



i i i i i i i i

You make **possible**

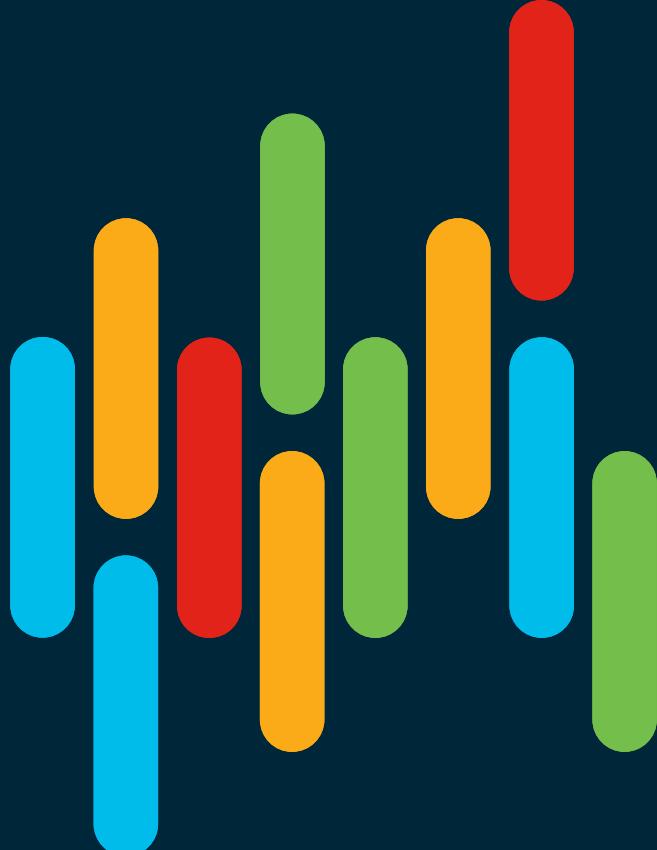


Advanced ASR 9000 Operation and Troubleshooting

Everything You Wanted To Know And More

Xander Thuijs, Distinguished Engineer
Aleksandar Vidakovic, Technical Leader

TECSPG-3001



cisco Live!

Barcelona | January 27-31, 2020

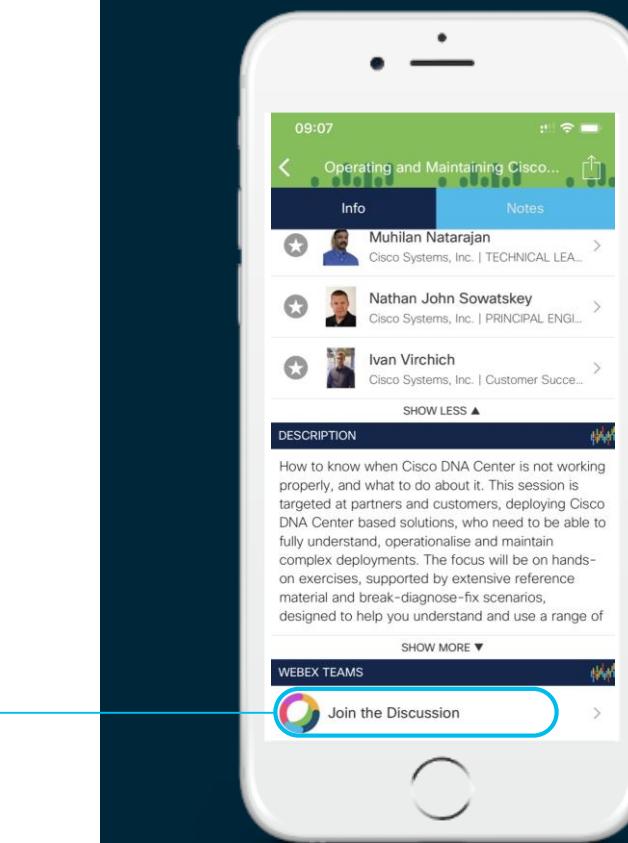
Cisco Webex Teams

Questions?

Use Cisco Webex Teams to chat with the speaker after the session

How

- 1 Find this session in the Cisco Events Mobile App
- 2 Click “Join the Discussion”
- 3 Install Webex Teams or go directly to the team space
- 4 Enter messages/questions in the team space

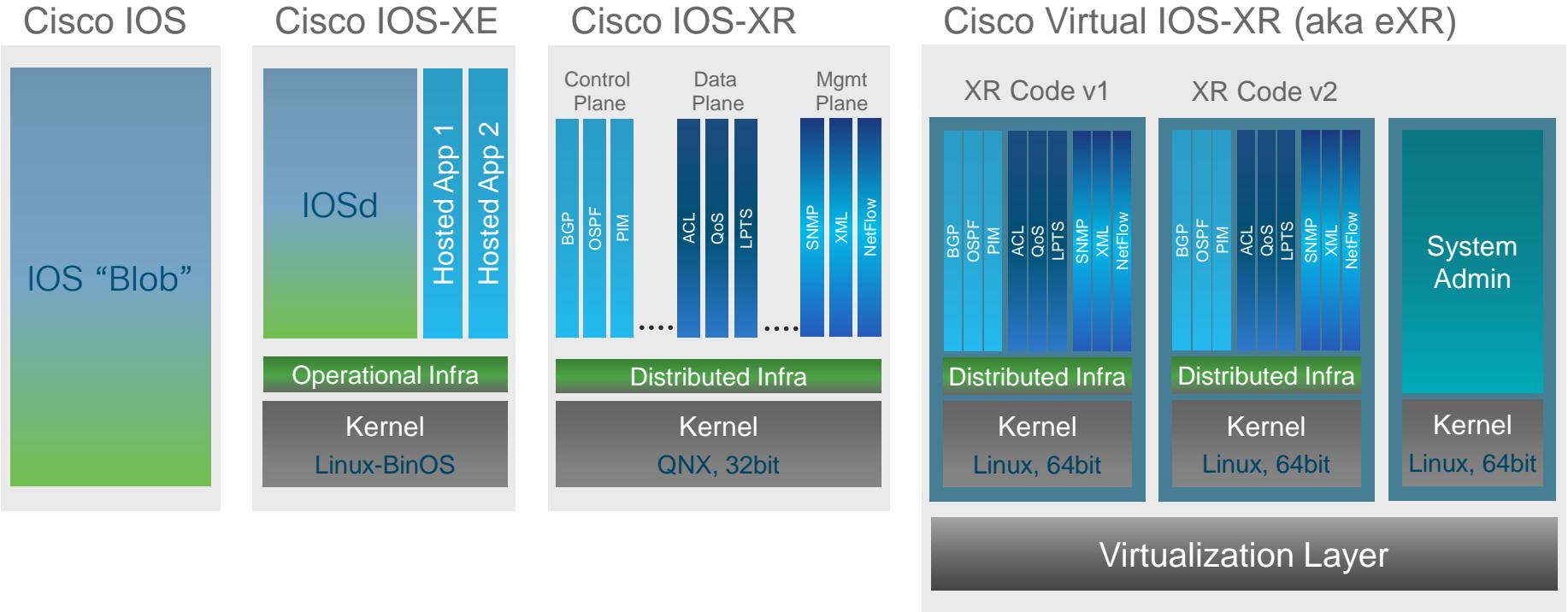


Agenda

- Introduction
- eXR Architecture (a)
- LC Architecture review (ASR9k vs NCS5500) (x)
- eXR install (CSM, ISSU, ISSU SMU) (x)
- Punt/Inject path (a)
- Packet tracer (a)
- Monitoring the system (Automation, PAM, MDT) (x)
- Usability (BGP GM,) (x)
- Load-balancing (a)

Enhanced (64-bit) IOS XR Architecture

Cisco IOS Evolution



Software forwarding on classic IOS

Bit of History on CEF and Fast switching

4. OSPF doesn't return in time?
5. HW timer fires
Calls function to spit CPU HOG
Hw timer is defined with "scheduler max time"

Software forwarding

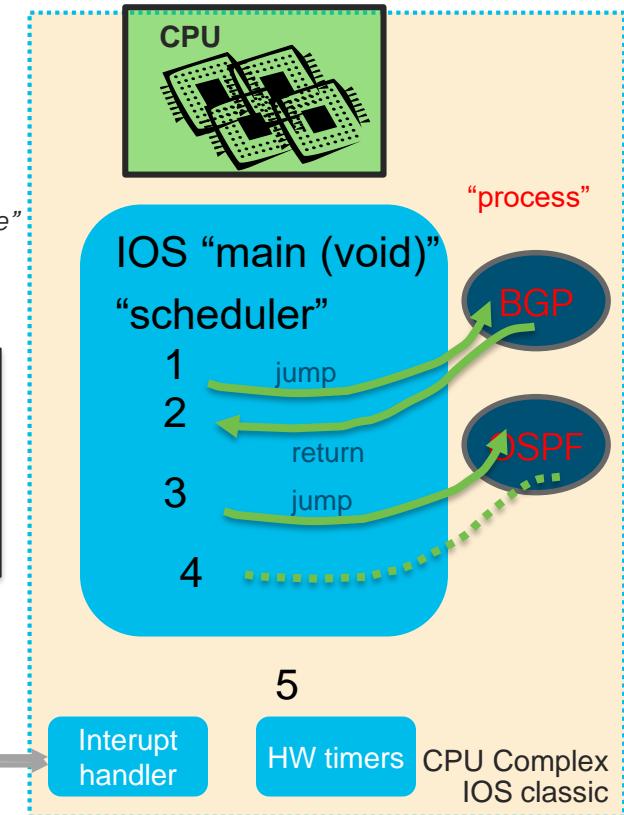
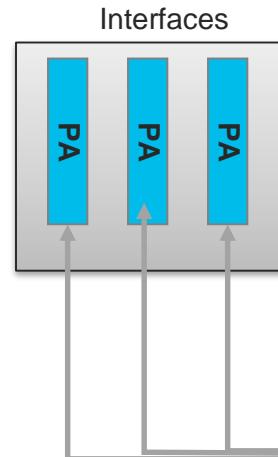
- Calls a receive interrupt (RX interrupt) to the CPU to handle the packet
- Time in the ISR (interrupt service routine) is limited.
- We need to turn around this packet quick during the RX interrupt.
- How? With all features and info? CEF! Prebuilt L2 headers, routing, features all in one entry!

CPU utilization for five seconds: 68% / 30%

28% used
Jump procs

Total CPU time used

Time spent in ISR
(fast switching)

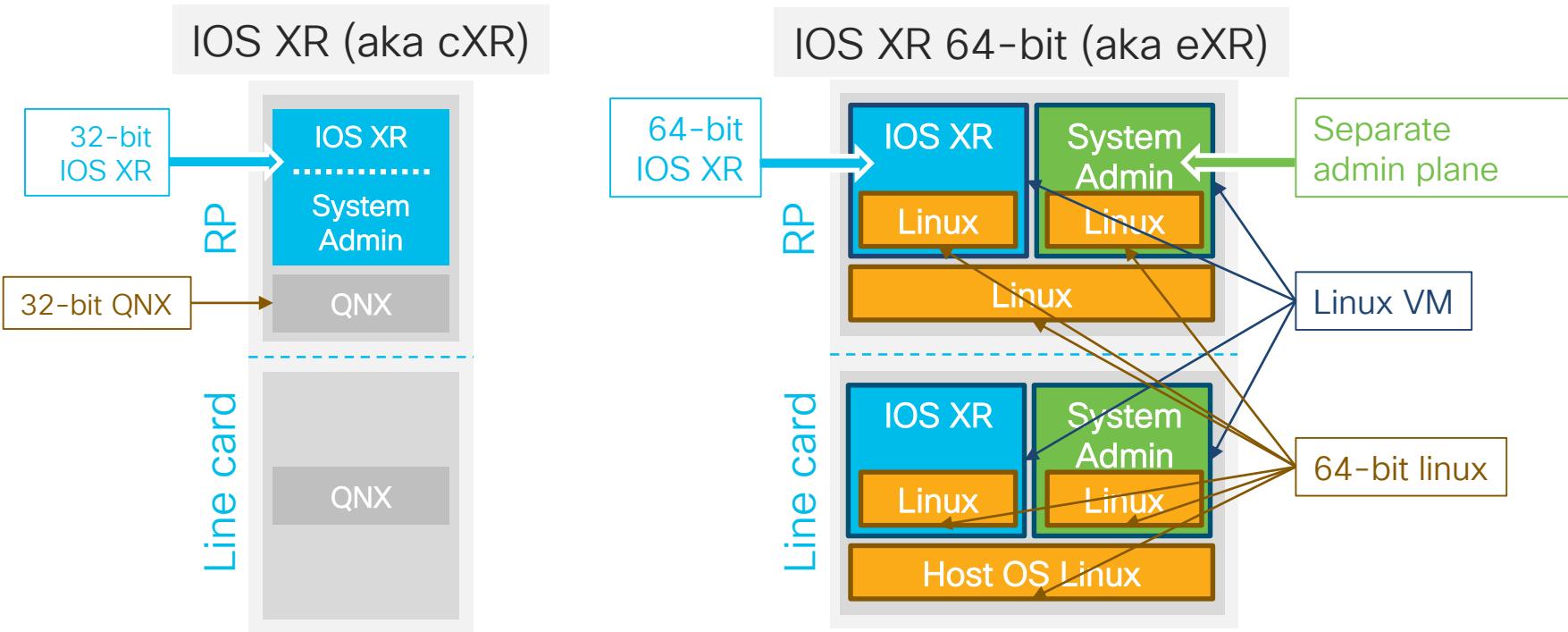


Packets not switched in ISR
Are handed over to IP Input (proc switching)

Why eXR?

- 64 Bit OS
- Route and feature scale
 - Max memory visible to a process in 32-bit OS (Text + Heap + Shared mem): 4GB
- Separation of Admin and Control plane
- In Service Software Upgrade (ISSU) support
- RPM based packaging
- 4th generation ASR 9000 line card and commons support
- Application hosting
- Higher scale/performance Telemetry (gRPC as transport)
- VM reload SMU cause less downtime than complete HW reload.

IOS XR 64 bit: Architecture Overview



IOS XR 64 bit: Architecture Overview

Cisco developed packages for core network functions (BGP, MPLS, etc.)

Yocto packages for standard Linux tools and libraries (bash, python, tcpdump, etc.).



XR VM
(Control Plane)

Admin VM
(Admin Plane)

Runs processes responsible to perform system diags, monitor env. variables, and manage hardware components

First VM to be booted by the Hypervisor, responsible for the start and maintenance of the Control Plane VM

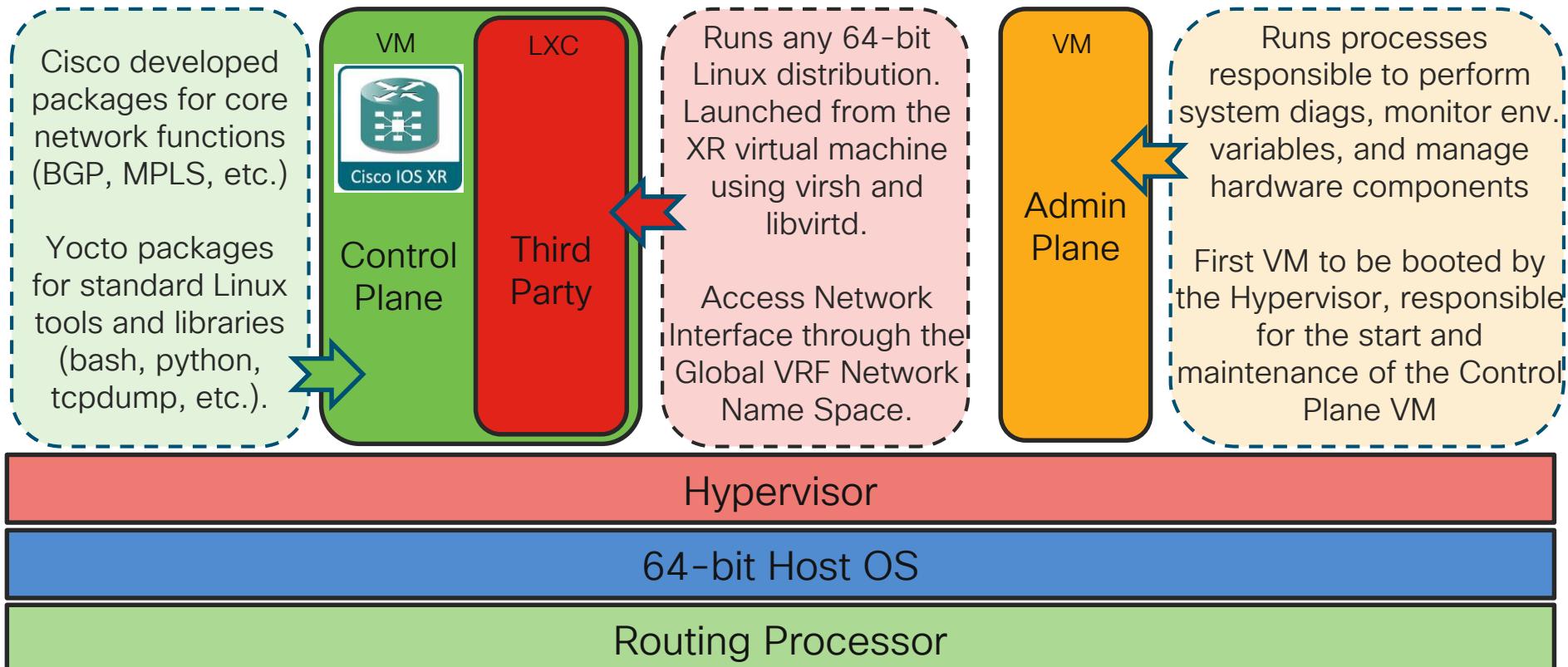


Hypervisor

64-bit Host OS

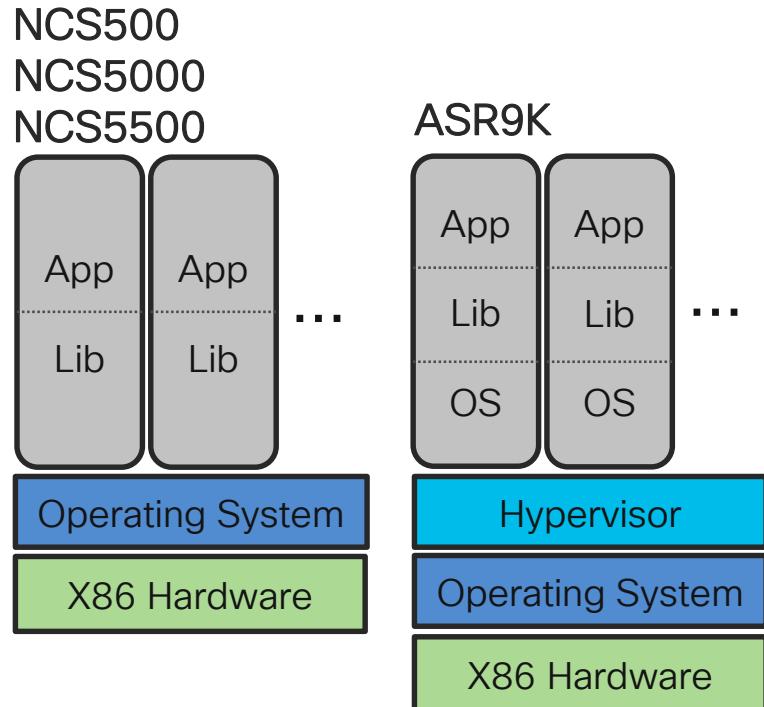
Routing Processor

IOS XR 64 bit: Architecture Overview



ASR9000 eXR vs NCS500/NCS5000/NCS5500

- With eXR, the Admin plane and the Control Plane run in isolated environment.
 - ASR9K uses Virtual Machines (VMs)
 - NCS5000 and NCS5500 use containers (LXCs)
- VMs and LXCcs Provide the same functionalities.
- LXCcs are lightweight no ISSU support.
- VMs are heavier but provide ISSU support.

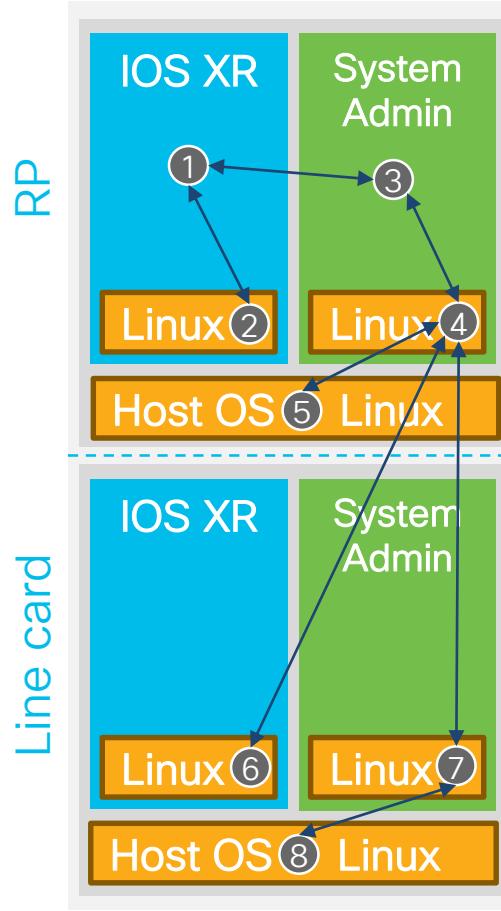


ASR9000 and eXR (in General)

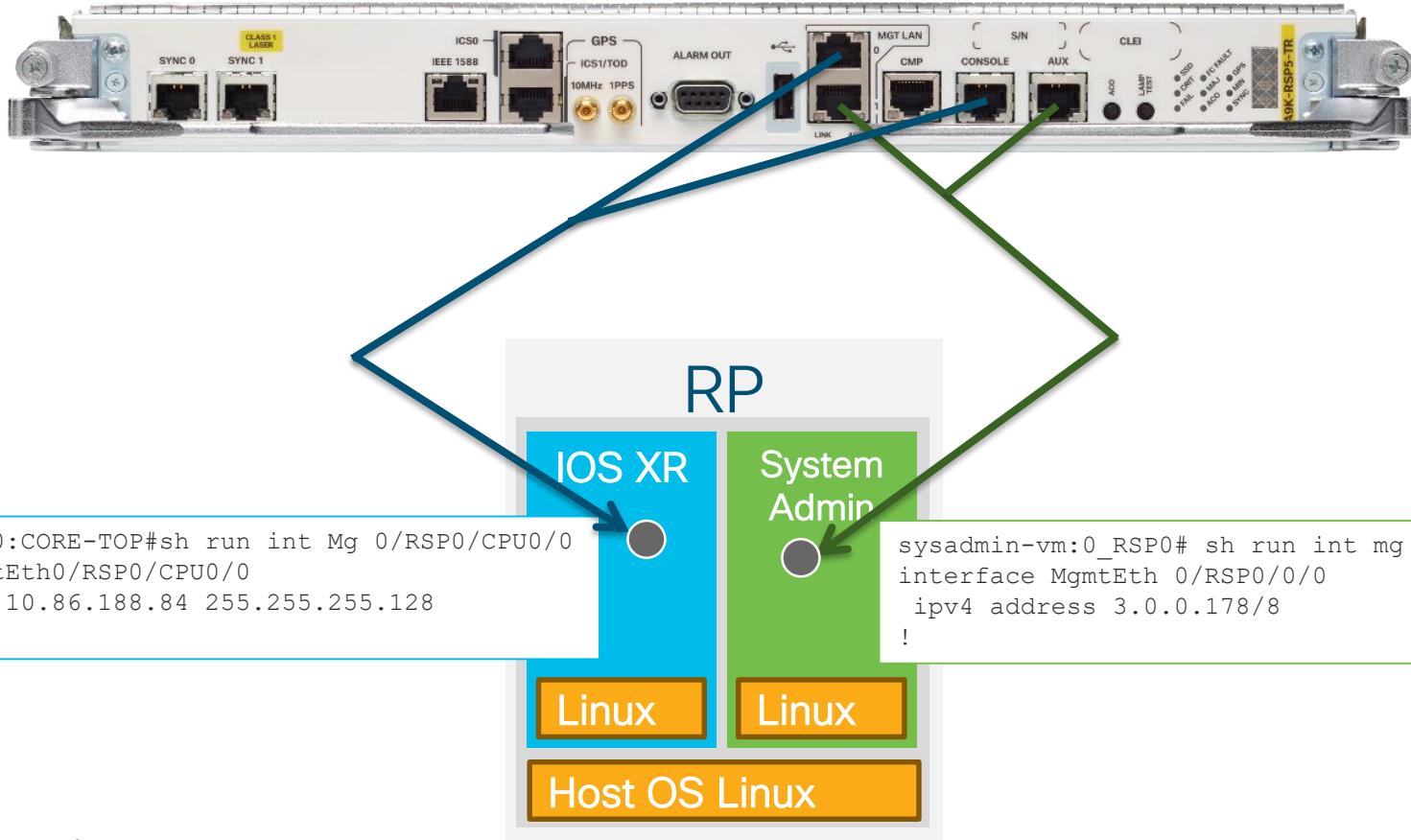
- eXR requires an x86 architecture
- Only tomahawk and lightspeed cards can run eXR.
- Typhoon uses a 32 bit processor on the LC
 - Attempts to do conversion and a formal VM layer for PPC processor did not succeed
- RSP440 could technically run eXR, but TR cards do not have enough memory (6G)
- ISSU requirements double memory and resource utilization!
- In eXR all existing sw is compiled for 64bit instead of 32bit. (relation to bugs pertaining to either)
- Redundancy:
 - Software Arbitration (vs HW arbitration)
 - Admin VM loads first ☐ XR VM state follows Admin VM

CLI Access

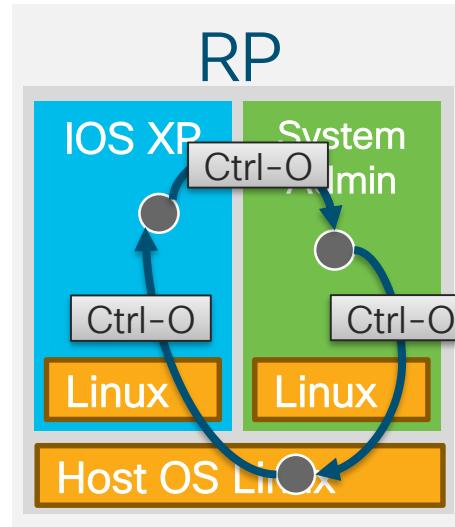
- All configuration & monitoring from XR VM
- Admin mode actions from 32-bit XR are in XR VM in 64-bit XR
 - E.g.: install
- Use scheme for advanced debugging



MgmtEth, Console, Aux Connections To VMs



Toggle Between 3 VMs on Console



CLI Access

RP XR VM, RP Admin VM, RP Host

```
RP/0/RSP0/CPU0:CORE-TOP#run
```

1

```
[xr-vm_node0_RSP0_CPU0:~]$
```

2

```
[xr-vm_node0_RSP0_CPU0:~]$exit
```

```
RP/0/RSP0/CPU0:CORE-TOP#
```

1

```
RP/0/RSP0/CPU0:CORE-TOP#admin
```

1

```
root connected from 192.0.16.4 using ssh on sysadmin-vm:0_RSP0
```

3

```
sysadmin-vm:0_RSP0# run
```

3

```
[sysadmin-vm:0_RSP0:~]$
```

4

```
[sysadmin-vm:0_RSP0:~]$chvrf 0 bash
```

4

```
[sysadmin-vm:0_RSP0:~]$ssh my_host
```

5

```
[host:~]$
```

Note

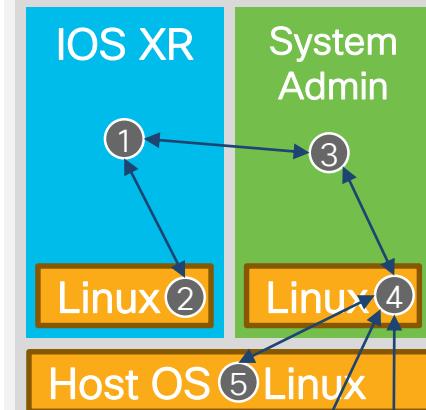
Exit from any prompt:
a) hit 'Ctrl-d'
b) type 'exit'

Note

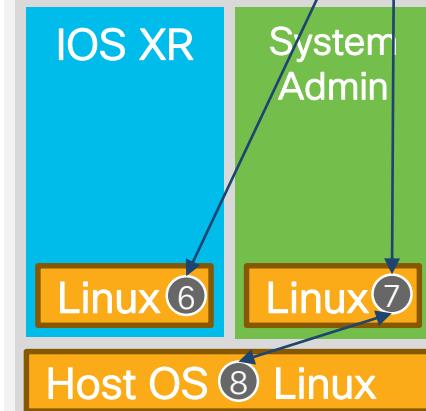
"chvrf 0 bash" enables ssh to hostnames:

- ssh my_host
- ssh lc<n>_xr
- ssh lc<n>_admin

RP



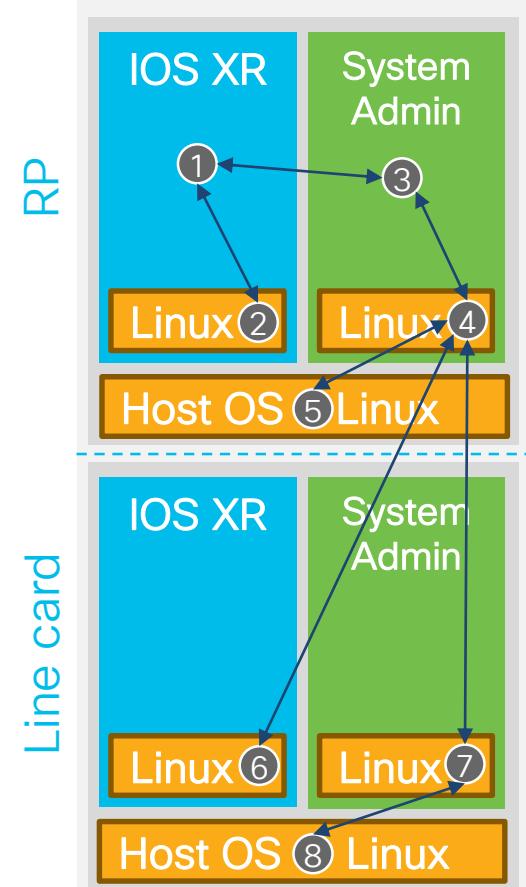
Line card



CLI Access

LC XR VM, LC Admin VM, LC Host

```
RP/0/RSP0/CPU0:CORE-TOP#  
RP/0/RSP0/CPU0:CORE-TOP#admin  
aleks connected from 192.0.16.4 using ssh on sysadmin-vm:0_RSP0  
sysadmin-vm:0_RSP0# run  
[sysadmin-vm:0_RSP0:~]$chvrf 0 bash  
[sysadmin-vm:0_RSP0:~]$ssh lc5_xr  
Last login: Sun Nov 10 17:02:15 2019 from 192.0.16.1  
[xr-vm_node0_5_CPU0  
[xr-vm_node0_5_CPU0:~]$exit  
Connection to lc5_xr closed.  
[sysadmin-vm:0_RSP0:~]$ssh lc5_admin  
Last login: Sun Nov 10 17:02:25 2019 from 192.0.16.1  
[sysadmin-vm:0_5:~]$  
[sysadmin-vm:0_5:~]$ ssh my_host  
Last login: Sun Nov 10 17:02:32 2019 from 10.0.2.15  
[host:~]$
```



CLI Access Example

How To Check Physical Memory On RP/LC

- On 32-bit IOS XR the total physical RAM is available to the CPU
- On 64-bit IOS XR the total physical RAM must be distributed between 4 entities:
 - host OS
 - Admin VM
 - XR VM
 - XR VM v2 (LC only; required during ISSU)
- command “`show memory summary [location <location>]`”:
 - 32-bit XR: shows the total physical RAM is available to the CPU
 - 64-bit XR: shows the total physical RAM available to XR VM on the CPU

CLI Access Example

RAM available to IOS XR on 32-bit vs 64-bit IOS XR

32-bit

```
RP/0/RSP1/CPU0:ariad#sh platform | i 8x100GE
Node          Type           State      Config State
0/3/CPU0     A9K-8X100GE-TR   IOS XR RUN PWR,NSHUT,MON
RP/0/RSP1/CPU0:ariad#sh memory summary location 0/3/CPU0
node:        node0_3_CPU0
```

```
Physical Memory: 24576M total
Application Memory : 24253M (18554M available)
Image: 82M (bootram: 82M)
Reserved: 224M, IOMem: 0, flashfsys: 0
Total shared window: 431M
```

64-bit

```
RP/0/RSP0/CPU0:CORE-TOP#sh platform | i 8x100GE
0/5/CPU0     A9K-8X100GE-SE   IOS XR RUN NSHUT
RP/0/RSP0/CPU0:CORE-TOP#sh memory summ loc 0/5/cpu0
node:        node0_5_CPU0
```

```
Physical Memory: 10691M total (4939M available)
Application Memory : 10691M (4939M available)
Image: 4M (bootram: 0M)
Reserved: 0M, IOMem: 0M, flashfsys: 0M
Total shared window: 311M
```

Both LCs are 8x100G Tomahawk LC

- TR/SE have the same size RAM available to LC CPU
- TR/SE differ in RAM available to NP

Different RAM size available to IOS XR

CLI Access Example

Physical Memory On LC (64-bit XR)

```
RP/0/RSP0/CPU0:CORE-TOP#admin
Sun Nov 10 17:46:22.305 UTC

root connected from 192.0.16.4 using ssh on sysadmin-vm:0_RSP0
sysadmin-vm:0_RSP0# run
Sun Nov 10 17:46:26.548 UTC+00:00

[sysadmin-vm:0_RSP0:~]$chvrf 0 bash
[sysadmin-vm:0_RSP0:~]$ssh lc5_admin
Last login: Sun Nov 10 17:03:14 2019 from 192.0.16.1
[sysadmin-vm:0_5:~]$ ssh my_host
Last login: Sun Nov 10 17:03:30 2019 from 10.0.2.15
[host:~]$ grep MemTotal /proc/meminfo
MemTotal: 24490700 kB
[host:~]$
```

XR VM vs Sysadmin VM

When do I need to go into Sysadmin VM?

- All Sysadmin VM commands can be executed from XR VM
 - Expect delay in output compared to running the command directly from Sysadmin VM mode
 - ➔ for interactive use, better go into Sysadmin VM
- Sysadmin syslog messages are imported into XR VM
- Sysadmin VM holds all chassis related info:
 - VMs status
 - Environment
 - FPD

XR VM vs Sysadmin VM CLI Example

Response delay between execution from XR VM vs execution in Admin VM

```
RP/0/RSP0/CPU0:CORE-TOP#admin sh vm ?  
Mon Nov 11 18:36:06.954 UTC
```

Possible completions:
location
| Output modifiers
<cr>

```
RP/0/RSP0/CPU0:CORE-TOP#admin sh vm  
Mon Nov 11 18:36:10.791 UTC
```

Location: 0/5			
Id	Status	IP Address	HB Sent/Recv
sysadmin	running	192.0.28.1	NA/NA
default-sdr	running	192.0.28.3	9705/9705

Location: 0/RSP0			
Id	Status	IP Address	HB Sent/Recv
sysadmin	running	192.0.16.1	NA/NA
default-sdr	running	192.0.16.4	194365/194365

```
RP/0/RSP0/CPU0:CORE-TOP#  
RP/0/RSP0/CPU0:CORE-TOP#
```

```
sysadmin-vm:0_RSP0# sh clock ; sh vm ?  
Possible completions:
```

location
| Output modifiers
<cr>

```
sysadmin-vm:0_RSP0# sh clock ; sh vm  
Mon Nov 11 18:42:34.288 UTC+00:00  
Mon Nov 11 18:42:34 UTC 2019
```

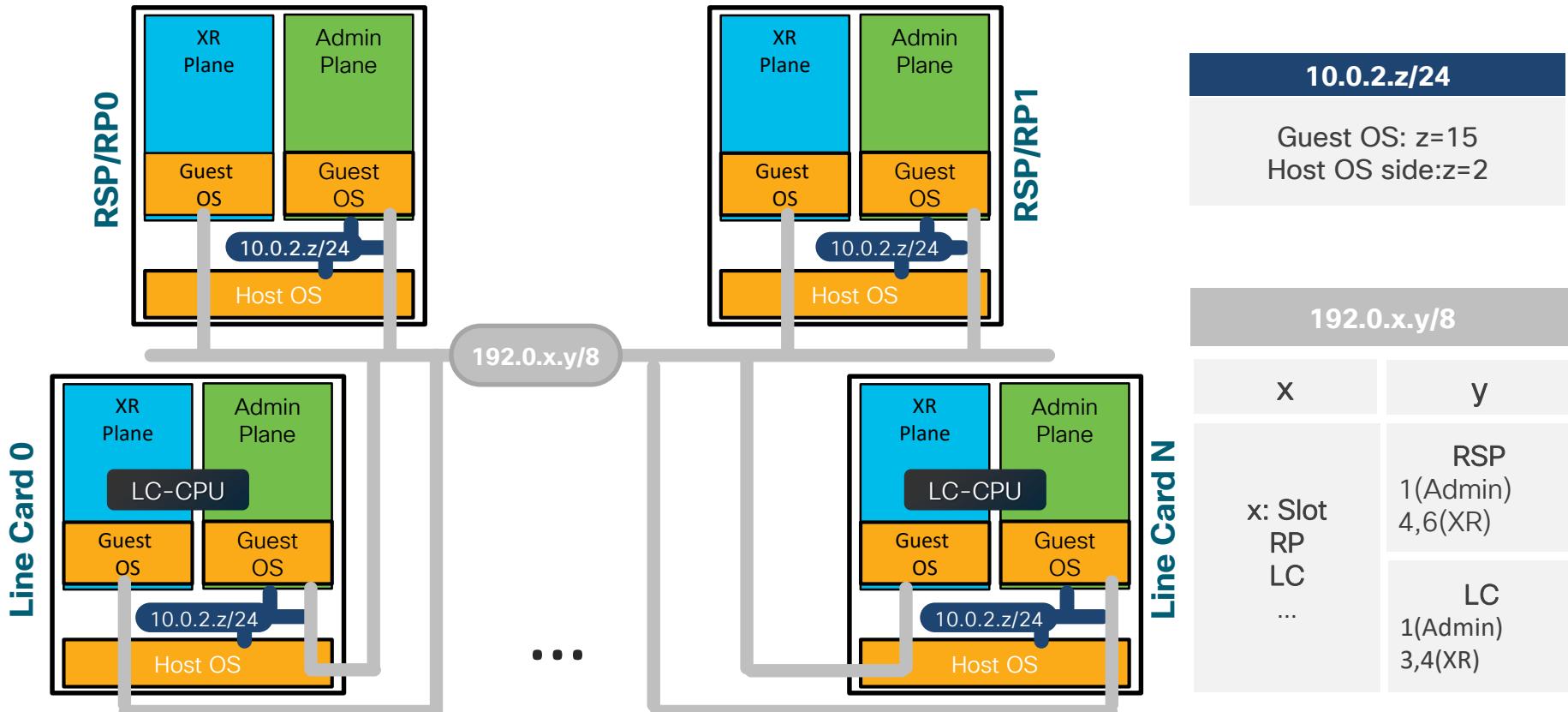
Location: 0/5			
Id	Status	IP Address	HB Sent/Recv
sysadmin	running	192.0.28.1	NA/NA
default-sdr	running	192.0.28.3	9743/9743

Location: 0/RSP0			
Id	Status	IP Address	HB Sent/Recv
sysadmin	running	192.0.16.1	NA/NA
default-sdr	running	192.0.16.4	195126/195126

```
sysadmin-vm:0_RSP0#
```

➔ Recommendation: In interactive session go to admin mode when planning on executing more than one admin command.

VM Internal Connection



CISCO Live!

XR VM Media Mapping

show media

Admin access point	Shell mount point	Size (RSP880)	Device Type	Comment
harddisk:	/misc/disk1/	5.5G	SSD	Used for dumping core/showtech/temp files
log: (read only)	/var/log/	469MB	SSD	System logs
disk0:	/misc/scratch/	968 MB	SSD	NP, PCIE, temp path for core files, TPA log, ZTP logs & other log files
apphost:	/misc/app_host	2.4GB	SSD	3rd party applications
rootfs: (read only)	/	3.8GB	SSD	This is the root of XR VM file system.
config: (read only)	/misc/config	469MB	SSD	Configuration repository
harddiska:	/eusb/	3.7 GB	eUSB	Spare space for to store user files. Device visible in XR plane only.

Sysadmin VM Media Mapping

admin show media

Admin access points	Shell mount points	Size (RSP880)	Device Type	Comment
harddisk:	/misc/disk1/	7.6 GB	SSD	Used for dumping core/showtech/temp files.
log: (read only)	/var/log/	469MB	SSD	System logs
disk0:	/misc/scratch/	968 MB	SSD	PCIE, temp path for core files & other log files
Install (no access)	/install_repo/	4.8 GB	SSD	Install repository
rootfs: (read only)	/	2.4 GB	SSD	This is the root of admin VM file system.
config: (read only)	/misc/config	469MB	SSD	Configuration repository
a9ksys: (read only)	/eusba:/	750 MB	eUSB	Used for storing OBFL files for RSP/RSP/FC. Device visible in admin plane only.
harddiskb:	/eusbb/	3GB	eUSB	Used for storing migration image during cXR to eXR or eXR to cXR migration. Post migration serves as disaster recovery partition. Device visible in admin plane only.

Copying Files Between VMs

The diagram illustrates the structure of the 'copy' command. It consists of two main parts: 'source' and 'destination'. The 'source' part is indicated by a blue bracket above the first half of the command, which includes 'copy <fs>:[<path>/]<file> location <location>'. The 'destination' part is indicated by a green bracket above the second half of the command, which includes '<fs>:<path>[/<file>] location <location>'.

```
copy <fs>:[<path>/]<file> location <location> <fs>:<path>[/<file>] location <location>
```

Example: Copy from Admin VM to XR VM

```
sysadmin-vm:0_RSP0# copy harddisk:dummy.txt location 0/RSP0 harddisk: location 0/RSP0/CPU0/VM1
Copying harddisk:dummy.txt to harddisk:
File copied successfully
sysadmin-vm:0_RSP0# exit
RP/0/RSP0/CPU0:CORE-TOP#dir harddisk:dummy.txt

Directory of harddisk:dummy.txt
44 -rw-r--r--. 1 15 Dec 23 11:49 dummy.txt

5678464 kbytes total (5343756 kbytes free)
RP/0/RSP0/CPU0:CORE-TOP#
```

Platform and VM status

XR VM Mode

```
RP/0/RSP0/CPU0:CORE-TOP#sh platform
Mon Nov 11 19:20:12.293 UTC
Node          Type           State        Config state
-----
0/RSP0/CPU0   A9K-RSP880-SE(Active)  IOS XR RUN    NSHUT
0/FT0         ASR-9010-FAN        OPERATIONAL  NSHUT
0/FT1         ASR-9010-FAN        OPERATIONAL  NSHUT
0/5/CPU0     A9K-8X100GE-SE      IOS XR RUN    NSHUT
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#sh platform vm
Mon Nov 11 19:20:17.839 UTC
Node name    Node type       Partner name  SW status   IP address
-----
0/5/CPU0     LC (ACTIVE)     NONE          FINAL Band  192.0.28.3
0/RSP0/CPU0   RP (ACTIVE)     NONE          FINAL Band  192.0.16.4
RP/0/RSP0/CPU0:CORE-TOP#
```

Platform and VM status

Sysadmin VM Mode

```
RP/0/RSP0/CPU0:CORE-TOP#admin show platform
Mon Nov 11 19:23:46.584 UTC
Location Card Type          HW State    SW State    Config State
-----
0/5      A9K-8X100GE-SE    OPERATIONAL  OPERATIONAL  NSHUT
0/RSP0   A9K-RSP880-SE    OPERATIONAL  OPERATIONAL  NSHUT
0/FT0    ASR-9010-FAN     OPERATIONAL  N/A          NSHUT
0/FT1    ASR-9010-FAN     OPERATIONAL  N/A          NSHUT

RP/0/RSP0/CPU0:CORE-TOP#admin sh vm
Mon Nov 11 19:23:58.151 UTC

Location: 0/5
Id          Status       IP Address   HB Sent/Recv
-----
sysadmin    running      192.0.28.1  NA/NA
default-sdr running      192.0.28.3  9992/9992

Location: 0/RSP0
Id          Status       IP Address   HB Sent/Recv
-----
sysadmin    running      192.0.16.1  NA/NA
default-sdr running      192.0.16.4  200100/200100
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#admin sh sdr
Mon Nov 11 19:39:18.526 UTC

SDR: default-sdr
Location  IP Address   Status      Boot Count  Time Started
-----
0/RSP0/VM1 192.0.16.4  RUNNING    1           11/10/2019 15:32:24
0/5/VM1    192.0.28.3  RUNNING    1           11/10/2019 15:34:44
RP/0/RSP0/CPU0:CORE-TOP#
```

Managing Node/VM Reload

Same effect because ASR9k is single rack

Operation	Effect	Sample Command
Power-cycle	Power-cycle chassis	<code>sysadmin-vm:0_RP0# hw-module location all reload</code>
	Power-cycle rack	<code>sysadmin-vm:0_RP0# reload rack 0</code>
	Power-cycle a card	<code>sysadmin-vm:0_RP0# hw-module location x/y reload</code>
Reload Admin VM	Reload RP Admin VM	<code>sysadmin-vm:0_RSP0# reload location 0/RSP0</code>
	Reload RP Admin VM	<code>sysadmin-vm:0_RSP0# reload location 0/5</code>
Reload XR VM	Reload all XR VM	<code>RP/0/RSP0/CPU0:CORE-TOP#reload location all</code>
	Reload LC XR VM	<code>RP/0/RSP0/CPU0:CORE-TOP#reload location 0/5/CPU0</code>
	Reload RM XR VM	<code>RP/0/RSP0/CPU0:CORE-TOP#reload location 0/RSP0/CPU0</code>
	RP switchover	<code>RP/0/RSP0/CPU0:CORE-TOP#redundancy switchover</code>

Managing Node/VM Shutdown

Operation	Effect	Sample Command
Node shutdown	Shutdown until next reload/power-cycle	<pre>sysadmin-vm:0_RP0# hw-module location all shutdown sysadmin-vm:0_RP0# hw-module location 0/RP0 shutdown sysadmin-vm:0_RP0# hw-module location 0/FC0 shutdown sysadmin-vm:0_RP0# hw-module location 0/5 shutdown</pre>
	Recover from shutdown	Power-cycle or reload node
	Permanent shutdown	<pre>sysadmin-vm:0_RSP0(config)# hw-module shutdown location 0/RP0 sysadmin-vm:0_RSP0(config)# hw-module shutdown location 0/FC0 sysadmin-vm:0_RSP0(config)# hw-module shutdown location 0/5</pre>
	Disable permanent shutdown	<pre>sysadmin-vm:0_RSP0(config)# no hw-module shutdown location 0/RP0 sysadmin-vm:0_RSP0(config)# no hw-module shutdown location 0/FC0 sysadmin-vm:0_RSP0(config)# no hw-module shutdown location 0/5</pre>

Linecard Architectures

ASR9000 NCS5500

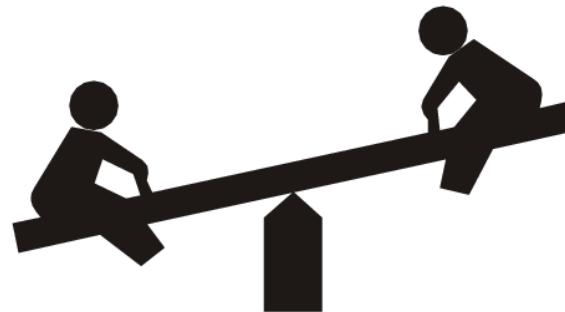
Forwarding Processor Terminology

“Custom”

Cisco-developed component

“Run to Completion”

Each core performs all operations for a packet



“Fabric Enabled”

Scalable via mesh or fabric

Still may be part of a “fixed” chassis

“Merchant”

Open market component

“Pipeline”

Each stage performs a single operation for every packet

“System on a Chip”

Single chip per router

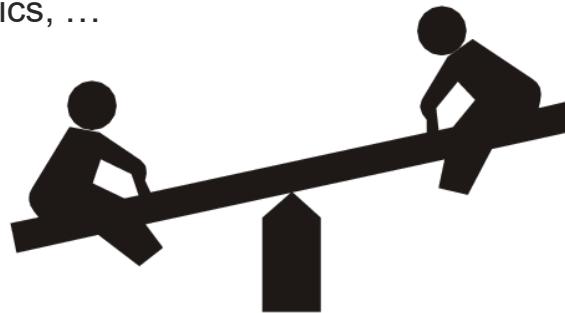
Forwarding Processor Characteristics

“Custom”

Up front Investment by Cisco, unit cost often lower
More aggressive technology
May apply to chips, memories, optics, ...

“Run to Completion”

High flexibility (e.g., extra stat updates)
10-15% ASIC gate penalty



“Fabric Enabled”

Enables scalable systems

“Merchant”

Vendor bears development cost
Available to entire industry

“Pipeline”

Less flexible
Wide range of pipeline flexibility

“System on a Chip”

Highest pps and bandwidth
Usually no off-chip resources
May scale via Ethernet fabric

Forwarding Components Examples

“Custom”

NCS 6000 nPower X1, X2

CPAK optics – 18 month ahead of CFP2

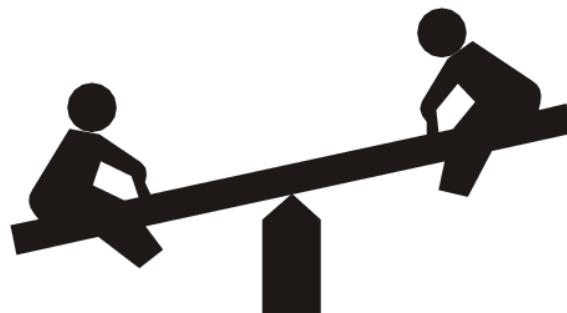
Atris / Serial Atris fast memory

ASR 9000 switch fabric

All Chassis / PCBs

“Run to Completion”

CRS, NCS 6000



“Fabric Enabled”

ASR 9000, NCS 5502/5508, NCS 6000, CRS

“Merchant”

Broadcom XGS (Trident/Tomahawk)

Broadcom DNX (Arad/Jericho)

NCS 6000 Fabric

EZ Chip (ASR 9000)

Cavium Octeon (CRS CGSE)

“Pipeline”

ASR 9000, NCS 5000, NCS 5500

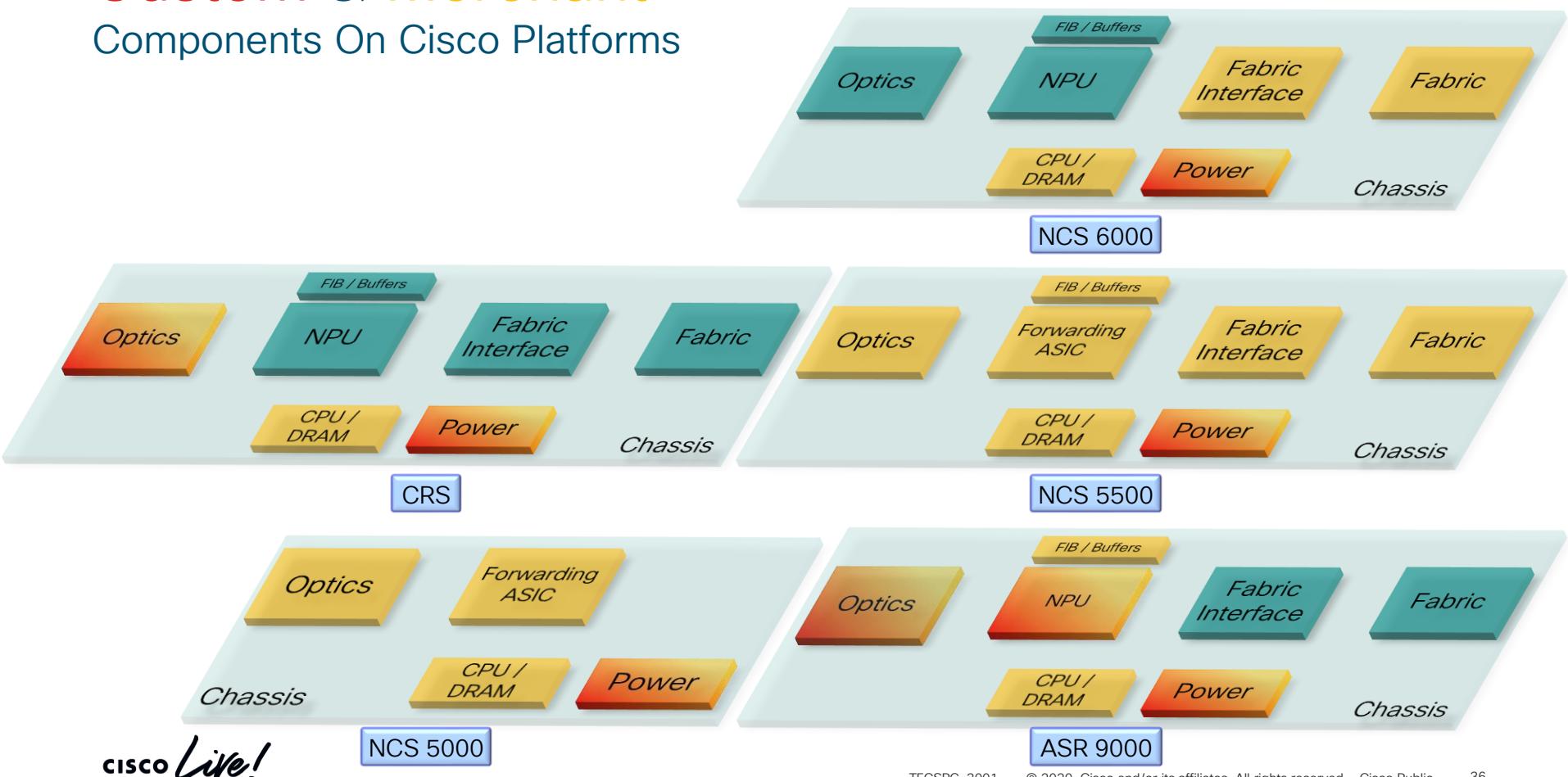
“System on a Chip”

ASR 9001

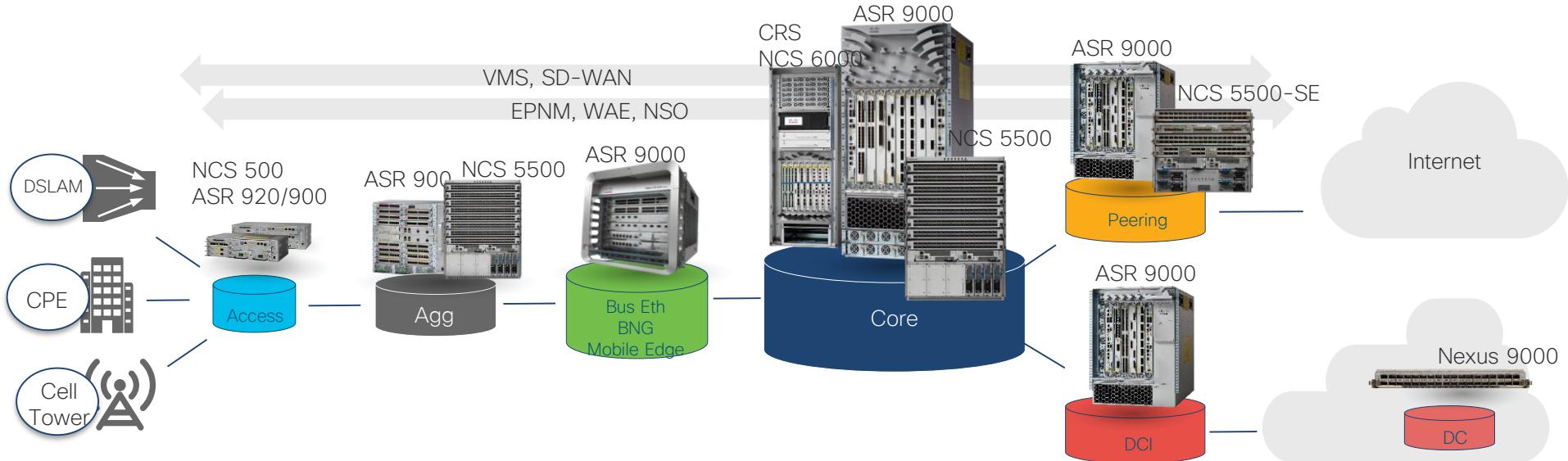
NCS 5000

NCS 5501

Custom & Merchant Components On Cisco Platforms



Service Provider Platforms per place in network



Capex



Density, Low Cost/Bit,
PayG, OpenPay

Opex



Low Watt/Bit, Automation,
ASR 9K - Swiss Army Knife

Security



Trustworthy systems in XR,
MacSec, Anti-DDoS

Converged



All in one - Multiple
services at scale

Time to market



Simplified Control & Transport
Plane with SR/EVPN

Programmable



On demand network SLA
with SR-TE and Open APIs

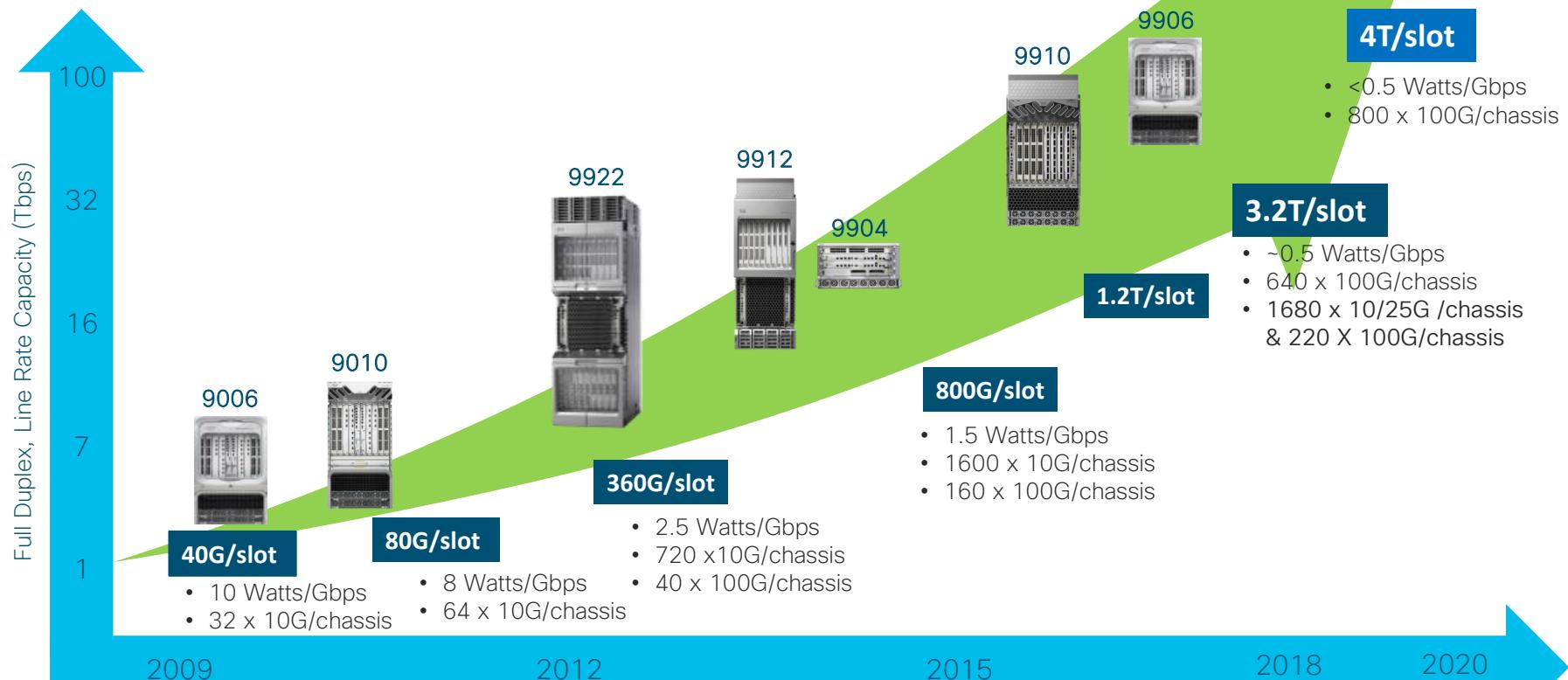
ASR9000

Cisco ASR 9000 portfolio

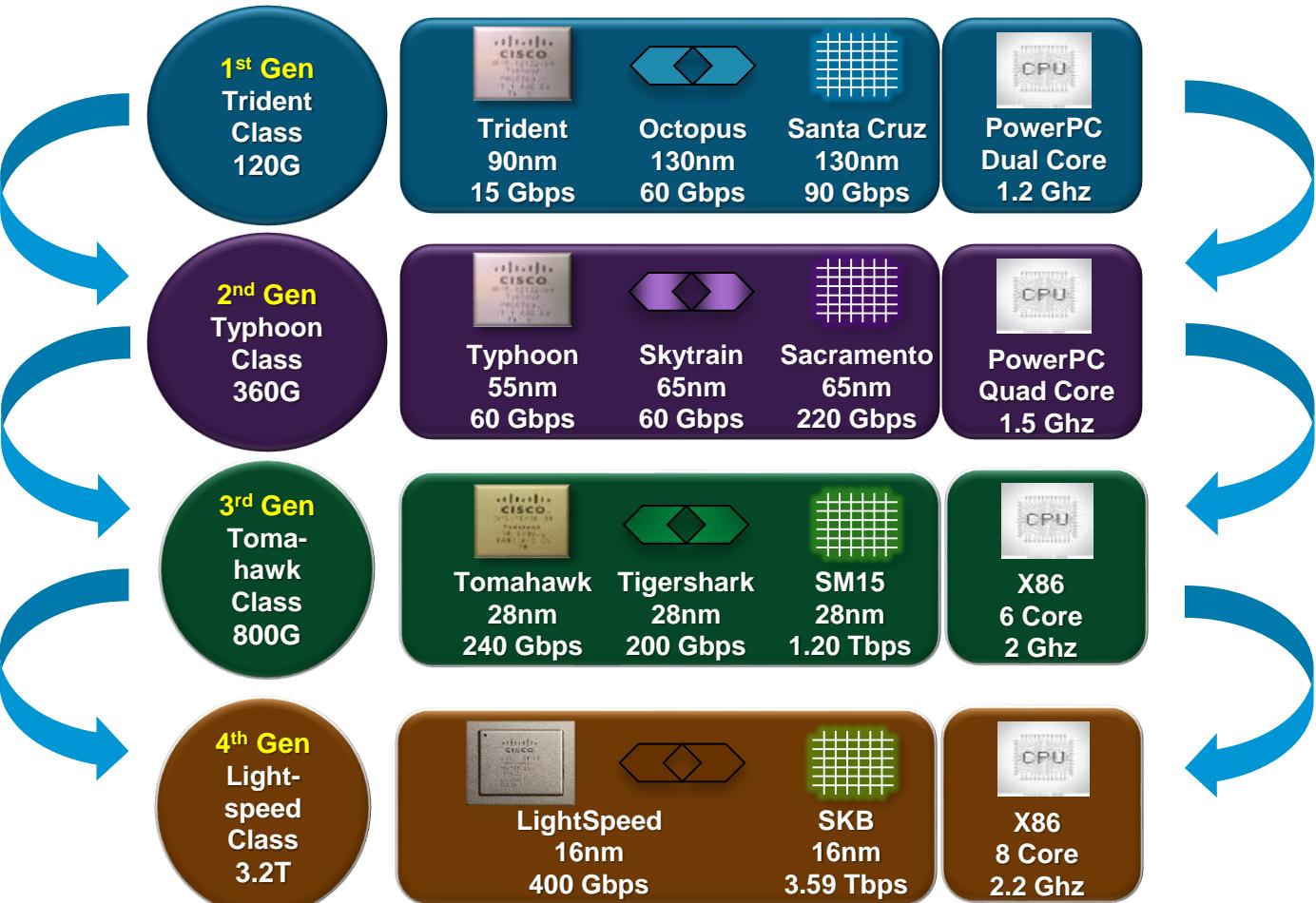
Compact & Powerful Access/Aggregation	Flexible Service Edge	High Density Service Edge and Core
<ul style="list-style-type: none"> Small footprint with full IOS-XR for distributed environments 	<ul style="list-style-type: none"> Optimized for ESE and MSE with high M-D scale for medium to large sites 	<ul style="list-style-type: none"> Scalable, ultra high density service routers for large, high-growth sites
One Platform, One OS, One Family		
 nV Satellite NCS5000		 ASR 9922
 ASR 9901 1HCY18	 ASR 9006 2HCY17	 ASR 9912
 ASR 9001	 ASR 9004	 ASR 9910
<u>Fixed 2RU</u>	<u>4 LC/10RU</u>	<u>8 LC/21RU</u>
240 Gbps	7 Tbps	14 Tbps
	32 Tbps	64 Tbps
<u>2 LC/6RU</u>	<u>4 LC/14RU</u>	<u>8 LC/21RU</u>
16 Tbps		
MSE	E-MSE	Peering
P/PE	CE	Mobility
		Broadband

