Tutorial

Improving Domain Name System resolution in your ISP network

Gael Hernandez, Interconnection Policy Manager
Dibya Khatiwada, Peering Coordinator

November 2019

Table of Contents

Disclaimer: This is not a technical tutorial on how the Domain Name System works. For further information on the topic, please read the appropriate RFCs.

Introduction

This tutorial covers the fundamentals of how networks peering with Packet Clearing House over an Internet exchange point can benefit from improved Border Gateway Protocol reachability and Domain Name System resolution.

At network level, direct BGP reachability via peering decreases the number of hops and latency between your network's edge router and our name server cluster. At DNS level, the resolution time for domains of which we are authoritative is reduced because DNS requests are answered locally by our name servers.

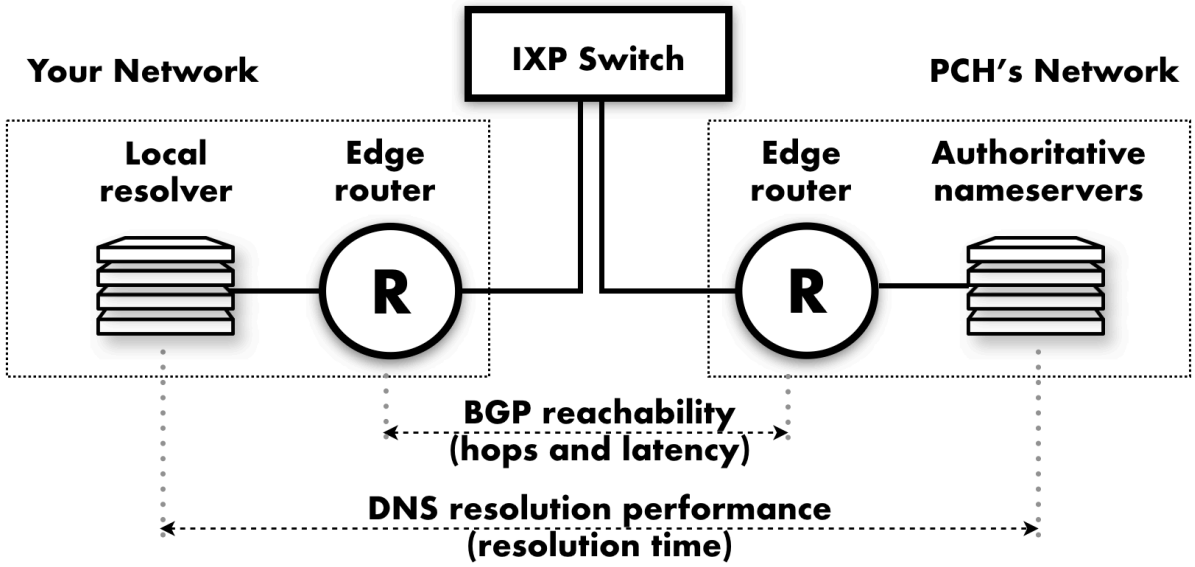

Figure 1: Diagram showing PCH network and an ISP network interconnected via an IXP switch.

The tutorial includes guidance and examples to troubleshoot and debug reachability and performance issues between your network and PCH's network.

PCH anycast DNS services

PCH operates the world's first and largest DNS content delivery network providing Internet users, in both well connected and poorly connected regions — with the quickest and most resilient DNS user experience.

PCH's global anycast network provides infrastructure for two of the thirteen letters of the root zone and more than 400 top-level domains including 112 country-code TLDs (see Appendix A). PCH also hosts critical-infrastructure domains, including the in-addr.arpa and ip6.arpa domains, and the domains of Internet exchange points, CERTs, and national governments. The map below shows the countries whose top-level domains use PCH's infrastructure.

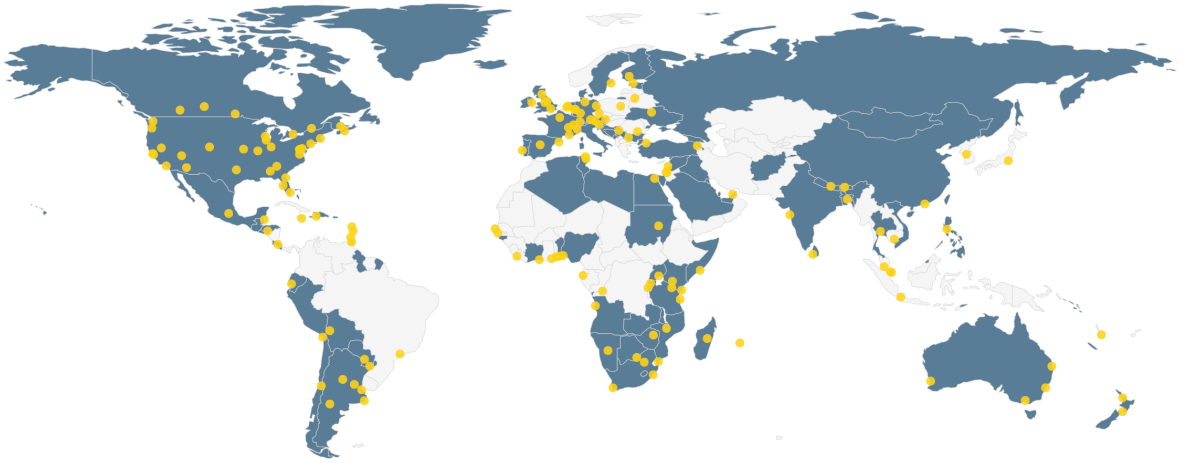

Figure 2: Diagram showing countries whose top-level domains use PCH infrastructure.

PCH's anycast network also provides infrastructure to the Quad9 project, a free and privacy-centric recursive DNS resolver that offers end users robust security protection. More information on the Quad9 project is available at https://www.quad9.net

Deployment of a DNS cluster at your local IXP

PCH owns and operates large clusters of DNS servers at major IXPs including those in Washington, Palo Alto, Frankfurt, Amsterdam, London, Paris, Tokyo, Singapore, Hong Kong and Johannesburg. We also operate smaller server clusters in many poorly-connected regions, where our services, unlike those of other DNS operators, remain available to local users during international connectivity disruptions.

At the time of writing this tutorial (Sep 2019), PCH network is available for public peering at 196 exchange points across 168 cities on six continents.[1]

If you are reading this tutorial you have most likely a DNS cluster already installed at your local IXP but if you are an Internet exchange point operator interested in hosting a DNS node, email us at info@pch.net and we can discuss technical and operational requirements.

Establishing BGP sessions with PCH's network

PCH operates two independent autonomous systems: AS42 for the anycast DNS network and AS3856 for research purposes, which includes a route collector and a looking glass. For maximum reliability of our services, we always aim to peer both bilaterally and through the route-server service if it is available at the exchange. Keep this in mind when troubleshooting reachability issues; you should be able to access the PCH network by either means.

