Use Cases: Application Hosting

This chapter describes use cases for running applications on IOS XR.

Running a Telemetry Receiver in a Linux Container (LXC)

For telemetry to work on Cisco IOS XR, it must use GPB (Google Protocol Buffer) over UDP, instead of TCP.

The procedure consists of the following steps:

1. Create a telemetry policy file.
2. Generate and compile a .proto file.
3. Configure the GPB encoder.
4. Launch a third-party container (LXC).
5. Configure the telemetry receiver.

Creating a Telemetry Policy File

A telemetry policy file is used to specify the kind of data to be generated and pushed to the telemetry receiver. The following steps describe how you can create the policy file for telemetry:

1. Determine the schema paths to stream data.

```
RP/0/RP0/CPU0:ios# schema-describe show interface
Wed Aug 26 02:24:40.556 PDT
RootOper.InfraStatistics.Interface(*).Latest.GenericCounters
```

2. Create a policy file that contains these paths:

```
{
"Name": "Test",
"Metadata": {
"Version": 25,
"Description": "This is a sample policy",
"Comment": "This is the first draft",
"Identifier": "<data that may be sent by the encoder to the mgmt stn"
```

```
},
"CollectionGroups": {
"FirstGroup": {
"Period": 30,
"Paths": [
"RootOper.InfraStatistics.Interface(*).Latest.GenericCounters"
   ]
  }
 }
}
```

3. Enter the XR Linux bash shell, and copy the policy file to IOS XR by using Secure Copy Protocol (SCP).

```
/* If you are using Cisco IOS XR Version 6.0.0, run the following command */
RP/0/RP0/CPU0:ios# run ip netns exec tpnns bash

/* If you are using Cisco IOS XR Version 6.0.2, run the following command */
RP/0/RP0/CPU0:ios# bash


[XR-vm_node0_RP0_CPU0:~]$ scp Test.policy cisco@10.0.0.1:/telemetry/policies
cisco@10.0.0.1's password:
Test.policy
100% 779 0.8KB/s 00:00
Connection to 10.0.0.1 closed by remote host.
```

   Where 10.0.0.1 is the IP address of the device on which you are copying the policy file.

4. Navigate to the IOS XR prompt and verify if the policy file has been successfully installed.

```
RP/0/RP0/CPU0:ios# show telemetry policies brief
Wed Aug 26 02:24:40.556 PDT
Name |Active?| Version | Description
----------------------------|-------|---------|------------------------------
Test N 1 This is a sample policy
```

Generating and Compiling a .proto File

The path in a policy file that you created needs a `.proto` file associated with it. The .proto file describes the GPB message format used to stream data. The following steps describe how you can generate and compile a .proto file for a telemetry receiver:

The `.proto` file is complied into a `.map` file. The compilation is done on a server.

1. Generate a .proto file.

```
telemetry generate gpb-encoding path
"RootOper.InfraStatistics.Interface(*).Latest.GenericCounters" file
disk0:generic_counters.proto
```

   The .proto file is generated by an on-box tool. The tool ignores naming parameters, and are hence optional.

Note The tool ignores text within quotes; therefore, the path should not contain quotes.

2. Compile the .proto file off the box.

   a. Cisco provides a telemetry compiler on Dev Hub. You can copy the directory to your Linux box, and run it, as shown here:

```
telemetry_protoc -f generic_counters.proto -o generic_counters.map
```

b. Access the copy of the .proto file from Dev Hub, and run the standard compiler on your Linux box, as shown here:

```
protoc python_out . -I=/
sw/packages/protoc/current/google/include/:.
generic_counters.proto ipv4_counters.proto
```

3. Copy the map file to IOS XR at `/telemetry/gpb/maps`.

Configuring the GPB Encoder

Configure the GPB encoder to activate the telemetry policy and stream data as outlined in the following steps:

1. Configure a loopback interface address for mapping the telemetry receiver to IOS XR, as shown here:

```
RP/0/RP0/CPU0:ios(config)# interface Loopback2
RP/0/RP0/CPU0:ios(config-if)# ipv4 address 2.2.2.2/32
RP/0/RP0/CPU0:ios(config-if)# no shut
RP/0/RP0/CPU0:ios(config-if)# commit
Fri Oct 30 07:51:14.785 UTC
RP/0/RP0/CPU0:ios(config-if)# exit
RP/0/RP0/CPU0:ios(config)# exit
RP/0/RP0/CPU0:ios# show ipv4 interface brief
Fri Oct 30 07:51:48.996 UTC

Interface                      IP-Address      Status                Protocol
Loopback0                      1.1.1.1         Up                    Up
Loopback1                      8.8.8.8         Up                    Up
Loopback2                      2.2.2.2         Up                    Up
GigabitEthernet0/0/0/0         192.164.168.10  Up                    Up
GigabitEthernet0/0/0/1         192.57.43.10    Up                    Up
GigabitEthernet0/0/0/2         unassigned      Shutdown              Down
MgmtEth0/RP0/CPU0/0            192.168.122.197 Up                    Up
RP/0/RP0/CPU0:ios#
```

2. Configure the encoder to stream the policy to the loopback interface of IOS XR that was just configured.

```
telemetry
          encoder gpb
            policy group alpha
              policy demo
              destination ipv4 2.2.2.2 port 5555
             !
      !
   !
```

Launching a Third-Party Container (LXC)

This section describes how you can launch a third-party container (LXC) on IOS XR.

1. Log into IOS XR.

```
RP/0/RP0/CPU0:ios# run
[xr-vm_node0_RP0_CPU0:~]$
```

2. Launch the third-party container.

```
[xr-vm_node0_RP0_CPU0:~]$ virsh -c lxc+tcp://10.11.12.15:16509/ -e ^Q console demo1
```

3. Log into the container when prompted.

```
Connected to domain demo
Escape character is ^Q
```

```
Kernel 3.14.23-WR7.0.0.2_standard on an x86_64
host login: Password:
```

You have successfully launched a third-party container.

Configuring the Telemetry Receiver

A telemetry receiver listens for streamed data on the specified interface IP address and port number, and it prints the header of the received packets. If .proto files are provided, they are compiled using the protoc compiler and the message contents are also printed. By default, only the first row of each table is printed, though the `print-all` option can be used to print the complete output.

To run a telemetry receiver within the container you launched, use the following steps:

1. Download all the receiver files to the third-party container. The receiver files are available on IOS XR at https://github.com/cisco/bigmuddy-network-telemetry-collector.
2. Run the receiver to stream and print data.

   ```
   python gpb_receiver.py ipaddress 2.2.2.2 port 5555 proto
   generic_counters.proto ipv4_counters.proto
   ```

   You can see data on the telemetry receiver, as shown here:

   ```
   Waiting for message
   Got message of length:1036bytes from address:('10.1.1.1', 5555)
   Encoding:2271560481
   Policy Name:demo
   Version:25
   Identifier:<data that may be sent by the encoder to the mgmt stn>
   Start Time:Wed Jan 21 09:54:33 1970
   End Time:Wed Aug 26 09:28:37 2015
   # Tables:1
   Schema
   Path:RootOper.InfraStatistics.Interface.Latest.GenericCounters
   # Rows:6
   Row 0:
   applique:0
   availability_flag:0
   broadcast_packets_received:0
   broadcast_packets_sent:0
   bytes_received:0
   bytes_sent:0
   carrier_transitions:0
   crc_errors:0
   framing_errors_received:0
   giant_packets_received:0
   input_aborts:0
   input_drops:0
   input_errors:0
   input_ignored_packets:0
   input_overruns:0
   input_queue_drops:0
   interface_name:Null0
   last_data_time:1440606516
   last_discontinuity_time:1440498130
   multicast_packets_received:0
   multicast_packets_sent:0
   output_buffer_failures:0
   output_buffers_swapped_out:0
   output_drops:0
   output_errors:0
   output_queue_drops:0
   ```

```
output_underruns:0
packets_received:0
packets_sent:0
parity_packets_received:0
resets:0
runt_packets_received:0
seconds_since_last_clear_counters:0
seconds_since_packet_received:4294967295
seconds_since_packet_sent:4294967295
throttled_packets_received:0
unknown_protocol_packets_received:0
Waiting for message
Got message of length:510bytes from address:('2.2.2.2', 5555)
Encoding:2271560481
Policy Name:demo
Version:25
Identifier:<data that may be sent by the encoder to the mgmt stn>
Start Time:Wed Jan 21 09:54:33 1970
End Time:Wed Aug 26 09:28:38 2015
# Tables:1
Schema Path:RootOper.InfraStatistics.Interface.Latest.Protocol
# Rows:5
Row 0:
bytes_received:0
bytes_sent:0
input_data_rate:0
input_packet_rate:0
interface_name:Loopback2
last_data_time:1440606517
output_data_rate:0
output_packet_rate:0
packets_received:0
packets_sent:0
protocol:24
protocol_name:IPV4_UNICAST
```

The telemetry receiver runs successfully within the third-party container (LXC).

