BGP RR Scale Considerations and KPI Monitoring

Contents

Introduction

This document describes the major contributors to the maximum scale a Border Gateway Protocol (BGP) Route-Reflectors (RR) can achieve and guidance on BGP RR performance monitoring.

HW/SW Platform Selection

A high-scale BGP RR is typically not in the forwarding path of packets carrying services provided by an Internet Service Provider. Therefore, the hardware requirements for a BGP RR and routers that are predominantly forwarding packets in the data path are different. Standard routers are built with a powerful data-path forwarding element and a comparatively moderate control-path element. A BGP RR performs all its tasks in a control plan.
Within the Cisco IOS® XR family of products, you can choose between 3 flavours of HW/SW platforms for a BGP RR role:

Physical Cisco IOS XR Router	Cisco IOS XRv 9000 Appliance	Cisco IOS XRv 9000 Router (aka XRv9k)
• Moderate control plane capacity (typically between 2 and 6 CPU cores allocated to RP XR VM) • Unused data-path capacity	• High control plane capacity (on Cisco UCS M5-based appliance 36 CPU cores are dedicated to RP XR VM) • Equal split between data-path and control-path capacity. • XRv9k image runs on barebone for maximum performance	• Customisable control plane capacity • Equal split between data-path and control-path power when using BGP RR image. • An additional layer of virtualisation impacts performance.

As of this writing, XRv9k Appliance is the optimal platform choice for BGP RR because it provides for highest control plane capacity with maximum performance.

Scale And Performance Considerations

The supported scale of data-plane entities is relatively easy to express because the performance of the data-path element seldom depends on the scale. For example, a TCAM lookup takes the same time regardless of the number of active TCAM entries.
The supported scale of control-plane entities is often much more complex because the scale and performance are interconnected. Consider a BGP RR with 1M routes. The work that a BGP process must carry out to maintain this BGP table depends on:

1. How many BGP peers are active?
2. What address families are active?
3. How are they distributed into update groups?
4. The complexity of RPLs (Route Policies)
5. Frequency of updates (incoming updates and also outgoing updates - advertisement interval).
6. TCP MSS, Interface/Path MTU - tuning this will aid in better performance
7. If dual-RP, is NSR enabled
8. Any known slow-peers, that aren't in separate update-group
9. Nexthop trigger-delay value

Number Of BGP Peers

The number of BGP peers is usually the first and unfortunately, often the only thing that comes to mind when considering the BGP scale. While the supported BGP scale cannot be represented without mentioning the number of BGP peers, it is not the most important factor. Many other aspects are equally relevant.

Address Families

The type of address family (AF) is an important factor in BGP performance considerations because in typical deployments it impacts the size of a single route. The number of IPv4 routes that can be packed into a single TCP segment is significantly higher than the number of VPNv4 routes. Hence for the same scale of BGP table changes, an IPv4 BGP RR has less work to do compared to a VPNv4 BGP RR. Obviously, in deployments where a significant number of communities is added to every route, the difference between AFs becomes less significant, but the size of a single route is then even bigger and requires consideration.

Number Of Update Groups

The BGP process prepares a single update for all members of the same update group. Then TCP process splits the update data into a required number of TCP segments (depending on TCP MSS) towards each member of the update group. You can see the active update groups and their members by using the show bgp update-group command. You can influence which and how many peers are members of an update group by creating a common outbound policy for a group of peers you want to be in the same update group. A single update sent by the BGP RR to a high number of BGP RR clients can trigger a burst of TCP ACKs that can be dropped in Local Packet Transport Service (LPTS) component of Cisco IOS XR routers.

Complexity of RPLs (Route Policies)

The complexity of route policies used by BGP impacts the BGP process performance. Every received or sent route must be evaluated against the configured route policy. A very long policy requires many CPU cycles to be spent on this action. A route policy that includes a regular expression is especially heavy on processing. A regular expression helps you express the route policy in a lesser number of lines but requires more CPU cycles while processing than the equivalent route policy that does not use regular expression.

Frequency Of Updates

The frequency of updates has an important bearing on the BGP scale. The number of updates is often difficult to predict. You can influence the frequency of updates by using the "**`advertisement-interval`**" command, which sets the minimum interval between the sending of BGP) routing updates. The default value towards iBGP peers is 0 seconds and 30 towards eBGP peers it is 30 seconds.

TCP MSS And Interface/Path MTU

Splitting an update into many TCP segments can put a lot of strain on TCP process resources in a high-scale and high-update frequency environment. A bigger path MTU and bigger TCP MSS are better for BGP and TCP performance.

