cisco Live!

#CiscoLive

The bridge to possible

Advanced QoS Troubleshooting
Using Case Studies on Nexus Cloud Scale ASICs

Manoj Kumar Shukla, Data Center BU Escalation Engineer
Kallol Bosu, Technical Leader, CX

BRKDCN-3004

This is an advanced session. Goal of this session is to discuss a few advanced QoS concepts and troubleshooting using real world case studies on Nexus 9K cloud scale ASICs.

Intended audience is network engineers and admins who are interested in deep dive QoS troubleshooting on standalone NX-OS.

Manoj Kumar Shukla

Data Center BU Escalation Engineer

CISCO CERTIFIED CCIE DATACENTER #40911

manoshuk@cisco.com

Manoj Kumar Shukla, CCIE No. 40911, is a Senior Escalation Engineer in Data Center BU, with over 9 years of experience in Data Center Networking.

Within Cisco, Manoj specializes in Routing and Switching portfolio on Enterprise and Data Centre products. Since joining Cisco, he has been handling customer service requests, Escalations, design discussions with customers. He is also been instrumental in driving technology, platform specific trainings within CX , BU and to external customers.

Kallol Bosu

Technical Leader, CX

CCIE# 62833

kbosu@cisco.com

Kallol is a Technical Leader in CX, with 7 years of experience in Enterprise and Data Center Networking. Within Cisco, he specializes in Enterprise and Data Center Switching/Routing technologies across various platforms.

Since joining Cisco, he has been handling customer service requests and Escalations. He is also driving technology & platform specific trainings within CX and quite a few initiatives related to troubleshooting documentation with Business Units.

Kallol holds a Masters degree in Software and Telecommunication. He is also a CCIE# 62833 in Service Provider track.

Agenda

Case Study #1

- Problem
- Solution using PFC and Watchdog timers

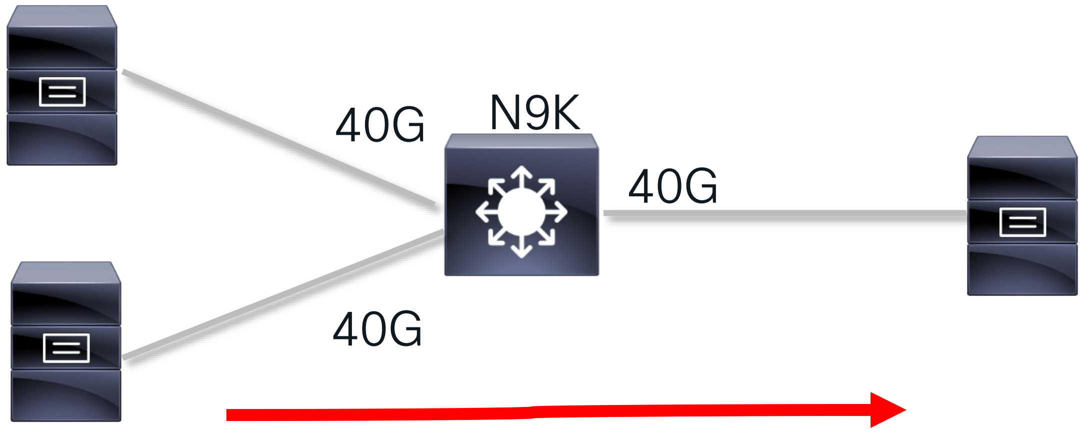
Case Study #2

- Problem
- Solution using Congestion Control (AFD, DPP, ETRAP)

Case Study #1

cisco Live!

