Scalable and Robust DDoS Detection via Universal Monitoring

Vyas Sekar
Joint work with:
Alan Liu, Vladimir Braverman JHU
Hun Namkung, Antonis Manousis, CMU

DDoS attacks are getting worse

Increasing in *number*

Increasing in *power*

Increasing in *diversity*

DDoS Attacks Cost $40,000 Per Hour

Incapsula, 11/12/2014

FBI WARNS OF INCREASE IN DDOS EXTORTION SCAMS

Threatpost, 7/31/2015

China Appears to Attack GitHub by Diverting Web Traffic

The New York Times, 3/30/2015

Half of companies experience more than five DDoS attacks a year.

Neustar, 2014

The DDoS That Almost Broke the Internet

Cloudflare, 3/27/2013

Wave of 100Gbps 'mega' DDoS attacks hits record level in 2014

Techworld, 7/16/2014

NTP ATTACKS: Welcome to The Hockey Stick Era

Arbor Networks, 2/14/2014

Tsunami SYN Flood Attack

Radware, 10/7/2014

Many attacks, many algorithms!

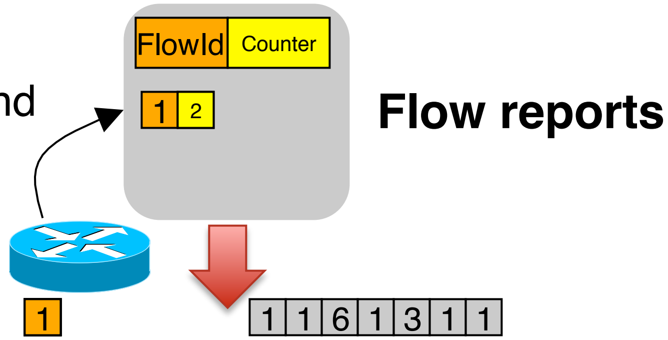
- **SYN Flood**
- **UDP Flood**
- **NTP Flood**
- **DNS Amplification**

- Who's sending a lot more traffic than 10min ago?
- Who's sending a lot to 10.0.1.0/16?
- Is there asymmetry in packet counts in directions?

Classical Netflow-style packet sampling

Sample packets at random, group into flows

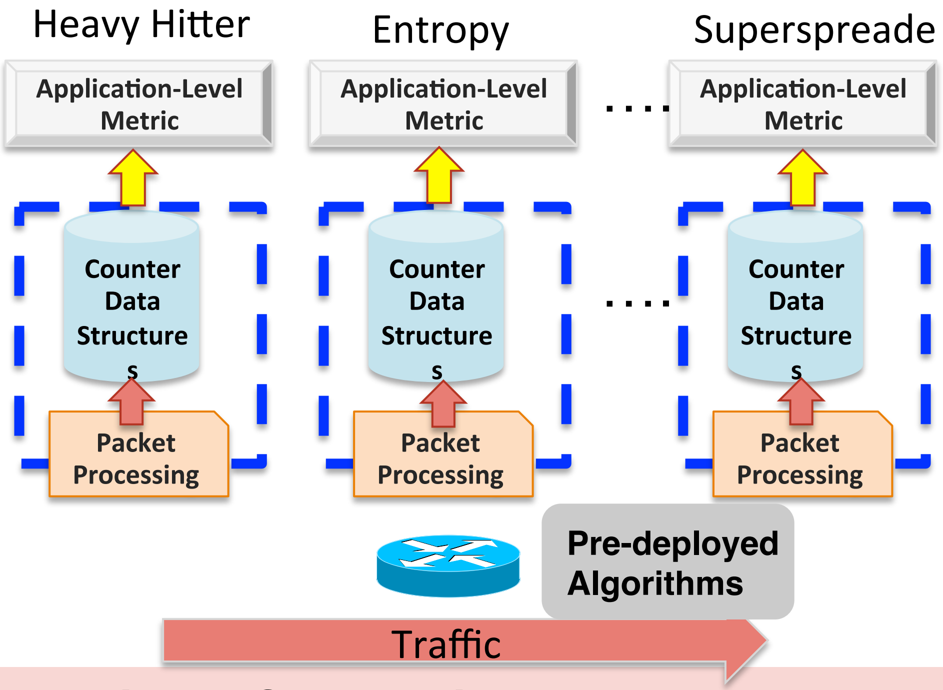
Flow = Packets with same pattern Source and Destination Address and Ports



