Use Cases: Application Hosting

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

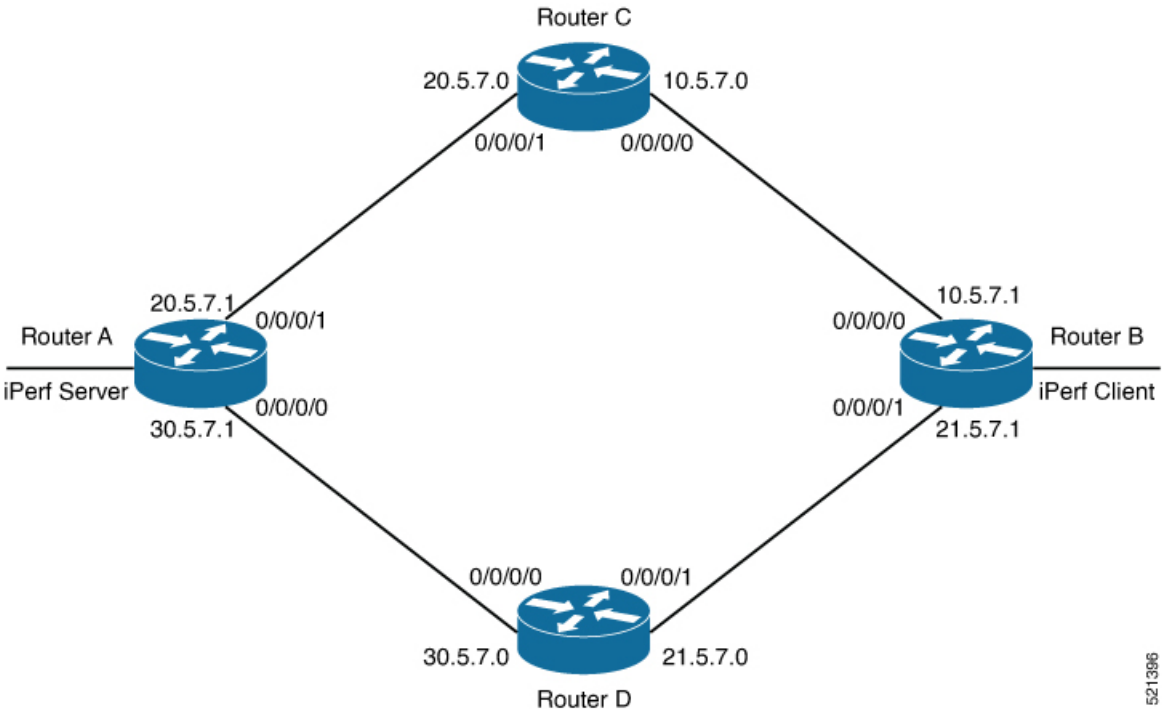
Hosting iPerf in Docker Containers to Measure Network Performance using Application Manager

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 1: 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.

