DNS Anycast with Cisco Prime Network Registrar

This chapter provides the knowledge and tools to configure Cisco Prime Network Registrar DNS services using with Anycast.

DNS Anycast with Cisco Prime Network Registrar

Anycast is a network and routing mechanism that enables a packet from a single client to go to one of many servers offering the same service. All the servers in the Anycast group are configured with the same Anycast IP address and the packet is routed from the client to the closest server by the best path as determined by routing algorithms. Anycast routing enables several important capabilities such as seamless redundancy, load balancing, and horizontal scaling by grouping multiple servers as a single service. Anycast DNS is simply an implementation of Anycast for DNS services. Anycast is used in conjunction with a routing protocol such as BGP (Border Gateway Protocol) to advertise the availability of the service to an adjoining router, which makes the Anycast DNS to work effectively.

Basic Requirements for DNS Anycast

The following is a list of requirements and recommendations for supporting Anycast DNS:

- Clients should be configured to resolve DNS queries via the caching server's (CDNS) Anycast address(es),
- Nameservers should advertise their Anycast address in NS and A RR’s,
- Nameservers should listen to DNS queries on the Anycast IP addresses,
- Nameservers should be configured with at least one Anycast IP address on a loopback interface,
- Additionally, the server should be configured with a management IP, which can be either a physical or an additional loopback interface,
- At least one physical IP must be defined on the DNS server for the exchange of routing information, as well as, system access and maintenance in the absence of the routes to the Anycast IP address(es),
- Nameservers should be configured to use the physical or management IP addresses for zone-transfers, zone updates, and/or query-source to ensure that these updates go to the intended server,

- Nameservers should Inject Anycast IP address(es) into the routed network using routing protocols such as RIP, OSPF, or BGP.

Anycast Routing with Quagga

Anycast can be manually configured, it is best implemented using routing protocols such as BGP or OSPF which announce the Anycast destination address to its gateway router. Using a routing protocol to announce availability of the DNS service ensures that routers don’t send DNS queries into a black hole if the service goes down. Because the CPNR DNS application does not have routing capabilities, some code, external to the DNS application must be added to the DNS environment (physical server or virtual machine). One prominent and open source product is Quagga.

Quagga

Quagga is a routing software suite, providing implementations of OSPFv2, OSPFv3, RIP v1 and v2, RIPng and BGP-4 for Unix platforms, Linux, Solaris and NetBSD. The solution described in this chapter uses BPG.

The Quagga architecture consists of a core daemon, zebra, which acts as an abstraction layer to the underlying Linux kernel and presents the Zserv API over a Unix or TCP stream to Quagga clients. It is these Zserv clients, which typically implement a routing protocol and communicate routing updates to the zebra daemon.

Quagga daemons are configurable via a network accessible CLI (called a 'vty'). The CLI follows a style similar to that of other routing software. There is an additional tool included with Quagga called 'vtysh', which acts as a single cohesive front-end to all the daemons, allowing one to administer nearly all aspects of the various Quagga daemons in one place.

Quagga does not ship with the CPNR 9.0 release. However, in a terminal session, administrators can YUM to install the Quagga components.

The Quagga, daemon can be found at http://www.nongnu.org/quagga/

Script

A sample python script, included in this chapter, starts and stops Quagga and monitors the DNS service to ensure it is operational. When Quagga is started, its BGP daemon sends an Anycast advertisement to the connected router making the DNS service available over the Anycast address. In the event the script detects the service to be unavailable, not responding to queries, the script will stop the Quagga daemon. Stopping Quagga breaks the tcp connection and the router will stop receiving BGP keep-alive messages. The router will then remove the dns service from its Anycast group and start sending dns queries to the next closest and available dns service.

The script uses Linux commands to start and stop the Quagga BGP daemon. The command used is; **/etc/init.d/bgpd start/stop/restart**. This is the standard way of starting/stopping services on Linux.

Cron Job

Along with the script, a cron job would be a good practice that monitors and restarts the script in the event the script stops working. An example of a cron job is outside the scope of this solution.

Router Configuration

Your configuration will probably be different based on your network requirements and variations in addressing schemes.

