Configure TCP Optimization Feature on Cisco IOS® XE SD-WAN cEdge Routers

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Problem

Solution

- Supported XE SD-WAN Platforms
- Caveats

Configure

- Use Case 1. Configure TCP Optimization on a Branch (all in one cEdge)
- Use Case 2. Configure TCP Optimization in Data Center with an External SN
- Failover Case

Verify

Troubleshoot

Using TCPOpt with other AppQoE / UTD Featutes starting from 17.2

Related Information

Introduction

This document describes the Transmission Control Protocol (TCP) Optimization feature on Cisco IOS® XE SD-WAN routers, which was introduced in 16.12 release in August 2019. The topics covered are prerequisites, problem description, solution, the differences in TCP optimization algorithms between Viptela OS (vEdge) and XE SD-WAN (cEdge), configuration, verification and list of related documents.

Prerequisites

Requirements

There are no specific requirements for this document.

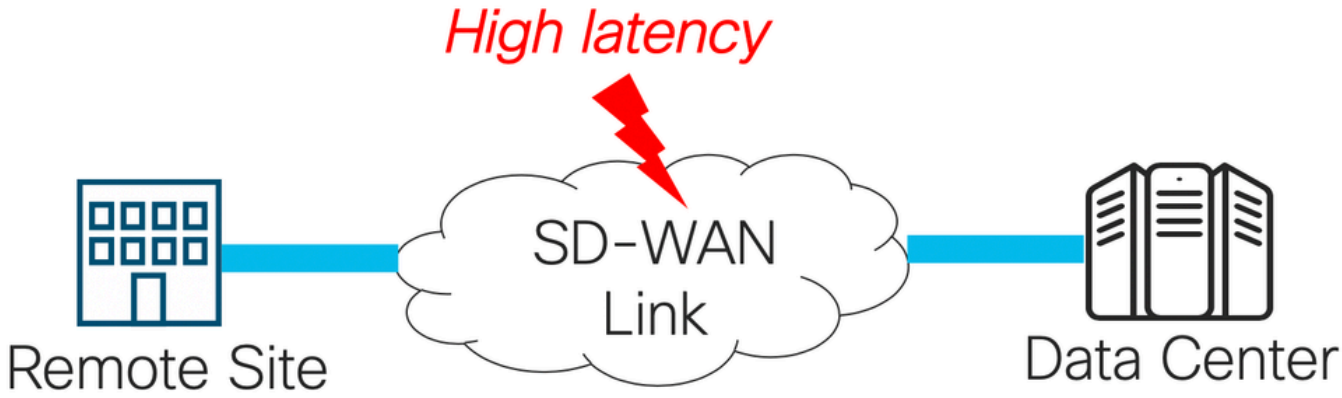
Components Used

The information in this document is based on Cisco IOS® XE SD-WAN.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Problem

High latency on a WAN link between two SD-WAN sides causes bad application performance. You have critical TCP traffic, which must be optimized.



Solution

When you use TCP Optimization feature, you improve the average TCP throughput for critical TCP flows between two SD-WAN sites.

Take a look at the overview and differences between TCP Optimization on cEdge Bottleneck Bandwidth and Round-trip (BBR) and vEdge (CUBIC)

Fast BBR propagation time algorithm is used in the XE SD-WAN implementation (on cEdge).

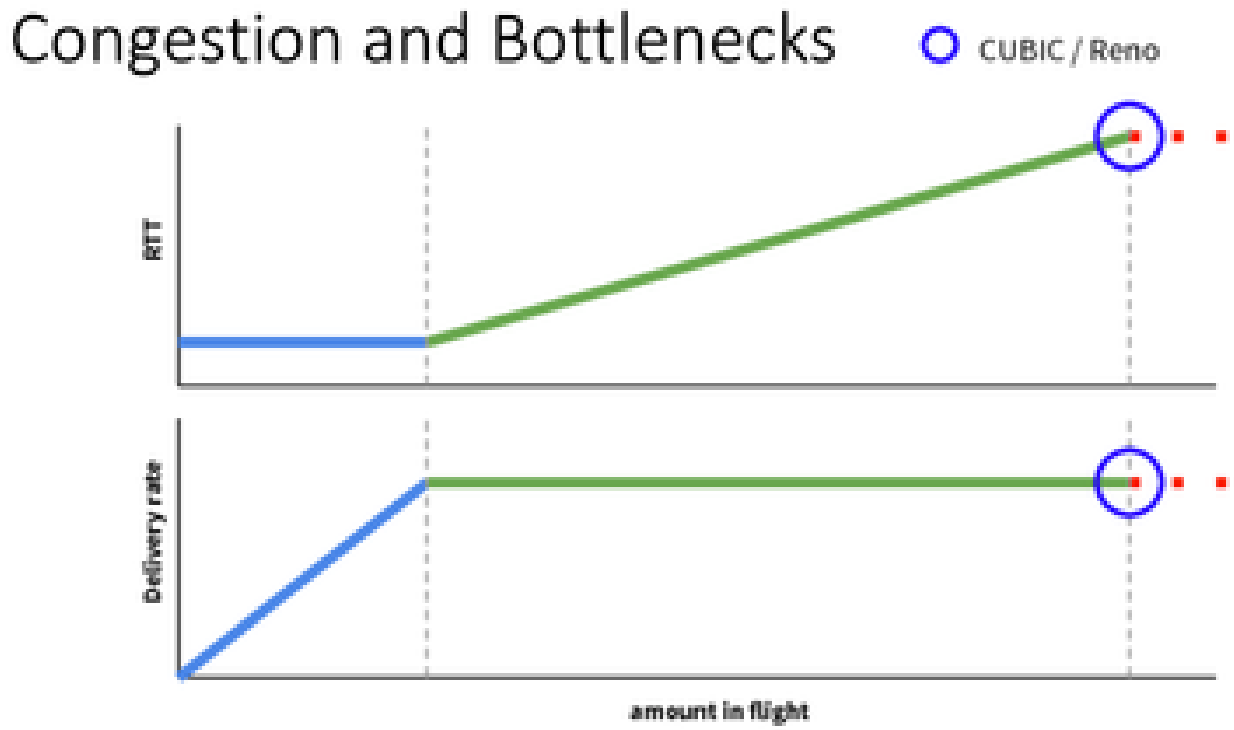
Viptela OS (vEdge) has a different, older algorithm, called CUBIC.

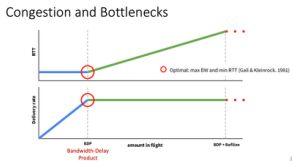
CUBIC takes mainly packet loss into consideration and is widely implemented across different client operating systems. Windows, Linux, MacOS, Android already have CUBIC built-in. In some cases, where you have old clients running TCP stack without CUBIC, enabling TCP optimization on vEdge brings improvements. One of the examples, where vEdge TCP CUBIC optimization benefited, is on submarines that use old client hosts and WAN links experiencing significant delays/drops. Note that only vEdge 1000 and vEdge 2000 support TCP CUBIC.

BBR is mainly focused on round-trip time and latency. Not on packet loss. If you send packets from US West to East coast or even to Europe across the public internet, in the majority of the cases you don't see any packet loss. Public internet is sometimes too good in terms of packet loss. But, what you see is delay/latency. And this problem is addressed by BBR, which was developed by Google in 2016.

In a nutshell, BBR models the network and looks at each acknowledgment (ACK) and updates max Bandwidth (BW) and minimum Round Trip Time (RTT). Then control sending is based on model: probe for max BW and min RTT, pace near estimate BW and keep inflight near Bandwidth-Delay-Product (BDP). The main goal is to ensure high throughput with a small bottleneck queue.

This slide from Mark Claypool shows the area, where CUBIC operates:





BBR operates in a better place, which is shown in this slide also from Mark Claypool:

If you want to read more about the BBR algorithm, you can find several publications about BBR linked at the top of the bbr-dev mailing list home page Here.