Estimate: FSD, Entropy, Heavy Hitters …

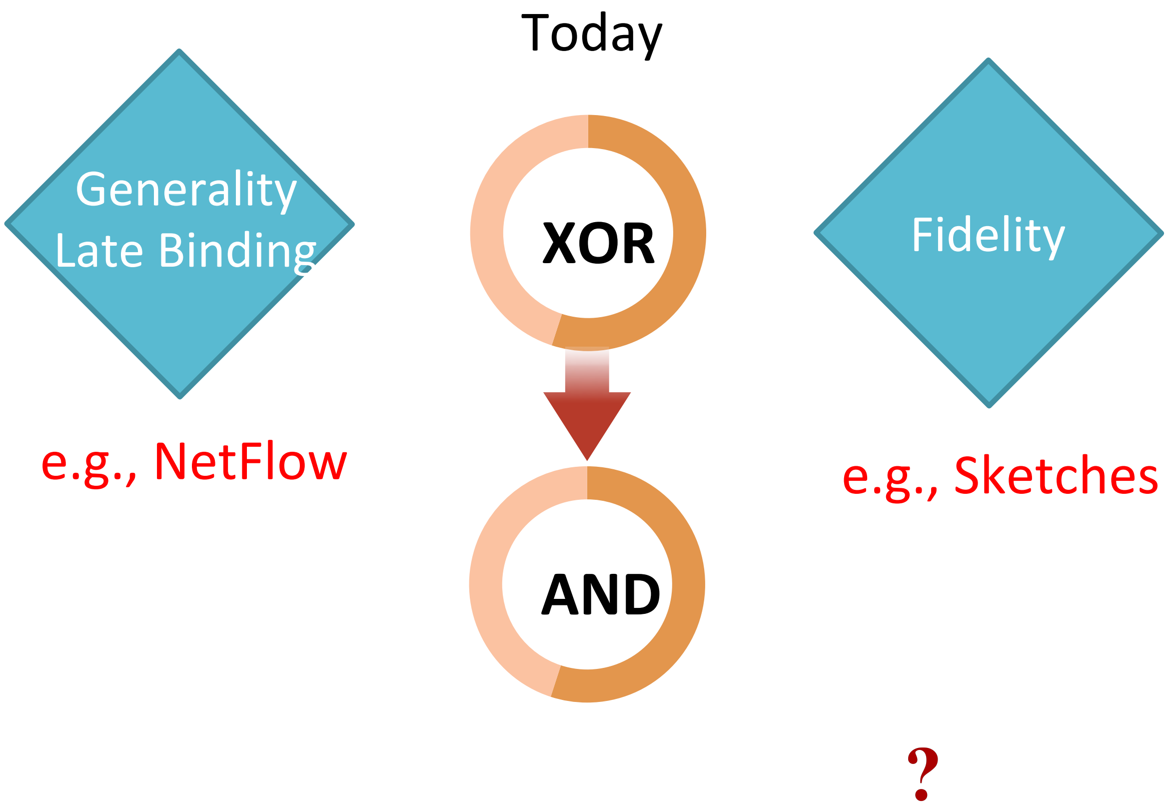
Prior work: Not good for fine-grained analysis!

Alternative: App-specific sketches



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

Driving question for our work



Many open questions..

Does such a construction exist?

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

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

Is it feasible to implement?

Roadmap for this talk

OpenNFP

Does such a construction exist?

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

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

Is it feasible to implement?

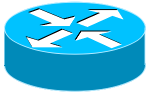
Concept of Universal Streaming

- Basic Streaming Algorithms:

(A stream of length m with n unique items)

1	1	5	1	3	3	1	2	4	6	5

……



frequency vector $< f_1, f_2 \dots f_n >$

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

F_2 : AMS Sketch, Count Sketch

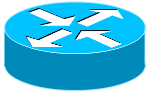
……

One algorithm solves one problem

- Universal Streaming?

1	1	5	1	3	3	1	2	4	6	5

……



frequency vector $< f_1, f_2 \dots f_n >$

Universality:
arbitrary g() function?

G-sum = $\sum_{i=1}^{n} g(f_i)$

Theory of Universal Streaming

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 polylogmithric memory.