Problem: Facing slowness with RDMA, copy operations of files, through N9K switch

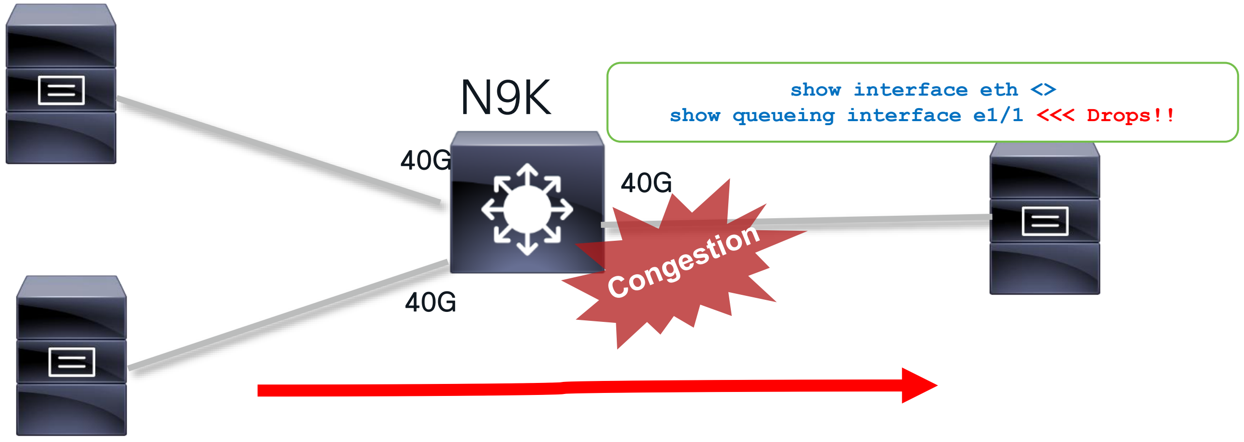


Troubleshooting Approach

Make sure following things are verified okay

- ❑ Basic IP connectivity is fine, L2/L3 is stable
- ❑ MTU is not the bottleneck
- ❑ No issues with servers/applications

Troubleshooting Approach (continued)



```
show interface eth <>
show queueing interface e1/1 <<< Drops!!
```

But we can't afford losing any packet for this flow

Potential Solutions?

1. Should we increase the buffer depth?

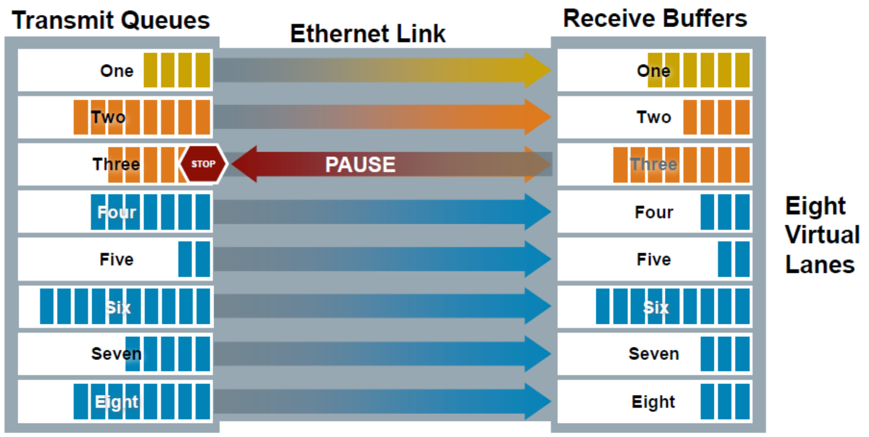
2. How about moving the affected flow to Priority queue?

!! Does not quite help

Solution- Priority Flow Control (PFC)

- A.k.a "Lossless Ethernet”
- PFC enables Flow Control on a Per-Priority basis
- Enable to have ability to have lossless and lossy priorities at the same time on the same wire
- Allows traffic to operate over a lossless priority independent of other priorities
- Other traffic assigned to other priority will continue to transmit and rely on upper layer protocols for retransmission

Flow Control Mechanism – 802.1Qbb



Solution- PFC Configuration

```
N9K(config)# class-map type qos c1
N9K(config-cmap-qos)# match cos 3
!
N9K(config)# policy-map type qos p1
N9K(config-pmap-qos)# class type qos c1
N9K(config-pmap-c-qos)# set qos-group 3
!
N9K(config)# class-map type network-qos match-any c1
N9K(config-cmap-nqos)# match qos-group 3
!
N9K(config)# policy-map type network-qos p1
N9K(config-pmap-nqos)# class type network-qos c-nq1
N9K(config-pmap-nqos-c)# pause pfc-cos 3
!
N9K(config)# system qos
N9K(config-sys-qos)# service-policy type network-qos p1
```

```
configure terminal
interface ethernet 1/1
      priority-flow-control mode on
```

Solution- PFC Verification

```
show interface priority-flow-control
slot  1
=======
============================================================
Port               Mode Oper(VL bmap)  RxPPP      TxPPP
============================================================
Ethernet1/1        Auto On   (2)      0          0
Ethernet1/2        Auto On   (2)      0          6 <<<
Ethernet1/3        Auto On   (2)      0          0
Ethernet1/4        Auto On   (2)      0          0
Ethernet1/5        Auto On   (2)      0          0
Ethernet1/6        Auto On   (2)      0          0
Ethernet1/7        Auto Off           0          0
Ethernet1/8        Auto Off           0          0
Ethernet1/9        Auto Off           0          0
Ethernet1/10       Auto Off           0          0
Ethernet1/11       Auto Off           0          0
**snip**
```

Consider enabling PFC Watchdog interval

Helps to avoid a complete Stall, following a PFC storm, caused by mad NIC , Routing loops.

