L'utilisation du CPU élevée de commutateur de gamme Catalyst 3850 dépannent

Contenu

Introduction

Ce document décrit comment dépanner des soucis d'utilisation du CPU, principalement dus aux interruptions, sur le nouveau Cisco IOS® - plate-forme XE. Supplémentaire, le document introduit plusieurs nouvelles commandes sur cette plate-forme qui sont intégrales afin de dépanner de tels problèmes.

Informations générales

Il est important de comprendre comment le Cisco IOS XE est construit. Avec le Cisco IOS XE, Cisco s'est déplacé à un kernel Linux et tous les sous-systèmes ont été décomposés en processus. Tous les sous-systèmes qui étaient Cisco IOS intérieur avant - comme les gestionnaires de modules, la Haute disponibilité (ha), et ainsi de suite - maintenant fonctionnent comme processus de logiciel dans le système d'exploitation Linux (SYSTÈME D'EXPLOITATION). Le Cisco IOS lui-même fonctionne en tant que démon dans le système d'exploitation Linux (IOSd). Le Cisco IOS XE retient non seulement le même aspect et impression du Cisco IOS classique, mais également son exécution, support, et Gestion.

Voici quelques définitions utiles :

- **Transmission du gestionnaire d'engine (ALIMENTÉ)** : C'est le coeur de la gamme Cisco Catalyst 3850 commutent et sont responsable de toute la programmation de matériel/expédition.

- **IOSd** : C'est le démon de Cisco IOS qui s'exécute sur le kernel Linux. Il est exécuté comme processus de logiciel dans le noyau.

- **Système de distribution de paquet (PDS)** : C'est l'architecture et le processus de la façon dont

des paquets sont livrés à et du divers sous-système. Comme exemple, il contrôle comment des paquets sont livrés du ALIMENTÉ à l'IOSd et vice versa.

- **Traitement** : Un traitement peut être considéré comme un pointeur. Il est des moyens de découvrir plus d'informations détaillées au sujet des variables spécifiques qui sont utilisées dans les sorties que la case produit. C'est semblable au concept des index locaux de la logique de cible (LTL) sur la gamme Cisco Catalyst 6500 commutent.

Étude de cas : Interruptions de Protocole ARP (Address Resolution Protocol)

Le dépannage et le processus de vérification dans cette section peuvent être largement utilisés pour l'utilisation du CPU élevée due aux interruptions.

Étape 1 : Identifiez le processus qui consomme des cycles CPU

Les affichages de commande **CPU de processus d'exposition** naturellement à quoi la CPU ressemble actuellement. Notez que la gamme Cisco Catalyst 3850 Swtich utilise quatre noyaux, et vous voient l'utilisation du CPU répertoriée pour chacun des quatre noyaux :

```
3850-2#show processes cpu sort | exclude 0.0
Core 0: CPU utilization for five seconds: 53%; one minute: 39%;  five minutes: 41%
Core 1: CPU utilization for five seconds: 43%; one minute: 57%;  five minutes: 54%
Core 2: CPU utilization for five seconds: 95%; one minute: 60%;  five minutes: 58%
Core 3: CPU utilization for five seconds: 32%; one minute: 31%;  five minutes: 29%
PID    Runtime(ms) Invoked  uSecs  5Sec     1Min     5Min     TTY   Process
8525   472560      2345554  7525   31.37    30.84    30.83    0     iosd
5661   2157452     9234031  698    13.17    12.56    12.54    1088  fed
6206   19630       74895    262    1.83     0.43     0.10     0     eicored
6197   725760      11967089 60     1.41     1.38     1.47     0     pdsd
```

De la sortie, il est clair que le démon de Cisco IOS consomme une partie importante de la CPU avec ALIMENTÉE, qui est le coeur de cette case. Quand l'utilisation du CPU est haute due aux interruptions, vous voyez qu'IOSd et utilisation FED par principale partie de la CPU, et ces sous-processus (ou un sous-ensemble de ces derniers) utilisent la CPU :

- Punject ALIMENTÉ TX
- Punject ALIMENTÉ RX
- Punject ALIMENTÉ complètent le niveau
- Punject ALIMENTÉ TX se terminent

