Next-gen Network Telemetry is Within Your Packets: In-band OAM

Frank Brockners

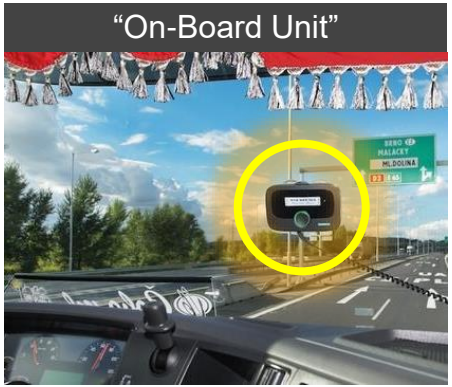
Open Networking Summit 2017

Let’s assume you’re interested in the behavior of your live user-data traffic.

What is the best source of information?

Well… probably the live user-data traffic itself.

How to send OAM information in packet networks?



In-band OAM (a.k.a. “in situ OAM”)

- OAM traffic embedded in the data traffic but not part of the payload of the packet
- OAM “effected by data traffic”
- Example: IPv4 route recording

Out-of-band OAM

- OAM traffic is sent as dedicated traffic, independent from the data traffic (“probe traffic”)
- OAM “not effected by data traffic”
- Examples: Ethernet CFM (802.1ag), Ping, Traceroute

Let’s add meta-data to all interesting live user-data traffic.

Remember RFC 791?

INTERNET PROTOCOL

DARPA INTERNET PROGRAM

PROTOCOL SPECIFICATION

September 1981

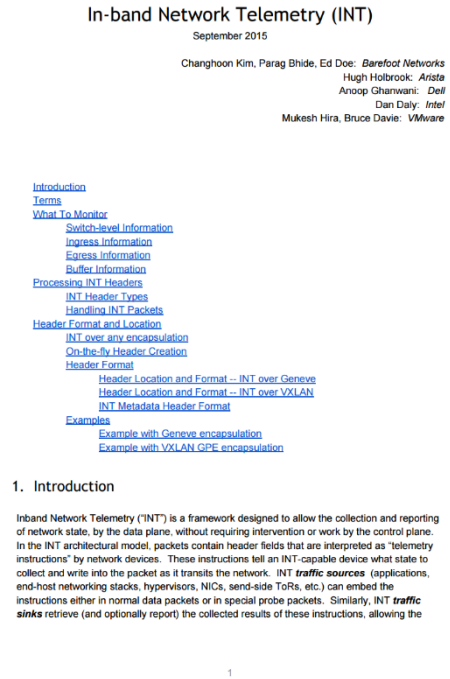
prepared for

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

The following internet options are defined:

CLASS	NUMBER	LENGTH	DESCRIPTION
0	0	-	End of Option list. This option occupies only 1 octet; it has no length octet.
0	1	-	No Operation. This option occupies only 1 octet; it has no length octet.
0	2	11	Security. Used to carry Security, Compartmentation, User Group (TCC), and Handling Restriction Codes compatible with DOD requirements.
0	3	var.	Loose Source Routing. Used to route the internet datagram based on information supplied by the source.
0	9	var.	Strict Source Routing. Used to route the internet datagram based on information supplied by the source.
0	7	var.	Record Route. Used to trace the route an internet datagram takes.
0	8	4	Stream ID. Used to carry the stream identifier.
2	4	var.	Internet Timestamp.

More recently: In-band Network Telemetry for P4



In-band Network Telemetry (INT)

September 2015

Changhoon Kim, Parag Bhide, Ed Doe: *Barefoot Networks*
Hugh Holbrook: *Arista*
Anoop Ghanwani: *Dell*
Dan Daly: *Intel*
Mukesh Hira, Bruce Davie: *VMware*

1. Introduction

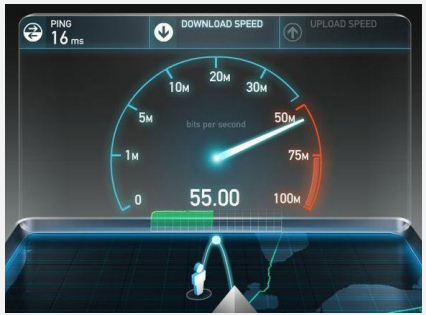
Inband Network Telemetry ("INT") is a framework designed to allow the collection and reporting of network state, by the data plane, without requiring intervention or work by the control plane. In the INT architectural model, packets contain header fields that are interpreted as "telemetry instructions" by network devices. These instructions tell an INT-capable device what state to collect and write into the packet as it transits the network. INT ***traffic sources*** (applications, end-host networking stacks, hypervisors, NICs, send-side ToRs, etc.) can embed the instructions either in normal data packets or in special probe packets. Similarly, INT ***traffic sinks*** retrieve (and optionally report) the collected results of these instructions, allowing the

1

http://p4.org/wp-content/uploads/fixed/INT/INT-current-spec.pdf

In-Band OAM - Motivation

- Service/Quality Assurance
  - Prove traffic SLAs, as opposed to probe-traffic SLAs; Overlay/Underlay
  - Service/Path Verification (Proof of Transit) – prove that traffic follows a pre-defined path
- Micro-Service/NFV deployments
  - Smart service selection based on network criteria (intelligent Anycast server/service selection)
- Operations Support
  - Network Fault Detection and Fault Isolation through efficient network probing
  - Path Tracing – debug ECMP, brown-outs, network delays
  - Derive Traffic Matrix
  - Custom/Service Level Telemetry



"Most large ISP's prioritize Speedtest traffic and I would even go as far to say they probably route it faster as well to keep ping times low."

Source: https://www.reddit.com/r/AskTechnology/comments/2i1nxc/can_i_trust_my_speedtestnet_results_when_my_isp/

Agenda

Why In-Band OAM? – Use-Case Examples

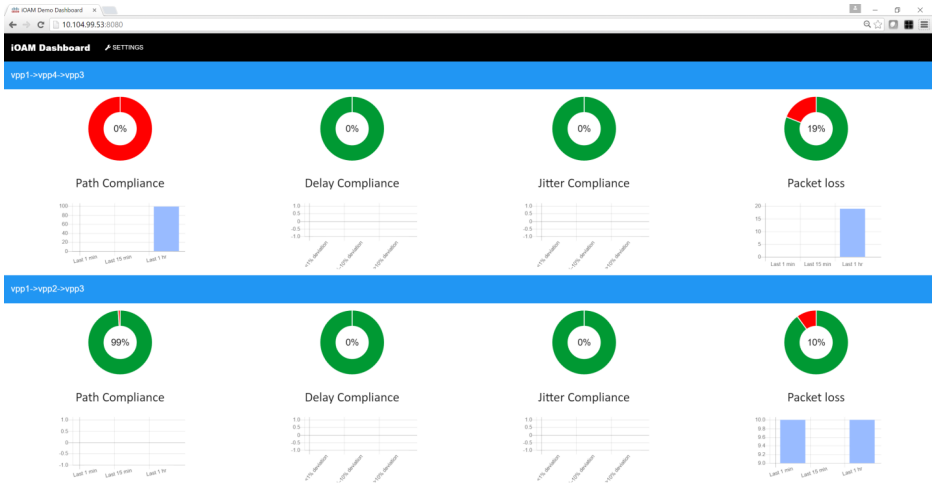
In-Band OAM – Technology & Evolving Standards

A Peek at the Implementation – IOS and OpenSource

In-Band OAM Solution Ecosystem & Advanced Use-Cases

Example: SLA Verification

- Overlay Networks, Service chains / path have associated SLA
  - KPIs for overlay networks and service chains are end-to-end delay, jitter and packet drops
- For Overlay Networks and NFV service chaining SLA-check is still very nacent
  - Either no end-to-end service chain SLA verification or SLA verification is based on probes (IPSLA) which can be easily cheated upon
  - Common interpretation is service-chain: Health = VM health
- Approach: In-band monitoring of customer traffic



Example: Proof of Transit

Consider TE, Service Chaining, Policy Based Routing, etc...

Example: Proof of Transit

Consider TE, Service Chaining, Policy Based Routing, etc...

“How do you **prove** that traffic follows the suggested path?”

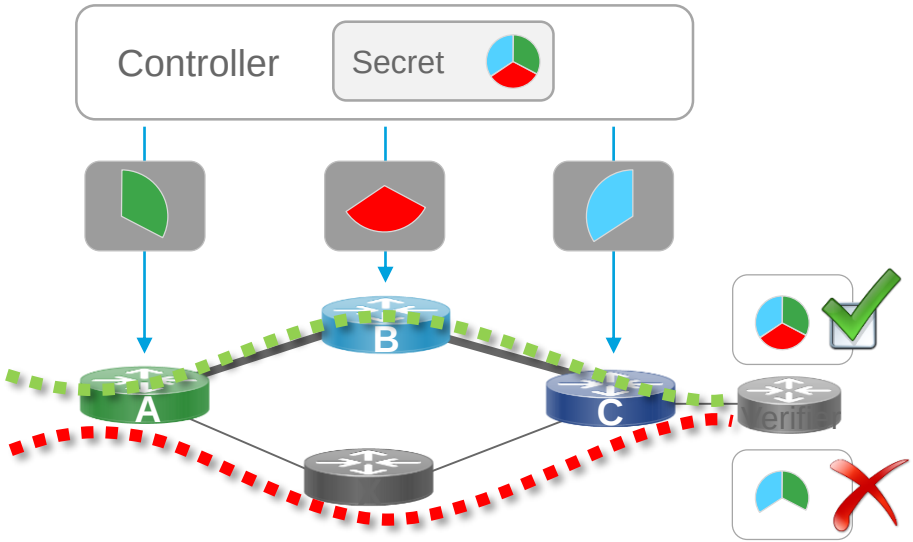
“… Willis says the first issue is connecting VNFs to the infrastructure. OpenStack does this in a sequential manner, with the sequence serially numbered in the VNF, but the difficulty comes when trying to verify that the LAN has been connected to the correct LAN port, the WAN has been connected to the correct WAN port and so on. "*If we get this wrong for a firewall function it could be the end of a CIO's career*," says Willis.”

LightReading Networking the Communications Industry

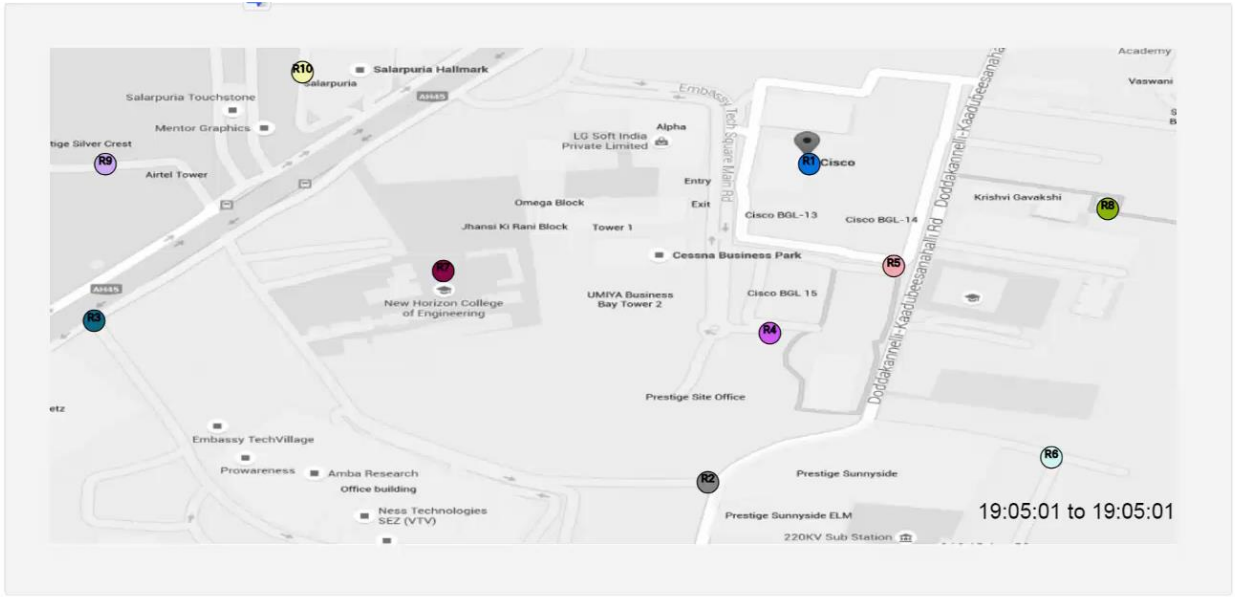
http://www.lightreading.com/nfv/nfv-specs-open-source/bt-threatens-to-ditch-openstack/d/d-id/718735
October 14, 2015
Light reading citing Peter Willis, Chief researcher for data networks, British Telecom

