The bridge to possible

Cisco SD-WAN - Hidden Complexity Revealed

How Cisco TAC Addresses Really Tricky Problems?

Denis Kodentsev, SD-WAN TAC Technical Leader, CCIE

BRKTRS-3050

cisco Live!

#CiscoLive

Who’s your speaker?

- Denis Kodentsev
  - SP/EN networking since 2000
  - with Cisco since 2007
  - CCIE since 2013
  - Designing Cisco SD-WAN networks since 2017
  - Tech Lead for SD-WAN TAC in Krakow, Poland

Cisco Webex App

https://ciscolive.ciscoevents.com/ciscolivebot/#BRKTRS-3050

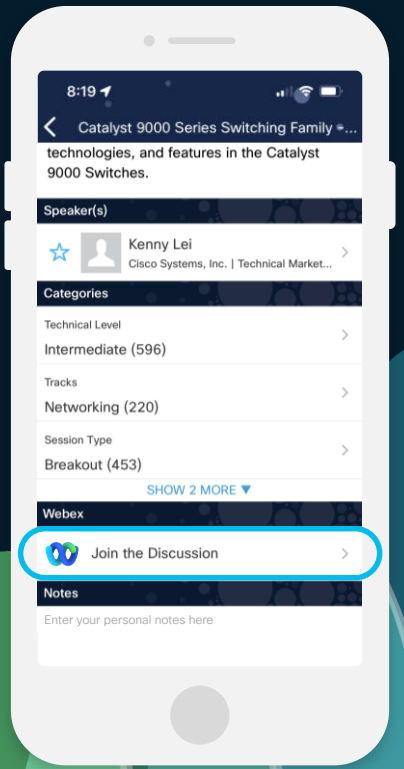
Questions?

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

How

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

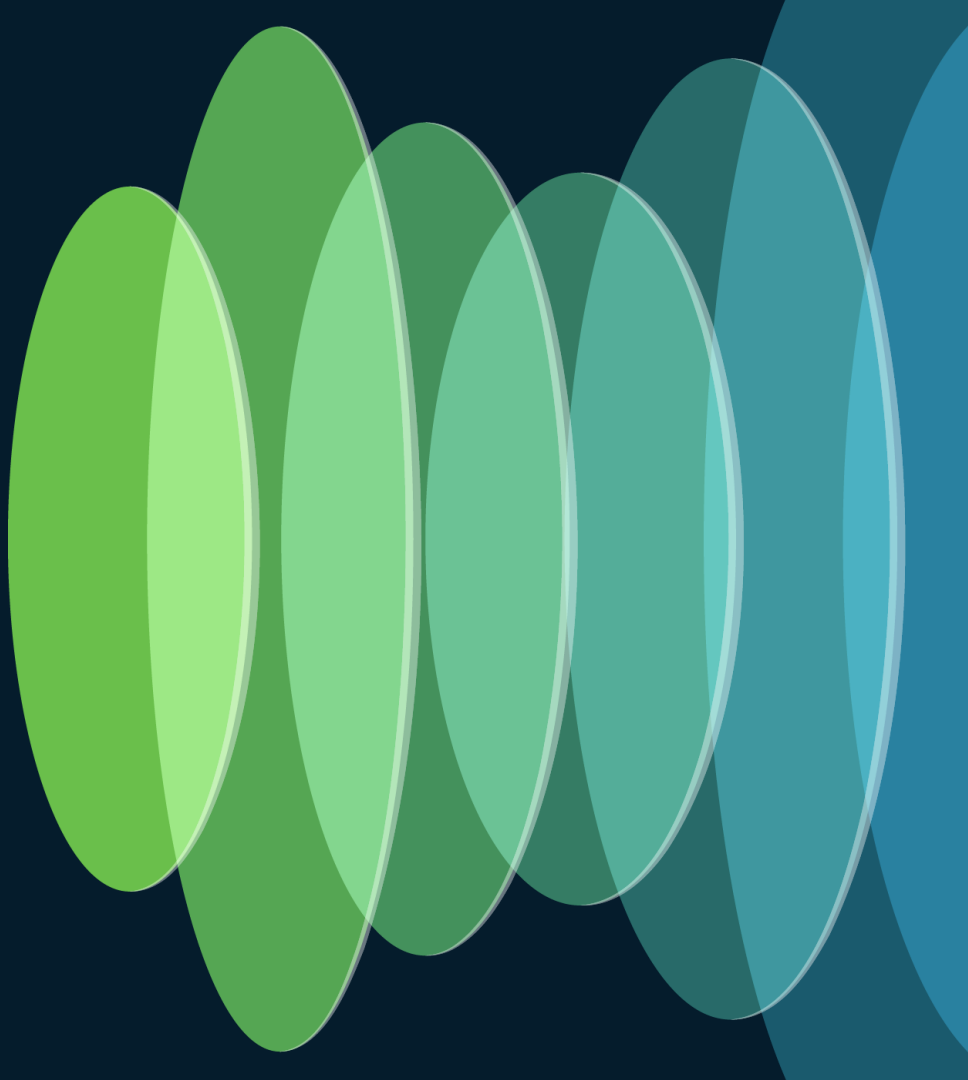
Webex spaces will be moderated by the speaker until June 7, 2024.



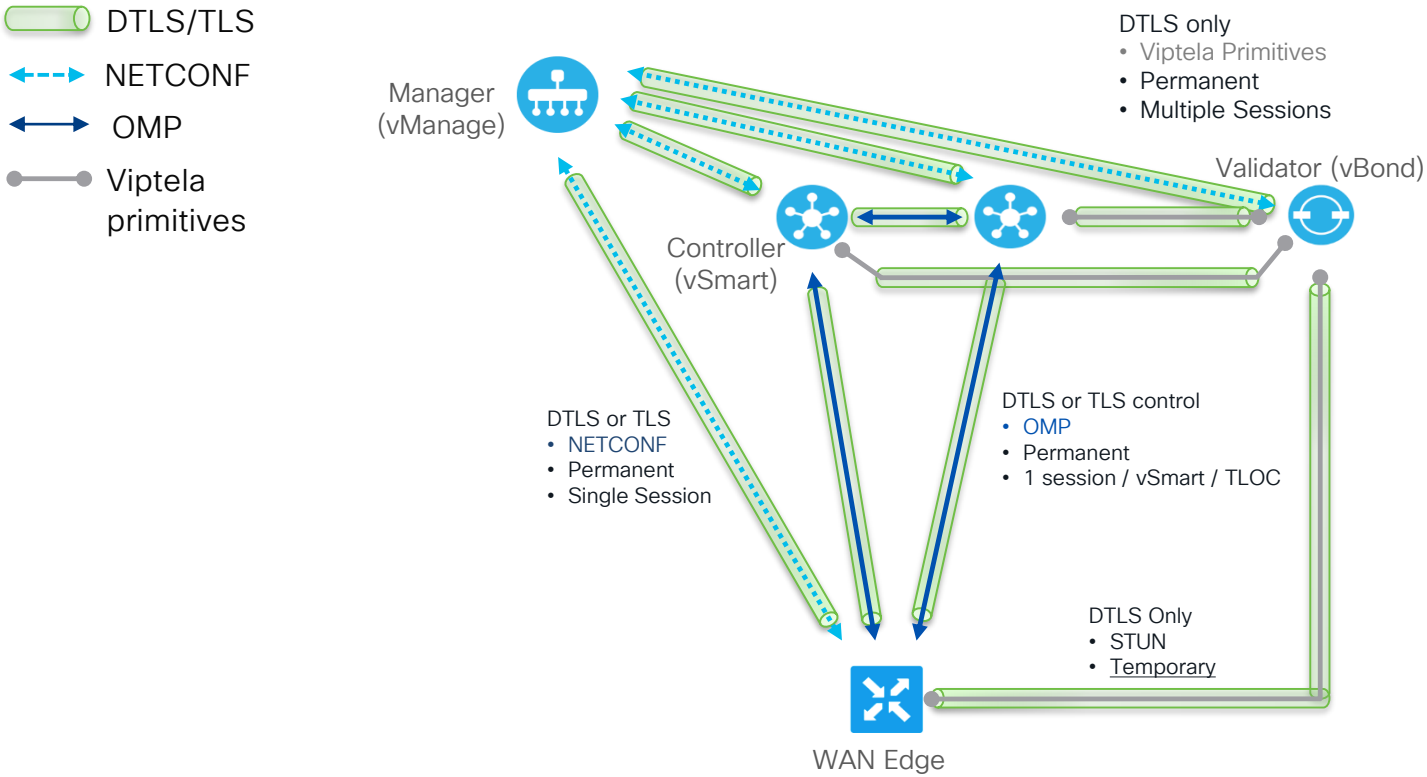
Agenda

- Best tools we use in TAC and their scope
- (Improve) your understanding of control-plane logs
- (Improve) your understanding of data-plane logs and outputs
- Personal know-how to share

Setting the stage

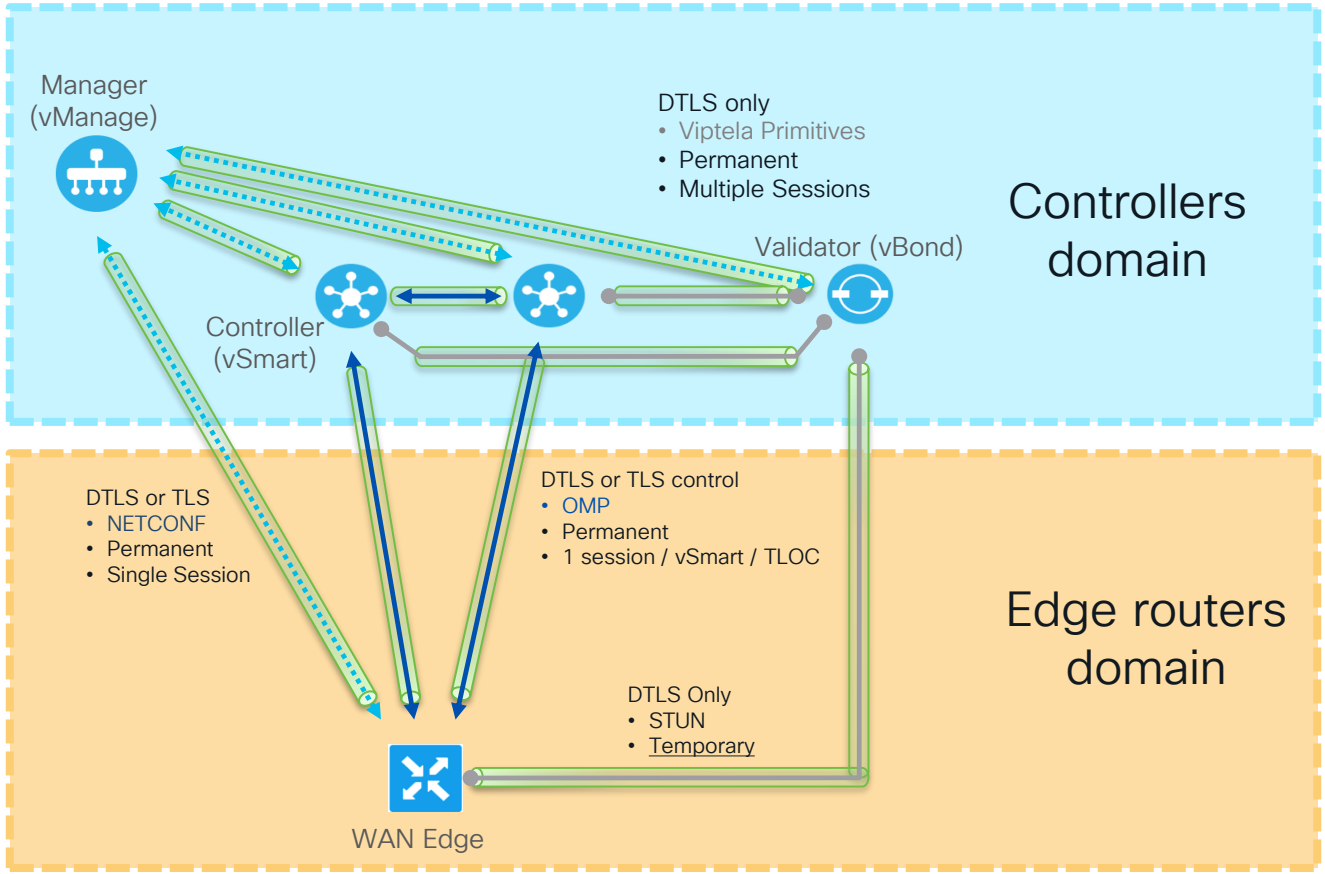


Let’s define the scope for the session



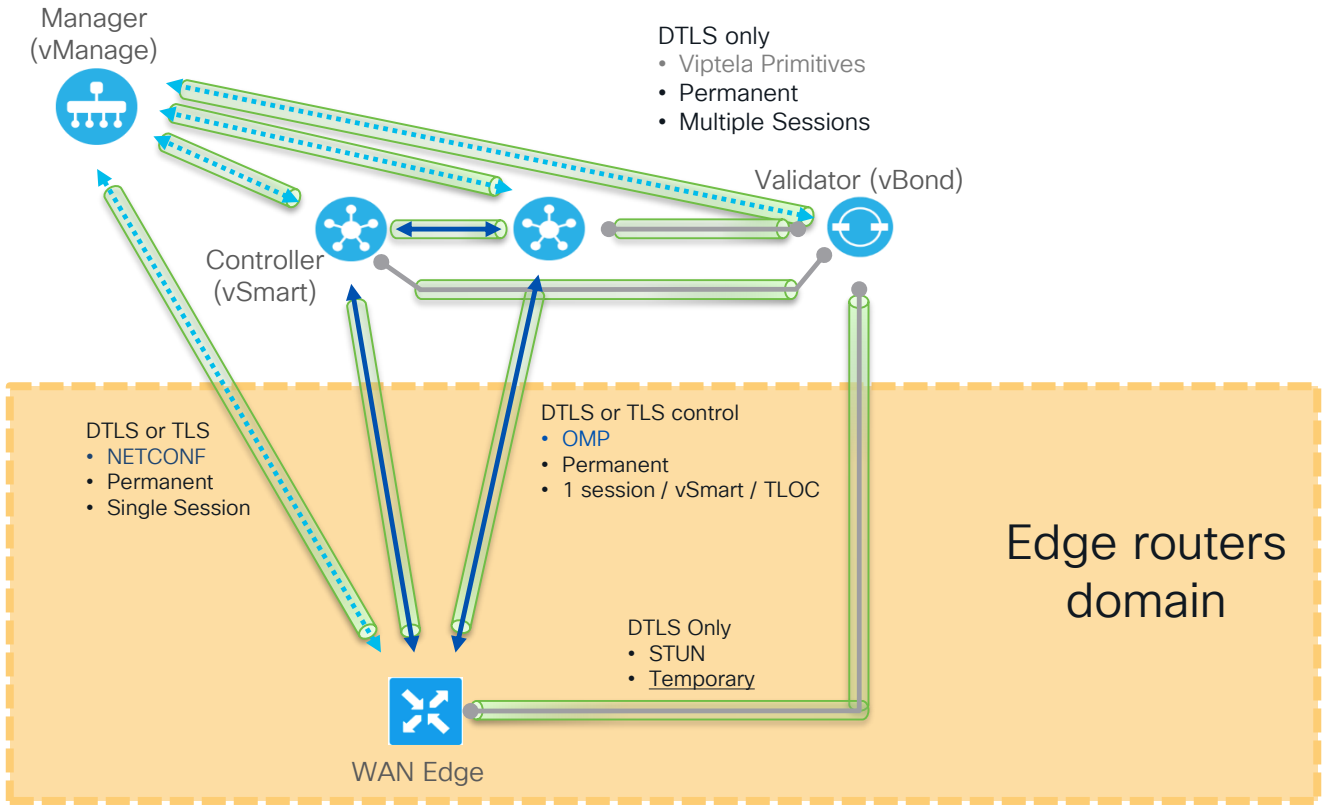
Two major domains to consider

DTLS/TLS
NETCONF
OMP
Viptela primitives



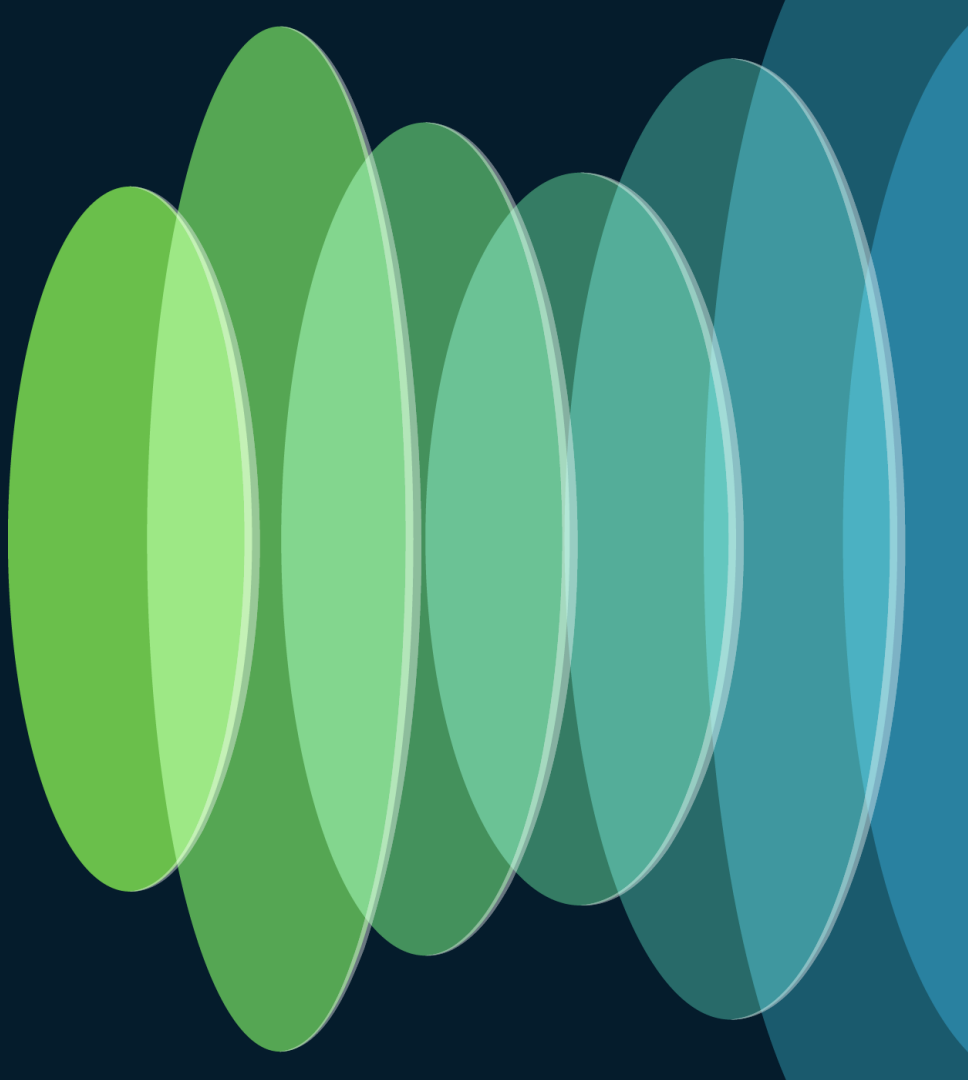
Let's focus on WAN Edges
both control-plane and data-plane

DTLS/TLS
NETCONF
OMP
Viptela primitives



It’s all about knowing your tools...

and when to use what?



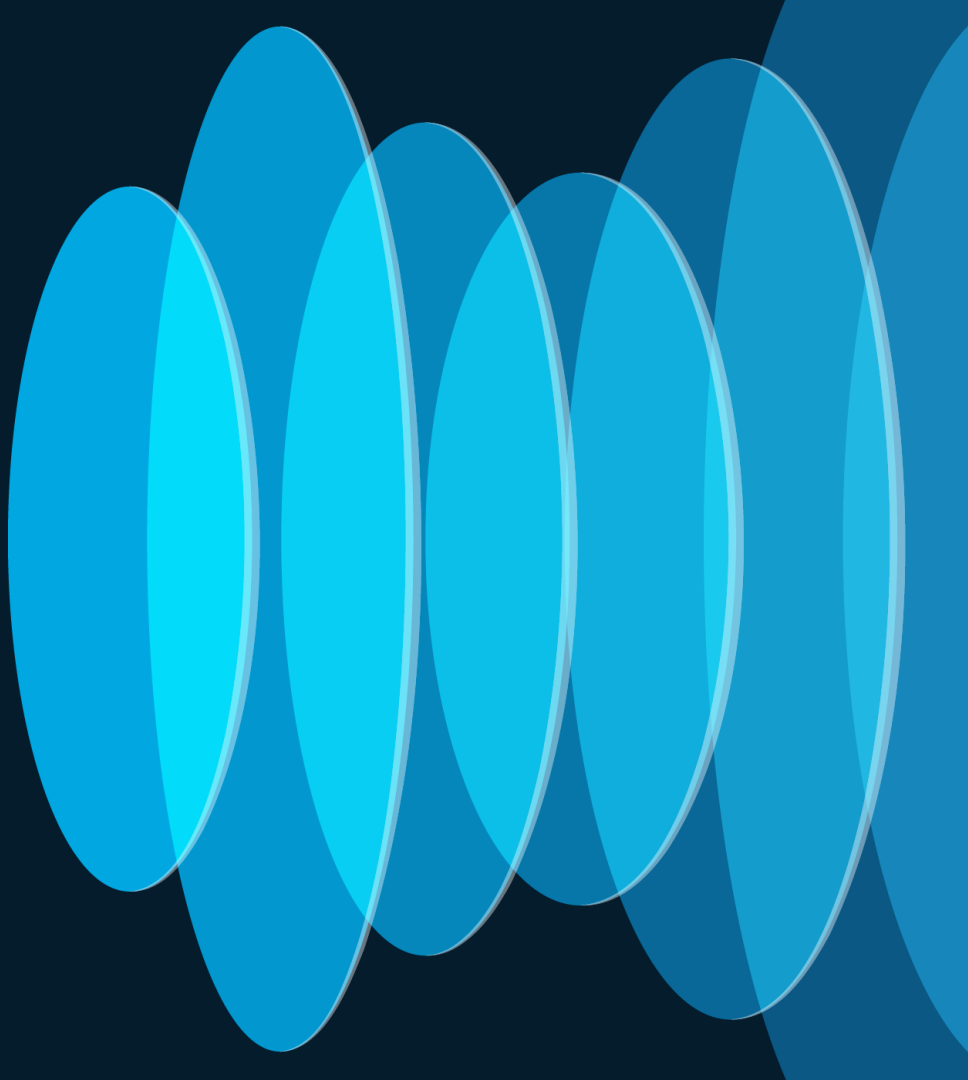
You followed troubleshooting best-practices and … still no luck?