Intuition behind Universality

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

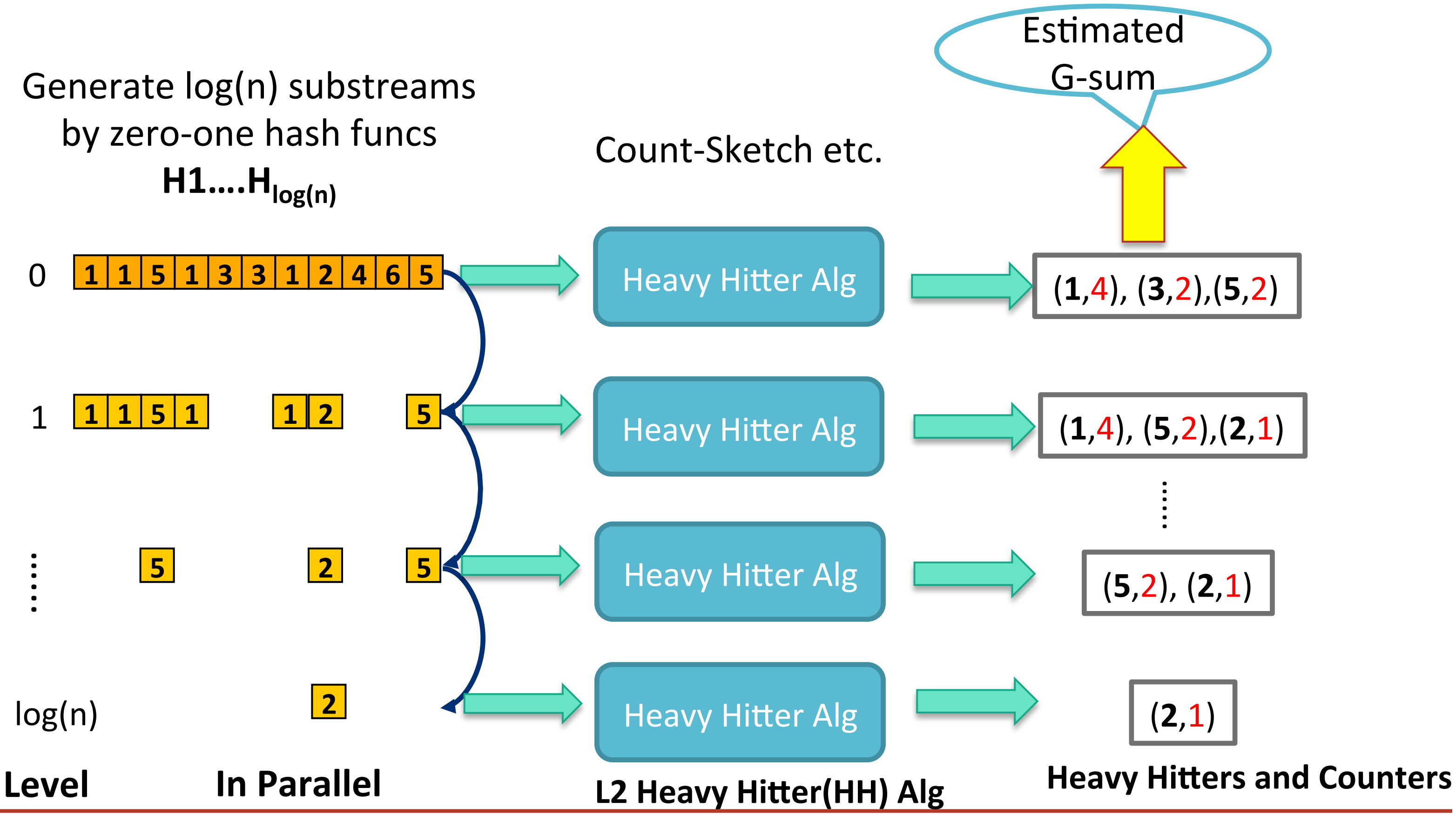
Case 1: there is one sufficiently large a g-heavy hitter

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

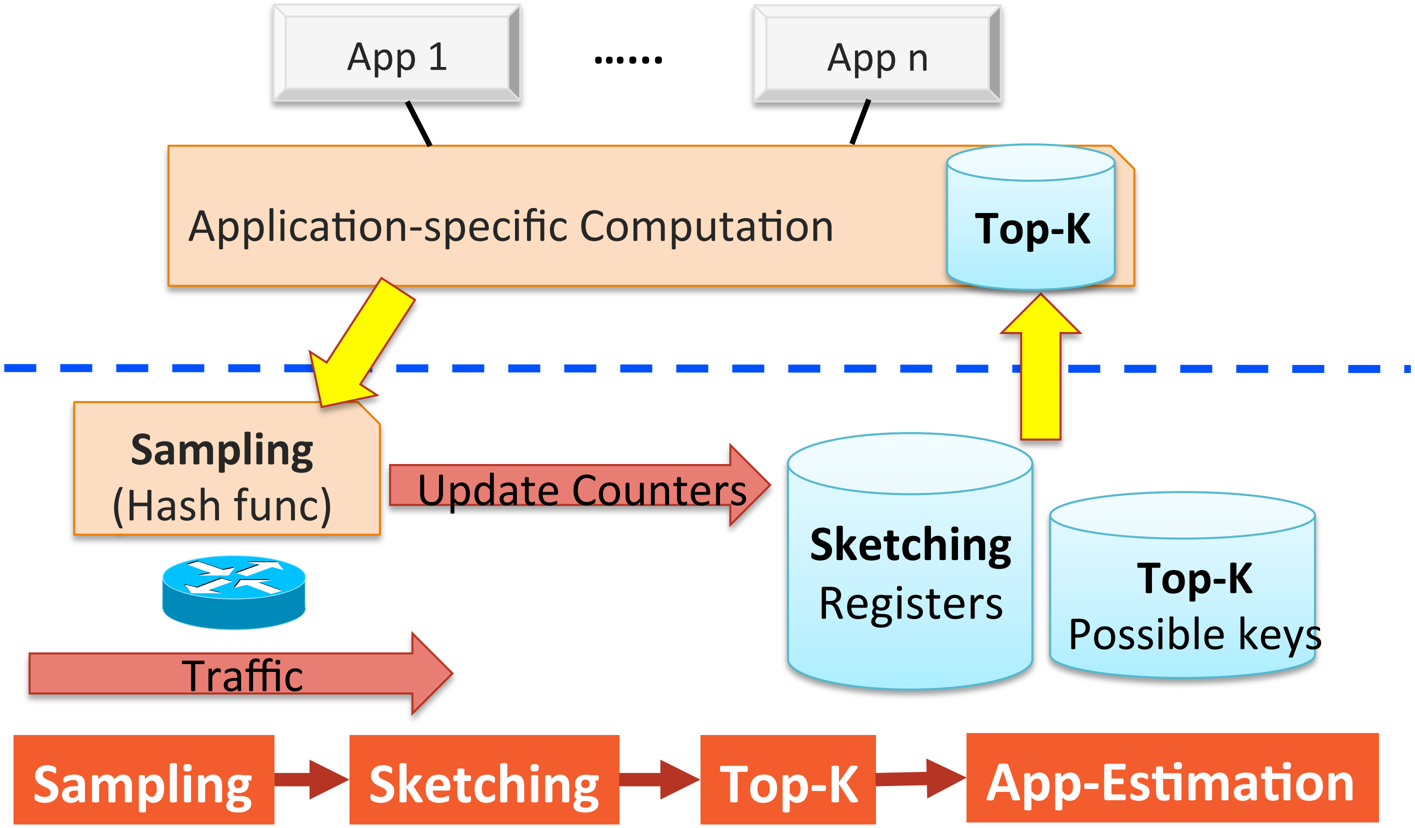
Case 2: there is NOT single sufficiantly large a g-heavy hitter