Vous pouvez zoomer dans l'un de ces processus avec la commande de ***<process> détaillée par CPU de processus d'exposition***. Puisqu'IOSd est responsable de la majorité de l'utilisation du CPU, voici un oeil plus attentif dans celui.

```
3850-2#show processes cpu detailed process iosd sort | ex 0.0
Core 0: CPU utilization for five seconds: 36%; one minute: 39%; five minutes: 40%
Core 1: CPU utilization for five seconds: 73%; one minute: 52%; five minutes: 53%
Core 2: CPU utilization for five seconds: 22%; one minute: 56%; five minutes: 58%
Core 3: CPU utilization for five seconds: 46%; one minute: 40%; five minutes: 31%
PID    T C  TID  Runtime(ms)Invoked uSecs 5Sec   1Min   5Min  TTY  Process
                                          (%)    (%)    (%)
```

```
8525   L          556160     2356540 7526  30.42   30.77  30.83  0   iosd
8525   L 1  8525  712558     284117  0     23.14   23.33  23.38  0   iosd
59     I          1115452    4168181 0     42.22   39.55  39.33  0   ARP Snoop
198    I          3442960    4168186 0     25.33   24.22  24.77  0   IP Host Track Proce
30     I          3802130    4168183 0     24.66   27.88  27.66  0   ARP Input
283    I          574800     3225649 0     4.33    4.00   4.11   0   DAI Packet Process

3850-2#show processes cpu detailed process fed sorted | ex 0.0
Core 0: CPU utilization for five seconds: 45%; one minute: 44%; five minutes: 44%
Core 1: CPU utilization for five seconds: 38%; one minute: 44%; five minutes: 45%
Core 2: CPU utilization for five seconds: 42%; one minute: 41%; five minutes: 40%
Core 3: CPU utilization for five seconds: 32%; one minute: 30%; five minutes: 31%
PID    T C  TID  Runtime(ms)Invoked uSecs 5Sec   1Min   5Min  TTY  Process
                                          (%)    (%)    (%)
5638   L          612840     1143306 536   13.22  12.90   12.93 1088 fed
5638   L 3  8998  396500     602433  0     9.87   9.63    9.61  0     PunjectTx
5638   L 3  8997  159890     66051   0     2.70   2.70    2.74  0     PunjectRx
```

La sortie (CPU d'IOSd sortie) prouve que le fureteur d'ARP, le processus de piste d'hôte IP, et l'entrée d'ARP sont élevés. C'est généralement - vu quand la CPU est due interrompu aux paquets d'ARP.

Étape 2 : Déterminez la file d'attente CPU qui entraîne l'état élevé d'utilisation du CPU

Le commutateur de gamme Cisco Catalyst 3850 a un certain nombre de files d'attente qui approvisionnent à différents types de paquets (ALIMENTÉ met à jour 32 files d'attente CPU RX, qui sont des files d'attente qui vont directement à la CPU). Il est important de surveiller ces files d'attente afin de découvrir quels paquets sont donnés un coup de volée à la CPU et ce qui sont traités par l'IOSd. Ces files d'attente sont par ASIC.

> **Note**: Il y a deux ASIC : 0 et 1. les ports 1 à 24 appartiennent à ASIC 0.

Afin de regarder les files d'attente, écrivez le ***<rx de direction de <queue> de cpuq de <port-asic> de port-ASIC de statistiques de coup de volée de show platform|***commande de ***tx>***.

En **statistiques port-ASIC de coup de volée de show platform 0** commandes de **rx de direction du cpuq -1**, les **-1** listes d'arguments toutes les files d'attente. Par conséquent, les listes de ces commandes toutes reçoivent des files d'attente pour Port-ASIC 0.

Maintenant, vous devez identifier qui alignent des poussers un grand nombre de paquets à un haut débit. Dans cet exemple, un examen des files d'attente a indiqué ce coupable :

```
<snip>
RX (ASIC2CPU) Stats (asic 0 qn 16 lqn 16):
RXQ 16: CPU_Q_PROTO_SNOOPING
----------------------------------------
Packets received from ASIC     : 79099152
Send to IOSd total attempts    : 79099152
Send to IOSd failed count      : 1240331
RX suspend count               : 1240331
RX unsuspend count             : 1240330
RX unsuspend send count        : 1240330
RX unsuspend send failed count : 0
RX dropped count               : 0
RX conversion failure dropped  : 0
```

```
RX pkt_hdr allocation failure  : 0
RX INTACK count                : 0
RX packets dq'd after intack   : 0
Active RxQ event               : 9906280
RX spurious interrupt          : 0
<snip>
```

Le nombre de file d'attente est 16 et le nom de file d'attente est **CPU_Q_PROTO_SNOOPING**.