Use Cases on Vagrant: Container Application Hosting

This section describes how you can use vagrant to run use cases for container application hosting.

Pre-requisites for Using Vagrant

Before you can start using vagrant, ensure that you have fulfilled the following requirements on your host device.

- Latest version of Vagrant for your operating system. We recommend Version 1.8.6.
- Latest version of a virtual box for your operating system. We recommend Version 5.1+.
- Minimum of 5 GB of RAM with two cores.
- (Optional) If you are using the Windows Operating System, we recommend that you download the Git bash utility for running the commands.

OSPF Path Failover by Running iPerf with Netconf on Vagrant

This section describes a use case for solving a path remediation problem by using iPerf and Netconf applications on vagrant.

Topology

The topology used for OSPF path remediation is illustrated in the following figure.

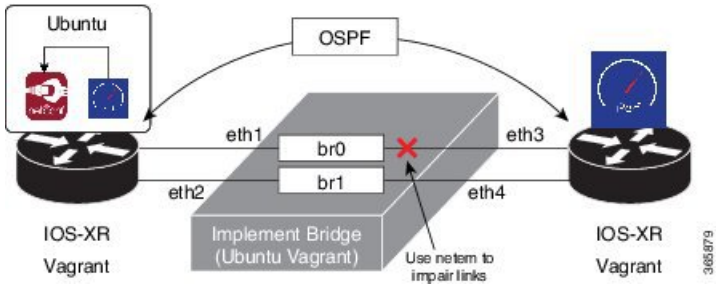
The router on the left is rtr1 and is the source of traffic. We run the pathchecker application inside an LXC on this router. Pathchecker uses an iPerf client to determine the health of the path.

The router on the right is rtr2 and is the destination for traffic. We run the pathchecker application inside an LXC on this router. Pathchecker uses an iPerf server that talks to the iPerf client on rtr1.

`devbox` serves two purposes in this topology:

- To create an LXC tar ball with pathchecker before being deployed to the routers.
- To bridge the two networks between the two routers over the parallel paths.

Figure 1: OSPF Path Failover with iPerf and Netconf on Vagrant



This example uses the following process for OSPF path failover:

1. Configure and establish OSPF neighbor relationship between two routers over two parallel paths.
2. Increase the cost of one path so that the other path is the preferred active path.
3. Use the pathchecker python application to monitor the OSPF active path by determining the bandwidth, jitter, packet loss and other parameters. Pathchecker uses the iPerf application to measure health of the active traffic path.
4. Use pathchecker to simulate network degradation by changing the OSPF active path cost during a Netconf session.

Procedure

Use the following steps to use iPerf with Netconf for OSPF path failover.

1. Generate an API key and a CCO ID by using the steps described on Github.
2. Download the latest stable version of the IOS-XRv vagrant box.

```
$ curl <cco-id>:<API-KEY>

$ BOXURL --output ~/iosxrv-fullk9-x64.box
```

```
$ vagrant box add --name IOS-XRv ~/iosxrv-fullk9-x64.box
```

3. Verify if the vagrant box has been successfully installed.

```
 AKSHSHAR-M-K0DS:~ akshshar$ vagrant box list
IOS-XRv (virtualbox, 0)
```

4. Create a working directory.

```
 AKSHSHAR-M-K0DS:~ akshshar$ mkdir ~/iosxrv
 AKSHSHAR-M-K0DS:~ akshshar$cd ~/iosxrv
```

5. Initialize the vagrant file with the new vagrant box.

```
 AKSHSHAR-M-K0DS:~ akshshar$ vagrant init IOS-XRv
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

6. Clone the repository containing the pathchecker code.

```
AKSHSHAR-M-K0DS:~ akshshar$ git clone https://github.com/ios-xr/pathchecker.git
Cloning into 'pathchecker'...
remote: Counting objects: 46, done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 46 (delta 8), reused 0 (delta 0), pack-reused 18
Unpacking objects: 100% (46/46), done.
Checking connectivity... done.
```

7. Navigate to the `pathchecker/vagrant` directory and launch `devbox`.

```
AKSHSHAR-M-K0DS:~ akshshar$ cd pathchecker/
AKSHSHAR-M-K0DS:pathchecker akshshar$ cd vagrant/
AKSHSHAR-M-K0DS:vagrant akshshar$ pwd
/Users/akshshar/pathchecker/vagrant

AKSHSHAR-M-K0DS:vagrant akshshar$ vagrant up devbox
Bringing machine 'devbox' up with 'virtualbox' provider...
==> devbox: Importing base box 'ubuntu/trusty64'...

---------------------------- snip output ---------------------------------

==> devbox: Running provisioner: file...
AKSHSHAR-M-K0DS:vagrant akshshar$
AKSHSHAR-M-K0DS:vagrant akshshar$
AKSHSHAR-M-K0DS:vagrant akshshar$ vagrant status
Current machine states:


rtr1                      not created (virtualbox)
devbox                    running (virtualbox)
rtr2                      not created (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

8. Launch an LXC within `devbox`.

```
AKSHSHAR-M-K0DS:vagrant akshshar$ vagrant ssh devbox

vagrant@vagrant-ubuntu-trusty-64:~$ sudo lxc-create -t ubuntu --name pathchecker
Checking cache download in /var/cache/lxc/trusty/rootfs-amd64 ...
Installing packages in template: ssh,vim,language-pack-en
Downloading ubuntu trusty minimal ...
I: Retrieving Release
I: Retrieving Release.gpg
I: Checking Release signature
...
vagrant@vagrant-ubuntu-trusty-64:~$ sudo lxc-start --name pathchecker
<4>init: hostname main process (3) terminated with status 1
<4>init: plymouth-upstart-bridge main process (5) terminated with status 1
<4>init: plymouth-upstart-bridge main process ended, respawning


Ubuntu 14.04.4 LTS nc_iperf console


pathchecker login: ubuntu
Password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

...
```

9. Install all the required iPerf and Netconf application dependencies within the LXC.

```
ubuntu@pathchecker:~$ sudo apt-get -y install python-pip python-lxml
 python-dev libffi-dev libssl-dev iperf git

ubuntu@pathchecker:~$ sudo pip install ncclient jinja2 cryptography==1.2.1
```

10. Retrieve the iPerf and Netconf application code from Github.

```
ubuntu@pathchecker:~$ git clone https://github.com/ios-xr/pathchecker.git
Cloning into 'pathchecker'...
remote: Counting objects: 46, done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 46 (delta 8), reused 0 (delta 0), pack-reused 18
Unpacking objects: 100% (46/46), done.
Checking connectivity... done.
ubuntu@pathchecker:~$
```

11. Change the SSH port inside the LXC.

When a container is deployed on XR, it shares the network namespace of XR. Since XR uses ports 22 and 57722 for internal processes, we change the port number to 58822 in this example.

```
ubuntu@pathchecker:~$ sudo sed -i s/Port\ 22/Port\ 58822/ /etc/ssh/sshd_config

ubuntu@pathchecker:~$ cat /etc/ssh/sshd_config | grep Port
Port 58822
```

12. Create the LXC tar ball.

a. Shut down the LXC.

```
ubuntu@pathchecker:~$ sudo shutdown -h now
ubuntu@pathchecker:~$
Broadcast message from ubuntu@pathchecker
 (/dev/lxc/console) at 10:24 ...

The system is going down for halt NOW!
```

b. Assume the root user role.

```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo -s
root@vagrant-ubuntu-trusty-64:~# whoami
root
```

c. Navigate to the `/var/lib/lxc/pathchecker/rootfs/` directory and package the `rootfs` into a tar ball.

```
root@vagrant-ubuntu-trusty-64:~# cd /var/lib/lxc/pathchecker/rootfs/
root@vagrant-ubuntu-trusty-64:/var/lib/lxc/pathchecker/rootfs/# tar -czvf
/vagrant/pathchecker_rootfs.tar.gz *
tar: dev/log: socket ignored
root@vagrant-ubuntu-trusty-64:/var/lib/lxc/pathchecker/rootfs/# exit
vagrant@vagrant-ubuntu-trusty-64:~$ exit
logout
Connection to 127.0.0.1 closed.

AKSHSHAR-M-K0DS:vagrant akshshar$ pwd
/Users/akshshar/pathchecker/vagrant
AKSHSHAR-M-K0DS:vagrant akshshar$ ls -l pathchecker_rootfs.tar.gz
-rw-r--r--  1 akshshar  staff  301262995 Jul 18 07:57 pathchecker_rootfs.tar.gz
AKSHSHAR-M-K0DS:vagrant akshshar$
```

