PINT: Probabilistic In-band Network Telemetry

Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minlan Yu, Michael Mitzenmacher

SIGCOMM 2020

Outline

- **Background**
- *PINT*
- Implementation and Evaluation
- Review

What is Network Telemetry?

Obtain information from switch:

- Port utilization (CONGA)
- Queue length/latency
- Drop rate
- …

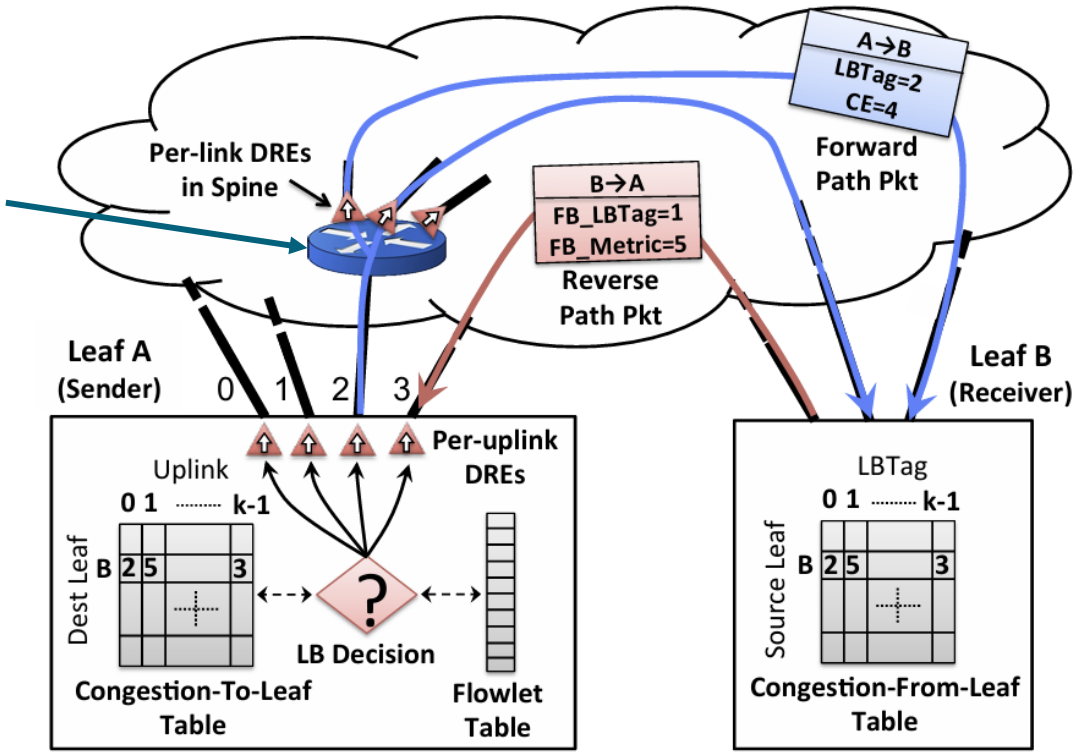


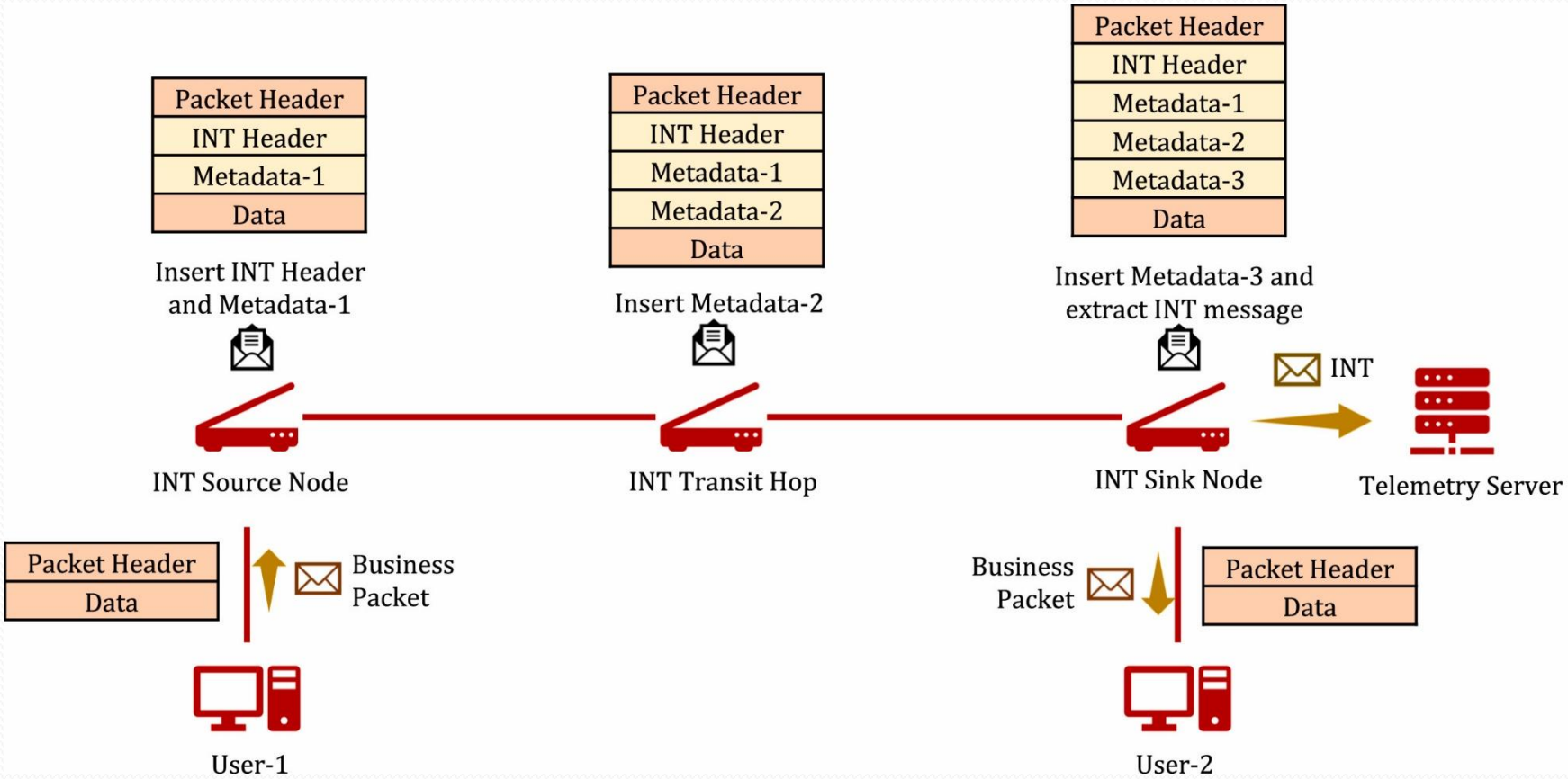
Figure 6: CONGA system diagram.