ASR 9000 Series Built to Last

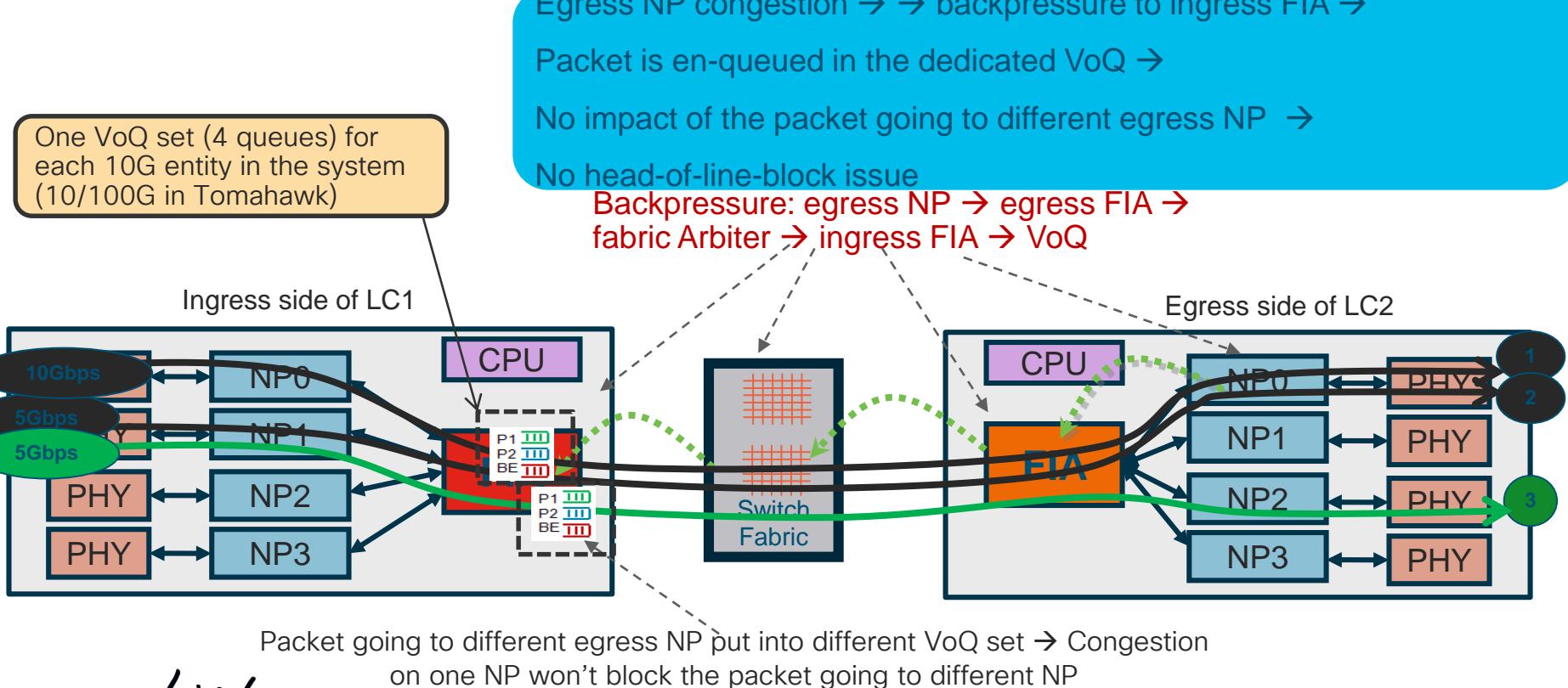
80x capacity increase in eight years; 95% power reduction/G



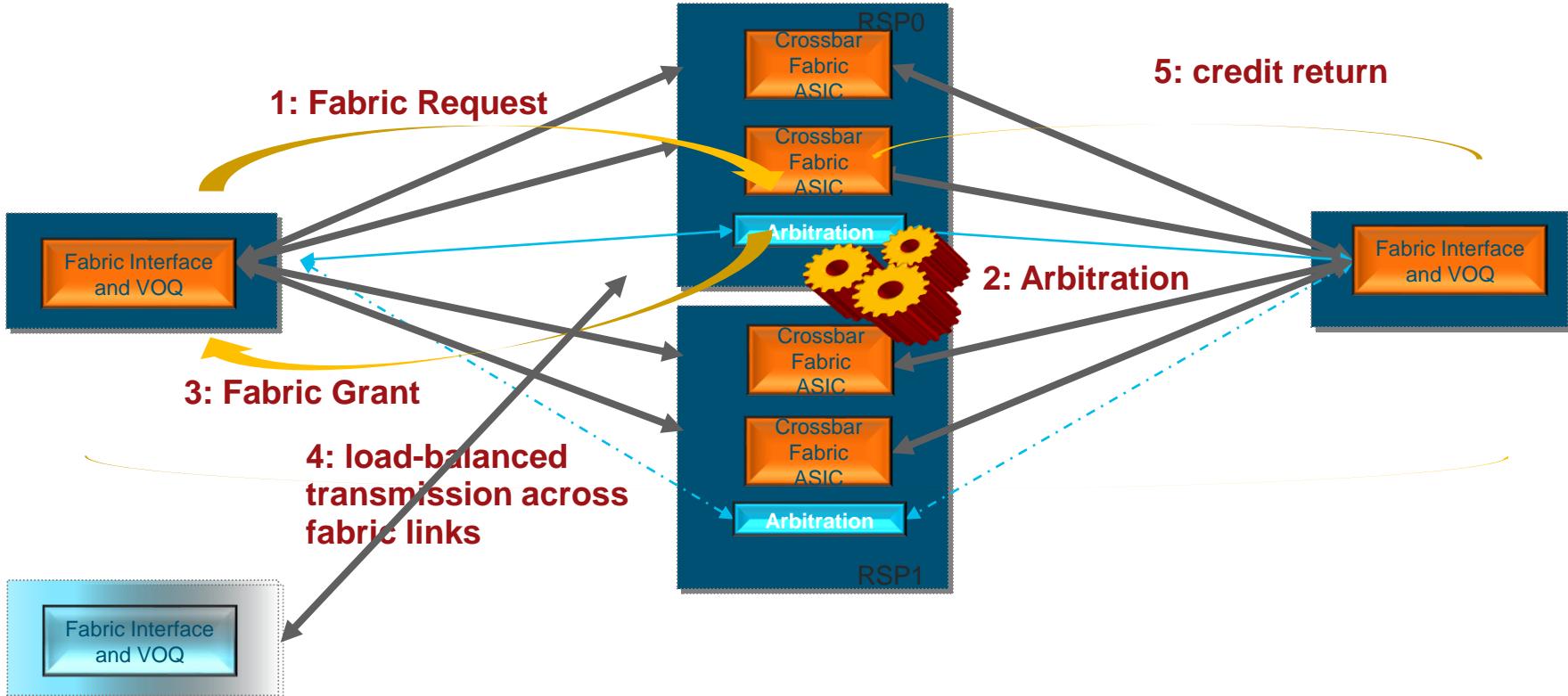
Edge Linecard Silicon Evolution



Typhoon and Tomahawk Linecards (ASR9000)

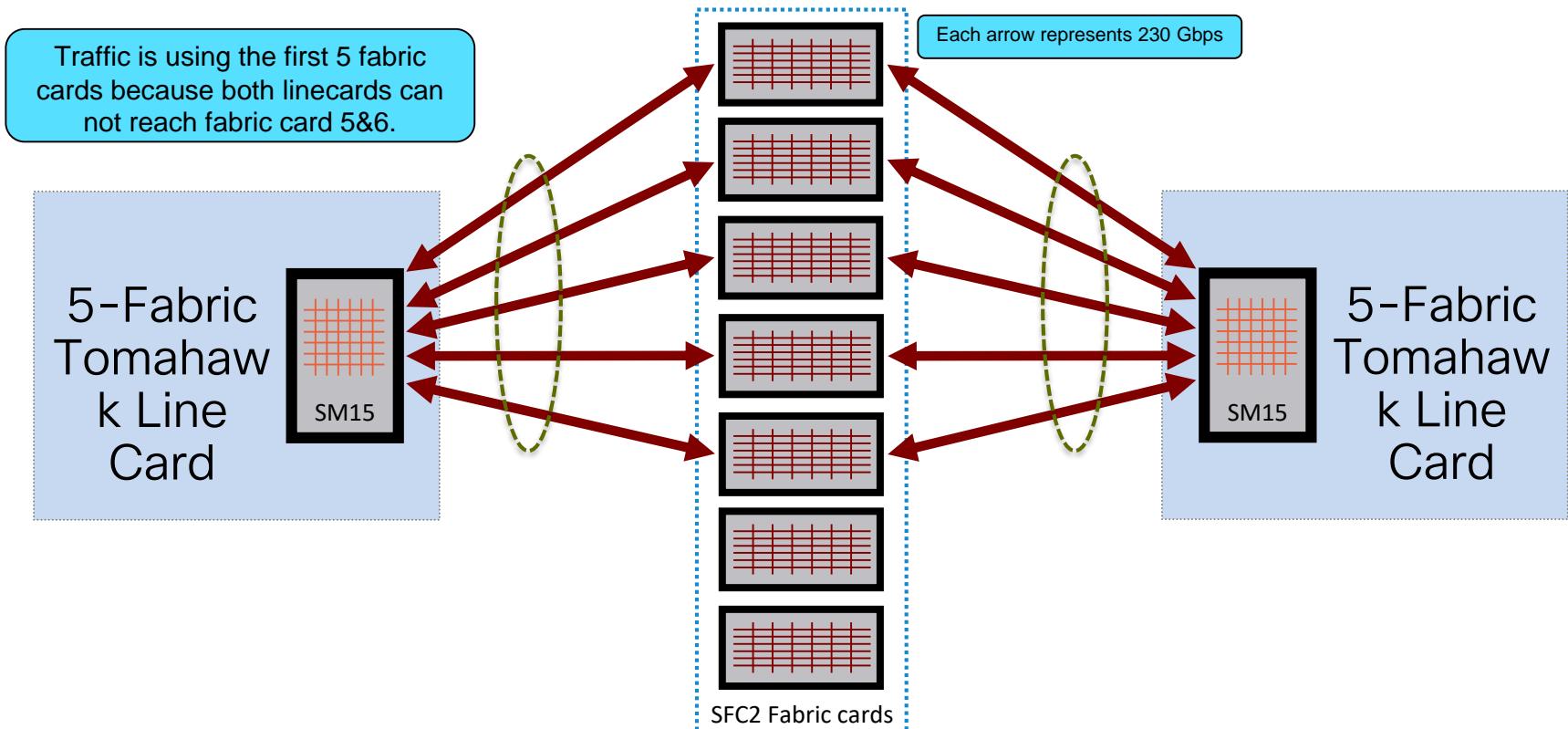


Fabric Arbitration Diagram

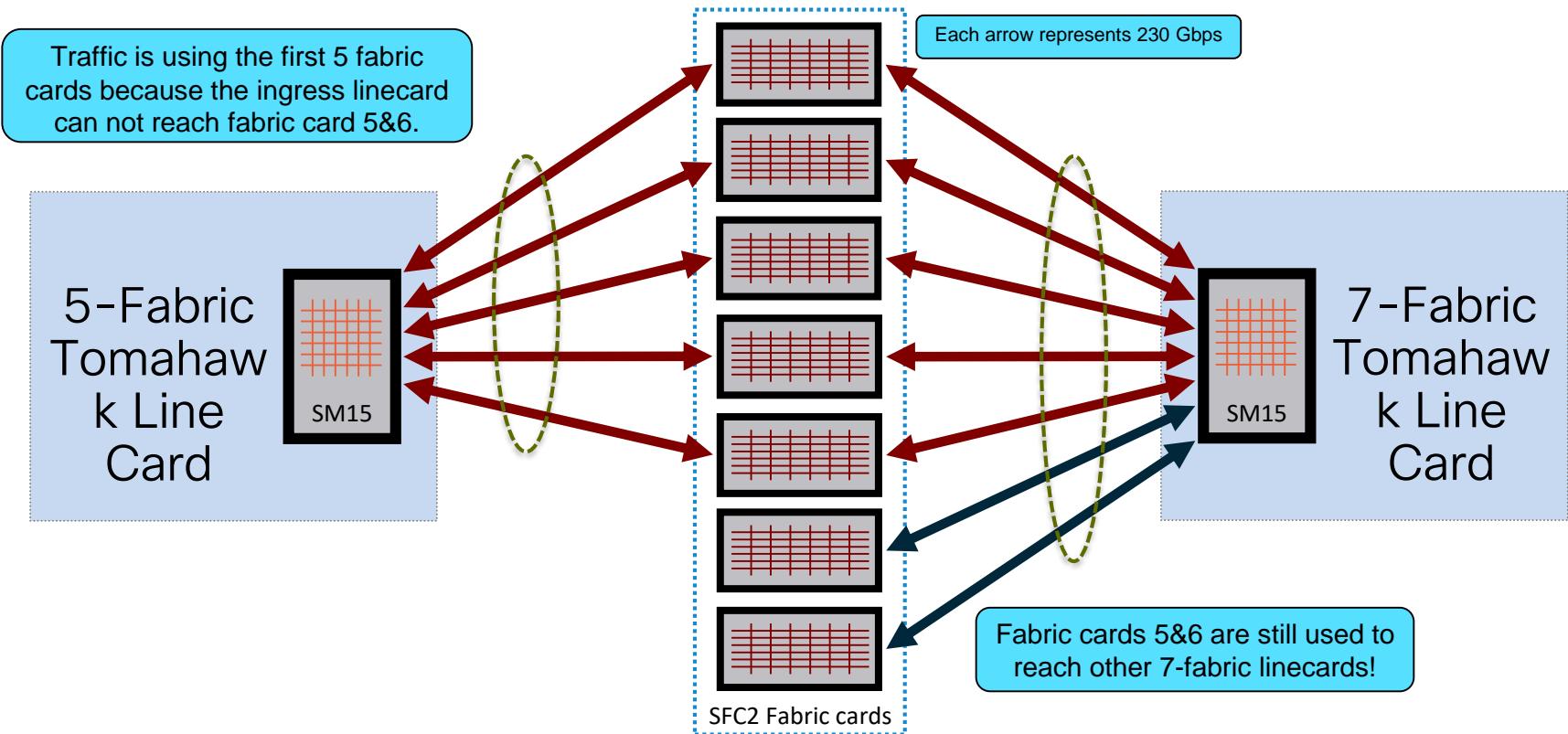


CISCO Live!

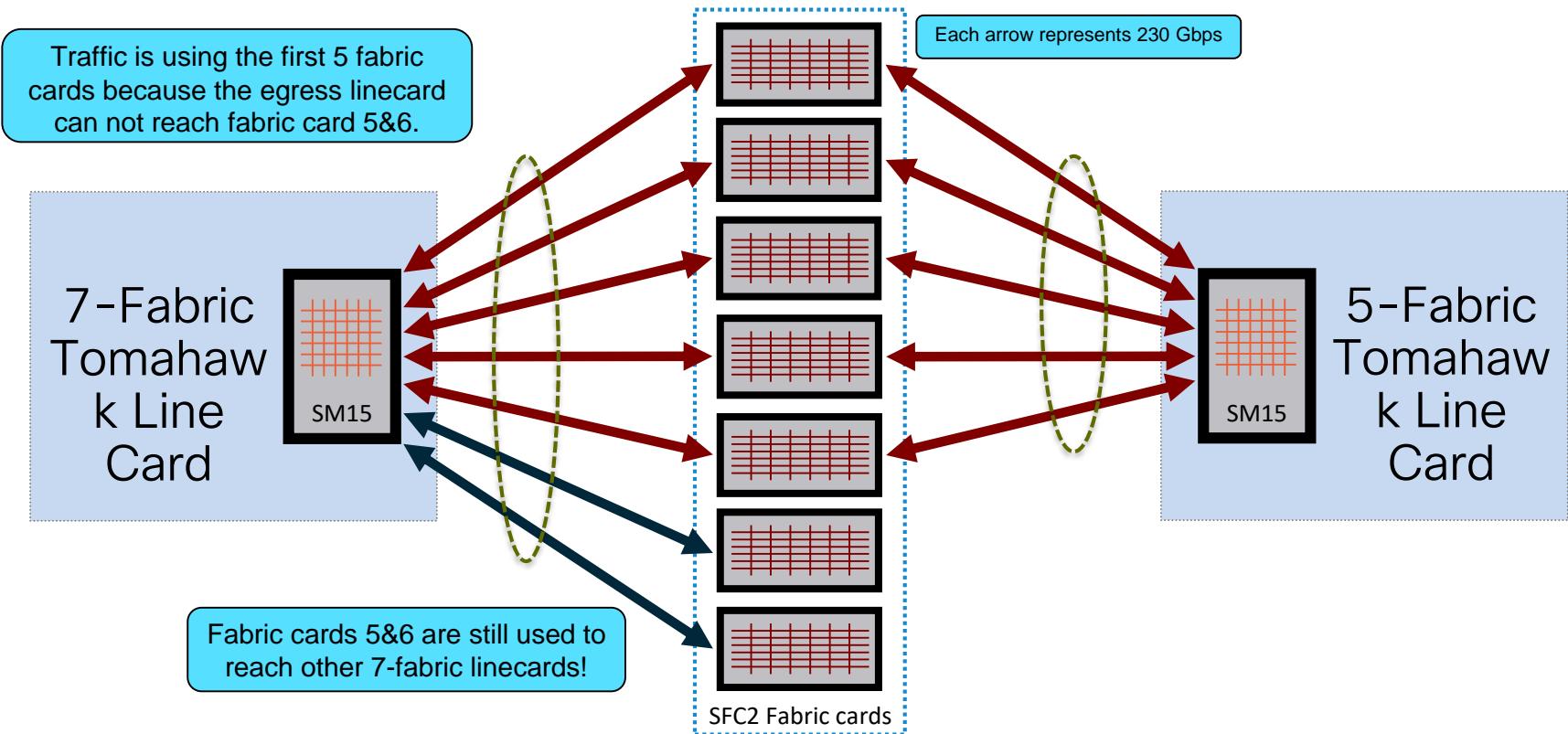
Fabric Interworking: 5-Fab LC to 5-Fab LC



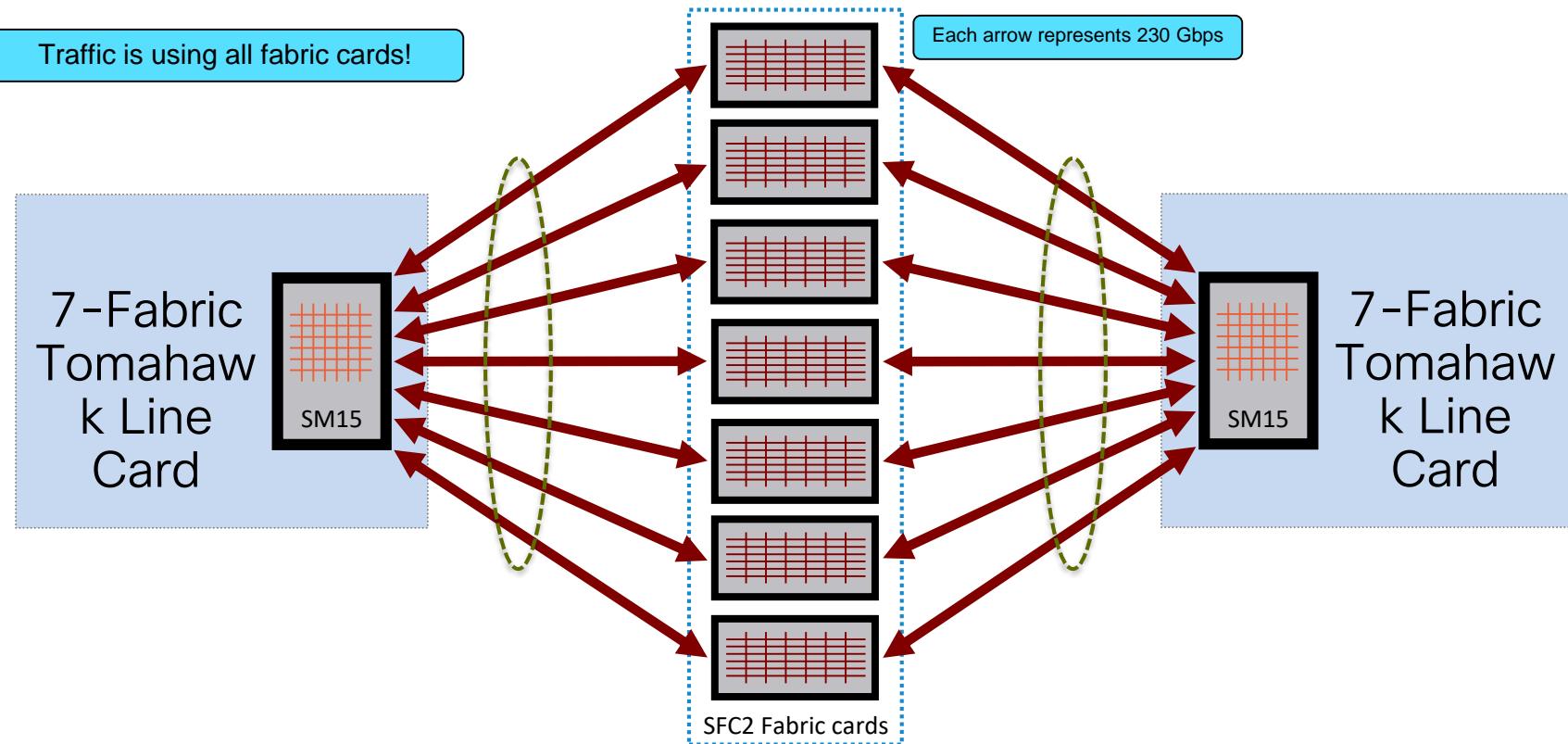
Fabric Interworking: 5-Fab LC to 7-Fab LC



Fabric Interworking: 7-Fab LC to 5-Fab LC



Fabric Interworking: 7-Fab LC to 7-Fab LC



High bandwidth mode?

- in high-bw mode, not a99-high-bw mode, mcast wont use 7 fabs, so there is a cap for mcast. that is limitation one.
- A rate-limiter is put in place to safeguard control packets getting dropped when you send line rate traffic to 7fab cards from 5 fabric cards. this can be alleviated by using qos policies to mark that traffic into p1/p2 fabric queues. (that is transit, not “for me”).
- this mainly pertains to 5fab 8x100 cards and 7fab cards like 12x100TH and 32x100LS.
- with 5 fab cards you can transport 1T. if all your “feeds” are on 5 fabs going to a 7fab card, then the limitation is there for the 1T out of that 7fab card.
- this is obviously with SFC2 cards. with sfc3's you get I want to say 600G per slot (so single fab card can carry all mod400 traffic).
- so in theory if you have more than 7 mod400's that can only do 5 fabs with sfc3 then you cap out in that card combination.

High bandwidth Mode!

- <https://community.cisco.com/t5/service-providers-documents/asr9k-chassis-fabric-modes/ta-p/3156749>
- Default: (=default in cXR)
 - 1024 VQI
 - Typhoon/Tomahawk, assumes 5 fabric
- HighBandWidth Mode (cfg: *fabric enable mode highbandwidth*) (=default in eXR)
 - 2048 VQI, Tomahawk only, mcast 5 fabs, ucast 5 or 7 fabs.
- A99-HighBandWidth Mode (cfg: *fabric enable mode A99-highbandwidth*)
 - *2048 VQI, TH only mcast 7 fab, ucast 7 fab*
- hw-module fia-intf-policer:
 - In default and HighBandWidth mode, the egress FIA rate limiter (enabled upon fabric degrade) calculation is based on status of first 5 fabric cards. However, in A99-HighBandWidth mode, the egress FIA rate limiter calculation is based on status of all 7 fabric cards.

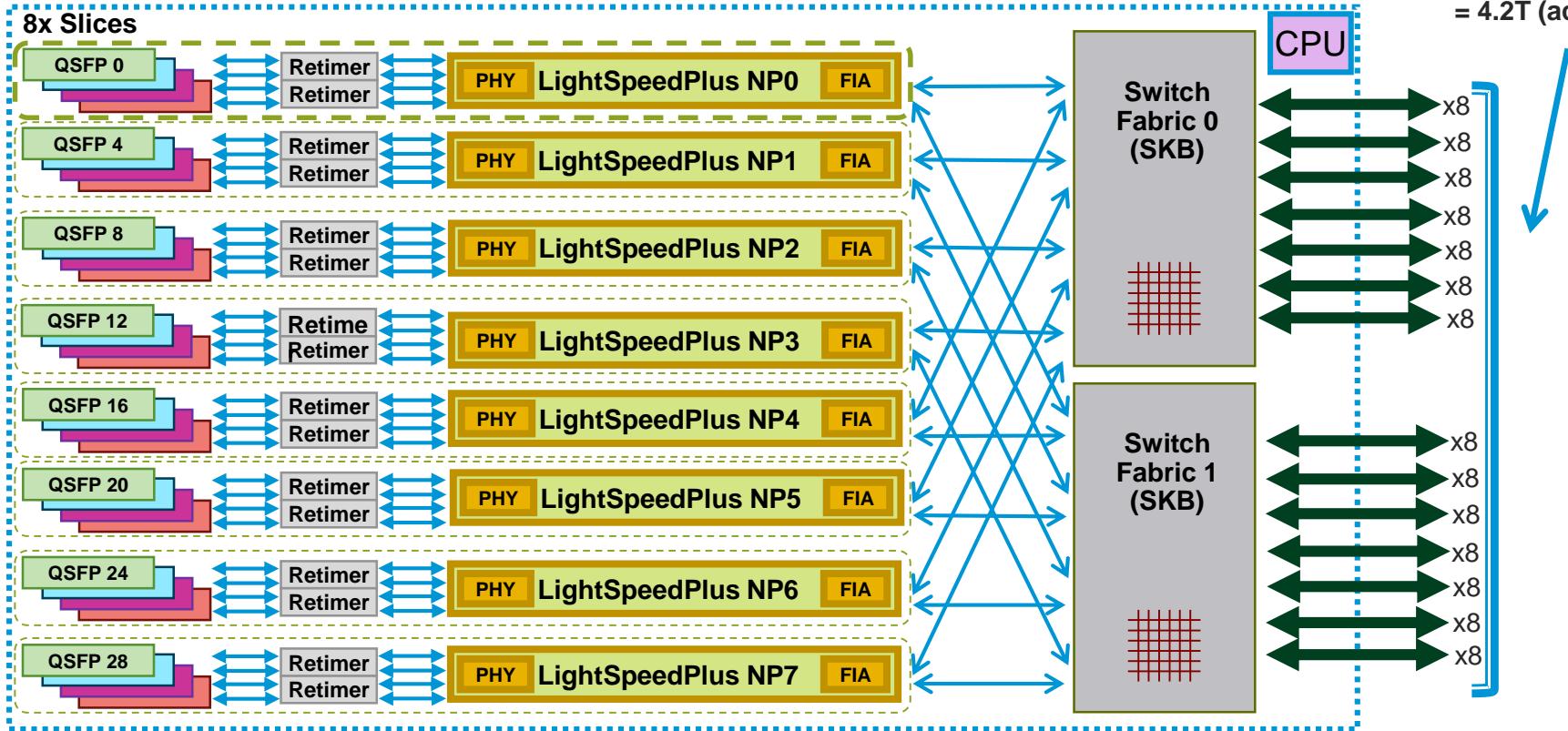
Lightspeed

32x100GE / 16x100GE / 8x100GE Linecard

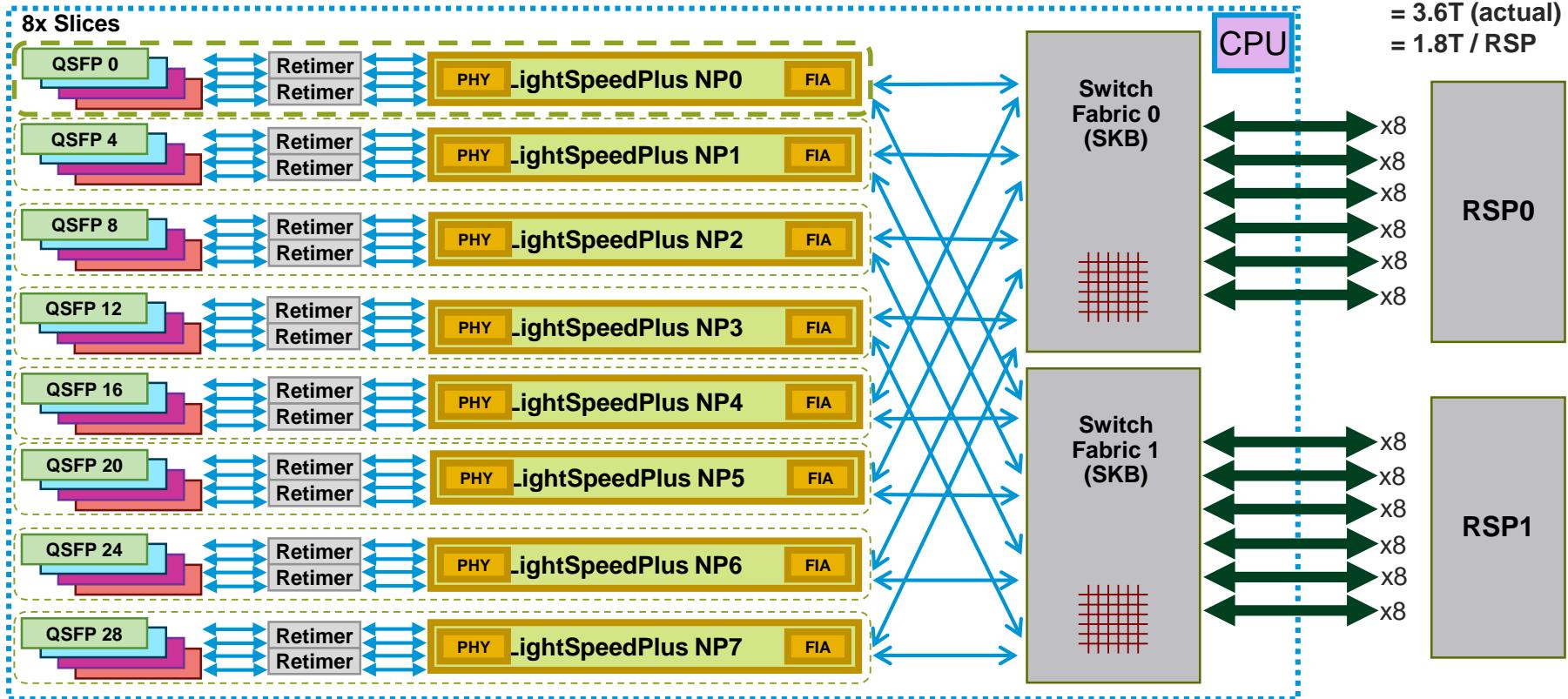


A99-32X100GE-X-SE/TR (7-fabric) LC Architecture (when used in 9922, 9912, 9910 & 9906)

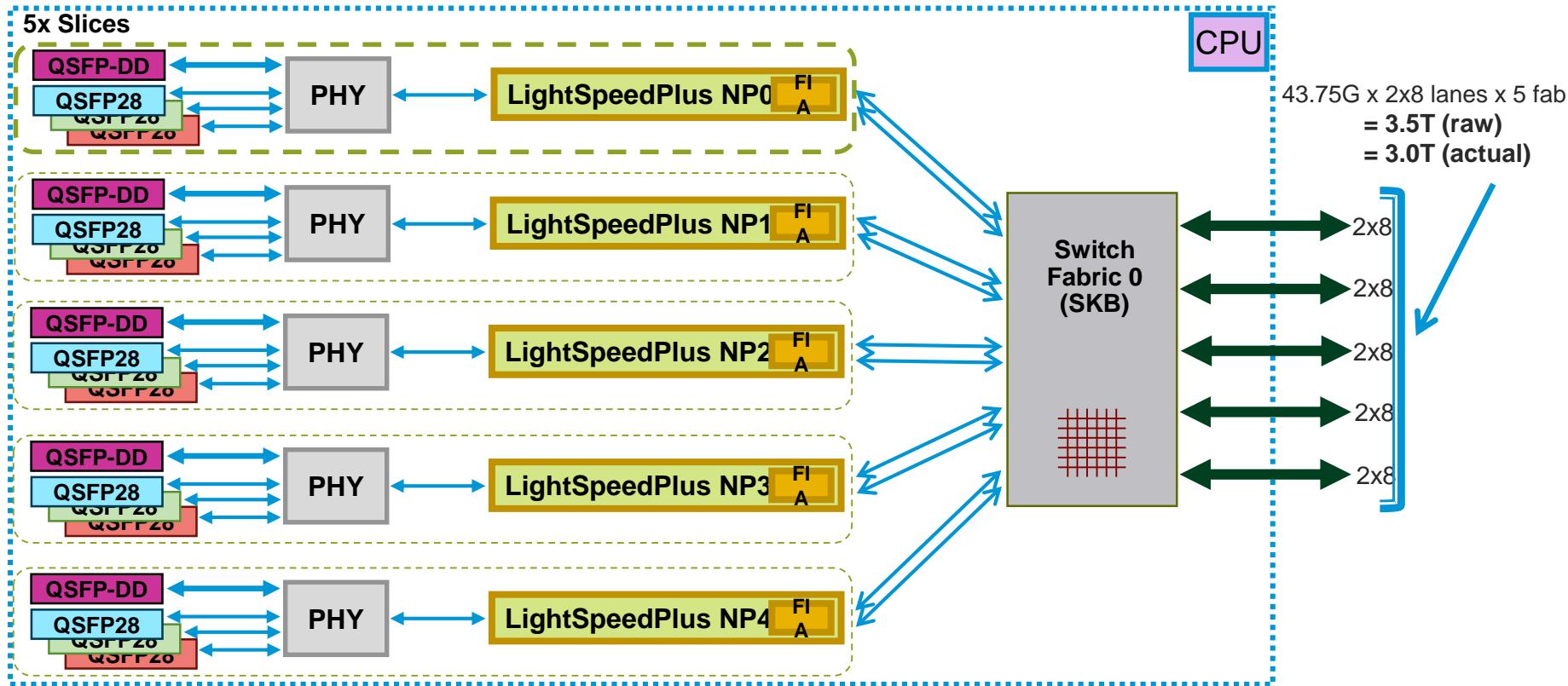
43.75G x 2x8 lanes x 7 fab
= 4.9T (raw)
= 4.2T (actual)



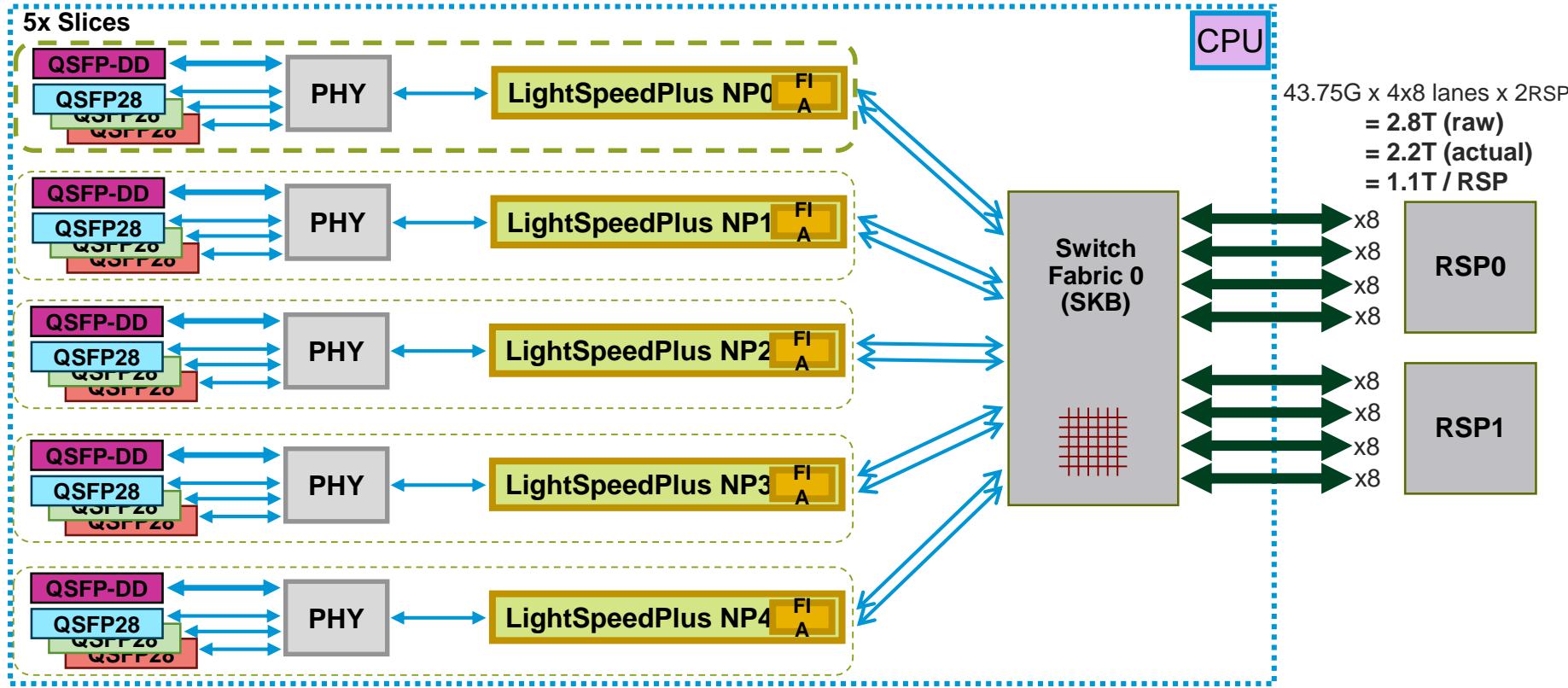
A99-32X100GE-X-SE/TR (7-fabric) LC Architecture (when used in 9904)



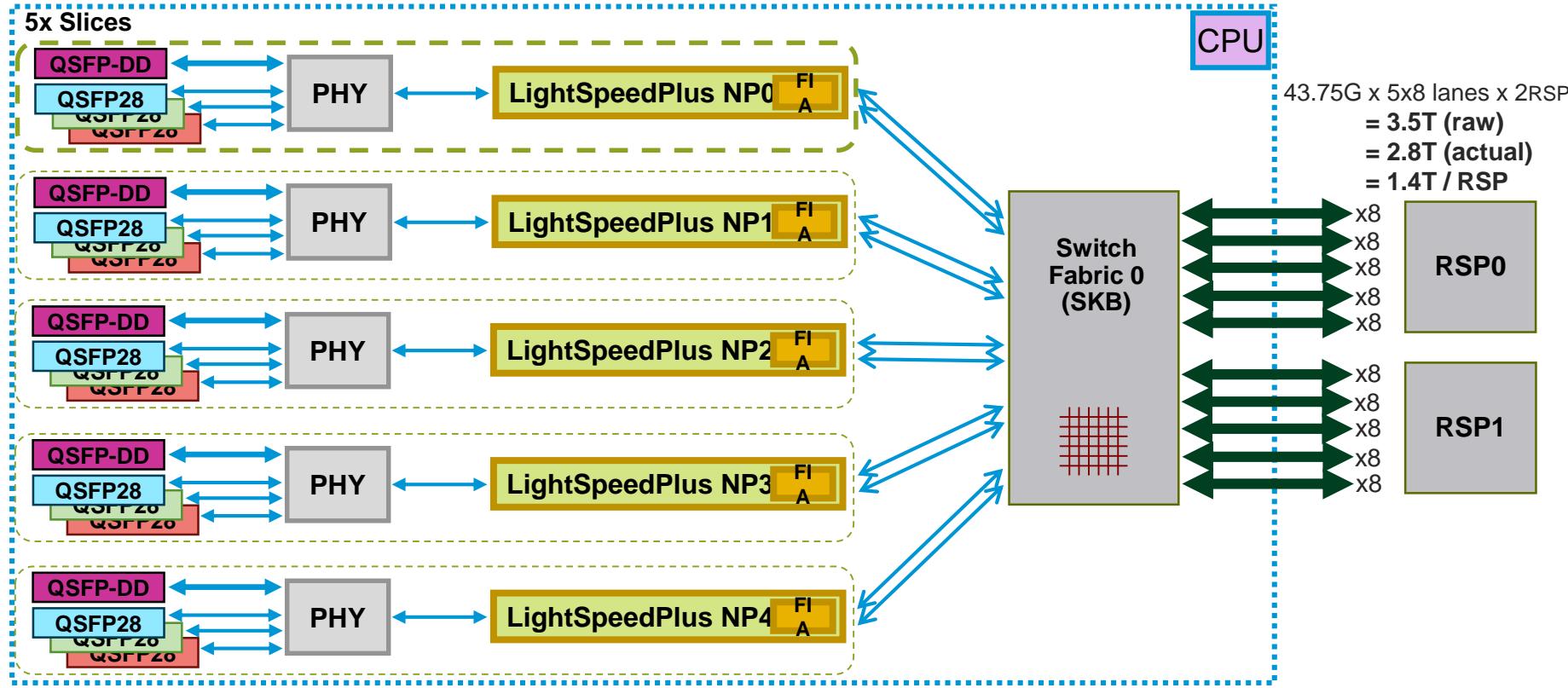
A9K-20HG-FLEX-SE/TR (5-fabric) LC Architecture (when used in 9922, 9912, 9910 & 9906)



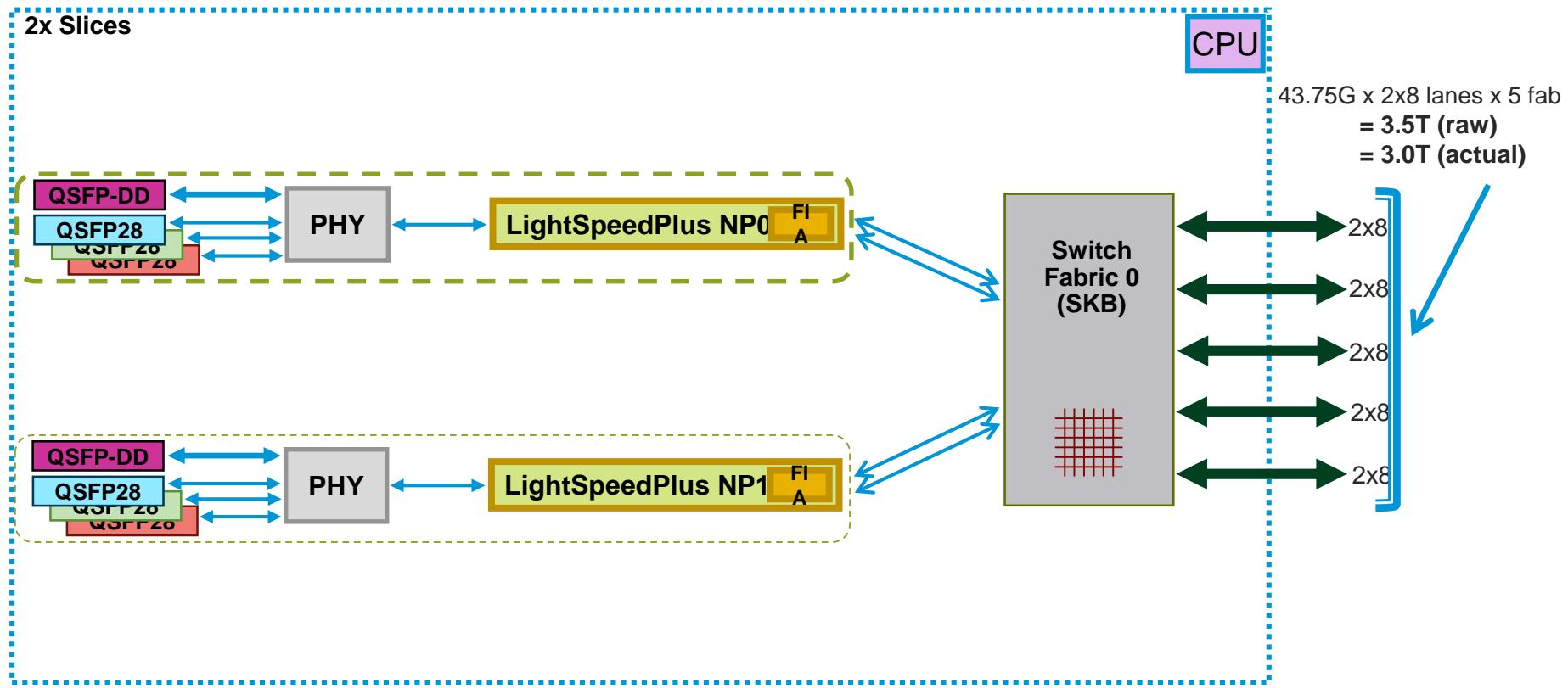
A9K-20HG-FLEX-SE/TR (5-fabric) LC Architecture (when used in 9010 & 9006)



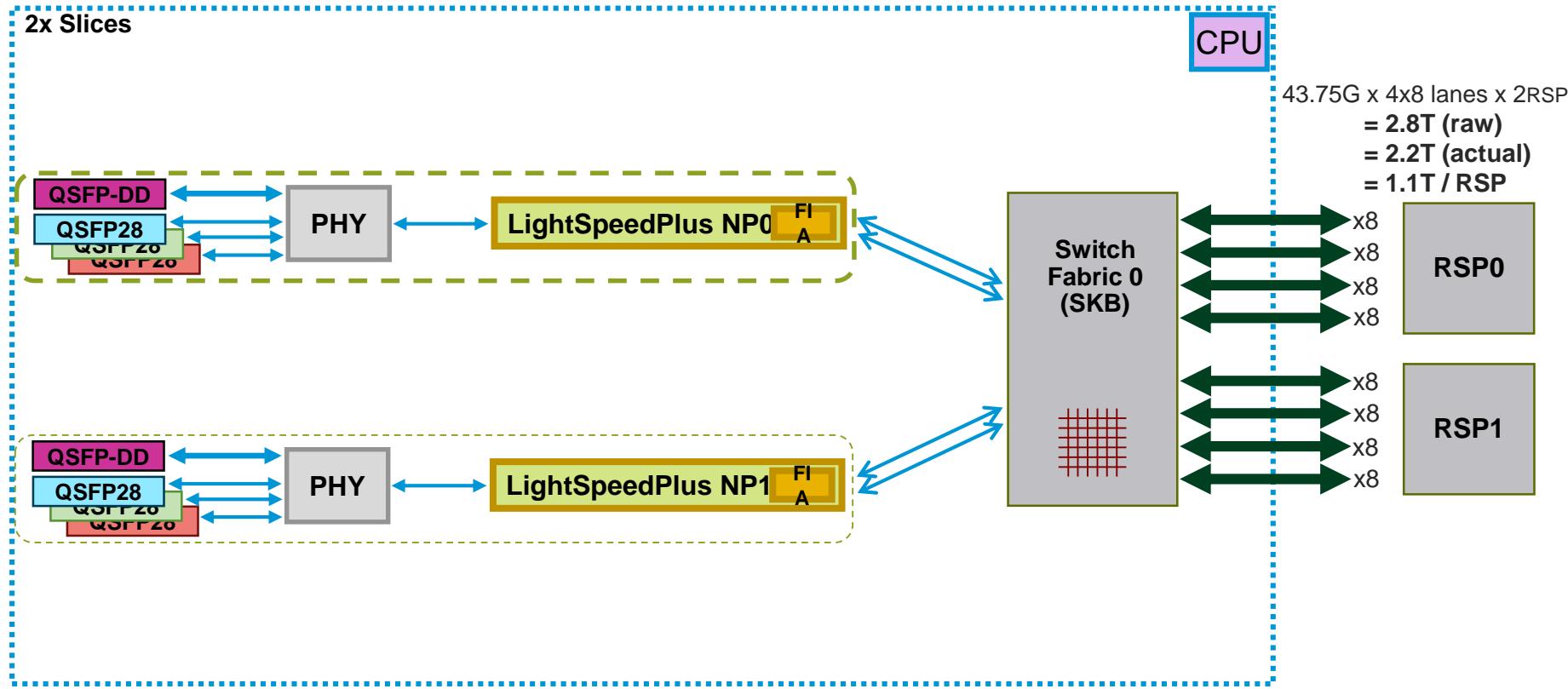
A9K-20HG-FLEX-SE/TR (5-fabric) LC Architecture (when used in 9904)



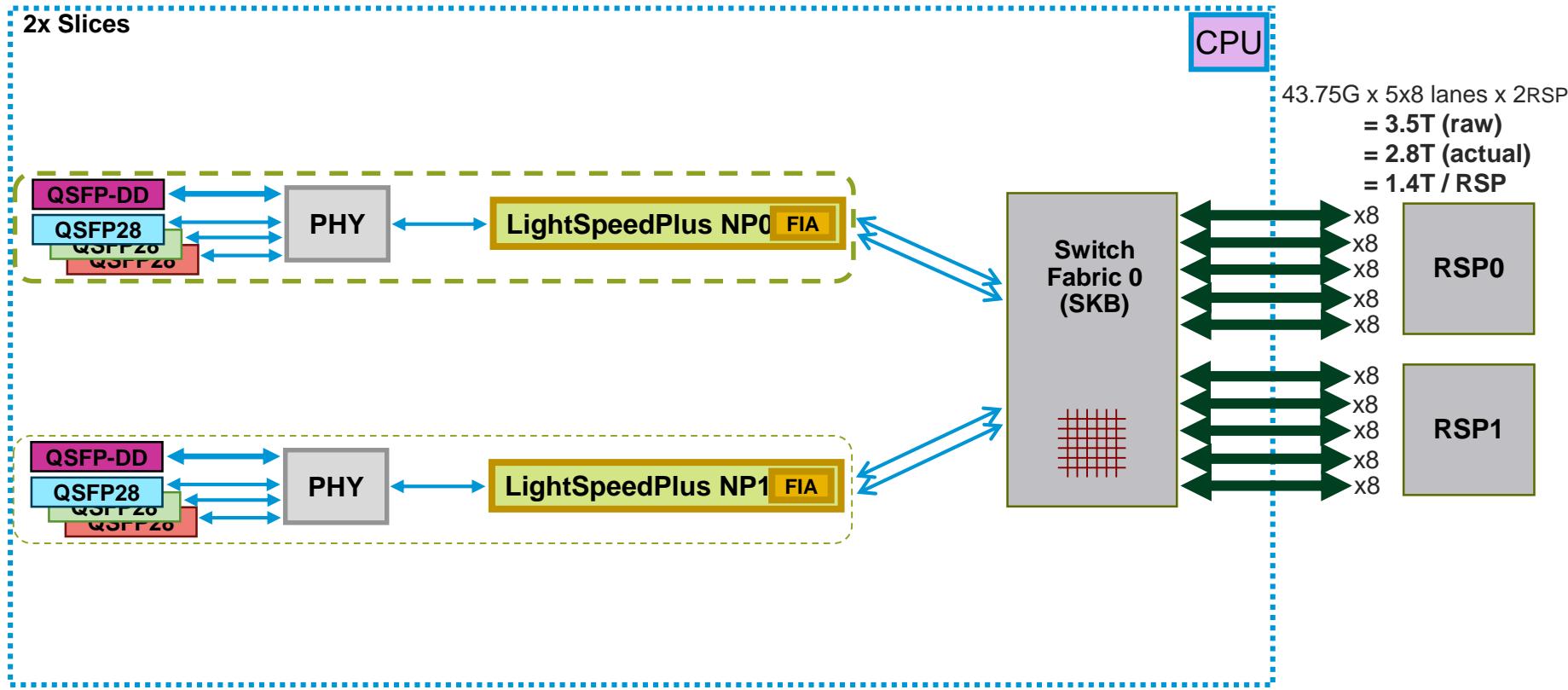
A9K-8HG-FLEX-SE/TR (5-fabric) LC Architecture (when used in 9922, 9912, 9910 & 9906)



A9K-8HG-FLEX-SE/TR (5-fabric) LC Architecture (when used in 9010 & 9006)



A9K-8HG-FLEX-SE/TR (5-fabric) LC Architecture (when used in 9004)



LightSpeed FAQ: Answers for The Impatient

1. What is Lightspeed?
 - A well known universal physical constant $c = 299,792,458$ meters per second (in vacuum)
2. I mean, what is ASR9000 LightSpeed?
 - It is the name of the fourth generation ASR9k hardware
 - It is a set of new linecards
 - 32 port 100G linecard: A99-32X100GE-SE & A99-32X100GE-TR
 - 16 port 100G linecard: A9K-16X100GE-SE & A9K-16X100GE-TR
 - 8 port 100G linecard: A9K-8X100GE-X-SE & A9K-8X100GE-X-TR
 - It is a set of new commons
 - Route Switch Processor: A9K-RSP5-SE & A9K-RSP5-TR
 - Route Processor: A99-RP3-SE & A99-RP3-TR
 - Switching Fabric Cards: A99-SFC3, A99-SFC3-S & A99-SFC3-T
 - ASR9922 Fan: ASR-9922-FAN-V3

LightSpeed FAQ: Answers for The Impatient

3. I want to run LightSpeed linecards in my ASR9922 and ASR9912. Do I need to upgrade my Commons?
 - This depends on which linecard you want to install:
 - 32x100GE: You need to upgrade to RP3 and SFC3
 - 16x100GE: You need to upgrade to SFC3. You can keep RP2 if you want
 - 8x100GE: You can stay with RP2 and SFC2
 - Also check the comments about the ASR9922 fan tray later in this FAQ
4. I want to run LightSpeed linecards in my ASR9910 and ASR9906. Do I need to upgrade my Commons?
 - This depends on which linecard you want to install:
 - 32x100GE & 16x100GE: You need to upgrade to RSP5 and SFC3
 - 8x100GE: You can stay with RSP4 and SFC2

LightSpeed FAQ: Answers for The Impatient

5. What about required Commons upgrades on ASR9904?
 - 32x100GE & 16x100GE: You need to upgrade to RSP5
 - 8x100GE: You can stay with RSP4. However, it is advisable to upgrade to RSP5 in order to ensure enough fabric bandwidth in case one RSP4 fails
6. I want to run LightSpeed linecards in my ASR9010 and ASR9006. Do I need to upgrade my Commons?
 - This depends on which linecard you want to install:
 - 32x100GE: This linecard is not supported in ASR9010 and ASR9006
 - 16x100GE: You need to upgrade to RSP5
 - 8x100GE: You can stay with RSP4. However, it is advisable to upgrade to RSP5 in order to ensure enough fabric bandwidth in case one RSP4 fails

LightSpeed FAQ: Answers for The Impatient

7. All this stuff is way too complicated. Do we have a tool / an app for this?
 - Well...

Linecard Type	Commons
32 Port	LightSpeed
16 Port	LightSpeed (Can keep RP2)
8 Port	Tomahawk

LightSpeed FAQ: Answers for The Impatient

8. How many Switching Fabric Cards do I need to install?

- This depends on the Chassis and on which linecard you want to install:
 - ASR9922 & ASR9912
 - 32x100GE 7-fabric linecard: You need to install 7 SFC3s
 - 16x100GE 5-fabric linecard: You need to install min 4 SFC3s
 - 8x100GE 5-fabric linecard: You need to install min 4 SFC3s
- ASR9910 & ASR9906
 - 32x100GE 7-fabric linecard: You need to install 2 RSP5s and 5 SFC3s
 - 16x100GE 5-fabric linecard: You need to install 2 RSP5s and min 2 SFC3s
 - 8x100GE 5-fabric linecard: You need to install 2 RSP5s and min 2 SFC3s

LightSpeed FAQ: Answers for The Impatient

9. All this stuff is way too complicated. Do we have a tool / an app for this?
 - Well...

Linecard Type	Fabrics
32 Port	7 7xSFC3 - or - 2xRSP5 & 5xSFC3
16 Port	4 4xSFC3 - or - 2xRSP5 & 2xSFC3
8 Port	4 4xSFC2/SFC3 - or - 2xRSP5 & 2xSFC3

LightSpeed FAQ: Answers for The Impatient

10. Can I mix RSPs/RPs/Fabrics from different generations, meaning keeping some Tomahawk Commons in the chassis and add LightSpeed Commons to it?
 - No, all Commons need to be from the same generation
 - Exception 1: ASR9922/ASR9912: SFC3 can be used with RP2
 - Exception 2: During hardware upgrades. Check the MOPs for more details
11. Can I keep Typhoon hardware in the same chassis?
 - No. LightSpeed can only co-exist with Tomahawk hardware
 - Rule since ASR9k day 1: Two consecutive linecard generations are supported in same chassis

LightSpeed FAQ: Answers for The Impatient

12. Do I need to upgrade my fans in order to accommodate LightSpeed linecards?

- On ASR-9922 chassis, you need to use the new Version 3 fans ASR-9922-FAN-V3 if you use the 32 port LightSpeed linecard
- ASR-9922 with 8 port and 16 port linecard requires only V2 fans (same as Tomahawk)
- On all other chassis types, there is no new requirement for fan versions. Just make sure you have the fans which support Tomahawk (V2 for ASR-9010/9006, V1 for all others)

Linecard Type	Commons
32 Port LS in ASR 9922	ASR-9922-FAN-V3
All other	Same requirements as Tomahawk

13. For the time being, I only want to add LightSpeed RSP5/RP3/SFC3, but no LightSpeed linecards. Do I need to upgrade my fans for this?

- No. LightSpeed commons do not require specific fan versions

LightSpeed FAQ: Answers for The Impatient

14. I want to run LightSpeed in my ASR9k system. Do I need to upgrade my Power System?
 - No. LightSpeed does not require a specific Power System version
 - You may use the existing power system supported with your ASR 9K chassis
 - Just use the Cisco Power Calculator (<http://tools.cisco.com/cpc>) to make sure you have enough power budget in your system to accommodate the new LightSpeed cards
15. What about the Software requirements?
 - LightSpeed support starts with IOS XR 64bit Release 6.5.15
 - Please note that LightSpeed is not supported with the classic 32bit IOS XR!
16. What kind of optics can I use with LightSpeed line cards?
 - LightSpeed supports QSFP28 optics, including breakouts

Key Lightspeed NPU Hardware Capabilities

- New in-house developed NPU
- 420G bandwidth (bi-directional), 300Mpps+ full duplex forwarding per LightSpeed NPU
 - (Tomahawk: 240G / 150Mpps+)
- 1.8Tb to 4.2Tb per slot fabric capacity
 - depending on chassis type
- 22 billion transistors
- 2M MAC addresses
- 6M v4 or 6M v6 routes
- 48K queues per NPU



LightSpeed - Preparing for Zettabyte Era

Innovation

Cisco NPU 4 in 1 (16nm): Integrated NPU, PHY, FIA and Memory

Native support for 10/25/40/100/400G

Integrated 100GE FEC

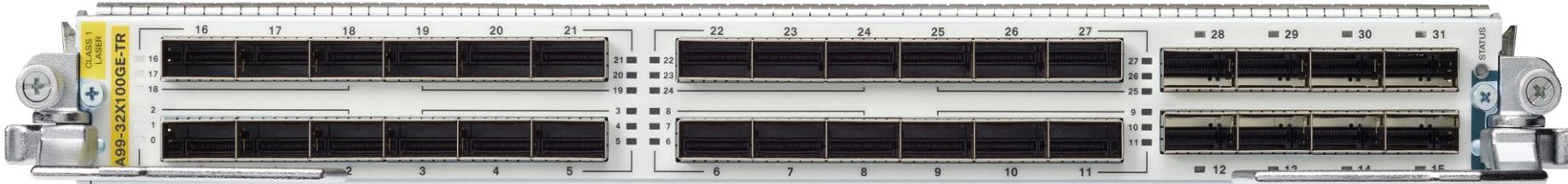
Leading the Market

4 x capacity increase per system

10GE, 40GE, 100GE and 400GE densities w/ rich features

Hitless FPD upgrade possible (no LC reload)

Sub-Second ISSU



Lower TCO

Low OPEX:

- Drastically lower power profile: ~ .5W/GE
- Improvement over Tomahawk w/ power down capability of the complete slice path including NP

Low CAPEX - Vortex and PAYG

Scale

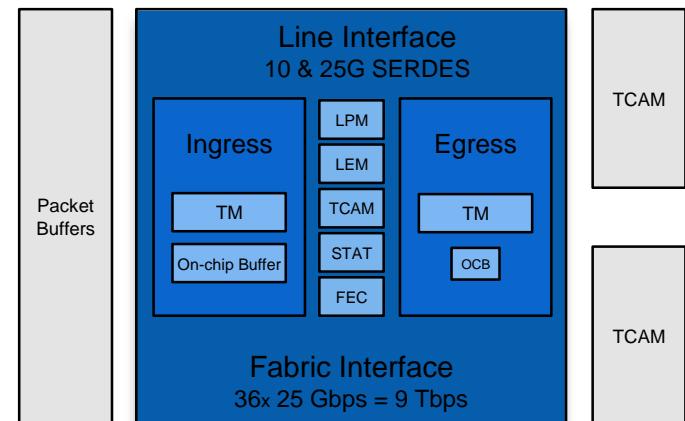
Ultra high control plane scale with eXR

HW acceleration for L2 classification, hashing, prefix lookup, ACL range compression, header re-write, flow ordering, statistics, policers, WRED

NCS5500

Forwarding ASIC Detail

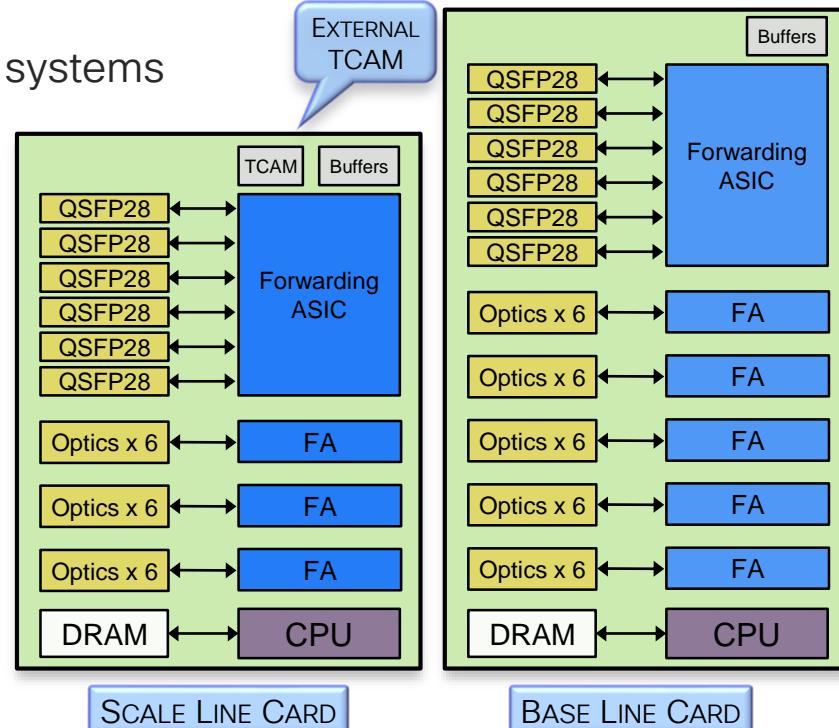
- Integrated Forwarding and Fabric Interface
- 720 Mpps packet forwarding
 - 600 Mpps with external TCAM forwarding
- Small internal buffers and TCAM
- Interface for external packet buffers, stats, and optional TCAM
 - Deep GDDR5 buffers (4GB)
- Traffic Manager
 - 96K queues
 - 3 million packet buffers
 - WRED, Hierarchical scheduling
- Line Interface
 - Supports 10G and 25G lanes
 - 4x 25 G for QSFP28 (100G or 4x 25G)
 - 4x 10G for QSFP+ (40G or 4x 10G)
- Fabric Interface
 - 36x 25G links into fabric



Scale Options

Base and Scale Forwarding

- Base and Scale options for modular and fixed systems
- Base line cards have highest density
 - On-chip FIB and small TCAM for ACLs/QoS
- Scale line cards increase FIB and ACL
 - 10 MB off-chip TCAM
 - Fewer forwarding ASICs per line card
- External TCAM is a shared resource
 - IPv4 & IPv6 route scale
 - Ingress ACL / QoS matching scale

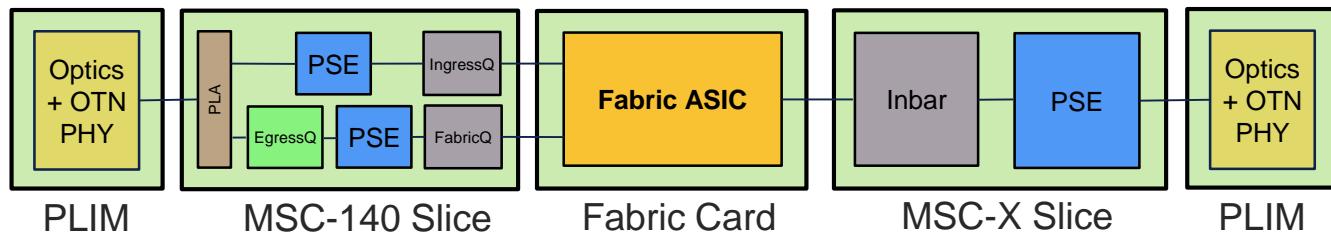


NCS 5500 Architecture

Comparison with Traditional XR Platforms

- Two-lookup architecture
 - Ingress to identify the destination chassis / line cards
 - Egress to identify the interface / VLAN / adjacency

CRS-3/X



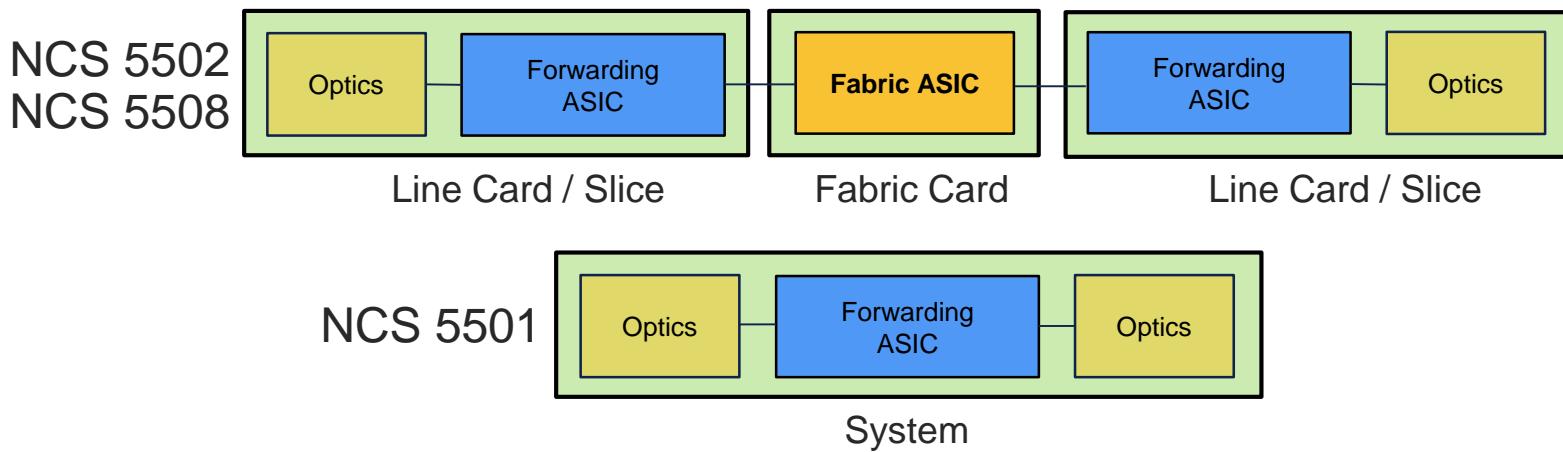
ASR9900



NCS 5500 Architecture

Comparison with Traditional XR Platforms

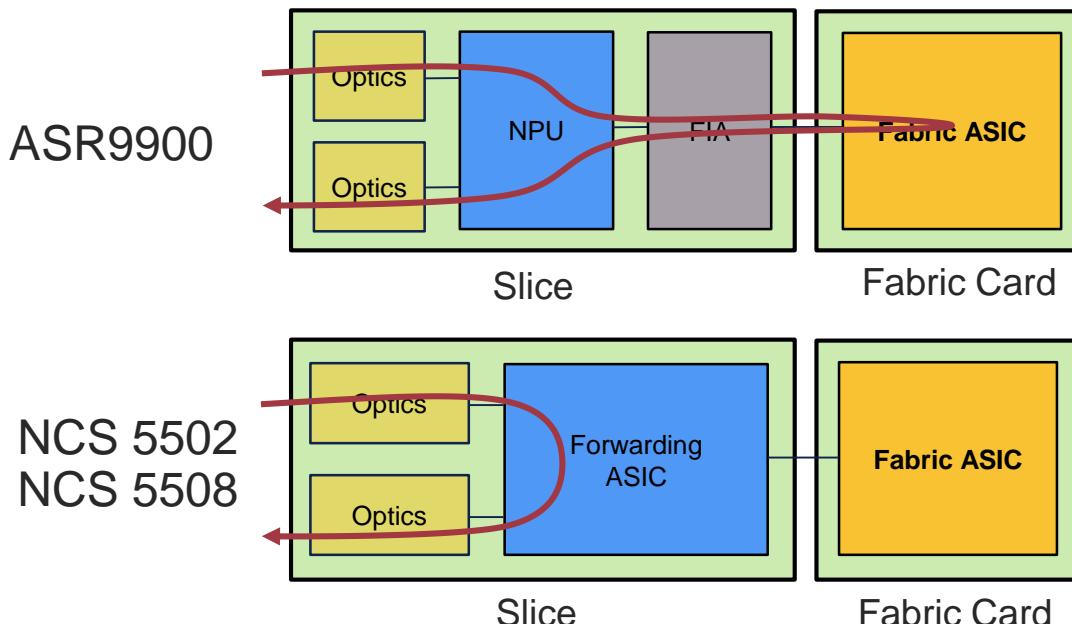
- Simplification: fewer internal elements
- Single-lookup architecture at ingress
 - VOQ-only Model



