPv6-only compute node will flow through the Jool software.

RFC6052 [3] "IPv6 Addressing of IPv4/IPv6 Translators" documents the 64:ff9b::/96 IPv6 prefix as a "Well-Known Prefix" for use in algorithmic mapping. As this range is assigned by IANA specifically for this purpose, we have chosen it as the range in which to generate synthesized AAAA records.

The NAT64 flow diagram provides an overview as to how translation is performed between address families.

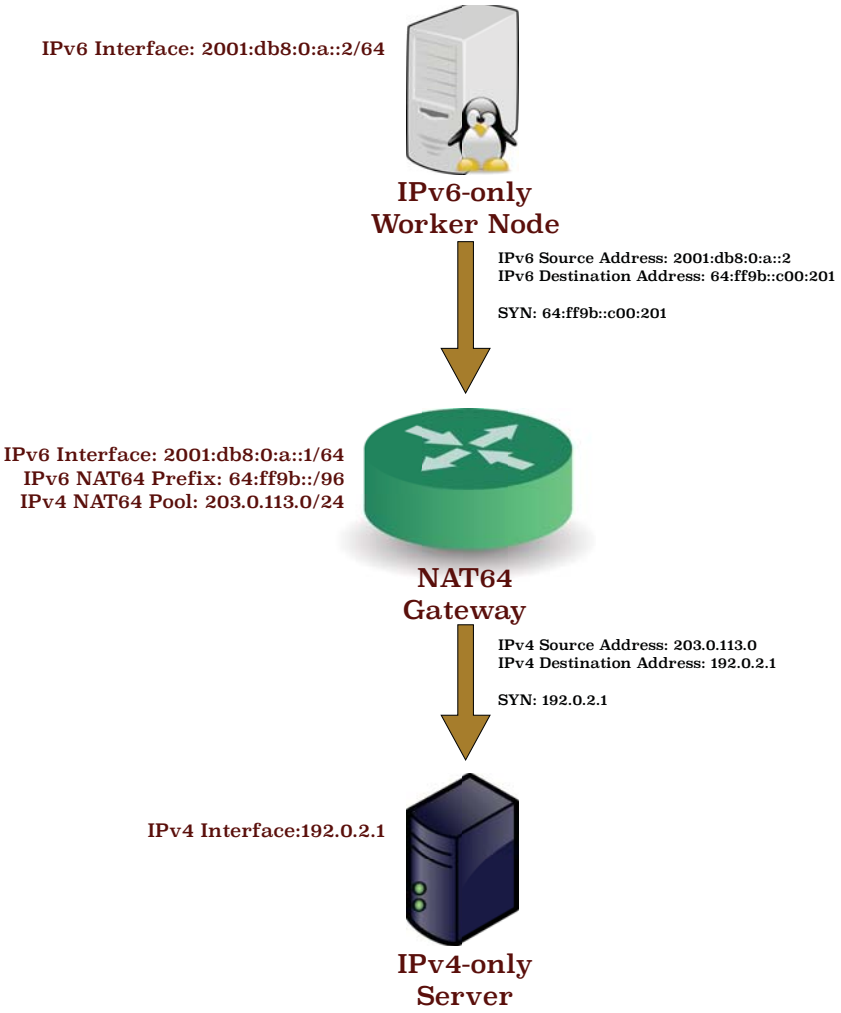

Figure 2. NAT64 Flow Diagram

4. Compute Node Network Setup

The site team opted to add a small amount of complexity to the IPv6-only nodes so that the freedom to selectively enable/disable NAT64 on a per-node basis was possible.

To that end, the following changes were made to each IPv6-only compute node to remove the IPv4 route and send traffic to NAT64 synthesised addresses via our Jool translation node at 2001:db8::6464 :

```
ip -6 route add 64:ff9b::/96 via 2001:db8::6464
ip route del default
echo "nameserver 2001:db8::16" > /etc/resolv.conf
```

We are unable to remove IPv4 completely from our infrastructure due to the use of StoRM/Lustre[4] for bulk data storage. Lustre 2.x is IPv4 only and we have deployed it using RFC1918 [5] address space. To ensure that these nodes are indeed IPv6-only as far as the rest of the WLCG is concerned, the IPv4 defaultroute is removed.