It detects whether packets in no-drop queue are being drained within a specific time period.

If packets are present in buffer longer than the configured time period and after the time period expires, all outgoing packets are dropped on the interfaces that match the PFC queue that is not being drained.

Solution- PFC Watchdog Configuration

```
priority-flow-control auto-restore multiplier value
priority-flow-control fixed-restore multiplier value
priority-flow-control watch-dog-interval {on | off}
priority-flow-control watch-dog interval value
priority-flow-control watch-dog shutdown-multiplier multiplier
```

```
configure terminal
interface ethernet 1/1
      priority-flow-control watch-dog-interval on
      [disable-action] << if queue shutdown is not intended
```

If “disable-action” is set, upon detecting queue stuck condition, switch logs a syslog message and does not take queue shutdown action

Case Study#1 CLI cheat-sheet

```
show queuing interface <>
show class-map type <network-qos|qos|queuing> <class-map-name>
show policy-map type <network-qos|qos|queuing> <policy-map-name>
!
show interface priority-flow-control
show queuing pfc-queue [interface] [ethernet|ii] [detail]
!
show tech-support ipqos >> bootflash:shtechqos.txt
```

Case Study#1 Summary and Take Away

PFC with watchdog timer is a go-to QoS solution for RDMA traffic, but the scope of this solution is not just limited to RDMA only.

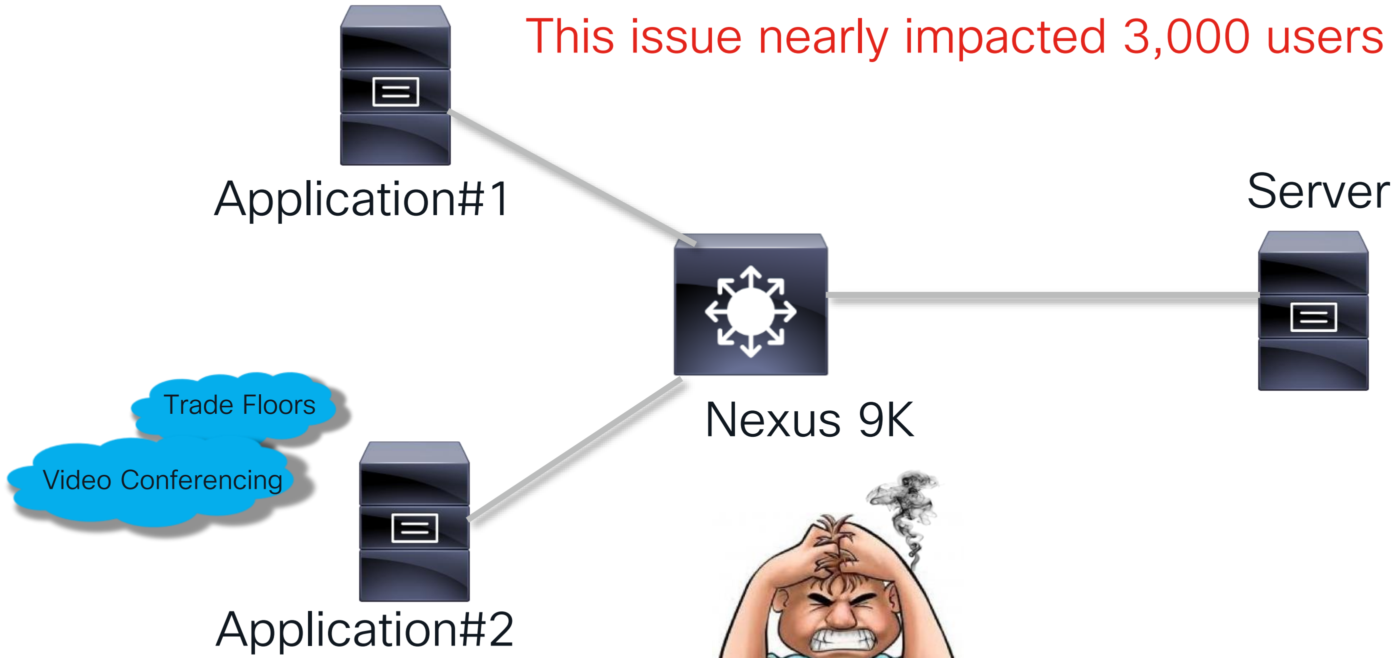
Enable PFC end to end to avoid any Head of Line (HoL) blocking issue.