13. Launch the two router topology.

a. Navigate to the `pathchecker/vagrant` directory and launch the vagrant instance.

```
AKSHSHAR-M-K0DS:vagrant akshshar$ pwd
/Users/akshshar/pathchecker/vagrant

AKSHSHAR-M-K0DS:vagrant akshshar$ vagrant up
Bringing machine 'rtr1' up with 'virtualbox' provider...
Bringing machine 'devbox' up with 'virtualbox' provider...
Bringing machine 'rtr2' up with 'virtualbox' provider...
```

b. Verify if the topology has been launched.

```
AKSHSHAR-M-K0DS:vagrant akshshar$ vagrant status
Current machine states:


rtr1                      running (virtualbox)
devbox                    running (virtualbox)
rtr2                      running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

14. Verify if OSPF is running on rtr1 and check the path state.

You can also see the cost of the OSPF path.

```
AKSHSHAR-M-K0DS:vagrant akshshar$ vagrant port rtr1
The forwarded ports for the machine are listed below. Please note that
these values may differ from values configured in the Vagrantfile if the
provider supports automatic port collision detection and resolution.

 22 (guest) => 2223 (host)
 57722 (guest) => 2200 (host)
 58822 (guest) => 58822 (host)
AKSHSHAR-M-K0DS:vagrant akshshar$ ssh -p 2223 vagrant@localhost
The authenticity of host '[localhost]:2223 ([127.0.0.1]:2223)' can't be established.
RSA key fingerprint is b1:c1:5e:a5:7e:e7:c0:4f:32:ef:85:f9:3d:27:36:0f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:2223' (RSA) to the list of known hosts.
vagrant@localhost's password:



RP/0/RP0/CPU0:rtr1# show  running-config  router ospf
Mon Jul 18 15:25:53.875 UTC
router ospf apphost
 area 0
  interface Loopback0
  !
  interface GigabitEthernet0/0/0/0
  !
  interface GigabitEthernet0/0/0/1
    cost 20
  !
 !
!

RP/0/RP0/CPU0:rtr1# show route 2.2.2.2
Mon Jul 18 15:26:03.576 UTC

Routing entry for 2.2.2.2/32
  Known via "ospf apphost", distance 110, metric 2, type intra area
  Installed Jul 18 15:18:28.218 for 00:07:35
  Routing Descriptor Blocks
   10.1.1.20, from 2.2.2.2, via GigabitEthernet0/0/0/0
      Route metric is 2
  No advertising protos.
RP/0/RP0/CPU0:rtr1#
```

15. Start the iPerf server on `rtr2` and configure it for receiving packets from `rtr1`.

Note iPerf was launched as a native application on `rtr2` while launching the vagrant instance.

```
AKSHSHAR-M-K0DS:vagrant akshshar$ vagrant ssh rtr2
Last login: Mon Jul 18 15:57:05 2016 from 10.0.2.2
xr-vm_node0_RP0_CPU0:~$
xr-vm_node0_RP0_CPU0:~$ iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 64.0 MByte (default)
```

16. Launch the pathchecker application within the LXC on `rtr1`.

a. Log in to the LXC on `rtr1`.

Password for user `ubuntu` is **ubuntu**.

```
AKSHSHAR-M-K0DS:vagrant akshshar$ ssh -p 58822 ubuntu@localhost
The authenticity of host '[localhost]:58822 ([127.0.0.1]:58822)' can't be
established.
RSA key fingerprint is 19:54:83:a9:7a:9f:0a:18:62:d1:f3:91:87:3c:e9:0b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:58822' (RSA) to the list of known hosts.
ubuntu@localhost's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.14.23-WR7.0.0.2_standard x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Mon Jul 18 15:19:45 2016 from 10.0.2.2
ubuntu@pathchecker:~$
```

b. Navigate to the pathchecker repository within the LXC, and check the contents of the pathchecker script.

```
ubuntu@pathchecker:~$ cd pathchecker/
ubuntu@pathchecker:~/pathchecker$ cat pc_run.sh
#!/bin/bash

./pathchecker.py --host 6.6.6.6 -u vagrant -p vagrant --port 830 -c 10 -o
apphost -a 0 -i GigabitEthernet0/0/0/0 -s 2.2.2.2  -j 4 -l 5 -f -t 10
```

`-I` represents the threshold for packet loss and has been set to 5% for this run. `-j` represents the jitter threshold that has a value of 4.

c. Start the pathchecker application by running the script.

```
ubuntu@pathchecker:~/pathchecker$ ./pc_run.sh
Error while opening state file, let's assume low cost state
Currently, on reference link GigabitEthernet0/0/0/0
Starting an iperf run.....
20160718162513,1.1.1.1,62786,2.2.2.2,5001,6,0.0-10.0,1311240,1048992
20160718162513,1.1.1.1,62786,2.2.2.2,5001,6,0.0-10.0,1312710,1048474
20160718162513,2.2.2.2,5001,1.1.1.1,62786,6,0.0-10.0,1312710,1048679,2.453,0,892,0.000,1

bw is
1025.5546875
jitter is
2.453
pkt_loss is
0.000
verdict is
False
Currently, on reference link GigabitEthernet0/0/0/0
Starting an iperf run.....
```

The pathchecker application is running on the path from `GigabitEthernet0/0/0/0` interface.

17. Open a parallel Git bash window and simulate impairment on the active path.

a. Access `devbox` through SSH.

```
AKSHSHAR-M-K0DS:vagrant akshshar$ cd pathchecker/vagrant

AKSHSHAR-M-K0DS:vagrant akshshar$ vagrant ssh devbox
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-87-generic x86_64)

 ...
```

b. View the impairment script and run it on `devbox`.

```
vagrant@vagrant-ubuntu-trusty-64:~$ ls
impair_backup.sh  impair_reference.sh  stop_impair.sh

vagrant@vagrant-ubuntu-trusty-64:~$ cat impair_reference.sh
#!/bin/bash
echo "Stopping all current impairments"
sudo tc qdisc del dev eth3 root &> /dev/null
sudo tc qdisc del dev eth4 root &> /dev/null
echo "Starting packet loss on reference link"
sudo tc qdisc add dev eth3 root netem loss 7%

vagrant@vagrant-ubuntu-trusty-64:~$ ./impair_reference.sh
Stopping all current impairments
Starting packet loss on reference link
```

The script creates a packet loss of 7% on the reference link.

18. Open the first Git bash window to view the pathchecker application running on `rtr1`.

```
Currently, on reference link GigabitEthernet0/0/0/0
Starting an iperf run.....
20160718164745,1.1.1.1,60318,2.2.2.2,5001,6,0.0-10.0,1311240,1048992
20160718164745,1.1.1.1,60318,2.2.2.2,5001,6,0.0-10.0,1312710,1048516
20160718164745,2.2.2.2,5001,1.1.1.1,60318,6,0.0-573.0,1312710,18328,5.215,0,892,0.000,1

bw is
1025.5546875
jitter is
5.215
pkt_loss is
0.000
verdict is
True
Woah! iperf run reported discrepancy, increase cost of reference link !
Increasing cost of the reference link GigabitEthernet0/0/0/0
Currently, on backup link
Starting an iperf run.....
20160718164755,1.1.1.1,61649,2.2.2.2,5001,6,0.0-10.0,1311240,1048992
20160718164755,1.1.1.1,61649,2.2.2.2,5001,6,0.0-10.0,1312710,1048577
20160718164755,2.2.2.2,5001,1.1.1.1,61649,6,0.0-583.3,1312710,18002,1.627,0,893,0.000,0

bw is
1025.5546875
jitter is
1.627
pkt_loss is
0.000
verdict is
False
Currently, on backup link
Starting an iperf run.....
20160718164805,1.1.1.1,59343,2.2.2.2,5001,6,0.0-10.0,1311240,1048992
20160718164805,1.1.1.1,59343,2.2.2.2,5001,6,0.0-10.0,1312710,1048520
20160718164805,2.2.2.2,5001,1.1.1.1,59343,6,0.0-593.4,1312710,17697,2.038,0,893,0.000,0
```

Pathchecker has initiated a failover from primary to secondary link.

19. Verify if the failover was successful on `rtr1`.

