Risoluzione dei problemi di utilizzo elevato della CPU dello switch Catalyst serie 3850

Sommario

[Introduzione](#)
[Premesse](#)
[Caso di studio: Address Resolution Protocol (ARP) Interrupts](#)
[Passaggio 1: Identificare il processo che utilizza i cicli della CPU](#)
[Passaggio 2: Determinare la coda di CPU che causa la condizione di utilizzo elevato della CPU](#)
[Passaggio 3: Scaricare il pacchetto inviato alla CPU](#)
[Passaggio 4: Utilizzare la traccia FED](#)
[Esempio di script EEM (Embedded Event Manager) per gli switch Cisco Catalyst serie 3850](#)
[Cisco IOS-XE 16.x o versioni successive](#)
[Informazioni correlate](#)

Introduzione

In questo documento viene descritto come risolvere i problemi relativi all'utilizzo della CPU, principalmente a causa di interruzioni, sulla nuova piattaforma Cisco IOS®-XE.

Premesse

È importante comprendere come è costruito Cisco IOS®-XE. Con Cisco IOS®-XE, Cisco è passato a un kernel Linux e tutti i sottosistemi sono stati suddivisi in processi. Tutti i sottosistemi precedentemente presenti in Cisco IOS®, ad esempio i driver dei moduli, l'alta disponibilità (HA, High Availability) e così via, vengono ora eseguiti come processi software nel sistema operativo Linux. Cisco IOS® funziona come daemon nel sistema operativo Linux (IOSd). Cisco IOS® XE conserva non solo lo stesso aspetto del classico Cisco IOS®, ma anche il suo funzionamento, supporto e gestione.

Inoltre, il documento introduce diversi nuovi comandi su questa piattaforma che sono integrali per risolvere i problemi di utilizzo della CPU.

Di seguito sono riportate alcune definizioni utili:

- Driver del motore di inoltro (FED): questo è il cuore dello switch Cisco Catalyst serie 3850 ed è responsabile di tutta la programmazione/inoltro hardware.

- Cisco IOSd®: questo è il daemon Cisco IOS® in esecuzione sul kernel Linux. Viene eseguito come processo software all'interno del kernel.

- Packet Delivery System (PDS): si tratta dell'architettura e del processo con cui i pacchetti vengono consegnati da e verso vari sottosistemi. Ad esempio, controlla il modo in cui i pacchetti vengono consegnati dalla FED all'IOSd e viceversa.

- Handle: un handle può essere considerato come un puntatore. È un mezzo per scoprire informazioni più dettagliate sulle variabili specifiche utilizzate negli output prodotti dalla casella. Questa procedura è simile al concetto di indici LTL (Local Target Logic) sugli switch Cisco Catalyst serie 6500.

Caso di studio: Address Resolution Protocol (ARP) Interrupts

Il processo di risoluzione dei problemi e di verifica descritto in questa sezione può essere ampiamente utilizzato per un elevato utilizzo della CPU a causa di interruzioni.

Passaggio 1: Identificare il processo che utilizza i cicli della CPU

Il comando **show process cpu** visualizza naturalmente l'aspetto corrente della CPU. Notare che lo switch Cisco Catalyst serie 3850 utilizza quattro core e l'utilizzo della CPU è elencato per tutti e quattro i core:

```
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
```

Dall'output, è chiaro che il daemon Cisco IOS® consuma una porzione maggiore della CPU insieme al FED, che è il cuore di questo dispositivo. Quando l'utilizzo della CPU è elevato a causa di interruzioni, Cisco IOSd® e FED utilizzano la maggior parte della CPU e questi sottoprocessi (o un sottoinsieme di essi) utilizzano la CPU:

- FED Punject TX
- FED Punject RX
- Rifornimento perforato FED
- Trasmissione puntiforme FED completata

È possibile eseguire lo zoom avanti su uno qualsiasi di questi processi con il comando **show process cpu detailed <*process*>**. Dal momento che Cisco IOSd® è responsabile della maggior parte dell'utilizzo della CPU, di seguito è riportata un'analisi più approfondita di questa caratteristica.

```
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
```

L'output (Cisco IOSd® CPU output) mostra che ARP Snoop, IP Host Track Process e ARP Input sono elevati. Questa condizione si verifica in genere quando la CPU viene interrotta a causa di pacchetti ARP.

