

パケットカラーリングテクニックまたはプラットフォームカウンタを使用したパケットドロップのトラブルシューティング

内容

[はじめに](#)

[前提条件](#)

[要件](#)

[使用するコンポーネント](#)

[背景説明](#)

[トポロジ](#)

[オプション 1Flow-idを使用したERSPANの設定](#)

[ステップ 1: ESPAN接続先の設定](#)

[ステップ 2aSRCに直接接続されたトラフィックのSpanソースの作成](#)

[ステップ2b:DSTに直接接続されたトラフィックのSPANソースの作成](#)

[ステップ 3: 迅速なWireshark分析](#)

[オプション 2プラットフォームカウンタ](#)

[プラットフォームカウンタのクリア](#)

[パケットが少ない、またはゼロのパケットサイズの特定](#)

[トラフィックフローの追跡](#)

[関連情報](#)

はじめに

このドキュメントでは、パケットカラーリングテクニックを使用してネットワークフローを追跡する方法について説明します。

前提条件

要件

- ACIの基礎知識
- エンドポイントグループと契約
- Wiresharkの基礎知識

使用するコンポーネント

このドキュメントは、特定のハードウェアやソフトウェアのバージョンに限定されるものではありません。

使用デバイス：

- バージョン5.3(2)を実行しているCisco ACI
- Span宛先
- Gen2スイッチ

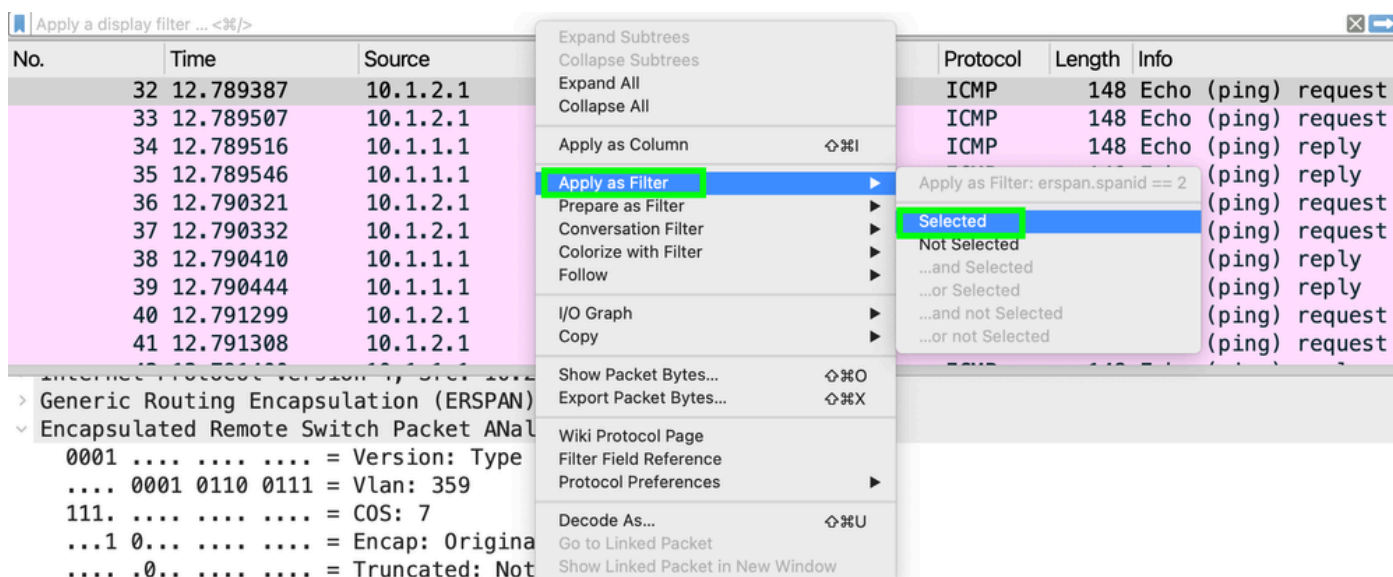
このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