PCH has an open peering policy: we attempt to peer with all participants at every Internet exchange point regardless of their volume or inbound/outbound traffic ratio as long as they

operate responsibly and in agreement with generally recognized best practices (see Appendix B for requirements).

When peering with us, please accept all prefixes we announce in our BGP session. A list of prefixes and ASNs is published via the AS macro AS-PCH (RADB) and is also available on demand at peering@pch.net. We recommend accepting all prefixes, since the number of services announced in your node might increase over time. You can set limits to the number of prefixes you receive from us or have a strict filtering policy, but please make sure you are aware of those when debugging a reachability issue.

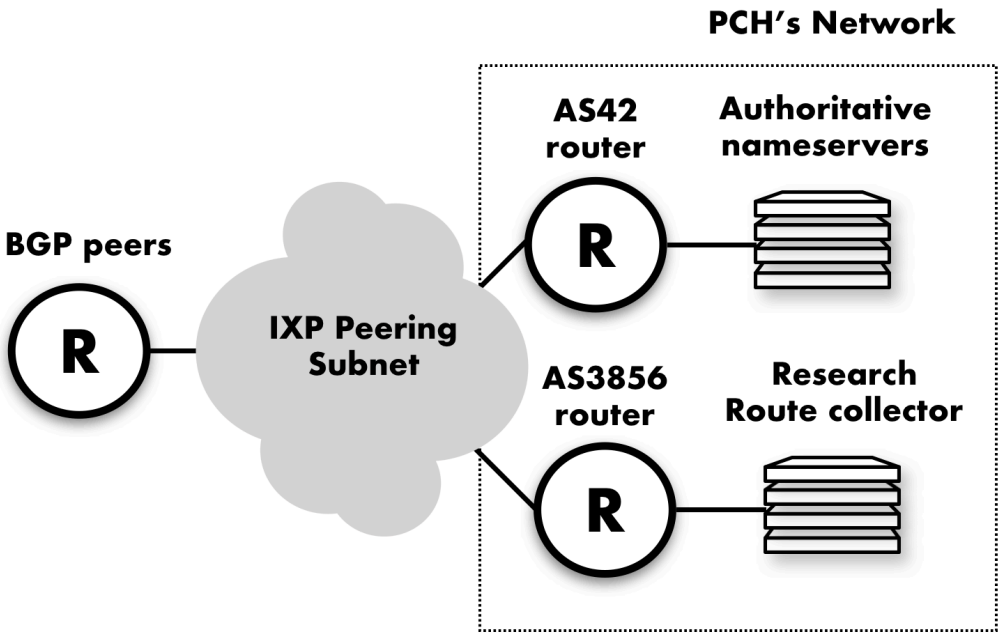

Figure 3: Diagram showing PCH's network connected to a BGP peer via an IXP peering subnet

We expect the peering network to send us its own prefixes as well as their customer prefixes. In turn we also expect our peering partner to propagate our prefixes to its customers to ensure routing efficiency across the downstream AS path.

Checking direct reachability to PCH's network

The simplest way to check that your network is reaching PCH over the IXP switch fabric with minimal latency is to perform a traceroute to anyns.pch.net.

```
$ traceroute -a anyns.pch.net
traceroute to anyns.pch.net (204.61.216.4), 64 hops max, 52 byte packets
 1  [AS0] 192.168.1.1 (192.168.1.1)  1.574 ms  1.286 ms  1.575 ms
 2  [AS17501] 1.217.166.202.ether.static.wlink.com.np (202.166.217.1)  2.059 ms  1.991 ms
2.148 ms
 3  [AS17501] xe-0-0-8-137.40.gw-jwl-core-02.wlink.com.np (202.79.40.137)  5.998 ms  3.704
ms  2.223 ms
 4  [AS4557] 198-32-231-20.setg.net (198.32.231.20)  2.571 ms  2.447 ms  2.650 ms
 5  [AS42] anyns.pch.net (204.61.216.4)  2.397 ms  2.614 ms  2.963 ms
```

```
$ traceroute6 anyns.pch.net
traceroute to anyns.pch.net (2001:500:14:6004:ad::1) from 2404:2c00:2::145, 30 hops max, 24
byte packets
 1  gw-noc-pool1.nren.net.np (2404:2c00:2::1)  0.868 ms  0.536 ms  0.443 ms
 2  npix-pts.woodynet.net (2404:2c00:ffff:e::20)  0.8 ms  0.83 ms  0.592 ms
 3  anyns.pch.net (2001:500:14:6004:ad::1)  0.869 ms  1.218 ms  0.969 ms
```

In the example above, the output shows that after two hops in the local network AS17501, packets reached the Nepal Internet exchange (peering subnet 198.32.231.0/24). The minimal latency (< 5ms) also confirms direct reachability.

Checking BGP announcements

In the case that a traceroute to anyns.pch.net shows that packets are not being routed via the Internet exchange (the second to last hop should be from an IP address contained in the peering subnet), we need to check BGP announcements and make sure that routers are learning the routes advertised by each other.

To this end, we first check that BGP sessions are up with the output of "show ip bgp summary." Then we use the "show ip bgp *ip-addr* advertised-routes" command to present the list of prefixes being advertised to you. We would generally email you that information, for example:

```
The BGP sessions have established to both of our routers. Can you please confirm if you are
receiving the below mentioned prefixes from us?

AS42:
198.32.231.18    4         45170   103047   103021    409148    0     0 9w2d             4
2404:2C00:FFFF:E::18
                 4         45170   102994   103057       519    0     0 9w2d             1


AS3856:
198.32.231.18    4 45170   103717    94258         0     0     0 09w2d02h         4
2404:2c00:ffff:e::18
                 4 45170   103719    94257         0     0     0 09w2d02h         1
```