Find heavy hitters on a series of sampled substreams of increasingly smaller size.

Universal Sketching Algorithm



Universal Monitoring Realization



Roadmap for this talk

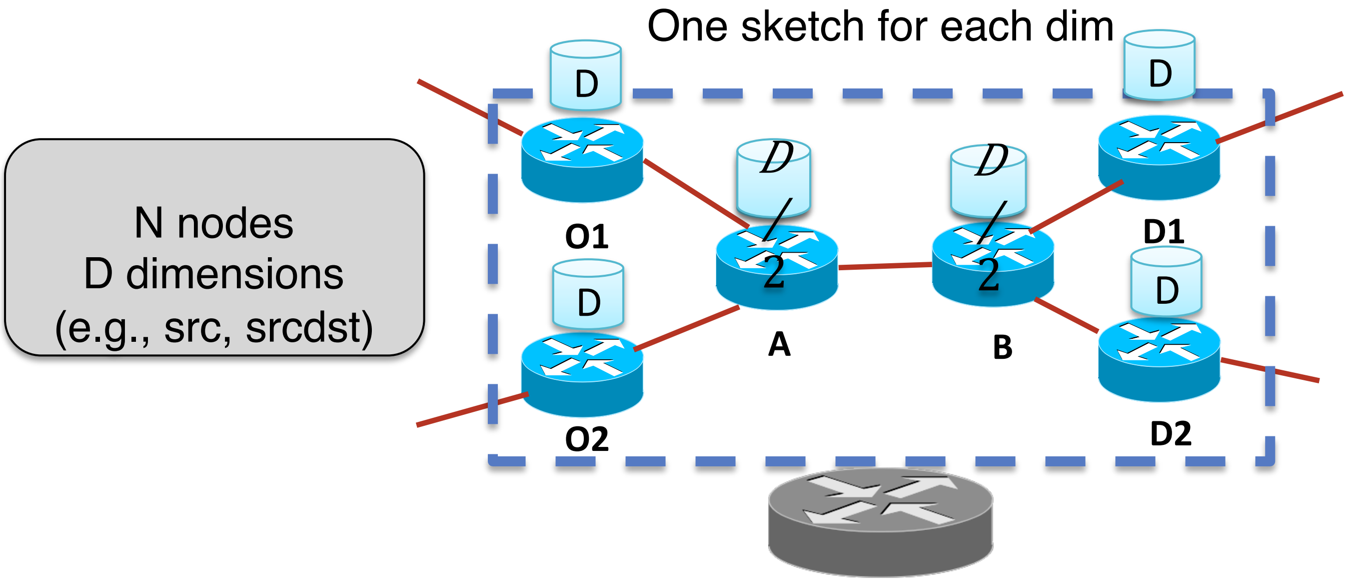
Does such a construction exist?

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

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

Is it feasible to implement?

Network-Wide Problem



Trivial sol: place D*N sketches
Our goal: Place s sketches, where s<<D*N
One-big-switch abstraction

Roadmap for this talk

Does such a construction exist?

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

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

Is it feasible to implement?

