Miglioramento del throughput su Catalyst 8000V con Multi-TxQs in AWS

Sommario

Introduzione

In questo documento viene descritto come abilitare e utilizzare Multi-TXQ su Catalyst 8000V implementato in ambienti AWS per migliorare le prestazioni di throughput.

Premesse

La presenza di più code semplifica e accelera il processo di mapping dei pacchetti in entrata e in uscita su una particolare vCPU. L'utilizzo di Multi-TXQ su Catalyst 8000V consente un utilizzo efficiente dei core nei core del dataplane disponibili allocati, con conseguente aumento delle prestazioni di throughput. In questo articolo viene fornita una breve panoramica sul funzionamento dei Multi-TXQ, sulla relativa configurazione, vengono mostrati esempi di configurazioni CLI per le implementazioni Autonomous e SD-WAN Catalyst 8000V e vengono esaminati i comandi di risoluzione dei problemi per individuare i colli di bottiglia delle prestazioni.

Comportamento di Catalyst 8000V quando non si utilizzano Multi-TxQs

Fino alla versione software 17.18, i pacchetti che entrano in Catalyst 8000V vengono distribuiti a tutte le vCPU (Packet Processing Core), indipendentemente dai flussi. Una volta che PP ha completato l'elaborazione del pacchetto, l'ordine del flusso viene ripristinato per essere inviato su un'interfaccia.

Prima di inserire il pacchetto in una coda di trasmissione (TxQ), Catalyst 8000V crea un TxQ per interfaccia. Pertanto, se è disponibile una sola interfaccia di uscita, più flussi confluiscono in un unico TxQ.

Catalyst 8000V non è in grado di trarre vantaggio da questo processo Multi-TxQ se è disponibile una sola interfaccia. Ciò determina colli di bottiglia delle prestazioni di throughput e una distribuzione del carico non uniforme tra i core del dataplane disponibili. Se viene utilizzata una sola interfaccia di uscita per trasmettere i dati dall'istanza C8000V, è disponibile un solo TxQ per trasmettere il traffico di rete e probabilmente causare la perdita dei pacchetti a causa di un riempimento più rapido della coda singola.

Per riferimento, il modello di architettura TxQ per Catalyst 8000V implementato in AWS è disponibile nella Figura 1.

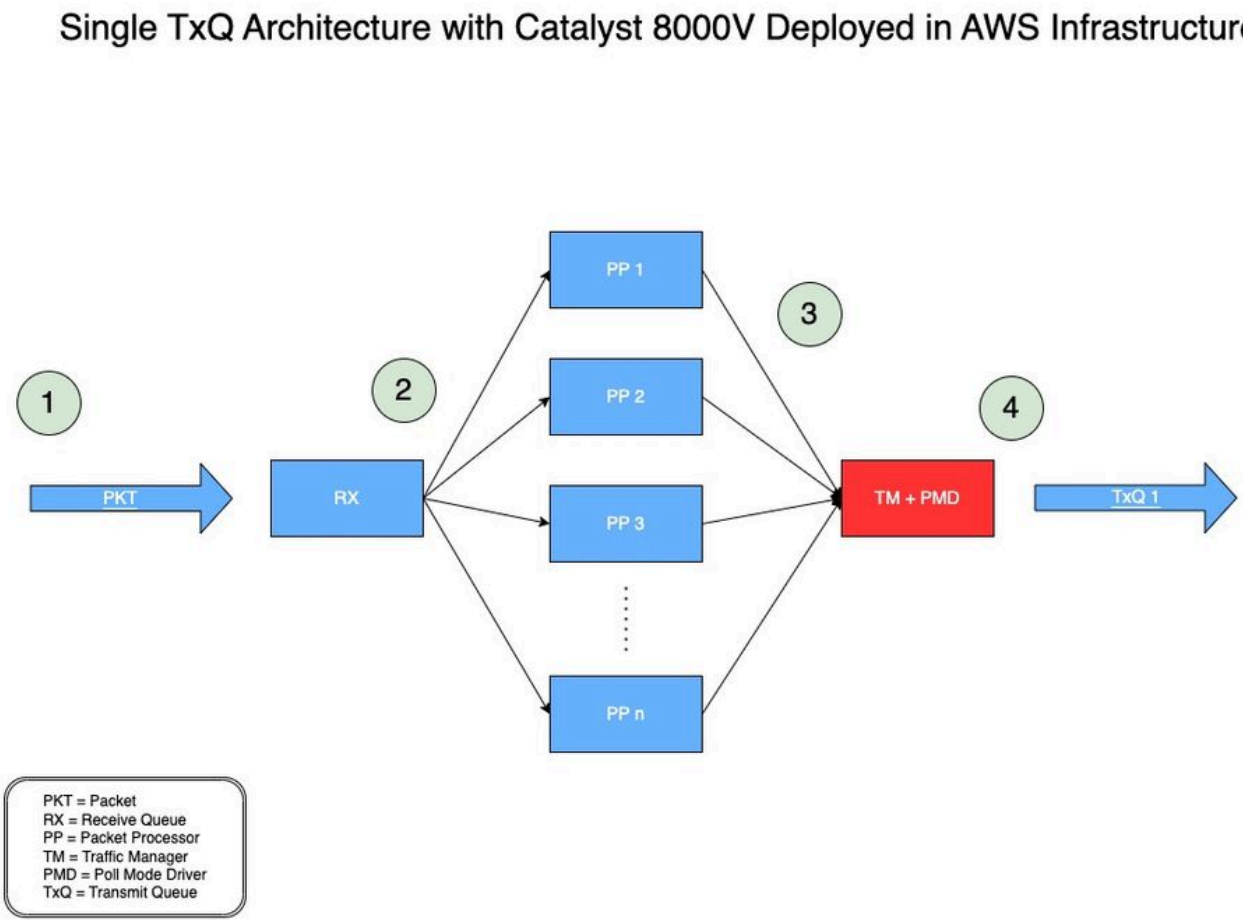


Figura 1: Singolo modello di architettura TxQ per Catalyst 8000V implementato in AWS.

1. Un pacchetto di rete (PKT) sta attraversando un VPC e viene ricevuto sull'interfaccia in entrata di un C800V.
2. Il PKT viene inserito in una coda di ricezione (RX) e quindi inoltrato a un motore di elaborazione pacchetti (PP) deciso da un algoritmo.
3. Dopo aver elaborato il pacchetto, il processore di pacchetti (PPP) lo invia al Traffic Manager (TM).
4. Al termine dell'elaborazione TM, un core è responsabile dell'inserimento del pacchetto nel TxQ disponibile, che viene quindi inoltrato all'interfaccia in uscita del Catalyst 8000V.