Enabling ECN on PFC no drop classes is a best practice to achieve better congestion control. Aggressive ECN threshold will kick in even before PFC and help to slow down
for TCP based application traffic.

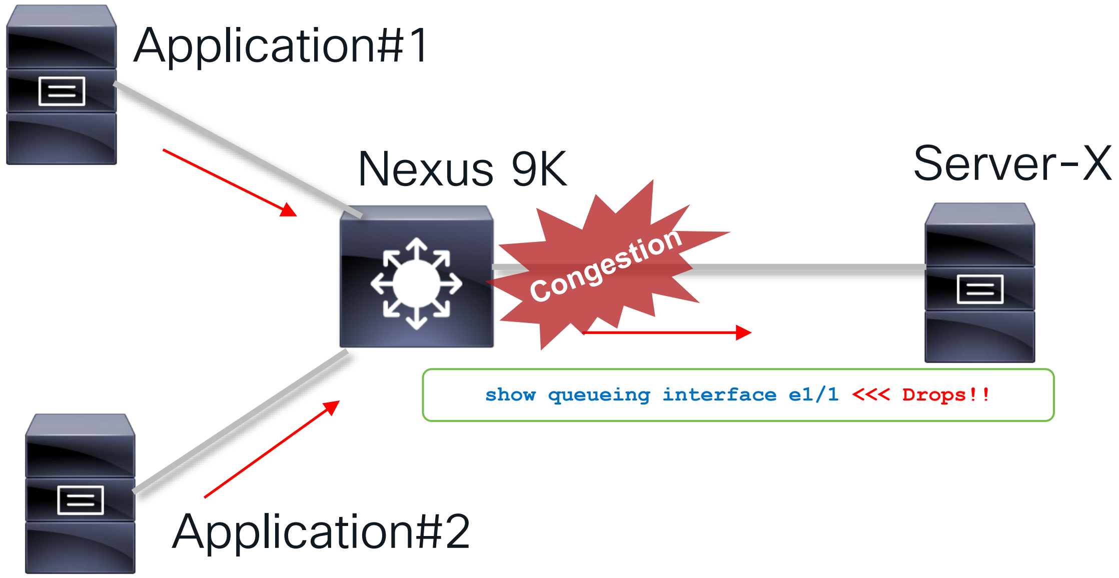
Case Study #2

Problem: Latency seen across many applications

This issue nearly impacted 3,000 users today



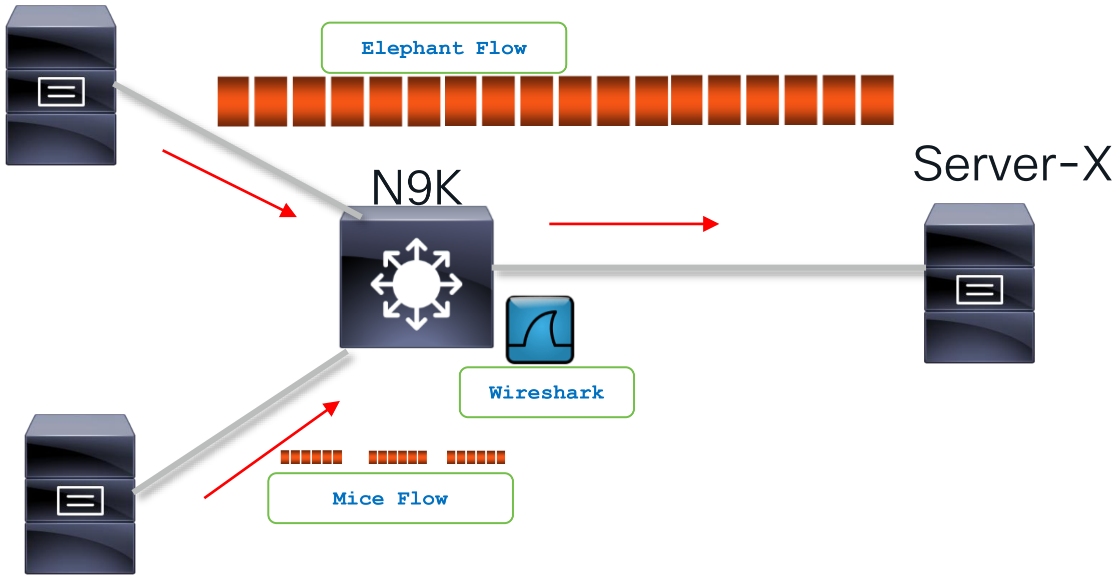
Gather more information about the problem



```
show queueing interface e1/1 <<< Drops!!
```

Troubleshooting Approach- Continued..