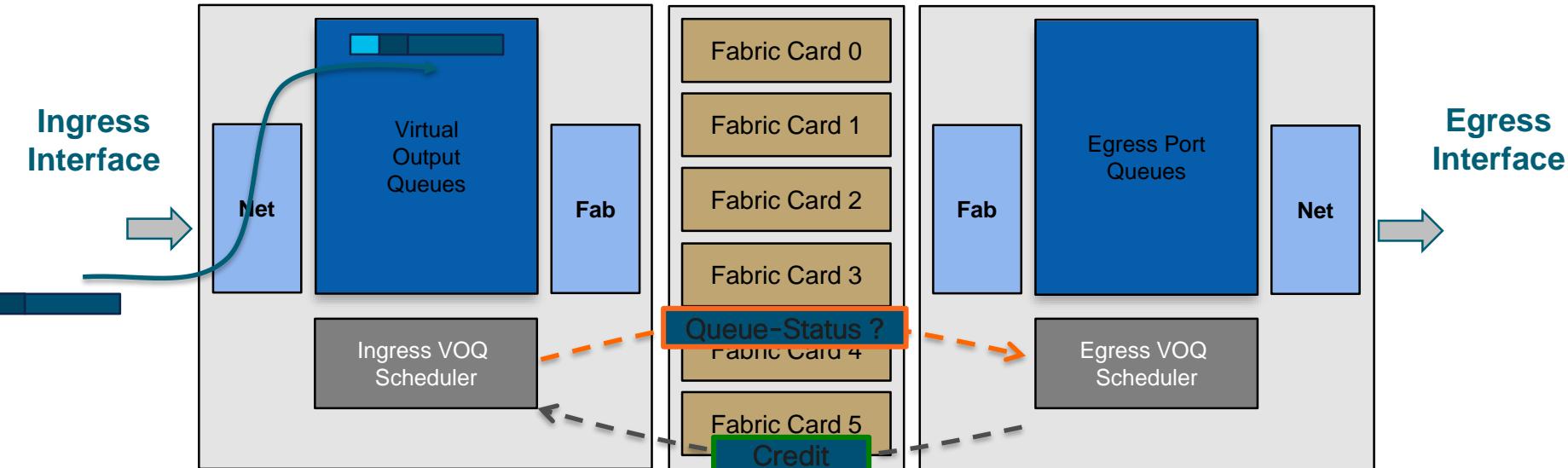
NCS 5500 Architecture

Local Forwarding

- Local traffic on NCS 5500 series can be routed by the FA without going through the fabric, lower latency

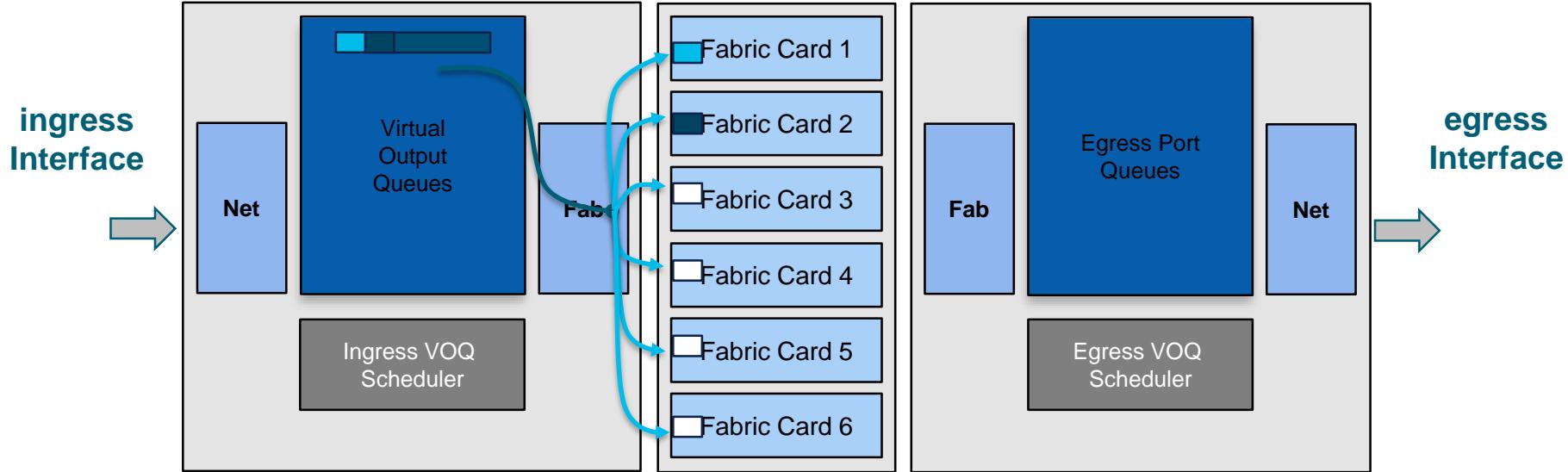


VOQ-only Architecture



- Packet is received on ingress interface, classified, and stored in internal buffer
- Single lookup
- Ingress VOQ scheduler polls Egress scheduler (maintaining a local VOQ DB)
- Egress answers with a credit-message

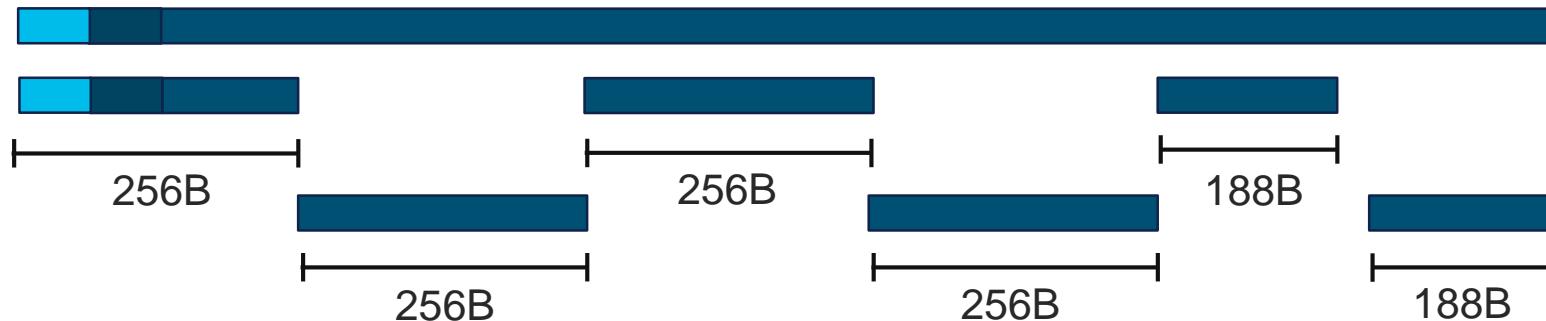
VOQ-only Architecture



- Packet is split in cells and load balanced among the fabric cards
- Cells are transported to the egress line card

VOQ-only Architecture

- Let's take the example of a 1400B packet

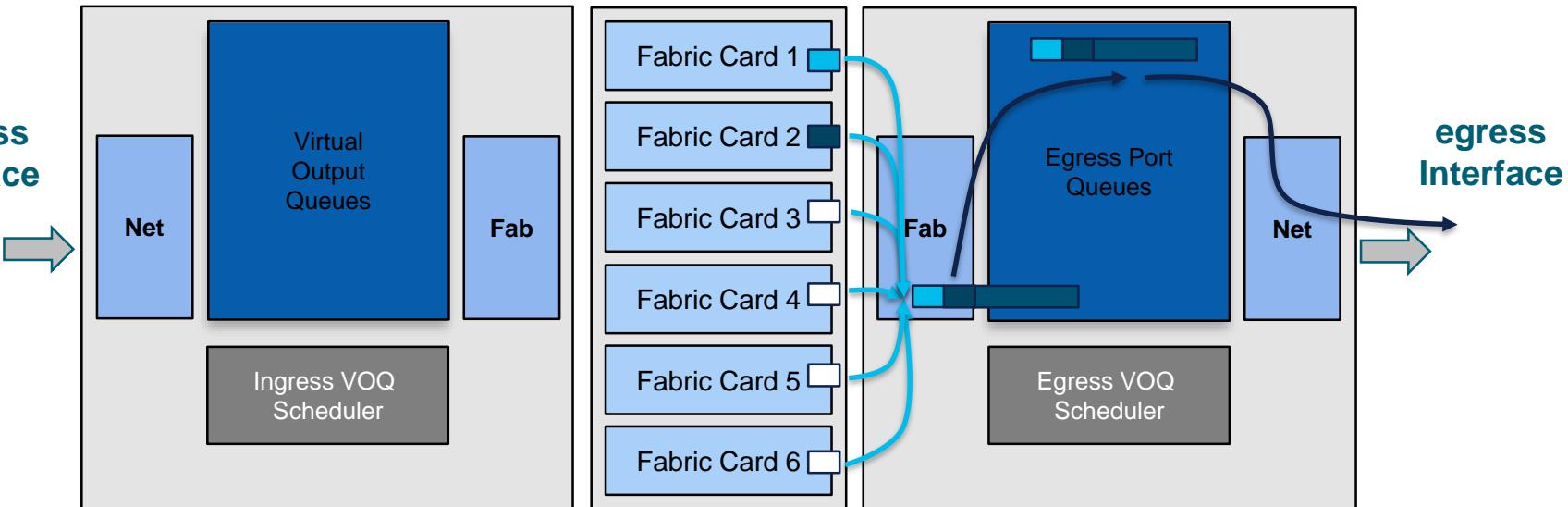


- If the last part is between 256B and 512B, we divide by 2

$$1400 - 4 \times 256 = 376 = 2 \times 188$$

VOQ-only Architecture

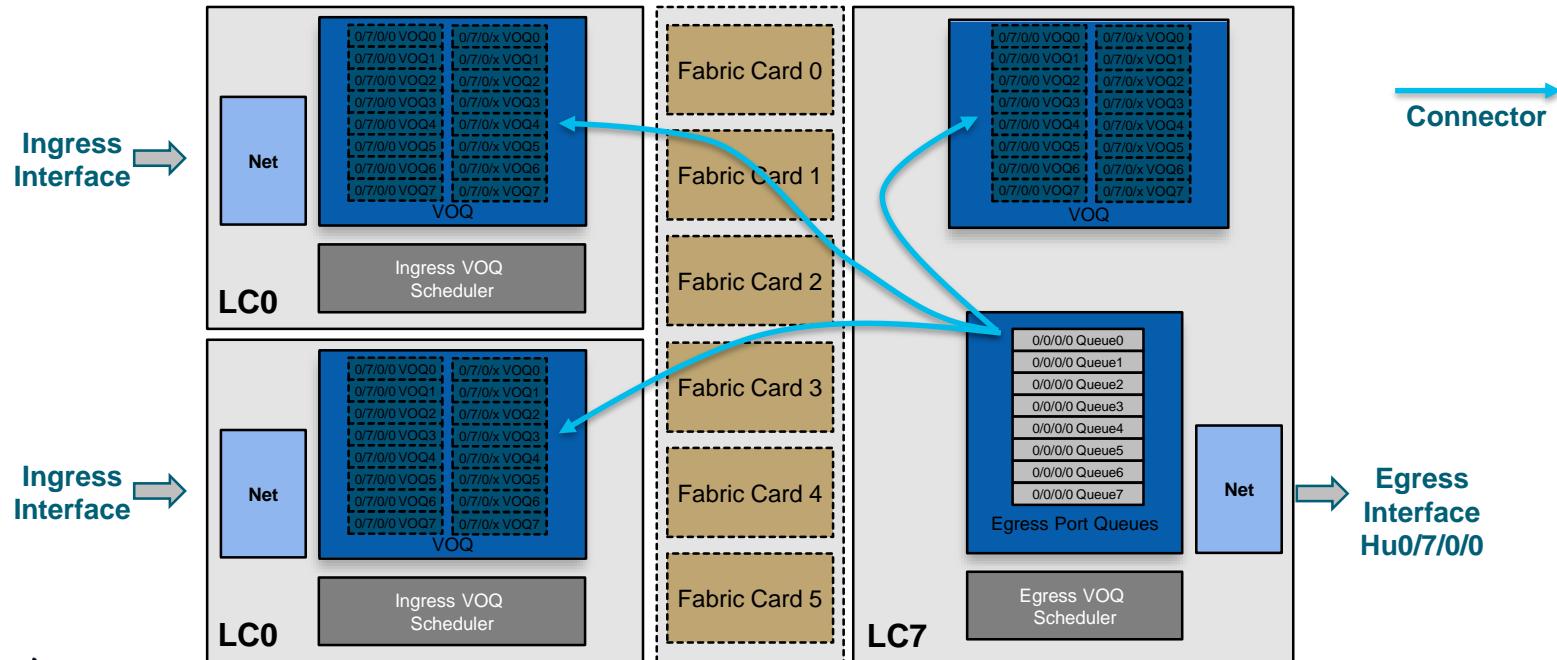
ingress Interface



- Cells are collected and packet re-assembled
- Packet is stored in the port queue
- Finally packet is transmitted through the egress interface

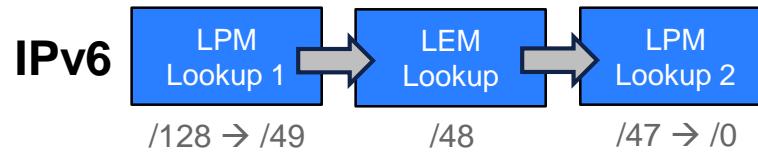
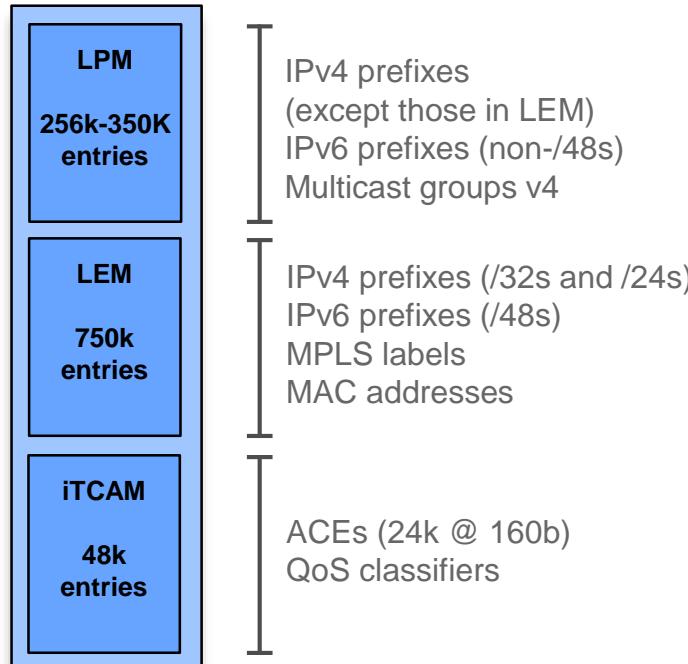
Virtual Output Queues (VOQ)

- All the queues of any egress interface are replicated to all ingress devices which want to send packet to that interface.
These queues are called VOQ (Virtual Output Queues)



Memory Structure for non-eTCAM Systems

In IOS XR 6.1.3: Host Optimized Mode

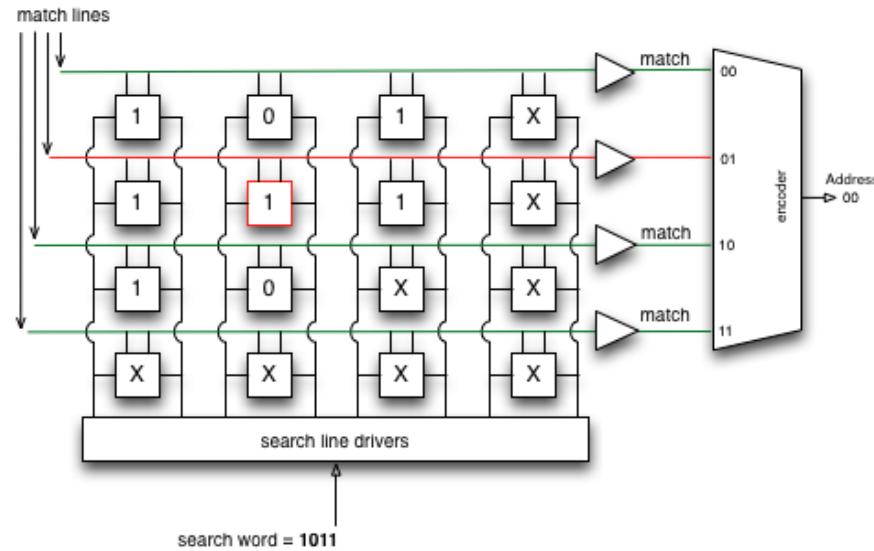


TCAM trouble with forwarding

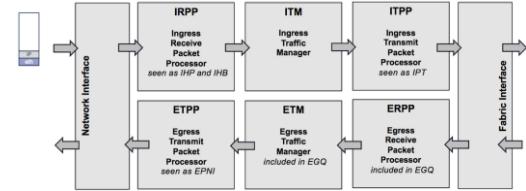
Longest Exact Match/Longest Prefix Match

Pfx/mask	TCAM format
101/3	101X
111/3	111X
10/2	10XX
0/0	XXXX

But this is an ORDERED list...!
(needs driver support)



DNX forwarding blocks



- **IRPP (Ingress receive packet processor)** - (IHB and IHP) has two cores and it manages the ingress packet classification.
- **ITM (Ingress traffic manager)** - Is where the VoQ (Virtual output queue) and it is stored into memory: 2 types of memory OCBM (on chip buffer memory) and External Dram.
 - If the queues are not congested it will go to the OCBM but if there is congestion, the packet will go to the external DRAM.
- **ITPP (Ingress transmit packet processor)** is part of IPT and is where the packet can be replicated (ingress multicast replication), split into cells and load balanced to the fabric using any of the 3 links to the fabric interface.
- **ERPP (Egres receive Packet Processor)** and **ETM (Egress traffic manager)** is part of EGQ block.
 - Here the first 128 bytes are extracted and we can do egress ACL and egress repliciaton for multicast is done here.
- **ETM (Egress traffic manager)** - This is only an internal buffer memory for unicast and multicast traffic
 - remember that if there is congestion ingress traffic manger will handle VoQ.
 - Scheduler works here and available credits are granted using ETM.
- **ETPP (Egress transmit packet processor)** is part of EPNI (Egress Porcess Negwork Interface) and is where headers sent from ingress are removed.
 - Encapsulation is constructed (Labels / GRE headers / Ethernet header).

Useful commands

NCS5508-B#show controllers fia diagshell 0 "diag counters g" location 0/1/CPU0

- this is an XR command that invokes a broadcom shell command which will show the internal databases and how much work they are currently using.

Sample output

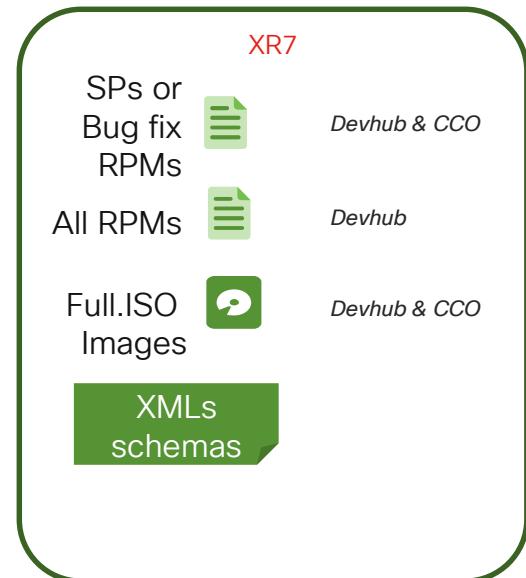
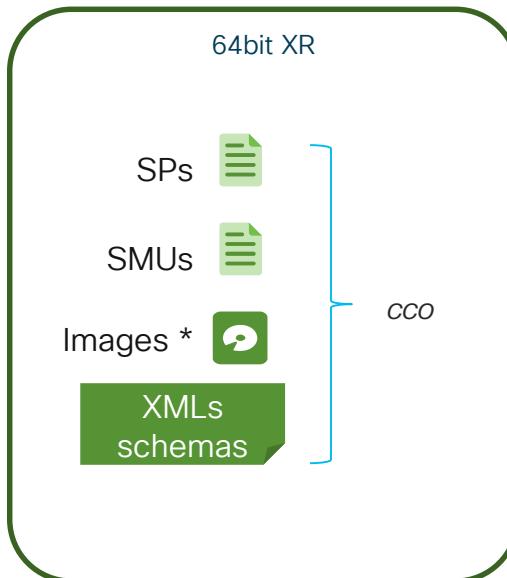
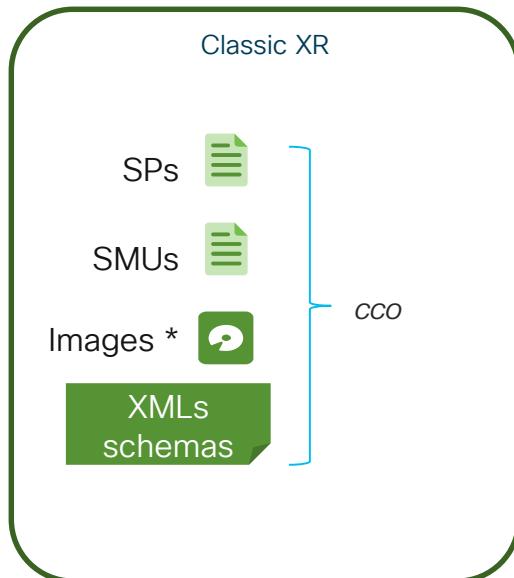
JERICHO NETWORK INTERFACE	
NBI	
RX_TOTAL_BYTE_COUNTER	= 69,744
RX_TOTAL_PKT_COUNTER	= 44
RX_TOTAL_DROPPED_EOPS	= 0
IRE	
CPU_PACKET_COUNTER	= 0
NTF_PACKET_COUNTER	= 44
OAMP_PACKET_COUNTER	= 0
OLP_PACKET_COUNTER	= 0
RCY_PACKET_COUNTER	= 10
IRE_FDT_INTERFACE_CNT	= 0
IDR	
MMU_IDR_PACKET_COUNTER	= 9,780
IDR_OCB_INTERFACE_COUNTER	= 0
IQM	
ENQUEUE_PKT_CNT	= 58
DEQUEUE_PKT_CNT	= 58
DELETED_PKT_CNT	= 0
ENQ_DISCARDED_PACKET_COUNTER	= 0
EQQ	
FQP_PACKET_COUNTER	= 58
POP_UNICAST_PKT_CNT	= 58
POP_DSCRD_UC_PKT_CNT	= 0
POP_UC_BYTES_CNT	= 107,932
POP_MC_PKT_CNT	= 0
POP_DSCRD_MC_PKT_CNT	= 0
POP_MC_BYTES_CNT	= 0
EHP_UNICAST_PKT_CNT	= 58
EHP_MC_HIGH_PKT_CNT	= 0
EHP_MC_LOW_PKT_CNT	= 0
DELETED_PKT_CNT	= 0
ROP_PKT_CNT	= 159,008
ROP_DSCRD_PKT_CNT	= 0
PRP_PKT_DSCRD_TDM_CNT	= 0
PRP_SOP_DSCRD_UC_CNT	= 0
FDA	
CELLS_IN_CNT_P1	= 0
CELLS_IN_CNT_P2	= 0
CELLS_IN_CNT_P3	= 0
CELLS_IN_TDM_CNT	= 0
CELLS_IN_MESHMC_CNT	= 0
CELLS_IN_IPT_CNT	= 451
IPT	
EQQ_PKT_CNT	= 73
ENQ_PKT_CNT	= 58
FDT_PKT_CNT	= 0
CRC_ERROR_CNT	= 0
CFG_EVENT_CNT	= 50 *
CFG_BYTE_CNT	= 34,172
FDT	
IPT_DESC_CELL_COUNTER	= 0
IRE_DESC_CELL_COUNTER	= 0
TRANSMITTED_DATA_CELLS_COUNTER	= 0
FDR	
P1_CELL_IN_CNT	= 0
P2_CELL_IN_CNT	= 0
P3_CELL_IN_CNT	= 0
CELL_IN_CNT_TOTAL	= 0
JERICHO FABRIC INTERFACE	

What does it mean?

- NBI is the Network buffer and Interlaken block.
- It is part of the Network interface (NIF).
- It manages the TX and RX buffers for the interface. The H in NBIH stands for high and L in NBIL is for Low.
 - RX_TOTAL_BYTE_COUNTER - Counter for total bytes sent from NIF to IRE.
 - RX_TOTAL_PKT_COUNTER - Counter for total packets sent from NIF to IRE.
 - RX_TOTAL_DROPPED_EOPS - This counter counts the number of EOPs dropped in all the RX PMH NIF ports.
 - TX_TOTAL_BYTE_COUNTER - Counter for total bytes sent from NIF to IRE
 - TX_TOTAL_PKT_COUNTER - Counter for total packets sent from EGQ to NIF.
- Counter explanation and details see community.cisco.com

IOS XR Install cXR/eXR/XR7 Operations Simplifications

CCO and Devhub Image download and content



Installable packages

Classic XR



A Pie could be a mini, SMU or optional pkg

eXR



An RPM could be a SMU or optional pkg
The infra is in the ISO

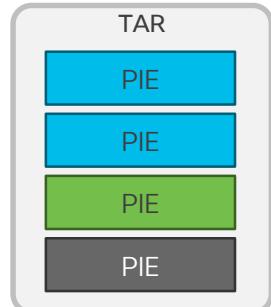
XR7



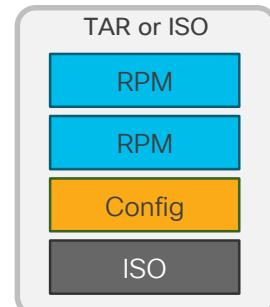
Granular packages

All packages are RPMs

TAR



TAR or ISO



ISO

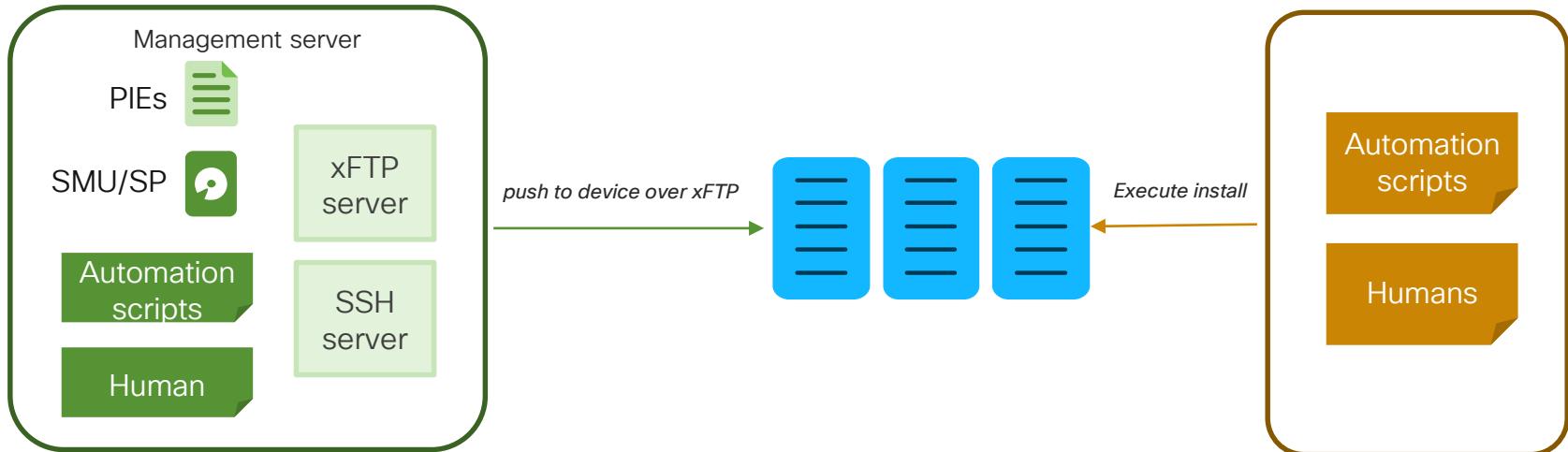


Monolithic image

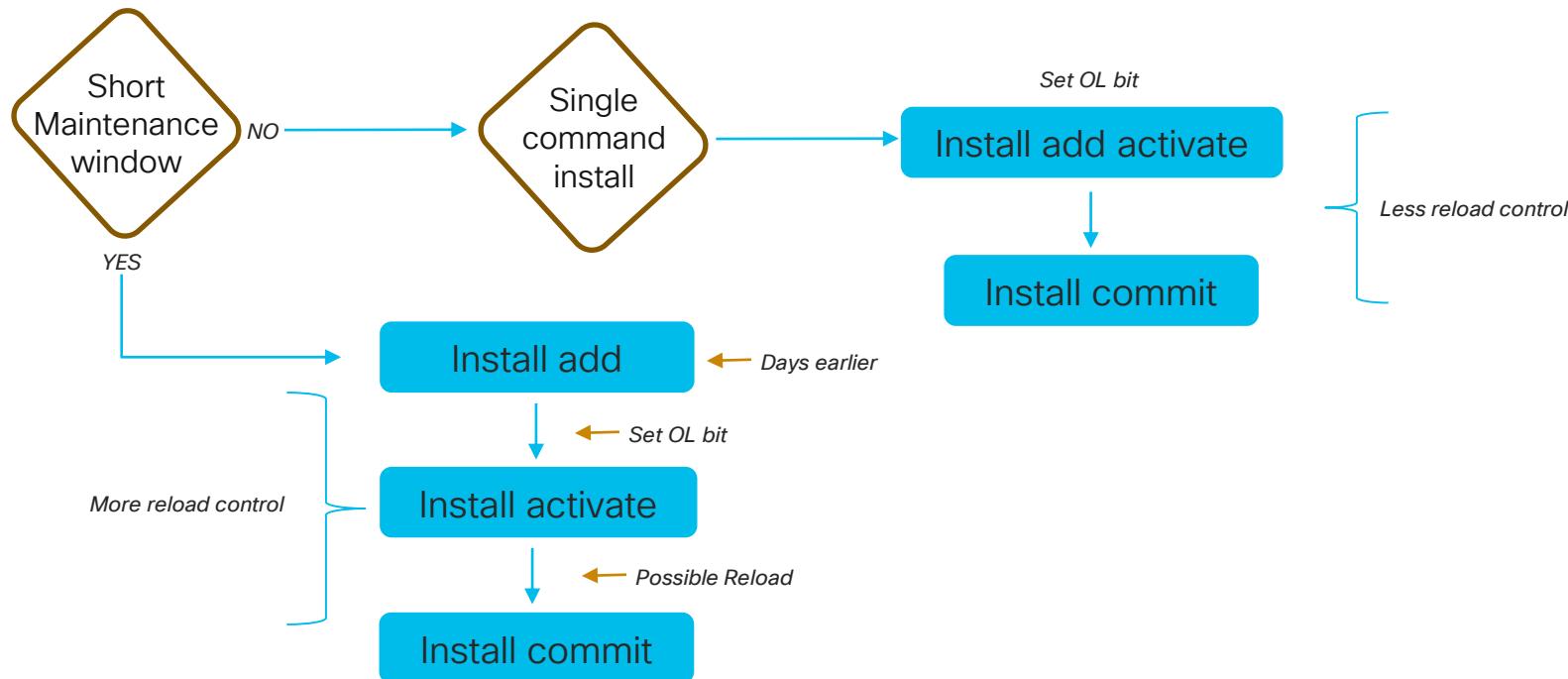
cXR	eXR	XR7
PIE	RPM	RPM
<p>Large envelope of components within an RPM, very involved package dependency, proprietary packaging, a patch introduces new binaries on top of an optional or mandatory package, the number of packages on system increase with new patches. Small set of starting point packages. Difficult to automate against.</p>	<p>Large envelope of components per RPM, portions have been modularized (Routing), open source packaging, but RPM as a container, not as a full-fledged RPM with metadata expressing dependencies, number of packages increase with more patches. Patches are cumulative. Small set of starting point RPMs. Less challenging to automate with.</p>	<p>Fix number of RPMs with accurate dependencies, completely modular, open source packaging and package management. A bug fix is a rev up of the RPM and not a small set of new binaries. RPM count on system is constant regardless of number of the patches. Large number of RPMs. DNF package management, calls for tools for package management, easy to automate against.</p>

- Install capabilities different in same if not all cases!
- Software delivery mechanism changes in XR7

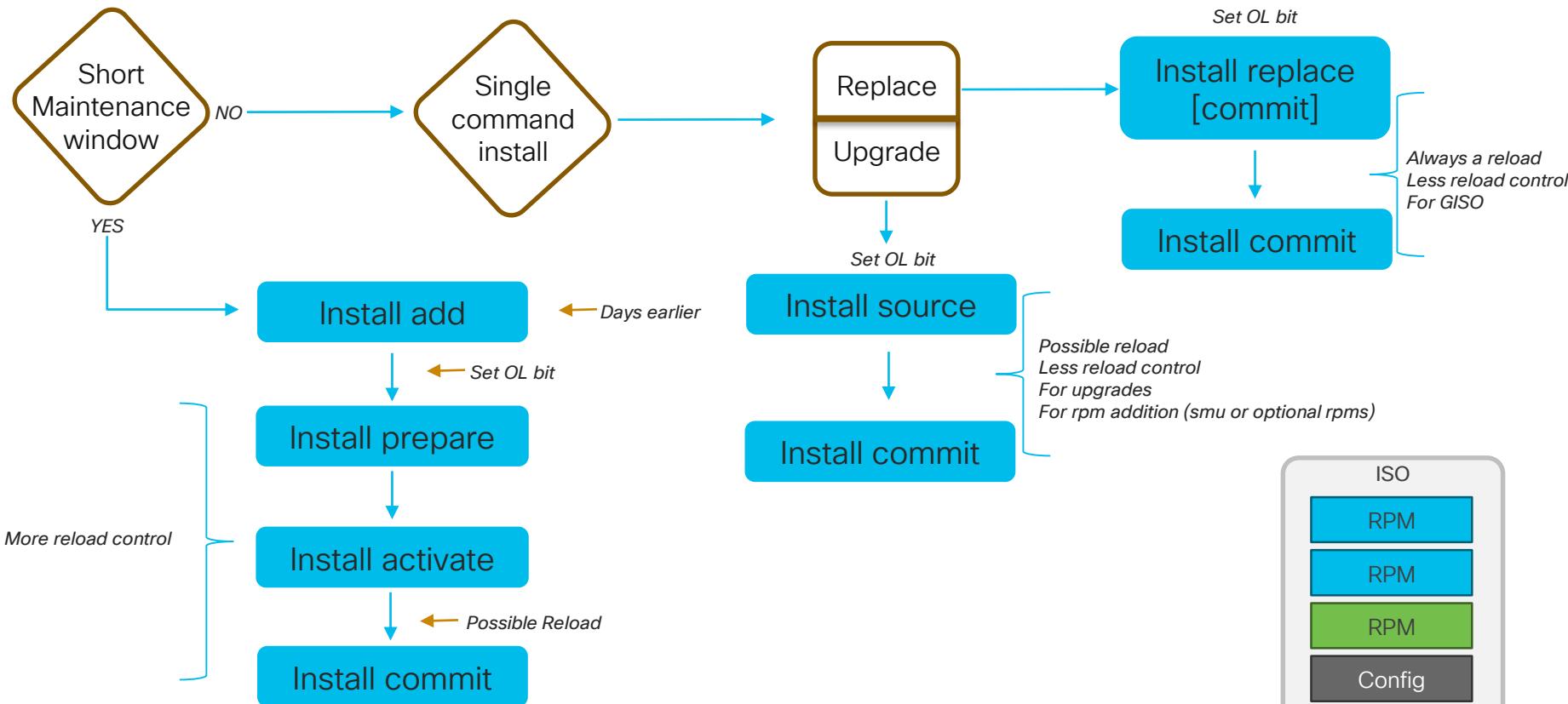
cXR Push pkg to device – Popular



cXR install workflow

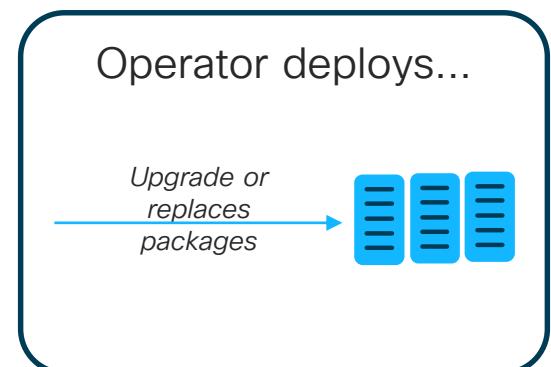
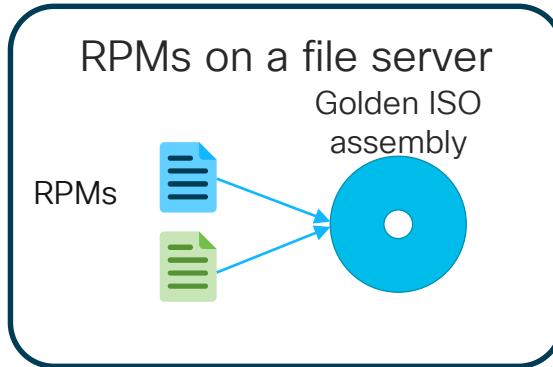
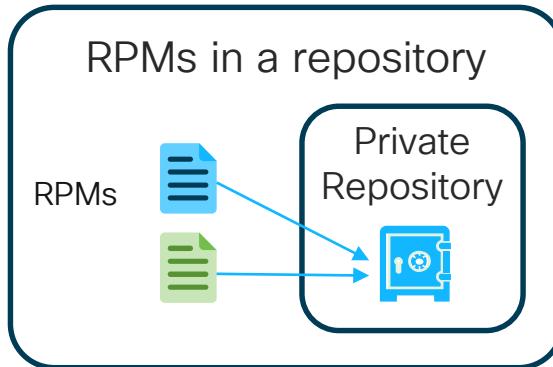
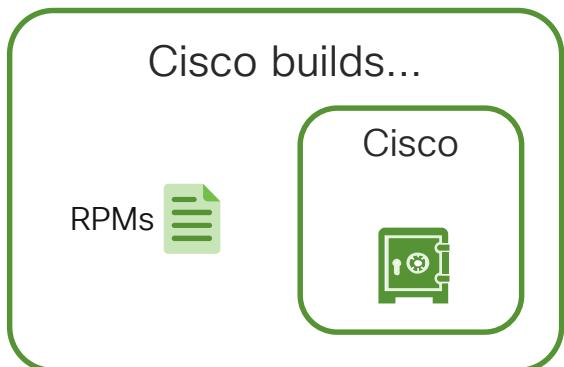


eXR install workflow

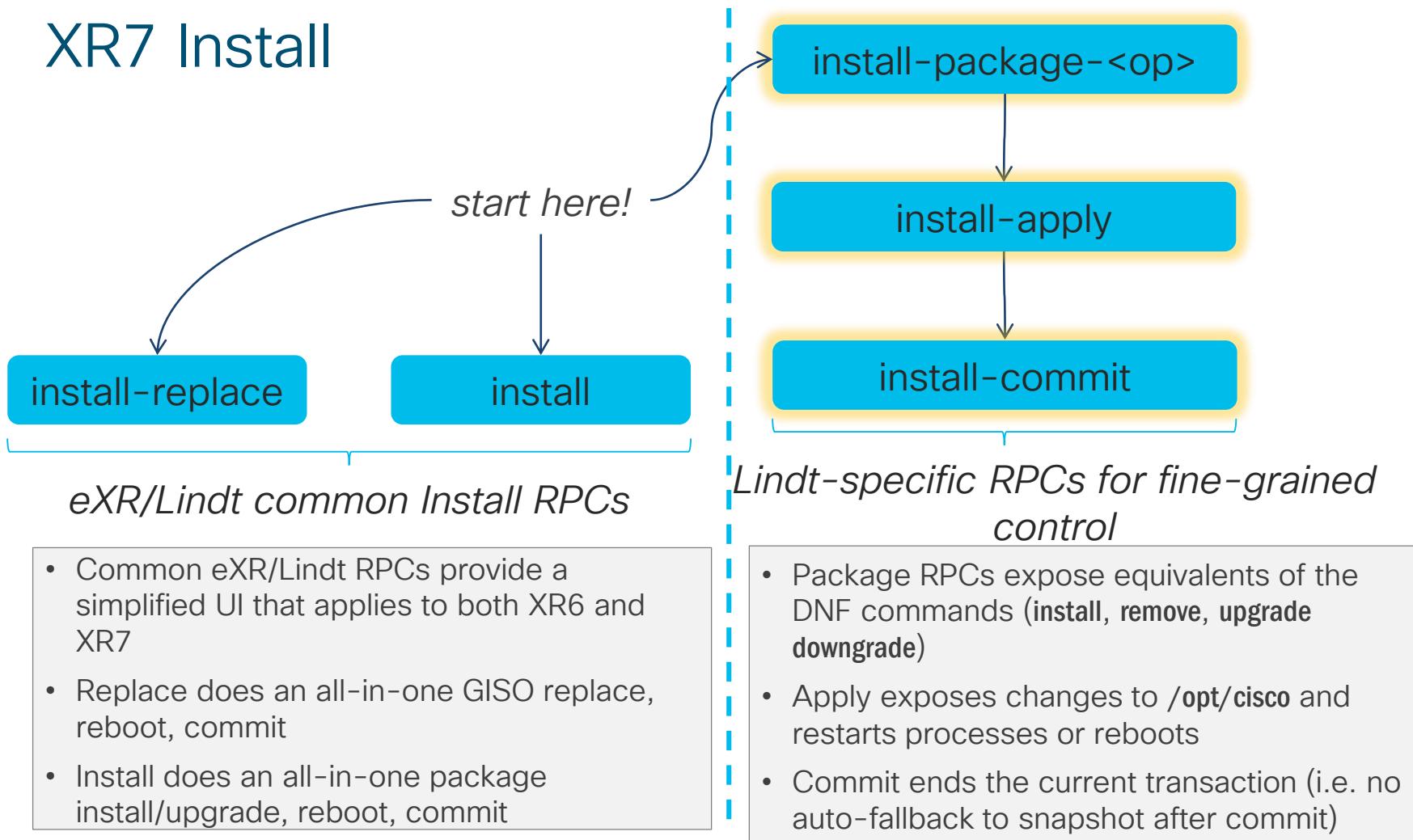


CISCO Live!

XR7 Upgrade workflow

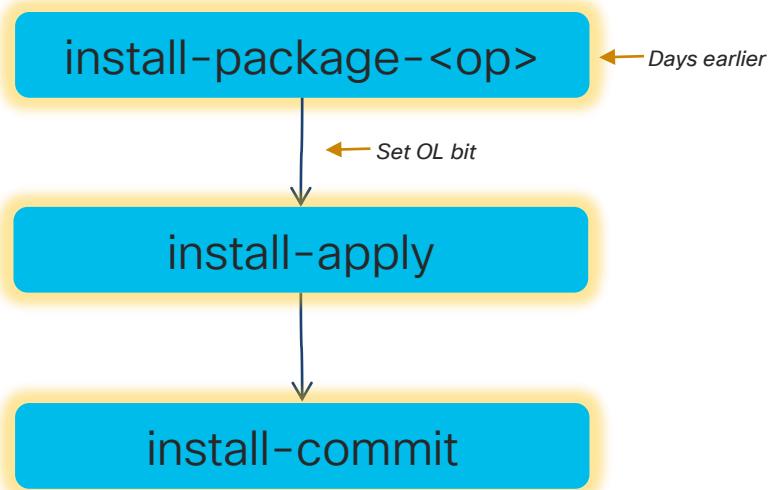
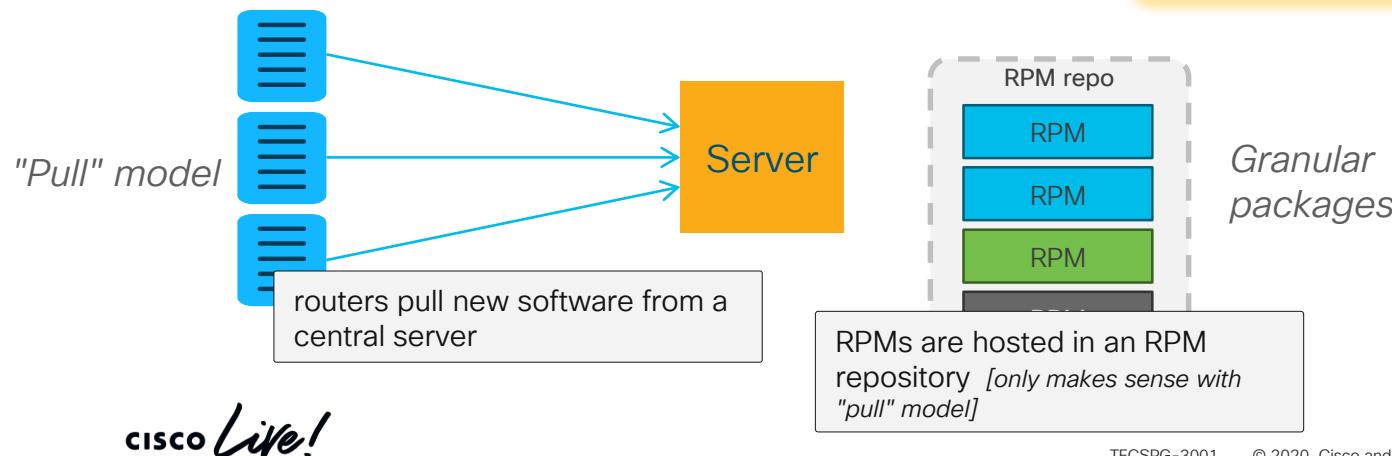


XR7 Install

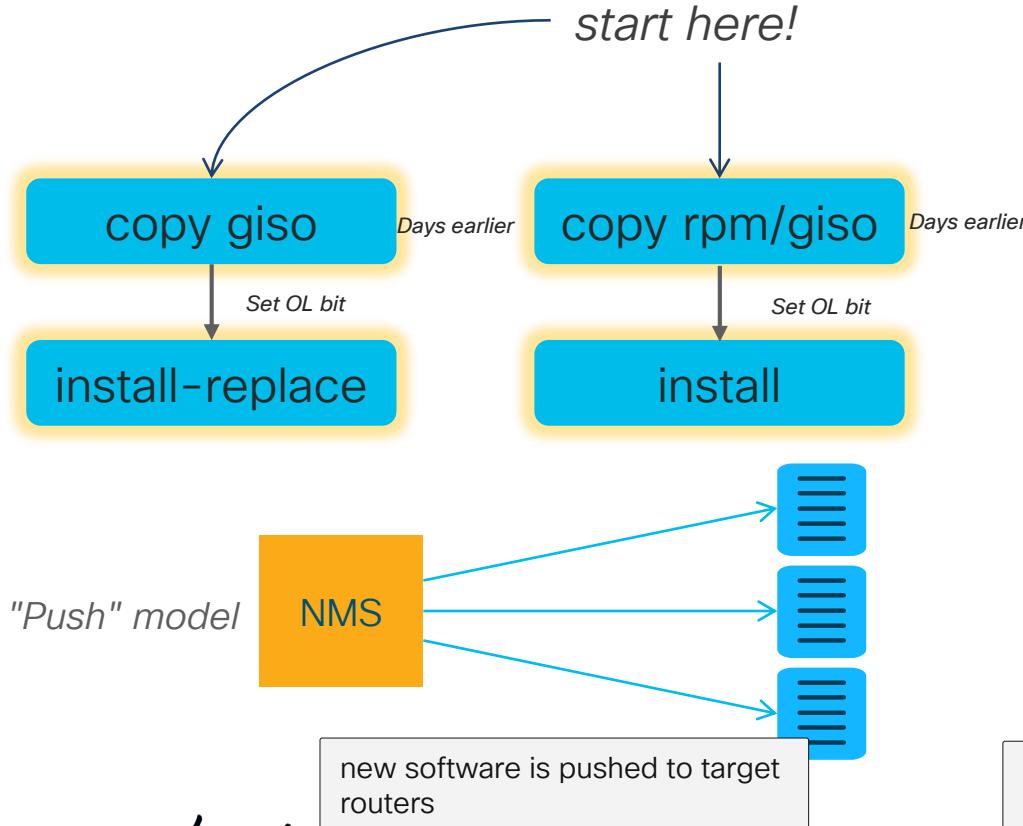


XR7 Install – Power user

- During installs, routers pull packages from private repo
- "Verbs" exactly match those supported by the underlying package manager
(e.g. "*dnf upgrade*" equivalent)

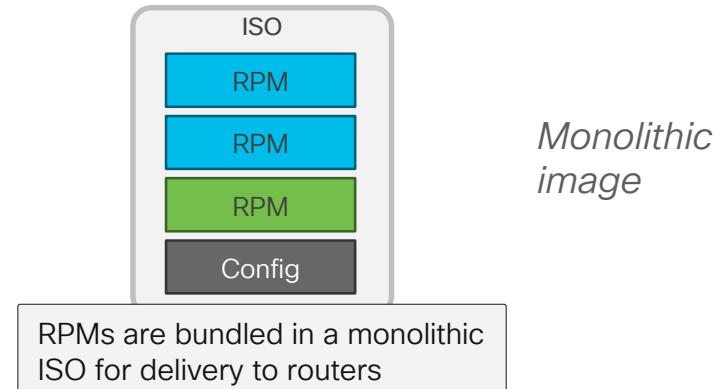


XR7 traditional workflow



- Upgrades software with a TAR, RPM or ISO
- Request a "replace" or "install" of the current software with the Golden ISO contents
- "I would like to run this software"

System computes minimal set of package changes to reach desired state



Golden ISO(GISO)

- GISO is a customised iso (iso 9660 file system), which includes desired version of **mini.iso + rpms + SMUs + router config + autorun scripts** [6.5.3]for making image management easy
- Compared to TAR format, iso has native support in Linux for boot/installation process, eg. bundling device config within an ISO
- RPMs and bug fixes can be delivered via GISO format helps to simplify during install process
- GISO is a single file with FCS image or image with bug fixes

Building a GISO

Requirement to Build a GISO

A Linux machine meets following requirements

- 'mount','rm','cp','umount','zcat','chroot','mkisofs' tools should be available
- User should have privilege to execute all of the above linux tool/cmd
- **Minimum python version 2.7**
- system should have at least 4 GB free disk space
- **gisobuild.py script (available on all eXR running routers in the location: /pkg/bin/gisobuild.py)**
 - mini iso is mandatory on file system [mandatory]
 - Desired rpms to build GISO [mandatory]
 - config file or running configuration [optional]
 - **auto-run script are optional [optional] -> 653**
 - Script helps to execute right after install automatically
- Right (rx) permissions for the rpms and the other files

Note: Kernel version of the system should be greater than 3.16 or greater than the version of kernel of cisco iso because of the initrd.img file creation during gisobuild process

- 2 files required to be downloaded from Cisco locations:

1. gisobuild.py
2. create_usb_zip



gisobuild.py

Download gisobuild.py from the cisco location:

```
usage: gisobuild.py [-h] -i BUNDLE ISO [-r RPMREPO] [-c XRCONFIG] [-s SCRIPT] [-l GISOLABEL] [-m] [-o] [-x] [-v]
```

Utility to build Golden/Custom iso. Please provide atleast repo path or config file along with bundle iso

optional arguments:

```
-h, --help                  show this help message and exit
-r RPMREPO, --repo RPMREPO          Path to RPM repository
-c XRCONFIG, --xrconfig XRCONFIG      Path to XR config file
-s SCRIPT, --script SCRIPT          Path to user executable script
-l GISOLABEL, --label GISOLABEL        Golden ISO Label
-m, --migration                 To build Migration tar only for ASR9k
-o, --optimize                  Optimize GISO by recreating and resigning initrd
-x, --x86 only                Use only x86_64 rpms even if arm is applicable for the platform
-v, --version                  Print version of this script and exit
```

required arguments:

```
-i BUNDLE_ISO, --iso BUNDLE_ISO          Path to Mini.iso/Full.iso file
```

Output of gisobuild.py

```
[root@nb-server3 giso]# /router/bin/python gisobuild.py -i ncs5500-mini-x-  
6.3.3.iso -r . -c running.cfg
```

System requirements check [PASS]

Info: Golden ISO label is not specified so defaulting to 0
Golden ISO build process starting...

Platform: ncs5500 Version: 6.3.3

XR-Config file (/auto/tftp-vista/narvenka/core/633/giso/running.cfg) will be
encapsulated in Golden ISO.

Scanning repository [/auto/tftp-vista/narvenka/core/633/giso]...

Building RPM Database...

Total 1 RPM(s) present in the repository path provided in CLI

Following XR x86_64 rpm(s) will be used for building Golden ISO:

(+) ncs5500-k9sec-4.2.0.0-r633.x86_64.rpm

...RPM compatibility check [PASS]

Building Golden ISO...

Exception why ? = [Errno 39] Directory not empty: '/auto/tftp-
vista/narvenka/core/633/giso/tmpdxF3_E/iso'
Summary

XR rpms:

ncs5500-k9sec-4.2.0.0-r633.x86_64.rpm

XR Config file:

router.cfg

...Golden ISO creation **SUCCESS**.

Golden ISO Image Location: /auto/tftp-vista/narvenka/core/633/giso/**ncs5500-**
goldenk9-x.iso-6.3.3.0 → **GISO**

Detail logs: /auto/tftp-vista/narvenka/core/633/giso/Giso_build.log-2019-03-
26:21:17:06.957192

Verifying GISO content

```
/root@nb-server3 giso]# mount -o loop ncs5500-mini-x-6.3.3.iso  
/mnt/g1  
[root@nb-server3 twitch]# tree /mnt/g1  
/mnt/g1  
└── boot  
    ├── bzImage  
    └── certs  
        ├── CertFile  
        ├── crl.der  
        └── Root_Certificate_Store.bin  
    └── grub  
        ├── device.map  
        ├── e2fs_stage1_5  
        ├── fat_stage1_5  
        ├── ffs_stage1_5  
        ├── iso9660_stage1_5  
        ├── jfs_stage1_5  
        ├── menu.lst  
        ├── menu.lst.install  
        ├── menu.lst.ucs  
        ├── menu.lst.ucs.install  
        ├── menu.lst.xboard  
        ├── minix_stage1_5  
        ├── reiserfs_stage1_5  
        ├── stage1  
        ├── stage2  
        ├── stage2_eltorito  
        ├── ufs2_stage1_5  
        └── vstafs_stage1_5  
            └── xfs_stage1_5  
        └── grub2  
            ├── bootx64.efi  
            ├── grub.cfg  
            ├── grub-usb.cfg  
            ├── grub-usb.efi  
            ├── NCS-55xx_pubkey.der  
            └── README  
            └── swims client.log  
        └── initrd.img  
        └── NCS-55xx_pubkey.der  
        └── signature.initrd.img  
        └── swims client.log  
    └── boot.catalog  
    └── giso  
        ├── info.txt  
        └── giso-summary.txt  
    └── iosxr_image.mdata.yml  
    └── iso  
        ├── info.txt  
        └── release-rpms-admin-arm.txt  
        └── release-rpms-admin-x86_64.txt  
        └── release-rpms-xr-x86_64.txt  
        └── router.cfg  
        └── xr_rpms  
            └── ncs5500-k9sec-4.2.0.0-r633.x86_64.rpm
```

GISO script removes Superseded RPMs Automatically



How to tell superseding RPMs? (6.5.3)

- Two smus (rpm) in the same component, the one with higher revision number supersedes the other one

For example:

ncs5500-dpa-3.1.0.**5**-r632.CSCvk42057.x86_64.rpm

ncs5500-dpa-3.1.0.**12**-r632.CSCvo28475.x86_64.rpm

Note: Latest 6.5.3 giso build script automatically remove superseded RPMs during build

Create USB boot

1. Create USB boot file

```
[root@nb-server3 giso]# ./create_usb_zip ncs5500 ncs5500-goldenk9-
x.iso-6.3.3.0
      adding: boot/ (stored 0%)
      adding: boot/install-image.iso (deflated 0%)
      adding: EFI/ (stored 0%)
      adding: EFI/boot/ (stored 0%)
      adding: EFI/boot/grub.cfg (deflated 66%)
      adding: EFI/boot/bootx64.efi (deflated 67%)
Zip file created - ncs5500-usb_boot.zip
[root@nb-server3 giso]#
```

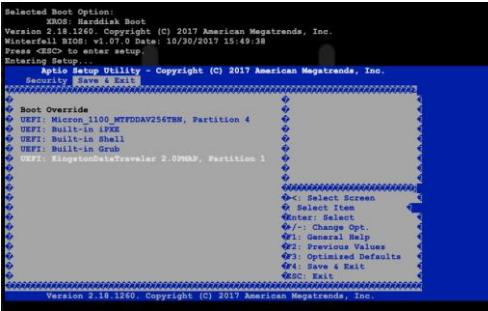
2. Insert the USB into the router this will mounted as /disk2:/ in the router, just unzip it. Now the bootable USB is ready.

```
[xr-vm_node0_RP0_CPU0:/disk2:]$unzip ncs5500-usb_boot.zip
Archive: ncs5500-usb_boot.zip
      creating: boot/
      inflating: boot/install-image.iso

      creating: EFI/
      creating: EFI/boot/
      inflating: EFI/boot/grub.cfg
      inflating: EFI/boot/bootx64.efi
```

Boot with USB GISO

1. Power cycle the router
2. Wait for <esc> prompt to get into BIOS
3. From BIOS select the USB device option to boot with



4. Now router will start booting from USB

GNU GRUB version 2.00

Press F2 to goto grub Menu..

Booting from USB..

Loading Kernel..

Kernel Secure Boot Validation Result: PASSED

GISO – 633 Vs 653

Key differences between releases

GISO features	633	653
Mini iso	✓	✓
Optional rpms	✓	✓
SMU's	✓	✓
Startup Config	✓	✓
Autorun script		✓
Image Label		✓
Supersede		✓

CSM-GISO

CSM 4.x supports GISO this will be extended with usb-boot capabilities

The screenshot shows a user interface titled "Create Golden ISO". At the top, there are four tabs: "RELEASE SOFTWARE" (selected), "TAR CONTENTS", "ISO IMAGE AND PACKAGE(S)", and "SERVER REPOSITORY". A sub-instruction below the tabs reads: "Select the ISO image and any additional packages from csm_data/repository that should be included." The main area is divided into two sections: "Available Files - showing 0" on the left and "Selected Files - showing 2" on the right. The "Available Files" section contains a "Filter" input field and a large empty rectangular area. The "Selected Files" section contains a "Filter" input field and a list with two items: "asr9k-k9sec-x64-2.1.0.1-r641.CSCvi85247.x86_64.rpm" and "asr9k-mini-x64-6.4.1.iso". Between the two sections are four control buttons: a top-right arrow, a right arrow, a left arrow, and a bottom-left arrow. At the bottom of the screen are "Previous" and "Next" navigation buttons.

iPXE Overview

- iPXE is an open source network boot firmware based gPXE/Etherboot. gPXE is an open-source PXE client firmware and bootloader derived from Etherboot. Supports HTTP/TFTP, IPv4, IPv6, iSCSI, SAN, FCoE etc..
- Requirements

Device	Infrastructure
MAC address of Mgmt Or Chassis Serial number	DHCP server HTTP/TFTP server

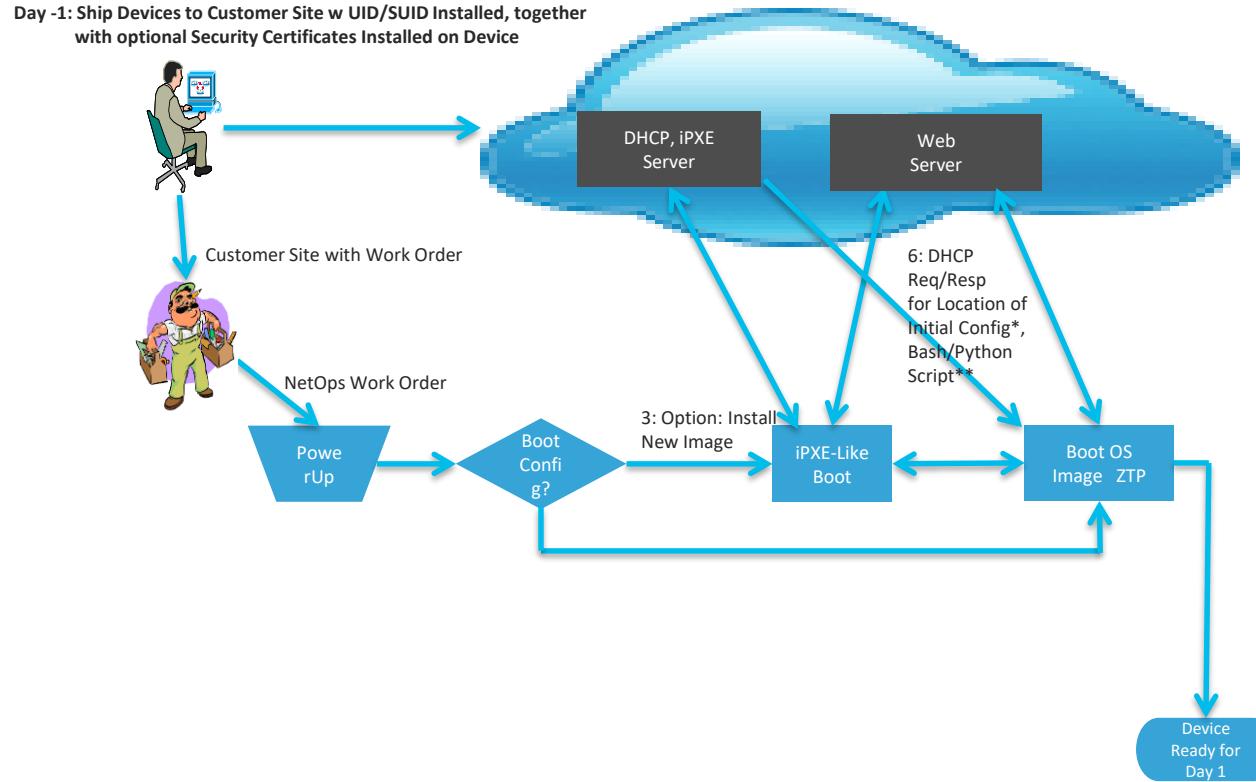
- PXE Operation

- When user selects PXE mode, then the PXE driver sends request to dhcp server with MAC address of RP as the identification.
- dhcp server looks for the entry in the dhcpcd.conf and then sends the image location.
- iPXE client on ASR9K download and install the image, eg. <http://asr9k-mini-x64.iso>

What is ZTP?

- “ZERO” touch provisioning.
- XR scripts that run on boot, invoke DHCP request
 - On initial system boot whether out of factory or after fresh install (re-bake) operation
- DHCP server returns a customer script or config file
 - Download configuration and apply
 - Download script and execute
- **How does ZTP recognize initial system boot and non-initial system boot?**
 - ZTP reads running-configuration and start ztp script if no 'username' is configured.

Zero Touch Provisioning – Device Day 0 Bootstrap



ZTP Summary of Operation

- Install new image using iPXE
 - DHCP request with use class “iPXE” and for identification send the MAC-address /Serial Number
 - DHCP reply with Image location example: <http://1.80.5.10/asr9k-mini-x64.iso>
 - iPXE client on the device download and install the image from web server
- Load device configuration
 - XR scripts that run on boot reads running-configuration and start ztp script if no 'username' is configured.
 - Invoke DHCP request with user class as “config”
 - DHCP server returns a customer script or configuration file
 - Download configuration and apply
 - Download script and execute
 - *When ZTP framework runs script, it needs to create user to gain access to XR (like CLI). So ZTP will create user called 'ztp-user' with root-!r privilege and random password*

Sample dhcpcd.conf for iPXE

```
host ASR9904-PE1 {  
    hardware ethernet 00:22:7e:0c:8c:30;  
    fixed-address 1.83.55.170;  
    option dhcp-client-identifier "FOX1739G951"; → option 61  
    if exists user-class and option user-class = "iPXE" {  
        filename "http://1.83.55.10/ankasr9k-mini-x64-6.1.1v/.iso"; - option 67  
    } else {  
        # Auto-provision request, script/config  
        filename "http://1.83.55.10/ankv/ZTP/PE1-ZTP-Script";  
        #filename "http://1.83.55.10/ankv/FOX1739G951.config";  
    }  
}
```

Sample case of Autorun Script

1. Create ssh keys for ssh access

```
#!/bin/bash  
  
export PATH=$PATH:/bin:/pkg/bin  
  
echo -ne "2048\n" | ce_cmd addkey rsa nooption general-keys
```

2. Add a customer rpm/files to specified location

Create a directory called other-rpms add the needed files in that.

```
#!/bin/bash  
export PATH=$PATH:/bin:/pkg/bin  
RPM_DIRECTORY="/disk2/other-rpms"  
echo -ne "2048\n" | ce_cmd addkey rsa nooption general-keys  
sleep 1  
if [ -d "$RPM_DIRECTORY" ] && [ "$(ls -A $RPM_DIRECTORY)" ]; then  
echo "copying the file"  
mkdir -p /harddisk:/additional_rpms/  
cp -r "$RPM_DIRECTORY"/* /harddisk:/additional_rpms/  
fi  
echo "end of testing" >> log.txt
```

Create anything wanted to be executed part of install, just add it part of a script and input during GISO build.