Example: Proof-of-Transit

- Question: With TE/SFC/PBR: How do you **prove** that traffic follows the correct path?
- Approach: OAM Meta-data added to all user traffic
  - Based on “Share of a secret”
  - Provisioned by controller over secure channel
  - Updated at every service hop
- Verifier checks whether collected meta-data allows retrieval of secret
  - ➡ Path verified



Example – Generic customer meta-data: Geo-Location within your packets



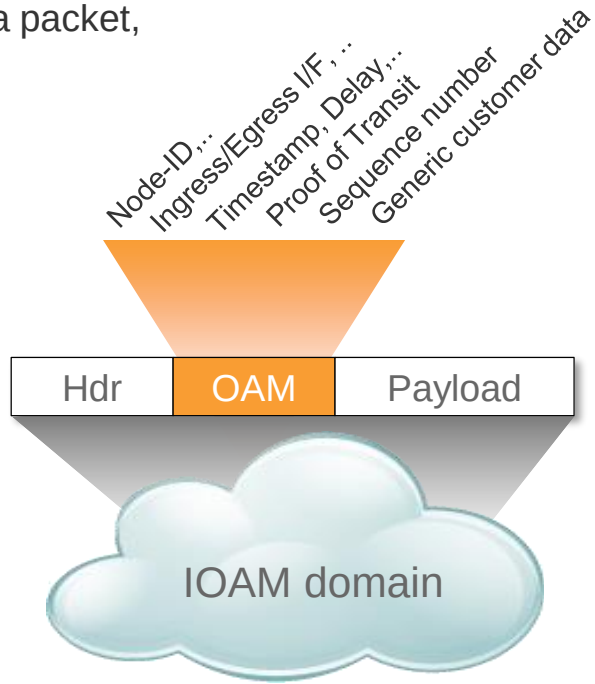
What if you could collect operational meta-data within your traffic?

Example use-cases...	Meta-data required...
• Path Tracing for ECMP networks	• Node-ID, ingress i/f, egress i/f
• Service/Path Verification	• Proof of Transit (random, cumulative)
• SLA proof: Delay, Jitter, Loss	• Sequence numbers, Timestamps
• Custom data: Geo-Location,..	• Custom meta-data
• Smart Service Selection	• Timestamps, custom meta-data
• Efficient Failure Detection/Isolation	• Node-ID, Timestamps

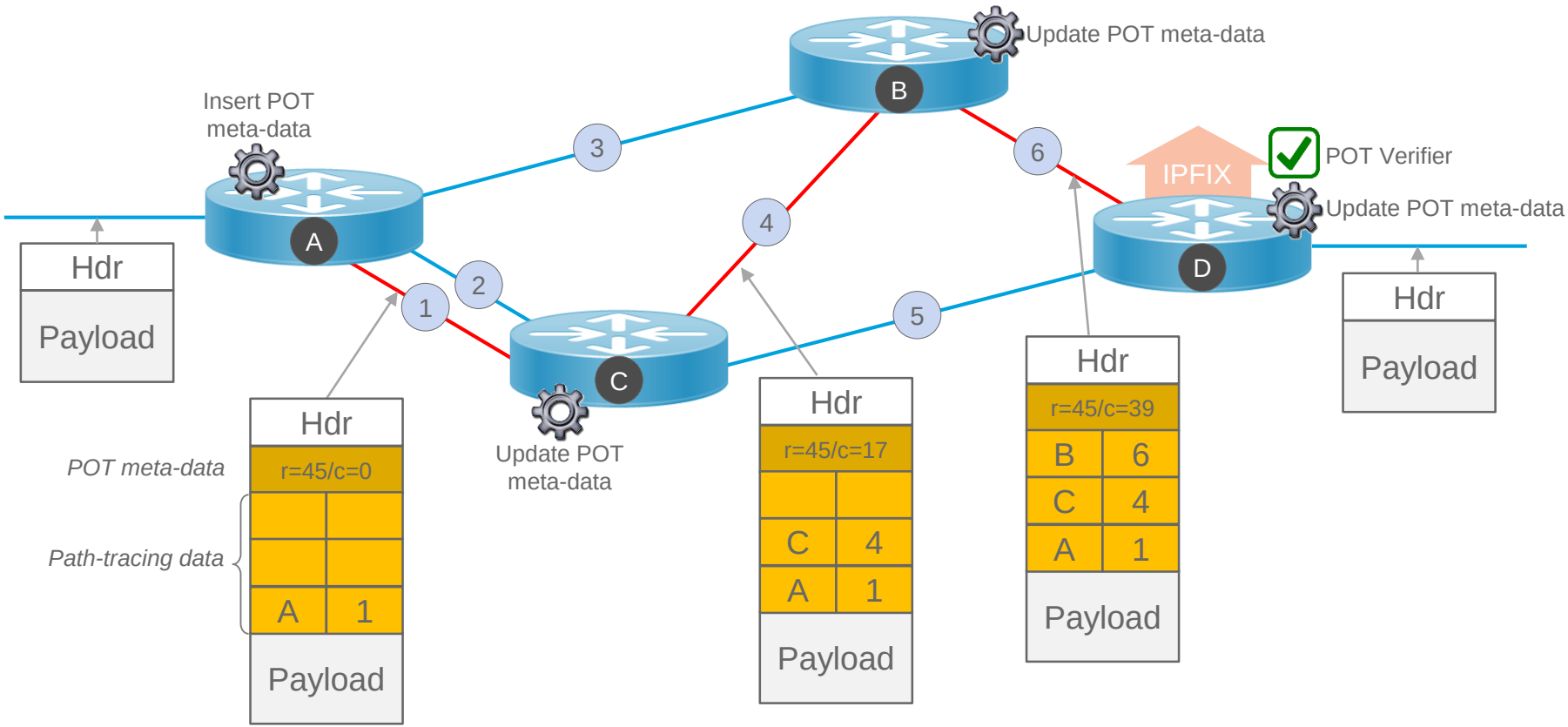
In-Band OAM (IOAM) in a Nutshell

- Gather telemetry and OAM information along the path within the data packet, (hence “in-band OAM”) as part of an existing/additional header
  - No extra probe-traffic (as with ping, trace, IPSLA)
- Transport options
  - IPv6: Native v6 HbyH extension header or double-encap
  - VXLAN-GPE: Embedded telemetry protocol header
  - NSH: Type-2 Meta-Data

  ... additional encapsulations being considered/WIP (incl. SRv6, GRE, MPLS)
- Deployment
  - Domain-ingress, domain-egress, and select devices within a domaininsert/remove/update the extension header
  - Information export via IPFIX/Flexible-Netflow/publish into Kafka
  - Fast-path implementation

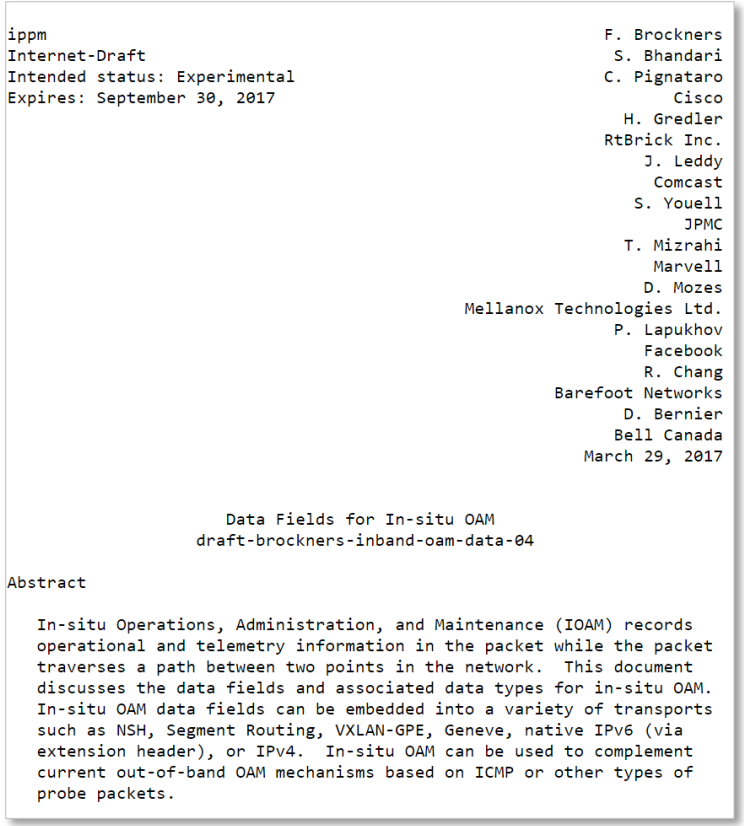


* FD.io/VPP open source reference implementation available



Latest IETF Drafts

- In-band OAM Authors: Cisco, Comcast, Facebook, JPMC, Bell Canada, Mellanox, Marvell, Barefoot, rtBrick
  - In-band OAM requirements: https://tools.ietf.org/html/draft-brockners-inband-oam-requirements-03.txt
  - In-band OAM data types: https://tools.ietf.org/html/draft-brockners-inband-oam-data-04.txt
  - In-band OAM transport: https://tools.ietf.org/html/draft-brockners-inband-oam-transport-03.txt
  - Proof-of-transit: https://tools.ietf.org/html/draft-brockners-proof-of-transit-03.txt
- In-band OAM manageability – YANG models and methods defined in IETF LIME WG
  - draft-ietf-lime-yang-connectionless-oam-03
  - draft-ietf-lime-yang-connectionless-oam-methods-00



ippm
Internet-Draft
Intended status: Experimental
Expires: September 30, 2017

F. Brockners
S. Bhandari
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JPMC
T. Mizrahi
Marvell
D. Mozes
Mellanox Technologies Ltd.
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
D. Bernier
Bell Canada
March 29, 2017

Data Fields for In-situ OAM
draft-brockners-inband-oam-data-04

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document discusses the data fields and associated data types for in-situ OAM. In-situ OAM data fields can be embedded into a variety of transports such as NSH, Segment Routing, VXLAN-GPE, Geneve, native IPv6 (via extension header), or IPv4. In-situ OAM can be used to complement current out-of-band OAM mechanisms based on ICMP or other types of probe packets.

In-situ OAM Data Fields Overview

- Per node scope
  - Hop-by-Hop information processing
    - Hop Limit
    - Node_ID (long/short)
    - Ingress Interface ID (long/short)
    - Egress Interface ID (long/short)
    - Timestamp
      - Wall clock (seconds, nanoseconds)
      - Transit delay
    - Queue length
    - Opaque data
    - Application Data (long/short)

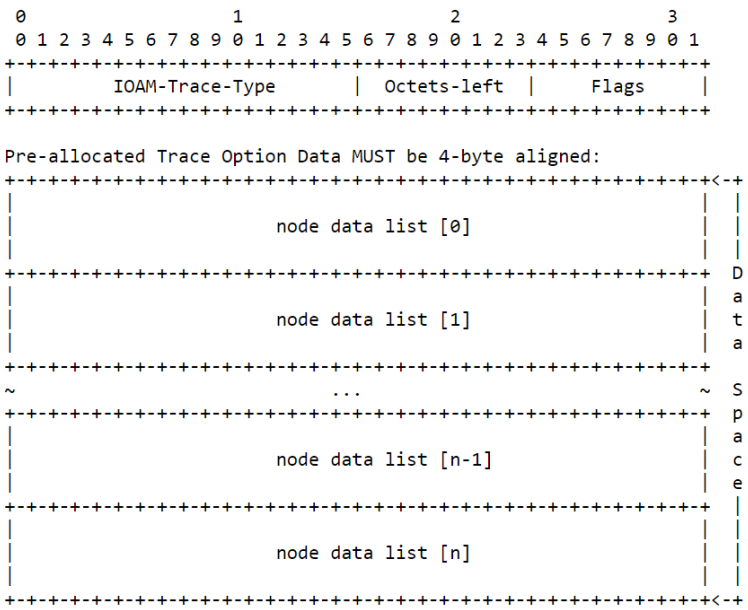
Two transport options:
- Pre-allocated array (SW friendly)
- Incrementally grown array (HW friendly)

