Tracking Packets' Paths and Latency via INT (In-band Network Telemetry)

3rd P4 Workshop

May/24, 2016

Jithin Thomas, Barefoot Networks

Petr Lapukhov, Facebook

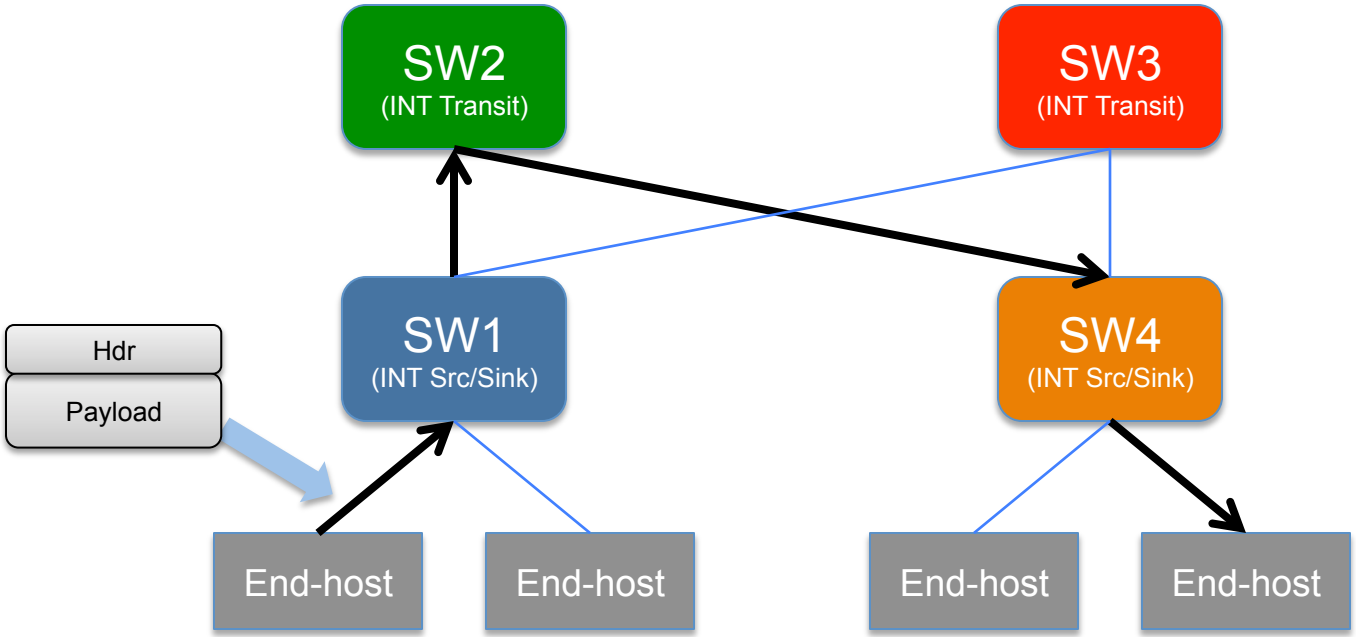
Agenda

- In-band Network Telemetry (INT)
- Facebook’s INT use cases
- Path and Latency Tracking (PLT) app
- PLT demo

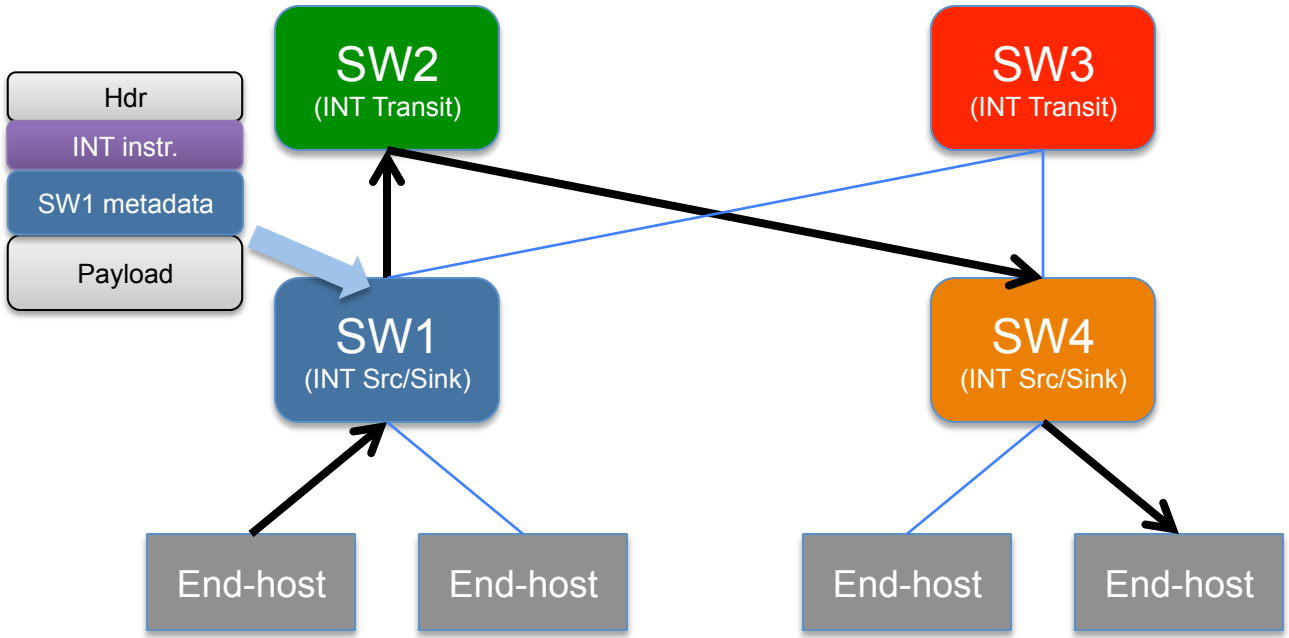
What is INT?

