Troubleshoot CPU Usage in Catalyst Switches on Cisco IOS XE 16.x

Contents

Introduction

This document describes how to troubleshoot high CPU usage due to interruptions on Cisco IOS® XE platforms running the 16.x releases.

Background Information

This document was contributed by Raymond Whiting and Yogesh Ramdoss, Cisco TAC Engineers.

This document also introduces several new commands on this platform that are integral in order to troubleshoot the high CPU usage concerns. It is important to understand how Cisco IOS XE is built. With Cisco IOS XE, Cisco has moved to a Linux kernel and all of the subsystems have been broken down into processes. All of the subsystems that were inside Cisco IOS earlier - such as the modules drivers, High Availability (HA), and so on - now run as software processes within the Linux Operating System (OS). Cisco IOS itself runs as a daemon within the Linux OS (IOSd). Cisco IOS XE retains not only the same look and feel of the classic Cisco IOS, but also its operation, support, and management.

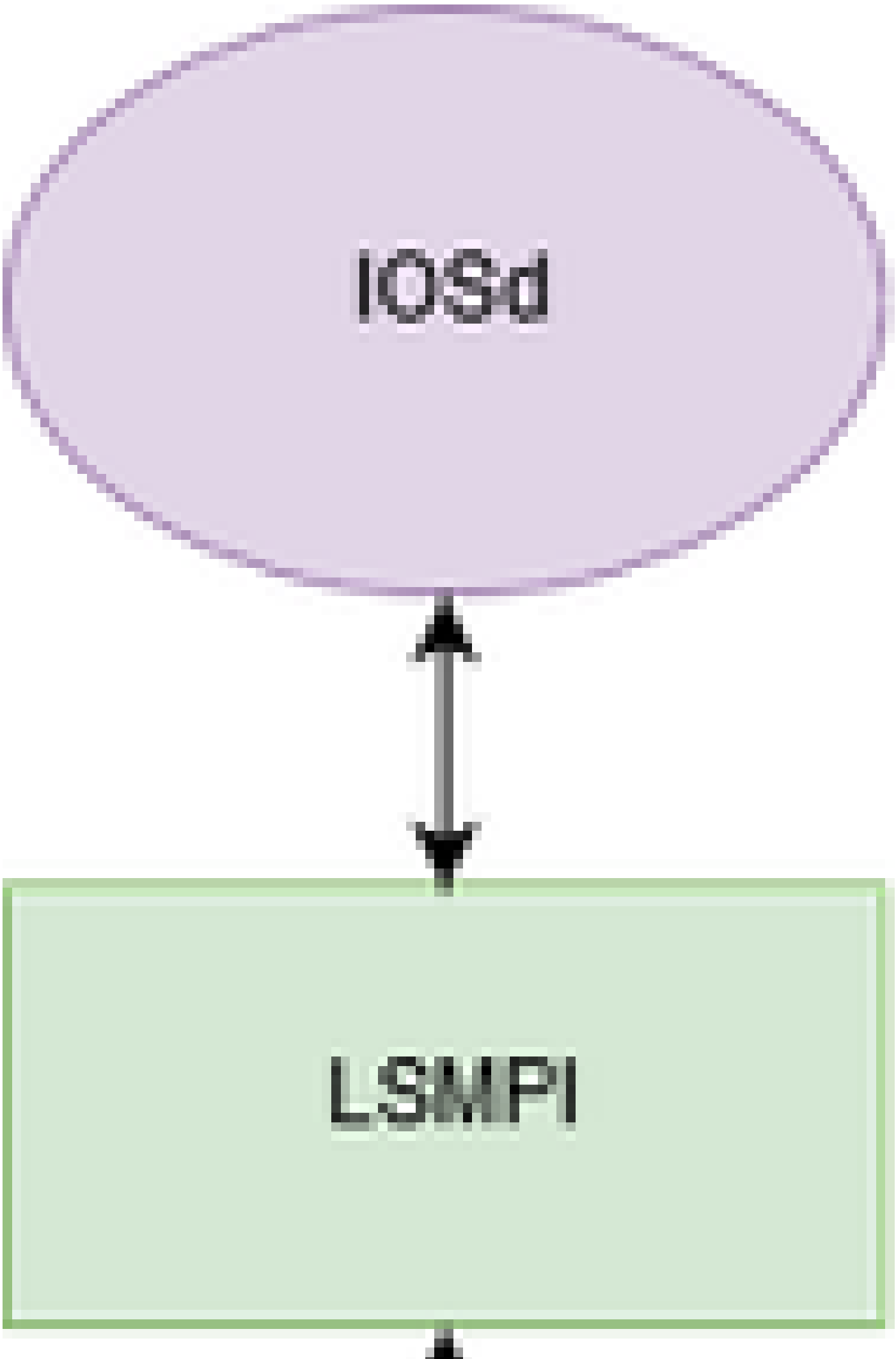
Here are some useful definitions:

- Forwarding Engine Driver (FED): This is the heart of the Cisco Catalyst switch and is responsible for all hardware programming/forwarding.
- IOSd: This is the Cisco IOS daemon that runs on the Linux kernel. It is run as a software process within the kernel.
- Packet Delivery System (PDS): This is the architecture and process of how packets are delivered to and from the various subsystems. As an example, it controls how packets are delivered from the FED to the IOSd and vice versa.
- Control Plane (CP): The control plane is a generic term used to group the functions and traffic that involve the CPU of the Catalyst Switch. This includes traffic such as Spanning Tree Protocol (STP), Hot Standby Router Protocol (HSRP), and routing protocols that are destined to the switch, or sent from the switch. This also includes application layer protocols like Secure Shell (SSH), and Simple Network Management Protocol (SNMP) that must be handled by the CPU.
- Data Plane (DP): Typically the data plane encompasses the hardware ASICs and traffic that is

forwarded without assistance from the Control Plane.

- Punt: Ingress protocol control packet intercepted by DP sent to the CP in order to process it.
- Inject: CP generated protocol packet sent to DP in order to egress out on IO interface(s).
- LSMPI: Linux Shared Memory Punt Interface.

A high-level diagram of the communication path between the Data Plane and Control Plane:


IOSd
LSWPI

command is used in order to display the current process state inside of the IOSd daemon. When you add the output modify | exclude 0.00, it filters out the processes that are currently idle.

There are two valuable pieces of information from this output:

- CPU utilization for five seconds: 91%/30%
  - The first number (91%) is the overall CPU utilization of the switch
  - The second number (30%) is the utilization caused by interrupts from the data plane
- The Address Resolution Protocol (ARP) Input process is currently the top Cisco IOS process that consumes the resources:

```
<#root>

Switch#

show processes cpu sort | ex 0.00


CPU utilization for five seconds:

91%/30%

; one minute: 30%; five minutes: 8%
 PID Runtime(ms)     Invoked      uSecs   5Sec   1Min   5Min TTY Process
  37       14645         325

45061 59.53% 18.86%  4.38%   0 ARP Input


 137        2288         115      19895  1.20%  0.14%  0.07%   0 Per-minute Jobs
 373        2626       35334         74  0.15%  0.11%  0.09%   0 MMA DB TIMER
 218        3123       69739         44  0.07%  0.09%  0.12%   0 IP ARP Retry Age
 404        2656       35333         75  0.07%  0.09%  0.09%   0 MMA DP TIMER
```

The show processes cpu platform sorted command is used to display what the process utilization from the Linux kernel looks like. From the output, it can be observed that the FED process is high, which is due to the ARP requests punted to the IOSd process:

```
<#root>

Switch#

show processes cpu platform sorted

CPU utilization for five seconds: 38%, one minute: 38%, five minutes: 40%
Core 0: CPU utilization for five seconds: 39%, one minute: 37%, five minutes: 39%
Core 1: CPU utilization for five seconds: 41%, one minute: 38%, five minutes: 40%
Core 2: CPU utilization for five seconds: 30%, one minute: 38%, five minutes: 40%
Core 3: CPU utilization for five seconds: 37%, one minute: 39%, five minutes: 41%
   Pid    PPid    5Sec    1Min    5Min  Status        Size  Name
-------------------------------------------------------------------------------
 22701   22439     89%     88%

88%  R       2187444224  linux_iosd-imag

 11626   11064     46%     47%

48%  S       2476175360  fed main event
```

```
  4585       2      7%      9%      9%  S                0  lsmpi-xmit
  4586       2      3%      6%      6%  S                0  lsmpi-rx
```