```
AKSHSHAR-M-K0DS:vagrant akshshar$ ssh -p 2223 vagrant@localhost
vagrant@localhost's password:



RP/0/RP0/CPU0:rtr1# show  running-config  router ospf
Mon Jul 18 17:50:47.851 UTC
router ospf apphost
 area 0
  interface Loopback0
  !
  interface GigabitEthernet0/0/0/0
   cost 30
  !
  interface GigabitEthernet0/0/0/1
   cost 20
  !
 !
!
```

The path cost from the GigabitEthernet0/0/0/0 interface is greater than that from the GigabitEthernet0/0/0/1 interface. Hence, failover takes place to the GigabitEthernet0/0/0/1 interface for traffic from `rt1` to `rtr2`.

20. Verify the OSPF path failover on **`rtr1`**.

The Loopback 0 interface IP address of `rtr1` in this example is 2.2.2.2

```
RP/0/RP0/CPU0:rtr1# show route 2.2.2.2
Mon Jul 18 18:01:49.297 UTC

Routing entry for 2.2.2.2/32
  Known via "ospf apphost", distance 110, metric 21, type intra area
  Installed Jul 18 16:47:45.705 for 01:14:03
  Routing Descriptor Blocks
  11.1.1.20, from 2.2.2.2, via GigabitEthernet0/0/0/1
      Route metric is 21
  No advertising protos.
RP/0/RP0/CPU0:rtr1#
```

The next hop for `rtr1` is 11.1.1.20 through the backup reference link: GigabitEthernet0/0/0/1

You have successfully configured OSPF path failover by using iPerf and Netconf on vagrant.

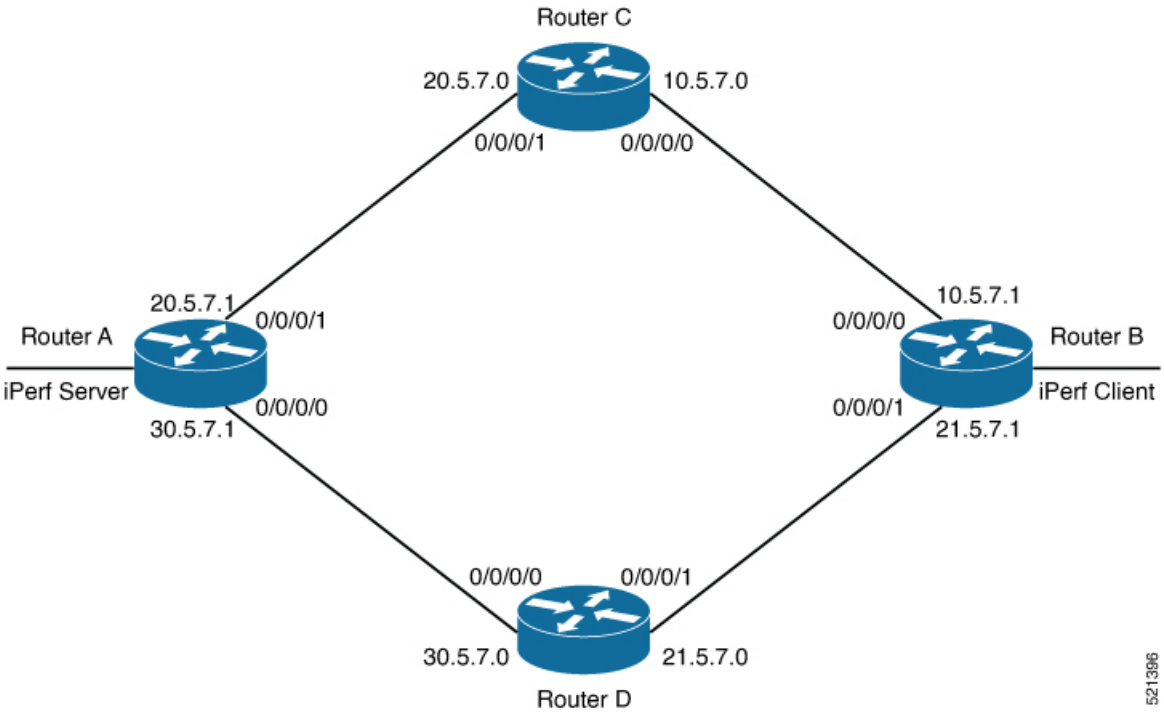
Hosting iPerf in Docker Containers to Measure Network Performance using Application Manager

Measuring the network performance is important to test the efficiency of the network. Network throughput, bandwidth, latency, and packet loss are some of the parameters used to measure the network performance. iPerf is a commonly used application for measuring network performance. The iPerf application is hosted on systems at both ends of the connection that is measured. One system is used as the server, and the other system is used as the client. At least one system must be a Cisco IOS XR router, the other system can be any other external entity like a controller or another router.

This use case illustrates the procedure for hosting the iPerf application in docker containers on two Cisco IOS XR routers, Router A and Router B to measure network performance. Router A hosts the iPerf server and Router B hosts the iPerf client.

In this usecase, we demonstrate the example of testing network bandwidth when a route update takes place. Router A hosts the iPerf Server and Router B hosts the iPerf Client. Router C and Router D are intermediate routers that allow traffic flow from Router A to Router B and vice-versa.

Figure 2: Hosting iPerf Application in Cisco IOS XR Routers



Verify Connection between Router A and Router B

The **ping** command verifies the connection between the IOS XR software on the routers, while the **bash ping** command verifies the connection between the linux kernel that hosts the IOS XR software on the routers.

Before you begin

Access global VRF on Cisco IOS XR Linux Shell. See Accessing Global VRF on the Cisco IOS XR Linux Shell.

Check the connection between Router A and Router B using the **ping** and **bash ping** commands.

```
Router#show ip route 30.5.7.1
Tue Dec  1 19:27:28.623 UTC

Routing entry for 30.5.7.0/31
  Known via "ospf 10", distance 110, metric 2, type intra area
  Installed Dec  1 18:09:44.525 for 01:17:44
  Routing Descriptor Blocks
    21.5.7.0, from 100.0.0.7, via FourHundredGigE0/0/0/1
```

```
      Route metric is 2
  No advertising protos.
Router#ping 30.5.7.1
Tue Dec  1 19:27:28.769 UTC
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 30.5.7.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/24/30 ms
Router#bash ping -c 5 30.5.7.1
PING 30.5.7.1 (30.5.7.1) 56(84) bytes of data.
64 bytes from 30.5.7.1: icmp_seq=1 ttl=254 time=31.9 ms
64 bytes from 30.5.7.1: icmp_seq=2 ttl=254 time=37.7 ms
64 bytes from 30.5.7.1: icmp_seq=3 ttl=254 time=30.5 ms
64 bytes from 30.5.7.1: icmp_seq=4 ttl=254 time=27.5 ms
64 bytes from 30.5.7.1: icmp_seq=5 ttl=254 time=30.3 ms

--- 30.5.7.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 27.549/31.621/37.719/3.371 ms
```

Install the iPerf Server Application

Step 1 Install the iPerf application RPM on Router A. Only the RPM file format is supported.

```
Router#appmgr package install rpm /misc/disk1/iperf-0.1.0-XR_7.3.1.x86_64.rpm

Router#show appmgr source-table
Thu Dec  3 09:57:40.808 UTC
Name            File
--------------- ------------------------------------------------------------------
iperf    iperf.tar.gz
Router#
```

Step 2 Configure the application to run as iPerf server.

```
Router#config
Thu Dec  3 09:57:54.034 UTC
Router(config)#appmgr
Router(config-appmgr)#application iperf-server-app
Router(config-application)#activate type docker source iperf docker-run-opts "--net=host" docker-run-cmd
 "iperf3 -s -d"
Router(config-application)#commit
Thu Dec  3 09:57:54.398 UTC
```

Step 3 Verify the basic details (application name and state) about the activated iPerf server application.