Cosa sono i Multi-TXQ nell'infrastruttura AWS

AWS ENA fornisce code di trasmissione multiple (Multi-TxQ) per ridurre il sovraccarico interno e aumentare la scalabilità. La presenza di più code semplifica e accelera il processo di mapping dei pacchetti in entrata e in uscita su una particolare vCPU. Il modello di riferimento di rete AWS e DPDK è basato sul flusso, in cui ogni vCPU elabora un flusso e trasmette i pacchetti da tale flusso a una coda di trasmissione assegnata (TxQ). La coppia di code RX/TX per ogni vCPU è valida in base al modello basato sul flusso.

Poiché Catalyst 8000V NON è basato sul flusso, l'istruzione "Coppia di code RX/TX per ciascuna vCPU" non è applicabile a Catalyst 8000V.

In questo caso, le code RX/TX non sono per vCPU ma per interfaccia. Le code RX/TX fungono da interfacce tra l'applicazione (Catalyst 8000V) e l'infrastruttura/hardware AWS per l'invio di traffico di dati/rete. AWS controlla la velocità e il numero di code RX/TX disponibili per interfaccia per singola istanza.

Catalyst 8000V deve avere più interfacce per creare più TxQ. Per mantenere l'ordine del flusso con più flussi in uscita da un'interfaccia (quando Catalyst 8000V abilita più TxQ seguendo questo processo), Catalyst 8000V esegue l'hash dei flussi in base alle 5 tuple per scegliere il TxQ appropriato. Un utente può creare più interfacce su Catalyst 8000V utilizzando la stessa NIC fisica collegata all'istanza utilizzando interfacce di loopback o indirizzi IP secondari.

Nella Figura 2, è possibile vedere come viene elaborato un pacchetto usando l'architettura Multi-TxQ con Catalyst 8000V in AWS.

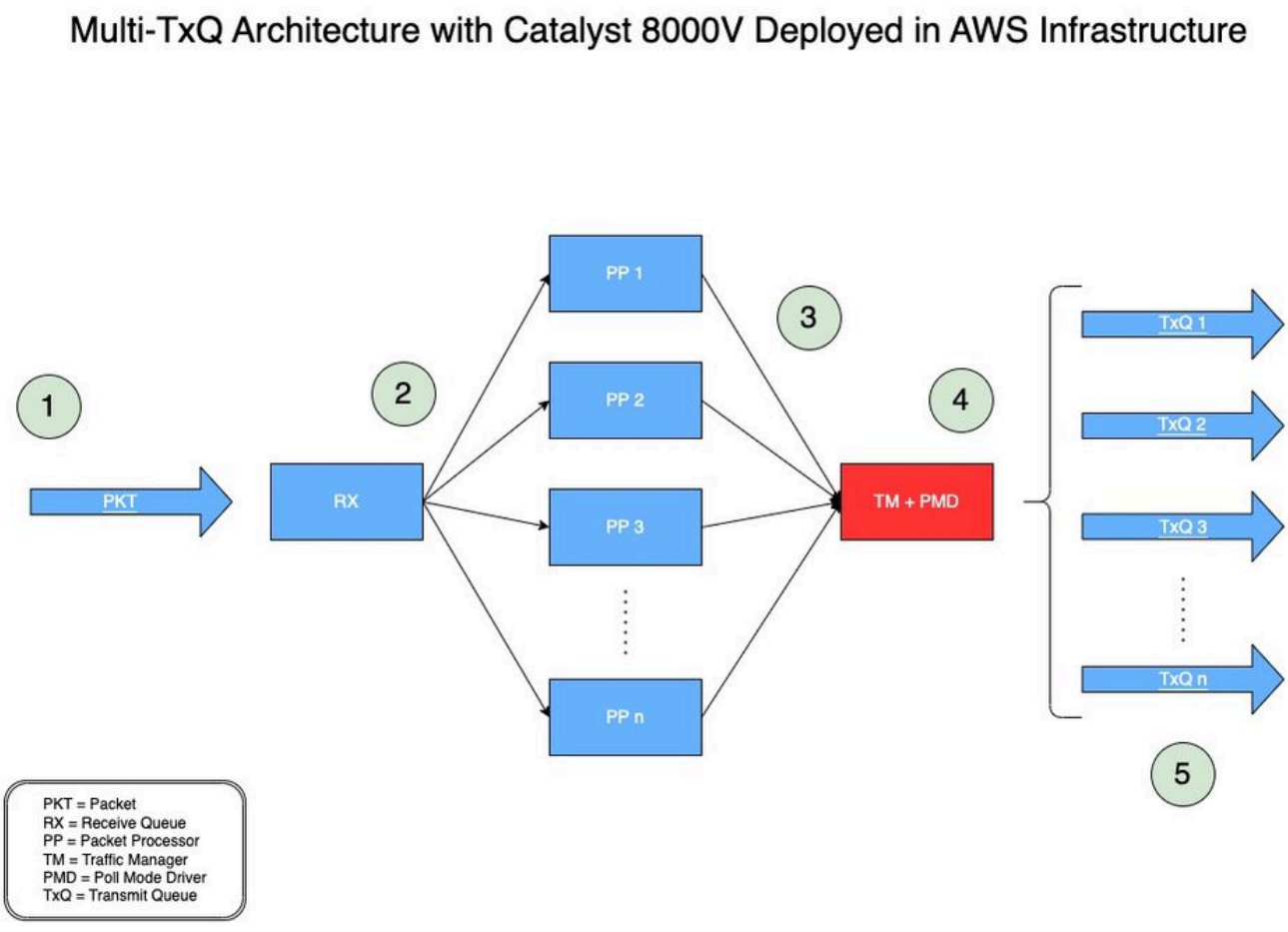


Figura 2: modello di architettura Multi-TxQ per Catalyst 8000V implementato in AWS.

1. Un pacchetto di rete (PKT) sta attraversando un VPC e viene ricevuto sull'interfaccia in entrata di un C800V.

2. Il PKT viene inserito in una coda di ricezione (RX) e quindi inoltrato a un motore di elaborazione pacchetti (PP) deciso da un algoritmo.
3. Dopo aver elaborato il pacchetto, il processore di pacchetti (PPP) lo invia al Traffic Manager (TM).
4. Al termine dell'elaborazione TM, prima di inserire il pacchetto in una coda di trasmissione (TxQ), il TM controlla l'intestazione del pacchetto e lo incapsula nell'hash (come spiegato nella sezione successiva). Per configurare il numero di TXQ supportati dall'istanza viene utilizzato un altro componente, il driver in modalità polling (PMD). Un core è dedicato alla funzione TM + PMD che esegue l'hashing e l'invio del pacchetto al TxQ assegnato.
5. Il TxQ viene scelto in base alle cinque tuple con hash e modulo con il numero di TxQ supportato dall'istanza. I pacchetti vengono inviati al TxQ selezionato e inoltrati all'interfaccia in uscita di Catalyst 8000V.

