Network Measurement

Changhoon Kim
chang@barefootnetworks.com

CS344, Spring 2019

**Courtesy Jennifer Rexford at Princeton University for several slides*

“If you can’t measure it, you can’t improve it”

-- Peter Drucker

“There are two possible outcomes. If the result confirms the hypothesis, then you’ve made a measurement; if the result contradicts the hypothesis, you’ve made a discovery.”

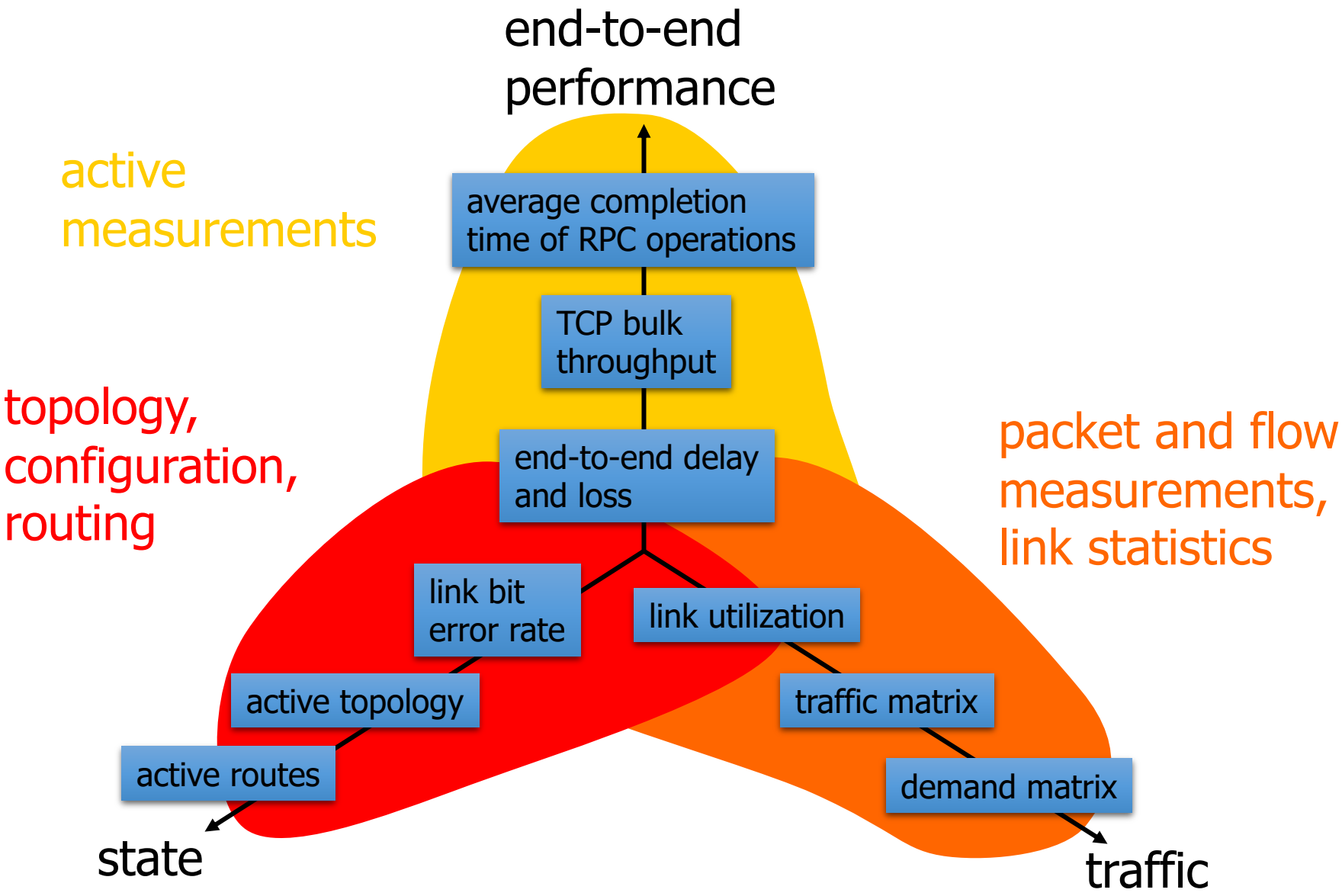
-- Enrico Fermi

Why Measure The Network?

- Network operation
  - Billing customers
  - Checking up on quality of service (QoS)
  - Detecting, diagnosing, and fixing problems
  - Planning outlay of new equipment and infra

- Scientific discovery
  - Characterizing traffic, topology, performance
  - Understanding performance and dynamics of new ideas

Taxonomy of Network Measurement – Take A



Taxonomy of Network Measurement – Take B

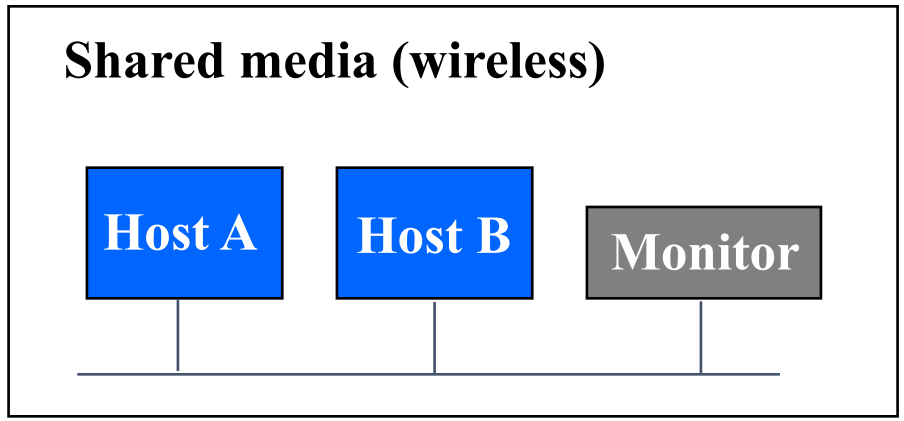
	Control Plane	Data Plane
	Topology, Configuration, Routing	End-to-end performance, Link statistics, Packets & flows and their attributes
Active	E.g., Route injection	E.g., Ping, Traceroute, iPerf
Passive	E.g., Routing table dumps, link-state monitoring, BGP update monitoring	E.g., mirroring, NetFlow, sFlow, heavy-hitter detection, tcpdump, TCP e-stats

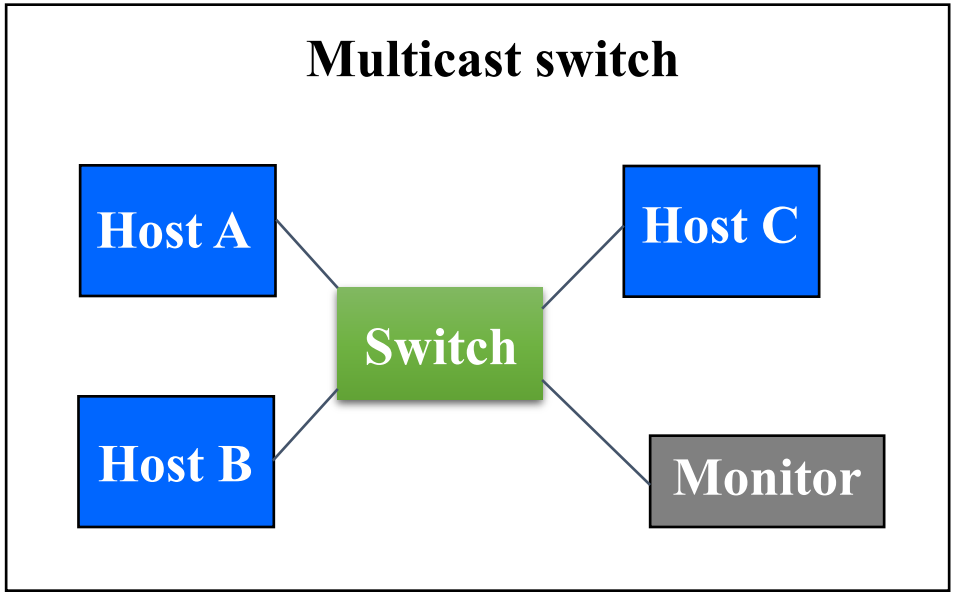
Packet Measurement

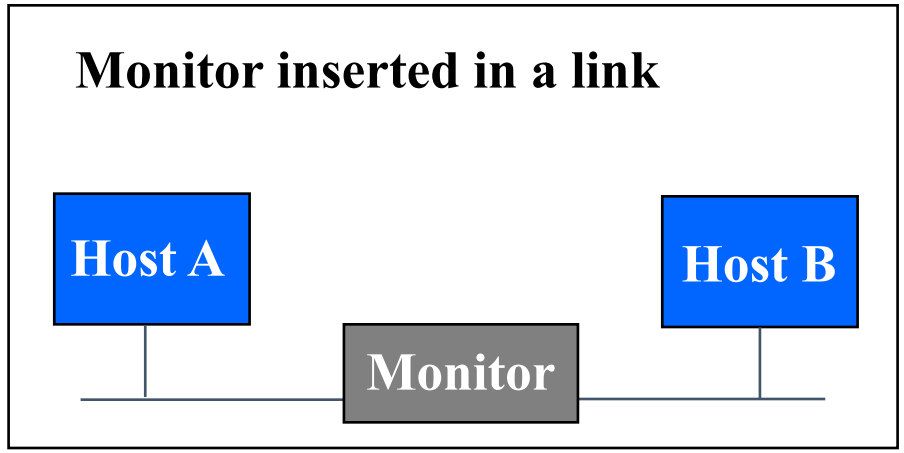
Packet Monitoring

- Definition
  - Passively collecting IP packets on one or more links
  - Recording IP, TCP/UDP, or application-layer traces
- Scope
  - Fine-grain information about user and application behavior
  - Passively monitoring the network infrastructure
  - Characterizing traffic and diagnosing problems