Une autre manière de découvrir la file d'attente de coupable est de sélectionner la commande de **client de coup de volée de show platform**.

```
3850-2#show platform punt client
 tag      buffer         jumbo    fallback     packets   received   failures
                                              alloc   free  bytes    conv  buf
 27       0/1024/2048     0/5       0/5         0     0          0     0     0
 65536    0/1024/1600     0/0       0/512       0     0          0     0     0
 65537    0/ 512/1600     0/0       0/512    1530  1530     244061     0     0
 65538    0/   5/5        0/0       0/5         0     0          0     0     0
 65539    0/2048/1600     0/16      0/512       0     0          0     0     0
 65540    0/ 128/1600     0/8       0/0         0     0          0     0     0
 65541    0/ 128/1600     0/16      0/32        0     0          0     0     0
 65542    0/ 768/1600     0/4       0/0         0     0          0     0     0
 65544    0/  96/1600     0/4       0/0         0     0          0     0     0
 65545    0/  96/1600     0/8       0/32        0     0          0     0     0
 65546    0/ 512/1600     0/32      0/512       0     0          0     0     0
 65547    0/  96/1600     0/8       0/32        0     0          0     0     0
 65548    0/ 512/1600     0/32      0/256       0     0          0     0     0
 65551    0/ 512/1600     0/0       0/256       0     0          0     0     0
 65556    0/  16/1600     0/4       0/0         0     0          0     0     0
 65557    0/  16/1600     0/4       0/0         0     0          0     0     0
 65558    0/  16/1600     0/4       0/0         0     0          0     0     0
 65559    0/  16/1600     0/4       0/0         0     0          0     0     0
 65560    0/  16/1600     0/4       0/0         0     0          0     0     0
s65561  421/ 512/1600     0/0       0/128  79565859 131644697  478984244     0 37467
 65563    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65564    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65565    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65566    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65581    0/   1/1        0/0       0/0         0     0          0     0     0
 131071   0/  96/1600     0/4       0/0         0     0          0     0     0
fallback pool: 98/1500/1600
jumbo pool:    0/128/9300
```

Déterminez la balise pour laquelle les la plupart des paquets ont été alloués. Dans cet exemple, il est **65561**.

Puis, sélectionnez cette commande :

```
3850-2#show pds tag all | in Active|Tags|65561
  Active   Client Client
    Tags   Handle Name                 TDA        SDA          FDA   TBufD        TBytD
   65561  7296672 Punt Rx Proto Snoop  79821397  79821397    0  79821397   494316524
```

Cette sortie prouve que la file d'attente est **fureteur Proto de Rx**.

Le s avant que les **65561** dans la sortie de l'ordre de **client de coup de volée de show platform** signifie que le traitement ALIMENTÉ est interrompu et accablé par le nombre de paquets entrant. Si le **s** ne disparaît pas, il signifie que la file d'attente est coincée de manière permanente.

Étape 3 : Videz le paquet envoyé à la CPU

Dans les résultats de la **balise de l'exposition PDS toute la** commande, notent un traitement, **7296672**, est signalée à côté du **fureteur Proto de Rx de coup de volée**.

Utilisez ce traitement dans la commande d'**évier de bout de paquet de *<handle> de* client de l'exposition PDS**. Notez que vous devez activer **mettez au point le PDS pktbuf-dernier** avant que vous utilisiez la commande. Autrement vous rencontrez cette erreur :

```
3850-2#show pds client 7296672 packet last sink
% switch-2:pdsd:This command works in debug mode only. Enable debug using
"debug pds pktbuf-last" command
```

Le débogage étant activé, vous voyez cette sortie :

```
3850-2#show pds client 7296672 packet last sink
Dumping Packet(54528) # 0 of Length 60
----------------------------------------
Meta-data
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0010 00 00 16 1d 00 00 00 00 00 00 00 00 55 5a 57 f0  ............UZW.
0020 00 00 00 00 fd 01 10 df 00 5b 70 00 00 10 43 00  .........[p...C.
0030 00 10 43 00 00 41 fd 00 00 41 fd 00 00 00 00 00  ..C..A...A......
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0050 00 00 00 3c 00 00 00 00 00 01 00 19 00 00 00 00  ...<............
0060 01 01 b6 80 00 00 00 4f 00 00 00 00 00 00 00 00  .......O........
0070 01 04 d8 80 00 00 00 33 00 00 00 00 00 00 00 00  .......3........
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0090 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00a0 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00  ................
Data
0000 ff ff ff ff ff ff aa bb cc dd 00 00 08 06 00 01  ................
0010 08 00 06 04 00 01 aa bb cc dd 00 00 c0 a8 01 0a  ................
0020 ff ff ff ff ff ff c0 a8 01 14 00 01 02 03 04 05  ................
0030 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11              ............
```