Application#1, Gigantic flow for data backup



Application#2, a series of small transactions

Potential Solutions?

1. Should we use Weighted Random Early Drop (WRED) ?

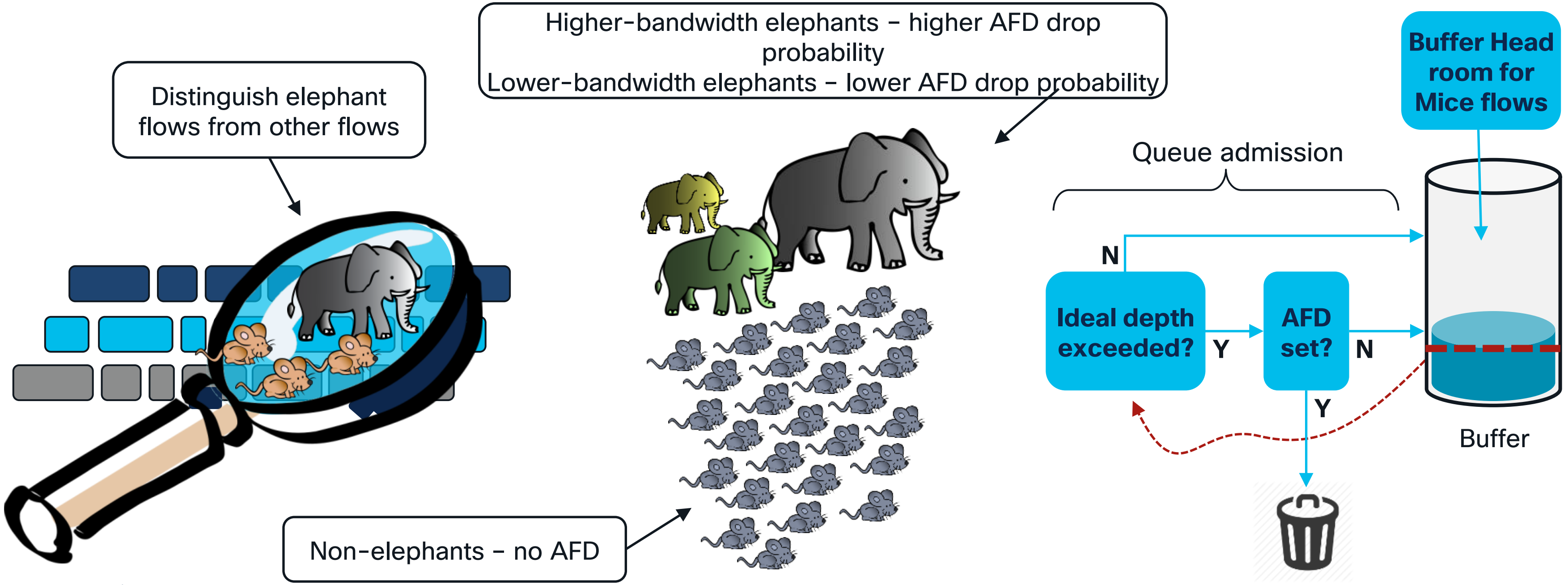
Does it help?

2. How about moving the affected mice flows to Priority queue?

Does it help?

Solution#1 Approximate Fair Drop (AFD)

Maintain throughput while minimizing buffer consumption by elephant flows – **keep buffer state as close to the ideal as possible**


Distinguish elephant flows from other flows
Higher-bandwidth elephants – higher AFD drop probability
Lower-bandwidth elephants – lower AFD drop probability
Non-elephants – no AFD
Queue admission
Buffer Head room for Mice flows
N
Ideal depth exceeded?
Y
AFD set?
N
Y
Buffer

Solution#1- AFD Deep Dive

ETRAP & AFD Algorithm

Flows with a byte count < ETRAP byte-count = Mice Flow
Flows with a byte count > ETRAP byte-count = Elephant Flow

Flow byte count < bandwidth threshold for the duration of age-period << NOT an Elephant Flow

Note !!