Framework designed to allow the collection and reporting of network state, by the data plane, without requiring intervention or work by the control plane.

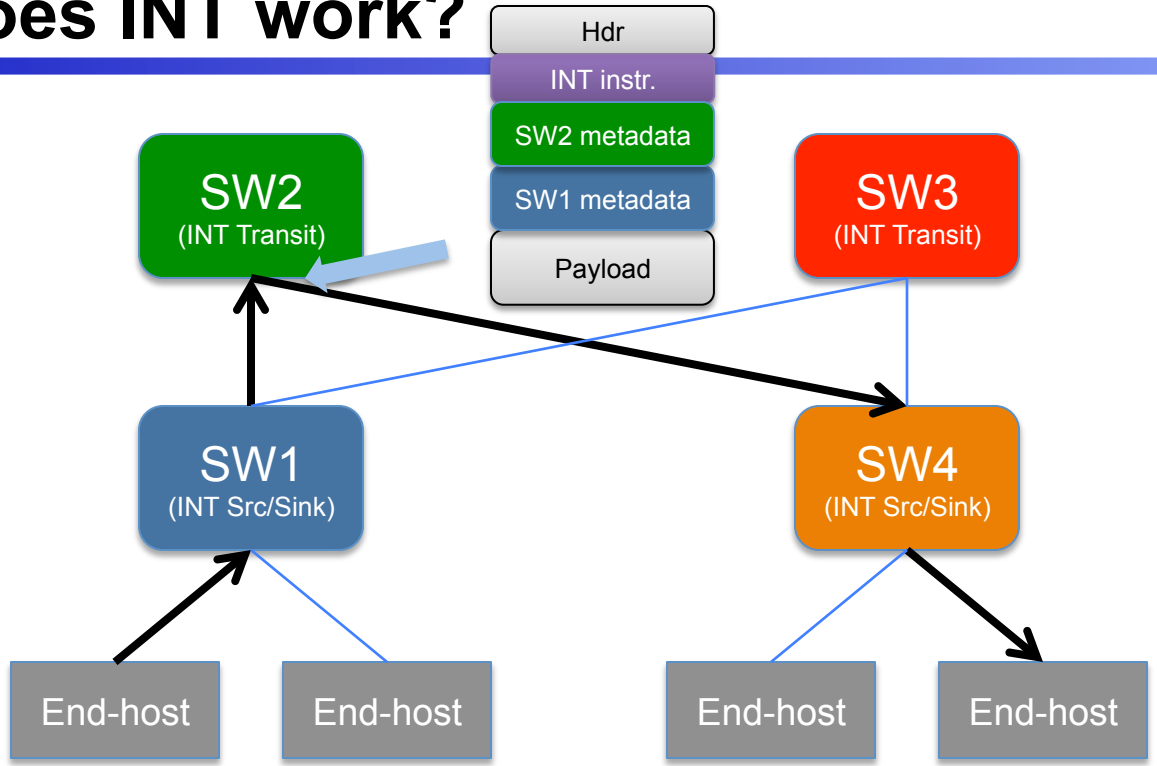
How does INT work?