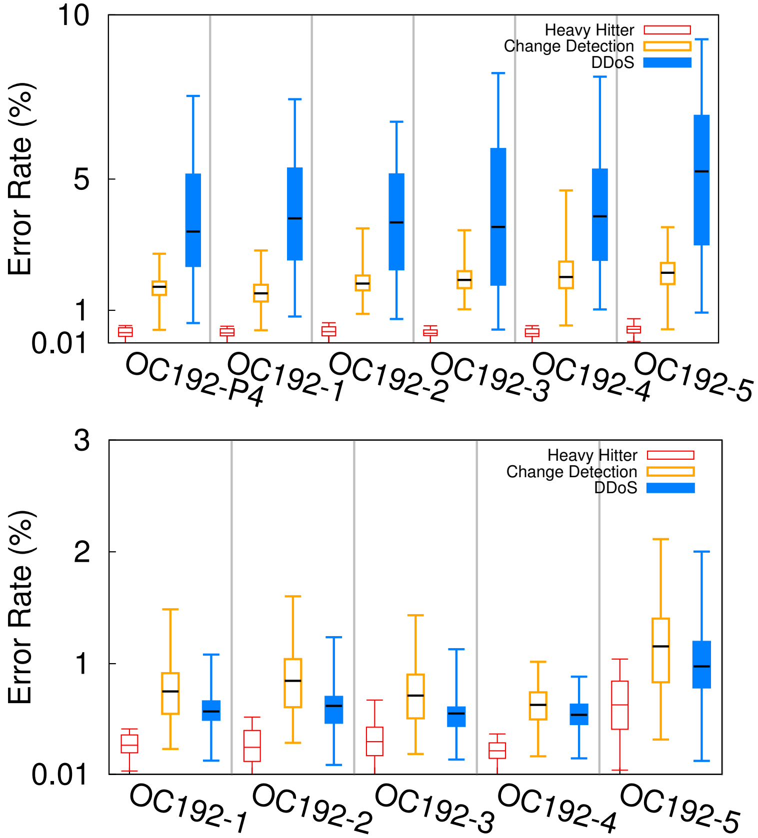
Evaluation Setup

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

UnivMon is Competitive

Key Takeaways:

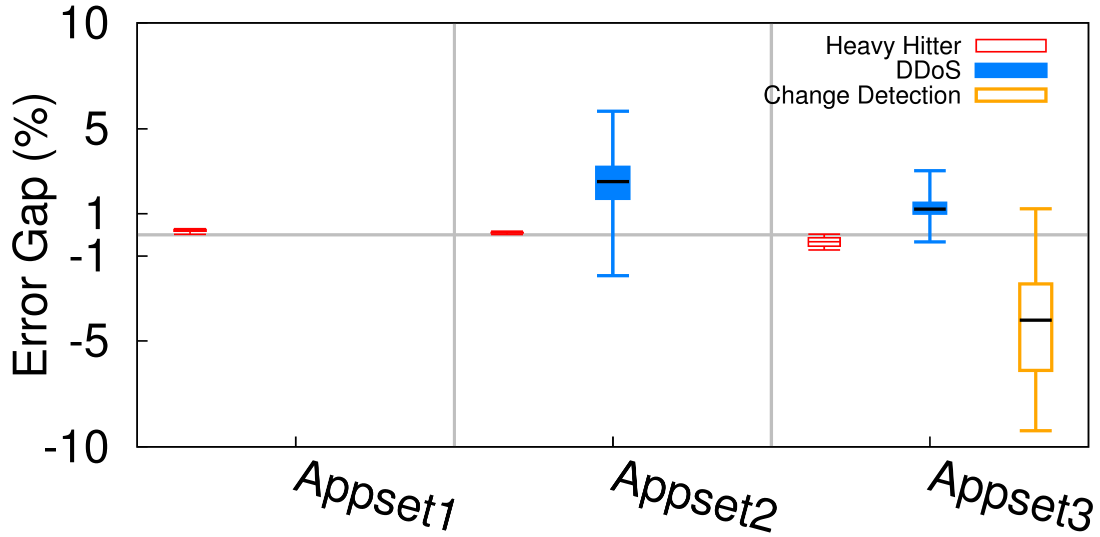
- ❖ Stable cross traces
- ❖ Error gap < 3.6%
- ❖ Good accuracy with limited memory



UnivMon
(Total 600KB)

OpenSketch
(600KB/task)

UnivMon is Better as Portfolio Grows!



{HH}   {HH,DDOS}   {HH,DDOS,Change}

Roadmap for this talk

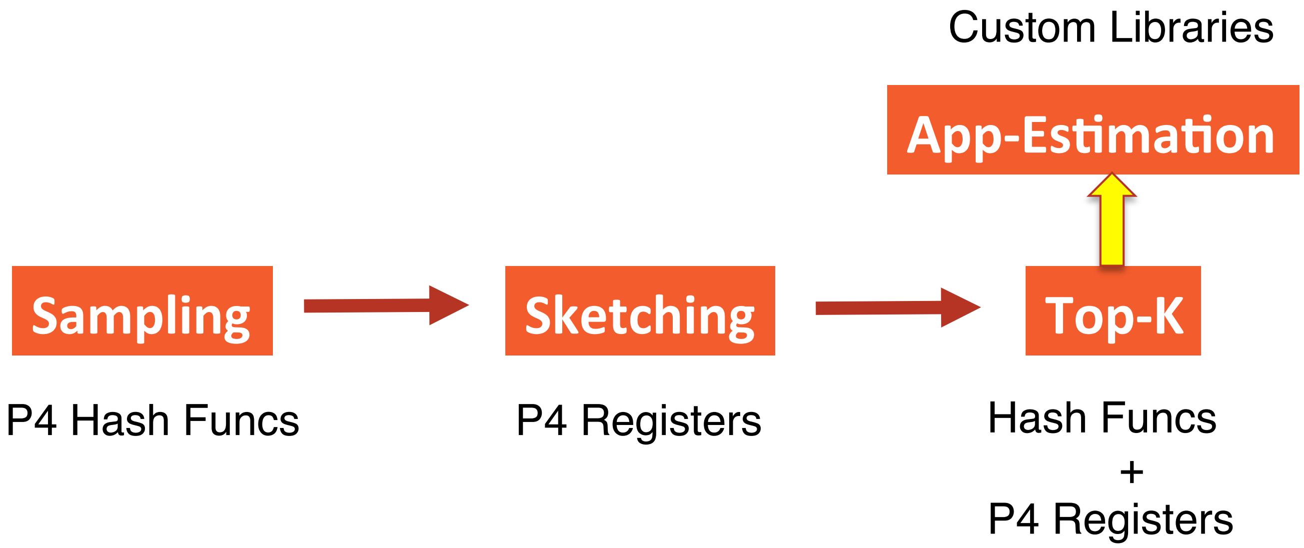
Does such a construction exist?

