Mejore el rendimiento de Catalyst 8000V con Multi-TxQs en AWS

Contenido

Introducción

Este documento describe cómo habilitar y utilizar Multi-TXQs en Catalyst 8000V implementado en entornos AWS para mejorar el rendimiento.

Antecedentes

La presencia de varias colas simplifica y acelera el proceso de asignación de paquetes entrantes y salientes a una vCPU determinada. El uso de Multi-TXQs en Catalyst 8000V permite una utilización eficiente del núcleo a través de los núcleos de plano de datos disponibles asignados, lo que se traduce en un mayor rendimiento. Este artículo proporciona una ligera descripción general del funcionamiento de Multi-TXQ, cómo se configura, muestra ejemplos de configuraciones CLI para implementaciones de Catalyst 8000V autónomas y de SD-WAN y revisa los comandos de solución de problemas para ayudar a encontrar cuellos de botella de rendimiento.

Comportamiento de Catalyst 8000V cuando no se utiliza Multi-TxQs

Hasta la versión de software 17.18, los paquetes que entran en Catalyst 8000V se distribuyen a todos los vCPU (núcleos de procesamiento de paquetes) independientemente de los flujos. Una vez que PP completa el procesamiento de paquetes, se restaura el orden de flujo para enviarlo a una interfaz.

Antes de colocar el paquete en una cola de transmisión (TxQ), el Catalyst 8000V crea un TxQ por interfaz. Por lo tanto, si solo hay una interfaz de salida disponible, entonces múltiples flujos van a un TxQ.

El Catalyst 8000V no puede aprovechar este proceso Multi-TxQ si solo hay una interfaz disponible. Esto se traduce en cuellos de botella de rendimiento y distribución de carga desigual en los núcleos de plano de datos disponibles. Si solo hay una interfaz de salida que se utiliza para transmitir datos desde la instancia de C8000V, solo hay un TxQ disponible para transmitir el tráfico de red y, posiblemente, hacer que los paquetes se descarten debido a que la única cola se llena más rápido.

Como referencia, puede encontrar el modelo de arquitectura TxQ única para Catalyst 8000V implementado en AWS aquí en la Figura 1.

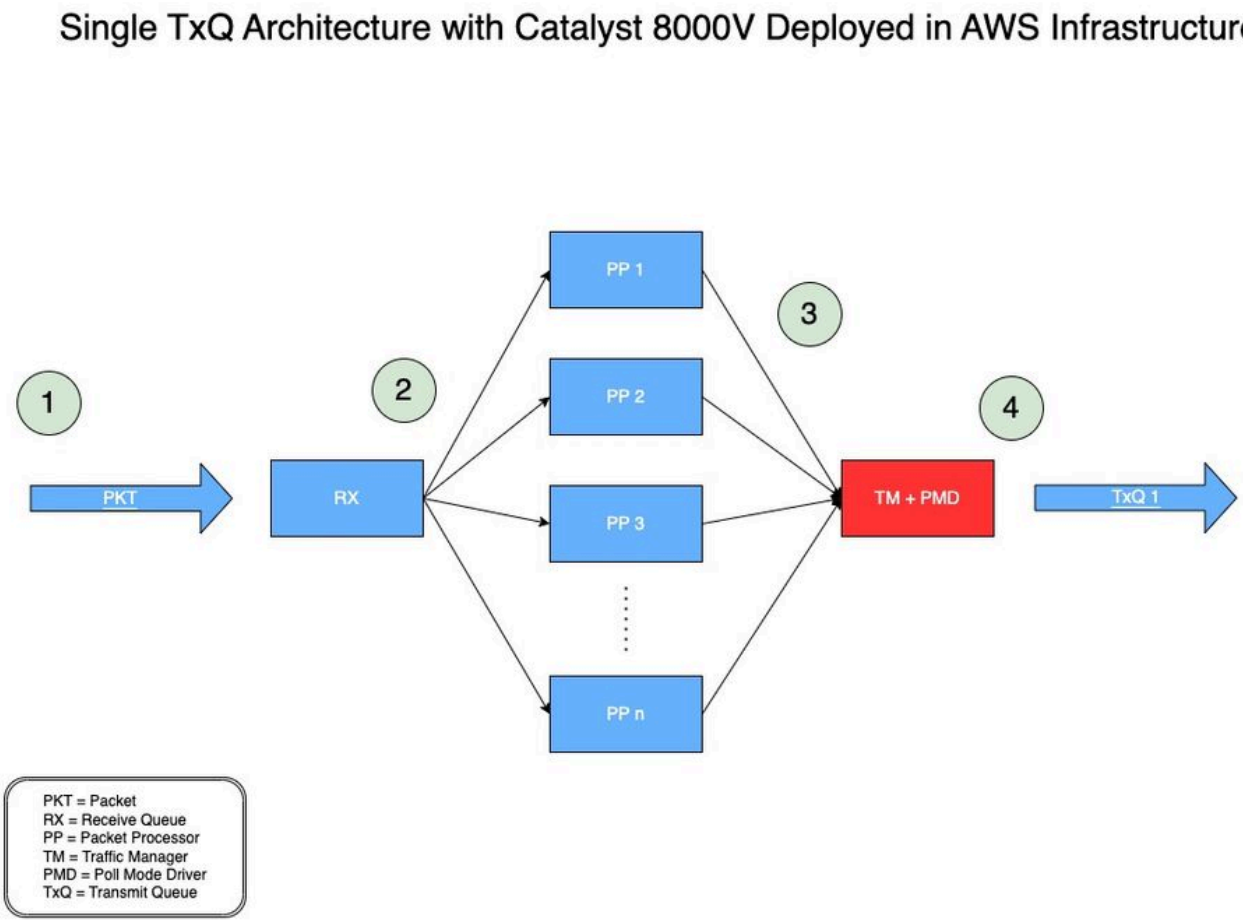


Figura 1: Modelo de arquitectura TxQ única para Catalyst 8000V implementado en AWS.

1. Un paquete de red (PKT) atraviesa una VPC y se recibe en la interfaz de entrada de un C8000V.
2. El PKT se coloca en una cola de recepción (RX) y luego se reenvía a un motor de procesador de paquetes (PSP) decidido por un algoritmo.
3. Una vez que el procesador de paquetes (PSP) procesa el paquete, lo envía a Traffic Manager (TM).
4. Al final del procesamiento de TM, un núcleo es responsable de colocar el paquete en el único TxQ disponible que luego se reenvía a la interfaz de egreso del Catalyst 8000V.

¿Qué son las Multi-TXQs en la Infraestructura AWS?

AWS ENA proporciona varias colas de transmisión (Multi-TxQ) para reducir la sobrecarga interna y aumentar la escalabilidad. La presencia de varias colas simplifica y acelera el proceso de asignación de paquetes entrantes y salientes a una vCPU determinada. El modelo de referencia de red AWS y DPDK se basa en el flujo, donde cada vCPU procesa un flujo y transmite paquetes de ese flujo a una cola de transmisión asignada (TxQ). El par de cola RX/TX para cada vCPU es válido según el modelo basado en el flujo.

Debido a que el Catalyst 8000V NO se basa en el flujo, la sentencia "RX/TX queue pair for each vCPU" no se aplica al Catalyst 8000V.

En este caso, las colas RX/TX no son por vCPU sino por interfaz. Las colas RX/TX actúan como interfaces entre la aplicación (Catalyst 8000V) y la infraestructura/hardware AWS para enviar tráfico de datos/red. AWS controla la velocidad y el número de colas RX/TX disponibles por interfaz en función de la instancia.

El Catalyst 8000V debe tener varias interfaces para crear varios TxQ. Para mantener el orden de flujo con varios flujos saliendo de una interfaz (una vez que el Catalyst 8000V habilita varios TxQs siguiendo este proceso), el Catalyst 8000V hace hash de los flujos basados en las 5 tuplas para elegir el TxQ apropiado. Un usuario puede crear varias interfaces en el Catalyst 8000V utilizando la misma NIC física conectada a la instancia mediante interfaces de loopback o direcciones IP secundarias.

En la Figura 2, puede encontrar cómo se procesa un paquete utilizando la arquitectura Multi-TxQ con Catalyst 8000V en AWS.

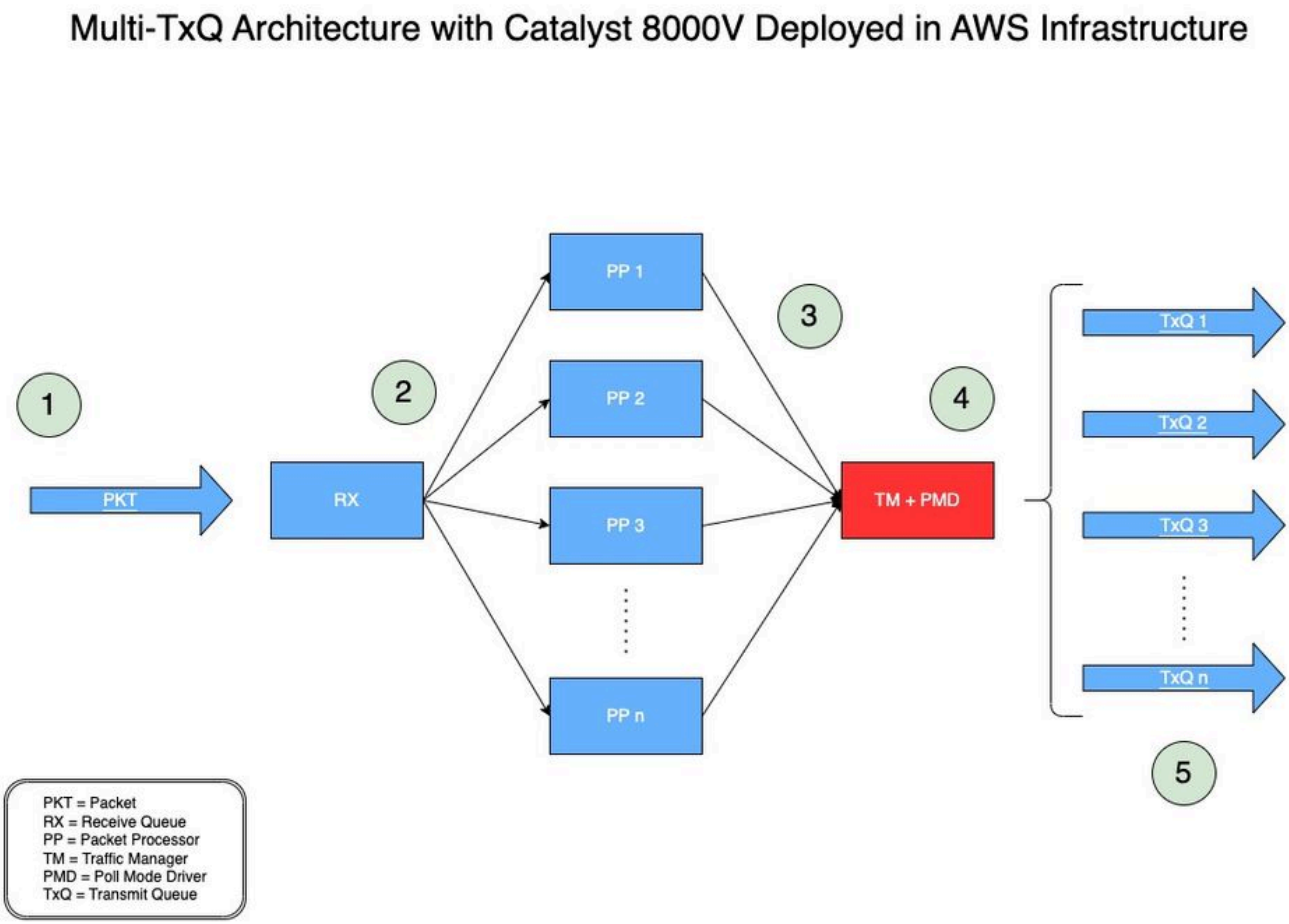


Figura 2: Modelo de arquitectura Multi-TxQ para Catalyst 8000V implementado en AWS.

1. Un paquete de red (PKT) atraviesa una VPC y se recibe en la interfaz de entrada de

un C8000V.
2. El PKT se coloca en una cola de recepción (RX) y luego se reenvía a un motor de procesador de paquetes (PSP) decidido por un algoritmo.
3. Una vez que el procesador de paquetes (PSP) procesa el paquete, lo envía a Traffic Manager (TM).
4. Al final del procesamiento de la TM, antes de colocar el paquete en una cola de transmisión (TxQ), la TM observa el encabezado del paquete y hace un hashes del paquete (explicado en la siguiente sección). Otro componente, el controlador de modo de sondeo (PMD), se utiliza para configurar el número de TXQ admitidos por la instancia. Un núcleo está dedicado a la función TM + PMD, que realiza el hashing y el envío del paquete al TxQ asignado.
5. El TxQ se selecciona en función de las cinco tuplas de troceo y módulo con el número de TxQ soportado por la instancia. Los paquetes se colocan en el TxQ seleccionado y se reenvían a la interfaz de salida de Catalyst 8000V.

