単一回線コマンドを使用したファブリックのトラブルシューティングと確認

内容

はじめに

このドキュメントでは、ファブリックの全体的な問題を検出するさまざまな方法について説明します。

前提条件

要件

- ACIに関する知識があることが推奨されます
- bashの基礎知識

使用するコンポーネント

このドキュメントの内容は、特定のソフトウェアやハード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4.2(3)を実行しているCisco ACI

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

情報の取得に使用されるツール

- sedによる代用： sed "s/<oldword >/<new work>/g" gは、複数回実行されることを定義します。
- sedは最初から最後まで行を取り込む: sed "/<beginning/,/end/p>"。-n (noprint)オプションと/p printフラグを組み合わせて、grepの機能を複製します。
- Sedは';'を使って複数回使うことができる。例： would : sed "s/<oldword >/<new work>/g; s/<oldword2 >/<new work2>/g"
- awk -F '<field_separator>' '{print $2}'：この特定の例では、定義されたFIELD_SEPARATORで行を分割し、2番目の区切りチャンクを印刷します。どちらの構文も全く同じことを行います。
- awk '{ print $1, $2 }'は、各入力レコードの最初の2つのフィールドをスペースで区切って印刷します。
- sort | uniq繰り返される行を報告します。-c（ハイフン）を使用すると、行の先頭に出現回数が付きます。
- sort -nrk <column>行を最も高くソートします。-nは数値による並べ替え、-kはキーを表します。これにより、列を変更できます。また、最小値を定義する場合は、-rを削除できます
- python -m json.tool:JSONをかわいらしい形式で表示します。

すべての1行のコマンドのリスト

ファブリック内のリーフのノードIDのみを取得します。

1. リスト内：

```
APIC#acidiag fnvread | grep leaf | awk '{print $1}'
```

2. カンマを区切り文字として使用した行：

```
APIC#acidiag fnvread | grep leaf | awk '{print $1}' | sed -z 's/\n/,/g;s/,$/\n/'
```

インターフェイスのリセットがあるかどうかを確認します。

この情報は、リセット回数が最も多いインターフェイスでソートされます。

```
APIC#moquery -c ethpm.PhysIf | egrep "dn|lastLinkStChg|resetCtr" | tr -d "\n" | sed "s/dn/\ndn/g;s/last
```

より遅いオプション

```
APIC#for leaf in `acidiag fnvread | grep leaf | awk '{print $1}'`; do echo; echo " -> leaf ID: $leaf  "
```

最も高い定格のインターフェイスを確認してください。

最も多くのトラフィックが受信されている場所を見つけるには、次の手順を実行します。

特定の値(b)を超える出力スループットですべてのインターフェイスのファブリックを照会します。 mの値は、bがgb、mb、またはkbのいずれであるかを定義します。

gbでフィルタするには、mを125000000に設定します。mbでフィルタするには、mを125000に設定します。kbでフィルタするには、mを125に設定します。

```
APIC#bash
APIC#b=1; m=125000;b=$((b*m)); printf "%-65s %25s\n", "Node/Interface" "Bits/Second"; icurl 'http://loc
```

```
APIC#for leaf in `acidiag fnvread | grep leaf | awk '{print $1}'`; do echo; echo " -> leaf ID: $leaf  "
```

すべてのSTPトポロジ変更を検出します。

1. このコマンドは各リーフに適用され、最近トポロジが変更されたかどうか、およびどのイン

ターフェイスが変更されたかを確認します。

```
APIC#for node in `acidiag fnvread | grep leaf | awk '{print $1}'`; do echo; echo "node ID: $node "; fab
```

2. このコマンドは各リーフに適用され、PVRSTP変更の最大件数と検出されたインターフェイスを確認します。

```
APIC#for node in `acidiag fnvread | grep leaf | awk '{print $1}'`; do echo; echo "node ID: $node "; fab
```

マルチキャストRPFドロップがあるかどうかを確認します。

これは、その時点で個々のリーフに対して行う必要があり、すべてのRPFドロップについて、有効になっているすべてのpim VRFを確認します。

```
APIC#for vrf in `show ip mroute summary vrf all | grep 'IP Multicast Routing Table for VR' | awk '{print
```

収集されるパケットが多すぎるかどうかを確認します。

1. ファブリックARPの光：

```
APIC#for leaf in `acidiag fnvread | grep leaf | awk '{print $1}'`; do echo; echo " -> leaf ID: $leaf  "
```

2. ARP収集の受信パケット：

```
APIC#for leaf in `acidiag fnvread | grep leaf | awk '{print $1}'`; do echo; echo " -> leaf ID: $leaf  "
```

QoSドロップ統計情報：

ファブリック全体で受信したQoSドロップを確認します。

```
APIC#moquery -c qosmIfClass -f 'qosm.IfClass.RxDropPacketsCount!="0"' | egrep "RxDropPacketsCount|dn"
```

ファブリック全体のQoS送信ドロップを確認します。

```
APIC#moquery -c qosmIfClass -f 'qosm.IfClass.TxDropPacketsCount!="0"' | egrep "TxDropPacketsCount|dn" |
```

インターフェイスの廃棄:

FCSエラー (非ストンプCRCエラー)

```
APIC#moquery -c rmonDot3Stats -f 'rmon.Dot3Stats.fCSErrors>="1"' | egrep "dn|fCSErrors"
```

FCS +ストンプCRCエラー

```
APIC#moquery -c rmonEtherStats -f 'rmon.EtherStats.cRCAlignErrors>="1"' | egrep "dn|cRCAlignErrors"
```

