# AWSでのマルチTxQによるCatalyst 8000Vのスループットの向上

## 内容



## はじめに

このドキュメントでは、AWS環境にデプロイされたCatalyst 8000Vでマルチ TXQを有効にして利用し、スループットパフォーマンスを向上させる方法について説明します。

## 背景説明

複数のキューが存在するため、着信パケットと発信パケットを特定のvCPUにマッピングするプロセスが簡素化され、高速化されます。Catalyst 8000Vでマルチ TXQを使用すると、割り当てられた使用可能なデータプレーンコア全体でコアを効率的に使用できるようになり、スループットパフォーマンスが向上します。この記事では、マルチTXQの動作方法と設定方法の概要を説明し、AutonomousとSD-WAN Catalyst 8000Vの両方の導入で使用できるサンプルCLI設定を示します。また、パフォーマンスのボトルネックを見つけるためのトラブルシューティングコマンドを紹介します。

# マルチTxQを使用していない場合のCatalyst 8000Vの動作

17.18のソフトウェアリリースまでは、Catalyst 8000Vに入るパケットは、フローに関係なくすべてのvCPU（パケット処理コア）に分散されます。PPがパケット処理を完了すると、フローの順序が復元され、インターフェイスに送信されます。

パケットを送信キュー(TxQ)に入れる前に、Catalyst 8000Vはインターフェイスごとに1つのTxQを作成します。したがって、使用できる出力インターフェイスが1つだけの場合、複数のフローが1つのTxQに入ります。

使用可能なインターフェイスが1つしかない場合、Catalyst 8000Vはこのマルチ TxQプロセスを利用できません。その結果、スループットパフォーマンスのボトルネックが発生し、使用可能なデータプレーンコア間で負荷分散が不均等になります。C8000Vインスタンスからデータを送信するために使用される出力インターフェイスが1つしかない場合、ネットワークトラフィックの送信に使用できるTxQは1つだけになり、単一のキューがすぐにいっぱいになるためにパケットがドロップされる可能性があります。

AWSにデプロイされたCatalyst 8000Vの単一TxQアーキテクチャモデルについては、図1を参照してください。

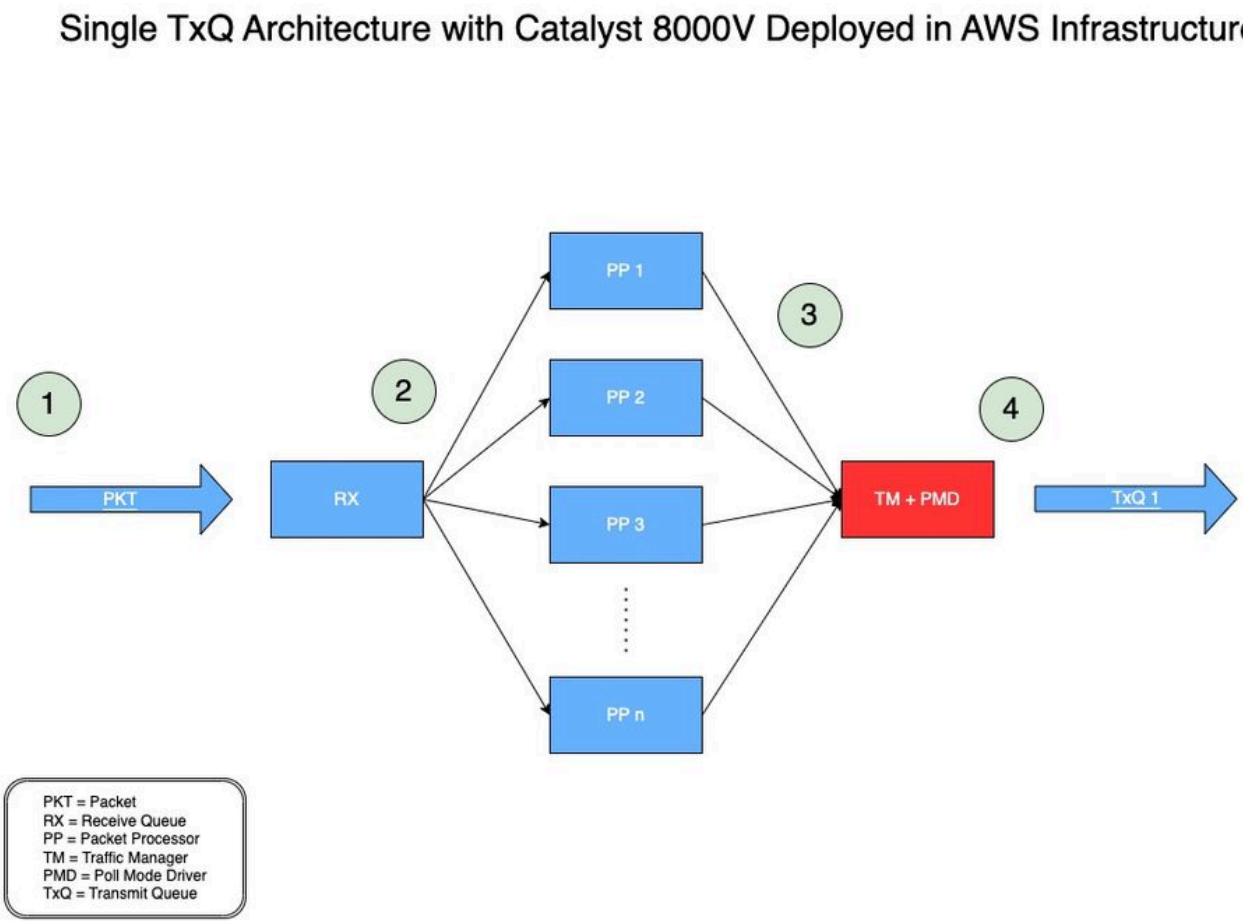


図1: AWSにデプロイされたCatalyst 8000Vの単一TxQアーキテクチャモデル。

1. ネットワークパケット(PKT)がVPCを通過し、C8000Vの入力インターフェイスで受信される。
2. PKTは受信キュー(RX)に格納され、アルゴリズムによって決定されたパケットプロセッサ(PP)エンジンに転送されます。
3. パケットプロセッサ(PP)はパケットを処理した後、パケットをTraffic Manager(TM)に送信します。
4. TM処理の終了時には、使用可能な1つのTxQにパケットを配置する責任が1つのコアにあり、それがCatalyst 8000Vの出力インターフェイスに転送されます。

# AWSインフラストラクチャのマルチTXQとは

AWS ENAは、内部のオーバーヘッドを削減し、拡張性を高めるために、複数の送信キュー（マルチTxQ）を提供しています。複数のキューが存在するため、着信パケットと発信パケットを特定のvCPUにマッピングするプロセスが簡素化され、高速化されます。AWSおよびDPDKネットワーク参照モデルはフローベースであり、各vCPUがフローを処理し、そのフローからパケットを割り当てられた送信キュー(TxQ)に送信します。 各vCPUのRX/TXキューペアは、フローベースモデルに基づいて有効です。

