Troubleshoot Control Plane Operations on Catalyst 9000 Switches

Contents

- Introduction
- Background Information
  - Terminology
- Catalyst 9000 CoPP
  - CoPP Implementation
  - Default Policy
  - Adjust CoPP
- Troubleshoot
  - Methodology
  - Useful Show Commands
    - Determine Overall and Historical Utilization
    - Check for Control Plane Policing
    - Gather information on punted traffic
    - Inspect CPU bound traffic
- Common Scenarios
  - Intermittent ICMP (Ping) loss to local IP
  - High ICMP redirects and Sluggish DHCP operation
- Additional Resources

Introduction

This document describes how to troubleshoot and validate control plane health on Catalyst 9000-family switches that run Cisco IOS® XE.

Background Information

The primary job of a switch is to forward packets as quickly as possible. Most packets are forwarded in hardware but certain types of traffic must be handled by the system CPU. Traffic that arrives at the CPU is handled as quickly as possible. A certain amount of traffic is expected to be seen at the CPU, but an overabundance leads to operational problems. The Catalyst 9000 family of switches incorporates a robust Control Plane Policing (CoPP) mechanism by default to prevent problems caused by traffic oversaturation of the CPU.

Unexepected problems arise in certain use-cases as a function of normal operation. The correlation between cause and effect is not obvious sometimes, which makes the problem difficult to approach. This document provides you with tools to validate control plane health, and provides a workflow on how to approach problems that involve the control plane punt or inject path. It also provides several common scenarios based on problems seen in the field.

Keep in mind that the CPU punt path is a limited resource. Modern hardware-forwarding switches can handle exponentially greater volume of traffic. The Catalyst 9000 family of switches support approximately 19,000 packets-per-second (pps) in aggregate at the CPU at any given time. Exceed this threshold, and punted traffic is policed without weight.

Terminology

- **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 together the functions and traffic that involve the CPU of the Catalyst Switch. This includes traffic such asSpanning 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), andSimple 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 which intercepted on DP sent to the CP to process it
- **Inject:**CP generated protocol packet sent to DP to egress out on IO interface(s)
- **LSMPI:**Linux Shared Memory Punt Interface

Catalyst 9000 CoPP

The foundation of CPU protection on the Catalyst 9000 family of switches is CoPP. With CoPP, a system-generated Quality of Service (QoS) policy is applied on the CPU punt/inject path. CPU-bound traffic is grouped into many different classes, and subsequently mapped across the individual hardware policers associated with the CPU. The policers prevent oversaturation of the CPU by a particular class of traffic.

CoPP Implementation

CPU-bound traffic is classified into queues. These queues/classes are system-defined and are not user-configurable. Policers are configured in hardware. The Catalyst 9000 family supports 32 hardware policers for 32 queues.

Specific values differ from platform to platform. In general, there are 32 system-defined queues. These queues relate to class-maps, which relate to policer indices. The policer indices have a default policer rate. This rate is user-configurable, though changes to the default CoPP policy increases susceptibility to an unexpected service impact.

System-Defined Values for CoPP

Class Maps Names	Policer Index (Policer No.)	CPU queues (Queue No.)
system-cpp-police-data	WK_CPP_POLICE_DATA(0)	WK_CPU_Q_ICMP_GEN(3) WK_CPU_Q_BROADCAST(12)

Class Maps Names	Policer Index (Policer No.)	CPU queues (Queue No.)
		WK_CPU_Q_ICMP_REDIRECT(6)
system-cpp-police-l2-control	WK_CPP_POLICE_L2_ CONTROL(1)	WK_CPU_Q_L2_CONTROL(1)
system-cpp-police-routing-control	WK_CPP_POLICE_ROUTING_CONTROL(2)	WK_CPU_Q_ROUTING_CONTROL(4) WK_CPU_Q_LOW_LATENCY (27)
system-cpp-police-control-low-priority	WK_CPP_POLICE_CO NTROL_LOW_PRI(3)	WK_CPU_Q_GENERAL_PUNT(25)
system-cpp-police-punt-webauth	WK_CPP_POLICE_PU NT_WEBAUTH(7)	WK_CPU_Q_PUNT_WEBAUTH(22)
system-cpp-police-topology-control	WK_CPP_POLICE_TOPOLOGY_CONTROL(8)	WK_CPU_Q_TOPOLOGY_CONTROL(15)
system-cpp-police-multicast	WK_CPP_POLICE_MULTICAST(9)	WK_CPU_Q_TRANSIT_TRAFFIC(18) WK_CPU_Q_MCAST_DATA(30)
system-cpp-police-sys- data	WK_CPP_POLICE_SYS _DATA(10)	WK_CPU_Q_LEARNING_CACHE_OVFL(13) WK_CPU_Q_CRYPTO_CONTROL(23) WK_CPU_Q_EXCEPTION(24) WK_CPU_Q_EGR_EXCEPTION(28) WK_CPU_Q_NFL_SAMPLED_DATA(26) WK_CPU_Q_GOLD_PKT(31) WK_CPU_Q_RPF_FAILED(19)
system-cpp-police-	WK_CPP_POLICE_DOT1X(11)	WK_CPU_Q_DOT1X_AUTH(0)

Class Maps Names	Policer Index (Policer No.)	CPU queues (Queue No.)
dot1x-auth		
system-cpp-police-protocol-snooping	WK_CPP_POLICE_PR(12)	WK_CPU_Q_PROTO_SNOOPING(16)
system-cpp-police-sw-forward	WK_CPP_POLICE_SW_FWD (13)	WK_CPU_Q_SW_FORWARDING_Q(14) WK_CPU_Q_LOGGING(21) WK_CPU_Q_L2_LVX_DATA_PACK(11)
system-cpp-police-forus	WK_CPP_POLICE_FORUS(14)	WK_CPU_Q_FORUS_ADDR_RESOLUTION(5) WK_CPU_Q_FORUS_TRAFFIC(2)
system-cpp-police-multicast-end-station	WK_CPP_POLICE_MULTICAST_SNOOPING(15)	WK_CPU_Q_MCAST_END_STATION_SERVICE(20)
system-cpp-default	WK_CPP_POLICE_DEFAULT_POLICER(16)	WK_CPU_Q_DHCP_SNOOPING(17) WK_CPU_Q_UNUSED(7) WK_CPU_Q_EWLC_CONTROL(9) WK_CPU_Q_EWLC_DATA(10)
system-cpp-police-stackwise-virt-control	WK_CPP_STACKWISE_VIRTUAL_CONTROL(5)	WK_CPU_Q_STACKWISE_VIRTUAL_CONTR (29)
system-cpp-police-l2lvx-control	WK_CPP_ L2_LVX_CONT_PACK(4)	WK_CPU_Q_L2_LVX_CONT_PACK(8)

Each queue relates to a traffic type or particular set of features. This is not an exhaustive list:

CPU Queues and Associated Feature(s)