NSR On Dual-RP Routers

NSR is a great feature for redundancy, but it does have an impact on BGP performance. On Cisco IOS XR routers both RPs are receiving simultaneously every BGP update directly from the NPU on the ingress line card, which means that the Active RP does not have to spend time on replicating the update to the Standby RP. However, every update generated by the Active RP must be sent to the Standby RP and from there to the BGP peer. This allows the Standby RP to be always up to date on the sequence and acknowledgement numbers but does have an impact on the overall BGP performance. This is why it is recommended that a BGP RR is a single-RP router.

Slow Peers

A slow peer can slow down the updates towards all members of the update group because the BGP process must keep the update in its memory until all peers have acknowledged it. If you know that some peers are much slower (for example routers in a legacy part of the network), separate them upfront into an update group. By default, Cisco IOS XR reports a slow peer via syslog message. You can create static slow peers (that never share the update group with others) or fine-tune the dynamic slow peer behaviour by using the BGP `slow-peer` configuration command in global or per-neighbour config mode. A good further read on this can be found in Troubleshoot Slow BGP Convergence Due to Suboptimal Route Policies on IOS-XR on the Cisco xrdocs.io portal.

Nexthop trigger-delay

If multiple BGP next-hops change in a short time interval and the critical nexthop trigger-delay value of zero is configured in an address family (AF) with a high number of routes, a full walk of the AF must be executed on every next-hop change event. Repeated walks of that AF increase the convergence time in address families with lower critical nexthop trigger-delay values. You can see the next-hop trigger-delay values by running the "show bgp all all nexthops" command.

Example Of Validated Multidimensional BGP RR Scale

Multidimensional scale results, especially for control plane features, are highly dependent on the specific test environment. Test results can vary significantly if some of the parameters are changed.

Parameter	Value	Value
Platform	XRv9k Appliance (UCS M5 based)	ASR9902
IOS XR release	7.5.2 + umbrella SMU for Cisco bug ID CSCwf09600. (constituents of this umbrella SMU are integrated in Cisco IOS XR release 7.9.2 and later)	7.11.2
Peers	VPNv4 eBGP: 2500 VPNv4 iBGP: 1700	VPNv4 iBGP: 2000
BGP Routes	Per session: 200 Total: 400k	Per session: 750 VPNv4: 1.36M

	Paths per route: 1	VPNv6: 150k IPv4: 950k IPv6: 200k Total: ~2.6M Paths per route: 1
IGP Routes	10k (ISIS)	10k (ISIS)
BGP update groups	1	1
BGP Timers	default	default
LPTS BGP-known policer rate	50,000	25,000
tcp num-thread configuration	16 16	16 16
BGP send-buffer-size	default	default
Key Performance Indicators (KPI) summary	• Test case with highest input and output packet rate: ◦ Input: 49.4kpps ◦ Output: 95kpps ◦ ==> LPTS drops (policer at 50kpps) ◦ ==> No drops in NetIO clients ◦ ==> Max XIPC Queue size (BGP): 1362 ◦ ==> Max XIPC Queue size (TCP): 1248	• Test case with highest input packet rate: ◦ Input: 16030 pkts/s ◦ Output: 31 pkts/s ◦ ==> No drops in LPTS nor NetIO clients ◦ ==> Max XIPC Queue size (BGP): 378 ◦ ==> Max XIPC Queue size (TCP): 1021 • Test case with highest output packet rate: ◦ Input: 12172 pkts/s ◦ Output: 23465 pkts/s ◦ ==> No drops

		in LPTS nor NetIO clients ◦ ==> Max XIPC Queue size (BGP): 109 ◦ ==> Max XIPC Queue size (TCP): 1518

Design Considerations

There are two approaches to BGP RR placement in the network:

1. Centralised/Flat BGP RR design.
2. Distributed/Hierarchical BGP RR design.

In a centralised/flat design, all BGP RR clients in the network establish BGP peering with a set (usually a pair) of BGP RR devices that hold the exact same information. This approach is simple to implement and works well in small to moderate scale networks. Any change in BGP table is propagated quickly to all BGP RR clients. As the number of BGP RR clients grows, the design can reach a scale limit when the number of TCP connections on the BGP RR devices grows to the extent where their performance is impacted.

In a distributed/hierarchical design, network is split into several regions. All routers in a region establish BGP peering with a set (usually a pair) of BGP RR devices that hold the exact same information. These BGP RR devices act as BGP RR clients to another set (usually a pair) of BGP RR devices. This design approach allows for easy network expansion, while keeping the number of TCP connections on every single BGP RR under certain limit.
Another design consideration is tailoring the scope of recipients of BGP updates. Depending on the VRF distribution among BGP RR clients, it’s worth considering the RT Constrained Route Distribution. If all BGP RR clients have interfaces in the same VRF, RT Constrained Route Distribution does not bring many benefits. However, if VRFs are sparsely distributed among all BGP RR clients, use of RT Constrained Route Distribution significantly reducesƒ the load on the bgp process on the BGP RR.