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#
```

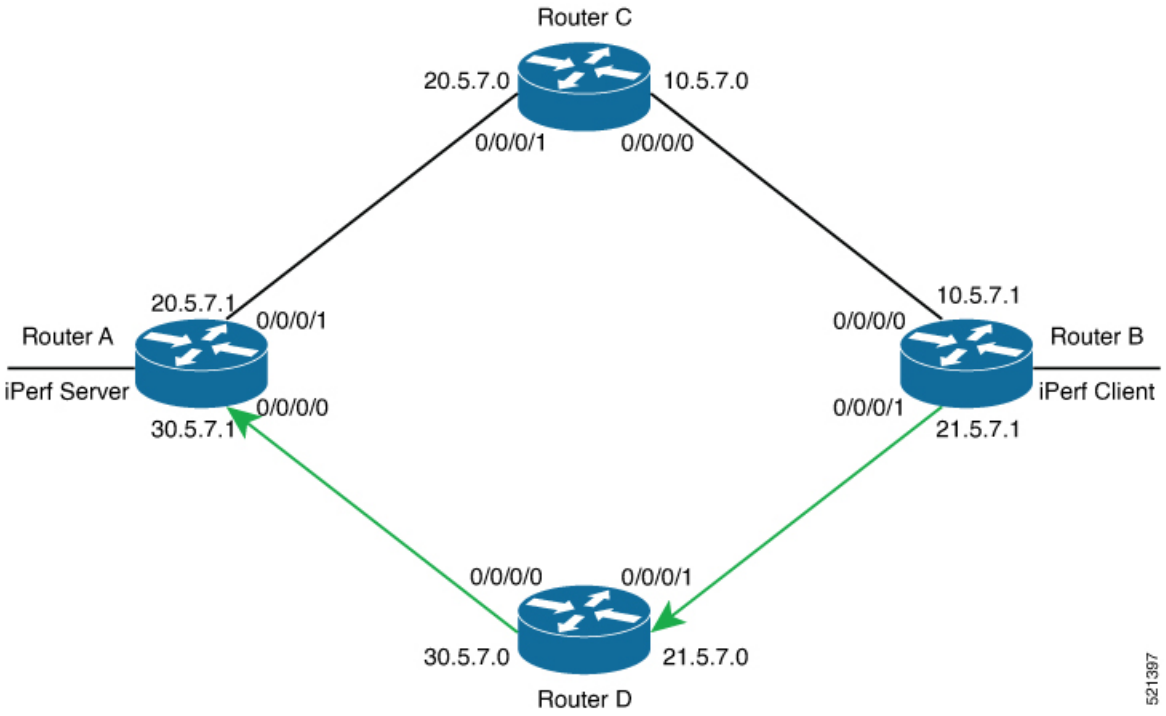
Verify Connection between the iPerf Server and iPerf Client Applications

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
```

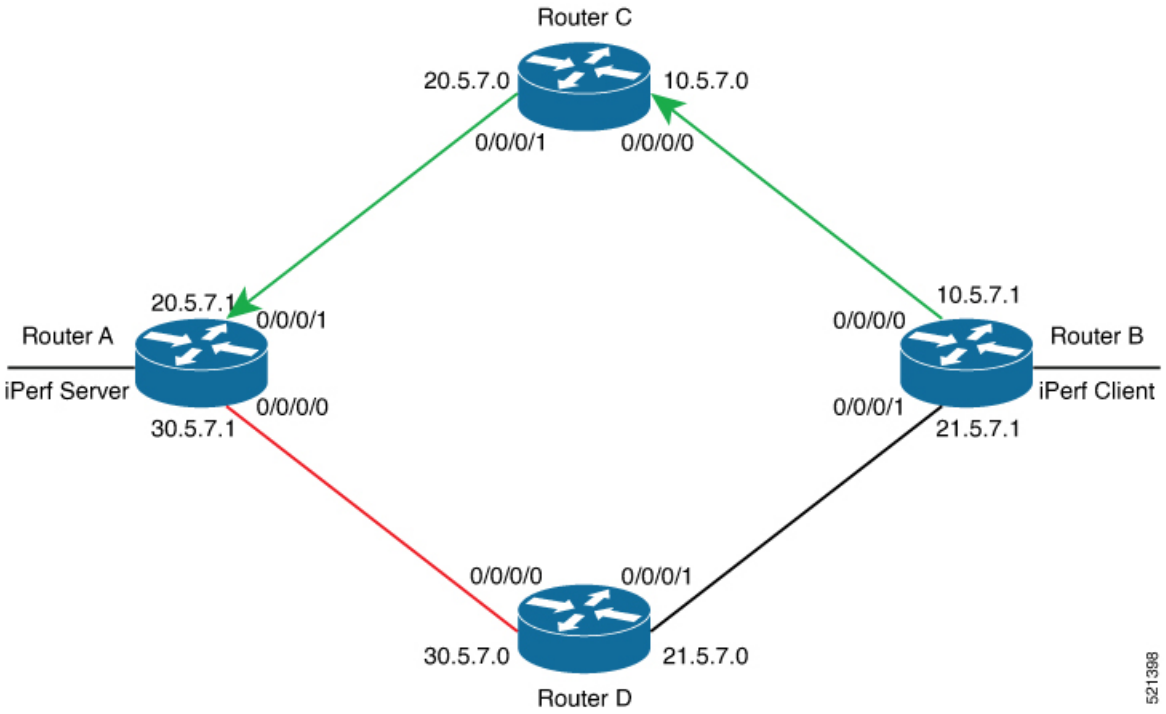
Step 3 Bring down the interface on Router D using the **shut** command to trigger a route update.

```
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#
```

CPU-Based Packet Generator

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
CPU-Based Packet Generator on NCS 5700 fixed port routers	Release 24.2.11	Introduced in this release on: NCS 5700 fixed port routers This feature support is now extended to NCS 5700 fixed port routers.

