Troubleshoot Packet Drops with Packet Coloring Techniques or Platform Counters

Contents

Introduction

This document describes how to track a network flow using packet coloring techniques.

Prerequisites

Requirements

- Basic knowledge of ACI
- Endpoint Groups and contract
- Wireshark basic knowledge

Components Used

This document is not restricted to specific hardware and software versions.

Devices used:

- Cisco ACI running version 5.3(2)
- Span destination
- Gen2 switches

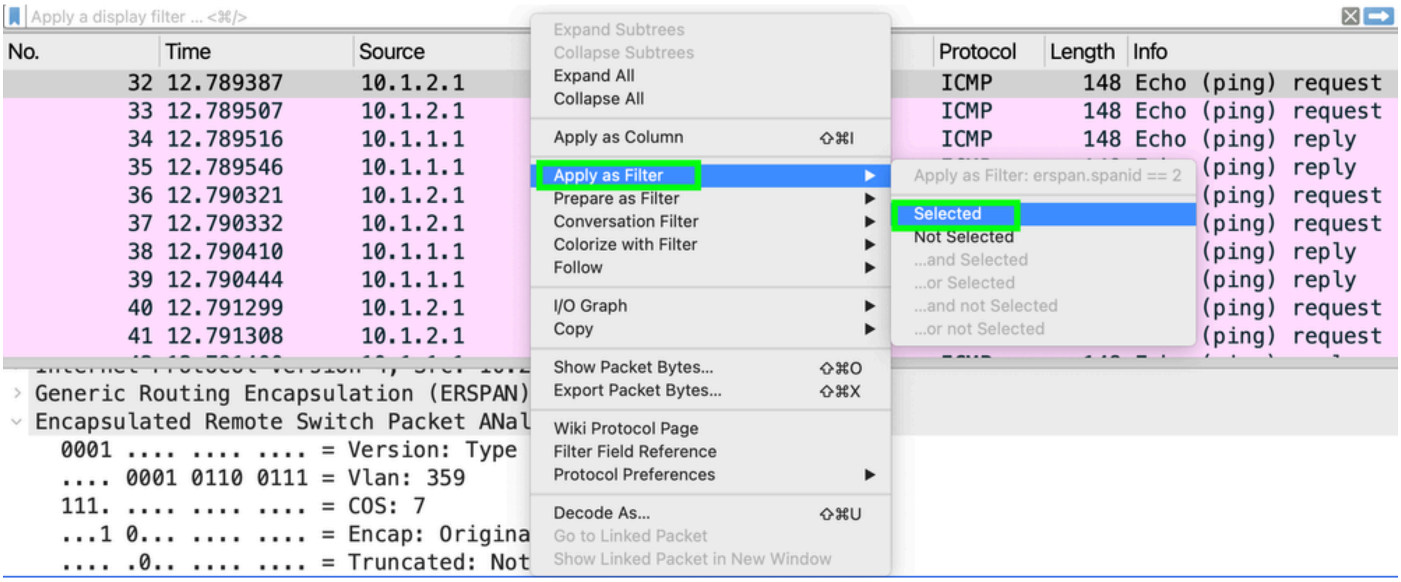
The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

How to create filters in Wireshark.

Open the **capture**. Using a frame inside the Encapsulated Remote Switch Packet, select the **SpanID line** and right-click.