```
Router#show appmgr application-table
Name                 Type    Config State  Status
--------             ------- ------------- -------------------------
iperf-server-app     Docker    Activated   Up 2 seconds
Router#
Thu Dec  3 09:57:54.398 UTC
Router#show appmgr application name iperf-server-app info summary
Thu Dec  3 09:58:15.569 UTC
Application: iperf-server-app
    Type: Docker
    Source: iperf
    Config State: Activated
    Container ID: 0118f9006cde2787e9809eb7c62ad8b552925b559a689c7aaa80f80d7ce43c02
```

```
    Image: alpine1:latest
    Command: "iperf3 -s -d"
    Status: Up 7 seconds
Thu Dec  3 09:57:54.398 UTC
Router#show appmgr application name iperf-server-app info detail
Thu Dec  3 09:58:26.401 UTC
Application: iperf-server-app
    Type: Docker
    Source: iperf
    Config State: Activated
    Docker Information:
        Container ID: 0118f9006cde2787e9809eb7c62ad8b552925b559a689c7aaa80f80d7ce43c02
        Container name: iperf-server-app
        Labels:
        Image: alpine1:latest
        Command: "iperf3 -s -d"
        Created at: 2020-12-03 09:58:08 +0000 UTC
        Running for: 18 seconds ago
        Status: Up 18 seconds
        Size: 0B
        Ports:
        Mounts:
        Networks: host
        LocalVolumes: 0
Router#show appmgr application name iperf-server-app stats
Thu Dec  3 09:58:39.594 UTC
Application Stats: iperf-server-app
   CPU Percentage: 0.00%
   Memory Usage: 624KiB / 31.23GiB
   Memory Percentage: 0.00%
   Network IO: 0B / 0B
   Block IO: 0B / 0B
   PIDs: 1
Router#
```

Step 4 Verify if the iPerf server is listening on the default port (5201) by using the netstat command inside the container.

The appmgr application exec name *app_name* docker-exec-cmd command can be used to execute any commands inside the container.

```
Router#appmgr application exec name iperf-server-app docker-exec-cmd name netstat -lnput
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.11:46727        0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:5201          0.0.0.0:*               LISTEN      -
udp        0      0 127.0.0.11:39552        0.0.0.0:*
Router#
```

Install the iPerf Client Application

Step 1 Install the iPerf application RPM on Router B.

```
Router#appmgr package install rpm /misc/disk1/iperf-0.1.0-XR_7.3.1.x86_64.rpm
Router#show appmgr source-table
Thu Dec  3 09:57:40.808 UTC
Name            File
--------------- ------------------------------------------------------------------
```

```
iperf      iperf.tar.gz
Router#
```

Step 2 Configure the application to run as iPerf client with a timeout (600s in this case).

```
Router#config
Thu Dec  3 09:57:54.034 UTC
Router(config)#appmgr
Router(config-appmgr)#application iperf-client-app
Router(config-application)#activate type docker source iperf docker-run-opts "--net=host" docker-run-cmd
 "iperf3 -c 30.5.7.1 -t 600"
Router(config-application)#commit
Thu Dec  3 09:57:54.398 UTC
```

Note Hosting the iPerf client application on Router B by providing the iPerf server physical interface IP address (30.5.7.1) establishes communication between Router B and Router A.

Step 3 Verify the basic details (application name and state) about the activated iPerf client application.

```
Router#show appmgr application-table
Thu Dec  3 09:59:47.628 UTC
Name                 Type    Config State  Status
--------             ------- ------------- -------------------------
iperf-client-app     Docker    Activated   Up 2 seconds
Router#
Thu Dec  3 09:57:54.398 UTC
Router#show appmgr application name iperf-client-app info summary
Thu Dec  3 09:59:54.534 UTC
Application: iperf-client-app
    Type: Docker
    Source: iperf
    Config State: Activated
    Container ID: 40e1730a97666b2b44c8c9313b94b0138925c9198ae63244ff3bd386132d9c9c
    Image: alpine1:latest
    Command: "iperf3 -c 30.5.7.1 -t 600"
    Status: Up 9 seconds
Router#show appmgr application name iperf-client-app info detail
Application: iperf-client-app
    Type: Docker
    Source: iperf
    Config State: Activated
    Docker Information:
        Container ID: 40e1730a97666b2b44c8c9313b94b0138925c9198ae63244ff3bd386132d9c9c
        Container name: iperf-client-app
        Labels:
        Image: alpine1:latest
        Command: "iperf3 -c 30.5.7.1 -t 600"
        Created at: 2020-12-03 09:59:45 +0000 UTC
        Running for: 20 seconds ago
        Status: Up 20 seconds
        Size: 0B
        Ports:
        Mounts:
        Networks: host
        LocalVolumes: 0
Router#show appmgr application name iperf-client-app stats
Thu Dec  3 10:00:18.079 UTC
Application Stats: iperf-client-app
   CPU Percentage: 0.11%
   Memory Usage: 720KiB / 31.23GiB
   Memory Percentage: 0.00%
   Network IO: 0B / 0B
   Block IO: 0B / 0B
```

```
   PIDs: 1
Router#
```

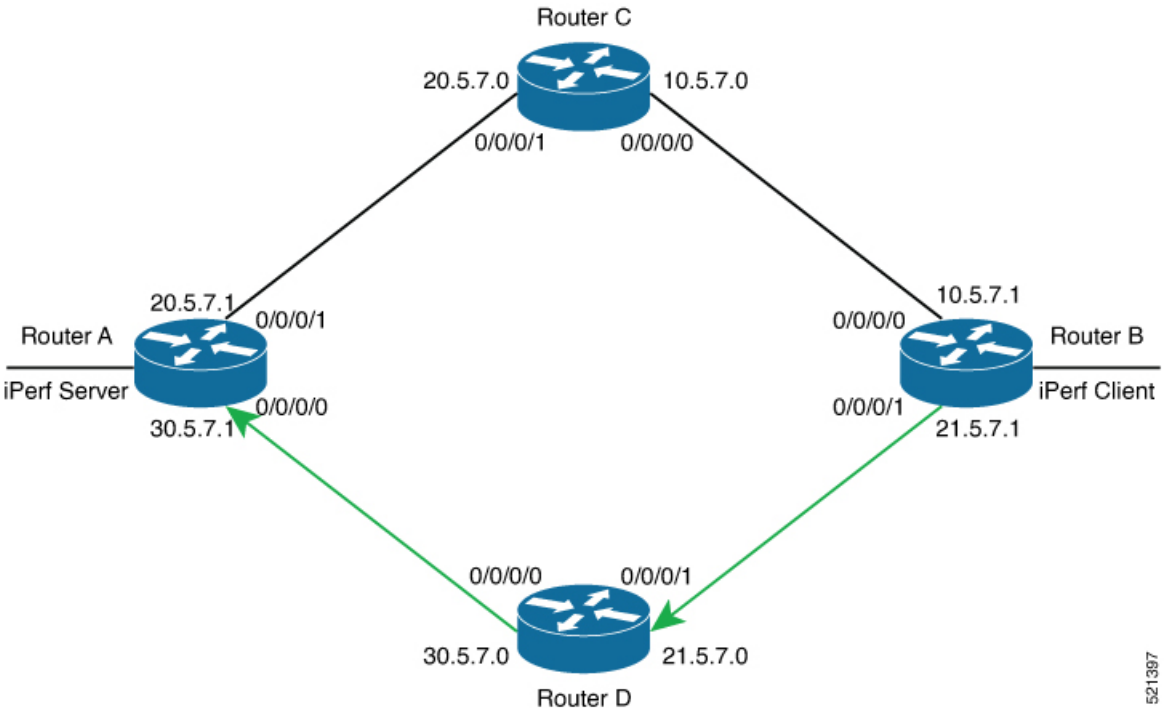
Verify Connection between the iPerf Server and iPerf Client Applications

Verify whether the connection is established between iPerf server and iPerf clients by executing the **bash netstat -anput** command on Router A. When the iPerf client is up and running, the entry in the **State** field displays "ESTABLISHED".

```
Router#bash netstat -anput
Thu Dec  3 10:00:33.535 UTC
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:646             0.0.0.0:*               LISTEN      8585/mpls_ldp
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      8567/ssh_server
tcp        0      0 0.0.0.0:830             0.0.0.0:*               LISTEN      8567/ssh_server
tcp6       0      0 :::5201                 :::*                    LISTEN      20829/iperf3
tcp6       0      0 :::22                   :::*                    LISTEN      8567/ssh_server
tcp6       0      0 :::830                  :::*                    LISTEN      8567/ssh_server
tcp6       0      0 30.5.7.1:5201           100.0.0.9:65322         ESTABLISHED 20829/iperf3
tcp6       0      0 30.5.7.1:5201           100.0.0.9:65302         ESTABLISHED 20829/iperf3
udp        0      0 0.0.0.0:646             0.0.0.0:*                           8585/mpls_ldp
udp        0      0 0.0.0.0:3232            0.0.0.0:*                           6833/pim
udp        0      0 0.0.0.0:3503            0.0.0.0:*                           10762/lspv_server
udp        0      0 0.0.0.0:68              0.0.0.0:*                           10704/xr_dhcpcd
udp        0      0 0.0.0.0:496             0.0.0.0:*                           6833/pim
udp6       0      0 :::3503                 :::*                                10762/lspv_server
```

Measure Network Performance

Step 1 Verify the traffic route from Router B to Router A using the **show ip route** command, on Router B.




