Troubleshooting Cisco Nexus 7000 Series Switches

Jarod Xueyi Zhu, Premium Engineer

VIP TAC 技术支持专家 ;7年TAC技术支持中心工作经验；Catalyst 6500 Team Leader

VCP,RHCE

Agenda

- Before You Get Started
  - NX-OS Troubleshooting Approach
  - Nexus 7000 Built-in Troubleshooting Tools
  - Architecture Overview
- Troubleshooting
  - CPU
  - Control Plane

Before You Get Started

Troubleshooting Mind Map

What ..

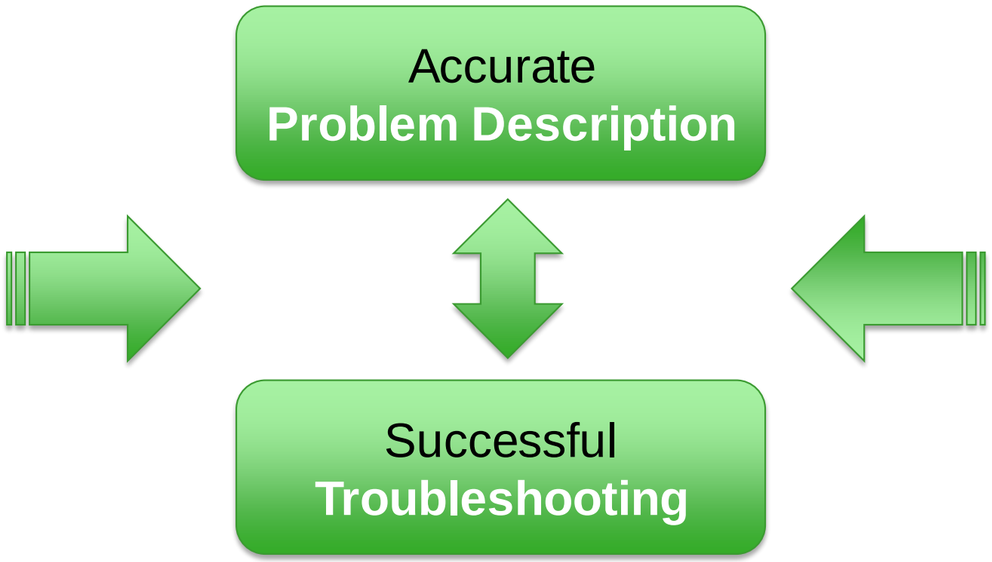
is broken or not functioning as expected

Why ..

is it broken and is there a workaround

When ..

the functionality broke or started to misbehave



Platform..

knowledge: Hardware, Software, Features, and Troubleshooting capabilities

Topology ..

knowledge and data path through topology

Interaction ..

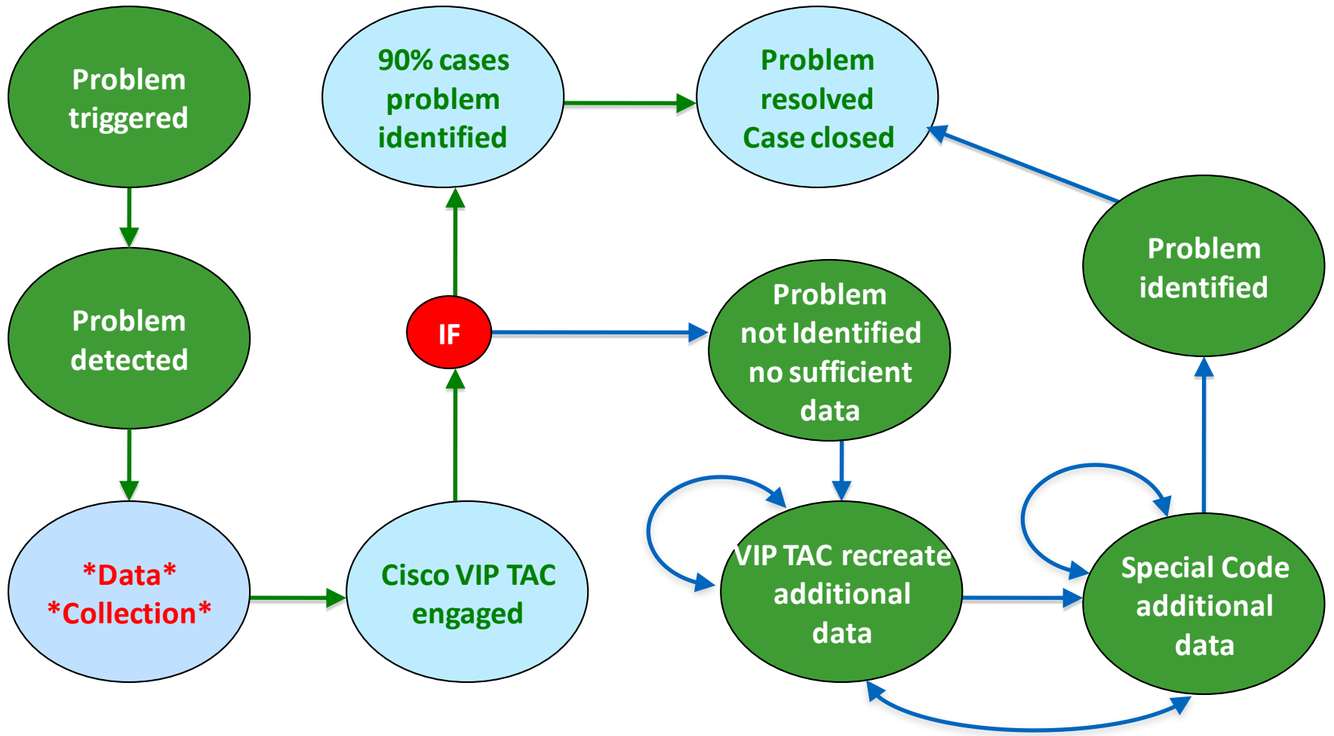
between Cisco devices and other vendors equipment's

Before You Get Started

NX-OS Approach

Facts

- NX-OS debugging and troubleshooting tools are very rich, and allow engineers to accurately assess the situation
- Customized ‘show techs” make the collection of related information accurate and quick.