- Set of nodes scope
  - Hop-by-Hop information processing
    - Service Chain Validation (Random, Cumulative)

- Edge to Edge scope
  - Edge-to-Edge information processing
    - Sequence Number

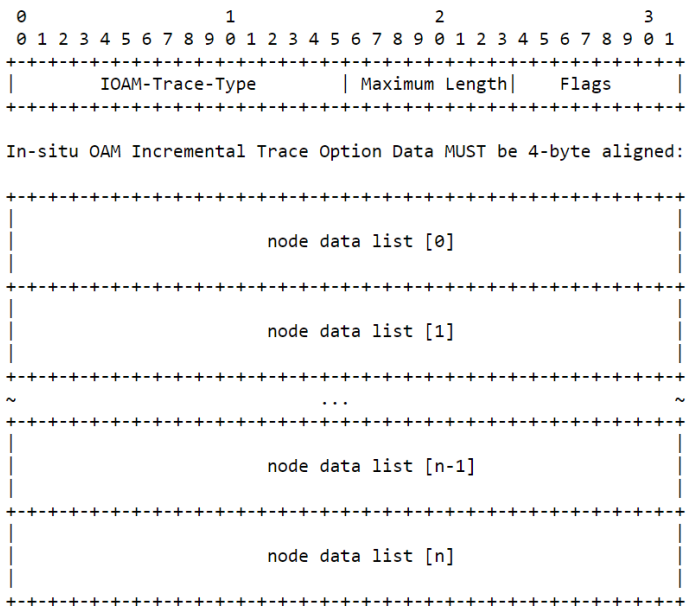
Pre-Allocated & Incremental Trace Option Header (per-node info)

Pre-allocated Trace Option header:



Software friendly

In-situ OAM Incremental Trace Option Header:



Hardware friendly

Trace Types

Bit 0 When set indicates presence of Hop_Lim and node_id in the node data.

Bit 1 When set indicates presence of ingress_if_id and egress_if_id in the node data.

Bit 2 When set indicates presence of timestamp seconds in the node data.

Bit 3 When set indicates presence of timestamp nanoseconds in the node data.

Bit 4 When set indicates presence of transit delay in the node data.

Bit 5 When set indicates presence of app_data in the node data.

Bit 6 When set indicates presence of queue depth in the node data.

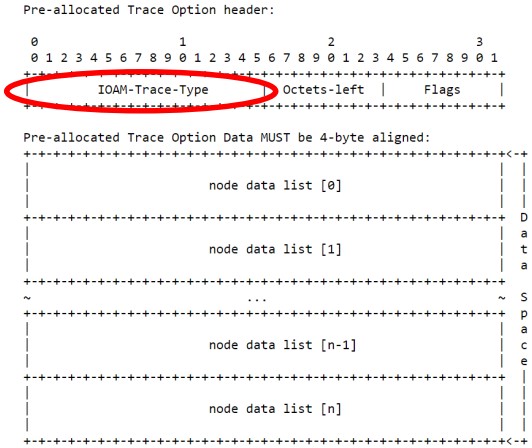
Bit 7 When set indicates presence of variable length Opaque State Snapshot field.

Bit 8 When set indicates presence of Hop_Lim and node_id wide in the node data.

Bit 9 When set indicates presence of ingress_if_id and egress_if_id wide in the node data.

Bit 10 When set indicates presence of app_data wide in the node data.

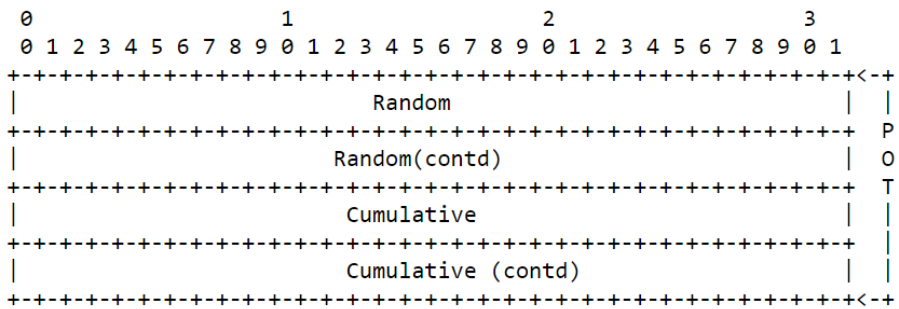
Bit 11-15 Undefined in this draft.



IOAM Data which in only updated by selected nodes: Proof-of-Transit

In-situ OAM Proof of Transit Option Header:

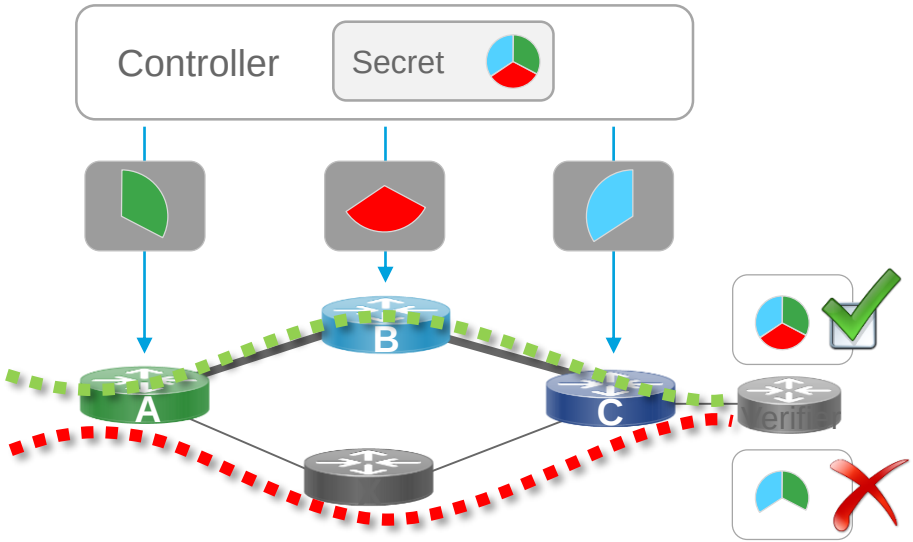
In-situ OAM Proof of Transit Option Data MUST be 4-byte aligned:



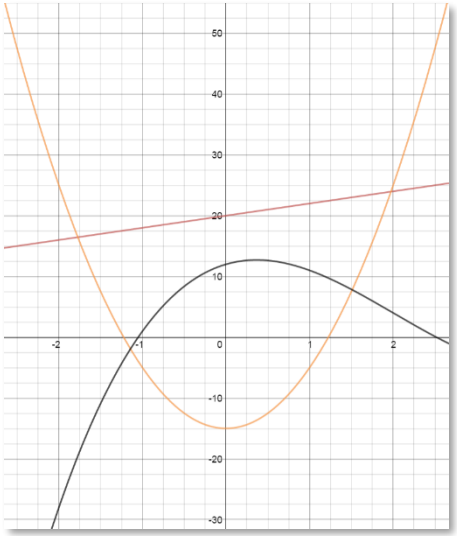
Proof of Transit:
Ensuring Path and/or Service Chain Integrity

- Meta-data added to all user traffic
  - Based on “Share of a secret”
  - Provisioned by controller over secure channel
  - Updated at every service hop
- Verifier checks whether collected meta-data allows retrieval of secret

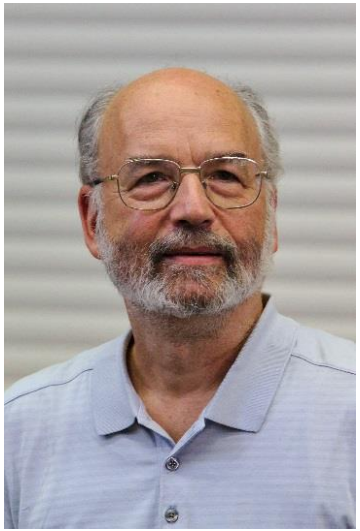
  ➡ Path verified



Solution Approach: Leveraging Shamir's Secret Sharing
Polynomials 101



- Parabola: Min 3 points
  $f2(x) = 10x^2 - 15$
- Line: Min 2 points
  $f1(x) = 2x + 20$
- Cubic function: Min 4 points
  $f3(x) = x^3 - 6x^2 + 4x - 12$

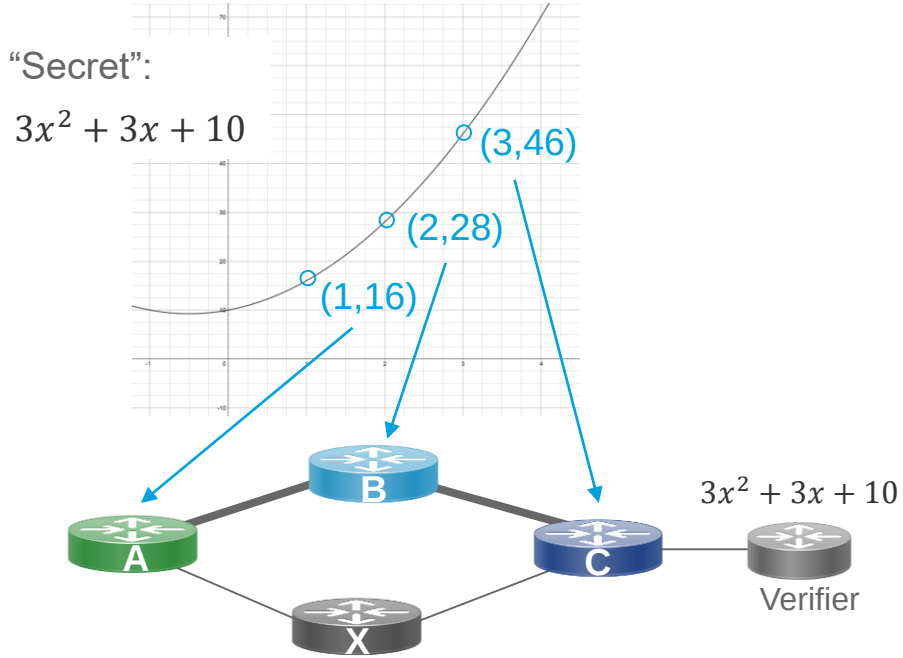


Adi Shamir

General: It takes k+1 points to defines a polynomial of degree k.

Solution Approach: Leverage Shamir's Secret Sharing
"A polynomial as secret"

- Each service is given a point on the curve
- When the packet travels through each service it collects these points
- A verifier can reconstruct the curve using the collected points
- Operations done over a finite field (mod prime) to protect against differential analysis



Operationalizing the Solution

- Leverage two polynomials:
  - POLY-1 secret, constant: Each hop gets a point on POLY-1
    Only the verifier knows POLY-1
  - POLY-2 public, random and per packet.
    Each hop generates a point on POLY-2 each time a packet crosses it.
- Each service function calculates (Point on POLY-1 + Point on POLY-2) to get (Point on POLY-3) and passes it to verifier by adding it to each packet.
- The verifier constructs POLY-3 from the points given by all the services and cross checks whether POLY-3 = POLY-1 + POLY-2
- Computationally efficient:
  2 additons, 1 multiplication, mod prime per hop

POLY-1
Secret – Constant

+

POLY-2
Public – Per Packet

=

POLY-3
Secret – Per Packet

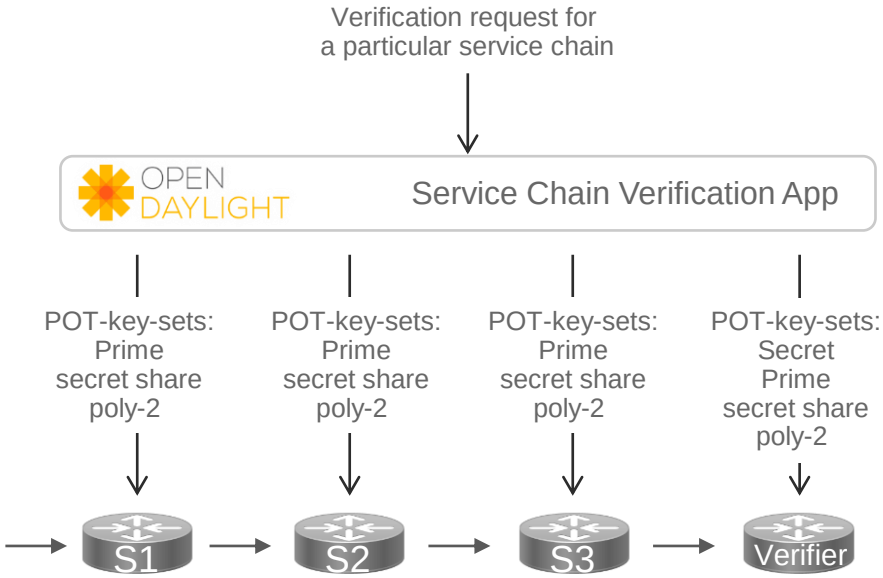
Meta Data for Service/Path Verification

- Verification secret is the independent coefficient of POLY-1
  - Computation/retrieval through a cumulative computation at every hop (“cumulative”)
- For POLY-2 the independent coefficient is carried within the packet (typically a combination of timestamp and random number)
  - n bits can service a maximum of 2^n packets
- Verification secret and POLY-2 coefficient (“random”) are of the same size
  - Secret size is bound by prime number