Sample Anycast Configuration using BGP

In this section we will describe the basic setup and configuration of Anycast using BGP on a Cisco router and Quagga host-based routing software. The purpose of this chapter is not to instruct administrators on the configuration of routers and BGP but instead to show a configuration that was successfully tested in the CPNR labs. Your network requirements may be different.

Border Gateway Protocol (BGP) is a standardized exterior gateway protocol designed to exchange routing and reachability information among Autonomous Systems (AS) on the Internet. This configuration uses a single AS this recipe is not intended to be solution deployed across autonomous systems.

The following steps need to be performed on the hosts DNS-1 and DNS-2:

Install Quagga Routing Software

This will install Quagga package like the following: **Quagga-0.99.15-7.el6_3.2.x86_64**

YUM install Quagga.

Create a Loopback Interface

Create a loopback interface alias on the system. Configure the anycast IP address on this loopback interface.

On RHEL, the interface configuration files are located at **/etc/sysconfig/network-scripts**. Create a file in that directory named ifcfg-lo:0 with the following contents:

DEVICE=lo:0

IPADDR=10.10.10.1

NETMASK=255.255.255.255

BOOTPROTO=none

ONBOOT=yes

Bring up the new loopback interface using the command:

ifup lo:0

Network Router Configuration

This router configuration used in the validation of this DNS Anycast solution. It is provided as a reference to assist in the development of DNS Anycast solution. While it is a complete configuration for this specific solution, it is only intended to be a reference for developing your solution.

csr1000v#sh run

Building configuration...

!

interface Loopback0

DNS Anycast with Cisco Prime Network Registrar

```
ip address 2.2.2.2 255.255.255.255
!
interface GigabitEthernet1
ip address 10.78.29.77 255.255.255.0 (Router)
negotiation auto
!
interface GigabitEthernet2
ip address 10.0.2.1 255.255.255.0 (Client)
negotiation auto
!
interface GigabitEthernet4 (DNS-2)
platform ring rx 256
ip address 10.0.3.1 255.255.255.0
negotiation auto
!
interface GigabitEthernet5 (DNS-3)
platform ring rx 256
ip address 10.0.5.1 255.255.255.0
negotiation auto
!
router ospf 1
router-id 2.2.2.2(is the loopback IP address)
redistribute bgp 65500 subnets
network 2.2.2.2 0.0.0.0 area 1
network 10.0.6.0 0.0.0.255 area 1
network 10.0.0.0 0.0.255.255 area 1
!
router bgp 65500
bgp log-neighbor-changes
neighbor IBGP peer-group
neighbor IBGP update-source Loopback0
neighbor ANY peer-group
neighbor 1.1.1.1 remote-as 65500
neighbor 1.1.1.1 peer-group IBGP
```

```
neighbor 1.1.1.1 update-source Loopback0
neighbor 10.0.3.2 remote-as65500
!(This should be the bgp AS in Quagga for DNS-2)
neighbor 10.0.3.2 peer-group ANY
neighbor 10.0.5.2 remote-as 65500
!(This should be the bgp AS in Quagga for DNS-3)
neighbor 10.0.5.2 peer-group ANY
!
address-family ipv4
redistribute ospf 1
neighbor IBGP next-hop-self
neighbor ANY next-hop-self
neighbor 1.1.1.1 activate
neighbor 10.0.3.2 activate
neighbor 10.0.5.2 activate
exit-address-family
!
virtual-service csr_mgmt
ip shared host-interface GigabitEthernet1
activate
!
ip default-gateway 10.78.28.1
ip forward-protocol nd
!
no ip http server
ip http secure-server
ip route 0.0.0.0 0.0.0.0 10.78.28.1
ip route 10.78.28.0 255.255.254.0 GigabitEthernet1 10.78.28.1
!
ip prefix-list anycast-ip seq 5 permit 10.10.10.1/32
!
control-plane
!
line con 0
```

```
stopbits 1
line vty 0 4
login local
!
!
end
```

Configure Quagga on DNS servers

Configure the Quagga configuration files on both the servers. The following example is for DNS-1. DNS-2 also needs to be configured similarly. The configuration files are located in **/etc/Quagga**.