Cette commande vide le dernier paquet reçu par l'évier, qui est IOSd dans cet exemple. Ceci prouve qu'il vide l'en-tête et il peut être décodé avec Wireshark terminal Terminal (TShark). **Les méta-données** sont pour l'usage interne par le système, mais les **données produites** fournissent les informations de paquet réelles. **Les méta-données**, cependant, demeurent extrêmement utiles.

Notez la ligne qui commence par **0070**. Utilisez les 16 premiers bits ensuite qui comme affiché ici :

```
3850-2#show platform port-asic ifm iif-id 0x0104d88000000033
Interface Table
Interface IIF-ID        : 0x0104d88000000033
Interface Name          : Gi2/0/20
Interface Block Pointer : 0x514d2f70
Interface State         : READY
Interface Stauts        : IFM-ADD-RCVD, FFM-ADD-RCVD
Interface Ref-Cnt       : 6
Interface Epoch         : 0
Interface Type          : ETHER
        Port Type         : SWITCH PORT
       Port Location     : LOCAL
        Slot              : 2
        Unit              : 20
       Slot Unit         : 20
```

```
       Acitve            : Y
       SNMP IF Index     : 22
       GPN               : 84
       EC Channel        : 0
       EC Index          : 0
       ASIC              : 0
       ASIC Port         : 14
       Port LE Handle    : 0x514cd990
Non Zero Feature Ref Counts
       FID : 48(AL_FID_L2_PM), Ref Count : 1
       FID : 77(AL_FID_STATS), Ref Count : 1
       FID : 51(AL_FID_L2_MATM), Ref Count : 1
       FID : 13(AL_FID_SC), Ref Count : 1
       FID : 26(AL_FID_QOS), Ref Count : 1
Sub block information
       FID : 48(AL_FID_L2_PM), Private Data &colon; 0x54072618
       FID : 26(AL_FID_QOS), Private Data &colon; 0x514d31b8
```

L'interface de coupable est identifiée ici. **Gig2/0/20** est où il y a un générateur du trafic ce trafic ARP de pompes. Si vous fermez ceci avalez, alors il résoudrait le problème et réduirait l'utilisation du CPU.

Étape 4 : Suivi ALIMENTÉ par utilisation

Le seul inconvénient avec la méthode discutée dans la dernière section est qu'il vide seulement le dernier paquet qui entre dans l'évier, et ce ne pourrait pas être le coupable.

Une meilleure manière de dépanner ceci serait d'utiliser une caractéristique appelée le **suivi ALIMENTÉ**. Le suivi est une méthode de capture de paquet (utilisant de divers filtres) qui sont poussés par ALIMENTÉ à la CPU. Le suivi ALIMENTÉ n'est pas aussi simple que la caractéristique de Netdr sur la gamme Cisco Catalyst 6500 commutent, cependant. Ici le processus est divisé en étapes :

1. Cheminement de détail d'enable. Par défaut, le suivi d'événement est allumé. Vous devez permettre au suivi de détail afin de capturer les paquets réels :

```
3850-2#set trace control fed-punject-detail enable
```

2. Réglez avec précision la mémoire tampon de capture. Déterminez comment profondément vos mémoires tampons sont pour le suivi et l'augmentation de détail comme nécessaires.

```
3850-2#show mgmt-infra trace settings fed-punject-detail
One shot Trace Settings:

Buffer Name: fed-punject-detail
Default Size: 32768
Current Size: 32768
Traces Dropped due to internal error: No
Total Entries Written: 0
One shot mode: No
One shot and full: No
Disabled: False
```

Vous pouvez changer la taille de mémoire tampon avec cette commande :

```
3850-2#set trace control fed-punject-detail buffer-size <buffer size>
```

Les valeurs disponibles à vous sont :

```
3850-2#set trace control fed-punject-detail buffer-size ?
 <8192-67108864>  The new desired buffer size, in bytes
 default          Reset trace buffer size to default
```

3. Ajoutez les filtres de capture. Vous devez maintenant ajouter de divers filtres pour la capture. Vous pouvez ajouter différents filtres et pour choisir **d'apparier toute l'**ou le **match any** ceux pour votre capture.