```
Router#show ip route 30.5.7.1
Thu Dec  3 10:08:01.859 UTC

Routing entry for 30.5.7.0/31
  Known via "ospf 10", distance 110, metric 2, type intra area
  Installed Dec  3 04:49:22.281 for 05:18:39
  Routing Descriptor Blocks
    21.5.7.0, from 100.0.0.7, via FourHundredGigE0/0/0/1
      Route metric is 2
  No advertising protos.
Router#
```

Step 2 Check the network performance between iPerf client and iPerf server (on Router B and Router A).

You can view the network monitoring parameters by executing the **show appmgr application name iperf-client-app logs** command, on Router B that hosts the iPerf client.

```
Router#show appmgr application name iperf-client-app logs
Tue Dec 1 12:50:27.862 UTC
Connecting to host 30.5.7.1, port 5201
[ 4] local 100.0.0.9 port 61384 connected to 30.5.7.1 port 5201
[ ID] Interval          Transfer     Bandwidth Retr     Cwnd
[ 4] 0.00-1.00 sec 1.05 MBytes     8.82 Mbits/sec 0     80.6 KBytes
[ 4] 1.00-2.00 sec 1.26 MBytes     10.6 Mbits/sec 0     136 KBytes
[ 4] 2.00-3.00 sec 1.18 MBytes     9.90 Mbits/sec 0     191 KBytes
[ 4] 3.00-4.00 sec 1.24 MBytes     10.4 Mbits/sec 0     246 KBytes
[ 4] 4.00-5.00 sec 1.18 MBytes     9.90 Mbits/sec 0     301 KBytes
[ 4] 5.00-6.00 sec 1.37 MBytes     11.5 Mbits/sec 0     362 KBytes
[ 4] 6.00-7.00 sec 1.37 MBytes     11.5 Mbits/sec 0     423 KBytes
[ 4] 7.00-8.00 sec 1.43 MBytes     12.0 Mbits/sec 0     486 KBytes
[ 4] 8.00-9.00 sec 1.30 MBytes     11.0 Mbits/sec 0     547 KBytes
[ 4] 9.00-10.00 sec 1.43 MBytes     12.0 Mbits/sec 0    611 KBytes
[ 4] 10.00-11.00 sec 1.62 MBytes     13.6 Mbits/sec 0   707 KBytes
[ 4] 11.00-12.00 sec 1.62 MBytes     13.6 Mbits/sec 0    875 KBytes
[ 4] 12.00-13.00 sec 1.93 MBytes     16.2 Mbits/sec 0    1.07 MBytes
```

```
[ 4] 13.00-14.00 sec 1.68 MBytes     14.1 Mbits/sec 0    1.29 MBytes
[ 4] 14.00-15.00 sec 1.06 MBytes     8.86 Mbits/sec 0    1.56 MBytes
[ 4] 15.00-16.00 sec 891 KBytes     7.30 Mbits/sec 0     1.83 MBytes
[ 4] 16.00-17.00 sec 970 KBytes     7.95 Mbits/sec 0     2.12 MBytes
[ 4] 17.00-18.00 sec 1.24 MBytes     10.4 Mbits/sec 0    2.58 MBytes
[ 4] 18.00-19.00 sec 885 KBytes     7.24 Mbits/sec 0     2.65 MBytes
[ 4] 19.00-20.00 sec 1.55 MBytes     13.0 Mbits/sec 0    3.10 MBytes
[ 4] 20.00-21.00 sec 820 KBytes     6.71 Mbits/sec 0     3.10 MBytes
[ 4] 21.00-22.00 sec 1.72 MBytes     14.4 Mbits/sec 6     2.42 MBytes
[ 4] 22.00-23.00 sec 0.00 Bytes     0.00 bits/sec 5       2.30 MBytes
[ 4] 23.00-24.00 sec 256 KBytes     2.10 Mbits/sec 0     1.35 MBytes
[ 4] 24.00-25.00 sec 1.56 MBytes     13.1 Mbits/sec 237   1.83 MBytes
[ 4] 25.00-26.00 sec 1.90 MBytes     15.9 Mbits/sec 0     2.17 MBytes
[ 4] 26.00-27.00 sec 382 KBytes     3.12 Mbits/sec 61     1.95 MBytes
[ 4] 27.00-28.00 sec 0.00 Bytes     0.00 bits/sec 0     1.39 MBytes
[ 4] 28.00-29.00 sec 3.35 MBytes     28.1 Mbits/sec 0     1.52 MBytes
[ 4] 29.00-30.00 sec 954 KBytes     7.82 Mbits/sec 0     1.58 MBytes
[ 4] 30.00-31.00 sec 1018 KBytes     8.34 Mbits/sec 0     1.64 MBytes
[ 4] 31.00-32.00 sec 1.24 MBytes     10.4 Mbits/sec 0     1.71 MBytes
[ 4] 32.00-33.00 sec 1.25 MBytes     10.5 Mbits/sec 0     1.76 MBytes
[ 4] 33.00-34.00 sec 1.61 MBytes     13.5 Mbits/sec 0     1.80 MBytes
[ 4] 34.00-35.00 sec 1.46 MBytes     12.2 Mbits/sec 0     1.82 MBytes
[ 4] 35.00-36.00 sec 1.18 MBytes     9.89 Mbits/sec 0     1.83 MBytes
[ 4] 36.00-37.00 sec 1.36 MBytes     11.4 Mbits/sec 0     1.84 MBytes
[ 4] 37.00-38.00 sec 1.36 MBytes     11.4 Mbits/sec 0     1.84 MBytes
[ 4] 38.00-39.00 sec 1.24 MBytes     10.4 Mbits/sec 0     1.84 MBytes
[ 4] 39.00-40.00 sec 1.25 MBytes     10.5 Mbits/sec 0     1.85 MBytes
[ 4] 40.00-41.00 sec 1.25 MBytes     10.5 Mbits/sec 0     1.86 MBytes
[ 4] 41.00-42.00 sec 1.40 MBytes     11.8 Mbits/sec 0     1.88 MBytes
[ 4] 42.00-43.00 sec 1.12 MBytes     9.37 Mbits/sec 0     1.91 MBytes
[ 4] 43.00-44.00 sec 1.12 MBytes     9.40 Mbits/sec 0     1.96 MBytes
[ 4] 44.00-45.00 sec 1.20 MBytes     10.1 Mbits/sec 0     2.02 MBytes
[ 4] 45.00-46.00 sec 1.27 MBytes     10.7 Mbits/sec 0     2.11 MBytes
[ 4] 46.00-47.00 sec 1.30 MBytes     10.9 Mbits/sec 0     2.22 MBytes
[ 4] 47.00-48.00 sec 1.25 MBytes     10.5 Mbits/sec 0     2.36 MBytes
[ 4] 48.00-49.00 sec 1.43 MBytes     12.0 Mbits/sec 0     2.53 MBytes
```

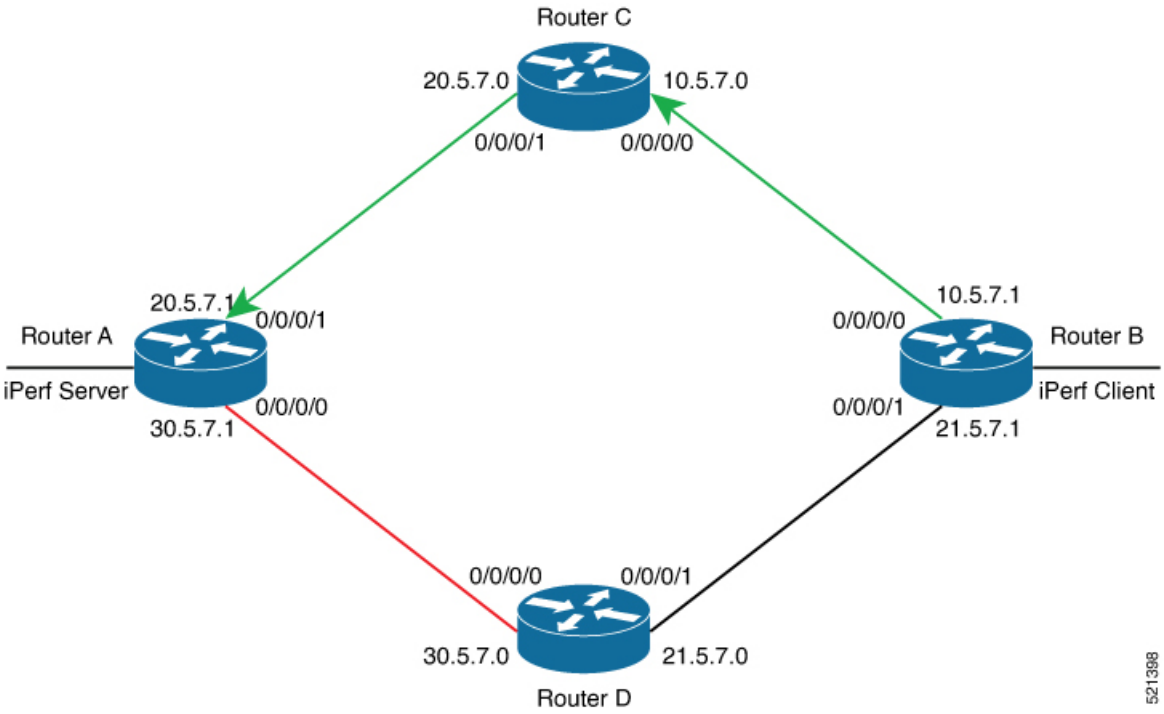
Step 3 Bring down the interface on Router D using the **shut** command to trigger a route update.