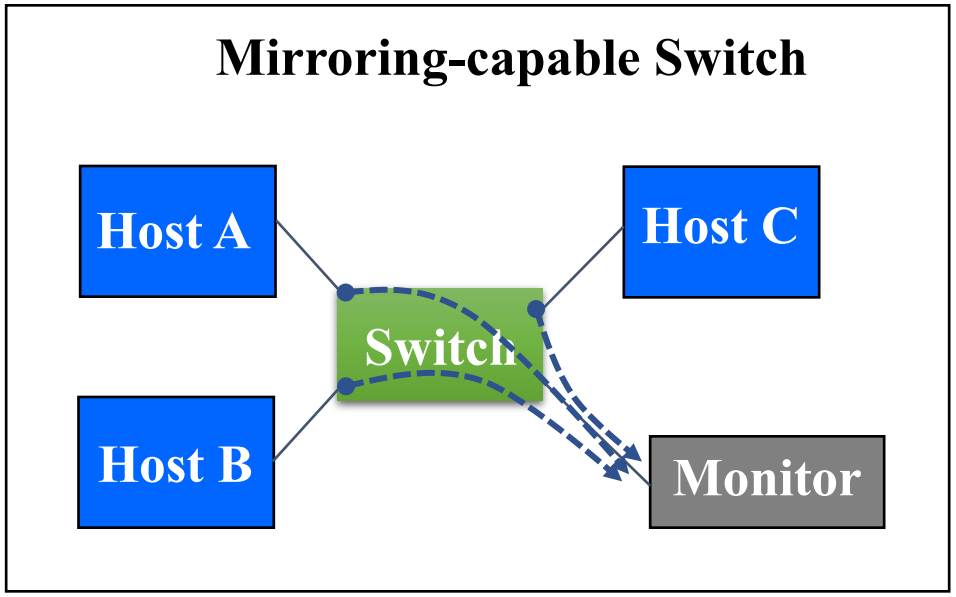
Monitoring a Link

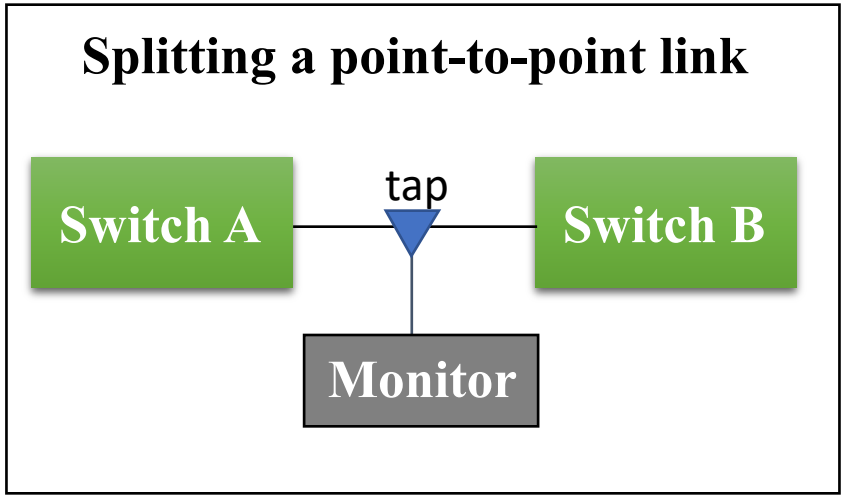


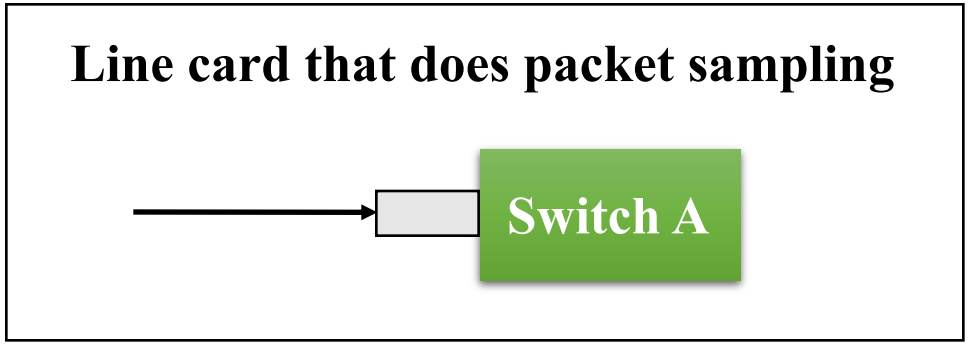




Monitoring a Link – cont'd







Selecting the Traffic

- Filter to focus on a subset of the packets
  - IP addresses/prefixes (e.g., to/from specific sites)
  - Protocol (e.g., TCP, UDP, or ICMP)
  - Port numbers (e.g., HTTP, DNS, BGP, Napster)

- Collect first *n* bytes of packet (snap length)
  - Medium access control header (if present)
  - IP header (typically 20 or 36 bytes)
  - IP+UDP header (typically 28 or 44 bytes)
  - IP+TCP header (typically 40 or 56 bytes)
  - Application-layer message (entire packet)

Analysis of IP Header Traces

- Source/destination addresses
  - Identity of popular servers & heavy apps/users
- Distribution of packet delay through the router
  - Identification of typical delays and anomalies
- Distribution of packet sizes
  - Workload models for routers
- Burstiness of the traffic on the link over time
  - Provisioning rules for allocating link capacity and queue sizes
- Throughput between pairs of src/dest addresses
  - Detection and diagnosis of performance problems

TCP Header Analysis

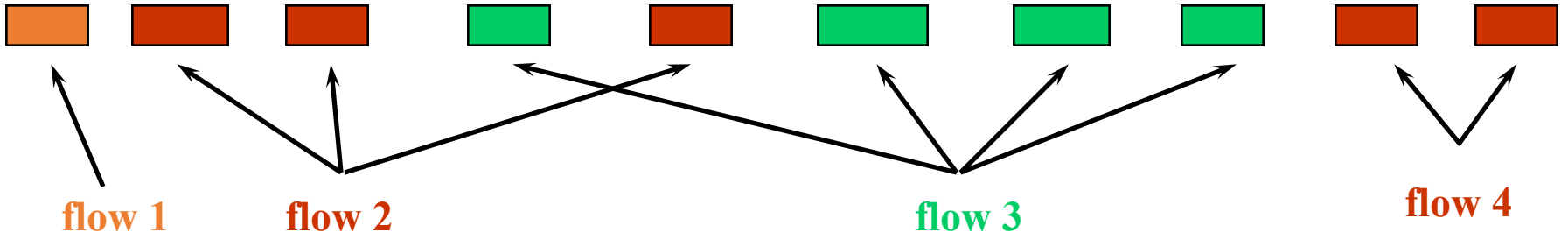
- Source and destination port numbers
  - Popular applications; parallel connections
- Sequence/ACK numbers and packet timestamps
  - Out-of-order/lost packets; throughput and delay
- Number of packets/bytes per connection
  - RPC/message transfer sizes; frequency of bulk transfers
- SYN flags from client machines
  - Unsuccessful requests; denial-of-service attacks
- FIN/RST flags from client machines
  - Frequency of Web transfers aborted by clients; application behavior

Packet Contents

- Application-layer header
  - HTTP request and response headers
  - SMTP commands and replies
  - DNS queries and responses; OSPF/BGP messages
- Application-layer body
  - HTTP resources (or checksums of the contents)
  - RPC (e.g., Thrift) messages, key-value caching operations, pub-sub messages
- This is getting harder due to end-to-end encryption

Flow Measurement

IP Flows



- Set of packets that "belong together"
  - Source/destination IP addresses and port numbers
  - Same protocol, ToS bits, …
  - Same input/output interfaces at a router (if known)

- Packets that are "close" together in time
  - Maximum spacing between packets (e.g., 30 sec)
  - E.g.: flows 2 and 4 are different flows due to time