Catalyst 8000Vはフローベースではないため、「各vCPUのRX/TXキューペア」という文はCatalyst 8000Vには適用されません。

この場合、RX/TXキューはvCPUごとではなく、インターフェイスごとに存在します。RX/TXキューは、アプリケーション(Catalyst 8000V)とAWSインフラストラクチャ/ハードウェア間のインターフェイスとして機能し、データ/ネットワークトラフィックを送信します。AWSは、インスタンスごとにインターフェイスごとに使用できるRX/TXキューの速度と数を制御します。

Catalyst 8000Vで複数のTxQを作成するには、複数のインターフェイスが必要です。インターフェイスから送信される複数のフローでフローの順序を維持するため（このプロセスの後でCatalyst 8000Vが複数のTxQをイネーブルにすると）、Catalyst 8000Vは5タプルに基づいてフローをハッシュし、適切なTxQを選択します。ユーザは、ループバックインターフェイスまたはセカンダリIPアドレスを使用して、インスタンスに接続されている同じ物理NICを使用して、Catalyst 8000V上に複数のインターフェイスを作成できます。

図2では、AWSでCatalyst 8000Vを使用するMulti-TxQアーキテクチャを使用してパケットがどのように処理されるかを確認できます。

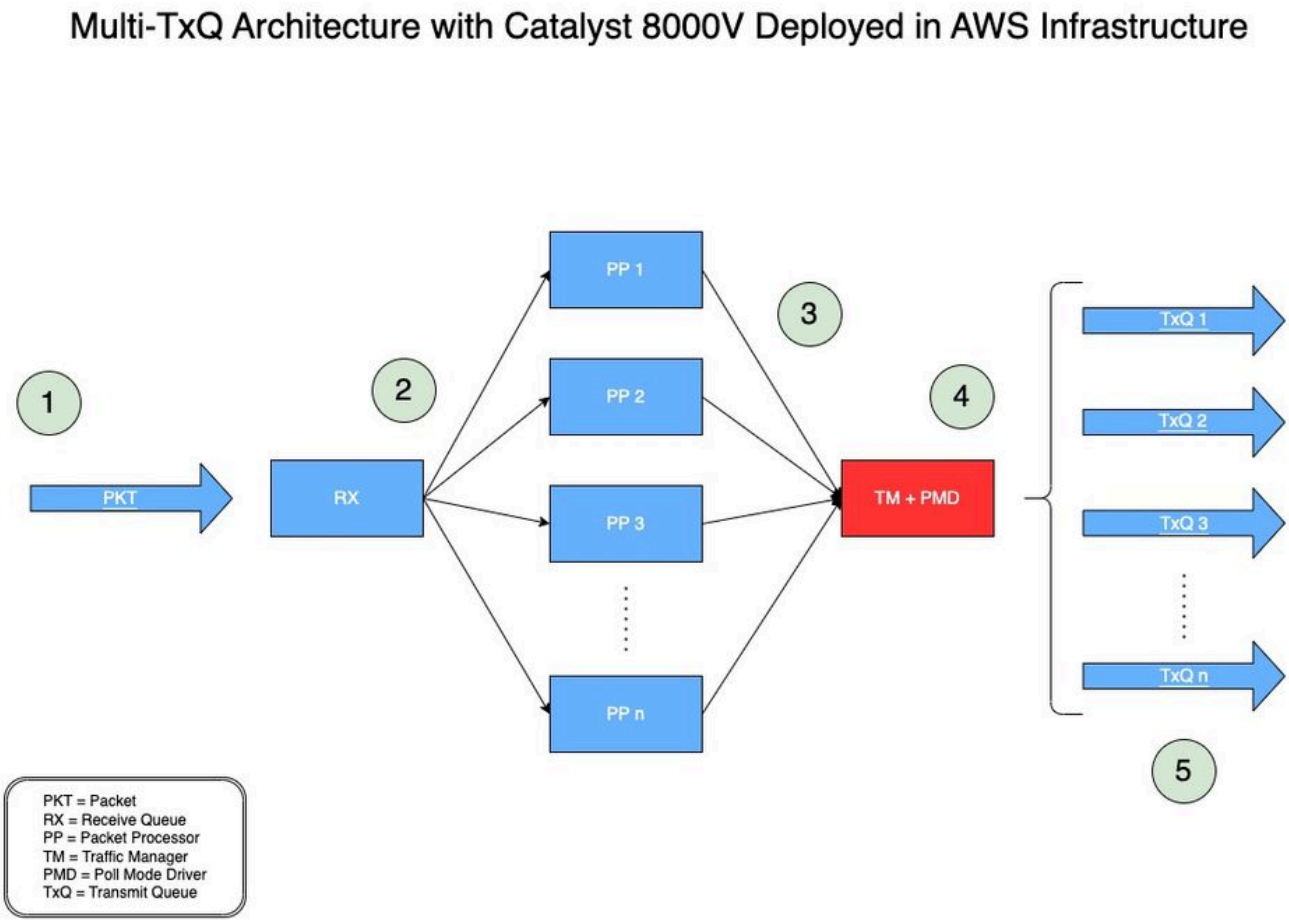



図2: AWSにデプロイされたCatalyst 8000Vのマルチ TxQアーキテクチャモデル。

1. ネットワークパケット(PKT)がVPCを通過し、C8000Vの入力インターフェイスで受信される。
2. PKTは受信キュー(RX)に格納され、アルゴリズムによって決定されたパケットプロセッサ(PP)エンジンに転送されます。
3. パケットプロセッサ(PP)はパケットを処理した後、パケットをTraffic Manager(TM)に送信します。
4. TM処理の最後に、パケットを送信キュー(TxQ)に入れる前に、TMはパケットヘッダーを参照してパケットをハッシュします（次のセクションで説明します）。 もう1つのコンポーネントであるPoll Mode Driver(PMD)は、インスタンスでサポートされるTXQの数を設定するために使用されます。1つのコアは、割り当てられたTxQに対してパケットのハッシュと送信を行うTM + PMD機能専用です。
5. TxQは、インスタンスでサポートされているTxQの数に基づいて、ハッシュおよびモジュロされた5つのタプルに基づいて選択されます。パケットは選択されたTxQに送信され、Catalyst 8000Vの出力インターフェイスに転送されます。

