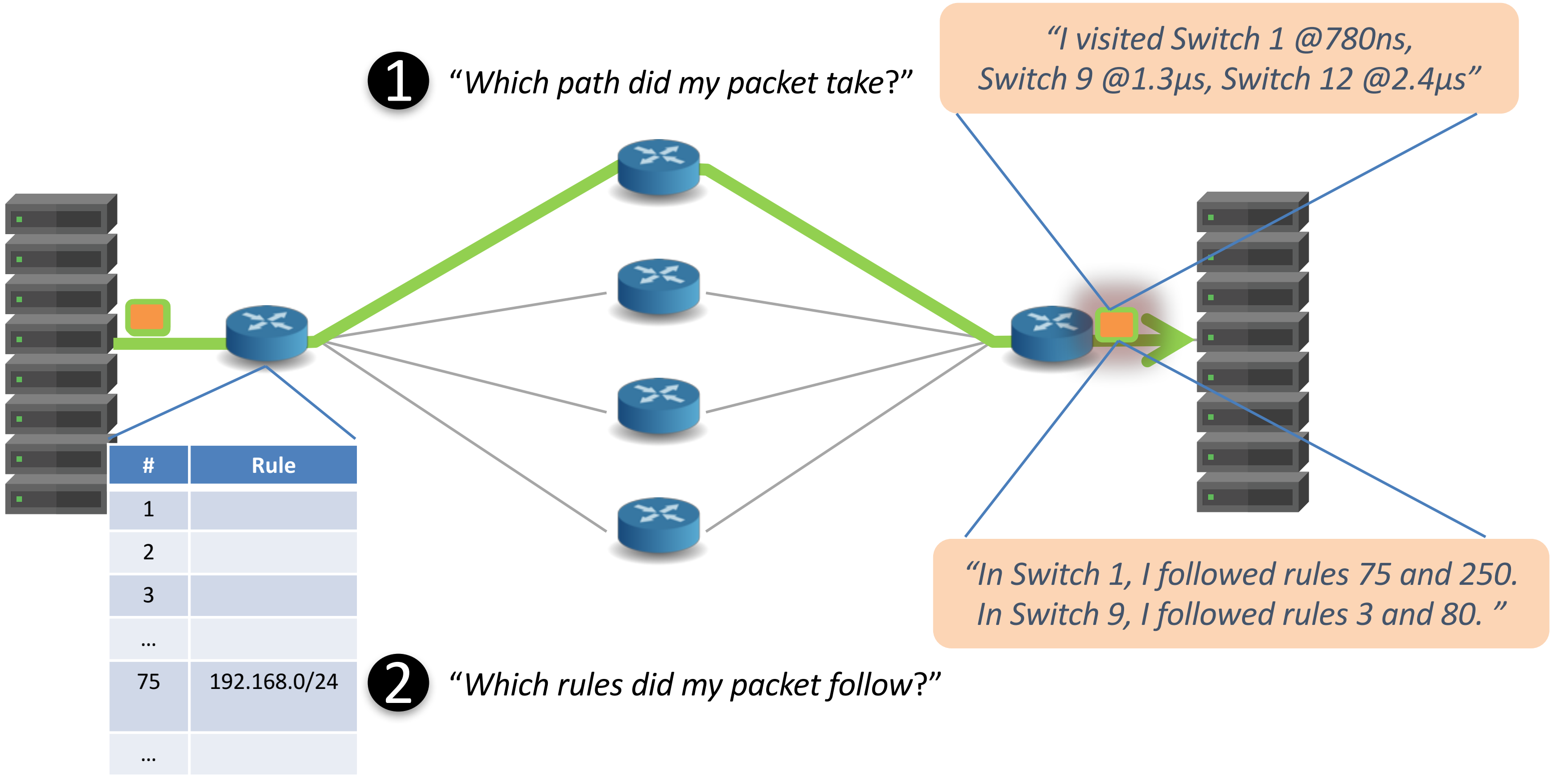
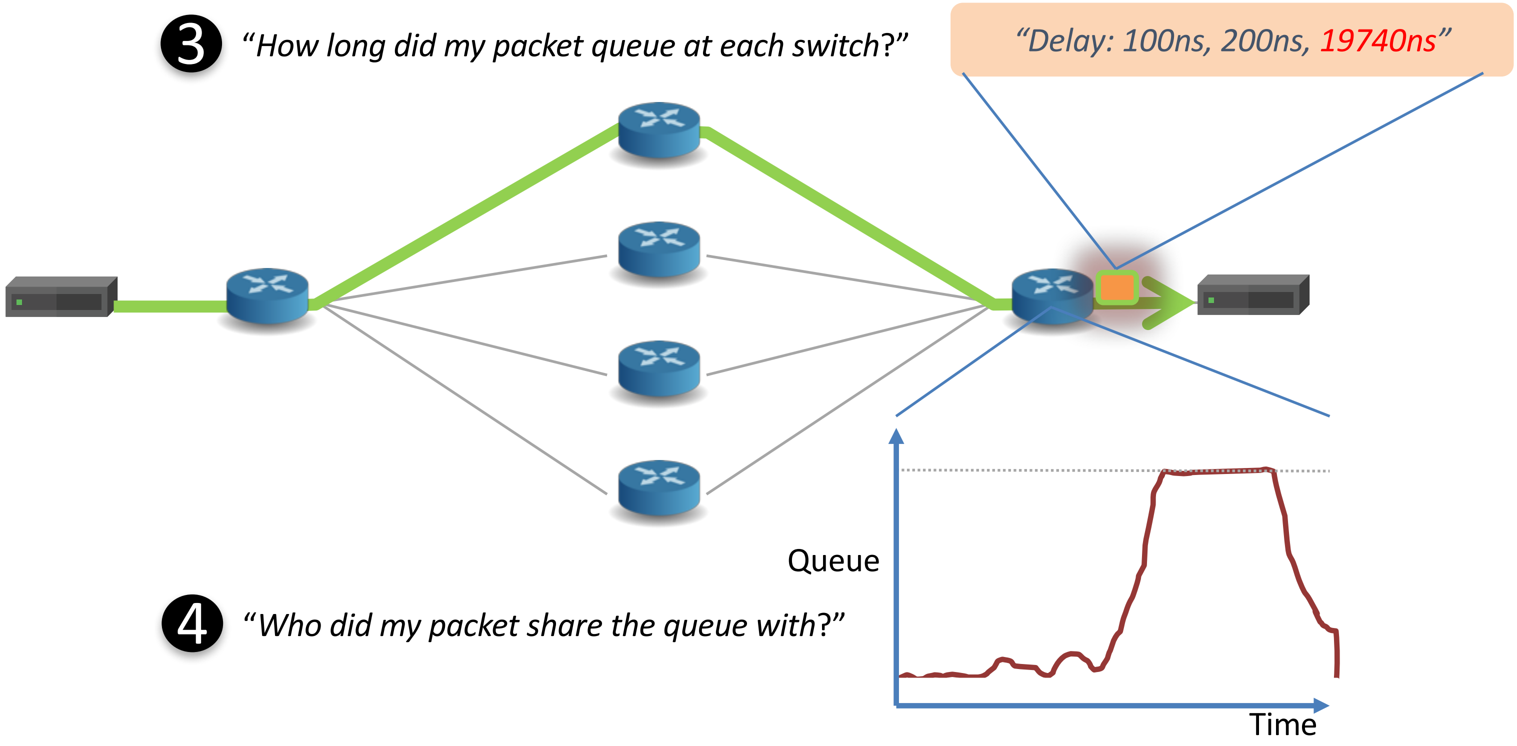
How can P4 programmability help this workshop?