Passaggio 2: Determinare la coda di CPU che causa la condizione di utilizzo elevato della CPU

Lo switch Cisco Catalyst serie 3850 ha un numero di code che rispondono a diversi tipi di pacchetti (il FED mantiene 32 code di CPU RX, ossia code che vanno direttamente alla CPU). È importante monitorare queste code per individuare i pacchetti indirizzati alla CPU e quelli elaborati da Cisco IOSd®. Queste code sono per ASIC.

Nota: sono disponibili due ASIC: 0 e 1. Le porte da 1 a 24 appartengono all'ASIC 0.

Per esaminare le code, immettere il comando **show platform punt statistics port-asic <*port-asic*> cpuq <*queue*> direction <*rx*|*tx*>**.

Nel comando **show platform punt statistics port-asic 0 cpuq -1 direction rx**, l'argomento -1 elenca tutte le code. Pertanto, questo comando elenca tutte le code di ricezione per Port-ASIC 0.

A questo punto, è necessario identificare la coda che invia un numero elevato di pacchetti a una velocità elevata. In questo esempio, un'analisi delle code ha rivelato questo colpevole:

```
<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>
```

Il numero di coda è 16 e il nome della coda è CPU_Q_PROTO_SNOOPING.

Un altro modo per scoprire la coda colpevole è immettere il comando **show platform punt client**.

```
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
```

Determinare il tag per il quale sono stati allocati la maggior parte dei pacchetti. In questo esempio, questo valore è 65561.

Immettere quindi questo comando:

```
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
```

Questo output mostra che la coda è Rx Proto Snoop.

Il valore s prima dello 65561 nell'output del comando **show platform punt client** indica che l'handle FED viene sospeso e sopraffatto dal numero di pacchetti in arrivo. Se la s non scompare, la coda è bloccata in modo permanente.

Passaggio 3: Scaricare il pacchetto inviato alla CPU

Nei risultati del comando **show pds tag all**, si noti che accanto allo snoop Punt Rx Proto viene visualizzato un handle, 7296672.

Usare questo handle nel comando **show pds client <*handle*> packet last sink**. Si noti che è necessario abilitare debug pds pktbuf-last prima di utilizzare il comando. In caso contrario, viene visualizzato il seguente errore:

```
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
```

Se il debug è abilitato, viene visualizzato questo output:

```
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 00 02 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              ............
```

Con questo comando viene eseguito il dump dell'ultimo pacchetto ricevuto dal sink, che in questo esempio è IOSd. Ciò mostra che scarica l'intestazione e può essere decodificata con Wireshark (TShark) basato su terminale. I metadati sono destinati all'uso interno da parte del sistema, ma l'output dei dati fornisce informazioni effettive sul pacchetto. I metadata, tuttavia, rimangono estremamente utili.

La linea che inizia con 0070. Usare i primi 16 bit successivi, come mostrato di seguito:

<#root>

```
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'interfaccia colpevole è identificata qui. Gig2/0/20 è dove c'è un generatore di traffico che pompa il traffico ARP. Se si arresta il sistema, il problema verrà risolto e l'utilizzo della CPU verrà ridotto al minimo.

Passaggio 4: Utilizzare la traccia FED

L'unico inconveniente del metodo illustrato nell'ultima sezione è che viene scaricato solo l'ultimo pacchetto che viene inserito nel sink, e non può esserne il responsabile.

Un modo migliore per risolvere il problema consiste nell'utilizzare una funzione denominata tracciamento FED. Il trace è un metodo di acquisizione dei pacchetti (utilizzando vari filtri) che viene eseguito dal FED sulla CPU. Tuttavia, il trace FED non è semplice come la funzione Netdr sugli switch Cisco Catalyst serie 6500.

Il processo è suddiviso in passaggi:

1. Abilitare il rilevamento dei dettagli. Per impostazione predefinita, la traccia degli eventi è attivata. Per acquisire i pacchetti effettivi, è necessario abilitare la traccia dei dettagli:

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

2. Ottimizzare il buffer di acquisizione. Determinare la profondità dei buffer per la traccia dei dettagli e aumentare se necessario.

```
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
```

È possibile modificare le dimensioni del buffer con questo comando:

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

I valori disponibili sono:

```
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. Aggiungere i filtri di acquisizione. A questo punto è necessario aggiungere diversi filtri per l'acquisizione. È possibile aggiungere filtri diversi e scegliere se far corrispondere tutti i filtri o solo quelli per la cattura.

   I filtri vengono aggiunti con questo comando:

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

   Queste opzioni sono attualmente disponibili:

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

```
field      field
offset     offset
```

Ora bisogna collegare le cose. Ricordare la coda colpevole identificata nel passaggio 2 di questo processo di risoluzione dei problemi? Poiché la coda 16 è la coda che invia un numero elevato di pacchetti alla CPU, ha senso tracciare questa coda e vedere quali pacchetti vengono puntati alla CPU da essa.