CPU queues (Queue No.)	**Feature(s)**
WK_CPU_Q_DOT1X_AUTH(0)	IEEE 802.1x Port-Based Authentication
WK_CPU_Q_L2_CONTROL(1)	Dynamic Trunking Protocol (DTP) VLAN Trunking Protocol (VTP) Port Aggregation Protocol (PAgP) Client Information Signaling Protocol (CISP) Message session relay protocol Multiple VLAN Registration Protocol (MVRP) Metropolitan Mobile Network (MMN) Link Level Discovery Protocol (LLDP) UniDirectional Link Detection (UDLD) Link Aggregation Control Protocol (LACP) Cisco Discovery Protocol (CDP) Spanning Tree Protocol (STP)
WK_CPU_Q_FORUS_TRAFFIC(2)	Host such as Telnet,Pingv4 and Pingv6, and SNMP Keepalive / loopback detection Initiate-Internet Key Exchange (IKE) protocol (IPSec)
WK_CPU_Q_ICMP_GEN(3)	ICMP - destination unreachable ICMP-TTL expired
WK_CPU_Q_ROUTING_CONTROL(4)	Routing Information Protocol version 1 (RIPv1) RIPv2 Interior Gateway Routing Protocol (IGRP) Border Gateway Protocol (BGP) PIM-UDP Virtual Router Redundancy Protocol (VRRP)

CPU queues (Queue No.)	Feature(s)
	Hot Standby Router Protocol version 1 (HSRPv1) HSRPv2 Gateway Load Balancing Protocol (GLBP) Label Distribution Protocol (LDP) Web Cache Communication Protocol (WCCP) Routing Information Protocol next generation (RIPng) Open Shortest Path First (OSPF) Open Shortest Path First version 3(OSPFv3) Enhanced Interior Gateway Routing Protocol (EIGRP) Enhanced Interior Gateway Routing Protocol version 6 (EIGRPv6) DHCPv6 Protocol Independent Multicast (PIM) Protocol Independent Multicast version 6 (PIMv6) Hot Standby Router Protocol next generation (HSRPng) IPv6 control Generic Routing Encapsulation (GRE) keepalive Network Address Translation (NAT) punt Intermediate System-to-Intermediate System (IS-IS)
WK_CPU_Q_FORUS_ADDR_RESOLUTION(5)	Address Resolution Protocol (ARP) IPv6 neighbor advertisement and neighbor solicitation
WK_CPU_Q_ICMP_REDIRECT(6)	Internet Control Message Protocol (ICMP) redirect
WK_CPU_Q_INTER_FED_TRAFFIC(7)	Layer 2 bridge domain inject for internal communication.

CPU queues (Queue No.)	Feature(s)
WK_CPU_Q_L2_LVX_CONT_PACK(8)	Exchange ID (XID) packet
WK_CPU_Q_EWLC_CONTROL(9)	Embedded Wirelss Controller (eWLC) [Control and Provisioning of Wireless Access Points (CAPWAP) (UDP 5246)]
WK_CPU_Q_EWLC_DATA(10)	eWLC data packet (CAPWAP DATA, UDP 5247)
WK_CPU_Q_L2_LVX_DATA_PACK(11)	Unknown unicast packet punted for map request.
WK_CPU_Q_BROADCAST(12)	All types of broadcast
WK_CPU_Q_OPENFLOW(13)	Learning cache overflow (Layer 2 + Layer 3)
WK_CPU_Q_CONTROLLER_PUNT(14)	Data - access control list (ACL) Full Data - IPv4 options Data - IPv6 hop-by-hop Data - out-of-resources / catch all Data - Reverse Path Forwarding (RPF) incomplete Glean packet
WK_CPU_Q_TOPOLOGY_CONTROL(15)	Spanning Tree Protocol (STP) Resilient Ethernet Protocol (REP) Shared Spanning Tree Protocol (SSTP)
WK_CPU_Q_PROTO_SNOOPING(16)	Address Resolution Protocol (ARP) snooping for Dynamic ARP Inspection (DAI)
WK_CPU_Q_DHCP_SNOOPING(17)	DHCP snooping
WK_CPU_Q_TRANSIT_TRAFFIC(18)	This is used for packets punted by NAT, which need to be handled in the software path.
WK_CPU_Q_RPF_FAILED(19)	Data – mRPF (multicast RPF) failed

CPU queues (Queue No.)	Feature(s)
WK_CPU_Q_MCAST_END_STATION _SERVICE(20)	Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD) control
WK_CPU_Q_LOGGING(21)	Access control list (ACL) logging
WK_CPU_Q_PUNT_WEBAUTH(22)	Web Authentication
WK_CPU_Q_HIGH_RATE_APP(23)	Broadcast
WK_CPU_Q_EXCEPTION(24)	IKE indication IP learning violation IP port security violation IP Static address violation IPv6 scope check Remote Copy Protocol (RCP) exception Unicast RPF fail
WK_CPU_Q_SYSTEM_CRITICAL(25)	Media Signaling/ Wireless Proxy ARP
WK_CPU_Q_NFL_SAMPLED_DATA(26)	Netflow sampled data and Media Services Proxy (MSP)
WK_CPU_Q_LOW_LATENCY(27)	Bidirectional Forwarding Detection (BFD), Precision Time Protocol (PTP)
WK_CPU_Q_EGR_EXCEPTION(28)	Egress resolution exception
WK_CPU_Q_STACKWISE_VIRTUAL _CONTROL(29)	Front side stacking protocols, namely SVL
WK_CPU_Q_MCAST_DATA(30)	Data - (S,G) creation Data - local joins Data - PIM Registration Data - SPT switchover

CPU queues (Queue No.)	Feature(s)
	Data - Multicast
WK_CPU_Q_GOLD_PKT(31)	Gold

Default Policy

By default, the system-generated CoPP policy is applied to the punt/inject path. The default policy can be viewed by using common MQC-based commands. It is also viewable within the switch configuration. The only policy that is allowed to be applied on ingress or egress of the CPU/control-plane is the system-defined policy.

Use **"show policy-map control-plane**" to view the policy applied to the control-plane:

<#root>

```
Catalyst-9600#

show policy-map control-plane


Control Plane

 Service-policy input: system-cpp-policy

   Class-map: system-cpp-police-ios-routing (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: none
      police:
          rate 17000 pps, burst 4150 packets
        conformed 95904305 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          drop

<snip>

   Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: any
```

Adjust CoPP

CoPP policer rates are user-configurable. Users also have the ability to disable queues.

This example demonstrates how to adjust an individual policer value. In this example, the adjusted class is "**system-cpp-police-protocol-snooping."**

<#root>

Device>

enable

Device#

configure terminal

Device(config)#

policy-map system-cpp-policy

Device(config-pmap)#

Device(config-pmap)#

class system-cpp-police-protocol-snooping

Device(config-pmap-c)#

Device(config-pmap-c)#

police rate 100 pps

Device(config-pmap-c-police)#

Device(config-pmap-c-police)#

exit

Device(config-pmap-c)#

exit

Device(config-pmap)#

exit

Device(config)#

Device(config)#

control-plane

Device(config-cp)#

Device(config)#

control-plane

Device(config-cp)#

service-policy input system-cpp-policy

Device(config-cp)#

```
Device(config-cp)#

end


Device#

show policy-map control-plane
```

This example demonstrates how to disable a queue entirely. Use caution when disabling queues, as this could lead to potential oversaturation of the CPU.

<#root>

```
Device>

enable

Device#

configure terminal


Device(config)#

policy-map system-cpp-policy

Device(config-pmap)#

Device(config-pmap)#

class system-cpp-police-protocol-snooping

Device(config-pmap-c)#


Device(config-pmap-c)#

no police rate 100 pps

Device(config-pmap-c)#

end
```

Troubleshoot

Methodology

CPU utilization is impacted by two basic activities- processes and interruption. Processes are structured activites the CPU performs while interruption refers to packets intercepted on the dataplane and sent to the CPU for action. Together, these activities comprise the total utilization of the CPU. Since CoPP is enabled by default, a service impact does not correlate with high CPU utilization necessarily. If CoPP does its job, CPU utilization is not greatly impacted. It is important to consider the overall utilization of the CPU, but overall utilization does not tell the entire story. The show commands and utilities in this section are used to quickly assess the health of the CPU and to identify relevant details about CPU-bound traffic.