Hash del traffico in Multi-TxQs

Al termine dell'elaborazione TM, come mostrato nel Passo 4 della Figura 2, prima di inserire il pacchetto in un TxQ, il TM controlla l'intestazione del pacchetto ed estrae le 5 tuple (indirizzo di destinazione, indirizzo di origine, protocollo, porta di destinazione e porta di origine) ed esegue l'hashing del pacchetto su un TxQ.

Il TxQ viene scelto in base alle cinque tuple con hash e modulo con il numero di TxQ supportato dall'istanza.

Versioni del software Catalyst 8000V che supportano Multi-TxQs

Le istanze AWS EC2 dello stesso tipo di famiglia di istanze supportano un numero diverso di TXQ, a seconda delle dimensioni dell'istanza. Il C8000V ha iniziato a supportare più TxQ a partire da IOS® XE 17.7.

A partire da IOS® XE 17.7, il modello C8000V supporta più TxQ sul modello C5n.9xlarge, che può contenere fino a 8 TXQ.

A partire da IOS® XE 17.9, il modello C8000V supporta le dimensioni dell'istanza C5n.18xlarge, che possono avere fino a 12 TXQ (il 50% in più rispetto a C5n.9xlarge).

Sebbene Multi-TxQ sia supportato da IOS® XE 17.7, si consiglia VIVAMENTE di utilizzare IOS® XE 17.9 sia per il ciclo di vita del software che per le prestazioni di throughput più elevate con il supporto di 12 TxQ.

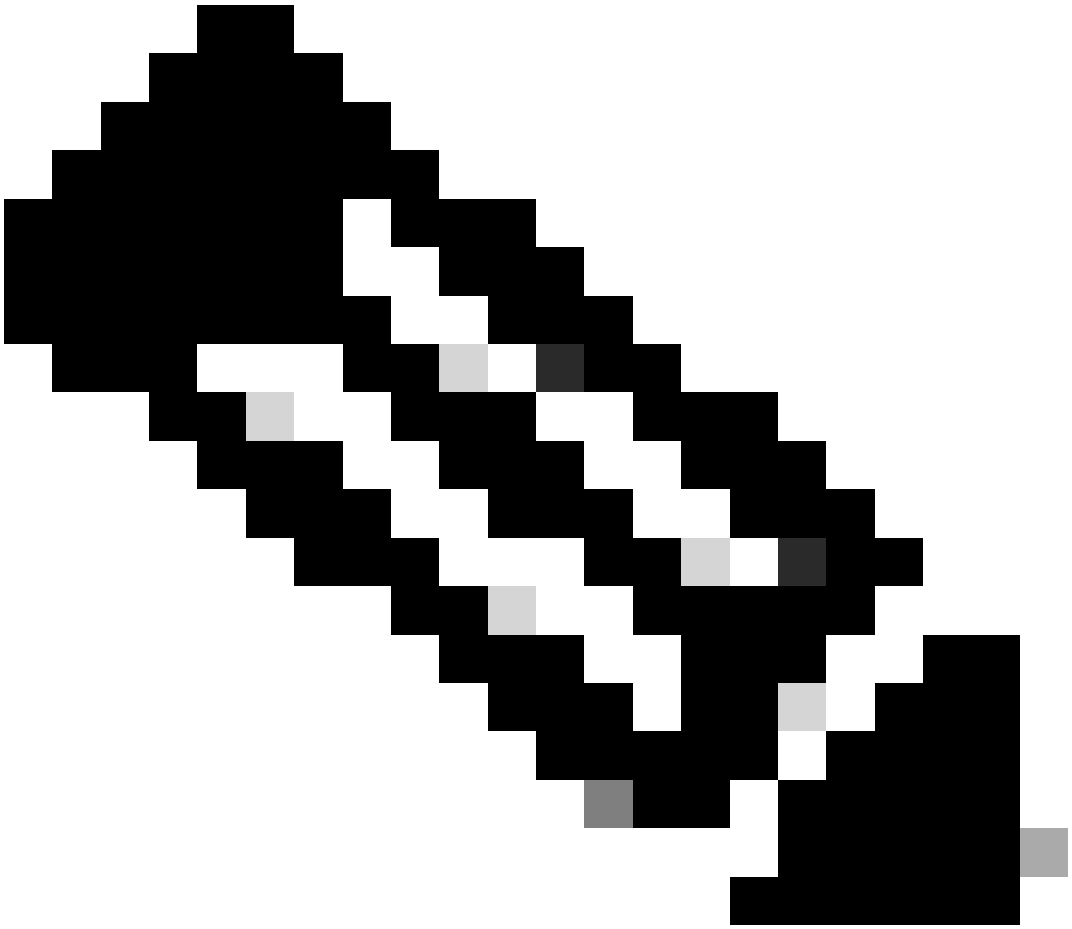
Come progettare lo schema di indirizzamento IP per calcolare l'hashing

Per eseguire l'hashing uniforme del traffico tra tutti i TxQ disponibili, è necessario utilizzare indirizzi IP speciali quando Catalyst 8000V sta terminando i tunnel IPsec/GRE.

Per generare questi indirizzi IP speciali, sono disponibili script pubblici da utilizzare per configurare le interfacce Catalyst 8000V responsabili della terminazione dei tunnel. In questa sezione vengono fornite istruzioni su come scaricare e utilizzare gli script per progettare gli indirizzi IP richiesti per l'hashing Multi-TxQ uniforme.

Se Catalyst 8000V gestisce traffico a testo non crittografato come TCP/UDP, non è necessario uno schema di indirizzamento IP speciale.

Le istruzioni originali sono disponibili qui: https://github.com/CiscoDevNet/python-c8000v-aws-multitx-queues/



> Nota: Per Catalyst 8000V con 17.18 o versioni successive, i pacchetti vengono distribuiti in modo diverso. Pertanto, è necessario utilizzare un algoritmo di hashing diverso.

Prerequisiti

- Deve disporre di un computer Linux/MacOS o Windows in grado di eseguire gli script

Python.

- Verificare che la versione Python sia 3.8.9 o successiva; controllare la versione Python con 'python3 —version'
- Installare PIP se non è già installato. In caso contrario, eseguire:
  - curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  - python3 get-pip.py

È possibile controllare la versione Python utilizzata dal computer con il comando 'python3 —version'.

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

Python 3.9.6
```

Una volta che la versione Python è stata verificata ed è in esecuzione, una versione uguale o superiore alla 3.8.9, installare la versione più recente di 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
```

Crea ambiente virtuale

Dopo aver installato i prerequisiti, creare l'ambiente virtuale e scaricare lo script di hashing dell'indirizzo IP utilizzato per generare lo schema di indirizzi IP univoco per Multi-TxQ.