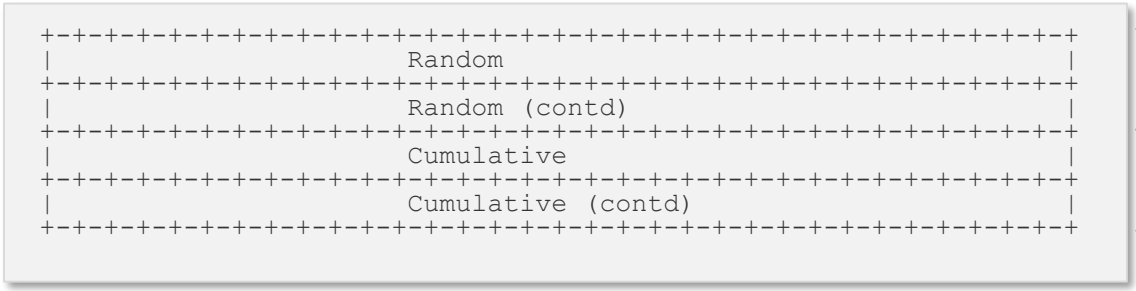
Transfer Rate	RND/ Secret Size	Max # of packets (assuming 64 byte packets)	Time that “random” lasts at maximum
1 Gbps	**64**	$2^{64} \approx 2 * 10^{19}$	approx. 10^{13} seconds, approx. 310000 years
10 Gbps	**64**	$2^{64} \approx 2 * 10^{19}$	approx. 10^{12} seconds, approx. 31000 years
100 Gbps	**64**	$2^{64} \approx 2 * 10^{19}$	approx. 10^{11} seconds, approx. 3100 years
10 Gbps	56	$2^{56} \approx 7 * 10^{16}$	approx. 10^{9} seconds, approx. 120 years
10 Gbps	48	$2^{48} \approx 2 * 10^{14}$	approx. 10^{7} seconds, approx. 5.5 months
10 Gbps	40	$2^{40} \approx 1 * 10^{12}$	approx. $5 * 10^{4}$ seconds, approx. 15 hours
1 Gbps	32	$2^{32} \approx 4 * 10^{9}$	2200 seconds, 36 minutes
10 Gbps	32	$2^{32} \approx 4 * 10^{9}$	220 seconds, 3.5 minutes
100 GBps	32	$2^{32} \approx 4 * 10^{9}$	22 seconds

Meta-Data Provisioning

- Meta-Data for Service Chain Verification (POT) provisioned through a controller (OpenDaylight App)
- Netconf/YANG based protocol
- Provisioned information from Controller to Service Function / Verifier
  - Service-Chain-Identifier
    (to be mapped to service chaining technology specific identifier by network element)
  - Service count (number of services in the chain)
  - 2 x POT-key-set (even and odd set)
    - Secret (in case of communication to the verifier)
    - Share of a secret, service index
    - 2nd polynomial coefficients
    - Prime number



Proof of Transit: Meta-Data Transport Options



- 16* Bytes of Meta-Data for POT
  - Random – Unique random number (e.g. Timestamp or combination of Timestamp and Sequence number)
  - Cumulative (algorithm dependent)

- Transport options for different protocols
  - Segment Routing: New TLV in SRH header
  - Network Service Header: Type-2 Meta-Data
  - In-band OAM for IPv6:
    Proof-of-transit extension header
  - VXLAN-GPE
    Proof-of-transit embedded-telemetry header
  - ... more to be added (incl. IPv4)

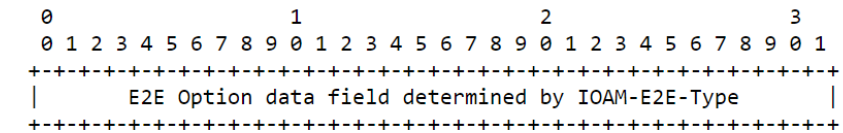
*Note: Smaller numbers are feasible, but require a more frequent renewal of the polynomials/secrets.

IOAM Data with E2E focus: Sequence Numbers

In-situ OAM Edge-to-Edge Option:

In-situ OAM Edge-to-Edge Option Header:

In-situ OAM Edge-to-Edge Option Data MUST be 4-byte aligned:



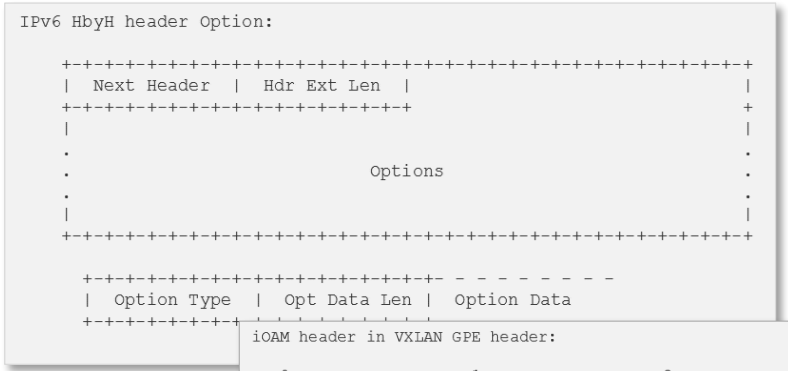
IOAM-E2E-Type: 8-bit identifier of a particular in situ OAM E2E variant.

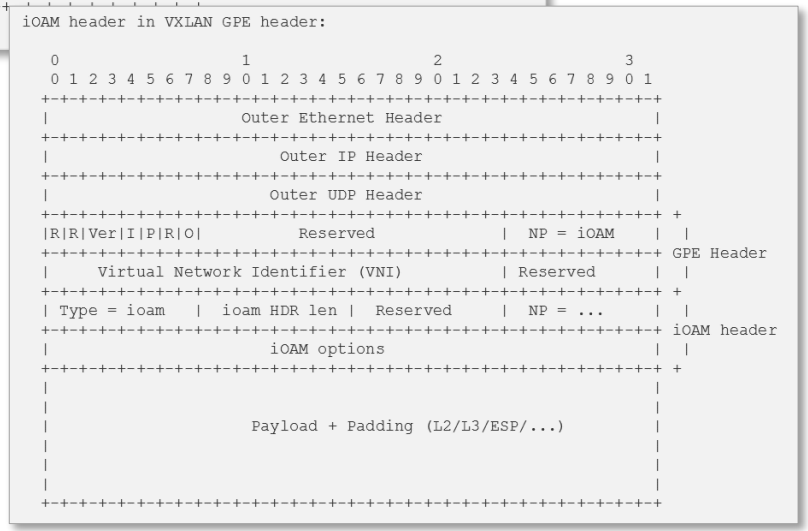
0: E2E option data is a 64-bit sequence number added to a specific tube which is used to identify packet loss and reordering for that tube.

Transport Options – IPv6, VXLAN-GPE, SRv6, NSH...

- IPv6
  - Native v6 HbyH extension header
  - Double-encapsulation: src=encap-node, dst=original
- VXLAN-GPE: Embedded telemetry protocol header;
- NSH: Type-2 Meta-Data
- SRv6: In-band OAM TLV in v6 SR-header SRH

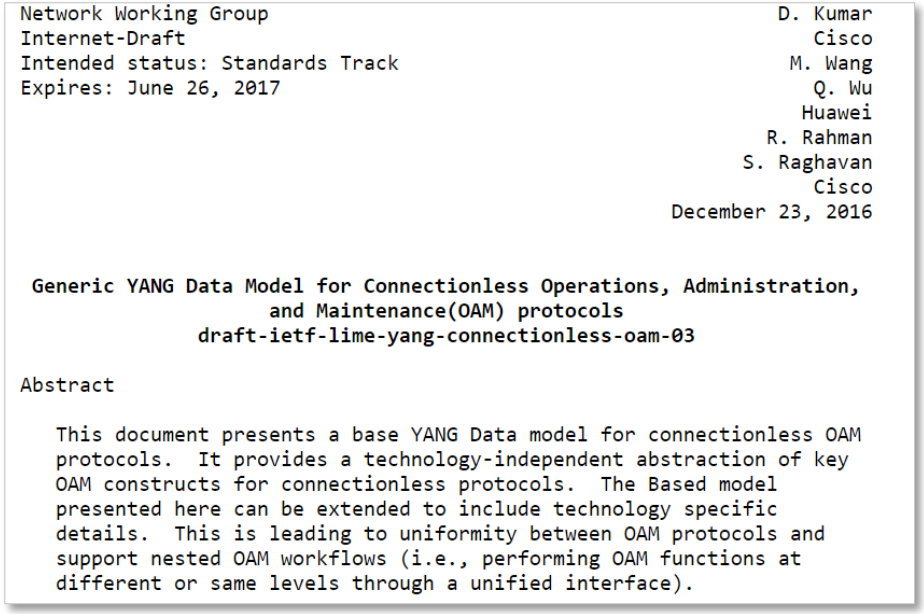
... additional encaps are in the works, incl. GRE,...





Configuration And Data Export

- In-band OAM Configuration and Data Exports included in IETF LIME WG Connectionless OAM YANG models
- draft-ietf-lime-yang-connectionless-oam-03
- draft-ietf-lime-yang-connectionless-oam-methods-00



Network Working Group — D. Kumar, Cisco
Internet-Draft
Intended status: Standards Track — M. Wang, Q. Wu, Huawei
Expires: June 26, 2017 — R. Rahman, S. Raghavan, Cisco
December 23, 2016

Generic YANG Data Model for Connectionless Operations, Administration, and Maintenance(OAM) protocols
draft-ietf-lime-yang-connectionless-oam-03

Abstract

This document presents a base YANG Data model for connectionless OAM protocols. It provides a technology-independent abstraction of key OAM constructs for connectionless protocols. The Based model presented here can be extended to include technology specific details. This is leading to uniformity between OAM protocols and support nested OAM workflows (i.e., performing OAM functions at different or same levels through a unified interface).

In-Band OAM Implementation

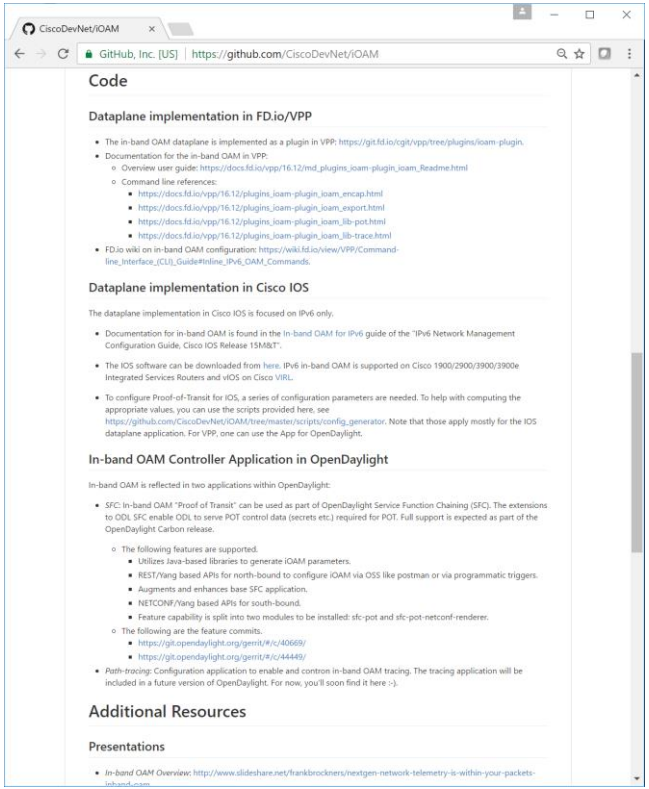
Implementation Approach

- In-band OAM mechanisms for IP (route recording) have been proposed in the past, but were always received with skepticism
  - Performance concerns
  - Footprint for a new technology (incl. new packet header) vs. installed base:
    “Is there a sufficiently large greenfield network to allow for the change?”