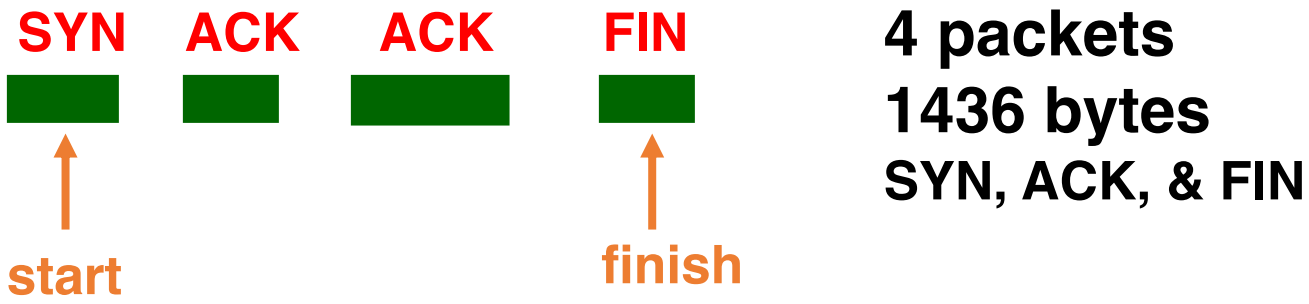
Flow Abstraction

- Not exactly the same as a “session” or “message”
  - Sequence of related packets may be multiple flows
  - Related packets may not follow the same links
  - “Message” or “session” is hard to measure from inside network
- Motivation for this abstraction
  - As close to a meaningful application-level traffic unit as possible from inside
  - Router optimization for forwarding/access-control
  - … might as well throw in a few counters

Traffic Statistics (e.g., Netflow)

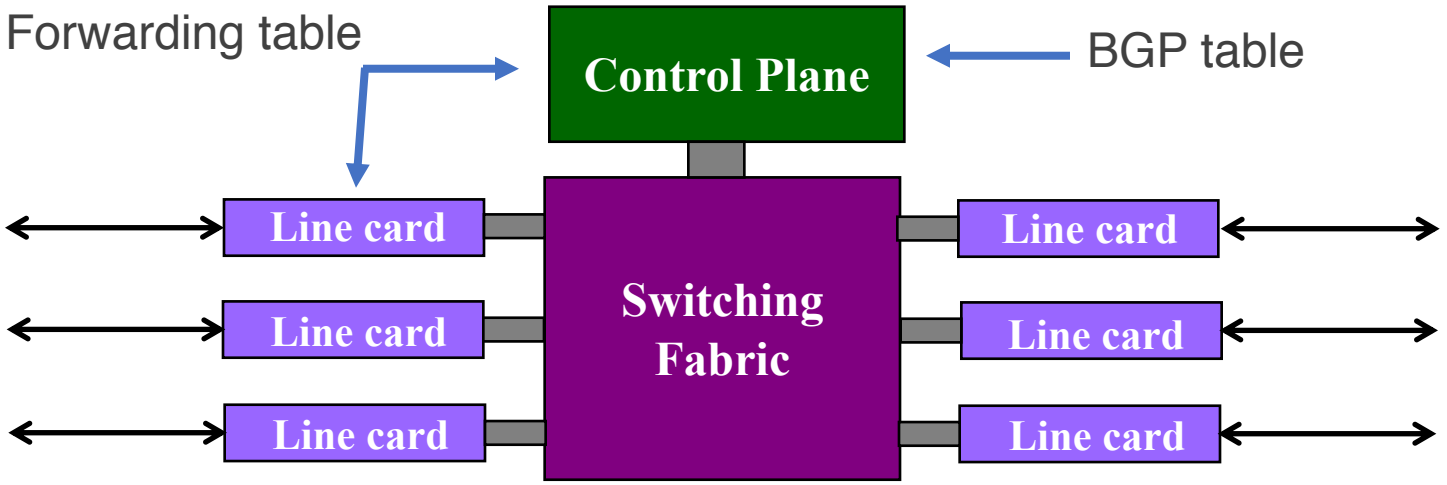
- Packet header info
  - Source and destination addresses and port #s
  - Other IP & TCP/UDP header fields (protocol, ToS)

- Aggregate traffic information
  - Start and finish time (time of first & last packet)
  - Total # of bytes and number of packets in the flow
  - TCP flags (e.g., logical OR over sequence of packets)



Recording Routing Information

- Input and output interfaces
  - Input interface is where packets entered the router
  - Output interface is “next hop” in forwarding table
- Source and destination IP prefix (mask length)
  - Longest prefix match on src and dest IP addresses

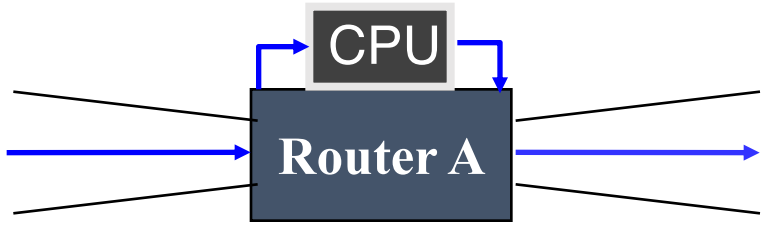


Packet vs. Flow Measurement

- Basic statistics (available from both techniques)
  - Traffic mix by IP addresses, port numbers, protocol
  - Average packet size
- Traffic over time
  - Both: traffic volumes on medium-to-large time scale
  - Packet: burstiness of the traffic on a small time scale
- Statistics per L4 connection
  - Both: volume of traffic transferred over the link
  - Packet: frequency of lost or out-of-order packets

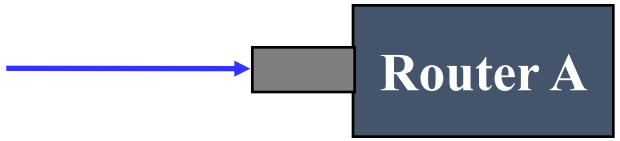
Collecting Flow Measurements

Route CPU that generates flow records



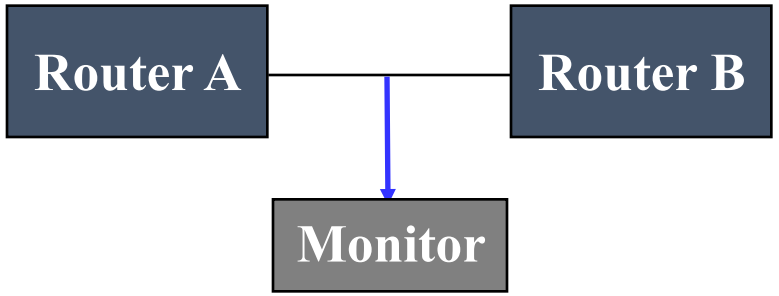
…may degrade forwarding performance

Line card that generates flow records



…more efficient to support measurement in each line card

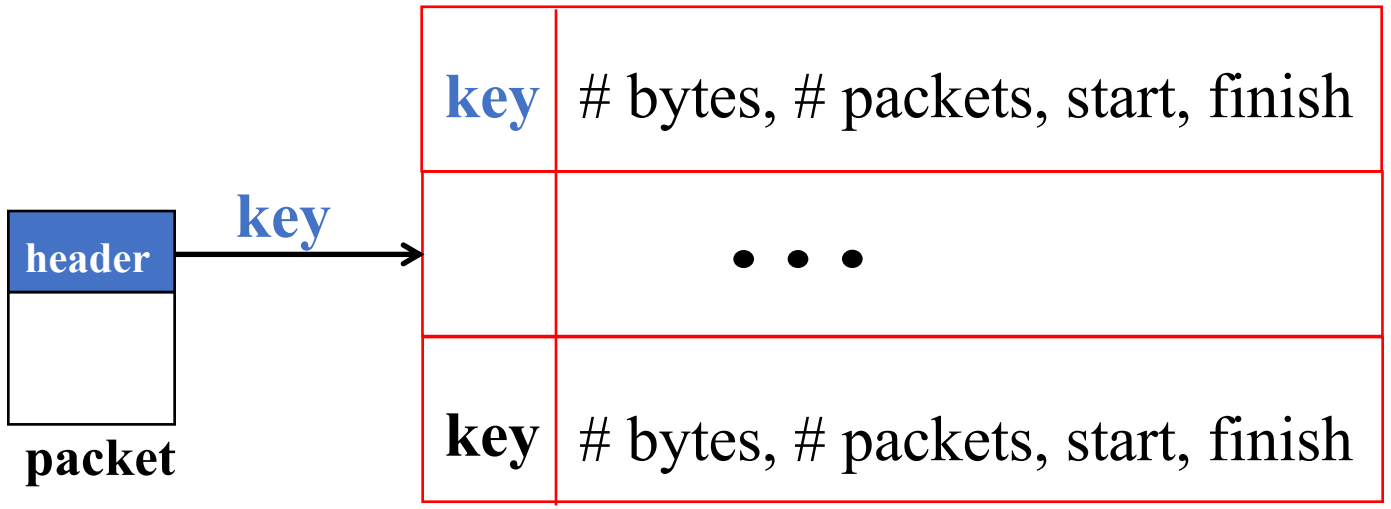
Packet monitor that generates flow records