Figure from CONGA, hope you still remember it

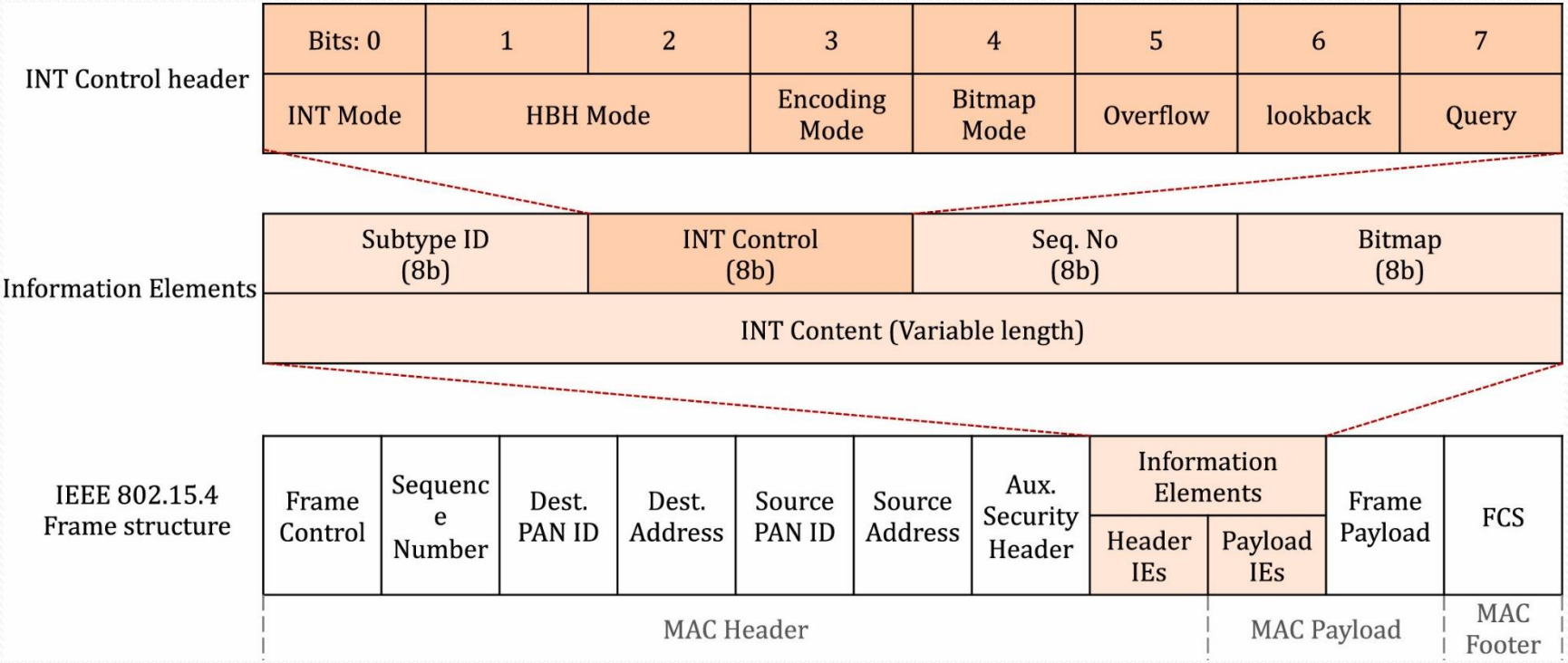
Why In-Network Telemetry?