Cómo se codifica el tráfico en Multi-TxQs

Al final del procesamiento de TM, como se muestra en el paso 4 de la figura 2, antes de colocar el paquete en un TxQ, el TM observa el encabezado del paquete y extrae las 5 tuplas (dirección de destino, dirección de origen, protocolo, puerto de destino y puerto de origen) y envía el paquete a un TxQ.

El TxQ se selecciona en función de las cinco tuplas de troceo y módulo con el número de TxQ soportado por la instancia.

Versiones de software de Catalyst 8000V compatibles con Multi-TxQs

Todas las instancias AWS EC2 del mismo tipo de familia de instancias admiten un número diferente de TXQ, dependiendo del tamaño de la instancia. El C8000V comenzó a admitir varios TxQ a partir de IOS® XE 17.7.

A partir de IOS® XE 17.7, el C8000V admite varios TxQ en C5n.9xlarge que pueden tener hasta 8 TXQ.

A partir de IOS® XE 17.9, el C8000V admite el tamaño de instancia grande C5n.18x, que puede tener hasta 12 TXQ (un 50% más que C5n.9xlarge).

Aunque Multi-TxQ es compatible con IOS® XE 17.7, se recomienda ALTAMENTE utilizar IOS® XE 17.9 tanto para el ciclo de vida del software como para capacidades de rendimiento de rendimiento superior con compatibilidad con 12 TxQ.

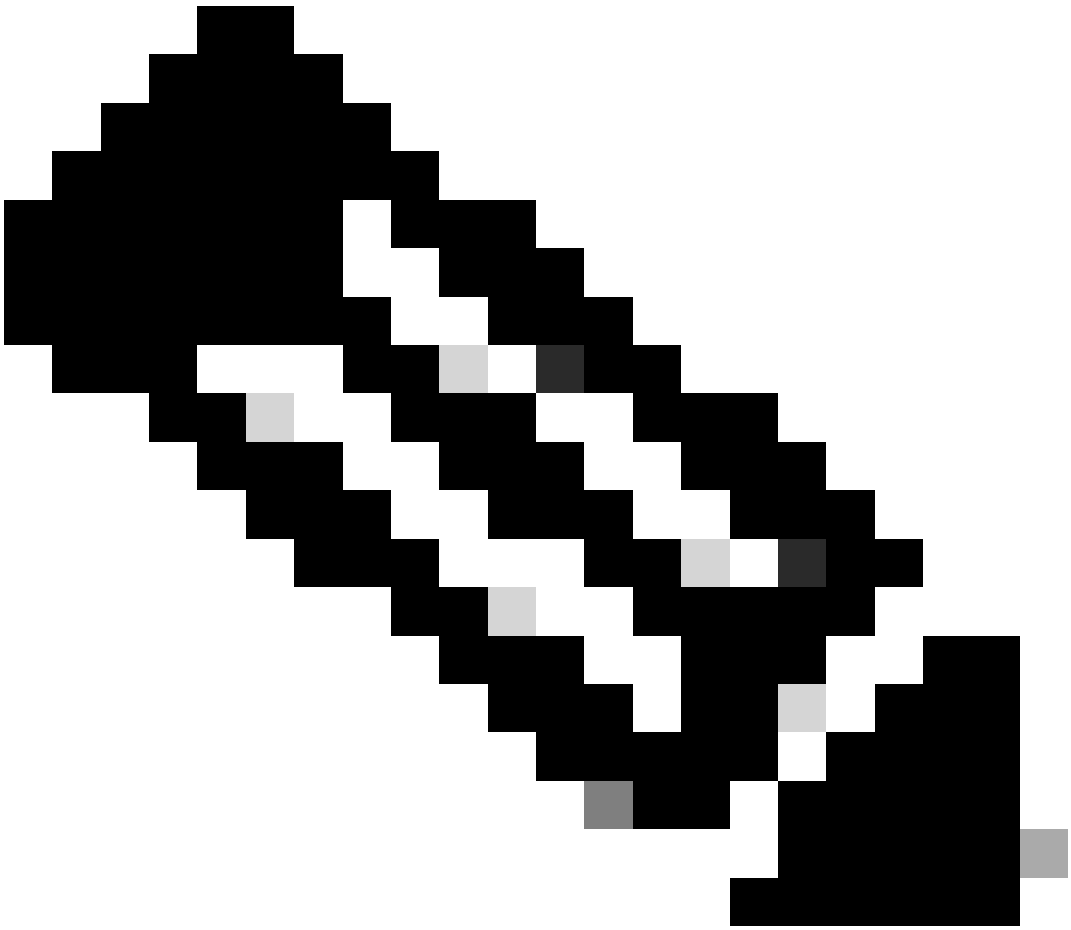
Cómo diseñar el esquema de direcciones IP para calcular el hash

Para realizar un hash uniforme del tráfico entre todos los TxQ disponibles, se deben utilizar direcciones IP especiales cuando Catalyst 8000V está terminando túneles IPsec/GRE.

Hay scripts públicos disponibles para generar estas direcciones IP especiales para ser utilizados en la configuración de las interfaces Catalyst 8000V que son responsables de terminar estos túneles. Esta sección proporciona instrucciones sobre cómo descargar y utilizar los scripts para diseñar las direcciones IP requeridas incluso para el hashing Multi-TxQ.

Si el Catalyst 8000V está manejando tráfico de texto claro como TCP/UDP, no se requiere ningún esquema de direccionamiento IP especial.

Las instrucciones originales se pueden encontrar aquí: https://github.com/CiscoDevNet/python-c8000v-aws-multitx-queues/



Nota: Para Catalyst 8000V que ejecuta 17.18 o posterior, los paquetes se distribuyen de manera diferente. Por lo tanto, se debe utilizar un algoritmo de hashing diferente.

Prerequisites

- Debe tener una máquina Linux/MacOS o Windows capaz de ejecutar scripts Python.
- Verifique que la versión de Python sea 3.8.9 o superior; comprobar la versión de Python con 'python3 —version'
- Instale PIP si aún no está instalado. Si no es así, ejecute:
  - curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  - python3 get-pip.py

Puede comprobar qué versión de Python utiliza su máquina con el comando 'python3 —version'.

```
user@computer ~ % python3 --version

Python 3.9.6
```

Una vez verificada la versión de Python y ejecutada, una versión igual o superior a 3.8.9, instala la última versión de PIP.

```
user@computer ~ % curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current

                                 Dload  Upload   Total   Spent    Left  Speed

100 2570k  100 2570k    0     0  6082k      0 --:--:-- --:--:-- --:--:-- 6135k
```

<#root>

```
user@computer ~ % python3 get-pip.py

Defaulting to user installation because normal site-packages is not writeable
Collecting pip
  Downloading pip-23.3.1-py3-none-any.whl.metadata (3.5 kB)
Downloading pip-23.3.1-py3-none-any.whl (2.1 MB)

   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 7.4 MB/s eta 0:00:00

Installing collected packages: pip

  WARNING: The scripts pip, pip3 and pip3.9 are installed in '/Users/name/Library/Python/3.9/bin' which

  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-scri

Successfully installed pip-23.3.1
```

[

```
notice
]
 A new release of pip is available: 21.2.4 -> 23.3.1
[
notice
]
 To update, run: /Applications/Xcode.app/Contents/Developer/usr/bin/python3 -m pip install --upgrade pi
```

Crear entorno virtual

Una vez que se hayan instalado los requisitos previos, cree el entorno virtual y descargue la secuencia de comandos de hash de direcciones IP utilizada para generar el esquema de direcciones IP únicas para Multi-TxQ.

Resumen de comandos:

1. python3 -m venv c8kv-hash
2. cd c8kv-hash
3. source bin/activate
4. git clone https://github.com/CiscoDevNet/python-c8000v-aws-multitx-queues/
5. cd c8kv-aws-pmd-hash
6. python3 -m pip install —upgrade pip
7. pip install -r requirements.txt

Los entornos virtuales de Python se utilizan para crear espacios de trabajo aislados que no afectan a otros proyectos o dependencias. Cree el entorno virtual 'c8kv-hash' mediante este comando:

```
user@computer Desktop % python3 -m venv c8kv-hash
```

Navegue dentro del entorno virtual hasta la carpeta 'c8kv-hash' (creada anteriormente).

```
user@computer Desktop % cd c8kv-hash
```

Active el entorno virtual.

```
user@computer c8kv-hash % source bin/activate
```

Clonar el repositorio que tiene el script de Python de hash Multi-TxQ.

```
(c8kv-hash) user@computer c8kv-hash % git clone https://github.com/CiscoDevNet/python-c8000v-aws-multit

Cloning into 'c8kv-aws-pmd-hash'...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (59/59), done.
remote: Total 82 (delta 34), reused 57 (delta 19), pack-reused 0
Receiving objects: 100% (82/82), 13.01 KiB | 2.60 MiB/s, done.
Resolving deltas: 100% (34/34), done.
```

Una vez clonado el repositorio, desplácese a la carpeta 'c8kv-aws-pmd-hash'. Dado que se encuentra en el entorno virtual creado, instale la última versión de PIP.

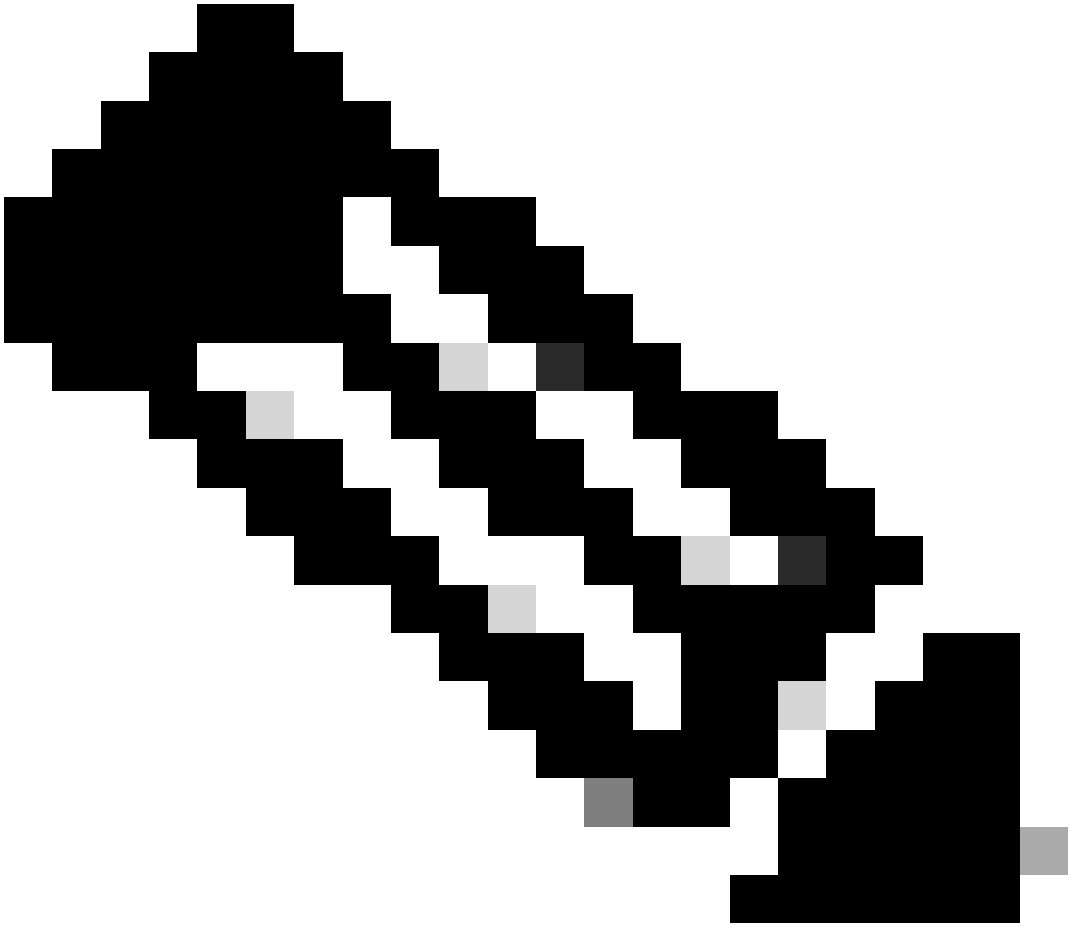
```
(c8kv-hash) user@computer c8kv-hash % cd c8kv-aws-pmd-hash
(c8kv-hash) user@computer c8kv-aws-pmd-hash % python3 -m pip install --upgrade pip

Requirement already satisfied: pip in /Users/name/Desktop/c8kv-hash/lib/python3.9/site-packages (21.2.4)
Collecting pip
  Downloading pip-23.3.1-py3-none-any.whl (2.1 MB)
     |████████████████████████████████| 2.1 MB 2.7 MB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 21.2.4
    Uninstalling pip-21.2.4:
      Successfully uninstalled pip-21.2.4
Successfully installed pip-23.3.1
```

Una vez actualizado PIP, instale las dependencias que se encuentran en el archivo requirements.txt de la carpeta.

```
(c8kv-hash) user@computer c8kv-aws-pmd-hash % pip install -r requirements.txt
Collecting crc32c==2.3 (from -r requirements.txt (line 1))
  Downloading crc32c-2.3-cp39-cp39-macosx_11_0_arm64.whl (27 kB)
Installing collected packages: crc32c
Successfully installed crc32c-2.3
```

El entorno virtual está ahora actualizado y se puede utilizar para generar el esquema de direcciones IP para Multi-TxQ.