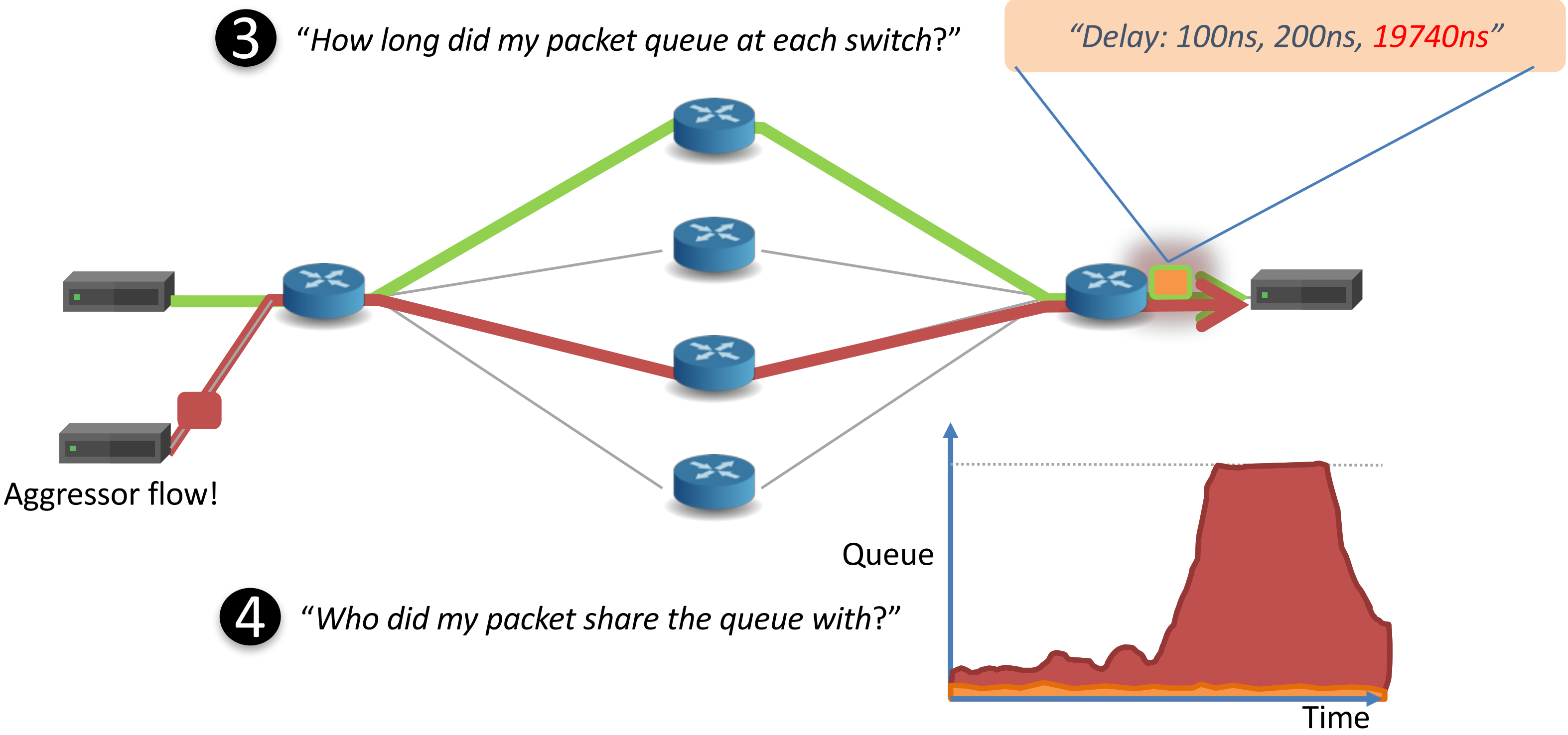
Changhoon Kim

BAREFOOT NETWORKS



1 "Which path did my packet take?"
"I visited Switch 1 @780ns, Switch 9 @1.3µs, Switch 12 @2.4µs"
| Rule
1
2
3
…
75 | 192.168.0/24
…
2 "Which rules did my packet follow?"
"In Switch 1, I followed rules 75 and 250. In Switch 9, I followed rules 3 and 80. "


3 "How long did my packet queue at each switch?"
"Delay: 100ns, 200ns, 19740ns"
Queue
4 "Who did my packet share the queue with?"
Time



3 "How long did my packet queue at each switch?"
"Delay: 100ns, 200ns, 19740ns"
Aggressor flow!
Queue
Time
4 "Who did my packet share the queue with?"

The network should answer these questions

1. *"Which path did my packet take?"*
2. *"Which rules did my packet follow?"*
3. *"How long did it queue at each switch?"*
4. *"Who did it share the queues with?"*