The command "show ip bgp *ip-addr* advertised-routes" shows the prefixes we are advertising via that BGP session. In the example below, we are advertising forty-one prefixes from AS42 and one prefix from AS3856. The equivalent command and output for v6 addresses/prefixes is showed in Appendix C due to its length.

```
router.as42# sh ip bgp neighbors 196.223.4.16 advertised-routes
BGP table version is 90, local router ID is 199.184.184.38
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  72.42.112.0/24   66.102.34.204            0             0 187 i
 *>  72.42.113.0/24   66.102.34.204            0             0 187 i
 *>  72.42.114.0/24   66.102.34.204            0             0 187 i
 *>  72.42.115.0/24   66.102.34.204            0             0 187 i
 *>  72.42.116.0/24   66.102.34.204            0             0 187 i
 *>  72.42.117.0/24   66.102.34.204            0             0 187 i
```

```
 *>  72.42.118.0/24   66.102.34.204           0             0 187 i
 *>  72.42.119.0/24   66.102.34.204           0             0 187 i
 *>  72.42.120.0/24   66.102.34.204           0             0 187 i
 *>  72.42.121.0/24   66.102.34.204           0             0 187 i
 *>  72.42.122.0/24   66.102.34.204           0             0 187 i
 *>  72.42.123.0/24   66.102.34.204           0             0 187 i
 *>  72.42.124.0/24   66.102.34.204           0             0 187 i
 *>  72.42.125.0/24   66.102.34.204           0             0 187 i
     Network          Next Hop            Metric LocPrf Weight Path
 *>  72.42.126.0/24   66.102.34.204           0             0 187 i
 *>  72.42.127.0/24   66.102.34.204           0             0 187 i
 *>i 114.69.222.0/24  66.102.34.200           0    100      0 i
 *>i 149.112.112.0/24 66.102.34.240           0    100      0 i
 *>i 149.112.149.0/24 66.102.34.240           0    100      0 i
 *>i 170.210.180.0/24 66.102.34.200           0    100      0 i
 *>i 189.201.244.0/23 66.102.34.200           0    100      0 i
 *>i 194.0.17.0       66.102.34.202           0    100      0 i
 *>i 194.0.27.0       66.102.34.200           0    100      0 i
 *>i 194.0.36.0       66.102.34.202           0    100      0 i
 *>i 194.0.42.0       66.102.34.202           0    100      0 i
 *>i 194.0.47.0       66.102.34.202           0    100      0 i
 *>i 194.117.58.0     66.102.34.200           0    100      0 i
 *>i 198.182.167.0    66.102.34.208           0    100      0 i
 *>i 199.4.137.0      66.102.34.200           0    100      0 i
 *>  199.7.91.0       66.102.34.218           0             0 27 i
 *>i 199.254.171.0    66.102.34.200           0    100      0 i
 *>i 200.108.148.0    66.102.34.200           0    100      0 i
 *>  203.119.88.0/23  66.102.34.204           0             0 187 i
 *>i 204.19.119.0     66.102.34.200           0    100      0 i
 *>i 204.26.57.0      66.102.34.202           0    100      0 i
 *>i 204.61.216.0/23  66.102.34.200           0    100      0 i
     Network          Next Hop            Metric LocPrf Weight Path
 *>  205.132.46.0     0.0.0.0                 0         32768 i
 *>i 206.51.254.0     66.102.34.202           0    100      0 i
 *>i 206.51.255.0     66.102.34.202           0    100      0 i
 *>i 207.34.6.0       66.102.34.202           0    100      0 i
 *>i 207.34.7.0       66.102.34.202           0    100      0 i

Total number of prefixes 41

route-collector.as3856# sh ip bgp neighbors 196.223.4.16 advertised-routes
BGP table version is 0, local router ID is 66.102.34.196
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 69.166.10.0/24   196.223.4.19             0         32768 i

Total number of prefixes 1
```

If the Quad9 service is enabled at your location (which is the default option in PCH deployments), then you should receive the following v6 prefix 2620:FE::/48 and four v4 prefixes 9.9.9.0/24, 149.112.112.0/24, 149.112.149.0/24 and 199.249.255.0.

```
     Network          Next Hop            Metric LocPrf Weight Path
*>  9.9.9.0/24       74.80.106.244           0             0 19281 i
*>  149.112.112.0/24 74.80.106.244           0             0 19281 i
*>  149.112.149.0/24 74.80.106.244           0             0 19281 i
*>  199.249.255.0    74.80.106.244           0             0 19281 i

*>  2620:FE::/48     2620:171:24::242        0             0 19281 i
```

We also send you the list of prefixes we are learning from you via the output of the “show ip bgp *ip-addr* routes” command. You should check that we are receiving all prefixes you are announcing to us. The equivalent command and output for IPv6 is shown below:

```
router.as42#sh ip bgp neighbors 196.223.4.16 routes
BGP table version is 90, local router ID is 199.184.184.38
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  41.191.68.0/22   196.223.4.16             0             0 37190 i
 *>  154.0.24.0/21    196.223.4.16             0             0 37190 i

Total number of prefixes 2
```

```
router.ktm#sh bgp ipv6 uni neighbors  2404:2c00:ffff:e::18 routes
BGP table version is 519, local router ID is 204.61.210.46
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>  2404:2C00::/32   2404:2C00:FFFF:E::18
                                               0             0 45170 i

Total number of prefixes 1
```

Checking that the BGP announcements are correct and that each network is advertising and receiving the expected number of prefixes and routes is a good operational practice. If your network does not advertise all your prefixes to us, the return path may use your transit provider rather than the peering session, creating an asymmetrical path. Asymmetrical paths can generate performance issues later and debugging such issues is more time-consuming than verifying the BGP announcements first.

To summarize, this is the checklist we recommend using when troubleshooting reachability issues:

- Check that BGP sessions are up with the command “show ip bgp summary”.
- Inspect the list of prefixes advertised in the session with the command "show ip bgp *ip-addr* advertised-routes”.
- Check the prefixes being received in the session with the command “show ip bgp *ip-addr* routes”. You should see 9.9.9.0/24 if Quad9 is enabled at your site.

Optimizing the use of PCH's DNS service

Once the edge router in your network is correctly configured and there is no BGP reachability issues, the next step is optimizing the use of our DNS resources.

First of all, let’s describe and illustrate a typical domain resolution process:

- User hosts send DNS requests to their local resolver (1)
- If the local resolver has the requested domain in cache it responds directly

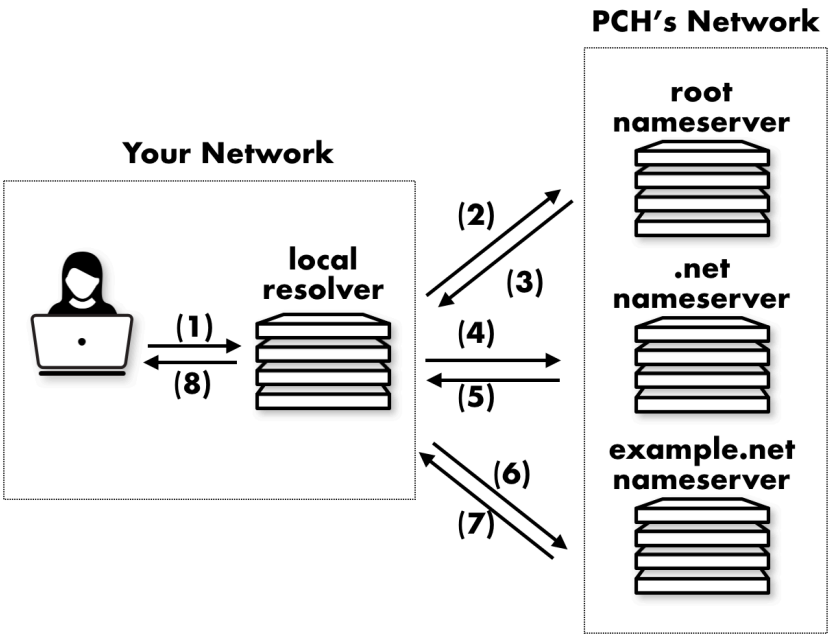

Figure 4: Diagram showing the interactions between a local resolver and PCH's network during a typical domain resolution process.