Guidelines:

- Determine if the problem relates to the control-plane. Most transit traffic is forwarded in hardware. Only certain traffic types and certain scenarios involve the CPU and control-plane, so keep this in mind throughout the investigation.
- Understand your utilization baseline. It is important to understand what normal utilization looks like so deviations from the norm can be identified.
- Validate overall utilization for both processes and interruption. Identify any processes that take up unexpected volumes of CPU cycles. If utilization falls outside of the expected range, this is potentially cause for concern. It is important to understand the average utilization for a system, so that deviations outside of the norm are recognized. Keep in mind that utilization alone is not a complete picture of control plane health.
- Determine if there is actively incrementing drops in CoPP. CoPP drops are not always indicative of a problem, but if you troubleshoot a problem related to a traffic class that is actively policed, this is a strong indicator of relevance.

Useful Show Commands

The switch allows for quick oversight of CPU health and CoPP statistics. There is also useful CLI to quickly determine the point of ingress of CPU-bound traffic.

Determine Overall and Historical Utilization

- **"Show processes cpu sorted"** is used to view overall CPU utilization. The "sorted" argument sorts process output based on percentage of usage. Processes using more CPU resources are at the top of the output. Utilization due to interrupts is also provided as a percentage.

<#root>

```
Catalyst-9600#

show processes cpu sorted


CPU utilization for five seconds: 92%/13%; one minute: 76%; five minutes: 73%

<<<--- Utilization is displayed for 5 second (both process and interrupt), 1 minute and 5 minute interva


                                                                                     92% refers to the c


                                                                                     The 13% value refer


PID Runtime(ms)    Invoked      uSecs   5Sec   1Min   5Min  TTY  Process

<<<--- Runtime statistics, as well as utilization averages are displayed here. The process is also ident


344   547030523  607054509        901  38.13% 30.61% 29.32%   0  SISF Switcher Th
345   394700227  615024099        641  31.18% 22.68% 21.66%   0  SISF Main Thread
```

```
 98   112308516  119818535        937   4.12%  4.76%  5.09%   0  Crimson flush tr
247    47096761   92250875        510   2.42%  2.21%  2.18%   0  Spanning Tree
123    35303496  679878082         51   1.85%  1.88%  1.84%   0  IOSXE-RP Punt Se
234         955       1758        543   1.61%  0.71%  0.23%   3  SSH Process
547     5360168    5484910        977   1.04%  0.46%  0.44%   0  DHCPD Receive
229    27381066  963726156         28   1.04%  1.34%  1.23%   0 IP Input
 79    13183805  108951712        121   0.48%  0.55%  0.55%   0 IOSD ipc task
  9     1073134     315186       3404   0.40%  0.06%  0.03%   0 Check heaps
 37    11099063  147506419         75   0.40%  0.54%  0.52%   0 ARP Input
312     2986160  240782059         12   0.24%  0.12%  0.14%   0 DAI Packet Proce
<snip>
565           0          1          0   0.00%  0.00%  0.00%   0 LICENSE AGENT
566          14       1210         11   0.00%  0.00%  0.00%   0 DHCPD Timer
567          40         45        888   0.00%  0.00%  0.00%   0 OVLD SPA Backgro
568          12       2342          5   0.00%  0.00%  0.00%   0 DHCPD Database
569           0         12          0   0.00%  0.00%  0.00%   0 SpanTree Flush
571           0          1          0   0.00%  0.00%  0.00%   0 EM Action CNS
572         681     140276          4   0.00%  0.00%  0.00%   0 Inline power inc
```

- "**Show processes cpu history**" provides a historical graph of CPU utilization over the last 60 seconds, 5 minutes and 72 hours.

<#root>

```
Catalyst-9600#
```

show processes cpu history

```
    999777776666688888666667777777777888887777766666999998888866
```

<<<--- The numbers at the top of each column represent the highest value seen throughout the time perio

```
    222555559999944444444440000088888888881111177777333335555500
```

It is read top-down. "9" over "2" in this example means "92%" for example.

```
100
90  *** ***** **********
80  ******** ***** ********** **********
70  ****************** ************************************
60  ************************************************************
50  ************************************************************
40  ************************************************************
30  ************************************************************
20  ************************************************************
10  ************************************************************
```

<<<--- The "*" represents the highest value during the given time period. This relates to a momentary sp

```
0....5....1....1....2....2....3....3....4....4....5....5....6
```

In this example, utilization spiked to 92% in the last 5 seconds.

```
          0    5    0    5    0    5    0    5    0    5    0
          CPU% per second (last 60 seconds)
         * = maximum CPU% # = average CPU%
```

```
   999898989999898998998998989889999989889898899999999899999999
   431823091102635316235129283771336574892809604014230901133511
100 ** *
90  ***** *****************************************************
80  ************#***#*#**#***####*##*****#**#***#***#**********
70  ###########################################################
```

<<<--- The "#" represents the average utilization. This indicates sustained utilization.

```
60  ###########################################################
```

In this example, within the last 5 minutes the average utilization was sustained around 70% while

```
50  ###########################################################
```

the maximum utilization spiked to 94%.

```
40  ###########################################################
30  ###########################################################
20  ###########################################################
10  ###########################################################
0....5....1....1....2....2....3....3....4....4....5....5....6
          0    5    0    5    0    5    0    5    0    5    0
          CPU% per minute (last 60 minutes)
         * = maximum CPU% # = average CPU%
```

```
   999999999999999999999999999999999999999999999999999999999999999999999999
   665656566646555666655656575654556567737555567574545545577595755464857 6757
100 ********** ***************** ******* ********* * ** ********* * *****
90  *********************************************************************
80  *********************************************************************
70  #####################################################################
60  #####################################################################
50  #####################################################################
40  #####################################################################
30  #####################################################################
20  #####################################################################
10  #####################################################################
0....5....1....1....2....2....3....3....4....4....5....5....6....6....7..
          0    5    0    5    0    5    0    5    0    5    0    5    0
               CPU% per hour (last 72 hours)
              * = maximum CPU% # = average CPU%
```

Check for Control Plane Policing

- Use **"show platform hardware fed <switch> active qos queue stats internal cpu policer"** to view aggregate CoPP statistics and additional information on queue/policer structure. This output provides a historic view of policer statistics since the last reset of the control plane. These counters are manually clearable as well. Generally, evidence of control-plane drops by policer points to a problem with the associated queue/class but make sure drops actively increment while the problem occurs. Run the command several times to observe for increasing Queue Drop values.