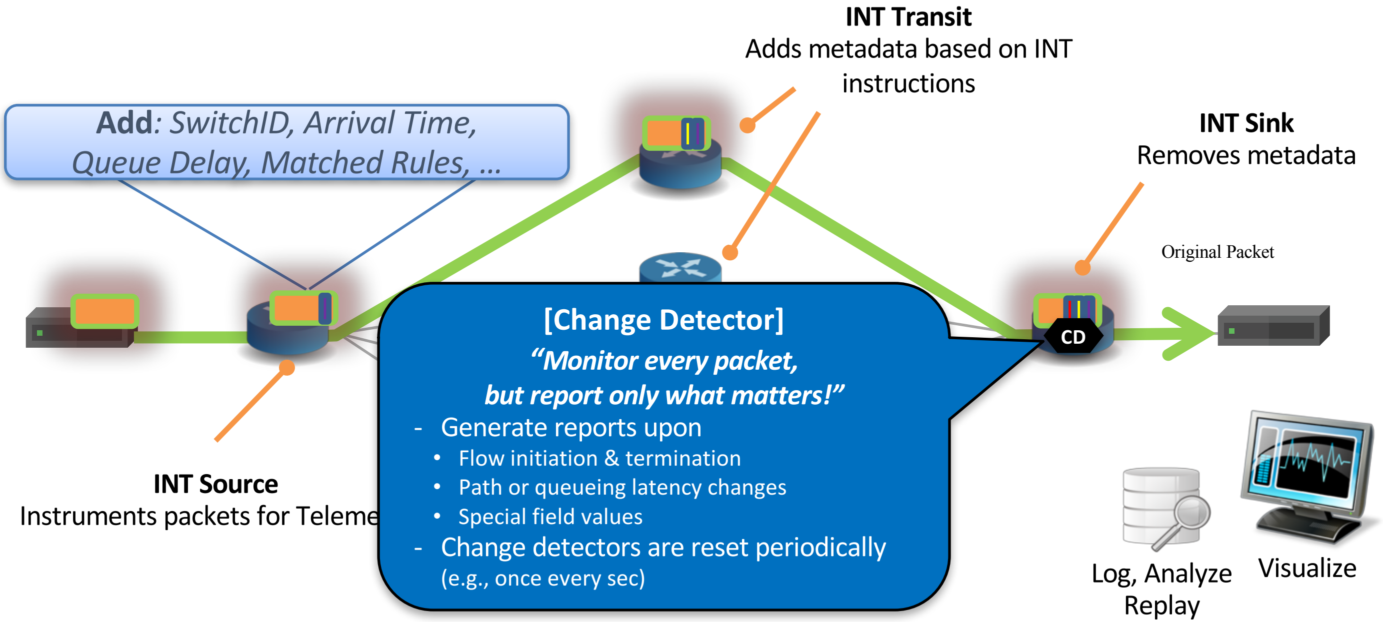
PISA + P4 can answer all four questions for the first time.
At full line rate. Without generating any additional packets!

DTEL: P4 library for data-plane telemetry

- ***"Track every flow"***: Flow Reporting
  - Monitor and report every flow's path and latency
  - Via end-to-end or hop-by-hop In-band Network Telemetry
- ***"Track every drop"***: Drop Reporting
  - Mirror every dropped packet along with the drop reason
- ***"Track every congestion"***: Congestion Reporting
  - Produce packet-level snapshots of a congested queue
  - Detect, characterize, and analyze microbursts

Flow Reporting: INT End-to-end Mode

- Leverages In-Band Network Telemetry (INT)
  https://github.com/p4lang/p4-applications/blob/master/docs/INT_v0_5.pdf