Before You Get Started

Troubleshooting Approach (Cont.)

Suggestions

- Identify detection and trigger time as accurately as possible to set ‘good’ start up point for collected data search and analysis
- Minimize delta time between trigger/detection time and data collection time
- Try to recall all activities before trigger/detection time
- Get proficient as much as possible with built-in tool box
- Get familiar with specific feature troubleshooting cli, feature show tech-support output for on-the-fly troubleshooting and analysis

Remember ..

- Internal data logs have limited size, adjust them ahead of time for relevant features you have deployed
- Even max-ed log size may not prevent data wrap up
- Use configuration rollback or other configuration backup method while troubleshooting and making configuration changes
- Forensic data survives reload or switchover via ‘Onboard logging’, ‘accounting-log’, ‘nvram’

Agenda

- Before You Get Started
  - Traditional Versus NX-OS Troubleshooting Approach
  - Nexus 7000 Built-in Troubleshooting Tools
  - Architecture Overview
- Troubleshooting
  - CPU
  - Control Plane

Built-In Troubleshooting Tools

Make Troubleshooting Easier and more Effective — Almost Fun to Do ☺

Powerful show cli

Standard CLI:

- Platform independent (PI) and dependent (PD) output
- Hardware keyword indicates platform hardware specific output

Engineering CLI

- Internal keyword
- No XML or SNMP support

Event-history logging

- Extensive feature and software component event-history logging
- Permanent engineering debugs output of process Finite State Machine (FSM)

Logflash logging

- Extensive system activity logging to dedicated logflash with filtering to display only ‘what I want to see

Built-In Troubleshooting Tools

Make Troubleshooting Easier and more Effective — Almost Fun to Do ☺

Onboard & Accounting

Onboard logging, accounting log logging (config and exec)

- Forensic data surviving reload and switchover
- Hardware component events and manipulation activity
- Use it to ‘recall’ all activity around ‘trigger and detection’ time

GOLD system

- A diagnostic framework to detect hardware failures while the system is online and operational
- Test types:
  - Bootup
  - Health Monitoring
  - On-demand
  - Scheduled

Standard tools

- Ping, Traceroute
- Span, Netflow, XML, EEM

Built-In Troubleshooting Tools

Make Troubleshooting Easier and more Effective — Almost Fun to Do ☺

Debugs

- Traditional feature related debugs e.g.

debug ip packet protocol igmp , debug ipv6 icmp, debug icmp

- NX-OS debugs with debug-filter, e.g.

debug-filter ip packet direction inbound

ASIC info

- Easy to read asic counters and registers
- Software copy not clear-on-read, must use clear cli to clear them
- Comprehensive per module, ASIC, port, counter category filtering

ELAM & Ethanalyzer

- Embedded Logic Analyzer Module (ELAM capture) provides detailed frame's internal header info
- Built-in wireshark analyzer capturing mgmt interface and CPU traffic. The output can be redirected to a text file with no performance impact

Agenda

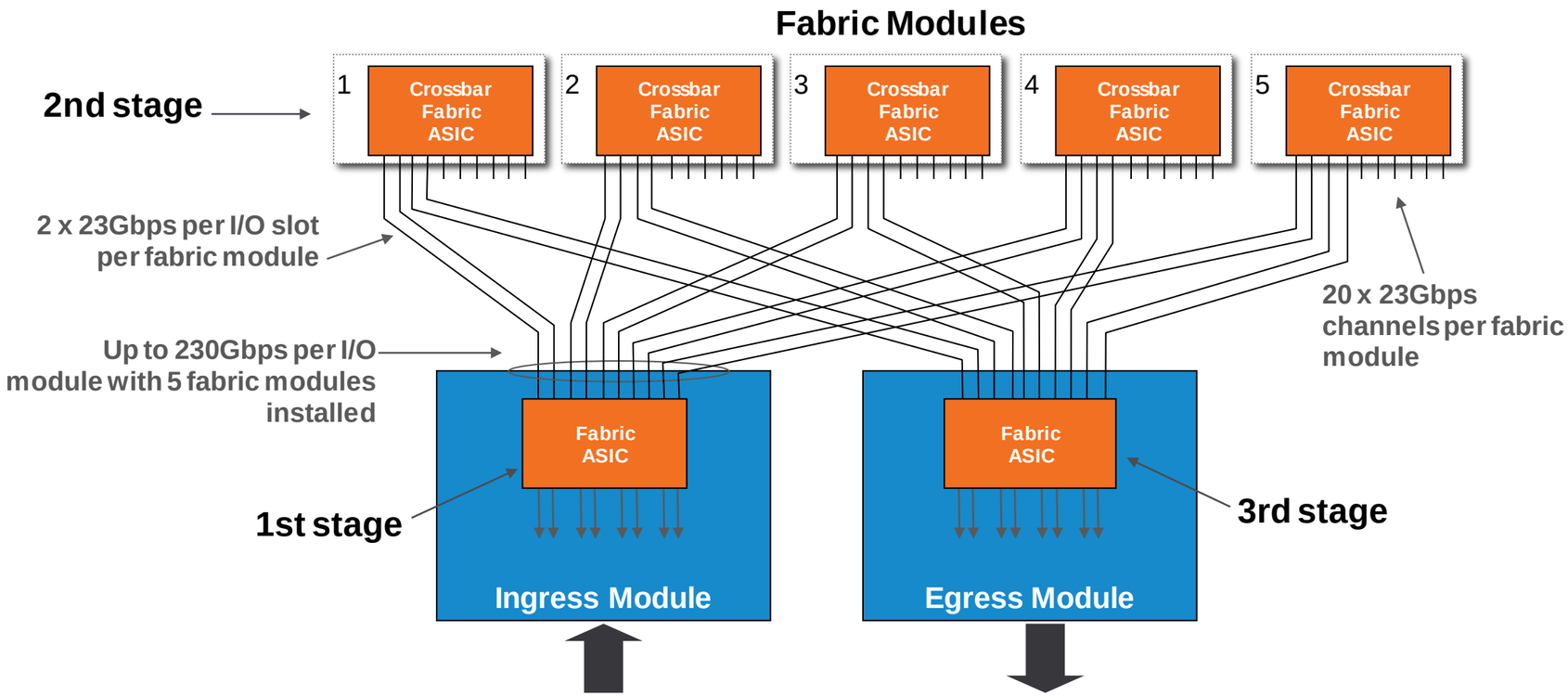
- Before You Get Started
  - Traditional Versus NX-OS Troubleshooting Approach
  - Nexus 7000 Built-in Troubleshooting Tools
  - Architecture Overview
- Troubleshooting
  - CPU
  - Control Plane