## トラフィックをマルチTxQにハッシュする方法

図2のステップ4に示すように、TM処理の最後に、パケットをTxQに配置する前に、TMはパケッ

トヘッダーを参照して5つのタプル（宛先アドレス、送信元アドレス、プロトコル、宛先ポート、および送信元ポート）を抽出し、パケットをTxQにハッシュします。

TxQは、インスタンスでサポートされているTxQの数に基づいて、ハッシュおよびモジュロされた5つのタプルに基づいて選択されます。

## マルチTxQをサポートするCatalyst 8000Vソフトウェアバージョン

同じインスタンスファミリタイプのAWS EC2インスタンスはすべて、インスタンスサイズに応じて異なる数のTXQをサポートします。C8000Vは、IOS® XE 17.7以降、複数のTxQのサポートを開始しました。

IOS® XE 17.7以降、C8000VはC5n.9xlargeで複数のTxQをサポートし、最大8つのTXQを使用できます。

IOS® XE 17.9以降、C8000VはC5n.18xlargeインスタンスサイズをサポートし、最大12個のTXQを使用できます（C5n.9xlargeよりも50 %多い）。

Multi-TxQはIOS® XE 17.7からサポートされていますが、ソフトウェアライフサイクルと高スループットパフォーマンス機能の両方を実現するIOS® XE 17.9と12 TxQのサポートを使用することを強くお勧めします。

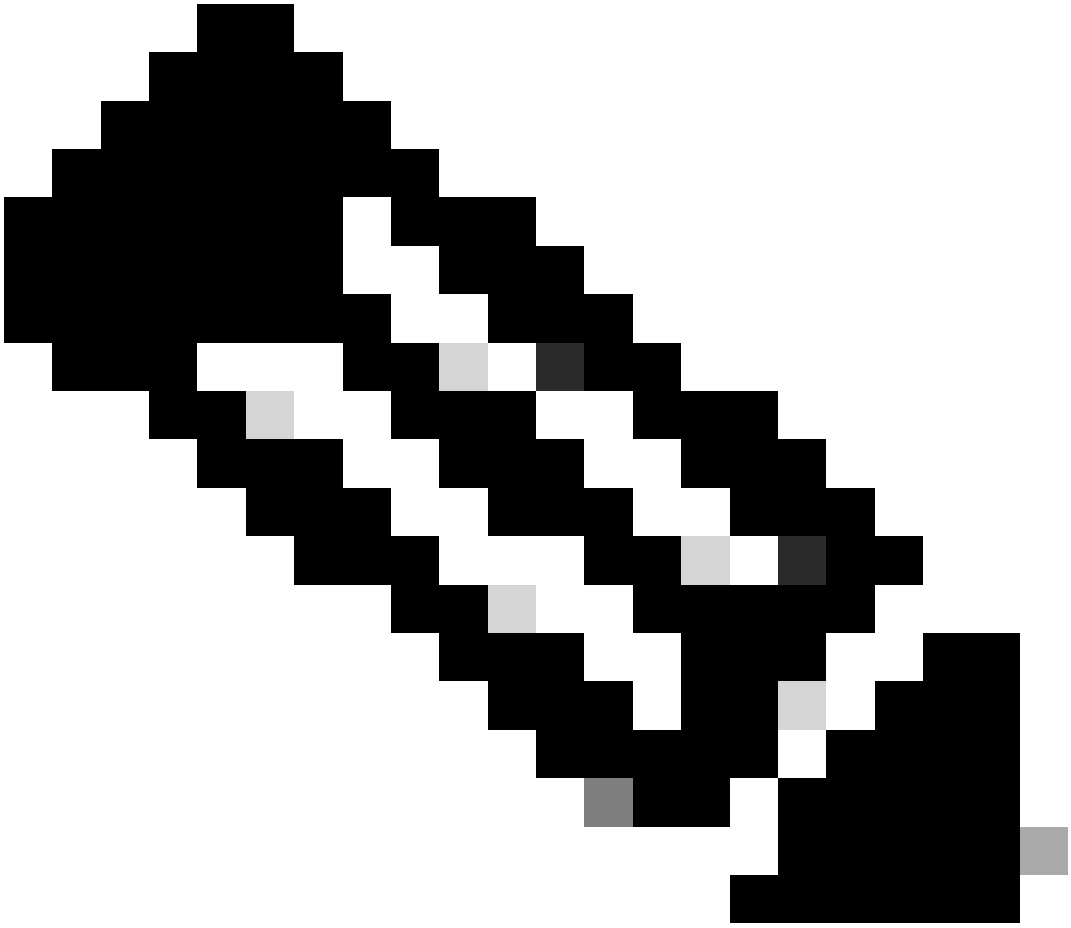
## ハッシュを計算するためのIPアドレッシング方式の設計方法

使用可能なすべてのTxQ間でトラフィックを均等にハッシュするには、Catalyst 8000VがIPsec/GREトンネルを終端するときに特別なIPアドレスを使用する必要があります。

これらのトンネルを終端するCatalyst 8000Vインターフェイスの設定に使用される特別なIPアドレスを生成するために、公開スクリプトを使用できます。このセクションでは、スクリプトをダウンロードして使用し、Multi-TxQハッシングに必要なIPアドレスを設計する方法について説明します。

Catalyst 8000VがTCP/UDPなどのクリアテキストトラフィックを処理している場合は、特別なIPアドレッシング方式は必要ありません。

元の手順については、https://github.com/CiscoDevNet/python-c8000v-aws-multitx-queues/を参照してください。



注:17.18以降を実行するCatalyst 8000Vでは、パケットの分散が異なります。したがって、別のハッシュアルゴリズムを使用する必要があります。

## 前提条件

- Linux/MacOS、またはPythonスクリプトを実行できるWindowsマシンが必要です。
- Pythonのバージョンが3.8.9以降であることを確認します。 'python3 —version'を使用してPythonのバージョンを確認します。
- PIPがインストールされていない場合はインストールします。そうでない場合は、次のコマンドを実行します。
  - curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  - python3のget-pip.py

マシンが使用しているPythonのバージョンは'python3 —version'コマンドで確認できます。

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

```
Python 3.9.6
```

Pythonのバージョンが確認され、実行されている3.8.9以降のバージョンでは、PIPの最新バージョンをインストールします。

```
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
```

# 仮想環境の作成

前提条件がインストールされたら、仮想環境を作成し、Multi-TxQの一意のIPアドレススキームを生成するために使用されるIPアドレスハッシュスクリプトをダウンロードします。

コマンドの要約：

1. python3 -m venv c8kv-hash
2. cd c8kvハッシュ
3. ソースビン/アクティブ化
4. git clone https://github.com/CiscoDevNet/python-c8000v-aws-multitx-queues/
5. cd c8kv-aws-pmd-hash
6. python3 -m pip install —pipをアップグレードする
7. pip install -r requirements.txt

Pythonの仮想環境は、他のプロジェクトや依存関係に影響を与えない独立したワークスペースを作成するために使用されます。次のコマンドを使用して、仮想環境「c8kv-hash」を作成します。