In summary:

Platform & Algorithm	Key input parameter	Use Case
cEdge (XE SD-WAN): BBR	RTT/Latency	Critical TCP traffic between two SD-WAN sites
vEdge (Viptela OS): CUBICP	Packet Loss	Old clients without any TCP optimization

Supported XE SD-WAN Platforms

In the XE SD-WAN SW Release 16.12.1d, these cEdge platforms support TCP Opmitization BBR:

- ISR4331
- ISR4351
- CSR1000v with 8 vCPU and min. 8 GB RAM

Caveats

- All platforms with DRAM less than 8 GB RAM are currently not supported.
- All platforms with 4 or less data cores are currently not supported.
- TCP Optimization does not support MTU 2000.
- Currently - no support for IPv6 traffic.
- Optimization for DIA traffic to a 3rd party BBR server not supported. You need to have a cEdge SD-

WAN routers on both sides.

- In the data center scenario currently, only one Service Node (SN) is supported per one Control Node (CN).
- Combined use case with security (UTD container) and TCP Optimization on the same device is not supported on 16.12 release. Staring from 17.2 the combined use case is supported. The order of operation is TCP Opt first, then Security. The packet will be punted to TCP Optimization container and then service chained (no second punt) to UTD security container. TCP Optimisation is done for complete flow, not for first few bytes. If UTD drops, then the complete connection is dropped.

Note: ASR1k does not currently support TCP Optimization. However, there is a solution for ASR1k, where the ASR1k send TCP traffic via AppNav tunnel (GRE encapsulated) to an external CSR1kv for optimization. Currently (Feb. 2020) only one CSR1k as single external service node is supported, but not well tested. This is described later in the configuration section.

This table summarizes caveats per release and underlines supported hardware platforms:

Scenarios	**Use Cases**	**16.12.1**	**17.2.1**	**17.3.1**	**17.4.1**	**Comments**
Branch-to-Internet	DIA	No	Yes	**Yes**	**Yes**	In 16.12.1 AppQoE FIA is not enabled on internet interface
	SAAS	No	Yes	**Yes**	**Yes**	In 16.12.1 AppQoE FIA is not enabled on internet interface
Branch-to-DC	Single Edge Router	No	No	No	**Yes**	Need to support multiple SN
	Multiple Edge Routers	No	No	No	**Yes**	Needs flow symmetry or Appnav flow sync. 16.12.1 not tested with
	Multiple SNs	No	No	No	**Yes**	vManage enhancement to accept multiple SN IPs
Branch-to-Branch	Full Mesh Network (Spoke-to-Spoke)	**Yes**	**Yes**	**Yes**	**Yes**	
	Hub-and-Spoke (Spoke-Hub-Spoke)	**No**	**Yes**	**Yes**	**Yes**	

BBR Support	TCP Opt with BBR	**Partial**	**Partial**	**Full**	**Full**	
Platforms	Platforms supported	**Only 4300 & CSR**	**All but ISR1100**	**All**	**All**	

Configure

A concept of SN and CN is used for TCP Optimization:

- SN is a daemon, which is responsible for the actual optimization of TCP flows.
- CN is known as AppNav Controller and is responsible for traffic selection and transport to/from SN.

SN and CN can run on the same router or separated as different nodes.

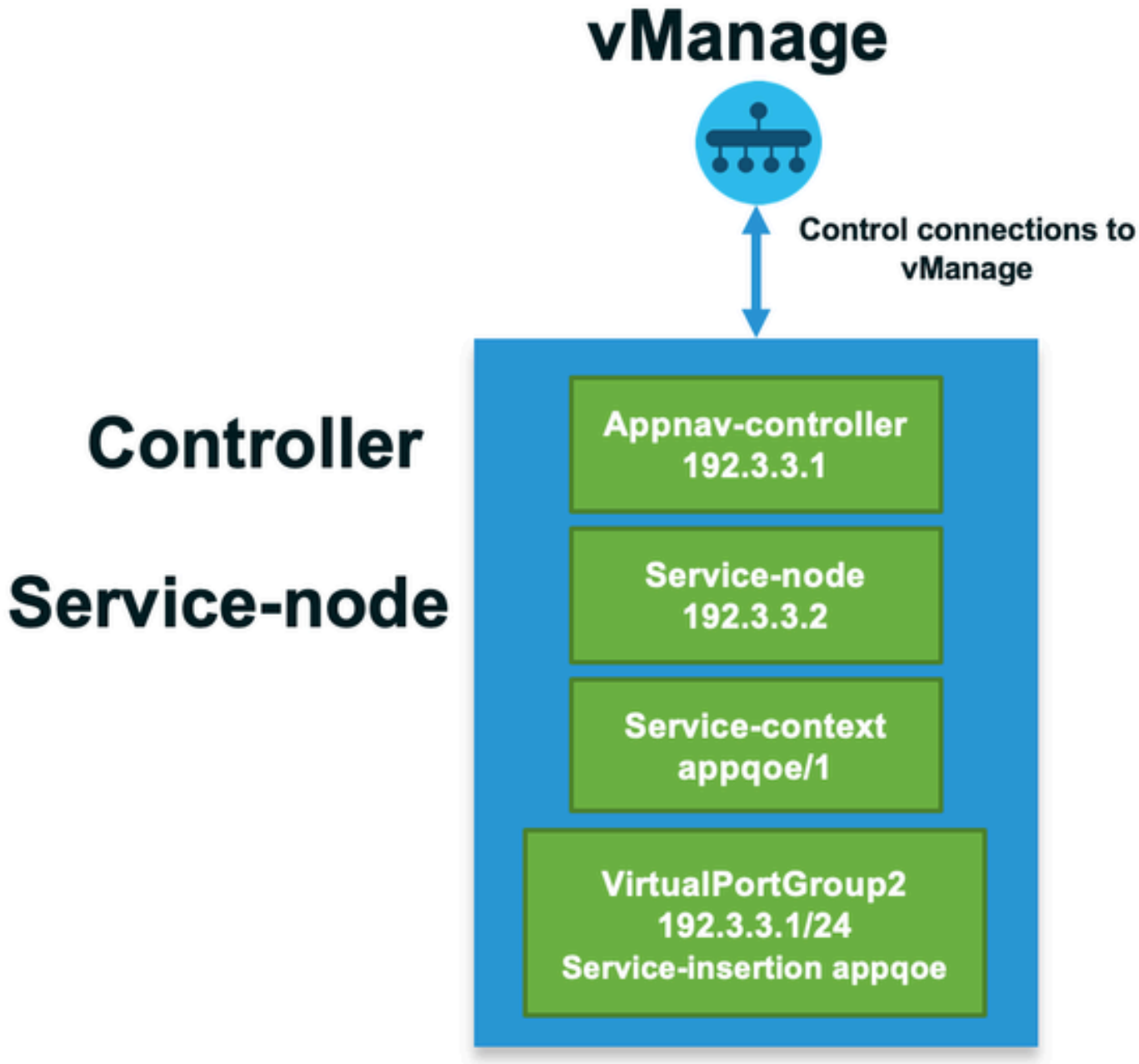
There are two main use cases:

1. Branch use case with SN and CN running on the same ISR4k router.
2. Data Center use case, where CN runs on ASR1k and SN runs on a separate CSR1000v virtual router.