Step 2. Investigate Why FED is Punting Packets to the Control Plane

From Step 1. You can conclude that the IOSd/ARP process runs high but is the victim of traffic that is introduced from the Data Plane. Further investigation as to why the FED process punts traffic to the CPU and where this traffic is coming from is needed.

The show platform software fed switch active punt cause summary gives a high-level overview of the punt reason. Any number that increments over multiple runs of this command indicates:

<#root>

Switch#

show platform software fed switch active punt cause summary

Statistics for all causes

Cause	Cause Info	Rcvd	Dropped
7	**ARP request or response**	**18444227**	**0**
11	For-us data	16	0
21	RP<->QFP keepalive	3367	0
24	Glean adjacency	2	0
55	For-us control	6787	0
60	IP subnet or broadcast packet	14	0
96	Layer2 control protocols	3548	0

Packets that are sent to the control plane from FED use a split queue structure in order to guarantee high-priority control traffic. It does not get lost behind lower-priority traffic, like ARP. A high-level overview of these queues can be viewed with the use of the show platform software fed switch active cpu-interface. After you run this command several times, it can be found that the Forus Resolution (Forus - which means traffic destined to the CPU) queue increments quickly.

<#root>

Switch#

show platform software fed switch active cpu-interface

queue	retrieved	dropped	invalid	hol-block
Routing Protocol	8182	0	0	0
L2 Protocol	161	0	0	0
sw forwarding	2	0	0	0
broadcast	14	0	0	0
icmp gen	0	0	0	0
icmp redirect	0	0	0	0
logging	0	0	0	0
rpf-fail	0	0	0	0
DOT1X authentication	0	0	0	0

```
Forus Traffic              16          0           0           0

Forus Resolution           24097779    0           0           0

Inter FED                  0           0           0           0
L2 LVX control             0           0           0           0
EWLC control               0           0           0           0
EWLC data                  0           0           0           0
L2 LVX data                0           0           0           0
Learning cache             0           0           0           0
Topology control           4117        0           0           0
Proto snooping             0           0           0           0
DHCP snooping              0           0           0           0
Transit Traffic            0           0           0           0
Multi End station          0           0           0           0
Webauth                    0           0           0           0
Crypto control             0           0           0           0
Exception                  0           0           0           0
General Punt               0           0           0           0
NFL sampled data           0           0           0           0
Low latency                0           0           0           0
EGR exception              0           0           0           0
FSS                        0           0           0           0
Multicast data             0           0           0           0
Gold packet                0           0           0           0
```

The use of the show platform software fed switch active punt cpuq all gives a more detailed view of these queues. Queue 5 is responsible for ARP, and as expected it increments across multiple runs of the command. The show plat soft fed sw active inject cpuq clear command can be used in order to clear the counters for easier reading.

<#root>

```
Switch#

show platform software fed switch active punt cpuq all

<snip>
CPU Q Id                     : 5

CPU Q Name                   : CPU_Q_FORUS_ADDR_RESOLUTION

Packets received from ASIC   : 21018219
Send to IOSd total attempts  : 21018219

Send to IOSd failed count    : 0
RX suspend count             : 0
RX unsuspend count           : 0
RX unsuspend send count      : 0
RX unsuspend send failed count : 0
RX consumed count            : 0
RX dropped count             : 0
RX non-active dropped count  : 0
RX conversion failure dropped  : 0
RX INTACK count              : 1050215
RX packets dq'd after intack : 90
Active RxQ event             : 3677400
RX spurious interrupt        : 1050016
<snip>
```