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

仮想環境の内部で、「c8kv-hash」フォルダ（前に作成）に移動します。

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

仮想環境をアクティブにします。

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

Multi-TxQハッシュPythonスクリプトを含むリポジトリをクローニングします。

```
(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.
```

リポジトリがクローンされたら、「c8kv-aws-pmd-hash」フォルダに移動します。これは作成さ

れた仮想環境にあるため、最新バージョンのPIPをインストールします。

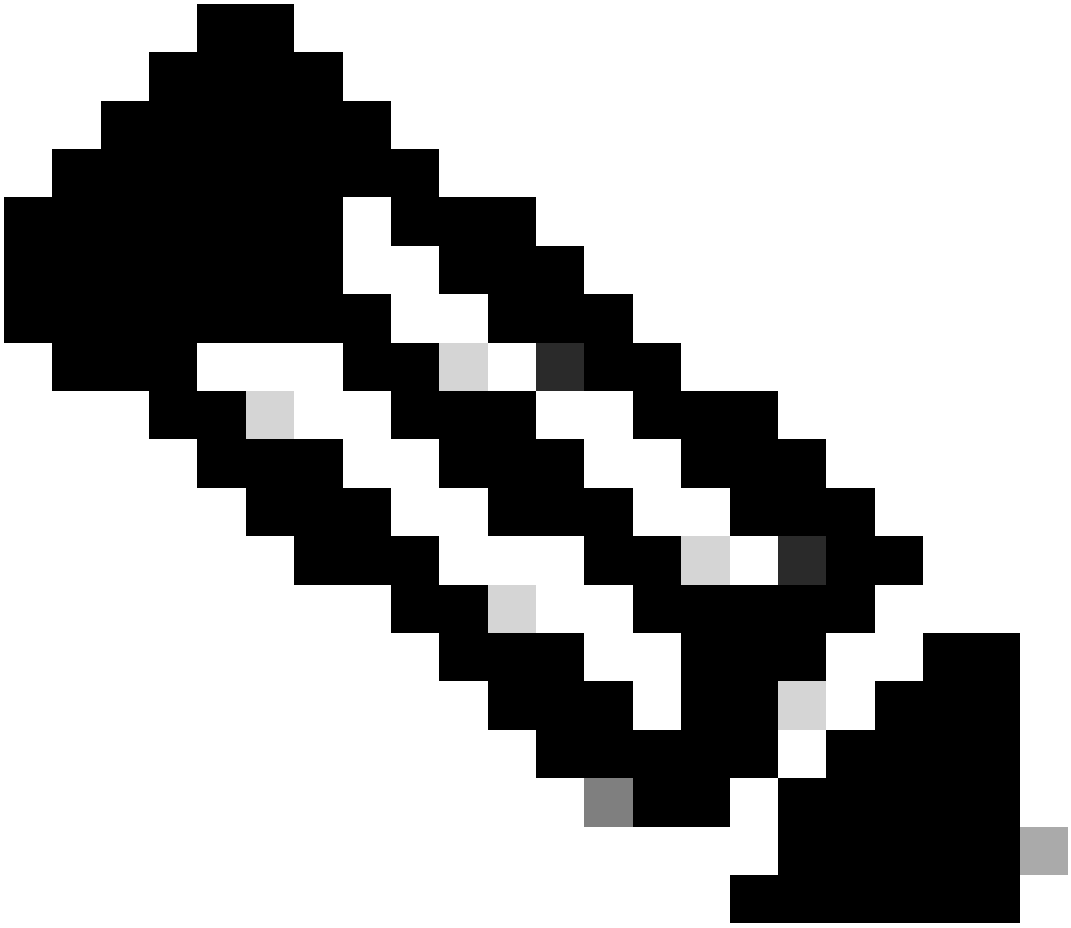
```
(c8kv-hash) user@computer c8kv-hash % cd c8kv-aws-pmd-hash
(c8kv-hash) user@computer c8kv-aws-pmd-hash % python3 -m pip install --upgrade pip

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
```

PIPがアップグレードされたら、フォルダのrequirements.txtファイルにある依存関係をインストールします。

```
(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
```

仮想環境は最新の状態になり、Multi-TxQのIPアドレス計画の生成に使用できます。

# 17.7および17.8リリース用のPythonハッシュインデックススクリプトを使用したIPアドレス方式の計算（非推奨）



注:7.7および17.8のハッシュスクリプトは近日中に廃止される予定です。17.9ハッシュスクリプトの使用を強く推奨します

---

コマンドの要約：

- 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'は、サポートされているPMD TXQに一致するように、モジュロ8を使用した17.7および17.8リリースに基づいてハッシュインデックスを生成します（変更しないでください）。

'—dest_network'は、宛先ネットワーク・アドレスのサブネットを定義します（ネットワークIPアドレス方式に基づいて変更します）。

'—src_network'は、送信元のネットワークアドレスサブネットを定義します（ネットワークIPアドレス方式に基づいて変更してください）。

'—unique_hash 1'は、一意にハッシュされたIPアドレスの1セット（8つのTXQに8ペア）を生成し

ます。これは変更できます。

<#root>

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

```
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**

# 17.9以降のリリースでPythonハッシュインデックススクリプトを使用してIPアドレス方式を計算する

コマンドの要約：

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

IOS® XEリリース17.9以降では、スクリプトは – old_crcオプションを指定せずにmodulo 12を使用し、サポートされているPMD TXQと一致することに注意してください。

'—dest_network'は、宛先ネットワーク・アドレスのサブネットを定義します（ネットワークIPアドレス方式に基づいて変更します）。

'—src_network'は、送信元のネットワークアドレスサブネットを定義します（ネットワークIPアドレス方式に基づいて変更してください）。

'—prot udp'は使用されるプロトコルを定義します。ユーザは、プロトコルパラメータを「gre」、「tcp」、「udp」、または任意の10進数値（オプション）として指定できます。

'—src_port'は使用される送信元ポートを定義します（オプション）。

'—dst_port'は使用する宛先ポートを定義します（オプション）。

'—unique_hash 1'は、一意にハッシュされたIPアドレスの1セット（12 TXQに対して12ペア）を生成します。これは変更できます。

<#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
```