- If the local resolver is configured as a forwarding resolver, then it forwards the query to the recursive name servers (this could be Quad9; see later for a detailed guide on using Quad9 as the "forwarding resolver").
- Otherwise, the local resolver sends a (recursive/non-recursive) request to a root server, TLD, and then walks down DNS tree (2,3), (4,5), (6,7). Finally it replies to the host (8)
- If there are multiple name server records for a domain, the local resolver distributes requests across all of them
  - BIND tries every server from time to time ensuring that the best-known server is used to resolve queries[2].
  - Other name server implementations may use different algorithms but with the same goal of using the closest topological server.

The process outlined above might differ slightly depending on the design of your DNS infrastructure (local resolver only, forwarding only or a combination of both) and the number of root and authoritative name servers available in your country.

In some cases, DNS requests may be resolved locally via peering if authoritative servers are available at the exchange. This is often the case in major Internet hubs where networks peer directly with DNS operators and many domains across the DNS tree can be resolved locally. In other cases, however, requests are routed via the transit provider and can be resolved at a distance topologically, including overseas as is often the case in poorly interconnected regions.

The best-case scenario is when all DNS requests (in diagram steps 2–7) are resolved locally via PCH's name servers, starting by the root, then the top-level domain and subsequent domains. It is important to remember that, because networks have generally configured a local resolver

with high caching capacity, some of the benefits of using PCH's DNS service may not be directly perceived.

The worst-case scenario is when no root servers or authoritative servers are locally available in the country. In that case, steps 2–7 require sending requests overseas via your transit provider. A combination of local and external resolution is often the case but this entirely depends on the presence of certain DNS operators in your country. Imagine, for instance, .com domains registered by local companies. In that case, requests for .com domains (steps 4 and 5) are sent to the nearest authoritative server unless the .com operator is present in the country. The following resolution would mostly be local (steps 6 and 7) since most hosting companies also operate their own DNS service.

It is good practice for networks to research which DNS root server's name servers are already available in their country, either connected through the Internet exchange or hosted by other network operators, before attempting to troubleshoot PCH's service.[3] As a network service provider, you can also contact your local IXP operator and request an increase in the number of letters of the root zone available at the IXP.

Checking your closest DNS root server

An instance of D and E root name servers may be available at the DNS cluster in your site, depending on the policy of the root server operator. The following combinations of dig and traceroute commands can help locate the anycast instance of the root server answering your requests:

```
$ dig @e.root-servers.net hostname.bind txt ch +short
"p01.ktm.eroot"

$ dig @d.root-servers.net hostname.bind txt ch +short
"kmnp2.droot.maxgigapop.net"
```

In this example, for E-root the closest anycast instance is replying from "ktm" which is the IATA code[4] for Kathmandu, Nepal. For D-root an interpretation of the result is not straight forward so an additional traceroute command can verify if the name servers are local.

```
$ traceroute -a d.root-servers.net
traceroute to d.root-servers.net (199.7.91.13), 64 hops max, 52 byte packets
 1  [AS198949] 192.168.1.254 (192.168.1.254)  2.153 ms  1.333 ms  1.172 ms
 2  [AS136762] 116.66.192.101 (116.66.192.101)  6.986 ms  2.069 ms  3.077 ms
 3  [AS136762] ae1.datahub-core1.subisu.net.np (116.66.193.101)  3.690 ms  3.085 ms  4.749
ms
 4  [AS4007] xe-0-0-3.2102.iris.subisu.net.np (182.93.92.22)  2.832 ms  3.244 ms  2.894 ms
 5  [AS4007] xe-0-0-3.2102.mbaluwatar1.subisu.net.np (182.93.92.21)  4.913 ms  3.042 ms
3.154 ms
 6  [AS136762] xe-0-0-3-ggc-rtr.subisu.net.np (116.66.193.34)  2.993 ms  2.991 ms  2.995 ms
 7  [AS4557] 198-32-231-20.setg.net (198.32.231.20)  4.378 ms !X *  8.286 ms !X

$ traceroute6 d.root-servers.net
traceroute to d.root-servers.net (2001:500:2d::d) from 2404:2c00:2::145, 30 hops max, 24
byte packets
 1  gw-noc-pool1.nren.net.np (2404:2c00:2::1)  0.787 ms  0.45 ms  0.366 ms
 2  npix-pts.woodynet.net (2404:2c00:ffff:e::20)  10.746 ms  2.35 ms  0.716 ms
 3  d.root-servers.net (2001:500:2d::d)  0.887 ms !S  1.292 ms !S  0.669 ms !S
```

The low latency shown in the output as well as the router's FQDN confirms that the instance is hosted in Nepal.

Verifying the latency to PCH-hosted DNS resources

The latency to PCH-hosted DNS resources depends on the topological distance between the local DNS resolver and PCH's name servers and the caching capacity of the resolver. Using the dig command with DNS resources known to be hosted locally can be an effective method of determining if the local resolver is using PCH-hosted resources.

As an example, let's query for the A/AAAA records of www.pch.net and analyse the results. To make the request more unique, let's force the name server to send the DNSSEC signature with the +dnssec modifier:

```
$ dig +dnssec A www.pch.net

; <<>> DiG 9.10.6 <<>> +dnssec A www.pch.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22625
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.pch.net.                   IN      A

;; ANSWER SECTION:
www.pch.net.            280     IN      A       206.220.231.147
www.pch.net.            280     IN      RRSIG   A 8 3 300 20190921170000 20190904170000 10308
pch.net. sefE5ws4kjz/5jhpqHNauDwhmT4LxuOzEpeZ/TNmNIWrI5zfCvkMZ7Kl
5CdFpbKsce5i6gudZSB7fxHI7MtJjJJ0Abyflfd2X9EbZ1qq+QLZGQ4i
qdZyKXzOtgETwlNlluUlsGyFijawhfaknBSup+tG3awtThwV62xeMpjJ
K8Mij1xJsU7OREqepC5Sk1VUn6X7cXcIGH54oiXAJdtURl190uBytmlo
idCSEklVFAxV8Mo5HMujNhjXDXBwdlkQk8Wf4KPrpgDjE5hrkG5VLa9d
3l+kFZeNdv5juqJZAWtuzSJPDKBiA8IMJ6YUCK3CXFa+XhRJgyE9DRry kgRplA==

;; Query time: 3 msec
;; SERVER: 9.9.9.9#53(9.9.9.9)
;; WHEN: Mon Sep 09 12:00:13 +0545 2019
;; MSG SIZE  rcvd: 351
```

```
$ dig -6 +dnssec A www.pch.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> -6 +dnssec A www.pch.net
;; global options: +cmd
;; connection timed out; no servers could be reached
dibya@sandbox:~$ dig +dnssec AAAA www.pch.net

; <<>> DiG 9.10.3-P4-Ubuntu <<>> +dnssec AAAA www.pch.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62497
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.pch.net.                   IN      AAAA

;; ANSWER SECTION:
www.pch.net.            300     IN      AAAA    2620:0:872::231:147
www.pch.net.            300     IN      RRSIG   AAAA 8 3 300 20191128110000 20191111110000
10308 pch.net. rYvuv8/f712cyAZncPtXXu+QgtnY07eeCWLl/u7SJd78lR1hMYgx7Apa PVkqtu4bwLwH8/
zx2foFfWYoUbdvi5xOA0fc6JHzvB3az+nYUMVGODb3
CFPiULzo+sw46frvFElbZUzq+ITXIT34bjQxwcp0lbewR+Y3YGmRmRok
1xIwkMeECYvPi8LcTiru8qeGF43FotUFdmuIHptltEMgJNOwuVjU8LuK
cV2mVIGXJF2gp1kkb9gcT5fLUwbto9EC8hFQ8HFezlNBP7UwAN7J2gCJ
U7rTmX7VKMCjz7exPF98SyExzCannQTdfqDDA0DF+B07kUWN1JX4sDJE i8vhpQ==

;; Query time: 4 msec
;; SERVER: 9.9.9.9#53(9.9.9.9)
;; WHEN: Fri Nov 15 12:47:39 +0545 2019
;; MSG SIZE  rcvd: 363
```