Calcular esquema de direcciones IP mediante script de índice de hash de Python para las versiones 17.7 y 17.8 (obsoleto)



Nota: LAS SECUENCIAS DE COMANDOS HASH 7.7 Y 17.8 DEJARÁN DE UTILIZARSE PRONTO. SE RECOMIENDA ENCARECIDAMENTE UTILIZAR UN SCRIPT HASH 17.9

Resumen de comandos:

- python3 c8kv_multitxq_hash.py —old_crc 1 —dest_network 192.168.1.0/24 —src_network 192.168.2.0/24 --unique_hash 1

'—old_crc 1' genera un índice hash basado en la versión 17.7 y 17.8 con el módulo 8 para que coincida con el TXQ PMD admitido (NO modificar)

'—dest_network' define la subred de direcciones de red de destino (modificación basada en el esquema de direcciones IP de la red)

'—src_network' define la subred de la dirección de red de origen (modificación basada en el esquema de direcciones IP de la red)

'—unique_hash 1' genera un conjunto (8 pares para 8 TXQ) de direcciones IP con hash único.

Esto se puede modificar.

<#root>

```
(c8kv-hash) user@computer c8kv-aws-pmd-hash % python3 c8kv_multitxq_hash.py --old_crc 1 --dest_network 1

Dest:            Src:             Prot  dstport  srcport          Hash: Rev-hash:

192.168.1.0      192.168.2.0      2     5
192.168.1.0      192.168.2.1      2     7
192.168.1.0      192.168.2.2      2     1
192.168.1.0      192.168.2.3      2     3
192.168.1.0      192.168.2.4      2     5
192.168.1.0      192.168.2.5      2     7
192.168.1.0      192.168.2.6      2     1
192.168.1.0      192.168.2.7      2     3
192.168.1.0      192.168.2.8      2     5
192.168.1.0      192.168.2.9      2     7
192.168.1.0      192.168.2.10     2     1
.
. ### trimmed output ###
.
192.168.1.255    192.168.2.247    5     2
192.168.1.255    192.168.2.248    5     4
192.168.1.255    192.168.2.249    5     6
192.168.1.255    192.168.2.250    5     0
192.168.1.255    192.168.2.251    5     2
192.168.1.255    192.168.2.252    5     4
192.168.1.255    192.168.2.253    5     6
192.168.1.255    192.168.2.254    5     0
192.168.1.255    192.168.2.255    5     2

Unique hash:

------ Tunnels set 0 ---------

192.168.1.37<===>192.168.2.37<===>0

192.168.1.129<===>192.168.2.129<===>1

192.168.1.36<===>192.168.2.36<===>2

192.168.1.128<===>192.168.2.128<===>3

192.168.1.39<===>192.168.2.39<===>4

192.168.1.131<===>192.168.2.131<===>5

192.168.1.38<===>192.168.2.38<===>6
```

192.168.1.130<===>192.168.2.130<===>7

Calcular esquema de direcciones IP mediante script de índice de hash de Python para las versiones 17.9 y posteriores

Resumen de comandos:

- python3 c8kv_multitxq_hash.py —dest_network 192.168.1.0/24 —src_network 192.168.2.0/24 —port udp —src_port 12346 —dst_port 12346 —unique_hash 1

Tenga en cuenta que en IOS® XE versión 17.9 y posteriores, la secuencia de comandos utiliza el módulo 12 sin la opción —old_crc, que coincide con el PMD TXQ compatible.

'—dest_network' define la subred de direcciones de red de destino (modificación basada en el esquema de direcciones IP de la red)

'—src_network' define la subred de la dirección de red de origen (modificación basada en el esquema de direcciones IP de la red)

'—port udp' define el protocolo utilizado. El usuario puede especificar el parámetro de protocolo como "gre" o "tcp" o "udp" o cualquier valor decimal (OPCIONAL)

'—src_port' define el puerto de origen utilizado (OPCIONAL)

'—dst_port' define el puerto de destino utilizado (OPCIONAL)

'—unique_hash 1' genera un conjunto (12 pares para 12 TXQ) de direcciones IP con hash único. Esto se puede modificar.

<#root>

```
(c8kv-hash) user@computer c8kv-aws-pmd-hash % python3 c8kv_multitxq_hash.py --dest_network 192.168.1.0/

Dest:            Src:             Prot  dstport  srcport          Hash: Rev-hash:

192.168.1.0      192.168.2.0      17    12346    12346       ==>    4        4       <-- Unique Hash Va
192.168.1.0      192.168.2.1      17    12346    12346       ==>    4        4
192.168.1.0      192.168.2.2      17    12346    12346       ==>    8        8       <-- Unique Hash Va
192.168.1.0      192.168.2.3      17    12346    12346       ==>    0        0       <-- Unique Hash Va
192.168.1.0      192.168.2.4      17    12346    12346       ==>    0        0
192.168.1.0      192.168.2.5      17    12346    12346       ==>    0        0
192.168.1.0      192.168.2.6      17    12346    12346       ==>    4        4
192.168.1.0      192.168.2.7      17    12346    12346       ==>    0        0
192.168.1.0      192.168.2.8      17    12346    12346       ==>    9        9       <-- Unique Hash Va
192.168.1.0      192.168.2.9      17    12346    12346       ==>    9        9
192.168.1.0      192.168.2.10     17    12346    12346       ==>    9        9
192.168.1.0      192.168.2.11     17    12346    12346       ==>    1        1       <-- Unique Hash Va
192.168.1.0      192.168.2.12     17    12346    12346       ==>    1        1
.
. ### trimmed output ###
.
```

```
192.168.1.255    192.168.2.250    17    12346    12346       ==>    1        1
192.168.1.255    192.168.2.251    17    12346    12346       ==>    1        1
192.168.1.255    192.168.2.252    17    12346    12346       ==>    9        9
192.168.1.255    192.168.2.253    17    12346    12346       ==>    1        1
192.168.1.255    192.168.2.254    17    12346    12346       ==>    5        5       <-- Unique Hash Va
192.168.1.255    192.168.2.255    17    12346    12346       ==>    9        9

Unique hash:

------ Tunnels set 0 ---------


192.168.1.38 <===> 192.168.2.38<===>0


192.168.1.37 <===> 192.168.2.37<===>1


192.168.1.53 <===> 192.168.2.53<===>2


192.168.1.39 <===> 192.168.2.39<===>3


192.168.1.48 <===> 192.168.2.48<===>4


192.168.1.58 <===> 192.168.2.58<===>5


192.168.1.42 <===> 192.168.2.42<===>6


192.168.1.46 <===> 192.168.2.46<===>7


192.168.1.40 <===> 192.168.2.40<===>8


192.168.1.43 <===> 192.168.2.43<===>9


192.168.1.36 <===> 192.168.2.36<===>10


192.168.1.56 <===> 192.168.2.56<===>11
```

Topología de ejemplo y configuración CLI mediante 8 TXQ con interfaces de loopback

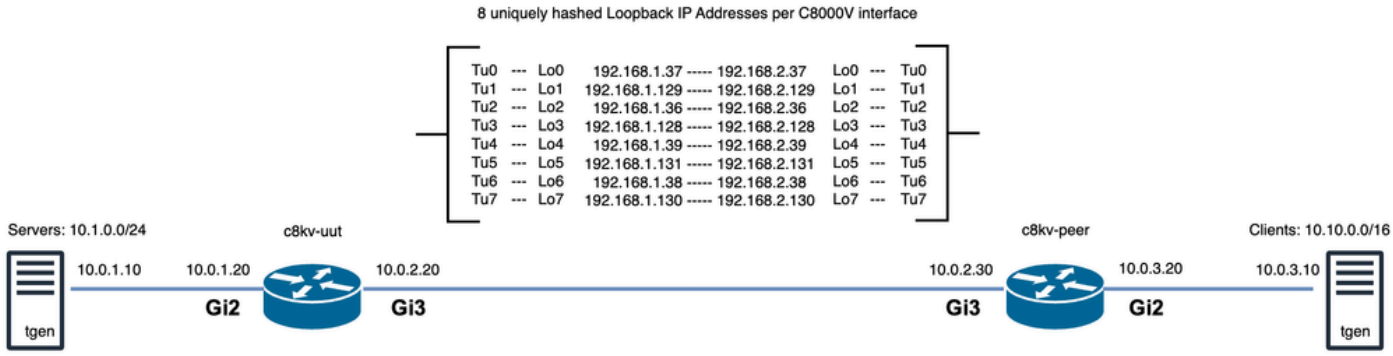


Figura 3: Topología de ejemplo que utiliza ocho TxQs mediante interfaces de loopback.

Ésta es una configuración CLI de ejemplo para 'c8kv-uut' (Figura 3) que crea ocho túneles IPsec con interfaces de loopback utilizando las direcciones IP hash calculadas (192.168.1.X) de la sección anterior.