## ループバックインターフェイスと8つのTXQを使用したトポロジおよびCLIの設定例

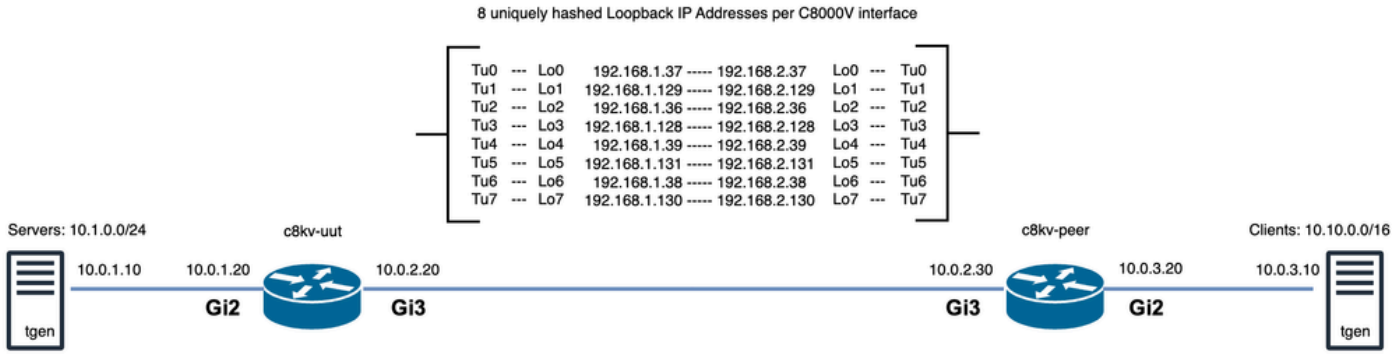



図3：ループバックインターフェイスを使用して8つのTxQを使用するトポロジ例。

これは、「c8kv-uut」（図3）のCLI設定例であり、前のセクションで計算されたハッシュIPアドレス(192.168.1.X)を使用して、ループバックインターフェイスを持つ8つのIPsecトンネルを作成します。

同様の設定が、残りの8個の計算されたハッシュIPアドレス(192.168.2.X)を使用して、他のルータエンドポイント(c8kv-peer)に適用されます。

```
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
```

# 12のTXQとループバックインターフェイスを使用したトポロジおよびCLIの設定例

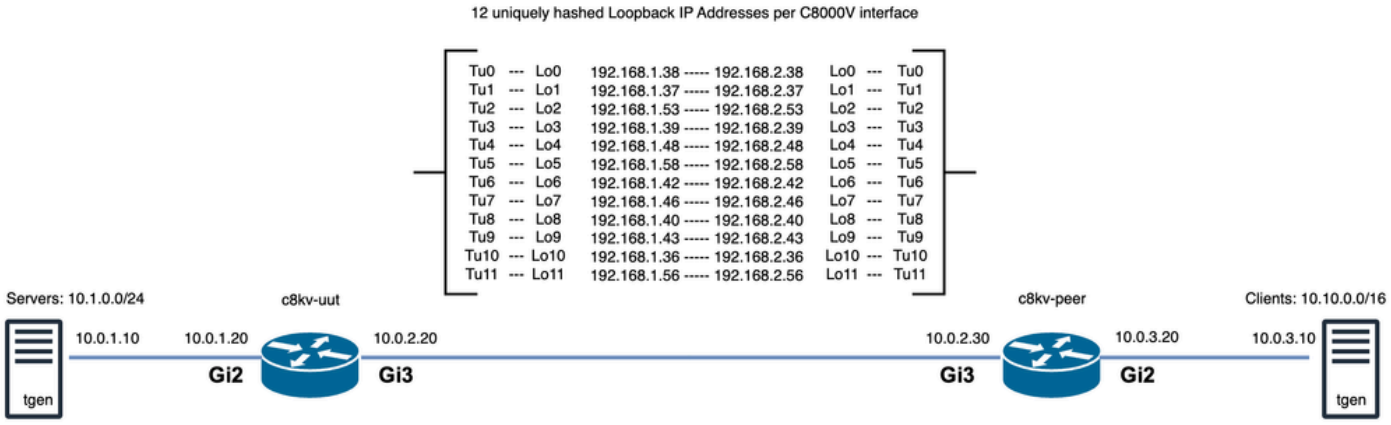



図 4： ループバックインターフェイスを使用する12個のTxQを使用するトポロジ例。

これは、「c8kv-uut」（図4）のCLI設定例であり、前のセクションで計算されたハッシュIPアドレス(192.168.1.X)を使用して、ループバックインターフェイスを持つ12のIPsecトンネルを作成します。

同様の設定が、残りの8個の計算されたハッシュIPアドレス(192.168.2.X)を使用して、他のルータエンドポイント(c8kv-peer)に適用されます。

```
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
```