<#root>

```
Catalyst9500#

show platform hardware fed active qos queue stats internal cpu policer


                         CPU Queue Statistics
============================================================================================
                                              (default) (set)     Queue        Queue
QId PlcIdx  Queue Name                Enabled   Rate     Rate      Drop(Bytes)  Drop(Frames)

<-- The top section of this output gives a historical view of CoPP drops. Run the command several times

--------------------------------------------------------------------------------------------

CPU queues correlate with a Policer Index (PlcIdx) and Queue (QId).

0    11     DOT1X Auth                  Yes     1000      1000      0             0

Note that multiple policer indices map to the same queue for some classes.

1    1      L2 Control                  Yes     2000      2000      0             0
2    14     Forus traffic               Yes     4000      4000      0             0
3    0      ICMP GEN                    Yes     750       750       0             0
4    2      Routing Control             Yes     5500      5500      0             0
5    14     Forus Address resolution    Yes     4000      4000      83027876      1297199
6    0      ICMP Redirect               Yes     750       750       0             0
7    16     Inter FED Traffic           Yes     2000      2000      0             0
8    4      L2 LVX Cont Pack            Yes     1000      1000      0             0
9    19     EWLC Control                Yes     13000     13000     0             0
10   16     EWLC Data                   Yes     2000      2000      0             0
11   13     L2 LVX Data Pack            Yes     1000      1000      0             0
12   0      BROADCAST                   Yes     750       750       0             0
13   10     Openflow                    Yes     250       250       0             0
14   13     Sw forwarding               Yes     1000      1000      0             0
15   8      Topology Control            Yes     13000     16000     0             0
16   12     Proto Snooping              Yes     2000      2000      0             0
17   6      DHCP Snooping               Yes     500       500       0             0
18   13     Transit Traffic             Yes     1000      1000      0             0
19   10     RPF Failed                  Yes     250       250       0             0
20   15     MCAST END STATION           Yes     2000      2000      0             0
21   13     LOGGING                     Yes     1000      1000      769024        12016
22   7      Punt Webauth                Yes     1000      1000      0             0
23   18     High Rate App               Yes     13000     13000     0             0
24   10     Exception                   Yes     250       250       0             0
25   3      System Critical             Yes     1000      1000      0             0
26   10     NFL SAMPLED DATA            Yes     250       250       0             0
27   2      Low Latency                 Yes     5500      5500      0             0
28   10     EGR Exception               Yes     250       250       0             0
29   5      Stackwise Virtual OOB       Yes     8000      8000      0             0
30   9      MCAST Data                  Yes     500       500       0             0
31   3      Gold Pkt                    Yes     1000      1000      0             0

* NOTE: CPU queue policer rates are configured to the closest hardware supported value
```

```
                      CPU Queue Policer Statistics
====================================================================
Policer    Policer Accept   Policer Accept  Policer Drop  Policer Drop
  Index        Bytes           Frames        Bytes          Frames
-------------------------------------------------------------------
0          59894            613             0             0
1          15701689         57082           0             0
2          5562892          63482           0             0
3          3536             52              0             0
4          0                0               0             0
5          0                0               0             0
6          0                0               0             0
7          0                0               0             0
8          2347194476       32649666        0             0
9          0                0               0             0
10         0                0               0             0
11         0                0               0             0
12         0                0               0             0
13         577043           8232            769024        12016
14         719225176        11182355        83027876      1297199
15         132766           1891            0             0
16         0                0               0             0
17         0                0               0             0
18         0                0               0             0
19         0                0               0             0

                  Second Level Policer Statistics

<-- Second level policer information begins here. Catalyst CoPP is organized with two policers to allow

====================================================================
20         2368459057       32770230        0             0
21         719994879        11193091        0             0

Policer Index Mapping and Settings
--------------------------------------------------------------------
level-2   :   level-1                      (default)   (set)
PlcIndex  :   PlcIndex                       rate      rate
--------------------------------------------------------------------
20        :   1  2  8                        13000     17000
21        :   0 4 7 9 10 11 12 13 14 15      6000      6000
====================================================================

               Second Level Policer Config
====================================================================
    level-1 level-2                                level-2
QId PlcIdx  PlcIdx  Queue Name                     Enabled
--------------------------------------------------------------------
0    11      21      DOT1X Auth                      Yes
1    1       20      L2 Control                      Yes
2    14      21      Forus traffic                   Yes
3    0       21      ICMP GEN                        Yes
4    2       20      Routing Control                 Yes
5    14      21      Forus Address resolution        Yes
6    0       21      ICMP Redirect                   Yes
7    16      -       Inter FED Traffic               No
8    4       21      L2 LVX Cont Pack                Yes
9    19      -       EWLC Control                    No
10   16      -       EWLC Data                       No
11   13      21      L2 LVX Data Pack                Yes
12   0       21      BROADCAST                       Yes
13   10      21      Openflow                        Yes
```

```
14    13      21      Sw forwarding              Yes
15    8       20      Topology Control           Yes
16    12      21      Proto Snooping             Yes
17    6       -       DHCP Snooping              No
18    13      21      Transit Traffic            Yes
19    10      21      RPF Failed                 Yes
20    15      21      MCAST END STATION          Yes
21    13      21      LOGGING                    Yes
22    7       21      Punt Webauth               Yes
23    18      -       High Rate App              No
24    10      21      Exception                  Yes
25    3       -       System Critical            No
26    10      21      NFL SAMPLED DATA           Yes
27    2       20      Low Latency                Yes
28    10      21      EGR Exception              Yes
29    5       -       Stackwise Virtual OOB      No
30    9       21      MCAST Data                 Yes
31    3       -       Gold Pkt                   No

                        CPP Classes to queue map

<-- Information on how different traffic types map to different queues are found here.
===========================================================================================

PlcIdx CPP Class                                : Queues
-------------------------------------------------------------------------------------------
0      system-cpp-police-data                   : ICMP GEN/ BROADCAST/ ICMP Redirect/
10     system-cpp-police-sys-data               : Openflow/ Exception/ EGR Exception/ NFL SAMPLED DATA
13     system-cpp-police-sw-forward             : Sw forwarding/ LOGGING/ L2 LVX Data Pack/ Transit Tr
9      system-cpp-police-multicast              : MCAST Data/
15     system-cpp-police-multicast-end-station  : MCAST END STATION /
7      system-cpp-police-punt-webauth           : Punt Webauth/
1      system-cpp-police-l2-control             : L2 Control/
2      system-cpp-police-routing-control        : Routing Control/ Low Latency/
3      system-cpp-police-system-critical        : System Critical/ Gold Pkt/
4      system-cpp-police-l2lvx-control          : L2 LVX Cont Pack/
8      system-cpp-police-topology-control       : Topology Control/
11     system-cpp-police-dot1x-auth             : DOT1X Auth/
12     system-cpp-police-protocol-snooping      : Proto Snooping/
6      system-cpp-police-dhcp-snooping          : DHCP Snooping/
14     system-cpp-police-forus                  : Forus Address resolution/ Forus traffic/
5      system-cpp-police-stackwise-virt-control : Stackwise Virtual OOB/
16     system-cpp-default                       : Inter FED Traffic/ EWLC Data/
18     system-cpp-police-high-rate-app          : High Rate App/
19     system-cpp-police-ewlc-control           : EWLC Control/
20     system-cpp-police-ios-routing            : L2 Control/ Topology Control/ Routing Control/ Low L
21     system-cpp-police-ios-feature            : ICMP GEN/ BROADCAST/ ICMP Redirect/ L2 LVX Cont Pack
```

Gather information on punted traffic

These commands are used to gather information on traffic punted to the CPU, including the type of traffic and the physical points of ingress.

- "**Show platform software fed <switch> active punt cpuq all**" or "**Show platform software fed <switch> active punt cpuq <0-31 Queue ID**>" can be used to see statistics related to all or to a specific CPU queue.