Monitor BGP Key Performance Indicators (KPI)

Monitoring of BGP RR’s Key Performance Indicators (KPI) is important for ensuring proper network operation.

A significant change in the network topology (for example a major DWDM link flap) can trigger routing updates that generate excessive traffic towards and/or from the BGP RR. Significant traffic hitting the BGP RR typically carries:

1. Updates from BGP peers.
2. TCP ACKs generated by the BGP peers, in response to updates sent by BGP RR & vice-versa

This section of the document explains the KPI that need to be monitored on a typical BGP RR and also how to tell which of the two significant BGP traffic types are causing high control plane traffic rate.

Path of BGP packets inside the router can be depicted as follows:

Punt
Ethernet controller -(packet)-> datapath forwarder -(packet)-> LPTS -(packet)-> SPP -(packet) -> NetIO -(packet)-> TCP -(message)-> BGP
Inject
BGP -(message)-> TCP -(packet)-> NetIO -(packet)-> SPP -(packet) -> datapath forwarder -(packet)-> Ethernet controller

KPIs can be split into:

Essentials:

1. Datapath Forwarder
2. LPTS (hardware punt policers settings, accept counters and drop counters)
3. SPP
4. NetIO
5. IPC queues (NetIO <==> TCP <==> BGP)
6. BGP InQ/OutQ sizes

Optional:

1. CPU utilisation
2. Memory utilisation
3. TCP statistics
4. BGP process performance
5. BGP convergence

Monitor Datapath Forwarder

On XRv9000 the datapath forwarder is the Data Plane Agent (DPA), while on ASR9000 platforms it is the Network Processor (NP).

Monitor XRv9000 Data Plane Agent (DPA)

Useful command to see the load and statistics of the DPA is:

```
show controllers dpa statistics global
```

This command shows all the non-zero counter, that give you insight into the type and number of packets punted from network interfaces to RP CPU, injected from RP CPU towards network interfaces, and the numer of packets dropped:

<#root>

```
RP/0/RP0/CPU0:xrv9k-01#

show controllers dpa statistics global

Index  Debug                                                        Count
-----------------------------------------------------------------------------
  350  TBPG L2 mailbox events                                           1

Index  Punt                                                         Count
-----------------------------------------------------------------------------
 1500  CDP                                                          46790
 1503  ARP                                                            212
 1611  IFIB                                                        595305
 1776  LLDP                                                         94037
 2001  IPv4 incomplete Tx adjacency                                     2

Index  Inject                                                       Count
-----------------------------------------------------------------------------
  744  CLNS multicast from fabric pre-route                        321250
  749  IPv4 from fabric                                            273993
  765  Inject to fabric                                            595245
  766  Inject to port                                              141033

Index  Drop                                                         Count
-----------------------------------------------------------------------------
  416  Egress UIDB in down state                                        1
  474  IPv4 egress UIDB down                                            2
  673  Pre-route PIT lookup missed                                      2
```

Monitor ASR9000 Network Processor (NP)

Useful commands to see the load and statistics of each NP in the system are:

```
show controllers np load all

show controllers np counters all
```

NP on ASR9000 has a rich set of counters that show you the number, rate and type of processed and dropped packets,.

<#root>

```
RP/0/RSP0/CPU0:ASR9k-B#

show controllers np load all

                Node: 0/0/CPU0:
----------------------------------------------------------------
           Load                Packet Rate
  NP0:  0% utilization             937 pps
  NP1:  0% utilization             538 pps
RP/0/RSP0/CPU0:ASR9k-B#
```

<#root>

```
RP/0/RSP0/CPU0:ASR9k-B#
```

show controllers np counters all

```
                Node: 0/0/CPU0:
----------------------------------------------------------------
Show global stats counters for NP0, revision v4
Last clearing of counters for this NP: NEVER
Read 92 non-zero NP counters:
Offset  Counter                                               FrameValue   Rate (pps)
-------------------------------------------------------------------------------------
  16  MDF_TX_LC_CPU                                             25475368          10
  17  MDF_TX_WIRE                                              681957877         267
  21  MDF_TX_FABRIC                                            683500690         267
  33  PARSE_FAB_RECEIVE_CNT                                    681767730         267
  37  PARSE_INTR_RECEIVE_CNT                                  1323024637         517
  41  PARSE_INJ_RECEIVE_CNT                                     13949634           5
  45  PARSE_ENET_RECEIVE_CNT                                   677655725         265
  49  PARSE_TM_LOOP_RECEIVE_CNT                                 49331192          19
  53  PARSE_TOP_LOOP_RECEIVE_CNT                                 1520846           1
 109  RSV_DROP_EGR_UIDB_NO_MATCH                                      10           0
 146  RSV_DROP_IPV4_RXADJ_DROP                                         1           0
 164  RSV_DROP_ING_LAG_NO_MATCH                                        3           0
 241  RSV_DROP_MPLS_LEAF_NO_MATCH                                   1312           0
<. . .>
```