# セカンダリIPアドレスを持つ12のTXQを使用したトポロジおよびCLIの設定例

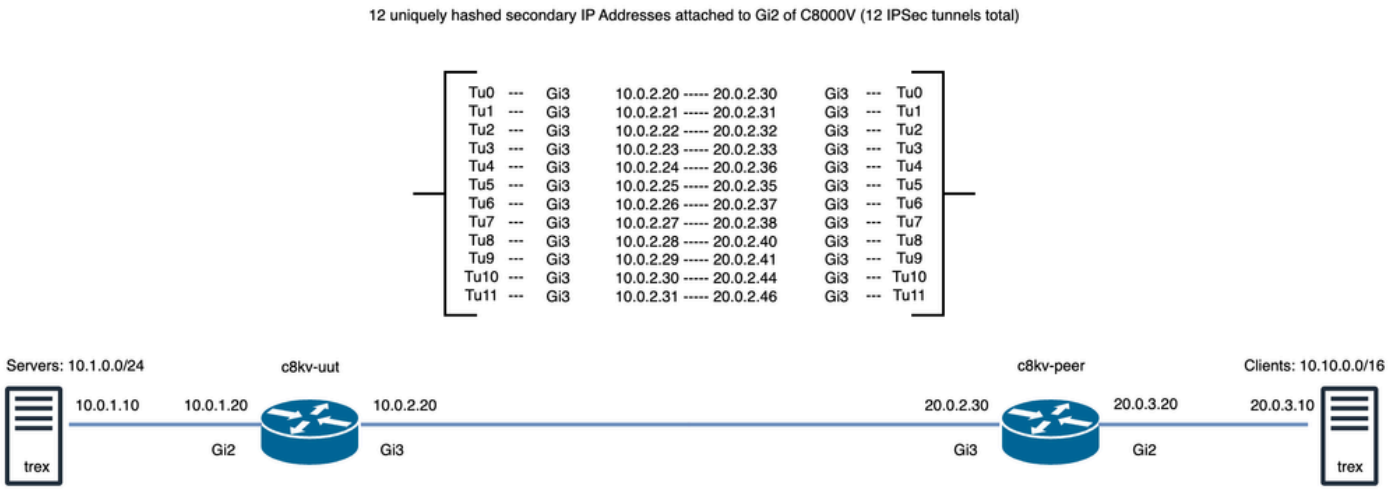
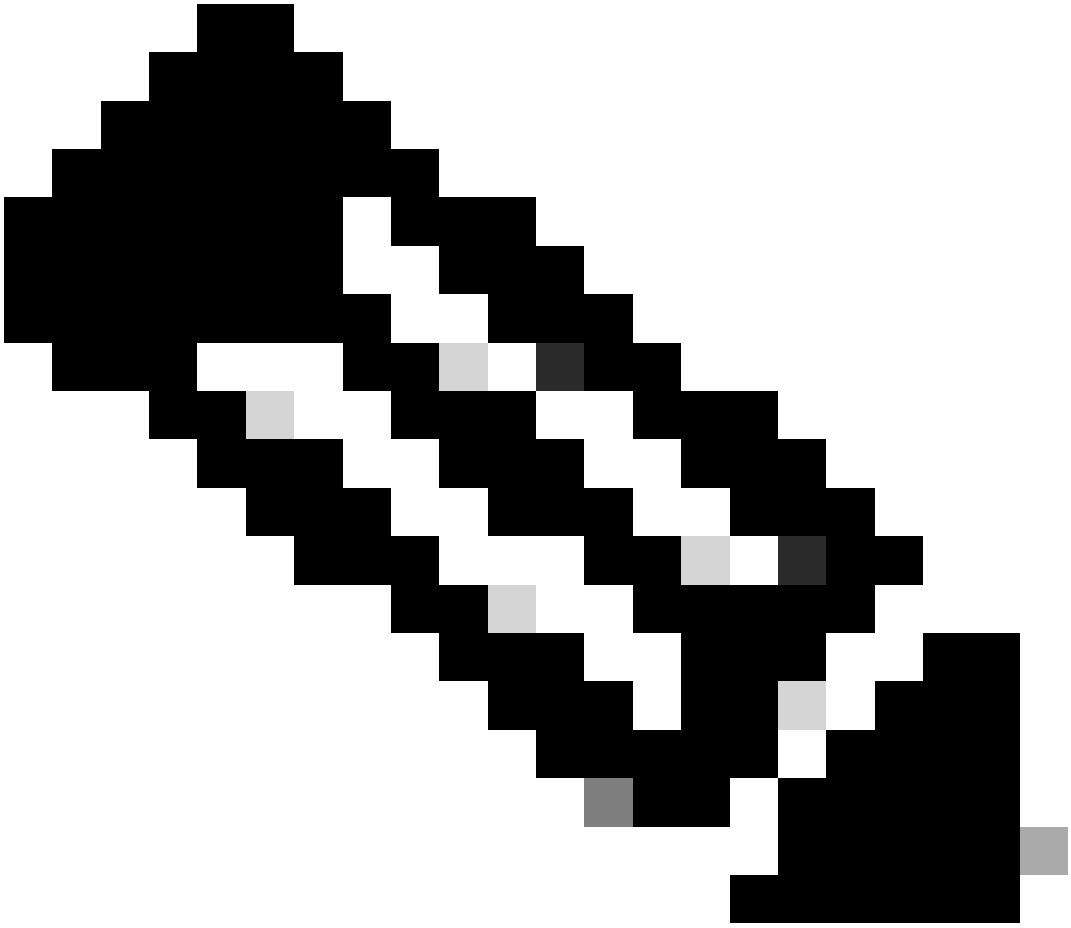



図 5. セカンダリIPアドレスを使用する12個のTxQを使用するトポロジ例。

ループバックアドレスをAWS環境で使用できない場合は、代わりにENIにアタッチされたセカンダリIPアドレスを使用できます。

これは、「c8kv-uut」（図5）のCLI設定例であり、計算されたハッシュIPアドレス(10.0.2.X)を使用して、GigabitEthernet3インターフェイスに接続された1個のプライマリIPアドレス+11個のセカンダリIPアドレスを送信元とする12個のIPsecトンネルを作成します。 同様の設定が、残りの12個の計算されたハッシュIPアドレス(20.0.2.X)を使用して、他のルータエンドポイント(c8kv-peer)に適用されます。



注：この例では、2つ目のC8000Vをトンネルエンドポイントとして使用していますが、TGWやDXなどの他のクラウドネットワーキングエンドポイントも使用できます。

```
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
```

# AWSでの一般的なCatalyst 8000Vのデプロイ

## 自律モード

前述のCLIの設定例とトポロジを参照してください。CLI設定は、ネットワークアドレッシング方式と生成されたハッシュIPアドレスに基づいてコピーおよび変更できます。