5. Jool Translator Network Setup (NAT64)

Jool is installed on a regular CentOS 7 host which is equipped with two interfaces; the first interface is connected to the same network as the compute node farm and is configured with an IPv6 address of 2001:db8::6464/64, the second interface is connected to an IPv4-only network with an IPv4 address of 192.0.2.1/24.

The Jool software can be configured to use any IPv6 /96 prefix for translation purposes. The site team chose to use the RFC6052 [3] “Well known prefix” of 64:ff9b::/96, which is the default, as the site team had no operational requirement to make the translation mechanism accessible to the outside world.

In order for Jool to function properly, there must be a pool of routable IPv4 addresses available for use:

```
/etc/modprobe.d/jool.conf:
options jool disabled=1 pool6=64:ff9b::/96 \
    pool4=192.0.2.4/30,192.0.2.8/29,192.0.2.16/28,192.0.2.32/28,192.0.2.48/29
```

This configuration tells Jool to perform address family translation on any packets destined for the network defined in pool6 and to source translated traffic from addresses specified in pool4.

The sharp-eyed reader may have noticed that the module is disabled at load time but this is due to the requirement to perform further tweaks on the host prior to enabling Jool; for example, due to the difficulties faced when trying to get our central Networks team to provision either static or dynamic routing to our network, we are required to proxy ARP for the IPv4 addresses specified in pool4 and so this is accomplished as part of our enable-jool.sh script which is called during the final stages of boot (post-networking):

```
#!/bin/bash
#
# Enable proxy ARP on VLAN842
echo 1 > /proc/sys/net/ipv4/conf/bond0.842/proxy_arp

# Install IPv4 routes for all of the Jool pool4 prefixes
ip route add 192.0.2.4/30 dev lo
ip route add 192.0.2.8/29 dev lo
ip route add 192.0.2.16/28 dev lo
ip route add 192.0.2.32/28 dev lo
ip route add 192.0.2.48/29 dev lo

#  Enable IPv4/IPv6 forwarding
echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding

# Turn off all firewalling
iptables -F
iptables -X
ip6tables -F
ip6tables -X

# Turn off LRO/GRO on all NICs
ethtool --offload enp10s0f0 lro off
ethtool --offload enp10s0f0 gro off
ethtool --offload enp10s0f1 lro off
ethtool --offload enp10s0f1 gro off

# Enable NAT64
jool --source-icmpv6-errors-better=on > /dev/null 2>&1
jool --enable > /dev/null 2>&1
```

There is no firewalling on the node for performance reasons principally because there is no requirement to block traffic on a host which is merely responsible for translating between IPv4 and IPv6. Firewalling can and should be enforced on the endpoints which are utilizing the services of the Jool Translator.

Turning off LRO/GRO (Large Receive Offload/Generic Receive Offload) on NIC hardware is recommended by the authors of Jool [1].

Turning Jool's source-icmpv6-errors-better parameter to on fixes traceroute from IPv6 hosts to IPv4 hosts and is recommended by the authors of Jool [1].

Finally, we enable Jool so that it can start translating.

6. PowerDNS Recursor Setup (DNS64)

PowerDNS is installed on a regular CentOS 7 host which is equipped with a single interface which is configured with an IPv6 address of 2001:db8::16/64 and an IPv4 address of 198.51.100.1/24.

The stock PowerDNS configuration must be modified to facilitate the use of DNS64 as described below.

The IPv6 prefix(es) used by the IPv6-only compute nodes need to be added to the allow from= line in /etc/pdns-recursor/recursor.conf.

```
/etc/pdns-recursor/recursor.conf:
...
allow from=<snip>,2001:db8::/64
...
```

Add lua-dns-script=/etc/pdns-recursor/dns64.lua to /etc/pdns-recursor/recursor.conf.

```
/etc/pdns-recursor/recursor.conf:
...
lua-dns-script=/etc/pdns-recursor/dns64.lua
...
```

dns64.lua should be obtained from https://github.com/PowerDNS/pdns/blob/master/pdns/dns64.lua and needs to be fixed up with the following changes.