背景説明

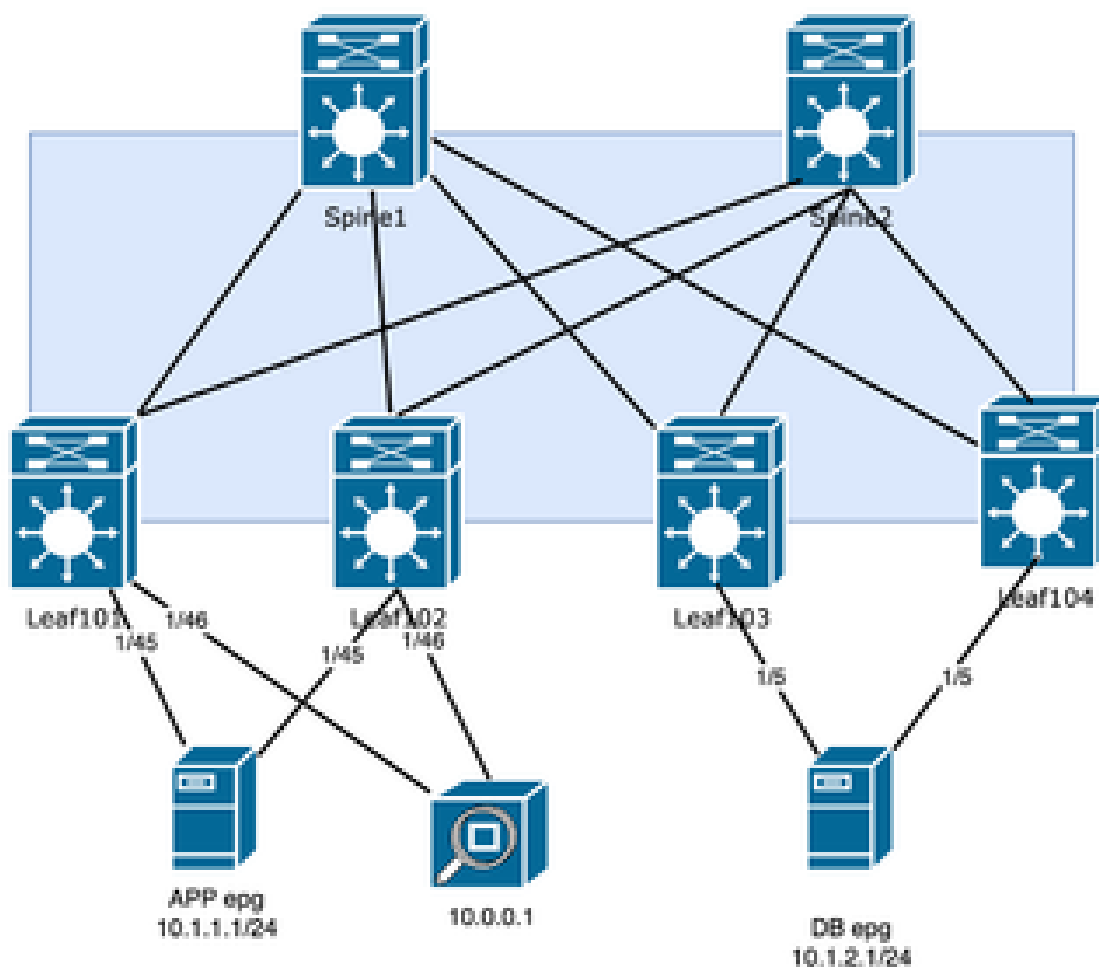
Wiresharkでのフィルタの作成方法

キャプチャを開きます。Encapsulated Remote Switch Packet(ERSP)内のフレームを使用して、SpanID行を選択し、右クリックします。

次の図に示すように、Apply as Filter > Selectedの順に選択します。



トポロジ



オプション 1Flow-idを使用したERSPANの設定

宛先サーバがすべてのトラフィックを処理できる場合、ERSPANヘッダーにはフローIDを定義するオプションが含まれます。このフローIDは、ファブリックへの着信トラフィックを識別するように設定できますが、発信トラフィックには異なるフローIDを設定できます。

ステップ 1 : ESPAN接続先の設定

1つの宛先グループのフローIDは1になります

Fabric > Access Policies > Policies > Troubleshooting > SPAN > SPAN Destination Groupsの下で

Create SPAN Destination Group



Name: All-dst-jr-flowid

Description: optional

Destination Type: **EPG** Access Interface

Destination EPG: jr ALL monitor

Tenant Application Profile EPG

SPAN Version: **Version 1** Version 2

Enforce SPAN Version: ☐

Destination IP: 10.0.0.1

Source IP/Prefix: 10.255.0.0/16

Flow ID: 1

TTL: 64

MTU: 8000

DSCP: Unspecified

Cancel

Submit

2番目の宛先グループで、flow-idを2に設定します。

Create SPAN Destination Group

Name:

Description:

Destination Type: EPG Access Interface

Destination EPG:
Tenant Application Profile EPG

SPAN Version: Version 1 Version 2

Enforce SPAN Version: ☐

Destination IP:

Source IP/Prefix:

Flow ID:

TTL:

MTU:

DSCP:

Cancel

Submit

ステップ 2aSRCに直接接続されたトラフィックのSPANソースの作成

Fabric > Access Policies > Policies > Troubleshooting > SPAN > SPAN Source Groupsの下で

Create SPAN Source Group

Name:

Description:

Admin State: Disabled Enabled

Filter Group:

Destination Group:

Create Sources

Name	Direction	Source EPG	Source Paths
------	-----------	------------	--------------

パスとEPGを追加して、トラフィックをさらにフィルタリングします。このラボの例は、Tenant jr Application Profile ALLおよびEPGアプリケーションです。

Create SPAN Source



Name:

Description:

Direction: Both Incoming Outgoing

Filter Group:

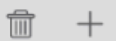
Span Drop Packets: ☐

Type: None EPG Routed Outside

Source EPG:

Tenant Application Profile EPG

Add Source Access Paths



Source Access Path

Pod-1/Node-101/VPC-ESX-169

Pod-1/Node-102/VPC-ESX-169

Cancel

OK

ステップ2b:DSTに直接接続されたトラフィックのSPANソースの作成

Fabric > Access Policies > Policies > Troubleshooting > SPAN > SPAN Source Groupsの下で

Create SPAN Source

Description: optional

Direction: **Both** Incoming Outgoing

Filter Group: select an option

Span Drop Packets: ☐

Type: None **EPG** Routed Outside

Source EPG: jr Tenant ALL Application Profile db EPG

Add Source Access Paths

Source Access Path
Pod-1/Node-103/eth1/6

パスだけでなくEPG DBも追加して、トラフィックのフィルタリングを強化します。

Create SPAN Source Group

Name: Src-epg-2

Description: optional

Admin State: Disabled **Enabled**

Filter Group: select an option

Destination Group: All-dst-jr-flowid2

Create Sources

Name	Direction	Source EPG	Source Paths
------	-----------	------------	--------------

ステップ 3 : 迅速なWireshark分析

この例では、ICMP要求パケットの数がICMP応答パケットの数と一致することを確認し、ACIアプリケーション内でパケットドロップが発生していないことを確認します。

Wiresharkでキャプチャを開き、SRCおよびDST IPとともに設定されたSPAN ID /Flow-IDを使用してフィルタを作成します。

```
<#root>
```

```
(erspan.spanid ==
```

```
and
```

```
) && (ip.src==
```

```
and ip.dst ==
```