…third party

Mechanics: Flow Cache

- Maintain a cache of active flows
  - Storage of byte/packet counts, timestamps, etc.
- Compute a key per incoming packet
  - Concatenation of source, destination, port #s, etc.
- Index into the flow cache based on the key
  - Creation or updating of an entry in the flow cache



Mechanics: Evicting Cache Entries

- Flow timeout
  - Remove flows not receiving a packet recently
  - Periodic sequencing to time out flows
  - New packet triggers the creation of a new flow
- Cache replacement
  - Remove flow(s) when the flow cache is full
  - Evict existing flow(s) upon creating a cache entry
  - Apply eviction policy (LRU, random flow, etc.)
- Long-lived flows
  - Remove flow(s) persisting a long time (e.g., 30 min)

Measurement Overhead

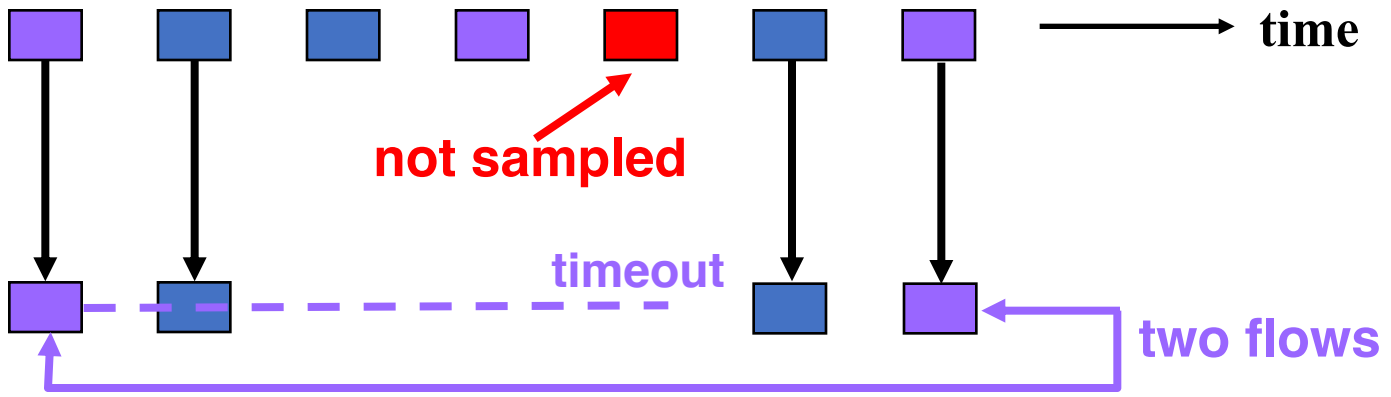
- Per-packet overhead
  - Computing the key and indexing flow cache
  - More work when the average packet size is small
  - May not be able to keep up with the link speed

- Per-flow overhead
  - Creation and eviction of entry in the flow cache
  - Volume of measurement data (# of flow records)
  - Larger # of flows when # of packets per flow is small
  - May overwhelm system collecting/analyzing data

Sampling: Packet Sampling

- Packet sampling before flow creation
  - 1-out-of-m sampling of individual packets
  - Create of flow records over the sampled packets

- Reducing overhead
  - Avoid per-packet overhead on (m-1)/m packets
  - Avoid creating records for many small flows



Advanced Topics and Recent Trends

More on Sampling …

- Sample all packets in a random subset of flows without maintaining any flow state in a device?
  - Apply hashing to 5 tuple of packets and sample those with certain hash results
- Sample packets or flows consistently across multiple hops?
  - Apply hashing only to invariant header fields (e.g., 5 tuple + IP identification, 5 tuple + TCP flag)
  - Use the same hash function and selection rule at every hop
- Sample just the first packet of each flow?
  - Use Bloom filter

Challenges of Flow Measurement Today

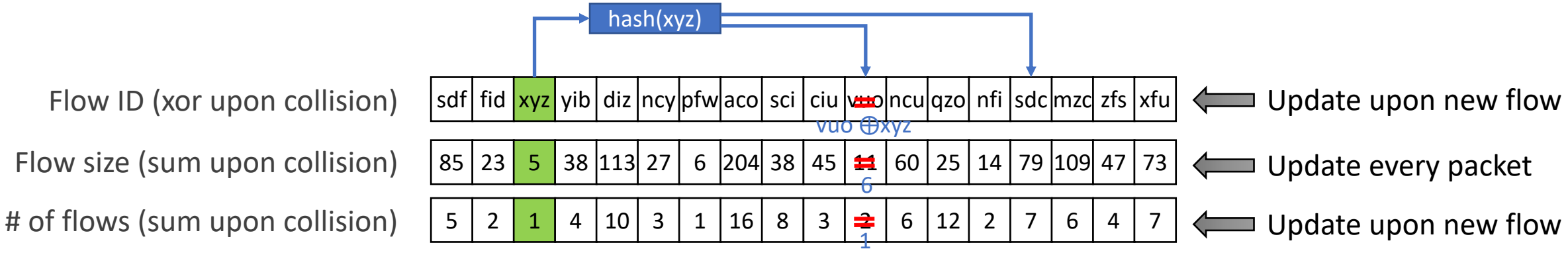
- Very high speed
  - 12.8Tbps or 6Bpps per switching chip (ASIC)
  - Even with four parallel pipelines (cores), one pipeline needs to handle 1.5Bpps; process one packet every 670 psecs!
  - DRAM read or write takes tens of nsecs; SRAM is the only viable option
- Limited amount of on-chip SRAM
  - Up to a few tens of MBs, shared with tables for various other features
- Hence, the followings have become very hard, motivating innovations
  - Flow cache – i.e., an exact match table with line-rate entry insertion
  - A large number of counters (e.g., per-flow counters)

Advanced Flow Measurement: Marple [Sigcomm'17]

- Real flow cache processing un-sampled traffic is too expensive; can we use a cheaper data structure?
  - A small hash table indexed by flow hash
  - Collisions will happen frequently
- Evict an entry upon collision
  - Evicted flow's information is delivered to local or remote monitor
  - Data-plane-based eviction is feasible, trading bandwidth for monitoring
- Partial flow statistics can be safely merged in many cases
  - Merge-able statistics include addition, max, min, set union, set intersection, exponentially-weighted moving average (e.g., rates), etc.
  - "Linear-in-state" statistics are provably merge-able

Advanced Flow Measurement: FlowRadar [NSDI'16]

- Again, let's just use a small hash table indexed by flow hash
- Embrace collisions; keep encoding values of colliding flows
  - Keep three hash tables (for flow ID, flow size, # of flows)
  - Produce *n* hashes for each packet and update *n* locations in each hash table
  - Periodically dump the hash tables to the controller
  - Controller decodes the hash-table contents in a "zig-zag" fashion
  - Decoding the hash-table contents at the network level works even better



Advanced Ideas for Keeping Counters

- Shallow counters on SRAM, backed by deep counters on DRAM
  - Occasionally export counter values from SRAM to DRAM using clever techniques

- Worth reading
  - CMA [IEEE Micro'02]
  - LR(t) [Sigmetrics'03]
  - Counter Braids [Sigmetrics'08]

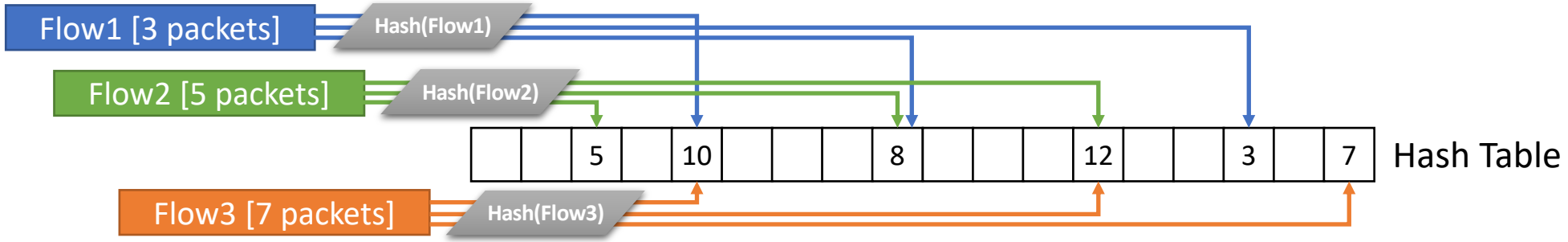
Getting Approximate Statistics

- Precise statistics are often too expensive and hard to get by; approximate answers are acceptable for many questions, such as …
  - *“How many unique flows are there?”*
  - *“Which are the top-N flows or elephant flows?”*
  - *“Which are top-N spreaders?”*
  - *“How varied are certain header fields?”*
  - *“What does the flow-size distribution look like?”*