Built-In Troubleshooting Tools

System Architecture — Multistage Switch Fabric

Facts

- Nexus 7000 implements 3-stage switch fabric
- Stages 1 and 3 on I/O modules
- Stage 2 on xbar modules

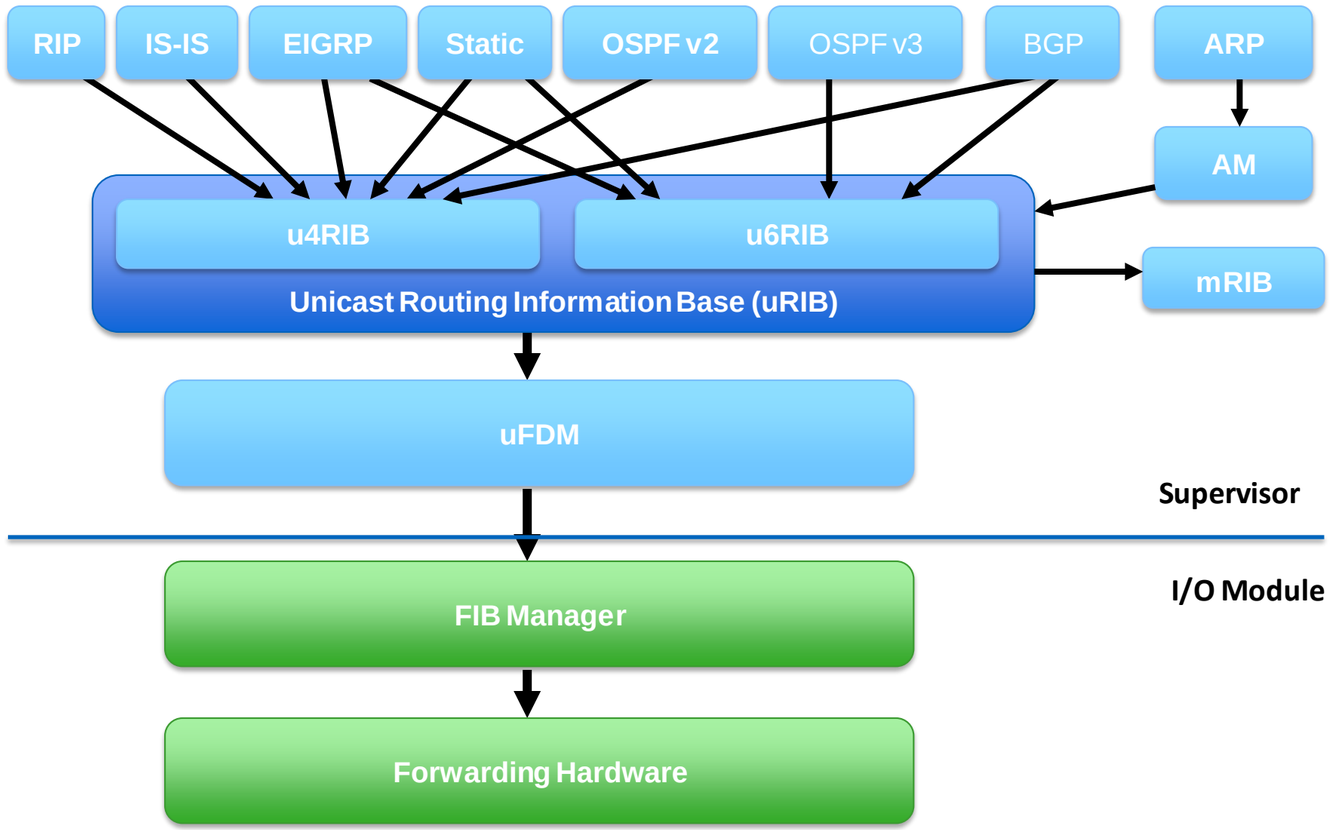


Built-In Troubleshooting Tools

Architecture — Unicast Routing Software Architecture

Facts

- uRIB digests all routing related information and builds the final routing table.
- Unicast Forwarding Distribution Module (UFDM) distributes forwarding information to Modules.
- FIB programs forwarding info on Modules.