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

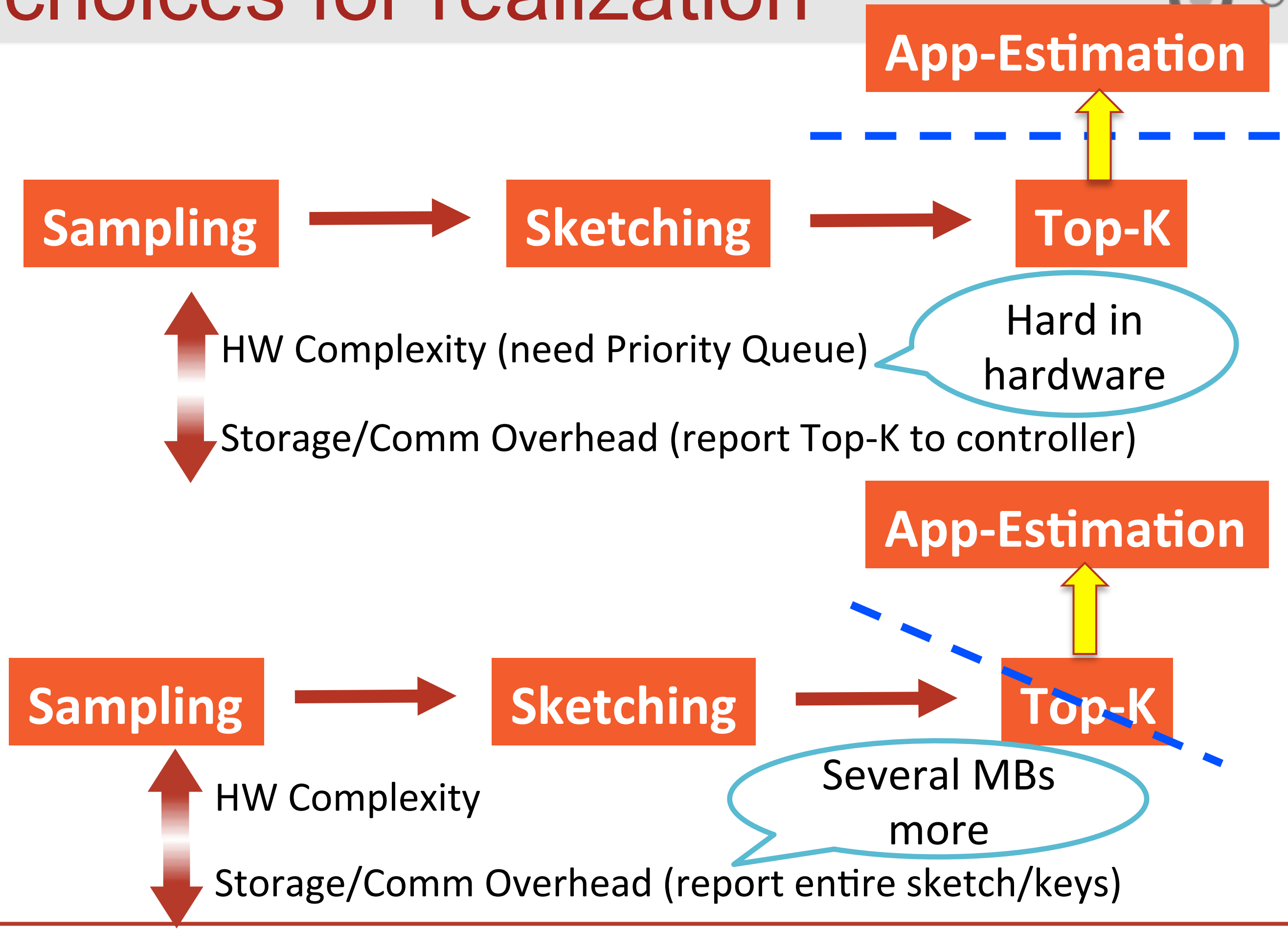
Is it competitive w.r.t. custom algorithms?

Is it feasible to implement?

Mapping Data Plane to P4



Design choices for realization



Implementation in Netronome: Step 1

Initial attempt: We tried with UnivMon P4 Code

Found out limitation of P4

- No Loop statement – out of space in Netronome
- Lack of Expressiveness, want to store seed values for hash. Store this at low level memory.