<#root>

```
C9300#

show platform software fed switch active punt cpuq all

Punt CPU Q Statistics
===========================================

CPU Q Id                       : 0
CPU Q Name                     : CPU_Q_DOT1X_AUTH
Packets received from ASIC     : 964
Send to IOSd total attempts    : 964
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                : 964
RX packets dq'd after intack   : 0
Active RxQ event               : 964
RX spurious interrupt          : 0
RX phy_idb fetch failed: 0
RX table_id fetch failed: 0
RX invalid punt cause: 0

CPU Q Id                       : 1
CPU Q Name                     : CPU_Q_L2_CONTROL
Packets received from ASIC     : 80487
Send to IOSd total attempts    : 80487
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                : 80474
RX packets dq'd after intack   : 16
Active RxQ event               : 80474
RX spurious interrupt          : 9
RX phy_idb fetch failed: 0
RX table_id fetch failed: 0
RX invalid punt cause: 0

CPU Q Id                       : 2
CPU Q Name                     : CPU_Q_FORUS_TRAFFIC
Packets received from ASIC     : 176669
Send to IOSd total attempts    : 176669
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                : 165584
```

```
RX packets dq'd after intack   : 12601
Active RxQ event               : 165596
RX spurious interrupt          : 11851
RX phy_idb fetch failed: 0
RX table_id fetch failed: 0
RX invalid punt cause: 0
<snip>
```

```
C9300#
```

show platform software fed switch active punt cpuq 16 <-- Queue ID 16 correlates with Protocol Snooping.

```
Punt CPU Q Statistics
===========================================

CPU Q Id                       : 16
CPU Q Name                     : CPU_Q_PROTO_SNOOPING
Packets received from ASIC     : 55661
Send to IOSd total attempts    : 55661
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                : 55659
RX packets dq'd after intack   : 9
Active RxQ event               : 55659
RX spurious interrupt          : 23
RX phy_idb fetch failed: 0
RX table_id fetch failed: 0
RX invalid punt cause: 0

Replenish Stats for all rxq:
-------------------------------------------
Number of replenish            : 4926842
Number of replenish suspend    : 0
Number of replenish un-suspend : 0
-------------------------------------------
```

- Use "s**how platform software fed <switch> active punt cause summary**" for a quick look at all of the different traffic types that have been seen at the CPU. Note that only non-zero causes are shown.

<#root>

```
C9300#
```

show platform software fed switch active punt cause summary

```
Statistics for all causes

Cause  Cause Info                    Rcvd                 Dropped
------------------------------------------------------------------------------
7      ARP request or response       142962               0
11     For-us data                   490817               0
21     RP<->QFP keepalive            448742               0
24     Glean adjacency               2                    0
```

```
55     For-us control                  415222              0
58     Layer2 bridge domain data packe 3654659             0
60     IP subnet or broadcast packet   37167               0
75     EPC                             17942               0
96     Layer2 control protocols        358614              0
97     Packets to LFTS                 964                 0
109    snoop packets                   48867               0
------------------------------------------------------------------------------
```

- Use the command **"show platform software fed <switch> active punt rates interfaces"** to quickly view the interfaces CPU-bound traffic ingresses the system. This command only shows interfaces with a non-zero input queue.

<#root>

```
C9300#
```

`show platform software fed switch active punt rates interfaces`

```
Punt Rate on Interfaces Statistics

Packets per second averaged over 10 seconds, 1 min and 5 mins
===========================================================================================
                             |            |  Recv |  Recv |  Recv |  Drop |  Drop   |  Drop
 Interface Name              |  IF_ID     |  10s  |  1min |  5min |  10s  |  1min   |  5min
===========================================================================================
 TenGigabitEthernet1/0/2       0x0000000a        5       5       5       0       0       0
 TenGigabitEthernet1/0/23      0x0000001f        1       1       1       0       0       0

-------------------------------------------------------------------------------------------
```

- Use **"show platform software fed <switch> active punt rates interfaces <IF-ID>"** to drill down and view the individual queues of the interface. This command shows aggregate statistics and can be used to view historic input queue activity and if traffic has been policed.

<#root>

```
C9300#
```

`show platform software fed switch active punt rates interfaces 0x1f <-- "0x1f" is the IF_ID of Te1/0/23,`

```
Punt Rate on Single Interfaces Statistics

Interface : TenGigabitEthernet1/0/23 [if_id: 0x1F]

  Received                         Dropped
  --------                         -------
   Total          : 1010652          Total           : 0
   10 sec average : 1                10 sec average  : 0
    1 min average : 1                 1 min average  : 0
    5 min average : 1                 5 min average  : 0

Per CPUQ punt stats on the interface (rate averaged over 10s interval)
==========================================================================
 Q  |         Queue                  |  Recv   |  Recv   |  Drop  |  Drop   |
 no |         Name                   |  Total  |  Rate   |  Total |  Rate   |
==========================================================================
```

```
 0   CPU_Q_DOT1X_AUTH                      0          0          0          0
 1   CPU_Q_L2_CONTROL                   9109          0          0          0
 2   CPU_Q_FORUS_TRAFFIC              176659          0          0          0
 3   CPU_Q_ICMP_GEN                        0          0          0          0
 4   CPU_Q_ROUTING_CONTROL            447374          0          0          0
 5   CPU_Q_FORUS_ADDR_RESOLUTION       80693          0          0          0
 6   CPU_Q_ICMP_REDIRECT                   0          0          0          0
 7   CPU_Q_INTER_FED_TRAFFIC               0          0          0          0
 8   CPU_Q_L2LVX_CONTROL_PKT               0          0          0          0
 9   CPU_Q_EWLC_CONTROL                    0          0          0          0
 10  CPU_Q_EWLC_DATA                       0          0          0          0
 11  CPU_Q_L2LVX_DATA_PKT                  0          0          0          0
 12  CPU_Q_BROADCAST                   22680          0          0          0
 13  CPU_Q_CONTROLLER_PUNT                 0          0          0          0
 14  CPU_Q_SW_FORWARDING                   0          0          0          0
 15  CPU_Q_TOPOLOGY_CONTROL           271014          0          0          0
 16  CPU_Q_PROTO_SNOOPING                  0          0          0          0
 17  CPU_Q_DHCP_SNOOPING                   0          0          0          0
 18  CPU_Q_TRANSIT_TRAFFIC                 0          0          0          0
 19  CPU_Q_RPF_FAILED                      0          0          0          0
 20  CPU_Q_MCAST_END_STATION_SERVICE    2679          0          0          0
 21  CPU_Q_LOGGING                       444          0          0          0
 22  CPU_Q_PUNT_WEBAUTH                    0          0          0          0
 23  CPU_Q_HIGH_RATE_APP                   0          0          0          0
 24  CPU_Q_EXCEPTION                       0          0          0          0
 25  CPU_Q_SYSTEM_CRITICAL                 0          0          0          0
 26  CPU_Q_NFL_SAMPLED_DATA                0          0          0          0
 27  CPU_Q_LOW_LATENCY                     0          0          0          0
 28  CPU_Q_EGR_EXCEPTION                   0          0          0          0
 29  CPU_Q_FSS                             0          0          0          0
 30  CPU_Q_MCAST_DATA                      0          0          0          0
 31  CPU_Q_GOLD_PKT                        0          0          0          0

--------------------------------------------------------------------------
```

Inspect CPU bound traffic

The Catalyst 9000 family of switches offers utilities to monitor and view CPU-bound traffic. Use these tools to understand what traffic is actively punted to the CPU.