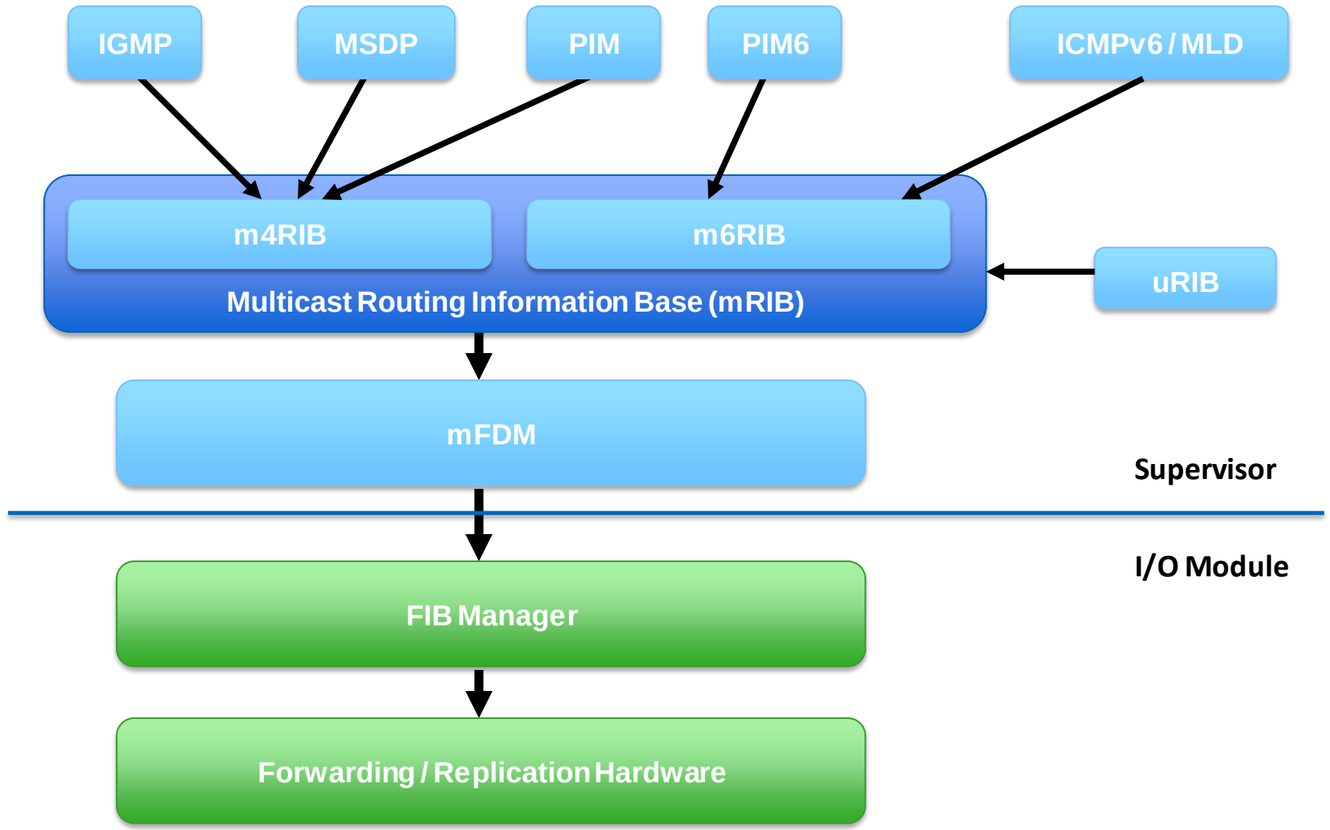


Built-In Troubleshooting Tools

Architecture — Multicast Routing Software Architecture

Facts

- mRIB adds routes, OIFs and handles updates when RPF changes
- mFDM distributes forwarding information to Modules.
- FIB programs forwarding info and MET tables on Modules



Agenda

- Before You Get Started
  - Traditional Versus NX-OS Troubleshooting Approach
  - Nexus 7000 Built-in Troubleshooting Tools
  - Architecture Overview
- Troubleshooting
  - CPU
  - Control Plane

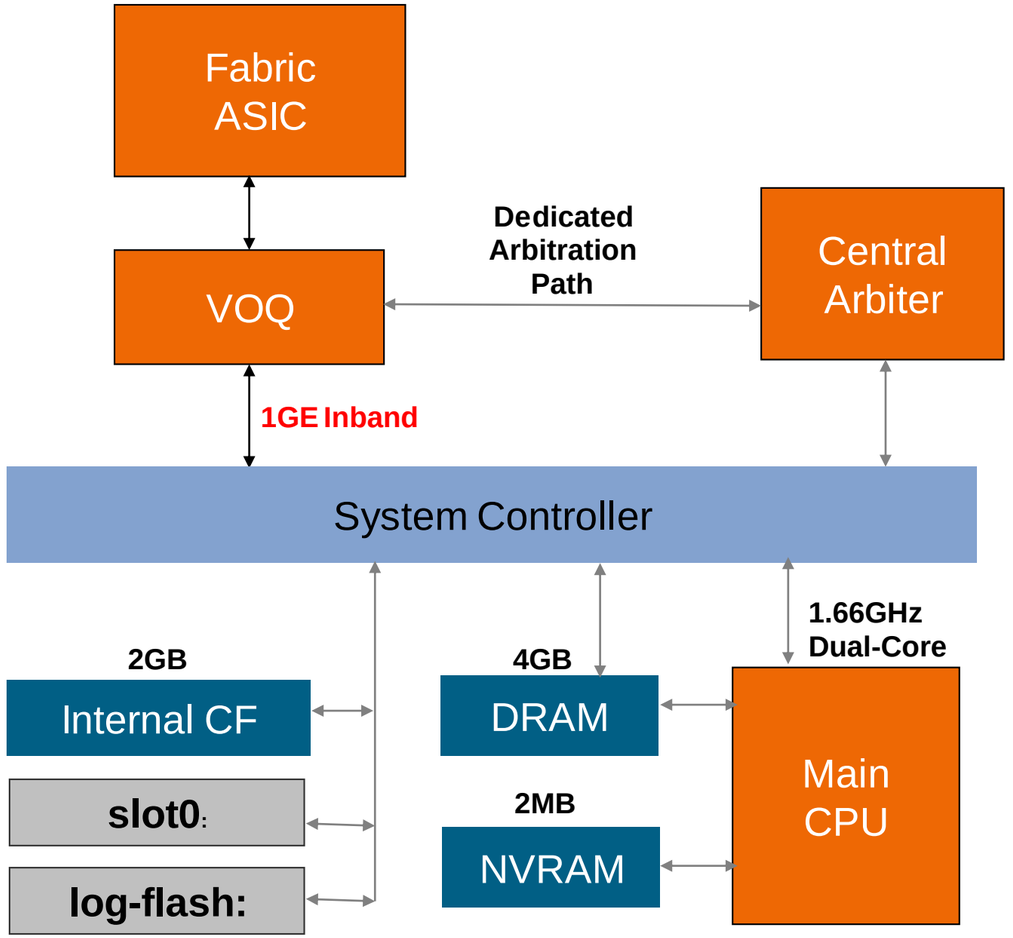
Troubleshooting

CPU — Is there a Problem?