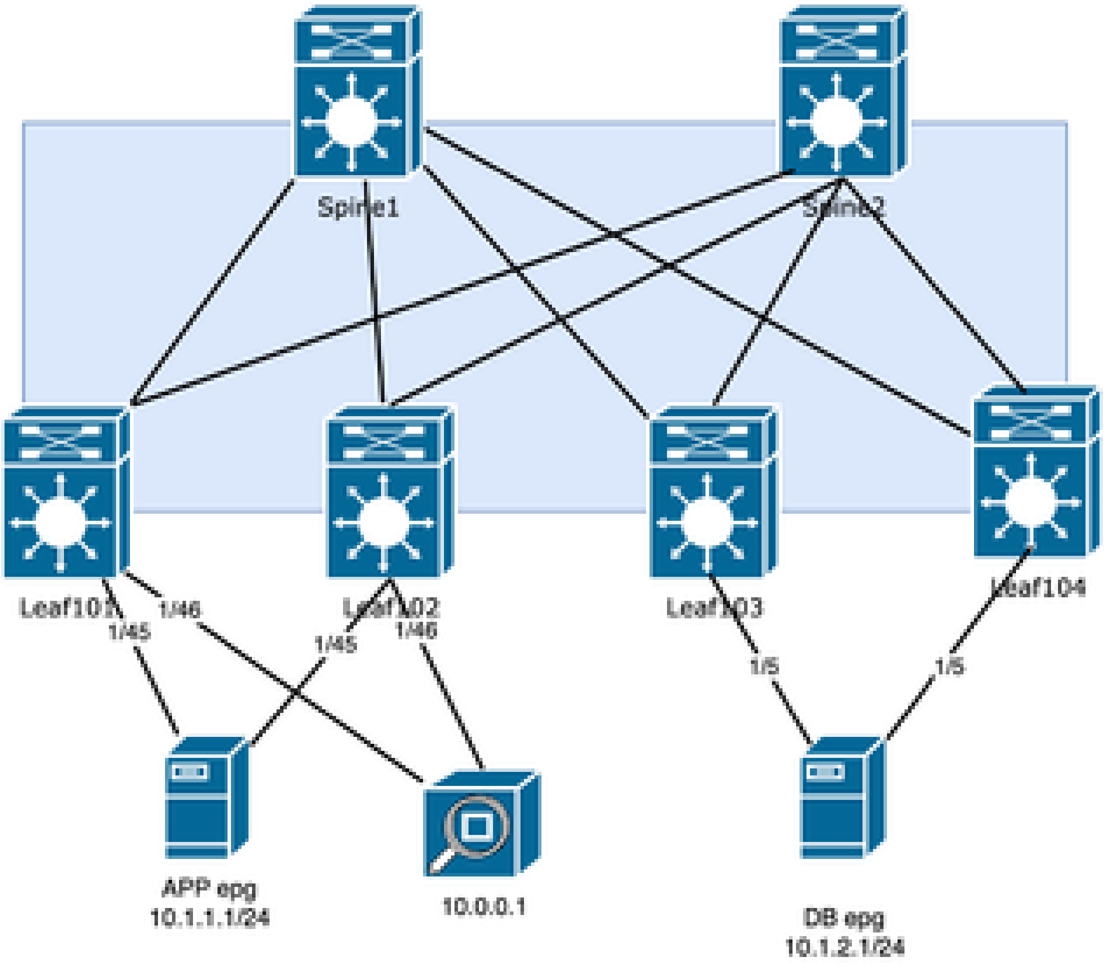
Select **Apply as Filter** > **Selected** as the picture shows:



```
No.          Time             Source                                   Protocol   Length  Info
          32 12.789387        10.1.2.1                                 ICMP          148  Echo (ping) request
          33 12.789507        10.1.2.1                                 ICMP          148  Echo (ping) request
          34 12.789516        10.1.1.1                                 ICMP          148  Echo (ping) reply
          35 12.789546        10.1.1.1                                                          (ping) reply
          36 12.790321        10.1.2.1                                                          (ping) request
          37 12.790332        10.1.2.1                                                          (ping) request
          38 12.790410        10.1.1.1                                                          (ping) reply
          39 12.790444        10.1.1.1                                                          (ping) reply
          40 12.791299        10.1.2.1                                                          (ping) request
          41 12.791308        10.1.2.1                                                          (ping) request

> Generic Routing Encapsulation (ERSPAN)
v Encapsulated Remote Switch Packet ANal
    0001 .... .... .... = Version: Type
    .... 0001 0110 0111 = Vlan: 359
    111. .... .... .... = COS: 7
    ...1 0... .... .... = Encap: Origina
    .... .0.. .... .... = Truncated: Not
```