Embedded Packet Capture (EPC)

EPC on the control plane can be done in either direction (or both). For punted traffic, capture inbound. EPC on the control plane can be saved to buffer or to file.

<#root>

```
C9300#

monitor capture CONTROL control-plane in match any buffer circular size 10


C9300#

show monitor capture CONTROL parameter <-- Check to ensure parameters are as expected.

   monitor capture CONTROL control-plane IN
   monitor capture CONTROL match any
   monitor capture CONTROL buffer size 10 circular
```

```
C9300#

monitor capture CONTROL start <-- Starts the capture.

Started capture point : CONTROL
C9300#

monitor capture CONTROL stop <-- Stops the capture.

Capture statistics collected at software:
    Capture duration - 5 seconds
    Packets received - 39
    Packets dropped - 0
    Packets oversized - 0

Bytes dropped in asic - 0

Capture buffer will exists till exported or cleared

Stopped capture point : CONTROL
```

The capture results can be viewed in either brief or detailed output.

<#root>

```
C9300#

show monitor capture CONTROL buffer brief

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

    1   0.000000 5c:5a:c7:61:4c:5f -> 00:00:04:00:0e:00 ARP 64 192.168.10.1 is at 5c:5a:c7:61:4c:5f
    2   0.030643 00:00:00:00:00:00 -> 00:06:df:f7:20:01 0x0000 30 Ethernet II
    3   0.200016 5c:5a:c7:61:4c:5f -> 00:00:04:00:0e:00 ARP 64 192.168.10.1 is at 5c:5a:c7:61:4c:5f
    4   0.400081 5c:5a:c7:61:4c:5f -> 00:00:04:00:0e:00 ARP 64 192.168.10.1 is at 5c:5a:c7:61:4c:5f
    5   0.599962 5c:5a:c7:61:4c:5f -> 00:00:04:00:0e:00 ARP 64 192.168.10.1 is at 5c:5a:c7:61:4c:5f
    6   0.800067 5c:5a:c7:61:4c:5f -> 00:00:04:00:0e:00 ARP 64 192.168.10.1 is at 5c:5a:c7:61:4c:5f
    7   0.812456 00:1b:0d:a5:e2:a5 -> 01:80:c2:00:00:00 STP 60 RST. Root = 0/10/00:1b:53:bb:91:00  Cost
    8   0.829809 10.122.163.3 -> 224.0.0.2    HSRP 92 Hello (state Active)
    9   0.981313 10.122.163.2 -> 224.0.0.13   PIMv2 72 Hello
   10   1.004747 5c:5a:c7:61:4c:5f -> 00:00:04:00:0e:00 ARP 64 192.168.10.1 is at 5c:5a:c7:61:4c:5f
   11   1.200082 5c:5a:c7:61:4c:5f -> 00:00:04:00:0e:00 ARP 64 192.168.10.1 is at 5c:5a:c7:61:4c:5f
   12   1.399987 5c:5a:c7:61:4c:5f -> 00:00:04:00:0e:00 ARP 64 192.168.10.1 is at 5c:5a:c7:61:4c:5f
   13   1.599944 5c:5a:c7:61:4c:5f -> 00:00:04:00:0e:00 ARP 64 192.168.10.1 is at 5c:5a:c7:61:4c:5f
<snip>

C9300#

show monitor capture CONTROL buffer detail | begin Frame 7

Frame 7: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface /tmp/epc_ws/wif_to_ts_p
    Interface id: 0 (/tmp/epc_ws/wif_to_ts_pipe)
        Interface name: /tmp/epc_ws/wif_to_ts_pipe
    Encapsulation type: Ethernet (1)
    Arrival Time: May  3, 2023 23:58:11.727432000 UTC
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1683158291.727432000 seconds
    [Time delta from previous captured frame: 0.012389000 seconds]
    [Time delta from previous displayed frame: 0.012389000 seconds]
    [Time since reference or first frame: 0.812456000 seconds]
    Frame Number: 7
```

```
    Frame Length: 60 bytes (480 bits)
    Capture Length: 60 bytes (480 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:llc:stp]
IEEE 802.3 Ethernet
    Destination: 01:80:c2:00:00:00 (01:80:c2:00:00:00)
        Address: 01:80:c2:00:00:00 (01:80:c2:00:00:00)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
    Source: 00:1b:0d:a5:e2:a5 (00:1b:0d:a5:e2:a5)
        Address: 00:1b:0d:a5:e2:a5 (00:1b:0d:a5:e2:a5)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Length: 39
    Padding: 00000000000000
Logical-Link Control
    DSAP: Spanning Tree BPDU (0x42)
        0100 001. = SAP: Spanning Tree BPDU
        .... ...0 = IG Bit: Individual
    SSAP: Spanning Tree BPDU (0x42)
        0100 001. = SAP: Spanning Tree BPDU
        .... ...0 = CR Bit: Command
    Control field: U, func=UI (0x03)
        000. 00.. = Command: Unnumbered Information (0x00)
        .... ..11 = Frame type: Unnumbered frame (0x3)
Spanning Tree Protocol
    Protocol Identifier: Spanning Tree Protocol (0x0000)
    Protocol Version Identifier: Rapid Spanning Tree (2)
    BPDU Type: Rapid/Multiple Spanning Tree (0x02)
    BPDU flags: 0x3c, Forwarding, Learning, Port Role: Designated
        0... .... = Topology Change Acknowledgment: No
        .0.. .... = Agreement: No
        ..1. .... = Forwarding: Yes
        ...1 .... = Learning: Yes
        .... 11.. = Port Role: Designated (3)
        .... ..0. = Proposal: No
        .... ...0 = Topology Change: No
    Root Identifier: 0 / 10 / 00:1b:53:bb:91:00
        Root Bridge Priority: 0
        Root Bridge System ID Extension: 10
        Root Bridge System ID: 00:1b:53:bb:91:00 (00:1b:53:bb:91:00)
    Root Path Cost: 19
    Bridge Identifier: 32768 / 10 / 00:1b:0d:a5:e2:80
        Bridge Priority: 32768
        Bridge System ID Extension: 10
        Bridge System ID: 00:1b:0d:a5:e2:80 (00:1b:0d:a5:e2:80)
    Port identifier: 0x8025
    Message Age: 1
    Max Age: 20
    Hello Time: 2
    Forward Delay: 15
    Version 1 Length: 0

C9300#

monitor capture CONTROL buffer display-filter "frame.number==9" detailed <-- Most Wireshark display filt

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

Frame 9: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface /tmp/epc_ws/wif_to_ts_p
    Interface id: 0 (/tmp/epc_ws/wif_to_ts_pipe)
        Interface name: /tmp/epc_ws/wif_to_ts_pipe
```

```
    Encapsulation type: Ethernet (1)
    Arrival Time: May  4, 2023 00:07:44.912567000 UTC
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1683158864.912567000 seconds
    [Time delta from previous captured frame: 0.123942000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 1.399996000 seconds]
    Frame Number: 9
    Frame Length: 64 bytes (512 bits)
    Capture Length: 64 bytes (512 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:vlan:ethertype:arp]
Ethernet II, Src: 5c:5a:c7:61:4c:5f (5c:5a:c7:61:4c:5f), Dst: 00:00:04:00:0e:00 (00:00:04:00:0e:00)
    Destination: 00:00:04:00:0e:00 (00:00:04:00:0e:00)
        Address: 00:00:04:00:0e:00 (00:00:04:00:0e:00)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: 5c:5a:c7:61:4c:5f (5c:5a:c7:61:4c:5f)
        Address: 5c:5a:c7:61:4c:5f (5c:5a:c7:61:4c:5f)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 10
    000. .... .... .... = Priority: Best Effort (default) (0)
    ...0 .... .... .... = DEI: Ineligible
    .... 0000 0000 1010 = ID: 10
    Type: ARP (0x0806)
    Padding: 0000000000000000000000000000
    Trailer: 00000000
Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: 5c:5a:c7:61:4c:5f (5c:5a:c7:61:4c:5f)
    Sender IP address: 192.168.10.1
    Target MAC address: 00:00:04:00:0e:00 (00:00:04:00:0e:00)
    Target IP address: 192.168.10.25
```

Capture results can either be written directly to file, or exported from buffer.