Una configuración similar se aplicaría en el otro extremo del router (c8kv-peer) con las ocho direcciones IP hash calculadas restantes (192.168.2.X).

```
ip cef load-sharing algorithm include-ports source destination 00ABC123

crypto keyring tunnel0
  local-address Loopback0
  pre-shared-key address 192.168.2.37 key cisco
crypto keyring tunnel1
  local-address Loopback1
  pre-shared-key address 192.168.2.129 key cisco
crypto keyring tunnel2
  local-address Loopback2
  pre-shared-key address 192.168.2.36 key cisco
crypto keyring tunnel3
  local-address Loopback3
  pre-shared-key address 192.168.2.128 key cisco
crypto keyring tunnel4
  local-address Loopback4
  pre-shared-key address 192.168.2.39 key cisco
crypto keyring tunnel5
  local-address Loopback5
  pre-shared-key address 192.168.2.131 key cisco
crypto keyring tunnel6
  local-address Loopback6
  pre-shared-key address 192.168.2.38 key cisco
crypto keyring tunnel7
  local-address Loopback7
  pre-shared-key address 192.168.2.130 key cisco

crypto isakmp policy 200
 encryption aes
 hash sha
 authentication pre-share
 group 16
 lifetime 28800
```

```
crypto isakmp profile isakmp-tunnel0
   keyring tunnel0
   match identity address 0.0.0.0
   local-address Loopback0
crypto isakmp profile isakmp-tunnel1
   keyring tunnel1
   match identity address 0.0.0.0
   local-address Loopback1
crypto isakmp profile isakmp-tunnel2
   keyring tunnel2
   match identity address 0.0.0.0
   local-address Loopback2
crypto isakmp profile isakmp-tunnel3
   keyring tunnel3
   match identity address 0.0.0.0
   local-address Loopback3
crypto isakmp profile isakmp-tunnel4
   keyring tunnel4
   match identity address 0.0.0.0
   local-address Loopback4
crypto isakmp profile isakmp-tunnel5
   keyring tunnel5
   match identity address 0.0.0.0
   local-address Loopback5
crypto isakmp profile isakmp-tunnel6
   keyring tunnel6
   match identity address 0.0.0.0
   local-address Loopback6
crypto isakmp profile isakmp-tunnel7
   keyring tunnel7
   match identity address 0.0.0.0
   local-address Loopback7

crypto ipsec transform-set ipsec-prop-vpn-tunnel esp-gcm 256
 mode tunnel
crypto ipsec df-bit clear

crypto ipsec profile ipsec-vpn-tunnel
 set transform-set ipsec-prop-vpn-tunnel
 set pfs group16

interface Loopback0
 ip address 192.168.1.37 255.255.255.255
!
interface Loopback1
 ip address 192.168.1.129 255.255.255.255
!
interface Loopback2
 ip address 192.168.1.36 255.255.255.255
!
interface Loopback3
 ip address 192.168.1.128 255.255.255.255
!
interface Loopback4
 ip address 192.168.1.39 255.255.255.255
!
interface Loopback5
 ip address 192.168.1.131 255.255.255.255
!
interface Loopback6
 ip address 192.168.1.38 255.255.255.255
```

```
!
interface Loopback7
 ip address 192.168.1.130 255.255.255.255
!

interface Tunnel0
 ip address 10.101.100.101 255.255.255.0
 load-interval 30
 tunnel source Loopback0
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.37
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel1
 ip address 10.101.101.101 255.255.255.0
 load-interval 30
 tunnel source Loopback1
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.129
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel2
 ip address 10.101.102.101 255.255.255.0
 load-interval 30
 tunnel source Loopback2
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.36
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel3
 ip address 10.101.103.101 255.255.255.0
 load-interval 30
 tunnel source Loopback3
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.128
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel4
 ip address 10.101.104.101 255.255.255.0
 load-interval 30
 tunnel source Loopback4
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.39
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel5
 ip address 10.101.105.101 255.255.255.0
 load-interval 30
 tunnel source Loopback5
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.131
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel6
 ip address 10.101.106.101 255.255.255.0
 load-interval 30
 tunnel source Loopback6
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.38
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel7
```

```
 ip address 10.101.107.101 255.255.255.0
 load-interval 30
 tunnel source Loopback7
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.130
 tunnel protection ipsec profile ipsec-vpn-tunnel
!


interface GigabitEthernet2
 mtu 9216
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet3
 mtu 9216
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!

 !  ### IP route from servers to c8kv-uut

ip route 10.1.0.0 255.255.0.0 GigabitEthernet2 10.0.1.10

!  ### IP routes from c8kv-uut to clients on c8kv-peer side, routes are evenly distributed to all 8 TXQ

ip route 10.10.0.0 255.255.0.0 Tunnel0
ip route 10.10.0.0 255.255.0.0 Tunnel1
ip route 10.10.0.0 255.255.0.0 Tunnel2
ip route 10.10.0.0 255.255.0.0 Tunnel3
ip route 10.10.0.0 255.255.0.0 Tunnel4
ip route 10.10.0.0 255.255.0.0 Tunnel5
ip route 10.10.0.0 255.255.0.0 Tunnel6
ip route 10.10.0.0 255.255.0.0 Tunnel7

 !  ### IP route from c8kv-uut Loopback int tunnel endpoint to c8kv-peer Loopback int tunnel endpoints

ip route 192.168.2.0 255.255.255.0 GigabitEthernet3 10.0.2.30
```

Topología de ejemplo y configuración CLI mediante 12 TXQ con interfaces de loopback

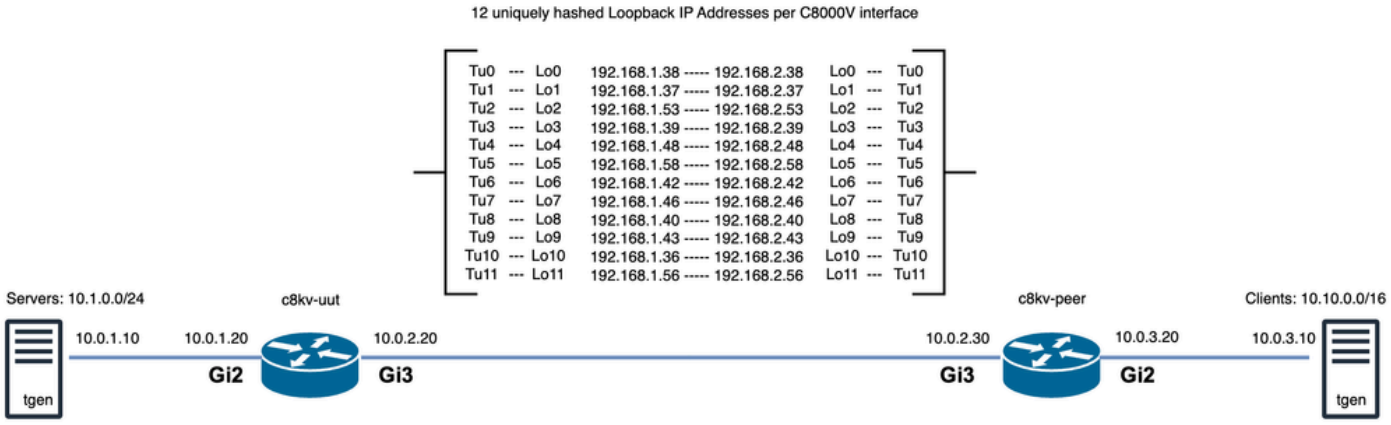


Figura 4. Topología de ejemplo que utiliza doce TxQs mediante interfaces de loopback.

Ésta es una configuración CLI de ejemplo para 'c8kv-uut' (Figura 4) que crea doce túneles IPsec con interfaces de loopback utilizando las direcciones IP hash calculadas (192.168.1.X) de la sección anterior.

Una configuración similar se aplicaría en el otro extremo del router (c8kv-peer) con las ocho direcciones IP hash calculadas restantes (192.168.2.X).

```
ip cef load-sharing algorithm include-ports source destination 00ABC123

crypto keyring tunnel0
  local-address Loopback0
  pre-shared-key address 192.168.2.38 key cisco
crypto keyring tunnel1
  local-address Loopback1
  pre-shared-key address 192.168.2.37 key cisco
crypto keyring tunnel2
  local-address Loopback2
  pre-shared-key address 192.168.2.53 key cisco
crypto keyring tunnel3
  local-address Loopback3
  pre-shared-key address 192.168.2.39 key cisco
crypto keyring tunnel4
  local-address Loopback4
  pre-shared-key address 192.168.2.48 key cisco
crypto keyring tunnel5
  local-address Loopback5
  pre-shared-key address 192.168.2.58 key cisco
crypto keyring tunnel6
  local-address Loopback6
  pre-shared-key address 192.168.2.42 key cisco
crypto keyring tunnel7
  local-address Loopback7
  pre-shared-key address 192.168.2.46 key cisco
crypto keyring tunnel8
  local-address Loopback8
  pre-shared-key address 192.168.2.40 key cisco
crypto keyring tunnel9
  local-address Loopback9
  pre-shared-key address 192.168.2.43 key cisco
```

```
crypto keyring tunnel10
  local-address Loopback10
  pre-shared-key address 192.168.2.36 key cisco
crypto keyring tunnel11
  local-address Loopback11
  pre-shared-key address 192.168.2.56 key cisco


crypto isakmp policy 200
 encryption aes
 hash sha
 authentication pre-share
 group 16
 lifetime 28800
crypto isakmp profile isakmp-tunnel0
   keyring tunnel0
   match identity address 0.0.0.0
   local-address Loopback0
crypto isakmp profile isakmp-tunnel1
   keyring tunnel1
   match identity address 0.0.0.0
   local-address Loopback1
crypto isakmp profile isakmp-tunnel2
   keyring tunnel2
   match identity address 0.0.0.0
   local-address Loopback2
crypto isakmp profile isakmp-tunnel3
   keyring tunnel3
   match identity address 0.0.0.0
   local-address Loopback3
crypto isakmp profile isakmp-tunnel4
   keyring tunnel4
   match identity address 0.0.0.0
   local-address Loopback4
crypto isakmp profile isakmp-tunnel5
   keyring tunnel5
   match identity address 0.0.0.0
   local-address Loopback5
crypto isakmp profile isakmp-tunnel6
   keyring tunnel6
   match identity address 0.0.0.0
   local-address Loopback6
crypto isakmp profile isakmp-tunnel7
   keyring tunnel7
   match identity address 0.0.0.0
   local-address Loopback7
crypto isakmp profile isakmp-tunnel8
   keyring tunnel8
   match identity address 0.0.0.0
   local-address Loopback8
crypto isakmp profile isakmp-tunnel9
   keyring tunnel9
   match identity address 0.0.0.0
   local-address Loopback9
crypto isakmp profile isakmp-tunnel10
   keyring tunnel10
   match identity address 0.0.0.0
   local-address Loopback10
crypto isakmp profile isakmp-tunnel11
   keyring tunnel11
   match identity address 0.0.0.0
   local-address Loopback11
```

```
crypto ipsec transform-set ipsec-prop-vpn-tunnel esp-gcm 256
 mode tunnel
crypto ipsec df-bit clear

crypto ipsec profile ipsec-vpn-tunnel
 set transform-set ipsec-prop-vpn-tunnel
 set pfs group16

interface Loopback0
 ip address 192.168.1.38 255.255.255.255
!
interface Loopback1
 ip address 192.168.1.37 255.255.255.255
!
interface Loopback2
 ip address 192.168.1.53 255.255.255.255
!
interface Loopback3
 ip address 192.168.1.39 255.255.255.255
!
interface Loopback4
 ip address 192.168.1.48 255.255.255.255
!
interface Loopback5
 ip address 192.168.1.58 255.255.255.255
!
interface Loopback6
 ip address 192.168.1.42 255.255.255.255
!
interface Loopback7
 ip address 192.168.1.46 255.255.255.255
!
interface Loopback8
 ip address 192.168.1.40 255.255.255.255
!
interface Loopback9
 ip address 192.168.1.43 255.255.255.255
!
interface Loopback10
 ip address 192.168.1.36 255.255.255.255
!
interface Loopback11
 ip address 192.168.1.56 255.255.255.255

interface Tunnel0
 ip address 10.101.100.101 255.255.255.0
 load-interval 30
 tunnel source Loopback0
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.38
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel1
 ip address 10.101.101.101 255.255.255.0
 load-interval 30
 tunnel source Loopback1
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.37
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel2
```

```
 ip address 10.101.102.101 255.255.255.0
 load-interval 30
 tunnel source Loopback2
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.53
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel3
 ip address 10.101.103.101 255.255.255.0
 load-interval 30
 tunnel source Loopback3
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.39
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel4
 ip address 10.101.104.101 255.255.255.0
 load-interval 30
 tunnel source Loopback4
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.48
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel5
 ip address 10.101.105.101 255.255.255.0
 load-interval 30
 tunnel source Loopback5
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.58
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel6
 ip address 10.101.106.101 255.255.255.0
 load-interval 30
 tunnel source Loopback6
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.42
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel7
 ip address 10.101.107.101 255.255.255.0
 load-interval 30
 tunnel source Loopback7
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.46
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel8
 ip address 10.101.108.101 255.255.255.0
 load-interval 30
 tunnel source Loopback8
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.40
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel9
 ip address 10.101.109.101 255.255.255.0
 load-interval 30
 tunnel source Loopback9
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.43
 tunnel protection ipsec profile ipsec-vpn-tunnel
```

```
!
interface Tunnel10
 ip address 10.101.110.101 255.255.255.0
 load-interval 30
 tunnel source Loopback10
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.36
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel11
 ip address 10.101.111.101 255.255.255.0
 load-interval 30
 tunnel source Loopback11
 tunnel mode ipsec ipv4
 tunnel destination 192.168.2.56
 tunnel protection ipsec profile ipsec-vpn-tunnel


interface GigabitEthernet2
 mtu 9216
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet3
 mtu 9216
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!

  !  ### IP route from c8kv-uut to local servers

ip route 10.1.0.0 255.255.0.0 GigabitEthernet2 10.0.1.10

!  ### IP routes from c8kv-uut to clients on c8kv-peer side, routes are evenly distributed to all 12 TX

ip route 10.10.0.0 255.255.0.0 Tunnel0
ip route 10.10.0.0 255.255.0.0 Tunnel1
ip route 10.10.0.0 255.255.0.0 Tunnel2
ip route 10.10.0.0 255.255.0.0 Tunnel3
ip route 10.10.0.0 255.255.0.0 Tunnel4
ip route 10.10.0.0 255.255.0.0 Tunnel5
ip route 10.10.0.0 255.255.0.0 Tunnel6
ip route 10.10.0.0 255.255.0.0 Tunnel7
ip route 10.10.0.0 255.255.0.0 Tunnel8
ip route 10.10.0.0 255.255.0.0 Tunnel9
ip route 10.10.0.0 255.255.0.0 Tunnel10
ip route 10.10.0.0 255.255.0.0 Tunnel11

 !  ### IP route from c8kv-uut Loopback int tunnel endpoint to c8kv-peer Loopback int tunnel endpoints

ip route 192.168.2.0 255.255.255.0 GigabitEthernet3 10.0.2.30
```