- Sketch
  - A compact, fixed-size data structure that can summarize traffic statistics while bounding an error margin

Example Sketches

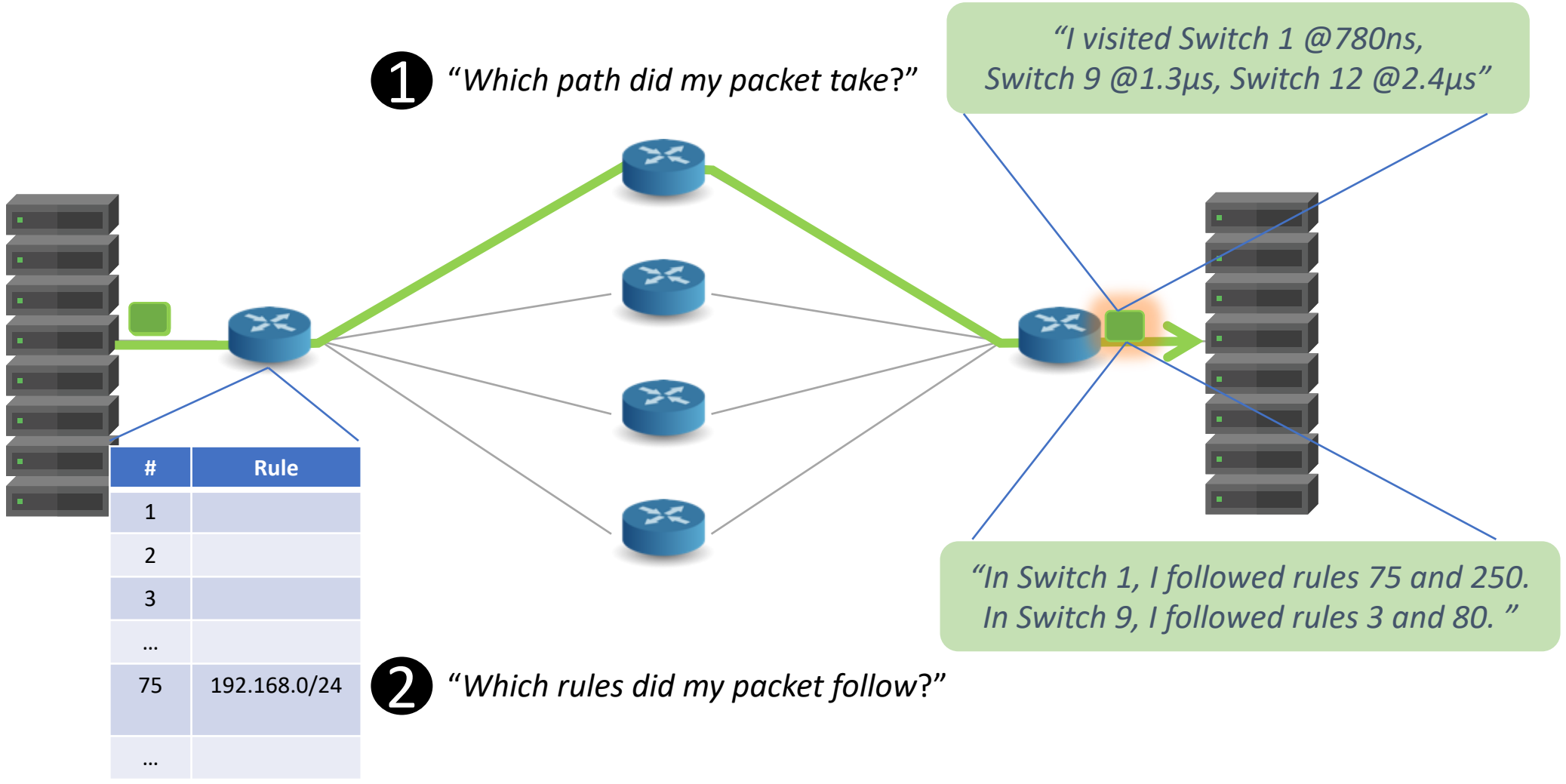
- CountMin sketch for elephant-flow detection



Size of flow1 = min(10, 8, 3) = 3, Size of flow2 = min(5, 8, 12) = 5, Size of flow3 = min(10, 12, 7) = 7

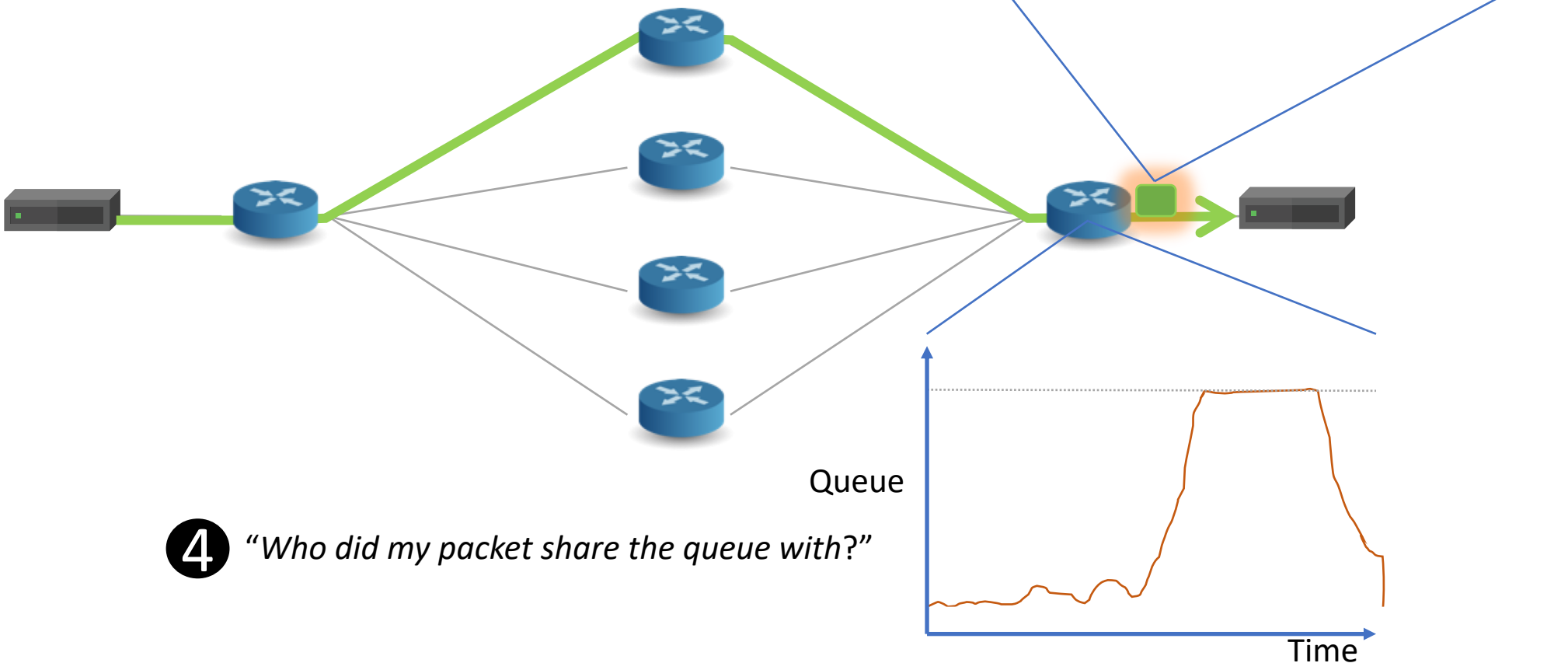
- Hyper-loglog sketch for approximate cardinality counting
  - *"To observe a very rare event, you must have had a lot of samples"*
  - Hash 5 tuple of each packet
  - Track the max number of leading zeros observed in the hash results
  - A larger number of leading zeros means "statistically" more flows
  - E.g., 10 leading zeros ~= 2^{11} flows

Data-plane Telemetry


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. "