There are a number of example configuration files in **/etc/Quagga**: one for each routing protocol that Quagga supports; one for zebra, the main process. For enabling anycast using BGP, we need to configure **zebra.conf** and **bgpd.conf**.

Quagga Zebra Configuration

```
# cat /etc/quagga/zebra.conf
hostname DNS-1
!
password zebra
enable password zebra
!
interface eth0
ip address 10.0.3.2/24
!
interface lo
!
line vty
!
```

Note Repeat the steps for any other Anycast servers that are part of the group.

Quagga BGP Configuration

```
#cat /etc/quagga/bgpd.conf
! -*- bgp -*-
!
```

```
! BGPd sample configuration file
!
!
hostname DNS-1
password zebra
log stdout
!
router bgp 65500
bgp router-id 10.78.29.79
bgp log-neighbor-changes
network 10.10.10.1/32
timers bgp 4 16
neighbor 10.0.3.1 remote-as 65500
neighbor 10.0.3.1 next-hop-self
neighbor 10.0.3.1 prefix-list DEFAULT in
neighbor 10.0.3.1 prefix-list ANYCAST out
!
address-family ipv4
network 10.0.3.1/24
neighbor 10.0.3.1 activate
exit-address-family
!
ip prefix-list ANYCAST seq 5 permit 10.10.10.1/32
ip prefix-list DEFAULT seq 5 permit 0.0.0.0/0
line vty
!
```

Start BGP daemon

Start the BGP daemon using the following command:

/etc/init.d/bgpd start

Run Diagnostics on Router

Run diagnostics on the router to make sure that the Anycast is setup properly.

The #sh ip bgp summary command output shows that router-1 has opened a BGP session with the two neighbors. The value "State/PfxRcd" indicates that the TCP session is up and the routers and hosts are

exchanging routes. This field should be a numeric value showing how many route prefixes have been received from the remote neighbor. The example value is 1. At this point the BGP connection of the with the DNS servers is in Established state.

The #sh ip bgp summary

BGP router identifier 2.2.2.2, local AS number 65500

BGP table version is 86, main routing table version 86

1 network entries using 248 bytes of memory

2 path entries using 240 bytes of memory

1/1 BGP path/bestpath attribute entries using 248 bytes of memory

0 BGP route-map cache entries using 0 bytes of memory

0 BGP filter-list cache entries using 0 bytes of memory

BGP using 736 total bytes of memory

BGP activity 16/15 prefixes, 61/59 paths, scan interval 60 secs

Neighbor	V	AS	MsgRcvd	MsgSent	TblVer	InQ	OutQ	Up/Down	State/PfxRcd
1.1.1.1	4	65500	0	0	1	0	0	4w0d	Idle
10.0.3.2	4	65500	137919	129519	86	0	0	1w0d	1
10.0.5.2	4	65500	137923	129519	86	0	0	1w0d	1

The command **#show ip bgp neighbors** shows information about the neighbors in detail.

The command **#show ip route** should have an entry for the Anycast address and the host via which it is currently routed.

#sh ip route

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP

D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area

N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2

E1 - OSPF external type 1, E2 - OSPF external type 2

i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2

ia - IS-IS inter area, * - candidate default, U - per-user static route

o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP

a - application route

+ - replicated route, % - next hop override, p - overrides from PfR

B 10.10.10.1/32 [200/0] via 10.0.3.2, 00:00:10

Monitor BGP Traffic Logs

To monitor the BGP traffic logs on the hosts DNS-1 and DNS-2 use the command **# telnet localhost bgpd**.

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

Hello, this is Quagga (version 0.99.15).

Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:

DNS-1> enable

DNS-1# terminal monitor

DNS-1# 2016/07/13 15:49:20 BGP: 10.0.5.1 send message type 4, length (incl. header) 19

2016/07/13 15:49:21 BGP: 10.0.5.1 rcv message type 4, length (excl. header) 0

2016/07/13 15:49:25 BGP: 10.0.5.1 send message type 4, length (incl. header) 19

2016/07/13 15:49:27 BGP: 10.0.5.1 rcv message type 4, length (excl. header) 0

Managing the Quagga BGP daemon

The following script will detect the DNS server status and shut down the Quagga BGP daemon in the event the DNS server is not responsive. Taking down the BGP Daemon will result in the TCP session going down and ultimately in the router taking this DNS server out of the Anycast list.