From here, there are a couple of options. ARP is broadcast traffic, so you can look for interfaces that have an abnormally high rate of broadcast traffic (also useful to troubleshoot Layer 2 loops). It is necessary to run this command multiple times in order to determine what interface actively increments.

```
<#root>

Switch#

show interfaces counters


Port            InOctets    InUcastPkts    InMcastPkts

InBcastPkts

Gi1/0/1    1041141009678              9              0

16267828358

Gi1/0/2             1254             11              0              1
Gi1/0/3                0              0              0              0
Gi1/0/4                0              0              0              0
```

The other option is to use the Embedded Packet Capture (EPC) tool in order to gather a sample of the packets that are seen at the control plane.

```
<#root>

Switch#

monitor capture cpuCap control-plane in match any file location flash:cpuCap.pcap

Switch#

show monitor capture cpuCap


Status Information for Capture cpuCap
  Target Type:
 Interface: Control Plane, Direction: IN
   Status : Inactive
  Filter Details:
    Capture all packets
  Buffer Details:
   Buffer Type: LINEAR (default)
  File Details:
   Associated file name: flash:cpuCap.pcap
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

This command configures an internal capture on the switch in order to capture any traffic that is punted to the control plane. This traffic is saved to a file on the flash. This is a normal wireshark pcap file that can be exported off a switch and opened in Wireshark for further analysis.

Start the capture, let it run for a few seconds, and stop the capture:

```
<#root>

Switch#

monitor capture cpuCap start

Enabling Control plane capture may seriously impact system performance. Do you want to continue? [yes/n

yes

Started capture point : cpuCap
*Jun 14 17:57:43.172: %BUFCAP-6-ENABLE: Capture Point cpuCap enabled.
Switch#

monitor capture cpuCap stop

Capture statistics collected at software:
        Capture duration - 59 seconds


Packets received - 215950

        Packets dropped - 0
        Packets oversized - 0

Bytes dropped in asic - 0

Stopped capture point : cpuCap
Switch#
*Jun 14 17:58:37.884: %BUFCAP-6-DISABLE: Capture Point cpuCap disabled.
```

It is also possible to view the capture file on the switch:

```
<#root>

Switch#

show monitor capture file flash:cpuCap.pcap

Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1   0.000000 Xerox_d7:67:a1 -> Broadcast    ARP 60 Who has 192.168.1.24? Tell 192.168.1.2
  2   0.000054 Xerox_d7:67:a1 -> Broadcast    ARP 60 Who has 192.168.1.24? Tell 192.168.1.2
  3   0.000082 Xerox_d7:67:a1 -> Broadcast    ARP 60 Who has 192.168.1.24? Tell 192.168.1.2
  4   0.000109 Xerox_d7:67:a1 -> Broadcast    ARP 60 Who has 192.168.1.24? Tell 192.168.1.2
  5   0.000136 Xerox_d7:67:a1 -> Broadcast    ARP 60 Who has 192.168.1.24? Tell 192.168.1.2
  6   0.000162 Xerox_d7:67:a1 -> Broadcast    ARP 60 Who has 192.168.1.24? Tell 192.168.1.2
  7   0.000188 Xerox_d7:67:a1 -> Broadcast    ARP 60 Who has 192.168.1.24? Tell 192.168.1.2
  8   0.000214 Xerox_d7:67:a1 -> Broadcast    ARP 60 Who has 192.168.1.24? Tell 192.168.1.2
  9   0.000241 Xerox_d7:67:a1 -> Broadcast    ARP 60 Who has 192.168.1.24? Tell 192.168.1.2
```

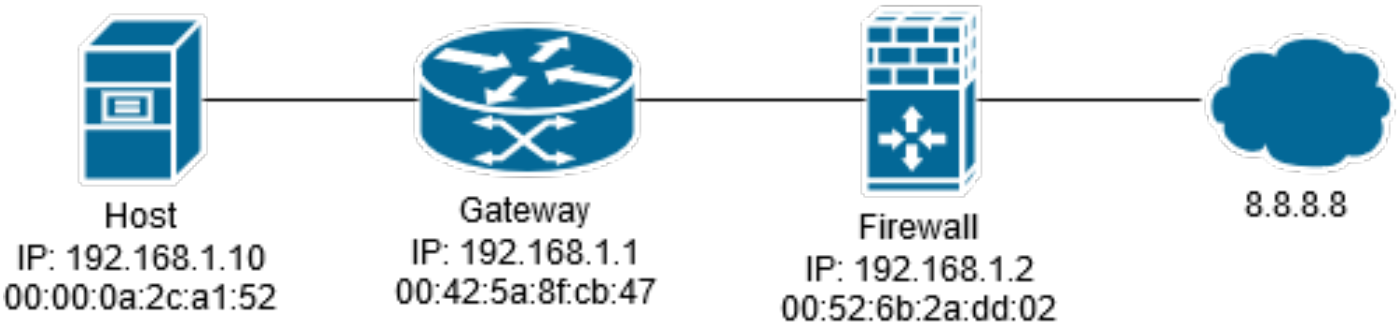
From this output, it is evident that the 192.168.1.2 host is the source of the constant ARPs that cause the high CPU on the switch. Use the show ip arp and show mac address-table address commands in order to track down the host and either remove it from the network or address the ARPs. It is also possible to get full details of each packet captured with the use of the detail option on the capture view command, show monitor capture file flash:cpuCap.pcap

detail. Refer to this guide for more information on packet captures on a Catalyst Switch.

Case Study 2. IP Redirects with CoPP

The latest generation Catalyst Switches are protected by Control Plane Policing (CoPP) by default. CoPP is used in order to protect the CPU from malicious attacks, and misconfigurations, that can jeopardize the ability of the switches in order to maintain critical functions such as spanning trees and routing protocols. These protections can lead to scenarios where the switch has only a slightly elevated CPU and clear interface counters, but traffic is dropped while traversing the switch. It is important to note the baseline CPU utilization on your device at the time of normal operations. It is not necessarily an issue to have elevated CPU utilization, and it depends on the features enabled on the device, but when this utilization increases without configuration changes, this can be a sign of concern.

Consider this scenario - hosts that live off of the Gateway switch report slow download speeds and ping loss to the internet. A general health check of the switch shows no errors on the interfaces or any ping loss when sourced from the gateway switch.



When you check the CPU, it shows slightly elevated numbers due to interrupts.

```
<#root>

Switch#

show processes cpu sorted | ex 0.00

CPU utilization for five seconds:

 8%/7%

; one minute: 8%; five minutes: 8%
 PID Runtime(ms)     Invoked      uSecs   5Sec   1Min   5Min TTY Process
 122      913359     1990893        458  0.39%  1.29%  1.57%   0

IOSXE-RP Punt

 Se
 147        5823       16416        354  0.07%  0.05%  0.06%   0 PLFM-MGR IPC pro
 404       13237      183032         72  0.07%  0.08%  0.07%   0 MMA DP TIMER
```

When you check the CPU interface, you see that the ICMP redirect counter is actively incremented.