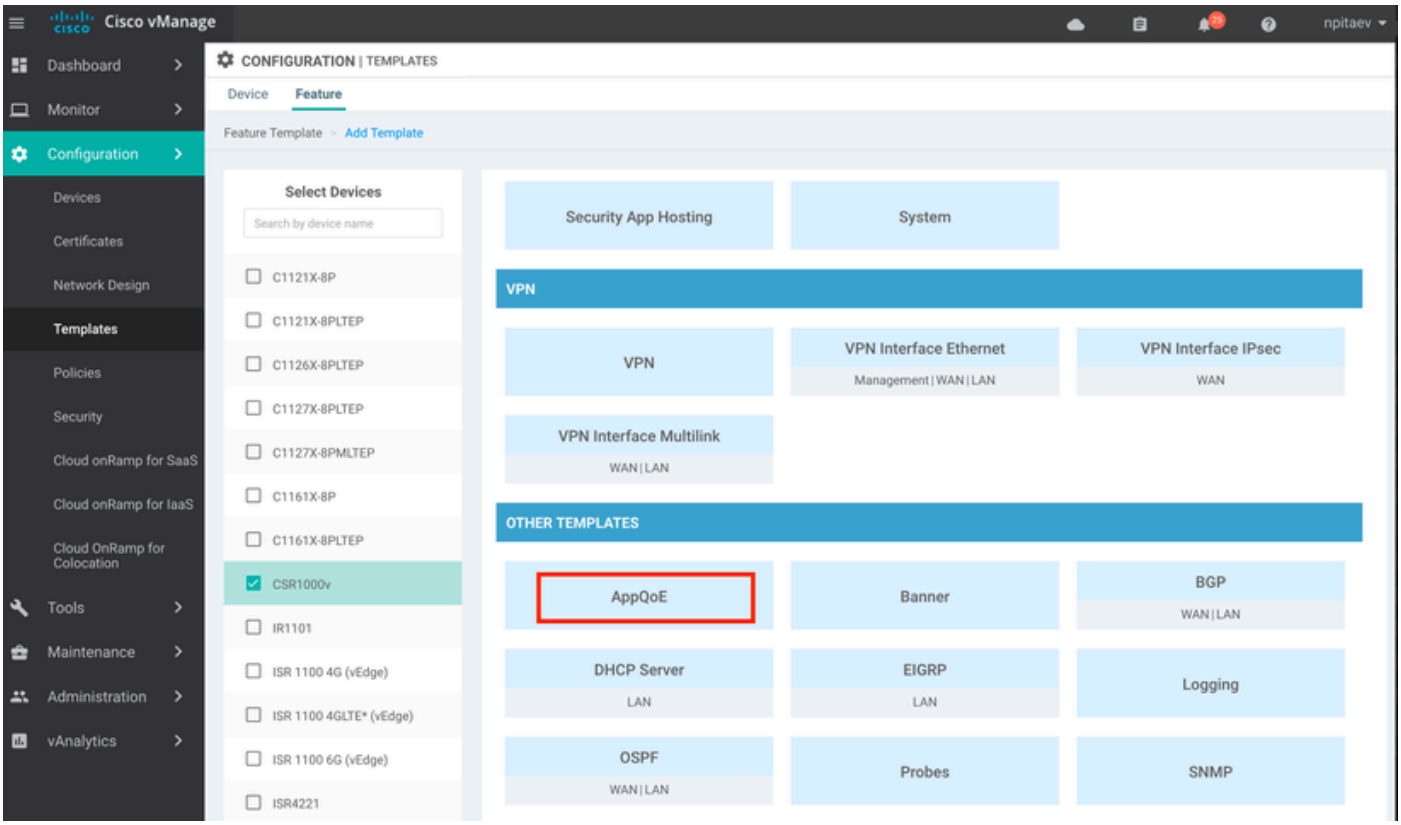
Both use cases are described in this section.

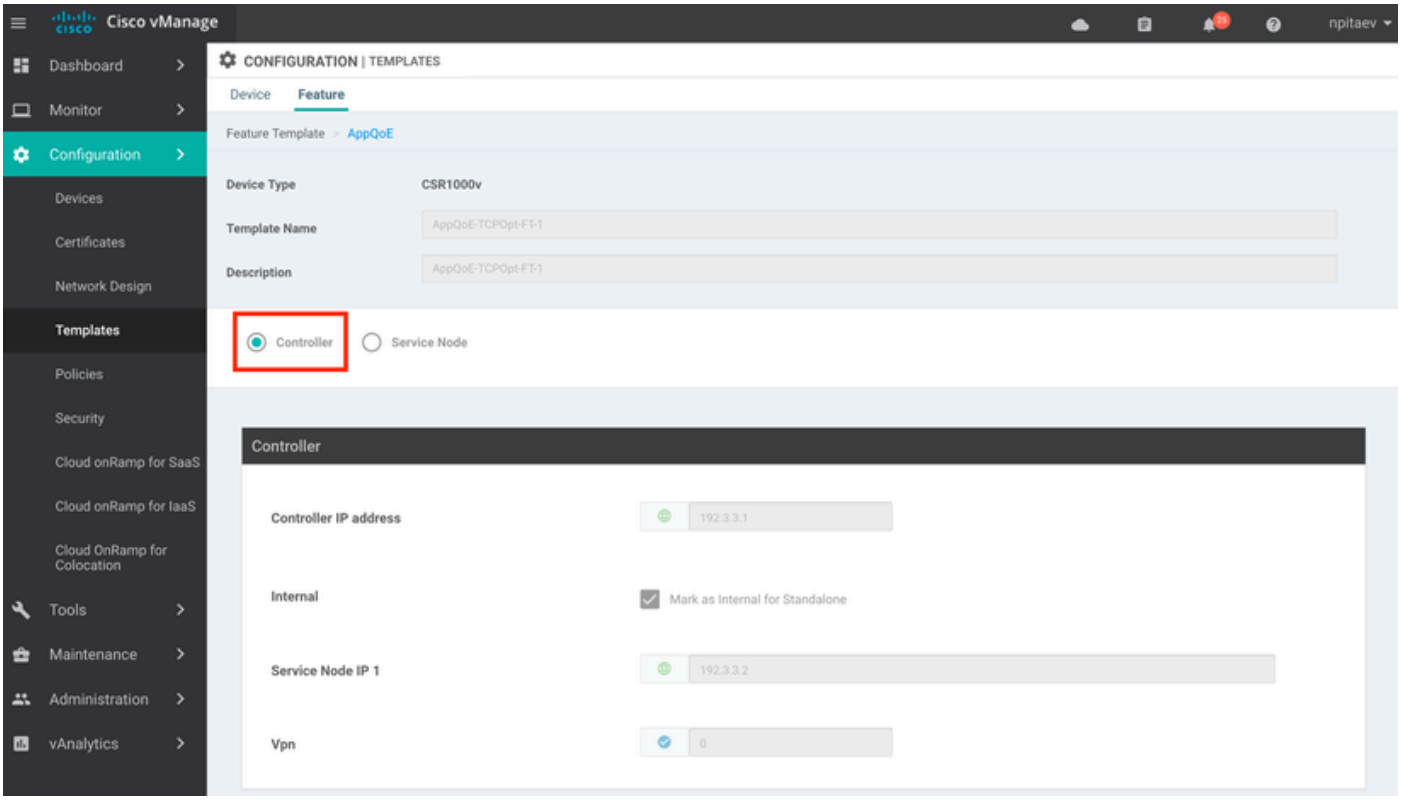
Use Case 1. Configure TCP Optimization on a Branch (all in one cEdge)

This image shows the overall internal architecture for a single standalone option at a branch:

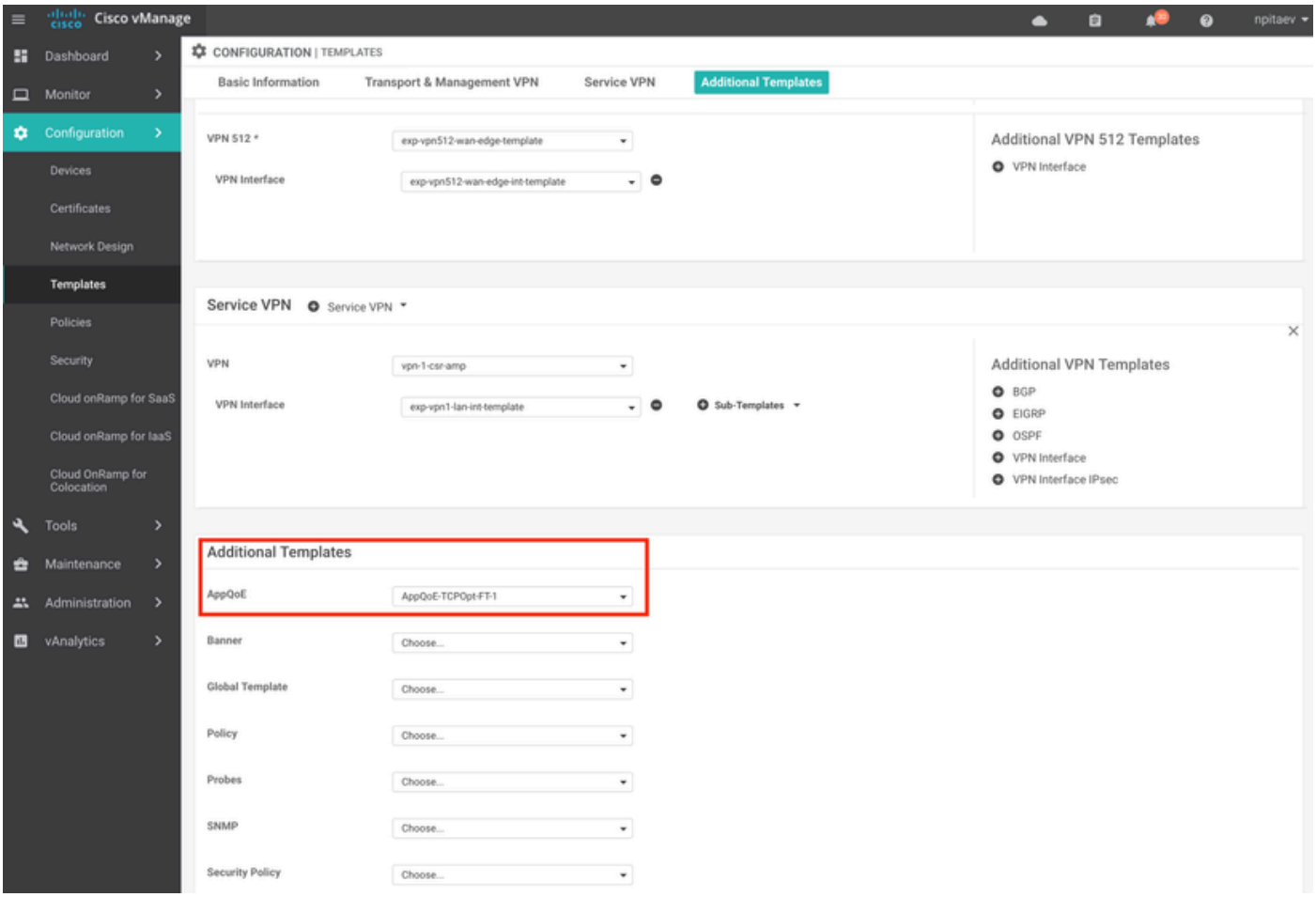


Step1. In order to configure TCP optimization, you need to create a feature template for TCP Optimization in vManage. Navigate to **Configuration > Templates > Feature Templates > Other Templates > AppQoE** as shown in the image.





Step 2. Add the AppQoE feature template to the appropriate device template under **Additional Templates**:

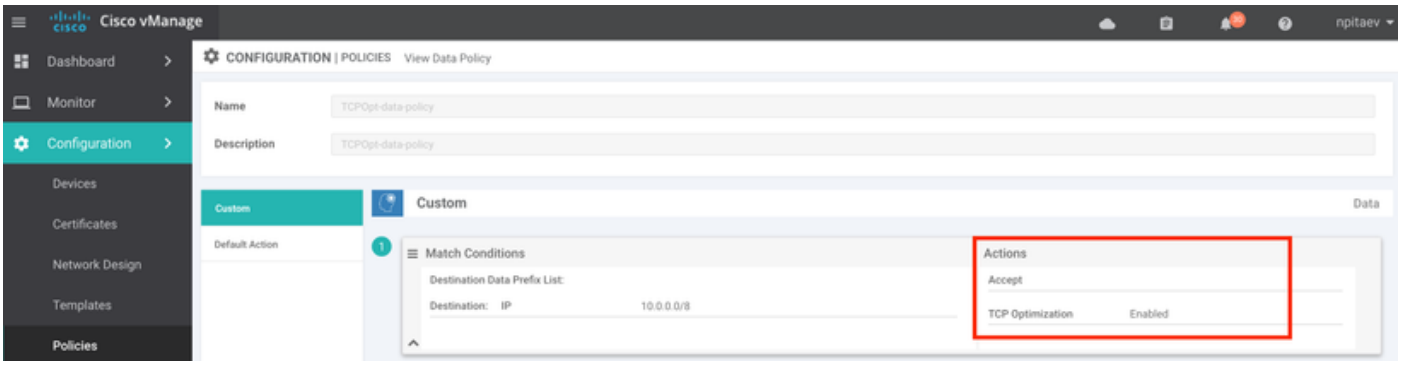


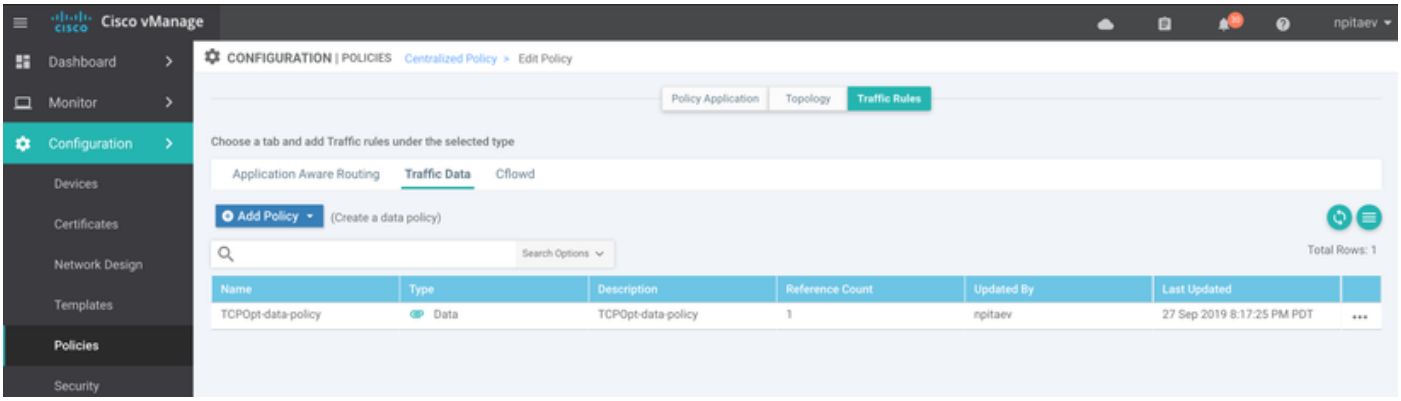
Here is the CLI preview of the Template Configuration:

```
service-insertion service-node-group appqoe SNG-APPQOE
 service-node 192.3.3.2
!
service-insertion appnav-controller-group appqoe ACG-APPQOE
 appnav-controller 192.3.3.1
!
service-insertion service-context appqoe/1
 appnav-controller-group ACG-APPQOE
 service-node-group SNG-APPQOE
 vrf global
 enable
!
!
interface VirtualPortGroup2
 ip address 192.3.3.1 255.255.255.0
 no mop enabled
 no mop sysid
 service-insertion appqoe
!
```

Step 3. Create a centralized data policy with the definition of the interesting TCP traffic for optimization.