Topology



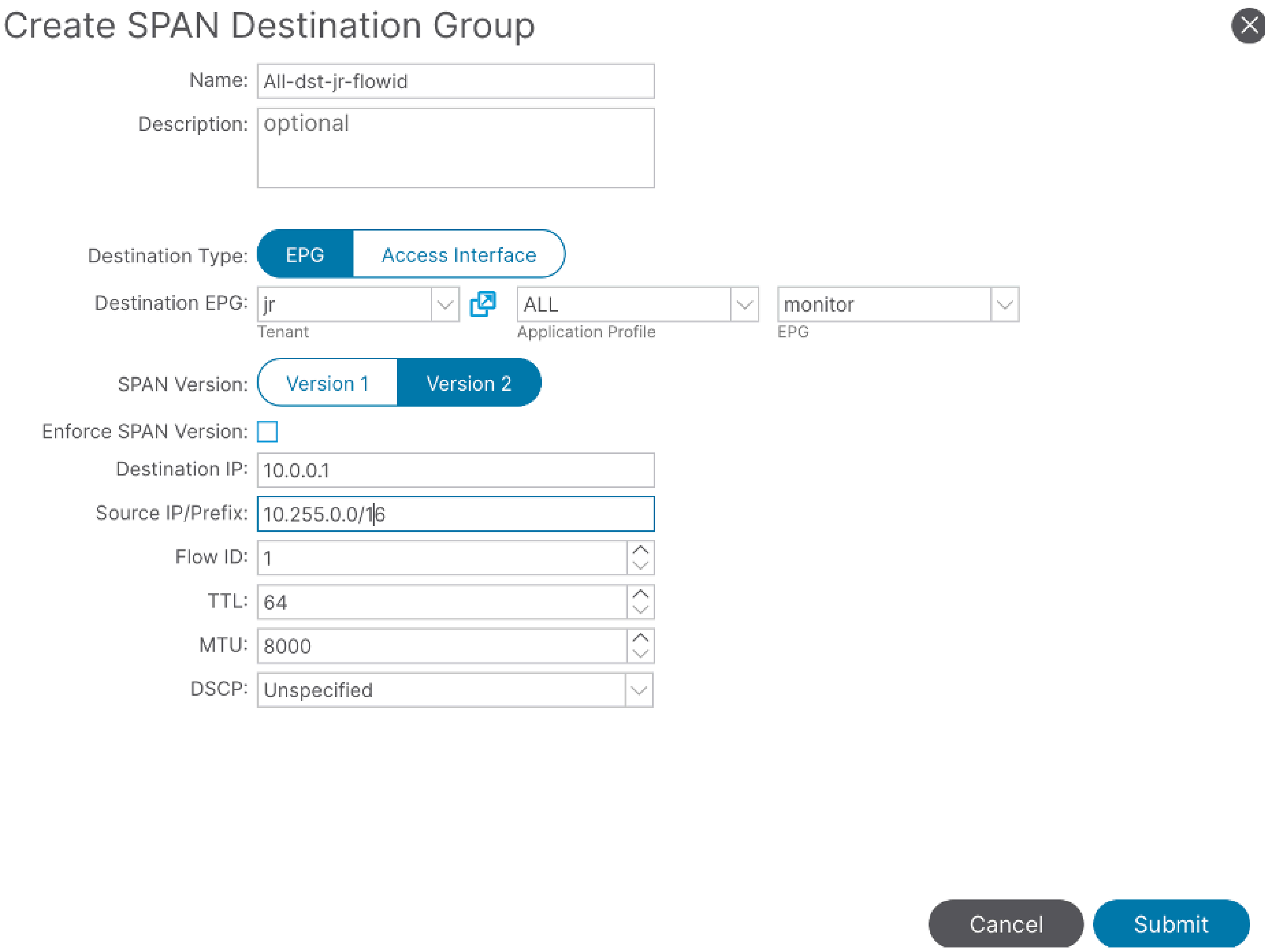
Option 1. ERSPAN Setup with Flow-id

If a destination server is capable of handling all the traffic, the ERSPAN header includes an option to define a Flow ID. This Flow ID can be configured to identify incoming traffic to the fabric, while a different Flow ID can be set up for outgoing traffic.

Step 1. ESPAN Destination Setup

One destination group is going to have the flow-id of 1

Under Fabric > Access Policies > Policies > Troubleshooting > SPAN > SPAN Destination Groups



Create SPAN Destination Group

Name: All-dst-jr-flowid

Description: optional

Destination Type: EPG | Access Interface

Destination EPG: jr (Tenant) | ALL (Application Profile) | monitor (EPG)