Riepilogo comandi:

1. python3 -m venv c8kv-hash
2. hash cd c8kv
3. raccoglitore origine/attiva
4. git clone https://github.com/CiscoDevNet/python-c8000v-aws-multitx-queues/
5. cd c8kv-aws-pmd-hash
6. python3 -m pip installazione —upgrade pip
7. pip install -r requisiti.txt

Gli ambienti virtuali in Python vengono utilizzati per creare spazi di lavoro isolati che non influiscono su altri progetti o dipendenze. Creare l'ambiente virtuale 'c8kv-hash' utilizzando questo comando:

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

Spostarsi all'interno dell'ambiente virtuale nella cartella 'c8kv-hash' (creata in precedenza).

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

Attivare l'ambiente virtuale.

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

Clonare il repository che dispone dello script Python di hashing 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 volta duplicato il repository, passare alla cartella 'c8kv-aws-pmd-hash'. Poiché si trova nell'ambiente virtuale creato, installare la versione più recente di PIP.

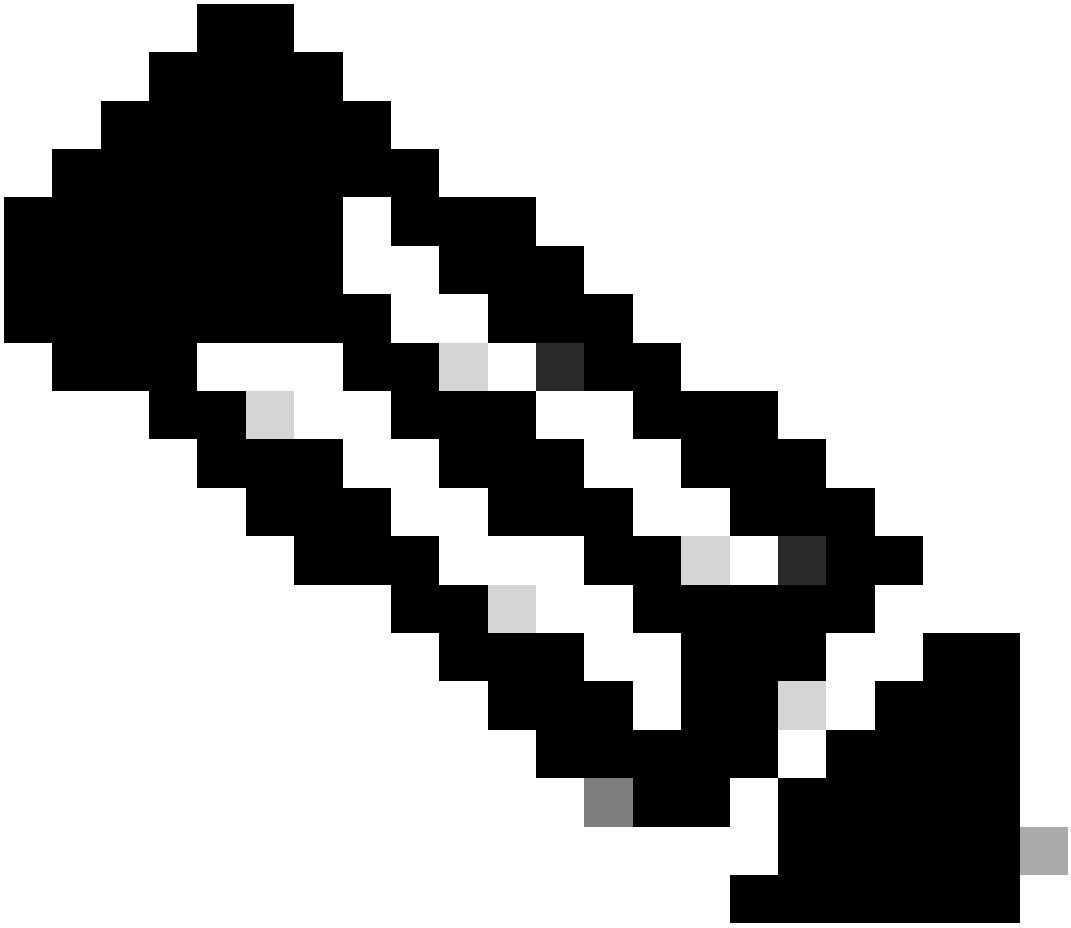
```
(c8kv-hash) user@computer c8kv-hash % cd c8kv-aws-pmd-hash
(c8kv-hash) user@computer c8kv-aws-pmd-hash % python3 -m pip install --upgrade pip

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 volta aggiornato PIP, installare le dipendenze trovate nel file requirements.txt nella cartella.

```
(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
```

L'ambiente virtuale è aggiornato e può essere utilizzato per generare lo schema di indirizzi IP per Multi-TxQ.

Calcola schema indirizzi IP tramite script Python Hash Index per le versioni 17.7 e 17.8 (deprecato)



Nota: GLI SCRIPT HASH 7.7 E 17.8 SARANNO PRESTO DEPRECATI. SI CONSIGLIA DI UTILIZZARE LO SCRIPT HASH 17.9

Riepilogo comandi:

- 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 indice hash basato sulla release 17.7 e 17.8 con modulo 8 che corrisponde al TXQ PMD supportato (NON modificare)

'—dest_network' definisce la subnet dell'indirizzo di rete di destinazione (modifica in base allo schema dell'indirizzo IP di rete)

'—src_network' definisce la subnet dell'indirizzo di rete di origine (da modificare in base allo schema dell'indirizzo IP di rete)

'—unique_hash 1' genera un set (8 coppie per 8 TXQ) di indirizzi IP con hash univoco. È possibile

modificare questa impostazione.

<#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

Calcola schema indirizzi IP tramite script Python Hash Index per le versioni 17.9 e successive

Riepilogo comandi:

- 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

In IOS® XE versione 17.9 e successive, lo script utilizza il modulo 12 senza l'opzione —old_crc, corrispondente al TXQ PMD supportato.

'—dest_network' definisce la subnet dell'indirizzo di rete di destinazione (modifica in base allo schema dell'indirizzo IP di rete)

'—src_network' definisce la subnet dell'indirizzo di rete di origine (da modificare in base allo schema dell'indirizzo IP di rete)

'—port udp' definisce il protocollo utilizzato. L'utente può specificare il parametro del protocollo come "gre" o "tcp" o "udp" o qualsiasi valore decimale (FACOLTATIVO)

'—src_port' definisce la porta di origine utilizzata (OPTIONAL)

'—dst_port' definisce la porta di destinazione utilizzata (OPTIONAL)

'—unique_hash 1' genera un set (12 coppie per 12 TXQ) di indirizzi IP con hash univoco. È possibile modificare questa impostazione.

<#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
```