Monitor LPTS

As a standard BGP RR is not in the forwarding path, all packets received on network interface are punted to control-plane. The data-path element on a BGP RR performs a small number of simple operations before packets are punted to control-plane. As the data-path element is unlikely to be a point of congestion, the only element on the line card that needs monitoring are the LPTS stats.

Please note that in case of XRv9k, hardware statisitics map to the vPP

Command:

```
  show lpts pifib hardware police location <location> | inc "Node|flow_type|BGP"
```

Example:

```
RP/0/RP0/CPU0:xrv9k-01#sh lpts pifib hardware police location 0/0/CPU0 | i "Node|flow_type|BGP"
             Node 0/0/CPU0:
flow_type     priority sw_police_id hw_policer_addr Cur. Rate  burst static_avgrate avgrate_type AggrAc
BGP-known     high     6            220             50000      1250  2500           Global       164013
BGP-cfg-peer  medium   7            221             4000       1000  2000           Global       355976
BGP-default   low      8            222             3000       750   1500           Global       521238
RP/0/RP0/CPU0:xrv9k-01#
```

What to look for:
If a significant jump in AggDrops against BGP-known flow type is observed, start looking for network

topology changes that have triggered such massive control plane churn.

Telemetry data path:

```
Cisco-IOS-XR-lpts-pre-ifib-oper:lpts-pifib
```

> **Note**: LPTS stats counters can be cleared. Your monitoring system must account for this possibility.

Monitor SPP

SPP is the first enttity on the route processor or line card CPU that receives the packet punted from the NP or DPA via internal fabric, and the last point in the software packet processing before it is handed over to the fabric for injection into the NP or DPA.

Relevant commands for SPP monitoring:

```
show spp node-counters

show spp client
```

The **show spp node-counters** command shows the rate of punted/injected packets and is easy to read and understand. For BGP sessions the relevant coounters are under **client/punt** and **client/inject** on the active RP.

The **show spp client** is more rich in output and gives a more detailed insight into number of packets enqueued/dropped towards clients, as well as the high watermark.

<#root>

```
RP/0/RP0/CPU0:xrv9k-01#

show spp node-counters

0/RP0/CPU0:

socket/rx
                Punted packets:          595305
               Punt bulk reads:               6
           Punt non-bulk reads:          595293
            Management packets:        74200158
         Management bulk reads:         1775930
     Management non-bulk reads:        70031734
-------------------------------
socket/tx
              Injected packets:          595245
            Management packets:       139939168
-------------------------------
xrv9k/classify
      Forwarded to SPP clients:        74795463
-------------------------------

client/inject
          Injected from client:       140534413
              Non-bulk injects:       140534413
-------------------------------
client/punt
              punted to client:        74795371
no client found - send to defa:              92

-------------------------------

0/0/CPU0:
<. . .>
```

<#root>

```
RP/0/RP0/CPU0:xrv9k-01#

show spp client

Sat Apr 20 17:11:40.725 UTC
0/RP0/CPU0:
Clients
=======
```

```
<. . .>
netio, JID 254 (pid 4591)
----------------------------------------------------------
Reconnect Pending: F, Exited: F, Keep Queues: F, Pakman Client: T
Quota Current: 0, Limit: 16384, Drops 0
Queues:
Key          Cur  Max      Enqueues   High Watermark (Timestamp)        Drops
0xffffffff    0   10             0    0 (22:13:52.195 Feb 21 24 UTC)       0
0x03000006    0 2048             0    0 (22:13:52.196 Feb 21 24 UTC)       0
0x03000007    0 3072        414881    1 (23:03:30.721 Feb 21 24 UTC)       0
0x03000008    0 1024             5    1 (13:41:28.389 Mar 13 24 UTC)       0
0x03000009    0 2048        180411    1 (23:03:31.565 Feb 21 24 UTC)       0
```

Monitor NetIO

While the LPTS policer only shows the count of packets accepted or dropped by a corresponding policer, at NetIO level we can see the rate of packets punted to RP CPU. Since on a typical BGP RR the great majority of received packets are BGP packets, the overall NetIO rate indicates very closely the rate of received BGP packets.