```
<#root>

Switch#
```

show platform software fed switch active cpu-interface

queue	retrieved	dropped	invalid	hol-block
Routing Protocol	12175	0	0	0
L2 Protocol	236	0	0	0
sw forwarding	714673	0	0	0
broadcast	2	0	0	0
icmp gen	0	0	0	0
icmp redirect	**2662788**	**0**	**0**	**0**
logging	7	0	0	0
rpf-fail	0	0	0	0
DOT1X authentication	0	0	0	0
Forus Traffic	21776434	0	0	0
Forus Resolution	724021	0	0	0
Inter FED	0	0	0	0
L2 LVX control	0	0	0	0
EWLC control	0	0	0	0
EWLC data	0	0	0	0
L2 LVX data	0	0	0	0
Learning cache	0	0	0	0
Topology control	6122	0	0	0
Proto snooping	0	0	0	0
DHCP snooping	0	0	0	0
Transit Traffic	0	0	0	0

While no drops are observed in FED, if you check CoPP, drops can be observed in the ICMP Redirect queue.

<#root>

Switch#

show platform hardware fed switch 1 qos queue stats internal cpu policer

CPU Queue Statistics

				(default)	(set)	
Queue						
QId	PlcIdx	Queue Name	Enabled	Rate	Rate	
Drop(Bytes)						
0	11	DOT1X Auth	Yes	1000	1000	0
1	1	L2 Control	Yes	2000	2000	0
2	14	Forus traffic	Yes	4000	4000	0
3	0	ICMP GEN	Yes	600	600	0
4	2	Routing Control	Yes	5400	5400	0
5	14	Forus Address resolution	Yes	4000	4000	0
6	**0**	**ICMP Redirect**	**Yes**	**600**	**600**	**463538463**
7	16	Inter FED Traffic	Yes	2000	2000	0
8	4	L2 LVX Cont Pack	Yes	1000	1000	0

<snip>

CoPP is essentially a QoS policy put on the control plane of the device. CoPP works just like any other QoS on the switch: when the queue for a specific traffic is exhausted, the traffic that uses that queue is dropped. From these outputs, you know that traffic is being software switched because of ICMP redirects, and you know that this traffic is being dropped because of the rate limit on the ICMP Redirect queue. You can accomplish a capture on the control plane in order to validate that the packets that hit the control plane are from the users.

In order to see what matching logic each class uses, you have a CLI in order to help identify packet types that hit a certain queue. Consider this example, in order to know what hits the system-cpp-routing-control class:

```
<#root>

Switch#

show platform software qos copp policy-info

Default rates of all classmaps are displayed:
 policy-map system-cpp-policy
 class system-cpp-police-routing-control
 police rate 5400 pps

Switch#

show platform software qos copp class-info

ACL representable classmap filters are displayed:

class-map match-any system-cpp-police-routing-control


 description Routing control and Low Latency
 match access-group name system-cpp-mac-match-routing-control
 match access-group name system-cpp-ipv4-match-routing-control
 match access-group name system-cpp-ipv6-match-routing-control
 match access-group name system-cpp-ipv4-match-low-latency
 match access-group name system-cpp-ipv6-match-low-latency



mac access-list extended system-cpp-mac-match-routing-control
 permit any host 0180.C200.0014
 permit any host 0900.2B00.0004
 ip access-list extended system-cpp-ipv4-match-routing-control
 permit udp any any eq rip
 <...snip...>
 ipv6 access-list system-cpp-ipv6-match-routing-control
 permit ipv6 any FF02::1:FF00:0/104
 permit ipv6 any host FF01::1
<...snip...>
 ip access-list extended system-cpp-ipv4-match-low-latency
 permit udp any any eq 3784
 permit udp any any eq 3785
 ipv6 access-list system-cpp-ipv6-match-low-latency
 permit udp any any eq 3784
 permit udp any any eq 3785
 <...snip...>



<#root>
```

```
Switch#
monitor capture cpuSPan control-plane in match any file location flash:cpuCap.pcap

Control-plane direction IN is already attached to the capture
Switch#
monitor capture cpuSpan start


Enabling Control plane capture may seriously impact system performance. Do you want to continue? [yes/n
Started capture point : cpuSpan
Switch#
*Jun 15 17:28:52.841: %BUFCAP-6-ENABLE: Capture Point cpuSpan enabled.
Switch#
monitor capture cpuSpan stop

Capture statistics collected at software:
        Capture duration - 12 seconds
        Packets received - 5751
        Packets dropped - 0
        Packets oversized - 0

Bytes dropped in asic - 0

Stopped capture point : cpuSpan
Switch#
*Jun 15 17:29:02.415: %BUFCAP-6-DISABLE: Capture Point cpuSpan disabled.
Switch#
show monitor capture file flash:cpuCap.pcap detailed

Starting the packet display ........ Press Ctrl + Shift + 6 to exit

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
<snip>
Ethernet II, Src: OmronTat_2c:a1:52 (00:00:0a:2c:a1:52), Dst: Cisco_8f:cb:47 (00:42:5a:8f:cb:47)