Topologia di esempio e configurazione CLI con 8 TXQ con interfacce di loopback

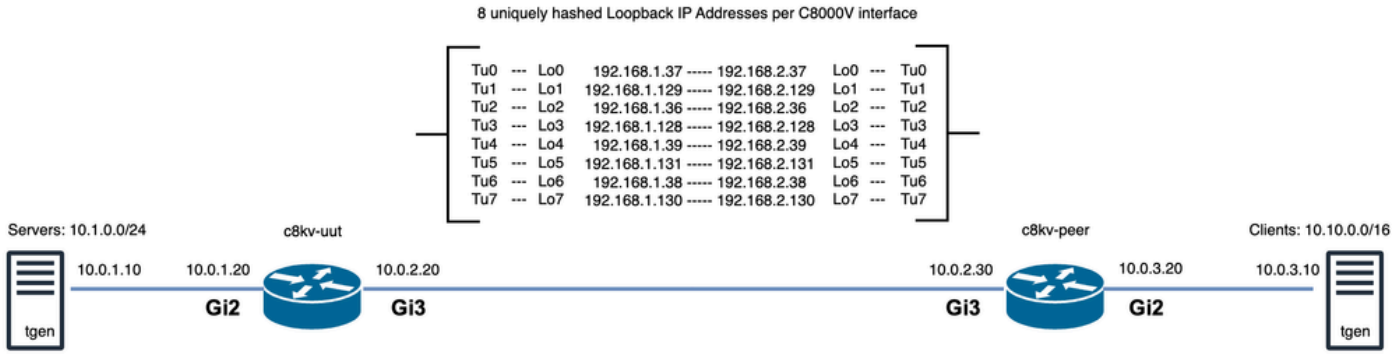


Figura 3: Topologia di esempio che utilizza otto TxQs utilizzando interfacce di loopback.

Questa è una configurazione CLI di esempio per 'c8kv-out' (Figura 3) che crea otto tunnel IPsec con interfacce loopback usando gli indirizzi IP hash calcolati (192.168.1.X) della sezione precedente.

Una configurazione simile sarebbe applicabile sull'altro endpoint del router (c8kv-peer) con gli altri otto indirizzi IP hash calcolati (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
```

Topologia di esempio e configurazione CLI con 12 TXQ con interfacce di loopback

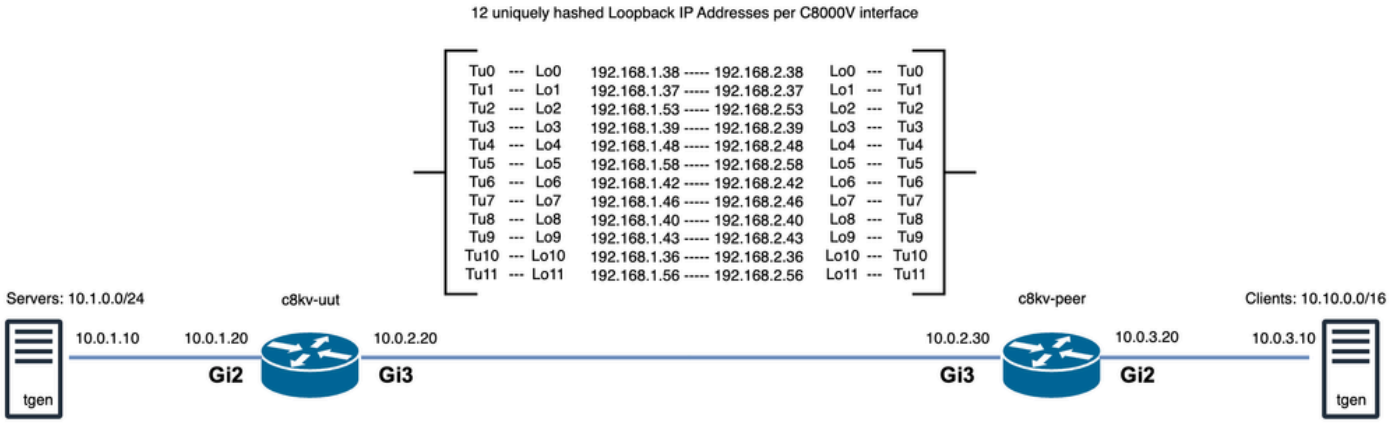


Figura 4. Topologia di esempio che utilizza dodici TxQ con interfacce di loopback.

Questa è una configurazione CLI di esempio per 'c8kv-out' (Figura 4) che crea dodici tunnel IPsec con interfacce di loopback utilizzando gli indirizzi IP hash calcolati (192.168.1.X) della sezione precedente.

Una configurazione simile sarebbe applicabile sull'altro endpoint del router (c8kv-peer) con gli altri otto indirizzi IP hash calcolati (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
```