- Cisco SD-WAN Troubleshooting TechNotes

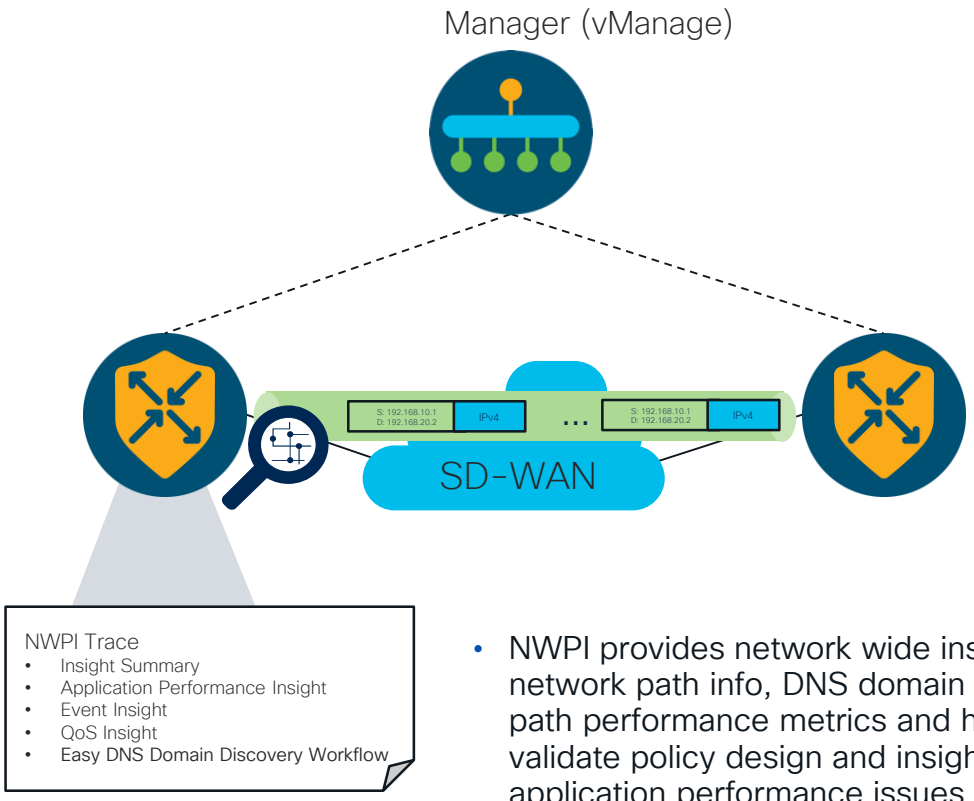
Cisco Live on-demand library:

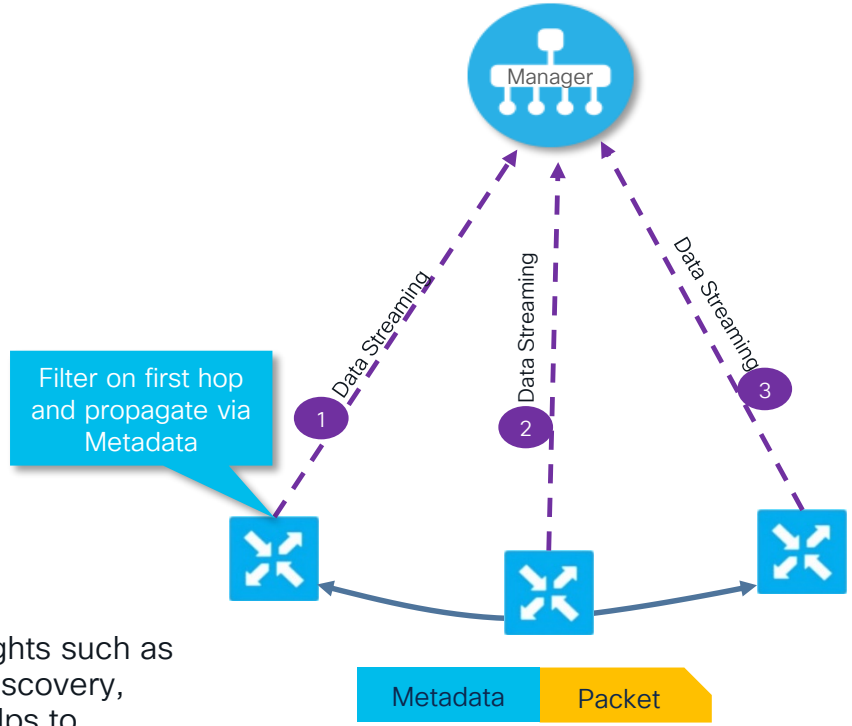
- Advanced SD-WAN Routing Troubleshooting
- SD-WAN - Advanced Troubleshooting with the power of NWPI and other features
- Advanced SD-WAN Policies Troubleshooting
- Automation and In-Depth Troubleshooting of Cat8k, ASR1k, ISR and SD-WAN Edge

Network Wide Path Insight (NWPI)



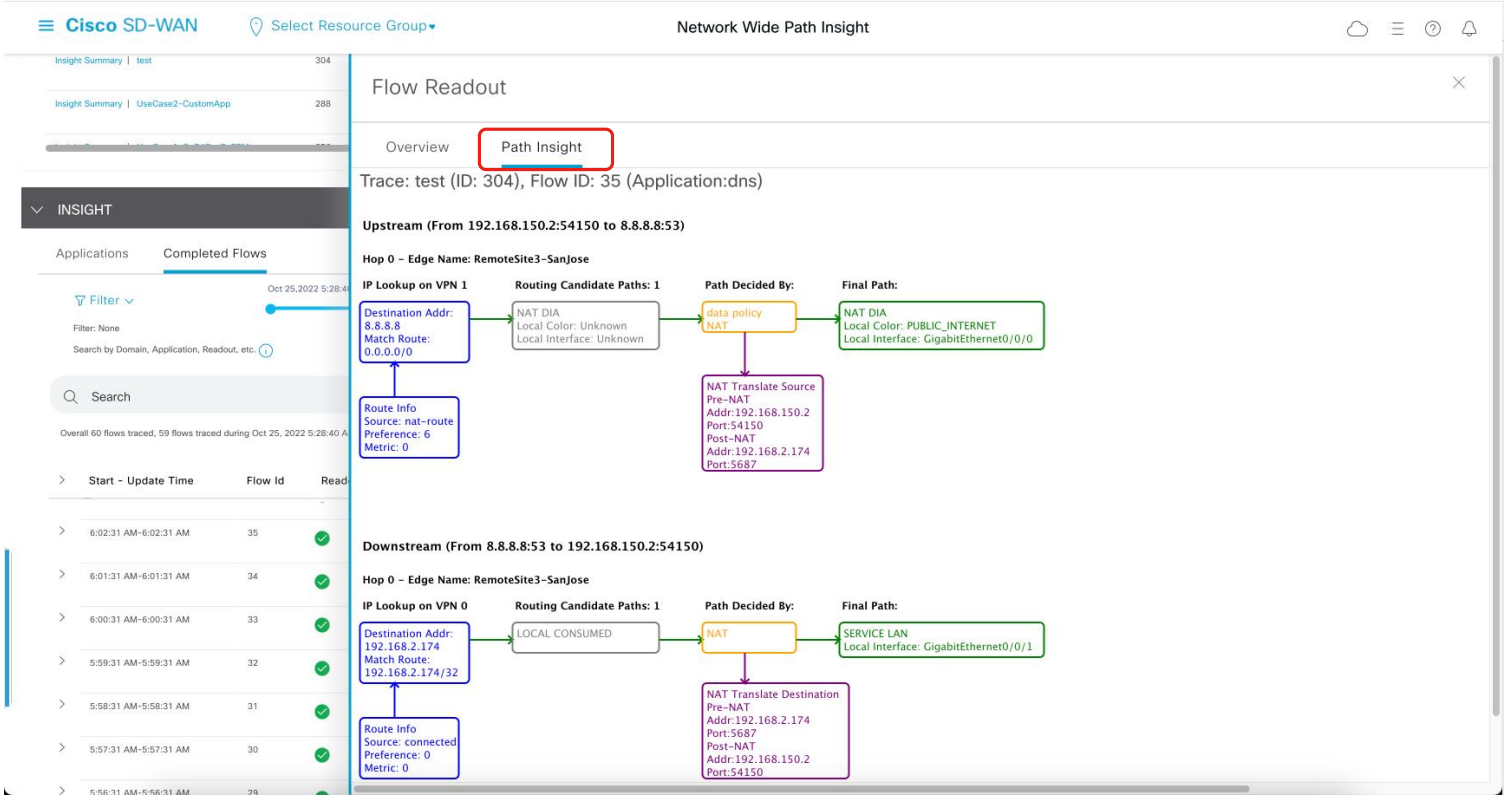
Network Wide Path Insight (NWPI)



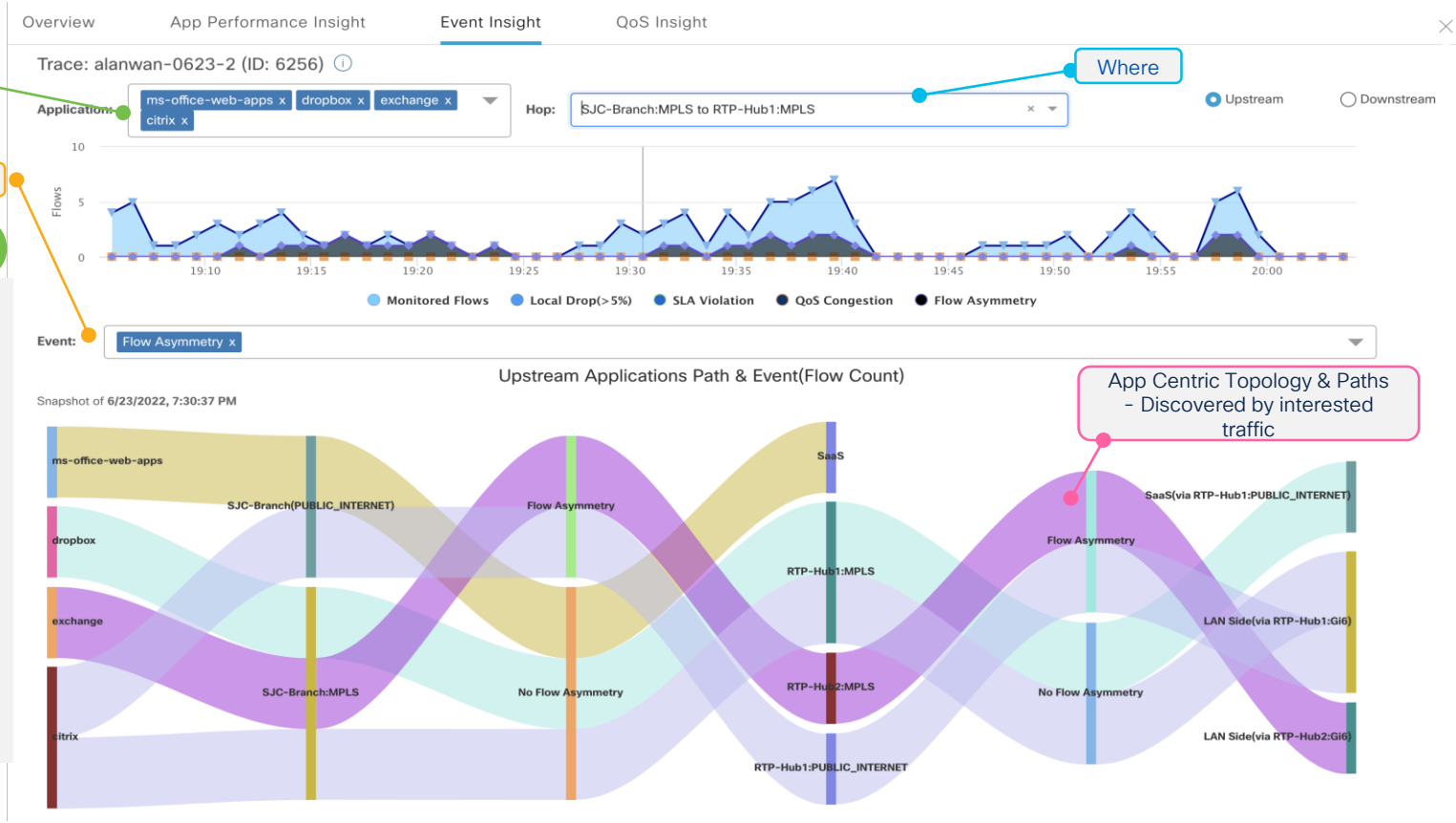


- NWPI provides network wide insights such as network path info, DNS domain discovery, path performance metrics and helps to validate policy design and insights for various application performance issues

NWPI: Path Insight



Insight Summary – Event Insight



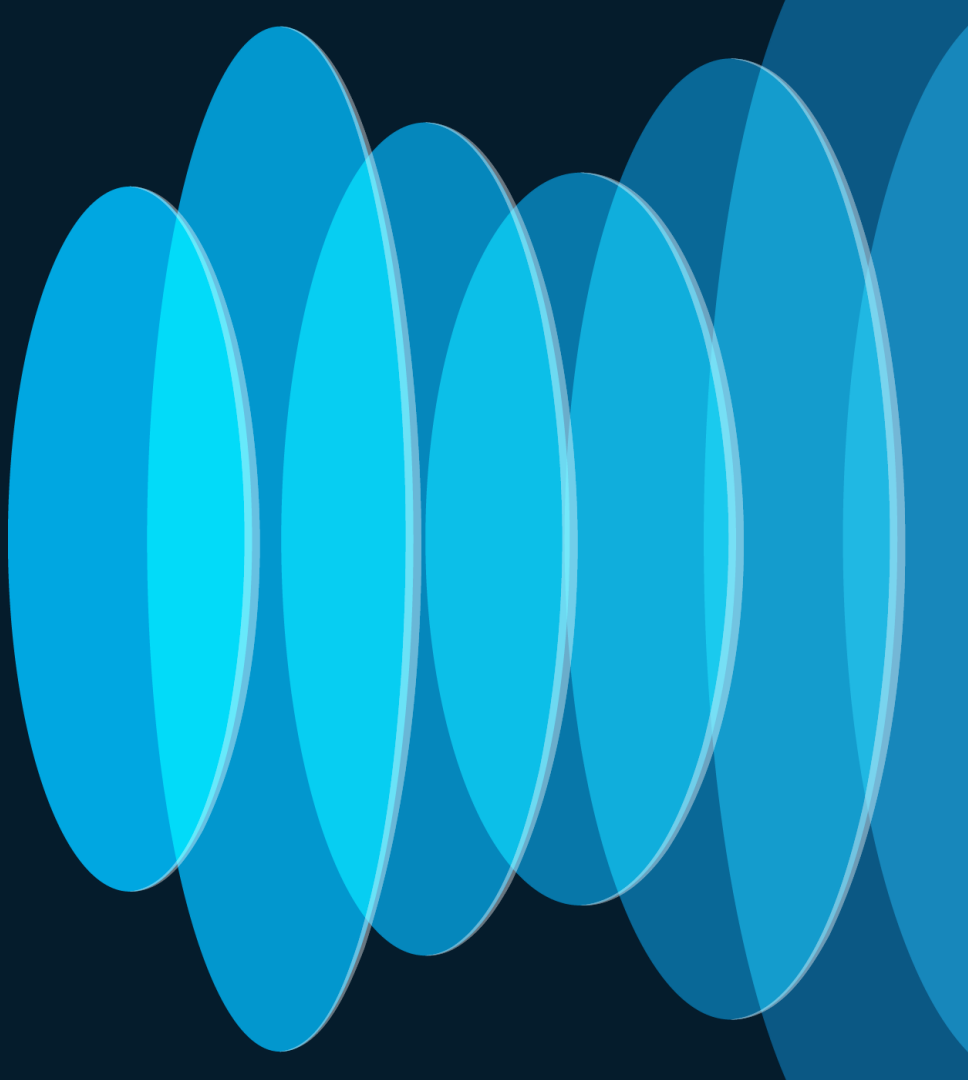
Readout

Office: SJC-Branch local breakout to SaaS Cloud via INET(DIA).

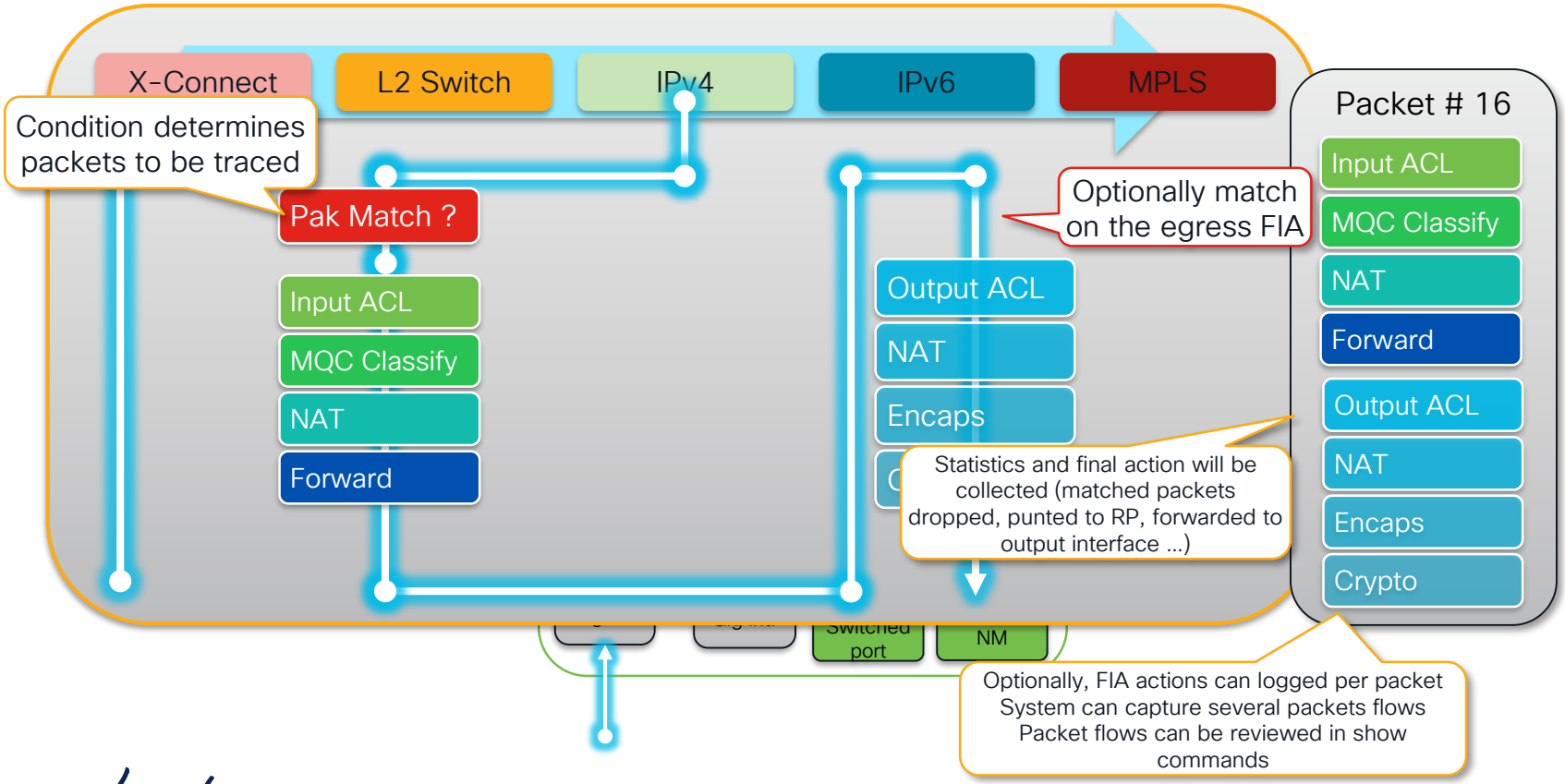
Dropbox: Backhaul from SJC-branch to RTP-Hub1 via MPLS, then breakout to SaaS Cloud via INET(DIA).