- **Load Balance**
  - CONGA actually leverages in-network telemetry information
- **Congestion Control**
  - HPCC: High Precision Congestion Control, SIGCOMM 2019
- **Failure Detection**
- **Troubleshooting**

In-Network Telemetry Workflow



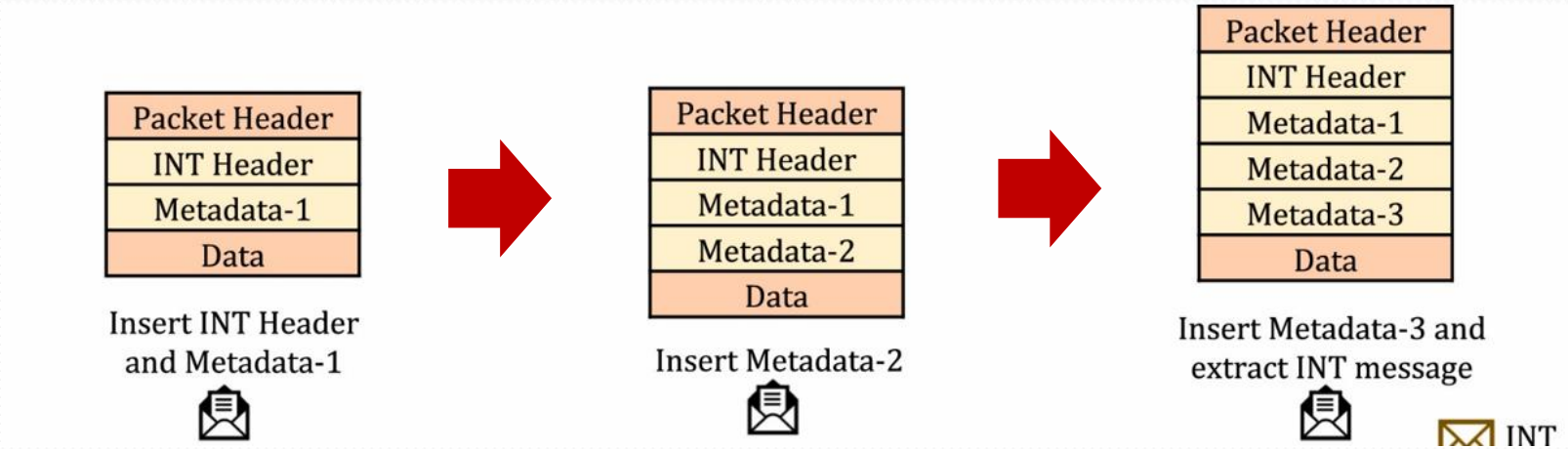
In-Network Telemetry Workflow



In-Network Telemetry Workflow

- INT Header is composed of INT Control header (8 bits), Sequence Number (8 bits) and Bitmap (8 bits).
  - INT Control header is used to define the telemetry mode and telemetry function.
  - Sequence Number is used to distinguish different INT telemetry data from the same node.
  - Each bit in Bitmap represents a predefined telemetry data.

Problem of INT



- Metadata grows **linearly with path length/hops**

High Overhead

- **Example**
  - 5 hops
  - 2 metadata per hop, 4 bytes each
  - MTU = 1000 bytes
- **Overhead**
  - 48 bytes
    - INT header: 8 bytes
    - INT metadata: 5 * 2 * 4 bytes
  - **~5% packet overhead**

High Overhead

- **Degraded Goodput**
  - Some bytes are not used to transfer real data
- **Increased latency**
  - Split packets
    - Particularly for small transfer: RPC
  - Hurt NIC run to completion feature
- Leading in some cases to a **25% increase** and **20% degradation** of flow completion time and goodput

The Question to Answer

- Can we preserve the benefits of in-band network telemetry, but at smaller overhead cost?
- Answer: **Probabilistic In-band Network Telemetry (*PINT*)**

Outline

- Background
- ***PINT***
- Implementation and Evaluation
- Review

Key Insights

- Many applications **do NOT need all per-packet per-switch telemetry information**
  - Congestion control → only bottleneck
  - Path tracing → multiple packets ok
  - Latency → approximate quantile ok
- **Approximate telemetry is sufficient!**