The total query time is 3ms and 4ms for A and AAAA respectively, an excellent resolution time for a local instance. However, one must still determine if the short time is due to the cache or to the A/AAAA and RRSIG resources being hosted locally.

One method of checking whether the request was served or cached is by comparing the 3ms with the latency in reaching the nameserver that replied to us. We can use dig to ask the list of authoritative name servers for pch.net and observe the query response time:

```
$ dig NS pch.net

; <<>> DiG 9.10.6 <<>> NS pch.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30168
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;pch.net.                       IN      NS

;; ANSWER SECTION:
pch.net.                40182   IN      NS      ns2.pch.net.
pch.net.                40182   IN      NS      ns3.pch.net.
pch.net.                40182   IN      NS      anyns.pch.net.

;; Query time: 5 msec
;; SERVER: 9.9.9.9#53(9.9.9.9)
;; WHEN: Mon Sep 09 12:00:35 +0545 2019
;; MSG SIZE  rcvd: 92
```

In this case, a latency of 5ms could suggest that the name server that answered this query is "as close as" the one that served the previous request. At this stage, note that we do not know which of the name server replied to our query (it could be anyns.pch.net, ns2.pch.net or ns3.pch.net). To find out, we can use the option +nssearch to compare the latency between the different name servers for pch.net. If there is a name server hosted locally, there should be at least one entry at a similar latency.

```
$ dig +nssearch pch.net. | awk -F ' ' ' {print $10, $11, $12, $13, $14}'

server 204.61.216.4 in 3 ms.
server 206.220.231.3 in 278 ms.
server 204.42.254.5 in 306 ms.

$ dig -x 204.61.216.4 +short
anyns.pch.net.

$ dig -6 +nssearch pch.net. | awk -F ' ' ' {print $10, $11, $12, $13, $14}'

server 2001:500:14:6004:ad::1 in 2 ms.
server 2620:0:872::231:3 in 257 ms.
server 2001:418:3f4::5 in 293 ms.
```

The result of the dig command shows that the name server with IP address 204.61.216.4 (anyns.pch.net) is at 3ms. A simple traceroute to 204.61.216.4 shows that the server is accessed locally to our network via the Internet exchange.

```
$ traceroute 204.61.216.4
traceroute to 204.61.216.4 (204.61.216.4), 64 hops max, 52 byte packets
 1  192.168.17.1 (192.168.17.1)  1.618 ms  1.501 ms  1.997 ms
 2  202.51.95.1 (202.51.95.1)  2.614 ms  3.780 ms  2.444 ms
 3  202.51.77.142 (202.51.77.142)  2.244 ms  2.104 ms  2.096 ms
 4  * as42.ktm.pch.net (198.32.231.20)  6.828 ms !X  2.897 ms !X

$ traceroute 2001:500:14:6004:ad::1
traceroute to 2001:500:14:6004:ad::1 (2001:500:14:6004:ad::1), 30 hops max, 80 byte packets
 1  gw-noc-pool1.nren.net.np (2404:2c00:2::1)  0.986 ms  0.929 ms  0.947 ms
 2  npix-pts.woodynet.net (2404:2c00:ffff:e::20)  3.397 ms  3.443 ms  3.330 ms
 3  anyns.pch.net (2001:500:14:6004:ad::1)  2.117 ms  1.485 ms *
```

The Quad9 recursive resolver project

Quad9 is a free, open, recursive DNS anycast resolver, primarily hosted on the PCH network infrastructure. It provides end users and organizations a DNS-based filtering service that blocks criminal sites such as malware distribution locations, phishing destinations, botnet command and control, and other hosts harmful to end users and network elements. Quad9 is an independent non-profit organization sponsored by PCH and other industry participants with the goal of making the Internet safer, more private, and more efficient. Quad9 services are entirely free of charge, though the project does appreciate sponsorship donations.

While providing security, Quad9 is dedicated to end user privacy and therefore neither stores nor transmits personally identifiable information (PII). It has been designed with GDPR

compliance as a baseline goal. See our privacy and data collection statements for more detail: https://www.quad9.net/privacy/ and https://www.quad9.net/policy/

The list of blocked domains is created/selected from a set of nineteen (as of Quarter 3, 2019) threat intelligence (TI) providers, in both the commercial and open community. Quad9 works in cooperation with these intelligence providers to feed back the effectiveness of blocked domains for continual tuning and improvement of the security protection process.

Quad9 can be used to provide no-cost basic cyber-security to ISP network end users as an additional value. Along with end user protections, Quad9 provides protections against network abuse, defending ISP infrastructure by blocking botnet control systems and user account fraud. Quad9 also reduces risk and management costs by offloading the most critical security and load-based issues involving recursive resolvers away from the core of an ISP network.

Quad9 offers an option for ISPs to acquire on-premise equipment for higher performance, which does have an installation and upkeep charge, but most organizations are able to use existing instances if they have good latency to a Quad9-served exchange. If you are interested in the on-premises option, get in touch with Quad9 at support@quad9.net.

Quad9 has several different service "flavors" — parameters other than the default set 9.9.9.9/149.112.112.112/2620:fe::fe. Choosing different sets of service IP addresses enables different service models, with advantages and disadvantages that an ISP must consider before use. See Appendix C for a list of these different variants. Notable is the ECS model, described below.

Because centralized services like Quad9 present an IP address outside the ASN of the ISP, a different model was developed called "ECS" (https://tools.ietf.org/html/rfc7871). This method includes some portion of the client IP address in the DNS query (in IPv4: 24 bits, and in IPv6: 56 bits) so that CDNs can attempt to provide content origins to clients more accurately. Quad9 supports ECS on the 9.9.9.11/149.112.112.11/2620:fe::11 service address set. Note that ECS responses may be slower than other service sets, as caching ability is reduced because of the additional data dimensions. ISPs that have installed CDN deployments within their own networks may find the ECS "flavor" of Quad9's service useful. Some CDNs use the origin IP address of the DNS server to direct clients to the most appropriate content.