A sample script is included in release 9.1 and It is located at :

/opt/nwreg2/local/examples/dns/dns_anycast_bgp.py

A cron job will need to be created to periodically run the script to check the status of the DNS server and start or stop the BGP daemon accordingly.

Sample Script

```
#
# Script to turn on BGP, detect if the DNS server is down, and shut down the quagga # BGP daemon
#
import dns.resolver
import os
import sys
import subprocess
if __name__ == "__main__":
run = True
if len(sys.argv) < 2:
print 'python %s <server ip> [retries]' % sys.argv[0]
print 'e.g. /usr/bin/python DNSQuery.py 10.104.245.91 5'
```

DNS Anycast with Cisco Prime Network Registrar

```
print 'or'
print 'e.g. /usr/bin/python DNSQuery.py 10.104.245.91'
sys.exit(0)
dns_server = sys.argv[1]
# The the program gets the number of query retries as input, use it.
# Else set the default as 1
if len(sys.argv) > 2:
query_retries = int (sys.argv[2])
else:
query_retries = 1
# Initialize the acceptable number of query failures before shutting down the
# BGP daemon
number_failures = 0
number_successes = 0
while run == True:
try:
request = dns.message.make_query("www.example.com.", dns.rdatatype.A)
response = dns.query.udp(request, dns_server, timeout=2)
number_successes = number_successes + 1
if (number_successes == query_retries):
print 'The DNS server is definitely up'
#The server seems up. If the BGP daemon is not running, start it
#Check if the BGP daemon is running
output = subprocess.check_output("/bin/ps -ef | /bin/grep -v grep | /bin/grep bgpd", shell=True)
print output
if 'bgpd.conf' in output:
print "The BGP daemon is already running"
run = False
# The check_output call threw an exception. If the exception is 1, this means the grep returned nothing.
# So start the BGP daemon.
except subprocess.CalledProcessError as grepexc:
print "error code", grepexc.returncode, grepexc.output
if grepexc.returncode == 1:
print 'BGP daemon was not running. Start it'
```

```
os.system('/etc/init.d/bgpd start')
run = False
#The DNS query threw an exception. Retry. Once the retries also fail, just stop the BGP daemon.
except dns.exception.DNSException as dnsexc:
print 'DNS Query failed. Retry.'
number_failures = number_failures + 1
if (number_failures == query_retries):
print 'All the query retries also failed.'
# The server looks like it is down. If the BGP daemon is running, stop it
# Check if the BGP daemon is running
try:
output = subprocess.check_output("ps -ef | grep -v grep | grep bgpd", shell=True)
if 'bgpd' in output:
os.system('/etc/init.d/bgpd stop')
run = False
except subprocess.CalledProcessError as grepexc:
if grepexc.returncode == 1:
print 'BGP daemon was not running'
run = False
#Encountered unexpected error.
except:
print 'Encountered unexpected exception. Check your system and script'
run = False
```

SNMP

SNMP traps can also be used to detect the health of the DNS server. If SNMP traps are enabled for the DNS server and the trap recipients are added, the CPNR DNS server generates traps in response to predetermined events that the application code detects and signals, e.g. Server stop, DNS masters not responding etc. These traps should routinely be monitored and an event requiring attention be detected, the following command can be executed to stop the BGP daemon on the host to take this DNS server out of the anycast group.

#/etc/init.d/bgpd stop

Once the issue impacting the server has been resolved, the following command can be executed to restart the BGP daemon on the host.

#/etc/init.d/bgpd start

Configure DNS Zones

While this is the conclusion of setting up the Anycast functionality, administrators will need to complete the configuration of the DNS servers. This can be found at http://www.cisco.com/c/en/us/support/cloud-systems-management/prime-network-registrar/tsd-products-support-series-home.html

Refer the below links for more information:

http://www.pacnog.org/pacnog6/IXP/Anycast-v10.pdf

http://www.nongnu.org/Quagga

http://www.linuxjournal.com/magazine/ipv4-anycast-linux-and-Quagga

http://ddiguru.com/blog/125-anycast-dns-part-5-using-bgp