❸ *"How long did my packet queue at each switch?"*

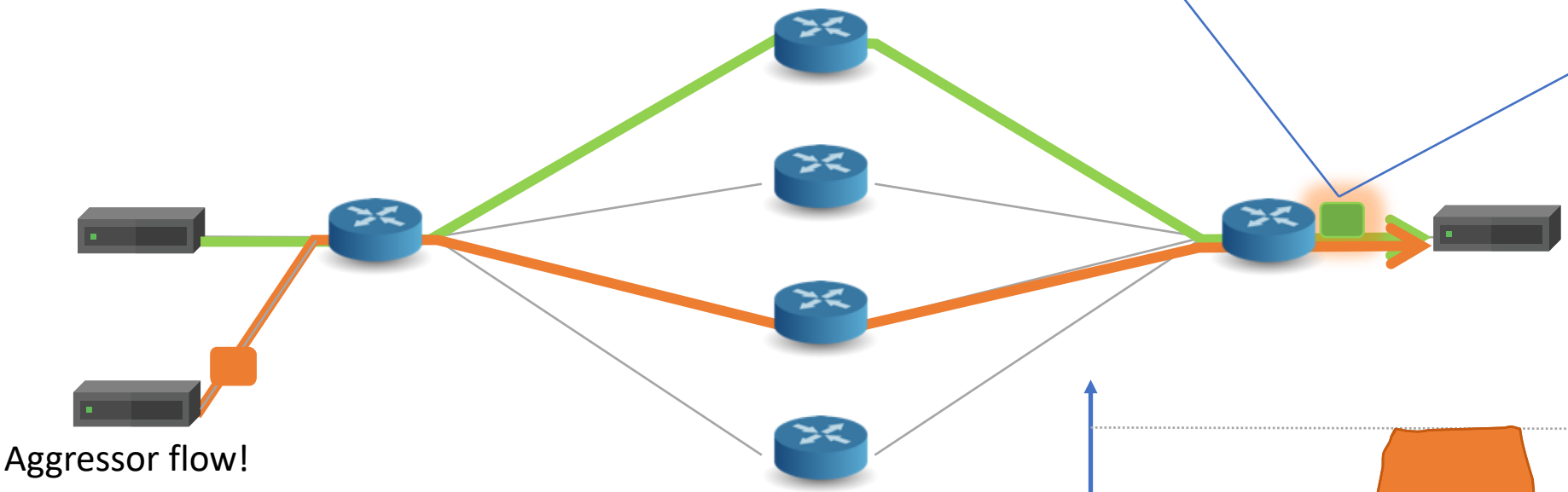
"Delay: 100ns, 200ns, 19740ns"



❹ *"Who did my packet share the queue with?"*

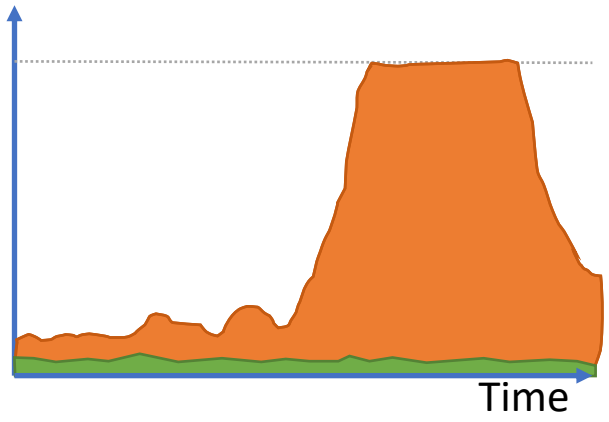
❸ *"How long did my packet queue at each switch?"*

"Delay: 100ns, 200ns, 19740ns"



Aggressor flow!

❹ *"Who did my packet share the queue with?"*



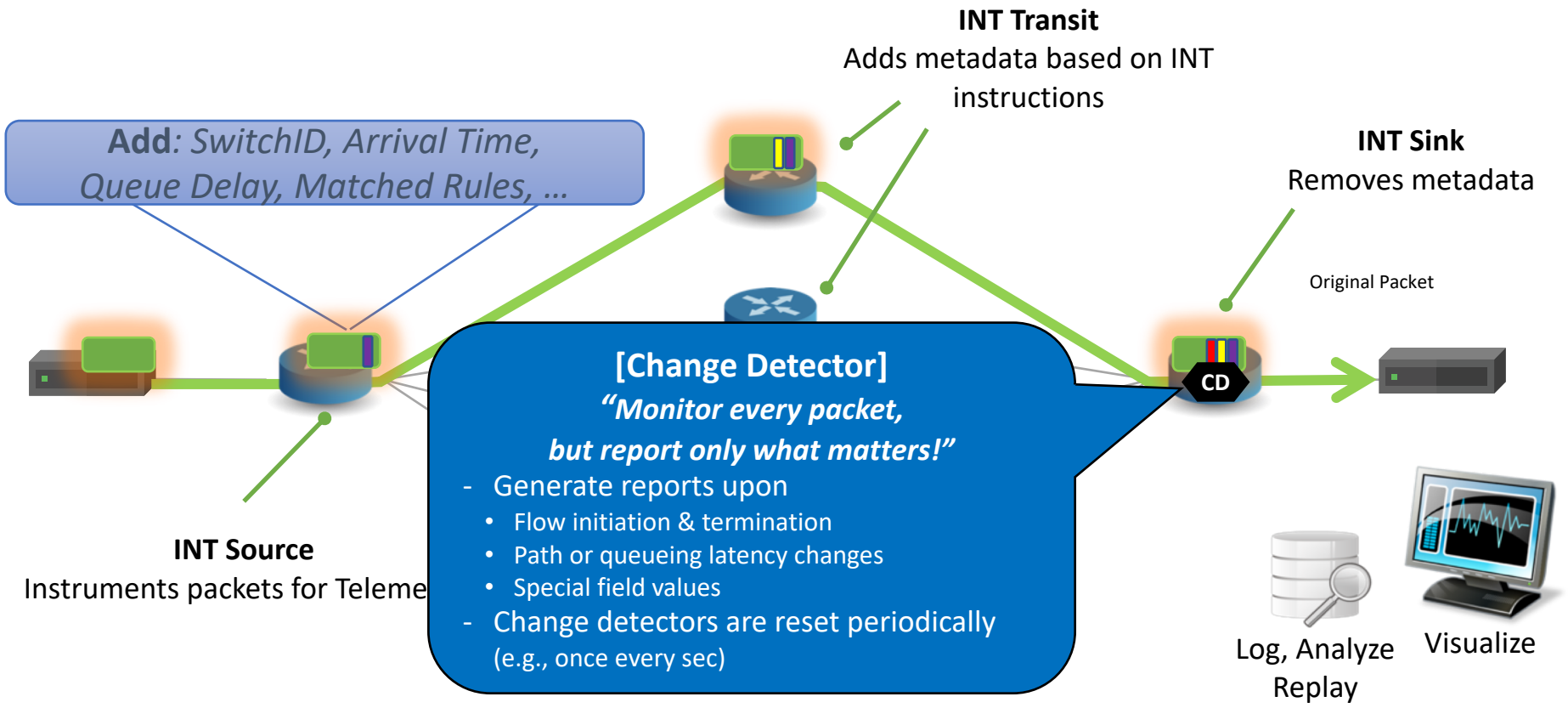
The network should answer these questions

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?"*

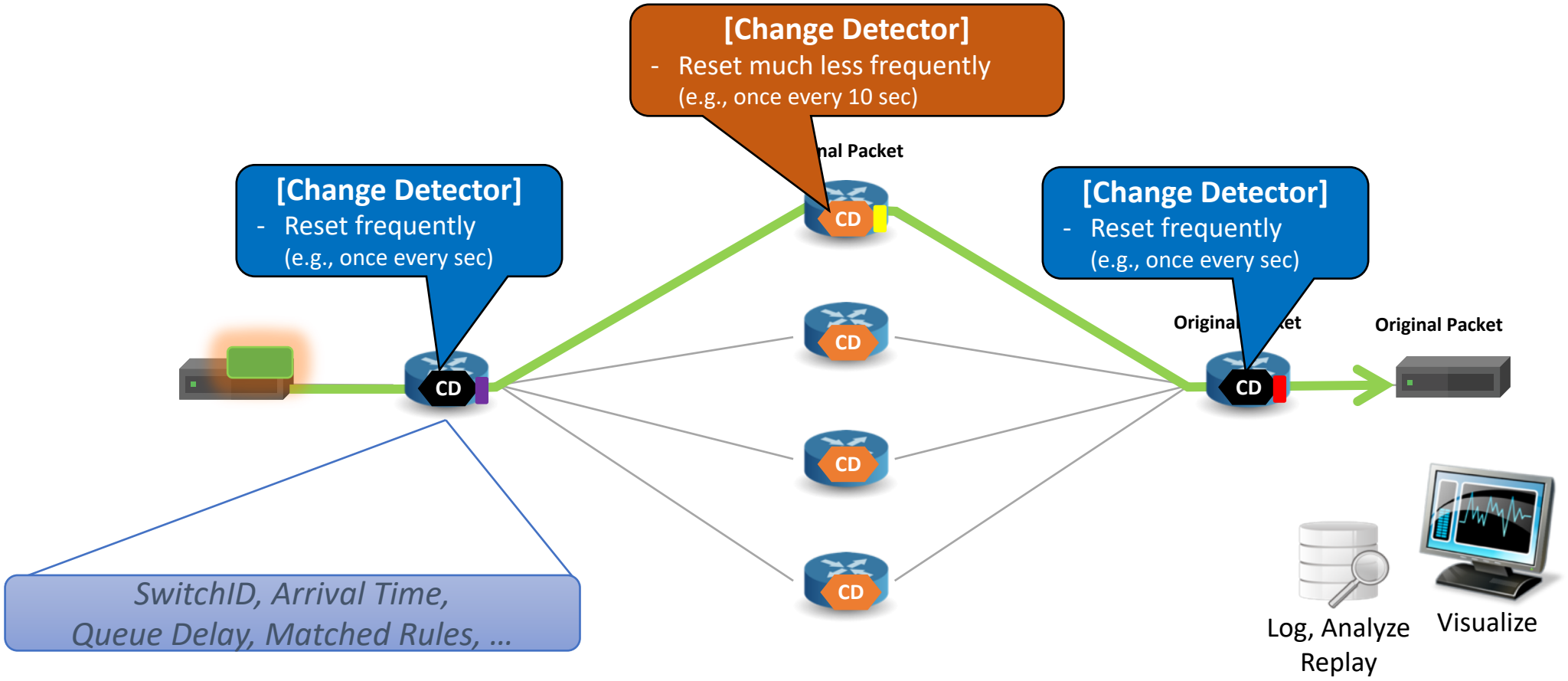
Programmable data planes can answer all four questions at full line rate, without impacting performance!