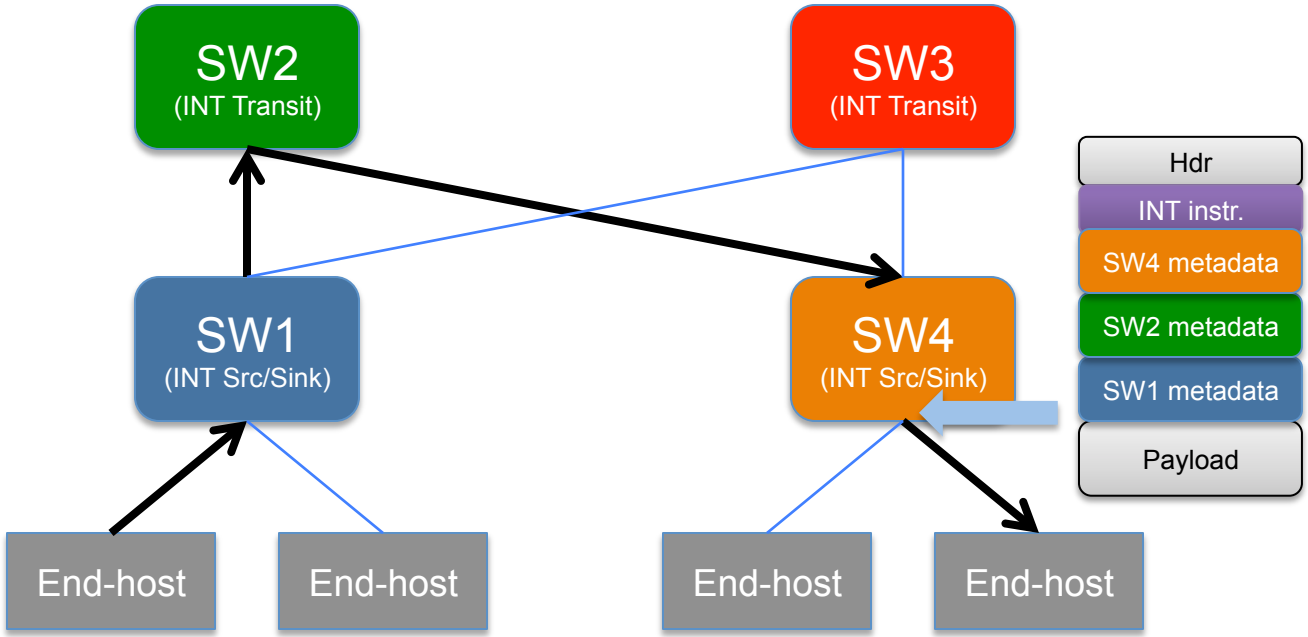
How does INT work?



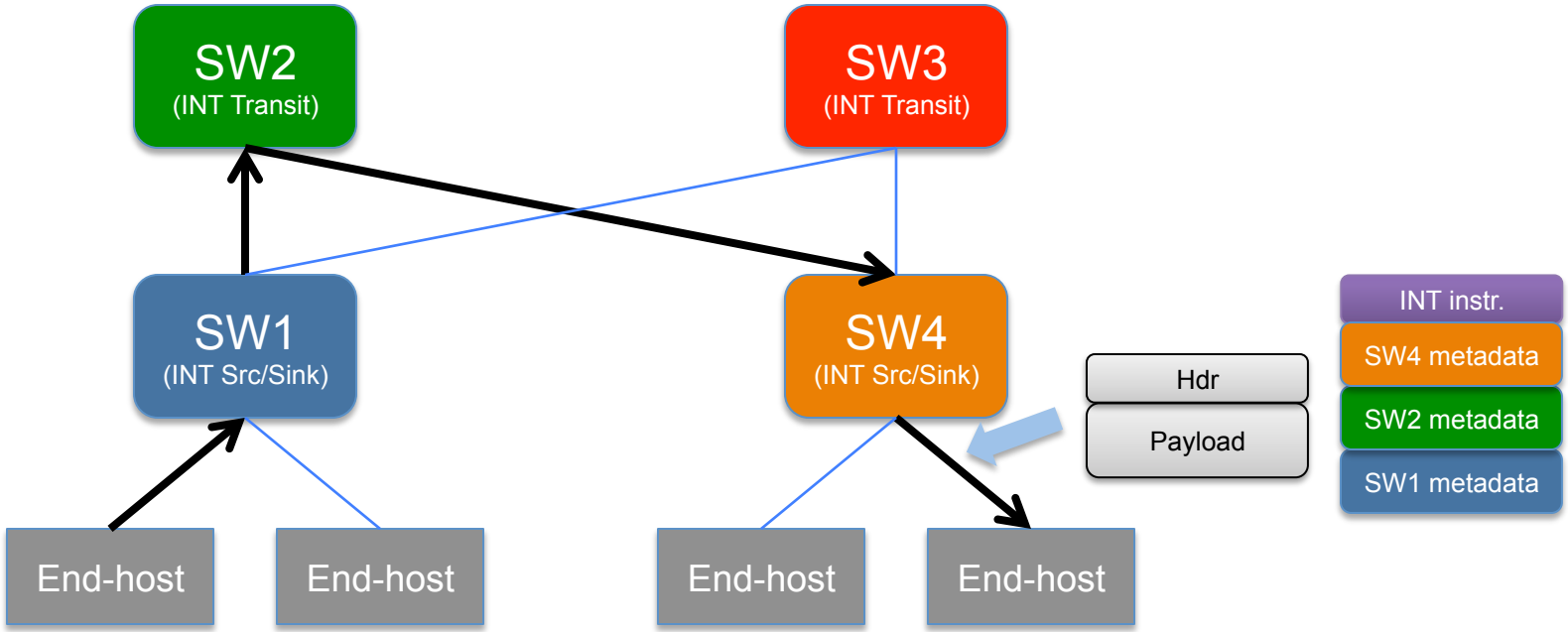
How does INT work?