Default ETRAP Parameters

```
N9K# show run all | in etrap
hardware qos etrap age-period 500 usec
hardware qos etrap byte-count 1048555
hardware qos etrap bandwidth-threshold 500 bytes
```

Approx. 1 MB

Solution#1- AFD Configuration

ETRAP & AFD Algorithm

```
N9K# show policy-map type queuing afd_8q-out

  Type queuing policy-maps
  ========================

  policy-map type queuing afd_8q-out
      class type queuing c-out-8q-q7
            priority level 1
     ***snip***
      class type queuing c-out-8q-q-default
            afd queue-desired 1 mbytes
            bandwidth remaining percent 100
```

The only parameter to configure for AFD within a class-based queue is to set the desired queue depth. It controls when AFD starts to apply algorithm-based drop to elephant flows. In this example, it starts when the queue grows to 1MB.

Solution#1- AFD Configuration (continued)

Apply the policy-map to given interface OR system QoS

```
N9k(config)# int e1/1
N9k(config-if)# service-policy type queuing output afd_8q-out
```

```
N9k(config)# System qos
N9k(config-if)# service-policy type queuing output afd_8q-out
```

- Define the ETRAP parameters - To differentiate Elephant/Mice Flow
- Activate AFD in a queuing policy-map by defining the desired queue depth.
- Apply the queuing policy to the system or egress interface in question.

Solution#1- AFD Verification

```
N9K# show interface hardware-mappings >>> to find out the slice
OR
N9K# show internal ethpm info interface ethernet 1/1
IF_STATIC_INFO: port_name=Ethernet1/1,if_index:0x1a000000,ltl=6144,slot=0,
nxos_port=0,dmod=1,dpid=20,unit=0,queue=65535,xbar_unitbmp=0x0,ns_pid=255,
slice_num=0,port_on_slice=20,src_id=40

N9K# attach module 1

module-1# show hardware internal tah afd hw-info asic 0 slice 0
Approximate Fair Dropping
=========================
unit: 0 slice: 0 slice_port: 4 oqueue: 3 <snipped>
mode           : 2
avg_timer: 0
avg_wt_alpha   : 0
drop_en        : 1 >> 1 -drop, 0 -ECN
mfair          : 2047
cur_uc_oq_qdepth : 0 bytes
```

Solution#1- AFD Verification (continued)

```
module-1# debug hardware internal dav dump asic 0 slice 0 table
tah_dav_lbx_etrapelephants | "key_vld=0x00000001”
```

Dump the HW hash table holding all elephant flows in ingress slice

```
module-1# debug hardware internal dav dump asic 0 slice 0 table
tah_dav_lbx_cnt_etrap_elephant_aliased
```

Dump HW counters for number of flows hashed

```
module-1# debug hardware internal dav dump asic 0 slice 0 table
tah_dav_lbx_cnt_etrap_elephant_dead
```