<#root>

Command:

show netio rates

Example:

<#root>

```
RP/0/RP0/CPU0:xrv9k-01#

show netio rates


Netio packet rate for node 0/RP0/CPU0
-----------------------------------
Current rate       (updated 0 seconds ago):
  Input:            7845 pkts/s
  Output:          10570 pkts/s
  Driver Output:  10598 pkts/s

 1 minute rate     (updated 0 seconds ago):
  Input:            7825 pkts/s
  Output:          10542 pkts/s
  Driver Output:  10569 pkts/s

 5 minute rate     (updated 0 seconds ago):
  Input:            7997 pkts/s
  Output:          10677 pkts/s
  Driver Output:  10704 pkts/s

RP/0/RP0/CPU0:xrv9k-01#
```

What to look for:

- If a significant jump in NetIO rate is observed, start looking for network topology changes that have triggered such massive control plane churn.

Telemetry data path:

- not applicable as telemetry must stream counter values, not rates. BGP-known LPTS policer accept counter can be used on the Telemetry collector to approximate the average rate of received BGP packets from known peers.

Monitor XIPC Queues

On the punt path packets received by NetIO from LPTS are passed on to TCP and BGP. It's important to monitor these queues:
1. TCP high priority queue though which NetIO delivers packets to TCP
2. BGP control queue
3. BGP data queue

On the inject path packets are created by TCP and passed onto NetIO. It's important to monitor these queues:

1. OutputL XIPC queue

Commands:

```
show netio clients
show processes bgp | i "Job Id"
show xipcq jid <bgp_job_id>

show xipcq jid <bgp_job_id> queue-id <n>
```

Examples:

NetIO to TCP, view from NetIO standpoint:

```
RP/0/RP0/CPU0:xrv9k-01#show netio clients
<. . .>
                   Input                Punt         XIPC InputQ      XIPC PuntQ
ClientID         Drop/Total          Drop/Total      Cur/High/Max     Cur/High/Max
<. . .>
tcp               L 0/340774            0/0          L 0/10/12800        0/0/0
                  H 0/44774091                       H 0/784/12800
```

TCP to NetIO, view from NetIO standpoint::

```
RP/0/RP0/CPU0:xrv9k-01#show netio clients
<. . .>
```

```
XIPC queues              Dropped/Queued    Cur/High/Max
------------------------------------------------------------
OutputL                  124860/9355257    0/14000/14000
```

NetIO to TCP, view from TCP process standpoint:

<#root>

```
RP/0/RP0/CPU0:xrv9k-01#
```

show processes tcp

```
 | i "Job Id"
                  Job Id: 430
RP/0/RP0/CPU0:xrv9k-01#
```

show xipcq jid

```
 430
Mon Apr 17 16:16:11.315 CEST
 Id              Name                  Size   Cur Size   Produced     Dropped     HWM
------ ------------------------------- ------  --------- ----------- ----------- --------
    17 XIPC_xipcq_12_0_9854_6506_i...  60000      0     39010269           0       960
    16 XIPC_xipcq_12_0_9854_6506_i...  24000      0     31518436           0      1364
    15 XIPC_tcp_124                     3200      0            0           0         0
    14 XIPC_tcp_125                     9600      0            0           0         0
    13 XIPC_tcp_psb_0                  25600      0            0           0         0
    10 XIPC_tcp_iq_9                  102400      0      9486010           0       312
    12 XIPC_tcp_iq_8                  102400      0      8892274           0       280
     9 XIPC_tcp_iq_5                  102400      0      8291392           0       289
    11 XIPC_tcp_iq_7                  102400      0      9700123           0       319
     8 XIPC_tcp_iq_6                  102400      0      9378703           0       332
     7 XIPC_tcp_iq_3                  102400      0      7221706           0       261
     6 XIPC_tcp_iq_4                  102400      0      9791070           0       308
     4 XIPC_tcp_ctrl_iq_1               4266      0            0           0         0
     5 XIPC_tcp_iq_2                  102400      0      9699027           0       295
     3 XIPC_tcp_ctrl_iq_0               4266      0       209909           0         9
     2 XIPC_tcp_i1                     12800      0     39460564           0       784
     1 XIPC_tcp_i0                     12800      0       212540           0        10
```

TCP to BGP:

<#root>

```
RP/0/RP0/CPU0:xrv9k-01#
```

show processes bgp

```
 | i "Job Id"
                  Job Id: 1078
RP/0/RP0/CPU0:xrv9k-01#
```

show xipcq jid

```
 1078
Mon Apr 17 16:09:33.046 CEST
 Id              Name                  Size   Cur Size   Produced     Dropped     HWM
------ ------------------------------- ------  --------- ----------- ----------- --------
```

```
     2 XIPC_xipcq_12_0_9854_6506_i...  60000      2    35546667            0     15034
     1 XIPC_xipcq_12_0_9854_6506_i...  24000      0     1369029            0        50
RP/0/RP0/CPU0:xrv9k-01#
```