   Des filtres sont ajoutés avec cette commande :

```
3850-2#set trace fed-punject-detail direction rx filter_add <filter>
```

   Ces options sont actuellement disponibles :

```
3850-2#set trace fed-punject-detail direction rx filter_add ?
 cpu-queue  rxq 0..31
 field      field
 offset     offset
```

   Maintenant vous devez joindre des choses ensemble. Souvenez-vous la file d'attente de coupable qui a été identifiée dans l'étape 2 de ceci dépannent le processus ? Puisque la file d'attente 16 est la file d'attente qui pousse un grand nombre de paquets vers la CPU, elle semble raisonnable de tracer cette file d'attente et de voir quels paquets sont donnés un coup de volée à la CPU par elle.

   Vous pouvez choisir de tracer n'importe quelle file d'attente avec cette commande :

```
 3850-2#set trace fed-punject-detail direction rx filter_add cpu-queue <start queue>
<end queue>
```

   Voici la commande pour cet exemple :

```
3850-2#set trace fed-punject-detail direction rx filter_add cpu-queue 16 16
```

Vous devez choisir une **correspondance tous** ou un **match any** pour vos filtres et puis activer le suivi :

```
3850-2#set trace fed-punject-detail direction rx match_all
3850-2#set trace fed-punject-detail direction rx filter_enable
```