Exchange/Citrix: Backhaul from SJC-branch to RTP-Hub1/Hub2 via MPLS/INET, then toward Campus/DC via LAN. **Both apps have some flows run into asymmetry on some hops.**

Packet-trace (aka fia-trace)



Recap: The Packet Tracer and FIA Debugger



Enabling Packet-trace

```
cedge1#debug platform condition ipv4 <ip_address>/32 both
cedge1#debug platform condition start
cedge1#debug platform packet-trace packet <number of packets> fia-trace
```

Optionally

```
cedge1# debug platform packet-trace copy packet both size <…>
```

Show commands:

```
cedge1#show platform packet-trace summary
Pkt    Input                     Output                    State  Reason
0      Gi2                       Gi3                       FWD
1      Tu3                       Gi2                       FWD
2      INJ.3                     Gi2                       FWD
3      internal0/0/recycle:0     Gi2                       FWD
4      Gi2.                      Tu3                       DROP   493 (NoStatsUpdate)
5      internal0/0/recycle:0     Gi2                       FWD
……
cedge1#debug platform condition stop
cedge1# show platform packet-trace packet <packet number>
```

Can you understand the Packet-trace output?

```
cedge1#show platform packet-trace packet 0
Packet: 0           CBUG ID: 35949496
Summary
  Input     : GigabitEthernet2
  Output    : GigabitEthernet3
  State     : FWD
  Timestamp
    Start   : 1214211941024994 ns (02/24/2020 11:03:14.435466 UTC)
    Stop    : 1214211941530105 ns (02/24/2020 11:03:14.435971 UTC)
Path Trace
  Feature: IPV4(Input)
    Input       : GigabitEthernet2
    Output      : <unknown>
    Source      : 192.168.11.254
    Destination : 192.168.17.254
    Protocol    : 1 (ICMP)
<remove>
  Feature: SDWAN ACL IN
    Interface    : GigabitEthernet2
    CG           : 3
    Seq          : 21
    Policy Flags : 0x100
    Action : SET_FWD_CLASS 3 Prec3
  Feature: SDWAN_ACL_IN
    Entry       : Input - 0x81845740
    Input       : GigabitEthernet2
    Output      : <unknown>
    Lapsed time : 815733 ns
<removed>
```

SD-WAN ACL matches flow and assign to QoS class ”Prec3”

Packet-trace – obvious parts

```
Feature: NBAR
    Packet number in flow: N/A
    Classification state: Final
    Classification name: ping
<removed>
Feature: SDWAN App Route Policy
    VRF          : 1
    CG           : 1
    Seq          : 65535
    SLA          : __all_tunnels__ (0)
    Policy Flags : 0x2
    SLA Strict   : No
    Preferred Color : 0x0 none
<removed>
Feature: SDWAN OCE
    Hash Value  : 0xaf6f0c4e
    Encap       : ipsec
    SLA         : 0
    SDWAN VPN   : 1
    SDWAN Proto : IPV4
    Out Label   : 1001
    Local Color : biz-internet
    Remote Color: biz-internet
    FTM Tunnel ID:15
    SDWAN Session Info
      SRC IP      : 172.16.11.254
      SRC Port    : 12346
      DST IP      : 172.16.17.254
      DST Port    : 12346
      Remote System IP : 172.16.255.17
```

NBAR classification is complete
Application is recognized

This flow does not match any app-route policies so it's load-balanced to all tunnels

Forwarding decision

Packet-trace – not-so-obvious parts

```
Feature: NBAR
    Packet number in flow: N/A
    Classification state: Final
    Classification name: ping
<removed>
Feature: SDWAN App Route Policy
    VRF           : 1
    CG            : 1
    Seq           : 65535
    SLA           : __all_tunnels__ (0)
    Policy Flags : 0x2
    SLA Strict    : No
    Preferred Color : 0x0 none
<removed>
Feature: SDWAN OCE
    Hash Value  : 0xaf6f0c4e
    Encap       : ipsec
    SLA         : 0
    SDWAN VPN   : 1
    SDWAN Proto : IPV4
    Out Label   : 1001
    Local Color : biz-internet
    Remote Color: biz-internet
    FTM Tunnel ID:15
    SDWAN Session Info
      SRC IP      : 172.16.11.254
      SRC Port    : 12346
      DST IP      : 172.16.17.254
      DST Port    : 12346
      Remote System IP : 172.16.255.17
```

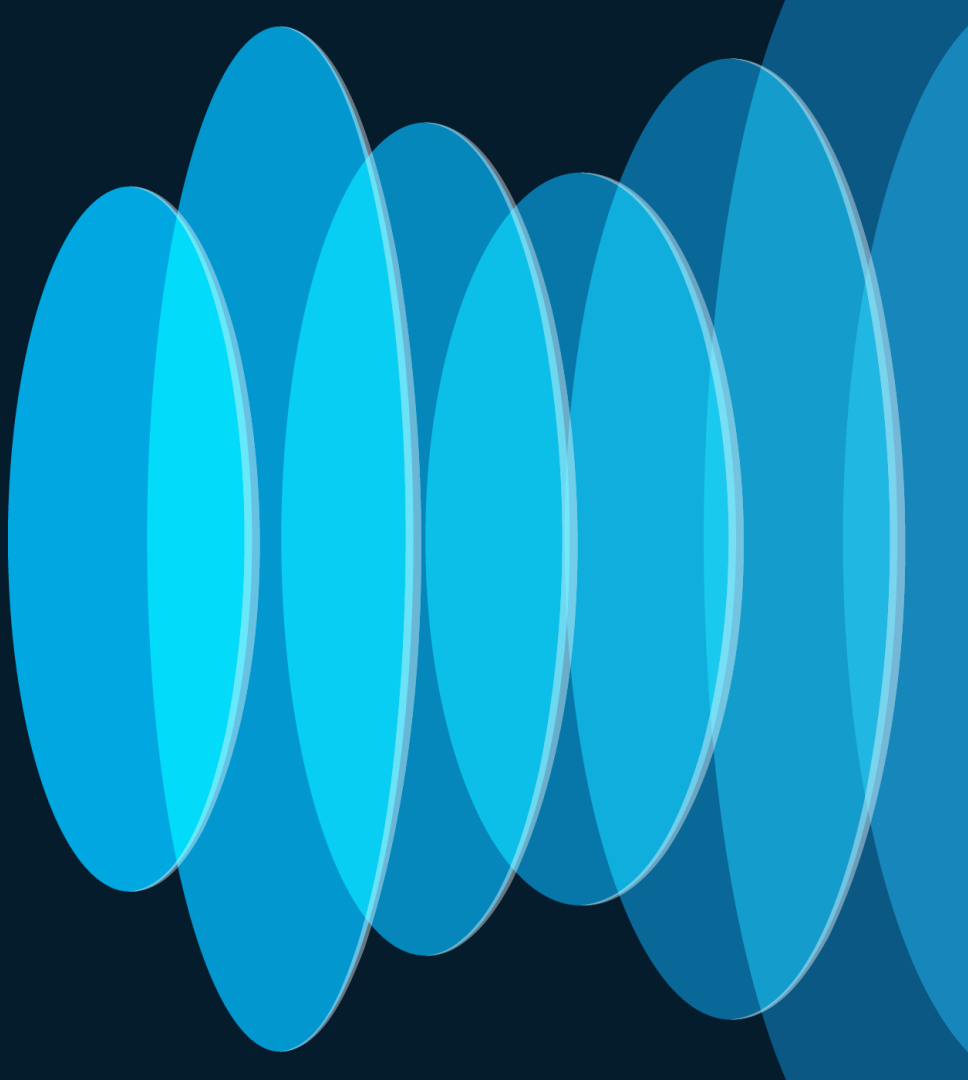
NBAR classification is complete
Application is recognized

This flow does not match any app-route policies so it's load-balanced to all tunnels

Wait…What is OCE? (more details later)

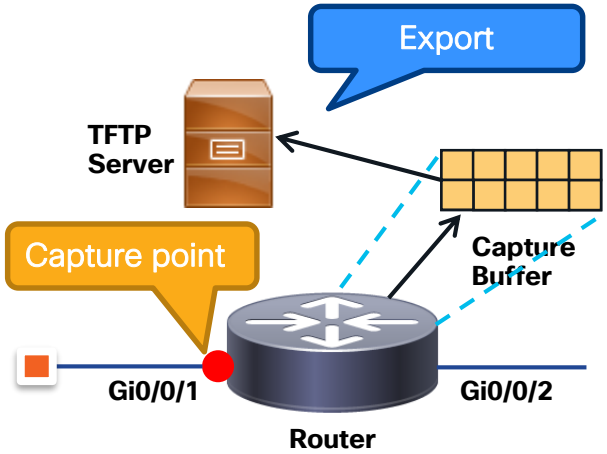
Forwarding decision

Embedded Packet Capture (EPC)



Embedded Packet Capture

- Use when you suspect traffic of interest is not reaching or not egressing your router
- Recommendations and caveats:
  - Circular option is your best friend with EPC
  - Be mindful of capture rate
  - Make sense to combine with packet-trace

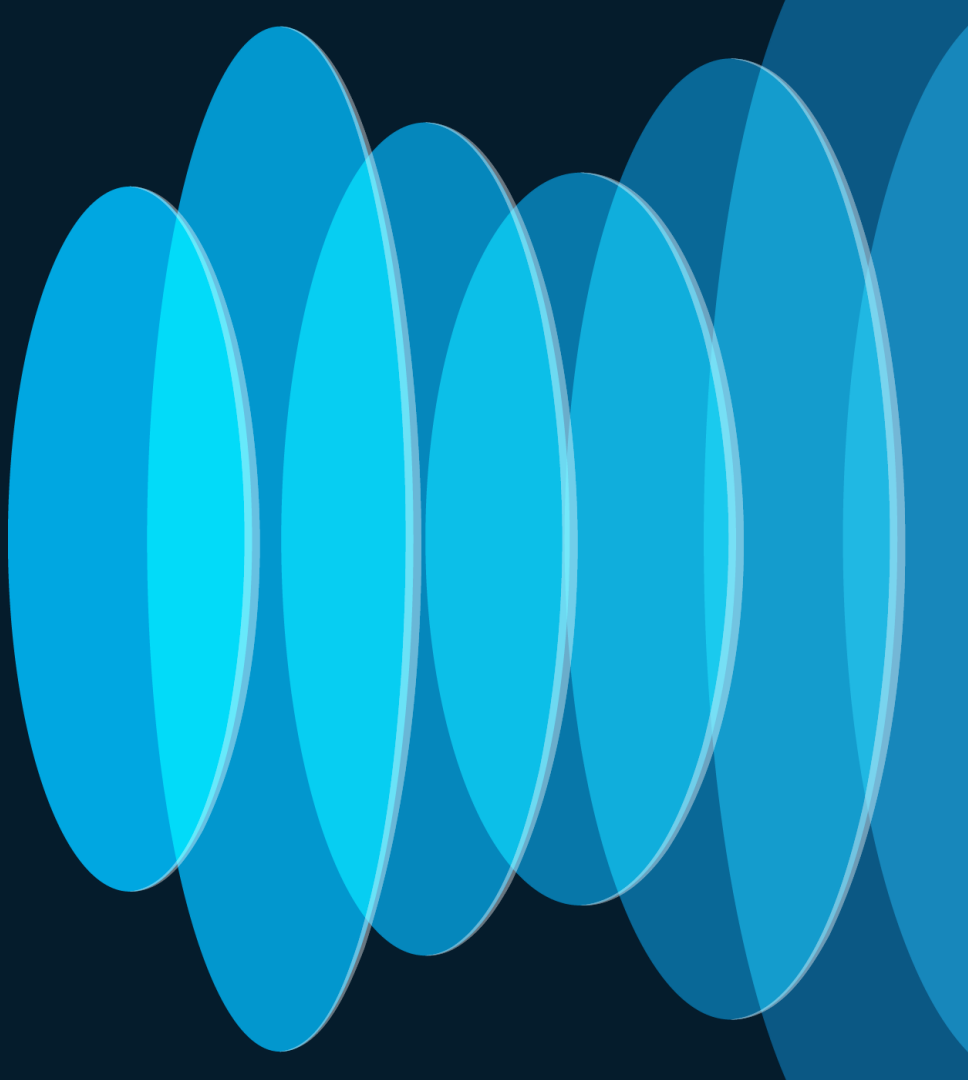



```
Device# monitor capture mycap match ipv4 host 1.1.1.1 host 2.2.2.2 bidirectional

Device# monitor capture mycap limit duration 1000
Device# monitor capture mycap interface GigabitEthernet 0/0/1 both
Device# monitor capture mycap buffer circular size 10
Device# monitor capture mycap start

<This is the timespan where the you're capturing the traffic of interest>

Device# monitor capture mycap stop

Device# monitor capture mycap export tftp://10.1.88.9/mycap.pcap
```

What about the control-plane?

bTrace

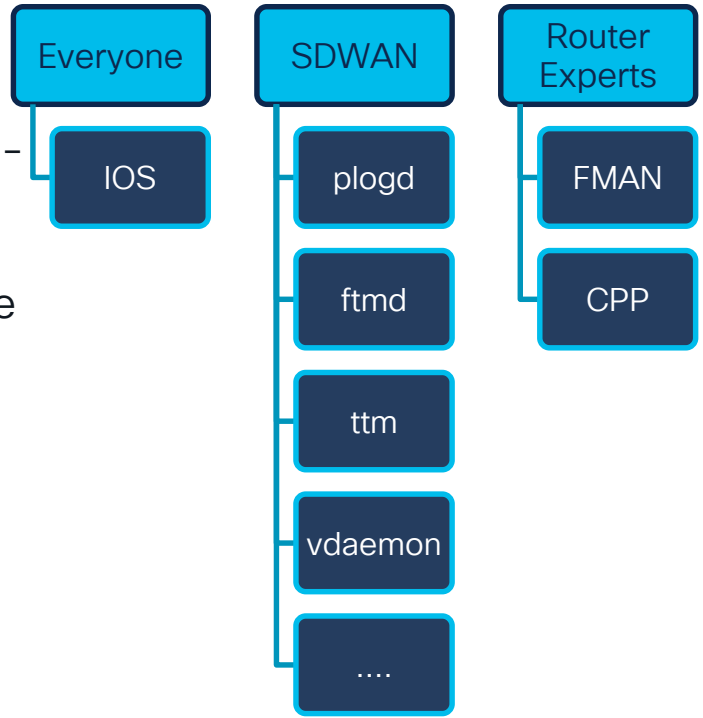
tool for process-specific troubleshooting



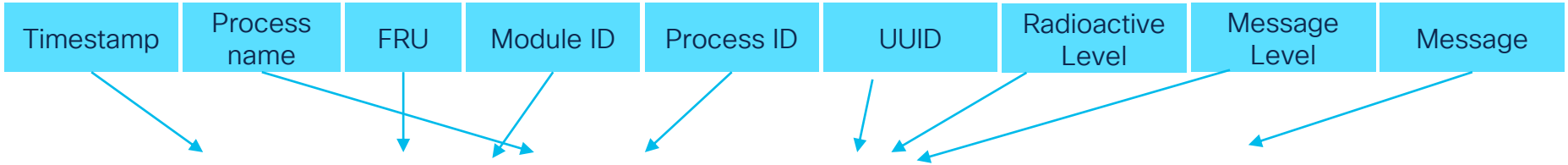
bTrace – who are the usual suspects for the tool?

bTrace stands for Binary Trace

- a dedicated binary log collected for every process running on top of IOS XE (including SD-WAN processes)
- Is your best friend to troubleshoot control-plane beyond regular show commands
- You better know where to look (the data available is huge). Process names might be cryptic ☹ Will address that later today
- Enabled with “Notice” level by default – remember to enable “Debug” level



Let’s try bTrace!



Message levels	
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice
6	Info
7	Debug

How to enable bTrace logging for a process (before 17.10):

debug platform software sdwan <sdwan process> <feature>

Example: **debug platform software sdwan ftmd bfd**

How to enable bTrace logging for a process (after 17.10):

set platform software trace <sdwan process> R0 <feature> debug

Example: **set platform software trace ftmd R0 ftmd-bfd debug**

Easy access with:

show logging process <process name> internal

Example: **show logging process ftmd internal**

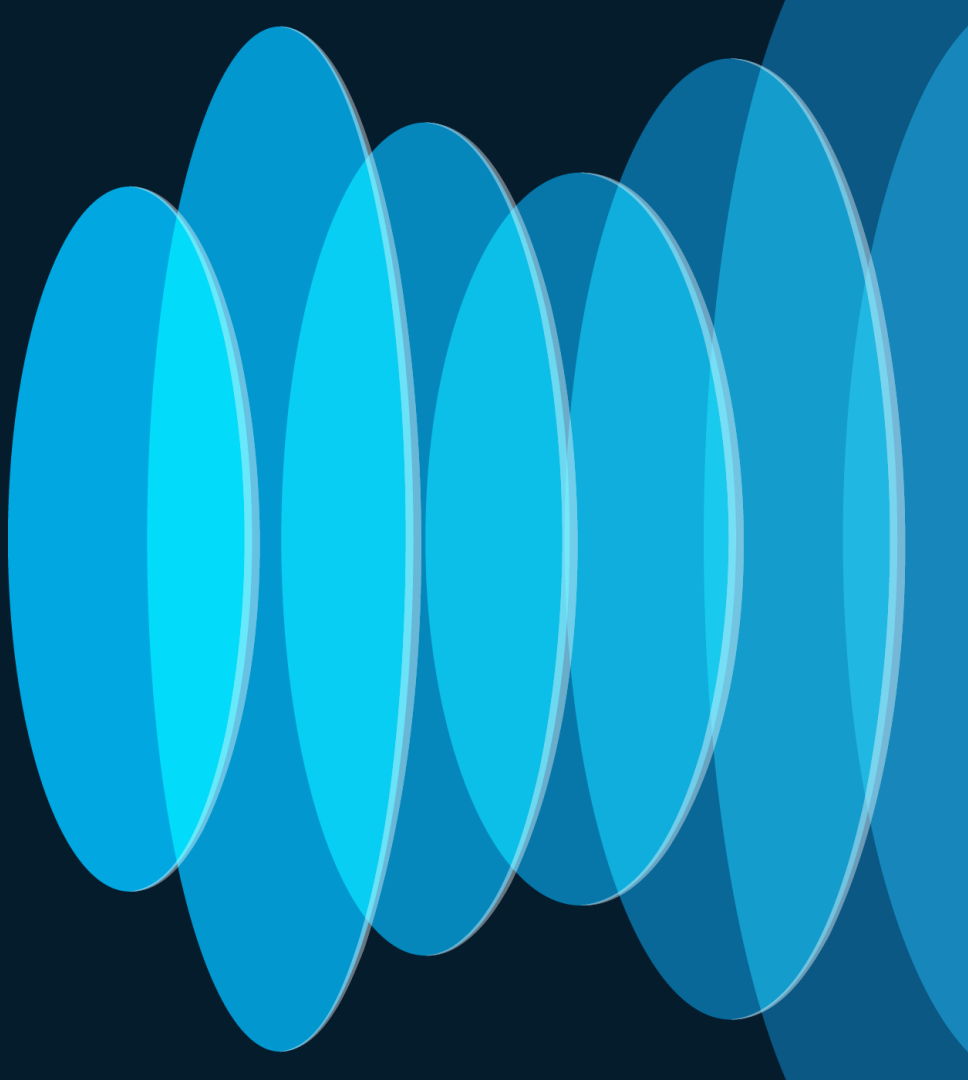
Better use ‘sdwan’ profile for bTrace

```
edge1#show logging profile ?
  all                    all processes
hardware-diagnostics  hardware diagnostics specific processes
  netconf-yang           netconf-yang specific processes
  restconf               restconf specific processes
  sdwan                  SDWAN specific processes
  wireless               Wireless specific processes

edge1#show logging profile sdwan internal start last 1 day
```

For the reference - profile “sdwan” includes:

plogd,viptela_start,cpp_cp,cxpd,ttm,dmiauthd,confd,nginx,pttcd,pubd,ndbmand,IOS,fman_rp,fman_fp,vip_confd_startup,vdaemon,fpmd,ftmd,ompd,binos,cfgmgr,dbgd

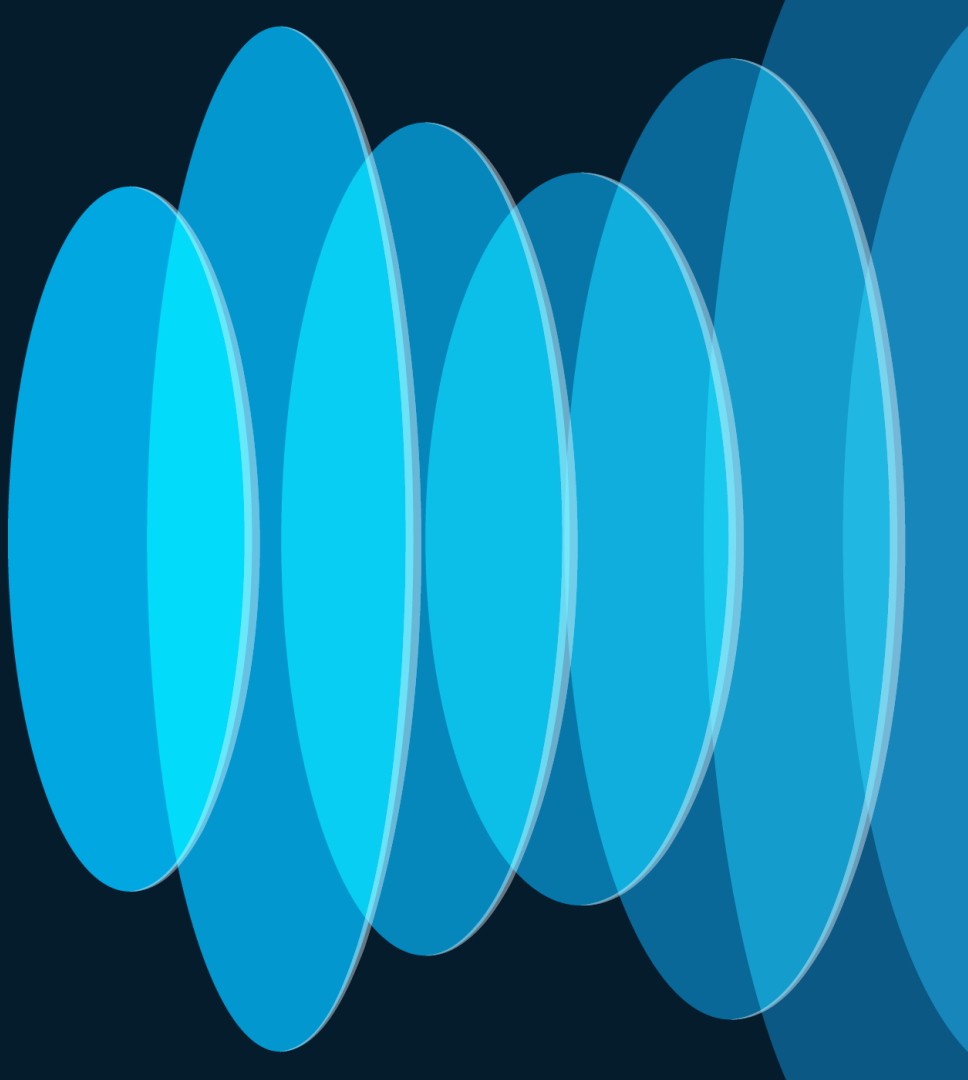
Have you noticed the blind spots with the tools?



