Single Line 명령으로 패브릭 문제 해결 및 확인

목차

소개

사전 요구 사항

- 요구 사항
- 사용되는 구성 요소

배경 정보

- 정보를 파악하는 데 사용되는 도구

모든 단일 행 명령 목록

- 패브릭 내에서 LEAF의 노드 ID만 가져옵니다.
- 인터페이스 재설정이 있는지 확인합니다.
- 가장 높은 등급의 인터페이스 확인:
- 모든 STP 토폴로지 변경 내용 찾기:
- 멀티캐스트 RPF 드롭이 있는지 확인합니다.
- Glean을 가져오는 패킷이 너무 많은지 확인합니다.
- QoS 삭제 통계:
- 인터페이스 삭제:
- FCS 오류(비압축 CRC 오류)
- FCS + 압축된 CRC 오류
- 출력 버퍼 삭제
- output errors
- 모든 인터페이스 카운터 지우기
- BGP 세션 문제:
- OSPF 세션 문제:
- CPU에 펀트된 Primarypackets
- 모든 특정 계약 관계가 구축된 apic에서 패브릭 전반 확인
- 엔드캡이 이미 구축된 위치를 확인하고 해당 epg 가져오기
- 패브릭 내 모든 노드의 메모리 사용률:
- istack에서 삭제 확인(예외 패킷이 펀팅됨)
- 계약 검증

소개

이 문서에서는 패브릭의 전반적인 문제를 탐지할 수 있는 다양한 방법을 설명합니다.

사전 요구 사항

요구 사항

- Cisco에서는 ACI에 대한 지식을 권장합니다.

- 바시에 대한 기본 지식

사용되는 구성 요소

이 문서는 특정 소프트웨어 및 하드웨어 버전으로 한정되지 않습니다.

사용된 장치:

- 버전 4.2(3)를 실행하는 Cisco ACI

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

배경 정보

정보를 파악하는 데 사용되는 도구

- 대용으로 사용: sed "s/<ldword >/<new work>/g" g는 두 번 이상 수행되었음을 정의합니다.
- 시작부터 끝까지 줄을 잡는 데 사용합니다. "/<beginning/,/end/p>"을 입력했습니다. grep의 기능을 복제하려면 -n(noprint) 옵션을 /p print 플래그와 결합합니다.
- Sed는 다음과 같이 ';'을 사용하여 두 번 이상 사용할 수 있습니다. 할 일: sed "s/<ldword >/<new work>/g; s/<oldword2>/<new work2>/g"
- awk -F '<field_separator>' '{print $2}' 이 특정 예에서는 정의된 FIELD_SEPARATOR로 행을 분할하고 두 번째로 구분된 청크를 인쇄합니다. 두 구문이 정확히 같은 일을 한다.
- awk '{ print $1, $2 }'는 각 입력 레코드의 처음 두 필드를 공백으로 인쇄합니다.
- 정렬 | uniq 반복되는 행을 보고합니다. 발생 -c 접두사 라인을 사용합니다.
- sort -nrk <column> - 라인을 가장 높게 정렬합니다. -n 숫자 정렬의 경우 -k - 열을 수정할 수 있도록 키 -r을 제거할 수 있습니다.
- python -m json.tool 예쁜 형식으로 JSON을 표시합니다.

모든 단일 행 명령 목록

패브릭 내에서 LEAF의 노드 ID만 가져옵니다.

1. 목록에서:

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

2. 구분 기호로 쉼표가 있는 줄:

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

인터페이스 재설정이 있는지 확인합니다.

이 정보는 재설정 수가 가장 많은 인터페이스를 기준으로 정렬됩니다.

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

저속 옵션

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

가장 높은 등급의 인터페이스를 확인합니다.

가장 많은 트래픽이 수신된 위치를 찾으려면

특정 값(b) 이상의 출력 처리량이 있는 모든 인터페이스에 대해 패브릭을 쿼리합니다. m 값은 b가 gb, mb 또는 kb인지 여부를 정의합니다.

gb에서 m을 125000000으로 설정하고, mb에서 m을 125000으로 설정하고, kb에서 m을 125로 설정합니다.

```
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  "
```

모든 STP 토폴로지 변경 내용 찾기:

1. 이 명령은 각 leaf에 적용되며 최근 토폴로지 변경 사항이 있는지, 어떤 인터페이스에 적용되는지 확인합니다.

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

2. 이 명령은 각 leaf에 적용되며 PVRSTP 변경의 최대 카운트와 어떤 인터페이스에 표시되는지 확인합니다.

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

멀티캐스트 RPF 드롭이 있는지 확인합니다.

이는 현재 개별 leaf에 대한 것이어야 하며 RPF 삭제에 대해 활성화된 모든 pim VRF를 확인합니다.

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

Glean을 가져오는 패킷이 너무 많은지 확인합니다.

1. 패브릭 ARP 글레인:

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

2. ARP가 수신한 패킷:

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

QoS 삭제 통계:

전체 패브릭에 대한 QoS 수신 삭제를 확인합니다.

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

전체 패브릭에 대한 QoS 전송 삭제를 확인합니다.

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

인터페이스 삭제:

FCS 오류(비압축 CRC 오류)

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

FCS + 압축된 CRC 오류

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

출력 버퍼 삭제

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

output errors

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

모든 인터페이스 카운터 지우기

패브릭 노드 목록을 가져오는 명령

```
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
```

이전 목록에 대한 카운터를 지우는 명령

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

BGP 세션 문제:

패브릭 언더레이에 문제가 있는 BGP 세션이 있는지 확인하기 위해

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

BGP 세션을 확인하려면

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

OSPF 세션 문제:

전체 상태가 아닌 세션을 식별합니다.

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

지속적으로 플랩하는 세션을 식별하고 최대 카운트에 대한 정보를 정렬합니다.

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

CPU에 펀트된 기본 패킷

⚠ 다음 사항을 고려하십시오. 이 명령은 500개의 패킷을 디버깅합니다. 금액이 낮은 경우 -c 뒤에 숫자를 수정합니다.

```
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
```

새 파일을 만들지 않고 직접 실행:

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

모든 특정 계약 관계가 구축된 apic에서 패브릭 전반 확인

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

엔드캡이 이미 구축된 위치를 확인하고 해당 epg 가져오기

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

패브릭 내 모든 노드의 메모리 사용률:

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

istack에서 삭제 확인(예외 패킷이 펀팅됨)

TX 삭제를 검토합니다. 명령을 sort -nk 6 -r과 함께 사용합니다.

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

계약 검증

특정 계약의 모든 관계를 검토합니다. 계약 및 테넌트의 이름을 바꾸는 스크립트를 사용합니다.

```
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
```

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.