È possibile scegliere di tracciare qualsiasi coda con questo comando:

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

Di seguito è riportato il comando per questo esempio:

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

È necessario scegliere una corrispondenza per tutti i filtri o una corrispondenza per tutti i filtri, quindi abilitare la traccia:

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

4. Visualizza pacchetti filtrati. È possibile visualizzare i pacchetti acquisiti con il comando **show mgmt-infra trace messages fed-punject-detail**.

```
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>
```

Questo output fornisce un'ampia quantità di informazioni e può in genere essere sufficiente per individuare l'origine e il contenuto dei pacchetti.

La prima parte del dump dell'intestazione è ancora una volta costituita dai metadati utilizzati dal sistema. La seconda parte è il pacchetto effettivo.

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

È possibile scegliere di tracciare questo indirizzo MAC di origine per trovare la porta colpevole (una volta identificato che questa è la maggior parte dei pacchetti che vengono puntualizzati dalla coda 16; questo output mostra solo un'istanza del pacchetto e gli altri output/pacchetti vengono ritagliati).

Tuttavia, c'è un modo migliore. Si noti che i log presenti dopo le informazioni dell'intestazione:

```
[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
```

Il primo log indica chiaramente da quale coda e tag proviene il pacchetto. Se non si era a conoscenza della coda in precedenza, questo è un modo semplice per identificare la coda che era.

Il secondo log è ancora più utile in quanto fornisce l'ID IIF (Interface ID Factory)-ID fisico per l'interfaccia di origine. Il valore esadecimale è un handle che può essere utilizzato per eseguire il dump delle informazioni relative alla porta:

```
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
        Active            : 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
```

Ancora una volta avete identificato l'interfaccia di origine e la causa del guasto.

La funzione di trace è un potente strumento fondamentale per la risoluzione dei problemi relativi all'utilizzo elevato della CPU e in grado di fornire numerose informazioni per risolvere correttamente questa situazione.

Esempio di script EEM (Embedded Event Manager) per gli switch Cisco Catalyst serie 3850

Utilizzare questo comando per attivare la generazione di un registro a una determinata soglia:

```
process cpu threshold type total rising


        interval


switch
```

Il log generato con il comando ha il seguente aspetto:

```
*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
```

Il log generato fornisce le seguenti informazioni:

- Utilizzo totale della CPU al momento del trigger. Questo valore è identificato dall'utilizzo totale della CPU (total/Intr) :50/0 in questo esempio.
- Primi processi: elencati nel formato PID/CPU%. In questo esempio sono:

```
8622/25 - 8622 is PID for IOSd and 25 implies that this process is using 25% CPU.
5753/12 - 5733 is PID for FED and 12 implies that this process is using 12% CPU.
```

Di seguito è riportato lo script EEM:

```
event manager applet highcpu
event syslog pattern "%CPUMEM-5-RISING_THRESHOLD"
 action 0.1 syslog msg "high CPU detected"
 action 0.2 cli command "enable"
 action 0.3 cli command "show process cpu sorted | append nvram:<filename>.txt"
 action 0.4 cli command "show process cpu detailed process <process name|process ID>
 sorted | nvram:<filename>.txt"
 action 0.5 cli command "show platform punt statistics port-asic 0 cpuq -1
 direction rx | append nvram:<filename>.txt"
 action 0.6 cli command "show platform punt statistics port-asic 1 cpuq -1
 direction rx | append nvram:<filename>.txt"
 action 0.7 cli command "conf t"
 action 0.8 cli command "no event manager applet highcpu"
```

Nota: il comando **process cpu threshold** non funziona attualmente nel treno 3.2.X. Un altro aspetto da tenere presente è che questo comando analizza l'utilizzo medio della CPU tra i quattro core e genera un registro quando questa media raggiunge la percentuale definita nel comando.

Cisco IOS-XE 16.x o versioni successive

Se sugli switch Catalyst 3850 è in esecuzione il software Cisco® IOS-XE versione 16.x o successive, vedere Risoluzione dei problemi di utilizzo elevato della CPU sulle piattaforme degli switch Catalyst con IOS-XE 16.x.

Informazioni correlate

- **Cos'è Cisco IOS XE?**
- **Switch Cisco Catalyst 3850 - Data sheet e documentazione**
- **Documentazione e supporto tecnico â€“ Cisco Systems**

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).