BGP data queue:

<#root>

RP/0/RP0/CPU0:xrv9k-01#

show xipcq jid

1078

queue-id 1

XIPC_xipcq_12_0_9854_6506_inst_1_data_toapp

:

```
Magic:              12344321
Version:            0
SHM Size:           192392
Owner PID:          9854
Owner JID:          1078
Queue ID:           1
Owner MQ handle:    483
User Context:       0x64
Interrupt Flag:     0
Sent-on-full Flag:  0
Max Queue Size:     24000
Queue Size:         0
Client Queue Size:  24000
High watermark:     50
Last Trigger Sent:  Mon Apr 17 16:11:10.009 CEST
MQ Send Errors:     0

Priority Queues:
      Prio       Size      Drops      Total
---------- ---------- ---------- ----------
    Unspec      24000          0          0
    Normal      24000          0    1396159
    Medium      24000          0          0
      High      24000          0          0
   Crucial      24000          0          0
RP/0/RP0/CPU0:xrv9k-01#
```

BGP control queue:

<#root>

RP/0/RP0/CPU0:xrv9k-01#

show xipcq jid

1078

queue-id

```
 2

XIPC_xipcq_12_0_9854_6506_inst_1_ctrl_toapp:
Magic:              12344321
Version:            0
SHM Size:           480392
Owner PID:          9854
Owner JID:          1078
Queue ID:           2
Owner MQ handle:    485
User Context:       0x64
Interrupt Flag:     0
Sent-on-full Flag:  0
Max Queue Size:     60000
Queue Size:         0
Client Queue Size:  60000
High watermark:     15034
Last Trigger Sent:  Mon Apr 17 16:12:49.483 CEST
MQ Send Errors:     0

Priority Queues:
      Prio       Size      Drops      Total
---------- ---------- ---------- ----------
    Unspec      60000          0          0
    Normal      60000          0   37313633
    Medium      60000          0          0
      High      60000          0          0
   Crucial      60000          0          0
RP/0/RP0/CPU0:xrv9k-01#
```

What to look for:

- there must be no drops in relevant queues
- in XIPC queue stats High watermark (HWM) must not exceed the 50% of queue size

For better tracking of high watermark value evolution, you must clear the high watermark value after every read. Note that this does not clear only the HWM counter but also clears all queue statistics. The format of the command for clearing the XIPC queue statistics is: clear xipcq statistics queue-name <queue_name>

As the queue name often includes the Process ID (PID), the queue name changes after process restart. Some examples of commands for clearing the relevant queues statistics:

```
clear xipcq statistics queue-name XIPC_tcp_i0
clear xipcq statistics queue-name XIPC_tcp_i1
clear xipcq statistics queue-name XIPC_xipcq_12_0_9854_6506_inst_1_data_toapp
clear xipcq statistics queue-name XIPC_xipcq_12_0_9854_6506_inst_1_ctrl_toapp
```

Telemetry path:

- There are no Telemetry sensor paths for XIPC.

Monitor BGP Input And Output Queues

BGP maintains an input and output queue for every BGP peer. Data sits in InQ when TCP has passed it to BGP, but BGP hasn't processed them yet. Data sits in OutQ while BGP waits on TCP to split the data into packets and transmit them. The instantaneous size of BGP InQ/OutQ provides a good indication of how busy the BGP process is.

Command:

```
show bgp <AFI> <SAFI> summary
```

Example:

```
RP/0/RP0/CPU0:xrv9k-01#show bgp all all summary
Address Family: VPNv4 Unicast
-----------------------------

BGP router identifier 192.168.0.1, local AS number 65000
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0x0
BGP main routing table version 2208096
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.

Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker         2208096    2208096    2208096    2208096     2208096     2208096

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
10.0.0.2          0 65000     180  601022  2208096    0    0 02:56:18        100
10.0.0.3          0 65000     180  601022  2208096    0    0 02:56:18        100
10.0.0.4          0 65000     180  601022  2208096    0    0 02:56:21        100
10.0.0.5          0 65000     180  601022  2208096    0    0 02:56:21        100
10.0.0.6          0 65000     180  601022  2208096    0    0 02:56:18        100
```

What to look for:

- The size of InQ/OutQ must be zero when the network is stable. It changes quickly when updates are exchanged.
- InQ/OutQ size must not monotonously increase over time.

Telemetry path:

- Cisco-IOS-XR-ipv4-bgp-oper:bgp

Monitor BGP Message Rates

Some BGP neighbors can continuously send updates or withdrawals if network topology is unstable. The BGP RR must then replicate such routing table change thousands of times to all its RR clients. Therefore it is important to monitor the message rates received from neighbors, to track sources of instabilities.