As an example; this data policy matches IP prefix 10.0.0.0/8, which includes source and destination addresses, and enables TCP optimization for it:





Here is the CLI preview of the vSmart Policy:

<#root>

```
policy
 data-policy _vpn-list-vpn1_TCPOpt_1758410684
  vpn-list vpn-list-vpn1
    sequence 1
     match
      destination-ip 10.0.0.0/8
     !
     action accept


tcp-optimization

     !
    !
  default-action accept
 !
 lists
  site-list TCPOpt-sites
   site-id 211
   site-id 212
  !
  vpn-list vpn-list-vpn1
   vpn 1
  !
 !
!
apply-policy
 site-list TCPOpt-sites
  data-policy _vpn-list-vpn1_TCPOpt_1758410684 all
 !
!
```

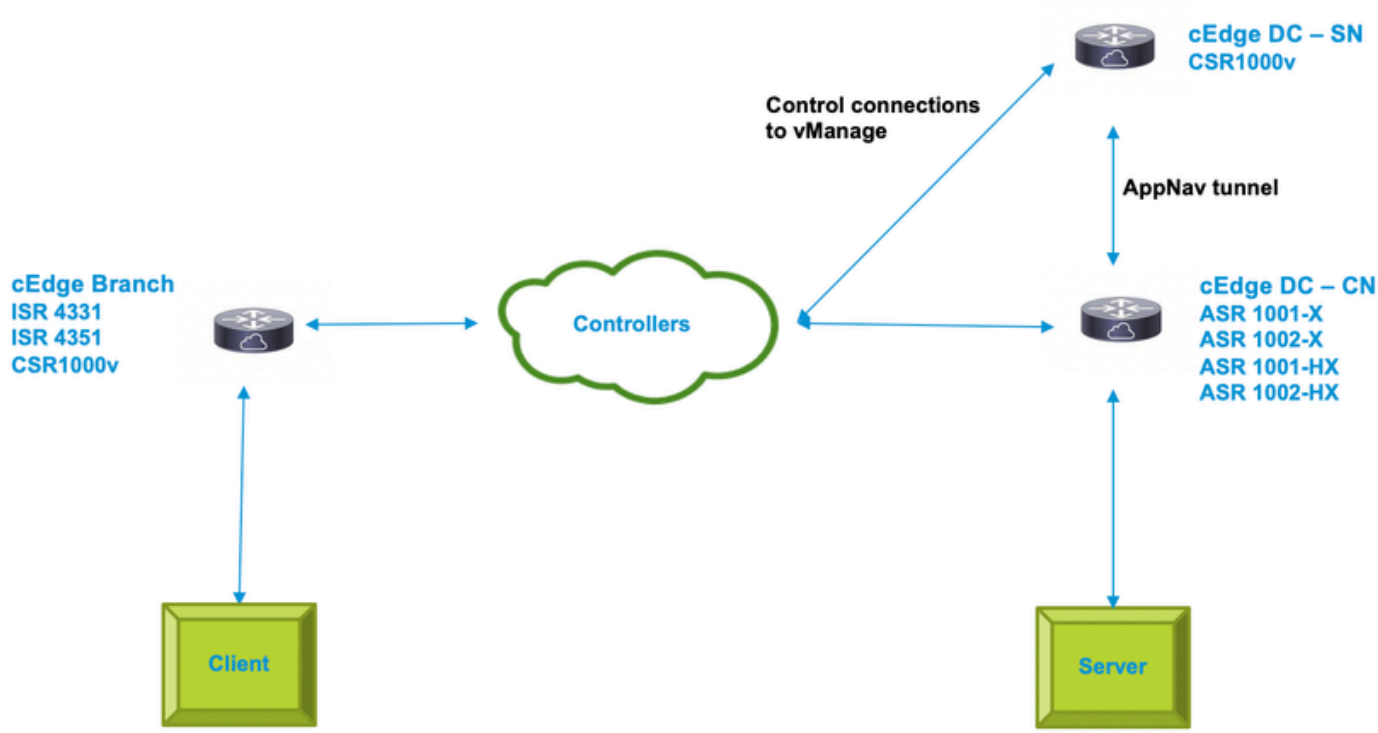
Use Case 2. Configure TCP Optimization in Data Center with an External SN

The main difference to the branch use case is the physical separation of SN and CN. In the all-in-one branch use case, the connectivity is done within the same router using Virtual Port Group Interface. In the data center use case, there is a AppNav GRE-encapsulated tunnel between ASR1k acting as CN and external CSR1k running as SN. There is no need for a dedicated link or cross-connect between CN and external SN, simple IP reachability is enough.

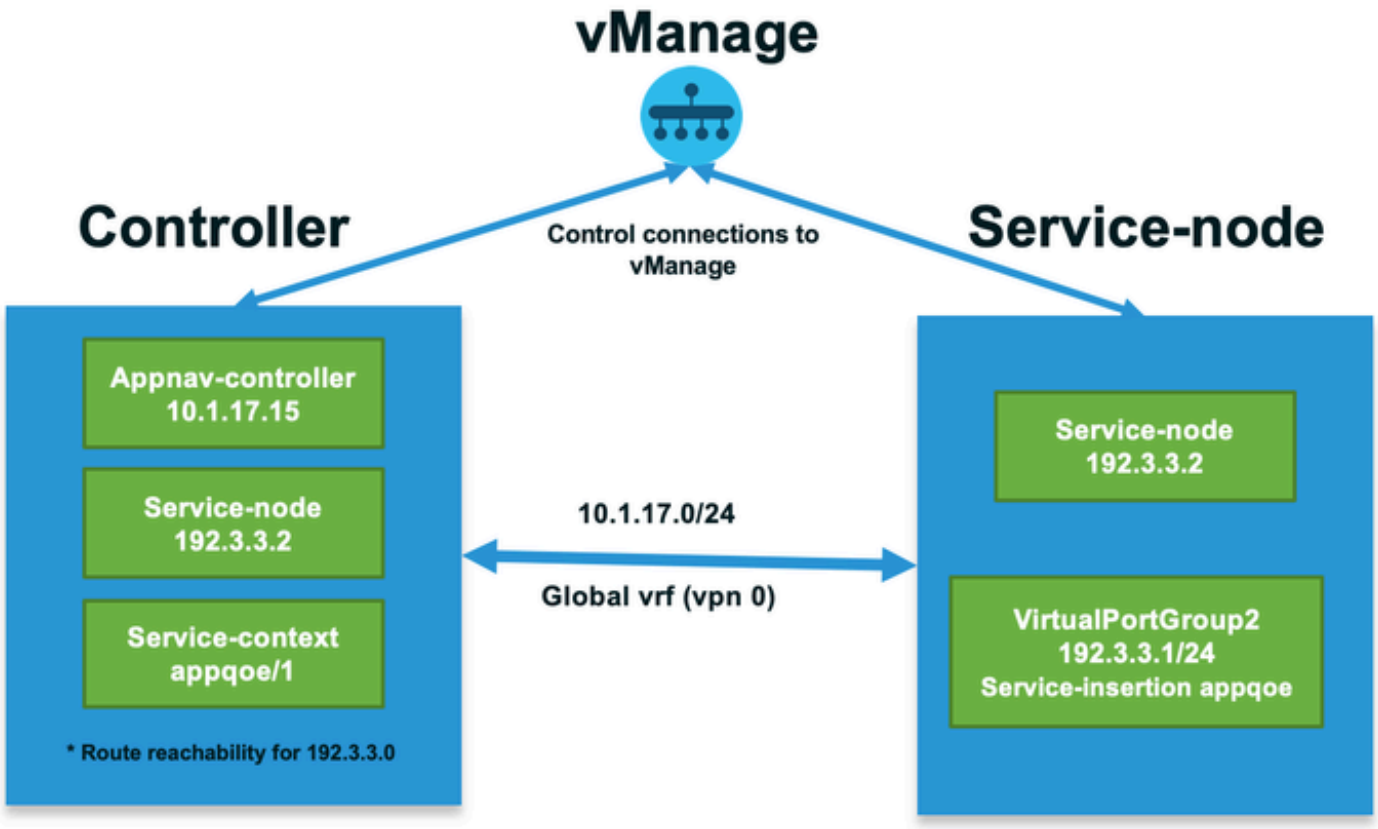
There is one AppNav (GRE) tunnel per SN. For future use, where multiple SNs are supported, it is recommended to use /28 subnet for the network between CN and SN.