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



Flow Reporting: INT Hop-by-hop Mode



Flexibility matters

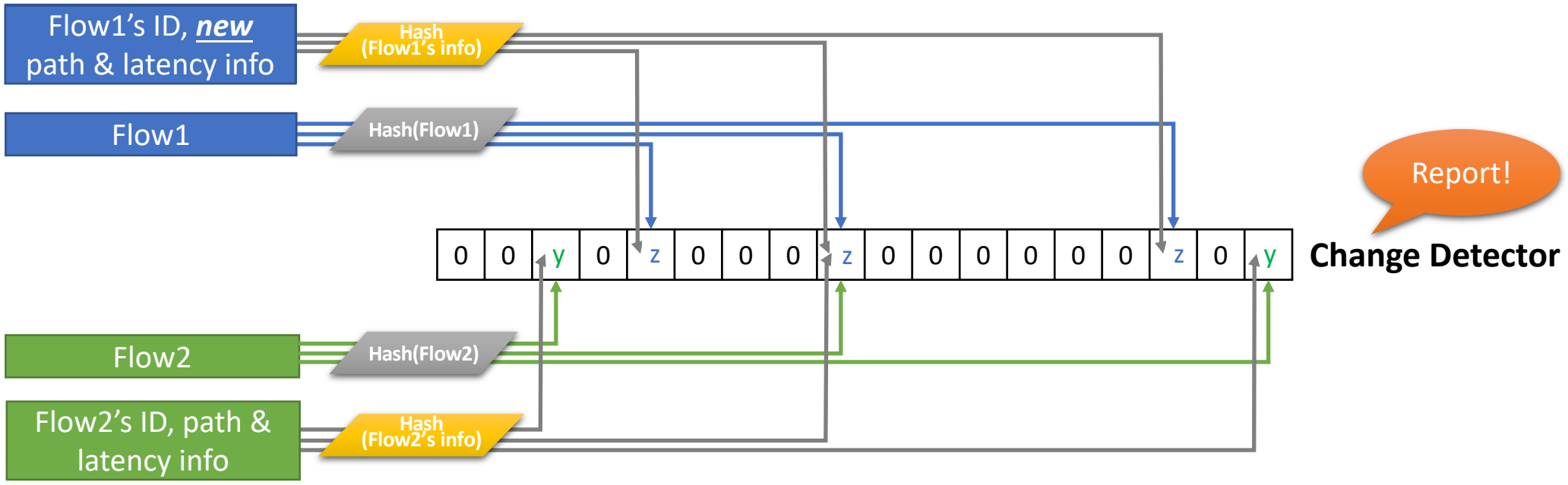
```
/* INT: add switch id */
action int_set_header_0() {
  add_header(int_switch_id_header);
  modify_field(int_switch_id_header.switch_id,
               global_config_metadata.switch_id);
}

/* 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

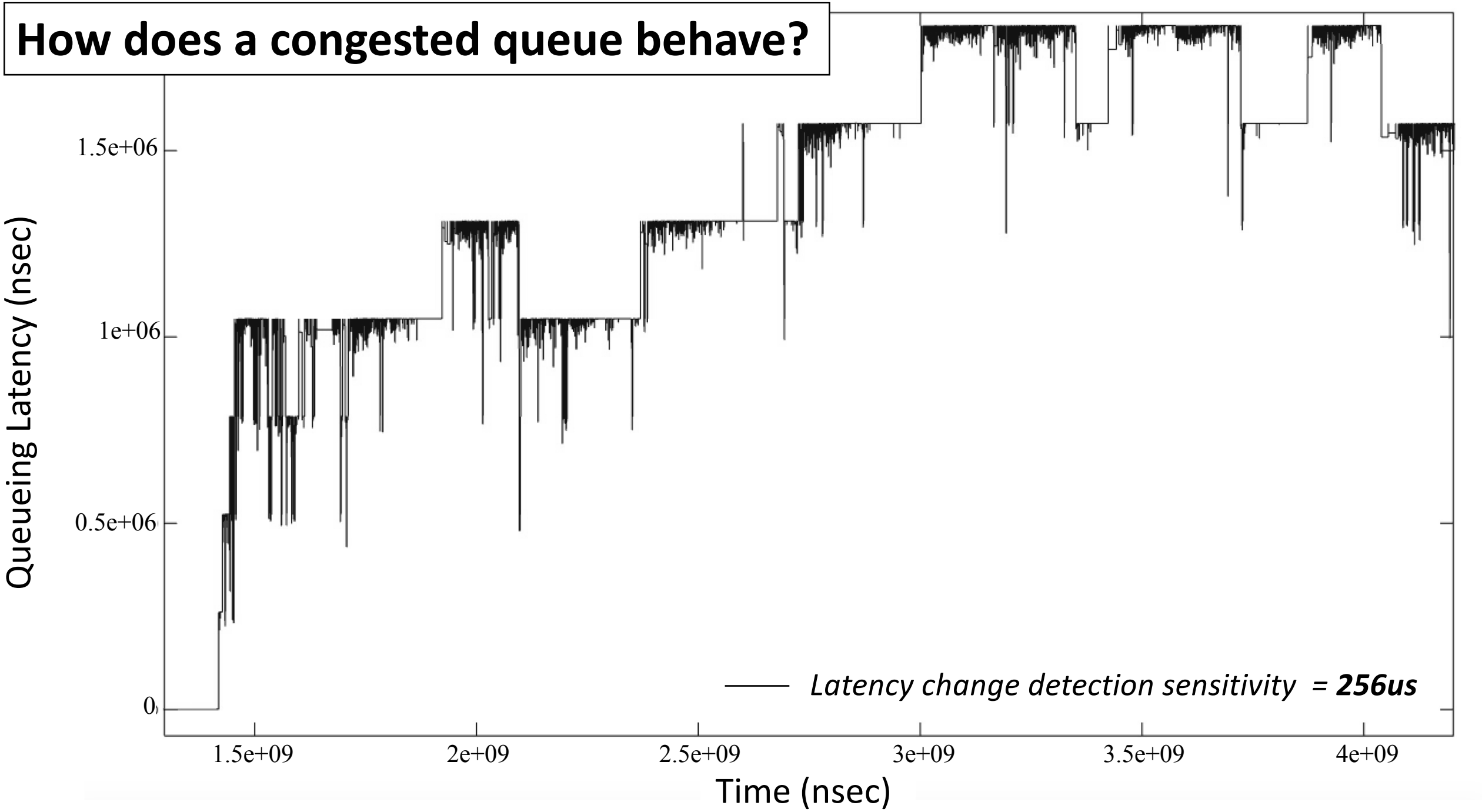
P4 code snippet: **switch ID**, **ingress** and **egress timestamp**

Change Detector

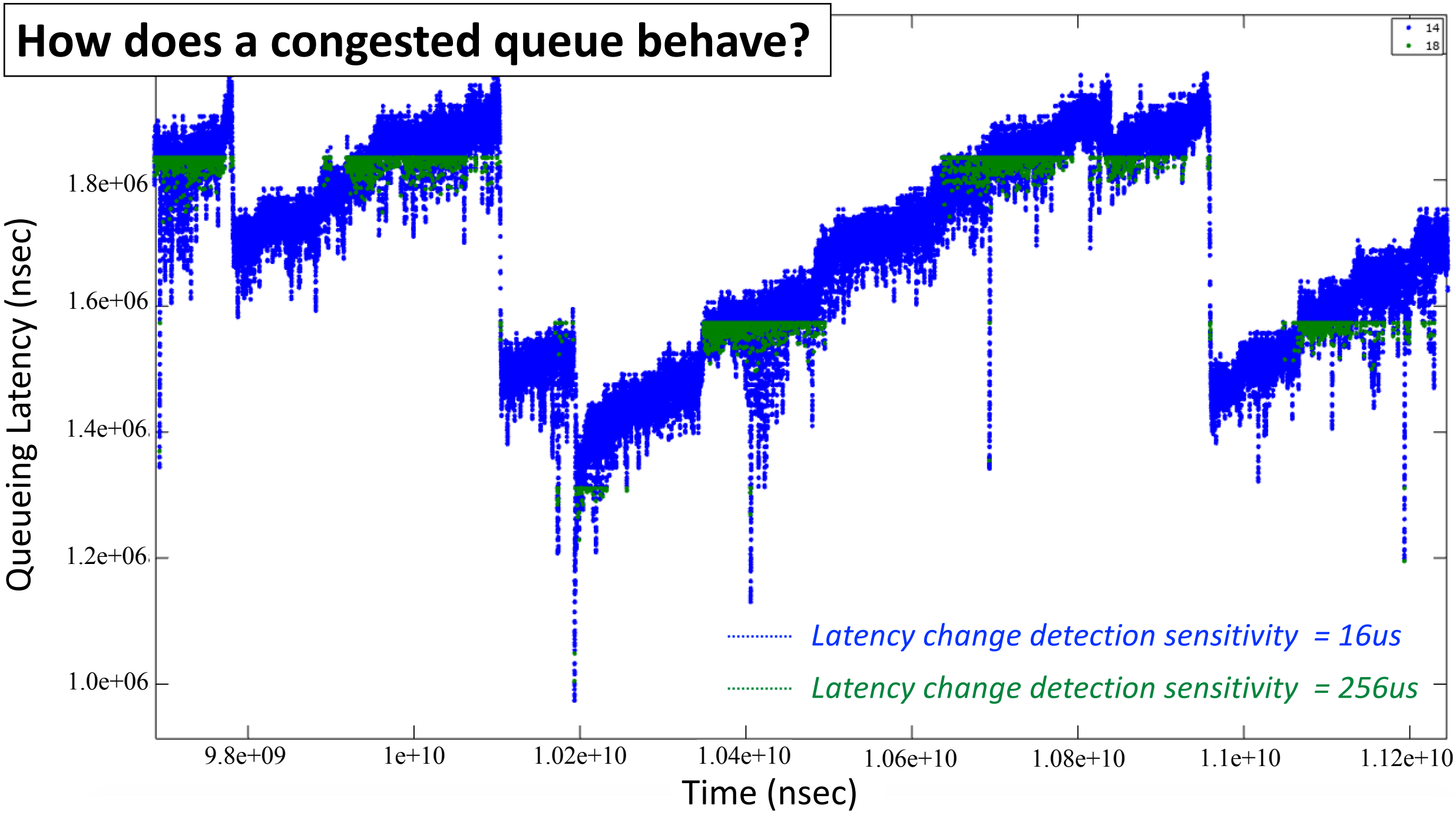
- Inspect every packet, and report only relevant ones
  - Keep the hash of flow into (ID, path, latency) in the hash table
  - Report when at least one of the cell values differ