<snip>
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 8.8.8.8



<snip>
```

When this host pings 8.8.8.8, it sends the ping to the MAC address of the gateway, as the destination address is outside of the VLAN. The gateway switch detects that the next hop is in the same VLAN, rewrites the destination MAC address to the firewall, and forwards the packet. This process can occur in hardware, but an exception to this hardware forwarding is the IP redirect process. When the switch receives the ping, it detects that it is routing traffic on the same VLAN and punts the traffic to the CPU in order to generate a redirect packet back to the host. This redirect message is to inform the host that there is a more optimal path to the destination. In this instance the layer 2 next-hop is by design and expected, the switch must be configured not to send the redirect messages and forward the packets in hardware. This is done when you disable the redirects on the VLAN interface.

```
interface Vlan1
 ip address 192.168.1.1 255.255.255.0


no ip redirects

end
```

When IP redirects are turned off, the switch rewrites the MAC address and forwards in the hardware.

Case Study 3. Intermittent High CPU

In the event that the high CPU on the switch is intermittent, it is possible to set up a script on the switch in order to automatically run these commands at the time of high CPU events. This is done with the use of the Cisco IOS Embedded Event Manager (EEM).

The entry-val is used in order to determine how high the CPU is before the script triggers. The script monitors the 5-second CPU average SNMP OID. Two files are written to the flash, tac-cpu-<timestamp>.txt contains the command outputs, and tac-cpu-<timestamp>.pcap contains the CPU ingress capture. These files can then be reviewed at a later date.

```
config t
no event manager applet high-cpu authorization bypass
event manager applet high-cpu authorization bypass
event snmp oid 1.3.6.1.4.1.9.9.109.1.1.1.1.3.1 get-type next entry-op gt entry-val 80 poll-interval 1 r
action 0.01 syslog msg "High CPU detected, gathering system information."
action 0.02 cli command "enable"
action 0.03 cli command "term exec prompt timestamp"
action 0.04 cli command "term length 0"
action 0.05 cli command "show clock"
action 0.06 regex "([0-9]|[0-9][0-9]):([0-9]|[0-9][0-9]):([0-9]|[0-9][0-9])" $_cli_result match match1
action 0.07 string replace "$match" 2 2 "."
action 0.08 string replace "$_string_result" 5 5 "."
action 0.09 set time $_string_result
action 1.01 cli command "show proc cpu sort | append flash:tac-cpu-$time.txt"
action 1.02 cli command "show proc cpu hist | append flash:tac-cpu-$time.txt"
action 1.03 cli command "show proc cpu platform sorted | append flash:tac-cpu-$time.txt"
action 1.04 cli command "show interface | append flash:tac-cpu-$time.txt"
action 1.05 cli command "show interface stats | append flash:tac-cpu-$time.txt"
action 1.06 cli command "show log | append flash:tac-cpu-$time.txt"
action 1.07 cli command "show ip traffic | append flash:tac-cpu-$time.txt"
action 1.08 cli command "show users | append flash:tac-cpu-$time.txt"
action 1.09 cli command "show platform software fed switch active punt cause summary | append flash:tac
action 1.10 cli command "show platform software fed switch active cpu-interface | append flash:tac-cpu-
action 1.11 cli command "show platform software fed switch active punt cpuq all | append flash:tac-cpu-
action 2.08 cli command "no monitor capture tac_cpu"
action 2.09 cli command "monitor capture tac_cpu control-plane in match any file location flash:tac-cpu
action 2.10 cli command "monitor capture tac_cpu start" pattern "yes"
action 2.11 cli command "yes"
action 2.12 wait 10
action 2.13 cli command "monitor capture tac_cpu stop"
action 3.01 cli command "term default length"
action 3.02 cli command "terminal no exec prompt timestamp"
action 3.03 cli command "no monitor capture tac_cpu"
```

Related Information

- Cisco IOS XE 16 At a Glance
- Catalyst 3850 Series Switch High CPU Usage Troubleshoot
- Embedded Packet Capture for Cisco IOS and Cisco IOS XE Configuration Example
- Configure Punt/Inject FED Packet Capture on Catalyst Switches Running Cisco IOS XE 16.x
- Cisco Technical Support & Download