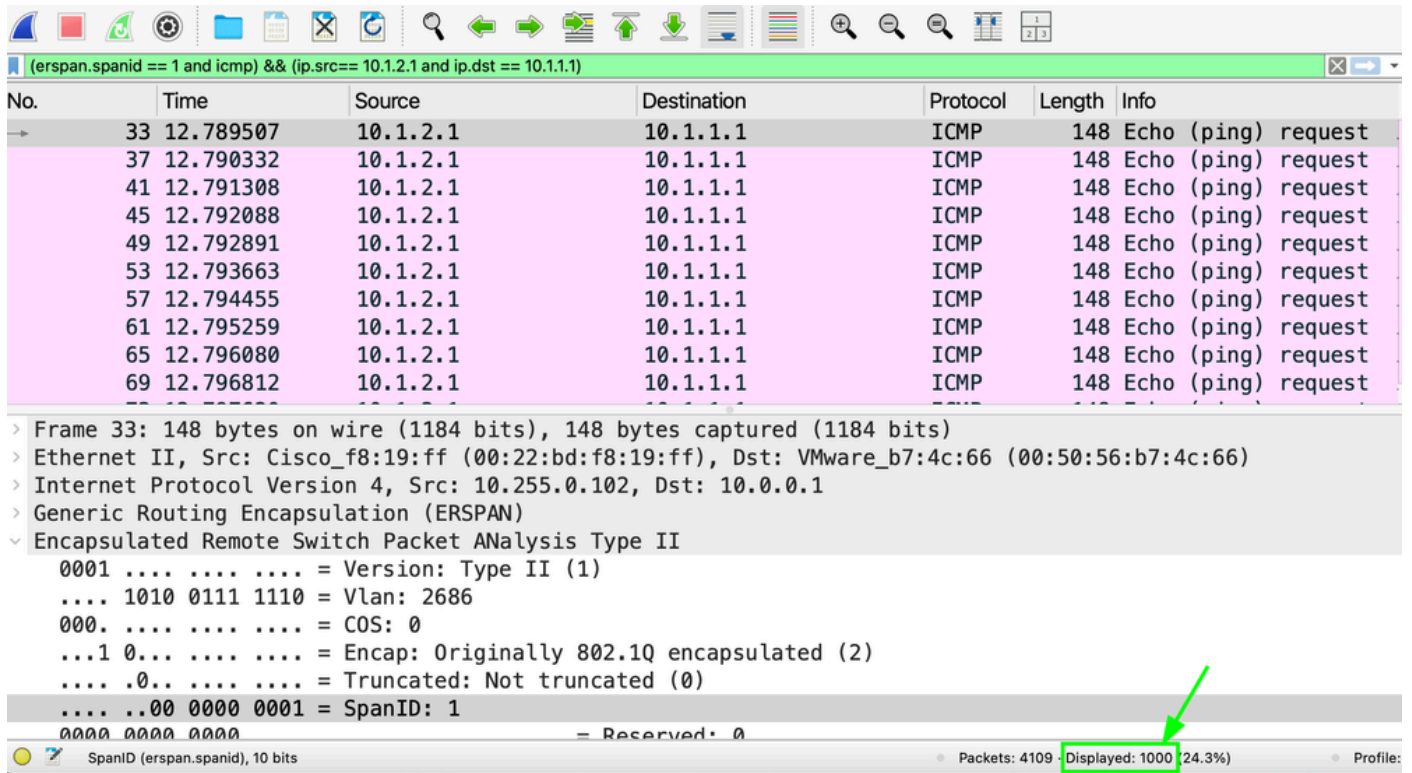
```
)
```

ラボでテストしたフローに使用するフィルタ：

```
<#root>
```

```
(erspan.spanid == 1 and icmp) && (ip.src== 10.1.2.1 and ip.dst == 10.1.1.1)
```

表示されたパケットが送信されたものと同じ量であることを確認します。



(erspan.spanid == 1 and icmp) && (ip.src == 10.1.2.1 and ip.dst == 10.1.1.1)

No.	Time	Source	Destination	Protocol	Length	Info
33	12.789507	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request
37	12.790332	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request
41	12.791308	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request
45	12.792088	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request
49	12.792891	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request
53	12.793663	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request
57	12.794455	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request
61	12.795259	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request
65	12.796080	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request
69	12.796812	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) request