4. Paquets filtrés par affichage. Vous pouvez afficher les paquets capturés avec la commande d'alimenter-punject-**détail de messages de suivi de mgmt-infra d'exposition**.

```
3850-2#show mgmt-infra trace messages fed-punject-detail
[11/25/13 07:05:53.814 UTC 2eb0c9 5661]
00 00 00 00 00 4e 00 40 07 00 02 08 00 00 51 3b
00 00 00 00 00 01 00 00 03 00 00 00 00 00 00 01
00 00 00 00 20 00 00 0e 00 00 00 00 00 01 00 74
00 00 00 04 00 54 41 02 00 00 00 00 00 00 00 00

[11/25/13 07:05:53.814 UTC 2eb0ca 5661]
ff ff ff ff ff ff aa bb cc dd 00 00 08 06 00 01
08 00 06 04 00 01 aa bb cc dd 00 00 c0 a8 01 0a
ff ff ff ff ff ff c0 a8 01 14 00 01 02 03 04 05
06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 f6 b9 10 32
[11/25/13 07:05:53.814 UTC 2eb0cb 5661] Frame descriptors:
[11/25/13 07:05:53.814 UTC 2eb0cc 5661]
=========
fdFormat=0x4    systemTtl=0xe
loadBalHash1=0x8        loadBalHash2=0x8
spanSessionMap=0x0      forwardingMode=0x0
destModIndex=0x0        skipIdIndex=0x4
srcGpn=0x54     qosLabel=0x41
srcCos=0x0      ingressTranslatedVlan=0x3
bpdu=0x0        spanHistory=0x0
sgt=0x0 fpeFirstHeaderType=0x0
srcVlan=0x1     rcpServiceId=0x2
wccpSkip=0x0    srcPortLeIndex=0xe
cryptoProtocol=0x0      debugTagId=0x0
vrfId=0x0       saIndex=0x0
pendingAfdLabel=0x0     destClient=0x1
appId=0x0       finalStationIndex=0x74
decryptSuccess=0x0      encryptSuccess=0x0
rcpMiscResults=0x0      stackedFdPresent=0x0
spanDirection=0x0       egressRedirect=0x0
redirectIndex=0x0       exceptionLabel=0x0
destGpn=0x0     inlineFd=0x0
suppressRefPtrUpdate=0x0        suppressRewriteSideEfects=0x0
cmi2=0x0        currentRi=0x1
currentDi=0x513b        dropIpUnreachable=0x0
srcZoneId=0x0   srcAsicId=0x0
originalDi=0x0  originalRi=0x0
srcL3IfIndex=0x2        dstL3IfIndex=0x0
dstVlan=0x0     frameLength=0x40
fdCrc=0x7       tunnelSpokeId=0x0

=========
[11/25/13 07:05:53.814 UTC 2eb0cd 5661]
[11/25/13 07:05:53.814 UTC 2eb0ce 5661] PUNT PATH (fed_punject_rx_process_packet:
830):RX: Q: 16, Tag: 65561
```

```
[11/25/13 07:05:53.814 UTC 2eb0cf 5661] PUNT PATH (fed_punject_get_physical_iif:
579):RX: Physical IIF-id 0x104d88000000033
[11/25/13 07:05:53.814 UTC 2eb0d0 5661] PUNT PATH (fed_punject_get_src_l3if_index:
434):RX: L3 IIF-id 0x101b6800000004f
[11/25/13 07:05:53.814 UTC 2eb0d1 5661] PUNT PATH (fed_punject_fd_2_pds_md:478):
RX: l2_logical_if = 0x0
[11/25/13 07:05:53.814 UTC 2eb0d2 5661] PUNT PATH (fed_punject_get_source_cos:638):
RX: Source Cos 0
[11/25/13 07:05:53.814 UTC 2eb0d3 5661] PUNT PATH (fed_punject_get_vrf_id:653):
RX: VRF-id 0
[11/25/13 07:05:53.814 UTC 2eb0d4 5661] PUNT PATH (fed_punject_get_src_zoneid:667):
RX: Zone-id 0
[11/25/13 07:05:53.814 UTC 2eb0d5 5661] PUNT PATH (fed_punject_fd_2_pds_md:518):
RX: get_src_zoneid failed
[11/25/13 07:05:53.814 UTC 2eb0d6 5661] PUNT PATH (fed_punject_get_acl_log_direction:
695): RX: : Invalid CMI2
[11/25/13 07:05:53.814 UTC 2eb0d7 5661] PUNT PATH (fed_punject_fd_2_pds_md:541):RX:
get_acl_log_direction failed
[11/25/13 07:05:53.814 UTC 2eb0d8 5661] PUNT PATH (fed_punject_get_acl_full_direction:
724):RX: DI 0x513b ACL Full Direction 1
[11/25/13 07:05:53.814 UTC 2eb0d9 5661] PUNT PATH (fed_punject_get_source_sgt:446):
RX: Source SGT 0
[11/25/13 07:05:53.814 UTC 2eb0da 5661] PUNT PATH (fed_punject_get_first_header_type:680):
RX: FirstHeaderType 0
[11/25/13 07:05:53.814 UTC 2eb0db 5661] PUNT PATH (fed_punject_rx_process_packet:916):
RX: fed_punject_pds_send packet 0x1f00 to IOSd with tag 65561
[11/25/13 07:05:53.814 UTC 2eb0dc 5661] PUNT PATH (fed_punject_rx_process_packet:744):
RX: **** RX packet 0x2360 on qn 16, len 128 ****
[11/25/13 07:05:53.814 UTC 2eb0dd 5661]
buf_no 0 buf_len 128

<snip>
```

Cette sortie fournit l'abondance des informations et devrait typiquement être assez pour découvrir où les paquets provenus et ce qui est contenu dans elles.

La première partie du vidage mémoire d'en-tête est de nouveau les **méta-données** qui est utilisé par le système. La deuxième partie est le paquet réel.

ff ff ff ff ff ff - destination MAC address
aa bb cc dd 00 00 - source MAC address

Vous pouvez choisir de tracer cette adresse MAC source afin de découvrir le port de coupable (une fois que vous avez identifié que c'est la majorité des paquets qui sont donnés un coup de volée de la file d'attente 16 ; cette sortie affiche seulement qu'un exemple du paquet et l'autres sortie/paquets sont coupés).

Cependant, il y a une meilleure manière. Notez que logs qui sont présents après les informations d'en-tête :

```
[11/25/13 07:05:53.814 UTC 2eb0ce 5661] PUNT PATH (fed_punject_rx_process_packet:
830):RX: Q: 16, Tag: 65561
[11/25/13 07:05:53.814 UTC 2eb0cf 5661] PUNT PATH (fed_punject_get_physical_iif:
```

```
579):RX: Physical IIF-id 0x104d88000000033
```

Le premier log vous indique clairement que de ce que la file d'attente et étiquettent ce paquet est livré. Si vous ne vous rendiez pas compte de l'eariler de file d'attente, c'est une méthode facile de l'identifier qui l'alignent étaient.