Two NICs are recommended on a CSR1k acting as SN. 2nd NIC for SD-WAN controller is needed if SN has to be configured / managed by vManage. If SN is going to be manually configured/managed, then 2nd NIC is optional.

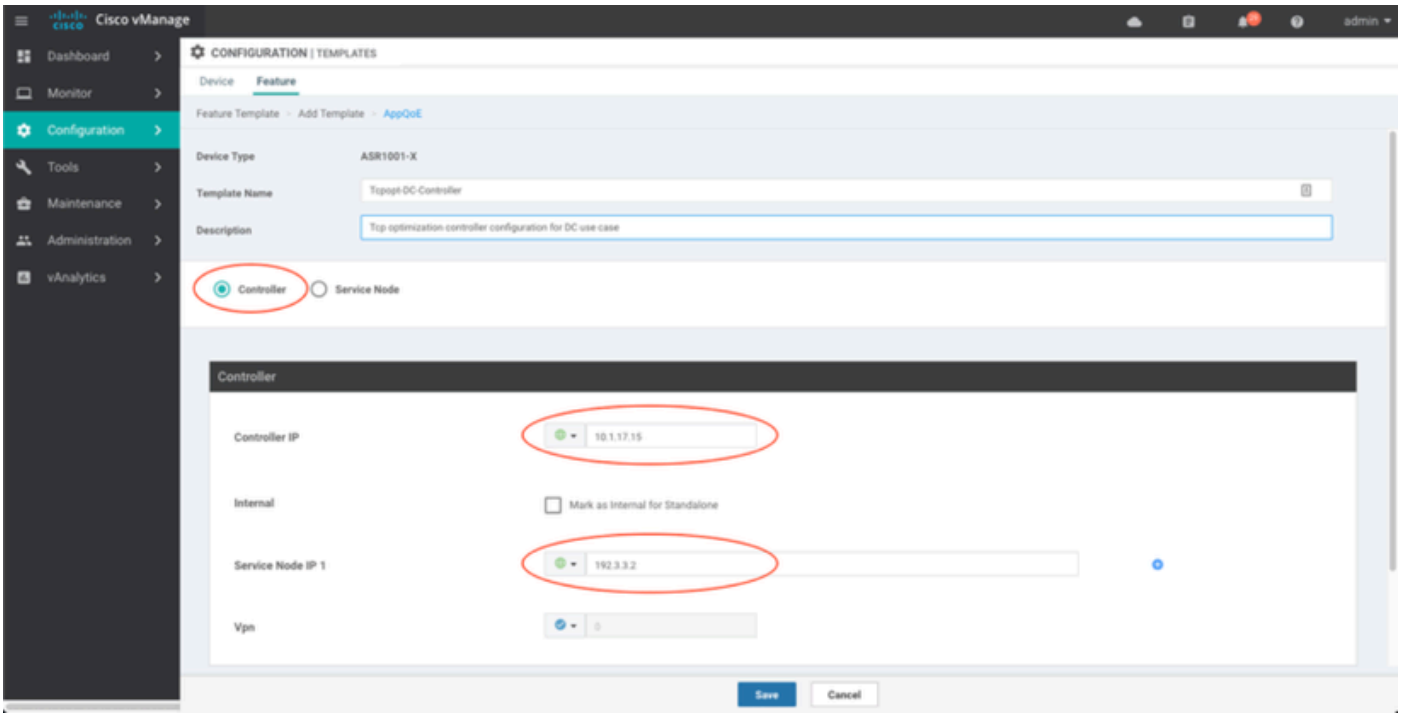
This image shows Data Center ASR1k running as CN and CSR1kv as Service Node SN :



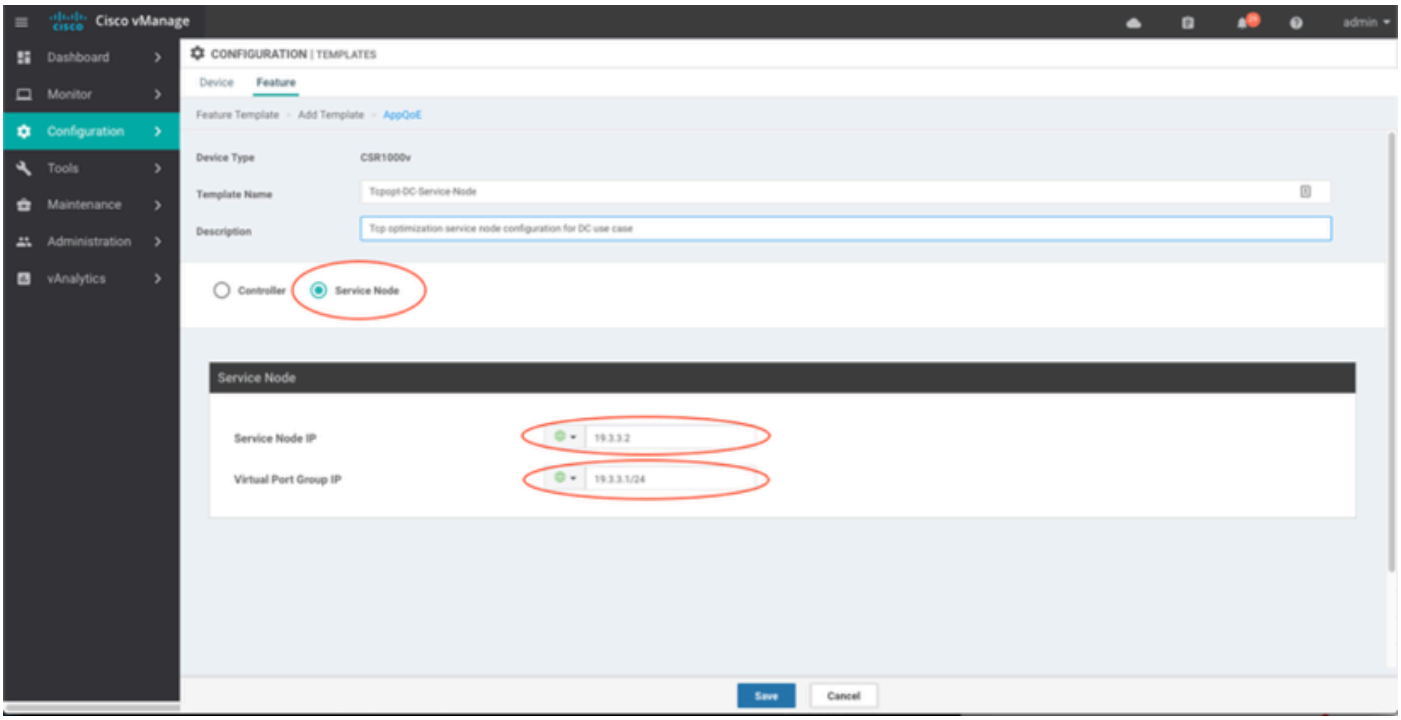
The topology for the data center use case with ASR1k and external CSR1k is shown here:



This AppQoE feature template shows ASR1k configured as Controller:



CSR1k configured as external Service Node is shown here:



Failover Case

Failover in the data center use case with CSR1k acting as SN, in case of external CSR1k failure:

- TCP sessions that already exist are lost because the TCP session on SN is terminated.
- New TCP sessions are sent to the final destination, but TCP traffic is not optimized (bypass).
- No blackholing for interesting traffic in case of SN failure.

Failover detection is based on AppNav heartbeat, which is 1 beat per second. After 3 or 4 errors, the tunnel is declared as down.

Failover in the branch use case is similar - in case of SN failure, the traffic is sent non-optimized directly to the destination.

Verify

Use this section in order to confirm that your configuration works properly.