Dump HW counters for number of flows dead

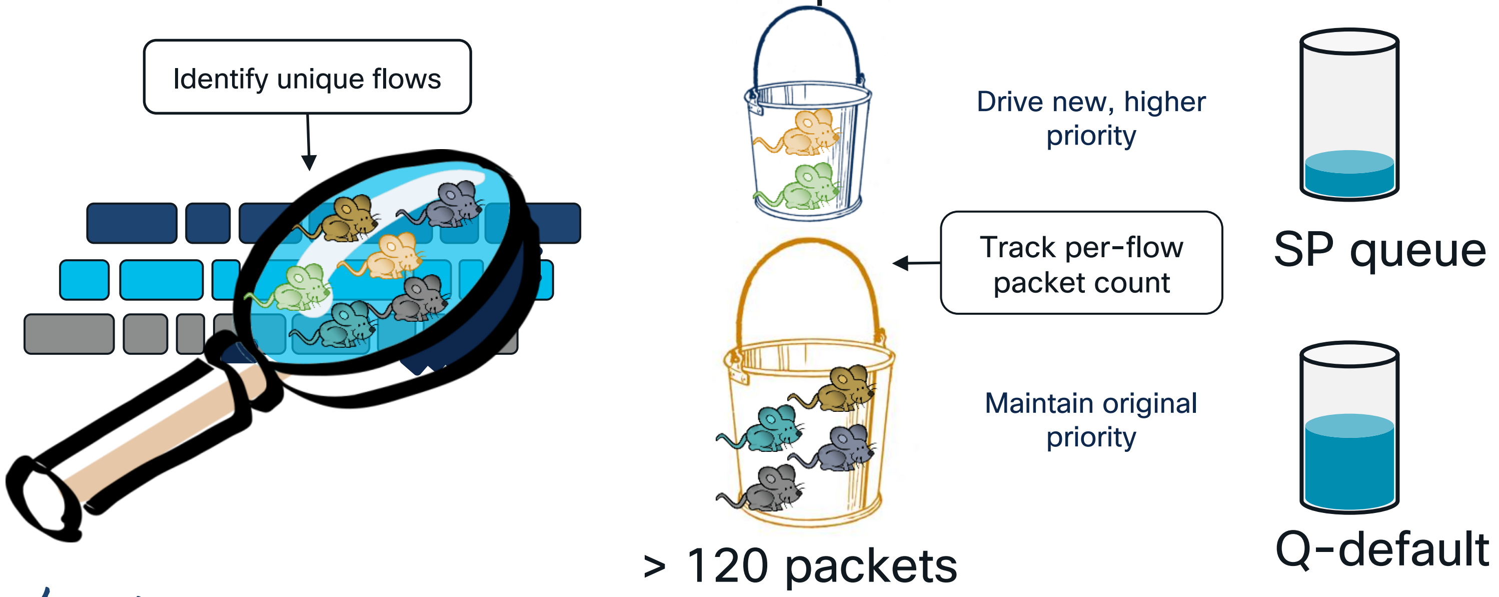
Challenges with Solution#1 (AFD)

1. Too many Mice flows from different critical applications
2. Need to prioritize traffic but cannot limit it to a particular network
3. With AFD alone, Elephant and Mice flows are in same queue

What if we can dedicate a separate queue for Mice flows ?

Solution#2 Dynamic Packet Prioritization (DPP)

- Prioritize initial packets of new / short-lived flows
- Up to first 120 packets of each flow assigned to higher-priority qos-group



<= 120 packets
Identify unique flows
Drive new, higher priority
Track per-flow packet count
SP queue
Maintain original priority
Q-default
> 120 packets

Solution#2- DPP Configuration

```
hardware qos dynamic-packet-prioritization age-period 5000 usec
hardware qos dynamic-packet-prioritization max-num-pkts 120
```

DPP knobs : Default values

```
policy-map type network-qos dpp
      class type network-qos-c-8q-nq-default
      dpp set-qos-group 7
      mtu 1500
!
system qos
      service-policy type network-qos dpp
```

In this example, DPP is enabled in class default. The first N packets (N= DPP max-num-pkts) in a flow in class-default will be set with qos-group 7. These will go into the queue for qos-group 7 on the egress port.

Solution#2- DPP Configuration (continued)

```
Queueing Policy on Egress Ports:

policy-map type queuing afd_8q-out
      class type queuing c-out-8q-q7
            priority level 1
      <snipped>
      class type queuing c-out-8q-q-default
            afd queue-desired 1 mbytes
```

First N packets will get into q7 and get prioritized

AFD can be optionally configured in same queuing policy

```
N9k(config)# int e1/1
N9k(config-if)# service-policy type queuing output afd_8q-out
```

Solution#2- DPP Verification

```
module-1# show hardware internal tah dpp hw-info asic 0 slice 0
Dynamic Packet Prioritization
==============================
unit: 0 slice: 0
prio_age_period: 0x5f5e10  << age-period
prio_disable   : 0
max_num_pkts   : 119 <<< packet-count threshold
En             : 1
Oclass         : 0x76543270 <<< note q 1 is mapped to q 7
use_local_crc. : 0
crc_hash_sel.  : 0
```

```
show queuing interface ethernet 1/1
```

To display QoS statistics of packets egressing out of the interface

Solution#2- DPP Verification (continued)

```
module-1# debug hardware internal dav dump asic 0 slice 0 table
tah_dav_lbx_cnt_num_flows_seen
ENTRY[0] = { value=0x00000000:0x00000002 }
```

Dump HW counters for number of flows hashed

```
module-1# debug hardware internal davdump asic 0 slice 0 table
tah_dav_lbx_cnt_highprio_pkt
ENTRY[0] = { value=0x00000000:0x000000f0 }
```