CSM

Cisco

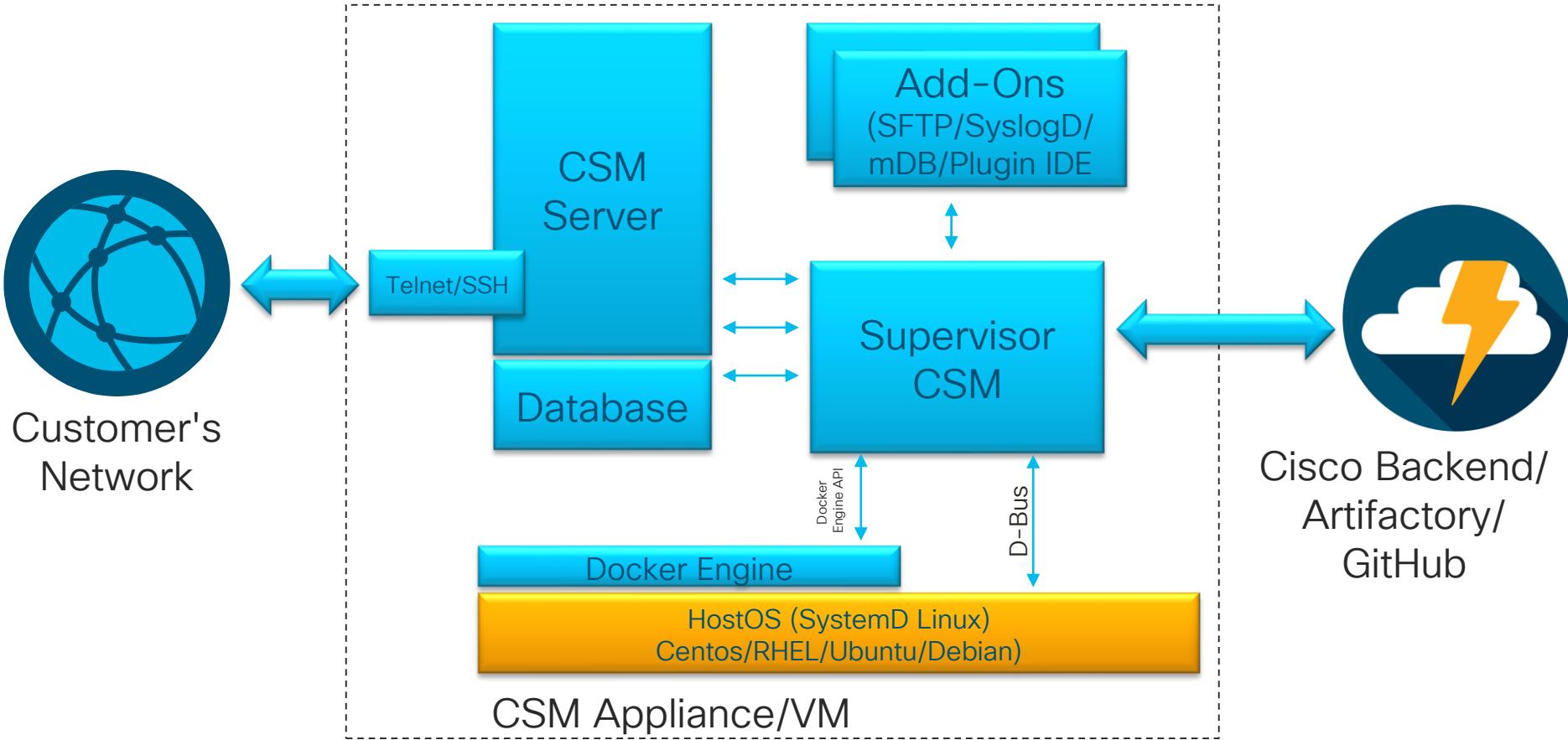
Software

Manager

What is Cisco Software Manager?

- CSM Server is a web-based, server-side automation and orchestration framework for software maintenance on Cisco Platforms.
- Provides IOS XR and IOS XR 64-bit SMUs & software upgrades across hundreds of routers in a scheduled manner through a simple point and click Web interface
- Automatically executes the customized Method of Procedure for software upgrades, including Pre-Checks and Post-Checks
- Supports the Classic XR to XR 64-bit migration

CSM 4.0 Architecture – Microservices



Key Success Factors

- Free or charge - easy to install
- Solves the software upgrade complexity
- Customer's feedback driven development (CFDD)
- Short TTM with new features and fixes. (CI/CD)
- Self-upgrade functionality.

Requirements

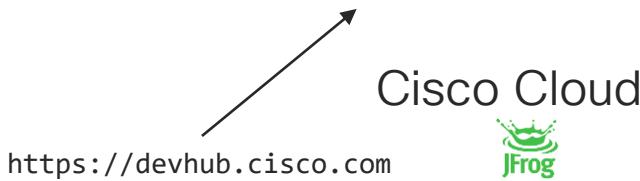
- CSM Works on most SystemD Linux distributions
- The installation procedure requires root privileges:
 - Stores the config in /etc/csm.json
 - Creates SystemD Service: csm-supervisor
- Minimal Hardware Requirements:
 - 2 CPU
 - 8 GB RAM
 - 30 GB HDD

Notable Features

- Manage Golden ISO
 - Create and manage Golden ISO images
- Create Tar File
 - Create a custom tar file from selected contents of downloaded tar files and/or other files in the internal repository.
- Custom Command Profile
 - Create profiles of CLI commands that can be chosen to run during pre-upgrade and post-upgrade install jobs. A diff of the outputs of the CLI commands can be viewed in the logs.
- Optimize Software
 - Optimize the provided software packages by determining if there are any missing pre-requisites.
- User Preferences
 - Exclude certain platforms from displaying on the CCO menu.
 - Define Cisco authentication. The authentication is needed for software download.

CSM Installation Script

```
sudo -E bash -c "$(curl -sL https://devhub.cisco.com/artifactory/software-manager-install-group/install.sh)"
```



<https://devhub.cisco.com/artifactory/software-manager-install-group/version/4/stable.json>

```
{
  "channel": "stable",
  "supervisor": "20",
  "csmserver": {
    "default": "4.0.1",
    "raspberrypi3": "4.0.1",
    "qemux86-64": "4.0.1"
  },
  "database": "1"
}
```

devhub-docker.cisco.com/csm-docker/amd64-csm-supervisor:20
devhub-docker.cisco.com/csm-docker/qemux86-64-csm-server:4.0.1
devhub-docker.cisco.com/csm-docker/amd64-csm-db:1

Can you imagine self-provisioned AWS instance?

CSM: List/Edit/Delete Hosts

List/Edit/Delete Hosts

Managed Hosts								
Region: ALL		Filter Hosts by Region ▾						
10 records per page		Search: <input type="text"/>						
Hostname	Region	Location	roles	Chassis	Platform	Software	Connection	Jump Server
9000v	SJ20-MBH			ASR-9904	ASR9K	6.1.3	telnet:1.78.28.103	SJ20-MBH
A Host	SJ20-MBH			ASR-9010	ASR9K	6.2.3	telnet:1.76.29.40 telnet:1.76.29.41 telnet:172.27.148.159:2039 telnet:172.27.148.159:2040	sj20lab-as2
A Host 2	SJ20-MBH			ASR-9904	ASR9K-X64	6.4.1	telnet:172.27.148.159:2040	Delete
A Host 3	SJ20-MBH			ASR-9904	ASR9K-X64	6.4.1	telnet:172.27.148.157 telnet:172.27.148.157 ssh:1.2.2.3 ssh:1.3.7.8	Delete
A Host 5	SJ20-MBH			ASR-9904	ASR9K-X64	6.4.1	telnet:172.27.148.157 telnet:172.27.148.157 telnet:172.27.148.159:2034 telnet:172.27.148.159:2036	Delete

CSM: Edit Current User

Edit User

Edit User

Username	johnsmith	
Privilege	Network Admin	
Authentication Method	Default	
Active	<input checked="" type="button"/> Yes <input type="button"/> No	

Information

Full Name	John Smith
Email Address	johnsmith@mycompany.com

Save Cancel

- Modify current user settings.
- To modify the password, click the Lock icon to reveal the password field. Then, make the necessary change.
- The email address specified will be used for email notification if it is enabled by the administrator during install operations.

CSM: Optimize Software

Optimize Software

The screenshot shows a software interface titled "Optimize Software". At the top, there are three buttons: "Server Repository" (with a server icon), "CCO" (with a globe icon), and "Optimize" (with a blue arrow). Below these buttons is a text input field containing the following list of software packages:

- asr9k-px-5.3.3.CSCvc38357.pie
- asr9k-px-5.3.3.CSCuz89632.pie
- asr9k-px-5.3.3.CSCux57115.pie

Below the input field, there is a note: "Paste software packages (SMUs/SPs) below or select them from above sources." Underneath the packages, there is a message: "Click 'Accept' to include all missing pre-requisites and remove all superseded packages, if any. Click 'Accept (Remove Unrecognized)' to remove entries marked as Unrecognized, if any. Entries classified as SMU/SP/Package will be included automatically." At the bottom of the interface are three buttons: "Accept" (green), "Accept (Remove Unrecognized)" (blue), and "Cancel".

- Provide a software package list (SMUs or SPs) using the following methods:
 - Enter them manually.
 - Select from a server repository.
 - Select from CCO.
- The SMU or SP names provided can be in different formats so long as the platform and release are identifiable.
- Click Optimize to optimize the list.
- If there are any missing pre-requisites or superseded packages, an information dialog will be displayed.
- To save the optimized list and use it for the Conformance report, click the Save as Software Profile button.

CSM: User Preferences

User Preferences

- Excludes certain platforms and releases from displaying on the CCO menu.
- Defines the CCO authentication information.

User Preferences

Filter Platforms

To exclude certain platforms from displaying on the CCO menu, check the checkbox next to the platform and release.

Search:

<input checked="" type="checkbox"/>	ASR9K-PX	6.0.2
<input checked="" type="checkbox"/>	ASR9K-PX	6.0.1
<input type="checkbox"/>	ASR9K-PX	5.3.31
<input type="checkbox"/>	ASR9K-PX	5.3.4
<input type="checkbox"/>	ASR9K-PX	5.3.3
<input type="checkbox"/>	ASR9K-PX	5.3.2
<input type="checkbox"/>	ASR9K-PX	5.3.1

CCO Authentication

Authentication information is used when downloading files from CCO.

Username: Save

Password: Validate

CSM: Manage Golden ISO

Manage Golden ISO

- Create a Golden ISO image file sourced from Release Software tar files and SMU files. These files must reside in the CSM Server internal repository.

The screenshot shows two related web pages for managing Golden ISO images.

Golden ISO Dashboard: This page displays a summary of job status: Scheduled (0), In Progress (0), Failed (1), and Completed (0). It includes a search bar, a dropdown for records per page (set to 10), and a table header with columns: Image Name, Scheduled Time, Packages, Server Repository, Created By, and Action. A message indicates "No data available in table". Navigation buttons for Previous and Next are present.

Action Menu: A dropdown menu titled "Action" is open, showing options: Create New Golden ISO, Delete All Scheduled Jobs, and Delete All Failed Jobs.

CSM Golden ISO Repository: This page provides details about a newly created Golden ISO: "(Created Golden ISO is stored in "/Users/alextang/PyCharm/csm_data/repository/golden_iso" and copied to the designated server repository)". It includes a "Return to Top" link, a search bar, and a table header with columns: Image Name, Size (Bytes), Packages, Downloaded Time, and Action. A message indicates "No data available in table". Navigation buttons for Previous and Next are present.

Create Golden ISO - Step 1

- Select a Release Software tar file from the CSM Server internal repository.

Create Golden ISO

RELEASE SOFTWARE TAR CONTENTS ISO IMAGE AND PACKAGE(S) SERVER REPOSITORY

Select from the available Release Software files in csm_data/repository to extract the feature RPMs.

Available Tar-Files - showing 2

Filter

ASR9K-x64-iosxr-px-6.4.2.tar	ASR9K-x64-iosxr-px-k9-6.4.2.tar
------------------------------	---------------------------------

►

►

◀

◀ ►

Selected Tar-Files - showing 0

Filter

--

Create Golden ISO - Step 2

- From the selected Release Software tar file, select the feature RPMs to be included in the Golden ISO.

Create Golden ISO

The following are the contents from the selected tar files. Select which feature RPMs should be included.

100 records per page Search:

<input type="checkbox"/>	File Name	Source Tar
<input checked="" type="checkbox"/>	asr9k-9000v-nV-x64-1.0.0.0-r642.x86_64.rpm	ASR9K-x64-iosxr-px-k9-6.4.2.tar
<input checked="" type="checkbox"/>	asr9k-isis-x64-1.1.0.0-r642.x86_64.rpm	ASR9K-x64-iosxr-px-k9-6.4.2.tar
<input type="checkbox"/>	asr9k-m2m-x64-2.0.0.0-r642.x86_64.rpm	ASR9K-x64-iosxr-px-k9-6.4.2.tar
<input checked="" type="checkbox"/>	asr9k-mpls-te-rsvp-x64-2.2.0.0-r642.x86_64.rpm	ASR9K-x64-iosxr-px-k9-6.4.2.tar
<input checked="" type="checkbox"/>	asr9k-ospf-x64-1.0.0.0-r642.x86_64.rpm	ASR9K-x64-iosxr-px-k9-6.4.2.tar
<input type="checkbox"/>	asr9k-bng-x64-1.0.0.0-r642.x86_64.rpm	ASR9K-x64-iosxr-px-k9-6.4.2.tar
<input type="checkbox"/>	asr9k-k9sec-x64-2.0.0.0-r642.x86_64.rpm	ASR9K-x64-iosxr-px-k9-6.4.2.tar

Showing 1 to 14 of 14 entries

1

Create Golden ISO - Step 3

- Select the mini ISO file and other SMU tar files to be included in the Golden ISO.

Create Golden ISO

RELEASE SOFTWARE TAR CONTENTS **ISO IMAGE AND PACKAGE(S)** SERVER REPOSITORY

Select the ISO image and any additional SMUs or RPMs from csm_data/repository that should be included.

Available Files - showing 5

6.4.2

asr9k-mini-x64-6.4.2.iso
asr9k-x64-6.4.2.CSCvh98967.tar
asr9k-x64-6.4.2.CSCvm79236.tar
asr9k-x64-6.4.2.CSCvn51436.tar
asr9k-x64-6.4.2.CSCvo63987.tar

Filter

asr9k-mini-x64-6.4.2.iso
asr9k-x64-6.4.2.CSCvh98967.tar
asr9k-x64-6.4.2.CSCvm79236.tar
asr9k-x64-6.4.2.CSCvn51436.tar
asr9k-x64-6.4.2.CSCvo63987.tar

Selected Files - showing 0

Filter

< >

< >

< >

Create Golden ISO - Step 4

- Specify the Golden ISO file name and select a server repository to store the generated file. Once the building process is started, the progress information will be displayed on the Managed Golden ISO Dashboard.

Create Golden ISO

The screenshot shows a user interface for creating a Golden ISO. At the top, there are four tabs: RELEASE SOFTWARE, TAR CONTENTS, ISO IMAGE AND PACKAGE(S), and SERVER REPOSITORY. The SERVER REPOSITORY tab is highlighted in blue. Below the tabs, a message says "Select the destination server repository for uploading the new golden ISO." There are four input fields:

- New Golden ISO Name:** A text input field containing "asr9k-6.4.2".
- For Migration from ASR9K to ASR9K-X64:** A button group with "Yes" (white) and "No" (blue).
- Server Repository:** A dropdown menu currently set to "TFTP Server".
- Server Directory:** An empty input field with a green upload icon and a red delete icon.

New Golden ISO Name	asr9k-6.4.2
For Migration from ASR9K to ASR9K-X64	Yes No
Server Repository	TFTP Server
Server Directory	<input type="text"/> ↑ ✖

Upload to other server repository

- Once the Golden ISO is created, it will be stored in the CSM internal repository as well as uploaded to the designated server repository. You may also choose to upload the same Golden ISO to other server repository by selecting the Upload action on the Golden ISO Dashboard.

CSM Golden ISO Repository (Created Golden ISO is stored in "/home/cisco/giso/csm_data/repository/golden_iso" and copied to the designated server repository)

Return to Top

10 records per page

Search:

Image Name	Size (Bytes)	Packages	Downloaded Time	Action
asr9k-golden-x64-6.4.1.v5.iso	1585752064	ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-9000v-nV-x64-1.0.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-bng-x64-1.0.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-eigrp-x64-1.0.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-isis-x64-1.1.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-k9sec-x64-2.1.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-li-x64-1.1.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-m2m-x64-2.0.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-mcast-x64-2.0.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-mgbl-x64-2.0.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-mlps-te-rsvp-x64-2.1.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-mlps-x64-2.0.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-optic-x64-1.0.0.0-r641.x86_64.rpm ASR9K-x64-iosxr-px-k9-6.4.1.tar/asr9k-ospf-x64-1.0.0.0-r641.x86_64.rpm asr9k-mini-x64-6.4.1.iso asr9k-x64-6.4.1.CSCvh98967.tar asr9k-x64-6.4.1.CSCvi85247.tar asr9k-x64-6.4.1.CSCvm65992.tar	05/07/2019 11:52 PM	<button>Select</button> <button>Upload</button> <button>Delete</button>

CSM: Create Tar File

Create Tar File

- Create a new tar file sourced from Release Software tar files or SMU files. These files must reside in the CSM Server internal repository.
- For software upgrade, it may be desirable to select only applicable software packages from the Release Software tar file instead of using the entire tar file. This will reduce the download time and the required storage capability on the device. Additional files (SMUs) can also be included.
- For SMU installation, it may be desirable to group all SMU files into a tar file as some software releases have limitation on the number of SMUs allowed during installation. To prepare a tar file for this situation, skip the first two screens of the Create Tar File Wizard.

Create Tar File – Step One: Select Tar Files

Create Tar File

RELEASE SOFTWARE TAR CONTENTS ADDITIONAL FILES SERVER REPOSITORY

Select from the available Release Software files in csm_data/repository to extract their contents.

Available Tar-Files - showing 3

Filter

ASR9K-iosxr-px-5.2.4-turboboot.tar
ASR9K-iosxr-px-5.3.4-turboboot.tar
ASR9K-iosxr-px-k9-6.1.2.tar

Selected Tar-Files - showing 1

Filter

ASR9K-iosxr-px-5.3.2.tar

Next

- Use the wizard to choose any (or none) of the tar files currently available in the internal repository (i.e. csm_data/repository).
- On the next screen, the contents of the chosen tar files will be displayed for further selection.

Create Tar File – Step Two: Choose TAR Contents

Create Tar File

RELEASE SOFTWARE TAR CONTENTS ADDITIONAL FILES SERVER REPOSITORY

The following are the contents from the selected tar files. Select which files should be included.

100 records per page Search:

<input type="checkbox"/> File Name	Source Tar
<input checked="" type="checkbox"/> asr9k-bng-px.pie-5.3.2	ASR9K-iosxr-px-5.3.2.tar
<input checked="" type="checkbox"/> asr9k-mpls-px.pie-5.3.2	ASR9K-iosxr-px-5.3.2.tar
<input type="checkbox"/> asr9k-doc-px.pie-5.3.2	ASR9K-iosxr-px-5.3.2.tar
<input type="checkbox"/> asr9k-optic-px.pie-5.3.2	ASR9K-iosxr-px-5.3.2.tar
<input type="checkbox"/> asr9k-mini-px.pie-5.3.2	ASR9K-iosxr-px-5.3.2.tar

Showing 1 to 5 of 5 entries

Previous Next

1

- This screen shows all the files contained by the tar files selected in Step One on the previous slide.
- Select the desirable packages from the selected tar files to be included in the new tar file.

Create Tar File – Step Three: Additional Files

Create Tar File

RELEASE SOFTWARE TAR CONTENTS **ADDITIONAL FILES** SERVER REPOSITORY

Select any additional files from csm_data/repository that should be included.

Available Files - showing 81

Filter

asr9k-px-5.1.0.CSCut52232.pie
asr9k-px-5.3.0.CSCuz09712.pie
asr9k-px-5.3.1.CSCuu91413.pie
asr9k-px-5.3.1.CSCuw01739.pie
asr9k-px-5.3.1.CSCuw46460.pie
asr9k-px-5.3.1.CSCuw50964.pie
asr9k-px-5.3.1.CSCuw65671.pie
asr9k-px-5.3.1.CSCuw94704.pie
asr9k-px-5.3.1.CSCux01983.pie
asr9k-px-5.3.2.CSCuu83507.pie

Selected Files - showing 3

Filter

asr9k-px-5.3.2.CSCur44662.pie
asr9k-px-5.3.2.CSCur86301.pie
asr9k-px-5.3.2.CSCus67318.pie

Next

The screenshot shows the 'Create Tar File' interface with the 'ADDITIONAL FILES' tab selected. On the left, under 'Available Files', there is a list of 81 files, mostly starting with 'asr9k-px'. On the right, under 'Selected Files', there are three files listed: 'asr9k-px-5.3.2.CSCur44662.pie', 'asr9k-px-5.3.2.CSCur86301.pie', and 'asr9k-px-5.3.2.CSCus67318.pie'. Between the two lists are four navigation buttons: a top-left arrow, a top-right arrow, a bottom-left arrow, and a bottom-right arrow. At the bottom of the screen are 'Previous' and 'Next' buttons.

- This screen shows all of the SMU files that are available in the internal CSM repository.
- Select SMU files to be included in the tar file.

Create Tar File – Step Four: Server Repository

Create Tar File

RELEASE SOFTWARE TAR CONTENTS ADDITIONAL FILES **SERVER REPOSITORY**

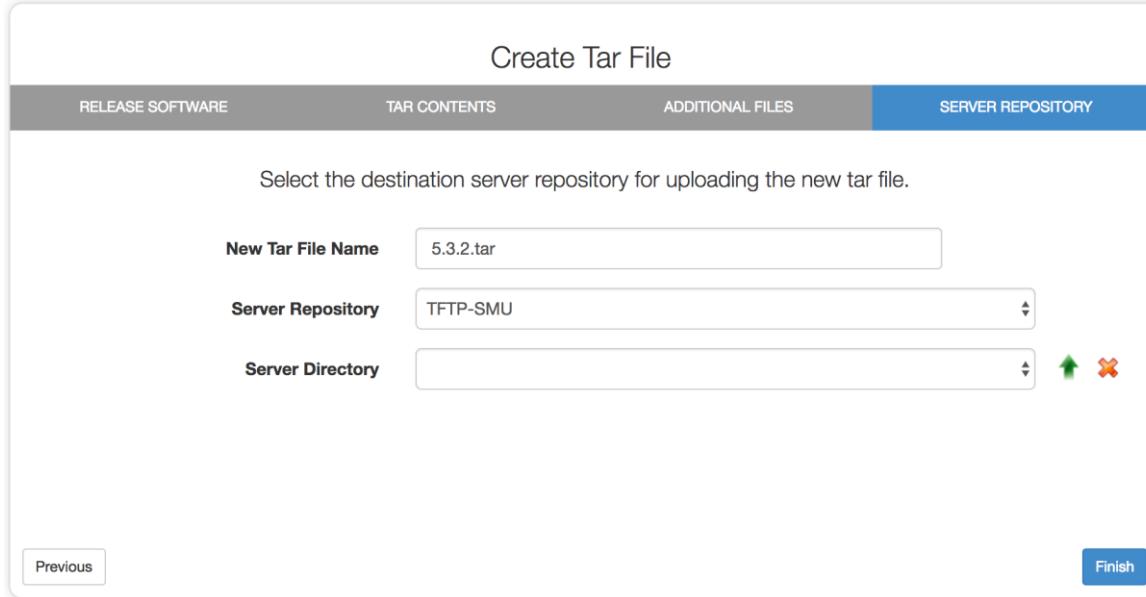
Select the destination server repository for uploading the new tar file.

New Tar File Name:

Server Repository: 

Server Directory:  

[Previous](#) [Finish](#)



- Specify the new tar file name.
- Select the Server Repository where the new tar file should be placed.
- Select an optional Server Directory if desired.

CSM: Custom Command Profile

Custom Command Profile

- Custom Command Profiles can be selected when doing a Pre-Upgrade or Post-Upgrade install job.
- The CLI commands listed in the chosen profile will be run on the host in addition to the default commands (e.g., show running config).
- If both Pre-Upgrade and Post-Upgrade are run, a diff will be calculated and logged.
- Custom Command Profiles can be exported and imported, making them easy to copy and share. Just click the Export button, select the profiles to export, and click ‘Export’. To import, click ‘Import’, select the previously exported .json file, and click ‘Import’.

Custom Command Profiles		Create ▾	Import	Export
10	records per page	Search:		
Profile Name	CLI Commands	Created By	Action	
<hr/>				
Core Device Profile	 show interfaces summary show ipv6 interface summary show ospf neighbor show ospfv3 neighbor show isis adjacency or show isis neighbors	root	Delete	
<hr/>				
Showing 1 to 1 of 1 entries		← Previous	1	Next →

Create a Custom Command Profile

Create Command Profile

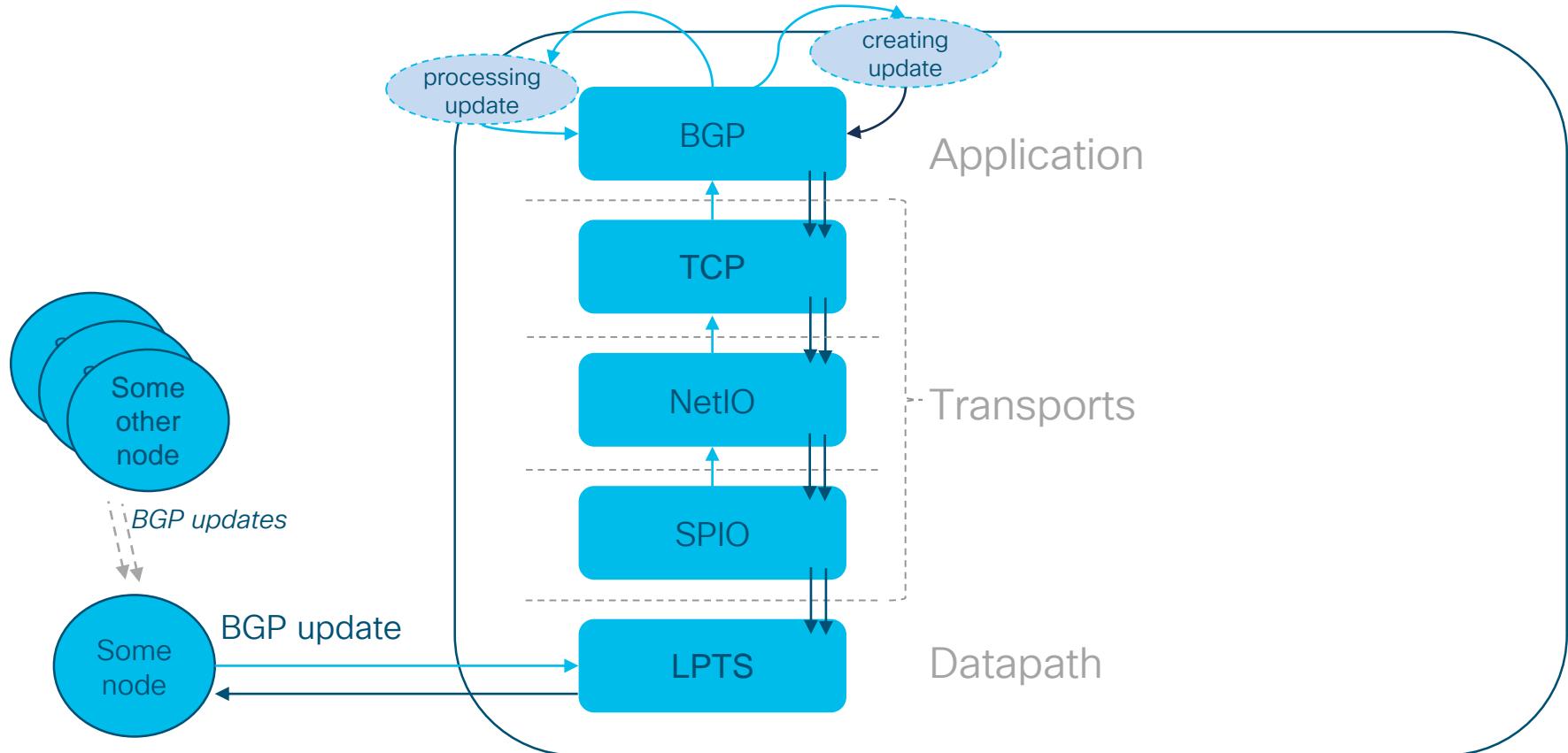
Profile Name	<input type="text" value="Core Device Profile"/>
CLI Commands	<pre>show interfaces summary show ipv6 interface summary show ospf neighbor show ospfv3 neighbor</pre>

Save **Cancel**

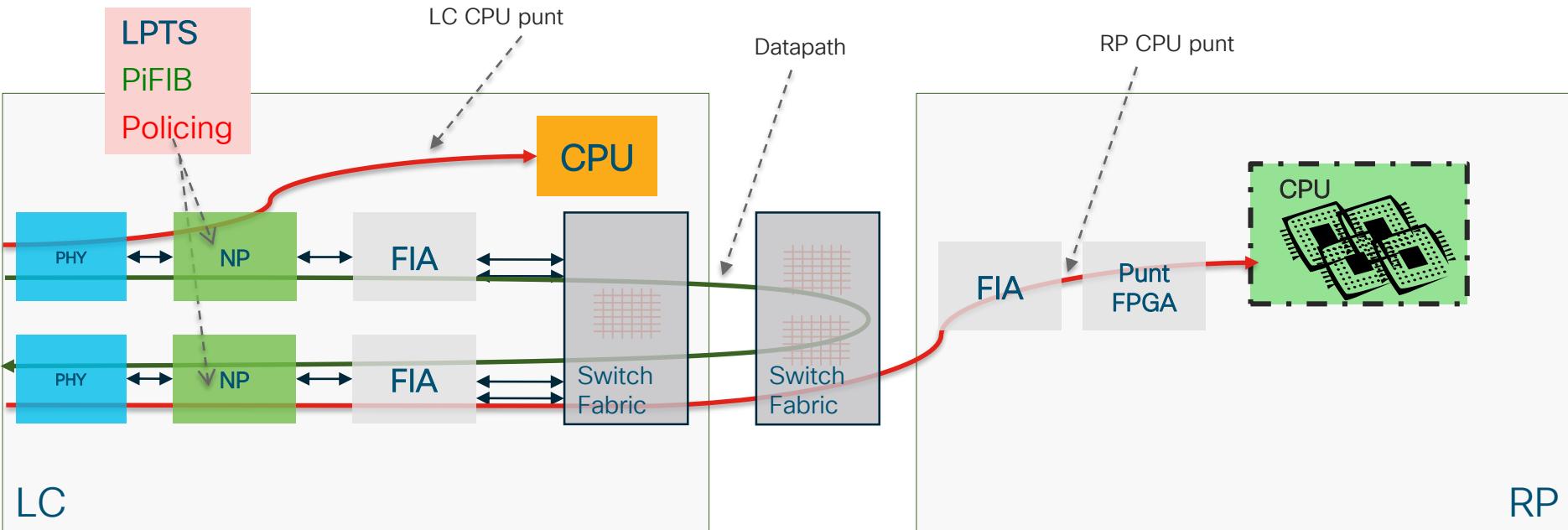
- From the Create tab, click Custom Command Profile.
- Specify the Profile Name.
- Specify the CLI commands to be executed during the Pre/Post-Upgrade install action.
- Click Save to create the custom command profile.

ASR9000 Punt/Inject Path

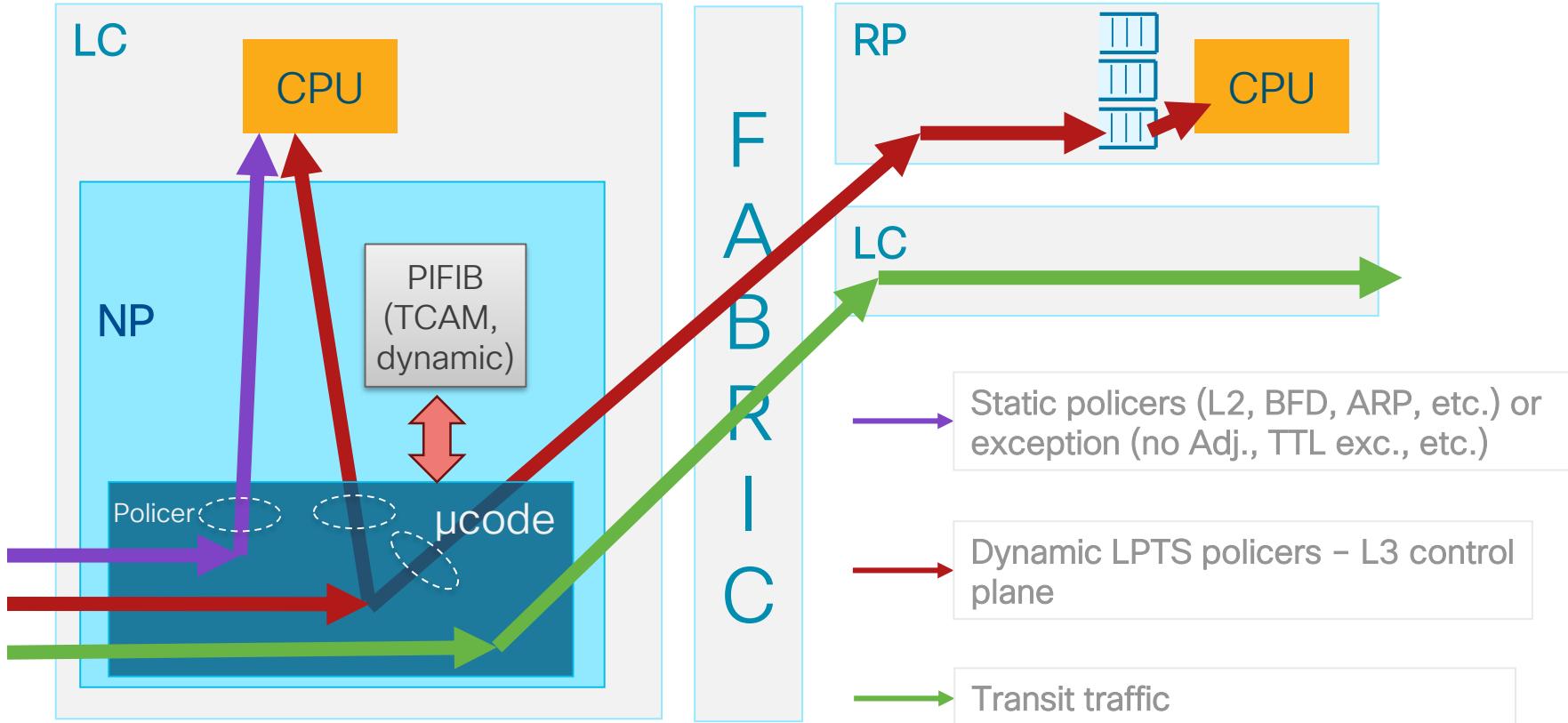
Order of Punt/Inject Operations w/ Packet Flow



Where is LPTS Executed?



Order of Punt/Inject Operations w/ Packet Flow



HW LPTS

Order of Punt/Inject Operations w/ Packet Flow

- Ingress NPU in the LC performs packet lookup using the HW FIB to determine how to switch the packet.
- If the incoming packet is part of transit traffic, they will be switched by the LC HW and sent to the egress LC through the fabric
- If the incoming packet is of specific L2/L3 type such as CDP, ARP, LACP PDU, BFD, CFM/OAM etc FIB will punt them to LC CPU for further processing. Also transit traffic to be forwarded, but frag required Packets with DF bit set packets, IP options packet, packets with RA, transit traffic dropped by ACL etc will be punted to LC CPU
- If FIB lookup determines that this is a “for-us” control/management plane packet, then further lookup has to be performed on the pre-IFIB table in the HW/TCAM to match it against a flow entry, perform policing on the packet stream, and ascertain the node/element and application to deliver
- For some of the “for-us” control packets, which needs to be delivered locally, requiring special handling such as ICMP echo, TTL expired packets, HW Pre-IFIB look-up will punt the packets to LC CPU
- “for us” control packets destined to the RP.
- IFIB slice lookup on a local node will provide transport and the associated application/server processes the packet needs to be delivered

Dynamic LPTS Entries - TCP Example

```
router bgp 65101
!
neighbor 192.85.8.11
!
neighbor 192.85.9.11
```

```
RP/0/RSP0/CPU0:ios#sh bgp neighbor | i "^[A-Z]|^ [A-Z]|host:"
Thu Jan 9 17:05:50.733 UTC
BGP neighbor is 192.85.8.11
    Remote AS 65101, local AS 65101, internal link
    Remote router ID 192.85.8.11
    For Address Family: IPv4 Unicast
        Local host: 192.85.8.1, Local port: 33726, IF Handle: 0x00000000
        Foreign host: 192.85.8.11, Foreign port: 179
BGP neighbor is 192.85.9.11
    Remote AS 65101, local AS 65101, internal link
    Remote router ID 192.85.9.11
    For Address Family: IPv4 Unicast
        Local host: 192.85.9.1, Local port: 22408, IF Handle: 0x00000000
        Foreign host: 192.85.9.11, Foreign port: 179
RP/0/RSP0/CPU0:ios#
```

```
RP/0/RSP0/CPU0:ios#sh tcp brief | i "179"
Thu Jan 9 16:10:18.108 UTC
0x00007fae08001308 0x60000000      0      0  ::::179          ::::0          LISTEN
0x00007fae1c00c468 0x00000000      0      0  ::::179          ::::0          LISTEN
0x00007fae14002868 0x60000000      0      0  192.85.9.1:22408  192.85.9.11:179  ESTAB
0x00007fae1c007d58 0x60000000  0      0  192.85.8.1:33726  192.85.8.11:179  ESTAB
0x00007fae08001028 0x60000000      0      0  0.0.0.0:179       0.0.0.0:0       LISTEN
0x00007fae1c008ef8 0x00000000      0      0  0.0.0.0:179       0.0.0.0:0       LISTEN
RP/0/RSP0/CPU0:ios#
```

TCP Extended Filter

```
router bgp 65101
bgp router-id 10.255.255.104
!
neighbor 192.85.8.11
!
neighbor 192.85.9.11
```

```
RP/0/RSP0/CPU0:ios#sh tcp extended-filters location 0/RSP0/CPU0
```

```
<..>
```

```
-----  
JID: 1087
```

```
Family: 2
```

```
VRF: 0x60000000
```

```
PCB: 0x7fae08001028
```

```
L4proto: 6
```

```
Lport: 179
```

```
Fport: 0
```

```
Laddr: 0.0.0.0
```

```
Faddr: 0.0.0.0
```

```
ICMP error filter mask: 0x12
```

```
LPTS options: 0x3 / 0x1 / 0x14 RX_FILTER / BOUND / r_ip l_ip
```

```
Flow Type: n/s
```

```
Filters:
```

Interface	ptype	Local	LPort	Remote	RPort	DUMMY	FLAGS	Action	Prio	TTL	FType
-----------	-------	-------	-------	--------	-------	-------	-------	--------	------	-----	-------

n/s	n/s	192.85.8.1/32	n/s	192.85.8.11/32	n/s	n/s	0 0 0	n/s	n/s	
-----	-----	---------------	-----	----------------	-----	-----	-------	-----	-----	--

n/s	n/s	192.85.9.1/32	n/s	192.85.9.11/32	n/s	n/s	0 0 0	n/s	n/s	
-----	-----	---------------	-----	----------------	-----	-----	-------	-----	-----	--

```
RP/0/RSP0/CPU0:ios#sh proc bgp | i Job
Job Id: 1087
```

```
RP/0/RSP0/CPU0:ios#
```

Dynamic LPTS Entries - Software

```
RP/0/RSP0/CPU0:ios#sh lpts bindings client-id tcp remote-address 192.85.8.11
Thu Jan  9 16:32:55.358 UTC
```

```
-----
Location      : 0/RSP0/CPU0
Client ID    : TCP
Cookie        : 0x080001028
Clnt Flags   : REUSEPORT
Layer 3       : IPV4
Layer 4       : TCP 6
VRF-ID        : default (0x60000000)
Local Addr   : 192.85.8.1
Remote Addr: 192.85.8.11
Local Port : 179
Remote Port: any
Filters       : Type / Intf / Pkt Type / Source Addr / Location
  xOPx / none
-----
```

```
Location      : 0/RSP0/CPU0
Client ID    : TCP
Cookie        : 0x1c007d58
Clnt Flags   : REUSEPORT
Layer 3       : IPV4
Layer 4       : TCP 6
VRF-ID        : default (0x60000000)
Local Addr   : 192.85.8.1
Remote Addr: 192.85.8.11
Local Port : 33726
Remote Port: 179
Filters       : Type / Intf / Pkt Type / Source Addr / Location
  xOPx / none
```

```
router bgp 65101
!
neighbor 192.85.8.11
!
neighbor 192.85.9.11
```

Dynamic LPTS Entries- HW - Tomahawk

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware entry brief location 0/1/CPU0
```

Node: 0/1/CPU0:

L3 - L3 Protocol; L4 - Layer4 Protocol; Intf - Interface;
Dest - Destination Node; V - Virtual;
na - Not Applicable or Not Available;
LU - Local chassis fabric unicast; ←
LM - Local chassis fabric multicast; ←
RU - Multi chassis fabric unicast;
RM - Multi chassis fabric multicast;
def - default

punt to active RP

punt to both active and standby RP

Offset	L3	VRF id	L4	Intf	Dest	laddr,Port	raddr,Port	acl name
<...>								
20	IPV4	default	TCP	any	LU(30)	any,33726	192.85.8.11,179	
22	IPV4	default	TCP	any	LU(30)	any,179	192.85.8.11,any	
36	IPV4	*	TCP	any	LU(30)	any,any	any,179	
38	IPV4	*	TCP	any	LU(30)	any,179	any,any	
21	IPV6	*	TCP	any	LU(30)	any,any	any,179	
22	IPV6	*	TCP	any	LU(30)	any,179	any,any	

```
router bgp 65101
!
neighbor 192.85.8.11
!
neighbor 192.85.9.11
```

- LM is used for BGP-known when both RPs are installed and NSR is enabled

Dynamic LPTS Entries - HW - Tomahawk

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware entry 14protocol tcp location 0/1/CPU0
```

```
Node: 0/0/CPU0:
```

```
<...>
```

```
VRF ID : 0x60000000
Destination IP : any
Source IP : 192.85.8.11
Is Fragment : 0
Interface : any
M/L/T/F : 0/IPv4_STACK/0/BGP-known
DestNode : 48
DestAddr : 48
SID : 7
L4 Protocol : TCP
TCP flag byte : any
Source port : Port:179
Destination Port : 33726
Ct : 0xc68e20
Accepted/Dropped : 2859/263
Lp/Sp : 1/255
# of TCAM entries : 1
Hpo/HAr/HBu/Cir/acl: 2490539/2500pps/1250ms/110pps/
State : Entry in TCAM
Rsp/Rtp : 10/30
```

```
router bgp 65101
bgp router-id 10.255.255.104
!
neighbor 192.85.8.11
!
neighbor 192.85.9.11
```

M - Fabric Multicast;
L - Listener Tag; T - Min TTL;
F - Flow Type;
DestNode - Destination Node;
DestAddr - Destination Fabric queue;
SID - Stream ID;
Po - Policier; Ct - Stats Counter;
Lp - Lookup priority; Sp - Storage Priority;
Ar - Average rate limit; Bu - Burst;
HAr - Hardware Average rate limit; HBu - Hardware Burst;
Cir - Committed Information rate in HAL
Rsp - Relative sorting position;
Rtp - Relative TCAM position;
na - Not Applicable or Not Available

Dynamic LPTS Entries - HW - Lightspeed

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware entry brief location 0/0/CPU0
```

Node: 0/0/CPU0:

* - Vital; L4 - Layer4 Protocol; Intf - Interface; L3 - Layer3 Protocol;
DestNode - Destination Node,

LU(0xY) Y is SFP, packet will reach to location sfp is pointing
MU(0xY) Y is FGID, packet will reach to all location fgid is pointing
Local - packet will be punted to line card;
VRF ID - vrf_id or hardware table id, same as UIDB Table ID;
Slice - slice/np number of the line card;
FlowType - Type of flow, entry belongs to;
na - Not Applicable or Not Available;
def - default

Local Address.Port,

Remote Address,Port/(BFD disc)	VRF ID	L3	L4	Intf	Slice	FlowType	DestNode(s)

<..>							
192.85.8.1,33726	192.85.8.11,179	0x00000000	IPv4	TCP	any	0	BGP-known
192.85.8.1,33726	192.85.8.11,179	0x00000000	IPv4	TCP	any	1	BGP-known
192.85.8.1,179	192.85.8.11	0x00000000	IPv4	TCP	any	0	BGP-cfg-peer
192.85.8.1,179	192.85.8.11	0x00000000	IPv4	TCP	any	1	BGP-cfg-peer
any any,179	any	IPv4	TCP	any	0	BGP-default	LU(0x30)
any any,179	any	IPv4	TCP	any	1	BGP-default	LU(0x30)
any,179 any	any	IPv4	TCP	any	0	BGP-default	LU(0x30)
any,179 any	any	IPv4	TCP	any	1	BGP-default	LU(0x30)
any any,179	any	IPv6	TCP	any	0	BGP-default	LU(0x30)
any any,179	any	IPv6	TCP	any	1	BGP-default	LU(0x30)
any,179 any	any	IPv6	TCP	any	0	BGP-default	LU(0x30)
any,179 any	any	IPv6	TCP	any	1	BGP-default	LU(0x30)

```
router bgp 65101
!
neighbor 192.85.8.11
!
neighbor 192.85.9.11
```

Dynamic LPTS Entries - HW - Lightspeed

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware entry 14protocol tcp location 0/0/CPU0
```

```
Node: 0/0/CPU0:
```

```
<...>
-----
L4 Protocol      : TCP
VRF ID          : 0x00000000
Destination IP   : 192.85.8.1
Source IP/BFD Disc: 192.85.8.11
Port/Type        : Port:33726
Source Port      : 179
Is Fragment     : 0
Is SYN           : any
Is Bundle        : na
Is Virtual       : na
Interface        : any
Slice            : 1
V/L/T/F        : 0/IPv4_STACK/0/BGP-known
DestNode         : LU(0x30)
DestAddr         : LU(0x30)
Accepted/Dropped : 0/0
Po/Ar/Bu        : 6/2500pps/1250ms
State            : pl_pifib_state_complete
-----
```

```
router bgp 65101
!
neighbor 192.85.8.11
!
neighbor 192.85.9.11
```

V - Vital; L - Listener Tag; T - Min TTL;
F - Flow Type;
DestNode - Destination Node;
DestAddr - Destination Fabric Address;
Po - Software Policer Id;
Ar - Average rate limit; Bu - Burst;
na - Not Applicable or Not Available

Dynamic LPTS Entries - HW

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware entry 14proto tcp loc 0/0/CPU0 | util egre -A4 "BGP-known" | util egre "BGP|Po"
Thu Jan  9 17:16:17.712 UTC
V/L/T/F      : 0/IPv4_STACK/0/BGP-known
Po/Ar/Bu     : 6/2500pps/1250ms
RP/0/RSP0/CPU0:ios#
```

- All BGP-known LPTS entries are sharing the same policer
- Consider ACL-based LPTS policer for more specific scenarios

Dynamic LPTS In Action

Lightspeed

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware police np np0 location 0/0/CPU0 | e "0"
          0"
Node 0/0/CPU0:
-----
flow_type priority sw_police_id hw_policer_addr Cur. Rate burst static_avgrate avgrate_type AggrAccepts AggrDrops TOS Value
-----
BGP-known   high     6           186        111      1250    2500      Np            3590       235      01234567
BGP-default  low      8           188        1500     750     1500      Static         2564       10       01234567
```

Anything suspicious in this output?

Tomahawk

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware police np np0 location 0/1/CPU0 | e "0"
          0"
Thu Jan  9 17:36:42.059 UTC
-----
Node 0/1/CPU0:
-----
Burst = 100ms for all flow types
-----
FlowType   Policer Type  Cur. Rate  Def. Rate Accepted   Dropped   TOS Value
-----
BGP-known   6       np      111      2500      2897       257      01234567
BGP-cfg-peer 7       np      99       2000      730        78      01234567
BGP-default  8       Static  1500     1500      2422       23      01234567
```

CISCO Live!

Dynamic LPTS In Action

Lightspeed

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware police np np0 location 0/0/CPU0 | e "0"
```

Node 0/0/CPU0:

flow_type	priority	sw_police_id	hw_policer_addr	Cur. Rate	burst	static_avgrate	avgrate_type	AggrAccepts	AggrDrops	TOS Value
BGP-known	high	6	186	111	1250	2500	Np	3590	235	01234567
BGP-default	low	8	188	1500	750	1500	Static	2564	10	01234567

```
lpts pifib hardware police location 0/0/CPU0
np np0 flow bgp known rate 111
np np0 flow bgp configured rate 99
```

Tomahawk

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware police np np0 location 0/1/CPU0 | e "0"
```

```
Thu Jan  9 17:36:42.059 UTC
```

Node 0/1/CPU0:

Burst = 100ms for all flow types

FlowType	Policer Type	Cur. Rate	Def. Rate	Accepted	Dropped	TOS Value
BGP-known	6	np	111	2500	2897	257
BGP-cfg-peer	7	np	99	2000	730	78
BGP-default	8	Static	1500	1500	2422	23

```
lpts pifib hardware police location 0/1/CPU0
np np0 flow bgp known rate 111
np np0 flow bgp configured rate 99
```

CISCO Live!

Static LPTS Entries

- Exception packets and selected hardcoded protocols

```
lpts punt police location 0/0/CPU0
protocol arp rate 123
exception ipv4 ttl-error rate 11
protocol dhcp broadcast request rate 2222
```

```
RP/0/RSP0/CPU0:ios#sh lpts pifib hardware static-police location 0/0/CPU0
Thu Jan  9 17:43:54.501 UTC
```

```
-----  
Node 0/0/CPU0:  
-----
```

```
Curr and Def Rate are in pps, Burst Rate (ms)
```

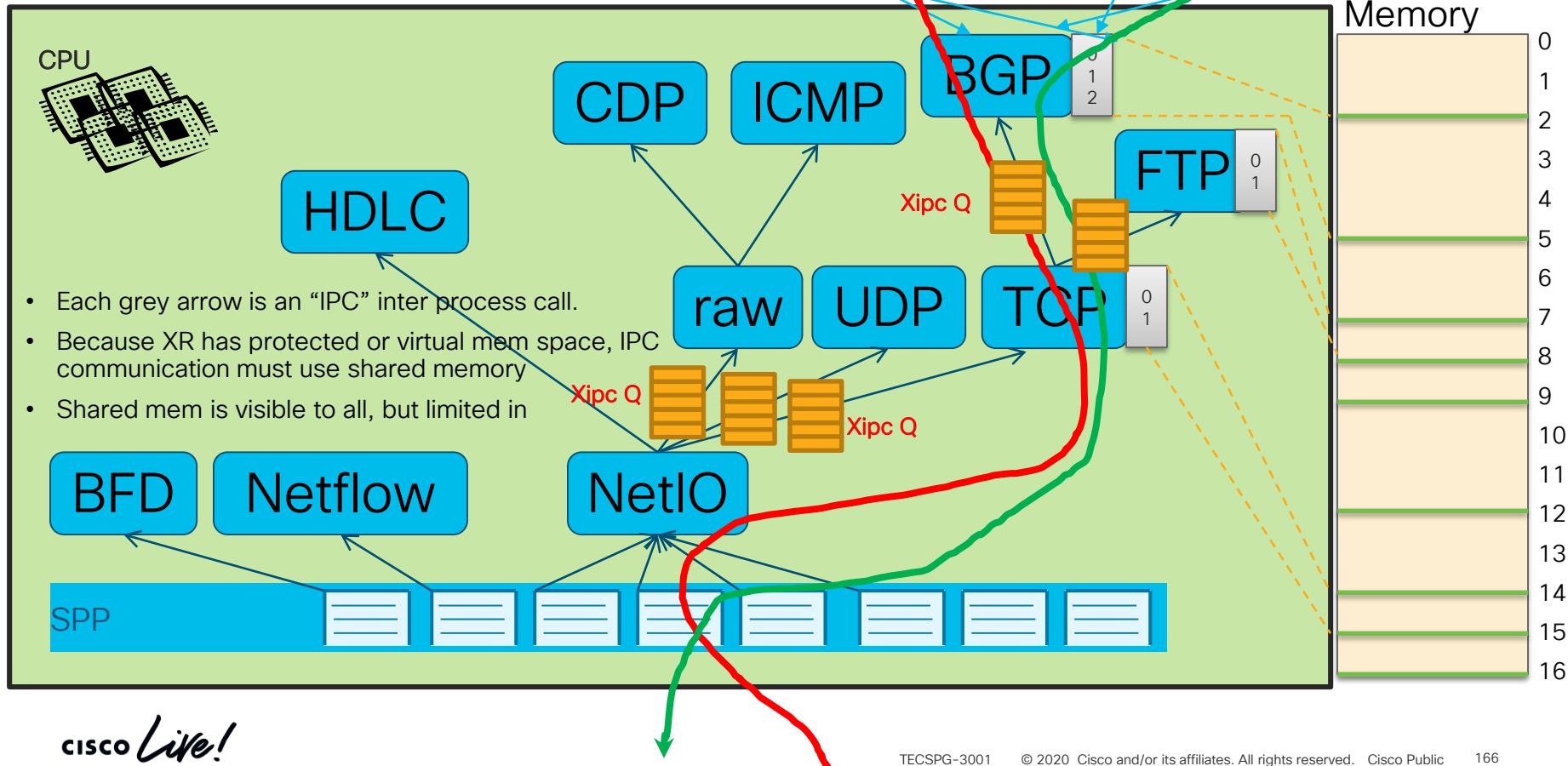
Index	Punt Reason	SID	Curr Rate	Def Rate	Burst Rate	Accepted	Dropped	Destination
0	PUNT_INVALID	NETIO_LO	100	100	20	0	0	Local
1	PUNT_PUNT_ALL	NETIO_HI	1000	1000	200	0	0	Local
2	PUNT_CDP	NETIO_CRUCIAL	1000	1000	200	0	0	Local
3	PUNT_ARP	ARP	123	2000	1000	22	0	Local
8	PUNT_BUNDLE_PROTO	LACP	1000	1000	200	0	0	Local
10	PUNT_DIAGS	DIAG	1000	1000	200	904	0	Local
14	PUNT_CFM_OTHER	CFM	5000	5000	1000	0	0	Local
15	PUNT_CFM_OTHER_RSP	CFM	5000	5000	1000	0	0	0/RSP0/CPU0
16	PUNT_DHCP_SNOOP_REQ	NETIO_MED	2000	2000	500	0	0	0/RSP0/CPU0
17	PUNT_DHCP_SNOOP_REPLY	NETIO_MED	2000	2000	500	0	0	0/RSP0/CPU0
35	PUNT_TTL_EXCEEDED	NETIO_LO	11	2000	400	0	0	Local
135	PUNT_DHCP_BCAST	NETIO_LO	2222	1000	200	0	0	Local
136	PUNT_DHCP_BCAST_REPLY	NETIO_LO	1000	1000	200	0	0	Local

LPTS Tuning

- Default tuning are good for typical PE
- Special network role may require tuning
 - E.g.: BGP RR → allow higher BGP known rate
- Too high rates allowed by LPTS → upper layers may experience congestion
 - Validate your LPTS tuning

SW Punt Path

Where are XIPC Queues



SPP Queues Towards Upper Layers

- BGP example → SPP delivers packets to NetIO via High priority queue

```
RP/0/RSP0/CPU0:ios#sh spp client location 0/RSP0/CPU0 | b netio_queue_high  
netio_queue_high, JID 440 (pid 6621)  
-----  
Reconnect Pending: F, Exited: F, Keep Queues: F, Pakman Client: T  
Quota Current: 0, Limit: 3072, Drops 0  
Queues:  
Key      Cur   Max      Enqueues      High Watermark (Timestamp)      Drops  
0xffffffff  0    10          0      0 (18:02:20.968 Jan 06 20 UTC)      0  
0x03000007  0  3072      29691      1 (18:33:05.670 Jan 06 20 UTC)      0
```

```
RP/0/RSP0/CPU0:ios#sh proc netio | i Job  
Job Id: 440  
RP/0/RSP0/CPU0:ios#
```

NetIO Queues Towards Upper Layers

- BGP example → NetIO delivers packets to TCP via High priority queue

```
RP/0/RSP0/CPU0:ios#sh netio clients location 0/RSP0/CPU0
<..>
          Input          Punt          XIPC InputQ      XIPC PuntQ
ClientID   Drop/Total    Drop/Total  Cur/High/Max  Cur/High/Max
-----
<..>
tcp        L 0/110857    0/0        L 0/30/8000    0/0/0
            H 0/72787
<..>
arp         0/129905     0/1        0/10/1000    0/1/1000
<..>
```

- XIPC Input queue → direct queue towards application
- XIPC PuntQ → queue for packets that NetIO keeps while it waits on application to perform an action
 - E.g.: ARP:
 - Static LPTS policer punts traffic because there's no adjacency
 - NetIO holds the packet, while it signals ARP process that adjacency resolution for IP address x.x.x.x is required
 - ARP generates a request → receives a reply → notifies NetIO → NetIO injects the packet into data-path

NetIO → TCP XIPCQ

```
RP/0/RSP0/CPU0:ios#sh proc tcp | i Job
```

Job Id: 323

```
RP/0/RSP0/CPU0:ios#sh xipcq jid 323
```

Id	Name	Size	Cur Size	Produced	Dropped	HWM
17	XIPC_xipcq_12_0_10292_7322_...	30000	0	94857	0	17
16	XIPC_xipcq_12_0_10292_7322_...	12000	0	5531	0	4
8	XIPC_tcp_124	2000	0	6	0	1
7	XIPC_tcp_125	6000	0	0	0	0
4	XIPC_tcp_iq_6	2666	0	42768	0	15
5	XIPC_tcp_iq_3	2666	0	4472	0	7
15	XIPC_tcp_iq_5	2666	0	77196	0	26
13	XIPC_tcp_ctrl_iq_0	2666	0	98	0	1
11	XIPC_tcp_iq_7	2666	0	36991	0	16
9	XIPC_tcp_psb_0	16000	0	0	0	0
14	XIPC_tcp_iq_2	2666	0	11666	0	15
10	XIPC_tcp_iq_4	2666	0	47511	0	19
6	XIPC_tcp_iq_8	2666	0	31466	0	17
3	XIPC_tcp_ctrl_iq_1	2666	0	0	0	0
12	XIPC_tcp_iq_9	2666	0	39074	0	29
2	XIPC_tcp_i1	8000	0	84210	0	11
1	XIPC_tcp_i0	8000	0	112255	0	30

```
RP/0/RSP0/CPU0:ios#
```

High-watermark: highest ever number of packets sitting in the queue

Max allowed number of packets in the queue

TCP → BGP XIPCQ

```
RP/0/RSP0/CPU0:ios#sh proc bgp | i Job
```

```
Job Id: 1087
```

```
RP/0/RSP0/CPU0:ios#sh xipcq jid 1087
```

Id	Name	Size	Cur Size	Produced	Dropped	HWM
2	XIPC_xipcq_12_0_10292_7322_...	30000	0	94882	0	26
1	XIPC_xipcq_12_0_10292_7322_...	12000	0	170203	0	33

```
RP/0/RSP0/CPU0:ios#
```

```
RP/0/RSP0/CPU0:ios#sh xipcq jid 1087 queue-id 1
```

```
XIPC_xipcq_12_0_10292_7322_inst_1_data_toapp:
```

```
Magic: 12344321
```

```
<..>
```

```
Max Queue Size: 12000
```

```
Queue Size: 0
```

```
Client Queue Size: 12000
```

```
High watermark: 33
```

```
Priority Queues:
```

Prio	Size	Drops	Total
Unspec	12000	0	0
Normal	12000	0	170226
Medium	12000	0	0
High	12000	0	0
Crucial	12000	0	0

```
RP/0/RSP0/CPU0:ios#sh xipcq jid 1087 queue-id 2
```

```
XIPC_xipcq_12_0_10292_7322_inst_1_ctrl_toapp:
```

```
Magic: 12344321
```

```
<..>
```

```
Max Queue Size: 30000
```

```
Queue Size: 0
```

```
Client Queue Size: 30000
```

```
High watermark: 26
```

```
Priority Queues:
```

Prio	Size	Drops	Total
Unspec	30000	0	0
Normal	30000	0	94929
Medium	30000	0	0
High	30000	0	0
Crucial	30000	0	0

IOS XR Embedded Packet Tracer

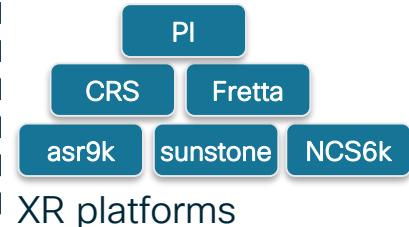
Introduction

- XR Embedded Packet Tracer empowers users to:
 - perform in-depth triaging of packet forwarding issues in data, punt and inject path
 - Validate service provisioning by tracing the matching flow through the system
 - Learn the platform and XR platform independent forwarding infrastructure
- Key functionalities:
 - Use flexible condition setting to specify a flow of interest.
 - Trace packets of that flow through the system.
 - Provide insight into the features executed on the packet

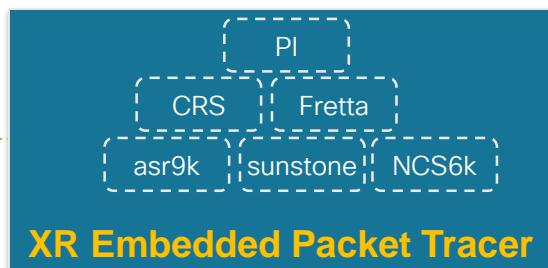
Packet Tracer Ecosystem

Automated

Manual



Individual device



Cross-platform

XR Embedded Packet Tracer

XE Packet Tracer

Nexus Packet Tracer

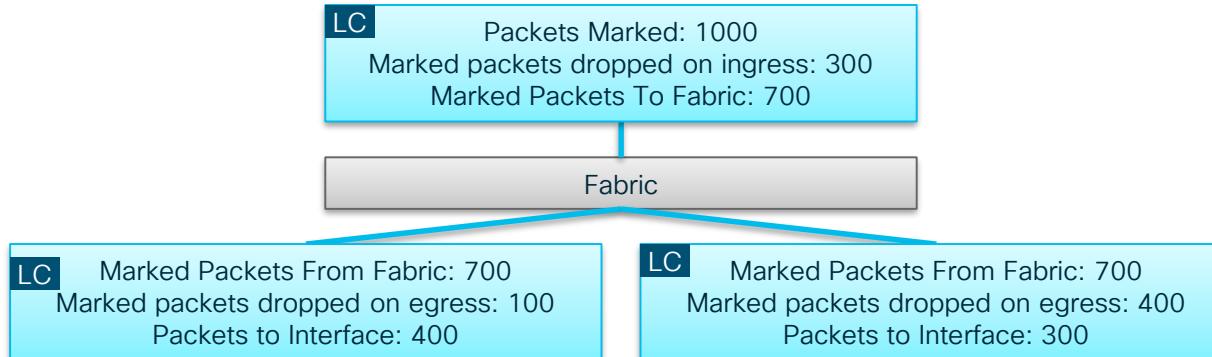
Catalyst Packet Tracer

Network Packet Tracer



Network level

XR Packet Tracer Use Cases



Supported Platforms/Releases

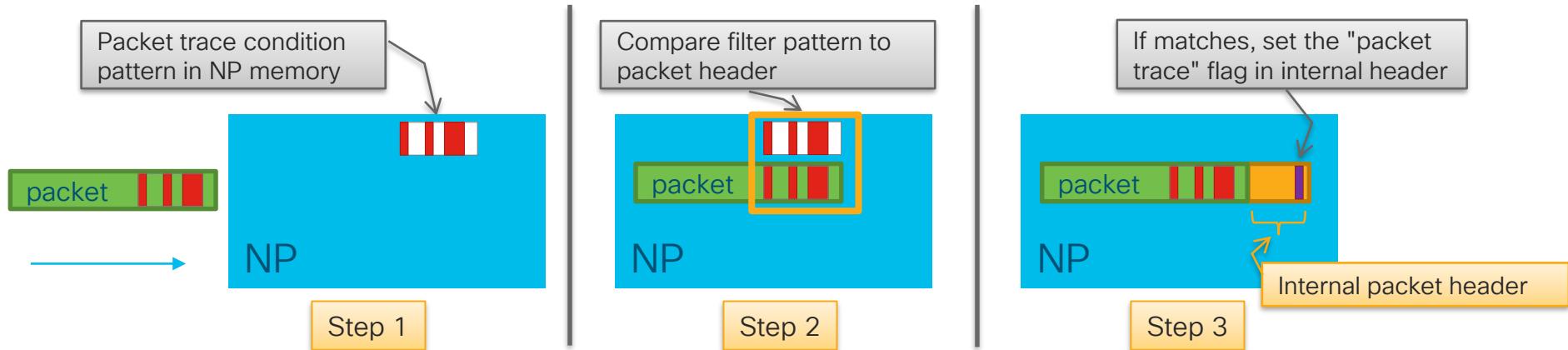
- XR Embedded Packet Tracer framework is platform independent
- XR Release 7.0.x:
 - Marking on ASR 9000 Tomahawk NP
 - Counting of packets on ASR 9000 Tomahawk NP
- XR Release 7.1.2:
 - Marking on ASR 9000 Tomahawk and Lightspeed NP
 - Counting of packets on ASR 9000 Tomahawk and Lightspeed NP

Available Counters

XR Release 7.0.x

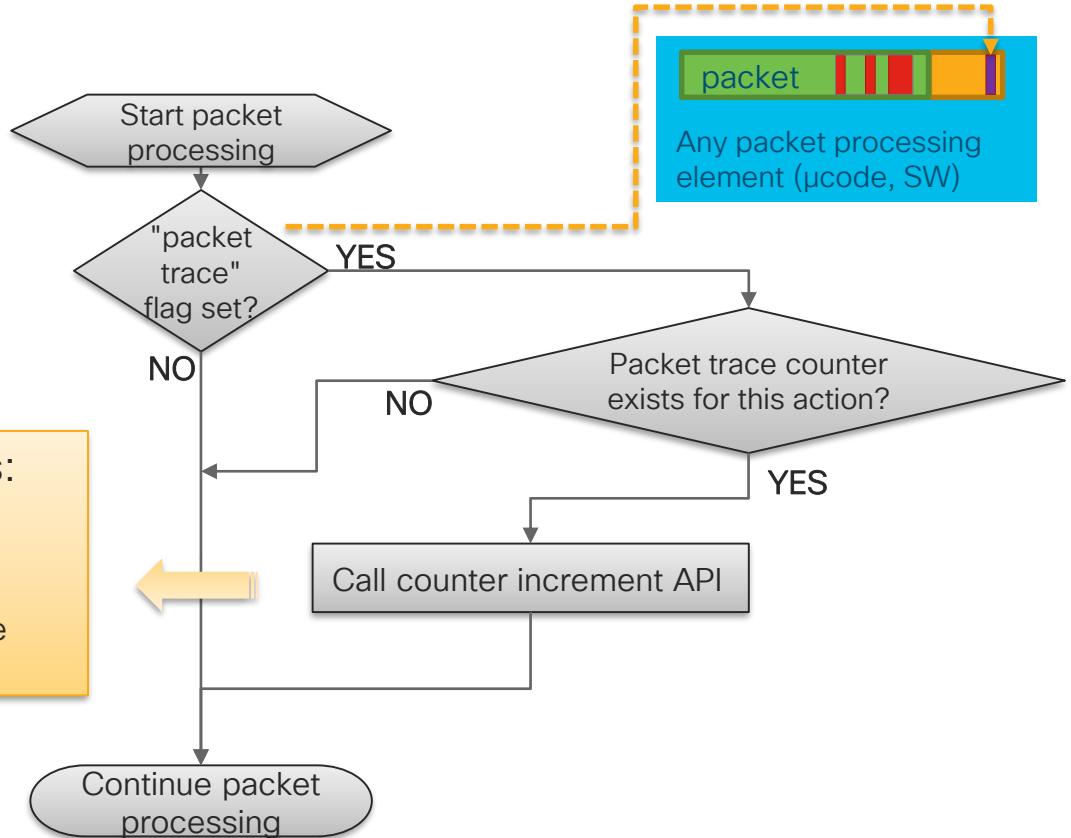
Name	Type	Description
PACKET_MARKED	Marking	Packet marked on network interface
PACKET_FROM_INJECT	Pass	Packet injected by linecard CPU
PACKET_ING_DROP	Drop	Packet dropped on ingress
PACKET_TO_FABRIC	Pass	Packet sent to router fabric
PACKET_TO_PUNT	Pass	Packet punted to linecard CPU
PACKET_FROM_FABRIC	Pass	Packet received from router fabric
PACKET_EGR_DROP	Drop	Packet dropped on egress
PACKET_TO_INTERFACE	Pass	Packet sent to network interface

Matching Packets In a Flow

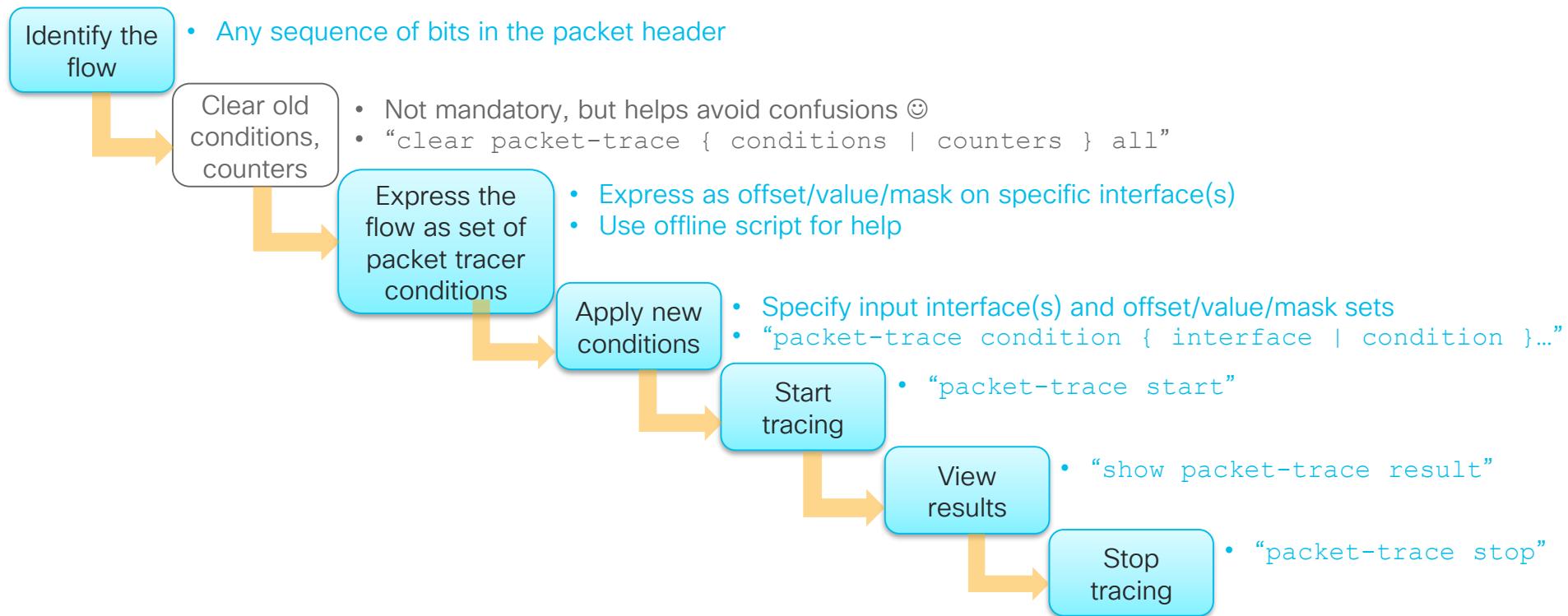


- Steps 1 to 3 executed before the packet header is parsed by the µcode
 - i.e. before sub-interface classification
- All subsequent elements in the packet path act based on the "packet trace" flag
- "Packet trace" flag is preserved in all internal headers (NP, fabric, punt PD, punt PI)

Accounting Of Matched Packets



Packet Trace Workflow



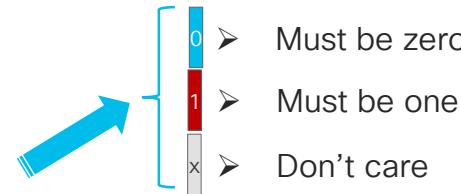
User Interaction

- User interaction with the packet trace framework is entirely contained in the user mode.
 - No need for any configuration changes.
- User can view the actions performed on packets through counters
- Counters identify the entity that has processed the packet and the action performed
- In 7.0.1 and 7.1.2 user interaction is provided via CLI
 - YANG models are planned for future

Packet Tracer CLIs

Command	Description
<code>packet-trace condition interface <interface></code>	Specify interface on which marking should be enabled
<code>packet-trace condition <id> offset <offset> value <value> mask <mask></code>	Specify the conditions as a set of offset/value/mask triplets
<code>packet-trace start</code>	Start packet marking
<code>packet-trace stop</code>	Stop packet marking
<code>show packet-trace status</code>	Display the status of packet tracer: <ul style="list-style-type: none">• Conditions buffered by packet trace master process• Tracing state (active/inactive), based on the execution of “start” and “stop” commands
<code>show packet-trace status [detail]</code>	Same as above, plus for every location: <ul style="list-style-type: none">• List every counting module and any errors it reported• List every marking module, active conditions and any errors it reported
<code>show packet-trace result</code>	Display all non-zero counters
<code>clear packet-trace conditions all</code>	Clear all conditions (only allowed when tracing stops)
<code>clear packet-trace counters all</code>	Clear all counters
<code>show packet-trace description [detail]</code>	Display all supported counters

Packet Trace Condition

- XR Packet Tracer is protocol agnostic.
- “Condition” is expressed in offset/value/mask triplets
- Consider a flow identified by a sequence of bits where:
- Offset is expressed in octets from the very start of the Ethernet frame
- Value defines the expected values on positions we care about starting from the offset
 - You can set 0 or 1 into positions you don't care about, in this example we use 0
- Mask defines which bits we care and which we don't care about starting from the offset



Pattern starts with octet 1 → offset: 1
Pattern value: b1011001110010000
 b 3 9 0 → value : 0xb390
Pattern mask : b1111101110110000
 f b b 0 → mask : 0xfb0

Packet Trace Condition

- Offline help script: <>will share via WebexTeams>>