Verify TCP Optimization on CLI with the use of this CLI command and see the summary of the optimized flows:

<#root>

```
BR11-CSR1k#

show plat hardware qfp active feature sdwan datapath appqoe summary

TCPOPT summary
----------------

  optimized flows    : 2

  expired flows      : 6033
  matched flows      : 0
  divert pkts        : 0
```

```
  bypass pkts        : 0
  drop pkts          : 0
  inject pkts        : 20959382
  error pkts         : 88

BR11-CSR1k#
```

This output gives detailed information about optimized flows:

```
<#root>

BR11-CSR1k#

show platform hardware qfp active flow fos-to-print all

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
     GLOBAL CFT  ~  Max Flows:2000000   Buckets Num:4000000
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Filtering parameters:
    IP1 : ANY
    Port1 : ANY
    IP2 : ANY
    Port2 : ANY
    Vrf id : ANY
    Application: ANY
    TC id: ANY
    DST    Interface id: ANY
    L3 protocol : IPV4/IPV6
    L4 protocol : TCP/UDP/ICMP/ICMPV6
    Flow type : ANY
Output parameters:
    Print CFT internal data ? No
    Only print summary ? No
    Asymmetric : ANY
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
keyID: SrcIP SrcPort DstIP DstPort L3-Protocol L4-Protocol vrfID
====================================================================
   key #0: 192.168.25.254 26113 192.168.25.11 22 IPv4 TCP 3
   key #1: 192.168.25.11 22 192.168.25.254 26113 IPv4 TCP 3
====================================================================
   key #0: 192.168.25.254 26173 192.168.25.11 22 IPv4 TCP 3
   key #1: 192.168.25.11 22 192.168.25.254 26173 IPv4 TCP 3
====================================================================
   key #0: 10.212.1.10 52255 10.211.1.10 8089 IPv4 TCP 2
   key #1: 10.211.1.10 8089 10.212.1.10 52255 IPv4 TCP 2

Data for FO with id: 2
------------------------

appqoe

: flow action DIVERT, svc_idx 0, divert pkt_cnt 1, bypass pkt_cnt 0, drop pkt_cnt 0, inject pkt_cnt 1,
====================================================================
   key #0: 10.212.1.10 52254 10.211.1.10 8089 IPv4 TCP 2
   key #1: 10.211.1.10 8089 10.212.1.10 52254 IPv4 TCP 2

Data for FO with id: 2
------------------------

appqoe
```

```
: flow action DIVERT, svc_idx 0, divert pkt_cnt 158, bypass pkt_cnt 0, drop pkt_cnt 0, inject pkt_cnt 2
=====================================================================
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
           Number of flows that passed filter: 4
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        FLOWS DUMP DONE.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

BR11-CSR1k#
```

Troubleshoot

The following CLI will help to identify issues with a specific TCP flow.

All example were taken from IOS XE SD-WAN 17.2.1 image running on ISR4431.

1. Find out the VRF Id (which is not qual to the VRF name). In this case we see in the output below, that service VPN (VRF) 1 has the VRF Id 2:

<#root>

AppQoE_R2#

show vrf detail

VRF 1 (

VRF Id = 2

); default RD 1:1; default VPNID <not set>
  New CLI format, supports multiple address-families
  Flags: 0x180C
  Interfaces:
    Gi0/0/3
…

2. Find out appropriate flow-id for the client in the service vpn – we see here 4 different flows between two windows clients using SMB share over TCP port 445:

<#root>

AppQoE_R2#

show sdwan appqoe flow vpn-id 2 client-ip 192.168.200.50

```
Optimized Flows
---------------
T:TCP, S:SSL, U:UTD
Flow ID      VPN  Source IP:Port        Destination IP:Port   Service
15731593842  2    192.168.200.50:49741  192.168.100.50:445    T
17364128987  2    192.168.200.50:49742  192.168.100.50:445    T
25184244867  2    192.168.200.50:49743  192.168.100.50:445    T
28305760200  2    192.168.200.50:49744  192.168.100.50:445    T
AppQoE_R2#
```

3. See TCP Opt stats for the appropriate flow:

```
<#root>

AppQoE_R2#

show sdwan appqoe flow flow-id 15731593842


VPN: 2  APP: 0 [Client 192.168.200.50:49741 - Server 192.168.100.50:445]

TCP stats
---------
Client Bytes Received   : 14114
Client Bytes Sent       : 23342
Server Bytes Received   : 23342
Server Bytes Sent       : 14114
TCP Client Rx Pause     : 0x0
TCP Server Rx Pause     : 0x0
TCP Client Tx Pause     : 0x0
TCP Server Tx Pause     : 0x0
Client Flow Pause State : 0x0
Server Flow Pause State : 0x0
TCP Flow Bytes Consumed : 0
TCP Client Close Done   : 0x0
TCP Server Close Done   : 0x0
TCP Client FIN Rcvd     : 0x0
TCP Server FIN Rcvd     : 0x0
TCP Client RST Rcvd     : 0x0
TCP Server RST Rcvd     : 0x0
TCP FIN/RST Sent        : 0x0
Flow Cleanup State      : 0x0
TCP Flow Events
  1. time:2196.550604   ::   Event:TCPPROXY_EVT_FLOW_CREATED
  2. time:2196.550655   ::   Event:TCPPROXY_EVT_SYNCACHE_ADDED
  3. time:2196.552366   ::   Event:TCPPROXY_EVT_ACCEPT_DONE
  4. time:2196.552665   ::   Event:TCPPROXY_EVT_CONNECT_START
  5. time:2196.554325   ::   Event:TCPPROXY_EVT_CONNECT_DONE
  6. time:2196.554370   ::   Event:TCPPROXY_EVT_DATA_ENABLED_SUCCESS

AppQoE_R2#
```

4. See also generic TCP Opt stats, which also show 4 optimized flows:

```
<#root>

AppQoE_R2#

show tcpproxy statistics


=========================================================
               TCP Proxy Statistics
=========================================================
Total Connections                  : 6
Max Concurrent Connections         : 4
Flow Entries Created               : 6
Flow Entries Deleted               : 2
```