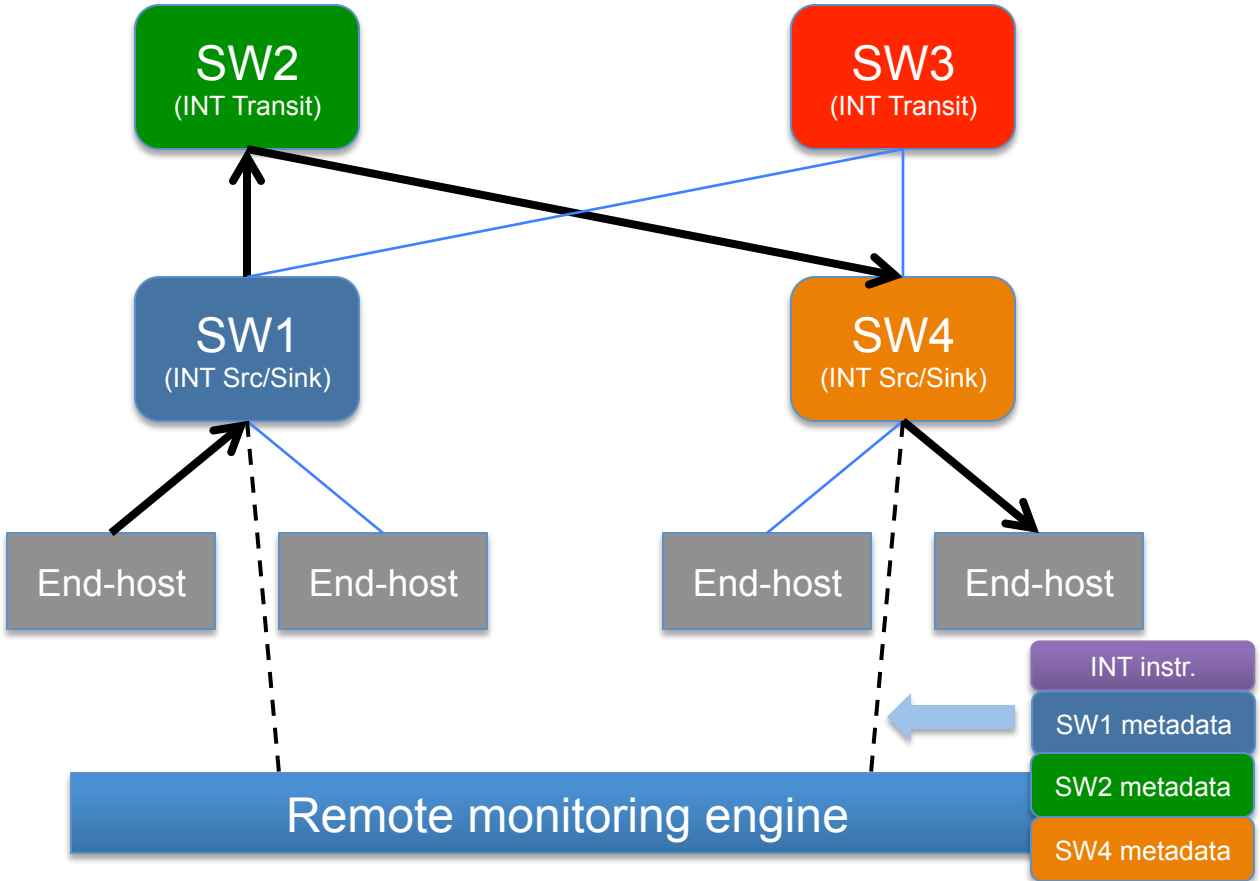
How does INT work?



How does INT work?



How does INT work?