Feature Name	Release Information	Feature Description
CPU-Based Packet Generator	Release 24.2.1	Introduced in this release on: NCS 5500 fixed port routers; NCS 5500 modular routers(NCS 5500 line cards; NCS 5700 line cards [Mode: Compatibility; Native]) You can now use a CPU-based packet generator for IOS-XR routers to simplify the diagnostic process for routers experiencing problems. This tool allows you to generate a wide range of traffic streams directly within the production environment without physically isolating the routers and moving them to a lab setup. This tool is beneficial in environments that use routers from different vendors or different models from the same vendor. The feature introduces the CLI Options command with different options to generate different types of packets.

Need for CPU-Based Packet Generator

Diagnosing network problems in production environments, such as traffic drops and mis-forwarding issues, is crucial for network management. Traditionally, routers are physically isolated for debugging, requiring moving equipment into lab environments with traffic generators.The CPU-Based Packet Generator can be used in the production environment, eliminating the need to isolate the routers to a lab environment for troubleshooting purposes.

Benefits of CPU-Based Packet Generator

- Versatile Traffic Crafting: Create complex nested packets, such as IPinIPinIPinIP, to test and diagnose a variety of scenarios.
- In-Production Diagnosis: Directly diagnose routers in a problem state without disrupting the network setup.

Restrictions of CPU-Based Packet Generator

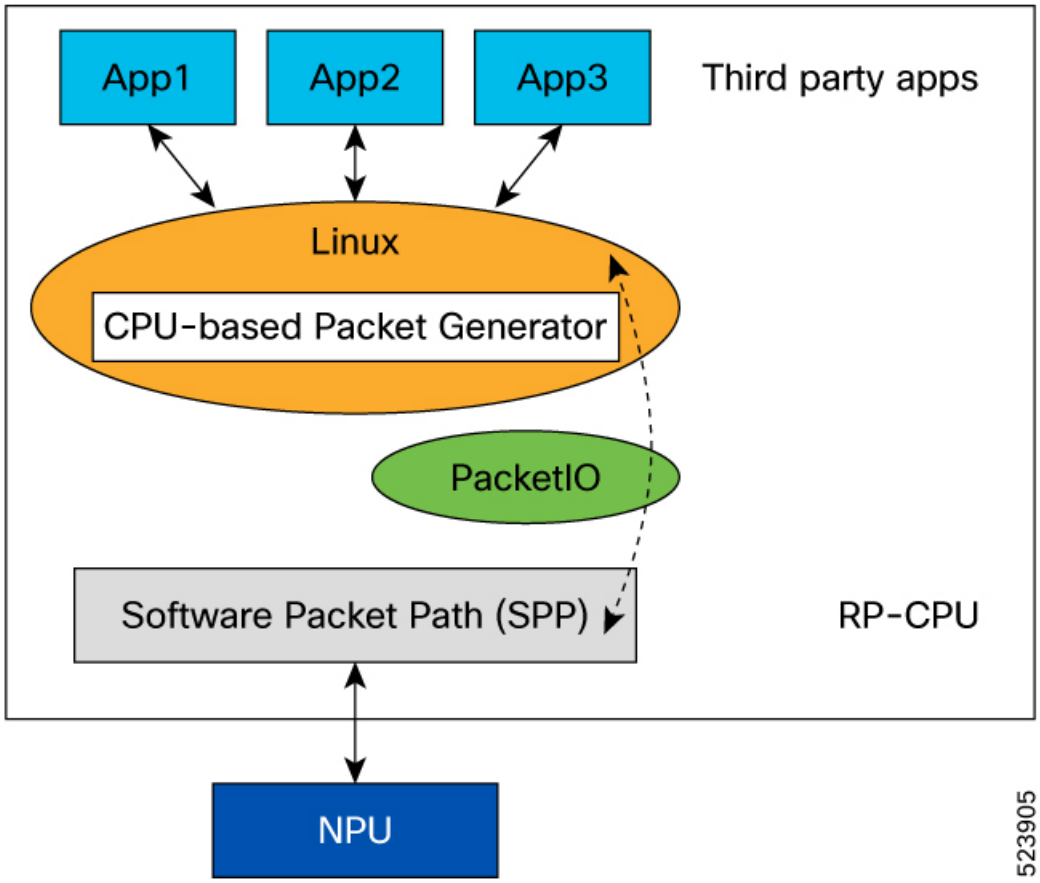
- CPU-based packet generators are not optimized for high-speed packet processing; therefore, they may not match the performance of NPU-based packet generators.
- CPU-based packet generators can potentially introduce higher CPU loads during operation, which may affect the router performance.