Understanding Quad9 deployment strategies

Quad9 can be used in two primary design models: as a "forwarding resolver" to which your existing DNS caching systems can be directed, or in "end-user direct queries" mode where your DHCP or other tools would hand out the Quad9 resolver addresses directly to clients. Both methods are outlined below.

Configuring your network to use Quad9 as "forwarding resolver"

The preferred method of implementing Quad9 in your network is simply to point any existing DNS infrastructure to Quad9. This approach provides the best latency (the servers directly

answering the queries are “inside” your network) while protecting your end users against malware and botnets. This method additionally provides privacy for end users, as no IP addresses for end users are then relayed outside the originating network.

A forwarding DNS server offers the same advantage of maintaining a cache to improve DNS resolution times for clients without doing any of the recursive querying itself. Instead, the local forwarding resolver sends all requests to an outside resolving server (Quad9) and then caches the results to use for later queries (steps 1a and 8a in the diagram below).

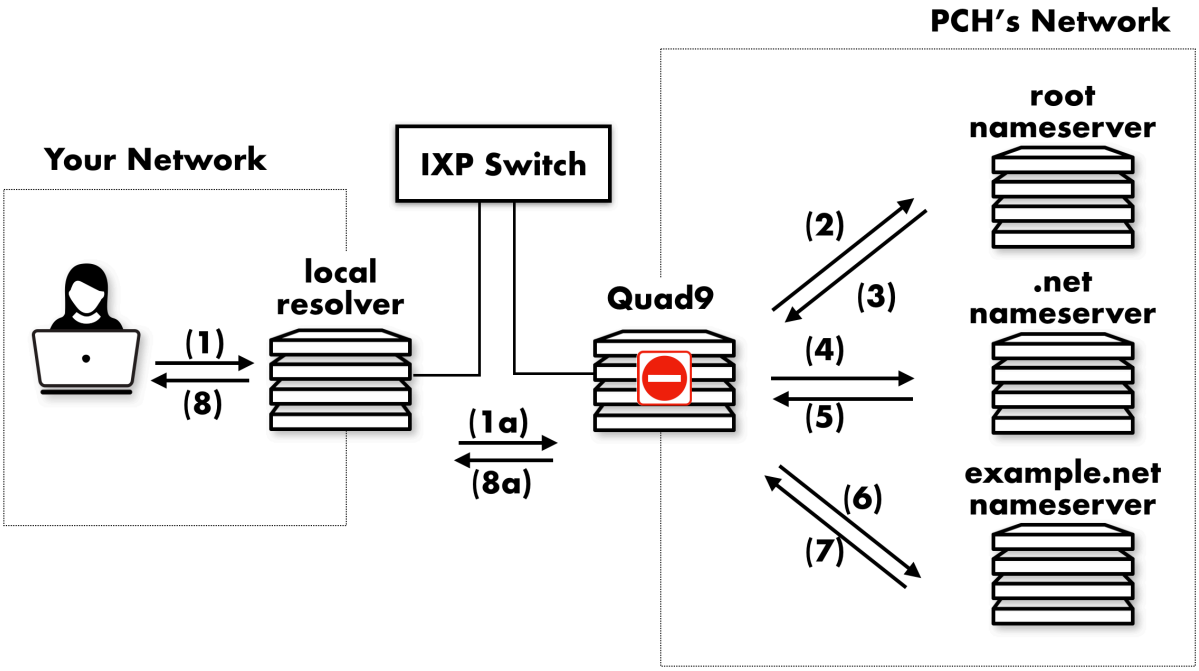

Figure 5: Diagram showing the interactions between a local resolver and Quad9 configured as forwarding resolver

This lets the forwarding server respond from its cache for repeated queries from clients, while not requiring all of the work of recursive queries. Thus the server makes only single requests (the forwarded client request, steps 1 and 8) instead of having to go through the entire recursion routine. This may be an advantage in environments where external bandwidth transfer is costly, where caching servers may need to be changed often, or when you wish to forward local queries to one server and external queries to another server.

Quad9 can be used as a forwarding resolver by a local cache resolver if one already exists. Platforms such as BIND, PowerDNS recursor, Unbound, Knot, dnsmasq, F-5, and most Microsoft products interoperate with Quad9 as “out of the box” caching solutions. For example, if you are running ISC BIND in your network, the following lines of configuration will setup your current server as a “forwarding cache” using Quad9 systems as the forwarding resolver:

- Step 1: Open your named.conf.options file and look for the options {…} block.
- Step 2: Create a block inside called forwarders that contains the IP addresses of the recursive name servers you want to forward requests to. In this case, we use Quad9’s secure DNS

service (9.9.9.9 & 149.112.112.112). If your network is IPv6 enabled, then add 2620:fe::fe in addition to the IPv4 addresses.

- Step 3: Set the forward directive to “only” so that this server forwards all requests and does not attempt to resolve requests on its own.

The configuration file should look similar to this one after you create the block and add the directive:

```
acl allowedhosts {
        192.0.2.0/24; # based on your own settings YMMV
        localhost;
        localnets;
};

options {
        directory "/var/cache/bind";

        recursion yes;
        allow-query { allowedhosts; };

        forwarders {
                9.9.9.9;
                149.112.112.112;
        };
        forward only;

        dnssec-validation auto;

        auth-nxdomain no;    # conform to RFC1035
        listen-on-v6 { any; };
};
```

Some recursive resolver packages now support DNS-over-TLS as an option to encrypt transactions sent to upstream forwarders like Quad9; this is a rapidly changing area, check with your software vendor for more details[5].

Checking if Quad9 is being used as forwarder resolver

The final step would be testing that the configuration is successful and that your local resolver is forwarding queries to Quad9. Several synthetic queries, implemented by Quad9, can be used to debug this scenario.[6]

In this case, a query of “whatismyip.on.quad9.net" returns the IP address as seen by the recursive resolver or an empty string if we are not querying Quad9 directly:

```
~ dig @172.16.0.1 +short A whatismyip.on.quad9.net
69.166.14.2

~ dig @9.9.9.9 +short A whatismyip.on.quad9.net
69.166.14.2
```

In the example above, both the local resolver in 172.16.0.1 and Quad9 reply with the same client IP address (69.166.14.2), confirming that Quad9 is configured as a forwarding resolver. An empty string on the first query would mean that Quad9 is not being used.

An additional synthetic query can be performed to determine which Quad9 node is replying to your queries. In this case, you can query "id.server.on.quad9.net" using the following command:

```
~ dig @9.9.9.9 +short TXT id.server.on.quad9.net
res200.pao.on.quad9.net.
```

The reply contains the IATA code of the instance where your query is being answered. In this case, "pao" represents Palo Alto in California, United States.

Setting up Quad9 on end-user client machines

End-users can also setup their laptops and mobile phones to use the Quad9 recursive resolver. Instructions for different operating systems are provided below:

Linux

Setup in Linux operating systems is done by editing the /etc/resolv.conf file to point your machine to the name server you want to use. In this example, we add the service addresses 9.9.9.9 and 149.112.112.112.