- Let’s stop arguing and go try…

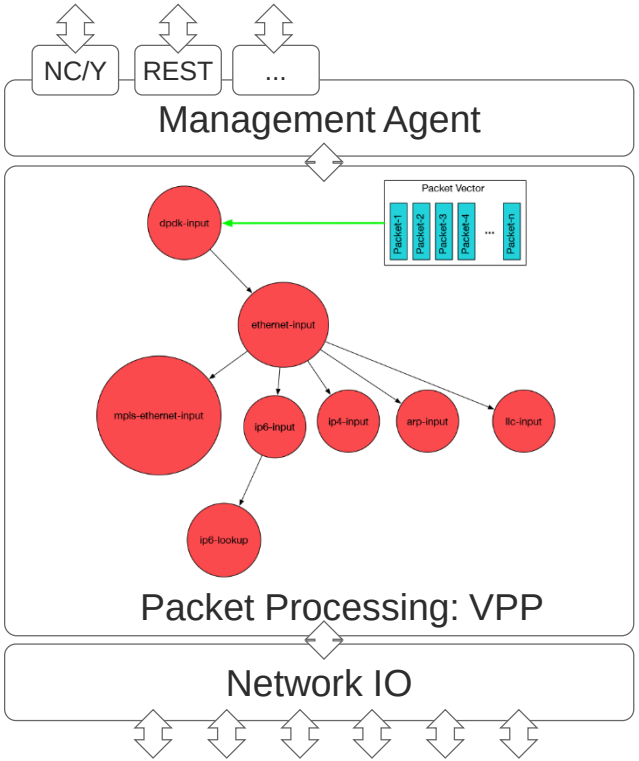
In-Band OAM Implementation

- Dataplane Implementation:
  - FD.io/VPP (see fd.io)
  - IOS (ISR-G2) – PI31 (CCO: End of July/16) – In-band OAM Config Guide for IOS
  - IOSv/VIRL
- Controller Implementation:
  - OpenDaylight (Carbon release)
- Supporting information for in-band OAM: https://github.com/CiscoDevNet/iOAM



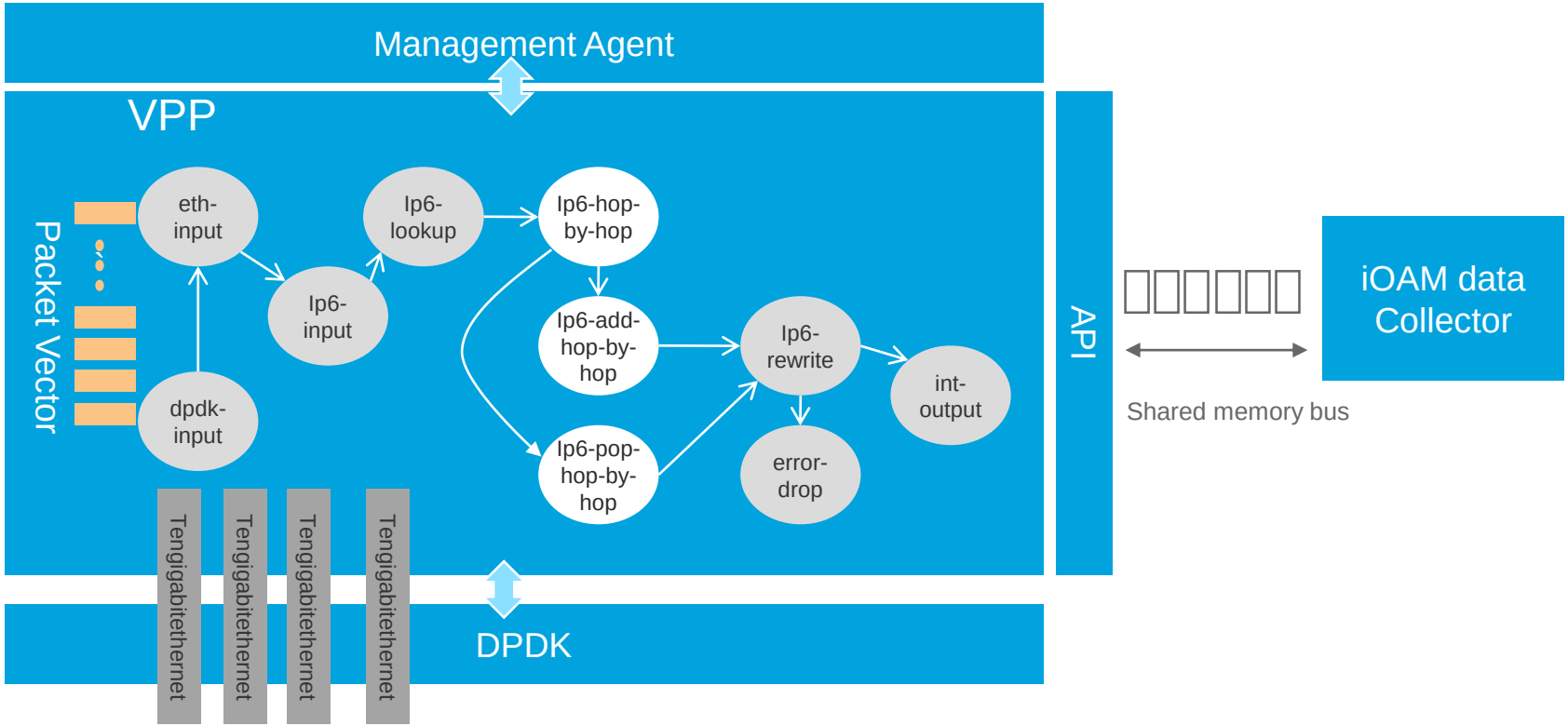
Introducing Vector Packet Processor - VPP

- VPP is a rapid packet processing development platform for highly performing network applications.
- It runs on commodity CPUs and leverages DPDK
- It creates a vector of packet indices and processes them using a directed graph of nodes – resulting in a highly performant solution.
- Runs as a Linux user-space application
- Ships as part of both embedded & server products, in volume; Active development since 2002
- Base iOAM code already available in fd.io open repositories (more coming soon)
- See also: FD.IO (The Fast Data Project)



Some initial iOAM already in FD.IO:
https://gerrit.fd.io/r/gitweb?p=vpp.git;a=blob_plain;f=vnet/vnet/ip/ip6_hop_by_hop.c;hb=HEAD

IOAM in VPP (native IPv6 encap)



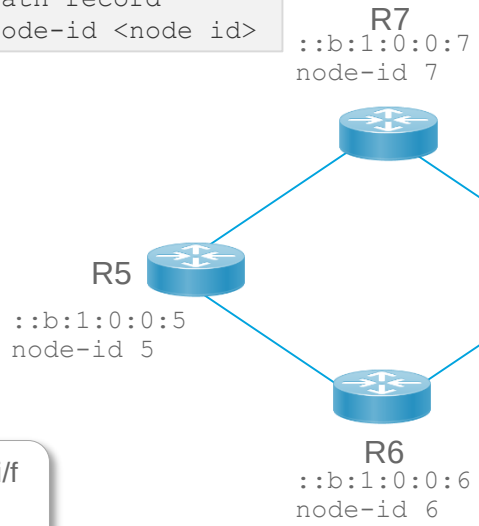
In-Band OAM Test Drive

- Extended Ping

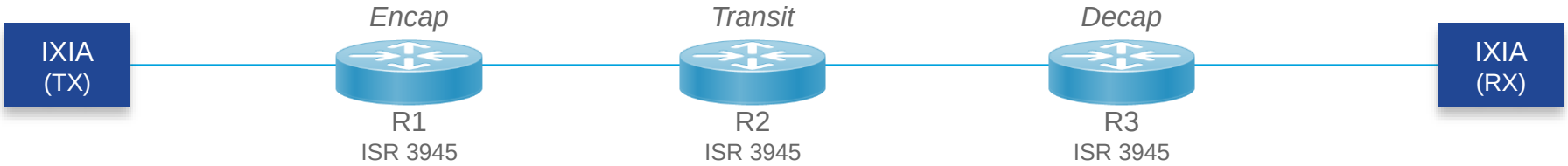
```
R5#ping
Protocol [ip]: ipv6
Target IPv6 address: ::b:1:0:0:8
Repeat count [5]: 1
Datagram size [100]:
Timeout in seconds [2]:
Extended commands? [no]: y
Source address or interface:
UDP protocol? [no]:
Verbose? [no]:
Precedence [0]:
DSCP [0]:
Include hop by hop Path Record option? [no]: y
Sweep range of sizes? [no]:
% Using size of 296 to accomodate extension headers

Type escape sequence to abort.
Sending 5, 296-byte ICMP Echos to ::B:1:0:0:8, timeout is 2 seconds:
5(5)----------(3)7(4)----------(3)8(5)----------(5)8(3)----------
(4)6(3)----------(5)5!
Success rate is 100 percent (1/1), round-trip min/avg/max = 4/6/10 ms
```

ingress i/f
node-id
egress i/f

```
ipv6 ioam path-record
ipv6 ioam node-id <node id>
```



Performance Test Results (native IPv6 encap)
In-band OAM with no performance degradation