- The probe packet rate is 80 kpps.

Topology of CPU-Based Packet Generator

The following diagram depicts the software architecture of CPU-based packet generator.

Figure 2: Architecture of CPU-Based Packet Generator



The Cisco IOS-XR PacketIO serves as a host for third-party applications on the XR platform, with PacketIO infrastructure facilitating packet transport and interactions between Linux and XR environments. Leveraging this existing infrastructure, the CPU-based packet generator is implemented as a Linux application and packaged within the supported XR platform base image, ensuring seamless distribution.

The Linux infrastructure maintains a database of all XR interfaces including bundles. The CPU-based packet generator is used to send a specific packet type over a chosen interface.

Capabilities of CPU-based Packet Generator

- **Support different packet types:** The CPU-based packet generator supports various packet types, including:
  - ARP

  - TCP
  - UDP
  - GRE
  - MPLS
  - IPinIP
  - ICMPv4
  - ICMPv6

- **Corrupt or error packet generation:** There are times when routers receive packets that are either corrupted or contain errors for various reasons. To identify and troubleshoot these issues, it becomes necessary to generate similar packets that can be used for debugging purposes. The CPU-based packet generator can create these packets and aid debugging.

  Examples include:

  - IPv4 packet with TTL 0
  - IPv4 packet with wrong checksum
  - IPv4 packet with mismatch between IP option length field and the IP header

How to Use CPU-based Packet Generator?

You can use CPU-based packet generator using:

- **CLI:** Use the **packetgen** command with different options to run the tool from XR bash environment. As the XR interfaces show up as Linux interfaces in bash environment, you can directly use the XR interface names.
- **pcap file:** Use an already captured pcap file in production routers and replay it.

  packetgen -i interface_name -pcap pcap_file

CLI Options

The following table outlines the different options available for the **packetgen** command.

Table 2: Packetgen CLI Options

Option	Description
-accounting	Turn on accounting for packets. Only works if packets come back to the packet generator.
-arp-destination-hw-address string	ARP target hardware address (default: uses interface MAC address)
-arp-destination-ip-address string	ARP target IP address (default: 127.0.0.1 or ::1)

Option	Description
-arp-operation uint	ARP operation (1: request, 2: reply , 3: rarp)
-arp-source-hw-address string	ARP sender hardware address (default : uses interface MAC)
-arp-source-ip-address string	ARP sender IP address (default: uses interface IP)
-burst int	Number of packets to be injected at a time. To be used in conjunction with -sleep.
-count int	Number of packets to be generated.
-data-type string	constant, incrementing, random (default: no payload)
-ethernet-dmac string	Destination MAC address (default: ff:ff:ff:ff:ff:ff)
-ethernet-smac string	Source MAC address (default: use interface MAC address)
-file string	Write packets to file
-gre	Enable GRE
-gre-checksum-present	Enable GRE checksum present bit
-gre-key-present	Enable GRE key present bit
-gre-over-mpls	Enable GRE over MPLS
-gre-protocol uint	Set the protocol type of the GRE payload (default: 0x0800 (IP)
-gre-seq-present	Enable GRE sequence number present bit
-gre-version uint	Set the GRE version number (default 0)
-header string	Custom header for all packets
-hex	Print hex dump of packets
-i string	Interface name for packet injection
-icmp-code uint	ICMP code (default: 0)
-icmp-type uint	ICMP type (default: 0)
-inc-dmac	Increment destination MAC
-inc-smac	Increment source mac
-inner-ethernet-dmac string	Inner Ethernet destination MAC address (default: ff:ff:ff:ff:ff:ff)
-inner-ethernet-smac string	Inner Ethernet source MAC address (default: ff:ff:ff:ff:ff:ff)