トンネルを正常に作成するには、C8000VとAWS VPCのルーティングテーブルの両方でIPルートを作成してください。

## SD-WANモード

これは、トポロジの例と、AWS VPC内にあるC8000V上のループバックインターフェイスを使用してTLOCを作成するSD-WAN設定です。

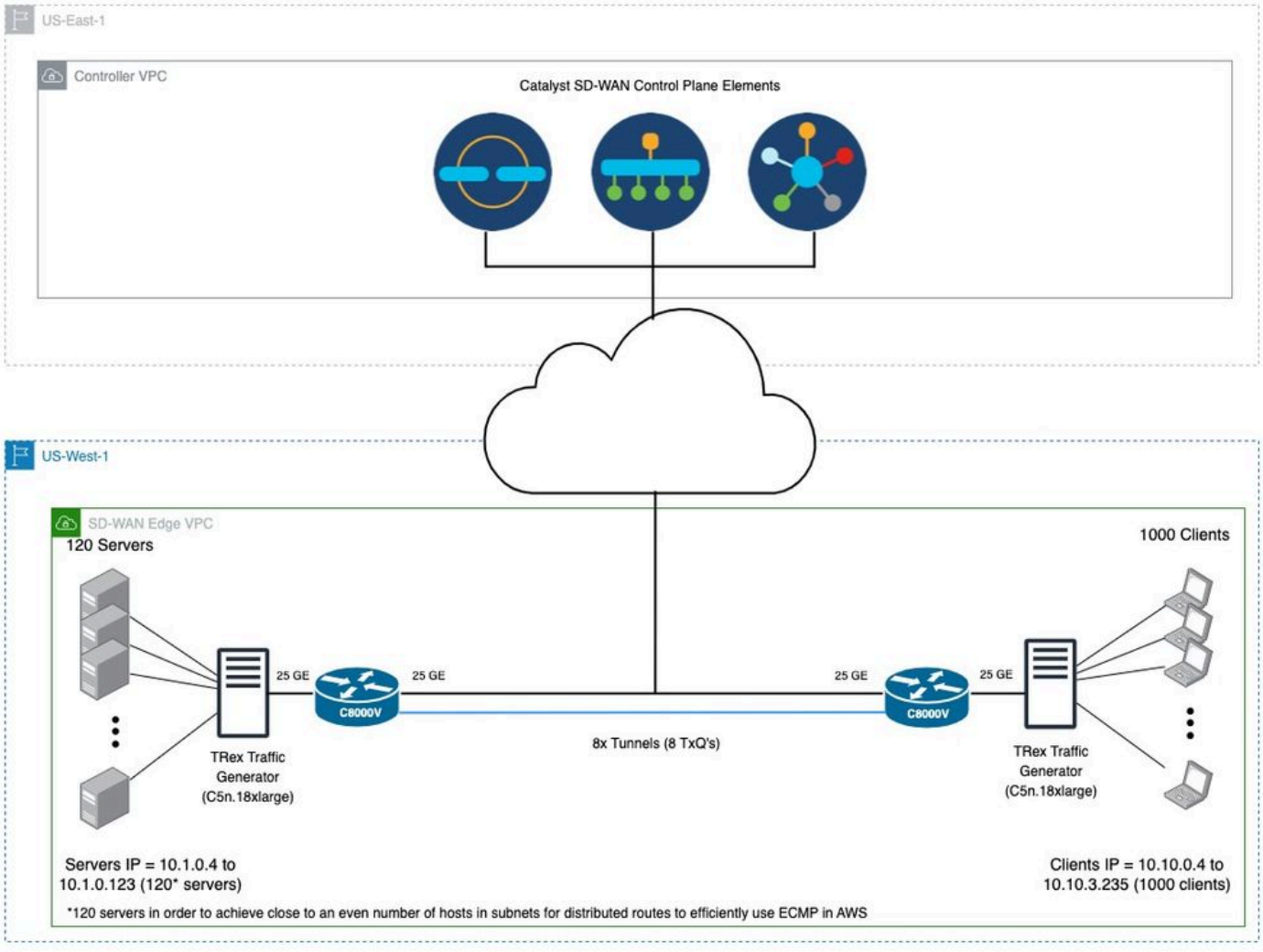
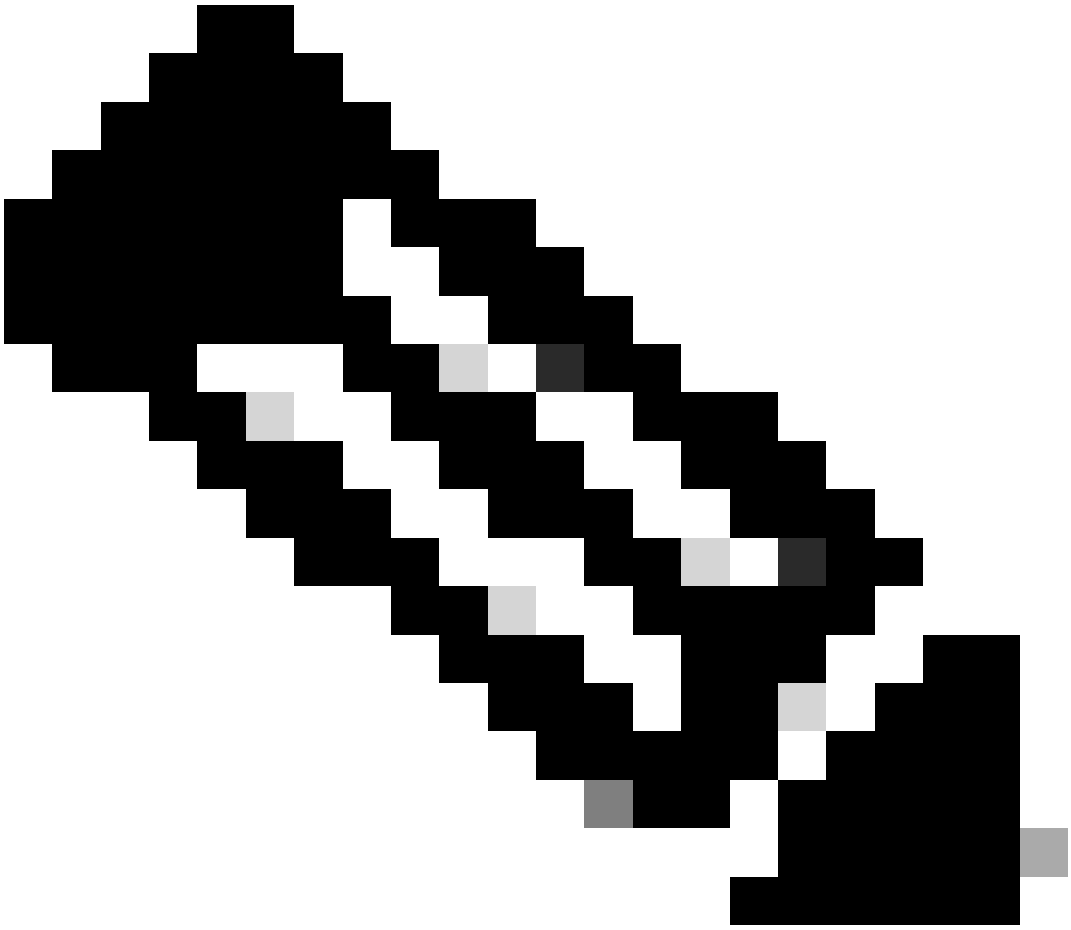


図 6.AWS VPC内のC8000V上のループバックインターフェイスでTLOCを使用するSD-WANトポロジの例。

---



注：図6では、黒色の接続はSD-WANコントロールプレーン要素とSD-WANエッジデバイス間のコントロール(VPN0)接続を表しています。青色の接続は、TLOCを使用する2つのSD-WANエッジデバイス間のトンネルを表します。

---

図6のSD-WAN CLIの設定例を参照してください（ここ）。

```
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 Tunnel1409501
no shutdown
ip unnumbered Loopback1
no ip redirects
ipv6 unnumbered Loopback1
no ipv6 redirects
tunnel source Loopback1
tunnel mode sdwan
exit
interface Tunnel1409502
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
```