- Step 1: Open the file with super user privileges in your text editor.
- Step 2: The file lists the DNS to use to resolve queries by setting the name server directives. You can comment out other name server directives and add our own:

```
nameserver 9.9.9.9
nameserver 149.112.112.112
# nameserver 209.244.0.3
# nameserver 192.0.2.1
```

- Step 3: Save and close the file
- Step 4: You can check which server you are querying with the tool dig. For an example, we try resolving a domain, for instance quad9.net:

```
~ dig quad9.net

; <<>> DiG 9.10.6 <<>> quad9.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19542
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;quad9.net.                     IN      A

;; ANSWER SECTION:
quad9.net.              582     IN      A       216.21.3.77

;; Query time: 17 msec
;; SERVER: 9.9.9.9#53(9.9.9.9)
;; WHEN: Fri Aug 30 12:49:33 IST 2019
;; MSG SIZE  rcvd: 54
```

Apple

Step-by-step instructions on how to setup Quad9 in your Apple machines are available online at https://quad9.net/apple/

Microsoft

Step-by-step instructions on how to setup Quad9 in machines with Windows operating systems are available online at https://quad9.net/microsoft/

Google Play

Step-by-step instructions on how to setup Quad9 in your Android mobile phone are available online at https://quad9.net/quad9-connect-on-google-play/

Adding encryption with DNS over TLS, DNS over HTTPS and DNSCrypt

Quad9 supports stub-to-recursive encryption with DNS-over-TLS (DoT), DNS-over-HTTPS (DoH), and DNSCrypt. If your cache or client supports these options, they can be enabled to prevent interception or re-write of queries from Quad9 systems.

You can find instructions on the use of DNS over TLS at https://www.quad9.net/private-dns-quad9-android9/; the use of DNS over HTTPS at https://quad9.net/doh-quad9-dns-servers/; and the use of DNSCrypt at https://www.quad9.net/privacy-dnscrypt-testing/.

Appendix A: List of ccTLDs that use PCH anycast platform

AF	Afghanistan	af	EC	Ecuador	ec
DZ	Algeria	dz	EG	Egypt	مصر
DZ	Algeria	الجزائر	EE	Estonia	ee
AO	Angola	ao	FI	Finland	fi
AG	Antigua and Barbuda	ag	FR	France	fr
AR	Argentina	ar	GE	Georgia	გე
AM	Armenia	am	GL	Greenland	gl
AM	Armenia	հայ	GT	Guatemala	gt
AU	Australia	au	GW	Guinea-Bissau	gw
BS	Bahamas	bs	GY	Guyana	gy
BD	Bangladesh	bd	HT	Haiti	ht
BD	Bangladesh	বাংলা	HN	Honduras	hn
BZ	Belize	bz	IS	Iceland	is
BJ	Benin	bj	IQ	Iraq	iq
BT	Bhutan	bt	IQ	Iraq	عراق
BO	Bolivia	bo	KE	Kenya	ke
BA	Bosnia and Herzegovina	ba	KI	Kiribati	ki
BW	Botswana	bw	KW	Kuwait	kw
BN	Brunei Darussalam	bn	LS	Lesotho	ls
BG	Bulgaria	бг	LY	Libya	ly
BI	Burundi	bi	LU	Luxembourg	lu
CA	Canada	ca	MG	Madagascar	mg
CV	Cape Verde	cv	MW	Malawi	mw
CL	Chile	cl	MV	Maldives	mv
CR	Costa Rica	cr	MT	Malta	mt
CI	Cote D'Ivoire	ci	MU	Mauritius	mu
HR	Croatia	hr	MX	Mexico	mx
CY	Cyprus	cy	MN	Mongolia	mn
DK	Denmark	dk	ME	Montenegro	me
DO	Dominican Republic	do	MZ	Mozambique	mz

NA	Namibia	na	LK	Sri Lanka	lk
NP	Nepal	np	LK	Sri Lanka	ලංකා
NC	New Caledonia	nc	LK	Sri Lanka	இலங்கை
NG	Nigeria	ng	SD	Sudan	sd
PS	Palestine	ps	SD	Sudan	سودان
PY	Paraguay	py	CH	Switzerland	ch
PE	Peru	pe	SY	Syria	sy
PH	Philippines	ph	SY	Syria	سورية
PT	Portugal	pt	TW	Taiwan	tw
PR	Puerto Rico	pr	TW	Taiwan	台湾
QA	Qatar	qa	TW	Taiwan	台灣
QA	Qatar	قطر	TZ	Tanzania	tz
RW	Rwanda	rw	TH	Thailand	th
KN	Saint Kitts and Nevis	kn	TH	Thailand	ไทย
VC	Saint Vincent and the Grenadines	vc	TL	Timor-Leste	tl
SM	San Marino	sm	TT	Trinidad and Tobago	tt
SA	Saudi Arabia	sa	TN	Tunisia	tn
SA	Saudi Arabia	السعودية	TN	Tunisia	تونس
RS	Serbia	rs	TR	Turkey	tr
RS	Serbia	срб	UG	Uganda	ug
SC	Seychelles	sc	UA	Ukraine	ua
SG	Singapore	sg	AE	United Arab Emirates	ae
SG	Singapore	சிங்கப்பூர்	AE	United Arab Emirates	امارات
SG	Singapore	新加坡	VN	Viet Nam	vn
SI	Slovenia	si	ZM	Zambia	zm
SB	Solomon Islands	sb	ZW	Zimbabwe	zw
SO	Somalia	so			
ZA	South Africa	za			
ES	Spain	es			

Appendix B: Reciprocal Peering Requirements for AS42, AS3856, and AS715

- The destination address of any traffic you send to us must be contained within a route that we currently BGP advertise to you.

- The source address of any traffic you send to us must be contained within a route you currently BGP advertise to us.

- You agree to augment equipment and circuits as necessary to maintain uncongested interconnection paths.

- You agree to maintain a 24x7 network operations centre (NOC) with standard technical, administrative, legal, and abuse Points of Contact.

- You agree to respond promptly to communications and work cooperatively to resolve security, stability, and abuse incidents in a timely fashion.

- You agree to BGP advertise only routes to prefixes delegated to you and your customers by a regional Internet registry.

- You agree to not engage in practices generally deemed abusive or fraudulent, such as pointing default, rewriting next-hop, or failing to implement BCP-38 filtering.

- In addition, although not mandatory, we prefer that peers that are large enough to meet us in multiple regions do so in at least two locations within each region and also maintain consistent route advertisements and origin autonomous systems within each region.