```
gpk-ads-063$ ./packet-trace-condition.py -h
CLI options:
-h, --help                      : print help
--dmac <value>                  : Destination MAC (char 'x' permitted as mask)
--smac <value>                  : Source MAC (char 'x' permitted as mask)
--mpls_labels <value,value>     : comma-separated list of MPLS labels
--pwethcw                        : Pseudowire Control Word is present after MPLS header
--inner_dmac <value>            : Destination MAC (char 'x' permitted as mask)
--inner_smac <value>            : Source MAC (char 'x' permitted as mask)
--vlans <value,value>           : comma-separated list of VLAN IDs
--TPIDs <value,value>           : comma-separated list of TPIDs (default: 0x8100)
--CoSs <value,value>            : comma-separated list of CoS values (default: 0)
--src_ipv4 <value>              : source IPv4 address
--src_ipv4_mask <value>         : source IPv4 address mask (default: 255.255.255.255)
--dst_ipv4 <value>              : destination IPv4 address
--dst_ipv4_mask <value>         : destination IPv4 address mask (default: 255.255.255.255)
--src_port <value>              : source L4 port
--dst_port <value>              : destination L4 port
```

Packet Trace Condition

- Example:

```
$ packet-trace-condition.py --vlans 110,2001 --dst_ipv4 10.1.2.4 --dst_ipv4_mask 255.255.255.252
# VLAN 110:
offset 14 value 0x006e mask 0xffff
# VLAN 2001:
offset 18 value 0x07d1 mask 0xffff
# Dst IPv4:
offset 38 value 0xa010204 mask 0xfffffffffc
```

- Disclaimer and recommendations:

- Use it “as is”
- Include fields that define encapsulation type because they impact the offset

```
$ packet-trace-condition.py --dst_ipv4 10.1.2.4 --dst_ipv4_mask 255.255.255.252
# Dst IPv4:
offset 30 value 0xa010204 mask 0xfffffffffc
```

- When using for the 1st time for particular topology double-check that offset is correct.
- In EoMPLS topologies, do not forget to include control word flag “--pwethcw” where applicable!

Packet Trace Condition

- Specify interfaces where condition is applied
 - Condition applies to physical interface
 - To trace on sub-interface, calculate in the encapsulation in the offset
 - For more secure match on sub-interface, include the dot1q/QinQ tag values in the condition
- Specify the offset/value/mask sets
 - Tip: stick to 1, 2 and 3 to specify condition IDs
 - Tomahawk NP supports up to three 4-octet value/mask sets

```
packet-trace condition interface Hu0/5/0/1
packet-trace condition interface Hu0/5/0/3
packet-trace condition 1 offset 14 value 0x7e4 mask 0xffff
packet-trace condition 2 offset 30 value 0xc0a80003 mask 0xffffffff
packet-trace condition 3 offset 34 value 0xc0a80002 mask 0xffffffff
```

Clear Conditions and Counters

- Packet trace counters can be cleared at any time

```
clear packet-trace counters all
```

- Packet trace conditions can be cleared when packet tracing is not active

```
clear packet-trace conditions all
```

Packet Trace Start / Stop

- Packet tracer master process sends the specified conditions via GSP channel when the user starts the packet tracer
- Upon receiving the conditions, prm_server_to process programs the condition and starts marking packets
- Packet trace flag is preserved in the NP/fabric/punt/inject packet headers

```
packet-trace start
```

- When packet tracing is stopped, marking modules clear the condition and stop marking packets

```
packet-trace stop
```

Show Packet Trace Status - Simple Output

- Displays conditions buffered by packet trace master and status
 - Status is derived from the packet-trace start/stop command

```
RP/0/RSP0/CPU0:CORE-TOP#show packet-trace status
```

Packet Trace Master Process:

Buffered Conditions:

```
Interface HundredGigE0_5_0_1
Interface HundredGigE0_5_0_3
1 offset 14 value 0x7e4 mask 0xffff
2 offset 30 value 0xc0a80003 mask 0xffffffff
3 offset 34 value 0xc0a80002 mask 0xffffffff
```

Conditions buffered by the
packet trace master process
on active RP

Status: Inactive

Status derived from the start/stop command

```
RP/0/RSP0/CPU0:CORE-TOP#
```

Show Packet Trace Status – Detailed Output

- Displays conditions buffered by packet trace master and status
 - Status is derived from the start/stop command
- For every location display:
 - registered counting modules
 - registered marking modules and active marking condition
 - errors

Show Packet Trace Status - Detailed Output

```
RP/0/RSP0/CPU0:CORE-TOP#show packet-trace status detail
Packet Trace Master Process:
```

Buffered Conditions:

```
Interface HundredGigE0_5_0_1
Interface HundredGigE0_5_0_3
1 offset 14 value 0x7e4 mask 0xffff
2 offset 30 value 0xc0a80003 mask 0xffffffff
3 offset 34 value 0xc0a80002 mask 0xffffffff
```

Status: **Active**

```
Location: 0/RSP0/CPU0
```

```
Available Counting Modules: 2
```

```
#1 spp_punt_inject
Last errors:
```

```
#2 netio
Last errors:
```

```
Available Marking Modules: 0
```

```
Location: 0/1/CPU0
```

```
Available Counting Modules: 3
```

```
#1 spp_punt_inject
Last errors:
```

```
#2 ssmh
Last errors:
```

```
#3 netio
Last errors:
```

```
Available Marking Modules: 0
```

```
Location: 0/5/CPU0
```

```
Available Counting Modules: 3
```

```
#1 spp_punt_inject
Last errors:
```

```
#2 ssmh
Last errors:
```

```
#3 netio
Last errors:
```

```
Available Marking Modules: 1
```

```
#1 prm_server_to
Interfaces: 2
  HundredGigE0/5/0/1
  HundredGigE0/5/0/3
```

```
Conditions: 3
```

```
1 offset 14 value 0x7e4 mask 0xffff
2 offset 30 value 0xc0a80003 mask 0xffffffff
3 offset 34 value 0xc0a80002 mask 0xffffffff
```

```
Last errors:
```

```
RP/0/RSP0/CPU0:CORE-TOP#
```

View Packet Trace Results

Counter type:

- **Pass** – this counter describes the action performed on the packet. Packet is passed on for further processing.
- **Drop** – this counter signals that the NP µcode made a decision to drop the packet. Drop reason is not provided. Try correlating the NP drop counters with the increment of this counter
- **Marking** – packet has matched the condition and was marked for tracing

```
show packet-trace results
```

Supported counters in 7.0.1:

Name	Type	Description
PACKET_MARKED	Marking	Packet marked on network interface
PACKET_FROM_INJECT	Pass	Packet injected by linecard CPU
PACKET_ING_DROP	Drop	Packet dropped on ingress
PACKET_TO_FABRIC	Pass	Packet sent to router fabric
PACKET_TO_PUNT	Pass	Packet punted to linecard CPU
PACKET_FROM_FABRIC	Pass	Packet received from router fabric
PACKET_EGR_DROP	Drop	Packet dropped on egress
PACKET_TO_INTERFACE	Pass	Packet sent to network interface

View Packet Trace Results

```
RP/0/RSP0/CPU0:CORE-TOP#show packet-trace results
```

Thu Jul 11 17:03:57.477 UTC

T: D - Drop counter; P - Pass counter

Location	Source	Counter	T	Last-Attribute	Count
0/5/CPU0	NP0	PACKET_MARKED	P	HundredGigE0_5_0_1	1000
0/5/CPU0	NP0	PACKET_TO_FABRIC	P		1000
0/5/CPU0	NP0	PACKET_FROM_FABRIC	P		1000
0/5/CPU0	NP0	PACKET_TO_INTERFACE	P	HundredGigE0_5_0_0	1000

Location

Counter name

Type (explained
on previous slide)

Counter value

Source identifies the NP
number on the location

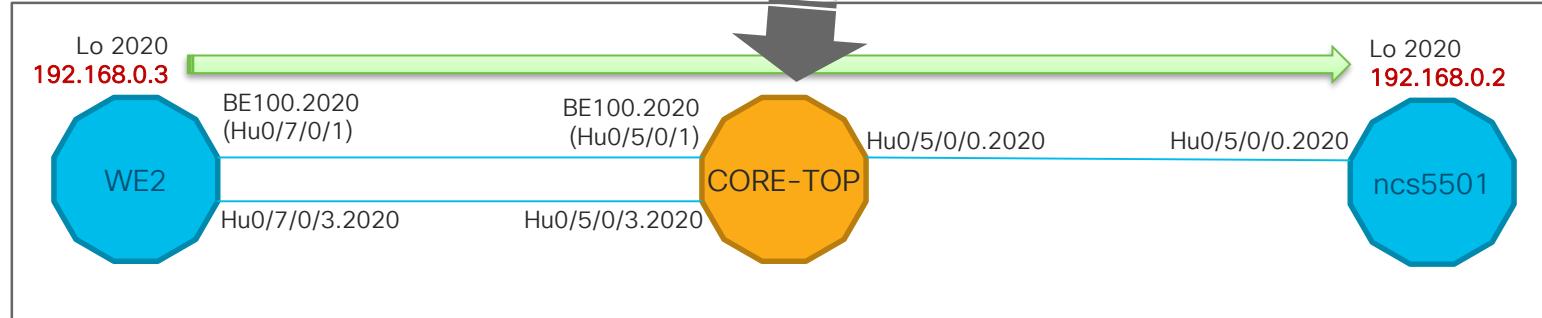
Order of counters tries to illustrate the order
of events in packet processing, but this may
not always be the case.

- With every counter update, packet trace framework also receives the timestamp and allows for a freeform “attribute” that describes more closely the action.
- The use of attribute and its meaning differs between counters
- Current CLI only exposes the last attribute to the user.
- In future releases, user will be able to see the last 1023 timestamps and attributes per counter.

Packet Trace Example 1: Transit Packets (Step-By-Step Guide)

Topology

```
interface Bundle-Ether100.2020
vrf packet-tracer-test
ipv4 address 10.10.11.10 255.255.255.0
encapsulation dot1q 2020
!
interface HundredGigE0/5/0/1.2020
vrf packet-tracer-test
ipv4 address 202.202.202.1 255.255.255.252
encapsulation dot1q 2020
```



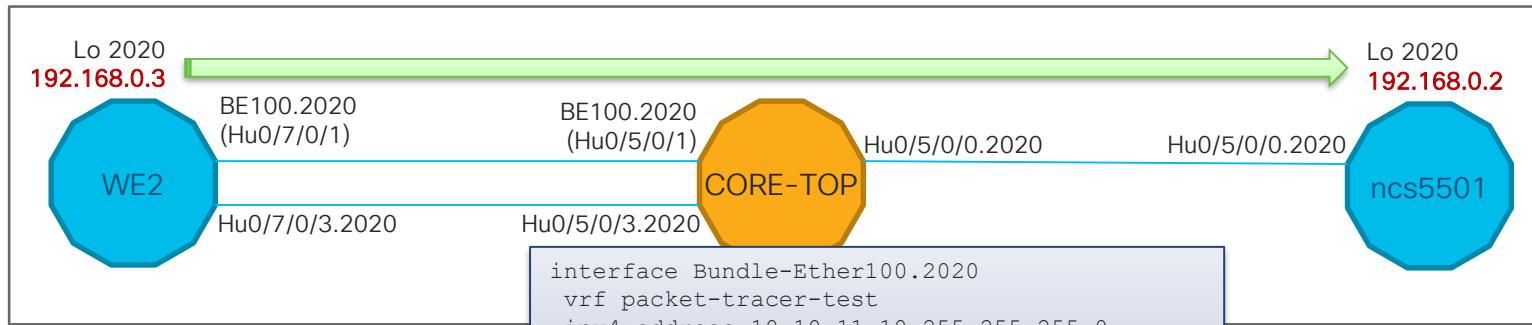
- In this example we will trace the packet flow from WE2 to ncs5501
 - source IPv4 address: 192.168.0.3 (Loopback 2020 of WE2)
 - destination IPv4 address: 192.168.0.2 (Loopback 2020 of ncs5501).
- Packet tracer will be activated on CORE-TOP router.
 - As we don't know on which of the two interfaces facing the WE2 can the flow come in, packet tracer will be activated on both interfaces facing the WE2.

Step #0: Stop prior tracing, clear conditions/counters

```
packet-trace stop  
clear packet-trace conditions all  
clear packet-trace counters all
```

Step #1: Specify the conditions

```
packet-trace condition interface hu0/5/0/1
packet-trace condition interface hu0/5/0/3
packet-trace condition 1 offset 14 value 0x7e4 mask 0xffff ← Vlan 2020
packet-trace condition 2 offset 30 value 0xc0a80003 mask 0xffffffff ← Src IPv4
packet-trace condition 3 offset 34 value 0xc0a80002 mask 0xffffffff ← Dst IPv4
```



```
interface Bundle-Ether100.2020
vrf packet-tracer-test
ipv4 address 10.10.11.10 255.255.255.0
encapsulation dot1q 2020
!
interface HundredGigE0/5/0/1.2020
vrf packet-tracer-test
ipv4 address 202.202.202.1 255.255.255.252
encapsulation dot1q 2020
```

Step #2: Verify the conditions were understood by the packet trace master process

```
RP/0/RSP0/CPU0:CORE-TOP#show packet-trace status
Thu Jul 11 16:48:15.987 UTC

Packet Trace Master Process:

Buffered Conditions:
  Interface HundredGigE0_5_0_1
  Interface HundredGigE0_5_0_3
  1 offset 14 value 0x7e4 mask 0xffff
  2 offset 30 value 0xc0a80003 mask 0xffffffff
  3 offset 34 value 0xc0a80002 mask 0xffffffff

Status: Inactive

RP/0/RSP0/CPU0:CORE-TOP#
```

Step #5: Start the flow

- Don't forget to start the flow



Step #6: Verify packet tracer results

```
RP/0/RSP0/CPU0:CORE-TOP#show packet-trace results
```

Thu Jul 11 17:03:57.477 UTC

T: D - Drop counter; P - Pass counter

Location	Source	Counter	T	Last-Attribute	Count
0/5/CPU0	NP0	PACKET_MARKED	-		
0/5/CPU0	NP0	PACKET_TO_FABRIC	P	HundredGigE0_5_0_1	1000
0/5/CPU0	NP0	PACKET_FROM_FABRIC	P		1000
0/5/CPU0	NP0	PACKET_TO_INTERFACE	P	HundredGigE0_5_0_0	1000

```
RP/0/RSP0/CPU0:CORE-TOP#
```

NP0 on 0/5/CPU0 has sent 1000 packets to interface(s), with last attribute HundredGigE0_5_0_0

NP0 on 0/5/CPU0 has received 1000 packets from fabric.

NP0 on 0/5/CPU0 has forwarded 1000 packets to fabric.

NP0 on 0/5/CPU0 has received 1000 packets, with last attribute HundredGigE0_5_0_1

Packet Trace Example 2: NP μcode drops: policer

NP μcode drops: policer

```
RP/RSP0/CPU0:CORE-TOP#sh packet-trace status
```

Packet Trace Master Process:

Buffered Conditions:

Interface HundredGigE0 5 0 7

1 offset 14 value 0x5dc00 mask 0xfffffff0

2 offset 24 value 0x964d3000 mask 0xffffffff

PW label

Last 4 octets of inner DMAC
In a PW with a control-word

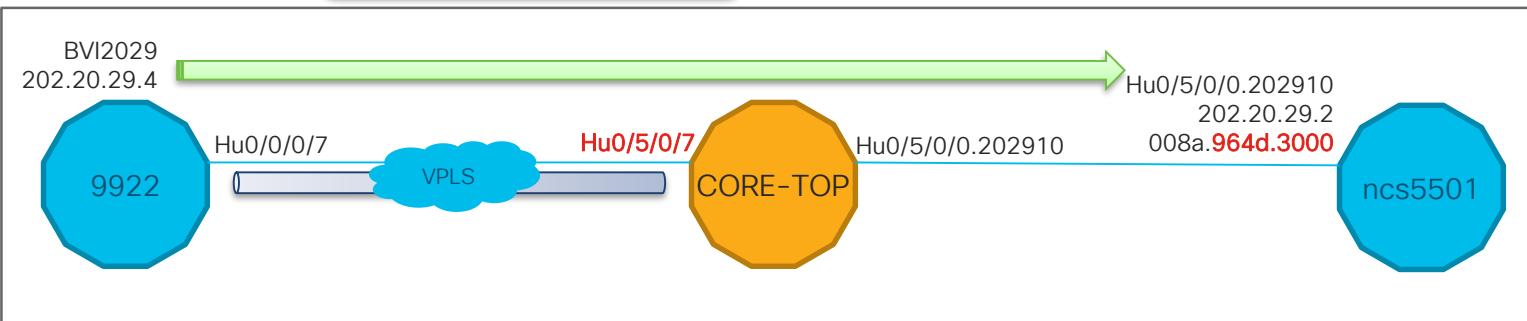
```
interface HundredGigE0/5/0/0.202910 12transport
encapsulation dot1q 2029 second-dot1q 10
rewrite ingress tag pop 2 symmetric
service-policy output pkt-trace-qos-drop-test
!
```

```
policy-map pkt-trace-qos-drop-test
class we2-to-ncs5501
```

police rate 1024 kbps

```
!
class class-default
police rate 64 kbps
```

traffic hits this
egress policer



NP μcode drops: policer

```
RP/0/RSP0/CPU0:CORE-TOP#sh policy-map interface hu0/5/0/0.202910 output
HundredGigE0/5/0/0.202910 output: pkt-trace-qos-drop-test

Class we2-to-ncs5501
<...snip...
Class class-default
  Classification statistics          (packets/bytes)      (rate - kbps)
    Matched                      :           1031/129886            8
    Transmitted                   :       N/A
    Total Dropped             :           36/4536                0
  Policing statistics            (packets/bytes)      (rate - kbps)
    Policed(conform)            :           995/125350            8
    Policed(exceed)             :           36/4536                0
    Policed(violate)            :           0/0                  0
    Policed and dropped        :           36/4536
```

NP µcode drops: policer

- Policer drop decision is made in the µcode → packet trace drop counter increments

```
RP/0/RSP0/CPU0:CORE-TOP#sh packet-trace results
```

```
Tue Aug 13 14:18:11.495 UTC
```

```
T: D - Drop counter; P - Pass counter
```

```
Location | Source | Counter
```

Location	Source	Counter	T	Last-Attribute	Count
0/5/CPU0	NP3	PACKET_MARKED	P	HundredGigE0_5_0_7	1026
0/5/CPU0	NP3	PACKET_TO_FABRIC	P		1026
0/5/CPU0	NP0	PACKET_FROM_FABRIC	P		1026
0/5/CPU0	NP0	PACKET_EGR_DROP	D		36
0/5/CPU0	NP0	PACKET_TO_INTERFACE	P	HundredGigE0_5_0_0	990

NP0 on 0/5/CPU0 has sent 990 packets to interface(s), with last attribute HundredGigE0_5_0_0

NP0 on 0/5/CPU0 has dropped 36 packets in egress direction (drop reason in future releases)

NP0 on 0/5/CPU0 has received 1026 packets from fabric.

NP3 on 0/5/CPU0 has forwarded 1026 packets to fabric.

NP3 on 0/5/CPU0 has received 1026 packets, with last attribute HundredGigE0_5_0_7

NP μcode drops: policer - XR 7.2.1 (7.1.2)

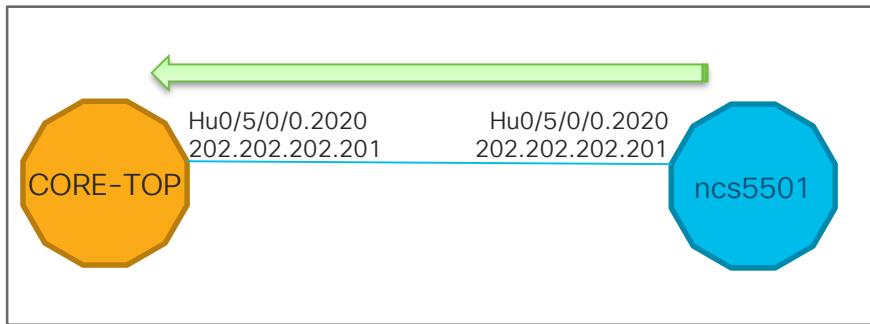
- Last-attribute shows the last drop reason
- Snapshot from a Lightspeed line card
 - Note same look and feel of output from Tomahawk and Lightspeed

```
RP/0/RSP0/CPU0:ios#sh packet-trace r
Fri Jan  3 13:27:04.630 UTC
T: D - Drop counter; P - Pass counter
Location      | Source        | Counter          | T | Last-Attribute           |       Count
-----+-----+-----+-----+-----+-----+
0/0/CPU0      NPO    PACKET_MARKED   P  HundredGigE0_0_0_0      12000
0/0/CPU0      NPO    PACKET_ING_DROP  D  IPv4 ingress QoS policer  11232
0/0/CPU0      NPO    PACKET_TO_FABRIC  P
0/1/CPU0      NPO    PACKET_FROM_FABRIC  P
0/1/CPU0      NPO    PACKET_TO_INTERFACE P
0/RSP0/CPU0    NETIO_PD   PKT_NETIO_PD_PUNT  P
0/RSP0/CPU0    NETIO_PD   PKT_NETIO_PD_INJECT P
RP/0/RSP0/CPU0:ios#
```

Packet Trace Example 3: NP µcode drops: ACL

NP μcode drops: ACL

```
interface HundredGigE0/5/0/0.2020
vrf packet-tracer-test
ipv4 address 202.202.202.201 255.255.255.252
ipv4 unreachables disable
encapsulation dot1q 2020
ipv4 access-group packet-trace-test-2 ingress
!
ipv4 access-list packet-trace-test-2
10 deny icmp host 202.202.202.202 host 202.202.202.201
20 permit ipv4 any any
!
```



NP µcode drops: ACL

```
RP/0/RSP0/CPU0:CORE-TOP#sh packet-trace status
```

Packet Trace Master Process:

Buffered Conditions:

Interface HundredGigE0 5 0 0	VLAN 2022
1 offset 14 value 0x7e4 mask 0xffff	Src IPv4 202.202.202.202
2 offset 30 value 0xcacacaca mask 0xffffffff	Dst IPv4 202.202.202.201
3 offset 34 value 0xcacacac9 mask 0xffffffff	

Future releases (7.1.2) will include an attribute describing drop reason

```
RP/0/RSP0/CPU0:CORE-TOP#sh packet-trace results
```

T: D - Drop counter; P - Pass counter

Location	Source	Counter		T	Last-Attribute		Count
0/5/CPU0	NP0	PACKET_MARKED	-	P	HundredGigE0 5 0 0	-	1000
0/5/CPU0	NP0	PACKET_ING_DROP		D			1000

```
RP/0/RSP0/CPU0:CORE-TOP#sh access-lists packet-trace-test-2 hardware ingress interface hu 0/5/0/0.2020 location
```

0/5/CPU0

Tue Aug 13 15:21:05.905 UTC

Stats mode: 1 (0 - Interface stats,1 - ACE stats)

ipv4 access-list packet-trace-test-2

10 deny icmp host 202.202.202.202 host 202.202.202.201 (1000 hw matches)

20 permit ipv4 any any (157 hw matches)

```
RP/0/RSP0/CPU0:CORE-TOP#
```

Packet Trace Example 4: NP TM drops

NP TM drops

- NP TM (traffic manager) drops are outside of NP µcode.
- Queuing in NP TM is implemented in hardware (no support for packet tracer).
- Consequence: Shaper, WRED, queue-limit drops are not accounted for in the packet tracer framework

Packet Trace Example 5: ICMP Echo Request

ICMP Echo Request Processing

- IOS XR reuses the packet buffer carrying the ICMP echo request to generate the ICMP echo reply
- Src/Dst IP is swapped, ICMP type/code is updated
- Packet trace flag is preserved in ICMP echo reply, allowing view of the full path of the ICMP echo request/reply processing

ICMP Echo Request Processing

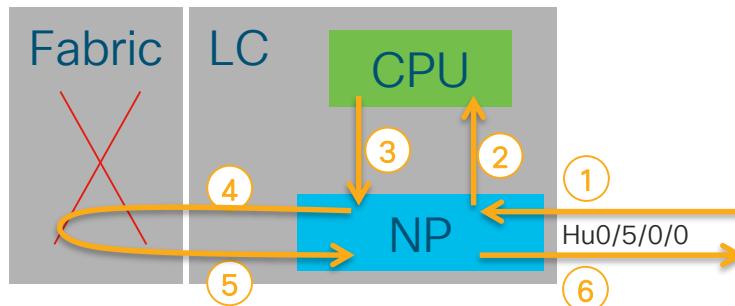
```
RP/0/RSP0/CPU0:CORE-TOP#packet-trace show results
```

T: D - Drop counter; P - Pass counter

Location | Source | Counter

Location	Source	Counter	T	Last-Attribute	Count
0/5/CPU0	NP0	PACKET_MARKED ①	P	HundredGigE0_5_0_0	8000
0/5/CPU0	NP0	PACKET_FROM_INJECT ③	P		8000
0/5/CPU0	NP0	PACKET_TO_FABRIC ④	P		8000
0/5/CPU0	NP0	PACKET_TO_PUNT ②	P		8000
0/5/CPU0	NP0	PACKET_FROM_FABRIC ⑤	P		8000
0/5/CPU0	NP0	PACKET_TO_INTERFACE ⑥	P	HundredGigE0_5_0_0	8000

```
RP/0/RSP0/CPU0:CORE-TOP#
```



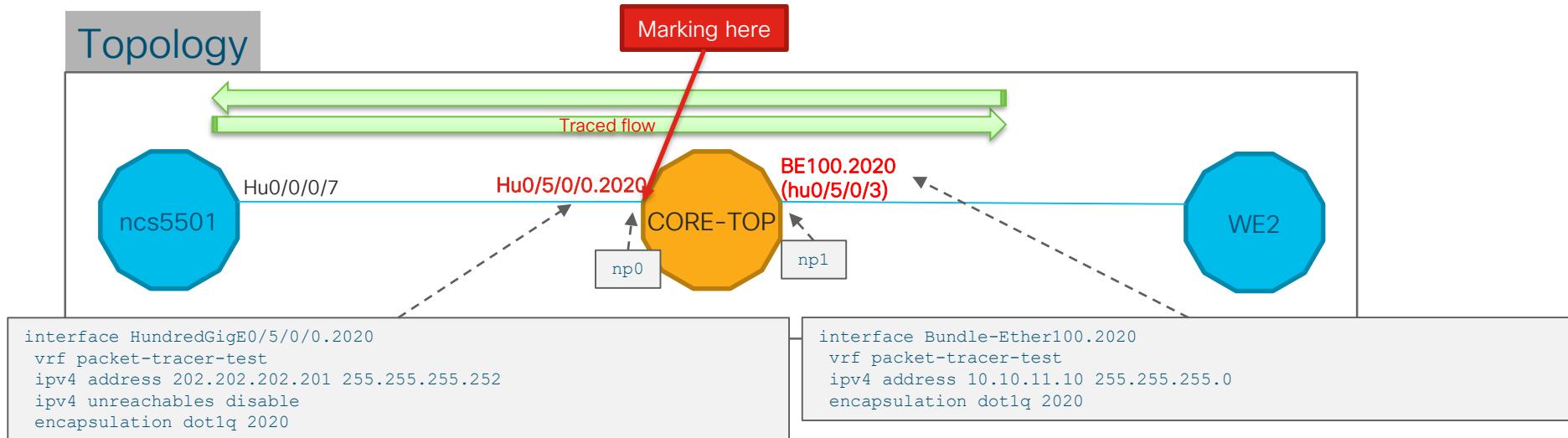
Packet Trace Example 6: TM or Pipeline Loopback

Pipeline and TM Loopback

- One of the most important limitations in XR release 7.0.1
- When NP microcode decides to send the packet into a loopback, the packet trace flag is lost on the loopback path.
- Pipeline loopback:
 - NP μcode creates a replica of the packet
 - Original packet is still traced, but the copy is not
 - Features: netflow, LI
- TM loopback:
 - No packet processing in the first pass; packet is sent directly to TM
 - Neither the original nor the copies can be traced (will be improved in 7.1.x to trace original packet)
 - Features: SPAN, multicast replication on NP

Pipeline and TM Loopback: Test Topology

- Test scenario: Bi-dir packet flow over a pair of L3 sub-ints



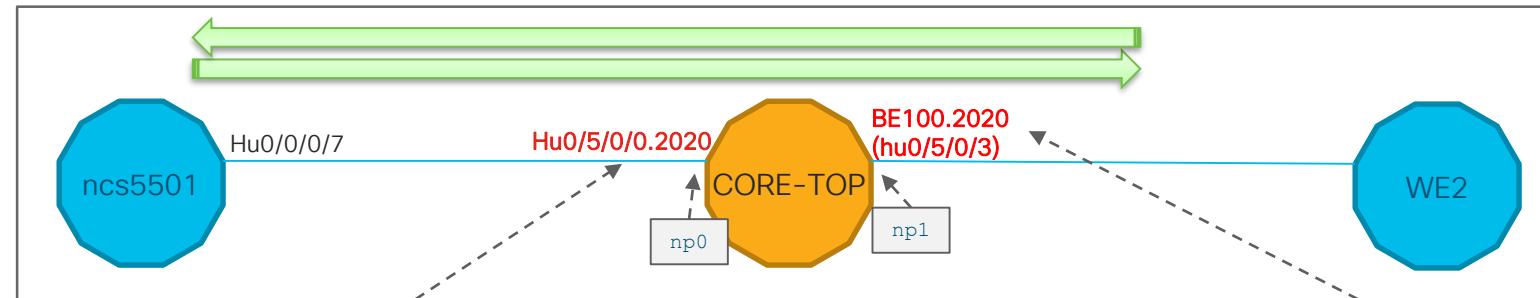
Pipeline and TM Loopback: Baseline captures

```
RP/0/RSP0/CPU0:CORE-TOP#sh controllers np counters np0 location 0/5/CPU0 | i "[0-9]{5}+[0-9]{3}|PIPE"
 17 MDF_TX_WIRE          63435112   935
 21 MDF_TX_FABRIC        2624266    926
 33 PARSE_FAB_RECEIVE_CNT 63397383   930
 37 PARSE_INTR_RECEIVE_CNT 4034605    519
 45 PARSE_ENET_RECEIVE_CNT 2464558    931
 549 MDF_PIPE_LPBK        502282     0
2130 ING_ICFD_QUEUE_PRI_1 2423371    925
2162 PKT_TRACE_ING_FROM_INTERFACE_21 2339607    925
2271 PKT_TO_FABRIC_TRACE 2149159    925
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#sh controllers np counters np1 location 0/5/CPU0 | i "[0-9]{5}+[0-9]{3}|PIPE"
 17 MDF_TX_WIRE          30625418   926
 21 MDF_TX_FABRIC        2340719    927
 33 PARSE_FAB_RECEIVE_CNT 30625203   926
 37 PARSE_INTR_RECEIVE_CNT 4017572    517
 45 PARSE_ENET_RECEIVE_CNT 2531965    927
 549 MDF_PIPE_LPBK        949370     0
2132 ING_ICFD_QUEUE_PRI_3 2339773    926
2270 PKT_FROM_FABRIC_TRACE 2149317    927
2293 PKT_TRACE_EGR_TO_INTERFACE_20 2149317    927
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#sh packet-trace results
Mon Aug 19 14:52:53.467 UTC
T: D - Drop counter; P - Pass counter
Location | Source | Counter           | T | Last-Attribute           |       | Count
-----+-----+-----+-----+-----+-----+-----+
0/5/CPU0 | NPO  | PACKET_MARKED      | P | HundredGigE0_5_0_0        |       | 5056
0/5/CPU0 | NPO  | PACKET_TO_FABRIC    | P |                           |       | 5057
0/5/CPU0 | NP1  | PACKET_FROM_FABRIC | P |                           |       | 5056
0/5/CPU0 | NP1  | PACKET_TO_INTERFACE | P | HundredGigE0_5_0_3        |       | 5055
RP/0/RSP0/CPU0:CORE-TOP#
```

The diagram illustrates the traffic flow through the Pipeline and TM Loopback architecture. It shows two main nodes: 'Ingress NP' and 'Egress NP'. Arrows indicate the flow of traffic from the Ingress NP through the NPs in the pipeline to the Egress NP. The NPs are represented by orange boxes labeled 'Ingress NP' and 'Egress NP'. The Ingress NP is connected to both NPs in the pipeline, and the Egress NP is connected to both NPs in the pipeline.

Pipeline Loopback: Apply Pipeline Loop Feature

- Enable netflow:



```
interface HundredGigE0/5/0/0.2020
vrf packet-tracer-test
ipv4 address 202.202.202.201 255.255.255.252
ipv4 unreachables disable
flow ipv4 monitor packet-trace-test sampler packet-trace-test ingress
encapsulation dot1q 2020
```

```
interface Bundle-Ether100.2020
service-policy input pkt-trace-qos-drop-test
vrf packet-tracer-test
ipv4 address 10.10.11.10 255.255.255.0
flow ipv4 monitor packet-trace-test sampler packet-trace-test egress
encapsulation dot1q 2020
```

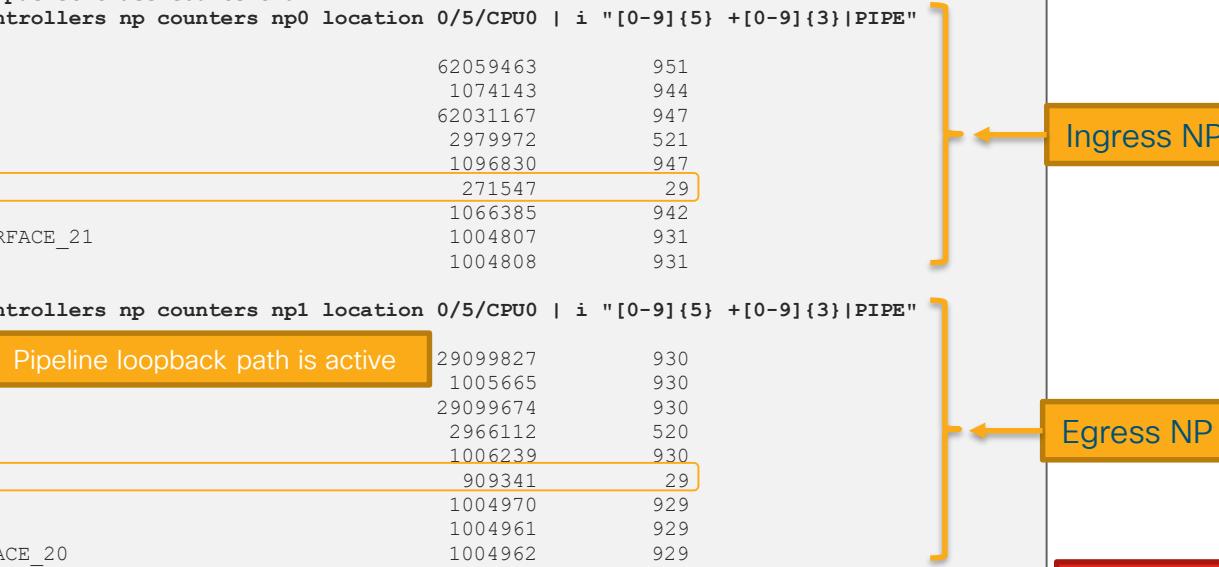
Pipeline Loopback: Impact on packet tracer

```

RP/0/RSP0/CPU0:CORE-TOP#clear packet-trace counters all
RP/0/RSP0/CPU0:CORE-TOP#sh controllers np counters np0 location 0/5/CPU0 | i "[0-9]{5}+[0-9]{3}|PIPE"
Mon Aug 19 14:19:05.702 UTC
 17 MDF_TX_WIRE                                62059463      951
 21 MDF_TX_FABRIC                             1074143       944
 33 PARSE_FAB_RECEIVE_CNT                     62031167      947
 37 PARSE_INTR_RECEIVE_CNT                   2979972       521
 45 PARSE_ENET_RECEIVE_CNT                  1096830       947
 549 MDF PIPE LPBK                            271547        29
2130 ING_ICFD_QUEUE_PRI_1                    1066385       942
2162 PKT_TRACE_ING_FROM_INTERFACE_21       1004807       931
2271 PKT_TO_FABRIC_TRACE                   1004808       931
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#sh controllers np1 location 0/5/CPU0 | i "[0-9]{5}+[0-9]{3}|PIPE"
Mon Aug 19 14:19:05.873 UTC
 17 MDF_TX_WIRE                                29099827      930
 21 MDF_TX_FABRIC                             1005665       930
 33 PARSE_FAB_RECEIVE_CNT                     29099674      930
 37 PARSE_INTR_RECEIVE_CNT                   2966112       520
 45 PARSE_ENET_RECEIVE_CNT                  1006239       930
 549 MDF PIPE LPBK                            909341        29
2132 ING_ICFD_QUEUE_PRI_3                    1004970       929
2270 PKT_FROM_FABRIC_TRACE                 1004961       929
2293 PKT_TRACE_EGR_TO_INTERFACE_20       1004962       929
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#sh packet-trace results
Mon Aug 19 14:19:06.028 UTC
T: D - Drop counter; P - Pass counter
Location | Source | Counter | T | Last-Attribute |
-----+-----+-----+-----+-----+
0/5/CPU0  NPO    PACKET_MARKED          P   HundredGigE0_5_0_0
0/5/CPU0  NPO    PACKET_TO_FABRIC       P
0/5/CPU0  NP1    PACKET_FROM_FABRIC     P
0/5/CPU0  NP1    PACKET_TO_INTERFACE    P   HundredGigE0_5_0_3
RP/0/RSP0/CPU0:CORE-TOP#

```

Pipeline loopback path is active



Outcome:
Original packets
are traced, copies
are not

TM Loopback: Apply TM Loopback feature

- Enable ingress SPAN on input interface:

```
interface HundredGigE0/5/0/0.2020
vrf packet-tracer-test
ipv4 address 202.202.202.201 255.255.255.252
ipv4 unreachables disable
monitor-session packet-tracer-test ethernet direction rx-only
encapsulation dot1q 2020
```

TM Loopback: Impact on packet tracer

```
RP/0/RSP0/CPU0:CORE-TOP#clear packet-trace counters all
RP/0/RSP0/CPU0:CORE-TOP#sh controllers np counters np0 location 0/5/CPU0 | i "[0-9]{5}+[0-9]{3}"
 17 MDF_TX_WIRE          63882788   949
 21 MDF_TX_FABRIC        3202444    1867
 33 PARSE_FAB_RECEIVE_CNT 63842862   944
 37 PARSE_INTR_RECEIVE_CNT 4279868    520
 45 PARSE_ENET_RECEIVE_CNT 2910383   944
 49 PARSE_TM_LOOP_RECEIVE_CNT 816228   1873
 53 PARSE_TOP_LOOP_RECEIVE_CNT 636570    927
 549 MDF_PIPE_LPBK        636575    927
 642 MDF_LB_TM1_LPBK0      675       927
 646 MDF_TM_LP_Q_PN       6705      927
 2071 PARSE_SPAN_LPBK      725       927
 2130 ING_ICFD_QUEUE_PRI_1 2866696   939
 2162 PKT_TRACE_ING_FROM_INTERFACE_21 2777707   927
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#sh controllers np1 location 0/5/CPU0 | i "[0-9]{5}+[0-9]{3}"
 17 MDF_TX_WIRE          31198038   1854
 21 MDF_TX_FABRIC        2778902    927
 33 PARSE_FAB_RECEIVE_CNT 31197809   1854
 37 PARSE_INTR_RECEIVE_CNT 4261985    517
 45 PARSE_ENET_RECEIVE_CNT 3104616    1854
 169 RSV_DROP_ACL_DENY    324892     927
 2130 ING_ICFD_QUEUE_PRI_1 324874     927
 2132 ING_ICFD_QUEUE_PRI_3 2777900    927
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#sh packet-trace results
T: D - Drop counter; P - Pass counter
Location | Source | Counter           | T | Last-Attribute           |           | Count
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
0/5/CPU0  NPO    PACKET_MARKED      P  HundredGigE0_5_0_0           124694
RP/0/RSP0/CPU0:CORE-TOP#
```

TM loopback path is active

Ingress NP

Egress NP

Outcome:
Neither the nor the replicated packets are traced

There's "noise" in np1 counters because the SPAN destination is also on this NP. It's an int with loopback internal and ethernet-services ACL that drops all traffic

PACKET_MARKED is the last packet-trace counter we can see
(resolved in 7.1.2 → original packet is traced)

TM Loopback: Apply TM Loopback feature

- Disable ingress SPAN on input interface
- Enable egress SPAN on output interface

```
interface HundredGigE0/5/0/0.2020
vrf packet-tracer-test
ipv4 address 202.202.202.201 255.255.255.252
ipv4 unreachables disable
no monitor-session packet-tracer-test ethernet direction rx-only
encapsulation dot1q 2020
interface Bundle-Ether100.2020
vrf packet-tracer-test
ipv4 address 10.10.11.10 255.255.255.0
monitor-session packet-tracer-test ethernet direction tx-only
encapsulation dot1q 2020
```

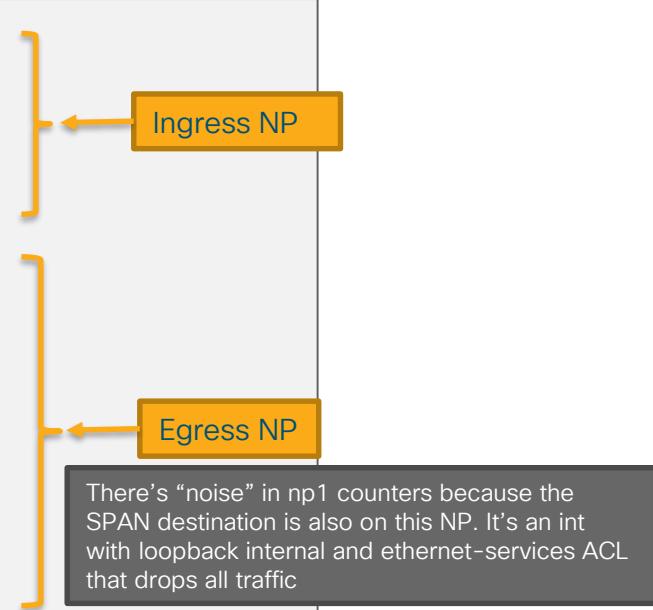
Packet Trace Condition

```

RP/0/RSP0/CPU0:CORE-TOP#clear packet-trace counters all
RP/0/RSP0/CPU0:CORE-TOP#sh controllers np counters np0 location 0/5/CPU0 | i "[0-9]{5}+[0-9]{3}"
  17 MDF_TX_WIRE                                64123883      932
  21 MDF_TX_FABRIC                             3627005       922
  33 PARSE_FAB_RECEIVE_CNT                     64082414      927
  37 PARSE_INTR_RECEIVE_CNT                   4452572       519
  45 PARSE_ENET_RECEIVE_CNT                  3149880       927
2130 ING_ICFD_QUEUEB_PRI_1                  3104431       921
2162 PKT_TRACE_ING_FROM_INTERFACE_21        3012129       921
2271 PKT_TO_FABRIC_TRACE                    2501097       921
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#sh controllers np counters np1 location 0/5/CPU0 | i "[0-9]{5}+[0-9]{3}"
Mon Aug 19 15:06:18.606 UTC
  17 MDF_TX_WIRE                                31664885      1843
  21 MDF_TX_FABRIC                             3059537       1843
  33 PARSE_FAB_RECEIVE_CNT                     31664649      1843
  37 PARSE_INTR_RECEIVE_CNT                   4434060       522
  45 PARSE_ENET_RECEIVE_CNT                  3571486       1843
  49 PARSE_TM_LOOP_RECEIVE_CNT                257854        1862
  53 PARSE_TOP_LOOP_RECEIVE_CNT              995570        922
169 RSV_DROP_ACL_DENY                         557293       922
433 RSV_EGR_SPAN                               46204        922
549 MDF_PIPE_LPBK                            995575       922
643 MDF_LB_TM1_LPBK1                          36463        718
646 MDF_TM_LP_Q_PN                           46193        921
2071 PARSE_SPAN_LPBK                          46219        922
2132 ING_ICFD_QUEUEB_PRI_3                  3058500      1843
2270 PKT_FROM_FABRIC_TRACE                 2501252       921
RP/0/RSP0/CPU0:CORE-TOP#
RP/0/RSP0/CPU0:CORE-TOP#sh packet-trace results
Mon Aug 19 15:06:18.768 UTC
T: D - Drop counter; P - Pass counter
Location | Source | Counter
----- | ----- | -----
0/5/CPU0  NPO    PACKET_MARKED             P   HundredGigE0_5_0_0
0/5/CPU0  NPO    PACKET_TO_FABRIC          P
0/5/CPU0  NP1    PACKET_FROM_FABRIC        P
RP/0/RSP0/CPU0:CORE-TOP#

```

TM loopback path is active



Outcome:
Neither the nor the replicated packets are traced

PACKET FROM FABRIC is the last packet-trace counter we can see (resolved in 7.1.2 → original packet is traced)

Packet Trace Restrictions And Limitations

Packet Tracer Restrictions and Limitations

- Packet tracer framework is available on all IOS XR platforms, but platform onboarding is executed in phases.
- Implementing packet tracer on a distributed forwarding architecture, with minimal performance impact was a challenge.
- Packet tracer may not always provide the exact reason for a packet drops, but it will at minimum show where were the packets seen last, therefore significantly reducing the need for additional troubleshooting.

Architectural Decisions

- Only one set of conditions can be active at a time on the router, applied to multiple interfaces (i.e. multiple marking modules of the same type).
- ➔ Only one flow can be traced at a time, but this flow can be traced on multiple locations.
- If distinct conditions sets would be allowed, user wouldn't be able to conclude from the packet trace results which counter increment pertains to which flow.
- Packet trace condition cannot be modified while tracing is active.

Marking in XR 7.0.1 – PI Restrictions

- Packet trace framework is not resolving logical interfaces. When tracing on a Bundle-Ether or PW-Ether interface, user must explicitly enable tracing on all member interfaces.

Marking in XR 7.0.1 - ASR9k PD Restrictions

- Packet marking and counting is supported only on ASR9000 Tomahawk line cards. (Lightspeed support n 7.1.2)
- Condition check and marking is performed at port level, before any other feature is executed on the packet.
 - Condition specification must take into account the dot1q or QinQ encapsulation.
- ASR9000 Tomahawk NP supports three offset/value/mask triplets, where the maximum length of each value and mask is 4 bytes.
- Marking disables the racetrack path through the NP microcode on interfaces specified in the packet-trace condition.
- Packets that looped back through NP are lost for tracing because the packet trace flag is not preserved on loopback path. (Improved in 7.1.2)
 - Pipeline loopback: original packet can still be traced
 - TM loopback: neither the original nor the copies are can be traced
- Not all L2 topologies support packet tracing. (Improved in 7.1.2)

Packet Trace Performance Impact

Packet Tracer Performance Impact - Lightspeed

Packet Tracer Disabled	<pre>RP/0/RSP0/CPU0:ios#show controllers np load np0 location 0/0/CPU0 Tue Jan 7 17:57:53.997 UTC Node: 0/0/CPU0: ----- Load Packet Rate NP0: 10% utilization 69846780 pps</pre>
Packet Tracer Enabled	<pre>RP/0/RSP0/CPU0:ios#show controllers np load np0 location 0/0/CPU0 Tue Jan 7 17:58:15.847 UTC Node: 0/0/CPU0: ----- Load Packet Rate NP0: 12% utilization 69816120 pps</pre>

Packet Tracer Performance Impact - Tomahawk

Packet Tracer Disabled	RP/0/RSP0/CPU0:ios#show controllers np load np0 location 0/1/CPU0 Node: 0/1/CPU0: ----- Load Packet Rate NPO: 19% utilization 44071296 pps
Packet Tracer Enabled	RP/0/RSP0/CPU0:ios#show controllers np load np0 location 0/1/CPU0 Node: 0/1/CPU0: ----- Load Packet Rate NPO: 62% utilization 44071480 pps

Monitoring the system (Automation, PAM, MDT)

PAM

Platform Automation Manager

PAM Overview

PAM? (Platform Automated Monitoring) - An on-box tool to monitor events and collect data automatically

Monitoring Agents

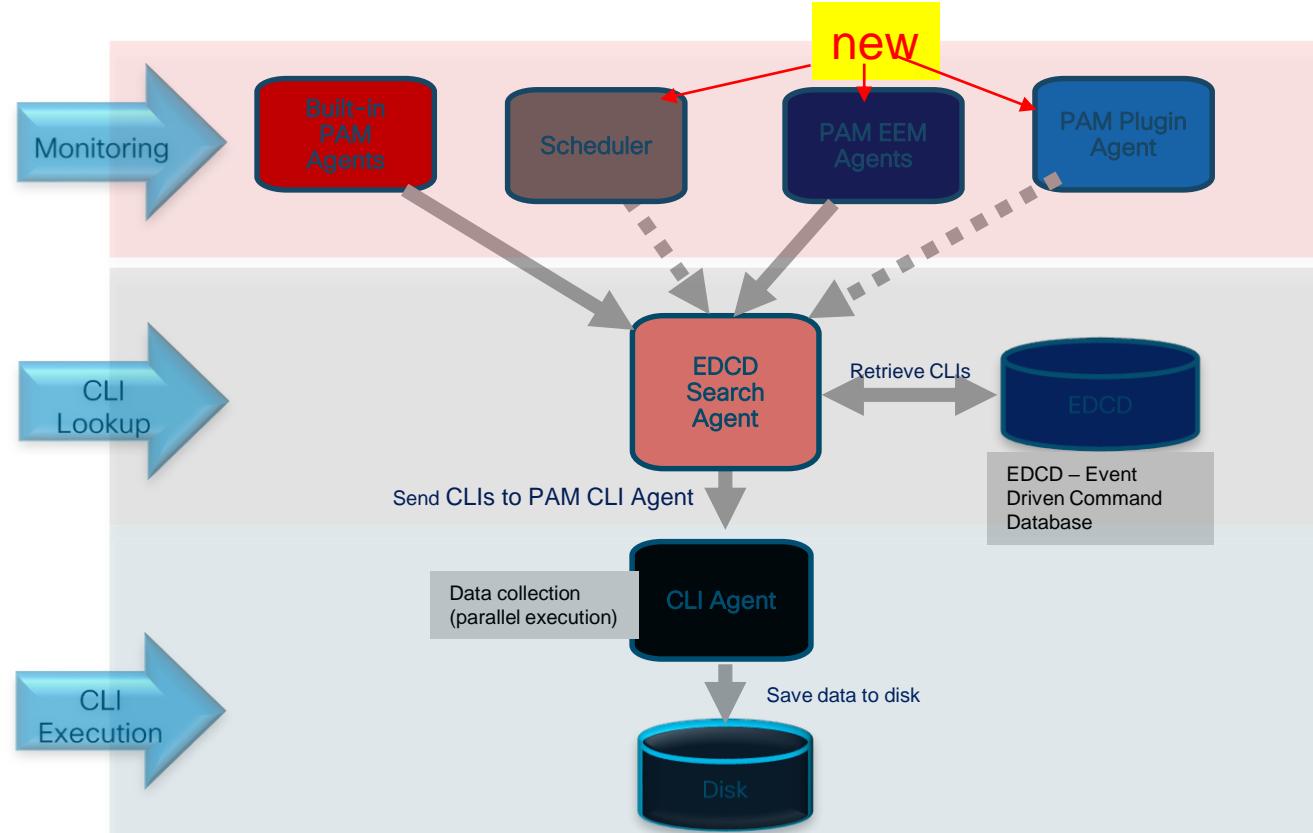
- **Built-in Agents** (on all the times)
Monitoring Crash/Tracebacks/CPU hogs/Memory leaks/Syslog
- **PAM Scheduler** – run jobs (CLI or scripts) periodically (namely, an XR cron agent)
- **PAM EEM Agent** – monitor syslog in real-time and trigger actions/data collection
- **PAM Plugin Agent** – spawn user-defined daemon dynamically

Data Collection Engine

- **EDCD (Event Driven CLI Database)**
- **CLI Agent** (i.e., for performing data collection)

API based/automation friendly/minimize runtime/scalable.

PAM-OBA Workflow



EDCD Ondemand Overview

EDCD is a database with each **Identifier** mapped to a list of CLIs (or scripts).

- Create an ondemand DB entry:

```
RP/0/RP0/CPU0:ios#edcd ondemand ?
add-update      Add or update ondemand EDCD entries
add-update-trigger Add or update ondemand EDCD entries, and then execute CLIs
delete         Delete ondemand EDCD entries
delete-all     Delete all EDCD entries
trigger        Trigger the collection of traces associated with given identifier
```

- Display ondemand DB entries:

```
RP/0/RP0/CPU0:ios#show edcd ondemand database ?
identifier  Select an identifier (if missing, all database entries will be displayed)
|          Output Modifiers
<cr>
```

```
RP/0/RP0/CPU0:ios#show edcd ondemand database
```

EDCD Ondemand – Create

- Create an ondemand DB entry:

```
RP/0/RP0/CPU0:ios#edcd ondemand add-update identifier schedule-1 commands \
    "show segment-routing traffic-eng ipv4 topology; \
     show segment-routing traffic-eng ipv6 topology
      show tech-support mpls traffic-eng"
```

- Verify DB entry:

```
RP/0/RP0/CPU0:ios#show edcd ondemand database identifier schedule-1
=====
Identifier: schedule-1
=====
1: show segment-routing traffic-eng ipv4 topology
2: show segment-routing traffic-eng ipv6 topology
3: show tech-support mpls traffic-eng
-----
RP/0/RP0/CPU0:ios#
```

EDCD Ondemand – Create (cont.)

- Verifying CLI – All CLI are verified before adding to EDCD. If invalid, the CLI will be rejected.

```
RP/0/RP0/CPU0:ios#edcd ondemand add-update identifier schedule-1 commands \
    "show segment-routing traffic-eng ipv4 topology; \
     show segment-routing traffic-eng ipv6 topology
      show tech-support traffic-eng"
```

Rejecting command: **show tech-support traffic-eng**

- Only XR CLIs are verified. Neither admin CLI, nor shell CLI are verified.
 - User need to verify them manually
- Duplicated CLIs will be removed.

EDCD Ondemand - Delete

- Two ways to delete

```
RP/0/RP0/CPU0:ios#edcd ondemand ?
  delete          Delete ondemand EDCD entries
  delete-all      Delete all EDCD entries
```

```
RP/0/RP0/CPU0:ios#edcd ondemand delete identifier admin-1 ?
  commands  Specify a list of commands that to be deleted
            (if missing all entries under this sub-pattern will be deleted)
<cr>
```

```
RP/0/RP0/CPU0:ios#edcd ondemand delete identifier admin-1 ?
  commands  Specify a list of commands that to be deleted
            (if missing all entries under this sub-pattern will be deleted)
```

- delete a single CLI

```
RP/0/RP0/CPU0:ios#edcd ondemand delete identifier admin-1 commands "show version"
```

Ondemand EDCD has been updated (execute 'show edcd ondemand database' to verify.)

- delete all CLIs for one identifier

```
RP/0/RP0/CPU0:ios#edcd ondemand delete identifier admin-1
```

Ondemand EDCD has been updated (execute 'show edcd ondemand database' to verify.)

EDCD Ondemand – Admin CLI



- Admin CLIs are supported. Admin CLI must be started with keyword admin. E.g.,

(Note: **admin** command cannot be verified)

```
RP/0/RP0/CPU0:ios#edcd ondemand add-update identifier admin-1 commands \
```

```
    "admin show tech-support install; admin show sdr; show version"
```

```
RP/0/RP0/CPU0:ios#show edcd ondemand database identifier admin-1
```

```
=====
Identifier: admin-1
=====
```

```
1: admin show tech-support install
```

```
2: admin show sdr
```

```
3: show version
```

EDCD Ondemand – Shell command

- Shell commands/scripts are supported, must be started with keyword **run**

(Note: shell command cannot be verified)

```
RP/0/RP0/CPU0:ios#edcd ondemand add-update identifier shell-1 commands \
    "run /pkg/bin/show_platform_sysdb -v; show platform"
```

```
RP/0/RP0/CPU0:ios#show edcd ondemand database identifier shell-1
```

```
=====
```

```
Identifier: shell-1
```

```
=====
```

```
1: run <ssh> "/pkg/bin/show_platform_sysdb -v"
```

```
2: show platform
```

```
-----
```

EDCD Ondemand – Trigger

```
RP/0/RP0/CPU0:ios#edcd ondemand trigger admin-1?  
  add-update-trigger  Add or update ondemand EDCD entries, and then execute CLIs  
  trigger            Trigger the collection of traces associated with given identifier
```

➤ Trigger w/o updating

```
RP/0/RP0/CPU0:ios#edcd ondemand trigger identifier triage-1
```

➤ Update and Trigger

```
RP/0/RP0/CPU0:ios#edcd ondemand add-update-trigger identifier triage-1 \  
  commands "admin show disk_status"
```

➤ Benefits:

- ❑ Reduce runtime - all show tech CLIs are executed in parallel.
- ❑ Simplify operations (both manual and automation)
 - E.g., “edcd ondemand trigger identifier triage-1” vs calling 20+ CLIs
- ❑ PAM tgz files will be automatically copied outside of the device (with PAM2.0 enabled)
 - Note: EDCD data is persistent cross reload/reimage (repeat testing DT images: DT1, DT2, DT3 ...)

EDCD Ondemand – Verify

- Location of EDCD logs: /harddisk:/cisco_support/
- File format:
 - PAM-<platform>-ondemand-xr-<identifier>-<tag>-<timestamp>.tgz

where tag is optional.

e.g., ls /harddisk:/cisco_support/PAM*

- PAM-xrv9k-ondemand-xr-triage1-2019Sep17-125851.tgz

- show edcd ondemand collection-status (useful when EDCD contains show tech)
 - RP/0/RP0/CPU0:ios#edcd ondemand trigger id showtech
 - RP/0/RP0/CPU0:ios#show edcd ondemand collection-status
 - /misc/disk1/cisco_support/showtech/ondemand-gsp-showtech_output-2019Sep17-170133.txt
 - /misc/disk1/cisco_support/showtech/ondemand-install-showtech_output-2019Sep17-170140.txt
 - Show Techs still getting collected: 2

EDCD Schedule Overview

- Allow to run CLIs/scripts periodically (simply data collection or monitoring).