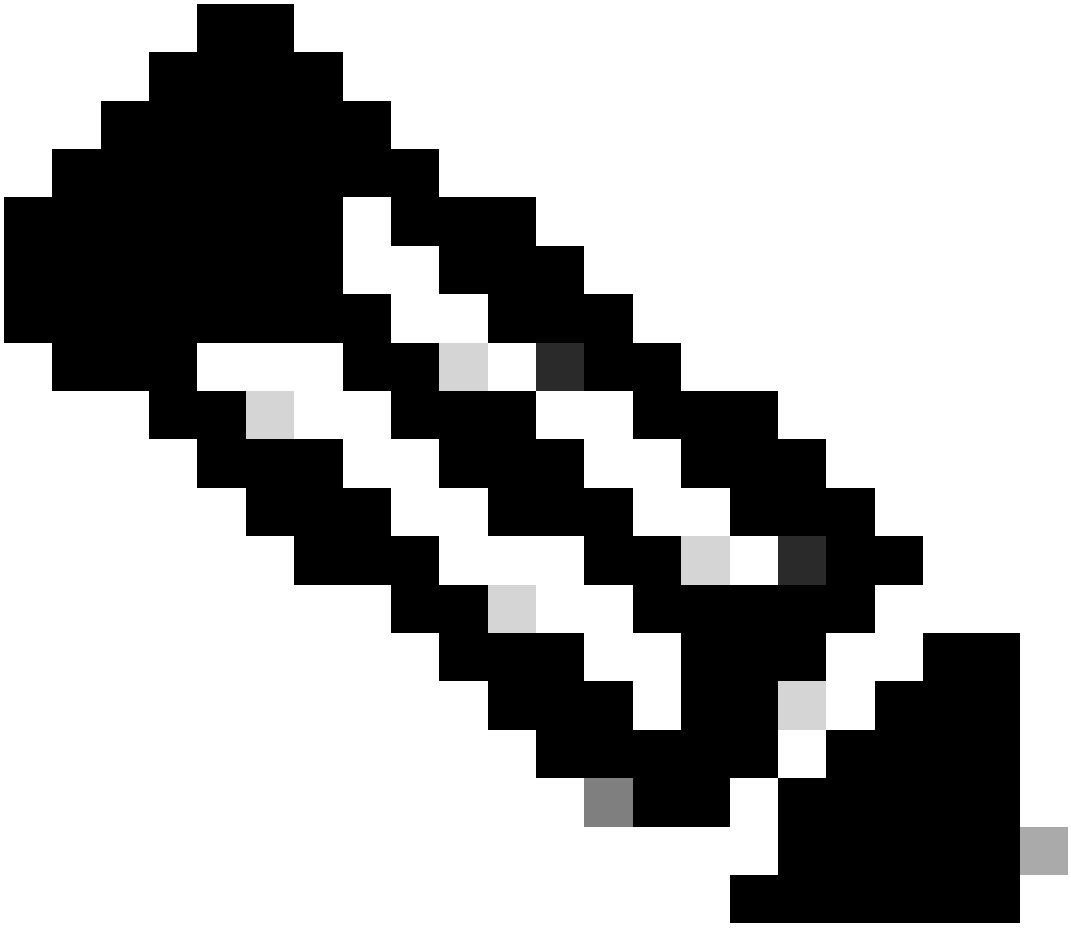
```
no allow-service netconf
no allow-service ntp
no allow-service ospf
no allow-service stun
allow-service https
no allow-service snmp
no allow-service bfd
exit
exit
interface Loopback4
tunnel-interface
encapsulation ipsec preference 150 weight 1
no border
color private5 restrict
no last-resort-circuit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

# AWSでのスループットパフォーマンスのトラブルシューティング



注：パブリッククラウド環境でパフォーマンステストを実行すると、スループットパフォーマンスに影響を与える可能性のある新しい変数が導入されます。この種のテストを実行する際に考慮すべきことがいくつかあります。

- テスト実行時のピアによる基礎となるリソースの使用
- 専用ホストを使用しない（専用ホストを使用するとクラウドコストが16倍に増加）
- クラウドは地域によって異なるため、パフォーマンスも異なる可能性があります
- 一部のインスタンスでは、機能プロファイルに関係なく数値が類似しています。これは、インスタンスのサイズごとのインターフェイスでのAWSスロットリングが原因である可能性があります
- AWSはEC2インスタンスの1秒あたりのパケット数をスロットリングするため、パケットのドロップも発生する可能性があります
- AWSはスロットリング率を開示しませんが、 ppsスロットリングによるドロップは「pps_allowance_exceeded」カウンタを介して観察できます

## 便利なCLIトラブルシューティングコマンド

スループットパフォーマンステストを実行する際に、これらのトラブルシューティングコマンドを使用して、パフォーマンス低下のボトルネックまたは原因を特定できます。

「show platform hardware qfp active statistics drop」:c8kvでドロップがあるかどうかを確認できます。重大なテールドロップや関連カウンタの増加がないことを確認する必要があります。

「show platform hardware qfp active statistics drop clear」：このコマンドはカウンタをクリアします。

「show platform hardware qfp active datapath infrastructure sw-cio」：このコマンドでは、パフォーマンス実行中に使用されているパケットプロセッサ(PP)、トラフィックマネージャ(TM)のパーセンテージに関する詳細情報が得られます。これにより、十分な処理能力があるかどうかをc8kvから判断できます。

「show platform hardware qfp active datapath util summary」：このコマンドを実行すると、c8kvがすべてのポートで送受信している入出力の全情報が得られます。

入力/出力レートを確認し、廃棄が発生していないか確認します。また、処理負荷のパーセンテージも必ず確認してください。100%に達した場合は、c8kvがその容量に達したことを意味します。

「show plat hardware qfp active infrastructure bqs interface GigabitEthernetX」：このコマンドでは、インターフェイスレベルの統計情報を、キュー番号、帯域幅、テールドロップごとにチェックできます。

「show controller」：このコマンドは、rx/txの正常なパケットと損失したパケットに関するきめの細かい情報を提供します。

このコマンドは、テールドロップが確認できないのにトラフィックジェネレータでドロップが表示されている場合に使用できます。

これは、データ使用率がすでに100 %に達していて、PPの使用率が100 %に達しているシナリオで発生する可能性があります。

rx_missed_errorsカウンタが増加し続けている場合は、これ以上のトラフィックを処理できないため、CSRがクラウドインフラストラクチャにバックプレッシャーをかけていることを意味しています。

「show platform hardware qfp active datapath infrastructure sw-hqf」 - AWSからのバックプレッシャによって発生した輻輳のチェックに使用できます。

「show plat hardware qfp active datapath infrastructure sw-nic」：複数のキュー間でトラフィックをロードバランシングする方法を決定します。17.7以降では、8つのマルチTXQがあります。

また、すべてのトラフィックを使用している特定のキューが1つあるか、ロードバランシングが適切に行われているかも判別できます。

"show controllers | in errors|exceeded|Giga":AWS側から実行されたppsスロットリングによるパケットのドロップを表示します。これは、 pps_allowance_exceededカウンタで確認できます。

# CLIの出力例

Tail dropsカウンタが増加し続ける出力例 – コマンドを複数回発行して、カウンタが増加しているかどうかを確認します。これにより、実際にテールドロップであるかどうかを確認できます。

<#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
```

次に示す出力例：リアルタイムデータを取得するには、30秒ごとにコマンドを発行します

<#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
```

次に示す出力例：入力/出力レートを確認して、廃棄がないかどうかを確認します。また、処理負荷のパーセンテージも必ず確認してください。100%に達した場合は、ノードがキャパシティに達したことを意味します。

<#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
```

インターフェイスレベルの統計情報の出力例を次に示します。

<#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
```

## RX/TX正常パケット、損失パケットの統計情報の出力例

### <#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
```

AWSからのバックプレッシャによって発生した輻輳を確認するためのサンプル出力:

<#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
```

トラフィックが複数のキュー間でどのようにロードバランスされるかを示す出力例:

```
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
```

pps_allowance_exceededカウンタを介して確認できる、AWS側から実行されたppsスロットリングによるパケットのドロップを示す出力例:

```
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
```

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。