Dump HW counters for number of packets prioritized

Case Study#2 CLI cheat-sheet

FYI

```
show queuing interface <>
show run all | in etrap
show class-map type <network-qos|qos|queuing> <class-map-name>
show policy-map type <network-qos|qos|queuing> <policy-map-name>
show interface hardware-mapping
show system internal ethpm info interface <>
show hardware internal tah afd hw-info asic <> slice <>
!
debug hardware internal dav dump asic <> slice <> table
!
show hardware internal tah dpp hw info asic <> slice <>
!
show tech-support
show tech-support ipqos >> bootflash:showtechQoS.txt
```

Case Study#2 Summary and Take Away

In a Data Center network, it is inevitable to have several elephant flows, and mice flows that will be mixed in the same queue on a network link.

Their different forwarding requirements should be factored into the queue management scheme.

Cisco Nexus 9000 cloud-scale ASICs introduce intelligent buffer management, including AFD with ETRAP and DPP. With these mechanisms, switches can distinguish mice flows and elephant flows and apply different queue management techniques to them.

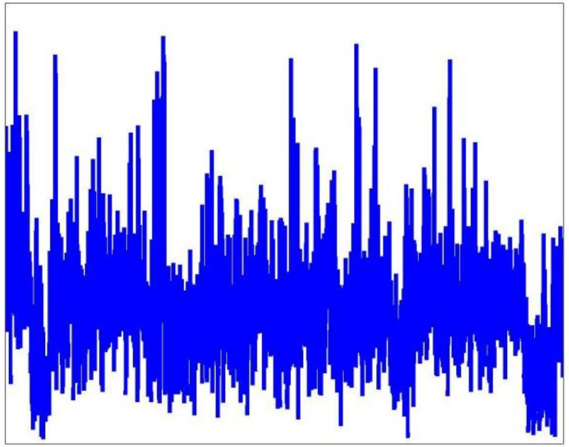
Additional Info: Microburst Monitoring

cisco Live!

Microburst

Why it is important to monitor ?

- Spike of activity –may result in the system resource exhaustion / saturation
- How short and how high? –Capacity of "weakest" system in the network
- Not captured by traditional load-monitoring tools



Microburst Monitoring

Allows monitoring traffic to detect unexpected data bursts

Detected when egress queue rises above a configured threshold

Microburst : Monitoring Configuration

```
N9K(config)# policy-map type queuing micro-burst-monitor
N9K(config-pmap-que)# class type queuing c-out-def
N9K(config-pmap-c-que)# burst-detect rise-thresh 208 bytes fall-thresh 208 bytes
```

```
Nexus3264Q# show queuing burst-detect detail
slot  1
--------------------------------------------------------------------------
        Out Of Band Statistics
--------------------------------------------------------------------------
Ethernet |Queue|Pipe |Start Depth|        Start Time         |Peak Depth|
Interface|     |     |  (bytes)  |                           | (bytes)  |
---------------------------------------------------------------------------
Ether1/23| 0   | XPE-A |   23000 | 2021/01/02 16:43:12:227129 | 24174    |
```

```
------------------------------------------------------------------------------------
Peak Time               |End Depth|             End Time             |Duration(nsecs)
|  (bytes)  |                                   |
------------------------------------------------------------------------------------
2021/01/02 16:43:12:239457 | 22850   | 2021/01/02 16:43:12:241236 | 14 msec
```

Microburst : What to do next?

It really depends on what type of traffic is causing the burst.
Based on that, you may do one of the followings–

1. Investigate further from application side, to understand why it is sending bursts, if that is legitimate or not and if that can be restricted.

2. Perform one of the approaches mentioned in case study#1 and 2 (AFD, DPP, PFC etc.), solely depending on how you want to treat that traffic.

Conclusion

The algorithm-based intelligent buffering and scheduling approach on Nexus 9000 Series Switches address the real-world network congestion problems caused by traffic bursts more efficiently, and demonstrated overall better application performance in comparison to the deep buffer approach.

Cisco Systems Speeding Applications in Data Center Networks
http://miercom.com/cisco-systems-speeding-applications-in-data-center-networks/

More details on QoS & Architecture

QoS fundamentals, basic configuration

End-to-End QoS Implementation and Operation with Cisco Nexus Switches -BRKDCN-3346 by Nemanja Kamenica

Cloud scale ASIC/N9K architecture, packet forwarding

Cisco Nexus 9000 Architecture -BRKARC-3222 by Timothy Stevenson

The bridge to possible

Thank you

cisco Live!

#CiscoLive