Topologia di esempio e configurazione CLI con 12 TXQ con indirizzi IP secondari

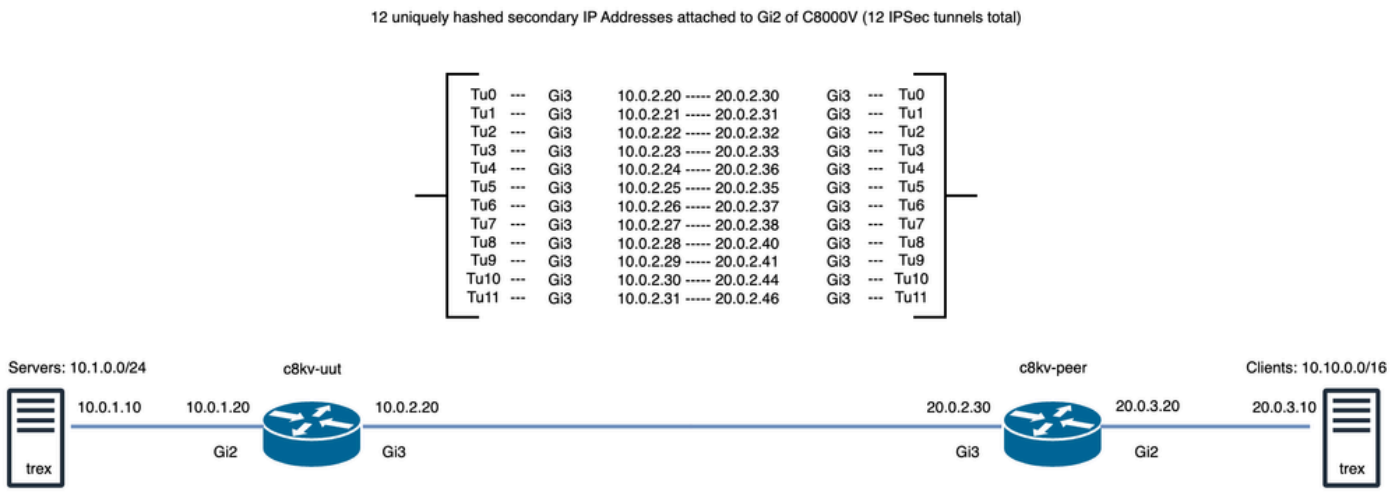
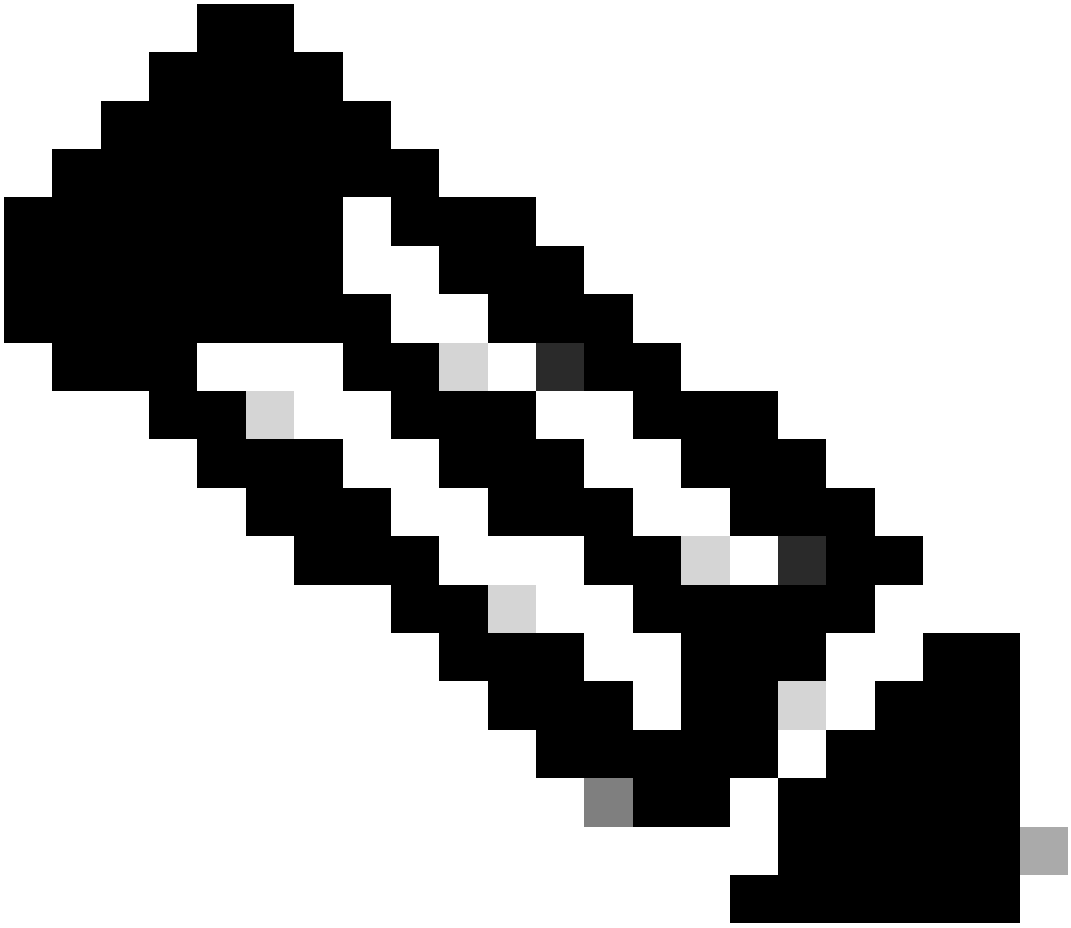


Figura 5. Topologia di esempio che utilizza dodici TxQ con indirizzi IP secondari.

Se non è possibile utilizzare gli indirizzi di loopback nell'ambiente AWS, è possibile utilizzare gli indirizzi IP secondari collegati all'ENI.

Questa è una configurazione CLI di esempio per 'c8kv-out' (Figura 5) che crea dodici tunnel IPsec la cui origine è 1 indirizzo IP primario + 11 indirizzi IP secondari collegati all'interfaccia Gigabit Ethernet3 utilizzando indirizzi IP con hash calcolato (10.0.2.X). Una configurazione simile verrebbe applicata all'altro endpoint del router (c8kv-peer) con gli altri dodici indirizzi IP hash calcolati (20.0.2.X).