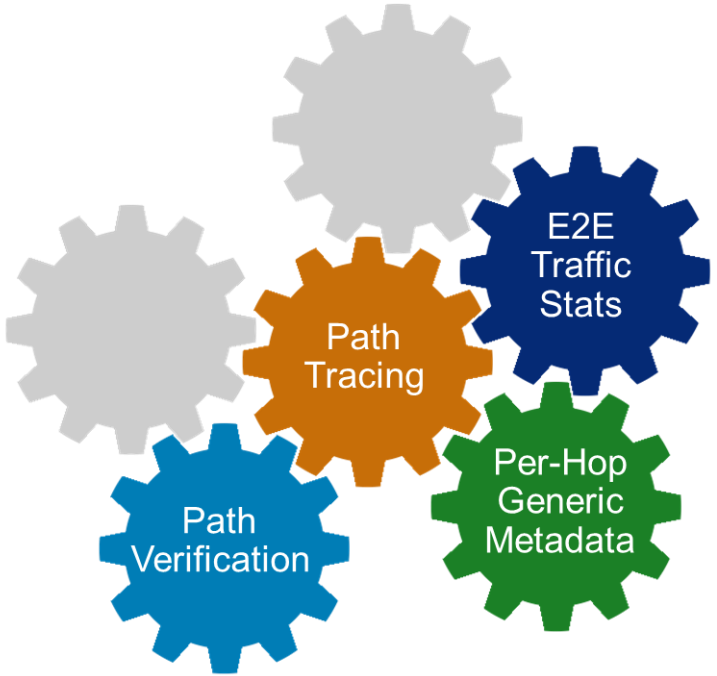


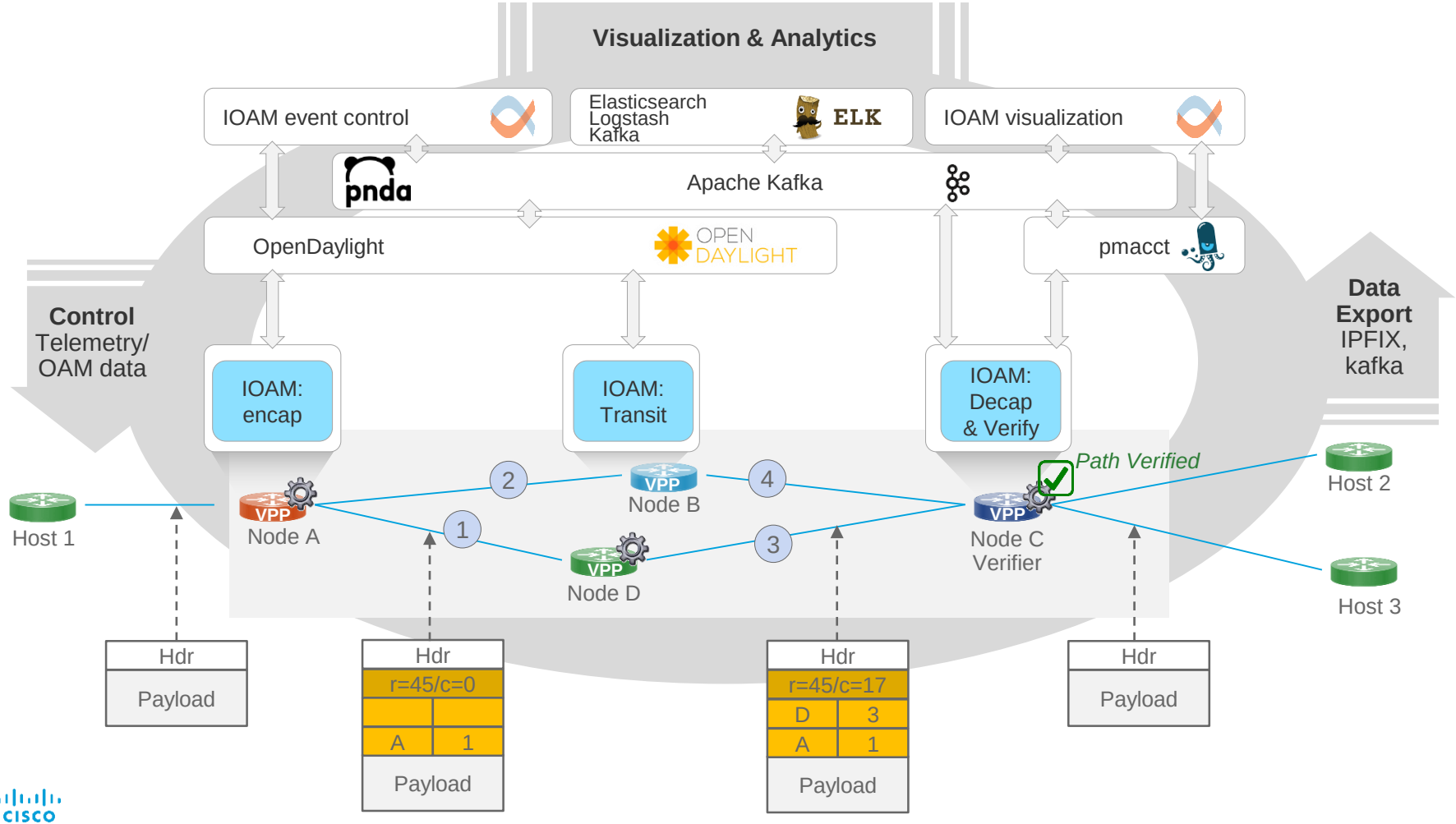
CPU Utilization in % at 600Mbps/no drops/IMIX traffic – 100 flows										
Traffic Bi Dir	**Frame size in Bytes**	**Traffic Load in %**	**Loss in %**	**RX Throughput in mbps**	**Latency (ns) AVG**	**Latency (ns) MIN**	**Latency (ns) MAX**	**CPU UTIL % ENCAP NODE (R1)**	**CPU UTIL % Inter NODE (R2)**	**CPU UTIL % DECAP NODE (R3)**
CEF IPv6	imix	60	0	1131	187401	54260	921680	23	23	23
GRE+IPv6	imix	60	0.04	1138	186753	56060	923120	24	24	25
iOAM6	imix	60	0	1138	186671	40640	917780	27	24	26

Network Operations and SLA diagnostics
In-Band OAM as part of a larger ecosystem

Examples of enhanced use-cases

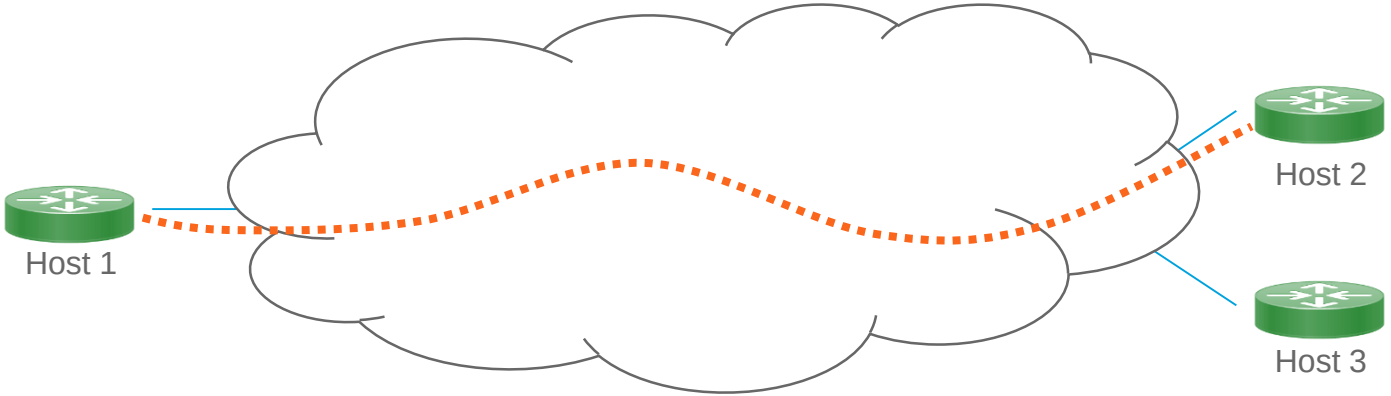
- SLA verification
  - Service chain verification
  - Identifying the reason for path verification failure (triggered OAM)
  - SLA check in overlay networks
- Efficient fault detection and fault isolaton
- Smart service-selection





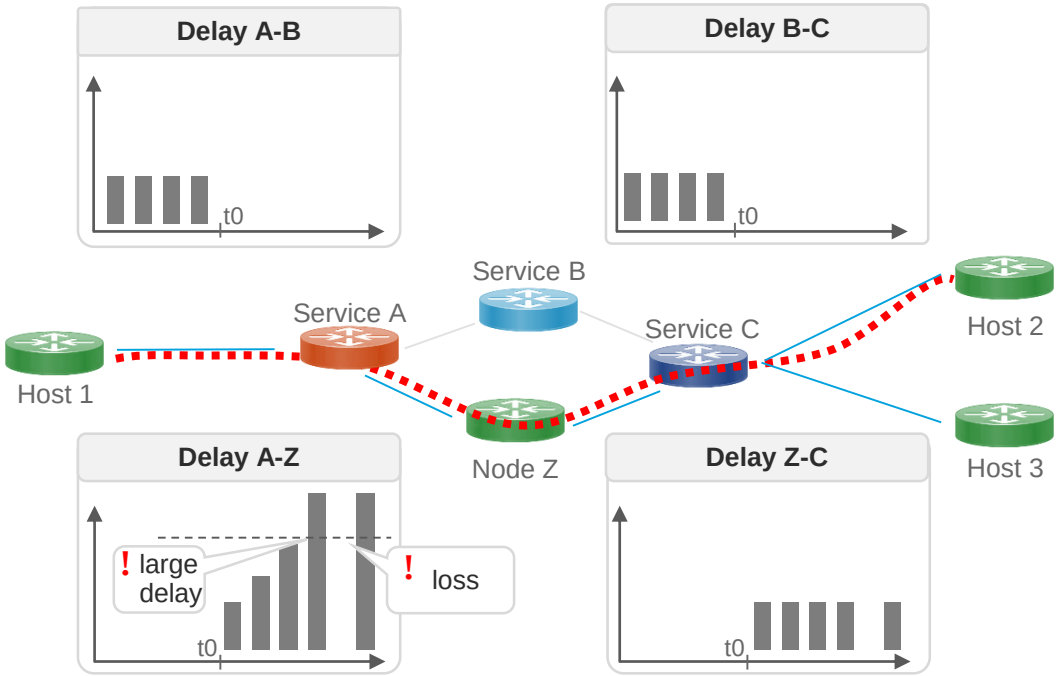
Example – SLA Verification

Example: OAM Analytics / SLA check
Identifying network & service chain bottlenecks



Observation: Suddenly connectivity issues between Host 1 and Host 2 – SLA violated

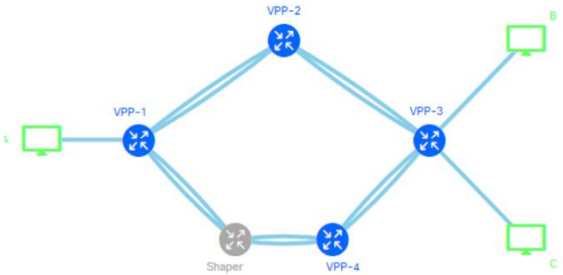
Detailed path tracing to identify the reason



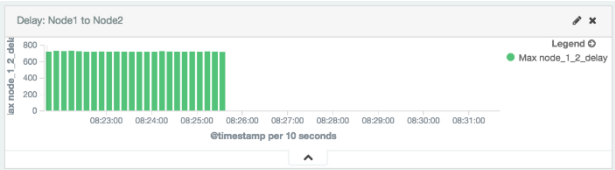
Observation

- Original path was A-B-C and fast
- At time t0 a path change happened: Traffic rerouted to A-Z-C
- Either Link A-Z is a low bandwidth link causing packet drops and/or Node Z is overloaded

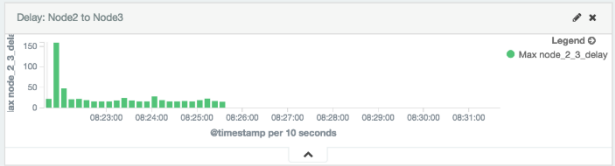
Per-link delay reveals root cause: Route change to a low bandwidth link