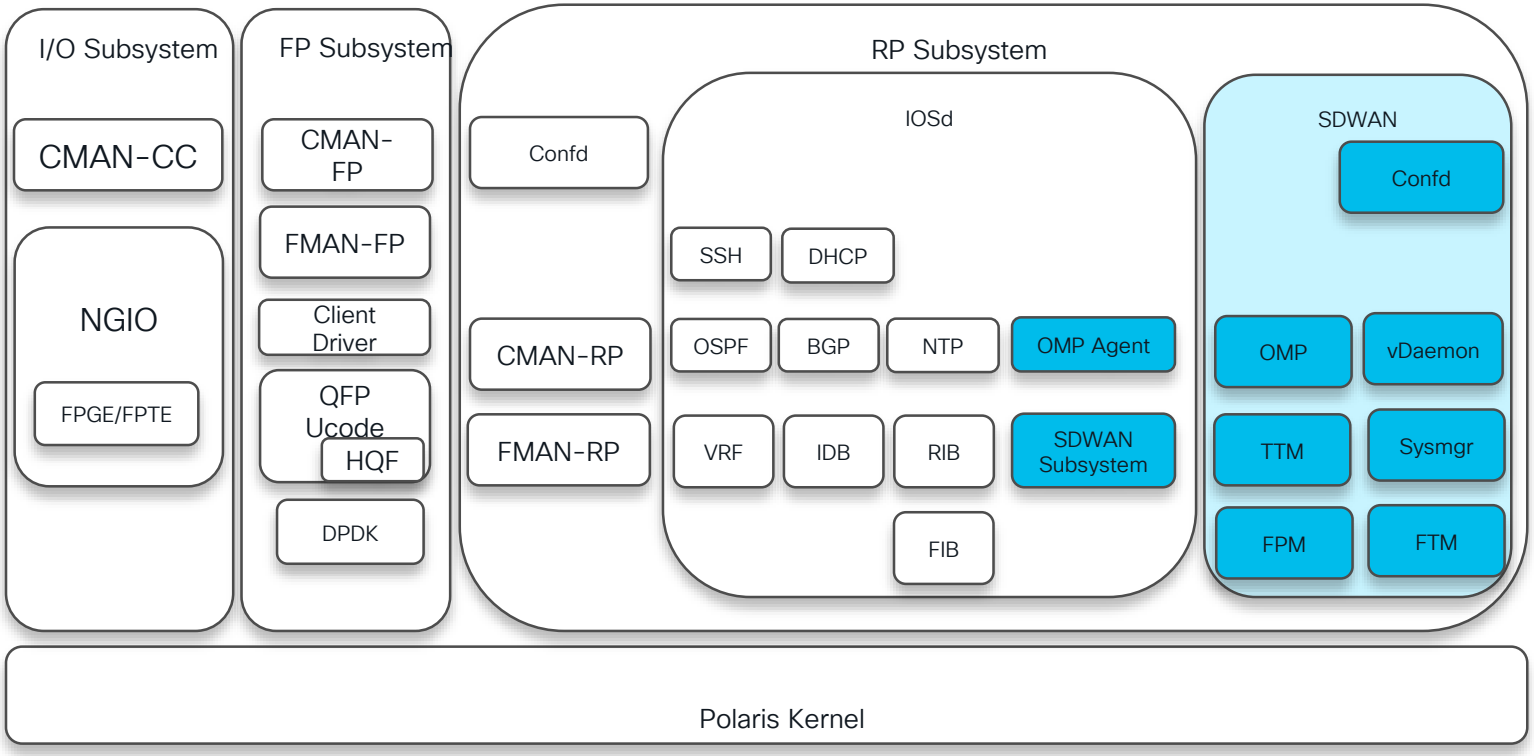
There’s no ”silver-bullet” for troubleshooting

- **Packet-trace** -> cryptic, device-specific, no control-plane visibility
- **Embedded Packet Capture** -> device-specific, limited rate, no correlation with Packet-trace/NWPI
- **NWPI** -> less detailed than Packet-trace, requires control-plane to be up and running, could be cryptic sometimes
- **bTrace** – how to identify the process name to trace? How to translate cryptic output?

Let’s improve your bTrace’s translation skills

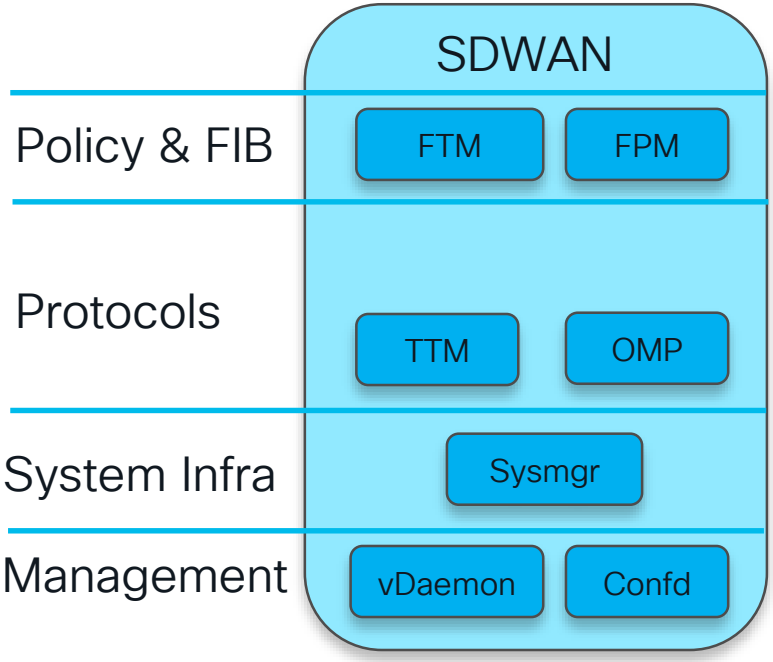


XE SD-WAN Software Architecture

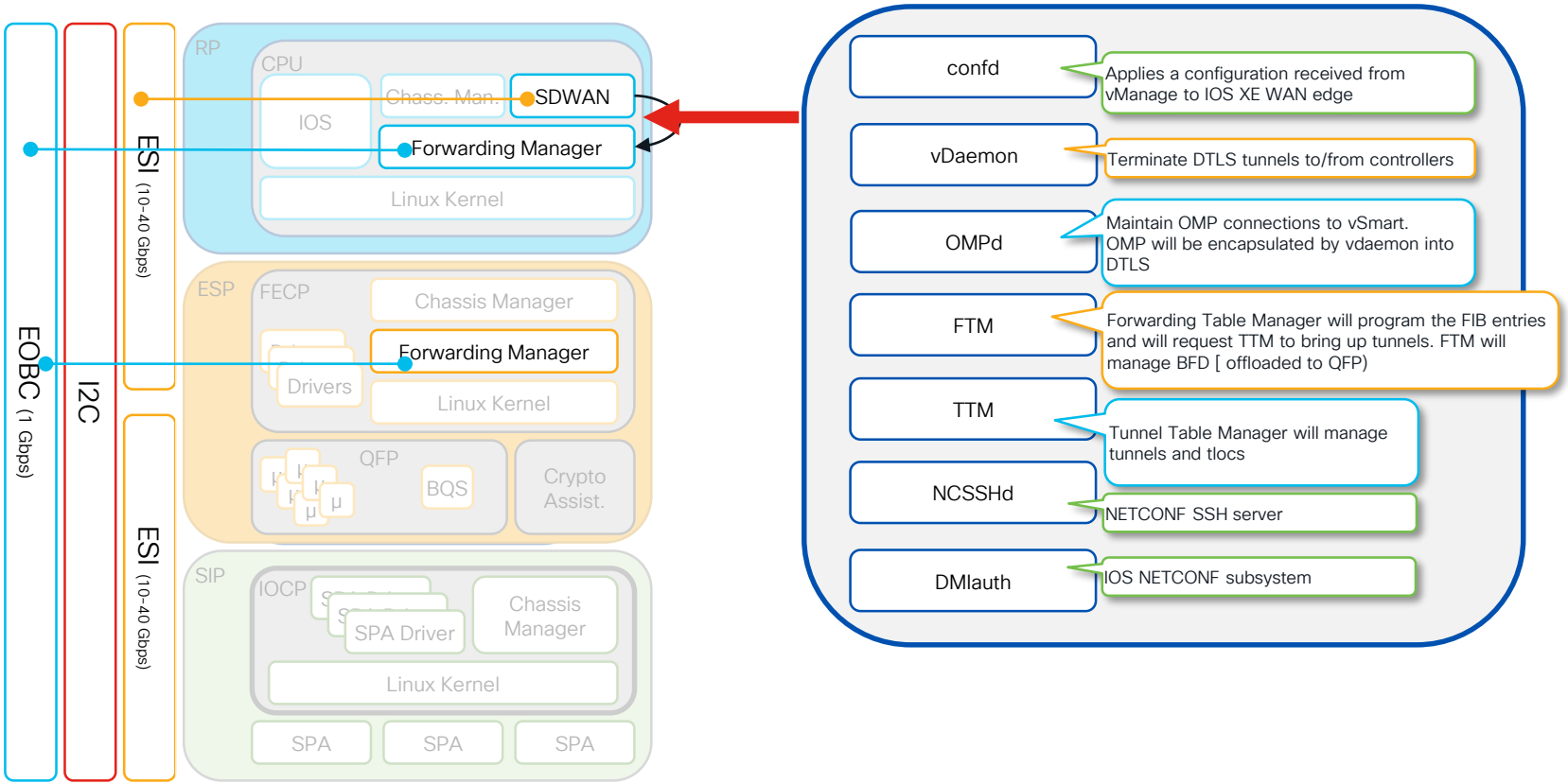


SD-WAN processes – deciphering acronyms

- vDaemon: SDWAN Software Process
- Confd: Configuration Process
- Sysmgr: System Manager Process
- TTM: Tunnel Table Manager
- OMP: Overlay Management Protocol
- FPM: Forwarding Policy Manager
- FTM: Forwarding Table Manager

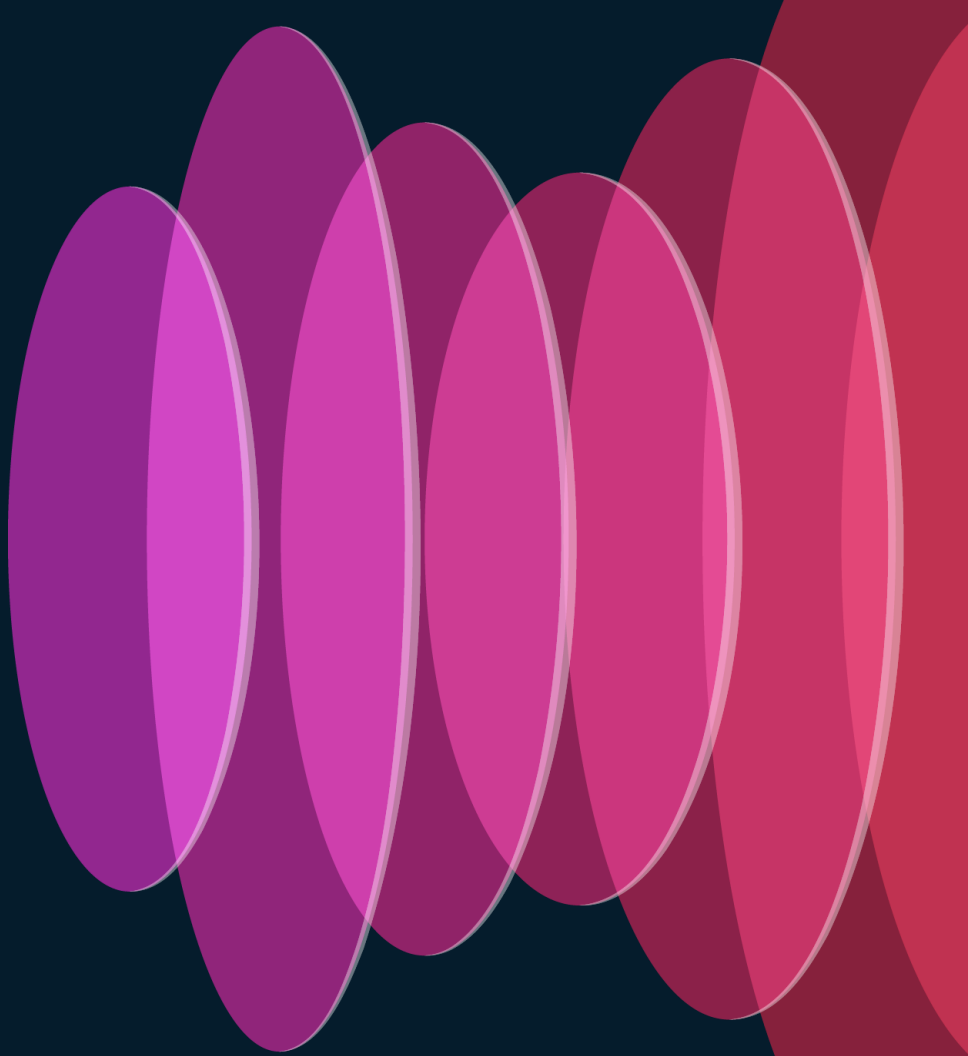


SDWAN processes – where they're? what they do?



vDaemon

The one for control-plane DTLS/TLS tunnels



Having issues with control-plane connections?

```
cE1_BR1#show sdwan control connections
                                                   PEER                 PEER                                          CONTROLLER
PEER    PEER PEER            SITE  DOMAIN PEER        PRIV  PEER           PUB                                         GROUP
TYPE    PROT SYSTEM IP       ID    ID     PRIVATE IP  PORT  PUBLIC IP      PORT  ORGANIZATION   LOCAL COLOR    PROXY STATE UPTIME     ID
------------------------------------------------------------------------------------------------------------------
vsmart  tls  10.0.0.101      101   1      192.168.2.4 23556 192.168.2.4    23556 poctool-1     mpls            No    up     4:20:41:30 1
vsmart  tls  10.0.0.101      101   1      192.168.2.4 23556 192.168.2.4    23556 poctool-1     biz-internet    No    up     4:20:41:25 1
vmanage tls  169.254.206.7   1     0      192.168.2.7 23756 192.168.2.7    23756 poctool-1      mpls           No    up     4:20:43:36 0
```

```
cE1_BR1#show sdwan control connection-history  | b PEER
                                                       PEER                  PEER
PEER    PEER     PEER          SITE  DOMAIN PEER        PRIVATE PEER          PUBLIC                        LOCAL     REMOTE REPEAT
TYPE    PROTOCOL SYSTEM IP     ID    ID     PRIVATE IP  PORT    PUBLIC IP     PORT   LOCAL COLOR  STATE     ERROR     ERROR  COUNT ORGANIZATION   DOWNTIME
------------------------------------------------------------------------------------------------------------------------------------------
vbond   dtls     0.0.0.0       0     0      192.168.2.2 12346   192.168.2.2   12346  biz-internet tear_down DISCVBD   NOERR  0     2022-12-16T17:06:10+0000
vbond   dtls     0.0.0.0       0     0      192.168.2.2 12346   192.168.2.2   12346  mpls         tear_down DISCVBD   NOERR  0     2022-12-16T17:06:06+0000
vsmart  dtls     10.0.0.102    102   1      192.168.2.5 12446   192.168.2.5   12446  biz-internet tear_down XTVSTRDN  NOERR  0     2022-12-16T17:05:53+0000
vsmart  dtls     10.0.0.102    102   1      192.168.2.5 12446   192.168.2.5   12446  mpls         tear_down XTVSTRDN  NOERR  0     2022-12-16T17:05:49+0000
vsmart  tls      10.0.0.101    101   1      192.168.2.4 23556   192.168.2.4   23556  biz-internet trying    DCONFAIL  NOERR  5     2022-12-16T17:05:28+0000
vsmart  tls      10.0.0.101    101   1      192.168.2.4 23556   192.168.2.4   23556  mpls         trying    DCONFAIL  NOERR  5     2022-12-16T17:05:24+0000
```

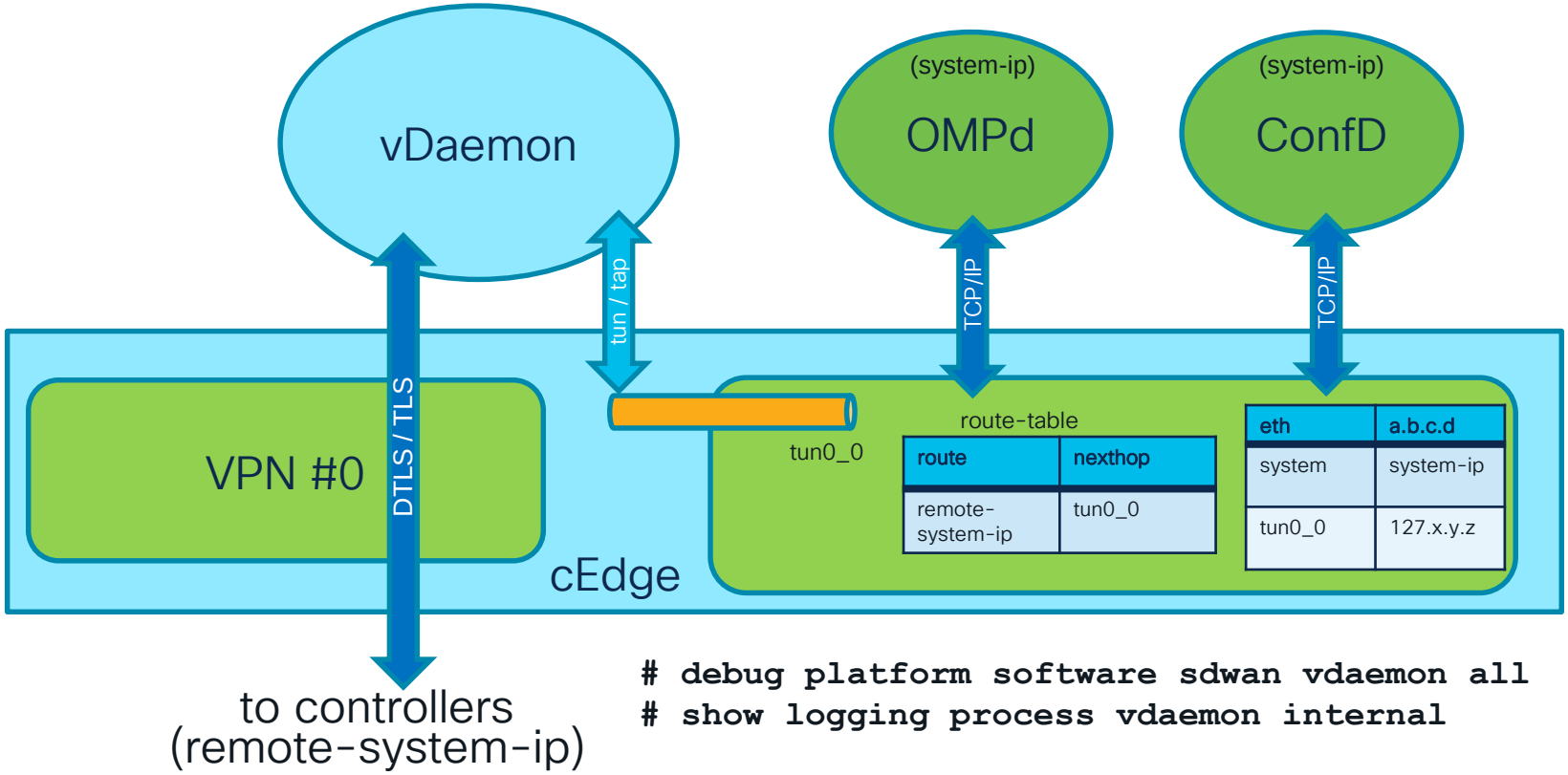
vBond:

```
show orchestrator connections
Show orchestrator connections-history
```

vSmart/vManage/vEdge:

```
show control connections
show control connection-history
```