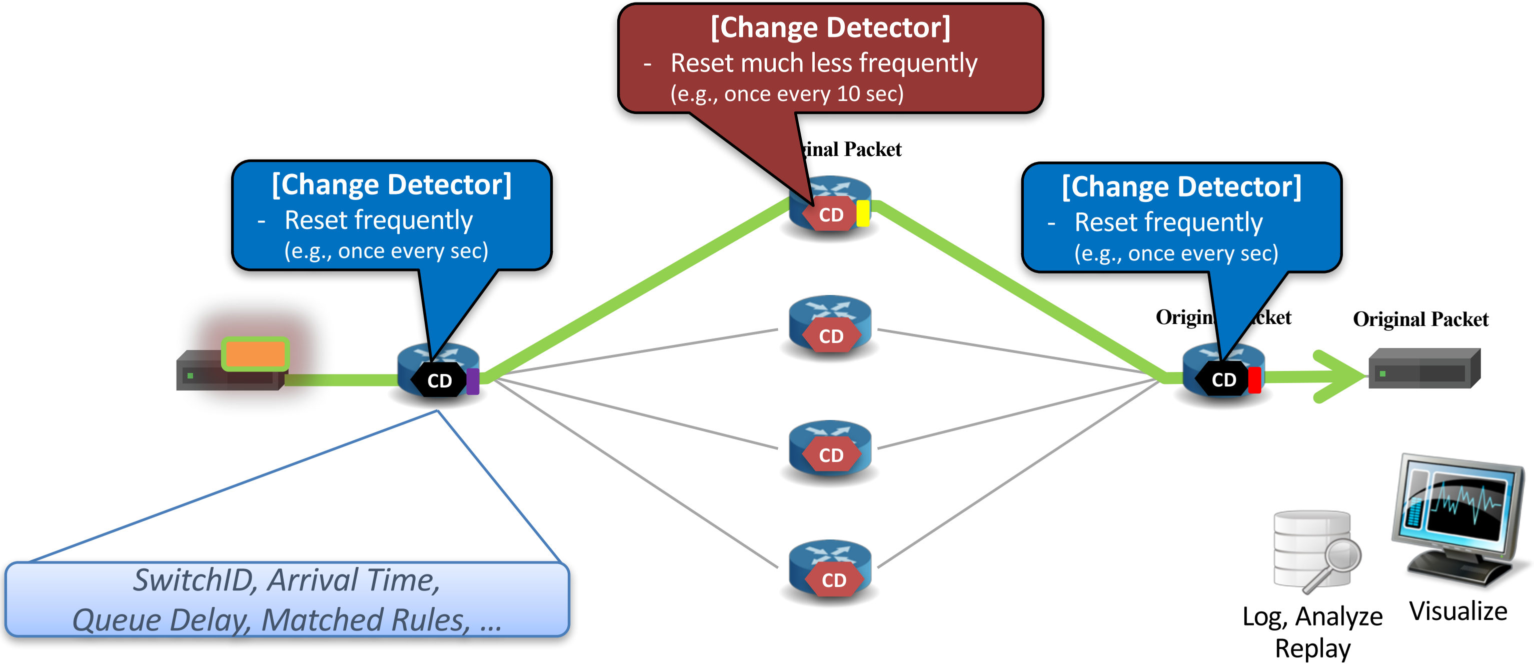


[Change Detector]

"Monitor every packet, but report only what matters!"

- Generate reports upon
  - Flow initiation & termination
  - Path or queueing latency changes
  - Special field values
- Change detectors are reset periodically (e.g., once every sec)

Flow Reporting: INT Hop-by-hop Mode



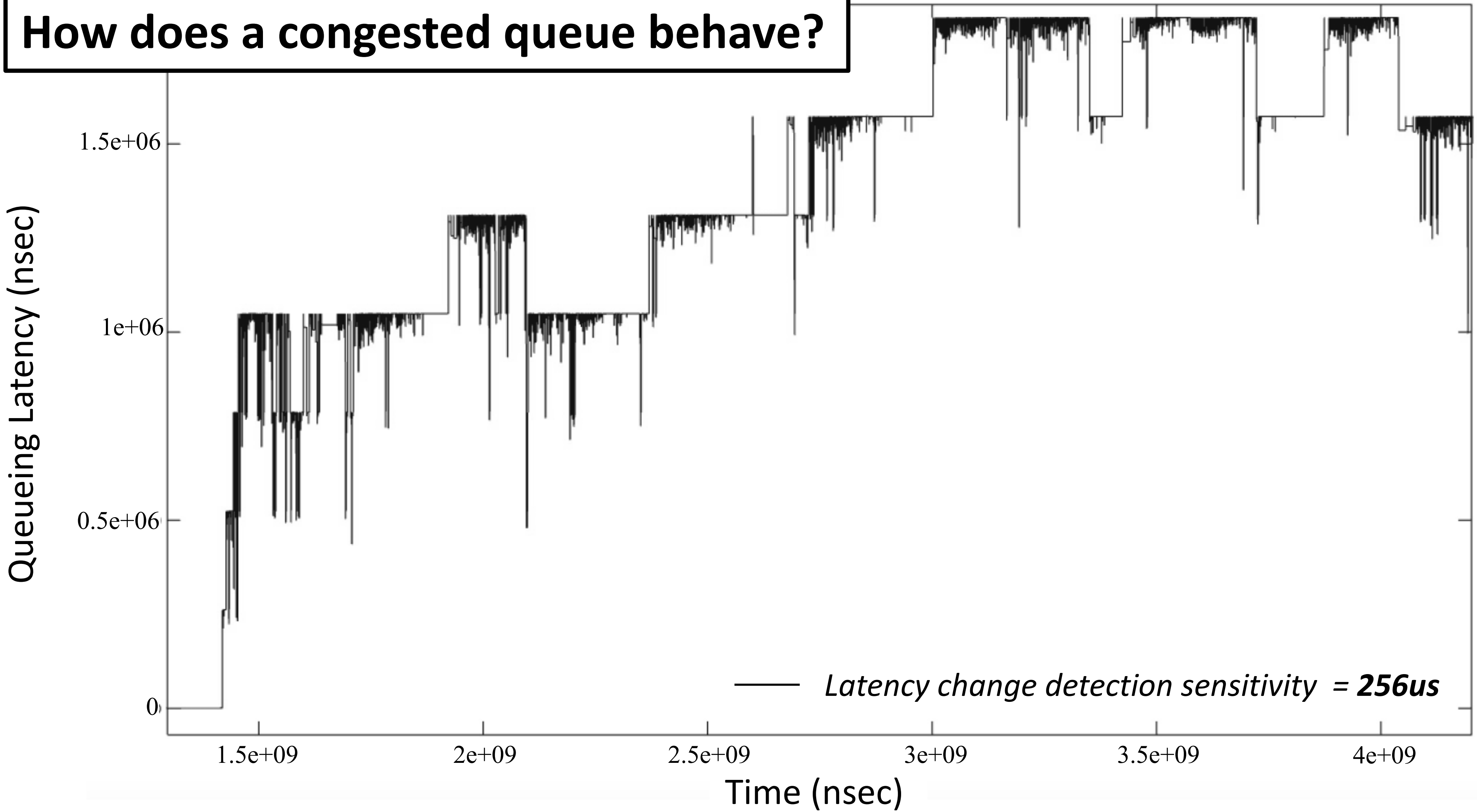
Flexibility matters

```
/* INT: add switch id */
action int_set_header_0() {
  add_header(int_switch_id_header);
  modify_field(int_switch_id_header.switch_id,
               global_config_metadata.switch_id);
}

/* INT: add ingress timestamp */
action int_set_header_1() {
  add_header(int_ingress_tstamp_header);
  modify_field(int_ingress_tstamp_header.ingress_tstamp,
               i2e_metadata.ingress_tstamp);
}

/* INT: add egress timestamp */
action int_set_header_2() {
  add_header(int_egress_tstamp_header);
  modify_field(int_egress_tstamp_header.egress_tstamp,
               eg_intr_md_from_parser_aux.egress_global_tstamp);
}
```