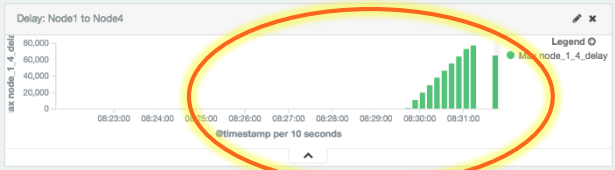
Delay 1-2



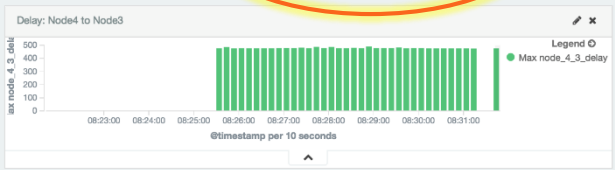
Delay 2-3



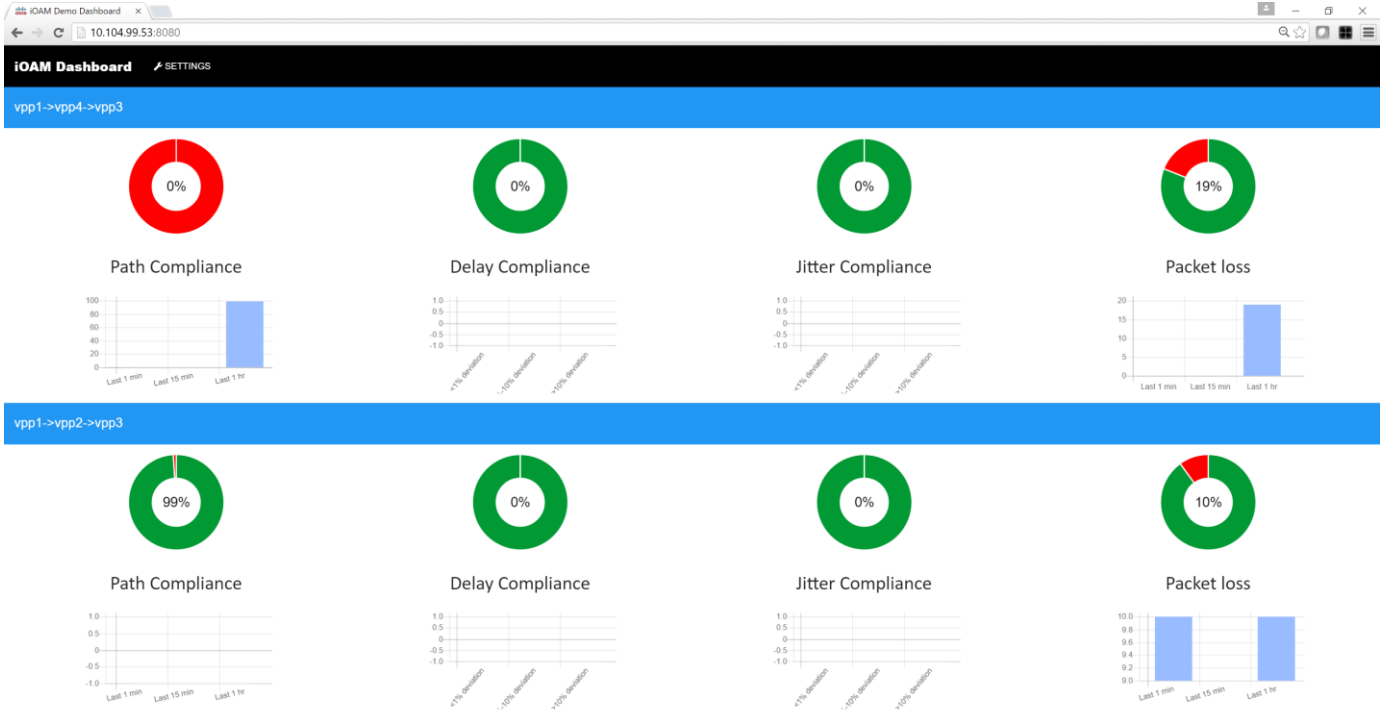
Delay 1-4



Delay 4-3



Overlay Networks:
Checking SLA Compliance on the real traffic



Example: Smart Traceroute – Fault Detection and Fault Isolation

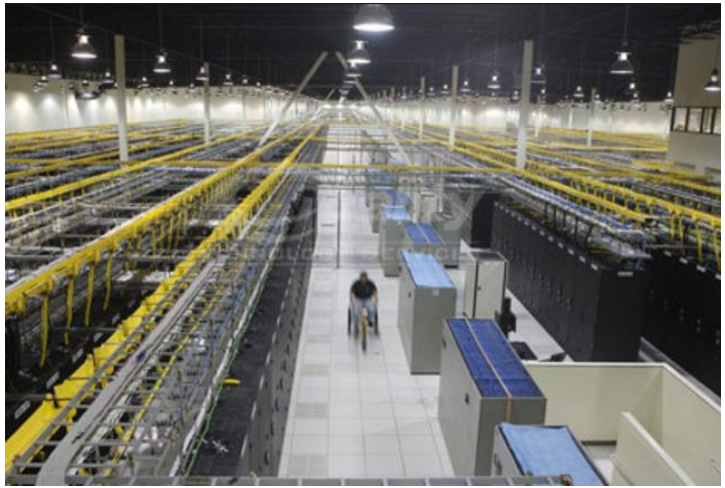
Active Network Probing:
Objective: Fast Failure Detection and Isolation

Current situation:

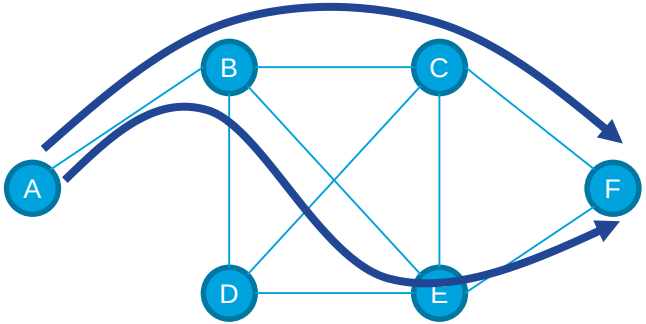
- Failure detection takes 10s of seconds in large networks
- Failure isolation (using ping and traceroute) takes several minutes: Probing based on traceroute is slow.

Objective & Approach

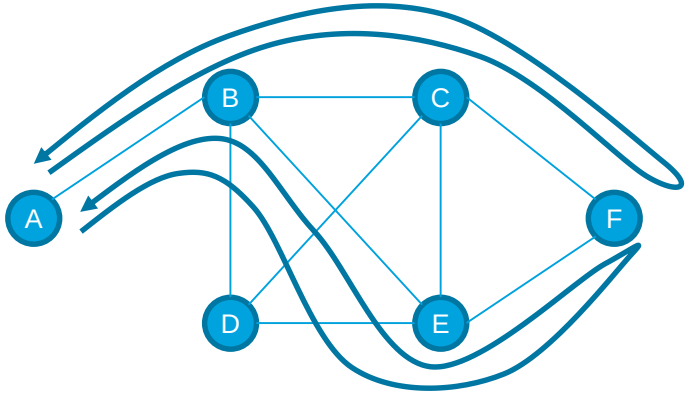
- Significantly improve failure detection and isolation times.
- Active network probing using UDP probe with in-band OAM



Normal network operation

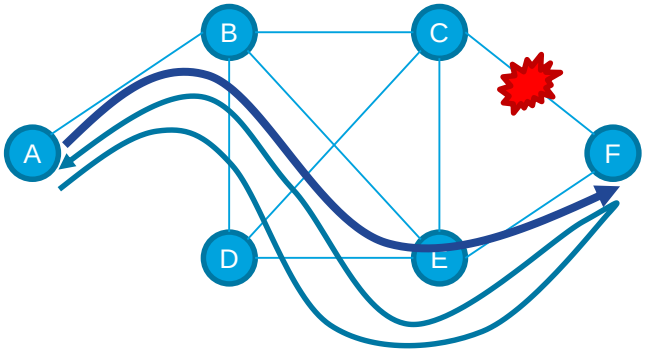


Customer traffic



Continous end point probing
using probes similar to customer traffic
(i.e. same ports)

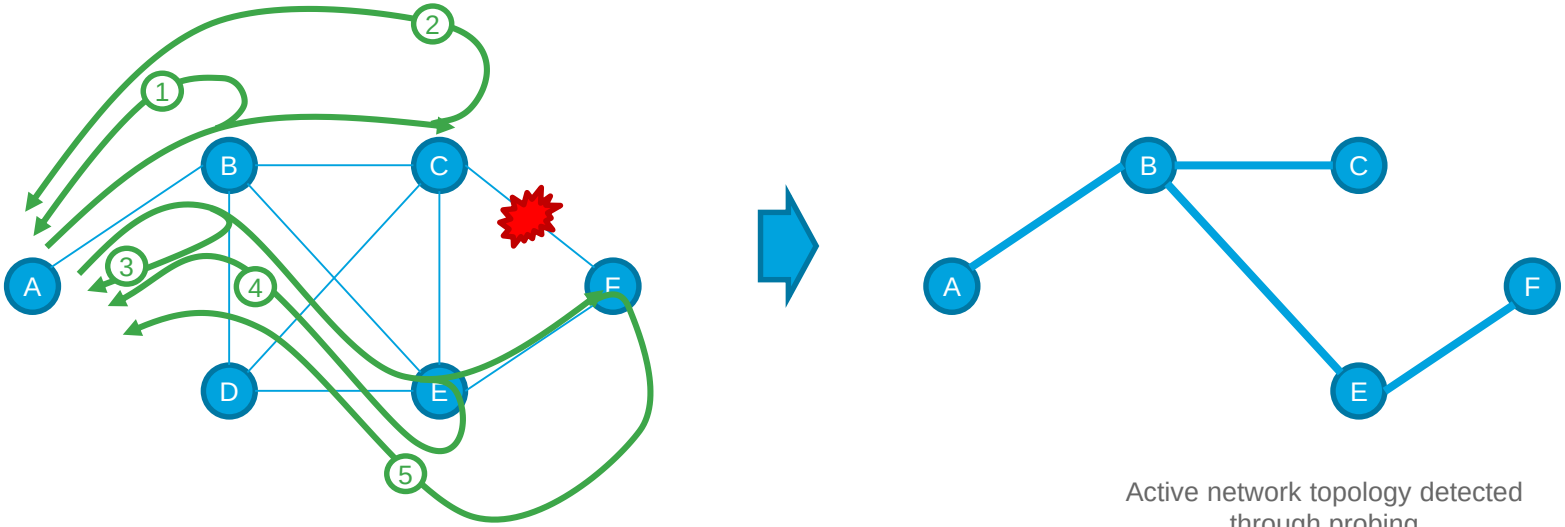
Network failure isolation



Observation from in-band OAM data:
Only subset of overall traffic arrives at F.

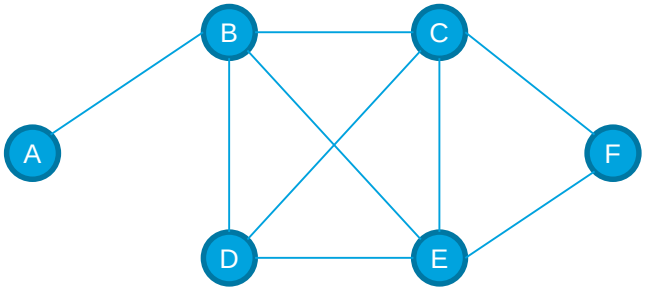
Detected either from live customer traffic
or probe traffic, or both.

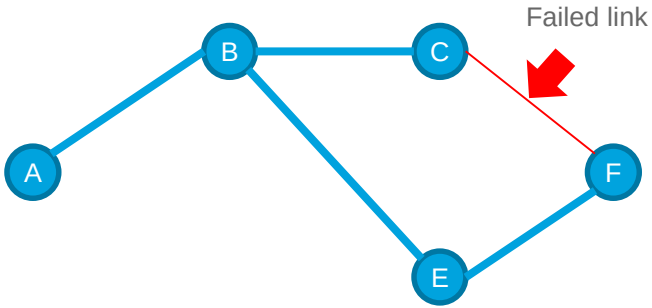
Network failure detection:
Probing with integrated loopback



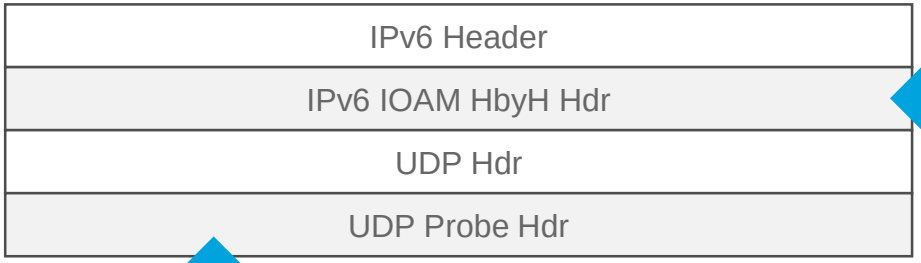
Active network topology detected through probing

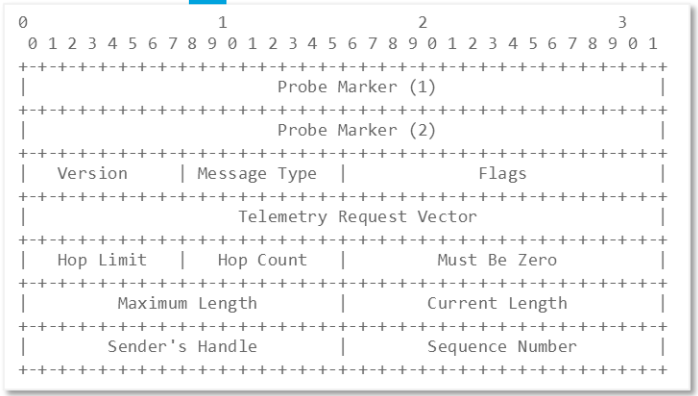
Network failure isolation: Correlate detected forwarding topology with original network topology