Topología de ejemplo y configuración CLI mediante 12 TXQ con direcciones IP secundarias

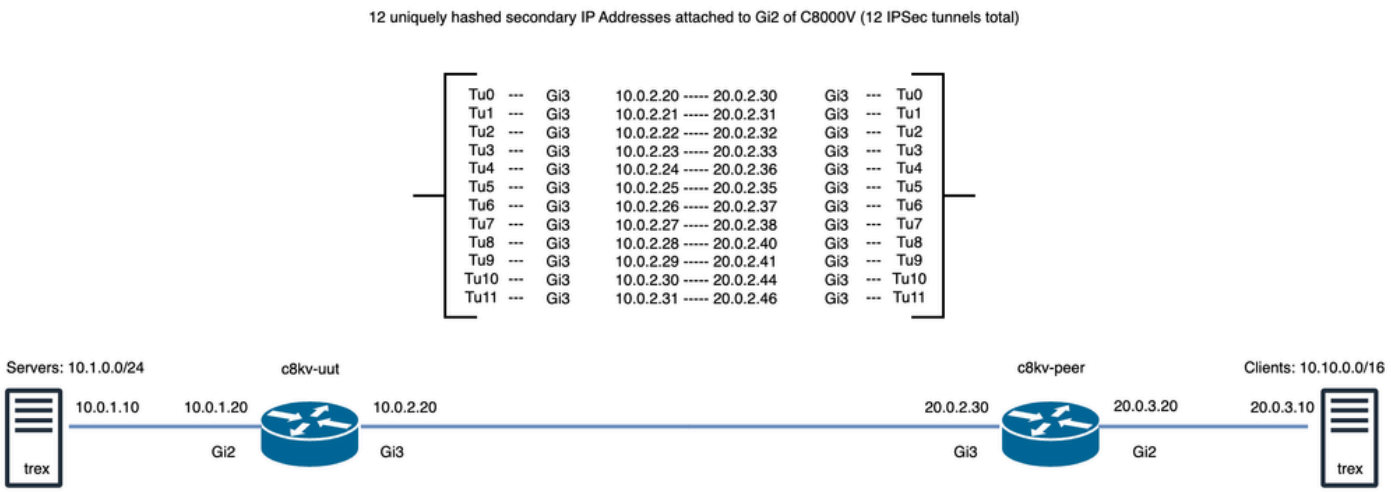
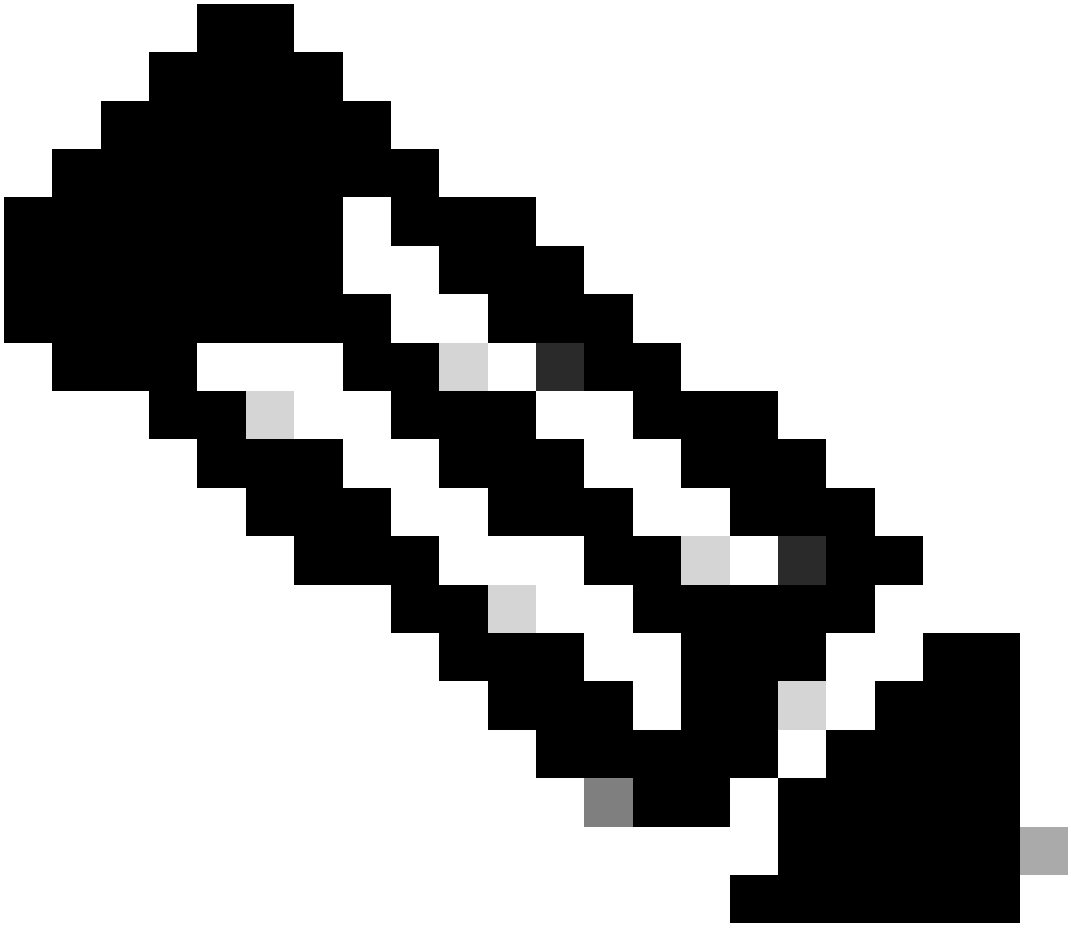


Figura 5. Topología de ejemplo que utiliza doce TxQs usando direcciones IP secundarias.

Si las direcciones de loopback no se pueden utilizar en el entorno AWS, se pueden utilizar en su lugar las direcciones IP secundarias conectadas a ENI.

Ésta es una configuración CLI de ejemplo para 'c8kv-uut' (Figura 5) que crea doce túneles IPsec con la fuente como 1 dirección IP principal + 11 direcciones IP secundarias conectadas a la interfaz GigabitEthernet3 mediante direcciones IP hash calculadas (10.0.2.X). Una configuración similar se aplicaría en el otro extremo del router (c8kv-peer) con las doce direcciones IP hash calculadas restantes (20.0.2.X).



Nota: En este ejemplo, estamos utilizando un segundo C8000V como terminal de túnel, pero también se pueden utilizar otros terminales de red en la nube, como TGW o DX.

```
ip cef load-sharing algorithm include-ports source destination 00ABC123

crypto keyring tunnel0
  local-address 10.0.2.20
  pre-shared-key address 20.0.2.30 key cisco
crypto keyring tunnel1
  local-address 10.0.2.21
  pre-shared-key address 20.0.2.31 key cisco
crypto keyring tunnel2
  local-address 10.0.2.22
  pre-shared-key address 20.0.2.32 key cisco
crypto keyring tunnel3
  local-address 10.0.2.23
  pre-shared-key address 20.0.2.33 key cisco
crypto keyring tunnel4
  local-address 10.0.2.24
  pre-shared-key address 20.0.2.36 key cisco
crypto keyring tunnel5
```

```
  local-address 10.0.2.25
  pre-shared-key address 20.0.2.35 key cisco
crypto keyring tunnel6
  local-address 10.0.2.26
  pre-shared-key address 20.0.2.37 key cisco
crypto keyring tunnel7
  local-address 10.0.2.27
  pre-shared-key address 20.0.2.38 key cisco
crypto keyring tunnel8
  local-address 10.0.2.28
  pre-shared-key address 20.0.2.40 key cisco
crypto keyring tunnel9
  local-address 10.0.2.29
  pre-shared-key address 20.0.2.41 key cisco
crypto keyring tunnel10
  local-address 10.0.2.30
  pre-shared-key address 20.0.2.44 key cisco
crypto keyring tunnel11
  local-address 10.0.2.31
  pre-shared-key address 20.0.2.46 key cisco


crypto isakmp policy 200
 encryption aes
 hash sha
 authentication pre-share
 group 16
 lifetime 28800
crypto isakmp profile isakmp-tunnel0
   keyring tunnel0
   match identity address 20.0.2.30 255.255.255.255
   local-address 10.0.2.20
crypto isakmp profile isakmp-tunnel1
   keyring tunnel1
   match identity address 20.0.2.31 255.255.255.255
   local-address 10.0.2.21
crypto isakmp profile isakmp-tunnel2
   keyring tunnel2
   match identity address 20.0.2.32 255.255.255.255
   local-address 10.0.2.22
crypto isakmp profile isakmp-tunnel3
   keyring tunnel3
   match identity address 20.0.2.33 255.255.255.255
   local-address 10.0.2.23
crypto isakmp profile isakmp-tunnel4
   keyring tunnel4
   match identity address 20.0.2.36 255.255.255.255
   local-address 10.0.2.24
crypto isakmp profile isakmp-tunnel5
   keyring tunnel5
   match identity address 20.0.2.35 255.255.255.255
   local-address 10.0.2.25
crypto isakmp profile isakmp-tunnel6
   keyring tunnel6
   match identity address 20.0.2.37 255.255.255.255
   local-address 10.0.2.26
crypto isakmp profile isakmp-tunnel7
   keyring tunnel7
   match identity address 20.0.2.38 255.255.255.255
   local-address 10.0.2.27
crypto isakmp profile isakmp-tunnel8
   keyring tunnel8
```

```
   match identity address 20.0.2.40 255.255.255.255
   local-address 10.0.2.28
crypto isakmp profile isakmp-tunnel9
   keyring tunnel9
   match identity address 20.0.2.41 255.255.255.255
   local-address 10.0.2.29
crypto isakmp profile isakmp-tunnel10
   keyring tunnel10
   match identity address 20.0.2.44 255.255.255.255
   local-address 10.0.2.30
crypto isakmp profile isakmp-tunnel11
   keyring tunnel11
   match identity address 20.0.2.46 255.255.255.255
   local-address 10.0.2.31


crypto ipsec transform-set ipsec-prop-vpn-tunnel esp-gcm 256
 mode tunnel
crypto ipsec df-bit clear

crypto ipsec profile ipsec-vpn-tunnel
 set transform-set ipsec-prop-vpn-tunnel
 set pfs group16

interface Tunnel0
 ip address 10.101.100.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.20
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.30
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel1
 ip address 10.101.101.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.21
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.31
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel2
 ip address 10.101.102.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.22
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.32
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel3
 ip address 10.101.103.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.23
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.33
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel4
 ip address 10.101.104.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.24
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.36
```

```
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel5
 ip address 10.101.105.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.25
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.35
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel6
 ip address 10.101.106.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.26
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.37
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel7
 ip address 10.101.107.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.27
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.38
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel8
 ip address 10.101.108.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.28
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.40
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel9
 ip address 10.101.109.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.29
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.41
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel10
 ip address 10.101.110.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.30
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.44
 tunnel protection ipsec profile ipsec-vpn-tunnel
!
interface Tunnel11
 ip address 10.101.111.101 255.255.255.0
 load-interval 30
 tunnel source 10.0.2.31
 tunnel mode ipsec ipv4
 tunnel destination 20.0.2.46
 tunnel protection ipsec profile ipsec-vpn-tunnel
!


interface GigabitEthernet2
 mtu 9216
```

```
 ip address dhcp
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet3
 mtu 9216
 ip address 10.0.2.20 255.255.255.0
 ip address 10.0.2.21 255.255.255.0 secondary
 ip address 10.0.2.22 255.255.255.0 secondary
 ip address 10.0.2.23 255.255.255.0 secondary
 ip address 10.0.2.24 255.255.255.0 secondary
 ip address 10.0.2.25 255.255.255.0 secondary
 ip address 10.0.2.26 255.255.255.0 secondary
 ip address 10.0.2.27 255.255.255.0 secondary
 ip address 10.0.2.28 255.255.255.0 secondary
 ip address 10.0.2.29 255.255.255.0 secondary
 ip address 10.0.2.30 255.255.255.0 secondary
 ip address 10.0.2.31 255.255.255.0 secondary
 load-interval 30
 speed 25000
 no negotiation auto
 no mop enabled
 no mop sysid
!

  !  ### IP route from c8kv-uut to local servers

ip route 10.1.0.0 255.255.255.0 GigabitEthernet2 10.0.1.10

!  ### IP routes from c8kv-uut to clients on c8kv-peer side, routes are evenly distributed to all 12 TX

ip route 10.10.0.0 255.255.0.0 Tunnel0
ip route 10.10.0.0 255.255.0.0 Tunnel1
ip route 10.10.0.0 255.255.0.0 Tunnel2
ip route 10.10.0.0 255.255.0.0 Tunnel3
ip route 10.10.0.0 255.255.0.0 Tunnel4
ip route 10.10.0.0 255.255.0.0 Tunnel5
ip route 10.10.0.0 255.255.0.0 Tunnel6
ip route 10.10.0.0 255.255.0.0 Tunnel7
ip route 10.10.0.0 255.255.0.0 Tunnel8
ip route 10.10.0.0 255.255.0.0 Tunnel9
ip route 10.10.0.0 255.255.0.0 Tunnel10
ip route 10.10.0.0 255.255.0.0 Tunnel11


!  ### IP route from c8kv-uut Gi3 int tunnel endpoint to c8kv-peer Gi3

int tunnel endpoints (secondary IP addresses on c8kv-peer side)

ip route 20.0.2.30 255.255.255.255 10.0.2.1
ip route 20.0.2.31 255.255.255.255 10.0.2.1
ip route 20.0.2.32 255.255.255.255 10.0.2.1
ip route 20.0.2.33 255.255.255.255 10.0.2.1
ip route 20.0.2.36 255.255.255.255 10.0.2.1
ip route 20.0.2.35 255.255.255.255 10.0.2.1
ip route 20.0.2.37 255.255.255.255 10.0.2.1
ip route 20.0.2.38 255.255.255.255 10.0.2.1
ip route 20.0.2.40 255.255.255.255 10.0.2.1
ip route 20.0.2.41 255.255.255.255 10.0.2.1
```

```
ip route 20.0.2.44 255.255.255.255 10.0.2.1
ip route 20.0.2.46 255.255.255.255 10.0.2.1
```