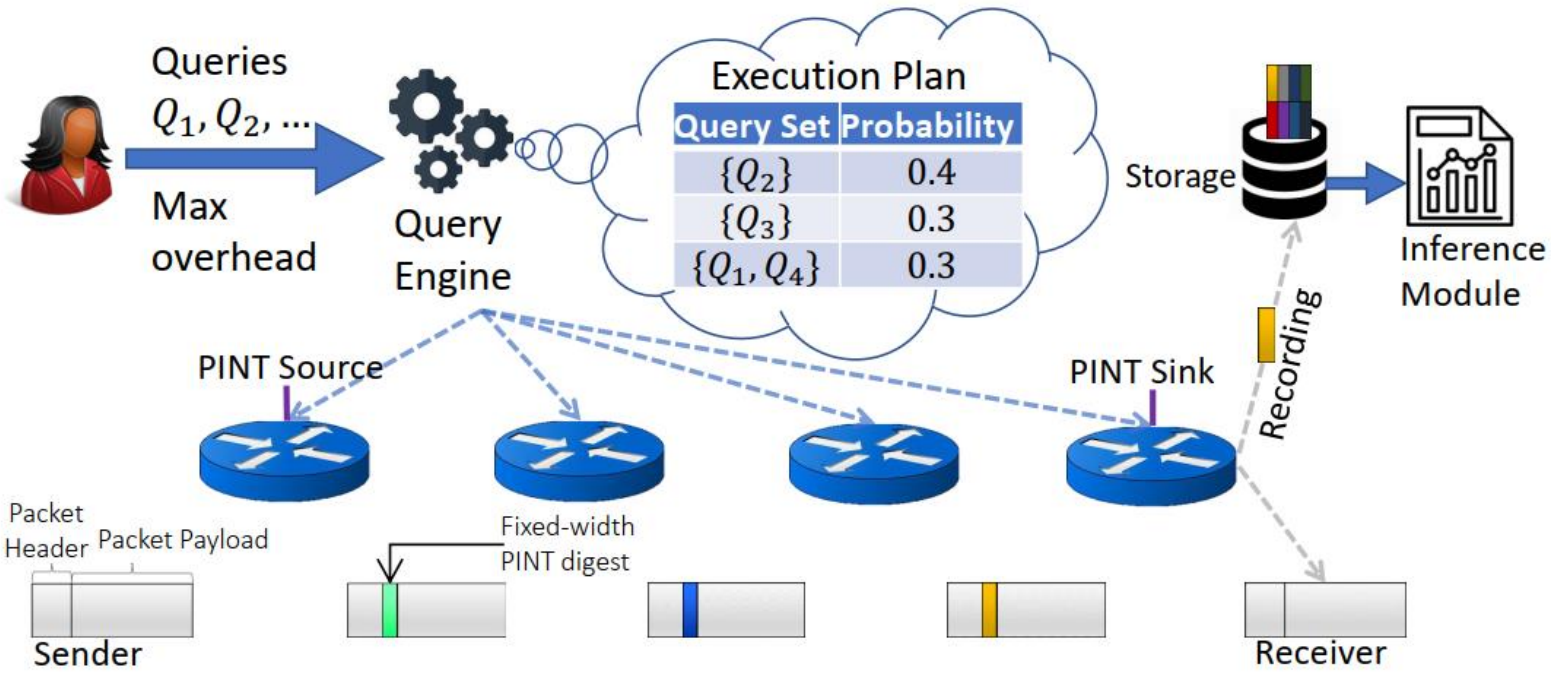
PINT's Core Idea

- The requested information is **probabilistically** encoded into several different packets so that a ***collection*** of a flow's packets provides the relevant data
  - **INT**: a query triggers every switch along the path to embed their own information
  - ***PINT***: spreads out the information over multiple packets to minimize the per-packet overhead

The Advantage of *PINT*

- **INT**
  - Per-packet overhead = O(path length)
- ***PINT***
  - Per-packet overhead = constant

PINT's Architecture



PINT's Key Components

- PINT Query/Query Engine
  - Specify multiple queries that should run concurrently under a ***global bit-budget***.
  - For example,
    - Switch latency
    - Queue length
    - …
  - Global bit-budget: 8bit

PINT's Key Components

- PINT Query/Query Engine
  - Generate the query plan:

Query	Probability
Latency	0.4
Queue	0.3
Others	0.3

Probability Execution Plan

PINT's Key Components

- PINT Digest
  - Fixed-width
  - Shared by all switch
  - **Modify but not add bits to digest**
  - For example,
    - Switch 1: digest = hash(latency)
    - Switch 2: digest = digest XOR hash(queue)
    - Switch 3: digest = digest XOR hash(…)
  - **Distributed Encoding but why XOR?**

Challenges

- How to coordinate among all switches?
  - **Hash-based Coordination**
- How to meet the constraints of switch?
  - **XOR-based Distributed Encoding**
- How to meet the bit constraints?
  - **Approximating Numeric Values**

Key Techniques

Technique	Purpose
Global Hashes	Decide **what information each packet should record**
Distributed Coding	Use **XOR-based encoding to distribute information across multiple packets**
Value Approximation	Use **sampling / sketches to approximately estimate values**

Use Case	Global Hashes	Distributed Coding	Value Approximation
Congestion Control	✗	✗	✓
Path Tracing	✓	✓	✗
Latency Quantiles	✓	✗	✓

Hash-based Coordination

- Problem: Switch must know which query to perform given a packet
- Solution: Use global hash function to coordinate switches **without extra bits**
- Example:
  - Suppose that we have three queries, each running with probability 1/3, and denote the query selection hash, mapping packet IDs to the real interval [0, 1], by q.
  - Then if q(*pj*) < 1/3, all switches would run the first query.
  - If q(*pj*) ∈ [1/3, 2/3] the second query, and otherwise the third.
  - q is **known by all switch**, so all switches compute the same q(*pj*).