<#root>

```
C9300#

monitor capture CONTROL export location flash:control.pcap <-- Exports the current buffer to file. Exter

Export Started Successfully

Export completed for capture point CONTROL
C9300#
C9300#

dir flash: | in control.pcap

475231  -rw-            3972   May 4 2023 00:00:38 +00:00  control.pcap
C9300#
```

FED CPU Packet Capture

The Catalyst 9000 family of switches supports a debug utility that allows enhanced visibility of packets to and from the CPU.

```
C9300#debug platform software fed switch active punt packet-capture ?
  buffer        Configure packet capture buffer
  clear-filter  Clear punt PCAP filter
  set-filter    Specify wireshark like filter (Punt PCAP)
  start         Start punt packet capturing
  stop          Stop punt packet capturing

C9300#$re fed switch active punt packet-capture buffer limit 16384
Punt PCAP buffer configure: one-time with buffer size 16384...done

C9300#show platform software fed switch active punt packet-capture status
Punt packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 0 packets. Capture capacity : 16384 packets

C9300#debug platform software fed switch active punt packet-capture start
Punt packet capturing started.

C9300#debug platform software fed switch active punt packet-capture stop
Punt packet capturing stopped. Captured 55 packet(s)
```

Buffer contents have brief and detailed options for output.

<#root>

```
C9300#

show platform software fed switch active punt packet-capture brief

Punt packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 55 packets. Capture capacity : 16384 packets

------ Punt Packet Number: 1, Timestamp: 2023/05/04 00:17:41.709 ------
 interface : physical: TenGigabitEthernet1/0/2[if-id: 0x0000000a], pal: TenGigabitEthernet1/0/2 [if-id:
 metadata  : cause: 109 [snoop packets], sub-cause: 1, q-no: 16, linktype: MCP_LINK_TYPE_IP [1]
 ether hdr : dest mac: 0000.0400.0e00, src mac: 5c5a.c761.4c5f
 ether hdr : vlan: 10, ethertype: 0x8100

------ Punt Packet Number: 2, Timestamp: 2023/05/04 00:17:41.909 ------
 interface : physical: TenGigabitEthernet1/0/2[if-id: 0x0000000a], pal: TenGigabitEthernet1/0/2 [if-id:
 metadata  : cause: 109 [snoop packets], sub-cause: 1, q-no: 16, linktype: MCP_LINK_TYPE_IP [1]
 ether hdr : dest mac: 0000.0400.0e00, src mac: 5c5a.c761.4c5f
 ether hdr : vlan: 10, ethertype: 0x8100

------ Punt Packet Number: 3, Timestamp: 2023/05/04 00:17:42.109 ------
 interface : physical: TenGigabitEthernet1/0/2[if-id: 0x0000000a], pal: TenGigabitEthernet1/0/2 [if-id:
 metadata  : cause: 109 [snoop packets], sub-cause: 1, q-no: 16, linktype: MCP_LINK_TYPE_IP [1]
 ether hdr : dest mac: 0000.0400.0e00, src mac: 5c5a.c761.4c5f
 ether hdr : vlan: 10, ethertype: 0x8100

------ Punt Packet Number: 4, Timestamp: 2023/05/04 00:17:42.309 ------
 interface : physical: TenGigabitEthernet1/0/2[if-id: 0x0000000a], pal: TenGigabitEthernet1/0/2 [if-id:
```

```
 metadata  : cause: 109 [snoop packets], sub-cause: 1, q-no: 16, linktype: MCP_LINK_TYPE_IP [1]
 ether hdr : dest mac: 0000.0400.0e00, src mac: 5c5a.c761.4c5f
 ether hdr : vlan: 10, ethertype: 0x8100

------ Punt Packet Number: 5, Timestamp: 2023/05/04 00:17:42.509 ------
 interface : physical: TenGigabitEthernet1/0/2[if-id: 0x0000000a], pal: TenGigabitEthernet1/0/2 [if-id:
 metadata  : cause: 109 [snoop packets], sub-cause: 1, q-no: 16, linktype: MCP_LINK_TYPE_IP [1]
 ether hdr : dest mac: 0000.0400.0e00, src mac: 5c5a.c761.4c5f
 ether hdr : vlan: 10, ethertype: 0x8100

C9300#
```

show platform software fed switch active punt packet-capture detailed <-- Detailed provides the same inf

```
Punt packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 55 packets. Capture capacity : 16384 packets

------ Punt Packet Number: 1, Timestamp: 2023/05/04 00:17:41.709 ------
 interface : physical: TenGigabitEthernet1/0/2[if-id: 0x0000000a], pal: TenGigabitEthernet1/0/2 [if-id:
 metadata  : cause: 109 [snoop packets], sub-cause: 1, q-no: 16, linktype: MCP_LINK_TYPE_IP [1]
 ether hdr : dest mac: 0000.0400.0e00, src mac: 5c5a.c761.4c5f
 ether hdr : vlan: 10, ethertype: 0x8100

 Packet Data Hex-Dump (length: 68 bytes) :
   000004000E005C5A  C7614C5F8100000A  0806000108000604  00025C5AC7614C5F
   C0A80A0100000400  0E00C0A80A190000  0000000000000000  0000000000000000
   E9F1C9F3

 Doppler Frame Descriptor :
   fdFormat                  = 0x4          systemTtl                 = 0xe
   loadBalHash1              = 0x20         loadBalHash2              = 0xc
   spanSessionMap            = 0            forwardingMode            = 0
   destModIndex              = 0            skipIdIndex               = 0
   srcGpn                    = 0x2          qosLabel                  = 0x83
   srcCos                    = 0            ingressTranslatedVlan     = 0x7
   bpdu                      = 0            spanHistory               = 0
   sgt                       = 0            fpeFirstHeaderType        = 0
   srcVlan                   = 0xa          rcpServiceId              = 0x1
   wccpSkip                  = 0            srcPortLeIndex            = 0x1
   cryptoProtocol            = 0            debugTagId                = 0
   vrfId                     = 0            saIndex                   = 0
   pendingAfdLabel           = 0            destClient                = 0x1
   appId                     = 0            finalStationIndex         = 0x74
   decryptSuccess            = 0            encryptSuccess            = 0
   rcpMiscResults            = 0            stackedFdPresent          = 0
   spanDirection             = 0            egressRedirect            = 0
   redirectIndex             = 0            exceptionLabel            = 0
   destGpn                   = 0            inlineFd                  = 0x1
   suppressRefPtrUpdate      = 0            suppressRewriteSideEfects = 0
   cmi2                      = 0            currentRi                 = 0x1
   currentDi                 = 0x527b       dropIpUnreachable         = 0
   srcZoneId                 = 0            srcAsicId                 = 0
   originalDi                = 0            originalRi                = 0
   srcL3IfIndex              = 0x27         dstL3IfIndex              = 0
   dstVlan                   = 0            frameLength               = 0x44
   fdCrc                     = 0x97         tunnelSpokeId             = 0
   isPtp                     = 0            ieee1588TimeStampValid    = 0
   ieee1588TimeStamp55_48    = 0            lvxSourceRlocIpAddress    = 0
   sgtCachingNeeded          = 0

 Doppler Frame Descriptor Hex-Dump :
   0000000044004E04  000B40977B520000  0000000000000100  000000070A000000
   0000000001000010  0000000074000100  0000000027830200  0000000000000000
```

Many display filters are available for use. Most common Wireshark display filters are supported.