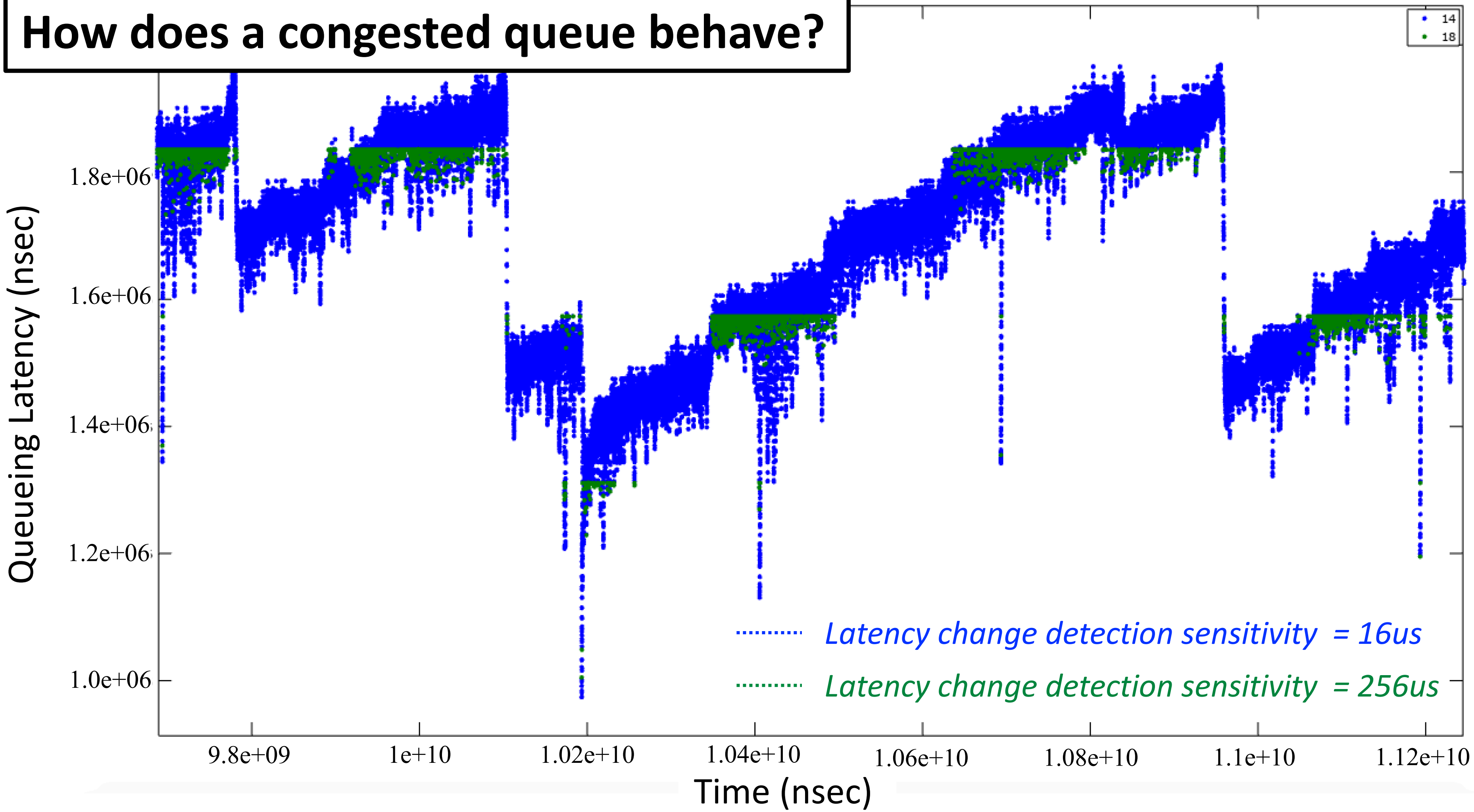
Programmable Telemetry

P4 code snippet: **switch ID**, **ingress** and **egress timestamp**

How does a congested queue behave?

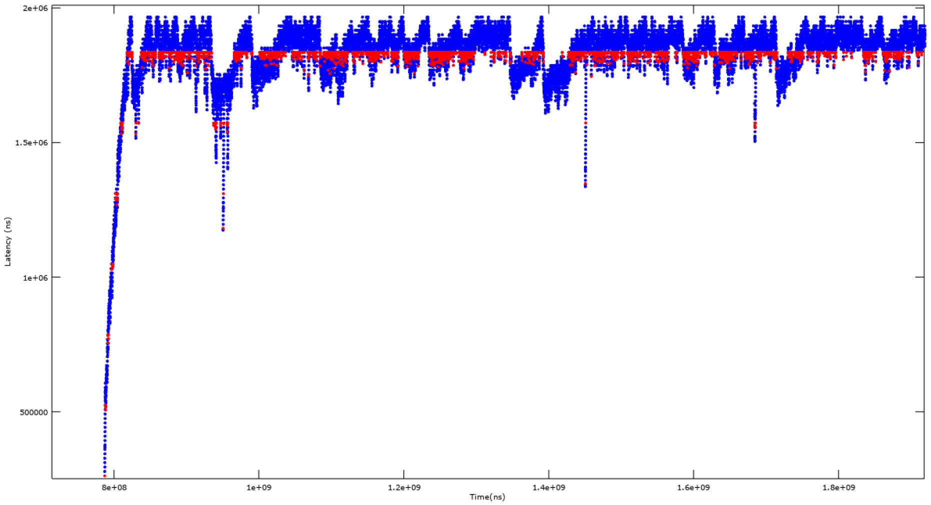


How does a congested queue behave?