Command:

```
show bgp <AFI> <SAFI> summary
```

Example:

```
RP/0/RP0/CPU0:xrv9k-01#show bgp all all summary
Address Family: VPNv4 Unicast
-----------------------------

BGP router identifier 192.168.0.1, local AS number 65000
BGP generic scan interval 60 secs
BGP table state: Active
Table ID: 0x0
BGP main routing table version 2208096
BGP scan interval 60 secs

BGP is operating in STANDALONE mode.

Process       RcvTblVer   bRIB/RIB   LabelVer  ImportVer  SendTblVer  StandbyVer
Speaker         2208096    2208096    2208096    2208096     2208096     2208096

Neighbor        Spk    AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  St/PfxRcd
10.0.0.2          0 65000     180  601022  2208096    0    0 02:56:18        100
10.0.0.3          0 65000     180  601022  2208096    0    0 02:56:18        100
10.0.0.4          0 65000     180  601022  2208096    0    0 02:56:21        100
10.0.0.5          0 65000     180  601022  2208096    0    0 02:56:21        100
10.0.0.6          0 65000     180  601022  2208096    0    0 02:56:18        100
```

RR clients queues have roughly the same amount of MsgSent but some neighbors can have a number of MsgRcvd higher than others. You must capture multiple snapshots of this command to assess the rate of messages.

Once you have identified the offending peers, you can then go through other commands like **show bgp neighbor <neighbor> detail** and **show bgp neighbor <neighbor> performance-statistics** or **show bgp recent-prefixes** to try to understand which prefixes are flapping and whether it is always the same ones or different ones.

Note: The MsgRcvd and MsgSent counters are per-neighbor but not per address-family. So when running a command like `show bgp all all summary`, you see the same counters per neighbor in the sections for the various address-families. They do not represent the number of messages received/sent from/to that neighbor for that address-family but across address-families.

Monitor CPU Utilisation

CPU utilisation must be monitored on every router, but on a router with a high number of CPU cores dedicated to control plane some readouts can be unintuitive. On an BGP RR with a high number of CPU cores dedicated to Routing Processor (RP), as in the case of XRv9k Appliance, active threads run on different CPU cores, while a number of CPU cores remains idle. As a consequence some CPU cores can be very busy, but the overall CPU utilisation calculated across all CPU cores remains moderate.

Therefore, for proper monitoring of CPU cores utilisation via CLI, use the **show processes cpu thread** command.

Monitor TCP Statistics

Cisco IOS® maintains detailed statistics on every TCP session. CLI command **show tcp brief** displays the list of all existing TCP sessions. In this summary output, for every TCP session you can see this information:

- **PCB:** unique TCP session identifier.
- **VRF-ID**: the ID of the VRF in which the session exists.
    - To see the correspondng VRF name, run this command:
    - show cef vrf all summary | utility egrep "^VRF:|Vrfid" | utility egrep -B1 <VRF-ID>
- **Recv-Q**: instantaneous size of the receive Q. Receive queue holds packets received from NetIO. The **tcp** process extracts the data from a packet and sends it to the corresponding application.
- **Send-Q**: instantaneous size of the send Q. Send queue holds data received from an application. The **tcp** process splits the data into TCP segments (dictated by negotiated maximum segment size - TCP MSS), encapsulates every segment into a layer 3 header of corresponding address family (IPv4 or IPv6) and sends the packet to NetIO.
- **Local Address**: local IPv4 or IPv6 address associated with the TCP socket. TCP sessions in LISTEN state are typically bound to "**any**" IP address, which is represented as "0.0.0.0" or "::" in case of IPv4 or IPv6 respectively.
- **Foreign Address**: remote IPv4 or IPv6 address associated with the TCP socket. TCP sessions in LISTEN state are typically bound to "**any**" IP address, which is represented as "0.0.0.0" or "::" in case of IPv4 or IPv6 respectively.
- **State**: TCP session state. Possible TCP session states are: LISTEN, SYNSENT, SYNRCVD, ESTAB, LASTACK, CLOSING, CLOSEWAIT, FINWAIT1, FINWAIT2, TIMEWAIT, CLOSED.

As the well-known BGP port number is 179, you can limit the displayed TCP sessions to those that are associated with the BGP application.