Quickly recapping INT

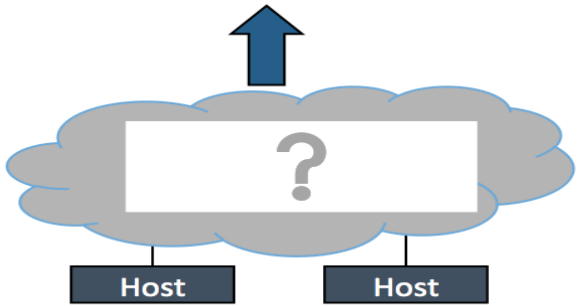
- Framework designed to allow the collection and reporting of network state, by the data plane, without requiring intervention or work by the control plane.
- Packets contain header fields that are interpreted as “telemetry instructions” by network devices.
- These instructions tell INT-capable devices what to collect; collected data is written into the packets themselves
- Examples of per-packet metadata one can collect via INT include switch IDs, input/output port IDs, hop latency, queue occupancy, arrival/departure timestamp, etc.

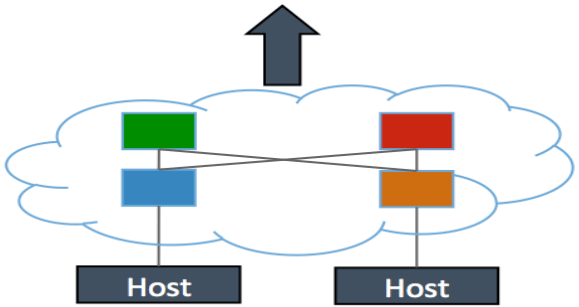
Why INT?

- Low visibility into network state
- Polling-based
- Aggregated counters
- Limited by control-plane speed

vs.

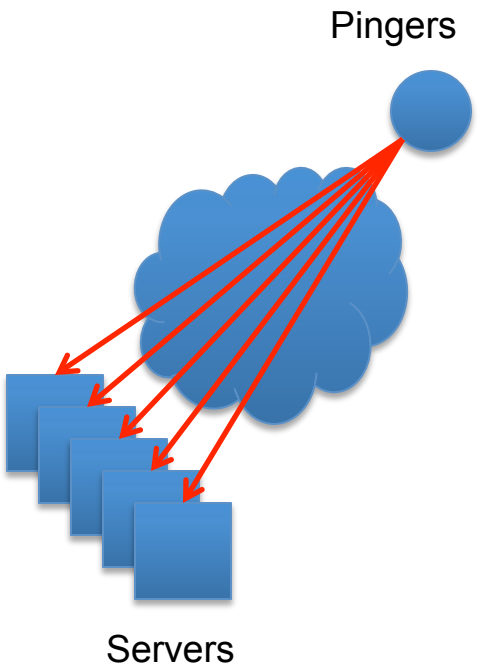
- Complete network visibility
- Real-time monitoring
- Per-packet metadata
- Full line rate, zero switch CPU involvement





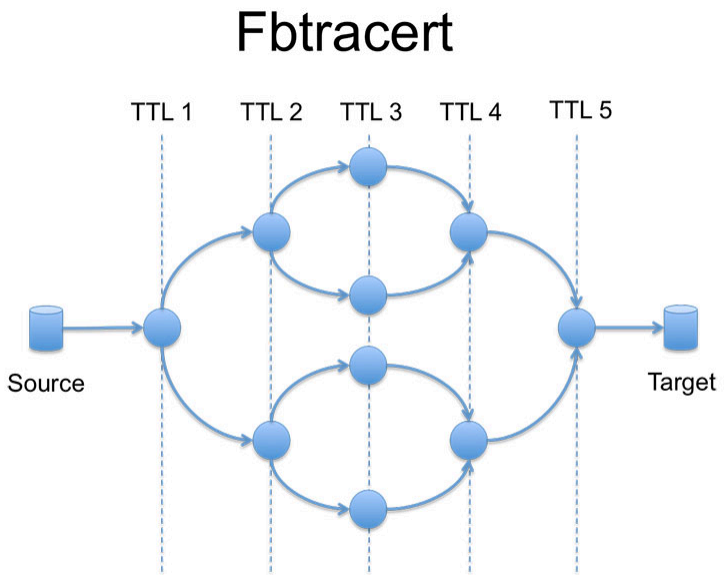
Facebook's INT Use Cases

- **NetNORAD**: Troubleshooting networks via end-to-end probing
- Hosts **send** and **receive** active probes
- PING on steroids
- Uses UDP probes
- Detect packet loss **quickly** and reliably
- Also track latency
- ~300-400Gbps of probe traffic