SPAN Version: Version 1 | Version 2

Enforce SPAN Version: □

Destination IP: 10.0.0.1

Source IP/Prefix: 10.255.0.0/16

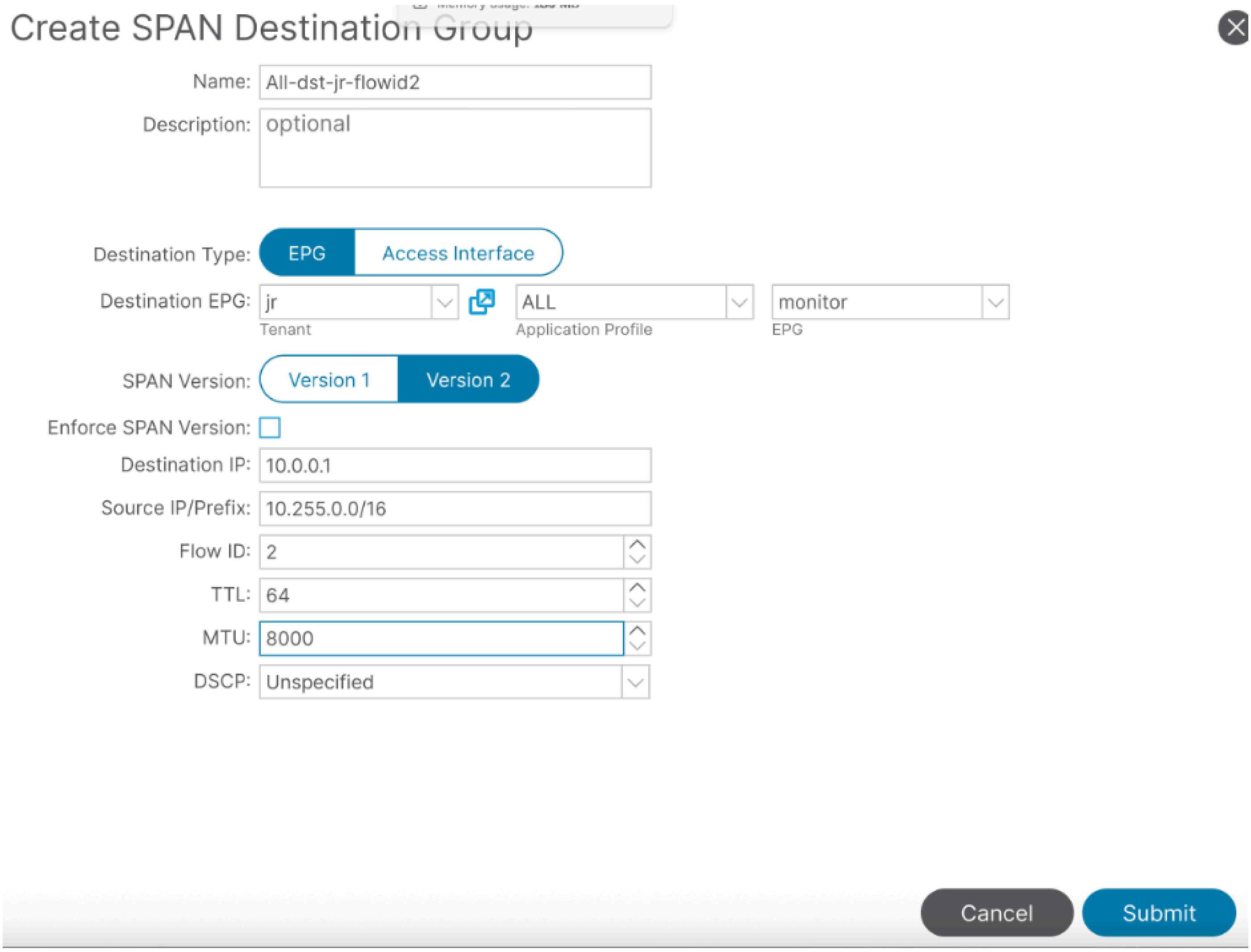
Flow ID: 1

TTL: 64

MTU: 8000

DSCP: Unspecified

On the second destination group, configure flow-id of 2:



Create SPAN Destination Group

Name: All-dst-jr-flowid2

Description: optional

Destination Type: EPG | Access Interface

Destination EPG: jr (Tenant) | ALL (Application Profile) | monitor (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: 2

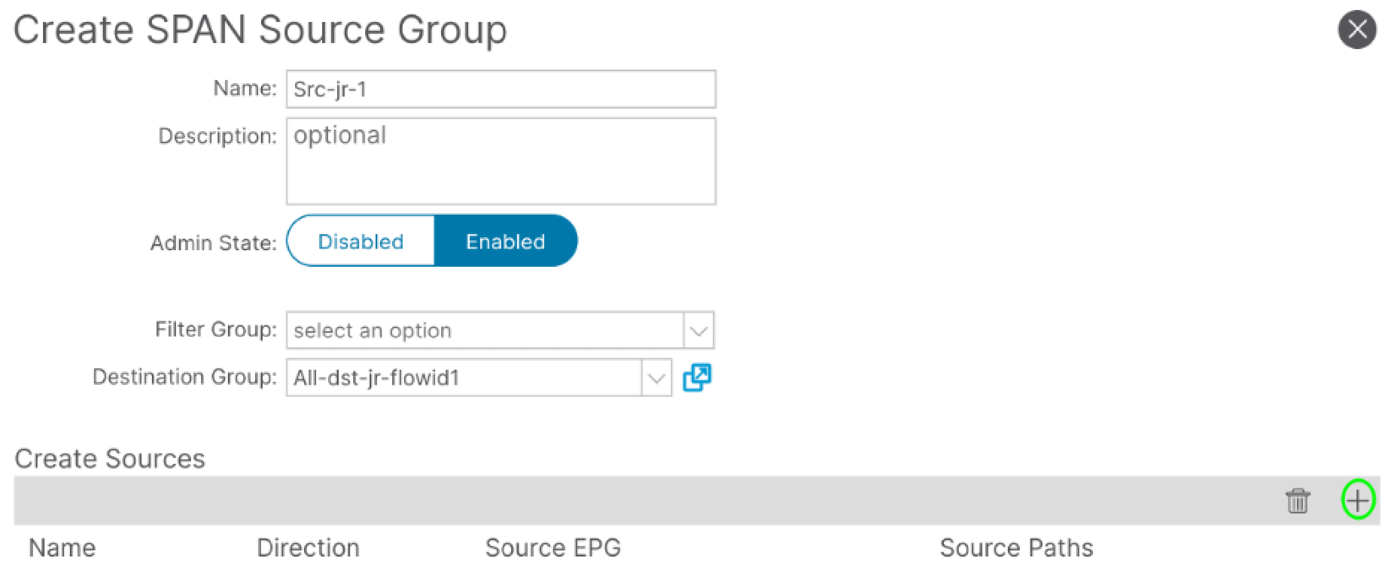
TTL: 64

MTU: 8000

DSCP: Unspecified

Step 2a. Create Span Source for the Traffic Directly Connected to the SRC

Under Fabric > Access Policies > Policies > Troubleshooting > SPAN > SPAN Source Groups