XOR-based Distributed Encoding

- **Static per-flow aggregation:** When the values are static for a given flow (i.e., do not change between packets), we can improve upon the dynamic aggregation approach using *distributed encoding*

XOR-based Encoding

- We have 3 packets:
  - P1 = A
  - P2 = B
  - P3 = A XOR B
- If Packet P2 is lost, we can still have
  - B = P1 XOR P3
- XOR encoding can be leveraged to **recover information**!
- XOR is **hardware-friendly!**

Approximating Numeric Values

- **Dynamic per-flow aggregation**

  packet1: latency = 5us
  packet2: latency = 200us
  packet3: latency = 10us
  packet4: latency = 30us

- **We cannot do:**

  5 XOR 200 XOR 10 XOR 30

Approximating Numeric Values

- Multiplicative approximation
- Additive approximation
- Randomized counting

Multiplicative Approximation

$$\text{digest } \mathfrak{a}(p_j, s) \triangleq \left[\log_{(1+\varepsilon)^2} v(p_j, s)\right]$$

$[\cdot]$ operator rounds the quantity to the closest integer

Decoding method: $(1+\varepsilon)^{2\cdot \mathfrak{a}(p_j,s)}$

Multiplicative Approximation

Latency	Log scale
1	0
2	1
4	2
8	3
16	4

- **Few bits**: 32bit latency becomes 8bit index
- Why it works?
  - For latency, **order-of-magnitude** differences matter more

Approximating Numeric Values

- Refer to the paper for more details
  - Additive approximation
  - Randomized counting

Revisit Key Techniques

Use Case	Global Hashes	Distributed Coding	Value Approximation
Congestion Control	✗	✗	✓
Path Tracing	✓	✓	✗
Latency Quantiles	✓	✗	✓

Congestion Control:

- Care end-to-end information and no need to run different query on switch
  - Global hash is not needed
- All packets of a flow carry different value, dynamic
  - Distributed coding (XOR) cannot be used
- Average queue length/trend is needed, do not need accurate value
  - Value approximation can be used

Revisit Key Techniques

Use Case	Global Hashes	Distributed Coding	Value Approximation
Congestion Control	✗	✗	✓
Path Tracing	✓	✓	✗
Latency Quantiles	✓	✗	✓

Path Tracing:

- Different packets can record path of different hops
  - Global Hash can be used
- Path does not change across packet
  - Distributed coding (XOR) can be used
- Exact path is needed
  - Value approximation cannot be used

Revisit Key Techniques

Use Case	Global Hashes	Distributed Coding	Value Approximation
Congestion Control	✗	✗	✓
Path Tracing	✓	✓	✗
Latency Quantiles	✓	✗	✓

Latency Quantiles: Average/P90/P99 latency

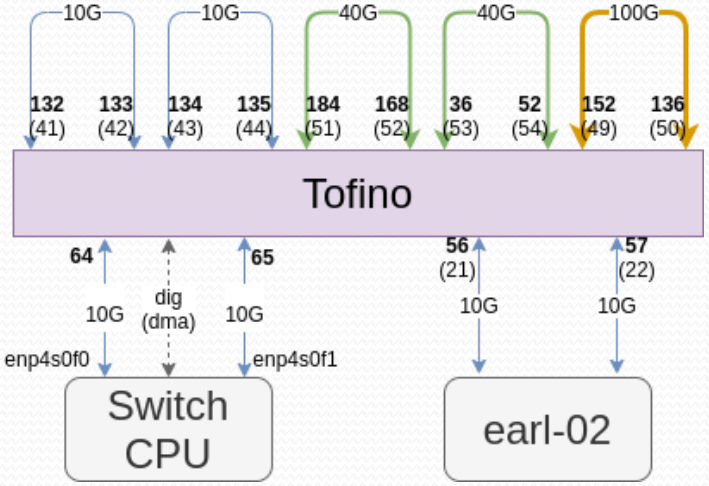
- Different packet should report different switch
  - Global hash is needed
- All packets of a flow carry different value, dynamic
  - Distributed coding (XOR) cannot be used
- Average latency/trend is needed, do not require exact value
  - Value approximation can be used