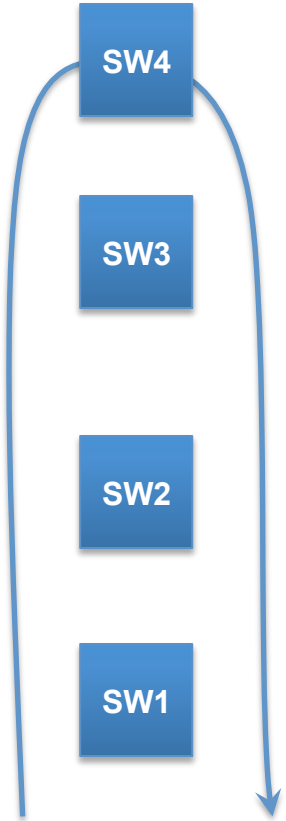
Facebook's INT Use Cases

- Fault **isolation** still an issue
- Currently use **traceroute**-like approach
- **Fbtracert**: explore multiple paths "quickly"
- Time-to-detect: ~20 seconds
- Time-to-isolate: ~2-5 minutes
- Work really well in data-center



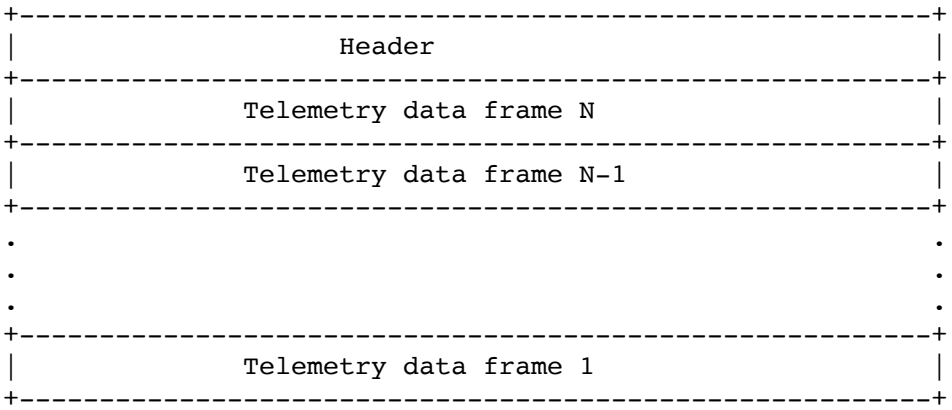
INT for advanced ping & traceroute

- Enhanced **ping**: record packet path
  - Device + Port
  - Analyzer can infer fault location
- Enhanced **traceroute**: loopback test
  - Turn packets around at some hop
  - Create looped flow **in data-plane**
- Goal: localize loss much faster
- Initiated by operator or software



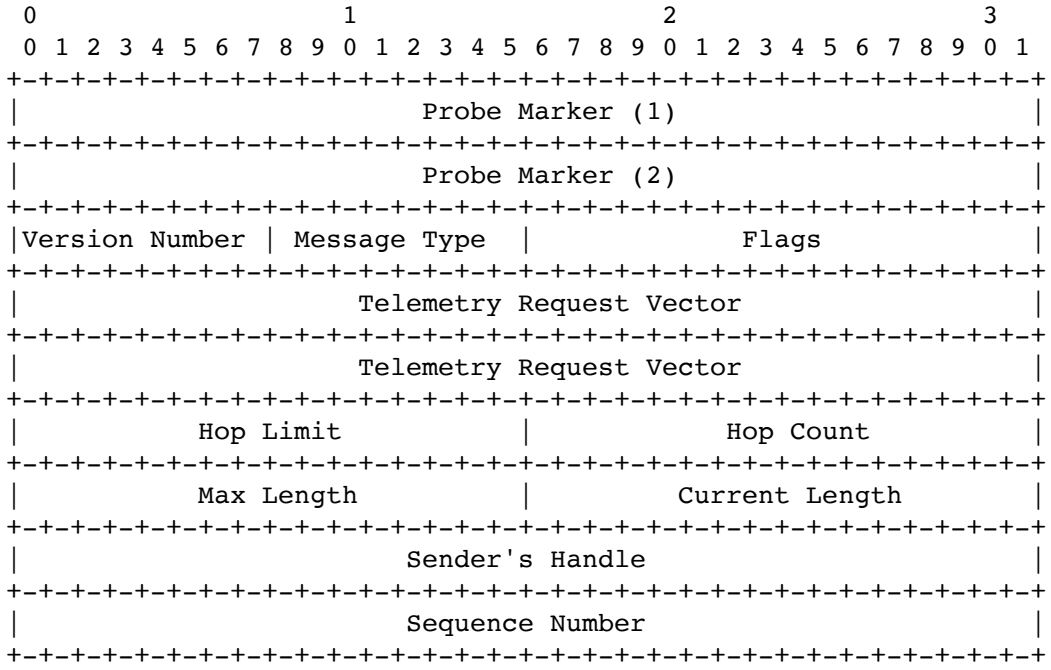
Implementation: draft-lapukhov-dataplane-probe