Create SPAN Source Group

Name: Src-jr-1

Description: optional

Admin State: Disabled | Enabled

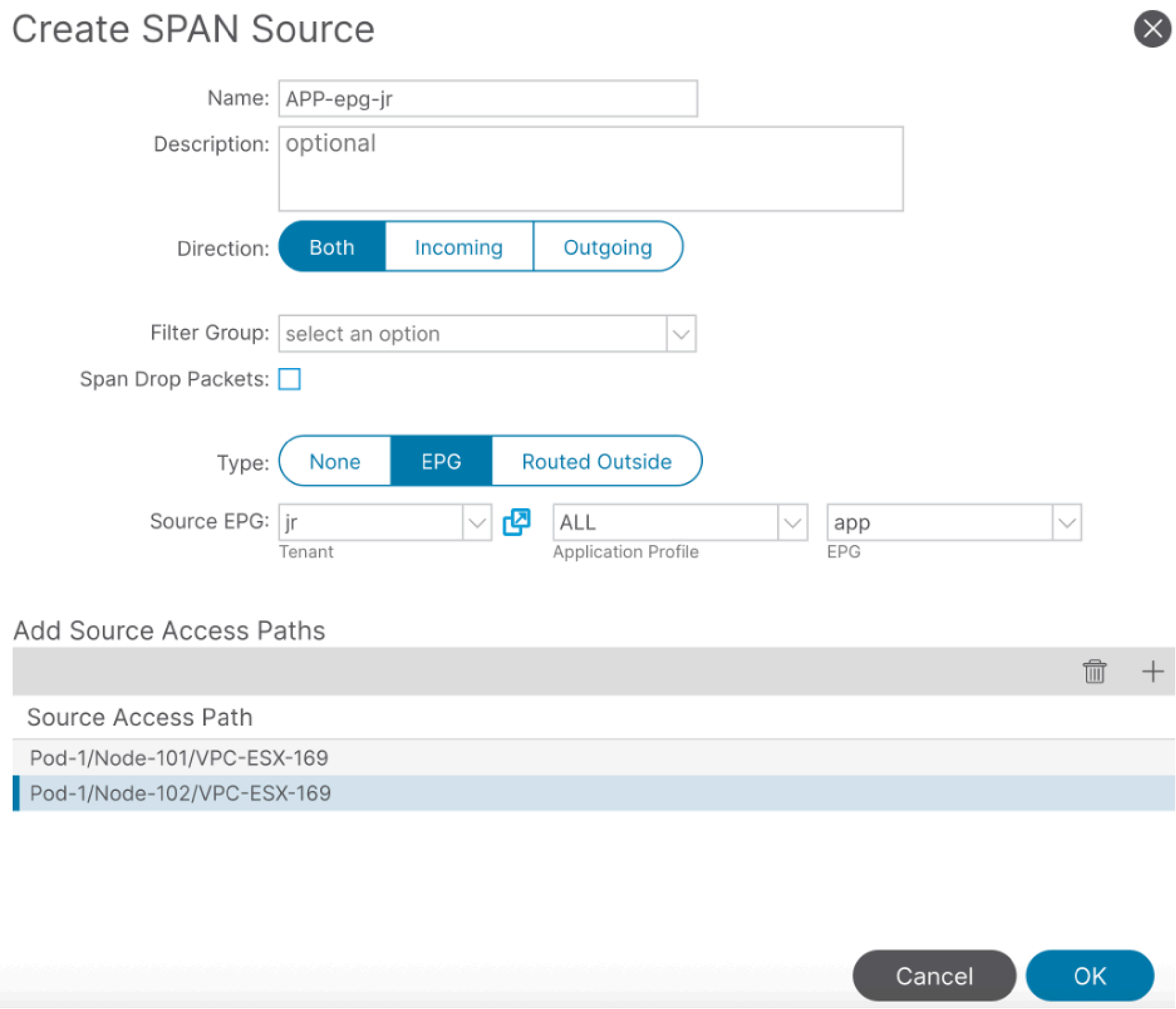
Filter Group: select an option

Destination Group: All-dst-jr-flowid1

Create Sources

Name	Direction	Source EPG	Source Paths

Filter the traffic more by adding the Path and the EPG. The lab example is Tenant jr Application Profile ALL and EPG app.