Should I Panic?

High CPU utilization is not automatically problem indication!

- NEXUS 7000 is dual core linux based system with robust preemptive scheduler (one functional unit for both rp and sp)
- Strict control-plane and data-plane separation
- Scheduler assures fair access to CPU for all processes
- Lower level processes (drivers) run in FIFO or non-preemptive mode



Troubleshooting

CPU — Causes

Process

Misbehaving process(s)

- Consume CPU cycles which impact normally-functioning processes
- Delay or prevent CPU from processing control traffic
- Usually triggered by a software bug, but it might be a product of a network event

Traffic

Unexpected traffic

- Excessive CPU bound traffic, control-plane churn
- Acess-list processing, hardware programming
- Possible typical data center traffic (arp, ipv6 nd, etc)

Troubleshooting

CPU — Supervisor, General Health Check

```
N7k-3-VDC3# show system resources
Load average:   1 minute: 0.64   5 minutes: 1.08   15 minutes: 1.30
Processes    :   3912 total, 2 running
CPU states   :   4.5% user,   5.0% kernel,   90.5% idle
Memory usage:   4115232K total,   3434268K used,    680964K free
```

How many processes were scheduled to run in average per whole system in last 1, 5 and 15 minutes

```
N7k-3-VDC3# show processes cpu history

     1 2 111 11111211233 1 1 111 1 1 1 6 112 1 1 21132 1 111 123
    919275058862141899918384800583739174756080779143297264026770
100
 90
 80
 70
 60                                  #
 50                                  #
 40                    #             #
 30                   ###            #            ##        ##
 20    # # # ##   ######       #      # ###   # #  ##       ###
 10 ##########################################################
    0....5....1....1....2....2....3....3....4....4....5....5....
               0    5    0    5    0    5    0    5    0    5

                CPU% per second (last 60 seconds)
                       # = average CPU%
```

How much of CPU cycles are used by user configured processes and kernel processes Output IS calibrated for 2 cores

CPU utilization 60 seconds ago

Troubleshooting

CPU — Identify the offending process(s)

```
N7K-3-VDC3# show system internal processes cpu
top - 14:01:06 up 21 days, 15:35,  4 users,  load average: 0.77, 0.73, 1.07
Tasks: 3257 total,   1 running, 422 sleeping,   0 stopped, 2834 zombie
Cpu(s):  5.8%us,  6.0%sy,  0.1%ni, 84.1%id,  0.4%wa,  0.1%hi,  3.4%si,
0.0%st
Mem:   4115232k total,  3875988k used,   239244k free,    82400k buffers
Swap:        0k total,        0k used,        0k free,  1817776k cached


PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
22683 root      20   0  182m  63m  14m S 93.7  1.6 636:17.84 netstack
29391 admin#03  20   0  5364 3312 1140 R 22.3  0.1   0:00.30 top
 3892 root      20   0  164m  54m  23m S  6.0  1.3   1095:00 netstack
 4149 root      20   0  111m  41m  19m S  4.5  1.0 994:43.26 stp
 3175 root      20   0 78100  19m  17m S  3.0  0.5 175:07.02 diagmgr
23028 root      20   0  101m  23m 9968 S  3.0  0.6 598:14.57 stp
 3181 root      20   0 77684 4564 3352 S  1.5  0.1   0:30.35 securityd
 3591 root      20   0  222m  13m 7132 S  1.5  0.3   0:09.61 igmp
 4753 root      20   0  162m  45m  16m S  1.5  1.1  34:59.22 netstack
    1 root      20   0  1988  612  532 S  0.0  0.0   0:16.32 init
```

Use X | no-more, where X is interval in seconds to get more snapshots

- Equivalent of Linux TOP monitoring tool output showing system processes across all vDCs
- Use it to cross check accuracy of ‘show system resources’ output
- Output is NOT calibrated for 2 cores so it would be expected to see 2 processes using 100% CPU
- Output show processes from all vDCs

Troubleshooting

CPU — Examine the offending process(s)

```
N7K-3-VDC3# show processes cpu | egrep "PID|--|ospf"
PID     Runtime(ms)   Invoked    uSecs  1Sec     Process
-----   -----------   --------   -----  ------   -----------
 9337           102         72   1418    0.0%   ospfv3
22916           118         62   1905   13.1%   ospf
```

```
N7K-3-VDC3# show system internal sysmgr service pid 22916
Service "__inst_001__ospf" ("ospf", 58):
        UUID = 0x41000119, PID = 22916, SAP = 320
        State: SRV_STATE_HANDSHAKED (entered at time Thu Mar  3 21:53:59 2012).
        Restart count: 1
        Time of last restart: Thu Mar  3 21:53:58 2011.
        The service never crashed since the last reboot.
        Tag = 6467
        Plugin ID: 1
```

```
N7K-3-VDC3# show system internal sysmgr service name ospfv3 tag 8893
Service "__inst_001__ospfv3" ("ospfv3", 59):
        UUID = 0x4100011A, PID = 9337, SAP = 328
        State: SRV_STATE_HANDSHAKED (entered at time Fri Mar 25 22:33:10 2012).
        Restart count: 2
        Time of last restart: Fri Mar 25 22:33:09 2011.
        The service never crashed since the last reboot.
        Tag = 8893
        Plugin ID: 1
```