Implementation in Netronome: Step 2

So we switched to Micro-c capabilities

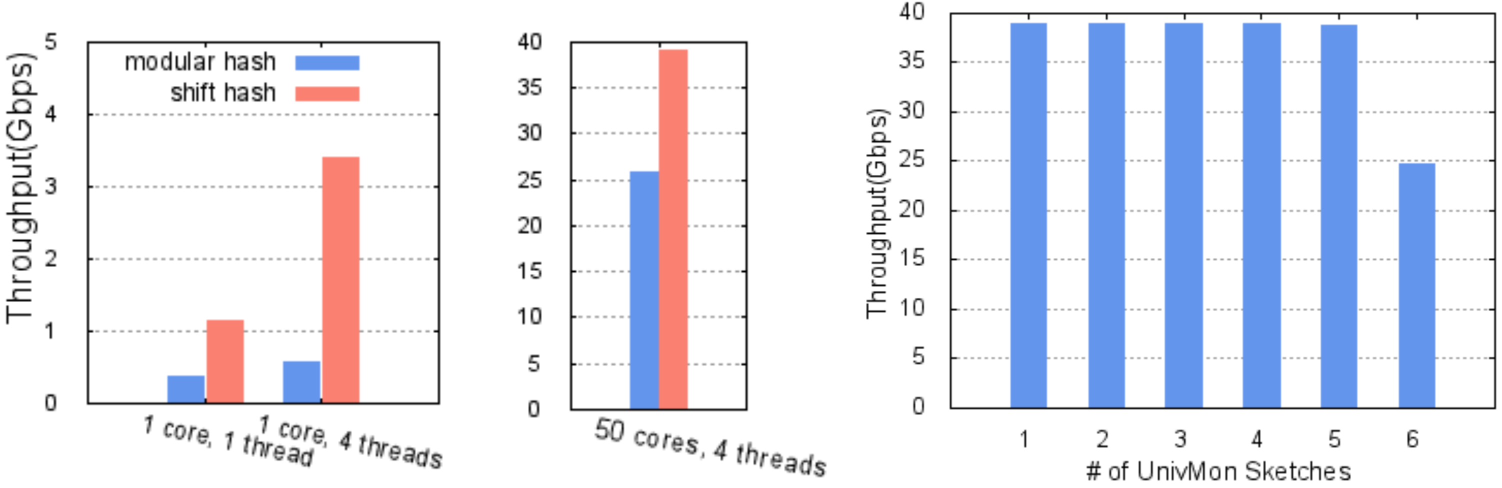
Some difficulties in porting/understanding APIs figuring out performance bottlenecks

We used the simulator to profile the bottleneck

Found out hash computation is the problem!

Optimizing hash operation

- Shift operation instead of modular operation
  - a,b : 64 bit random integer, x : 32bit key
  - Ha,b(x) = ((ax + b) % p) % m
  - Ha,b(x) = ((ax + b) >> 32) & n



Takeaway from basic improvements

- Shift operation is much faster than modular operation in Netronome
- UnivMon can exploit parellelism with Netronome. Atomic engine did a great job to solve synchronization issues with sketch counters
- Limitation : Shift operation can't guarantee enough randomness of hash functions and fair accuracy of sketching

Ideas: Use Tabular Hash

- Memory read is faster than modular operation and it has higher independence

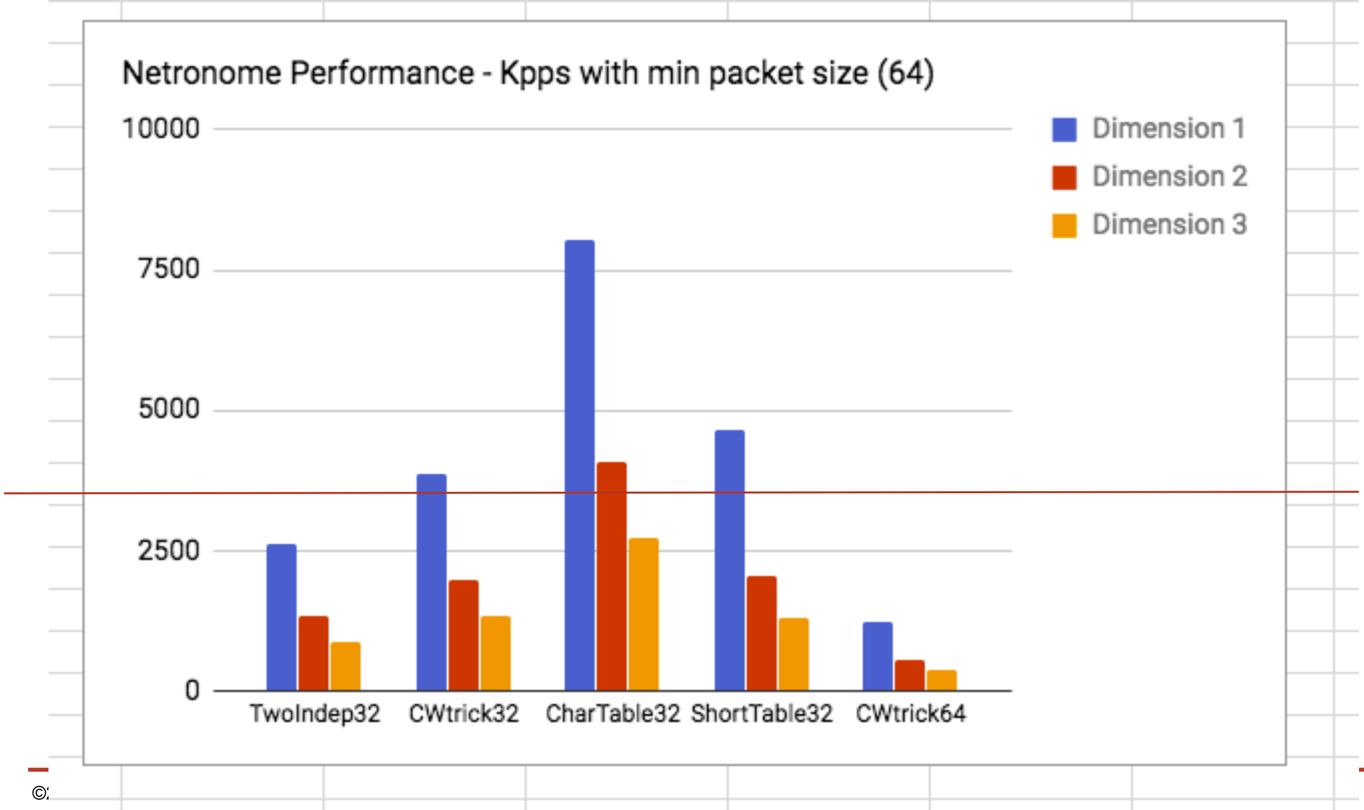
	32bit				64bit		
Name	Two Indep	CW trick 32	Char Table 32	Short Table 32	CW trick 64	Char Table 64	Short Table 64
Independent	2-independent	5-independent	5-independent	5-independent	5-independent	5-independent	5-independent
Key Idea	(ax+b) % p % m	multiplication & shift	tabulation	tabulation	multiplication & shift	tabulation	tabulation
Memory	0		20KB	384KB		12MB	100MB
Instructions		53 (8 multiplication)	37	12	243	85	37
memory lookup			7	3		15	7
Memory Needed for Netronome (X 27)			540KB	10MB		204MB	2.7GB
Implemented	O	O	O	O	O	X	X