Create SPAN Source

Name: APP-epg-jr

Description: optional

Direction: Both | Incoming | Outgoing

Filter Group: select an option

Span Drop Packets: □

Type: None | EPG | Routed Outside

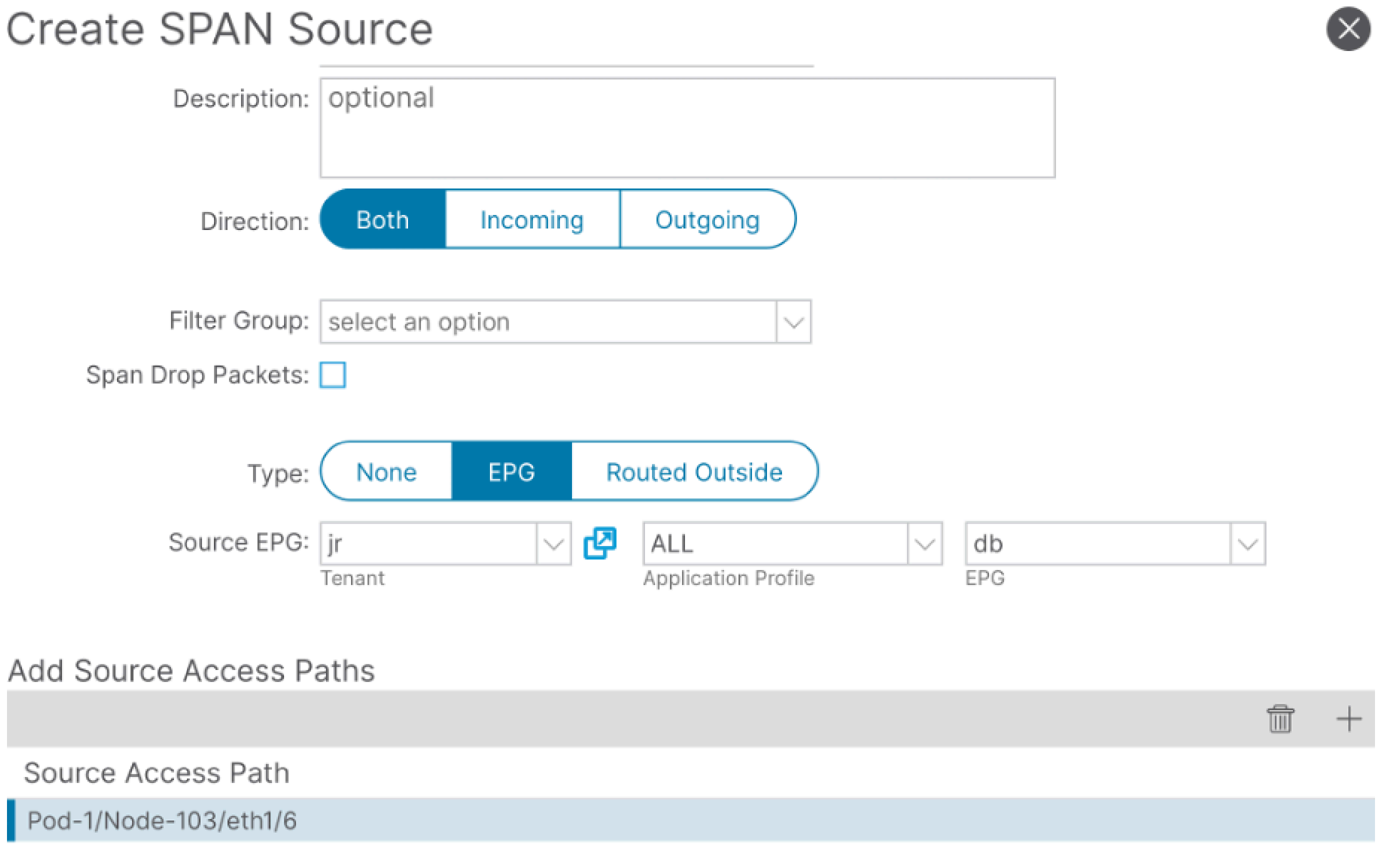
Source EPG: jr (Tenant) | ALL (Application Profile) | app (EPG)

Add Source Access Paths

Source Access Path
Pod-1/Node-101/VPC-ESX-169
Pod-1/Node-102/VPC-ESX-169

Step 2b. Create Span Source for the Traffic Directly Connected to the DST

Under Fabric > Access Policies > Policies > Troubleshooting > SPAN > SPAN Source Groups



Create SPAN Source

Description: optional

Direction: [✓ Both] Incoming Outgoing

Filter Group: select an option

Span Drop Packets: □

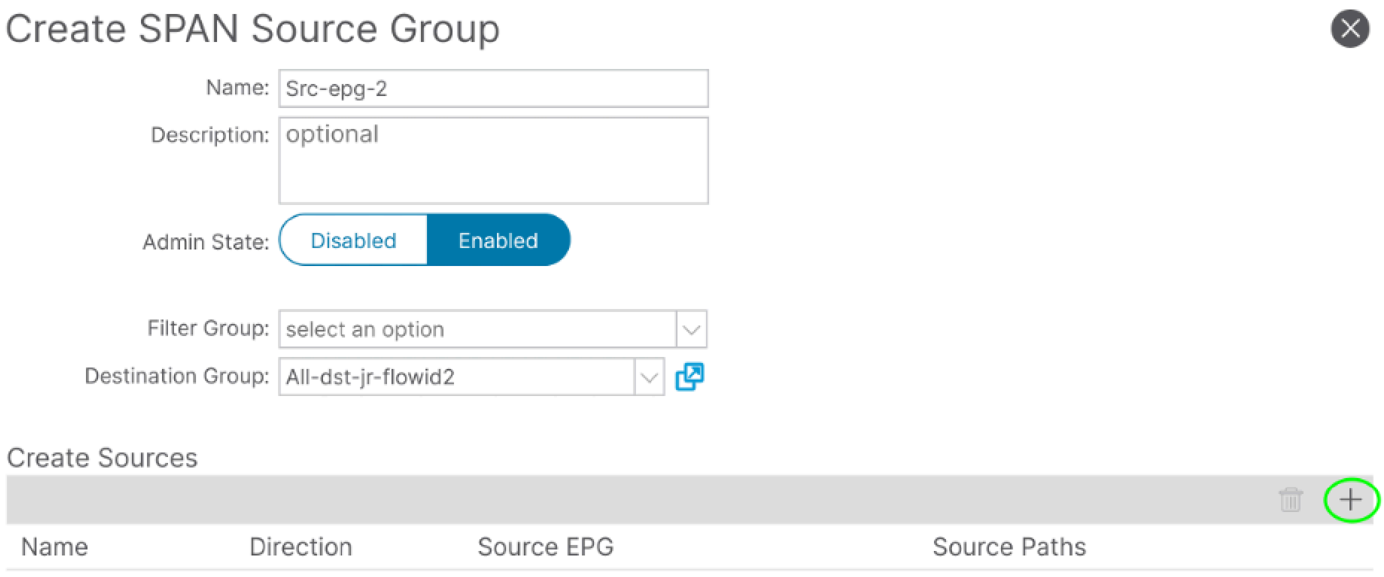
Type: None [✓ EPG] Routed Outside

Source EPG: jr (Tenant) ALL (Application Profile) db (EPG)

Add Source Access Paths

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

Filter the traffic more by adding not only the Path but also the EPG DB:



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

Step 3. Quick Wireshark Analysis

In this example, you are verifying that the number of ICMP request packets matches the number of ICMP response packets, ensuring that there are no packet drops within the ACI fabric.