Option	Description
-inner-ip-checksum uint	Inner IP checksum (default: compute checksum automatically)
-inner-ip-dont-fragment uint	Set inner IP Don't Fragment flag as 1
-inner-ip-dst string	Inner destination IP address (default: 127.0.0.1 or ::1)
-inner-ip-flow-label uint	Inner IPv6 Flow Label value (default: 0)
-inner-ip-frag-offset uint	Inner IP fragment offset in units of 64-bits (e.g. 1 = 64 bits)
-inner-ip-protocol string	Inner IP protocol . Supports protocol text (TCP, UDP) and code (63 for TCP) (default: TCP)
-inner-ip-src string	Inner source IP address (default: 127.0.0.1 or ::1)
-inner-ip-tos uint	Inner IP Type Of Service (TOS) value (default: 0)
-inner-ip-traffic-class uint	ip-traffic-class (traffic-class) value (default: 0)
-inner-ip-ttl uint	Inner IP time to live (ttl). (Default ttl = 64
-inner-ip-version int	Inner IP version (default: 4)
-inner-vlan-id uint	Inner VLAN id (default: 0)
-inner-vlan-tpid uint	Inner VLAN ethernet type (default: 33024 :Dot1Q)
-inner-vlan-vpri uint	Inner VLANpriority (default: 0
-ip-checksum string	IP checksum (default: compute checksum automatically)
-ip-dont-fragment string	Set IP flag -ip-dont-fragment 0 -> 000 Nothing set -ip-dont-fragment 1 -> 001 More Fragments -ip-dont-fragment 2 -> 010 Dont Fragment -ip-dont-fragment 4 -> 100 set reserved bit
-ip-dst string	Destination IP address (default: 127.0.0.1 or ::1)
-ip-flow-label string	IPv6 Flow Label value (default: 0)
-ip-frag-offset string	Fragment offset in units of 64-bits (1 = 64 bits)
-ip-protocol string	IP protocol. Supports protocol text (TCP, UDP, GRE, VXLAN, ICMP, NDP) and code (63 for TCP) (default: TCP)
-ip-src string	Source IP address (default: use interface ip)
-ip-tos string	IP Type Of Service value (default: 0)
-ip-traffic-class string	IP traffic class (traffic-class) value (default: 0)

Option	Description
-ip-ttl string	IP time to live (ttl). (Default ttl = 64
-ip-version string	IP version should always be set for accurate IP packet creation, ip version (default: 4).
-mpls-exp string	Comma separated MPLS EXP (Experimental) value (default: 0)
-mpls-label string	Comma separated list of Multiprotocol Label Switching (MPLS) labels to be added to the packet. Specified from top to bottom
-mpls-ttl string	Comma separated MPLS TTL (Time To Live) value (default: 64)
-ndp string	Specify the neighbor discovery protocol: nbr-solicit, nbr-advt
-ndp-target-address string	NDP target address (default: for advertisement source IP, for solicitation destination IP
-pcap string	File to replay pcap
-progress	Display a progress bar
-seed int	Seed for pseudo random payload generator
-size int	Size of payload
-sleep string	Time duration to sleep during each burst. To be used together with -burst.
-stdout	Print packets to stdout
-tcp-dport int	TCP destination port (default: 40000)
-tcp-flags string	Set TCP control flags: • U (Urgent): Indicates that the data should be processed urgently. • A (Acknowledgement): Acknowledges the receipt of data. • P (Push): Instructs the sender to push the data to the receiving application immediately. • R (Reset): Resets the connection. • S (Synchronize): Synchronizes sequence numbers to initiate a connection. • F (Finish): Indicates the sender has finished sending data and wants to terminate the connection.
-tcp-sport int	TCP source port (default: 40000)
-udp-dport int	UDP destination port (default: 40000)
-udp-sport int	UDP source port (default: 40000)
-vlan-id uint	VLAN id (default: 0)

Option	Description
-vlan-tpid uint	VLAN ethernet type (default: 33024 :Dot1Q)
-vlan-vpri uint	VLAN priority (default: 0
-vxlan-udp-dport int	UDP destination port for VXLAN (default: 4789)
-vxlan-udp-sport int	UDP source port for VXLAN (default: 0)
-vxlan-vni uint	VXLAN VNI (default: 0)

Sample Commands

This section lists sample commands for some common packet types.