PID – Process ID

Runtime – total non-idle time process has been actively using CPU

Invoked – number of times process has been context switched voluntary (finished job) and involuntary (scheduler interrupt)

uSecs - average amount of time process was running during a single context switch

Useful process level details

For testing purposes, process was manually restarted using ‘restart ospfv3 8893’ cli

Troubleshooting

CPU — Traffic Causes High CPU Utilization and Control-Plane Instability

Attackers

Typical “offending” datacenter

- ARP, ND (IPv6)
- DHCP traffic
- Glean traffic (no ARP or ND)
- Malicious traffic to 224.0.0.0/24 subnet
- Fragments or malicious L2 mcast or ‘other’ traffic

Remember:
misbehaving “expected” traffic, such as OSPF packets, might be a dangerous attacker as well

Defense

- CPU protection via CoPP policers
- CPU protection via L2/L3 hardware rate-limiters (RL)
- CoPP and RL default settings may need tweaking based on network requirement specifics
  - Both are configured/enabled per M1 I/O Module
  - Total inband traffic allowed is the sum across all M1 I/O Modules

Troubleshooting

CPU — Traffic Causes High CPU Utilization and Control-Plane Instability

Problem

OSPF neighbor's failing to come up.

- Syslog messages report OSPF neighbor failures
- CPU states show high utilization caused by OSPF and Netstack process.

```
N7K-1-VDC2# show system resources
Load average:   1 minute: 2.92   5 minutes: 2.38   15 minutes: 2.27
Processes    :   1267 total, 4 running
CPU states   :   34.0% user,   42.5% kernel,   23.5% idle
Memory usage:   4115232K total,   3638780K used,    476452K free

N7K-1-VDC2# show processes cpu sort

PID    Runtime(ms)   Invoked   uSecs  1Sec    Process
-----  -----------  --------  -----  ------  -----------
 3981          127       276    462   43.2%  ospf
 3841          267        78   3427   16.4%  netstack
 2941     34146488   7377876   4628    0.9%  platform
 3982          118       245    485    0.9%  ospfv3

2011 Mar 26 15:38:56.395 N7K-1-VDC2 %OSPF-5-NBRSTATE:  ospf-6467 [3981]  Process
6467, Nbr 192.251.19.22 on Vlan19 from INIT to DOWN, DEADTIME
2011 Mar 26 15:38:56.584 N7K-1-VDC2 %OSPF-5-NBRSTATE:  ospf-6467 [3981]  Process
6467, Nbr 192.251.19.22 on Vlan19 from DOWN to INIT, HELLORCVD
2011 Mar 26 15:39:33.865 N7K-1-VDC2 %OSPF-5-NBRSTATE: ospf-6467  [3981]  Process
6467, Nbr 192.251.19.22 on Vlan19 from INIT to DOWN, DEADTIME
2011 Mar 26 15:39:35.754 N7K-1-VDC2 %OSPF-5-NBRSTATE:  ospf-6467 [3981]  Process
6467, Nbr 192.251.19.22 on Vlan19 from DOWN to INIT, HELLORCVD
```

Troubleshooting

CPU Traffic — Inband stats

```
N7K-1# show hardware internal cpu-mac inband stats | egrep "  Rx|
Tx|counters|Throttle|Tick|rate|total|good|XOFF p|XON p"
RMON counters                              Rx                   Tx
total packets                       779905245           1421785114
good packets                        779905245           1421650279
good octets (hi)                            0                    0
good octets (low)                172303021767         192965708376
total octets (hi)                           0                    0
total octets (low)               172302724342         192974265660
XON packets                                 0                67627
XOFF packets                                0                67208
Interrupt counters
Error counters
Throttle statistics
Throttle interval ..... 2 * 100ms
Packet rate limit ..... 32000 pps
Tick counter ..... 12414130
Rx packet rate (current/max)  4993 / 20296 pps
Tx packet rate (current/max)  60 / 3474 pps
--snip--
```

The Challenge

how to identify offending traffic type and its source

Total number of frames received and send by CPU

Hard coded maximum limit, with larger packet size, this number may not be reached

How many times did throttling kicked in

CPU bound traffic current pps /maximum pps reached

Troubleshooting

CPU Traffic — Pktmgr debugs

```
N7K-1-VDC2# show system internal pktmgr interface vlan 64
Vlan64, ordinal: 117
  SUP-traffic statistics: (sent/received)
    Packets: 3771848 / 40687558
    Bytes: 304360445 / 36018498390
    Instant packet rate: 0 pps / 4951 pps
    -- snip --
```

Use this cli first without specific interface to identify the 'offending' traffic - the one with the highest rate. Alternatively, use 'show system internal pktmgr internal vdc inband' which identifies vDC interfaces and number of packet sent to the CPU

```
N7K-1-VDC2# debug pktmgr frame
2011 Mar 26 21:22:30.599670 netstack: In  Vlan 64 0x0800 992  7
0000.1301.1301 -> 0100.5e00.0005 Vlan64
```

debug-filter pktmgr vlan 64

Offending host mac

```
N7K-1-VDC2# show ip arp vlan 64 | i 0000.1301.1301

N7K-1-VDC2# show mac address-table address 0000.1301.1301 vlan 64
VLAN     MAC Address      Type      age     Secure NTFY Ports/SWID.SSID.LID
---------+-----------------+--------+---------+------+----+------------------
-
64       0000.1301.1301    dynamic   0          F     F  Eth2/9
```

No ARP entry??