```
RP/0/RP0/CPU0:ios#edcd scheduler ?
add-update  Add or update a job
delete      Delete a job
delete-all   Delete all jobs

RP/0/RP0/CPU0:ios#edcd scheduler add-update cadence '*/10 * * * *' ?
command     Command to be executed at the above cadence
identifier   An identifier linked to a list of CLIs (defined in ondemand EDCD)
<cr>
```

- Display scheduler entries

```
RP/0/RP0/CPU0:ios#show edcd scheduler
<Job ID>: <job content>
1: */30 * * * * root /pkg/bin/pam_is_active_rp && /pkg/bin/edcd_cli.py ondemand --operation trigger -i schedule-1
RP/0/RP0/CPU0:ios#
```

EDCD Scheduler – Add job (shell command)



```
RP/0/RP0/CPU0:ios#edcd scheduler add-update cadence '*/10 * * * *' ?
```

command Command to be executed at the above cadence

identifier An identifier linked to a list of CLIs (defined in ondemand EDCD)

<cr>

e.g., collect show memory summary every 10 minute:

```
RP/0/RP0/CPU0:ios#edcd scheduler add-update cadence '*/10 * * * *' \
  command "show_memory_ng -n 4288 -s >> /harddisk:/show_memory.log"
```

Every 10 minute

Use single quotes

Output saved to this file

Use double quotes

EDCD Scheduler - Add job (CLIs via EDCD)



Schedule jobs to execute CLIs defined in EDCD

```
RP/0/RP0/CPU0:ios#show edcd ondemand database identifier schedule-1
=====
Identifier: schedule-1
=====
1: show segment-routing traffic-eng ipv4 topology
2: show segment-routing traffic-eng ipv6 topology
3: show tech-support mpls traffic-eng
-----
```

E.g., collect above CLIs every 20 minute:

```
RP/0/RP0/CPU0:ios#edcd scheduler add-update cadence '*'/20 * * * '*' \
identifier schedule-1
```

Note: the data will be saved under /harddisk:/cisco_support/PAM*tgz

Single or double quotes



EDCD Scheduler – Delete jobs

Delete a single job

```
RP/0/RP0/CPU0:ios#edcd scheduler delete job-id 1
```

Delete all jobs

```
RP/0/RP0/CPU0:ios#edcd scheduler delete-all
```

PAM EEM Overview

Enable to monitor syslog and trigger actions in real-time.

- Sample EEM XR config:

```
RP/0/RP0/CPU0:ios(config) #event manager pam-agent run  
RP/0/RP0/CPU0:ios(config) #event manager pam-agent register syslog <keyword>  
RP/0/RP0/CPU0:ios(config) #commit
```

Syslog keyword

- EDCD EEM syntax

```
RP/0/RP0/CPU0:ios#edcd eem add-update keyword <keyword> sub-pattern <sub-  
pattern> identifier <identify> tag <tag> throttle-window <minutes>
```

Sub-pattern to only match
<sub-pattern>
(note: <keyword> will
match any process name)

To trigger all CLIs
defined under
<identifier>

tag is to help identify
PAM logs (i.e., <tag>
will be part of PAM
*.tgz file). Optional

Repeated event is
ignored if occurred
within <minutes>
minutes. Optional

- EDCD EEM Sample

```
RP/0/RP0/CPU0:ios#edcd eem add-update keyword OS-SYSMGR-4-PROC_RESTART_NAME sub-pattern bgp  
identifier bgp-test tag bgp-edcd throttle-window 60
```

PAM EEM Sample

```
RP/0/RP0/CPU0:ios#show edcd ondemand database identifier bgp-test
=====
Identifier: bgp-test
=====
1: show platform
2: show install active
3: show health gsp
4: show health sysdb
5: show health cfgmgr
6: show tech routing bgp
7: show tech gsp
=====
```

Identifier name

```
RP/0/RP0/CPU0:ios#show edcd eem database
=====
Keyword: 'OS-SYSMGR-4-PROC_RESTART_NAME'
=====
Subpattern: 'bgp':
=====
Commands:
  show platform
  show install active
  show health gsp
  show health sysdb
  show health cfgmgr
  show tech routing bgp
  show tech gsp
identifier: bgp-test
tag: bgp-edcd
throttle_window (minutes): 60
do_async: False
```

Syslog
keyword

```
RP/0/RP0/CPU0:ios#show running-config | inc event
event manager pam-agent run
event manager pam-agent register syslog OS-SYSMGR-4-PROC_RESTART_NAME 1
```

PAM EEM - Trigger Action

➤ Trigger EEM Action:

```
RP/0/RP0/CPU0:ios#process restart bgp    <<< this will trigger EEM action
```

However, the following process restart won't trigger since sub-pattern doesn't match `bgp`:

```
RP/0/RP0/CPU0:ios#process restart ipv4_io
```

➤ Verify EEM Execution Status:

```
RP/0/RP0/CPU0:ios#show edcd eem logs
```

```
RP/0/RP0/CPU0:ios#show logging | inc PAM
```

```
RP/0/RP0/CPU0:ios#run ls /harddisk:/cisco_support/PAM*
```

Or

```
RP/0/RP0/CPU0:ios#dir harddisk:/cisco_support/PAM*
```

PAM EEM - Additional Options

```
RP/0/RP0/CPU0:ios#edcd eem add-update keyword <keyword> sub-pattern default  
identifier test1 ?
```

parallel-execution Allow to execute all commands in parallel (by default only shot-tech CLIs are executed in parallel), optional

tag Specify a tag name which will be part of log file (tgz), i.e. to help identify the logs. Optional

throttle-window Throttle window (minutes) within which the same commands can only be executed once. Optional

<cr>

PAM Plugin Agent

- Allow to run daemons (or processes cross reload automatically).
- Use case scenarios
 - Collect data constantly
 - Monitor events constantly
 - Run triggers repeatedly
- Wiki: https://apps.na.collabserv.com/wikis/home?lang=en-us#/wiki/W9c1adb76657b_4e91_925d_df6d5939390f/page/How%20to%20add%20PAM%20plugin%20daemon%20processes
- Usage
 - Three positional arguments: **add**, **remove**, and **show**
 - [xr-vm_node0_RP0_CPU0:~]\$ pam_plugin_cli -h
 - usage: pam_plugin_cli [-h] [-d] {add,remove,show} ...
 - positional arguments:
 - {add,remove,show}
 - add add plugin (daemon process)
 - remove remove plugin (daemon processes)
 - show show plugins
- Submode
- [xr-vm_node0_RP0_CPU0:~]\$ pam_plugin_cli <add|remove|show> -h

PAM Plugin Agent – Samples

- Allow to run daemons (or processes) cross reload automatically)

Setup (add) a new daemon

```
pam_plugin_cli add -s <full path of the daemon process>
```

e.g.,

```
[node0_RP0_CPU0:~]$ pam_plugin_cli add -s /harddisk:/check_routes.pl
```

```
check_routes.pl has been added to the pool, and is being processed. Please run  
pam_plugin_cli show --status, and pam_plugin_cli show --logs
```

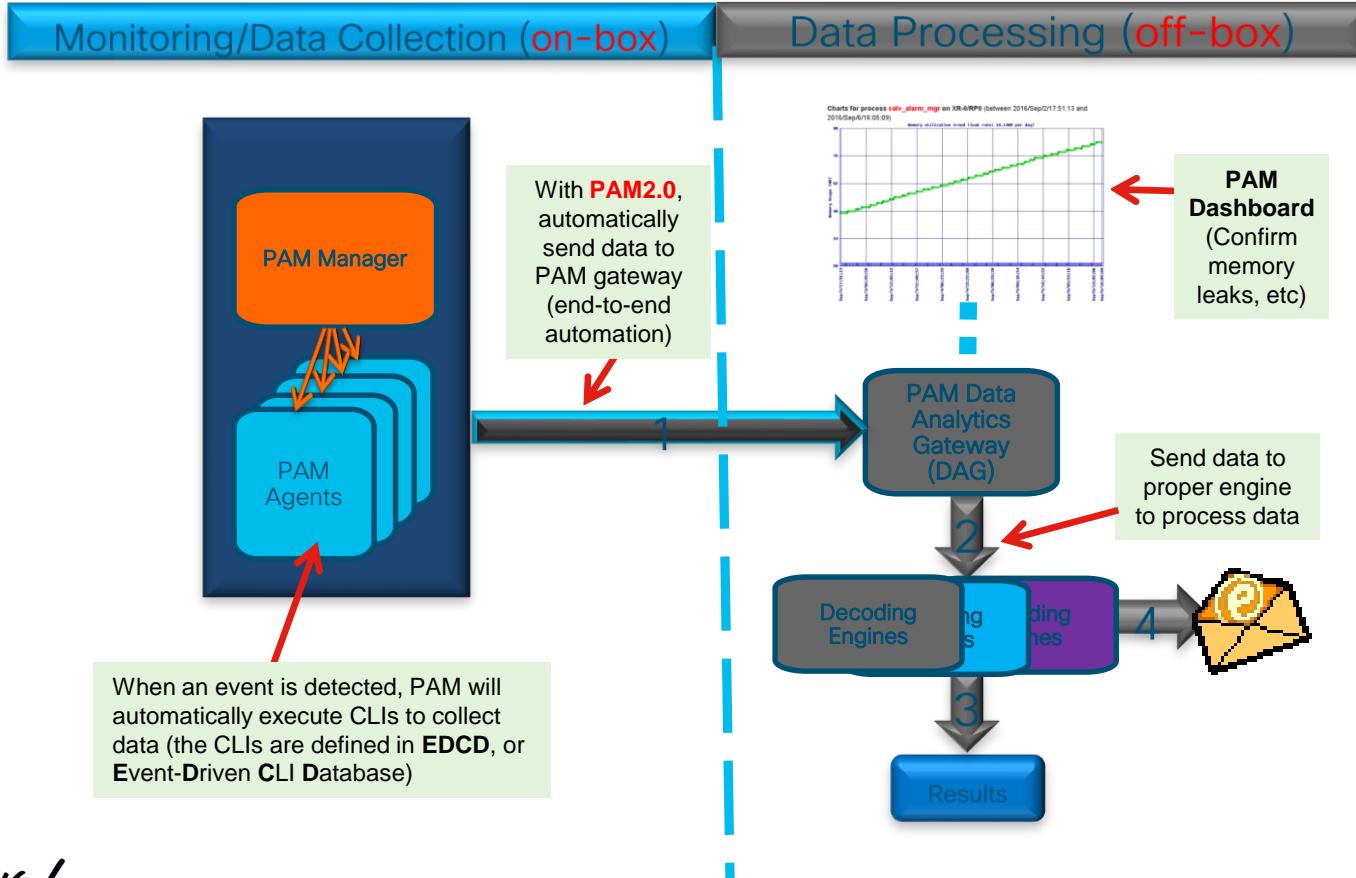
Display status

```
[xr-vm_node0_RP0_CPU0:~]$pam_plugin_cli show -A  
There are 2 plugin(s) active.  
check_routes.pl is running. pid: ['15758']  
show_sdr is running. pid: ['27386']
```

Display logs (history)

```
[xr-vm_node0_RP0_CPU0:~]$pam_plugin_cli show -L
```

PAM Overall Workflow (devtest)



MDT

Model Driven Telemetry

Fundamentals of Modern Telemetry



Push Not Pull

Performance



Analytics-Ready Data

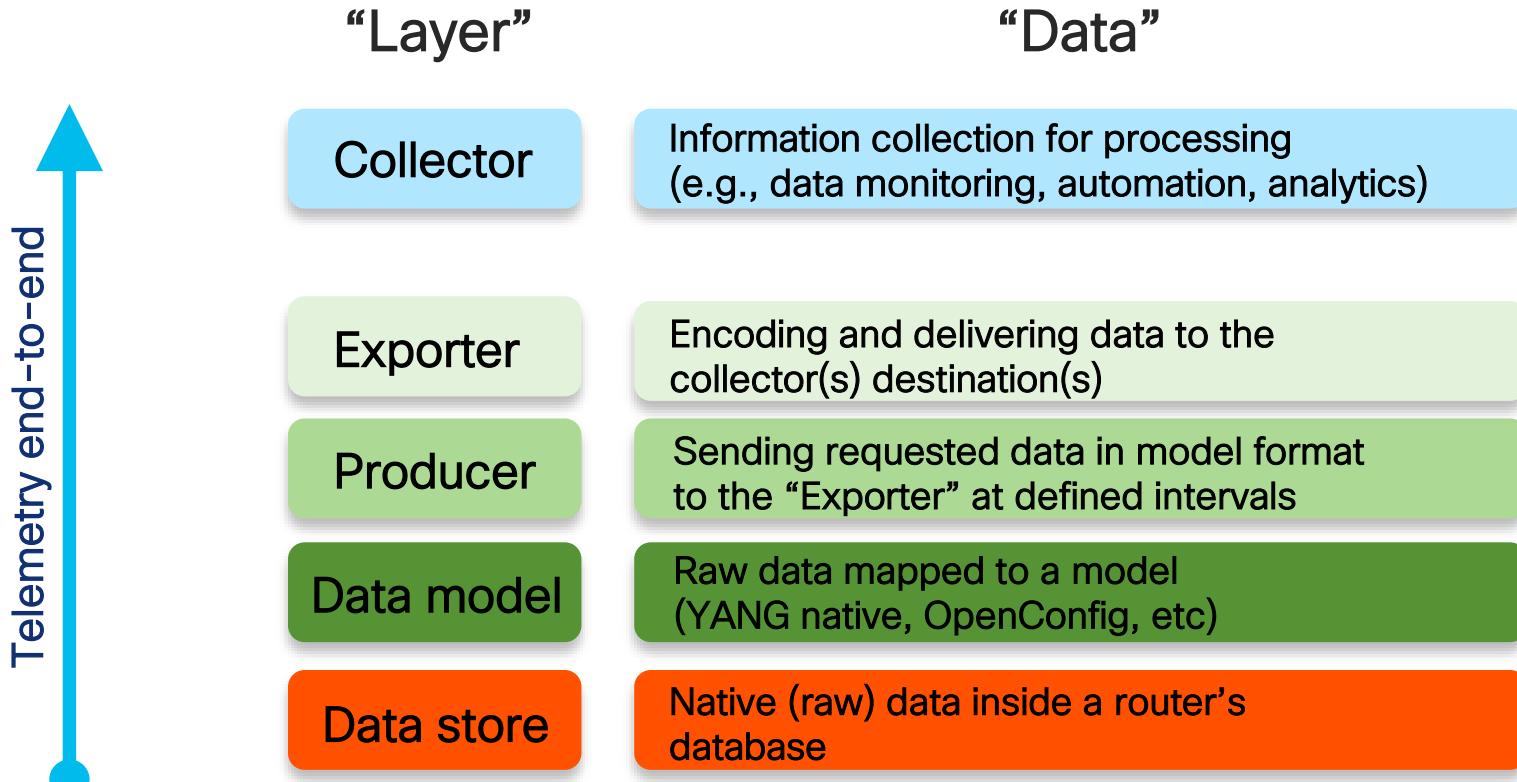
Tool-Chains



Data-Model Driven

Automation

“OSI model” of Telemetry



Step 1: define what you want to stream

Define the data you want to stream.

Hint: see in couple of slides how to find what you want

Data model

Raw data mapped to a model
(YANG native, OpenConfig, etc)

telemetry model-driven

sensor-group SGROUP1

YANG Model

sensor-path Cisco-IOS-XR-infra-statsd-oper:infra-

statistics/interfaces/interface/latest/generic-counters

subtree path

Step 2: define where and how

Exporter

Encoding and delivering data to the collector(s) destination(s)

Define the destination(s), encoding and transport protocols

Hint: you can have several destinations under a single destination-group, or several destination-groups in your config.

Another hint: Port/Encoding/Transport must be the same on the receiver side ☺

telemetry model-driven

destination-group DGROUP

address family ipv4 192.168.1.1 port 2104

----- and/or -----

address family ipv6 2001:db8::1 port 2104

encoding self-describing-gpb

protocol tcp

Specify where you want to send your data

Specify how you want your data to look like

Specify how you want your data to be delivered

Step 3: tie everything together

Producer

Sending requested data in model format
to the "Exporter" at defined intervals

The final step is to tie all together.

Hint: try to create separate subscriptions for each sensor-path, it will enhance your telemetry experience. See in multithreading section of chapter two (p. 62)

telemetry model-driven

subscription Sub1

sensor-group-id SGROUP1 sample-interval 30000

= 30 seconds

destination-id DGROUP1

There is no “gold” rule about minimum sample-interval. It depends on collection and encoding time, as you need to have “sample interval” bigger than “Collection+Encoding”. For most cases it is less than 10s or 5s. Sometimes collection plus encoding can be as long as 100s. More about this later in this deck.

If you don't feel comfortable talking about this with your customer, reach out to us.

CISCO Live!

How to find the data you want to stream?

All the information is formatted using YANG models. Don't stream everything, find what you need inside the model and configure the full path for your telemetry. Use [pyang](#) or [yang-explorer](#) for this, it shows any model in a nice way.

Hint: mostly a manual process so far, we're thinking about how to automate this. Meantime, we work on a case-by-case basis or you can use ANX tool (see p. 160)

```
$ pyang -f tree Cisco-IOS-XR-infra-statsd-oper.yang --tree-path infra-statistics/interfaces/interface/latest/generic-counters
```

```
module: Cisco-IOS-XR-infra-statsd-oper
  +-ro infra-statistics
    +-ro interfaces
      +-ro interface* [interface-name]
        +-ro latest
          +-ro generic-counters
            +-ro packets-received?          uint64
            +-ro bytes-received?           uint64
            +-ro packets-sent?             uint64
            +-ro bytes-sent?               uint64
            +-ro multicast-packets-received? uint64
            +-ro broadcast-packets-received? uint64
<output snipped for brevity>
```

You can start using pyang as is, without “--tree-path” to see all paths. This one just gives more exact view

You can see what is inside the path (this info will be streamed)

Where to find those models?

A reasonable question. There are two ways you should remember (there are more, these two are the easiest). The first one, go here: <https://github.com/YangModels/yang/tree/master/vendor/cisco/xr> and get models for the release you want.

Hint: it contains _all_ models for the release. Some might work on ASR9k or NCS55xx only.

YangModels / yang

Code Issues 20 Pull requests 1 Projects 0 Wiki Insights

Branch: master yang / vendor / cisco / xr / 631 /

einarnn First commit of IOS-XR 6.3.1 models.

Cisco-IOS-XR-aaa-localsd-cfg.incompatible First commit of IOS-XR 6.3.1 models.

Cisco-IOS-XR-plat-chas-invmgr-oper.incompatible First commit of IOS-XR 6.3.1 models.

Cisco-IOS-XR-Ethernet-SPAN-cfg.yang First commit of IOS-XR 6.3.1 models.

Cisco-IOS-XR-Ethernet-SPAN-datatype.yang First commit of IOS-XR 6.3.1 models.

Cisco-IOS-XR-Ethernet-SPAN-oper-sub1.yang First commit of IOS-XR 6.3.1 models.

Cisco-IOS-XR-Ethernet-SPAN-oper-sub2.yang First commit of IOS-XR 6.3.1 models.

Cisco-IOS-XR-Ethernet-SPAN-oper-sub3.yang First commit of IOS-XR 6.3.1 models.

Cisco-IOS-XR-Ethernet-SPAN-oper.yang First commit of IOS-XR 6.3.1 models.

- **Telemetry only cares about operational (*-oper.yang) models. (not *-cfg.yang, not *-act.yang)**
- 143 oper YANG models published for XR 6.1.1
- 151 oper YANG are for XR 6.1.2
- 177 oper YANG are for XR 6.2.1
- 180 oper YANG for XR 6.2.2
- 198 oper YANG for XR 6.3.1

Use pyang to see the content of each model. Compare names in the command you want to find the model for and the output of the YANG model. Later on, an auto tool will be created for a more convenient way.

Openconfig models for telemetry

```
VOSIPCHU-M-C1GV:631 vosipchu$ pyang -f tree openconfig-platform.yang
module: openconfig-platform
  +-rw components
    +-rw component* [name]
      +-rw name?      string
      +-rw config
        +-rw name?    string
        +-rw type?    union
        +-rw id?      string
        +-rw description? string
        +-rw mfg-name? string
        +-rw version?  string
        +-rw serial-no? string
        +-rw part-no?  string
      +-ro state
        +-ro name?    string
        +-ro type?    union
        +-ro id?      string
        +-ro description? string
        +-ro mfg-name? string
        +-ro version?  string
        +-ro serial-no? string
        +-ro part-no?  string
      +-ro properties
        +-rw property* [name]
          +-rw name      -> ../config/name
          +-rw config
            +-rw name?    string
            +-rw value?   union
            +-rw configurable? boolean
          +-ro state
            +-ro name?    string
            +-ro value?   union
            +-ro configurable? boolean
      +-rw subcomponents
        +-rw subcomponent* [name]
          +-rw name      -> ../config/name
          +-rw config
            +-rw name?    -> ../../../../../../component/config/name
          +-ro state
            +-ro name?    -> ../../../../../../component/config/name
augment /oc-if:interfaces/oc-if:interface/oc-if:state:
  +-ro hardware-port? -> /components/component/name
```

Applied config operational data

Derived state operational data

With OpenConfig you don't have separate models for Configuration and Monitoring (oper data). Everything is contained within a single model:

- Configuration is under "config" section.
- Operational data is under "state" section.

Under "state" section you can find two types of operational data:

- Applied config (similar to "sh run XXX")
- Derived state (that's the real time oper data, like stats).

So, if you plan to use OpenConfig models in your demo, be sure to find needed paths and check them.

Models supported on your router

Another option is to get models supported on your exact router with its running release. Send “Netconf Hello” to it and the router will reply with all its capabilities plus all the supported models.

```
webserver$ ssh cisco@10.30.111.9 -p 830 -s netconf
```

```
cisco@10.30.111.9's password:
```

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
<capability>http://cisco.com/ns/yang/Cisco-IOS-XR-infra-statsd-oper?module=Cisco-
IOS-XR-infra-statsd-oper&revision=2015-11-09</capability>
```

NETCONF <Hello>
Capabilities Lists
All Supported
YANG Models

<...further output omitted for brevity...>

You will see a long list of models supported on the platform

How to understand which model you need to use: finding XML schema

This is a manual and not very easy process so far (yes, there is some work to make this process easier, but no dates yet).

The first step is to understand which “show output” you want to stream using Model Driven Telemetry. Let’s say we want to find the right path for RSVP interface output:

```
RP/0/RP0/CPU0:NCS5501_bottom#sh rsvp interface bundle-ether 26
```

```
Thu Dec 7 18:27:56.646 PST
```

```
*: RDM: Default I/F B/W % : 80% [cfgd] (max resv/bc0), 0% [default] (bc1)
```

Interface	MaxBW (bps)	MaxFlow (bps)	Allocated (bps)	MaxSub (bps)
Bundle-Ether26	240G	240G	20M (0%)	0

After the “show output” was defined you need to find the corresponding XML schema inside the router, using “schema-describe” command:

```
RP/0/RP0/CPU0:NCS5501_bottom#schema-describe sh rsvp interface bundle-ether 26
```

```
Thu Dec 7 18:28:25.325 PST
```

```
Action: get
```

```
Path: RootOper.RSVP.InterfaceBrief({'InterfaceName': 'Bundle-Ether26'})
```

How to understand which model you need to use: finding XML schema contents

After you found the corresponding XML schema, go and check the content of it. To do this you need to run “m2mcon” and use “get”:

```
RP/0/RP0/CPU0:NCS5501_bottom#run m2mcon
Thu Dec 7 18:29:35.191 PST
Enter 'help' or '?' for help
m2mcon>
m2mcon> get RootOper.RSVP.InterfaceBrief({"InterfaceName": "Bundle-Ether26"})
Result:
[{"RootOper.RSVP.InterfaceBrief":{"InterfaceName": "Bundle-Ether26"},
 {"BandwidthInformation": {"DSTEMode": "PreStandard",
 "PreStandardDSTEInterface": {"AllocatedBitRate": 20000000,
 "IsMaxBandwidthAbsolute": true,
 "IsMaxSubpoolBandwidthAbsolute": true,
 "MaxBandwidth": 240000000000,
 "MaxFlowBandwidth": 240000000000,
 "MaxSubpoolBandwidth": 0}},
 "InterfaceName": "Bundle-Ether26"}]
```

Fields to remember

At this step you have your show output, the schema you need and the fields of this schema. You need to find corresponding YANG model. Do this search based on the name (usually, it is clear which model you need), for RSVP you have just a single model: “Cisco-IOS-XR-ip-rsvp-oper.yang”

How to understand which model you need to use: finding YANG path

Use pyang tool to go through the model, try to find the path (section) that contains the same fields. In our case it will be:

```
VOSIPCHU-M-C1GV:631 vosipchu$ pyang -f tree Cisco-IOS-XR-ip-rsvp-oper.yang --tree-path rsvp/interface-briefs/interface-brief
module: Cisco-IOS-XR-ip-rsvp-oper
+--ro rsvp
  +--ro interface-briefs
    +--ro interface-brief* [interface-name]
      +--ro interface-name      xr:Interface-name
      +--ro bandwidth-information
        | +--ro pre-standard-dste-interface
        | | +--ro allocated-bit-rate?          uint64
        | | +--ro max-flow-bandwidth?         uint64
        | | +--ro max-bandwidth?             uint64
        | | +--ro max-subpool-bandwidth?       uint64
        | | +--ro is-max-bandwidth-absolute?  boolean
        | | +--ro is-max-subpool-bandwidth-absolute?  boolean
        | +--ro standard-dste-interface
        | | +--ro allocated-bit-rate?          uint64
        | | +--ro max-flow-bandwidth?         uint64
        | | +--ro max-bandwidth?             uint64
        | | +--ro max-pool0-bandwidth?        uint64
        | | +--ro max-pool1-bandwidth?        uint64
        | | +--ro is-max-bandwidth-absolute?  boolean
        | | +--ro is-max-bc0-bandwidth-absolute?  boolean
        | | +--ro is-max-bc1-bandwidth-absolute?  boolean
        | +--ro dste-mode?                 Rsvp-mgmt-dste-modes
      +--ro interface-name-xr?          string
```

```
[{"AllocatedBitRate": 20000000, "IsMaxBandwidthAbsolute": true, "IsMaxSubpoolBandwidthAbsolute": true, "MaxBandwidth": 2400000000000, "MaxFlowBandwidth": 2400000000000, "MaxSubpoolBandwidth": 0}],
```

How to understand which model you need to use: final thoughts

If you see that fields between XML and YANG schemas correspond to each other, then bingo, you found the right path (as you may guess, IOS XR Native YANG models are based on internal XML schemas)!

Sometimes it might not work as easy as in our example here.

You might not find the corresponding schema or get several schemas in the reply.

In first situation try to search for “close by” show commands, you might get your schema there ☺

In the second situation go through each schema one-by-one and compare with show output.

There might be situations where you don’t have any schema. In that case, please, contact PM/TME of that product (better to add the name of the customer to make process faster and smoother ;). They know what to do next.

Encoding options

Encoding (or “serialization”) translates data (objects, state) into a format that can be transmitted across the network. When the receiver decodes (“de-serializes”) the data, it has a semantically identical copy of the original data.

IOS XR today has 3 options for encoding.

- **Compact GPB** (Google Protocol buffers aka “protobufs”)
- **Key-Value GPB** (aka Self-Describing GPB)
- **JSON** (starting with 6.3.1)

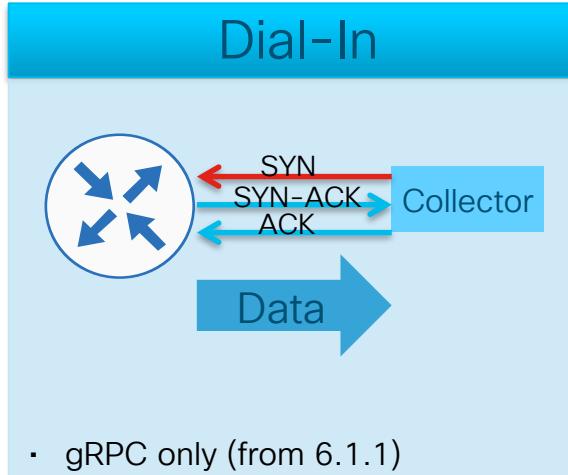
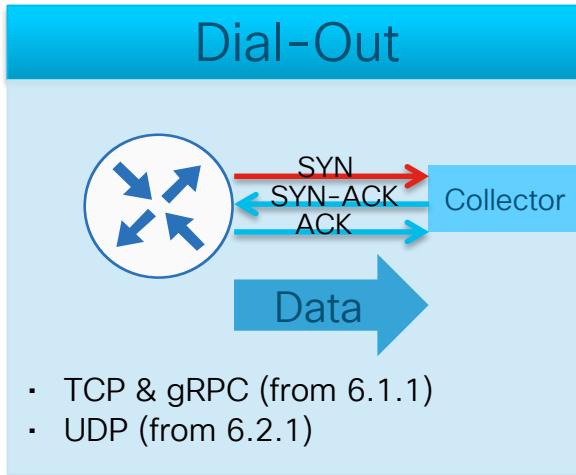
GPB was selected for encoding because it is both faster and more efficient compared with other encoding options (i.e. XML mostly and partially JSON). Also, the early adopter customers already had GPB in their networks which helped both sides speed up development and adoption. JSON was added later (in XR 6.3.1) for a specific customer.

KV-GPB looks very similar to JSON, the major difference is that values are encoded by Varints (Varints are a method of serializing integers using one or more bytes. Smaller numbers take a smaller number of bytes. See here:

<https://developers.google.com/protocol-buffers/docs/encoding>). KV-GPB is slightly more efficient compared with JSON. cGPB is the most efficient encoding option on the wire, as it is totally binary after encoding (tags and values are encoded). But it requires decoding and encoding files at sender and receiver respectively (This is set up manually, see p. 148).

More about encoding and what to do with each in the chapter four.

Transport Options



IOS XR has two options for transport: Dial-In and Dial-out. The difference being which endpoint sends SYN packet first. In both cases data will be pushed out of the router. NOTE: gRPC is required for Dial-in.
gRPC runs over HTTP2 and gives a number of benefits.

1. Support for Dial-IN
2. Feedback from the receiver to the sender (Uses internal signaling, similar to TCP window size control)
3. Binary mode
4. header compression
5. others

Dial-Out and Dial-IN descriptions

In the **Dial-OUT mode**, telemetry is configured on a Telemetry-capable router. Sensor-paths, and Telemetry Destinations are configured and bound together into one or more Telemetry Subscriptions.

A Telemetry-capable router every 30 seconds sends a TCP SYN trying to establish a Telemetry Session to each Telemetry Destination for each Telemetry Subscription. For each established Telemetry Session, the Telemetry-capable router streams Telemetry Messages per the Telemetry Subscription sample-interval configuration. Such Telemetry Subscriptions are considered “*persistent subscriptions*”.

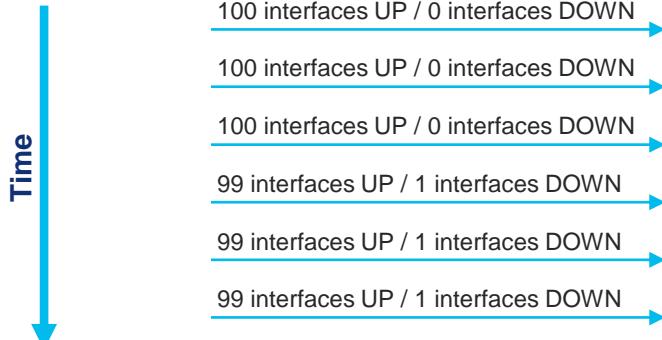
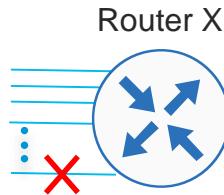
If a Telemetry Session terminates, the Telemetry-capable Device will again send TCP SYN packets every 30 seconds trying to re-establish a new Telemetry Session to the Telemetry Destination. A 30-seconds interval was selected based on a number of tests and feedback from early adopters. For UDP Dial-out mode streaming will be constant and all the time, as there is no connection-oriented protocols involved.

The **Dial-IN mode** of operation allows an MDT Collector to establish an MDT Session, and subscribe dynamically to one or more *pre-configured* sensor-paths or mdt-subscriptions. The MDT-capable router will stream Telemetry messages via the same MDT session (in-band). Such subscriptions are considered “*dynamic subscriptions*”. A dynamic subscription terminates either when the MDT Collector cancels the subscription, or when the MDT Session terminates.

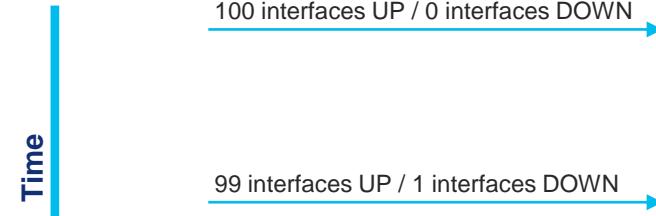
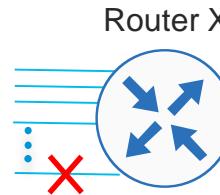
Event-Driven Telemetry overview

Event Driven Telemetry (starts with XR 6.3.1) is complimentary to MDT. The main goal is to send updates whenever they happen. At the start, EDT sends the dump of all the data for the configured sensor-path and only updates are streamed after.

Model-Driven Telemetry



Event-Driven Telemetry



Event-Driven Telemetry configuration

A configuration example:

```
telemetry model-driven
sensor-group interface
  sensor-path Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface
!
subscription interface
  sensor-group-id interface sample-interval 0 ## "0" means EDT
!
```

IOS XR 6.3.1 includes support for:

- 1. Interface changes**
 - Sensor path: Cisco-IOS-XR-pfi-im-cmd-oper:interfaces/interface-xr/interface
 - Sensor path: Cisco-IOS-XR-ipv6-ma-oper:ipv6-network/nodes/node/interface-data/vrfs/vrf/global-briefs/global-brief (*)
- 2. RIB changes**
 - Sensor path: Cisco-IOS-XR-ip-rib-ipv4-oper:rib/vrfs/vrf/afs/af/safs/saf/ip-rib-route-table-names/ip-rib-route-table-name/routes/route
 - Sensor path: Cisco-IOS-XR-ip-rib-ipv6-oper:ipv6-rib/vrfs/vrf/afs/af/safs/saf/ip-rib-route-table-names/ip-rib-route-table-name/routes/route
- 3. Syslog updates**
 - Sensor path: Cisco-IOS-XR-infra-syslog-oper:syslog/messages/message

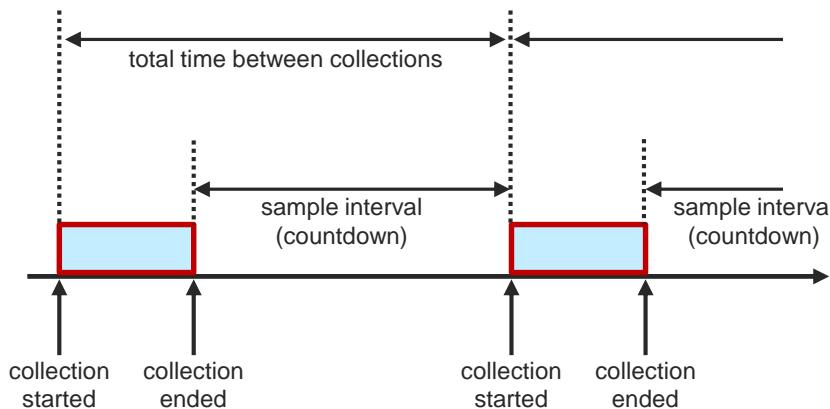
If your customer wants some specific event-driven feature, please, let us know !

Sample intervals variations

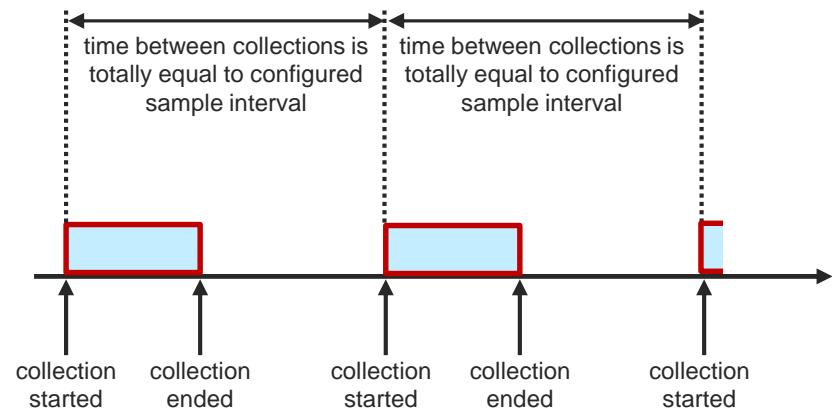
There are two commands exist to configure *sample-interval* :

- sensor-group-id health sample-interval <time_in_ms>
- sensor-group-id health strict-timer

With just “sample-interval” configured you will have sample-interval to start countdown only after collection is done (so, you will have your time as “Sample-interval”+“collection-time”)



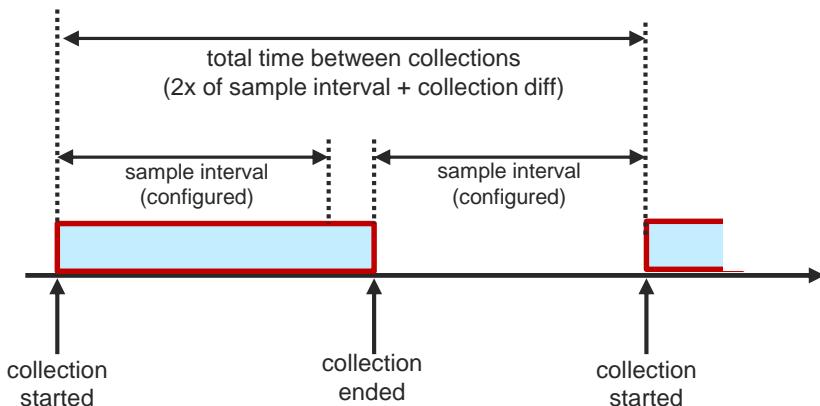
When you have “sample-interval” and “strict-timer” configured, collection time will not be included, you will observe constant behavior with your timer.



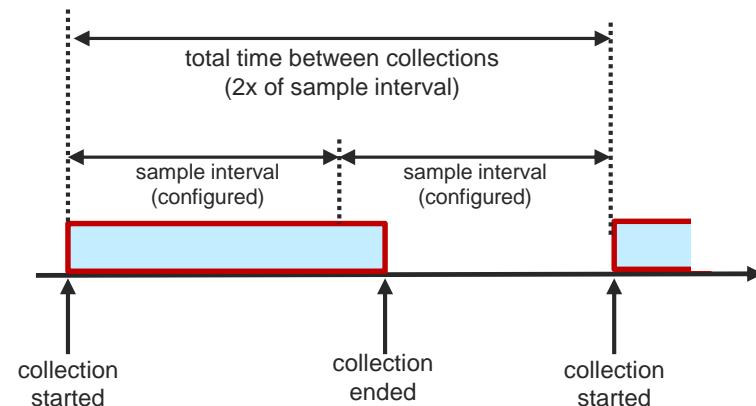
How MDT behaves in case of missed collections

In any case we don't send anything in case of missed collection. All the information is seen internally only.

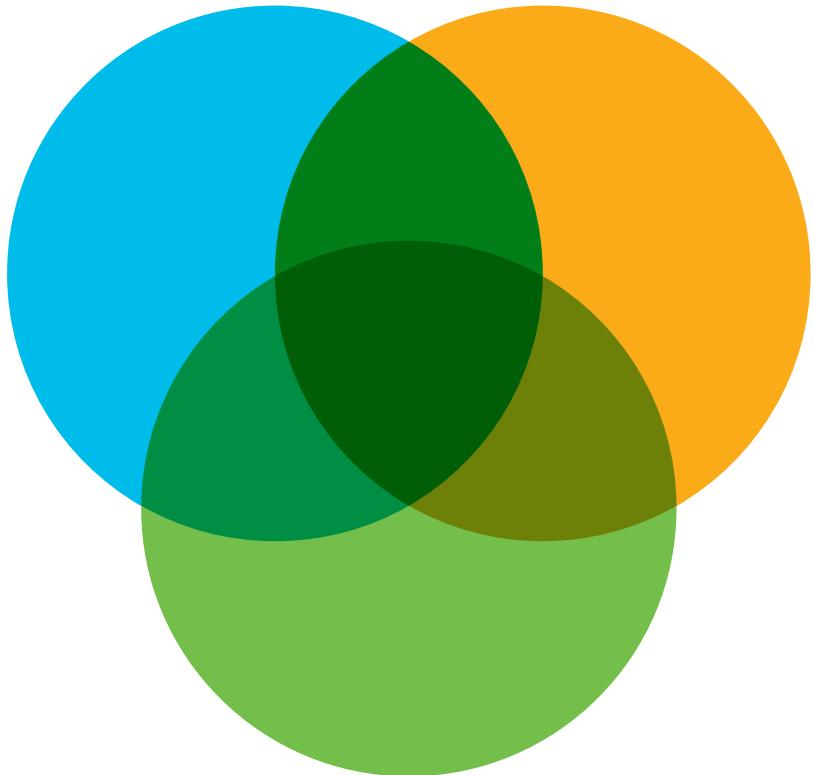
For collections without “strict-timer” configured as soon as current collection stops, the system will start new sample-interval timer countdown. As soon as our collection finishes after configured sample-interval, MDT will find this with the internal math and update that we have “+1” missed collection.



For collections with “strict-timer” configured, the system keeps tracking of sample-interval timer configured and will always try to align for that border. If current collection took longer, the system will update internally that there was a “+1” missed collection.



MDT internals

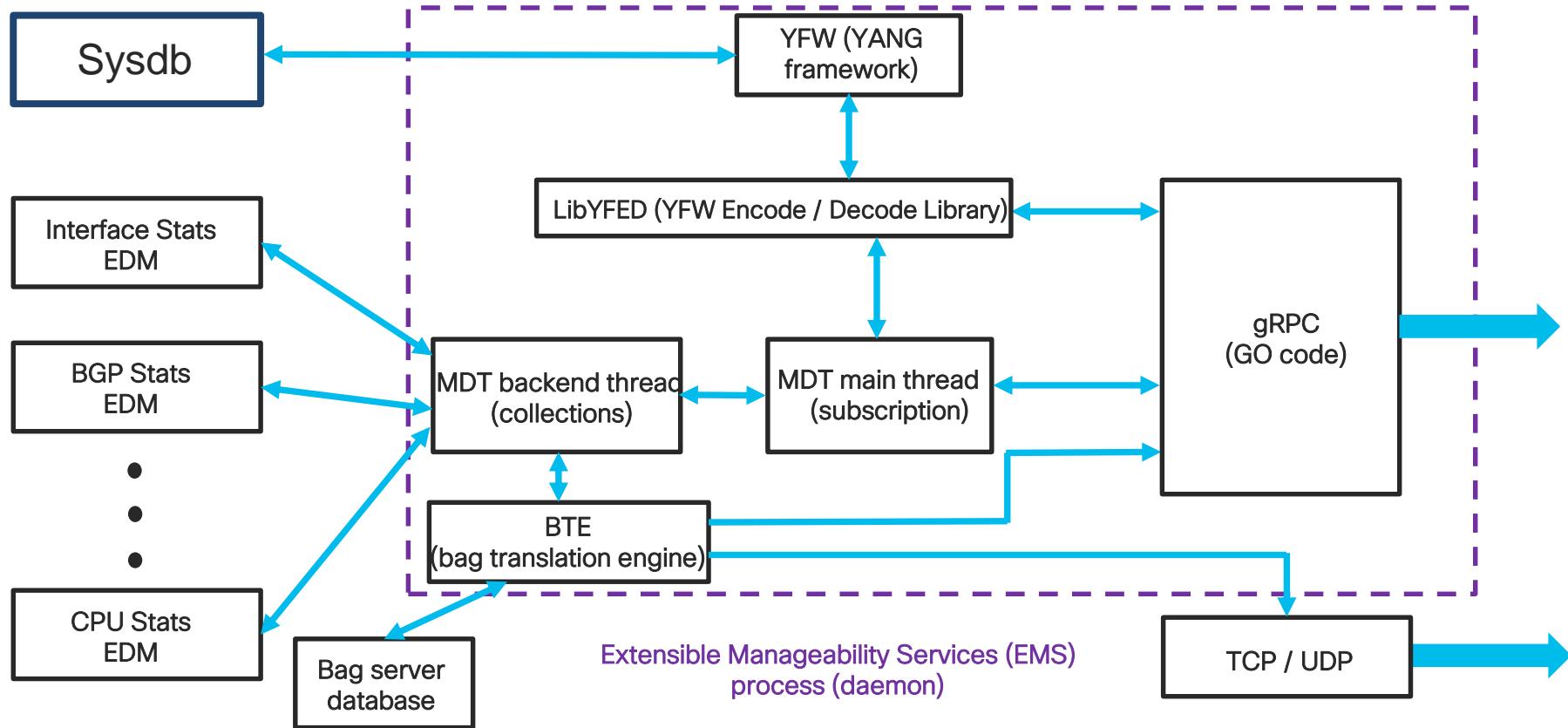


Introduction for the chapter

The goal of this chapter is to explain how Streaming Telemetry works internally. We all know that it is faster than SNMP. But how? This chapter should give you a solid overview.



MDT internals overview (eXR)



Main elements of MDT

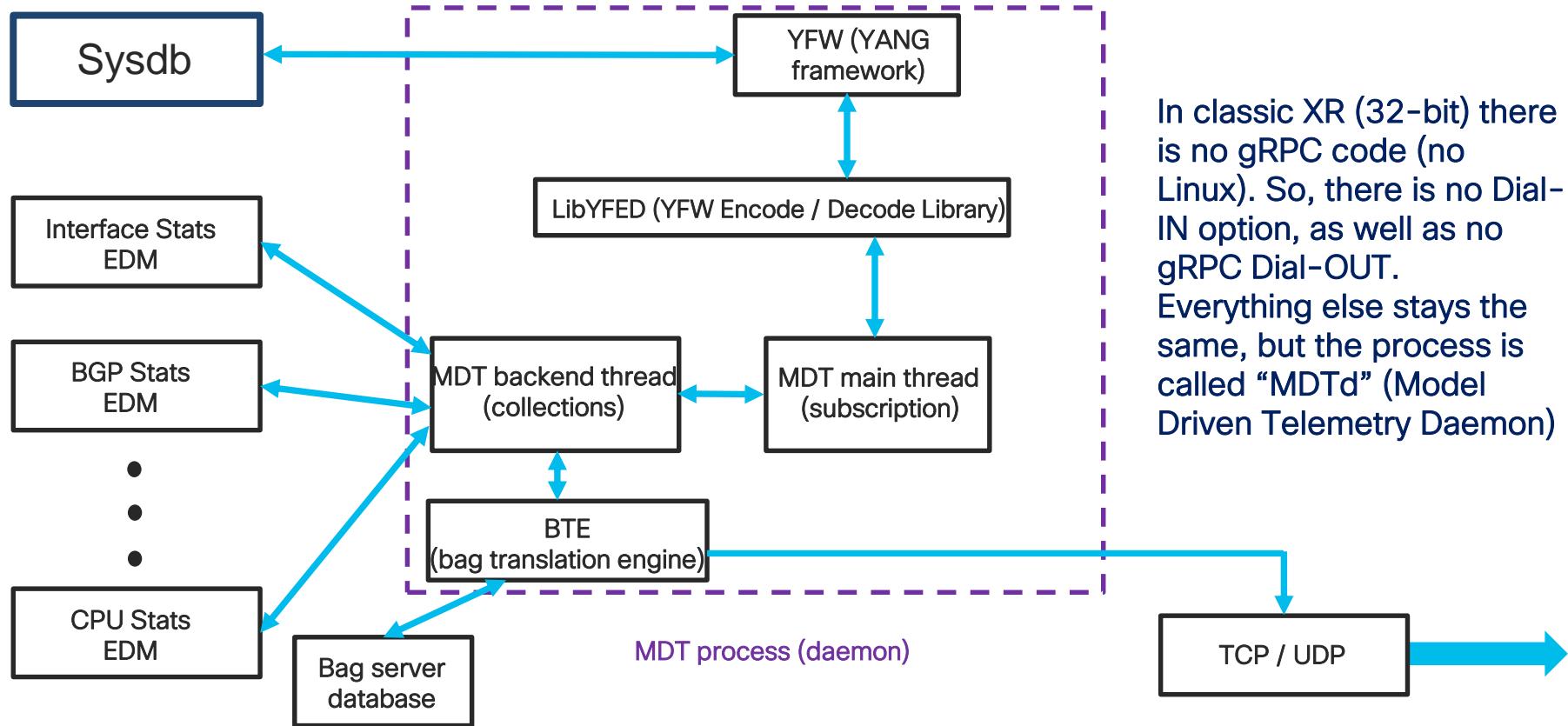
Streaming Telemetry in IOS eXR is a number of threads within the EMS (Extensible Manageability Services) process.

EMS by itself is an eXR daemon that provides an extensible interface for users to manage a device. Architecture-wise EMS is designed to support "Any Yang Model", "Any Encoding" and "Any Transport". Telemetry is a part of this architecture, where each thread is responsible to make sure that Telemetry is tuned to be highly scalable and perform much better comparing to SNMP.



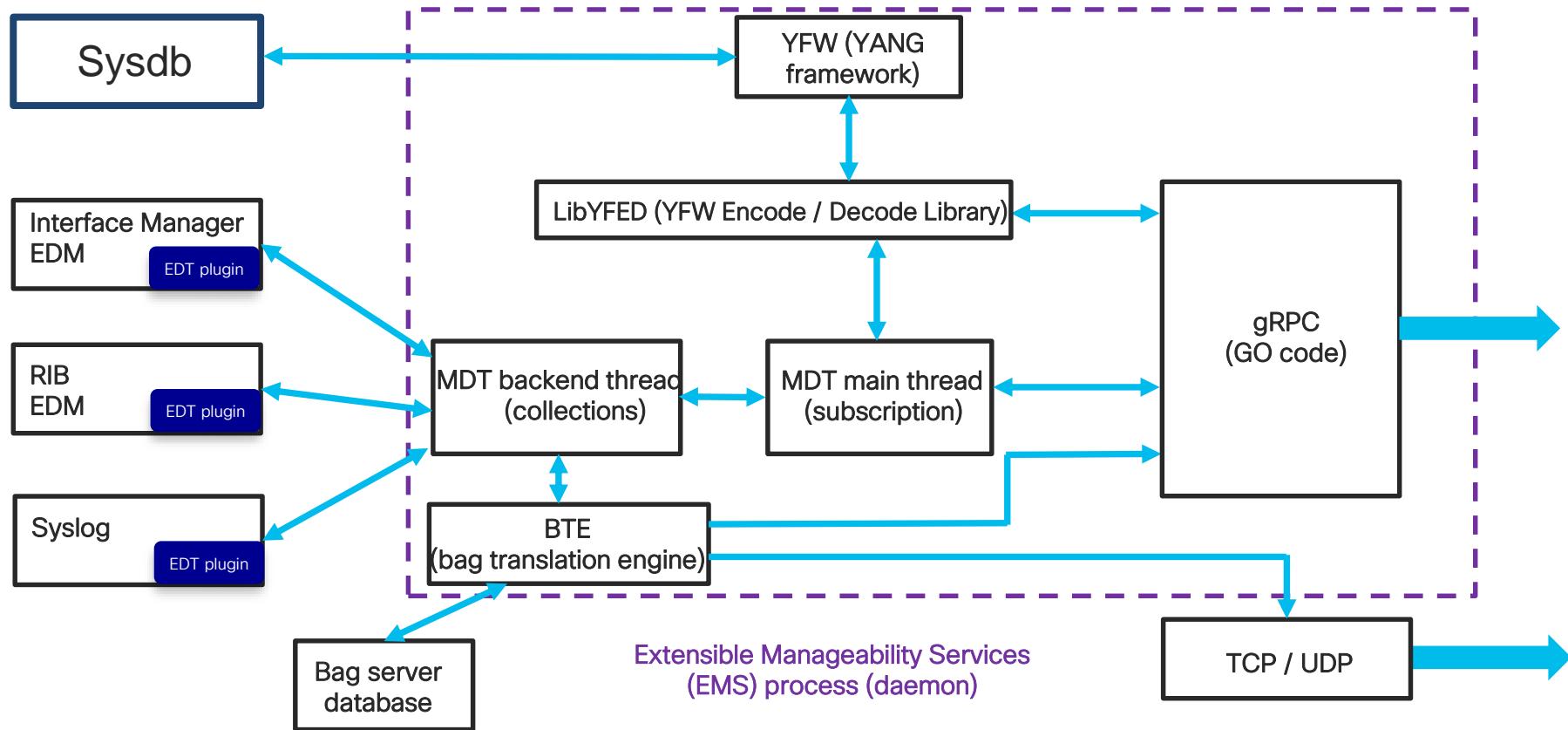
Let's have a brief overview of each component of MDT. Understanding of these components will greatly help you with troubleshooting and better/deeper discussions with your customers

MDT internals overview (classic XR)



In classic XR (32-bit) there is no gRPC code (no Linux). So, there is no Dial-IN option, as well as no gRPC Dial-OUT. Everything else stays the same, but the process is called “MDTd” (Model Driven Telemetry Daemon)

EDT internals overview



Tricks with streaming through gRPC

There are two important things to know when you have Streaming Telemetry with gRPC:

1. Make sure that Linux has the route to your collector (as well as collector has the path back).
2. Make sure there are no MTU interop problems between Linux and IOS XR (when you have streaming through Mgmt ports).

How to get into linux to check these points? Use “bash” command (to get into 3d party namespace):

```
RP/0/RP0/CPU0:NCS5502_bottom#bash  
Thu Nov 9 10:56:54.878 PST  
[NCS5502_bottom:~]$
```

From there we can start our check

Find Linux Interfaces

Here is how to check interfaces mirrored into Linux:

```
[NCS5502_bottom:~]$ ifconfig
```

```
Hg0_0_0_0 Link encap:Ethernet HWaddr 00:8a:96:6a:6c:dc
inet addr:192.168.13.3 Mask:255.255.255.0
inet6 addr: fe80::28a:96ff:fe6a:6cdc/64 Scope:Link
inet6 addr: 2222:beaf:13::3/64 Scope:Global
UP RUNNING NOARP MULTICAST MTU:9192 Metric:1
RX packets:111000348611 errors:0 dropped:0 overruns:0 frame:0
TX packets:120524278070 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:117676419373109 (107.0 TiB) TX bytes:127025705461727 (115.5 TiB)
```

```
Mg0_RP0_CPU0_0 Link encap:Ethernet HWaddr 2c:5a:0f:77:66:aa
inet addr:10.30.110.43 Mask:255.255.255.224
inet6 addr: fe80::2e5a:fff:fe77:66aa/64 Scope:Link
UP RUNNING NOARP MULTICAST MTU:1514 Metric:1
RX packets:696026 errors:0 dropped:0 overruns:0 frame:0
TX packets:1794949 errors:0 dropped:0 overruns:0 carrier:3
collisions:0 txqueuelen:1000
RX bytes:47634249 (45.4 MiB) TX bytes:2373441110 (2.2 GiB)
```

```
fwd_ew Link encap:Ethernet HWaddr 00:00:00:00:00:0b
inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
UP RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:140 (140.0 B)
```

```
fwdintf Link encap:Ethernet HWaddr 00:00:00:00:00:0a
inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
UP RUNNING NOARP MULTICAST MTU:9174 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:16 errors:0 dropped:1 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:1176 (1.1 KiB)
```

```
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
inet6 addr: 2500::3/128 Scope:Global
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

```
lo:0 Link encap:Local Loopback
inet addr:172.16.0.3 Mask:255.255.255.255
UP LOOPBACK RUNNING MTU:65536 Metric:1
```

What do you see?

You need to remember this when looking through interfaces:

1. **Data Plane interfaces**: Usual data plane ports on an IOS XR router (like “HundredGigE0/0/0/0”).
 - Only active ports are shown (e.g. “Hg0_0_0_0”).
 - For active bundles, the bundle itself as well as all member interfaces are shown.
 - Loopback ports also present there.
2. **“fwd_ew”**: This is the interface responsible for East-West communication.
 - In other words, when an app is talking with an XR protocol. This is N/A for Telemetry, but just good to know.
 - Loopback1 is, by default, used as the source of this interface (if you don’t have it, East-West communication will not happen). You can also change this behavior.
3. **“fwdintf”**: This is the interface responsible for forwarding of data traffic.
 - All LC traffic is routed over this interface. For example, when we stream information using gRPC over Data Plane ports, this interface is used to deliver Telemetry to Data Plane ports.
 - By default, with no config, you won’t be able to see this interface.
 - As soon as you configure any first Loopback, it will be inserted as the source address for this interface.
 - Other option is to configure the interface you want explicitly (through TPA config, see next slide).
4. **“Mg0_RP0_CPU0_0”**: This is the interface responsible for communication with Mgmt Port.
 - For example, when we stream information out using gRPC over Mgmt Plane ports, this interface is used to deliver traffic).

It is important to understand Linux routes

Check the routes in Linux:

```
NCS5502_bottom:~]$ route  
Kernel IP routing table  
Destination     Gateway      Genmask      Flags Metric Ref  Use Iface  
default         0.0.0.0      U            0      0      0 fwdintf  
10.30.110.32   *           255.255.255.224 U      0      0      0 Mg0_RP0_CPU0_0
```

Every packet out of network will go there (default route). It is not configured by default.

The network behind Mgmt port

Traffic to LC ports
Traffic to Mgmt ports

```
[NCS5502_bottom:~]$ ip route  
default dev fwdintf scope link src 172.16.0.3  
10.30.110.32/27 dev Mg0_RP0_CPU0_0 proto kernel scope link src 10.30.110.43
```

IP address of fwdintf (either configure manually, or the first active Loopback will be used (except Loo1))

Mgmt IP address

Update routing table if you need this

You can configure interface for fwdintf manually (it has to have IP address configured!):

```
tpa
vrf default
address-family ipv4
update-source Bundle-Ether13
!
interface Bundle-Ether13
description 5502_BOTTOM==TO==5501_TOP
ipv4 address 192.168.13.3 255.255.255.0
load-interval 30
```

Now check the route table on Linux:

```
[NCS5502_bottom:~]$ ip route
default dev fwdintf scope link src 192.168.13.3
10.30.110.32/27 dev Mg0_RP0_CPU0_0 proto kernel scope link src 10.30.110.43
```

IP address of fwdintf was changed according to the config above

Summary of Linux routing for Streaming Telemetry with gRPC:

- If you have your collector behind your Mgmt Plane port, you're good to go.
- If you have your collector behind your Data Plane ports, you should:
 - Either configure a dedicated port under TPA (and make sure you have routing from it to your collector and back)
 - Or make sure your loopback port has this routing connectivity with the collector.

More about gRPC and Telemetry here: <https://xrdocs.github.io/telemetry/tutorials/2017-05-05-mdt-with-grpc-transport-tricks/>

MTU in Linux and IOS XR

There is a different behavior for MTU on Linux vs XR:

- IOS XR MTU is a Layer 2 MTU (it includes L2 Header).
- Linux MTU is a Layer 3 MTU (it doesn't include L2 Header).

Linux will copy MTU number from your XR port. For example, you have your management port with MTU configured for 1514B and Linux will copy this number.

It means that whenever gRPC starts setting up a connection, Linux TCP will use Linux value for TCP MSS.

It shouldn't be an issue for many cases, except when your router has a collection that needs to use full MTU-size packet. In that case you will observe 14 bytes-less window of drops, while Linux still thinks that the packet is below MTU and no fragmentation needed, but the port in XR will see an incoming packet with the size bigger than MTU (and, hence, it will be silently dropped).

This is critical for NCS1k product line where Telemetry is streamed out through Mgmt Plane ports.

The best way to deal with this situation is to go into bash and configure MTU on Mgmt Port to be 14 bytes less than MTU on XR port:

```
RP/0/RP0/CPU0:NCS5502_bottom#bash  
[NCS5502_bottom:~]$ ifconfig Mg0_RP0_CPU0_0 mtu 1500
```

This will not survive a reboot, so, you might want to automate this.

It was supposed to be optimized (fixed) in IOS XR 6.3.1, but seems that some work is still needed to be done. (*Will be updated when the info is available*)

MDT troubleshooting

Troubleshooting: checking “sh run” output

Make a quick check of your running telemetry configuration. It is very quick and you can double check yourself that you've configured sensor-groups, sensor-paths, destination, encoding, transport and subscription.

It should look similar to this:

```
telemetry model-driven
destination-group DGroup1
address-family ipv4 10.30.110.38 port 57500
encoding self-describing-gpb
protocol grpc no-tls
!
sensor-group health
  sensor-path Cisco-IOS-XR-shellutil-oper:system-time/uptime
!
subscription health
  sensor-group-id optics sample-interval 10000
  destination-id DGroup1
```

Troubleshooting: check telemetry applied in EMSd

After you've checked running config, let's make sure that it is properly applied inside the internal telemetry process (EMSD). The goal is to see that your config was fully installed (you should find all you configured before right there):

```
RP/0/RP0/CPU0:NCS5502_bottom#sh telem model-driven trace config last 100
Nov 16 20:47:41.579 m2m/mdt/config 0/RP0/CPU0 t23312 63 [ems_mdt_cfg_apply_func]: item ord_M/mdt/ord_d/destination/DGroup1/ord_A/enter, event SET, datatype -1401912336
Nov 16 20:27:19.882 m2m/mdt/config 0/RP0/CPU0 t20564 167055 [ems_mdt_cfg_apply_func]: item ord_M/mdt/ord_d/destination/DGroup1/ord_b/dest-address/ipaddr/ipv4/10.30.110.38,0xe09c/encoding, event CREATE_SET, datatype -1401914976
Nov 16 20:47:41.579 m2m/mdt/config 0/RP0/CPU0 t23312 67 [ems_mdt_cfg_apply_func]: item ord_M/mdt/ord_d/destination/DGroup1/ord_b/dest-address/ipaddr/ipv4/10.30.110.38,0xe09c/encoding, event SET, datatype -1401911856
Nov 16 20:47:41.579 m2m/mdt/config 0/RP0/CPU0 t23312 71 [ems_mdt_cfg_apply_func]: item ord_M/mdt/ord_n/sensor-group/health/ord_A/enter, event SET, datatype -1401911376
Nov 16 20:47:41.579 m2m/mdt/config 0/RP0/CPU0 t23312 73 [ems_mdt_cfg_apply_func]: item ord_M/mdt/ord_n/sensor-group/health/ord_a/sensor-path/Cisco-IOS-XR-shellutil-oper:system-time%2fuptime/enter, event SET, datatype -1401911136
Nov 16 20:47:41.579 m2m/mdt/config 0/RP0/CPU0 t23312 77 [ems_mdt_cfg_apply_func]: item ord_M/mdt/subscription/health/ord_A/enter, event SET, datatype -1401910656
Nov 16 20:47:41.579 m2m/mdt/config 0/RP0/CPU0 t23312 79 [ems_mdt_cfg_apply_func]: item ord_M/mdt/subscription/health/ord_c/sensor-group/health/sample-interval, event SET, datatype -1401910416
Nov 16 20:47:41.579 m2m/mdt/config 0/RP0/CPU0 t23312 81 [ems_mdt_cfg_apply_func]: item ord_M/mdt/subscription/health/ord_c/destination-id/DGroup1/enter, event SET, datatype -1401910176
Nov 16 20:47:41.618 m2m/mdt/config 0/RP0/CPU0 t23298 277 [ems_mdt_cfg_apply_func]: ==Processing cfg done, total tuple count: 3 + 10 ==
Nov 16 20:47:41.618 m2m/mdt/config 0/RP0/CPU0 t23298 278 [ems_mdt_cfg_apply_func]: No More config batches to follow...
```

Troubleshooting: checking Sensor-Paths (1/6)

Let's now verify that our sensor paths were correctly resolved internally:

It should have an output similar to this:

```
RP/0/RP0/CPU0:NCS5502_bottom#sh tele model-driven sensor-group optics
Thu Nov 9 19:25:38.251 PST
Sensor Group Id:optics
Sensor Path: Cisco-IOS-XR-controller-optics-oper:optics-oper/optics-ports/optics-port/optics-info
Sensor Path State: Resolved
```

You should see "Resolved" which means that the sensor-path was properly matched with EDM bags.
You can also see SysDB path to that oper data (the same show command, but add "internal" at the end):

```
RP/0/RP0/CPU0:NCS5502_bottom#sh tele model-driven sensor-group optics internal
Thu Nov 9 19:25:38.251 PST
Sensor Group Id:optics
Sensor Path: Cisco-IOS-XR-controller-optics-oper:optics-oper/optics-ports/optics-port/optics-info
Sensor Path State: Resolved
Sysdb Path: /oper/optics/if/<optics_controller_oper_OpticsPort_port_name>/optics_info
Yang Path: Cisco-IOS-XR-controller-optics-oper:optics-oper/optics-ports/optics-port/optics-info
```

As you can see, the full SysDB path to the needed EDM is shown here. Let's see what we can do with that info.

Troubleshooting: checking Sensor-Paths (2/6)

To see the operational data that will be streamed out, you need to go into SysDB:

```
RP/0/RP0/CPU0:NCS5502_bottom#run sysdbcon -m
Thu Nov  9 19:33:01.889 PST
Read-only sysdbcon
Please enter initial bind point>/
```

Just hit “enter” here. You’re defining your start point for searching, it will be “/”

You can now follow your SysDB path discovered before. There are several rules for proper experience with sysdbcon:

1. When you’re copying SysDB path output collected above, don’t include the beginning “/”, start with the first word.
2. You will mostly use “get” or “list” in your searches. Usually start with “list” to see options and when you’re at the end of the SysDB path, use “get” to see the content of the bag (a “bag” is the data inside EDM you’re trying to push out using Telemetry)
3. Each time you see a “*” or words with “< >”, it is a sign for you to use the “list” option.

Troubleshooting: checking Sensor-Paths (3/6)

Start with “list”:

```
sysdbcon:[m]/> list oper/optics/if/  
[external dir]      'Optics0_0_0_0/'  
[external dir]      'Optics0_0_0_1/'  
[external dir]      'Optics0_0_0_2/'  
[external dir]      'Optics0_0_0_3/'  
<<<removed for breavity>>>
```

Select a single node and proceed to the “get”:

```
sysdbcon:[m]/> get oper/optics/if/Optics0_0_0_0/optics_info  
[bag (len 2236)]      'oper/optics/if/Optics0_0_0_0/optics_info' (optics_edm_info v1.3 XDR)  
  'optics_edm_info' {  
    [enum (len 4)]    'TransportAdminState' 0x00000002  
    [bool]          'OpticsPresent' TRUE  
    [enum (len 4)]    'OpticsType' 0x00000001  
    [string (len 15)] 'DerivedOpticsType'  '100G QSFP28 AOC'  
    [string (len 0)]  'OpticsModule' "  
    [enum (len 4)]    'DWDMCarrierBand'    0x00000000
```

As the result, you will see all the information that will be pushed by telemetry. Remember where the “bag” is located and how to find the name (in purple above). We will need this information later.

Troubleshooting: checking Sensor-Paths (4/6)

Here is an example of finding an issue. At this step there is no Netflow configuration on the router, the SysDB path is shown as resolved (EDM exists, but it doesn't have any data!). As the result, you will see no information streamed.