Implementación típica de Catalyst 8000V en AWS

Modo autónomo

Consulte las configuraciones y topologías CLI de ejemplo anteriores. La configuración de CLI se puede copiar y modificar en función del esquema de direcciones de red y las direcciones IP hash generadas.

Para crear túneles correctamente, asegúrese de crear rutas IP tanto en el C8000V como en las tablas de routing en AWS VPC.

Modo SD-WAN

Este es un ejemplo de topología y configuración SD-WAN que crea TLOCs usando interfaces de loopback en C8000V ubicados en un VPC AWS.

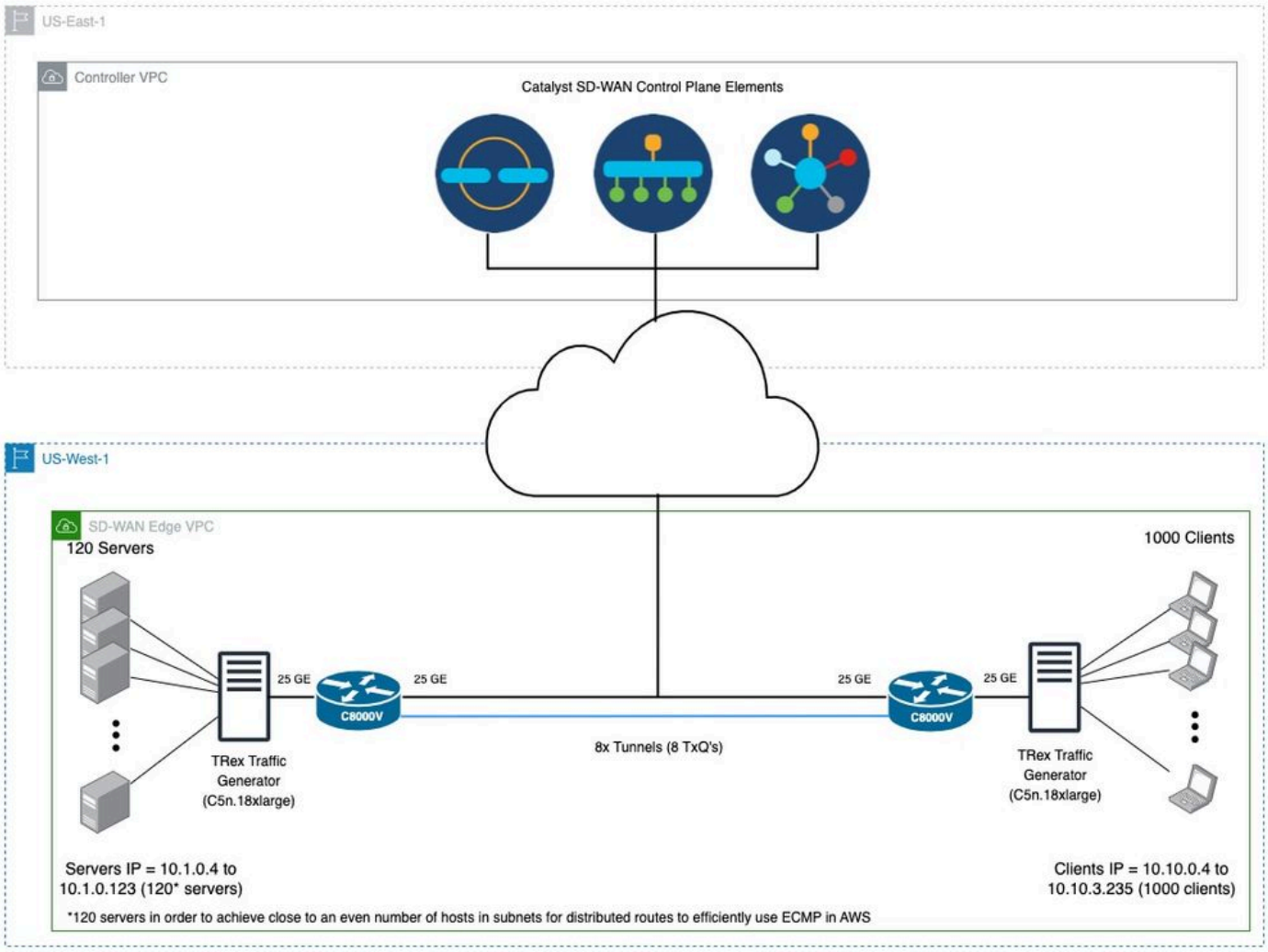
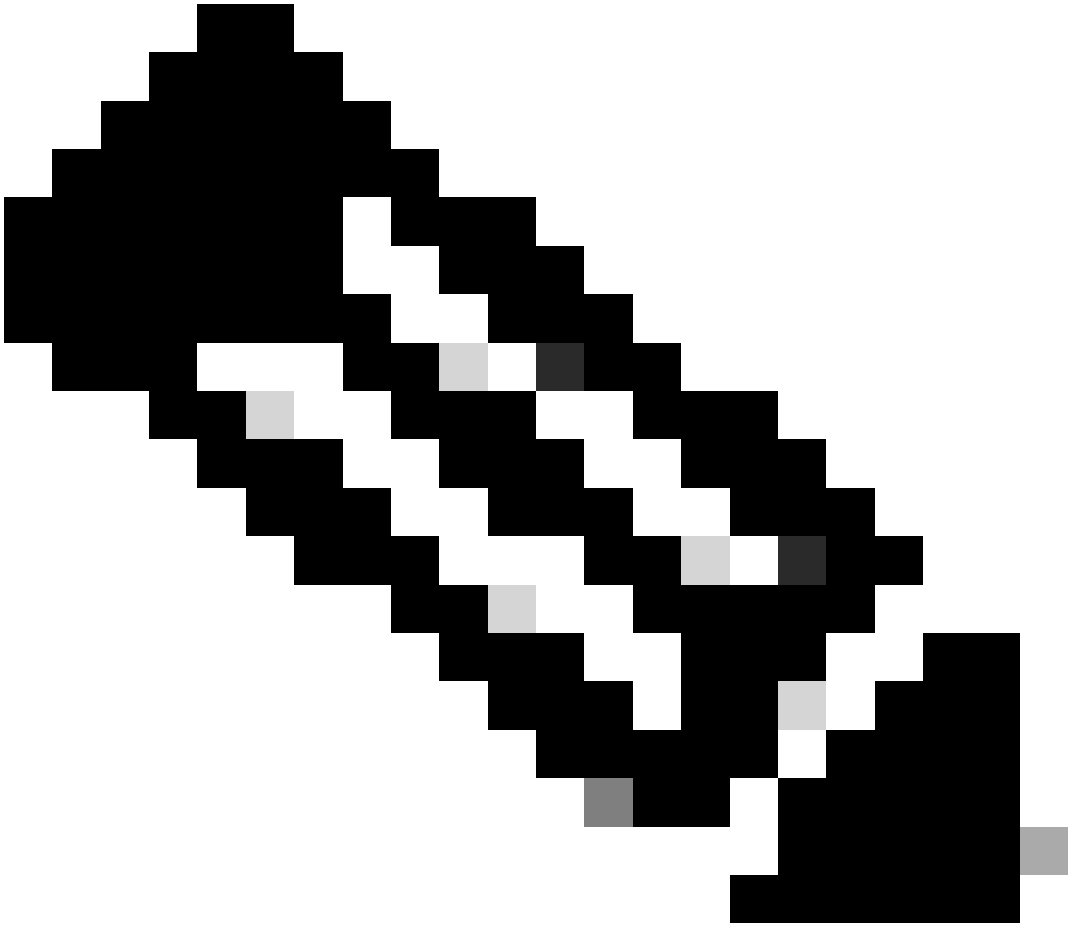


Figura 6. Topología de ejemplo de SD-WAN que utiliza TLOC con interfaces de loopback en C8000V ubicadas en un VPC AWS.



Nota: En la figura 6, la conexión de color negro representa la conexión de control (VPN0) entre los elementos del plano de control SD-WAN y los dispositivos periféricos SD-WAN. Las conexiones de color azul representan túneles entre los dos dispositivos periféricos SD-WAN que utilizan TLOC.