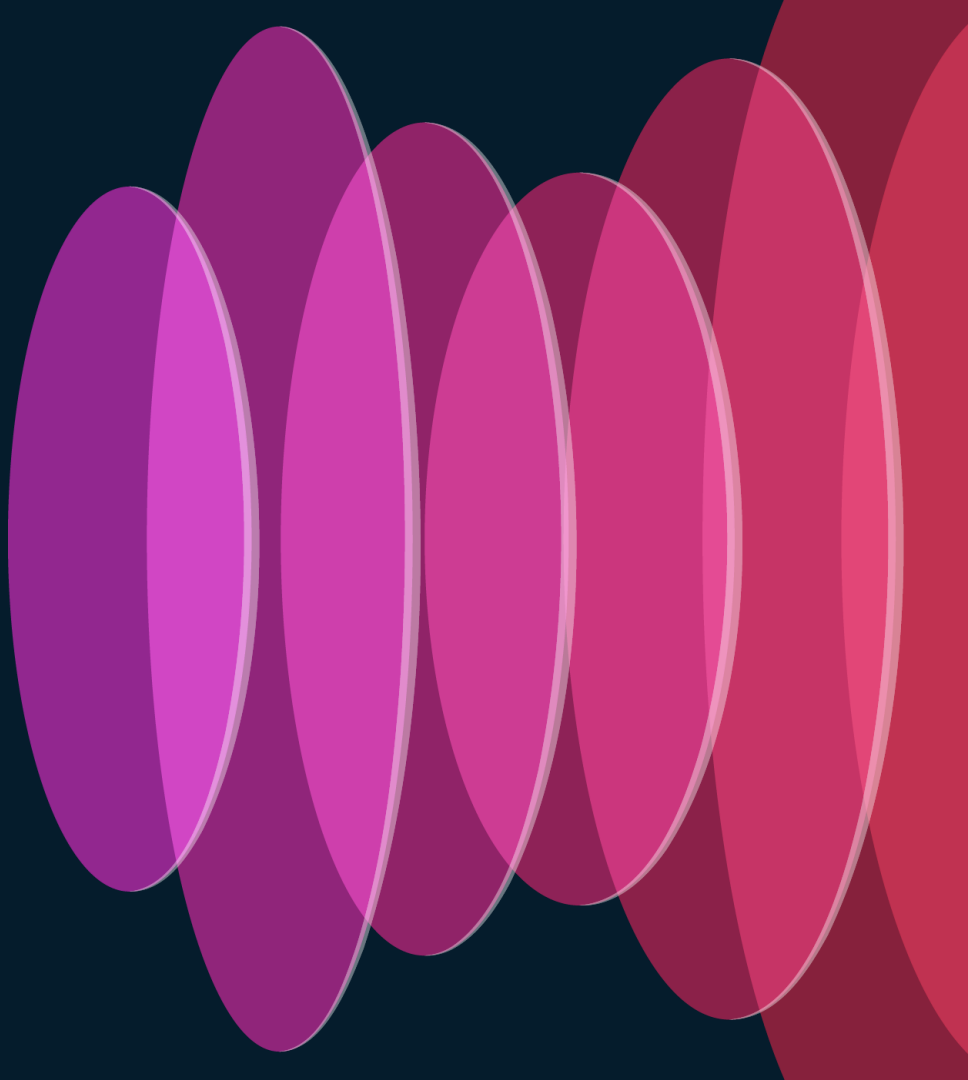
vDaemon process in a nutshell



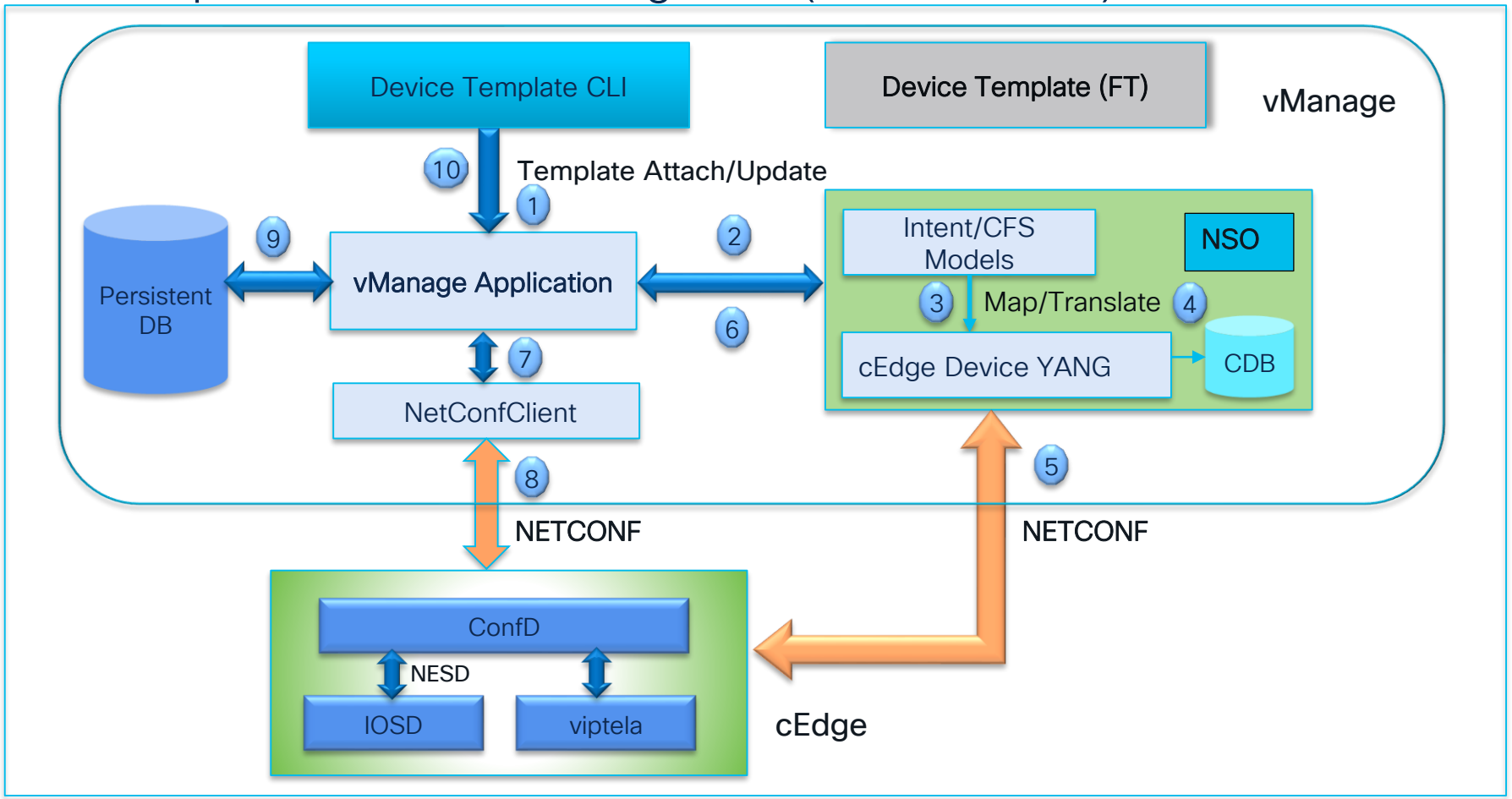

```
# debug platform software sdwan vdaemon all
# show logging process vdaemon internal
```

ConfD

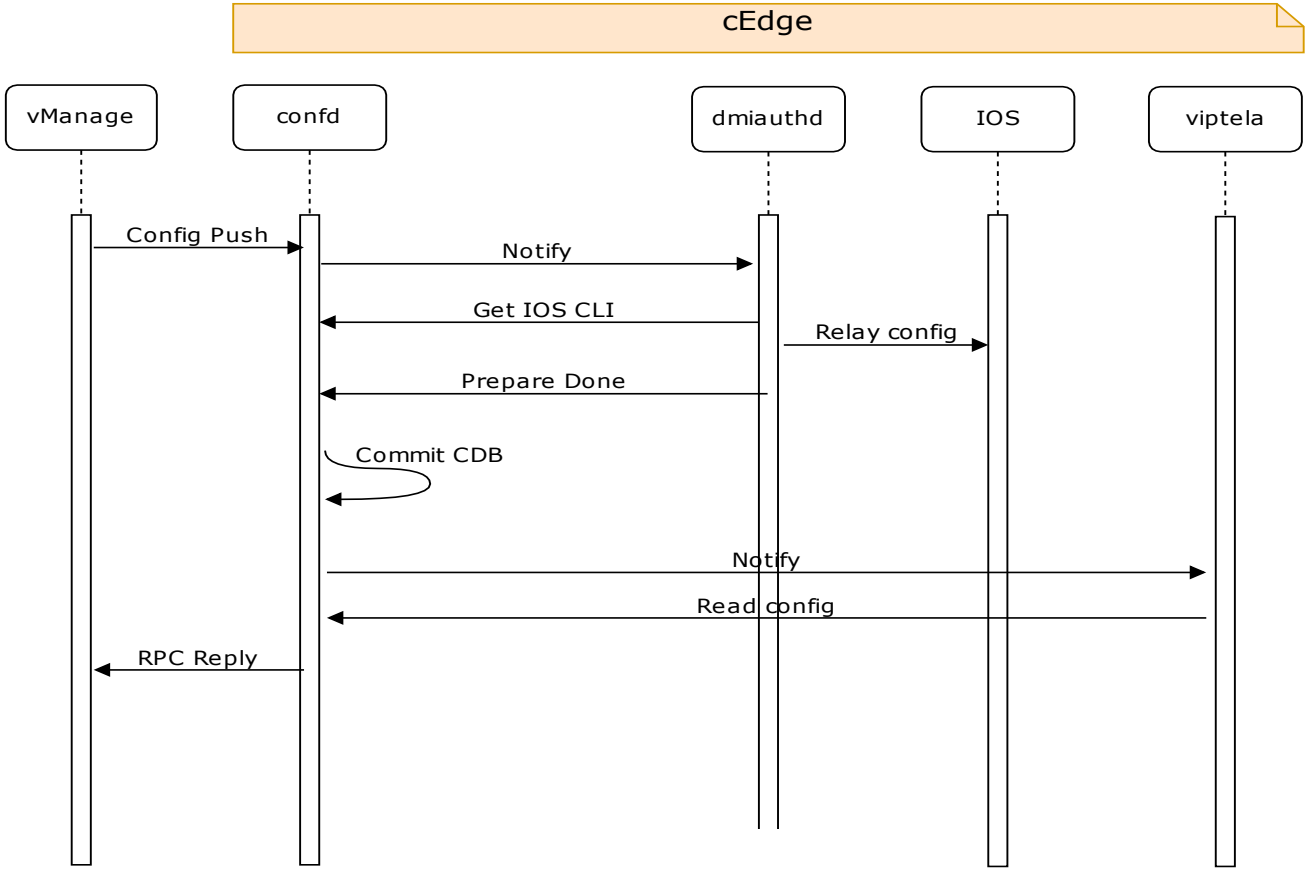
The one to apply configuration changes



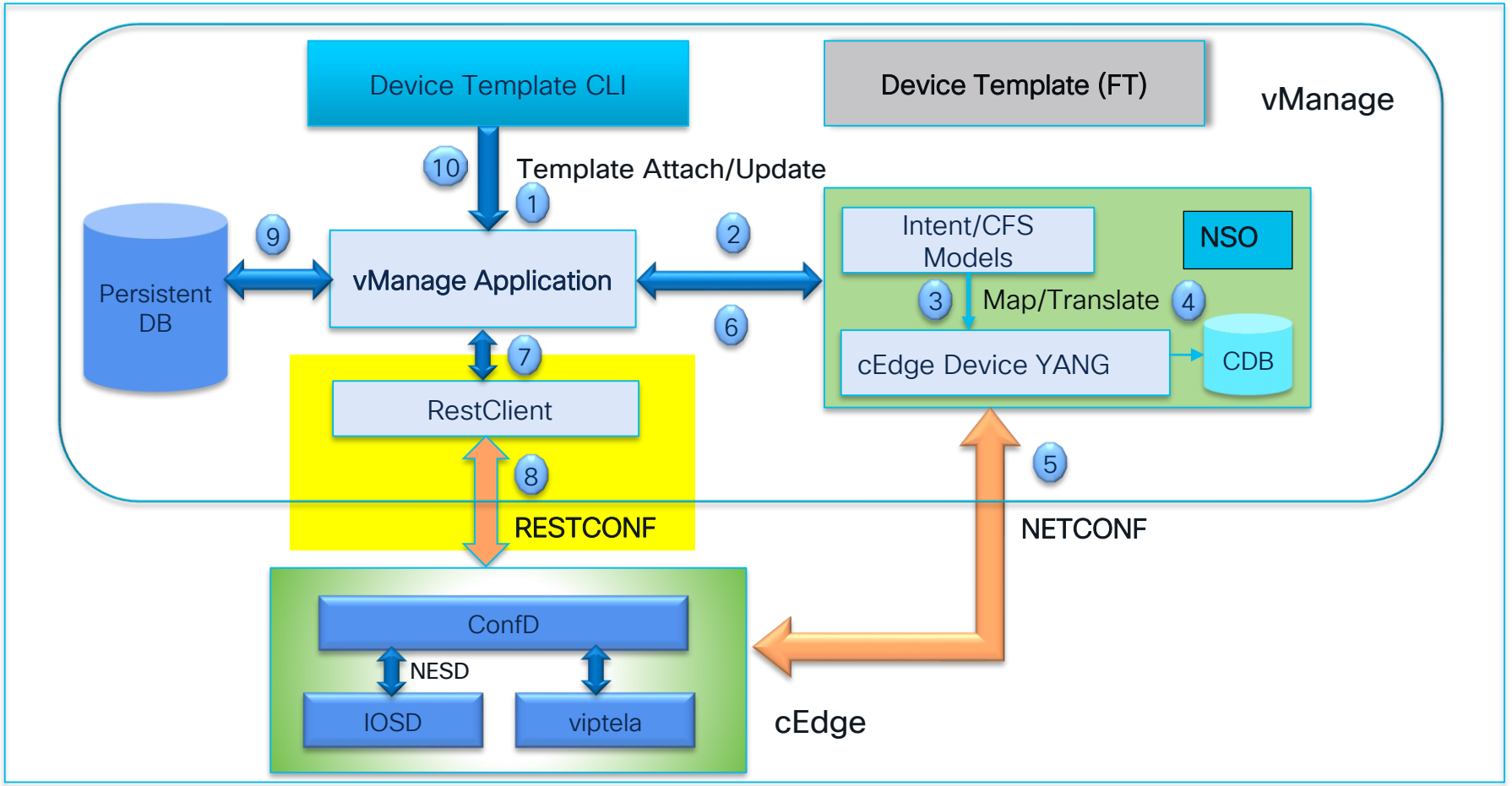
Template attach and Config Push (Device Online) - Pre 20.6



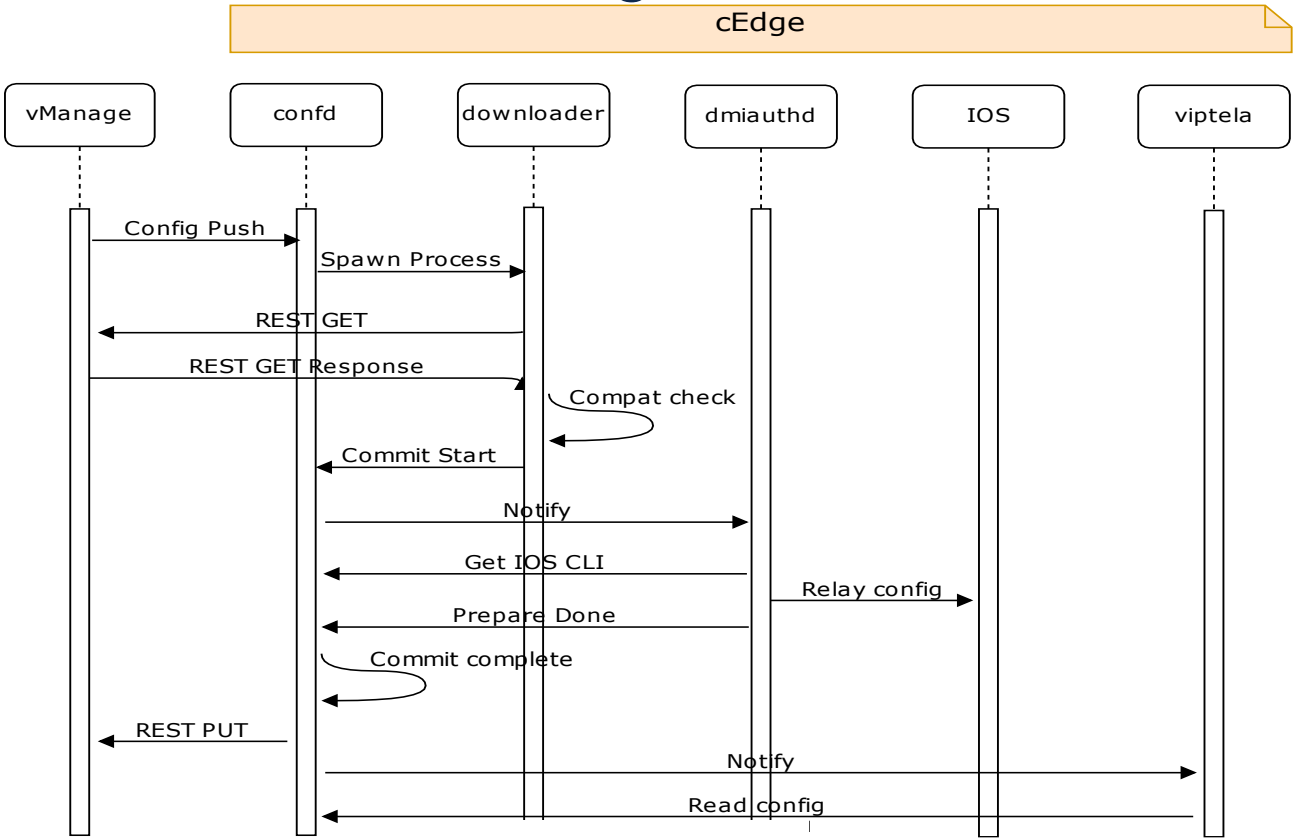
ConfD behavior before 17.6.x (push-mode)



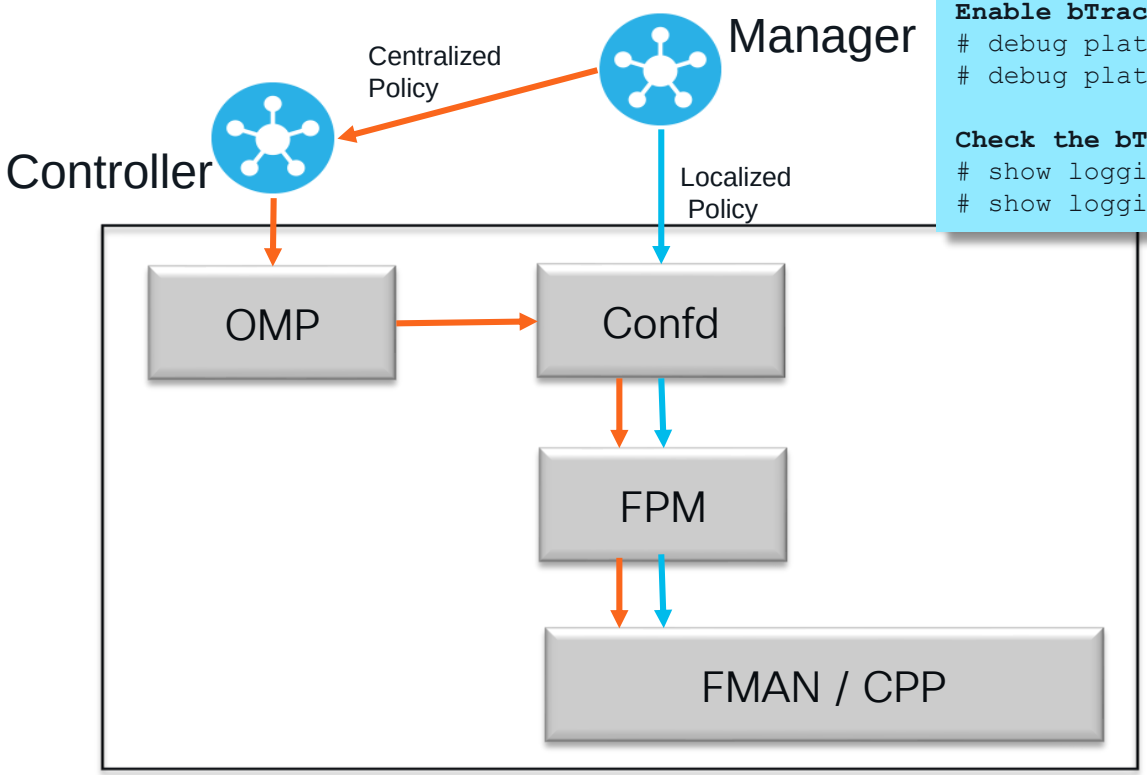
Template attach and Config Pull (Device Online) - 20.6+



ConfD behavior starting 17.6.x – cont.