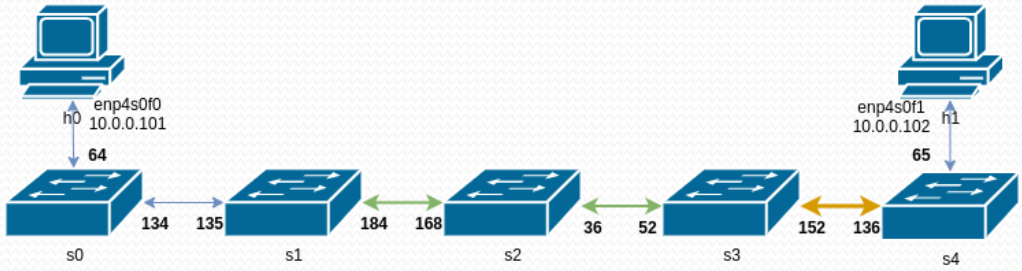
Outline

- Background
- *PINT*
- **Implementation and Evaluation**
- Review

Implementation

- Fully open-sourced
- https://github.com/ProbabilisticINT/Tofino-PINT





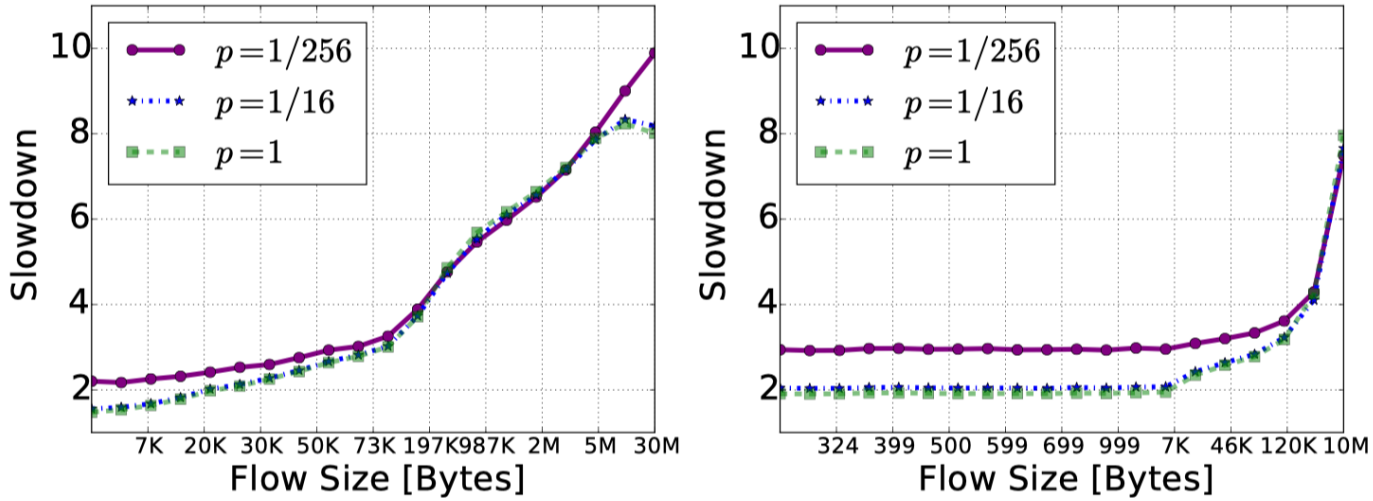
Evaluation of *PINT*

- Congestion Control
- Latency Measurements
- Path Tracing
- Combined Experiments

Congestion Control

- Evaluate how *PINT* affects the performance of HPCC
  - HPCC is a CC algorithm based on INT signals
- ns3 simulation: FatTree topology with 16 Core switches, 20 Agg switches, 20 ToRs, and 320 servers (16 in each rack)
  - 400Gbps bandwidth
  - 12µs maximum base RTT
- Two workloads
  - Web Search (same as in the DCTCP paper)
  - Hadoop

Congestion Control



(a) Web search workload

(b) Hadoop workload

Figure 8: The 95th-percentile slowdown of running *PINT*-based HPCC (at 50% network load) on p-fraction of the packets. On both workloads, the performance of running it on $1/16$ of the packets produces similar results to running it on all.

Latency Measurements

- Same topology
- Same workloads
- PINT in four scenarios
  - using b = 4 b = 8 bit-budgets
  - with sketches (denoted PINT S), and without
  - Sketch is a data structure to improve the results of statistics results

Latency Measurements

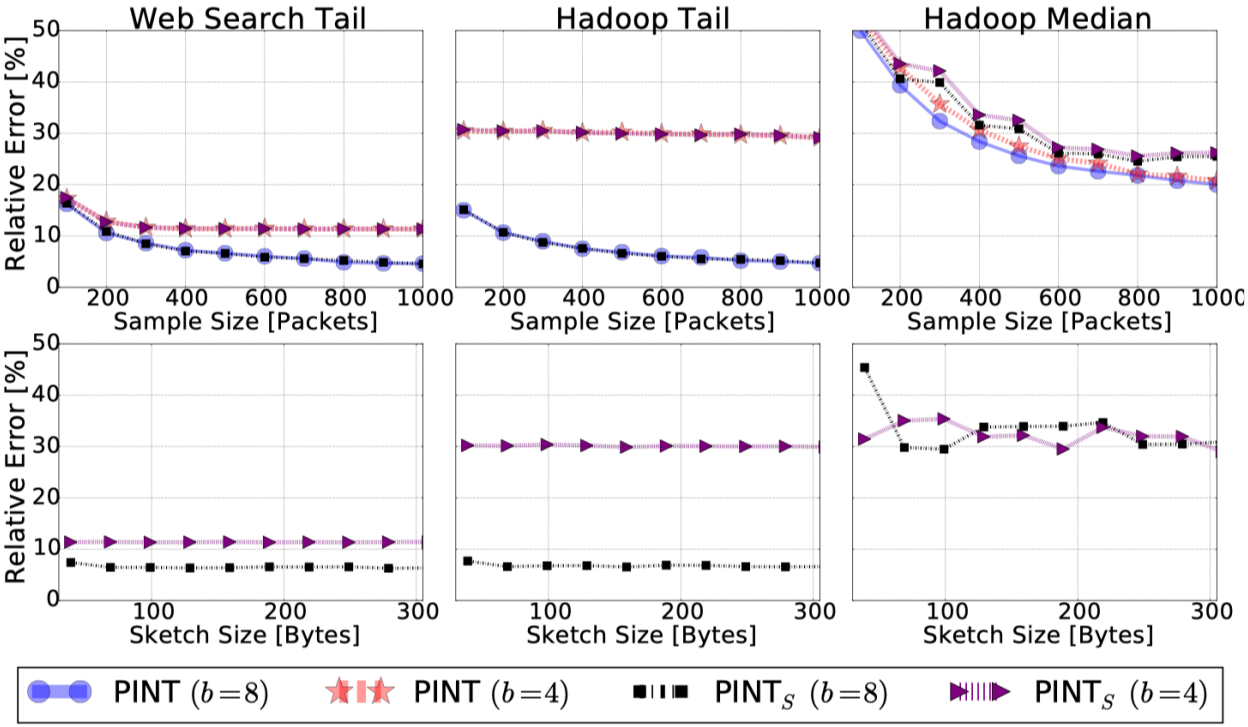


Figure 9: *PINT* error on estimating latency quantiles with a sketch (PINT$_S$) and without. In the first row, the sketch has 100 digests; in the second, the sample has 500 packets.