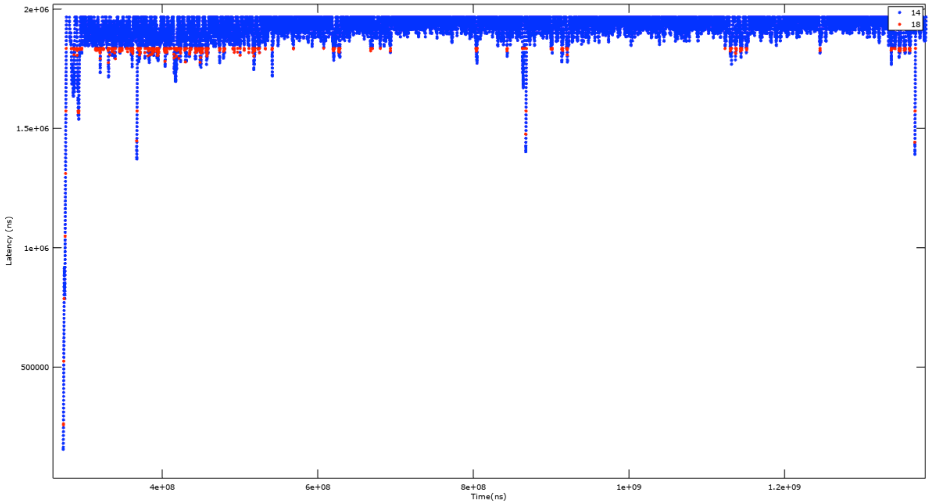


Results with more connections

25 connections



50 connections

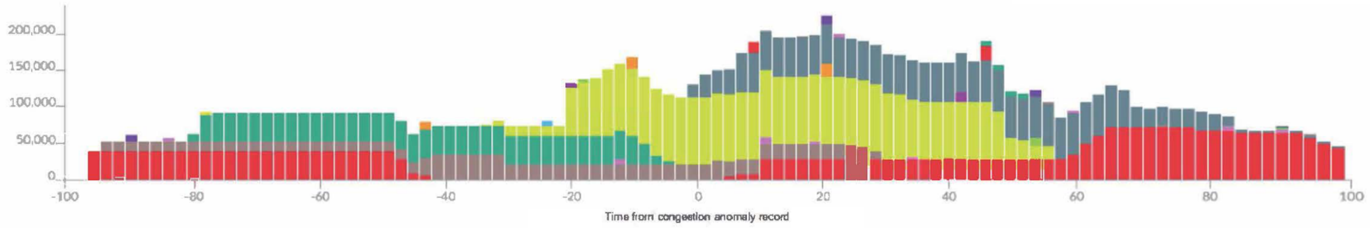


Visualizing microbursts (to the nanosecond)

Anomaly Records — BAREFOOT Deep Insight

Timestamp	Switch Id	Queue
July 25, 2017 - 18:17:51.513 UTC		

Queue Occupancy Over Time (bytes)



17 Affected Flows

Flow	kB in Queue	% of Queue Buildup	Packet Drops
10.32.2.2:46380 -> 10.36.1.2:5101 TCP	3282	29	0
10.32.2.2:46374 -> 10.36.1.2:5101 TCP	3073.5	27	25
10.32.2.2:46386 -> 10.36.1.2:5101 TCP	2092.5	18	27
10.32.2.2:46388 -> 10.36.1.2:5101 TCP	1456.5	13	0
10.32.2.2:46390 -> 10.36.1.2:5101 TCP	1227	11	36
10.32.2.2:46372 -> 10.36.1.2:5101 TCP	45	0	0
10.32.2.2:46392 -> 10.36.1.2:5101 TCP	37.5	0	39
10.35.1.2:34256 -> 10.36.1.2:5102 TCP	34.5	0	0

What does this mean?

- Switch can literally inspect every single packet and export just relevant information
- No loss of visibility due to sampling, probing, or control-plane-based polling
- Always-on drop, congestion, and flow tracking is possible
- This is already available on off-the-shelf brand-name switches
- Huge opportunities for Big-data processing and machine-learning experts

Smarter and Faster Congestion Notification

- Smarter congestion notification
  - Use more accurate and relevant congestion signals such as queue growth or decrease velocity (e.g., direction, not just values), fair rates for flows, etc.
  - Use additional congestion information such as app-pool-level queue occupancy
  - Differentiated treatment for lossy and lossless traffic to improve fairness between the two traffic classes

- Faster congestion notification
  - Use early indicators of congestion such as dequeue-time queue depth, link utilization
  - Directly generate congestion notifications (e.g., CNPs) for heavy hitters
    - Up to O(10^3) ~ O(10^4) heavy flows

Smarter Way of Reacting To Congestion

- Dynamic, congestion-aware load balancing
  - Path-level (global) or hop-level (local) congestion-aware next-hop resolution
  - Offered with flowlet switching, addressing out-of-order delivery upon path changes
  - E.g., HULA prototypes available. CONGA is feasible.

- Smart and safe pausing
  - Rich PFC statistics and anomaly (e.g., PFC deadlock) detection
  - Congestion isolation; having upstream device “surgically” pause or rate-limit heavy flows (similar to IEEE 802.1Qcz)

- Enhanced in-cast mitigation
  - Burst absorption via “packet parking”; use generic external memory (DRAM/HBM) as temporary packet buffers accessible via data path (Ethernet)

Want to find more resources or follow up?

- **Technical References**
  - In-band Network Telemetry (P4.org App WG)
    https://github.com/p4lang/p4-applications/tree/master/docs
  - Telemetry Report Format Specification (P4.org App WG)
    https://github.com/p4lang/p4-applications/tree/master/docs
  - In-situ OAM (IETF)
    https://tools.ietf.org/html/draft-brockners-inband-oam-data-07
  - Cisco Nexus 34180, INT Configuration Guide
    https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus3000/sw/programmability/9_x/b_Cisco_Nexus_3000_Series_NX-OS_Programmability_Guide_9x/b_Cisco_Nexus_3000_Series_NX-OS_Programmability_Guide_9x_chapter_011110.pdf