Format for inband telemetry request/ collection



Implementation: draft-lapukhov-dataplane-probe

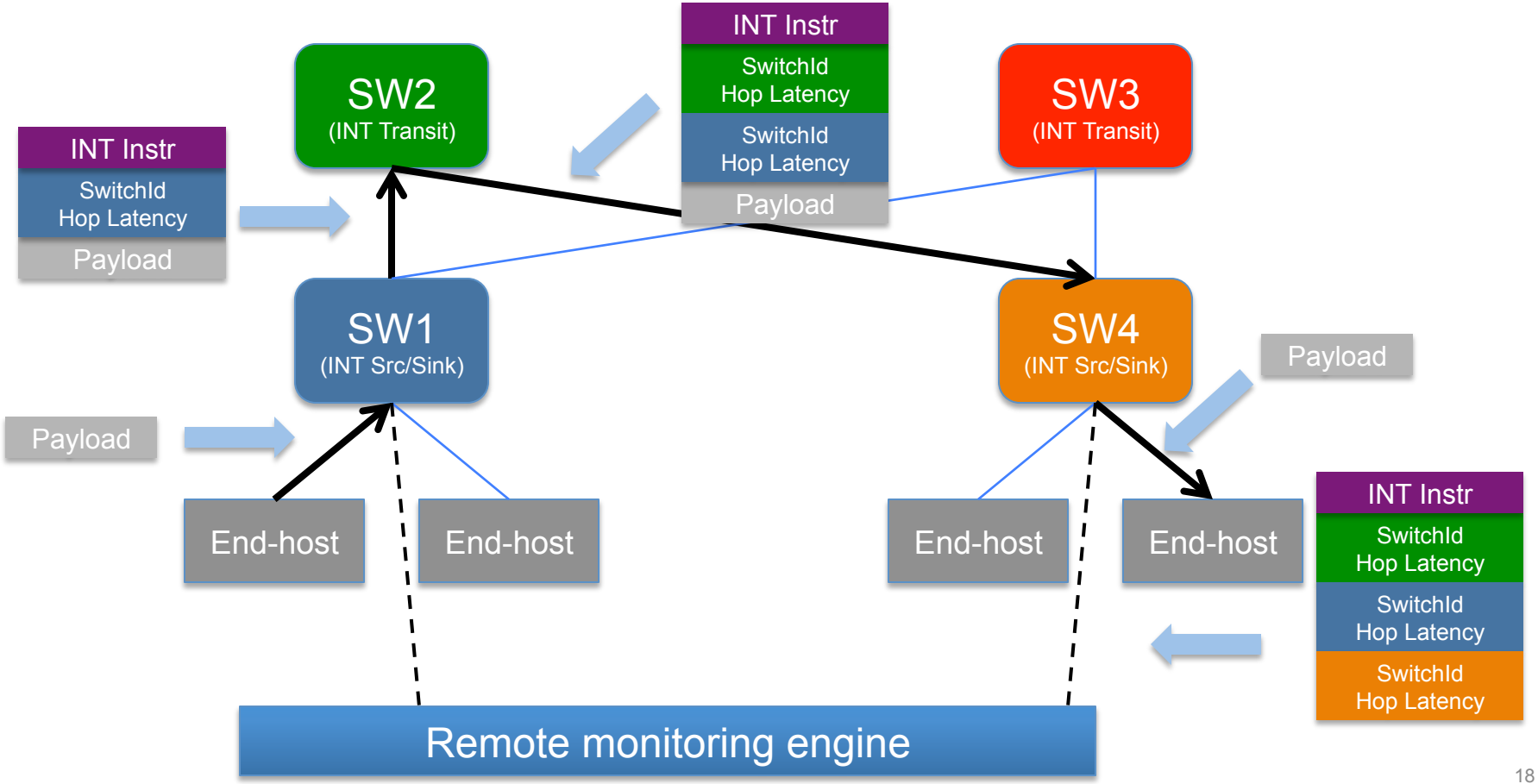
Telemetry Request Header



What is Path and Latency Tracking (PLT)?

- An application of INT
- Collect the physical path and hop latencies for every packet; INT Sinks export only meaningful records to the monitor
- INT Sinks generate a report each time they observe
  - A new connection
  - Path of an existing connection changes
  - Hop/end-to-end latency of an existing connection fluctuates significantly
- Reports are sent to a remote monitoring system as mirrored packets

How does PLT work?