Latency Measurements

- When getting enough packets, the error of the aggregation becomes stable and converges to the error arising from compressing the values.
- By compressing the incoming samples using a sketch (e.g., that keeps 100 identifiers regardless of the number of samples), PINT accuracy degrades only a little even for small 100B sketches.

Path Tracing

- Mininet (emulator):
  - Use two large diameter (denoted D) ISP topologies (Kentucky Datalink and US Carrier) from Topology Zoo
  - Fat Tree topology with K=8
- 3 variants of PINT
  - Using 1-bit, 4-bit, and two independent 8-bit hash functions (denoted by 2 × (b = 8))
  - Compare PINT to two state-of-the-art IP Traceback solutions
    - PPM
    - AMS2

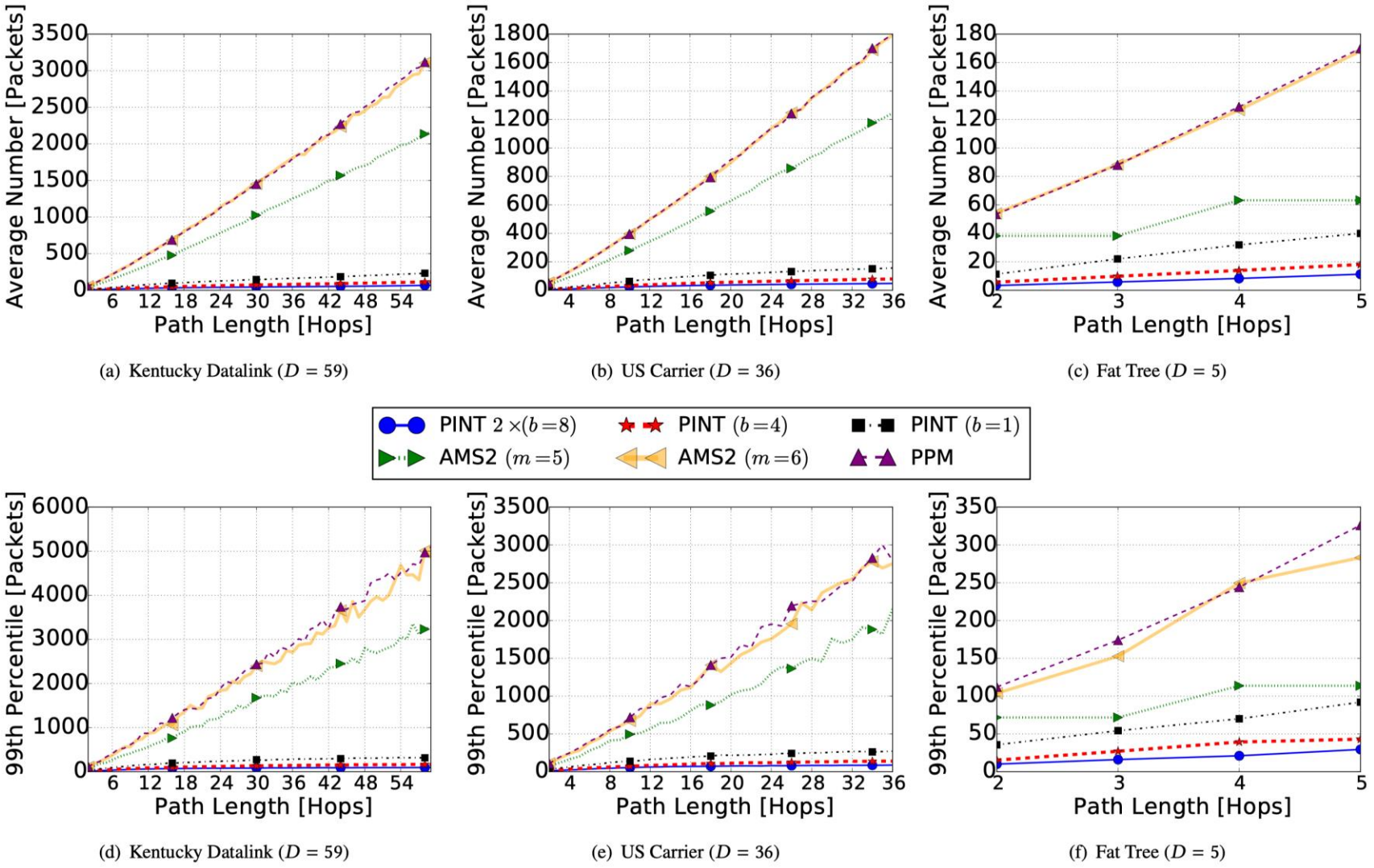


Figure 10: Comparison of the number of packets required (lower is better) for path decoding of different algorithms, including *PINT* with varying bit-budget.