The variable 'prefix' must be defined as "64:ff9b::" in /etc/pdns-recursor/dns64.lua to match the "Well known prefix" of RFC6052 configured for Jool in the previous section.

```
/etc/pdns-recursor/dns64.lua:
...
prefix = "64:ff9b::"
...
```

In order for IPv4 reverse DNS entries to be visible to IPv6-only compute nodes, the parameter passed to newDN() needs to be changed from "f.f.7.7.b.1.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa." to "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.b.9.f.f.4.6.0.0.ip6.arpa." in /etc/pdns-recursor/dns64.lua (once again to match the "Well known prefix" of RFC6052).

```
/etc/pdns-recursor/dns64.lua:
...
if dq.qtype == pdns.PTR and
    dq.qname:isPartOf(newDN("0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.b.9.f.f.4.6.0.0.ip6.arpa."))
    then
...
```

7. The Final Piece of the Puzzle (CLAT)

464XLAT, and the principal topic of this paper, consists of two key components:

- A PLAT (Provider-side Translator) which has been described in this paper as NAT64 and has been implemented at the Queen Mary tier two site using Jool.
- A CLAT (Client-side Translator) which has been implemented at the Queen Mary tier two site using Tore Andersen's clatd [6] which depends heavily on Nathan Lutchansky's TAYGA [7] software and works as follows:

The CLAT resides on the IPv6-only compute node and when it is started, it creates a virtual network interface, complete with IPv4 address and IPv4 defaultroute; which IPv4 addresses are ultimately used in these cases is configurable although the site team had no operational requirement to change the defaults in this case.

Any IPv4 packets sent by applications running on the node which enter the virtual IPv4 interface have the source IPv4 address translated to an IPv6 address configured on the IPv6-only compute node and the destination IPv6 address is translated as per the algorithm used by the DNS64 resolver (64:ff9b::/96 + 192.0.2.10 = 64:ff9b::192.0.2.10).

The CLAT passes the translated packet through to the host networking stack which handles it as per any regular IPv6 packet; as described earlier, traffic to 64:ff9b::/96 is handled by the PLAT which, in turn, translates the IPv6 packet back to IPv4.

It should be noted that DNS64 is optional when using a CLAT; likewise, DNS64 negates the requirement for a CLAT if compute nodes do not need to communicate with IPv4 literals.

8. Monitoring and Logging

It can be useful to log the DNS requests made by IPv6-only compute nodes and the responses from the remote authoritative servers to determine which endpoints are IPv4-only (only A record returned), dual-stack (A and AAAA records returned) and those which are IPv6-only (only AAAA records returned); this provides a metric by which the WLCG transition to IPv6 can be easily measured.

Due to the requirement of the Queen Mary Network Acceptable Use Policy and that of our upstream, JANET, we have an operational requirement to log the origin of every communication which is initiated from our network; either to a specific host or to a specific user.

The Jool userspace utility has the ability to extract and export data concerning the sessions currently traversing the translation mechanism; this permits the ability to retain logs pursuant to our obligations under the Acceptable Use Policies which we are required to follow and it also provides the site team and the WLCG community with a very useful source of data in the form of IPv4-only hosts which have yet to be made either dual-stack or IPv6-only in order to complete the WLCG transition to IPv6 and will allow us to decommission the DNS64/NAT64/464XLAT services on our network.

Exporting the information is relatively simple:

```
jool -s -n --csv | <script which does something with stdin>
```

9. Operational Experiences

The deployment of IPv6-only compute nodes at Queen Mary has gone incredibly smoothly.

Given the wide range of VOs that the Queen Mary tier two site supports, the site team decided that deploying IPv6-only compute nodes without any transition mechanism such as that described in this paper would have caused significant problems as evidenced by the parts of the WLCG which are still IPv4-only.