Nota: In questo esempio viene utilizzato un secondo C8000V come endpoint del tunnel, ma è possibile utilizzare anche altri endpoint di rete cloud come 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
```

Installazione tipica di Catalyst 8000V in AWS

Modalità autonoma

Vedere gli esempi precedenti di configurazioni e topologie CLI. La configurazione CLI può essere copiata e modificata in base allo schema di indirizzamento di rete e agli indirizzi IP con hash generati.

Per una corretta creazione del tunnel, accertarsi di creare route IP sia sul C800V che sulle tabelle di routing sul VPC AWS.

Modalità SD-WAN

Questa è una topologia di esempio e una configurazione SD-WAN che crea TLOC utilizzando interfacce di loopback su C8000V in un VPC AWS.

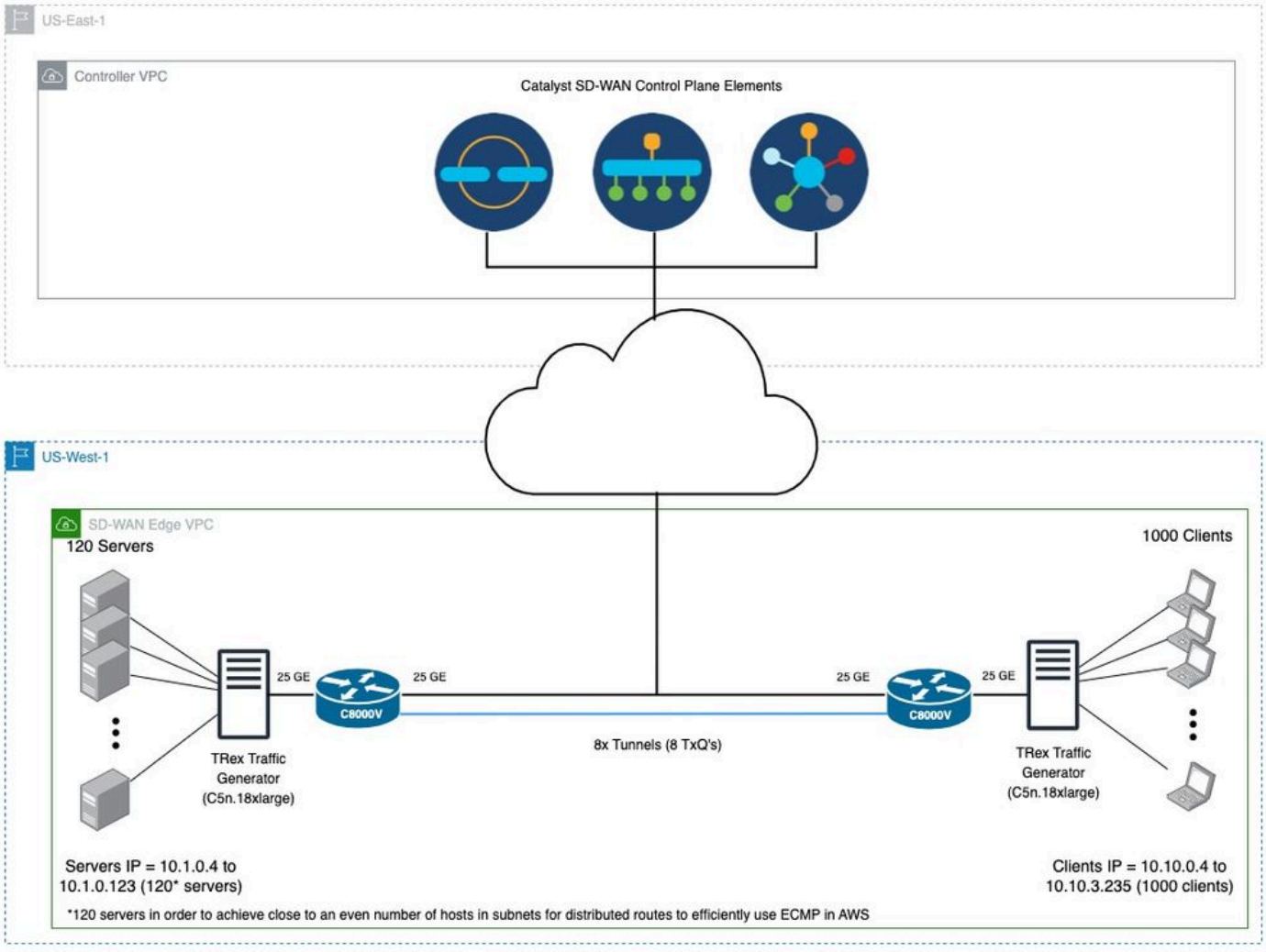
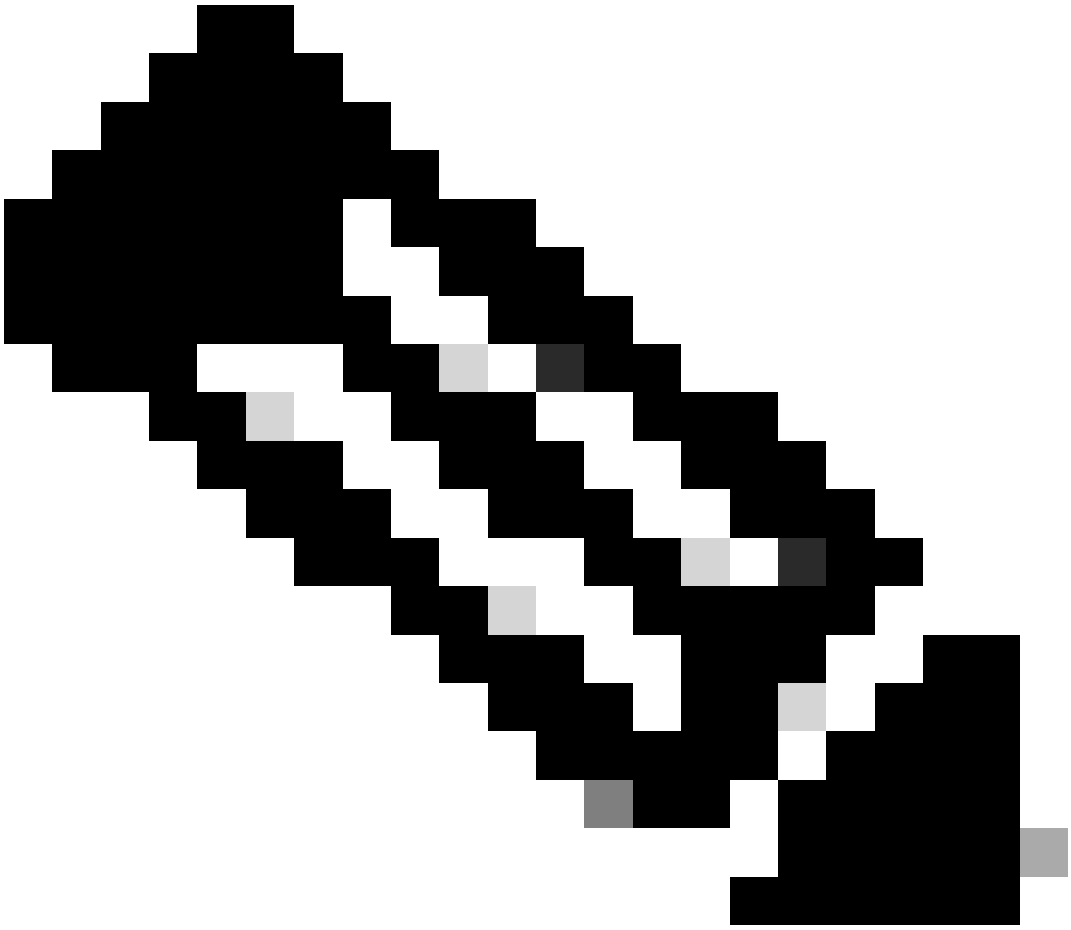


Figura 6. Topologia SD-WAN di esempio che utilizza TLOC con interfacce di loopback su C8000V in un VPC AWS.



Nota: Nella Figura 6, la connessione di colore nero rappresenta la connessione di controllo (VPN0) tra gli elementi del control plane SD-WAN e i dispositivi periferici SD-WAN. Le connessioni blu rappresentano i tunnel tra i due dispositivi periferici SD-WAN che utilizzano i TLOC.

È possibile trovare un esempio di configurazione della CLI SD-WAN per la Figura 6 (qui).