Combined Experiments

- Test the performance of PINT when running all three use cases concurrently

Combined Experiments

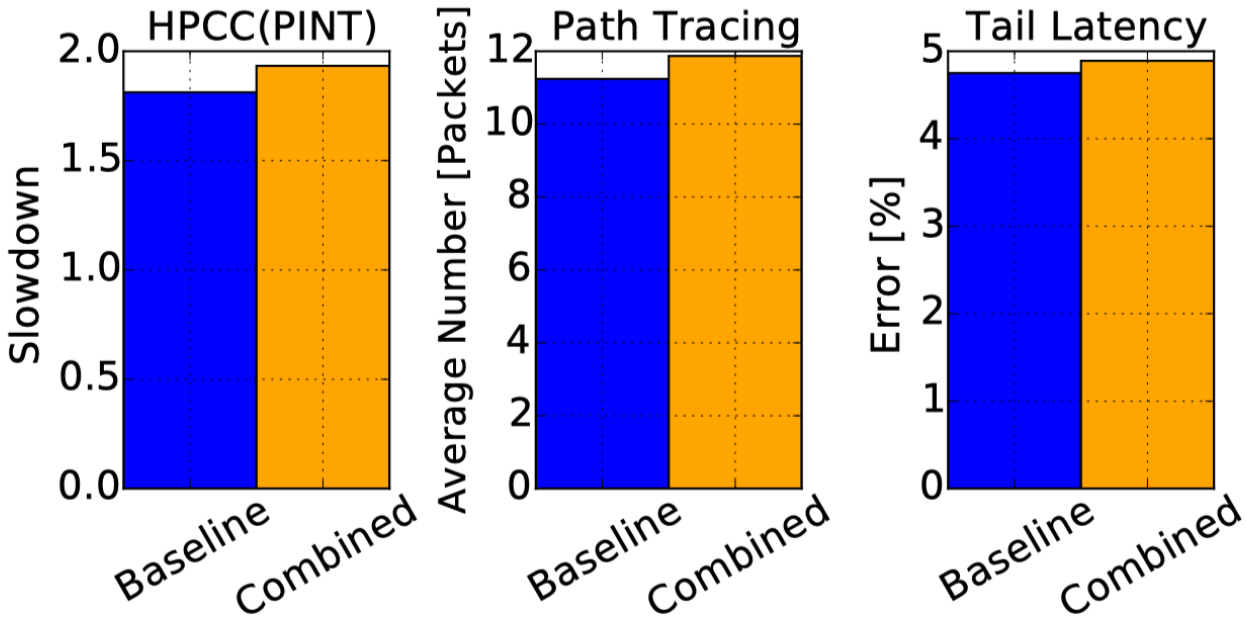


Figure 11: The performance of each query in a concurrent execution (FatTree topology + Hadoop workload) compared to running it alone.

Combined Experiments

- **The performance of PINT is close to a Baseline of running each query separately**
- For estimating median latency, the relative error increases by only 0.7% from the Baseline to the combined case.
- In case of HPCC, we that observe short flows become 6.6% slower while the performance of long flows does not degrade.
- For path tracing, the number of packets increases by 0.5% compared with using two 8bit hashes

Limitations

- **Tracing short flows**
  - *PINT* requires multiple packets but small flows can consist of just a single packet
- **Data plane complexity**
  - PINT consumes many programmable resources on switches, restricting other applications deployed on switches
- **Tracing flows with multipath routing**
  - Path change may make values (i.e., switch IDs) for some hops become different
  - Further mechanisms are required

Outline

- Background
- *PINT*
- Implementation and Evaluation
- **Review**

Review

- In-Network/band Telemetry
  - Required by many applications, such as CC/LB/Debugging
  - Problem: **High per-packet bit overhead**
- ***PINT*: Probabilistic In-band Network Telemetry**
  - Insight: Many applications do **NOT** need all per-packet per-switch telemetry information
  - Requested information is **probabilistically** encoded into several different packets

Topic Review

Software Defined Network

What We Have Learned?

- Programmability
  - **Control Plane** Programmability: OpenFlow
  - **Data Plane** Programmability: P4
- Two SDN applications
  - **In-networking Aggregation**: ATP
  - **In-networking Telemetry**: PINT

Some Key Points

- OpenFlow & P4
  - Functions of control/data plane
  - Why we need control plane programmability?
  - Why we need control plane programmability?

Some Key Points

- ATP
  - What is in-networking aggregation? How does SwitchML work? What's the problem of SwitchML?
  - ATP Packet Format & Switch Memory Layout
  - The overall workflow of ATP

Some Key Points

- PINT
  - What is in-networking telemetry? What is the existing problem?
  - Why do different applications require different key technologies of PINT?

Reading Materials
(For Group Presentation. Not in the Exam)

- **More In-network Applications**
  - **Congestion Control**: HPCC: High Precision Congestion Control, SIGCOMM 2019
  - **L4 Load Balancer**: SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs, SIGCOMM 2017
  - **KV Store**: NetCache: Balancing Key-Value Stores with Fast In-Network Caching, SOSP 2017