Source Port

Troubleshooting

CPU Traffic — Other Capture methods

Debug the offending process

```
N7K-1-VDC2# debug-filter ip ospf interface vlan 64
N7K-1-VDC2# debug logfile offending_traffic
N7K-1-VDC2# show debug logfile offending_traffic

2011 Mar 26 23:33:25.992586 ospf: 6467 [3981]
(default) rcvd: prty:7 ver:2 t:HELLO len:44
rid:0.0.0.0 area:0.0.0.0 crc:0xfdd2 aut:0 aukid:0 from
192.253.64.254/Vlan64
2011 Mar 26 23:33:25.992780 ospf: 6467 [3981] Invalid
src address 192.253.64.254, should not be seen on
Vlan64
```

Ethanalyzer

- Ethanalyzer can be used to capture the traffic that is taking the inband interface to the CPU.
- Write the capture output to a pcap file and open it using wireshark for analysis
- If still more digging is needed, use a more specific trigger to narrow down the search offending host (s)

Agenda

- Before You Get Started
  - Traditional Versus NX-OS Troubleshooting Approach
  - Nexus 7000 Built-in Troubleshooting Tools
  - Architecture Overview
- Troubleshooting
  - CPU
  - Control Plane

Troubleshooting

CoPP — Essentials

Goal

CoPP protects the SUP against the following classes of traffic

- Control Plane packets, such as Protocols Hellos and other Receives
- Data Plan transit packets, such as Glean, Exceptions, and Redirects
- Management Plane packets, such as SNMP, and SSH

Operation

- NX-OS device segregates different packets destined to the inband interface into different classes.
- Once these classes are identified, the NX-OS device polices or marks down packets, which ensure that the supervisor module is not overwhelmed.
- CoPP policer is attached to the interface “control-plane”

Implementation

CoPP Policing is implemented on each forwarding engine independently:

- the configured policer’s values apply on a per forwarding engine basis and the aggregate traffic prone to hit the CPU is the sum of the conformed/transmit traffic on all of the forwarding engines
- CoPP can be modified from the default VDC only.

Troubleshooting

CoPP — Tighten the grip on Received packets (OSPF example)

Problem

Flapping OSPF neighbors!!

- A faulty OSPF neighbor or an offending server is blasting the switch with Hello packets.
- Default CoPP is rate-limiting as designed, but that results on dropping legitimate neighbors packets as well.

```
N7K-1# show policy-map interface control-plane module 2 | egrep "service-
policy|critical|ospf|police cir 39600|malicious"
  service-policy  input: copp-system-policy
    class-map copp-system-class-critical (match-any)
      match access-grp name copp-system-acl-ospf
      match access-grp name copp-system-acl-ospf6
      police cir 39600 kbps , bc 250 ms

N7K-1# show class-map type control-plane copp-system-class-critical | egrep
class|ospf
    class-map type control-plane match-any copp-system-class-critical
      match access-grp name copp-system-acl-ospf
      match access-grp name copp-system-acl-ospf6

N7K-1# show ip access-lists copp-system-acl-ospf
IP access list copp-system-acl-ospf
        10 permit ospf any any
```

No “malicious” class to block malicious traffic

Troubleshooting

CoPP — Tighten the grip on Received packets (OSPF example) Cont.

Modify

copp-system-acl-ospf
to permit the neighbors only

```
N7K-1# show ip access-lists copp-system-acl-ospf
IP access list copp-system-acl-ospf
        10 permit ospf any any
        20 permit ip 40.9.0.0/16 224.0.0.5/32
        30 permit ip 40.9.0.0/16 224.0.0.6/32
```

Remove

Add neighbors

Create

copp-system- acl-malicious
access-list

```
N7K-1# show ip access-lists copp-system-acl-malicious
IP access list copp-system-acl-malicious
        10 permit ip any 224.0.0.0/24
```

Add

copp-system-class-malicious class, right before the last class default, with zero-rate policer to block all malicious traffic.

```
N7K-1# show policy-map interface control-plane module 2 | egrep
"service-policy|critical|ospf|police cir 39600|malicious|police cir 1 "
  service-policy  input: copp-system-policy
    class-map copp-system-class-critical (match-any)
      match access-grp name copp-system-acl-ospf
      match access-grp name copp-system-acl-ospf6
      police cir 39600 kbps , bc 250 ms
    class-map copp-system-class-malicious (match-any)
      match access-grp name copp-system-acl-malicious
      police cir 1 bps , bc 200 ms
```

Troubleshooting

CoPP — Tighten the grip on Received packets (OSPF example) Cont.

Verify

Check the CoPP policer for drops

- The new class-map shows high rate of dropped packets.
- Furthermore, the statistics results point to the module where the offending device is connected .

```
N7K-1# show policy-map interface control-plane module 2 class copp-system-class-
  malicious

control Plane
  service-policy  input: copp-system-policy
    class-map copp-system-class-malicious (match-any)
      match access-grp name copp-system-acl-malicious
      police cir 1 bps , bc 200 ms
      module 2 :
        conformed 0 bytes; action: drop
        violated 1799505072 bytes; action: drop

N7K-1# show policy-map interface control-plane module 1 class copp-system-class-
  malicious

control Plane
  service-policy  input: copp-system-policy

    class-map copp-system-class-malicious (match-any)
      match access-grp name copp-system-acl-malicious
      police cir 1 bps , bc 200 ms
      module 1 :
        conformed 0 bytes; action: drop
        violated 0 bytes; action: drop
```

Troubleshooting

Control Plan — Hardware Rate-limiters

Essentials

- Rate-limiters can prevent redirected packets for egress exceptions from overwhelming the supervisor module
- As with CoPP policers, modifying the default rates should be carefully planned before any configuration changes.