出力バッファドロップ

```
APIC#moquery -c rmonEgrCounters -f 'rmon.EgrCounters.bufferdroppkts>="1"' | egrep "dn|bufferdroppkts"
```

出力エラー

```
APIC#moquery -c rmonIfOut -f 'rmon.IfOut.errors>="1"' | egrep "dn|errors"
```

すべてのインターフェイスカウンタのクリア

ファブリックノードのリストを取得するコマンド

```
APIC# acidiag fnvread | egrep " active" | egrep "leaf|spine" | awk '{print $1}' | sed -e 'H;${x;s/\n/,/
101,102,103,204,205,206,301,1001,1002,2001,2002
```

前のリストのカウンタをクリアするコマンド

```
APIC# fabric 101,102,103,204,205,206,301,1001,1002,2001,2002 clear counters interface all
```

BGPセッションの問題：

ファブリックアンダーレイに問題のあるBGPセッションがあるかどうかを確認するには

```
APIC#moquery -c bgpPeerEntry -f 'bgp.PeerEntry.operSt!="established" and  bgp.PeerEntry.dn*"overlay-1"'
```

BGPセッションを確認します

```
APIC#moquery -c bgpPeerEntry -f 'bgp.PeerEntry.operSt!="established" | egrep "dn|operSt" | tr -d "\n" |
```

OSPFセッションの問題：

full状態ではないセッションを特定します。

```
APIC#moquery -c ospfAdjEp -f 'ospf.AdjEp.operSt!="full"' | egrep "dn|peerIp" | tr -d '\n' | sed "s/dn
```

絶えずフラップするセッションを特定し、最も高いカウントで情報をソートします。

```
APIC#moquery -c ospfAdjStats -f 'ospf.AdjStats.stChgCnt!="0"' | egrep "dn|stChgCnt" | tr -d "\n" | tr -
```

CPUにパントされるプライマリパケット

⚠ 考慮する：このコマンドは500パケットをデバッグします。金額を小さくするには、-cの後の数値を変更します。

```
lEAF#tcpdump -i kpm_inb -c 500 > /tmp/cpu-dp.txt
```

```
LEAF#cat /tmp/cpu-dp.txt | grep IP | awk '{print $3 , $4 , $5}' | grep -v $HOSTNAME | awk -F ':' '{prin
```

新規ファイルを作成せずに直接実行：

```
LEAF#tcpdump -i kpm_inb -c 500 | grep IP | awk '{print $3 , $4 , $5}' | grep -v $HOSTNAME | awk -F ':'
```

特定のすべての契約関係が導入されているapicからファブリック全体を確認します。

```
APIC#moquery -c actrlRule -f 'actrl.Rule.sPcTag=="32783" and actrl.Rule.dPcTag=="46" and actrl.Rule.sco
```

encapがすでに展開されている場所を確認し、対応するepgを取得します。

```
APIC#moquery -c l2CktEp -f 'l2.CktEp.encap=="vlan-3018"'
```

ファブリック内のすべてのノードのメモリ使用率：

```
APIC#bash
APIC# clear ; echo -e "Node ID\t\tFree Memory\tUsed Memory" ; moquery -c procSysMemHist15min -f 'proc.Sy
```

lstackでのドロップの確認（例外パケットがパントされる）

TXドロップを確認します。sort -nk 6 -rとともにコマンドを使用します。

```
 APIC# cat istack_debug | egrep "Protocol:|x_pkts_dropped" | tr -d "\n" | sed "s/Protocol/\nProtocol/g"
```

契約の検証

特定の契約のすべての関連を確認します。コントラクトとテナントの名前を置き換えるスクリプトを使用します。

```
APIC# CONTRACT='brc-<contract-name>'
APIC# TN='<tenant>'
#CHECK CONSUMERS
#To get the count of epgs consuming a contract (excluding vzany consumer):
APIC#icurl -g 'http://localhost:7777/api/node/mo/uni/tn-'$TN'/'$CONTRACT'.json?query-target=subtree&tar

#To list all epg objects consuming a contract (excluding vzany consumers):
APIC#icurl -g 'http://localhost:7777/api/node/mo/uni/tn-'$TN'/'$CONTRACT'.json?query-target=subtree&tar

#To get the count of vzanys consuming a contract:
APIC#icurl 'http://localhost:7777/api/node/mo/uni/tn-'$TN'/'$CONTRACT'.json?query-target=children&targe

#To list all vzany objects consuming a contract:
APIC#icurl 'http://localhost:7777/api/node/mo/uni/tn-'$TN'/'$CONTRACT'.json?query-target=children&targe

#CHECK PROVIDERS
#To get the count of epgs providing a contract (excluding vzany consumer):
APIC#icurl -g 'http://localhost:7777/api/node/mo/uni/tn-'$TN'/'$CONTRACT'.json?query-target=subtree&tar

#To list all epg objects providing a contract (excluding vzany consumers):
APIC#icurl -g 'http://localhost:7777/api/node/mo/uni/tn-'$TN'/'$CONTRACT'.json?query-target=subtree&tar

#To get the count of vzanys providing a contract:
APIC#icurl 'http://localhost:7777/api/node/mo/uni/tn-'$TN'/'$CONTRACT'.json?query-target=children&targe

#To list all vzany objects providing a contract:
APIC#icurl 'http://localhost:7777/api/node/mo/uni/tn-'$TN'/'$CONTRACT'.json?query-target=children&targe
```

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。