Le deuxième log est bien plus utile, parce qu'il fournit l'usine d'ID d'interface physique (IIF) - ID pour l'interface de source. La valeur hexadécimale est un traitement qui peut être utilisé afin de vider des informations sur ce port :

```
3850-2#show platform port-asic ifm iif-id 0x0104d88000000033
Interface Table
Interface IIF-ID        : 0x0104d88000000033
Interface Name          : Gi2/0/20
Interface Block Pointer : 0x514d2f70
Interface State         : READY
Interface Stauts        : IFM-ADD-RCVD, FFM-ADD-RCVD
Interface Ref-Cnt       : 6
Interface Epoch         : 0
Interface Type          : ETHER
        Port Type         : SWITCH PORT
       Port Location     : LOCAL
        Slot              : 2
        Unit              : 20
       Slot Unit         : 20
       Acitve            : Y
       SNMP IF Index     : 22
       GPN               : 84
       EC Channel        : 0
       EC Index          : 0
       ASIC              : 0
       ASIC Port         : 14
       Port LE Handle    : 0x514cd990
Non Zero Feature Ref Counts
       FID : 48(AL_FID_L2_PM), Ref Count : 1
       FID : 77(AL_FID_STATS), Ref Count : 1
       FID : 51(AL_FID_L2_MATM), Ref Count : 1
       FID : 13(AL_FID_SC), Ref Count : 1
       FID : 26(AL_FID_QOS), Ref Count : 1
Sub block information
       FID : 48(AL_FID_L2_PM), Private Data &colon; 0x54072618
       FID : 26(AL_FID_QOS), Private Data &colon; 0x514d31b8
```

Vous avez identifié de nouveau l'interface et le coupable de souce.

Le suivi est un outil puissant qui est essentiel afin de dépanner des problèmes élevés d'utilisation du CPU et fournit l'abondance des informations afin de résoudre avec succès une telle situation.

Script du gestionnaire d'événement inclus par échantillon (EEM) pour le commutateur de gamme Cisco Catalyst 3850

Employez cette commande afin de déclencher un log à générer à un seuil spécifique :

```
process cpu threshold type total rising <CPU %> interval <interval in seconds>
switch <switch number>
```

Le log généré avec la commande ressemble à ceci :

```
*Jan 13 00:03:00.271: %CPUMEM-5-RISING_THRESHOLD: 1 CPUMEMd[6300]:  Threshold: :
50, Total CPU Utilzation(total/Intr) :50/0, Top 3 processes(Pid/Util) : 8622/25,
5753/12, 9663/0
```

Le log généré fournit ces informations :

- Toute l'utilisation du processeur au moment du déclencheur. Ceci est identifié par **CPU totale Utilzation (total/Intr) : 50/0** dans cet exemple.
- Processus supérieurs - ceux-ci sont répertoriés dans le format de **PID/CPU%**. Ainsi dans cet exemple, ceux-ci sont :

```
*Jan 13 00:03:00.271: %CPUMEM-5-RISING_THRESHOLD: 1 CPUMEMd[6300]:  Threshold: :
50, Total CPU Utilzation(total/Intr) :50/0, Top 3 processes(Pid/Util) : 8622/25,
5753/12, 9663/0
```

Le script EEM est affiché ici :

```
*Jan 13 00:03:00.271: %CPUMEM-5-RISING_THRESHOLD: 1 CPUMEMd[6300]:  Threshold: :
50, Total CPU Utilzation(total/Intr) :50/0, Top 3 processes(Pid/Util) : 8622/25,
5753/12, 9663/0
```

> **Note**: La commande de **seuil CPU de processus** ne fonctionne pas actuellement dans la série 3.2.X. Un autre point à se souvenir est que cette commande regarde l'utilzation moyen CPU parmi les quatre noyaux et génère un log quand cette moyenne atteint le pourcentage qui a été défini dans la commande.

Cisco IOS XE 16.x ou versions ultérieures

Si vous avez des Commutateurs du Catalyst 3850 qui exécutent la version 16.x de Logiciel Cisco IOS XE version 2 ou plus tard, voyez pour dépanner l'utilisation du CPU élevée dans des plates-formes de commutateur Catalyst exécutant IOS-XE 16.x.

Informations connexes

- Quel est Cisco IOS XE ?
- Cisco Catalyst 3850 Commutateurs - Fiches techniques et littérature
- Support et documentation techniques - Cisco Systems