Appendix C: Router commands for v6 prefixes

Below the router commands and outputs showing the v6 prefixes advertised by our routers.

```
## Advertised prefixes from AS42

router.ktm#sh bgp ipv6 uni neighbors 2404:2c00:ffff:e::18 advertised-routes
BGP table version is 519, local router ID is 204.61.210.46
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
 *>i 2001:500:14::/48 2620:171:24::12          0    100      0 i
 *>  2001:500:2D::/48 2620:171:24::56          0             0 10886 i
 *>i 2001:500:48::/48 2620:171:24::232         0    100      0 i
 *>i 2001:500:49::/48 2620:171:24::232         0    100      0 i
 *>i 2001:500:4A::/48 2620:171:24::232         0    100      0 i
 *>i 2001:500:4B::/48 2620:171:24::232         0    100      0 i
 *>i 2001:500:4C::/48 2620:171:24::232         0    100      0 i
 *>i 2001:500:4D::/48 2620:171:24::232         0    100      0 i
 *>i 2001:500:4E::/48 2620:171:24::232         0    100      0 i
 *>i 2001:500:4F::/48 2620:171:24::232         0    100      0 i
 *>i 2001:500:7D::/48 2620:171:24::12          0    100      0 i
 *>i 2001:500:83::/48 2620:171:24::248         0    100      0 i
 *>i 2001:500:A5::/48 2620:171:24::232         0    100      0 i
 *>  2001:500:A8::/48 2620:171:24::80          0             0 21556 i
 *>i 2001:500:E1::/48 2620:171:24::232         0    100      0 i
 *>i 2001:678:3::/48  2620:171:24::16          0    100      0 i
 *>i 2001:678:28::/48 2620:171:24::12          0    100      0 i
 *>i 2001:678:4C::/48 2620:171:24::16          0    100      0 i
 *>i 2001:678:60::/48 2620:171:24::16          0    100      0 i
 *>i 2001:678:78::/48 2620:171:24::16          0    100      0 i
 *>i 2001:678:94::/48 2620:171:24::12          0    100      0 i
 *>  2620:FE::/48     2620:171:24::242         0             0 19281 i
 *>  2620:171:24::/48 ::                       0         32768 i
 *>i 2620:171:800::/48 2620:171:24::12          0    100      0 i
 *>i 2620:171:801::/48 2620:171:24::16          0    100      0 i
 *>i 2620:171:804::/48 2620:171:24::16          0    100      0 i
 *>i 2620:171:805::/48 2620:171:24::16          0    100      0 i
 *>i 2620:171:806::/48 2620:171:24::12          0    100      0 i
 *>i 2620:171:807::/48 2620:171:24::12          0    100      0 i
 *>i 2620:171:808::/48 2620:171:24::12          0    100      0 i
 *>i 2620:171:A00::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A01::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A02::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A03::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A04::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A05::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A06::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A07::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A08::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A09::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A0A::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A0B::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A0C::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A0D::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A0E::/48 2620:171:24::20          0    100      0 i
 *>i 2620:171:A0F::/48 2620:171:24::20          0    100      0 i
 *>i 2800:110:10::/48 2620:171:24::12           0    100      0 i
 *>i 2801:140:10::/48 2620:171:24::12           0    100      0 i
 *>i 2A01:8840:5::/48 2620:171:24::2            0    100      0 i
 *>i 2A01:8840:15::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:19::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:1D::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:21::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:25::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:29::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:2D::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:31::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:35::/48 2620:171:24::20          0    100      0 i
```

```
 *>i 2A01:8840:39::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:3D::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:41::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:45::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:4D::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:51::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:55::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:59::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:5D::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:61::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:65::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:69::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:6D::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:71::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:75::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:79::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:7D::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:81::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:85::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:89::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:8D::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:91::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:95::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:99::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:9D::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:A1::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:A5::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:A9::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:AD::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:B1::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:B5::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:B9::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:BD::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:C1::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:C5::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:C9::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:CD::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:D1::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:D5::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:D9::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:DD::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:E1::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:E5::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:E9::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:ED::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:F1::/48 2620:171:24::20          0    100      0 i
 *>i 2A01:8840:1C1::/48 2620:171:24::20          0    100      0 i

Total number of prefixes 105


## Advertised v6 prefix from AS3856

route-collector.ktm.pch.net# sh ipv6 bgp neighbors  2404:2c00:ffff:e::18 advertised-routes
BGP table version is 0, local router ID is 74.80.106.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 2620:0:870::/48  2404:2c00:ffff:e::21
                                         0         32768 i
```

While we attempt to advertise the same prefixes in all locations, the specific number of prefixes advertised might be different from the list above.

Appendix D: Quad9 System Service Variants

Below are outlined the service addresses and offerings for each of the variants of the Quad9 system.

	Blocklist	DNSSEC validation	EDNS Client Subnet	Service Addresses (*)(**)
Quad9 secure	Yes	Yes	No	IPv4: 9.9.9.9 & 149.112.112.112 IPv6: 2620:fe::fe & 2620:fe::9
Quad9 non-secure	No	No	No	IPv4: 9.9.9.10 & 149.112.112.10 IPv6: 2620:fe::10 & 2620:fe::fe:10
Quad9 secure + ECS	Yes	Yes	Yes	IPv4: 9.9.9.11 & 149.112.112.11 IPv6: 2620:fe::11 & 2620:fe::fe:11

(*) All service addresses offer the following encryption options:

- DNS-over-TLS (DoT) supported on port 853
- DNS-over-HTTPS (DoH) supported on port 443
- DNSCrypt supported on port 8443

(**) Use only one of these sets of addresses—secure or unsecured. Mixing secure and unsecured IP addresses in your configuration may lead to your system being exposed without the security enhancements, or your private data may not be fully protected. For consistent results, do not specify secure Quad9 addresses in combination with any other open recursive resolver because this may result in your systems being unprotected for 50% of your queries.

Secure addresses offer:

- Domain blocklist from nineteen different malware/threat providers
- DNSSEC validation on lookups

EDNS Client Subnet (ECS) transmission service addresses offer:

- First 24 bits of client subnet (IPv4) transmitted to authoritative servers
- First 56 bits of client subnet (IPv6) transmitted to authoritative servers

Non-secure addresses offer:

- No DNSSEC validation and no blocklist, which might be useful for testing validation

Notes

[1] An updated list of our peering locations is available at https://www.pch.net/about/peering.

[2] This behaviour can seem inefficient as we are querying servers with high RTTs instead of using only the ones closest to us. However, it also ensures that changes in the topology of the DNS authoritative servers, often happening transparently to network operators, do not have a negative impact on the overall performance of domain resolution.

[3] A helpful resource listing global and local locations of root server operators is https://www.root-servers.org.

[4] IATA codes are abbreviations that the International Air Transport Association publishes to facilitate air travel. They are typically one, two, three, or four character combinations that uniquely identify locations, equipment, companies, and times to standardize international flight operations. The Internet industry has borrowed them to identify the location of network equipment by using them in the fully qualified domain name. As an example, "ae-5.r02.amstnl02.nl.bb.gin.ntt.net" contains the IATA code ‘ams’ which stands for Amsterdam.

[5] Configuration instructions to implement DNS over TLS using BIND 9 and stunnel are available at https://kb.isc.org/docs/aa-01386

[6] Synthetic queries are processed by external programs and do not generate a query to the public DNS system.