```
Router(config)#interface FourhundredGig0/0/0/0
Router(config-if)#shut
Router(config-if)#commit
```

Note Because of the interface shutdown, the route to 30.5.7.1 needs to be updated and hence momentarily there will be no route to this address.

Step 4 During the route update, check the network performance by executing the **show appmgr application name** *app_name* **logs** command.

You will notice that the entries in the **Bandwidth** field is Zero for a short duration, when the new route is installed.




```
Router#show appmgr application name iperf-client-app logs
Tue Dec 1 12:59:40.349 UTC
Connecting to host 30.5.7.1, port 5201
[ 4] local 100.0.0.9 port 61384 connected to 30.5.7.1 port 5201
15
[ ID] Interval          Transfer Bandwidth     Retr     Cwnd
[ 4] 0.00-1.00 sec 1.05 MBytes 8.82 Mbits/sec 0     80.6 KBytes
[ 4] 1.00-2.00 sec 1.26 MBytes 10.6 Mbits/sec 0     136 KBytes
[ 4] 2.00-3.00 sec 1.18 MBytes 9.90 Mbits/sec 0     191 KBytes
[ 4] 3.00-4.00 sec 1.24 MBytes 10.4 Mbits/sec 0     246 KBytes
[ 4] 4.00-5.00 sec 1.18 MBytes 9.90 Mbits/sec 0     301 KBytes
[ 4] 5.00-6.00 sec 1.37 MBytes 11.5 Mbits/sec 0     362 KBytes
[ 4] 6.00-7.00 sec 1.37 MBytes 11.5 Mbits/sec 0     423 KBytes
[ 4] 7.00-8.00 sec 1.43 MBytes 12.0 Mbits/sec 0     486 KBytes
[ 4] 8.00-9.00 sec 1.30 MBytes 11.0 Mbits/sec 0     547 KBytes
[ 4] 9.00-10.00 sec 1.43 MBytes 12.0 Mbits/sec 0     611 KBytes
[ 4] 10.00-11.00 sec 1.62 MBytes 13.6 Mbits/sec 0     707 KBytes
[ 4] 11.00-12.00 sec 1.62 MBytes 13.6 Mbits/sec 0     875 KBytes
[ 4] 12.00-13.00 sec 1.93 MBytes 16.2 Mbits/sec 0     1.07 MBytes
[ 4] 13.00-14.00 sec 1.68 MBytes 14.1 Mbits/sec 0     1.29 MBytes
[ 4] 14.00-15.00 sec 1.06 MBytes 8.86 Mbits/sec 0     1.56 MBytes
[ 4] 15.00-16.00 sec 891 KBytes 7.30 Mbits/sec 0     1.83 MBytes
[ 4] 16.00-17.00 sec 970 KBytes 7.95 Mbits/sec 0     2.12 MBytes
[ 4] 17.00-18.00 sec 1.24 MBytes 10.4 Mbits/sec 0     2.58 MBytes
[ 4] 18.00-19.00 sec 885 KBytes 7.24 Mbits/sec 0     2.65 MBytes
[ 4] 19.00-20.00 sec 1.55 MBytes 13.0 Mbits/sec 0     3.10 MBytes
[ 4] 20.00-21.00 sec 820 KBytes 6.71 Mbits/sec 0     3.10 MBytes
[ 4] 21.00-22.00 sec 1.72 MBytes 14.4 Mbits/sec 6     2.42 MBytes
[ 4] 22.00-23.00 sec 0.00 Bytes 0.00 bits/sec 5     2.30 MBytes
[ 4] 23.00-24.00 sec 256 KBytes 2.10 Mbits/sec 0     1.35 MBytes
[ 4] 24.00-25.00 sec 1.56 MBytes 13.1 Mbits/sec 237     1.83 MBytes
[ 4] 25.00-26.00 sec 1.90 MBytes 15.9 Mbits/sec 0     2.17 MBytes
[ 4] 26.00-27.00 sec 382 KBytes 3.12 Mbits/sec 61     1.95 MBytes
[ 4] 27.00-28.00 sec 0.00 Bytes 0.00 bits/sec 0     1.39 MBytes
[ 4] 28.00-29.00 sec 3.35 MBytes 28.1 Mbits/sec 0     1.52 MBytes
```