```
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
```

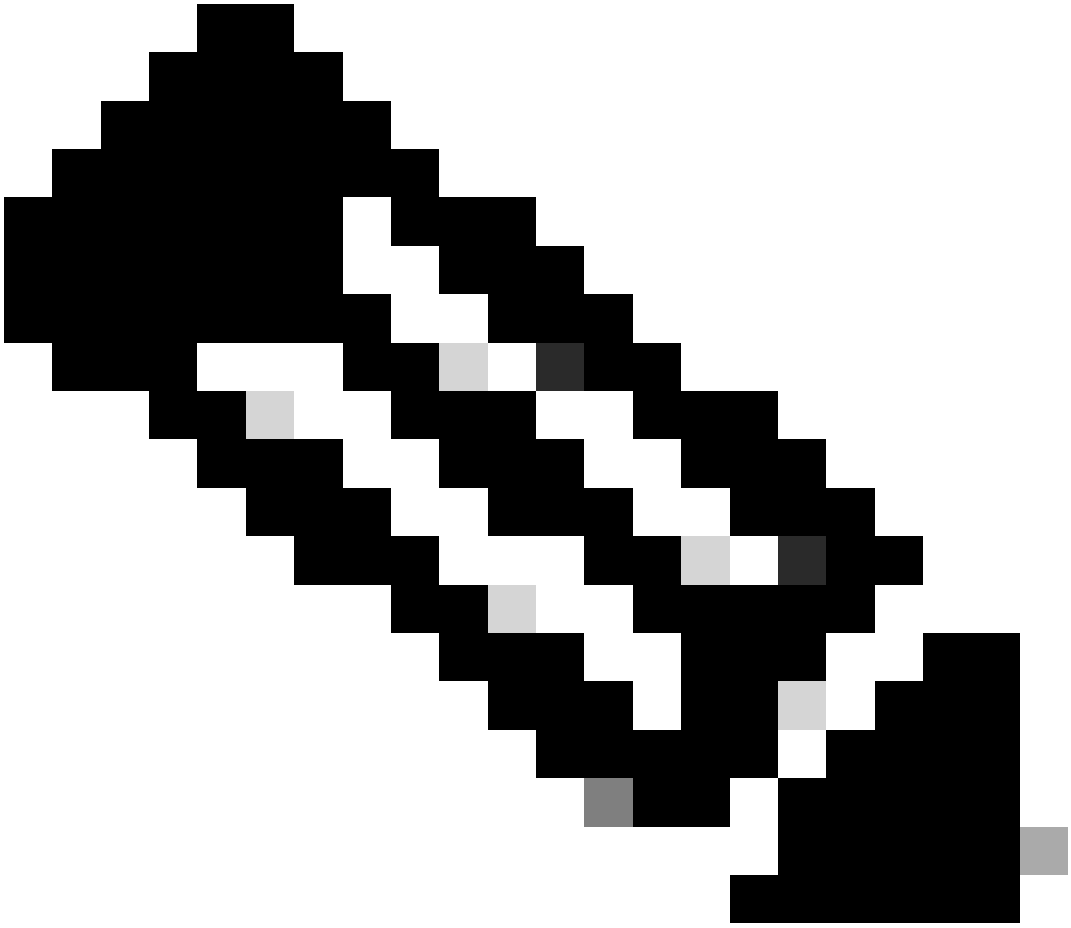
```
no allow-service sshd
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback4
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private5 restrict
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
!
```

Risoluzione dei problemi relativi alle prestazioni del throughput in AWS



Nota: L'esecuzione di test delle prestazioni in ambienti cloud pubblici introduce nuove variabili che possono influire sulle prestazioni di throughput. Di seguito sono riportati alcuni aspetti da considerare durante l'esecuzione di questi tipi di test:

- Utilizzo delle risorse sottostanti da parte dei peer al momento dell'esecuzione dei test
- Non utilizzare host dedicati (l'uso di host dedicati aumenta il costo del cloud di 16x)
- Il cloud viene eseguito in aree diverse, le prestazioni potrebbero variare
- In alcuni casi, i numeri sono simili indipendentemente dal profilo della feature; ciò è probabilmente dovuto alla limitazione di AWS sull'interfaccia per dimensione dell'istanza
- AWS limita la velocità dei pacchetti al secondo sulle istanze EC2, causando anche il rifiuto dei pacchetti
- AWS non indica la velocità di limitazione, ma è possibile osservare le riduzioni dovute alla limitazione di PPS tramite il contatore 'pps_allowy_exceeded'

Comandi utili per la risoluzione dei problemi della CLI

Quando si eseguono i test delle prestazioni del throughput, questi comandi di risoluzione dei problemi possono essere utilizzati per individuare i colli di bottiglia o i motivi del peggioramento delle prestazioni.

"show platform hardware qfp active statistics drop" - ci permette di capire se ci sono cadute sul c8kv. Dobbiamo assicurarci che non ci siano gocce di coda significative o contatori rilevanti in aumento.

"show platform hardware qfp active statistics drop clear" - Questo comando cancella i contatori.

"show platform hardware qfp active datapath infrastructure sw-cio" - Questo comando ci fornisce informazioni dettagliate sulla percentuale di processore pacchetti (PP), Traffic Manager (TM) utilizzati durante le esecuzioni delle prestazioni. In questo modo è possibile determinare se la capacità di elaborazione è sufficiente o meno rispetto a c8kv.

"show platform hardware qfp active datapath summary" - Questo comando ci fornisce le informazioni complete sull'input/output che il datapath c8kv trasmette/riceve da tutte le porte.

Verificare la velocità di input/output e controllare se si verifica una caduta. Verificare inoltre la percentuale di carico di elaborazione. Se raggiunge il 100%; significa che c8kv ha raggiunto la sua capacità.

"show plat hardware qfp active infrastructure bqs interface Gigabit EthernetX" - Questo comando ci permette di controllare le statistiche a livello di interfaccia in termini di numero di coda, larghezza di banda, velocità.

"show controller" - Questo comando ci fornisce informazioni molto granulari sui pacchetti buoni rx/tx, e sui pacchetti mancanti.

Questo comando può essere usato in uno scenario in cui non si notano cali di coda ma il generatore del traffico mostra ancora il rilascio.

Questo può accadere in uno scenario in cui l'utilizzo dei dati sta già raggiungendo il 100% e allo stesso modo il PP al 100%.

Se i contatori rx_missing_errors continuano ad aumentare, significa che il CSR sta eseguendo il backpressuring dell'infrastruttura cloud in quanto non è in grado di elaborare altro traffico.

"show platform hardware qfp active datapath infrastructure sw-hqf" - può essere usato per verificare eventuali congestioni causate dalla pressione di AWS.

"show platform hardware qfp active datapath infrastructure sw-nic" - Determina in che modo viene bilanciato il carico del traffico tra più code. Dopo la 17.7 abbiamo 8 Multi-TXQ.

Inoltre, può determinare se esiste una particolare coda che sta prendendo tutto il traffico o sta bilanciando correttamente il carico.

"show controller | in errors|exceeded|Giga" - Mostra le perdite di pacchetti dovute alla limitazione PPS eseguita dal lato AWS, che può essere rilevata tramite il contatore pps_allowedx.

Output CLI di esempio

Output di esempio in cui il contatore Tail drops continua ad aumentare - Eseguire il comando più volte per verificare se i contatori sono in aumento, consentendo così di confermare che si tratta effettivamente di tail drops.

<#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
```

Output di esempio mostrato qui - Eseguire il comando ogni 30 secondi per ottenere i dati in tempo reale

<#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
```

Uscita di esempio mostrata qui - Assicurarsi di controllare la velocità di input/output e vedere se c'è una caduta. Verificare inoltre la percentuale di carico di elaborazione. Se raggiunge il 100%; significa che il nodo ha raggiunto la propria capacità.

<#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
```

Di seguito è riportato un esempio di output per le statistiche a livello di interfaccia:

<#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
```

Output di esempio per le statistiche dei pacchetti validi RX/TX, dei pacchetti perduti

<#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
```

Output di esempio per controllare se si verifica una congestione dovuta alla contropressione da 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
```

Output di esempio per il bilanciamento del carico del traffico tra più code:

```
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
```

Output di esempio che mostra le perdite di pacchetti causate dalla limitazione PPS eseguita dal lato AWS, rilevabili tramite il contatore pps_allowedx:

```
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
```

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).