ConfD for policy download



Enable bTrace for confd:

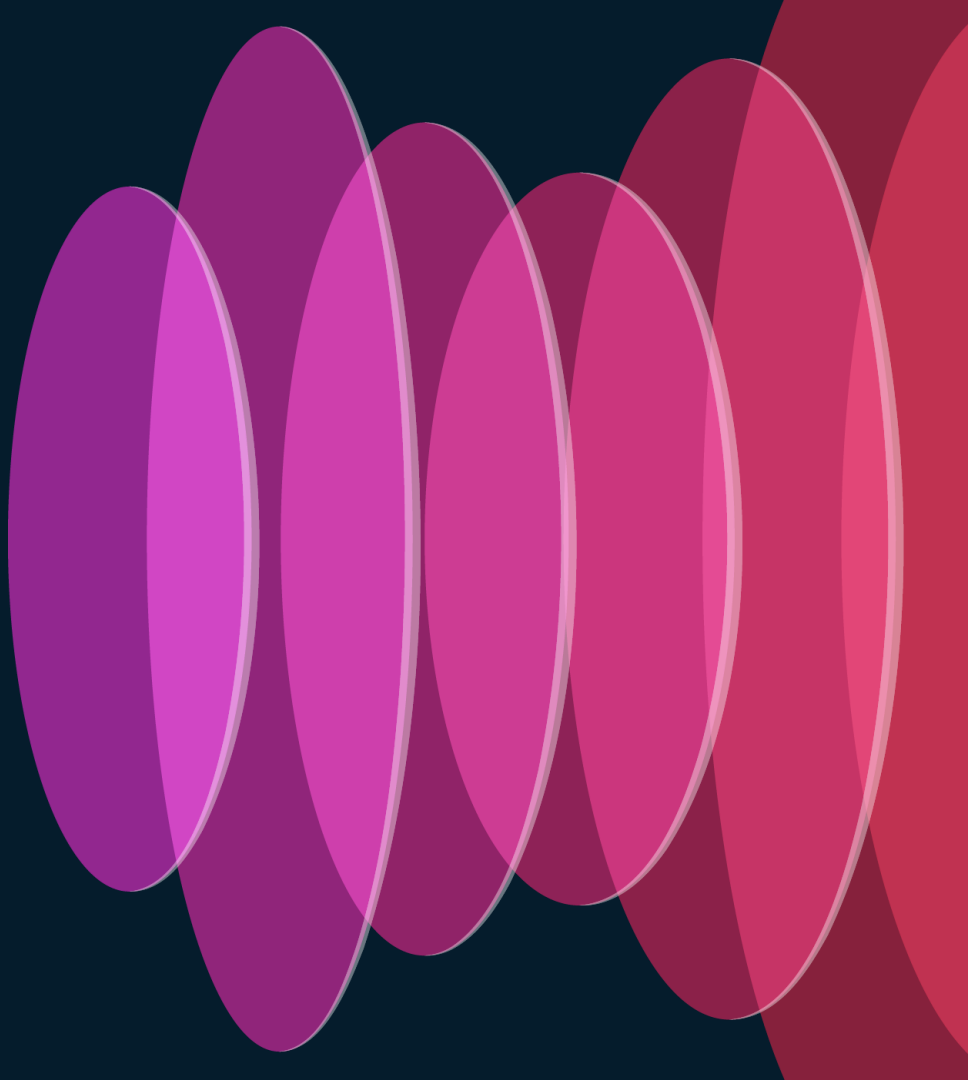
```
# debug platform software sdwan confd <>
# debug platform software sdwan config-mgr <>
```

Check the bTrace for ConfD:

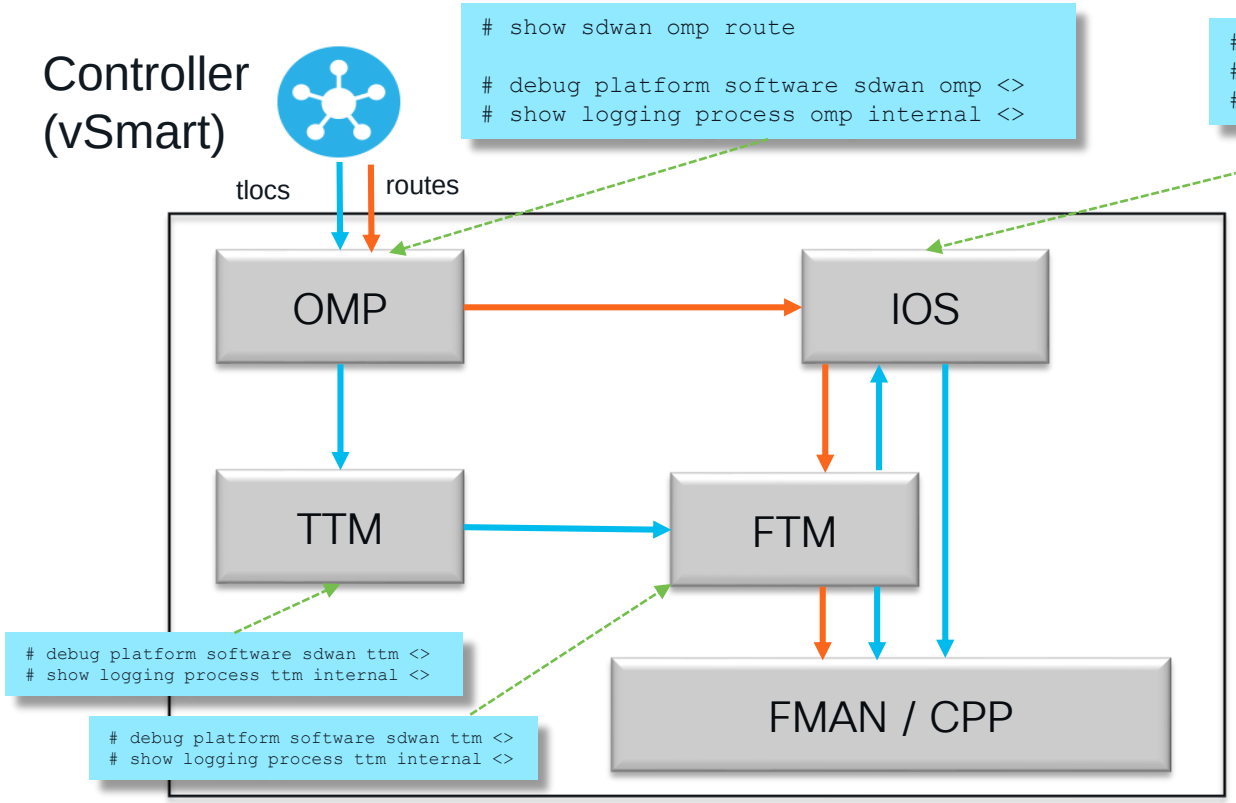
```
# show logging process confd internal <>
# show logging process config-mgr internal <>
```

OMPd, TTMd, FTMd

The ones for SDWAN FIB



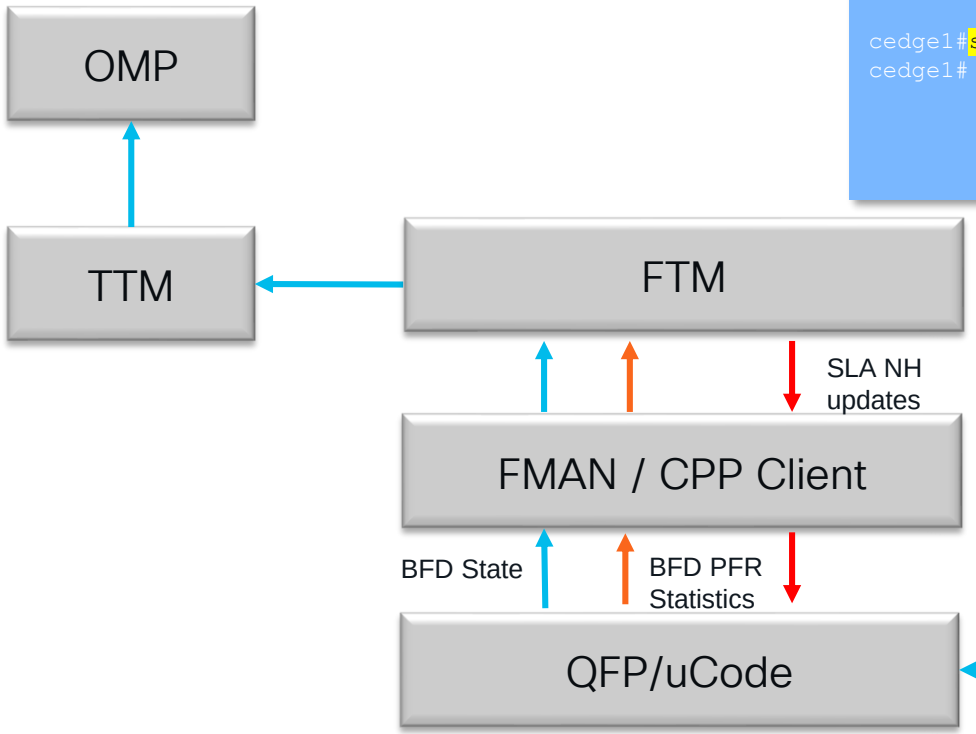
OMP: TLOCs and Route download



1. Routes come from:
   - IOS for connected networks
   - OMP for remote locations
2. IOS combines both into RIB
3. Based on multiple inputs (RIB, TTM, BFD) FTM builds and dynamically updates SDWAN FIB for all overlay routes (excluding locally connected routes – these will be handled by IOS CEF)*
4. FMAN/CPP takes both inputs from SDWAN FIB and IOS CEF and programs the data-plane

***NAT DIA route** is also installed via FTM and missing from CEF.

BFD and App-Route statistics



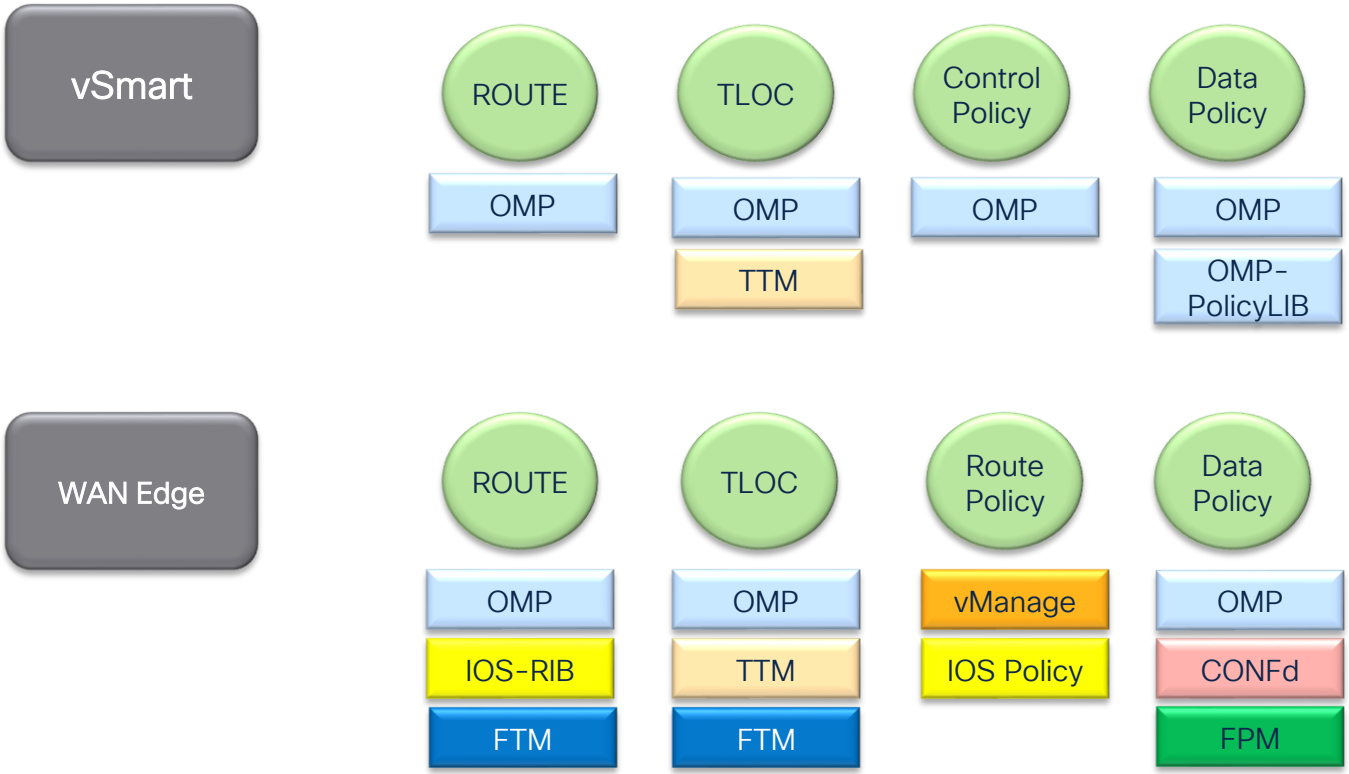

```
cedge1#show bfd summary
cedge1#
```

SDWAN BFD is not managed by IOS!
It's managed by FTM!

```
# debug platform software sdwan ftmd bfd
# show logging process ftmd internal
```

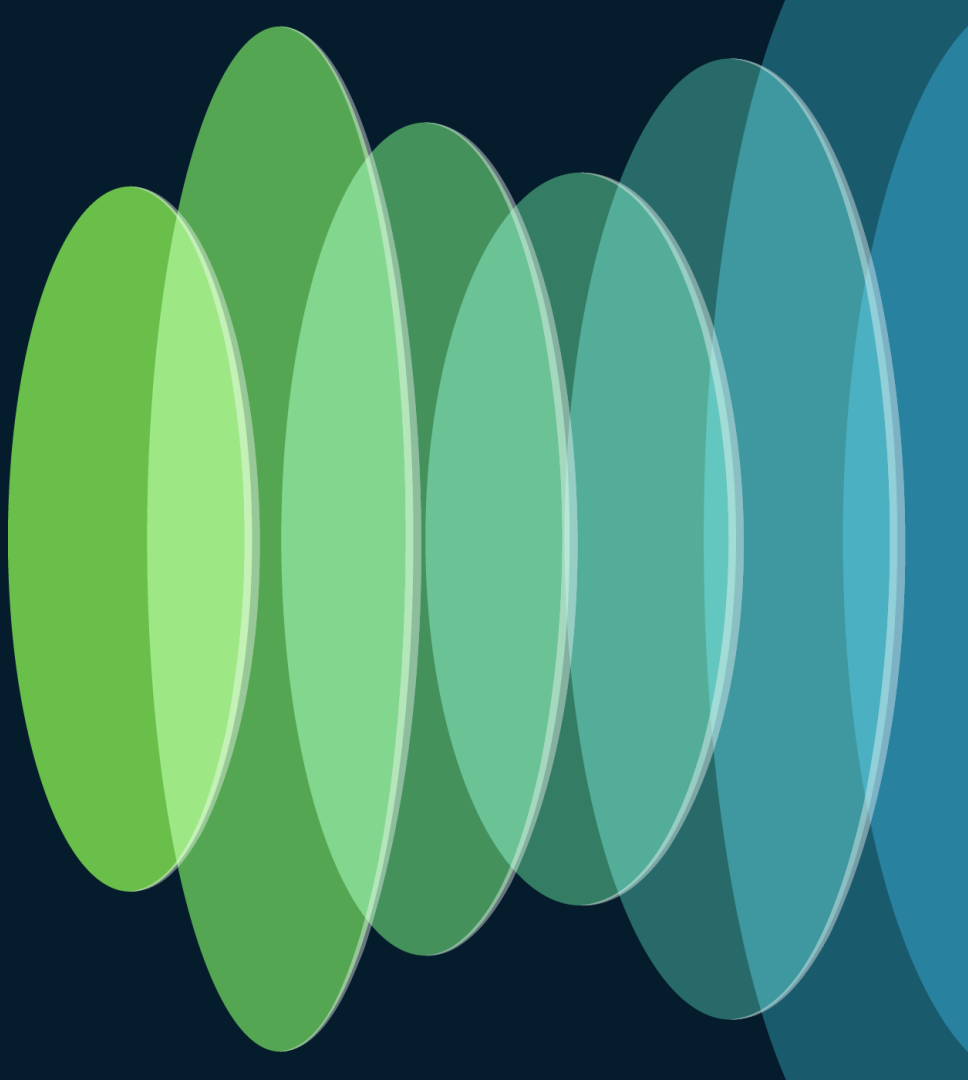
BFD echo / pmtu packets

OMP-related issues.
Which process to look for and where?

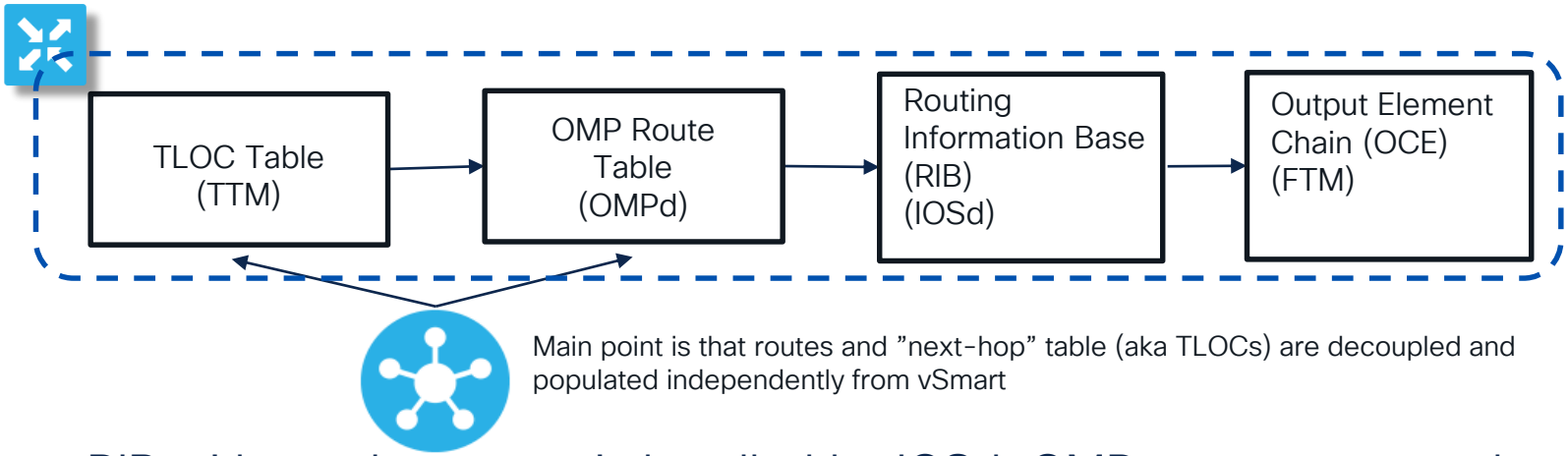


Connecting the dots...

How SD-WAN FIB really works? And why it works that way?