Current Flow Entries : 4

Current Connections : 4

```
Connections In Progress             : 0
Failed Connections                  : 0
SYNCACHE Added                      : 6
SYNCACHE Not Added:NAT entry null   : 0
SYNCACHE Not Added:Mrkd for Cleanup : 0
SYN purge enqueued                  : 0
SYN purge enqueue failed            : 0
Other cleanup enqueued              : 0
Other cleanup enqueue failed        : 0
Stack Cleanup enqueued              : 0
Stack Cleanup enqueue failed        : 0
Proxy Cleanup enqueued              : 2
Proxy Cleanup enqueue failed        : 0
Cleanup Req watcher called          : 135003
Total Flow Entries pending cleanup  : 0
Total Cleanup done                  : 2
Num stack cb with null ctx          : 0
Vpath Cleanup from nmrx-thread      : 0
Vpath Cleanup from ev-thread        : 2
Failed Conn already accepted conn   : 0
SSL Init Failure                    : 0

Max Queue Length Work               : 1
Current Queue Length Work           : 0
Max Queue Length ISM                : 0
Current Queue Length ISM            : 0
Max Queue Length SC                 : 0
Current Queue Length SC             : 0
Total Tx Enq Ign due to Conn Close  : 0
Current Rx epoll                    : 8
Current Tx epoll                    : 0

Paused by TCP Tx Full               : 0
Resumed by TCP Tx below threshold   : 0
Paused by TCP Buffer Consumed       : 0
Resumed by TCP Buffer Released      : 0
SSL Pause Done                      : 0
SSL Resume Done                     : 0
SNORT Pause Done                    : 0
SNORT Resume Done                   : 0
EV SSL Pause Process                : 0
EV SNORT Pause Process              : 0
EV SSL/SNORT Resume Process         : 0
Socket Pause Done                   : 0
Socket Resume Done                  : 0
SSL Pause Called                    : 0
SSL Resume Called                   : 0
Async Events Sent                   : 0
Async Events Processed              : 0
Tx Async Events Sent                : 369
Tx Async Events Recvd               : 369
Tx Async Events Processed           : 369
Failed Send                         : 0

TCP SSL Reset Initiated             : 0
TCP SNORT Reset Initiated           : 0
```

```
TCP FIN Received from clnt/svr      : 0
TCP Reset Received from clnt/svr    : 2
SSL FIN Received -> SC              : 0
SSL Reset Received -> SC            : 0
SC FIN Received -> SSL              : 0
SC Reset Received -> SSL            : 0
SSL FIN Received -> TCP             : 0
SSL Reset Received -> TCP           : 0
TCP FIN Processed                   : 0
TCP FIN Ignored FD Already Closed   : 0
TCP Reset Processed                 : 4
SVC Reset Processed                 : 0

Flow Cleaned with Client Data       : 0
Flow Cleaned with Server Data       : 0
Buffers dropped in Tx socket close  : 0
TCP 4k Allocated Buffers            : 369
TCP 16k Allocated Buffers           : 0
TCP 32k Allocated Buffers           : 0
TCP 128k Allocated Buffers          : 0
TCP Freed Buffers                   : 369
SSL Allocated Buffers               : 0
SSL Freed Buffers                   : 0
TCP Received Buffers                : 365
TCP to SSL Enqueued Buffers         : 0
SSL to SVC Enqueued Buffers         : 0
SVC to SSL Enqueued Buffers         : 0
SSL to TCP Enqueued Buffers         : 0
TCP Buffers Sent                    : 365

TCP Failed Buffers Allocations      : 0
TCP Failed 16k Buffers Allocations  : 0
TCP Failed 32k Buffers Allocations  : 0
TCP Failed 128k Buffers Allocations : 0
SSL Failed Buffers Allocations      : 0
Rx Sock Bytes Read < 512            : 335
Rx Sock Bytes Read < 1024           : 25
Rx Sock Bytes Read < 2048           : 5
Rx Sock Bytes Read < 4096           : 0
SSL Server Init                     : 0
Flows Dropped-Snort Gbl Health Yellow        : 0
Flows Dropped-Snort Inst Health Yellow       : 0
Flows Dropped-WCAPI Channel Health Yellow    : 0
Total WCAPI snd flow create svc chain failed : 0
Total WCAPI send data svc chain failed       : 0
Total WCAPI send close svc chain failed      : 0
Total Tx Enqueue Failed                      : 0
Total Cleanup Flow Msg Add to wk_q Failed    : 0
Total Cleanup Flow Msg Added to wk_q         : 0
Total Cleanup Flow Msg Rcvd in wk_q          : 0
Total Cleanup Flow Ignored, Already Done     : 0
Total Cleanup SSL Msg Add to wk_q Failed     : 0
Total UHI mmap                               : 24012
Total UHI munmap                             : 389

Total Enable Rx Enqueued                     : 0
Total Enable Rx Called                       : 0
Total Enable Rx Process Done                 : 0
Total Enable Rx Enqueue Failed               : 0
Total Enable Rx Process Failed               : 0
Total Enable Rx socket on Client Stack Close : 0
Total Enable Rx socket on Server Stack Close : 0
```

```
AppQoE_R2#
```

Using TCPOpt with other AppQoE / UTD Featutes starting from 17.2

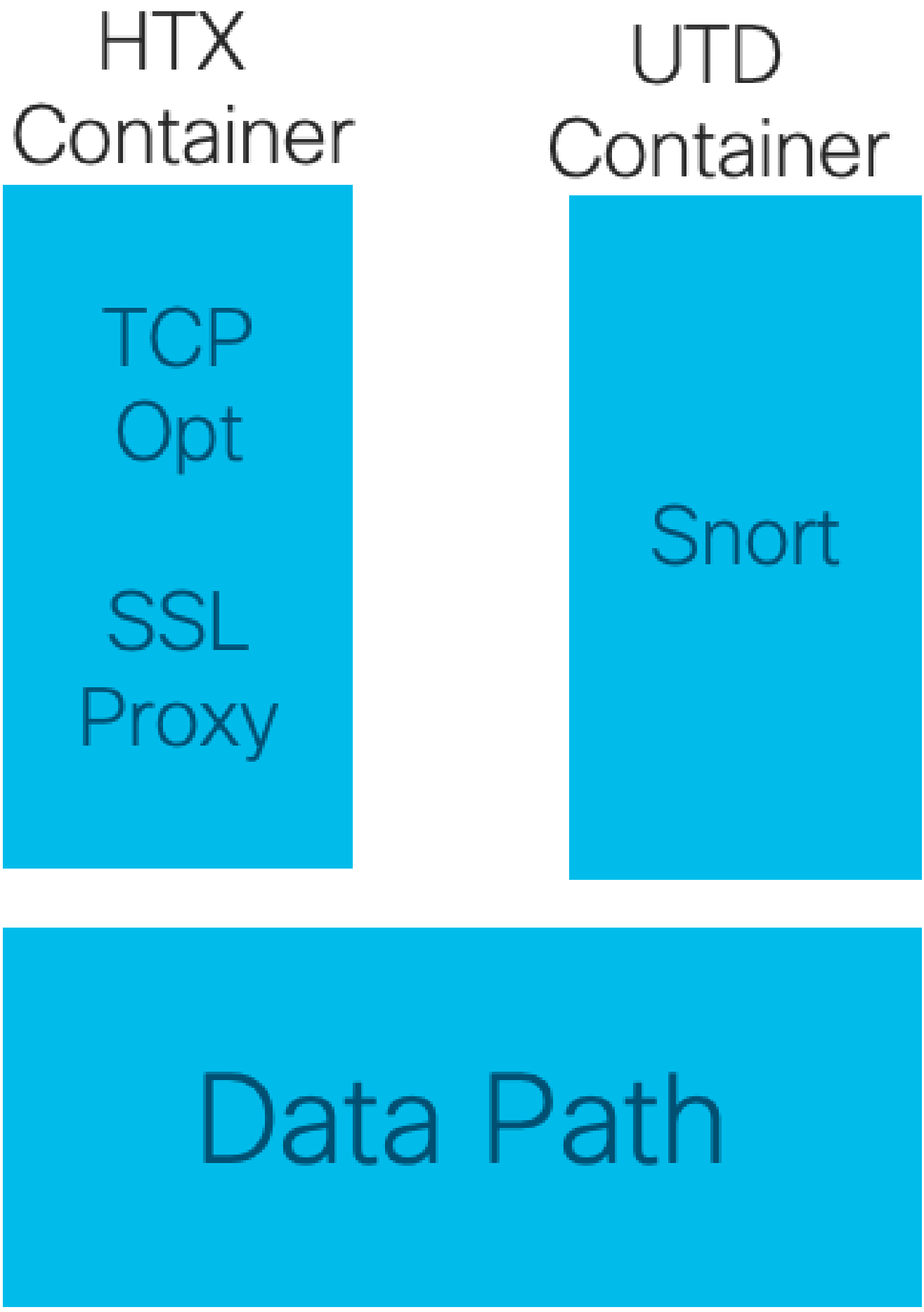
In 16.12 the primary use case for TCPOpt is Branch-to-Branch. There is a separate redirection to TCP Proxy and separate redirection to UTD container in 16.12, that's why TCP Opt does not work together with Security in 16.12

In 17.2 a centralized policy path is implemented, which will detect the need for TCP Opt and Security.

Releated packets will redirect to service plane (punted) only once.

Different flow options are possible staring from 17.2:

1. TCP Opt only
2. UTD Only
3. TCP Opt -> UTD
4. TCP Opt -> SSL Proxy -> UTD


HTX
Container
UTD
Container
TCP
Opt
SSL
Proxy
Snort
Data Path

Related Information

- Release Notes for Cisco IOS XE SD-WAN Release 16.12.x
- **Technical Support & Documentation - Cisco Systems**