Open the **capture** on wireshark to create the filter using the SPAN ID /Flow-ID configured along with SRC and DST IP:

<#root>

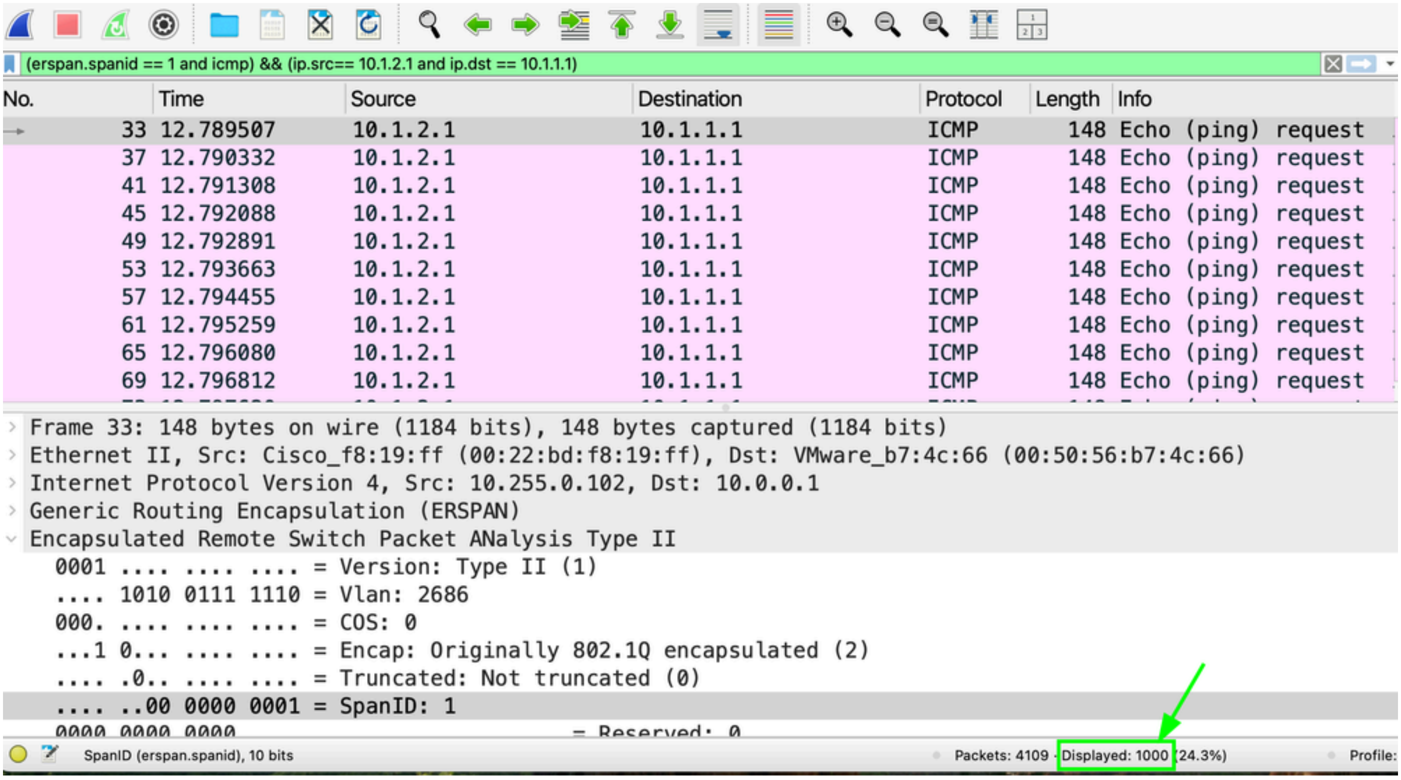
```
(erspan.spanid == <id selected on ERSPAN DST group> and <protocol>) && (ip.src== <ip src> and ip.dst ==
```

Filter Used for the Lab tested flow:

<#root>

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

Verify the Displayed packet is the same amount as sent:



```
(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
```