Puede encontrar un ejemplo de configuración de la CLI de SD-WAN para la figura 6 (aquí).

```
csr_uut#show sdwan run
system
system-ip             29.173.249.161
site-id               5172
admin-tech-on-failure
sp-organization-name  SP_ORG_NAME
organization-name     ORG_NAME
upgrade-confirm       15
vbond X.X.X.X
!
memory free low-watermark processor 68484
service timestamps debug datetime msec
```

```
service timestamps log datetime msec
no service tcp-small-servers
no service udp-small-servers
platform console virtual
platform qfp utilization monitor load 80
platform punt-keepalive disable-kernel-core
hostname csr_uut
username ec2-user privilege 15 secret 5 $1$4P16$..ag88eFsOMLIemjNcWSt0
vrf definition 11
address-family ipv4
exit-address-family
!
address-family ipv6
exit-address-family
!
!
vrf definition Mgmt-intf
address-family ipv4
exit-address-family
!
address-family ipv6
exit-address-family
!
!
no ip finger
no ip rcmd rcp-enable
no ip rcmd rsh-enable
no ip dhcp use class
ip route 0.0.0.0 0.0.0.0 X.X.X.X
ip route 0.0.0.0 0.0.0.0 X.X.X.X
ip route 0.0.0.0 0.0.0.0 X.X.X.X
ip route vrf 11 10.1.0.0 255.255.0.0 X.X.X.X
ip route vrf Mgmt-intf 0.0.0.0 0.0.0.0 X.X.X.X
no ip source-route
ip ssh pubkey-chain
username ec2-user
key-hash ssh-rsa 353158c28c7649710b3c933da02e384b ec2-user
!
!
!
no ip http server
ip http secure-server
ip nat settings central-policy
ip nat settings gatekeeper-size 1024
ipv6 unicast-routing
class-map match-any class0
match dscp 1
!
class-map match-any class1
match dscp 2
!
class-map match-any class2
match dscp 3
!
class-map match-any class3
match dscp 4
!
class-map match-any class4
match dscp 5
!
class-map match-any class5
match dscp 6
```

```
!
class-map match-any class6
match dscp 7
!
class-map match-any class7
match dscp 8
!
policy-map qos_map1
class class0
priority percent 20
!
class class1
bandwidth percent 18
random-detect
!
class class2
bandwidth percent 15
random-detect
!
class class3
bandwidth percent 12
random-detect
!
class class4
bandwidth percent 10
random-detect
!
class class5
bandwidth percent 10
random-detect
!
class class6
bandwidth percent 10
random-detect
!
class class7
bandwidth percent 5
random-detect
!
!
interface GigabitEthernet1
no shutdown
ip address dhcp
no mop enabled
no mop sysid
negotiation auto
exit
interface GigabitEthernet2
no shutdown
ip address dhcp
load-interval 30
speed 10000
no negotiation auto
service-policy output qos_map1
exit
interface GigabitEthernet3
shutdown
ip address dhcp
load-interval 30
speed 10000
no negotiation auto
exit
```

```
interface GigabitEthernet4
no shutdown
vrf forwarding 11
ip address X.X.X.X 255.255.255.0
load-interval 30
speed 10000
no negotiation auto
exit
interface Loopback1
no shutdown
ip address 192.168.1.21 255.255.255.255
exit
interface Loopback2
no shutdown
ip address 192.168.1.129 255.255.255.255
exit
interface Loopback3
no shutdown
ip address 192.168.1.20 255.255.255.255
exit
interface Loopback4
no shutdown
ip address 192.168.1.128 255.255.255.255
exit
interface Loopback5
no shutdown
ip address 192.168.1.23 255.255.255.255
exit
interface Loopback6
no shutdown
ip address 192.168.1.131 255.255.255.255
exit
interface Loopback7
no shutdown
ip address 192.168.1.22 255.255.255.255
exit
interface Loopback8
no shutdown
ip address 192.168.1.130 255.255.255.255
exit
interface Tunnel1
no shutdown
ip unnumbered GigabitEthernet1
tunnel source GigabitEthernet1
tunnel mode sdwan
exit
interface Tunnel14095001
no shutdown
ip unnumbered Loopback1
no ip redirects
ipv6 unnumbered Loopback1
no ipv6 redirects
tunnel source Loopback1
tunnel mode sdwan
exit
interface Tunnel14095002
no shutdown
ip unnumbered Loopback2
no ip redirects
ipv6 unnumbered Loopback2
no ipv6 redirects
tunnel source Loopback2
```

```
tunnel mode sdwan
exit
interface Tunnel14095003
no shutdown
ip unnumbered Loopback3
no ip redirects
ipv6 unnumbered Loopback3
no ipv6 redirects
tunnel source Loopback3
tunnel mode sdwan
exit
interface Tunnel14095004
no shutdown
ip unnumbered Loopback4
no ip redirects
ipv6 unnumbered Loopback4
no ipv6 redirects
tunnel source Loopback4
tunnel mode sdwan
exit
interface Tunnel14095005
no shutdown
ip unnumbered Loopback5
no ip redirects
ipv6 unnumbered Loopback5
no ipv6 redirects
tunnel source Loopback5
tunnel mode sdwan
exit
interface Tunnel14095006
no shutdown
ip unnumbered Loopback6
no ip redirects
ipv6 unnumbered Loopback6
no ipv6 redirects
tunnel source Loopback6
tunnel mode sdwan
exit
interface Tunnel14095007
no shutdown
ip unnumbered Loopback7
no ip redirects
ipv6 unnumbered Loopback7
no ipv6 redirects
tunnel source Loopback7
tunnel mode sdwan
exit
interface Tunnel14095008
no shutdown
ip unnumbered Loopback8
no ip redirects
ipv6 unnumbered Loopback8
no ipv6 redirects
tunnel source Loopback8
tunnel mode sdwan
exit
no logging console
aaa authentication enable default enable
aaa authentication login default local
aaa authorization console
aaa authorization exec default local none
login on-success log
```

```
license smart transport smart
license smart url https://smartreceiver.cisco.com/licservice/license
line aux 0
!
line con 0
stopbits 1
!
line vty 0 4
transport input ssh
!
line vty 5 80
transport input ssh
!
sdwan
interface GigabitEthernet1
tunnel-interface
encapsulation ipsec
color private1 restrict
allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface GigabitEthernet2
exit
interface GigabitEthernet3
exit
interface Loopback1
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private2 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
```

```
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback2
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private3 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback3
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private4 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
```

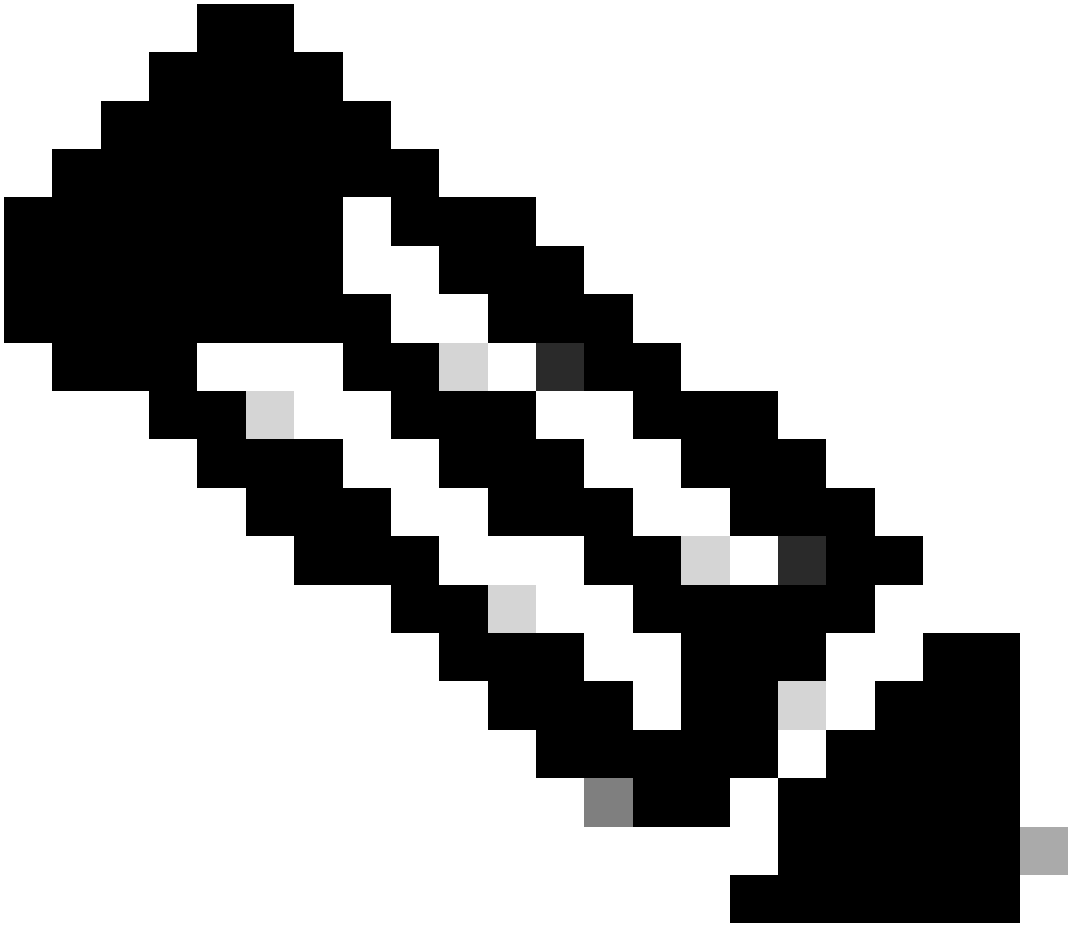
```
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback4
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private5 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback5
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private6 restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
```

```
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback6
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color red restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback7
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color blue restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
```

```
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback8
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color green restrict
no last-resort-circuit
no low-bandwidth-link
max-control-connections       0
no vbond-as-stun-server
vmanage-connection-preference 0
port-hop
carrier                       default
nat-refresh-interval          5
hello-interval                1000
hello-tolerance               12
bind                          GigabitEthernet2
no allow-service all
no allow-service bgp
allow-service dhcp
allow-service dns
allow-service icmp
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
appqoe
no tcpopt enable
no dreopt enable
no httpopt enable
!
omp
no shutdown
send-path-limit  16
ecmp-limit       16
graceful-restart
no as-dot-notation
timers
graceful-restart-timer 43200
exit
address-family ipv4
advertise connected
advertise static
!
address-family ipv6
advertise connected
advertise static
!
!
!
security
```

```
ipsec
replay-window  8192
integrity-type ip-udp-esp esp
!
!
sslproxy
no enable
rsa-key-modulus      2048
certificate-lifetime 730
eckey-type           P256
ca-tp-label          PROXY-SIGNING-CA
settings expired-certificate  drop
settings untrusted-certificate drop
settings unknown-status       drop
settings certificate-revocation-check none
settings unsupported-protocol-versions drop
settings unsupported-cipher-suites drop
settings failure-mode         close
settings minimum-tls-ver      TLSv1
dual-side optimization enable
!
policy
app-visibility
flow-visibility
!
```

Resolución de problemas de rendimiento en AWS



Nota: La realización de pruebas de rendimiento en entornos de nube pública introduce nuevas variables que podrían afectar al rendimiento. A continuación se indican algunos aspectos que se deben tener en cuenta al realizar este tipo de pruebas:

- Uso de recursos subyacente por los pares en el momento de ejecutar las pruebas
- No utilizar hosts dedicados (el uso de hosts dedicados aumenta el coste de la nube 16 veces)
- La nube se ejecuta en diferentes regiones, el rendimiento puede variar
- En algunos casos, los números son similares independientemente del perfil de la función; esto se debe posiblemente a la limitación de AWS en la interfaz por tamaño de instancia
- AWS regula la velocidad de paquetes por segundo en las instancias EC2, lo que también puede provocar la caída de paquetes
- AWS no revela la velocidad de aceleración, pero las caídas debido a la limitación de pps se pueden observar a través del contador 'pps_alloween_exceeded'

Comandos útiles de resolución de problemas CLI

Al realizar pruebas de rendimiento de rendimiento, estos comandos de solución de problemas se pueden utilizar para identificar cuellos de botella o motivos de degradación del rendimiento.

"show platform hardware qfp active statistics drop" - nos permite entender si hay caídas en el c8kv. Tenemos que asegurarnos de que no haya caídas de cola significativas ni ningún contador relevante que aumente.

"show platform hardware qfp active statistics drop clear" - Este comando borra los contadores.

"show platform hardware qfp active datapath infrastructure sw-cio" - Este comando nos da información detallada sobre el porcentaje de procesador de paquetes (PSP), Traffic Manager (TM) que se utiliza durante las ejecuciones de rendimiento. Esto nos permite determinar si hay suficiente capacidad de procesamiento o no desde el c8kv.

"show platform hardware qfp active datapath util summary" - Este comando nos da la información completa de la entrada/salida que el c8kv está transmitiendo/recibiendo de todos los puertos.

Asegúrese de comprobar la velocidad de entrada/salida y ver si hay alguna caída. Además, asegúrese de verificar el porcentaje de carga de procesamiento. Si alcanza el 100%; significa que el c8kv ha alcanzado su capacidad.

"show plat hardware qfp active infrastructure bqs interface GigabitEthernetX" - Este comando nos permite verificar las estadísticas de nivel de interfaz en términos de número de cola, ancho de banda y caídas de cola.

"show controller" - Este comando nos da mucha información granular sobre los paquetes rx/tx buenos, paquetes perdidos.

Este comando se puede utilizar en un escenario donde no vemos ninguna caída de cola pero el generador de tráfico todavía nos muestra caída.

Esto puede suceder en un escenario en el que la utilización de los datos ya está alcanzando el 100% e igualmente el PP al 100%.

Si los contadores rx_missing_errors continúan aumentando, implica que el CSR está presionando hacia atrás la infraestructura de la nube ya que no puede procesar más tráfico.

"show platform hardware qfp active datapath infrastructure sw-hqf" - se puede utilizar para verificar cualquier congestión que ocurra debido a la contrapresión de AWS.

"show plat hardware qfp active datapath infrastructure sw-nic" - Determina cómo se equilibra la carga del tráfico a través de múltiples colas. Post 17.7, tenemos 8 Multi-TXQs.

Además, puede determinar si hay alguna cola en particular que esté tomando todo el tráfico o si se está balanceando la carga correctamente.

"show controllers | in errors|exceeded|Giga" - Muestra las caídas de paquetes debido a la

aceleración de pps realizada desde el lado AWS, que se puede observar a través del contador pps_allow_exceeded.

Ejemplo de salida CLI

Salida de ejemplo donde el contador de caídas de cola sigue aumentando- Ejecute el comando varias veces para ver si los contadores aumentan, lo que nos permite confirmar que efectivamente se trata de caídas de cola.