Table 3: Sample Packetgen Commands

Packet Type	Sample Command
ARP	packetgen -i enp0s8 -ip-ttl 32 -arp-operation 1 -progress -count 10000 -inc-smac -arp-destination-ip-address 192.168.56.1
TCP	packetgen -i enp0s8 -ip-ttl 32 -tcp-sport 40000 -progress -count 10000 -inc-smac
UDP	packetgen -i enp0s8 -ip-ttl 32 -udp-sport 40000 -progress -count 10000 -inc-smac
ICMP - PING	packetgen -i enp0s8 -ip-ttl 32 -icmp-type 8 -progress -count 10000 -ip-dst 192.168.56.1
GRE	packetgen -i enp0s8 -ip-ttl 32 -gre -count 100 -inner-ip-ttl 32 -tcp-sport 3222 -progress
IP in IP	packetgen -i enp0s8 -count 100 -tcp-sport 3222 -progress -ip-src="1.1.1.1,2.2.2.2"
ETHER-IP	packetgen -i enp0s8 -ip-ttl 32 -count 100 -inner-ip-version 6 -tcp-sport 3222 -progress -inner-ethernet-smac ff:ff:ff:ff:ff:ff
VLAN	packetgen -i enp0s8 -ip-ttl 32 -tcp-sport 40000 -progress -count 10000 -inc-smac -vlan-id 2
QinQ	packetgen -i enp0s8 -ip-ttl 32 -tcp-sport 40000 -progress -count 10000 -inc-smac -vlan-id 2 -inner-vlan-id 2
VXLAN	packetgen -i enp0s8 -ip-ttl 32 -tcp-sport 40000 -progress -count 10000 -inc-smac -vxlan-vni 3 -vxlan-udp-sport 4444 -inner-ip-version 4 -inner-ethernet-smac ff:ff:ff:ff:ff:ff -data-type constant
NDP	packetgen -i enp0s8 -ip-version 6 -ndp nbr-advt -count 100 -ip-checksum 1 -progress
MPLS	packetgen -i enp0s8 -ip-version 4 -mpls-label 1,2,3,4,5 -tcp-sport 4556 -count 1000 -progress

Command Example

This section shows an example command to send an ICMP ping request from source address 10.0.0.1 to destination address 10.0.0.2 via interface Hu0_0_0_25.

```
Router# bash
[ios:~]$ packetgen -i Hu0_0_0_25 -ip-ttl 32 -progress -count 50 -icmp-type 8 -ip-dst 10.0.0.2
 -ip-src 10.0.0.1 --ethernet-smac 78:c5:51:84:48:c4 --ethernet-dmac 00:00:00:1e:ca:fc
INFO[0000] [ETH IP ICMP]
INFO[0000] Setting SRC IP to 10.0.0.1
INFO[0000] Setting DST IP to 10.0.0.2
INFO[0000] Opening Handle Hu0_0_0_25
INFO[0000] Opened Handle Hu0_0_0_25
INFO[0000] Starting Packet Injection
Sending Packets... 2% | | (1/50, 254 packet/s) [0s:0s] /* Truncated output. */

Address Age Hardware Addr State Type Interface
10.0.0.1 - 78c5.5184.48c4
Interface ARPA HundredGigE0/0/0/25
10.0.0.2 00:50:23 0000.001e.cafc Dynamic ARPA HundredGigE0/0/0/25

Source stats:
Stat Name      Port Name            Control Packet Tx.  Control Packet Rx.  Ping Reply Tx.
20.0.0.2/
Card01/Port01  Ethernet - VM - 001  51                  51                  50

Interface stats:
Input    Punt XIPC  InputQ     XIPC         PuntQ
ClientID Drop/Total Drop/Total Cur/High/Max Cur/High/Max
-------------------------------------------------------------------------------
ipv6_icmp 0/0        0/0       0/0/1000     0/0/1000
icmp      0/50       0/0       0/15/1000    0/0/1000
```