<#root>

```
C9300#

show platform software fed switch active punt packet-capture display-filter-help

FED Punject specific filters :
   1. fed.cause              FED punt or inject cause
   2. fed.linktype           FED linktype
   3. fed.pal_if_id          FED platform interface ID
   4. fed.phy_if_id          FED physical interface ID
   5. fed.queue              FED Doppler hardware queue
   6. fed.subcause           FED punt or inject sub cause
Generic filters supported :
   7. arp                    Is this an ARP packet
   8. bootp                  DHCP packets [Macro]
   9. cdp                    Is this a CDP packet
  10. eth                    Does the packet have an Ethernet header
  11. eth.addr               Ethernet source or destination MAC address
  12. eth.dst                Ethernet destination MAC address
  13. eth.ig                 IG bit of ethernet destination address (broadcast/multicast)
  14. eth.src                Ethernet source MAC address
  15. eth.type               Ethernet type
  16. gre                    Is this a GRE packet
  17. icmp                   Is this a ICMP packet
  18. icmp.code              ICMP code
  19. icmp.type              ICMP type
  20. icmpv6                 Is this a ICMPv6 packet
  21. icmpv6.code            ICMPv6 code
  22. icmpv6.type            ICMPv6 type
  23. ip                     Does the packet have an IPv4 header
  24. ip.addr                IPv4 source or destination IP address
  25. ip.dst                 IPv4 destination IP address
  26. ip.flags.df            IPv4 dont fragment flag
  27. ip.flags.mf            IPv4 more fragments flag
  28. ip.frag_offset         IPv4 fragment offset
  29. ip.proto               Protocol used in datagram
  30. ip.src                 IPv4 source IP address
  31. ip.ttl                 IPv4 time to live
  32. ipv6                   Does the packet have an IPv4 header
  33. ipv6.addr              IPv6 source or destination IP address
  34. ipv6.dst               IPv6 destination IP address
  35. ipv6.hlim              IPv6 hop limit
  36. ipv6.nxt               IPv6 next header
  37. ipv6.plen              IPv6 payload length
  38. ipv6.src               IPv6 source IP address
  39. stp                    Is this a STP packet
  40. tcp                    Does the packet have a TCP header
  41. tcp.dstport            TCP destination port
  42. tcp.port               TCP source OR destination port
  43. tcp.srcport            TCP source port
  44. udp                    Does the packet have a UDP header
  45. udp.dstport            UDP destination port
  46. udp.port               UDP source OR destination port
  47. udp.srcport            UDP source port
  48. vlan.id                Vlan ID (dot1q or qinq only)
  49. vxlan                  Is this a VXLAN packet
```

```
C9300#

show platform software fed switch active punt packet-capture display-filter arp brief

Punt packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 55 packets. Capture capacity : 16384 packets

------ Punt Packet Number: 1, Timestamp: 2023/05/04 00:17:41.709 ------
 interface : physical: TenGigabitEthernet1/0/2[if-id: 0x0000000a], pal: TenGigabitEthernet1/0/2 [if-id:
 metadata  : cause: 109 [snoop packets], sub-cause: 1, q-no: 16, linktype: MCP_LINK_TYPE_IP [1]
 ether hdr : dest mac: 0000.0400.0e00, src mac: 5c5a.c761.4c5f
 ether hdr : vlan: 10, ethertype: 0x8100

------ Punt Packet Number: 2, Timestamp: 2023/05/04 00:17:41.909 ------
 interface : physical: TenGigabitEthernet1/0/2[if-id: 0x0000000a], pal: TenGigabitEthernet1/0/2 [if-id:
 metadata  : cause: 109 [snoop packets], sub-cause: 1, q-no: 16, linktype: MCP_LINK_TYPE_IP [1]
 ether hdr : dest mac: 0000.0400.0e00, src mac: 5c5a.c761.4c5f
 ether hdr : vlan: 10, ethertype: 0x8100
<snip>
```

Filters can be applied as capture filters, as well.

<#root>

```
C9300#

show platform software fed switch active punt packet-capture set-filter arp <-- Most common Wireshark fi

Filter setup successful. Captured packets will be cleared

C9300#$e fed switch active punt packet-capture status
Punt packet capturing: disabled. Buffer wrapping: disabled
Total captured so far: 0 packets. Capture capacity : 16384 packets
Capture filter : "arp"
```

Common Scenarios

Intermittent ICMP (Ping) loss to local IP

Traffic that is forwarded to a local IP on a switch is punted in the Forus (literally “for us”) queue. Seeing incrementation in the Forus CoPP queue relates to dropped packets destined for the local switch. This is relatively straight-forward and easy to conceptualize.

In some conditions, though, there could be loss to locally destined traffic that does not neatly correlate with Forus drops.

With sufficient CPU-bound traffic flow, the punt path becomes oversaturated beyond the ability of CoPP to prioritize which traffic is policed. Traffic is ‘silently’ policed on a first-in, first-out basis.

In this scenario, evidence of control-plane policing in high volume is seen, but the traffic type of interest (Forus in this example) does not actively increment necessarily.

In summary, if there is an exceptionally high volume of CPU-bound traffic, evidenced by both active CoPP policing and demonstrated with a packet capture or FED punt debug, there could be loss that does not align to the queue you are troubleshooting. In this scenario, determine why there is an excessive amount of CPU-bound traffic and take measures to ease the burden on the control-plane.

High ICMP redirects and Sluggish DHCP operation

CoPP on the Catalyst 9000 series switch is organized into 32 hardware queues. Those 32 hardware queues align to 20 individual policer indices. Each policer index correlates with one or more hardware queues.

Functionally, this means multiple traffic classes share a policer index and are subject to a common aggregate policer value.

A common problem seen on switches with DHCP relay agents enabled involves sluggish DHCP response. Clients are able to get IPs sporadically, but it takes several attempts to complete and some clients time out.

The ICMP redirect queue and the Broadcast queue share a policer index, so a high volume of traffic that is received on and routed out of the same Switch Virtual Interface (SVI) impacts applications that rely on broadcast traffic. This is especially noticeable when the switch acts as a relay agent.

This document offers an in-depth explanation of the concept, and how to mitigate: Troubleshoot DHCP Issues on Catalyst 9000 DHCP Relay Agents

Additional Resources

Troubleshoot Slow Or Intermittent DHCP on Catalyst 9000 DHCP Relay Agents

Configure FED CPU Packet Capture on Catalyst 9000 Switches

Catalyst 9300 Switches: Configuring Control Plane Policing

Configuring Packet Capture - Network Management Configuration Guide, Cisco IOS XE Bengaluru 17.6.x (Catalyst 9300 Switches)

Operate and Troubleshoot DHCP Snooping on Catalyst 9000 Switches