Why is PLT useful?

- **Real-time anomaly detection and alert generation**
  - Congested connections
  - Congestion at switches
  - Unused (dead) links and switches
  - Imbalance of link utilization (ECMP and LAG)
  - Loops

- **Interactive analysis**
  - On-demand path visualization (for connections, src-dst ToR pairs, etc.)
  - Traffic matrix generation
  - Triage incidents of congestion - determine whether issue was caused by network/application behavior
  - And more…

PLT P4 design

- **Sending a report for each packet in the network could result in an excessive amount of redundant data.**

- **We have implemented a smart de-duplication logic in P4 to determine when reports need to be sent to the monitor**
  - Will be open-sourced in the near future

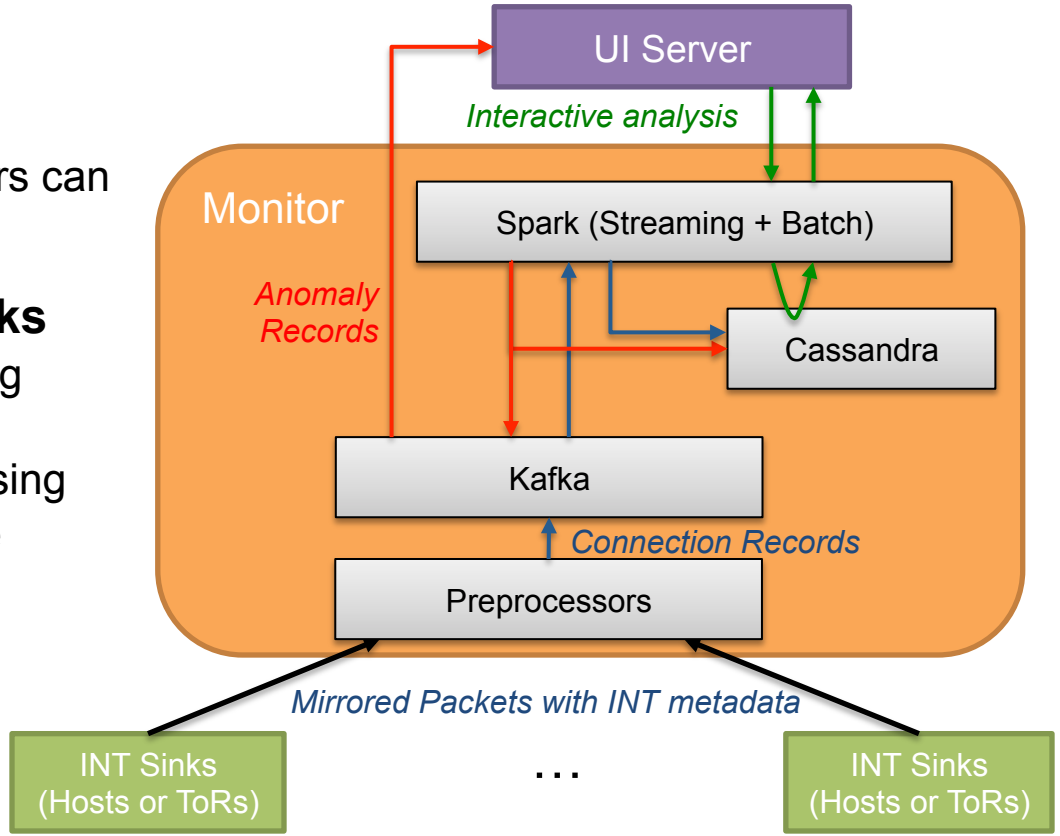
- **Reports are sent to the monitor as mirrored packets**

PLT monitoring engine

- **Scalable distributed system designed to**
  - consume the PLT reports in real time
  - generate alerts and
  - save the reports to a durable data store
- **Based completely on open-source tools**
- **Modular design implies that users can choose to seamlessly replace any component in the stack, if required.**
- **Will be open-sourced in the near-future**

PLT monitoring engine

- **Low-overhead design**
  - E.g. A dozen dedicated in-rack servers can cover a cluster (20 racks)
- **Open-source-based building blocks**
  - *Kafka:* Distributed pub-sub messaging
  - *Spark Streaming*: Near-real-time, in-memory large-data stream processing
  - *Cassandra*: Backend NoSQL storage
- **Will be open-sourced soon**



PLT demo

- **Demo**

With PLT, you can …

- Confirm the effect of device configuration or policy changes real time
- Verify and audit network behavior thoroughly
- Quantify RIB-FIB inconsistency
- Identify connections affected by planned maintenance or un-planned events
- Detect hop-latency increases and identify the victims; use the info to justify SLAs