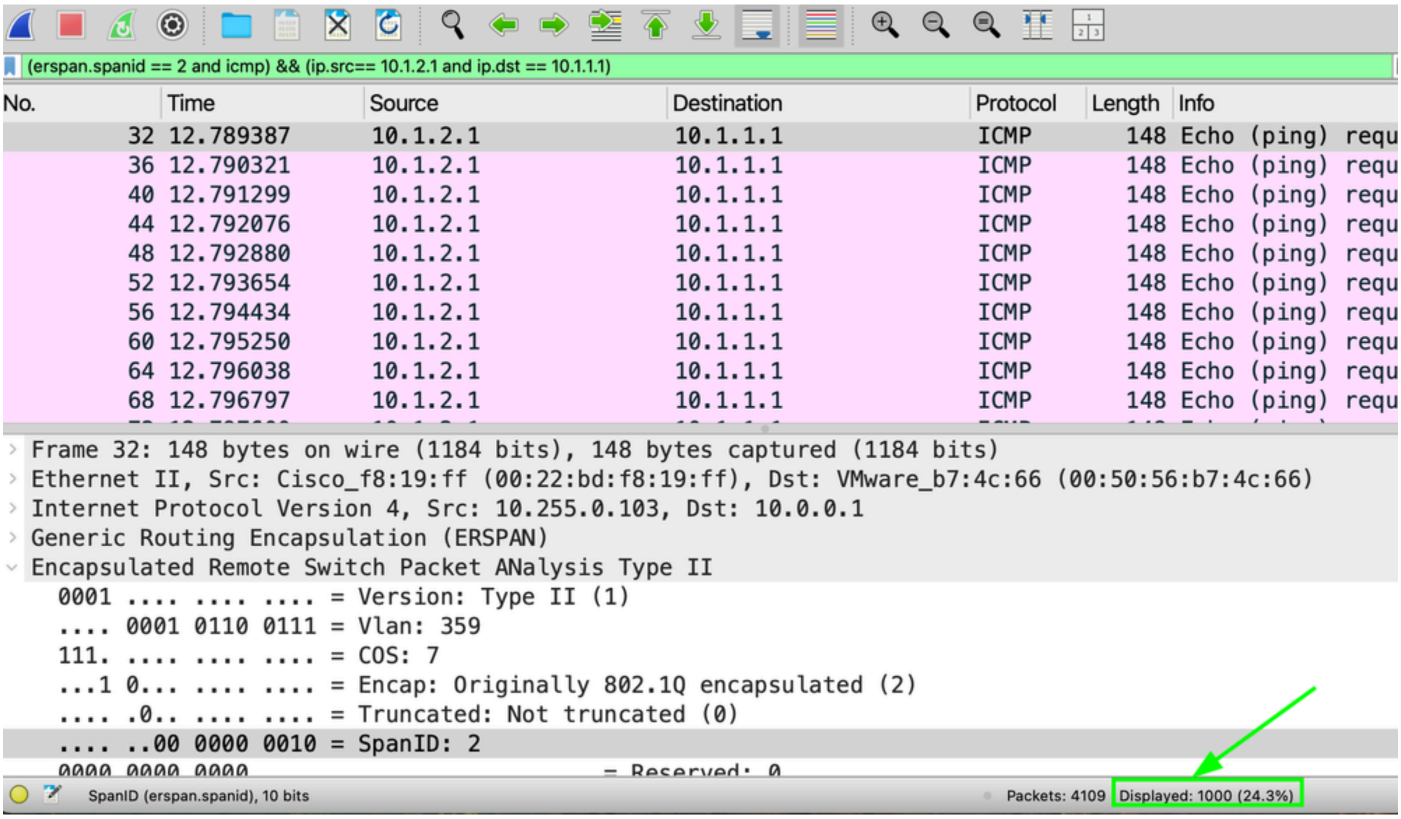
```
> Frame 33: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)
> Ethernet II, Src: Cisco_f8:19:ff (00:22:bd:f8:19:ff), Dst: VMware_b7:4c:66 (00:50:56:b7:4c:66)
> Internet Protocol Version 4, Src: 10.255.0.102, Dst: 10.0.0.1
> Generic Routing Encapsulation (ERSPAN)
v Encapsulated Remote 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:
```

The next SPAN ID must have the same amount; if it does not, the packet was dropped inside the fabric.

Filter:

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



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

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

```
> 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)
v 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
```

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

Option 2. Platform Counters

This method takes advantage that Nexus is tracking the performance of individual interfaces with different Packet sizes, but the method does require that at least a queue has a low amount of traffic, if not zero.

Clear Platform Counters

Go into the individual switch and clear the individual interface that connects to the devices.

<#root>

```
Switch#

vsh_lc -c “clear platform internal counters port <port id>”
```

<#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”
```

Identify a Packet Size with Low or Zero Packets

Find a packet size that has possibly no counters in all Leafs for both RX and TX:

<#root>

```
vsh_lc -c ‘show platform internal counters port <id>’ | grep X_PKT
```

In the next example, packet size greater than 512 and lower than 1024:

<#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
```

The step needs to be performed in the link where the packets are being forwarded to it.

Track Traffic Flow

From server 10.1.2.1, 1000 packets are sent with a packet size of 520.

Verify on Leaf 103 interface 1/6, where traffic is initiated on RX:

<#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 packet RX, but only 647 were sent as a reply.

The next step is to check the other servers' outgoing interfaces:

For Leaf102:

<#root>

```
MXS2-LF102#

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

                 RX_PKT_512           0
                 TX_PKT_512        1000
```

The fabric did not drop the Request.

For Leaf 101, RX packets 647 and is the same amount of packets TX by ACI.

<#root>

```
MXS2-LF101#

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

                 RX_PKT_512        647
                 TX_PKT_512          0
```

Related Information

[Troubleshoot ACI Intra-Fabric Forwarding - Intermittent Drops]