<#root>

```
csr_uut#show platform hardware qfp active statistics drop
Last clearing of QFP drops statistics : never
-----------------------------------------------------------------------
Global Drop Stats Packets Octets
-----------------------------------------------------------------------
Disabled 30 3693
IpFragErr 192 290976
Ipv4NoRoute 43 3626
Ipv6NoRoute 4 224
SdwanImplicitAclDrop 31 3899
```

TailDrop 19099700 22213834441

```
UnconfiguredIpv6Fia 3816 419760
```

Ejemplo de resultado mostrado aquí: ejecute el comando cada 30 segundos para obtener los datos en tiempo real

<#root>

```
csr_uut#show platform hardware qfp active datapath infrastructure sw-cio
Credits Usage:

ID Port Wght Global WRKR0 WRKR1 WRKR2 WRKR3 WRKR4 WRKR5 WRKR6 WRKR7 WRKR8 WRKR9 WRKR10 WRKR11 WRKR12 WR
1 rcl0 16: 455 0 4 1 2 3 2 2 4 4 4 4 0 4 23 512
1 rcl0 32: 496 0 0 0 0 0 0 0 0 0 0 0 0 0 16 512
2 ipc 1: 468 4 2 4 3 0 1 1 4 0 2 0 4 0 18 511
3 vxe_punti 4: 481 0 0 0 0 0 0 0 0 0 0 0 0 0 31 512
4 Gi1 4: 446 0 0 1 1 0 2 3 0 3 2 0 1 1 52 512
5 Gi2 4: 440 4 4 4 3 2 1 1 3 2 4 4 3 2 59 504
6 Gi3 4: 428 1 1 1 0 4 4 1 0 4 4 0 0 2 43 494
7 Gi4 4: 427 1 1 0 1 4 2 0 4 3 4 1 1 7 56 512

Core Utilization over preceding 12819.5863 seconds
-------------------------------------------------
ID: 0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

% PP

```
: 6.11 6.23 6.09 6.09 6.04 6.05 6.06 6.07 6.05 6.03 6.04 6.06 0.00 0.00
% RX: 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 2.23
```

% TM:

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 4.79 0.00
% IDLE: 93.89 93.77 93.91 93.91 93.96 93.95 93.94 93.93 93.95 93.97 93.96 93.94 95.21 97.77
```

Salida de muestra que se muestra aquí - Asegúrese de verificar la velocidad de entrada/salida y ver si hay alguna caída. Además, asegúrese de verificar el porcentaje de carga de procesamiento. Si alcanza el 100%; significa que el nodo ha alcanzado su capacidad.

<#root>

```
csr_uut#show platform hardware qfp active datapath util summary
CPP 0: 5 secs 1 min 5 min 60 min

Input: Total (pps)

900215 980887 903176 75623
(bps) 10276623992 11197595912 10310265440 863067008

Output: Total (pps)

900216 937459 865930 72522
(bps) 10276642720 10712432752 9894215928 828417104

Processing: Load (pct)

56 58 54 4
```

Salida de ejemplo que se muestra aquí para las estadísticas de nivel de interfaz:

<#root>

```
csr_uut#sh plat hardware qfp active infrastructure bqs interface GigabitEthernet2
Interface: GigabitEthernet2, QFP interface: 7
Queue: QID: 111 (0x6f)
bandwidth (cfg) : 0 , bandwidth (hw) : 1050000000
shape (cfg) : 0 , shape (hw) : 0
prio level (cfg) : 0 , prio level (hw) : n/a
limit (pkts ) : 1043
Statistics:
depth (pkts ) : 0

tail drops (bytes): 0 , (packets) : 0


total enqs (bytes): 459322360227 , (packets) : 374613901
licensed throughput oversubscription drops:
(bytes): 0 , (packets) : 0
Schedule: (SID:0x8a)
Schedule FCID : n/a
bandwidth (cfg) : 10500000000 , bandwidth (hw) : 10500000000
shape (cfg) : 10500000000 , shape (hw) : 10500000000
Schedule: (SID:0x87)
Schedule FCID : n/a
bandwidth (cfg) : 200000000000 , bandwidth (hw) : 200000000000
shape (cfg) : 200000000000 , shape (hw) : 200000000000
Schedule: (SID:0x86)
Schedule FCID : n/a
```

```
bandwidth (cfg) : 500000000000 , bandwidth (hw) : 500000000000
shape (cfg) : 500000000000 , shape (hw) : 500000000000


csr_uut#sh plat hardware qfp active infrastructure bqs interface GigabitEthernet3 | inc tail
tail drops (bytes): 55815791988 , (packets) : 43177643
```

Ejemplo de salida para los paquetes correctos RX/TX, estadísticas de paquetes perdidos

<#root>

```
c8kv-aws-1#show controller
GigabitEthernet1 - Gi1 is mapped to UIO on VXE
rx_good_packets 346
tx_good_packets 243
rx_good_bytes 26440
tx_good_bytes 31813
rx_missed_errors 0
rx_errors 0
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0packets 0
rx_q0bytes 0
rx_q0errors 0
tx_q0packets 0
tx_q0bytes 0
GigabitEthernet2 - Gi2 is mapped to UIO on VXE
rx_good_packets 96019317
tx_good_packets 85808651
rx_good_bytes 12483293931
tx_good_bytes 11174853219
```

rx_missed_errors 522036

```
rx_errors 0
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0packets 0
rx_q0bytes 0
rx_q0errors 0
tx_q0packets 0
tx_q0bytes 0
GigabitEthernet3 - Gi3 is mapped to UIO on VXE
rx_good_packets 171596935
tx_good_packets 191911304
rx_good_bytes 11668588022
tx_good_bytes 13049984257
```

rx_missed_errors 21356065

```
rx_errors 0
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0packets 0
rx_q0bytes 0
rx_q0errors 0
```

```
tx_q0packets 0
tx_q0bytes 0
GigabitEthernet4 - Gi4 is mapped to UIO on VXE
rx_good_packets 95922932
tx_good_packets 85831238
rx_good_bytes 12470124252
tx_good_bytes 11158486786
```

rx_missed_errors 520328

```
rx_errors 46
tx_errors 0
rx_mbuf_allocation_errors 0
rx_q0packets 0
rx_q0bytes 0
rx_q0errors 0
tx_q0packets 0
tx_q0bytes 0
```

Salida de muestra para comprobar si hay alguna congestión debido a la contrapresión de AWS:

<#root>

```
csr_uut#show platform hardware qfp active datapath infrastructure sw-hqf
Name : Pri1 Pri2 None / Inflight pkts
GigabitEthernet4 : XON XON XOFF / 43732

HQF[0] IPC: send 514809 fc 0 congested_cnt 0
HQF[0] recycle: send hi 0 send lo 228030112
fc hi 0 fc lo 0
cong hi 0 cong lo 0
HQF[0] pkt: send hi 433634 send lo 2996661158
fc/full hi 0 fc/full lo 34567275
```

cong hi 0 cong lo 4572971630********Congestion counters keep incrementing**

```
HQF[0] aggr send stats 3225639713 aggr send lo state 3225206079
aggr send hi stats 433634
max_tx_burst_sz_hi 0 max_tx_burst_sz_lo 0
HQF[0] gather: failed_to_alloc_b4q 0
HQF[0] ticks 662109543, max ticks accumulated 348
HQF[0] mpsc stats: count: 0
enq 3225683472 enq_spin 0 enq_post 0 enq_flush 0
sig_cnt:0 enq_cancel 0
deq 3225683472 deq_wait 0 deq_fail 0 deq_cancel 0
deq_wait_timeout
```

Salida de ejemplo para cómo se equilibra la carga del tráfico a través de varias colas:

```
um-csr-uut#sh plat hardware qfp active datapath infrastructure sw-nic
pmd b1c5a400 device Gi1
RX: pkts 50258 bytes 4477620 return 0 badlen 0
```

```
pkts/burst 1 cycl/pkt 579 ext_cycl/pkt 996
Total ring read 786244055, empty 786197491
TX: pkts 57860 bytes 6546349
pri-0: pkts 7139 bytes 709042
pkts/send 1
pri-1: pkts 3868 bytes 451352
pkts/send 1
pri-2: pkts 1875 bytes 219403
pkts/send 1
pri-3: pkts 2417 bytes 242527
pkts/send 1
pri-4: pkts 8301 bytes 984022
pkts/send 1
pri-5: pkts 10268 bytes 1114859
pkts/send 1
pri-6: pkts 1740 bytes 175353
pkts/send 1
pri-7: pkts 22252 bytes 2649791
pkts/send 1
Total: pkts/send 1 cycl/pkt 1091
send 56756 sendnow 0
forced 56756 poll 0 thd_poll 0
blocked 0 retries 0 mbuf alloc err 0
TX Queue 0: full 0 current index 0 hiwater 0
TX Queue 1: full 0 current index 0 hiwater 0
TX Queue 2: full 0 current index 0 hiwater 0
TX Queue 3: full 0 current index 0 hiwater 0
TX Queue 4: full 0 current index 0 hiwater 0
TX Queue 5: full 0 current index 0 hiwater 0
TX Queue 6: full 0 current index 0 hiwater 0
TX Queue 7: full 0 current index 0 hiwater 0
pmd b1990b00 device Gi2
RX: pkts 1254741010 bytes 511773562848 return 0 badlen 0
pkts/burst 16 cycl/pkt 792 ext_cycl/pkt 1342
Total ring read 1012256968, empty 937570790
TX: pkts 1385120320 bytes 564465308380
pri-0: pkts 168172786 bytes 68650796972
pkts/send 1
pri-1: pkts 177653235 bytes 72542203822
pkts/send 1
pri-2: pkts 225414300 bytes 91947701824
pkts/send 1
pri-3: pkts 136817435 bytes 55908224442
pkts/send 1
pri-4: pkts 256461818 bytes 104687120554
pkts/send 1
pri-5: pkts 176043289 bytes 71879529606
pkts/send 1
pri-6: pkts 83920827 bytes 34264110122
pkts/send 1
pri-7: pkts 160636635 bytes 64585622696
pkts/send 1
Total: pkts/send 1 cycl/pkt 442
send 1033104466 sendnow 41250092
forced 1776500651 poll 244223290 thd_poll 0
blocked 1060879040 retries 3499069 mbuf alloc err 0
TX Queue 0: full 0 current index 0 hiwater 31
TX Queue 1: full 718680 current index 0 hiwater 255
TX Queue 2: full 0 current index 0 hiwater 31
TX Queue 3: full 0 current index 0 hiwater 31
TX Queue 4: full 15232240 current index 0 hiwater 255
TX Queue 5: full 0 current index 0 hiwater 31
```

```
TX Queue 6: full 0 current index 0 hiwater 31
TX Queue 7: full 230668 current index 0 hiwater 224
pmd b1712d00 device Gi3
RX: pkts 1410702537 bytes 498597093510 return 0 badlen 0
pkts/burst 18 cycl/pkt 269 ext_cycl/pkt 321
Total ring read 1011915032, empty 934750846
TX: pkts 754803798 bytes 266331910366
pri-0: pkts 46992577 bytes 16616415156
pkts/send 1
pri-1: pkts 49194201 bytes 17379760716
pkts/send 1
pri-2: pkts 46991555 bytes 16616509252
pkts/send 1
pri-3: pkts 49195026 bytes 17381741474
pkts/send 1
pri-4: pkts 48875656 bytes 17283423414
pkts/send 1
pri-5: pkts 417370776 bytes 147056906106
pkts/send 6
pri-6: pkts 46992860 bytes 16617923068
pkts/send 1
pri-7: pkts 49191147 bytes 17379231180
pkts/send 1
Total: pkts/send 2 cycl/pkt 0
send 339705775 sendnow 366141927
forced 3138709511 poll 2888466204 thd_poll 0
blocked 1758644571 retries 27927046 mbuf alloc err 0
TX Queue 0: full 0 current index 0 hiwater 0
TX Queue 1: full 0 current index 0 hiwater 0
TX Queue 2: full 0 current index 0 hiwater 0
TX Queue 3: full 0 current index 0 hiwater 0
TX Queue 4: full 0 current index 1 hiwater 0
TX Queue 5: full 27077270 current index 0 hiwater 224
TX Queue 6: full 0 current index 0 hiwater 0
TX Queue 7: full 0 current index 0 hiwater 0
```

Salida de ejemplo que muestra las caídas de paquetes debido a la aceleración de pps realizada desde el lado AWS, que se puede observar a través del contador pps_allow_exceeded:

```
C8k-AWS-2#show controllers | in errors|exceeded|Giga

GigabitEthernet1 - Gi1 is mapped to UIO on VXE
  rx_missed_errors 1750262
  rx_errors 0
  tx_errors 0
  rx_mbuf_allocation_errors 0
  rx_q0_errors 0
  rx_q1_errors 0
  rx_q2_errors 0
  rx_q3_errors 0
  bw_in_allowance_exceeded 0
  bw_out_allowance_exceeded 0
  pps_allowance_exceeded 11750
  conntrack_allowance_exceeded 0
  linklocal_allowance_exceeded 0
```

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).