  - Very low false negatives, reasonably low false positives when # of flows is smaller than the soft capacity of change detector



How does a congested queue behave?

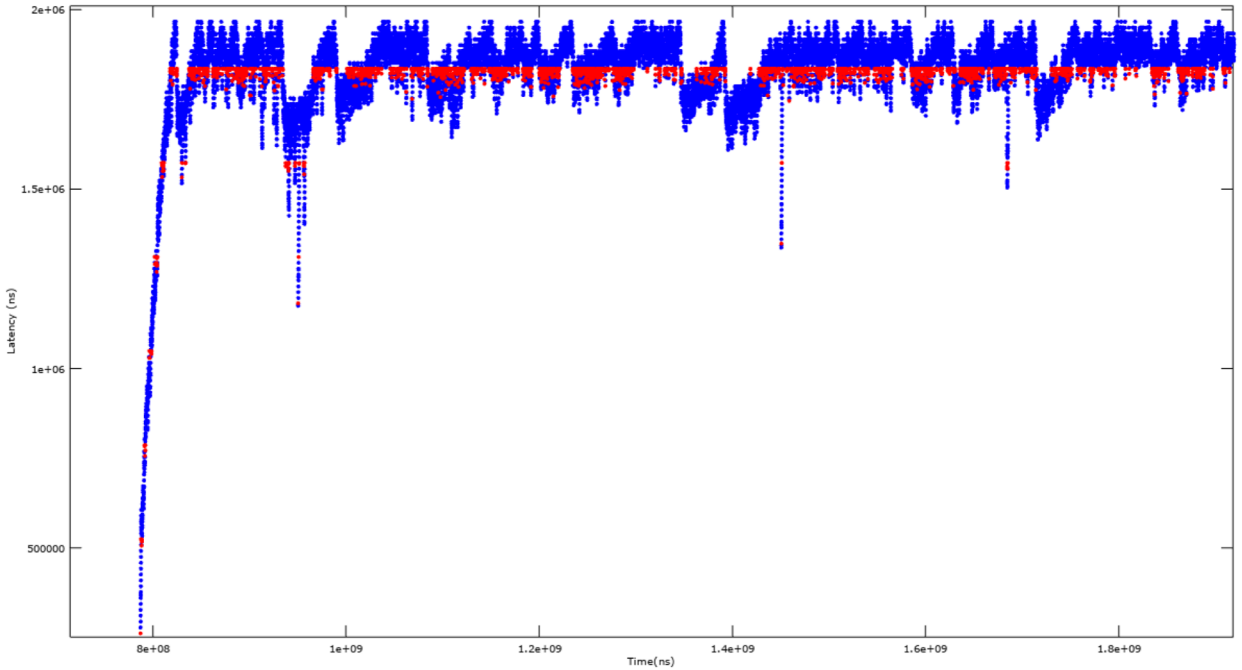


How does a congested queue behave?

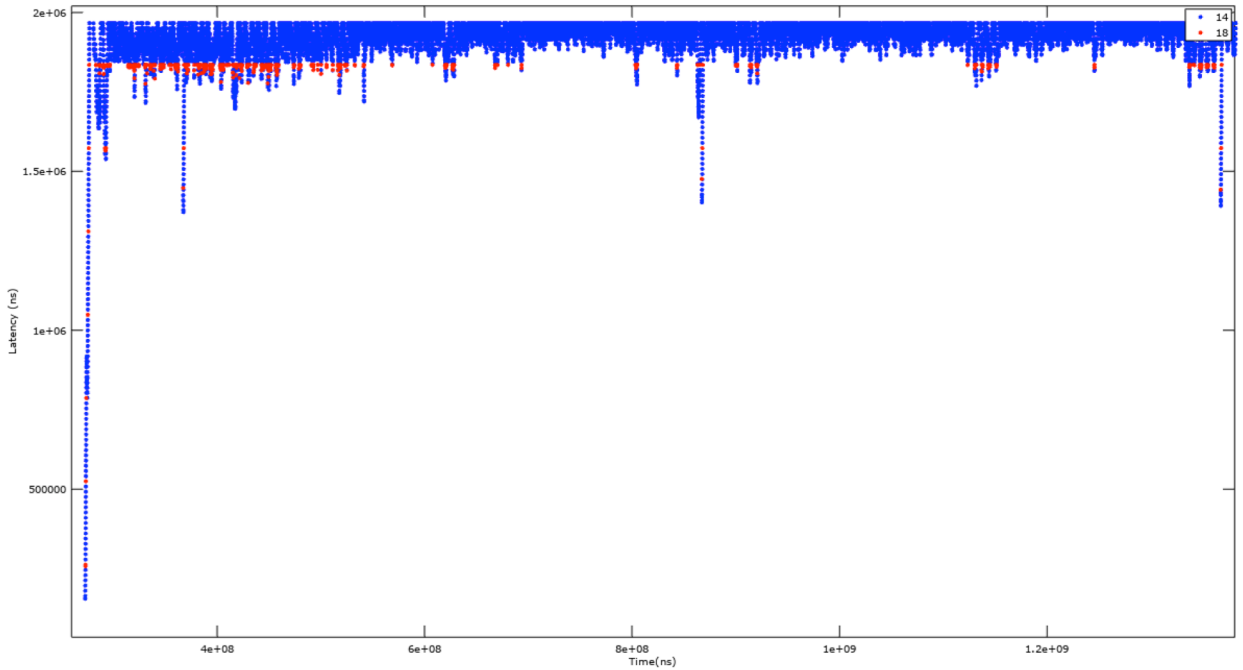


Results with more connections

25 connections (still somewhat meta-stable)



50 connections (stable)



Conclusions

- Measurement is crucial to network operations
  - Measure, model, control
  - Detect, diagnose, fix
- Network measurement is challenging
  - Very high speed vs. limited h/w technology
  - Large volume of measurement data with multi dimensions
- Programmable data planes open up exciting new possibilities
  - “Data-plane Telemetry” for ground-truth information
- Great way to understand the network, applications, users
  - Popular applications, traffic characteristics