Example:

```
RP/0/RSP0/CPU0:ASR9k-B#show tcp brief | include "PCB|:179 "
   PCB             VRF-ID     Recv-Q Send-Q Local Address          Foreign Address        State
0x00007ff7d403bde0 0x60000000      0      0  :::179                 :::0                   LISTEN
0x00007ff7d403b020 0x60000002      0      0  :::179                 :::0                   LISTEN
0x00007ff7d403d130 0x60000000      0      0  192.168.0.4:50144      192.168.0.5:179        ESTAB
0x00007ff7a4025650 0x60000000      0      0  0.0.0.0:179            0.0.0.0:0              LISTEN
0x00007ff7a4024a50 0x60000002      0      0  0.0.0.0:179            0.0.0.0:0              LISTEN
```

You can use the displayed PCB value to obtain the statistics for a particular TCP session. CLI commands that provide insight into TCP process statistics:

Global:

```
show tcp statistics clients location <active_RP>

show tcp statistics summary location <active_RP>
```

Per PCB:

```
show tcp brief | i ":179"

show tcp detail pcb <pcb> location 0/RP0/CPU0

show tcp statistics pcb <pcb> location <active_RP>
```

Global TCP statistics commands show the overall health of TCP sessions. Apart from the data packet statistics (in/out), you can see for example whether there are packets with checksum errors, malformed packets, packets dropped due to authentication errors, out-of-order packets, packets with data after window, which gives you an indication of the behaviour of TCP peers.

In the per-PCB commands, you can see important parameters of a TCP session, like MSS, maximum round-trip time, and so on.

Relevant counters in the output of `show tcp detail pcb` command are:

- **Retrans Timer Starts**: indicates how many times the retransmission timer started.
- **Retrans Timer Wakeups**: indicates how many times did the retransmission timer run out, triggering a retransmission of the TCP segment.
- **Current send queue size in bytes**: unacknowledged bytes from the peer.
- **Current receive queue size in bytes/packets**: bytes/packets yet to be read by the application (BGP).
- **mis-ordered bytes**: bytes that are queued in the save queue due to a hole in TCP receive window.

<#root>

```
RP/0/RSP0/CPU0:ASR9k-B#

show tcp detail pcb 0x4a4400e4

==============================================================
Connection state is ESTAB, I/O status: 0, socket status: 0
Established at Sat Apr 20 18:26:26 2024

PCB 0x4a4400e4, SO 0x4a42c0ac, TCPCB 0x4a43b708, vrfid 0x60000000,
Pak Prio: Normal, TOS: 64, TTL: 255, Hash index: 402
Local host: 10.10.10.229, Local port: 179 (Local App PID: 856311)
Foreign host: 10.10.10.254, Foreign port: 46980
(Local App PID/instance/SPL_APP_ID: 856311/0/0)


Current send queue size in bytes: 0 (max 16384)

Current receive queue size in bytes: 0 (max 65535)



mis-ordered: 0 bytes

Current receive queue size in packets: 0 (max 60)



Timer          Starts    Wakeups         Next(msec)

Retrans          2795          0                  0

SendWnd          1341          0                  0
TimeWait            0          0                  0
AckHold           274          2                  0
KeepAlive         333          1             299983
PmtuAger            0          0                  0
GiveUp              0          0                  0
Throttle            0          0                  0
FirstSyn            0          0                  0

   iss: 2030796738  snduna: 2034498828  sndnxt: 2034498828
sndmax: 2034498828  sndwnd: 3291        sndcwnd: 4200
   irs: 285455091   rcvnxt: 285455710   rcvwnd: 64917   rcvadv: 285520627
```

```
SRTT: 162 ms, RTTO: 415 ms, RTV: 253 ms, KRTT: 0 ms
minRTT: 0 ms, maxRTT: 247 ms

ACK hold time: 200 ms, Keepalive time: 300 sec, SYN waittime: 30 sec
Giveup time: 0 ms, Retransmission retries: 0, Retransmit forever: FALSE
Connect retries remaining: 0, connect retry interval: 0 secs
<...>
RP/0/RSP0/CPU0:ASR9k-B#
```

Monitor Memory Utilisation

The BGP route table is stored in BGP process heap memory. The routing table is stored in RIB process heap memory.

Useful commands for heap memory monitoring:

```
show memory summary

show memory summary detail

show memory-top-consumers

show memory heap summary all
```

Telemetry sensor path:

```
Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/detail
```

FIB stores forwarding entries in shared memory space.

Useful commands for shared memory monitoring:

```
show memory summary

show memory summary detail

show shmwin summary
```

Monitor BGP Process Performance

Useful command that provides internal data on BGP process performance:

```
show bgp process performance-statistics
```

```
show bgp process performance-statistics detail
```

Monitor BGP Convergence

Another useful command is the one that shows the overall status of BGP convergence: show bgp convergence

When the network is stable, you see something like this:

```
RP/0/RP0/CPU0:ASR9k-B#show bgp convergence
Mon Dec 18 13:55:47.976 UTC
Converged.
All received routes in RIB, all neighbors updated.
All neighbors have empty write queues.
RP/0/RP0/CPU0:ASR9k-B#
```