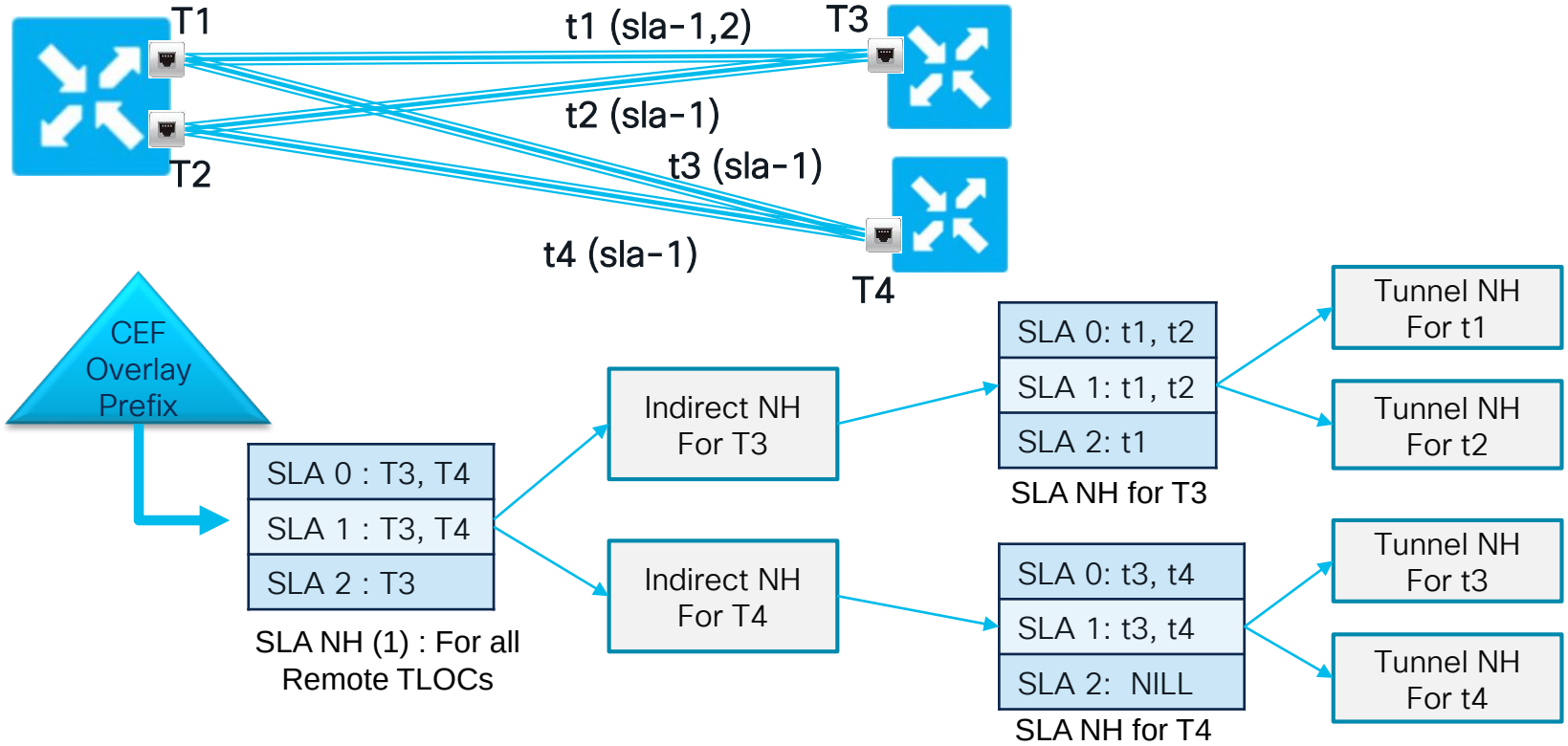


SD-WAN FIB = Output Element Chain (OCE)



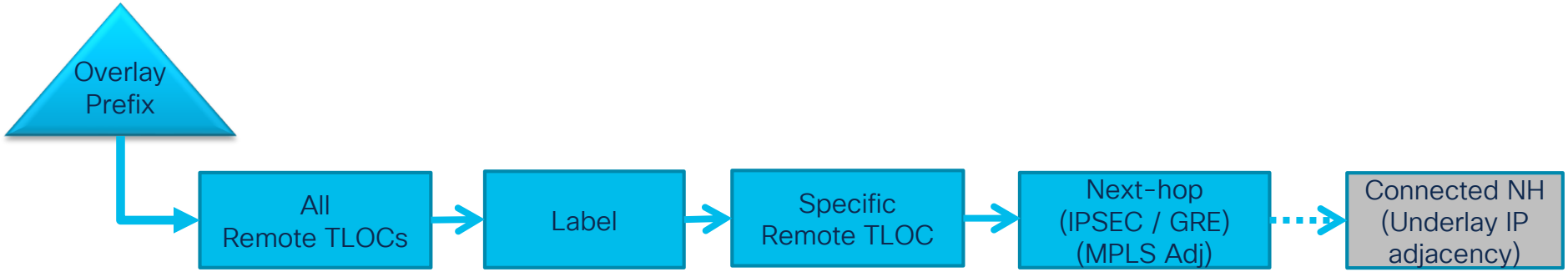
- RIB with overlay routes is handled by IOSd. OMP routes are populated into RIB via OMP-agent sitting next to IOSd
- FIB: IOSd's CEF is used for LAN-side routes while Output Chain Element(OCE) is used for overlay routes (not locally originated). OCE is populated by FTM (based on TTM, OMP and FPM inputs)

SD-WAN FIB = Output Element Chain (OCE)



5-level FIB hierarchy. Why so complex?

- Here is the SDWAN FIB (OCE) chain for a routes that are learnt via OMP
- FTM is constantly updating the OCE based on various events and inputs



- **SLA NH (1):** Corresponds to set of Remote TLOC's advertising the route
- **Indirect NH:** Gives the Label to be used for the chosen Remote TLOC for a particular VRF
- **SLA NH (2):** Set of local tunnels that can be used to reach Remote TLOC
- **IPSEC/GRE NH:** Provides Tunnel Encapsulation and connected NH for underlay routing

How to check OCE for a specific prefix (the hard way)?

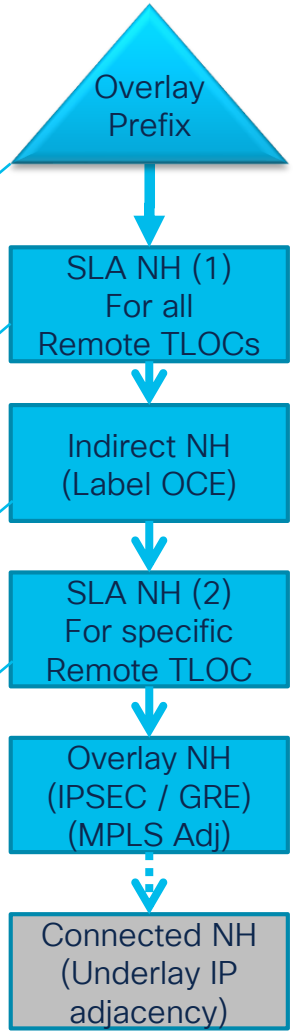
```
# show platform software ip f0 cef table * summary
…
Name              VRF id  Table id    Protocol         Prefixes    State
1                 3       3           IPv4             17          hw: 0x55ee0d19ab30 (created)
```

```
# show platform software ip f0 cef table index 3 prefix 10.10.101.0/24 detail
…
10.10.101.0/24 -> OBJ_SDWAN_NH_SLA_CLASS (0xf80002ff), urpf: 0
```

```
# show platform software sdwan fp active next-hop sla id 0xf80002ff
…
SLA_0: num_nhops 2, Fallback_sla_flag TDL_FALSE, nhobj_type SDWAN_NH_INDIRECT
  ECMP: 0xf80002df 0xf80002df 0xf80002df 0xf80002df
…
        0xf80002ef 0xf80002ef 0xf80002ef 0xf80002df
  SLA_1: num_nhops 0, Fallback_sla_flag TDL_FALSE, nhobj_type ADJ_DROP
```

```
# show platform software sdwan fp active next-hop indirect id 0xf80002df
…
  nhobj_type: SDWAN_NH_LOCAL_SLA_CLASS, nhobj_handle: 0xf808005f
  label: 1003, dst_vpn: 1, sys-ip: 1.1.1.9, sla_class: 1
```

```
# show platform software sdwan fp active next-hop sla id 0xf808005f
…
  SLA_0: num_nhops 1, Fallback_sla_flag TDL_FALSE, nhobj_type SDWAN_NH_OVERLAY
  ECMP: 0xf800025f 0xf800025f 0xf800025f 0xf800025f
…
  SLA_1: num_nhops 0, Fallback_sla_flag TDL_FALSE, nhobj_type ADJ_DROP
```



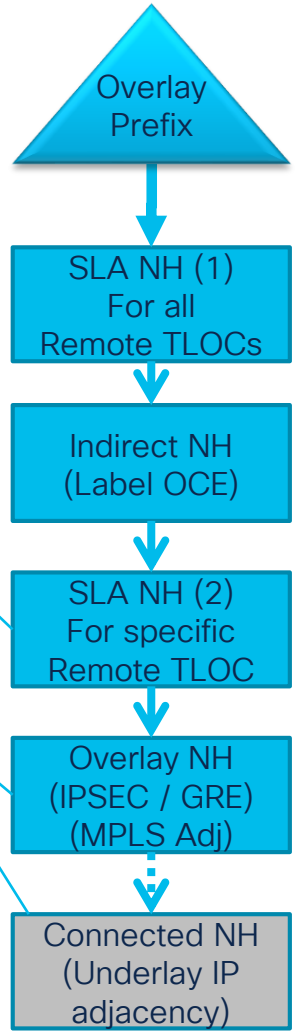
How to check OCE for a specific prefix (the hard way)? Cont.

```
# show platform software sdwan fp active next-hop sla id 0xf808005f
…
  SLA_0: num_nhops 1, Fallback_sla_flag TDL_FALSE, nhobj_type SDWAN_NH_OVERLAY
  ECMP: 0xf800025f 0xf800025f 0xf800025f 0xf800025f
…
  SLA_1: num_nhops 0, Fallback_sla_flag TDL_FALSE, nhobj_type ADJ_DROP
```

```
# show platform software sdwan fp active next-hop overlay id 0xf800025f
…
    encap type: ipsec
    src-ip: 10.1.2.7, src-port: 12386
    dst-ip: 10.1.3.8, dst-port: 12366
    local_system_ip: 1.1.1.7
    remote_system_ip: 1.1.1.9
    local_color: 5 [public-internet], remote_color: 5 [public-internet]
    wan_ifindex: 7 [GigabitEthernet1], tun_ifindex: 19 [Tunnel1]
    tun_adj_id: 0, l2_adj_id: 0x37, l2_oce_id: 0x0, l2_oce_type:
    bfd-ld: 20012, ipsec_flow_id: 603979800, session_id: 12
    nh_overlay_h: 0xf800025f
```

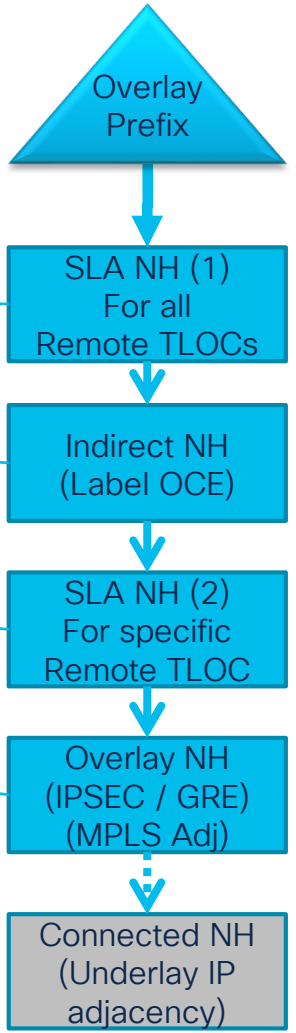
```
# show platform software sdwan session | include Session|Color|12
RemoteSysIP  Color        Proto SrcIp    SPort  DstIp    DPort  DPubIp PPort BFD-LD TUN-ID   SA-ID     WAN-Intf (nexthop)
1.1.1.9  public-internet IPSEC 10.1.2.7 12386 10.1.3.8 12366 10.1.3.8 12366 20012   12  603979800 GigabitEthernet1 (10.1.2.1)
```

```
# show platform software sdwan f0 session | i Src|20012
Src-IP    Dst-IP   Src-Port Dst-Port  Encap  SA-I      BFD-LD L2-ADJ L2-OCE  L2-OCE-TYP   AOM ID   NH Handle   Status
10.1.2.7 10.1.3.8   12386     12366   ipsec 0x24000018  20012  0x37   0x0                    617    0xf800025f  Done
```

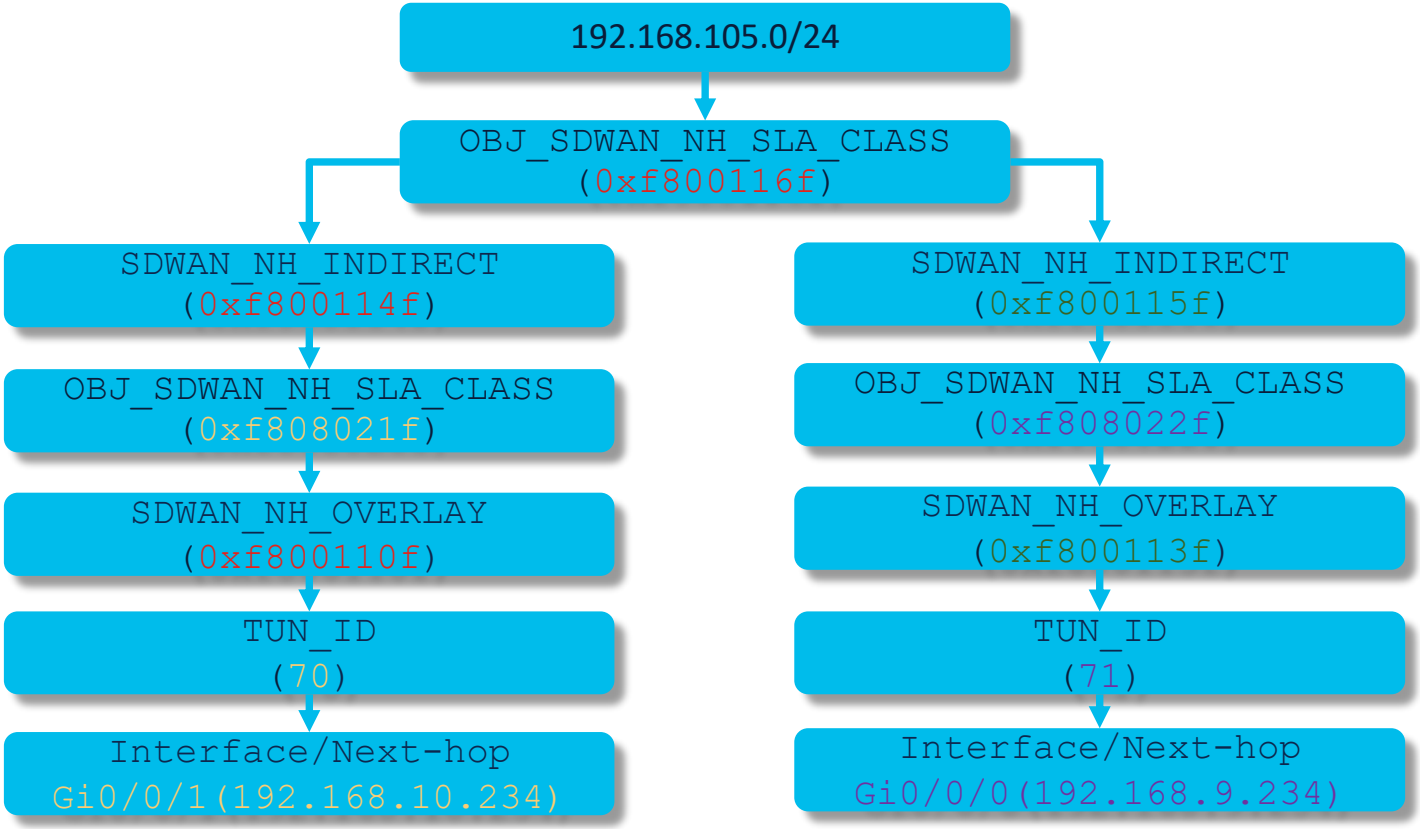


OR... you can check it the easy way ☺

```
# show platform software sdwan omp route vrf 1 10.10.101.0/24
…
10.10.101.0/24 -> OBJ_SDWAN_NH_SLA_CLASS (0xf80002ff), urpf: 0
…
=== OCE ===
OCE Type: SDWAN SLA OCE, Number of children: 1
  SLA valid_mask: 0x1
  Fallback SLA valid_mask: 0
  SLA NH OCE Handle : 0xf80002ff
  MAX index : 256
  SLA next HW OCE Ptrs: 0xd9bb2890, 0xd9bb2890, 0xd9bb2890, 0xd9bb2890
=== OCE ===
OCE Type: Label OCE, Number of children: 1
  Label flags: 0
  Num Labels: 1
  Num Bk Labels: 0
  Out Labels: 1003
  Out Backup Labels:
  Next HW OCE Ptr: 0xd9b4dc20
=== OCE ===
OCE Type: SDWAN SLA OCE, Number of children: 1
  SLA valid_mask: 0x1
  Fallback SLA valid_mask: 0
  SLA NH OCE Handle : 0xf808005f
  MAX index : 256
  SLA next HW OCE Ptrs: 0xd9bb2760, 0xd9bb2760, 0xd9bb2760, 0xd9bb2760
=== OCE ===
OCE Type: Adjacency, Number of children: 0
Adj Type: SDWAN Adjacency
Encap Len: 20
L3 MTU: 1442
Adj Flags: 0x2000
    ADJ_IS_SDWAN_IPSEC
Fixup Flags: 0x0000
Output UIDB: 65519
Interface Name: Tunnel1
Encap: 45 00 00 00 00 00 40 00 ff 89 00 00 0a 01 02 07
       0a 01 03 08
….
```

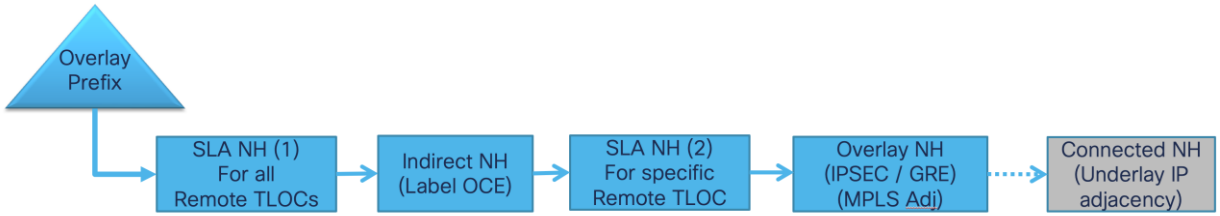


OCE is not really a linked list – it's a tree!



Let's review the packet-trace again

```
<removed>
Feature: NBAR
    Packet number in flow: N/A
    Classification state: Final
    Classification name: ping
<removed>
Feature: SDWAN App Route Policy
    VRF           : 1
    CG            : 1
    Seq           : 65535
    SLA           : __all_tunnels__ (0)
    Policy Flags : 0x2
    SLA Strict    : No
    Preferred Color : 0x0 none
<removed>
Feature: SDWAN OCE
    Hash Value  : 0xaf6f0c4e
    Encap       : ipsec
    SLA         : 0
    SDWAN VPN   : 1
    SDWAN Proto : IPV4
    Out Label   : 1001
    Local Color : biz-internet
    Remote Color: biz-internet
    FTM Tunnel ID:15
    SDWAN Session Info
      SRC IP      : 172.16.11.254
      SRC Port    : 12346
      DST IP      : 172.16.17.254
      DST Port    : 12346
      Remote System IP : 172.16.255.17
```



AAR results in SLA#0 (default)

Output Chain Element's points to a specific tunnel (hash)

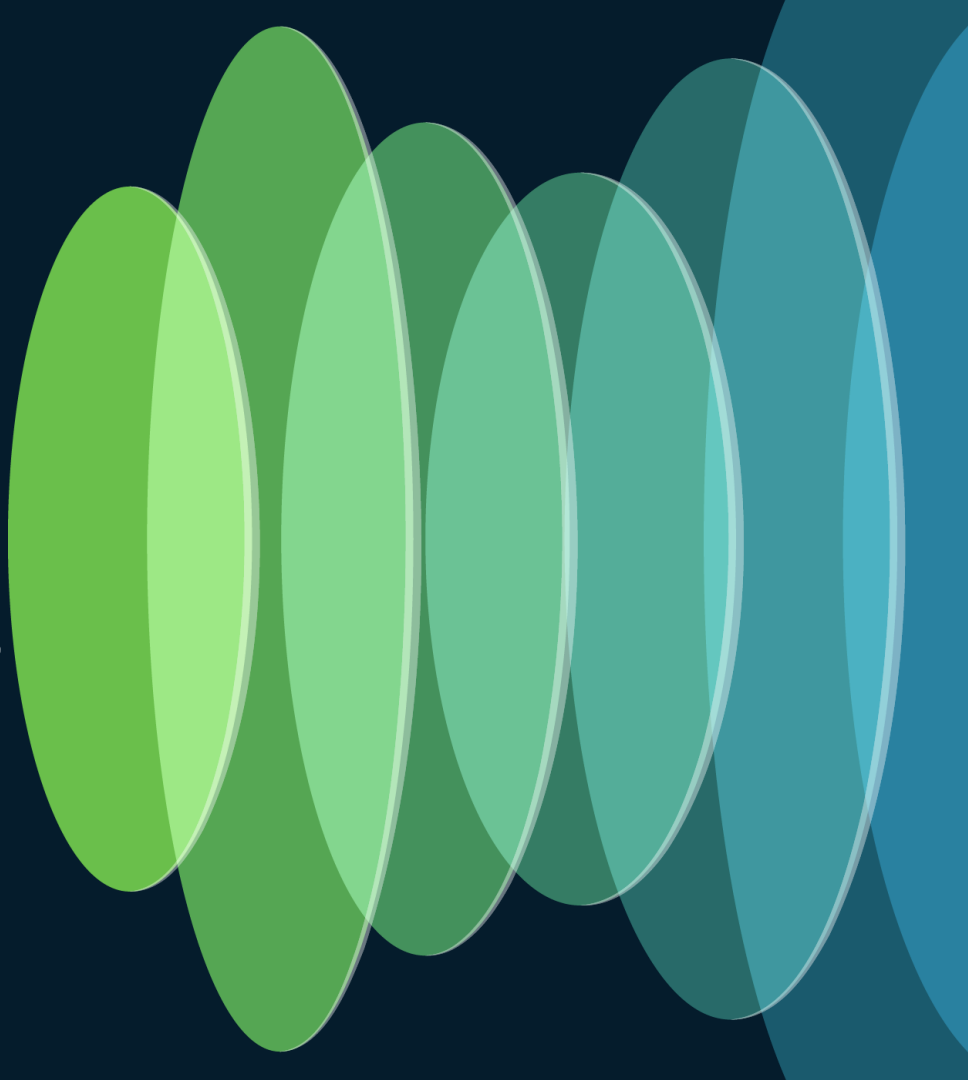
SLA#0 is used

Indirect NH (label OCE)

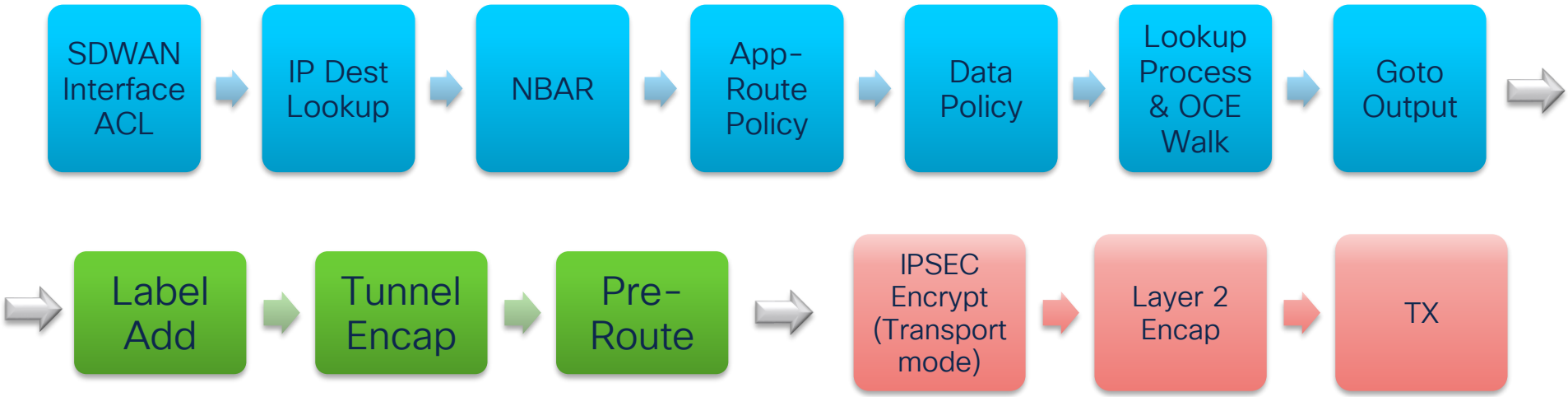
Tunnel ID

Overlay NH – points to a specific tunnel

Anything else to “de-cipher” a packet-trace output?