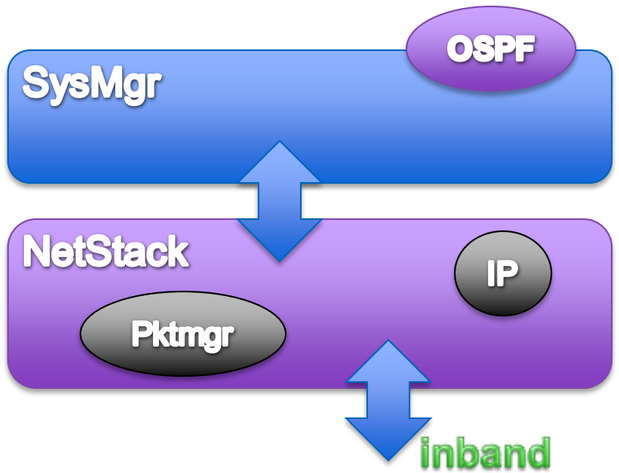
```
N7K-1# show hardware rate-limiter ?
  [snip]
  access-list-log  Packets copied to supervisor for access-list logging
  copy             Data and control packets copied to supervisor
  f1               Control packets from F1 modules to supervisor
  layer-2          Layer-2 control and Bridged packets
  layer-3          Layer-3 control and Routed packets
  module           Optionally specify a module number
  receive          Packets redirected to supervisor
  |                Pipe command output to filter

N7K-1# show hardware rate-limiter layer-2 mcast-snooping module 1
Units for Config: packets per second
Allowed, Dropped & Total: aggregated since last clear counters
Rate Limiter Class                          Parameters
------------------------------------------------------------
layer-2 mcast-snooping                      Config     : 1500
                                            Allowed    : 302128
                                            Dropped    : 0
                                            Total      : 302128
```

Troubleshooting

Control Plan — Verifying Software Services health (OSPF example)

Sysmgr

The System Manager handles processes and monitors their health. It keeps the mapping of PIDs to UUIDs.




```
N7K-1-PeerA# show system internal sysmgr service name ospf
Service "__inst_001__ospf" ("ospf", 14):
        UUID = 0x41000119, PID = 3725, SAP = 320
        State: SRV_STATE_HANDSHAKED (entered at time Wed Mar 14 15:47:34 2012).
        Restart count: 1
        Time of last restart: Wed Mar 14 15:47:33 2012.
        The service never crashed since the last reboot.
        Tag = 1
        Plugin ID: 1

N7K-1-PeerA# show system internal sysmgr service all | egrep -i netstack|name
Name                       UUID       PID     SAP   state  Start count      Tag  Plugin ID
netstack              0x00000221     5588     246   s0009            1      N/A           0
```

NOTE

Remember this:

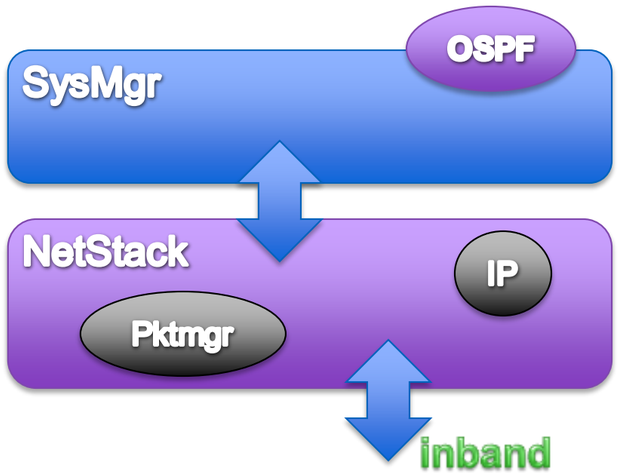
SAP = 320

Troubleshooting

Control Plan — Verifying Software Services health (OSPF example) Cont.

NetStack

Netstack is a full Network Stack designed with Modularity, High availability, and Virtualization implementation goals.




```
N7K-1-PeerA# show ip client ospf
Client: ospf-6467, uuid: 1090519321, pid: 3981, extended pid: 3981
 Protocol: 89, client-index: 19, routing VRF id: 65535
 Data MTS-SAP: 2339
 Data messages, send successful: 209867328, failed: 13263152

N7K-1-PeerA# show system internal pktmgr client 0x221
Client uuid: 545, 4 filters, pid 3841
 Filter 1: EthType 0x0800,
 Rx: 299923608, Drop: 0
 Filter 2: EthType 0x86dd,
 Rx: 1412579, Drop: 0
 [snip]
 Total Rx: 301346464, Drop: 0, Tx: 144295338, Drop: 0
 COS=0 Rx: 15993531, Tx: 87699456    COS=1 Rx: 1903980, Tx: 0
 COS=2 Rx: 0, Tx: 0    COS=3 Rx: 0, Tx: 0
 COS=4 Rx: 0, Tx: 0    COS=5 Rx: 3694169, Tx: 1
 COS=6 Rx: 56191519, Tx: 56595881    COS=7 Rx: 223563265, Tx: 0
```

Check for OSPF IP client failures

Check for L2 client packet drops

Troubleshooting

Control Plan — Verifying Software Services health (OSPF example) Cont.

MTS

- "Messages and Transactional Services". MTS offers SAPs (Service Access Points) to allow services to exchange messages
- MTS provides complete fault isolation by handling data structure communications.

```
N7K-1-PeerA# show system internal mts sup sap 320 stats
msg  tx: 3328
byte tx: 396657
msg  rx: 527
byte rx: 65045
opc sent to myself: 8927
max_q_size q_len limit (soft q limit): 1024
max_q_size q_bytes limit (soft q limit): 15%
max_q_size ever reached: 17
max_fast_q_size (hard q limit): 4096
rebind count: 0
Waiting for response: none
buf in transit: 0
bytes in transit: 0
```

Make sure the counters are incrementing (no memory leak)

```
N7k# show system internal mts buffers summary
node    sapno   recv_q  pers_q  npers_q log_q
sup     320     0       0       4592    0
```

npers high value indicates OSPF MTS buffer leak

Unicast L2 and L3 Forwarding, ARP

Control Plan — Golden rule

In case the issue you have encountered is urgent, complicated or you can't figure it out, collect **show tech-support** output asap!

Related show tech(s)

```
N7K-1-VDC2# show tech-support sysmgr
N7K-1-VDC2# show tech-support netstack detail
N7K-1-VDC2# show tech-support pktmgr
N7K-1-VDC2# show tech-support <service>
```

Thank you.