```
RP/0/RSP0/CPU0:ASR9006#sh tele model-driven sensor-group test internal
Thu Nov 9 20:00:14.460 PST
Sensor Group Id:test
  Sensor Path: Cisco-IOS-XR-asr9k-netflow-oper:net-flow/statistics/statistic/producer
Sensor Path State: Resolved
  Sysdb Path: /oper/netflow/node/<netflow_asr9k_oper_Statistics_nodeid>/producer/stats
  Yang Path: Cisco-IOS-XR-asr9k-netflow-oper:net-flow/statistics/statistic/producer/statistics
```

When you try to find the data in SysDB, you will fail:

```
RP/0/RSP0/CPU0:ASR9006#run sysdbcon -m
Thu Nov 9 20:00:22.955 PST
Read-only sysdbcon
Please enter initial bind point>/
Enter 'help' or '?' for help
sysdbcon:[m]/> list oper/netflow/node/
%SYSDB-SYSDB-4-SC_LIST : [Sysdbcon] While executing the list request the following error occurred:
'sysdb' detected the 'warning' condition 'A SysDB client tried to access a nonexistent item or list an empty directory'
```

As you can see, “Resolved” doesn’t always mean everything is good. If you see resolved, but no data streamed, check SysDB for the oper data. If that doesn’t exist – check that you configured the needed feature on your router.

Troubleshooting: checking Sensor-Paths (5/6)

Finally, after Netflow was configured, the output in SysDB is this:

```
sysdbcon:[m]/> get oper/netflow/node/831/producer/stats
[bag (len 152)]      'oper/netflow/node/831/producer/stats' (nf_producer_stats v1.0 XDR)
'nf_producer_stats' {
    [uint64 (len 8)]  'IPv4IngressFlows'   0x0000000000000000
    [uint64 (len 8)]  'IPv4EgressFlows'   0x0000000000000000
    [uint64 (len 8)]  'IPv6IngressFlows'   0x0000000000000000
    [uint64 (len 8)]  'IPv6EgressFlows'   0x0000000000000000
    [uint64 (len 8)]  'MPLSIngressFlows'  0x0000000000000000
    [uint64 (len 8)]  'MPLSEgressFlows'  0x0000000000000000
    [uint64 (len 8)]  'DropsNoSpace'     0x0000000000000000
    [uint64 (len 8)]  'DropsOthers'      0x0000000000000000
    [uint64 (len 8)]  'UnknownIngressFlows' 0x0000000000000000
    [uint64 (len 8)]  'UnknownEgressFlows' 0x0000000000000000
    [uint64 (len 8)]  'WaitingServers'    0x0000000000000000
    [uint64 (len 8)]  'SppRxCounts'      0x0000000000000000
    [uint64 (len 8)]  'FlowPacketCounts'  0x0000000000000000
    [string (len 0)]  'LastCleared'     "
}
[end bag]
```

Troubleshooting: checking Sensor-Paths (6/6)

There are two situations possible where sensor path is not resolved.

When the path within the YANG model is wrong (but the model is right):

```
RP/0/RP0/CPU0:NCS5502_bottom#sh tele model-driven sensor-group test internal
Fri Nov 17 10:58:04.355 PST
Sensor Group Id:test
  Sensor Path: Cisco-IOS-XR-shellutil-oper:foobar
  Sensor Path State: Not Resolved
Status: "unknown-element"
```

You will see "unknown-element". So, go back to your model and check the path ("pyang" is your friend there)

Another possible situation is when the model is not supported on that platform at all:

```
RP/0/RP0/CPU0:NCS5502_bottom#sh tele model-driven sensor-group test internal
Fri Nov 17 10:58:04.355 PST
Sensor Group Id:test
  Sensor Path: Cisco-IOS-XR-foobar:barfoo
  Sensor Path State: Not Resolved
Status: Module 'Cisco-IOS-XR-foobar' not recognized or supported for the specified operation
```

In this case go back to chapter one ("High-level MDT overview") and find out how to get models per platform.

Sensor-Paths enhancement: filtering capability

There is one specific case exists with Sensor Paths and Telemetry.

Starting with XR 6.2.2 we support path filtering (as a limited availability as of today).

You can see this feature this way:

1. For interfaces you can go both ways:

- **using regex “*”** (e.g.: Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface[interface-name='HundredGigE*']/latest/generic-counters)
- **or using a specific value** (e.g.: Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface[interface-name='HundredGigE0/0/0/0']/latest/generic-counters)

2. For other sensor paths you should go just with specifying an **exact element**, e.g:

Cisco-IOS-XR-ip-rib-ipv4-oper:rib/vrfs/vrf[vrf-name='MDT']/afs/af/safs/saf/ip-rib-route-table-names/ip-rib-route-table-name/protocol/bgp/as/information

The main reason why you might be interested in doing this it to get some additional optimization from MDT.

The optimization is based on two things:

- You will have BTE ready to have full description from the very beginning. All the paths with multi-selection options (“*” in SysDB paths) require BTE to wait for the first transmission to find all the configured elements (as in the BGP example above, BTE would have to wait for the first transmission to find all the VRFs configured and would re-use that knowledge during next transmissions)
- You will have MDT Backend Thread request only data you specified (exact value or a range for interfaces).

** we will have global regex support added later, as of today, think about “interfaces” as the only path for “*”. Others are just with exact values.*

Troubleshooting: gRPC session establishment (1/5)

If you plan to use gRPC Dial-IN, then you have to have this configuration on your router:

```
grpc
port 57400
! address-family ipv6 # grpc dial-in supports ipv4 or ipv6; uncomment this for ipv6 support
```

Now make sure you have connectivity to your collector from Linux:

```
NCS5502_bottom:~]$ route
Kernel IP routing table
Destination     Gateway      Genmask      Flags Metric Ref  Use Iface
default         *          0.0.0.0      U     0      0      fwdintf
10.30.110.32   *          255.255.255.224 U     0      0      0 Mg0_RP0_CPU0_0

RP/0/RP0/CPU0:NCS5502_bottom#bash
Thu Nov  9 20:32:25.119 PST
[NCS5502_bottom:~]$ ping 10.30.110.38
PING 10.30.110.38 (10.30.110.38) 56(84) bytes of data.
64 bytes from 10.30.110.38: icmp_seq=1 ttl=64 time=0.800 ms
64 bytes from 10.30.110.38: icmp_seq=2 ttl=64 time=0.974 ms
```

If your collector is behind a data plane port, you will either need to add a port through TPA config or make sure you have connectivity through Loopback (that is used as the source interface of “fwdintf” - see “MDT internals” chapter, p.48)

Troubleshooting: gRPC session establishment (2/5)

After you checked that you have a two-way L3 path, you can have a look into transport part of gRPC (dial-out mode here on):

```
RP/0/RP0/CPU0:NCS5502_bottom#sh grpc trace ems last 50
Nov 16 20:20:20.805 ems/conf 0/RP0/CPU0 t20564 EMS-CONF: ==Start Processing emsd cfg ==
Nov 16 20:20:20.805 ems/conf 0/RP0/CPU0 t20564 EMS-CONF: start_guard done creating guard
Nov 16 20:20:20.805 ems/conf 0/RP0/CPU0 t22349 EMS-CONF:guard start guard at tid(140562636044032) for type(2), wait for (5)s
Nov 16 20:20:20.805 ems/conf 0/RP0/CPU0 t20564 EMS-CONF: calling Go callback to end CONFIG
Nov 16 20:20:20.805 ems/conf 0/RP0/CPU0 t20564 EMS-CONF: emsd_cfg_notif_pc:2493 called with (21,0)
Nov 16 20:20:20.805 ems/conf 0/RP0/CPU0 t20564 EMS-CONF: emsd_cfg_notif_pc:2605 start/restart service
Nov 16 20:20:20.805 ems/conf 0/RP0/CPU0 t20564 EMS-CONF: IsRdyToStartSrv:362 No cfg change. Not restarting server now
Nov 16 20:20:20.805 ems/conf 0/RP0/CPU0 t20564 EMS-CONF: ==Processing cfg done==
Nov 16 20:20:20.805 ems/conf 0/RP0/CPU0 t20564 EMS-CONF:stop_guard stop guard for 140562636044032
Nov 16 20:20:20.842 ems/conf 0/RP0/CPU0 t20569 EMS-CONF:emsd_is_active_role get proc role (1)
Nov 16 20:20:20.847 ems/grpc 0/RP0/CPU0 t24858 EMS-GRPC: grpc: addrConn.resetTransport failed to create client transport: connection error: desc = "transport: dial tcp 10.30.110.38:57500: getsockopt: connection refused"; Reconnecting to {"10.30.110.38:57500" <nil>}
Nov 16 20:20:20.847 ems/grpc 0/RP0/CPU0 3444# t24858 EMS-GRPC: Failed to dial 10.30.110.38:57500: context canceled; please retry.
===== for TLS setup you might also see this =====
Nov 17 15:51:02.340 ems/grpc 0/RP0/CPU0 t27814 EMS-GRPC: Failed to dial 10.30.110.38:57500: connection error: desc = "transport: x509: certificate signed by unknown authority"; please retry.
```

This output shows the start of the gRPC process. You don't need to go through each line, but you can see that it failed to connect to the collector (as it is down now). As always, just pay attention to errors/failures/etc.
gRPC will try again in 30 seconds.

Troubleshooting: gRPC session establishment (3/5)

Another way is to look at gRPC from the MDT (EMSD) side:

```
RP/0/RP0/CPU0:NCS5502_bottom#sh tele model-driven trace go-info last 50
Nov 16 20:27:19.826 m2m/mdt/go-info 0/RP0/CPU0 t29422 167020 [mdt_go_trace_info]: dialReqAgent:376 1: req 505, entries when dequeue is 0
Nov 16 20:27:19.912 m2m/mdt/go-info 0/RP0/CPU0 t20569 167335 [mdt_go_trace_info]: emsMdtConnEstablish:206 Send dial req of '10.30.110.38:57500' to normal dialer
Nov 16 20:27:19.912 m2m/mdt/go-info 0/RP0/CPU0 75305# t21083 167338 [mdt_go_trace_info]: queueDialRequest:363 0: req 508, dialer entries in queue 0.
Nov 16 20:27:19.912 m2m/mdt/go-info 0/RP0/CPU0 77438# t21083 167339 [mdt_go_trace_info]: mdtConnEstablish:96 0: Dialing out to 10.30.110.38:57500, req 508
Nov 16 20:27:19.912 m2m/mdt/go-info 0/RP0/CPU0 68910# t21083 167340 [mdt_go_trace_info]: mdtConnEstablish:134 0: req 508, TLS is false, cert , host .
Nov 16 20:27:19.912 m2m/mdt/go-info 0/RP0/CPU0 72110# t21083 167341 [mdt_go_trace_info]: mdtDialer:210 0: namespace /var/run/netns/global-vrf
Nov 16 20:27:19.912 m2m/mdt/go-info 0/RP0/CPU0 74243# t21083 167342 [mdt_go_trace_info]: getAddDialerChanWithLock:141 0:Found existing dialer for namespace /var/run/netns/global-vrf
Nov 16 20:27:19.912 m2m/mdt/go-info 0/RP0/CPU0 76376# t21083 167343 [mdt_go_trace_info]: mdtDialer:222 0: Reuse dialerChan 0xc820088188 for /var/run/netns/global-vrf
Nov 16 20:27:19.912 m2m/mdt/go-info 0/RP0/CPU0 79574# t23342 167344 [mdt_go_trace_error]: mdtConnEstablish:164 0: grpc service call failed, ReqId 508, 10.30.110.38:57500, rpc error: code = 14
desc = grpc: the connection is unavailable
===== for TLS setup you might also see this ======
Nov 17 15:48:32.303 m2m/mdt/go-info 0/RP0/CPU0 190062# t23279 62335 [mdt_go_trace_info]: mdtConnEstablish:96 1: Dialing out to 10.30.110.38:57500, req 501
Nov 17 15:48:32.303 m2m/mdt/go-info 0/RP0/CPU0 192195# t23279 62336 [mdt_go_trace_info]: mdtConnEstablish:134 1: req 501, TLS is false, cert , host .
```

Here we check whether EMSd sees the process correctly and we don't have any issues in interop. As before, try to see what has failed or look for errors, in most cases description of errors explains the problem.

Troubleshooting: gRPC session establishment (4/5)

Now let's have a look on a successful call using gRPC:

```
RP/0/RP0/CPU0:NCS5502_bottom#sh grpc trace ems las 50
Nov 16 20:51:21.069 ems/conf 0/RP0/CPU0 t23319 EMS-CONF:emsd_is_active_role get proc role (1)
Nov 16 20:51:21.071 ems/info 0/RP0/CPU0 t23834 EMS_INFO: mdtSendAgent:486 New chanID 1, Limits(HWCA) 4000, 1000, 200, 100 total 13000, 12000, 6000
===== for TLS setup you might also see this =====
Nov 17 15:53:32.307 ems/info 0/RP0/CPU0 t28163 EMS_INFO: nsDialerCheckAddTLSOption:349 mdtDialout: TLS pem: /misc/config/grpc/dialout/dialout.pem, Server host: chkpt1.com
```

That's it. After you have your connection up, your trace is stopped for EMS part of the gRPC.
Pay attention to the channel ID, you need to check that it is the same from MDT side (see next slide)

Troubleshooting: gRPC session establishment (5/5)

This is a successful connection from MDT (EMSD) side:

```
RP/0/RP0/CPU0:NCS5502_bottom#sh tele model-driven trace go-info last 50
Nov 16 20:50:51.072 m2m/mdt/go-info 0/RP0/CPU0 6987# t24038 717 [mdt_go_trace_info]: dialReqAgent:376 1: req 501, entries when dequeue is 0
Nov 16 20:51:21.069 m2m/mdt/go-info 0/RP0/CPU0 t23319 724 [mdt_go_trace_info]: emsMdtConnEstablish:209 Send dial req of '10.30.110.38:57500' to retry dialer
Nov 16 20:51:21.069 m2m/mdt/go-info 0/RP0/CPU0 t23833 729 [mdt_go_trace_info]: queueDialRequest:363 1: req 501, dialer entries in queue 0.
Nov 16 20:51:21.069 m2m/mdt/go-info 0/RP0/CPU0 t23833 730 [mdt_go_trace_info]: mdtConnEstablish:96 1: Dialing out to 10.30.110.38:57500, req 501
Nov 16 20:51:21.069 m2m/mdt/go-info 0/RP0/CPU0 t23833 731 [mdt_go_trace_info]: mdtConnEstablish:134 1: req 501, TLS is false, cert , host .
Nov 16 20:51:21.069 m2m/mdt/go-info 0/RP0/CPU0 t24038 732 [mdt_go_trace_info]: mdtDialer:210 1: namespace /var/run/netns/global-vrf
Nov 16 20:51:21.069 m2m/mdt/go-info 0/RP0/CPU0 t24038 733 [mdt_go_trace_info]: getAddDialerChanWithLock:141 1:Found existing dialer for namespace /var/run/netns/global-vrf
Nov 16 20:51:21.069 m2m/mdt/go-info 0/RP0/CPU0 6993# t24038 734 [mdt_go_trace_info]: mdtDialer:222 1: Reuse dialerChan 0xc820088270 for /var/run/netns/global-vrf
Nov 16 20:51:21.072 m2m/mdt/go-info 0/RP0/CPU0 108529# t23834 735 [mdt_go_trace_info]: mdtConnEstablish:176 %!d(string=10.30.110.38:57500): %!s(int=1), chanstat buffered num 4000, gHardlimit 13000
Nov 16 20:51:21.072 m2m/mdt/go-info 0/RP0/CPU0 t23834 736 [mdt_go_trace_info]: mdtConnEstablish:179 mdtDialout: chanstat gResume 6000, condResume 200 alwaysResume 100
Nov 16 20:51:21.072 m2m/mdt/go-info 0/RP0/CPU0 t23834 745 [mdt_go_trace_info]: mdtConnEstablish:184 1: Ready for mdt dialout data, req 501, chan 1, 10.30.110.38:57500
Nov 16 20:51:21.072 m2m/mdt/go-info 0/RP0/CPU0 t23834 748 [mdt_go_trace_info]: dialReqAgent:376 1: req 501, entries when dequeue is 0
Nov 16 20:51:21.072 m2m/mdt/go-info 0/RP0/CPU0 t23834 760 [mdt_go_trace_info]: mdtSendAgent:467 Start a send agent for req 501, chan 1

===== for TLS setup you might also see this =====
Nov 17 15:53:32.305 m2m/mdt/go-info 0/RP0/CPU0 t28163 62609 [mdt_go_trace_info]: mdtConnEstablish:134 1: req 502, TLS is true, cert /misc/config/grpc/dialout/dialout.pem, host chkpt1.com.
```

Connection was successful and you see that channel number is the same as we saw from the previous output. At this point we can tell that gRPC works fine and we can move on.

Troubleshooting: BTE encoding of data

The final piece of internal processes within EMSd is BTE (the thread responsible for encoding). BTE initially will notify that it can't resolve your sensor path, but as soon as YFW gives the information about SysDB path, BTE consults the bag server and is able to resolve:

```
RP/0/RP0/CPU0:NCS5502_bottom#sh tele model-driven bte trace last 10
Nov 16 20:51:21.072 mdt/bte/errors 0/RP0/CPU0 t24155 Cannot find ucode for schema path /oper/clock/gl/system_uptime, client Cisco-IOS-XR-shellutil-oper:system-time/uptime
Nov 16 20:51:22.834 mdt/bte/info 0/RP0/CPU0 t24156 Successfully updated uCode entry for discriminant Cisco-IOS-XR-shellutil-oper:system-time/uptime with data size 888
```

Troubleshooting: How to read subscription (1/4)

There is a very convenient way to run a show command and get an almost complete picture about Telemetry on a router. Here is the CLI config for the explanation. Let's go step by step for the show output and describe all the lines.

```
telemetry model-driven
destination-group DGroup1
address-family ipv4 10.30.110.38 port 57500
encoding self-describing-gpb
protocol grpc no-tls
!
!
sensor-group health
sensor-path Cisco-IOS-XR-shellutil-oper:system-time/uptime
sensor-path Cisco-IOS-XR-ip-rib-ipv4-oper:rib/vrfs/vrf[vrf-name='Default']/afs/af/safs/saf/ip-rib-route-table-names/ip-rib-route-table-name/protocol/bgp/as/information
!
subscription health
sensor-group-id health sample-interval 5000
destination-id DGroup1
!
```

After basic explanation let's have a look at couple of examples of finding the source of problems.

Troubleshooting: How to read subscription (2/4)

```
RP/0/RP0/CPU0:NCS5502_bottom#sh tele m subscription health internal
Fri Nov 17 21:53:37.856 PST
Subscription: health
-----
State: Active
DSCP/Qos marked value: Default
Sensor groups:
Id: health
Sample Interval: 5000 ms
Sensor Path: Cisco-IOS-XR-shellutil-oper:system-time/uptime
Sensor Path State: Resolved
Sensor Path: Cisco-IOS-XR-ip-rib-ipv4-oper:rib/vrfs/vrf[vrf-name='Default']/afs/af/safs/saf/ip-rib-route-table-names/ip-rib-route-table-name/protocol/bgp/as/information
Sensor Path State: Resolved

Destination Groups:
Group Id: DGroup1
Destination IP: 10.30.110.38
Destination Port: 57500
Encoding: self-describing-gpb
Transport: grpc
State: Active
No TLS
Total bytes sent: 8010
Total packets sent: 45
Last Sent time: 2017-11-17 21:53:34.2034410100 -0800
```

Use "internal" to get access to the extended info

Subscription under attention

"Active" means all is good. If you see "Paused", it means at least one collection inside has issues with connectivity

it is possible to change QoS values for streamed telemetry data (per subscription)

configured sample interval

Resolved models in that subscription

Configured address and port of the collector

Encoding, transport and state

This is the sum from all collections in this subscription. Remember, that with TCP/UDP dial-out mode you have 12B for header added to each packet (you will see 12B per packet difference between here and collections)

Troubleshooting: How to read subscription (3/4)

Collection Groups:

Id: 3
Sample Interval:
Encoding:
Num of collection:
Collection time:
Total time:
Total Deferred:
Total Send Errors:
Total Send Drops:
Total Other Errors:
No data Instances:
Last Collection Start:
Last Collection End:
Sensor Path:

5000 ms

self-describing-gpb

45

Min: 7 ms Max: 13 ms

Min: 7 ms Avg: 9 ms Max: 13 ms

0

0

0

0

0

2017-11-17 21:53:34.2034393100 -0800

2017-11-17 21:53:34.2034403100 -0800

Cisco-IOS-XR-shellutil-oper:system-time/uptime

You may have different sensor groups under the subscription with different sample intervals

Number of collections done at the moment

MIN/MAX time to collect data from the EDM

MIN/MAX/AVG total time (collection time above plus encoding time in BTE plus send)

Info duplicated from fields below

Difference here b/w values should be in range of the collection time above

EDM path for your sensor path configured

Sysdb Path:

/oper/clock/gl/system_uptime

Count:

45 Method: GET Min: 7 ms Avg: 9 ms Max: 13 ms

Item Count:

45 Status: Active

Missed Collections:0 send bytes: 8010 packets: 45 dropped bytes: 0

success

errors

deferred/drops

Gets

45

0

List

0

0

Datalist

0

0

Finddata

0

0

GetBulk

0

0

Encode

0

0

Send

0

Gives you time as well as the method of collection involved (GET in this example)

Status is active – everything works fine

Information about collections, including missed ones (missed are seen when your collection time is more than configured sample interval. If you observe it, increase your sample interval)

Troubleshooting: How to read subscription (4/4)

```
Id: 4
Sample Interval:      5000 ms
Encoding:             self-describing-gpb
Num of collection:   45
Collection time:     Min:      6 ms Max:     13 ms
Total time:          Min:    10 ms Avg:    13 ms Max:    33 ms
Total Deferred:      0
Total Send Errors:   0
Total Send Drops:    0
Total Other Errors:  0
No data Instances:   0
Last Collection Start: 2017-11-17 21:53:34.2034560100 -0600
Last Collection End:  2017-11-17 21:53:34.2034570100 -0800
Sensor Path:         Cisco-IOS-XR-ip-rib-ipv4-oper-rib/rfs/vrf[vrf-name='Default']/afs/af/safs/saf/ip-rib-route-table-names/ip-rib-
route-table-name/protocol/bgp/as/information
```

```
Sysdb Path:          /oper/ipv4-rib/g1/vrf/*/afi/*/safi/*/table/*/proto/bgp/*/info
Count:                45 Method: FINDDATA Min: 6 ms Avg: 9 ms Max: 13 ms
Item Count:           0 Status: Filter Error
Missed Collections:  0 send bytes: 0 packets: 0 dropped bytes: 0
```

	success	errors	deferred/drops
Gets	0	0	0
List	225	0	0
Datalist	0	0	45
Finddata	0	0	0
GetBulk	0	0	0
Encode	0	0	0
Send	0	0	0

"Gets": when BackEnd thread asks for the data from the EDM in a one-by-one mode (not very efficient, but most of the EDMs work in this mode). Errors means something is bad within the data (go to the feature owner, not EMSd people)

Despite the name, **"list"** works the same as **"gets"** – information is collected one-by-one. Errors are seen when there is something wrong with the bag (or, as an example, when you don't have feature configured. Path will be resolved, MDT will send requests, but no data will be pushed – errors counter will be increasing)

Both **finddata** and **datalist** are used when MDT can request EDMs in blocks and have replies in blocks. Usually, it happens with paths which contain **"**"** (see the SysDB path for this Sensor Path). Errors here are seen because of Filter Error (there is no vrf-name "Default". Yes, it is case-sensitive). Sometimes, you might see **"**"** in the path, see Finddata or Datalist requests, and Gets/Lists counters also increasing. It means that EDM doesn't support bulk replies (MDT tried to get this way, but no luck).

Encode Errors are seen when there is a mismatch b/w internal versions of EMSd and EDM. **Encode deferred/drops** – when EDM bag is not registered in the Bag Server. **Send errors** also deal with code mismatch. **Send Drops** are increasing when remote server's buffer is full

GetBulk is something in the middle between Finddata/Datalist and Gets. It is applicable just for very few features and tries to optimize internal transport

Troubleshooting: Subscription Example #1 (1/2)

Collection Groups:

```
Id: 6
Sample Interval: 10000 ms
Encoding: self-describing-gpb
Num of collection: 2
Collection time: Min: 6 ms Max: 6 ms
Total time: Min: 10 ms Avg: 11 ms Max: 13 ms
Total Deferred: 0
Total Send Errors: 0
Total Send Drops: 0
Total Other Errors: 0
No data Instances: 2
```

```
Last Collection Start:2017-11-18 08:54:34.3039765436 -0800
Last Collection End: 2017-11-18 08:54:34.3039778436 -0800
```

```
Sensor Path: Cisco-IOS-XR-asic-errors-oper:asic-errors/nodes/node/asic-information/instances/instance/error-path/instance-summary
```

You see success in List and Finddata and errors there.. Hmm, what's going on? Let's dig into SysDB path to see

Sysdb Path:	/oper/asic-errors/nodes/*/*/instance/*/asic-errors/1,0
Count:	2 Method: FINDDATA Min: 6 ms Avg: 6 ms Max: 6 ms
Item Count:	0 Status: Paused
Missed Collections:	0 send bytes: 0 packets: 0 dropped bytes: 0 deferred/drops
	success errors deferred/drops
Gets	0
List	4
Datalist	0
Finddata	2
GetBulk	0
Encode	0
Send	0

Troubleshooting: Subscription Example #1 (2/2)

This is the SysDB path that was resolved for your YANG path (you take it from the output on previous page):

Sysdb Path: /oper/asic-errors/node/*/*/instance/*/asic-errors/1,0

Let's use "run sysdbcon -m" to find out the state:

```
RP/0/RP0/CPU0:NCS5502_bottom#run sysdbcon -m
Sat Nov 18 08:56:45.846 PST
Read-only sysdbcon
Please enter initial bind point>/
Enter 'help' or '?' for help
sysdbcon:[m]> list oper/asic-errors/node/
[dir] '0/'
sysdbcon:[m]> list oper/asic-errors/node/0/
[external dir] 'Jericho/'
sysdbcon:[m]> list oper/asic-errors/node/0/Jericho/
sysdbcon:[m]> list oper/asic-errors/node/0/Jericho/instance/
%SYSDB-SYSDB-4-SC_LIST : [Sysdbcon] While executing the list request the following error occurred:  
'sysdb' detected the 'warning' condition 'A verifier or EDM callback function returned: 'not found''
```

Okay, seems that we found the issue – we got resolved first two "stars" (between /node/ and /instance/) that gave us "success" in the output, but after that something went wrong. Result – nothing was sent out.

In such situations go to the feature owner and ask to fix this [please keep “exr-mdt@cisco.com” (mailer list) in copy] or remove from the platform list if that is not supposed to be supported.

Troubleshooting: Subscription Example #2 (1/3)

Collection Groups:

```
Id: 7
Sample Interval: 10000 ms
Encoding: self-describing-gpb
Num of collection: 0
Collection time: Min: 0 ms Max: 0 ms
Total time: Min: 0 ms Avg: 0 ms Max: 0 ms
Total Deferred: 2
Total Send Errors: 0
Total Send Drops: 0
Total Other Errors: 0
No data Instances: 0
Last Collection Start: 2017-11-18 09:09:04.3910294436 -0600
Last Collection End: 2017-11-18 09:09:04.3910307436 -0600
Sensor Path: Cisco-IOS-XR-dnx-port-mapper-oper:oor/nodes/node
```

	success	errors	deferred/drops
Gets	2	0	
List	2	0	
Datalist	0	0	
Finddata	2	0	
GetBulk	0	0	
Encode	0	0	
Send	0	0	

This one looks good, we have success with Gets and List and Finddata. But why don't we see the data out? And what is the Encode Drop here?

Let's start with SysDB path as before

Troubleshooting: Subscription Example #2 (2/3)

This is the SysDB path that was resolved for your YANG path (you take it from the output on previous page):

Sysdb Path: /oper/dpa/oor/node/*/summary

Let's use "run sysdbcon -m" to find out the state:

```
RP/0/RP0/CPU0:NCS5502_bottom#run sysdbcon -m
Sat Nov 18 09:14:21.502 PST
Read-only sysdbcon
Please enter initial bind point>/
Enter 'help' or '?' for help
sysdbcon:[m]> list oper/dpa/oor/node/
[external dir]      '1000/'
sysdbcon:[m]> get oper/dpa/oor/node/1000/summary
[bag (len 48)] 'oper/dpa/oor/node/1000/summary' (oor_summary v1.2 XDR)
'oor_summary'{
  [uint32 (len 4)]  'RED'      0 (0x00000000)
  [uint32 (len 4)]  'GREEN'    50 (0x00000032)
  [uint32 (len 4)]  'YELLOW'   0 (0x00000000)
}
[end bag]
sysdbcon:[m]> exit
```

Okay, seems that everything looks good, right?

Next step, let's check that the EDM bag is properly registered (bag name is in bold/red above)

Troubleshooting: Subscription Example #2 (3/3)

Let's check that bag is properly registered:

```
RP/0/RP0/CPU0:NCS5502_bottom#sh bag database bag oor_summary  
Sat Nov 18 09:15:10.569 PST  
Bag not found: oor_summary
```

And here it is. EDM contains all the data, but the bag is not registered and we can't send this data out.
In such situations go to the feature owner and ask to fix this [please keep "exr-mdt@cisco.com"(mailer list) in copy]

The main rule in troubleshooting is when you see errors and no data out, start with checking the SysDB path.
MDT uses the same data as you will see there. If you can't resolve the data – the problem is most probably there. In this case contact the feature owner. Then check that the bag is registered. Finally, make sure that the receiver has buffer and can process your streams (collector-end problems, this feedback is with gRPC only).

```
RP/0/RP0/CPU0:NCS5502_bottom#sh tele model-driven destination internal  
Sat Nov 18 09:44:54.721 PST
```

Dest Id	Sub	IP	Port	Trans	State Chan	Buff	Full
test	subs0	10.30.110.38	57500	grpc	1	62	0 0

Should be both "0"

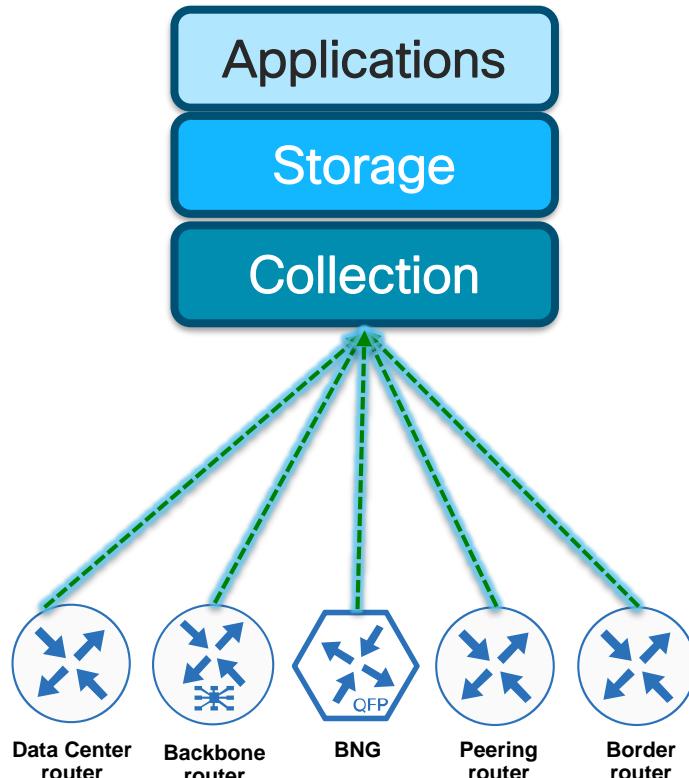
Collectors and analytics platforms

Introduction for this section

The goal of this section is to give you an overview of open sourced tools that you can use for your MDT demo, as well as how to install them and explain to interested parties.



A Very Basic Analytics Platform Architecture



When you start thinking about the receiving side of Telemetry, you might have this high-level architecture in mind.

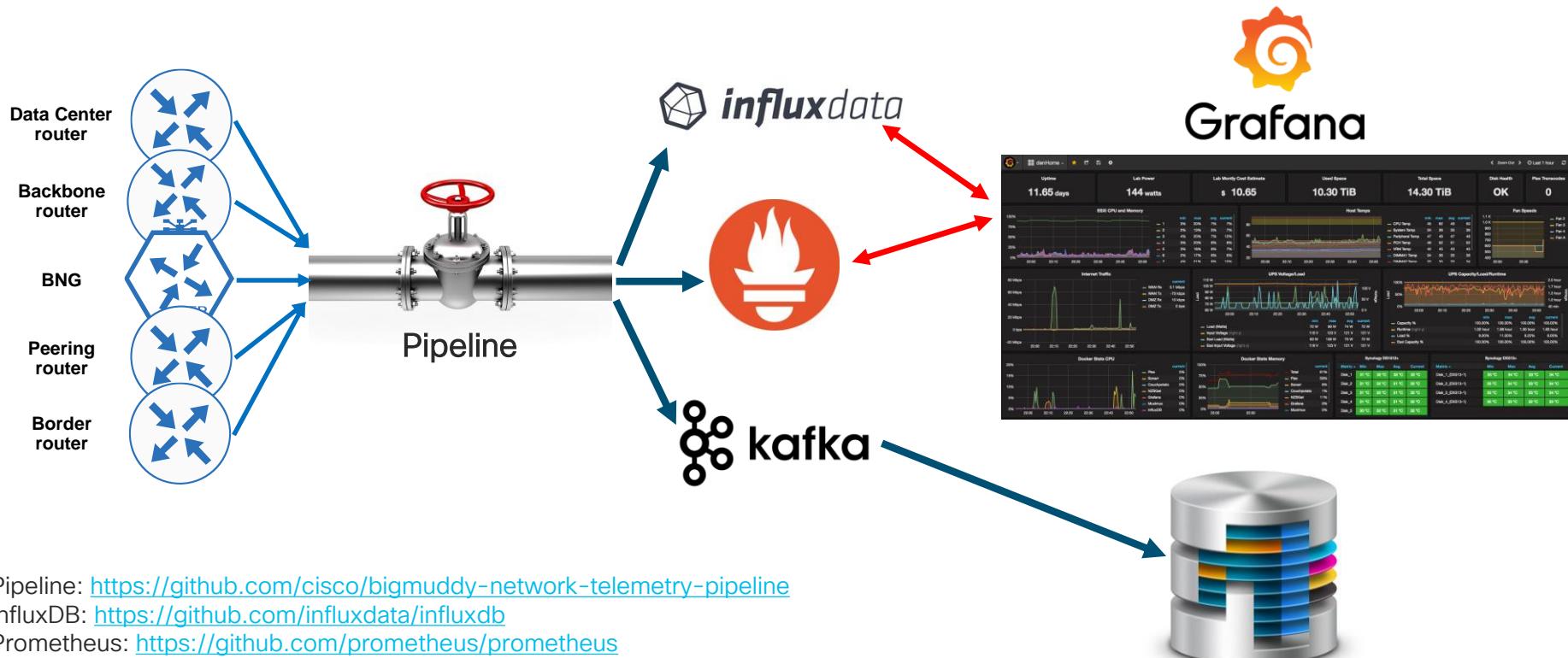
The first layer (Collection) – you need to collect all the streams and transform them from GPB/JSON into a format that will be supported by the layer above. Filtering can also happen here.

The second layer (Storage) is usually a TSDB ([time-series database](#)). The goal is to take the data from the layer below and keep it together with timestamps. You can also have other type of databases here (think of telemetry as a big data solution)

The third layer (Applications) is your “business logic” or tools that you will use with the data stored in a database.

Each layer can be an open-sourced solution, a commercial one or a home-grown solution.

An Open Source end-to-end solution from GitHub



TSDBs basic comparison

Data	InfluxDB	Prometheus	Elasticsearch
Website	https://influxdata.com/	https://prometheus.io/	https://www.elastic.co/products/elasticsearch
Category	Real-time Analytics	Monitoring System	Real-time Search
Supported Measurements	metrics, events	metrics	metrics, events
Sharding and Replication	Manual	Manual (supports federation)	Automatic
High Availability (HA)	Double writing 2 servers	Double writing 2 servers	Clustering
Underlying Technology	Golang	Golang	Java
Operational Complexity	Low (medium with HA)	Low	Medium
Storage Backend	Custom	Custom	Document
Supported Data Types	int64, float64, bool, and string	float64	string, int32, int64, float32, float64, bool, null
Bytes per point after compression	2.2	1.3	22
Metric Precision	nano second	milli second	milli second
Recording type	events	fixed interval	events
Write Performance - Single Node	470k metrics / sec (custom HW)	800k metrics / sec	30k metrics / sec
Write Performance - 5 Node Cluster	-	-	120k metrics / sec (calculated)
Query Performance (1 host, 12hr by 1m)	3.78 ms (min), 8.17 (avg)	tbd	13.23 ms (min), 28.6 (avg)
Query Performance	Medium to Fast	Moderate	Moderate
Query Language	InfluxQL (SQL like)	PromQL	Query DSL
Data Model	metric names, fields, labels	metric names, labels	metric names, labels
Query Language Functionality	4/5	5/5	3/5
Query Language Usability	5/5	4/5	3/5
Community Size	large	large	large
License	MIT	Apache 2	Apache 2
Latest Version	v1.3.5	v2.0.0-beta.2	v5
Maturity	Stable	Stable	Stable

Full table: <https://tinyurl.com/jsd4esy>

Good to read: <https://tinyurl.com/ybaw4ww6>

DB ranking: <https://tinyurl.com/ya8rrrjp>

InfluxDB vs Elasticsearch: <https://tinyurl.com/y7yxjf6v>

InfluxDB vs OpenTSDB: <https://tinyurl.com/y8ofbjyy>

InfluxDB vs Cassandra: <https://tinyurl.com/y83vv9ys>

Open telemetry stack: systems requirements



Grafana uses TSDB for the work, server requirements are small (250MB of RAM, 1CPU – the answer from the developer: <https://tinyurl.com/ybdhexqu>)



Prometheus
<https://prometheus.io>

Prometheus doesn't give exact requirements, but in many discussions shows 64GB of RAM and 32 CPU: <https://tinyurl.com/yd7l38ed>)



InfluxDB requirements:
<https://tinyurl.com/yd9f3ntj>

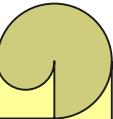
Server	CPU, cores	RAM, GB	IOPS
Low load [5k writes, 100k series, 5 queries per second]	2-4	2-4	500
Moderate load [250k writes, 1m series, 25 queries per second]	4-6	8-32	500-1000
High load [>250k writes, >1m series, >25 queries per second]	8+	32+	1000+

Pipeline

Pipeline is a lightweight program, written in Go (with concurrency). You can also start several instances [on a single server] if needed.

BGP Graceful Maintenance

BGP Graceful Maintenance overview



Advertise routes with a lower preference to allow alternate routes to take over before taking down a link or router.

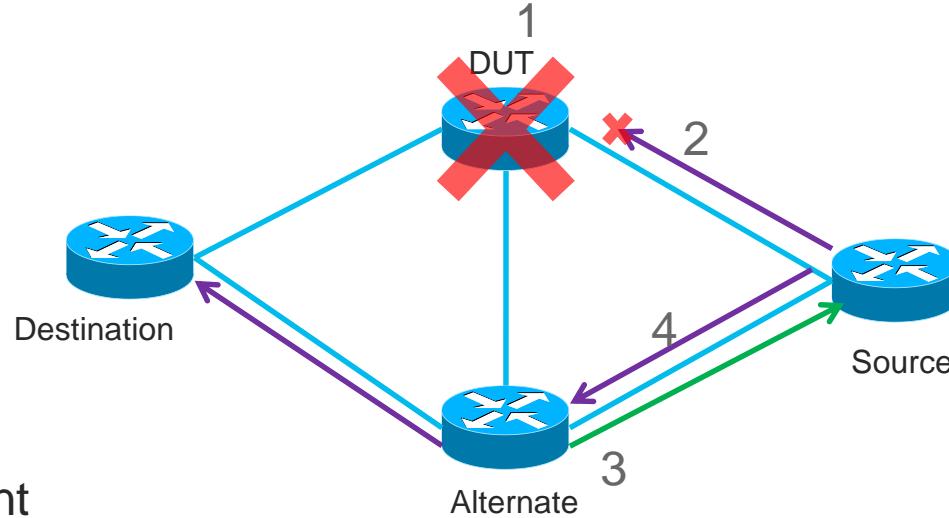
This avoids transient incorrect routing during convergence when a link or router is taken out of service.

This is important when BGP convergence takes many seconds or even minutes.

The Problem

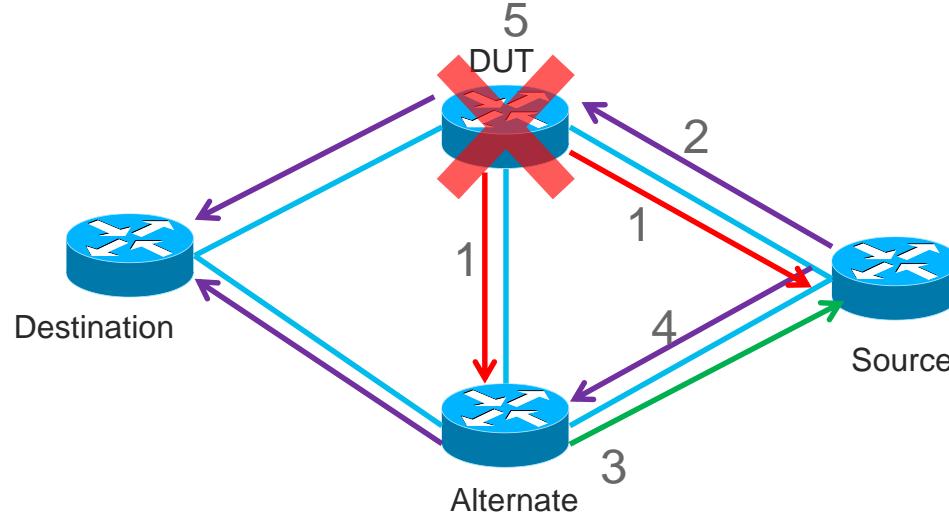
1. Take down DUT
2. Packets drop
3. Alternate announces
4. Packets resume

A router is unable to forward to a destination if a withdrawal arrives before the advertisement of its alternate route



The Solution

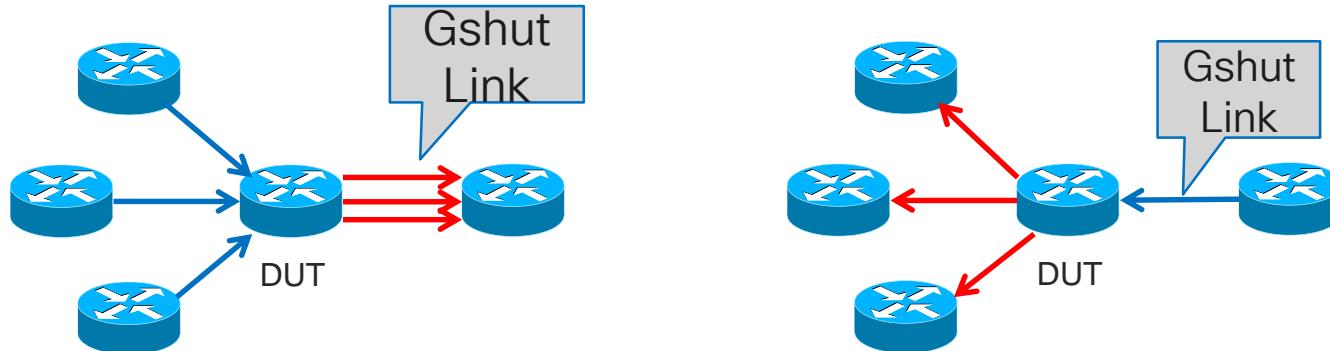
1. Announce Lower Preference
2. Packets continue
3. Alternate announces
4. Packets reroute
5. Take down DUT



Putting a BGP session into Graceful Maintenance

Two things happen

- Routes from that neighbor are sent to others with GSHUT
- Routes to that neighbor are sent with GSHUT



→ Regular route advertisement

→ Route with GSHUT community and possibly other preference lowering attributes added

GSHUT attribute vs. community

Routes received from a GSHUT neighbor are marked with a GSHUT attribute to distinguish them from routes received with the GSHUT community.

When a neighbor is taken out of maintenance, the attribute on its paths is removed, but not the community.

```
RP/0/0/CPU0:R1#sh bgp 5.5.5.5/32
...
Local, (Received from a RR-client)
12.12.12.5 from 12.12.12.5 (192.168.0.5)
    Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best, import-candidate, graceful-shut
    Received Path ID 0, Local Path ID 1, version 13
...
...
```

Attribute

```
RP/0/0/CPU0:R4#sh bgp 5.5.5.5
...
10.10.10.1 from 10.10.10.1 (192.168.0.5)
    Received Label 24000
    Origin incomplete, metric 0, localpref 100, valid, internal, best, group-best, import-candidate
    Received Path ID 0, Local Path ID 1, version 4
    Community: graceful-shutdown
    Originator: 192.168.0.5, Cluster list: 192.168.0.1
...

```

Community

Lowering the route preference

By default, just add the GSHUT community

Optionally, can also add LOCAL_PREF and AS_PATH prepends.

Note: This is done after outbound policy is applied.

```
neighbor 666.0.0.1
  remote-as 65000
  graceful-maintenance
  local-preference 4
  as-prepends 3
```

Any routes with the GSHUT attribute

AND

All routes if this neighbor has graceful-maintenance activated

will be sent to this neighbor with

GSHUT community, LOCAL_PREF=4 and 3 AS_PATH prepends.

(LOCAL_PREF will not be sent to EBGP neighbors)

Activating graceful-maintenance

Globally: for all neighbors

```
router bgp 65000  
  graceful-maintenance activate all-neighbors retain-routes
```

Without this, only neighbors with graceful-maintenance configuration will activate

RIB will retain BGP routes when BGP process is stopped

Activate a single neighbor

```
neighbor 666.0.0.1  
  remote-as 65000  
  graceful-maintenance  
    activate
```

Activate a whole neighbor group

```
neighbor-group test  
  graceful-maintenance  
    activate
```

graceful-maintenance and Graceful Shutdown

The “graceful-maintenance activate” configuration does not disappear after shutdown.

It also works at session bringup time.

Use the “no” keyword to remove graceful-maintenance activation after bringup is complete.

Removing the global activation

```
RP/0/0/CPU0:R1(config)#router bgp 1  
RP/0/0/CPU0:R1(config-bgp)#no graceful-maintenance
```

For a single neighbor

```
RP/0/0/CPU0:R1(config-bgp)# neighbor 666.0.0.1  
RP/0/0/CPU0:R1(config-bgp-nbr)#graceful-maintenance  
RP/0/0/CPU0:R1(config-nbr-gshut)#no activate
```

When to Shutdown

A router or link can be shut down after the network has converged as a result of a graceful-maintenance activation.

Convergence can take from less than a second up to more than an hour.

Unfortunately, a single router cannot know when a whole network has converged.

After graceful-maintenance activation, it can take a few seconds to start sending updates. Then The “InQ” and “OutQ” of neighbors in the “show bgp summary” command indicates the level of BGP messaging.

Neighbors should stop sending traffic to this router. If they do not have alternate routes, they will not stop.

Don't forget

Sending GSHUT to another AS can cause churn in far away networks

If this router is the only source of a route, then GSHUT does not help.

`send-community-gshut-ebgp` must be configured under neighbor address family of an EBGP neighbor for this router to add the GSHUT community.

This does not affect a GSHUT community that was received on a route from another source. For that, use `send-community-ebgp`.

```
RP/0/0/CPU0:R1(config-bgp)# neighbor 666.0.0.1
RP/0/0/CPU0:R1(config-bgp-nbr)/address-family ipv4 unicast
RP/0/0/CPU0:R1(config-bgp-nbr-af)send-community-gshut-ebgp
```

BGP Graceful Maintenance -ENHANCEMENTS

- Current implementation allows graceful maintenance to be activated either for single BGP neighbor or for all the BGP neighbors
- To simplify the use of BGP GSHUT, we added more granularity by enabling graceful maintenance at:
 - interface (physical and logical) level and
 - line-card location level

Note: applicable for all the directly connected e-BGP neighbor sessions

CSCvn44072- graceful draining of bgp routes through CLI at linecard and interface level

CSCvo66735- Need a CLI to display bgp graceful maintenance state

Feature Scope

- i-BGP and e-BGP multi-hop neighbors are not supported
- e-BGP neighbor session over the bundle interface –
 - BGP ignores location level configuration for neighbor session going over bundle interface
 - User is required to configure the graceful maintenance for the neighbor at interface level
- Transition from multi-hop to non multi-hop DC e-BGP neighbor not supported
 - If neighbor needs graceful maintenance, user needs to remove and add the neighbor config

Configuration

```
\  
RP/0/RP0/CPU0:xrg-301-ncs5508(config-bgp)#graceful-maintenance ?  
  activate All neighbors with graceful-maintenance config  
RP/0/RP0/CPU0:xrg-301-ncs5508(config-bgp)#graceful-maintenance activate  
RP/0/RP0/CPU0:xrg-301-ncs5508(config-gshut-activate)#[br/>  all-neighbors Also neighbors without graceful-maintenance config  
interface      Enable graceful-maintenance on all neighbors whose session is going over this interface  
location       Enable graceful-maintenance on all neighbors whose session is going over this line card location  
retain-routes  Keep BGP routes in RIB once BGP process stops  
<cr>  
  
RP/0/0/CPU0:R3(config-bgp)#graceful-maintenance activate  
RP/0/RP0/CPU0:xrg-301-ncs5508(config-gshut-activate)#location ?  
  0/0/CPU0    Configure line card location  
  0/1/CPU0    Configure line card location  
  0/RP0/CPU0  Configure line card location  
  0/RP1/CPU0  Configure line card location  
  WORD        Configure line card location  
  
RP/0/RP0/CPU0:xrg-301-ncs5508(config-bgp)#graceful-maintenance activate  
RP/0/RP0/CPU0:xrg-301-ncs5508(config-gshut-activate)#interface ?  
  BVI          Bridge-Group Virtual Interface  
  Bundle-Ether Aggregated Ethernet interface(s) | short name is BE  
  Bundle-POS   Aggregated POS interface(s) | short name is BP  
  FastEthernet FastEthernet/IEEE 802.3 interface(s) | short name is Fa  
  FiftyGigE    FiftyGigabitEthernet/IEEE 802.3 interface(s) | short name is  
  .....  
  .....  
  .....  
<all the logical and physical interfaces on the router>
```

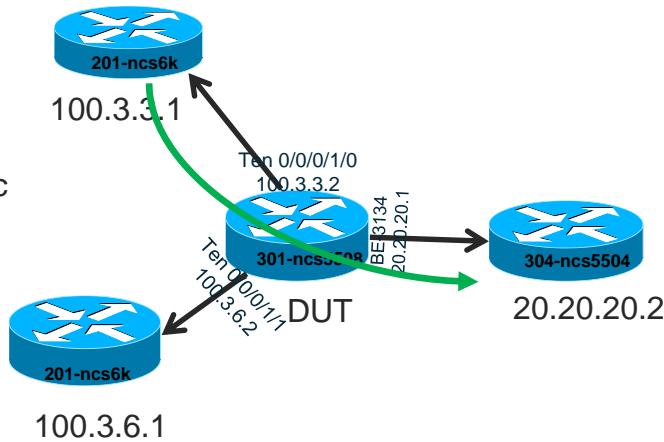
New graceful-maintenance activate submode

New CLI options

Initiating Graceful Maintenance on the DUT with Physical Interface

Source on 201-ncs8k
Destination on 304-ncs5504

Before activating GSHUT on DUT, traffic goes via DUT link Ten 0/0/0/1/0



Initiating Graceful Maintenance on the DUT with Physical Interface

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh bgp sessions
```

```
Sat Mar 2 06:07:03.176 UTC
```

Neighbor	VRF	Spk	AS	InQ	OutQ	NBRState	NSRState
20.20.20.2	default	0	6534	0	0	Established	None
100.3.3.1	default	0	65001	0	0	Established	None
100.3.6.1	default	0	65001	0	0	Established	None

```
RP/0/RP1/CPU0:xrg-301-ncs5508#
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh bgp neighbor 100.3.3.1 | inc Handle
```

```
Sat Mar 2 06:12:31.445 UTC
```

```
Local host: 100.3.3.2, Local port: 179, IF Handle: 0x000000270
```

BGP neighbor 100.3.3.1
session using interface
Ten0/0/0/1/0

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh im database ifhandle 0x000000270
```

```
Sat Mar 2 06:12:43.639 UTC
```

```
View: OWN - Owner, L3P - Local 3rd Party, G3P - Global 3rd Party, LDP - Local Data Plane
```

```
GDP - Global Data Plane, RED - Redundancy, UL - UL
```

```
Node 0/0/CPU0 (0x0)
```

```
Interface TenGigE0/0/0/1/0, ifh 0x000000270 (up, 1514)
```

```
Interface flags: 0x0000000000110049f (ROOT_IS_HW|IFCONNECTOR  
|IFINDEX|BROADCAST|CONFIG|HW|VIS|DATA|CONTROL)
```

```
Encapsulation: ether
```

```
Interface type: IFT_TENGETHERNET
```

```
Control parent: None
```

```
Data parent: None
```

```
Views: GDP|LDP|L3P|OWN
```

```
Protocol Caps (state, mtu)
```

```
-----
```

```
None qos_out (up, 1514)
```

```
None ether (up, 1514)
```

```
arp arp (up, 1500)
```

```
ipv4 ipv4 (up, 1500)
```

```
ether_sock ether_sock (up, 1500)
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508#
```

Initiating Graceful Maintenance on the DUT with Physical Interface

```
RP/0/RP0/CPU0:xrg-201-ncs6k#sh bgp 34.34.34.34
Fri Mar 1 22:05:55.061 UTC
BGP routing table entry for 34.34.34.34/32
Versions:
  Process          bRIB/RIB  SendTblVer
  Speaker          177        177
Last Modified: Aug 19 17:42:06.558 for 27w5d
Paths: (2 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
    Path #1: Received by speaker 0
    Advertised to update-groups (with more than one peer):
      0.2
      65004 6534
      100.3.3.2 from 100.3.3.2 (1.1.1.1) 
        Origin IGP, localpref 100, valid, external, best, group-best, multipath
        Received Path ID 0, Local Path ID 0, version 177
        Origin-AS validity: not-found
    Path #2: Received by speaker 0
    Not advertised to any peer
    65004 6534
      100.3.6.2 from 100.3.6.2 (1.1.1.1)
        Origin IGP, localpref 100, valid, external, multipath
        Received Path ID 0, Local Path ID 0, version 0
        Origin-AS validity: not-found
RP/0/RP0/CPU0:xrg-201-ncs6k#
```

On 201-ncs6k, bestpath
points to path from
100.3.3.2

Initiating Graceful Maintenance on the DUT with Physical Interface

Before activating gshut, make sure on DUT, neighbor is configured with send-community-ebgp for the address family

```
router bgp 65004
neighbor 100.3.3.1
remote-as 65001
address-family ipv4 unicast
send-community-ebgp
route-policy ALLOW_ALL in
route-policy ALLOW_ALL out
!
!
!
```

On the neighbor configure a route policy matching on the GSHUT community to lower the preference of those routes received with gshut community

```
route-policy gshut
if community matches-any gshut then
    set local-preference 88
endif
pass
end-policy
```

```
neighbor 100.3.3.2
address-family ipv4 unicast
route-policy gshut in
```

Initiating Graceful Maintenance on the DUT with Physical Interface

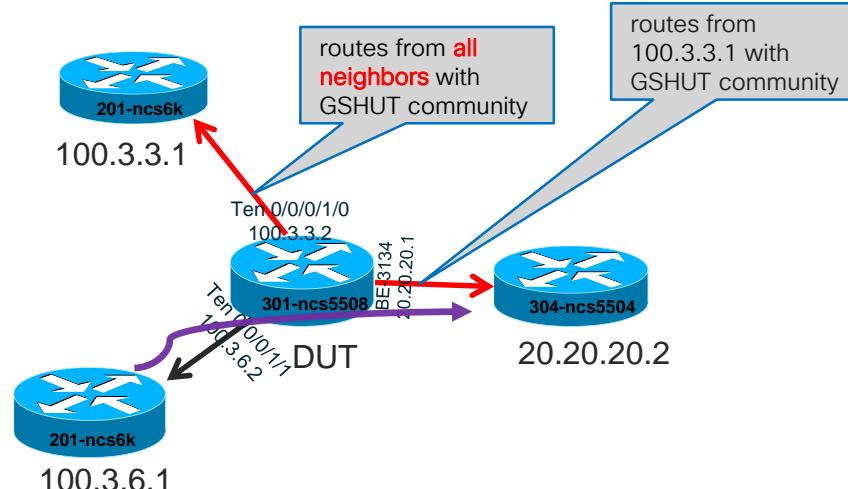
Global
configuration :
interface option

```
router bgp 65000
  graceful-maintenance activate interface interface Ten 0/0/0/1/0
```

Source on 201-ncs8k
Destination on 304-ncs5504

On DUT, neighbor 100.3.3.1 graceful activated due to interface config

Alternate path chosen for the traffic
Traffic goes via DUT link
Ten0/0/0/1/1



Route advertisement with GSHUT community and possibly other preference lowering attributes added

Initiating Graceful Maintenance on the DUT with Physical Interface

```
RP/0/RP1/CPU0:xrg-301-ncs5508(config)#router bgp 65004
RP/0/RP1/CPU0:xrg-301-ncs5508(config-bgp)#graceful-maintenance activate
RP/0/RP1/CPU0:xrg-301-ncs5508(config-gshut-activate)#interface tenGigE 0/0/0/1/0
RP/0/RP1/CPU0:xrg-301-ncs5508(config-gshut-activate)#commit
Sat Mar  2 06:23:28.468 UTC
RP/0/RP1/CPU0:xrg-301-ncs5508(config-gshut-activate)#
RP/0/RP1/CPU0:xrg-301-ncs5508#sh bgp neighbor 100.3.3.1 | inc Grace
Sat Mar  2 06:27:25.225 UTC
Graceful Maintenance active without neighbor config, Interface cfg applied
RP/0/RP1/CPU0:xrg-301-ncs5508#
```

Due to interface config, bgp neighbor 100.3.3.1 got gshut activated

On 201-nca6k
Check 201-ncs6k receives routes with gshut community and bestpath changes to alternate path

```
RP/0/RP0/CPU0:xrg-201-ncs6k#sh bgp 34.34.34.34
Fri Mar  1 22:38:29.693 UTC
BGP routing table entry for 34.34.34.34/32
Versions:
  Process          bRIB/RIB  SendTblVer
  Speaker           179      179
Last Modified: Aug 19 17:58:59.558 for 27w5d
Paths: (2 available, best #2)
  Advertised to update-groups (with more than one peer):
    0.2
    Path #1: Received by speaker 0
    Not advertised to any peer
    65004 6534
      100.3.3.2 from 100.3.3.2 (1.1.1.1)
        Origin IGP, localpref 88, valid, external
        Received Path ID 0, Local Path ID 0, version 0
        Community: graceful-shutdown
        Origin-AS validity: not-found
    Path #2: Received by speaker 0
    Advertised to update-groups (with more than one peer):
      0.2
      65004 6534
        100.3.6.2 from 100.3.6.2 (1.1.1.1)
          Origin IGP, localpref 100, valid, external, best, group-best
          Received Path ID 0, Local Path ID 0, version 179
          Origin-AS validity: not-found
RP/0/RP0/CPU0:xrg-201-ncs6k#
```

GSHUT community received in the route and local pref 88 set for the path

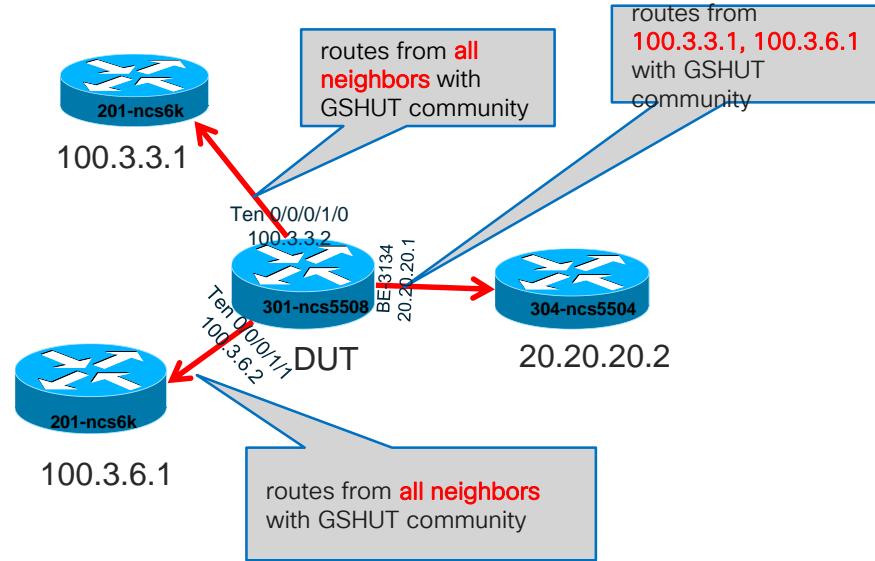
Best path changes to alternate path

Initiating Graceful Maintenance on the DUT with Line-Card Location

Global configuration :
interface option

```
router bgp 65000  
graceful-maintenance activate location 0/0/cpu0
```

On DUT, neighbor 100.3.3.1, 100.3.6.1 graceful activated due to location config
Neighbor 20.20.20.2 did not get activated since session with bundle interface is ignored for location option



→ Route advertisement with GSHUT community and possibly other preference lowering attributes added

Initiating Graceful Maintenance on the DUT with Line-Card Location

```
BGP neighbor 100.3.3.1 and 100.3.3.2 sessions are going over line card location 0/0/cpu0
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh bgp sessions
```

```
Sat Mar 2 06:07:03.176 UTC
```

Neighbor	VRF	Spk	AS	InQ	OutQ	NBRState	NSRState
20.20.20.2	default	0	6534	0	0	Established	None
100.3.3.1	default	0	65001	0	0	Established	None
100.3.6.1	default	0	65001	0	0	Established	None

```
RP/0/RP1/CPU0:xrg-301-ncs5508#
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh bgp neighbor 100.3.3.1 | inc Handle
```

```
Sat Mar 2 06:12:31.445 UTC
```

```
Local host: 100.3.3.2, Local port: 179, IF Handle: 0x000000270
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh im database ifhandle 0x000000270 | inc ifh
```

```
Sat Mar 2 06:12:43.639 UTC
```

```
Interface TenGigE0/0/0/1/0, ifh 0x000000270 (up, 1514)
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh bgp neighbor 100.3.6.1 | inc Handle
```

```
Sat Mar 2 06:12:31.445 UTC
```

```
Local host: 100.3.3.2, Local port: 179, IF Handle: 0x000000278
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh im database ifhandle 0x000000278 | inc ifh
```

```
Sat Mar 2 06:12:43.639 UTC
```

```
Interface TenGigE0/0/0/1/1, ifh 0x000000278 (up, 1514)
```

Initiating Graceful Maintenance on the DUT with Line-Card Location

On DUT

```
RP/0/RP1/CPU0:xrg-301-ncs5508(config)#router bgp 65004
RP/0/RP1/CPU0:xrg-301-ncs5508(config-bgp)#graceful-maintenance activate
RP/0/RP1/CPU0:xrg-301-ncs5508(config-gshut-activate)#location 0/0/cpu0
RP/0/RP1/CPU0:xrg-301-ncs5508(config-gshut-activate)#commit
RP/0/RP1/CPU0:xrg-301-ncs5508(config-gshut-activate)#+
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh bgp neighbor 100.3.3.1 | inc Grace
Sat Mar 2 06:27:25.225 UTC
```

Graceful Maintenance active without neighbor config, Location cfg applied

Due to location config, bgp neighbor 100.3.3.1
got gshut activated

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh bgp neighbor 100.3.6.1 | inc Grace
Sat Mar 2 06:27:25.225 UTC
```

Graceful Maintenance active without neighbor config, Location cfg applied

Due to location config, bgp neighbor 100.3.6.1
got gshut activated

On 201-ncs6k

```
RP/0/RP0/CPU0:xrg-201-ncs6k#sh bgp 34.34.34.34
routing table entry for 34.34.34.34/32
...
Paths: (2 available, best #2)
  Advertised to update-groups (with more than one peer):
    0.2
  Path #1: Received by speaker 0
  Not advertised to any peer
  65004 6534
    100.3.3.2 from 100.3.3.2 (1.1.1.1)
      Origin IGP, localpref 88, valid, external
      Received Path ID 0, Local Path ID 0, version 0
      Community: graceful-shutdown
      Origin-AS validity: not-found
  Path #2: Received by speaker 0
  Advertised to update-groups (with more than one peer):
    0.2
  65004 6534
    100.3.6.2 from 100.3.6.2 (1.1.1.1)
      Origin IGP, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 0, version 185
      Community: graceful-shutdown
      Origin-AS validity: not-found
```

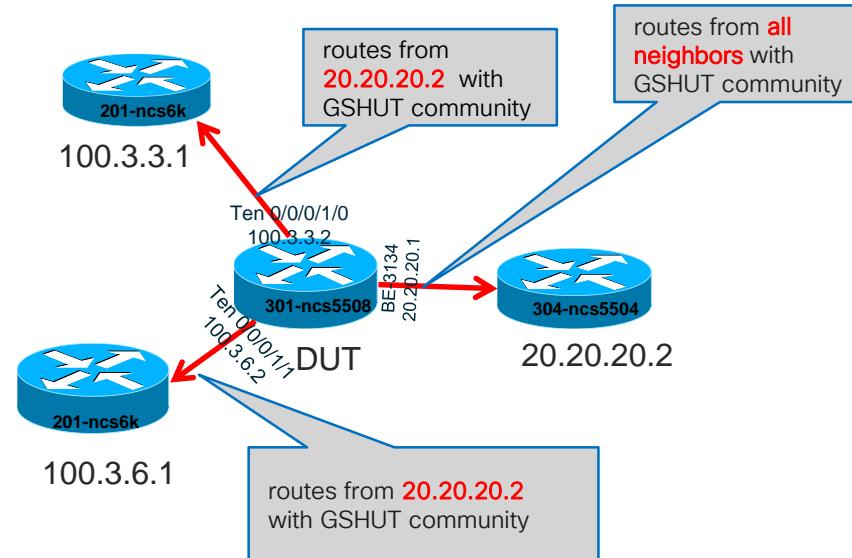
GSHUT community received in the
route received from both 100.3.3.2
and 100.3.6.2

Initiating Graceful Maintenance on the DUT with Bundle Interface

Global configuration :
interface option

```
router bgp 65000  
graceful-maintenance activate interface Bundle-Ether 3134
```

On DUT, neighbor 20.20.20.2
graceful activated due to bundle
interface config



→ Route advertisement with GSHUT community and possibly other preference lowering attributes added

Initiating Graceful Maintenance on the DUT with Bundle Interface

```
BGP neighbor 20.20.20.2 session is going over bundle interface
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508#sh bgp sessions
```

```
Sat Mar 2 06:07:03.176 UTC
```

Neighbor	VRF	Spk	AS	InQ	OutQ	NBRState	NSRState
20.20.20.2	default	0	6534	0	0	Established	None
100.3.3.1	default	0	65001	0	0	Established	None
100.3.6.1	default	0	65001	0	0	Established	None

```
RP/0/RP1/CPU0:xrg-301-ncs5508#
```

```
RP/0/RP0/CPU0:xrg-301-ncs5508#sh bgp neighbor 20.20.20.2 | inc Handle
```

```
Sat Mar 2 07:08:31.120 UTC
```

```
Local host: 20.20.20.1, Local port: 179, IF Handle: 0x080000d4
```

```
RP/0/RP0/CPU0:xrg-301-ncs5508#
```

```
RP/0/RP0/CPU0:xrg-301-ncs5508#sh im database ifhandle 0x080000d4 | inc ifh
```

```
Sat Mar 2 07:08:25.212 UTC
```

```
Interface Bundle-Ether3134, ifh 0x080000d4 (up, 1514)
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508(config)#router bgp 65004
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508(config-bgp)#graceful-maintenance activate
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508(config-gshut-activate)#interaface bundle-ether 3134
```

```
RP/0/RP1/CPU0:xrg-301-ncs5508(config-gshut-activate)#commit
```

```
RP/0/RP0/CPU0:xrg-301-ncs5508#sh bgp neighbor 20.20.20.2 | inc Grace
```

```
Sat Mar 2 07:11:39.110 UTC
```

```
Graceful Maintenance active without neighbor config, Interface cfg applied
```

```
RP/0/RP0/CPU0:xrg-301-ncs5508#
```

