One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon

Zaoxing Liu, Antonis Manousis, Greg Vorsanger, Vyas Sekar, and Vladimir Braverman
Many Monitoring Requirements

Traffic Engineering
“Flow Size Distribution”

Worm Detection
“SuperSpreaders”

Anomaly Detection
“Entropy”, “Traffic Changes”

Accounting
“Heavy Hitters”

• Who’s sending a lot more traffic than 10min ago? (Change)
• Who’s sending a lot from 10.0.1.0/16? (Heavy Hitter)
• Are you being DDoS-ed?
Traditional: Packet Sampling

Sample packets at random, group into flows

Flow = Packets with same pattern Source and Destination Address and Ports

Flow reports

Prior work: Not good for fine-grained analysis!
Alternative: App-Specific Sketches

Higher Complexity with more applications
Higher development time as new applications appear
Tight Binding between monitoring data and control plane
Motivating Question

Can we achieve this?

e.g., NetFlow

e.g., Sketches

Generality
Late Binding

Fidelity

Today

XOR

AND
UnivMon Vision

- One Sketch for multiple tasks
- Naturally Late-binding
Many Natural Challenges!

Does such a construction exist?

If it exists, is it feasible to implement?

Does it extend to a network-wide setting? e.g., Multiple paths, Multiple dimensions

Is it competitive w.r.t. custom algorithms?
This Talk

Does such a construction exist?

If it exists, is it feasible to implement?

Does it extend to a network-wide setting? e.g., Multiple paths, Multiple dimensions

Is it competitive w.r.t. custom algorithms?
This Talk

Does such a construction exist?

If it exists, is it feasible to implement?

Does it extend to a network-wide setting?
  e.g., Multiple paths, Multiple dimensions

Is it competitive w.r.t. custom algorithms?
Concept of Universal Streaming

- Basic Streaming Algorithms:
  (A stream of length $m$ with $n$ unique items)

\[ 1 \ 1 \ 5 \ 1 \ 3 \ 3 \ 1 \ 2 \ 4 \ 6 \ 5 \ \ldots \]

frequency vector \(< f_1, f_2 \ldots f_n >\)

- Universal Streaming?

\[ 1 \ 1 \ 5 \ 1 \ 3 \ 3 \ 1 \ 2 \ 4 \ 6 \ 5 \ \ldots \]

frequency vector \(< f_1, f_2 \ldots f_n >\)

Frequency Moments $F_k = \sum_{i=1}^{n} f_i^k$

$F_2$ : AMS Sketch, Count Sketch

\ldots

One algorithm solves one problem

Universality:
 arbitrary $g()$ function?

\[ G\text{-sum} = \sum_{i=1}^{n} g(f_i) \]
Theory of Universal Streaming [BO’10, BO’13]

**Thm 1:**
There exists a universal approach to estimate $G$-sum when $g()$ function is non-decreasing such that $g(0)=0$, and $g(f_i)$ doesn’t grow monotonically faster than $f_i^2$.

**Thm 2:**
A universal sketch construction can be used to estimate $G$-sum with high probability using polylogarithmic memory.
Intuition of Universal Sketch

Informal Definition: Item $i$ is a $g$-heavy hitter if changing its frequency $f_i$ significantly affects its G-sum.

Case 1: there is one sufficiently large a $g$-heavy hitter

Most of mass is concentrated in this heavy hitter. Use L2 Heavy-Hitter algorithm to find such a heavy hitter.

Case 2: there is NO single sufficiently large $g$-heavy hitter

Find heavy hitters on a series of sampled substreams of increasingly smaller size.
Universal Sketch Data Structure

Generate $\log(n)$ substreams by zero-one hash funcs $H_1, \ldots, H_{\log(n)}$

Levels

In Parallel

Generate $\log(n)$ substreams by zero-one hash funcs $H_1, \ldots, H_{\log(n)}$

In Parallel

Levels
This Talk

Does such a construction exist?

If it exists, is it feasible to implement?

Does it extend to a network-wide setting? e.g., Multiple paths, Multiple dimensions

Is it competitive w.r.t. custom algorithms?
How to Map to P4

Heavy Hitter Alg

Estimated G-sum

(1,4), (3,2), (5,2)

(1,4), (5,2), (2,1)

(2,1)

Sampling → Sketching → Top-K

App-Estimation
Top-K Stage on Switch

Sampling ➔ Sketching ➔ Top-K

App-Estimation

HW Complexity (need Priority Queue)

Storage/Comm Overhead (report to controller)

Hard in hardware
Split Top-K Stage

Sampling → Sketching → Top-K → App-Estimation

- HW Complexity (w/o Priority Queue)
- Storage/Comm. Overhead (report to controller)
- Several MBs more
Implementation Summary

- Sampling
  - (Hash func)
- Traffic
- Update Counters
- Sketching
  - P4 register
- Top-K
  - Possible keys
- Application-specific Computation

Sampling → Sketching → Top-K → App-Estimation
This Talk

Does such a construction exist?

If it exists, is it feasible to implement?

Does it extend to a network-wide setting? e.g., Multiple paths, Multiple dimensions

Is it competitive w.r.t. custom algorithms?
Network-wide Problem

N nodes
D dimensions (e.g., src, srcdst)

One sketch for each dim

**Trivial sol:** place D*N sketches

**Our goal:** Place s sketches, where s<<D*N

**One-big-switch** abstraction
This talk

Does such a construction exist?

If it exists, is it feasible to implement?

Does it extend to a network-wide setting? e.g., Multiple paths, Multiple dimensions

Is it competitive w.r.t. custom algorithms?
Evaluation Setup

- Traces: CAIDA backbone traces
  - Split into different “epoch” durations
- Memory setup: 600KB—5MB
- Application metrics: HH, Change, DDoS, etc.
- Custom algorithms from OpenSketch
UnivMon is Competitive Per-App

Error Rate Comparison

Max error gap < 3.6%; Results hold across multiple traces
UnivMon Better for Larger Portfolio

Clear advantages when handling more applications
Memory needs are reasonable

- Memory Usage (KB)
  - OS-trace1
  - OS-trace2
  - OS-trace3
  - OS-trace4
  - OS-trace5
  - UM-trace1
  - UM-trace2
  - UM-trace3
  - UM-trace4
  - UM-trace5

- Monitoring Time Interval:
  - 5s
  - 30s
  - 1m
  - 5m

Slow increase (logarithmically) and supports larger windows
Conclusions

• Network management needs many metrics

• Traditional: Generality XOR Fidelity
  • E.g., NetFlow vs Custom Sketches

• New opportunity: Universal Sketches!
  • Generality AND Fidelity AND Late Binding

• UnivMon brings this opportunity to fruition
  • Practical, realizable in P4
  • Comparable (and better) than custom
  • Amenable to “network-wide” abstractions
  • Many exciting future directions:
    • Theoretical improvements, Native multidimensional, etc.
Network-wide coordination helps

Network Wide Evaluation (600KB per sketch)

Ingress
Greedy-D.&C.
Q.&S.
UnivMon

Average Memory(KB)

ATT-N.A.  GEANT  BellSouth