2.7GB runs out of memory

204MB is possible to implement

Now all of tables are in the DRAM of NIC

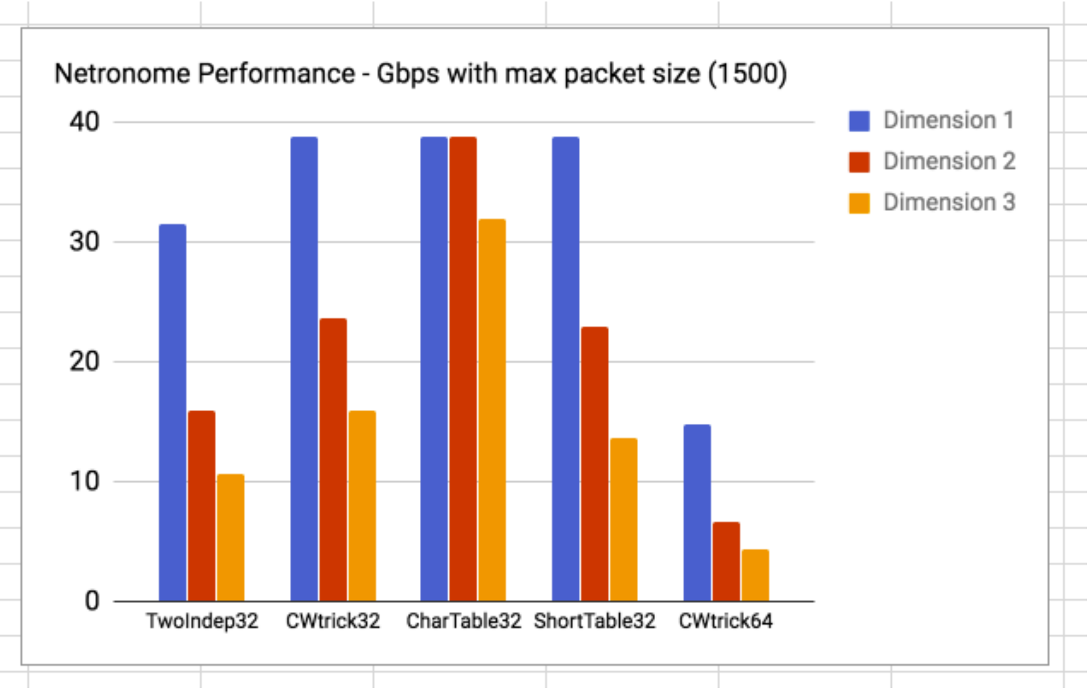
Tabular hash results - Kpps



Baseline(forwarding) max pps : 17300K pps

Line Rate with max packet size (1500) : 3245

Tabular hash throughput results



- Char Table 32 -> can cover 2 dimension with line rates
- 64bit is slower

Some lessons and takeaways

Simulator helped!

We could profile the bottleneck of our implementation with built-in simulator of IDE

UnivMon is feasible on NFP at line-rate

with 3 dimensions and 5-independent hash function

C programming with Netronome has greater flexibility!

Ongoing and Next Steps

APIs to write applications and queries on UnivMon

Suite of DDoS detection applications

Continue profiling and benchmarking

- Other platforms as well (e.g., openvswitch, fd.io)

Conclusions

- DDoS Detection needs more flexibility and programmability
- Today: General XOR Flexible
  Vision: General + Flexible via Universal Monitoring
- Initial promise: Feasible, accurate, possible to implement
- Ongoing and future work:
  Performance profiling, “Northbound” APIs etc.