ASR9000 Load Balancing

ASR9000 Load Balancing Modalities

Dynamic
(Traffic based)

Static
(port, VC, etc. based)

NP μcode hash

PHY hash

Ingress NP

- ECMP/UCMP path selection
- LAG member selection
- Fabric link selection (multicast only)

Typhoon 100G PHY

SGMII link selection (ingress only)

Round robin

Ingress FIA

Fabric link selection

Per L2VPN VC

Ingress NP

- ECMP/UCMP path selection
- LAG member selection

Port hash

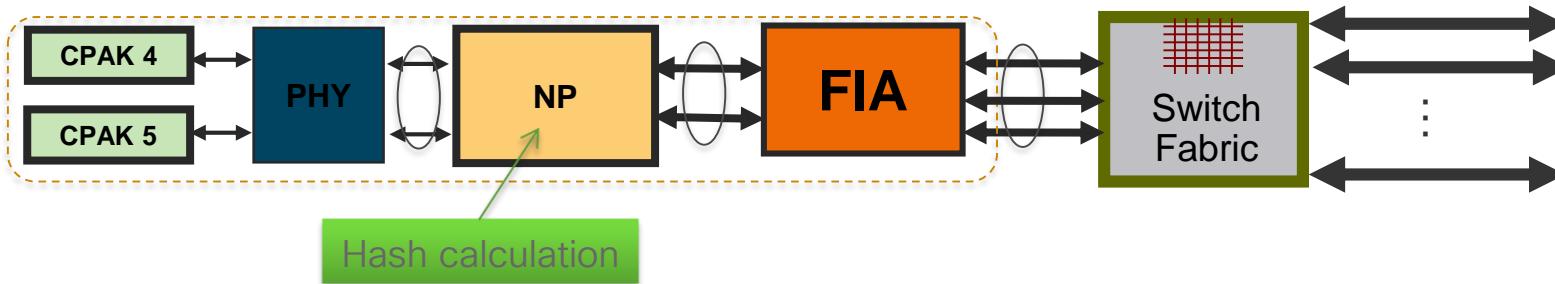
Satellite ICL Bundle

ICL bundle member selection

(Default for L2VPN; other LB algorithms also available)

ASR9k Load-balancing Essentials

NP µcode hash

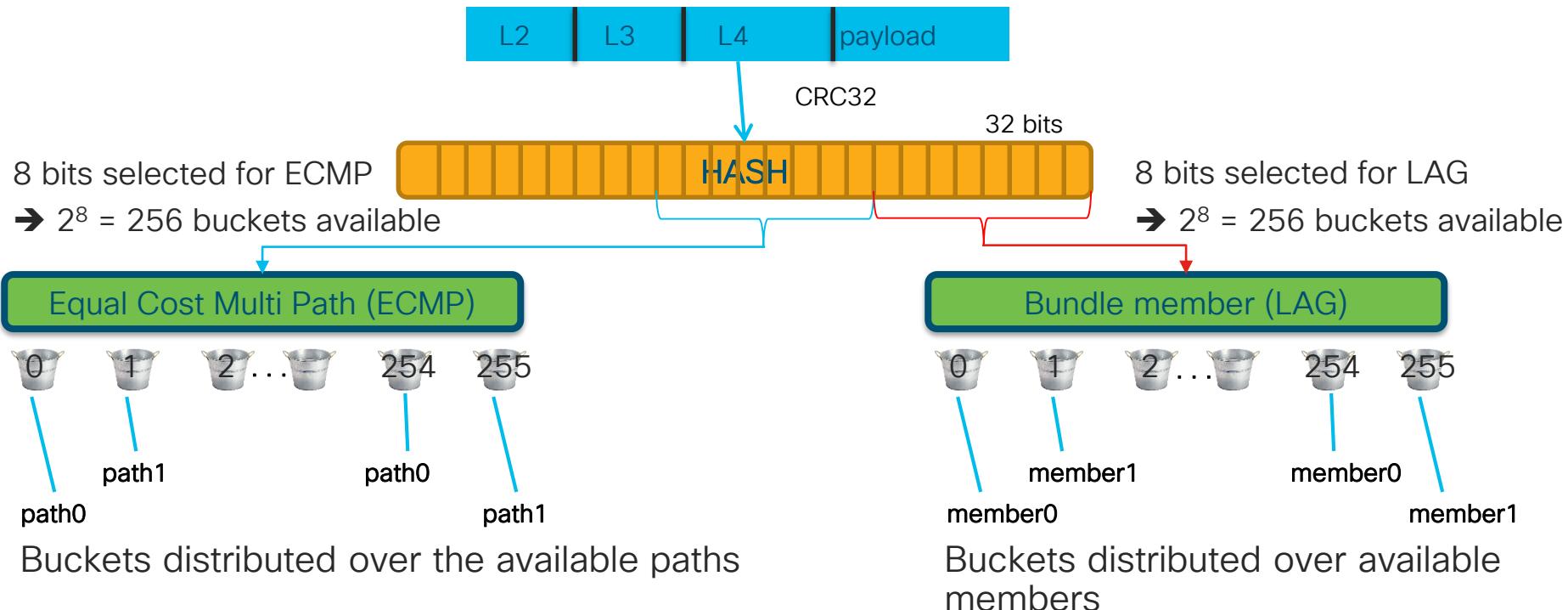


- Every packet received on an NPU undergoes a 32-bit CRC based **hash** computation.
 - Fields used as input for the hash calculation depend on packet encapsulation
- Different sets of **8 bits** of the hash are used for different purposes:
 - Bundle member selection
 - Non recursive Path (IGP/OSPF/ISIS)
 - Recursive path (eg bGP)

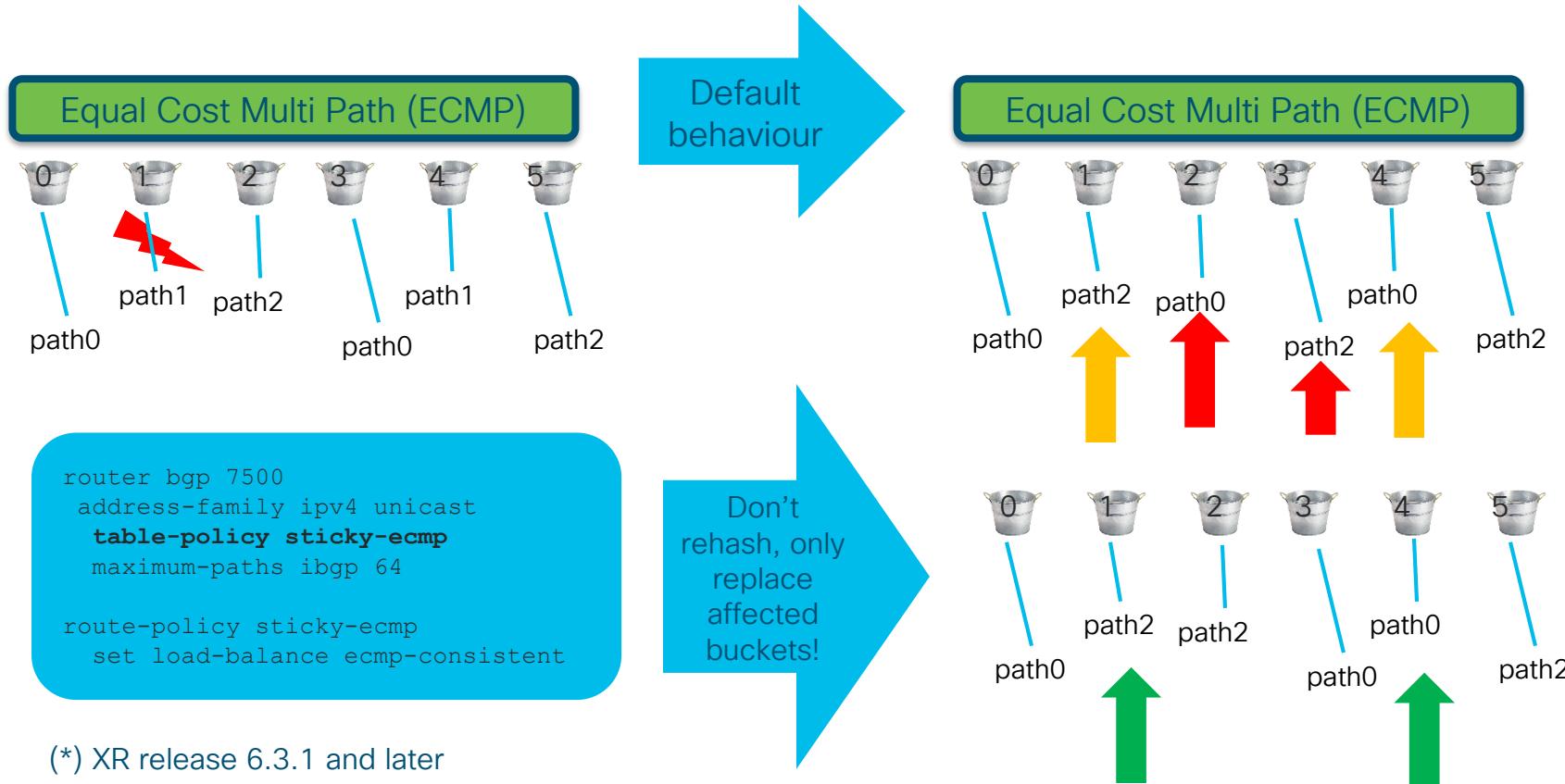
```
→ HASH_LAG_MASK(x) ((x) & 0xFF)
→ HASH_IGP_MASK(x) (((x) & 0xFF00) >> 8)
→ HASH_BGP_MASK(x) (((x) & 0x1FE000) >> 13)
```

ASR9k Load-balancing Essentials

Concept of a hash and its usage: ECMP and LAG

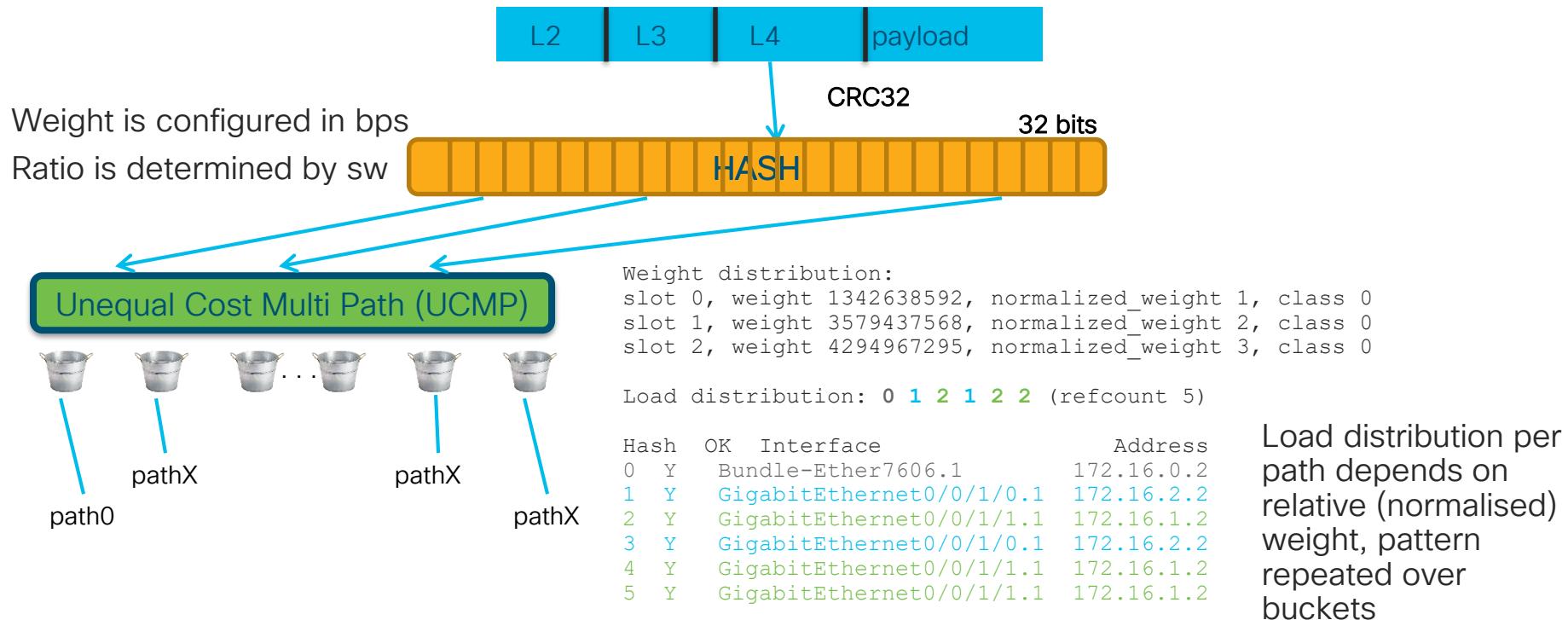


Rehashing consequences and sticky ECMP*



ASR9k Load-balancing Essentials

Concept of a hash and its usage: UCMP



ECMP/UCMP Hash Input

A: IPv4 Unicast or IPv4 to MPLS

- No or unknown Layer 4 protocol: IP SA, DA and Router ID
- UDP or TCP: IP SA, DA, Src Port, Dst Port and Router ID

B: IPv4 Multicast

- For (S,G): Source IP, Group IP, next-hop of RPF
- For (*,G): RP address, Group IP, next-hop of RPF

C: MPLS to MPLS or MPLS to IPv4

- # of labels <= 4:
 - Payload is IP: same as IPv4 unicast
 - Payload is other: 4th label, Router ID
- # of labels > 4 : 4th label and Router ID

Bundle Hash Input

L3 bundle:

- follows “A” or “C”, depending on packet type

L2 access bundle:

- SMAC, DMAC, RID (default)
- IP SA, IP DA, RID (if configured under I2vpn)

MPLS enabled core facing bundle with L2 access:

- PW label (default)
- SMAC, DMAC, RID (if configured under I2vpn)
- IP SA, IP DA, RID (if configured under I2vpn)

L2VPN vs L3VPN MPLS Loadbalancing

- When a labeled packet arrives on the interface the ASR9000 advances a pointer for at max 4 labels.
- If the number of labels ≤ 4 and the next nibble seen right after that label is
 - 4: default to IPv4 based balancing
 - 6: default to IPv6 based balancing
- This means that if you have a P router that has no knowledge about the MPLS service of the packet, that nibble can either mean the IP version (in MPLS/IP) or it can be the DMAC (in EoMPLS).
- RULE: If you have EoMPLS services AND macs are starting with a 4 or 6. You HAVE to use Control-Word



- Control Word inserts additional zeros after the inner label showing the P nodes to go for label based balancing.
- In EoMPLS, the inner label is VC label. So LB per VC then. More granular spread for EoMPLS can be achieved with FAT PW (label based on FLOW inserted by the PE device who owns the service).
 - Take note of the knob to change the code: PW label code 0x11 (17 dec, as per draft specification). (IANA assignment is 0x17)

Using CRC for hashing vs MD5

CRC

```
uint32_t  
calculate_crc32 (uint8_t *buffer, uint16_t size)  
{  
    uint32_t accumulator;  
    uint16_t i;  
  
    accumulator = 0xffffffff;  
  
    for (i = 0; i < size; i++) {  
        accumulator = (accumulator << 8) ^ crc_table\  
                      [((accumulator >> 24) & 0xff) ^  
buffer[i]];  
    }  
  
    return (~accumulator);  
}
```

MD5

```
MD5Update (context, input, inputLen)  
{  
    MD5_CTX         *context;          /* context */  
    unsigned char   *input;            /* input block */  
    unsigned int    inputLen;          /* length of input  
block */  
    {  
        unsigned int i, indx, partLen;  
  
        /* Update number of bits */  
        if ((context->count[0] += ((UINT4)inputLen << 3))  
            < ((UINT4)inputLen << 3))  
            context->count[1]++;  
        context->count[1] += ((UINT4)inputLen >> 29);  
  
        partLen = 64 - indx;  
  
        /* Transform as many times as possible. */  
        if (inputLen >= partLen) {  
            MD5_memcpy  
                ((POINTER)&context->buffer[indx], (POINTER)input, partLen);  
            MD5Transform (context->state, context->buffer);  
            .....  
        }  
    }  
}
```

```
xthuijs-ads:easy_md5 xthuijs$ cat md5.c | wc -l  
367
```

→ 11 lines vs 367

Different results between MD5 and CRC also

CRC 32

```
XTHUIJS:fatlabel$ ./fatlabel -n 192.168.1.1 -f -x
```

Hash calculator FAT (version 1.3), (c) Jul-2016,
xander thuijs CCIE#6775 Cisco Systems Int.

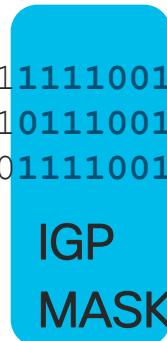
CRC32 = 0x8845F35F

```
XTHUIJS:fatlabel$ ./fatlabel -n 192.168.1.2 -f -x
```

CRC32 = 0x114CA2E5

```
XTHUIJS:fatlabel$ ./fatlabel -n 192.168.1.3 -f -x
```

CRC32 = 0x664B9273



0100001011111111**1111001**0011101
100100101000010101**0111001**1011010
001000101010110000**1111001**1100111

MD5

```
XTHUIJS-M-9180:easy_md5 xthuijs$ ./easy_md5
```

v0.1 MD5 Authenticator calculator
xander thuijs, CCIE6775 Cisco Systems June 2007
Packet contents: (type none<enter> for empty input)
192.168.1.1

Calculated Authenticator:

dc 7c ae 41 1c f5 1d b7 71 7f 6c 38 50 3d 13 4b

Packet contents: (type none<enter> for empty input)
192.168.1.2

Calculated Authenticator:

cd 23 c5 c3 91 16 e0 1d 35 79 29 21 33 51 ac fa

Loadbalancing knobs and what they affect

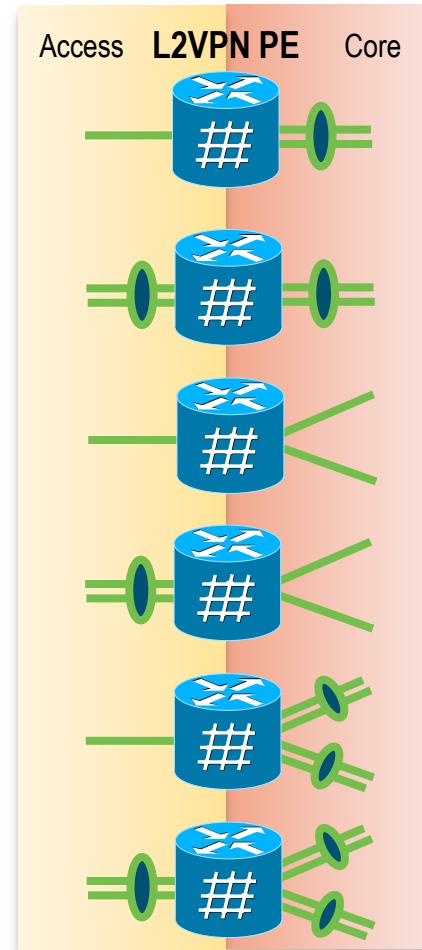
- L2vpn loadbalancing src-dest-ip
 - For L2 bundle interfaces egress out of the AC
 - FAT label computation on ingress from AC towards core
 - Note: upstream loadbalancing out of core interface does not look at fat label (inserted after hash is computed)
- On bundle (sub)interfaces:
 - Loadbalance on srcIP, dest IP, src/dest or fixed hash value (tie vlan to hash result)
Used to be on L2transport only, now also on L3 (XR53).
- GRE (no knob needed anymore)
 - Encap: based on incoming ip
 - Decap: based on inner ip
 - Transit: based on inner payload if incoming is v4 or v6 otherwise based on GRE header
 - So running mpls over GRE will result in LB skews!
 - Tunnelkey possible via **cef loadbalancing fields l4 gtp**

Loadbalancing FAQ

- If packets are fragmented, L4 is omitted from the hash calculation
- Include IPv6 flow label in hash calculation (XR release 6.0 and later)

```
cef load-balancing fields ipv6 flow-label
```

- Show cef exact route or bundle hash BE<x> can be used to feed info and determine actual path/member
 - this is a shadow calculation; *should* yield the same result as HW
- Mixing Bundle members between trident/typhoon/tomahawk is not recommended.



ASR9000 L2VPN Load-Balancing (cont.)

- ASR9000 PE **Imposition** load-balancing behaviors
 - Per-PW based on MPLS VC label (default)
 - Per-Flow based on L2 payload information; i.e. DMAC, SMAC, RID
 - Per-Flow based on L3/L4 payload information; i.e. L3 D_IP / L3 S_IP / L4 D_port / L4 S_port⁽¹⁾, RTR ID
- ASR9000 PE **Disposition** load-balancing behaviors
 - Per-Flow load-balancing based on L2 payload information; i.e. DMAC, SMAC (default)
 - Per-Flow load-balancing based on L3/L4 payload information; i.e. L3 D_IP / L3 S_IP / L4 D_port / L4 S_port

PE Per-Flow load-balance configuration based on L2 payload information

```
!  
12vpn  
load-balancing flow src-dst-mac  
!
```

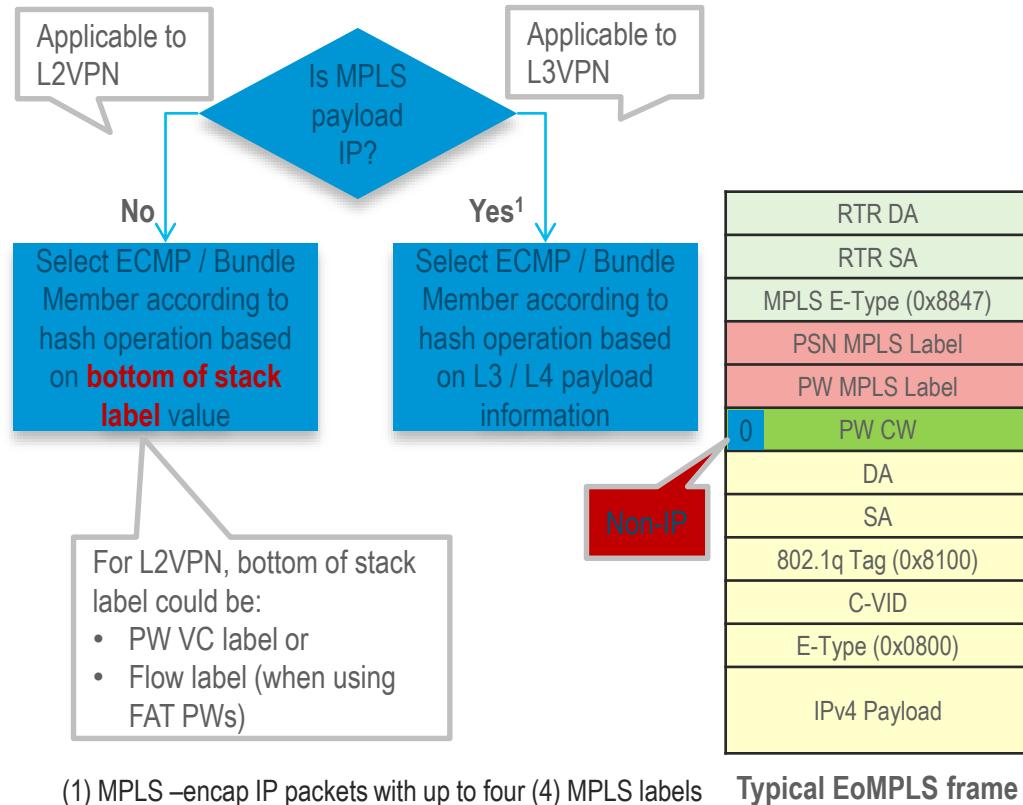
PE Per-Flow load-balance configuration based on L3/L4 payload information

```
!  
12vpn  
load-balancing flow src-dst-ip  
!
```

(1) Typhoon/Tomahawk LCs required for L3&L4 hash keys. Trident LCs only capable of using L3 keys

ASR9000 L2VPN Load-Balancing (cont.)

- ASR9000 as P router load-balancing behaviors
 - Based on L3/L4 payload information for IP MPLS payloads
 - Based on Bottom of Stack label for Non-IP MPLS payloads
 - Based on 4th label is label stack is > 4
- IP MPLS payloads identified based on version field value (right after bottom of stack label)
 - Version = 4 for IPv4
 - Version = 6 for IPv6
 - Anything else treated as Non-IP
- L2VPN (PW) traffic treated as Non-IP
 - PW Control-Word strongly recommended to avoid erroneous behavior on P router when DMAC starts with 4/6



ASR9000 L2VPN Load-Balancing (cont.)

- In the case of VPWS or VPLS, at the ingress PE side, it's possible to change the load-balance upstream to MPLS Core in three different ways:
 1. At the L2VPN sub-configuration mode with “load-balancing flow” command with either src-dst-ip or src-dst-mac [default]
 2. At the pw-class sub-configuration mode with “load-balancing” command where you can choose flow-label or pw-label.
 3. At the bundle interface sub-configuration mode with “bundle load-balancing hash” command with either dst-ip or src-ip.
- It's important to not only understand these commands but also that 1 is weaker than 2 which is weaker than 3.

ASR9000 L2VPN Load-Balancing (cont.)

For example, if you had the configs in all 3 locations

- Because of the priorities, on the egress side of the encapsulation PE (to the MPLS Core), we will do per-dst-ip load-balance.
- If the bundle-specific configuration is removed, we will do per **src-dst-ip load-balancing** across ECMP and LAG.
- **PW-label** will be assigned based on the **src-dst-ip**.

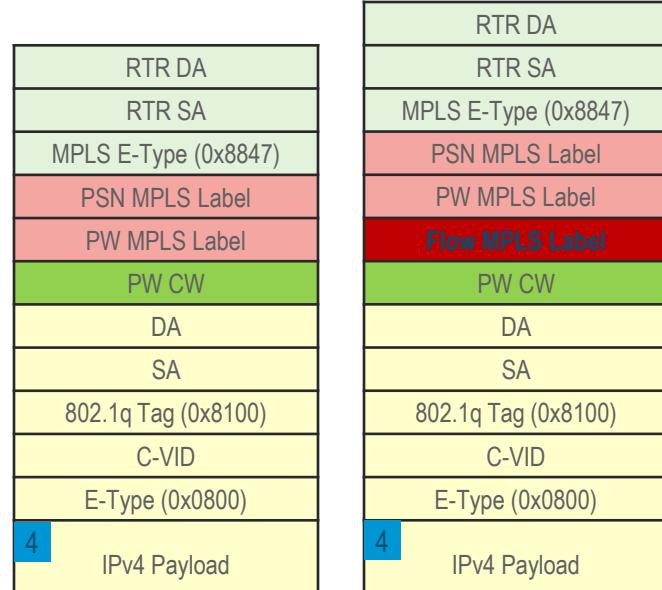
```
interface Bundle-Ether1
  bundle load-balancing hash dst-ip
!
L2vpn
  load-balancing flow src-dst-ip
  pw-class FAT
  encapsulation mpls
  control-word
  transport-mode ethernet
  load-balancing
  pw-label
```

Latest additions to the LoadBalancing family!

- Starting XR 6.5, if an EFP receives an MPLS encapped frame up to 4 labels can be examined to derive IPv4
- Starting XR 6.5, if an EFP receives a PPPoE encapped frame, and if the ppp protocol is IPv4 (0x0021), the L3/L4 info can be derived
 - Note this is Typhoon/Tomahawk specific
 - (Lightspeed and NCS5500 evaluated)
- Starting 6.2.2 Tomahawk can find IPv4/6 up to 8 labels for hash calculation
 - Otherwise uses Labels 3-5
 - (used to be up to 4 labels and inner single label)
- PWHE loadbalancing (more details to follow)
 - The ability to hash on payload as opposed to “per-pw”.

Flow Aware Transport PWs

- Problem: How can LSRs load-balance traffic from flows in a PW across core ECMPs and Bundle interfaces?
- LSRs load-balance traffic based on IP header information (IP payloads) or based on bottom of stack MPLS label (Non-IP payloads)
 - PW traffic handle as Non-IP payload
- RFC6391 defines a mechanism that introduces a Flow label that allows P routers to distribute flows within a PW
 - PEs push / pop Flow label
 - P routers not involved in any signaling / handling / manipulation of Flow label
- NOTE: pseudowire headend (**PWHE**) can be configured with fat label, but it is not supported in the forwarding path. If multiple paths to reach the TLDPE peer can be reached (as defined also by the generic interface list) loadbalancing is done on PW label (if non IP) or based on the ingress hash computation



EoMPLS frame without
Flow Label

EoMPLS frame with
Flow Label

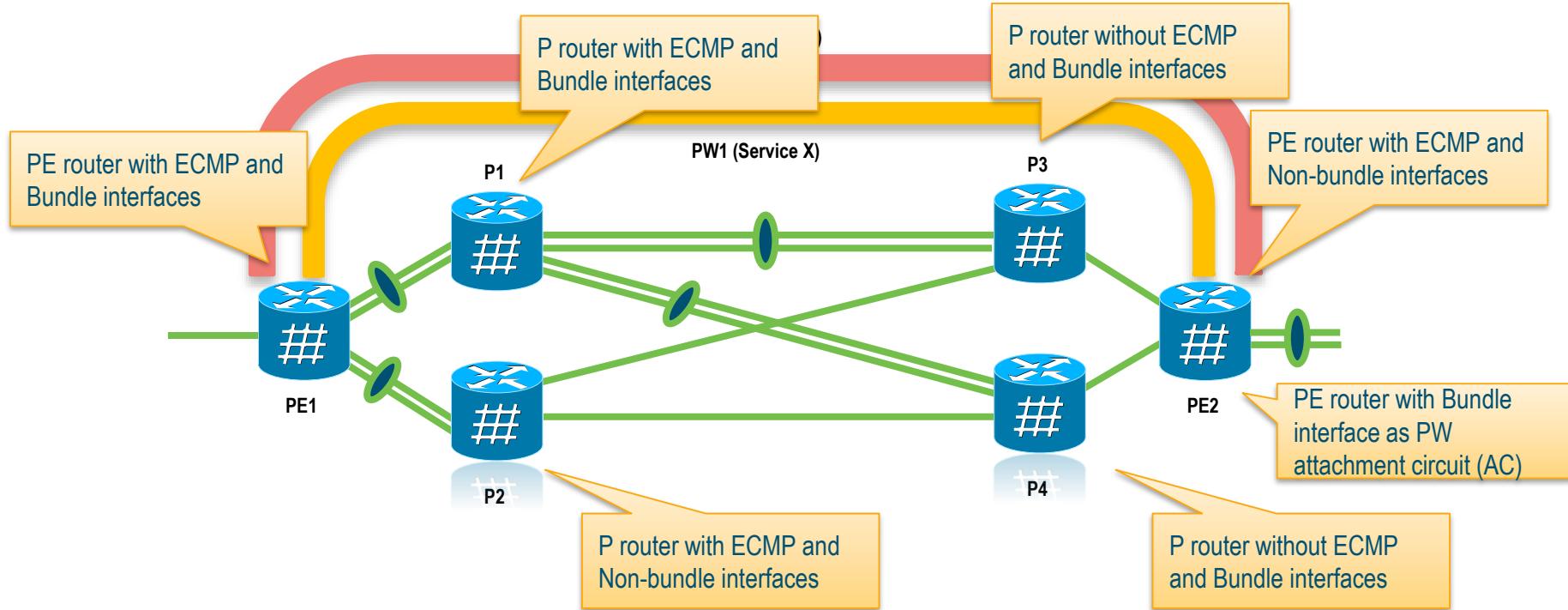
Flow Aware Transport PWs (cont.)

- ASR9000 PE capable of negotiating (via LDP – RFC6391) the handling of PW Flow labels
- ASR9000 also capable of manually configure imposition and disposition behaviors for PW Flow labels
- Flow label value based on L2 or L3/L4 PW payload information
- ASR9000 PE capable of load-balancing regardless of the presence of Flow Label
 - Flow label aimed at assisting P routers

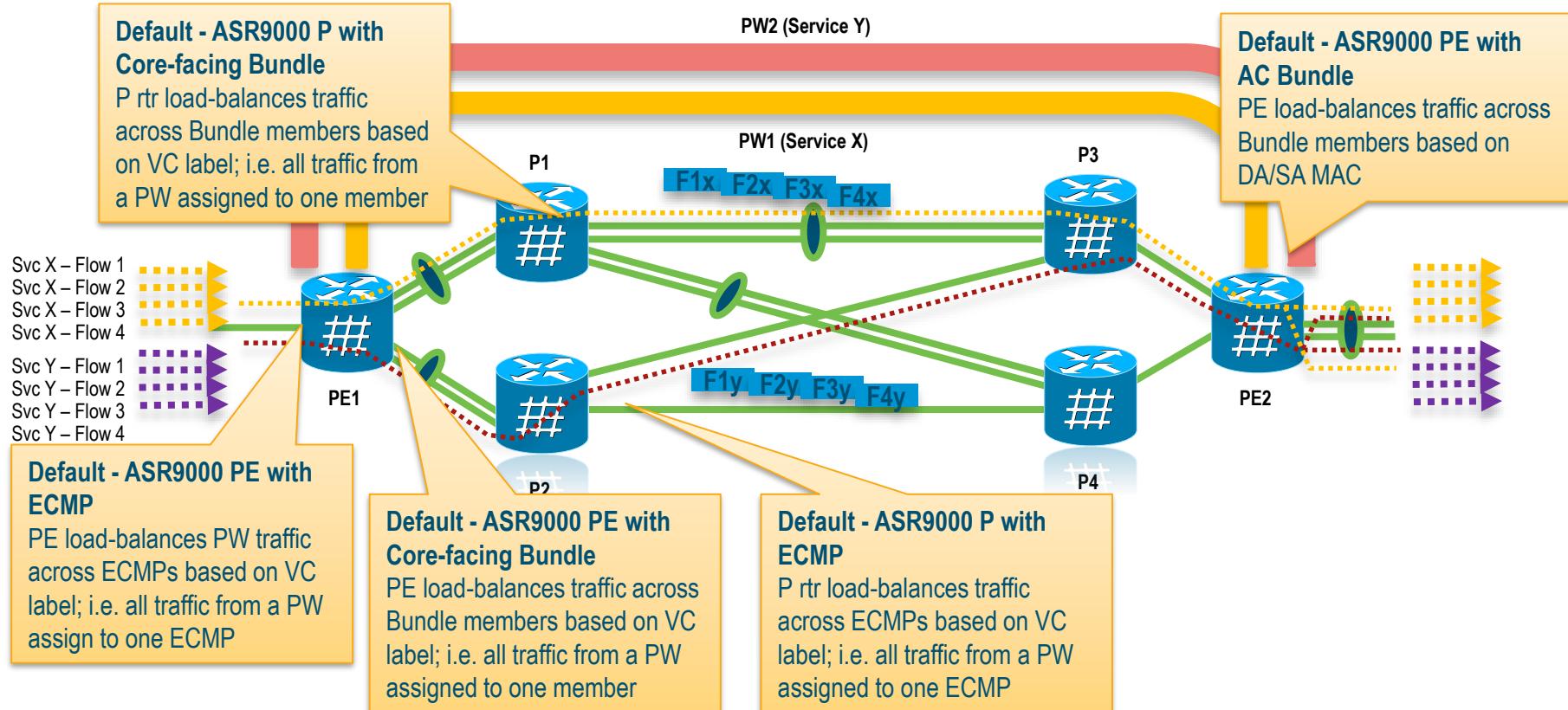
```
!  
12vpn  
pw-class sample-class-1  
encapsulation mpls  
load-balancing flow-label both  
!  
pw-class sample-class-1  
encapsulation mpls  
load-balancing flow-label tx  
!  
pw-class sample-class-1  
encapsulation mpls  
load-balancing flow-label rx
```

```
!  
12vpn  
pw-class sample-class  
encapsulation mpls  
load-balancing flow-label both static  
!
```

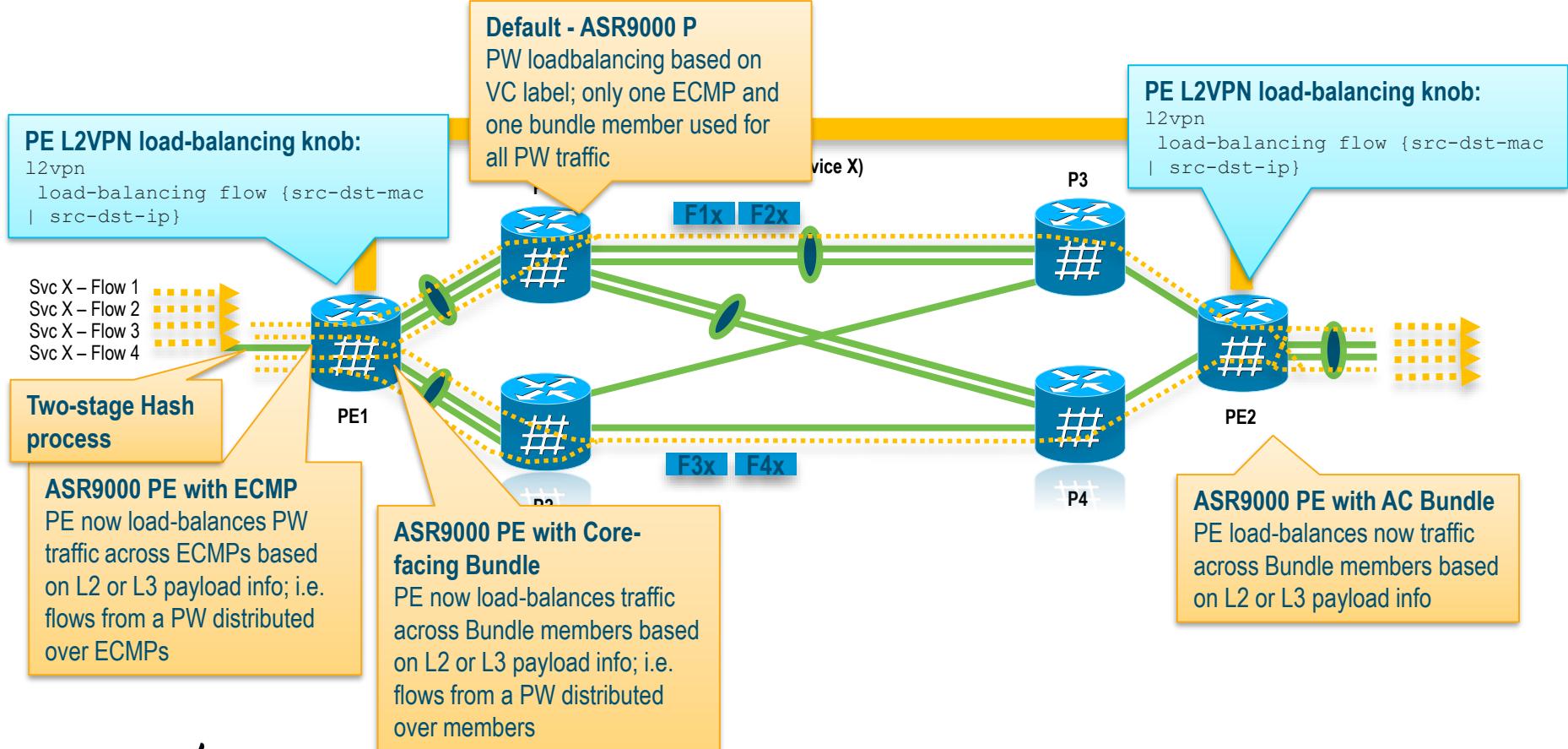
L2VPN Load-balancing (E2E Scenario)



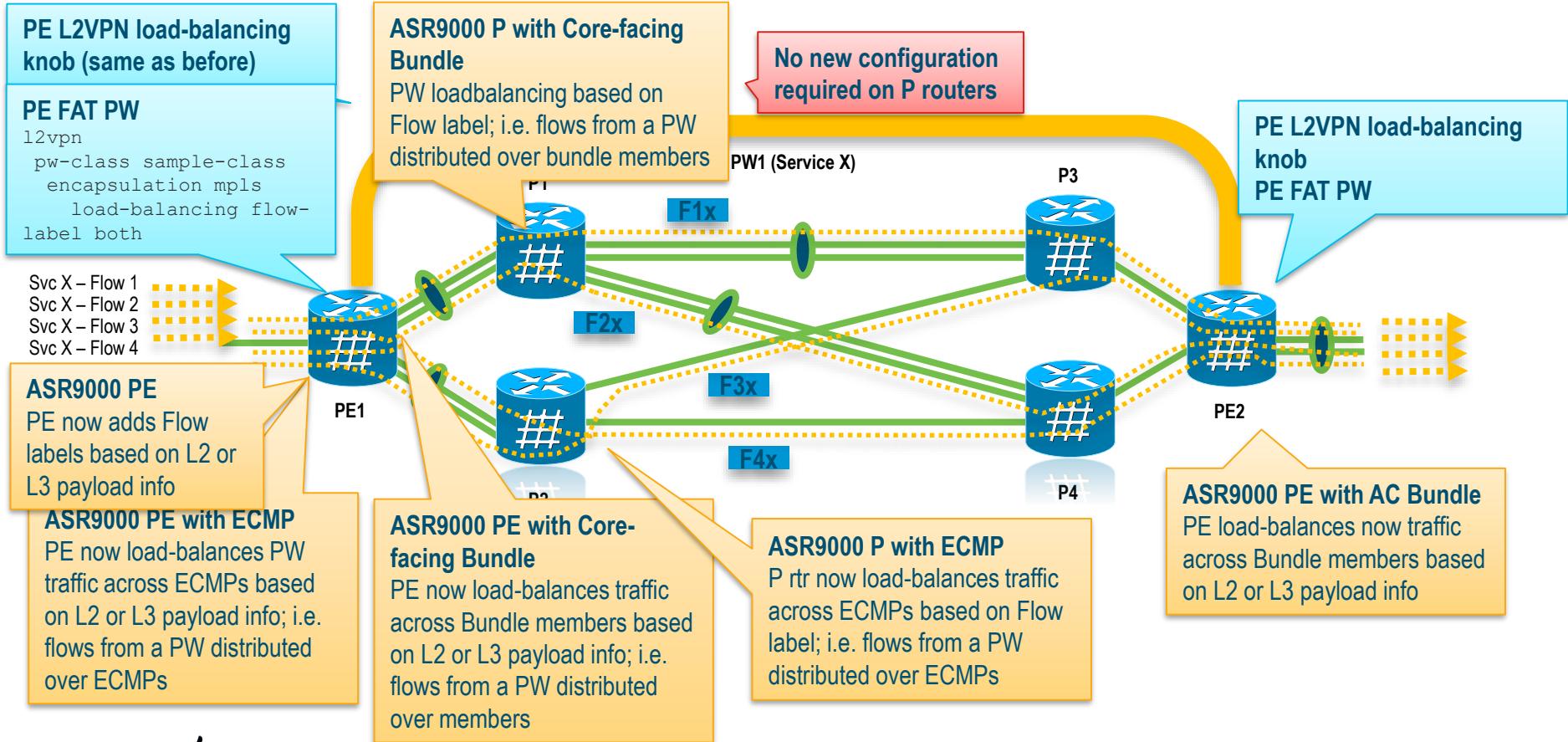
L2VPN Load-balancing (Per-VC LB)



L2VPN Load-balancing (L2/L3 LB)



L2VPN Load-balancing (L2/L3 LB + FAT)



L2VPN LB Summary

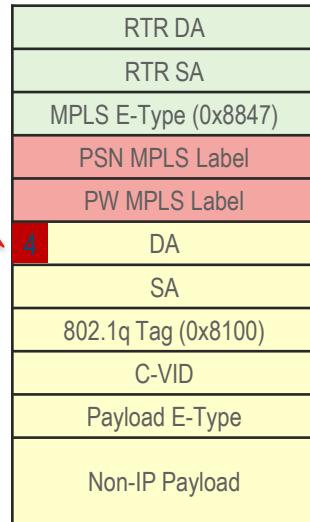
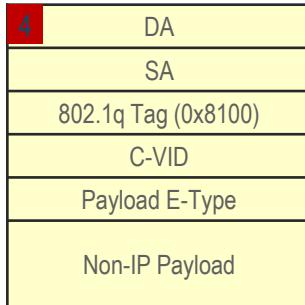
- ASR9000 as L2VPN PE router performs multi-stage hash algorithm to select ECMPs / Bundle members
 - User-configurable hash keys allows for the use of L2 fields or L3/L4 fields in PW payload in order to perform load-balancing at egress imposition
- ASR9000 (as PE) complies with RFC6391 (FAT PW) to POP/PUSH Flow labels and aid load-balancing in the Core
 - PE load-balancing is performed irrespective of Flow PW label presence
 - FAT PW allows for load-balancing of PW traffic in the Core WITHOUT requiring any HW/SW upgrades in the LSR
 - Cisco has prepared a draft to address current gap of FAT PW for BGP-signaled PWs
- ASR9000 as L2VPN P router performs multi-stage hash algorithm to select ECMPs / Bundle members
 - Always use bottom of stack MPLS label for hashing
 - Bottom of stack label could be PW VC label or Flow (FAT) label for L2VPN

Significance of PW Control-Word

Problem:

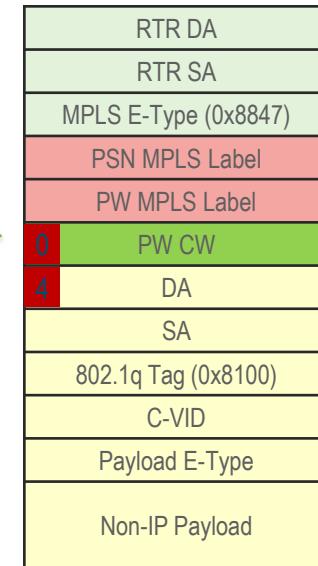
DANGER for LSR

LSR will confuse payload as IPv4
(or IPv6) and attempt to load-
balance based off incorrect fields



Solution:

Add PW Control Word in front of PW payload. This guarantees that a zero will always be present and thus no risk of confusion for LSR

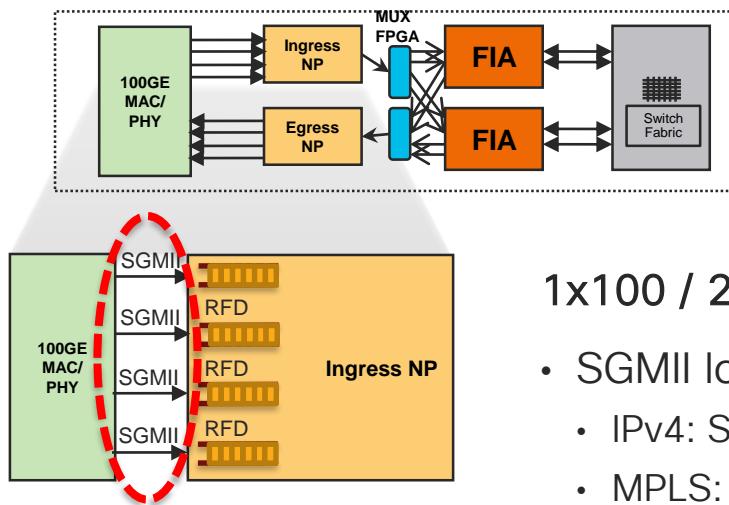


PHY Hash (100G Typhoon)

Dynamic
(Traffic based)

PHY hash

Typhoon 100G PHY
SGMII link selection
(ingress only)



1x100 / 2x100 Typhoon LC before NP:

- SGMII load-balancing (ingress only):
 - IPv4: SrcIP+DstIP
 - MPLS: innermost or 4th MPLS label
- RFD Buffer allocation per SGMII
 - When 525 RFD buffers are in use in an SGMII pool, Early Frame Discard (EFD) kicks in

Satellite ICL Bundle Load-Balancing

Static
(port, VC, etc.
based)

Port hash

Satellite ICL Bundle

ICL bundle member
selection

hash_value = (satellite_port_number + 2) % number_of_ICL_members

Eg: on a bundle ICL with 4 members, hash value of satellite port 13 is:

$$X = (13 + 2) \% 4 = 15 \% 4 = 3$$

# of ICL members	Satellite Port	Hash
4	1	3
4	2	0
4	3	1
4	4	2

asr9k



Satellite

1 2 3 ... 40

ASR9000 Unicast Fabric Load Balancing

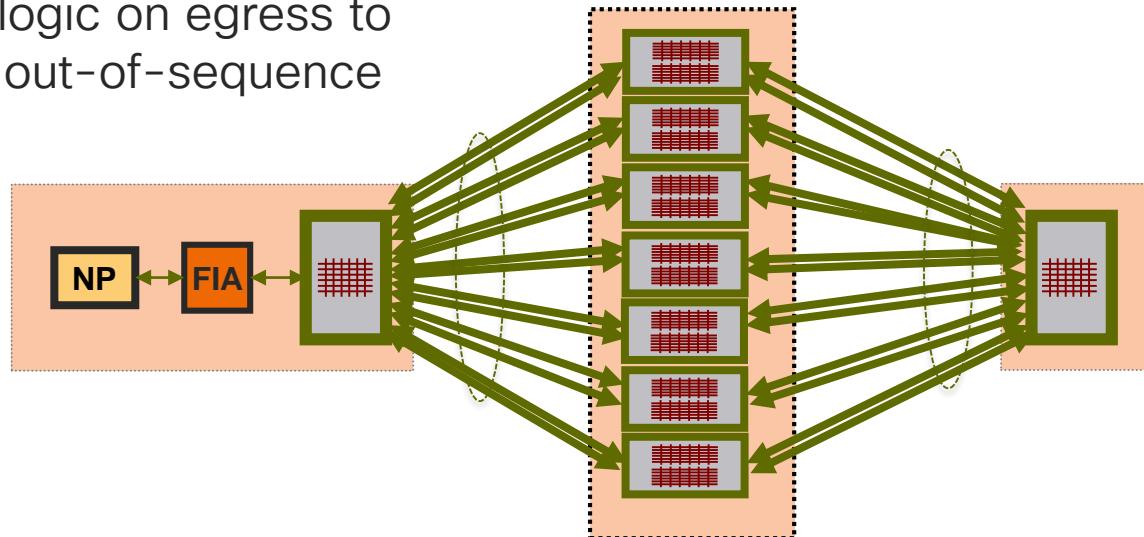
Static
(port, VC, etc.
based)

Round robin

Ingress FIA

Fabric link selection

- Round-robin across all fabric links; optimal fabric BW utilisation
- Special logic on egress to prevent out-of-sequence



ASR9000 Multicast Fabric Load Balancing

Dynamic
(Traffic based)

NP µcode hash

Ingress NP

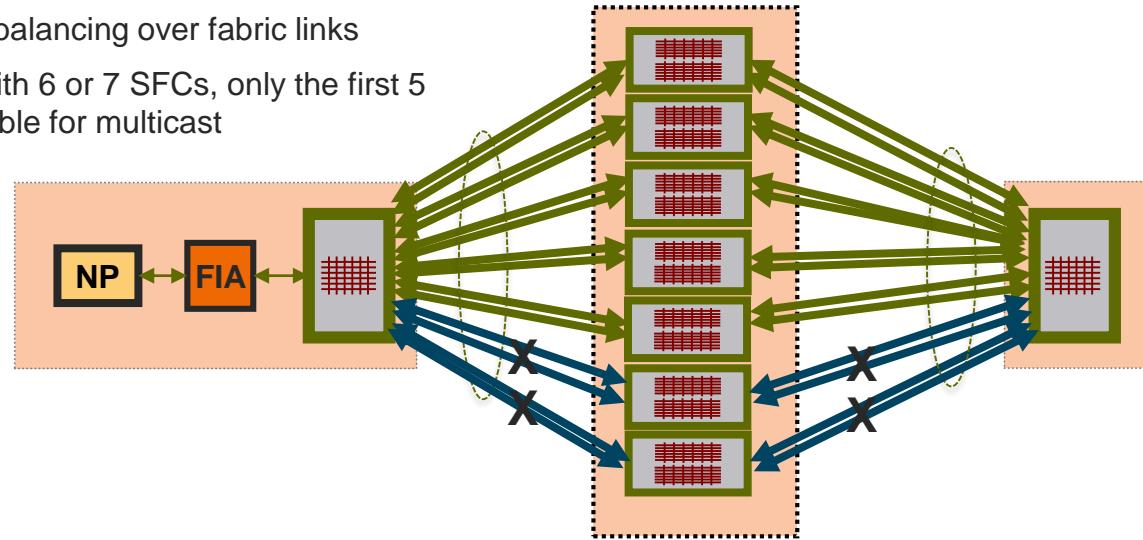
Fabric link selection

Hash input:

- For (S,G): Source IP, Group IP, next-hop of RPF
- For (*,G): RP address, Group IP, next-hop of RPF

Algorithm:

- Per-flow load balancing over fabric links
- On asr9922 with 6 or 7 SFCs, only the first 5 SFCs are eligible for multicast



Important ASR9k MFIB Data-Structures

- FGID = Fabric Group ID
 1. FGID Index points to (slotmask, fabric-channel-mask)
 2. Slotmask, fabric-channel-mask = simple bitmap
- MGID = Multicast Group ID (S,G) or (*,G)
- 4-bit RBH
 1. Used for multicast load-balancing chip-to-chip hashing
 2. Computed by ingress NP ucode using these packet fields:
 3. IP-SA, IP-DA, Src Port, Dst Port, Router ID
- FPOE = FGID + 4-bit RBH

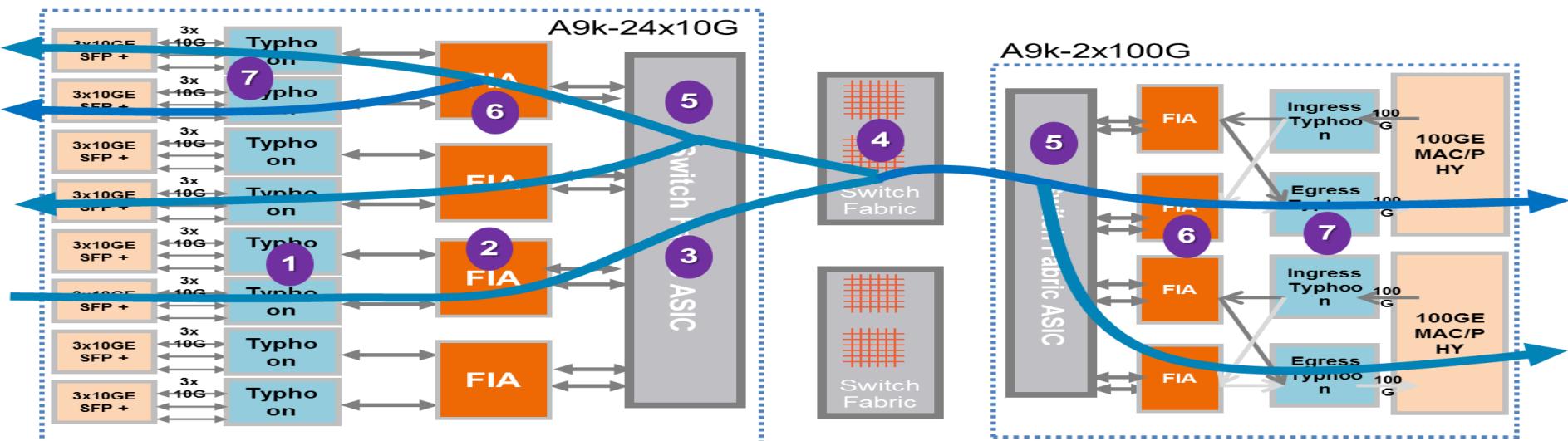
(S,G)	MGID	FGID
10.0.0.1, 230.0.0.1	12345	0x82
10.1.1.1, 230.12.1.1	3456	0x209
172.16.1.3, 229.1.1.3	23451	0x100

Multicast Replication Model Overview

- Ingress NPU:
 - MFIB (S,G) route lookup yields {FGID, MGID, Olist, 4-bit RBH} data-structures
 - Ingress NPU adds FGID, MGID, 4-bit RBH in fabric header to FIA
- Ingress FIA:
 - Load-balance multicast traffic from FIA to LC Fabric
- Ingress LC Fabric:
 - Reads FPOE bits in the fabric header AND reads 3-bits of derived RBH
 - It will load-balance MGID towards any of the fabric channels
 - Now it send traffic to central fabric over 1 of the fabric channels per MGID
- Fabric Replication to Egress LC Fabric:
 - Receives 1 copy from ingress LC
 - Reads fabric header FGID slotmask value to lookup the FPOE table to identify which fabric channel output ports to replicate to
 - Now it replicates 1 copy to egress LCs with multicast receivers
- *For a given multicast flow, the choice of selecting a path from a list of paths between source and destination NPs is done using key inserted in fabric header. This key is called RBH or Result Bundle Hash. NP calculates RBH value. This means that multiple RBH values have to be calculated and used when more than one flows exists between source and destination NP.*
- Egress LC Fabric Replication to FIA:
 - Egress LC fabric is connected to all the FIAs (ie. upto 6 FIAs in A9k-36x10G) card
 - All MGIDs (ie. mroute) are mapped into 4k FPOE table entries in LC fabric
 - Looks up FPOE index and replicate the packets mapped to egress FIAs with MGID receiver
- Egress FIA Replication to NPU
 - Egress FIA has 256k MGIDs (ie. mroutes), 1 MGID is allocated per mroute
 - Each MGID in the FIA is mapped to its local NPUs
 - Performs a 19-bit MGID lookup of incoming mcast packet from LC fabric
 - Replicates 1 copy to each Typhoon NPU with mroute receivers
- Egress NPU Multicast OIF Replication
 - Egress NPU performs L2/L3/MPLS multicast OIF replication (2nd stage lookup)
 - MGID lookup yields OIF count (ie. replication interface count)
 - When OIF count == 1, then NPU replicate all L2/L3/MPLS multicast traffic in 1st pass
 - When OIF count > 1, then NPU replicate all L2/L3/MPLS multicast traffic in 2nd pass
 - (S,G), (*,G)

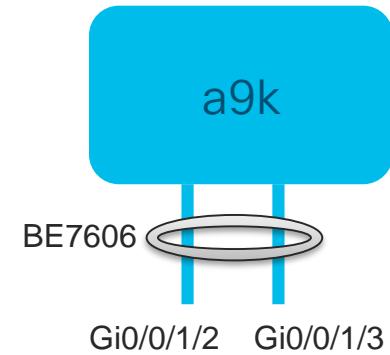
Multicast Replication Model Overview

- Multicast Replication in ASR9k is like an SSM tree
- 2-stage replication model:
 1. Fabric replication
 2. Egress NP OIF replication



Multicast Bundle Load-Balancing

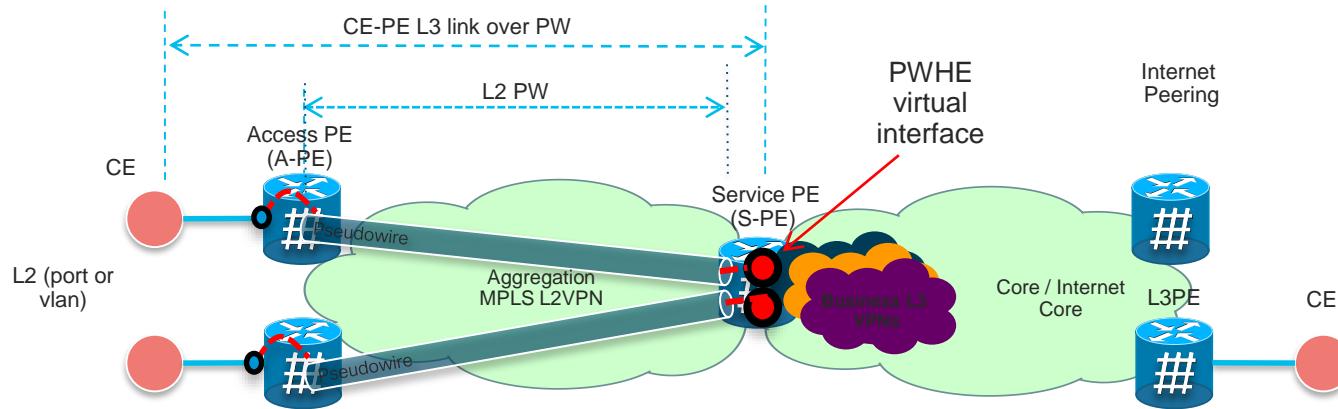
```
RP/0/RSP0/CPU0:a9k#sh mrib route 172.16.255.3 230.9.0.8 detail  
(172.16.255.3, 230.9.0.8) Ver: 0x3f71 RPF nbr: 0.0.0.0 Flags: C RPF,  
PD: Slotmask: 0x0  
MGID: 16905  
Up: 00:04:00  
Outgoing Interface List  
Bundle-Ether7606.1 (0/0/CPU0) Flags: LI, Up: 00:04:00
```



```
RP/0/RSP0/CPU0:a9k#sh mrib platform interface bundle-ether 7606.1 detail  
< ..output omitted.. >  
-----  
Route OLE on Bundle-Ether7606.1 (0x620)  
Route: 0xe0000000:(172.16.255.3, 230.9.0.8)/32  
UL Interface: GigabitEthernet0/0/1/2 (0x4000140)  
Bundle Member: GigabitEthernet0/0/1/2 (0x4000140)  
Raw hash: 0x1977fd33  
Intrf Handle: 0x5000147b  
Entry Handle: 0x50001493  
-----
```

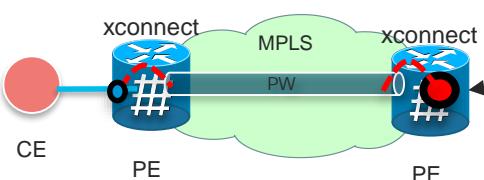
PWHE Load Balancing

Pseudo-Wire Head-End



- Unified MPLS end-to-end transport architecture
- Flexible service edge placement with virtual PWHE interface
 - L2 and L3 interface/sub-interface
 - Feature parity as regular L3 interface: QoS, ACL, Netflow, BFD, etc
 - CE-PE routing is over MPLS transport network. It doesn't need direct L3 link any more
- CE-PE virtual link is protected by the MPLS transport network

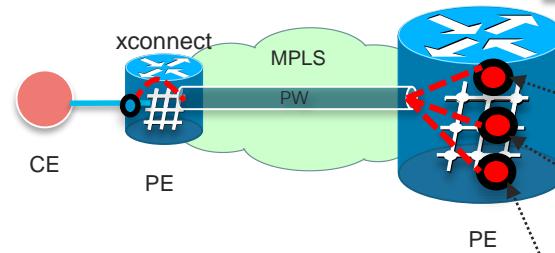
Pseudo-Wire Head-End Configuration Examples



PWHE L3/L2 Main Interface Example

```
interface pw-ether 200
vrf vrf0001
ipv4 address 11.0.0.1/24
ipv6 address 2001:dal::1/64
attach generic-interface-list pwhe_gi_2
!
generic-interface-list pwhe_gi_2
interface Bundle-Ether100
interface TenGigE0/5/0/12
```

```
l2vpn
xconnect group pwhe
p2p pwhe-red
interface pw-ether 100
neighbor 100.100.100.100 pw-id 1
```



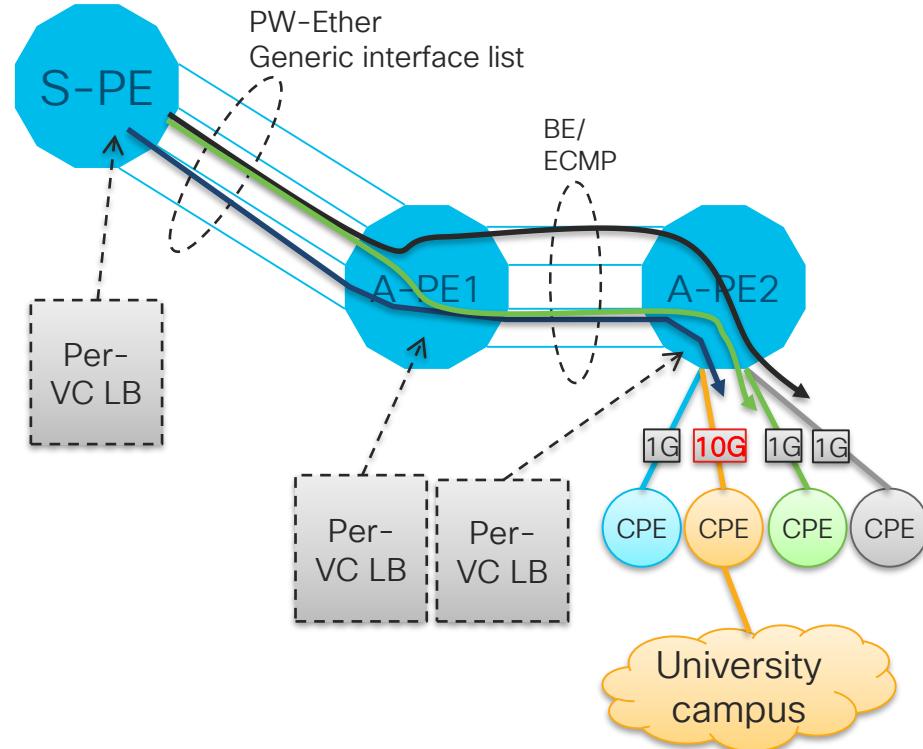
PWHE L3/L2 Sub-interface Example

```
interface pw-ether 100.100
encap dot1q 100
vrf vpn-red
ipv4 address 10.1.1.2/24
!
interface pw-ether 100.200 12transport
encap dot1q 200
!
interface pw-ether 100.300 12transport
encap dot1q 300
```

```
l2vpn
xconnect group cisco
p2p service2
interface PW-Ether100.200
neighbor ipv4 1.1.1.1 pw-id 22
bridge group group1
bridge-domain domain2
interface PW-Ether100.300
vfi cisco
neighbor 192.0.0.1 pw-id 100
neighbor 192.0.0.2 pw-id 100
```

PWHE Load-Balancing

- L2VPN load-balancing
 - Only Per-VC
- Flow Label
 - Supported in 6.3.3, 6.5.x, 6.6.x, etc.
- BGP MP
 - No support



PWHE Troubleshooting

```
show generic-interface-list [name <name>]
show im database interface pw-ether <n> verbose
show interface pw-ether <n> detail
show proc pwhe_ma

show l2vpn ea pwhe summary location <location>
show l2vpn pwhe interface pw-ether <n> detail
show l2vpn generic-interface-list private
show l2vpn ma pwhe interface pw-ether <n> private
show l2vpn ma pwhe peer private
show l2vpn ma pwhe trace

show mpls lsd forwarding pw-list
```

Complete your online session survey



- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on cisco.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at cisco.com.

Continue your education



Demos in the
Cisco Showcase



Walk-In Labs



Meet the Engineer
1:1 meetings



Related sessions



Thank you





i i i i i i i i

You make **possible**