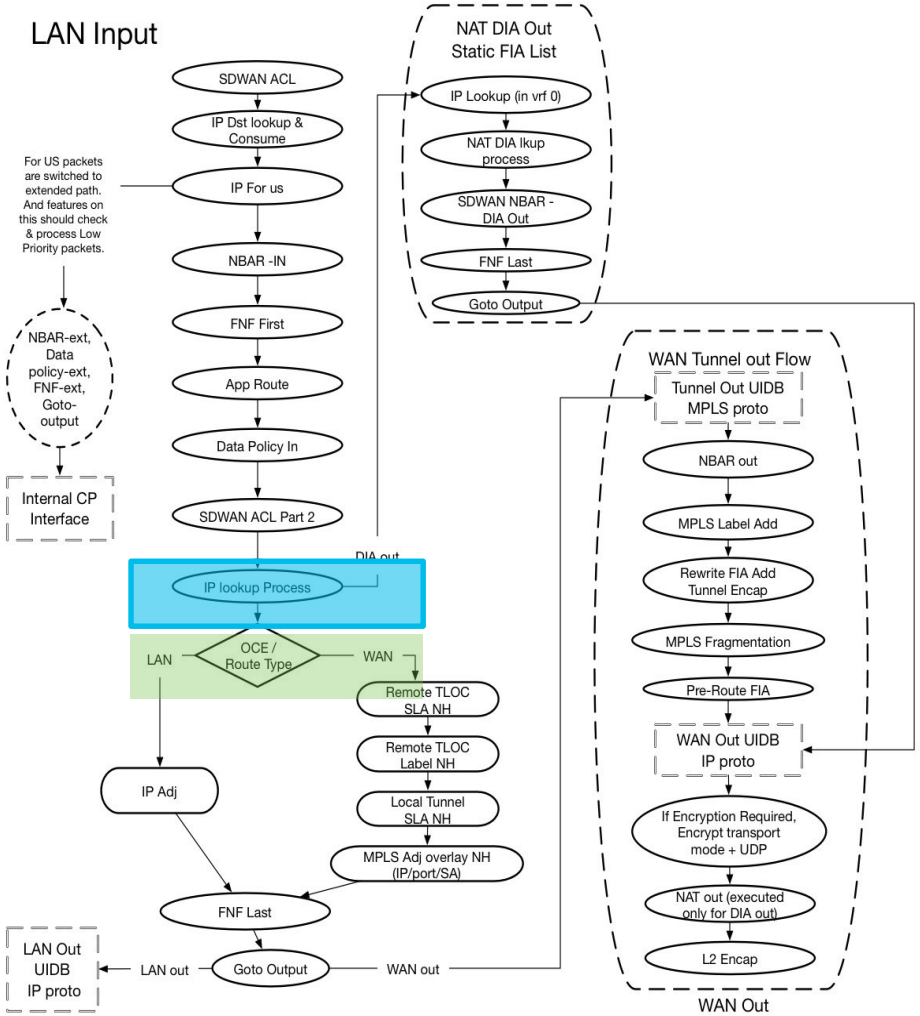
Life of a Packet (FIAs): From LAN to WAN



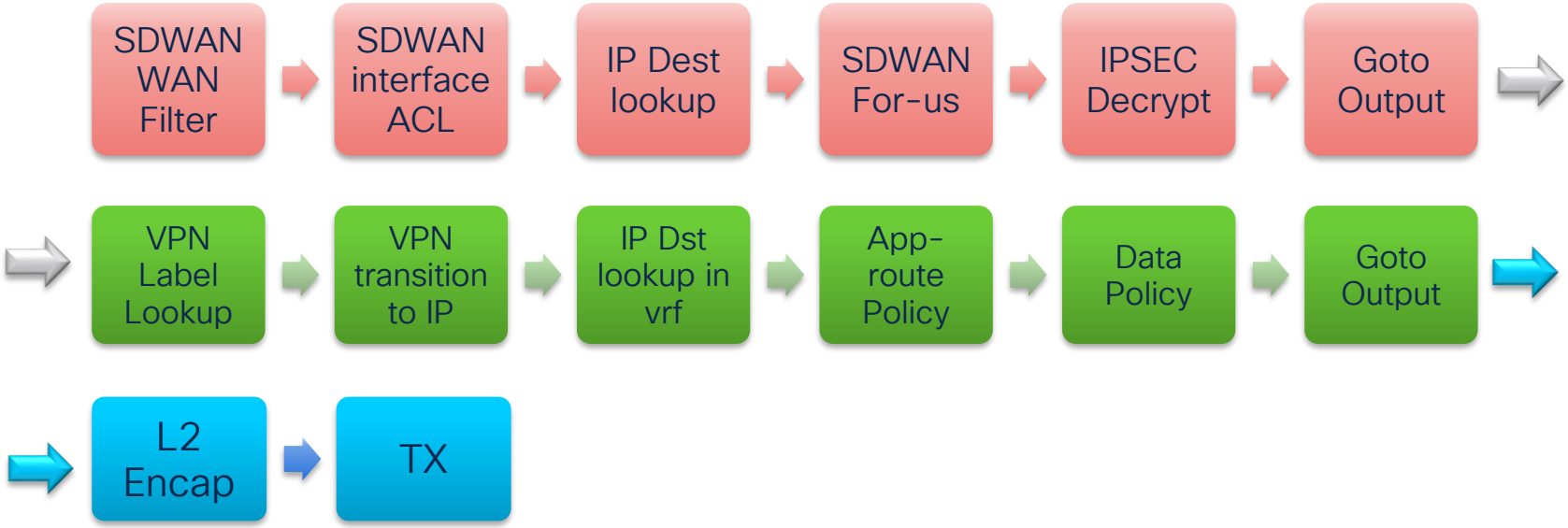
Packet processing “from service”

Key aspects:

- NAT DIA modifies the “normal” input processing bypassing OCE/CEF FIB
- “to WAN” uses OCE FIB for forwarding decision
- “to LAN” uses CEF FIB for forwarding decision



Life of a Packets (FIAs): From WAN to LAN

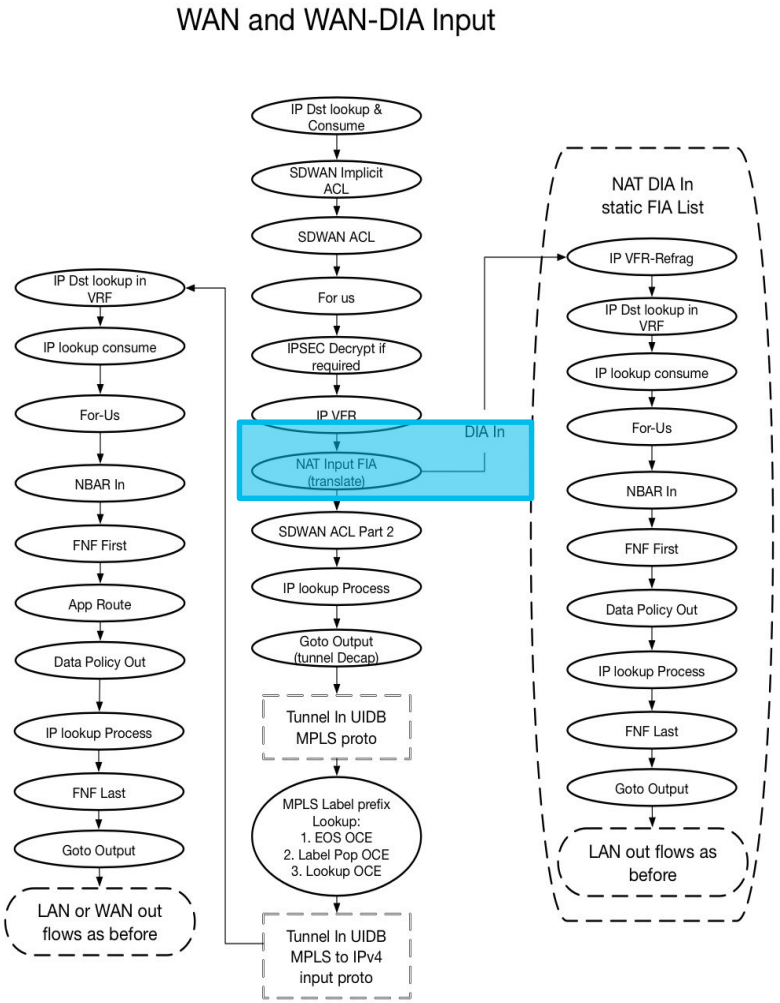


Color Coding: LAN Interface | Tunnel Interface | WAN Interface

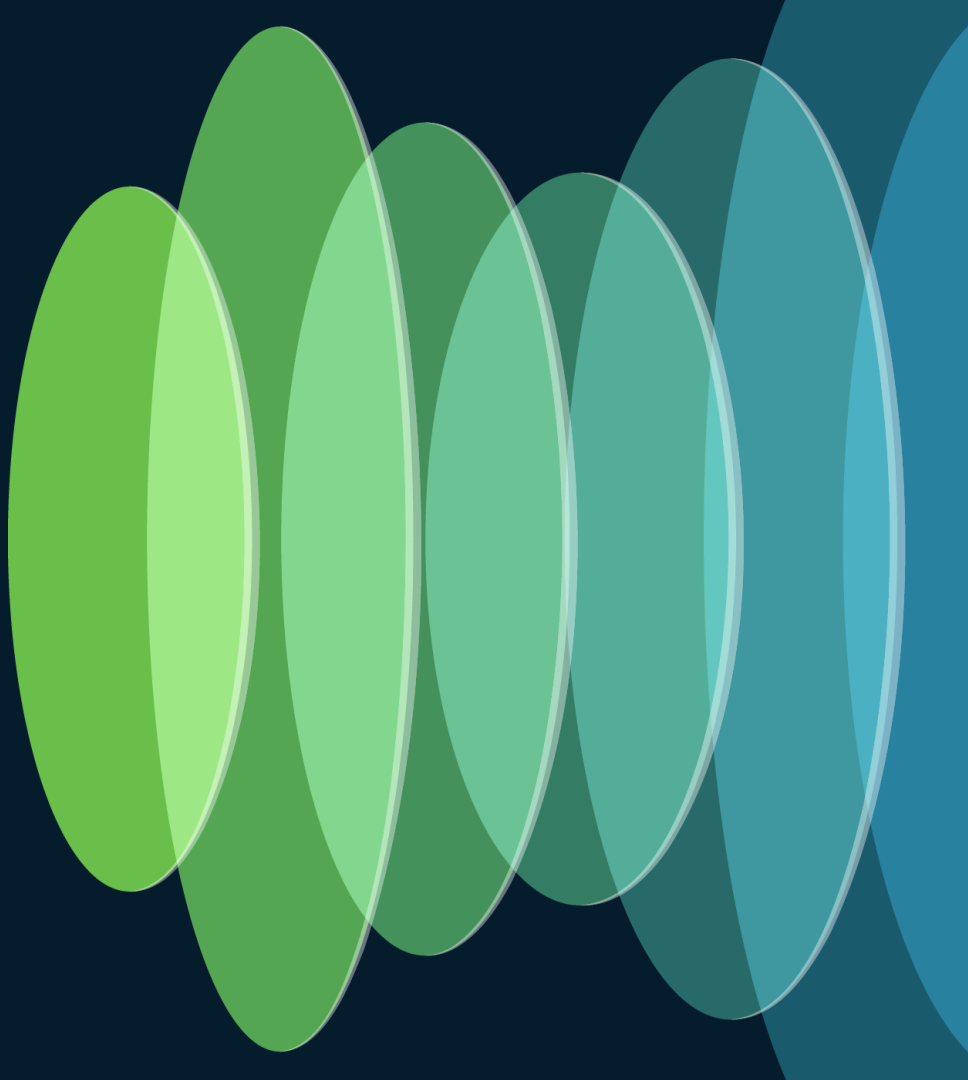
Packet processing “from tunnel”

Key aspects:

- Like “from service” the NAT DIA modifies the “normal” input processing
  - With NAT DIA in place OCE FIB lookup is skipped and the traffic forwarded according to CEF FIB handling transport VPN #0
- Unlike “from service” behavior only OCE FIB is used (overlay routes FIB)



One more thing...



How to chase intermittent (come and go) issues?

- Normally, you would just run a subset of tools we've discussed earlier in the session, collect the output and analyze later.. or send it to the TAC ☺
- The challenge with intermittent issues you need run these tools only when AND where the issue is present.

How would you address the challenge?

- Step 1 (mandatory) – identify which of the following has a clear indication of issue presence:
  - Syslog message (show logging) – use EEM with syslog tracking + Tools
  - Any other show command – use EEM with cron + TCL + Tools

Case-study: Identify the microburst traffic causing tail-drops (EPC + EEM + TCL)

- Step 2 –
  - Clear counters to reset tail-drops counters to zero
  - Run EPC for the interface with **circular** option so it will capture the traffic until stopped

```
#  clear counters
#  monitor capture TEST interface gigabitEthernet 0/0/0 out buffer circular size
100 limit pps 1000000
```

- The EPC capture should be running until the issue comes to play.

So, how can we detect the moment to stop the EPC with no logging message available?

Case-study: Identify the microburst traffic causing tail-drops (EPC + EEM + TCL)

- Step 3 – build a TCL script to check if the tail/wred-drop counter IS non zero (=increased). Save it to the router's bootflash:

```
#  set syslog [open "syslog: " w+]
puts $syslog "%EEM-7-CHECK-TAIL-DROPS: Checking for new packet
drops."

set out [exec "sh policy-map interface gigabitEthernet 0/0/0"]

foreach line [split $out "\n"] {
set drops_string  ""
set dscp_value  ""
set taildrops_packets ""
set randomdrops_packets ""
regexp -all -line {([0-9]+/[0-9]+.*[0-9]+/[0-9]+.* [0-9]+)} $line all drops_string
  if {[info exists drops_string ]} {
    if {![string equal "" $drops_string ]} {
      regexp -all -line {(af[\d][\d]|cs[\d]|ef|default)} $line dscp_value
      regexp -all -line {([0-9]+/[0-9]+)} $drops_string randomdrops_combined
taildrops_combined
…
```

```
…
      regexp -all -line {(^[0-9]+)} $taildrops_combined taildrops_packets
      regexp -all -line {(^[0-9]+)} $randomdrops_combined
randomdrops_packets
      if { $randomdrops_packets != "0" || $taildrops_packets != "0"} {
        puts $syslog "%EEM-7-CHECK-TAIL-DROPS: New packet drops
detected for DSCP: $dscp_value. Current random-drops (pkts)
$randomdrops_packets tail-drops (pkts): $taildrops_packets"
        puts $syslog "%EEM-7-CHECK-TAIL-DROPS: Stopping packet
capture. Please export pcap file manually."
        set out [exec "monitor capture stop"]
        }
    }
  }
}
close $syslog
```

Case-study: Identify the microburst traffic causing tail-drops (EPC + EEM + TCL)

- Step 4 – Configure EEM to run the TCL script every minute (minimum). Script will stop EPC once it detects a tail or wred drop.

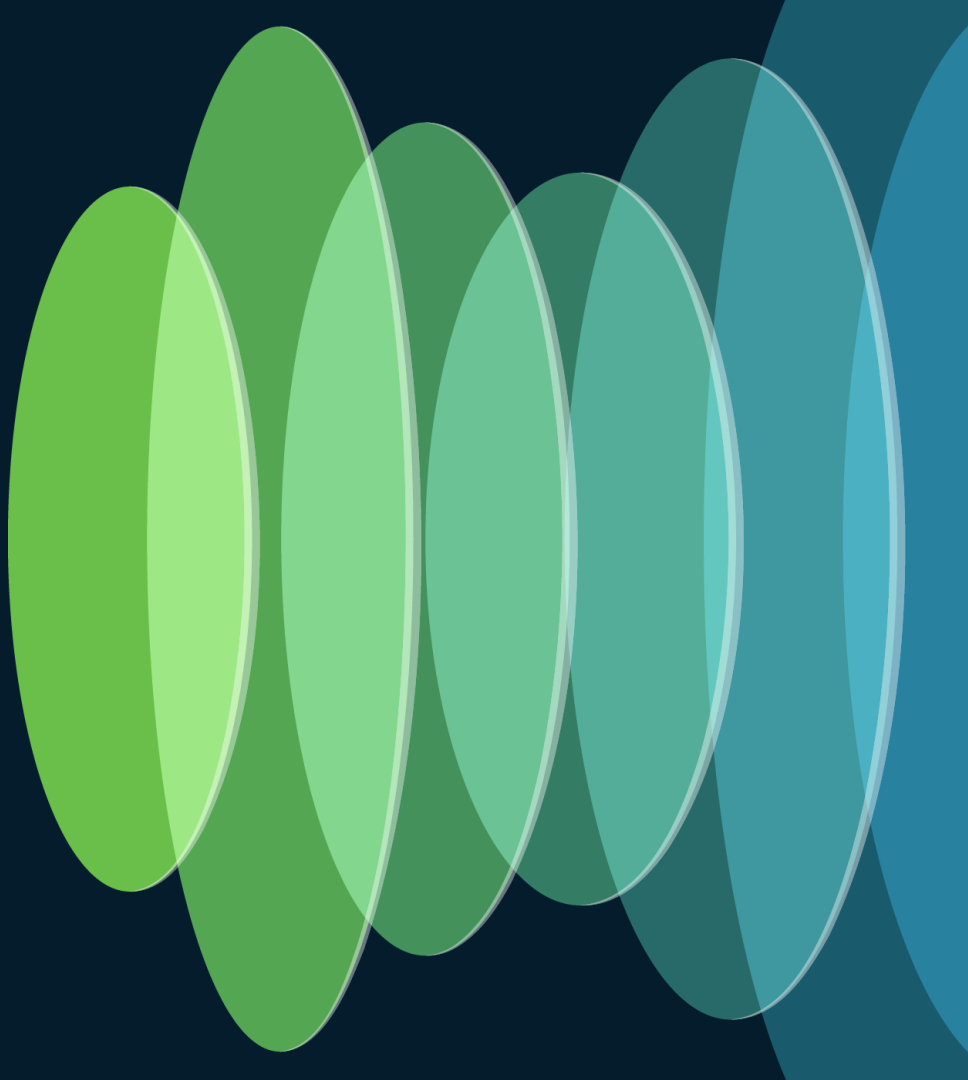
```
event manager applet taildrops_check authorization bypass
 event timer cron cron-entry "*/1 * * * *"
 action 0.2 cli command "enable"
 action 0.4 cli command "tclsh bootflash:taildrops.tcl"
…
```

- Step 5 – Bring cup of your favorite drink and check periodically for syslog message

"%EEM-7-CHECK-TAIL-DROPS: Stopping packet capture. Please export pcap file manually."

The captured pcap file will contain the bursty traffic to analyze ☺

To wrap up...



Key take-aways and Call to Action

- Know your tools and when to use (and not to use) them
- If you have to choose one single tool - use NWPI first
- bTrace – best tool to debug (and learn) SD-WAN proccess interactions and control-plane
- Packet-trace (fia-trace) is your best friend to debug and learn packet processing inside a specific IOS XE Edge Router
- Use EPC – make sure you're seeing the traffic you're expecting to see… and not seeing which you don't expect ☺

Complete Your Session Evaluations

Complete a minimum of 4 session surveys and the Overall Event Survey to be entered in a drawing to **win 1 of 5 full conference passes** to Cisco Live 2025.

Earn 100 points per survey completed and compete on the Cisco Live Challenge leaderboard.

Level up and earn **exclusive prizes!**

Complete your surveys in the **Cisco Live mobile app.**

Continue your education

- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

Contact me at: LinkedIn

Thank you