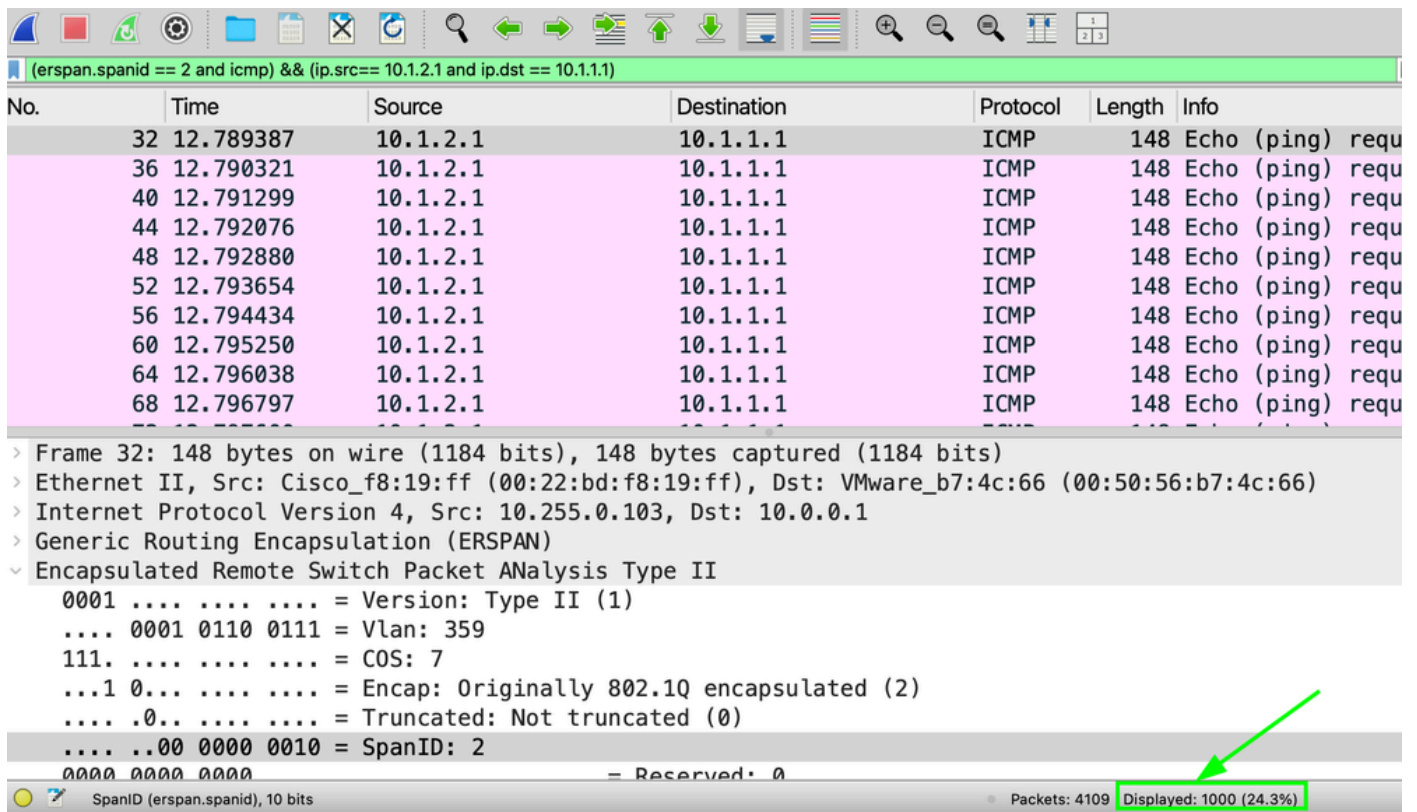
> Frame 33: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)
 > Ethernet II, Src: Cisco_f8:19:ff (00:22:bd:f8:19:ff), Dst: VMware_b7:4c:66 (00:50:56:b7:4c:66)
 > Internet Protocol Version 4, Src: 10.255.0.102, Dst: 10.0.0.1
 > Generic Routing Encapsulation (ERSPAN)
 > Encapsulated Remote Switch Packet Analysis Type II
 0001 = Version: Type II (1)
 1010 0111 1110 = Vlan: 2686
 000. = COS: 0
 ...1 0... = Encap: Originally 802.1Q encapsulated (2)
 0.. = Truncated: Not truncated (0)
 00 0000 0001 = SpanID: 1
 0000 0000 0000 = Reserved: 0

SpanID (erspan.spanid), 10 bits Packets: 4109 Displayed: 1000 (24.3%) Profile:

次のSPAN IDの値は同じでなければなりません。同じでない場合、パケットはファブリック内で廃棄されています。

[Filter] :

(erspan.spanid == 2 and icmp) && (ip.src == 10.1.2.1 and ip.dst == 10.1.1.1)



The image shows a Wireshark packet capture interface. The top toolbar contains various icons for file operations, packet navigation, and analysis. Below the toolbar is a green filter bar with the expression: `(erspan.spanid == 2 and icmp) && (ip.src == 10.1.2.1 and ip.dst == 10.1.1.1)`.