```
[ 4] 29.00-30.00 sec 954 KBytes 7.82 Mbits/sec 0     1.58 MBytes
[ 4] 30.00-31.00 sec 1018 KBytes 8.34 Mbits/sec 0     1.64 MBytes
[ 4] 31.00-32.00 sec 1.24 MBytes 10.4 Mbits/sec 0     1.71 MBytes
[ 4] 32.00-33.00 sec 1.25 MBytes 10.5 Mbits/sec 0     1.76 MBytes
[ 4] 33.00-34.00 sec 1.61 MBytes 13.5 Mbits/sec 0     1.80 MBytes
[ 4] 34.00-35.00 sec 1.46 MBytes 12.2 Mbits/sec 0     1.82 MBytes
[ 4] 35.00-36.00 sec 1.18 MBytes 9.89 Mbits/sec 0     1.83 MBytes
[ 4] 36.00-37.00 sec 1.36 MBytes 11.4 Mbits/sec 0     1.84 MBytes
[ 4] 37.00-38.00 sec 1.36 MBytes 11.4 Mbits/sec 0     1.84 MBytes
[ 4] 38.00-39.00 sec 1.24 MBytes 10.4 Mbits/sec 0     1.84 MBytes
[ 4] 39.00-40.00 sec 1.25 MBytes 10.5 Mbits/sec 0     1.85 MBytes
[ 4] 40.00-41.00 sec 1.25 MBytes 10.5 Mbits/sec 0     1.86 MBytes
[ 4] 41.00-42.00 sec 1.40 MBytes 11.8 Mbits/sec 0     1.88 MBytes
[ 4] 42.00-43.00 sec 1.12 MBytes 9.37 Mbits/sec 0     1.91 MBytes
[ 4] 43.00-44.00 sec 1.12 MBytes 9.40 Mbits/sec 0     1.96 MBytes
[ 4] 44.00-45.00 sec 1.20 MBytes 10.1 Mbits/sec 0     2.02 MBytes
[ 4] 45.00-46.00 sec 1.27 MBytes 10.7 Mbits/sec 0     2.11 MBytes
[ 4] 46.00-47.00 sec 1.30 MBytes 10.9 Mbits/sec 0     2.22 MBytes
[ 4] 95.00-96.00 sec 1.48 MBytes 12.4 Mbits/sec 0     1.82 MBytes
[ 4] 96.00-97.00 sec 1.25 MBytes 10.5 Mbits/sec 0     1.83 MBytes
[ 4] 97.00-98.00 sec 1.25 MBytes 10.5 Mbits/sec 0     1.83 MBytes
[ 4] 98.00-99.00 sec 1.49 MBytes 12.5 Mbits/sec 0     1.84 MBytes
[ 4] 99.00-100.00 sec 1.25 MBytes 10.5 Mbits/sec 0     1.86 MBytes
[ 4] 100.00-101.00 sec 1.21 MBytes 10.2 Mbits/sec 0     1.89 MBytes
[ 4] 101.00-102.00 sec 1.34 MBytes 11.2 Mbits/sec 0     1.94 MBytes
[ 4] 102.00-103.00 sec 1.25 MBytes 10.5 Mbits/sec 0     2.01 MBytes
[ 4] 103.00-104.00 sec 1.30 MBytes 10.9 Mbits/sec 0     2.09 MBytes
[ 4] 104.00-105.00 sec 1.25 MBytes 10.5 Mbits/sec 0     2.17 MBytes
[ 4] 105.00-106.00 sec 1.39 MBytes 11.6 Mbits/sec 0     2.33 MBytes
[ 4] 106.00-107.00 sec 1.01 MBytes 8.47 Mbits/sec 0     2.46 MBytes
[ 4] 107.00-108.00 sec 526 KBytes 4.31 Mbits/sec 0     2.54 MBytes
[ 4] 108.00-109.00 sec 0.00 Bytes 0.00 bits/sec 0     2.54 MBytes
[ 4] 109.00-110.00 sec 0.00 Bytes 0.00 bits/sec 0     2.54 MBytes
[ 4] 110.00-111.00 sec 0.00 Bytes 0.00 bits/sec 0     2.54 MBytes
[ 4] 111.00-112.00 sec 0.00 Bytes 0.00 bits/sec 1     1.41 KBytes
[ 4] 112.00-113.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 113.00-114.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 114.00-115.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 115.00-116.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 116.00-117.00 sec 0.00 Bytes 0.00 bits/sec 1     1.41 KBytes
[ 4] 117.00-118.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 118.00-119.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 119.00-120.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 120.00-121.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 121.00-122.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 122.00-123.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 123.00-124.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 124.00-125.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 125.00-126.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 126.00-127.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 127.00-128.00 sec 0.00 Bytes 0.00 bits/sec 1     1.41 KBytes
[ 4] 128.00-129.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 129.00-130.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 130.00-131.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 131.00-132.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 132.00-133.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 133.00-134.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 134.00-135.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 135.00-136.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 136.00-137.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 137.00-138.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 138.00-139.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 139.00-140.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
```

```
[ 4] 140.00-141.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 141.00-142.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 142.00-143.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 143.00-144.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 144.00-145.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 145.00-146.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 146.00-147.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 147.00-148.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 148.00-149.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 149.00-150.00 sec 0.00 Bytes 0.00 bits/sec 0     1.41 KBytes
[ 4] 150.00-151.00 sec 700 KBytes 5.73 Mbits/sec 847    600 KBytes
[ 4] 151.00-152.00 sec 954 KBytes 7.82 Mbits/sec 993    1.32 MBytes
[ 4] 152.00-153.00 sec 509 KBytes 4.17 Mbits/sec 0     1.79 MBytes
[ 4] 153.00-154.00 sec 1.08 MBytes 9.07 Mbits/sec 0     1.85 MBytes
[ 4] 154.00-155.00 sec 1.38 MBytes 11.6 Mbits/sec 0     1.90 MBytes
[ 4] 155.00-156.00 sec 1.55 MBytes 13.0 Mbits/sec 0     1.98 MBytes
[ 4] 156.00-157.00 sec 1.16 MBytes 9.71 Mbits/sec 0     2.04 MBytes
[ 4] 157.00-158.00 sec 1.21 MBytes 10.2 Mbits/sec 0     2.10 MBytes
[ 4] 158.00-159.00 sec 1.26 MBytes 10.6 Mbits/sec 0     2.17 MBytes
[ 4] 159.00-160.00 sec 1.14 MBytes 9.56 Mbits/sec 0     2.23 MBytes
[ 4] 160.00-161.00 sec 1.29 MBytes 10.8 Mbits/sec 0     2.27 MBytes
[ 4] 161.00-162.00 sec 1.24 MBytes 10.4 Mbits/sec 0     2.34 MBytes
[ 4] 162.00-163.00 sec 1.42 MBytes 11.9 Mbits/sec 0     2.41 MBytes
[ 4] 163.00-164.00 sec 1.11 MBytes 9.34 Mbits/sec 0     2.46 MBytes
[ 4] 164.00-165.00 sec 1.39 MBytes 11.7 Mbits/sec 0     2.56 MBytes
[ 4] 165.00-166.00 sec 995 KBytes 8.16 Mbits/sec 0     2.69 MBytes
[ 4] 166.00-167.00 sec 1.88 MBytes 15.7 Mbits/sec 0     2.94 MBytes
[ 4] 167.00-168.02 sec 950 KBytes 7.69 Mbits/sec 0     3.12 MBytes
[ 4] 168.02-169.00 sec 1.79 MBytes 15.2 Mbits/sec 0     3.12 MBytes
[ 4] 169.00-170.01 sec 1.27 MBytes 10.6 Mbits/sec 0     3.12 MBytes
[ 4] 170.01-171.00 sec 1.25 MBytes 10.5 Mbits/sec 23     1.60 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-600.00 sec   704 MBytes  9.84 Mbits/sec  12069   sender
[  4]   0.00-600.00 sec   702 MBytes  9.82 Mbits/sec          receiver

iperf Done.

<!-On Router A!>
Router#show appmgr application name iperf-server-app stats
Thu Dec  3 11:45:47.790 UTC
Application Stats: iperf-server-app
   CPU Percentage: 0.00%
   Memory Usage: 816KiB / 31.23GiB
   Memory Percentage: 0.00%
   Network IO: 0B / 0B
   Block IO: 0B / 0B
   PIDs: 1
<!-On Router B!>
Router#show appmgr application name iperf-client-app stats
Thu Dec  3 11:45:59.418 UTC
Application Stats: iperf-client-app
   CPU Percentage: 0.00%
   Memory Usage: 0B / 0B
   Memory Percentage: 0.00%
   Network IO: 0B / 0B
   Block IO: 0B / 0B
   PIDs: 0
```

Stop iPerf Applications

Stop the iPerf applications on Router A and Router B using the **appmgr application stop name** *app_name* command. The **application stop** command can only be used for applications that are registered, activated, and are currently running. The **application stop** command stops only the application and does not clean up the resources used by the application.

You can verify the status of the application using the **show appmgr application-table** command. The **Status** is displayed as **Exited** if the application has been stopped successfully.

```
Router#appmgr application stop name iperf-server-app
Mon Nov 30 13:38:36.202 UTC
Router#show appmgr application-table
Mon Nov 30 13:38:36.999 UTC
Name                 Type    Config State  Status
--------             ------- ------------- -------------------------
iperf-server-app     Docker    Activated   Exited (1) Less than a se
Router#
```

Start iPerf Applications

Start or restart an application that has been stopped (and not deactivated) using the **appmgr application start name** *app_name* command.

```
Router#appmgr application start name iperf-server-app
Tue Dec 1 13:06:21.996 UTC
Router#show appmgr application-table
Mon Nov 30 13:38:36.999 UTC
Name                 Type    Config State  Status
--------             ------- ------------- -------------------------
iperf-server-app     Docker    Activated   UP(1) Less than a second
Router#
```

Deactivate iPerf Applications

Step 1 Deactivate the iPerf applications using the **no appmgr application** *app_name* command. You deactivate the installed application when you want to release all resources used by the application.

```
Router#config
Router(config)#no appmgr application iperf-server-app
Router(config)#commit
```

Step 2 Verify the status of the application by using the **show appmgr application-table** *app_name* **stats** command.

```
Router#show appmgr application-table
Mon Nov 30 13:39:51.197 UTC
Router#
```

Note You can activate a deactivated application using the **appmgr application** *app_name* **activate type docker source** *source_name* command.

Uninstall iPerf Applications

Uninstall the applications using the **appmgr package uninstall package** *package_name* command.

After the application is successfully uninstalled, executing the **show appmgr source-table** command displays no result.

```
Router#appmgr package uninstall package iperf
table
Mon Nov 30 13:41:05.155 UTC
Router#show appmgr source-table
Mon Nov 30 13:41:05.936 UTC
Router#
```