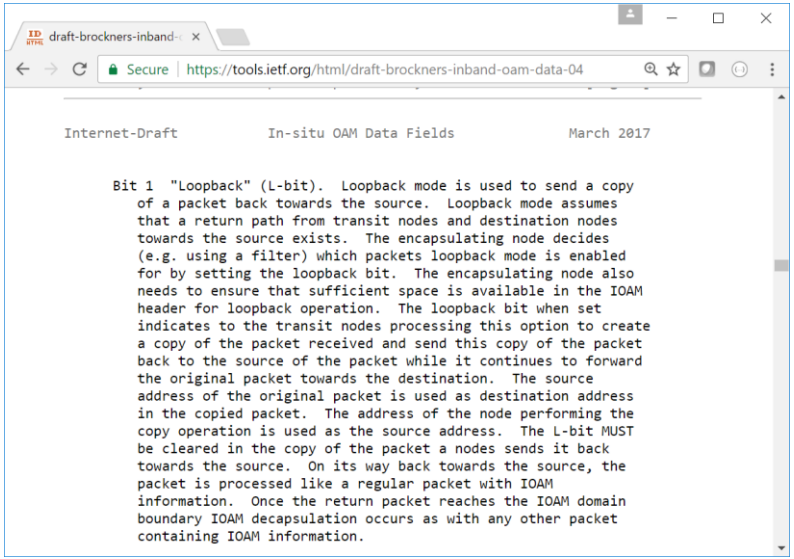


iOAM Ping UDP-Probe and Loopback Flag





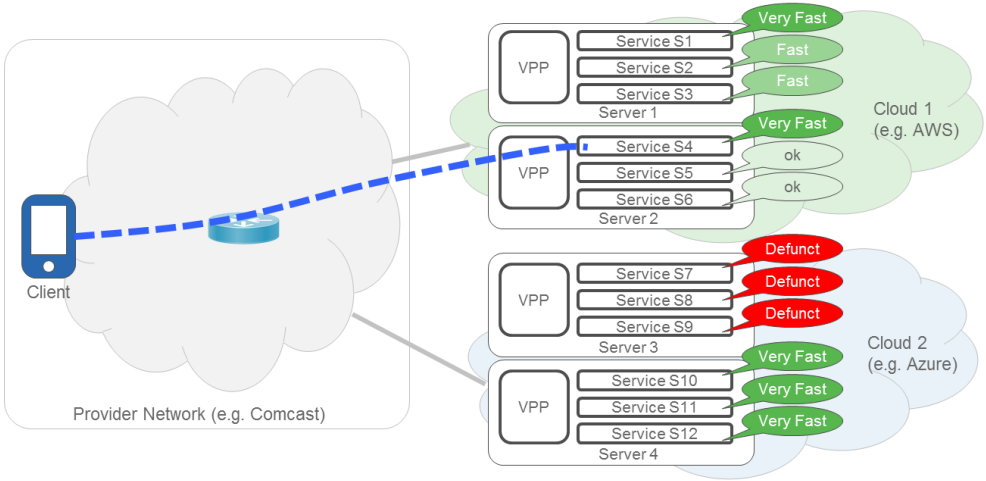
https://tools.ietf.org/html/draft-lapukhov-dataplane-probe-01



Example: M-AnyCast: Intelligent Service Selection

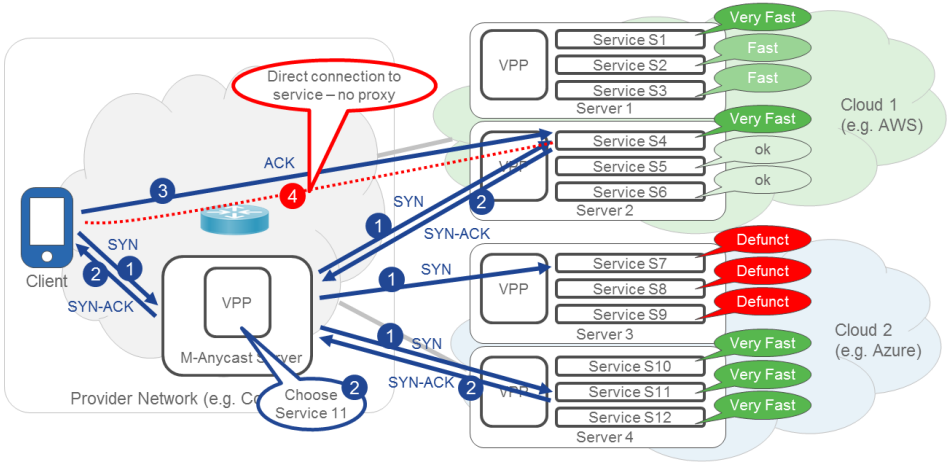
Service Selection: Problem Statement

- Highly redundant micro-services (e.g. video-chunck servers) hosted as containers in multiple public clouds
  - All Services have an IPv6 address. All Services share the same secondary anycast address.
  - Performance characteristics of individual service instances differ: Access latencies, server load, service load, service liveliness
- Client needs to choose an appropriate service to connect to

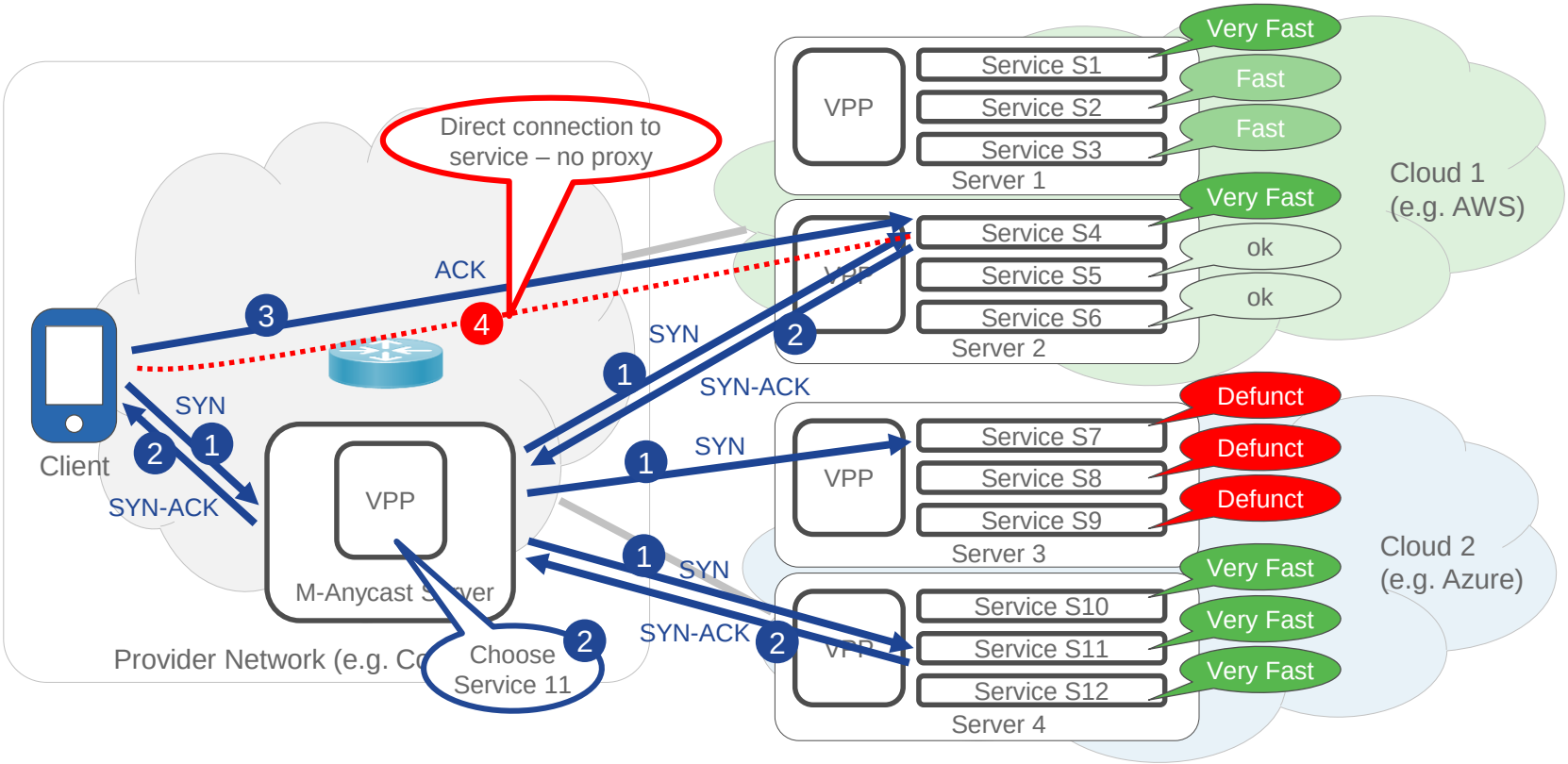


Service Selection: Solution Concept

- Introduce M-Anycast Server
- Leverage Segment Routing to steer traffic, in-band OAM for optimized service selection
- M-Anycast Server:
  - Hosts the Anycast address of the services
  - Intercepts TCP-SYN, replicates the SYN and sends to a selected subset of all services using SR
  - Chooses an appropriate SYN-ACK using embedded in-band OAM data and forwards that SYN-ACK to the client

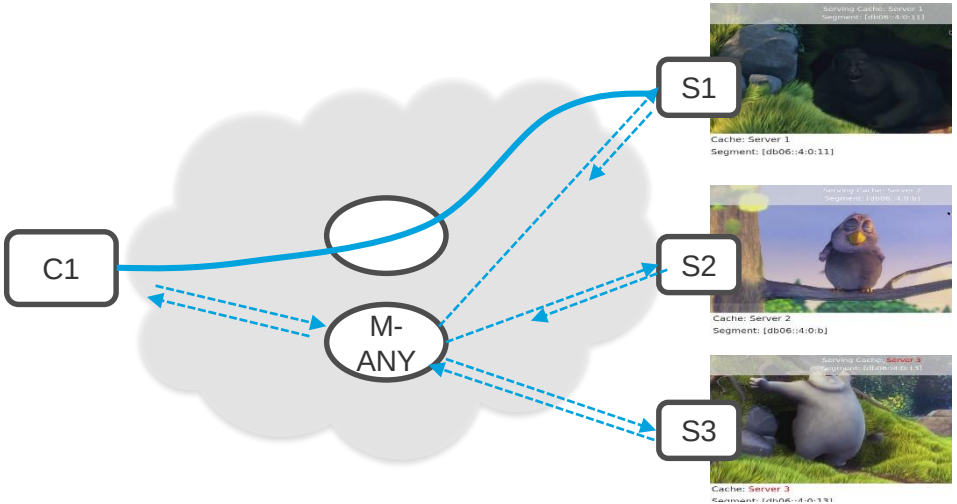


Service Selection: Solution Concept



Demo - Overview

- Highly resilient video playout combining IPv6 addressible video services with M-anycast as an efficient and fail safe service selection and load-balancing method
  - 6CN: IPv6 addressible video-segments form the service
  - M-Anycast solution implemented in FD.io/VPP:
    - M-Anycast server – Server selector with IOAM and SRv6 spray functions
    - VPP co-located with client and servers to perform SRv6 and IOAM functions



Demo Video: https://youtu.be/-jqww8ydWQk

Summary

In-Band OAM – Implementation Status

- Dataplane Implementation:
  - IOS
    - IOS (ISR-G2) – PI31 (CCO: End of July/16) – In-band OAM Config Guide for IOS
    - IOSv/VIRL
  - OpenSource
    - FD.io/VPP (see fd.io)
    - OpenDaylight Control App (proof of transit / tracing)
  - Supporting information for in-band OAM: https://github.com/CiscoDevNet/iOAM
- Standardization IETF
  - Authored by: Cisco, Comcast, Facebook, JPMC, Bell Canada, Mellanox, Marvell, Barefoot, rtBrick
  - Documents:
    - In-band OAM requirements: https://tools.ietf.org/html/draft-brockners-inband-oam-requirements-03.txt
    - In-band OAM data types: https://tools.ietf.org/html/draft-brockners-inband-oam-data-04.txt
    - In-band OAM transport: https://tools.ietf.org/html/draft-brockners-inband-oam-transport-03.txt
    - Proof-of-transit: https://tools.ietf.org/html/draft-brockners-proof-of-transit-03.txt
- Videos:
  Google+ In-Band OAM group:
  https://plus.google.com/u/0/b/112958873072003542518/112958873072003542518/videos?hl=en
  Youtube In-Band OAM channel:
  https://www.youtube.com/channel/UC0WJOAKBTrftyosP590RrXw
- Blogs:
  blogs.cisco.com/getyourbuildon/a-trip-recorder-for-all-your-traffic
  blogs.cisco.com/getyourbuildon/verify-my-service-chain

In-Band OAM: Summary

- Enhanced visibility for all your traffic:
  New sources of data for SDN applications
- Network provided telemetry data gathered and added to live data
  - Complement out-of-band OAM tools like ping and traceroute
  - Path / Service chain verification
- Record the packet's trip as meta-data within the packet
  - Record path and node (i/f, time, app-data) specific data hop-by-hop and end to end
  - Export telemetry data via Netflow/IPFIX/Kafka to Controller/Apps
- In-band OAM can be implemented without forwarding performance degradation
- IOS and OpenSource (FD.io/VPP, OpenDaylight) Implementations

Thank You