It was noted that every single session which has been established via the Jool Translator node would have meant job failure if such a translation mechanism had not been deployed; although, one concern that the site team did have was the possibility that users were specifying IPv4 literals in their jobs, such as “wget http://192.0.2.10/” instead of “wget http://www.example.com/” as there is no way for DNS64 to work around that because the DNS64 resolver is never asked to resolve a hostname.

The principal use-case for 464XLAT is to handle the requirement for a client to connect to IPv4 literals (such as 192.0.2.10) in addition to hostnames (such as www.example.com); in reality, we have not experienced any WLCG job failures on IPv6-only nodes where we have selectively disabled the CLAT.

The deployment of a CLAT on each compute node should be considered optional as we have not identified any jobs submitted to the Queen Mary tier two site which require communication with IPv4 literals.

There is no requirement for DNS64 if a CLAT is deployed on each compute node but as there is currently no CLAT implementation which resides in kernelspace; the site team decided on using DNS64 with a kernel-based NAT64 implementation in order to translate the majority of traffic that is not targeted at IPv4 literals as already described and to supplement this with a

userspace CLAT per compute node in order to handle the remaining traffic that is targeted at IPv4 literals.

As with any device in the traffic path which does not blindly forward traffic, the translation mechanism will incur some performance overhead; although, this has been proven to be extremely small [9].

10. Future Work

We intend to migrate all compute nodes at the Queen Mary tier two site to IPv6-only which will then permit us to investigate the feasibility of assigning unique IPv6 addresses on a per-job basis which would allow us to identify network usage at a much finer-grained level.

Implementing SIIT-DC BRs [10] on our network border for the purpose of completely removing IPv4 from our dual-stack hosts without impacting on connectivity to IPv4 hosts.

A number of the IPMI management interfaces do not support IPv6, so IPv4 will continue to remain on the out-of-band management network until they are retired.

11. Conclusions

Details of a successful implementation of DNS64/NAT64/464XLAT at the Queen Mary WLCG tier two site have been presented. We have demonstrated that a CLAT is unnecessary for WLCG jobs - simplifying deployment for other sites. We have identified parts of the WLCG experiment infrastructure that need to be dual stack to support IPv6 only sites. Sites should consider deploying NAT64/DNS64 instead of NAT - as the proportion of traffic using IPv6 grows, the performance impact of NAT will diminish.

12. Reference

[1] Jool SIIT/NAT64 Implementation for Linux (co-developed by ITESM and NIC México): `http://www.jool.mx/`
Getting Rid Of Receive Offloads: `http://www.jool.mx/en/offloads.html`
–global Userspace Flags: `https://www.jool.mx/en/usr-flags-global.html`
[2] PowerDNS Recursor 4.x (developed by PowerDNS.COM BV): `https://www.powerdns.com/recursor.html`
[3] RFC6052 IPv6 Addressing of IPv4/IPv6 Translators: `https://www.ietf.org/rfc/rfc6052.txt`
[4] Upgrading and Expanding Lustre Storage for use with the WLCG: D P Traynor, T S Froy, C J Walker. *Journal of Physics: Conference Series* This issue.
[5] RFC1918 Address Allocation for Private Internets: `https://www.ietf.org/rfc/rfc1918.txt`
[6] clatd (developed by Tore Anderson): `https://github.com/toreanderson/clatd`
[7] TAYGA (developed by Nathan Lutchansky): `http://www.litech.org/tayga/`
[8] RFC6877 Combination of Stateful and Stateless Translation (464XLAT): `https://www.ietf.org/rfc/rfc6877.txt`
[9] Adira Quintero, Francisco Sans, Eric Gamess,"Performance Evaluation of IPv4/IPv6 Transition Mechanisms", International Journal of Computer Network and Information Security(IJCNIS), Vol.8, No.2, pp.1-14, 2016.DOI: 10.5815/ijcnis.2016.02.01
[10] RFC7756 Stateless IP/ICMP Translation for IPv6 Internet Data Centre Environments: `https://www.ietf.org/rfc/rfc7756.txt`