The main packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
32	12.789387	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ
36	12.790321	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ
40	12.791299	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ
44	12.792076	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ
48	12.792880	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ
52	12.793654	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ
56	12.794434	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ
60	12.795250	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ
64	12.796038	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ
68	12.796797	10.1.2.1	10.1.1.1	ICMP	148	Echo (ping) requ

Below the packet list, the details pane shows the structure of the selected packet (No. 32):

- > Frame 32: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)
- > Ethernet II, Src: Cisco_f8:19:ff (00:22:bd:f8:19:ff), Dst: VMware_b7:4c:66 (00:50:56:b7:4c:66)
- > Internet Protocol Version 4, Src: 10.255.0.103, Dst: 10.0.0.1
- > Generic Routing Encapsulation (ERSPAN)
- > Encapsulated Remote Switch Packet ANalysis Type II
 - 0001 = Version: Type II (1)
 - 0001 0110 0111 = Vlan: 359
 - 111. = COS: 7
 - ...1 0... = Encap: Originally 802.1Q encapsulated (2)
 -0.. = Truncated: Not truncated (0)
 -00 0000 0010 = SpanID: 2
 - 0000 0000 0000 = Reserved: 0

The bottom status bar shows: `SpanID (erspan.spanid), 10 bits` and `Packets: 4109 Displayed: 1000 (24.3%)`. A green arrow points to the `Displayed: 1000 (24.3%)` text.

オプション 2プラットフォームカウンタ

この方法は、Nexusがパケットサイズの異なる個々のインターフェイスのパフォーマンスを追跡していることを利用しますが、ゼロではないにしても、少なくともキューのトラフィック量が少なくなることが必要です。

プラットフォームカウンタのクリア

個々のスイッチに移動し、デバイスに接続されている個々のインターフェイスをクリアします。

```
<#root>
```

```
Switch#
```

```
vsh_lc -c "clear platform internal counters port
```

```
"
```

```
<#root>
```

```
LEAF3#
```

```
vsh_lc -c "clear platform internal counters port 6"
```

```
LEAF1#
```

```
vsh_lc -c "clear platform internal counters port 45"
```

```
LEAF2#
```

```
vsh_lc -c "clear platform internal counters port 45"
```

パケットが少ない、またはゼロのパケットサイズの特定

RXとTXの両方のすべてのリーフにカウンタがない可能性のあるパケットサイズを見つけます。

```
<#root>
```

```
vsh_lc -c 'show platform internal counters port
```

```
' | grep X_PKT
```

次の例では、パケットサイズが512より大きく1024より小さい場合です。

```
<#root>
```

```
LEAF101#
```

```
vsh_lc -c "show platform internal counters port 45 " | grep X_PKT
```

RX_PKTOK	1187
RX_PKTTOTAL	1187
RX_PKT_LT64	0
RX_PKT_64	0
RX_PKT_65	1179
RX_PKT_128	8
RX_PKT_256	0
RX_PKT_512	0 <<
RX_PKT_1024	0
RX_PKT_1519	0
RX_PKT_2048	0
RX_PKT_4096	7
RX_PKT_8192	43
RX_PKT_GT9216	0
TX_PKTOK	3865

TX_PKTTOTAL	3865
TX_PKT_LT64	0
TX_PKT_64	0
TX_PKT_65	3842
TX_PKT_128	17
TX_PKT_256	6
TX_PKT_512	0 <<
TX_PKT_1024	10
TX_PKT_1519	3
TX_PKT_2048	662
TX_PKT_4096	0
TX_PKT_8192	0
TX_PKT_GT9216	0

このステップは、パケットが転送されるリンクで実行する必要があります。

トラフィックフローの追跡

サーバ10.1.2.1から、1000個のパケットが520のパケットサイズで送信されます。

トラフィックがRXで開始されるリーフ103インターフェイス1/6で確認します。

```
<#root>
```

```
MXS2-LF103#
```

```
vsh_lc -c "show platform internal counters port 6 " | grep X_PKT_512
```

RX_PKT_512	1000
TX_PKT_512	647

1000パケットのRXが送信されましたが、応答として送信されたのは647パケットだけです。

次に、他のサーバの発信インターフェイスを確認します。

Leaf102の場合：

```
<#root>
```

```
MXS2-LF102#
```

```
vsh_lc -c "show platform internal counters port 45 " | grep X_PKT_512
```

RX_PKT_512	0
TX_PKT_512	1000

ファブリックは要求をドロップしませんでした。

リーフ101の場合、RXパケットは647で、ACIによるTXと同じ量のパケットです。

<#root>

MXS2-LF101#

```
vsh_lc -c "show platform internal counters port 45 " | grep X_PKT_512
```

RX_PKT_512	647
TX_PKT_512	0

関連情報

[ACI Intra-Fabric Forwardingのトラブルシューティング：断続的なドロップ](#)

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。