New Directions in Traffic Measurement and Accounting

C. Estan and G. Varghese

Presented by
Aaditeshwar Seth

1

Need for traffic measurement

- Internet backbone monitoring
  - Short term
    - Detect DoS attacks
  - Long term
    - Traffic rerouting and link upgrades
    - Accounting for usage based pricing
- System tasks
  - Develop reports on *flow* usage: Size of *flows*

2

Relation to stream databases

- Large number of incoming tuples
- Map operator to differentiate between flows
- Sum operator to find size of flows

3

Issues

- Scalability with growing link speeds and number of flows
- Database solutions
  - Indexing and query optimization
- Networking solutions
  - Same, but not stated

4

Netflow

- Uses sampling
  - Inaccurate
- Tracks all flows
  - Resource intensive
  - Hence, expensive: Use *DRAM* → slower
  - Also, too much state needs to be reported back

5

[Estan and Verghese, 2002]

- Reduce the problem
  - Only track large flows
- #1: Identify large flows
  - Reduces resource requirements
- #2: For these flows, do accurate tracking
  - Reduced resource requirements allow faster and expensive *SRAM* to be used

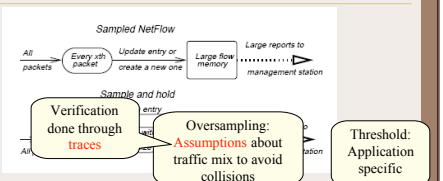
6

Overview

- Algorithms
  - Sample and hold
  - Multistage filters
- Optimizations
  - Entry preservation
  - Early removal
  - Shielding
  - Conservative update of counters
- Analytical comparisons
- Experimental comparisons
- Implementation issues

7

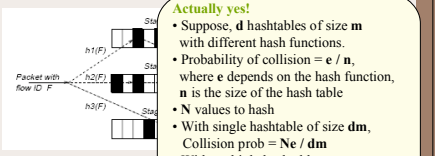
Sample and hold



- Implementation simplicity
- *P* = Probability of sampling a byte. Adjust this. How? P = O / T
- High *P* implies more *false positives*
- Low *P* implies more latency in tracking, false negatives with bursty traffic can creep in

8

Multistage filters



Actually yes!
- Suppose, **d** hashtables of size **m** with different hash functions.
- Probability of collision = **e / n**, where **e** depends on the hash function, **n** is the size of the hash table
- **N** values to hash
- With single hashtable of size **dm**, Collision prob = **Ne / dm**
- With multiple hashtables, Collision prob = **$(Ne / m)^d$**

- Higher accuracy
- **Having more hashtables to avoid [illegible] large hashtable: General theorem?**
- Reduces both false positives and false negatives
- Logarithmic scaling
- Measurement interval also depend on traffic mix. How to infer?

9

Optimizations

- Entry preservation
  - Retain all flows > T and all flows added in the current interval
  - Large flows are measured accurately
- Early removal
  - For flows added in the current interval, retain only flows > R
  - Prevents small flows from carrying forward
- Shielding
  - Do not hash large flows. Use a different hashtable for computing large flows?
  - Prevents piggy-backing of small flows on large flows
- Conservative update of counters
  - Update smallest counter. Trim other counters
  - Prevents false positives

10

Analytical comparisons

Measure	Sample and hold	Multistage filters	Sampling
Relative error	$\frac{\sqrt{2}}{Mz}$	$\frac{1+10 \cdot log_{10}(n)}{Mz}$	$\frac{1}{\sqrt{Mx}}$
Memory accesses	1	$1 + log_{10}(n)$	$\frac{1}{x}$

- Constraint: Only use **M** memory entries
- z: % of link capacity used by a flow
- r: implementation factor
- n: number of flows
- x: sampling rate (1 in x get sampled)
- *Accuracy comes at the cost of processing*

11

Analytical comparisons: Cont…

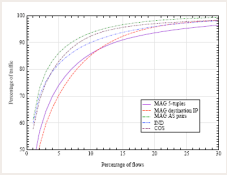
Measure	Sample and hold	Multistage filters	Sampled NetFlow
Exact measurements	$\lesssim longlived\%$	longlived%	0
Relative error	$1.41/O$	$\lesssim 1/u$	$0.0088/\sqrt{zt}$
Memory bound	$2O/z$	$2/z + 1/z \ log_{10}(n)$	min(n,486000 t)
Memory accesses	1	$1 + log_{10}(n)$	$1/x$

- Netflow memory not limited
- t: measurement interval
- O: oversampling
- u: zC/T = how much larger is a flow than the threshold

12

Experimental comparisons

- 3 traces
- 3 definitions of flows
  - 5 tuple: Src IP, Src port, Dst IP, Dst port, Proto
  - Dst IP
  - Src AS, Dst AS
- Few flows dominate
- Underutilized links



13

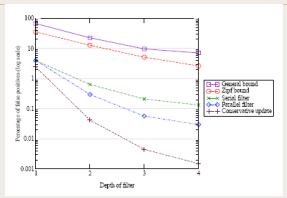
Experimental comparisons: Cont…

Algorithm	Maximum memory usage (entries)/ Average error				
	MAG 5-tuple	MAG destination IP	MAG+ AS pair	IND 5-tuple	COS 5-tuple
General bound	16,385 / 25%	16,385 / 25%	16,385 / 25%	16,385 / 25%	16,385 / 25%
Zipf bound	8,148 / 25%	7,441 / 25%	5,489 / 25%	6,303 / 25%	5,081 / 25%
Sample and hold	2,303 / 24.33%	1,964 / 24.07%	714 / 24.40%	1,313 / 23.83%	710 / 22.17%
+ preserve entries	3,832 / 4.67%	3,213 / 3.28%	1,038 / 1.32%	1,894 / 3.04%	1,017 / 6.61%
+ early removal	2,659 / 3.89%	2,294 / 3.16%	803 / 1.18%	1,525 / 2.92%	859 / 5.46%

- Relative error = Average error / Threshold
- Over estimation with general and Zipfian distribution is due to underutilized links
  - Bounds calculated using the link capacity

14

Experimental comparisons: Cont…



- Percentage of false positives decreases exponentially with depth of filter
- Tighter bounds
  - Bounds calculated using the maximum traffic

15

Experimental comparisons: Cont…

Group (flow size)	Unidentified flows / Average error		
	Sample and hold	Multistage filters	Sampled NetFlow
> 0.1%	0%/0.075%	0%/0.037%	0%/9.02%
0.1 . . . 0.01%	1.8%/7.09%	0%/1.090%	0.02%/22%
0.01 . . . 0.001%	77%/61.2%	55%/43.9%	18%/50.3%

- Flow Ids are 5 tuples
- Error = Sum of errors / Sum of flow sizes
- Algorithms good for large and very large flows

16

Experimental comparisons: Cont…

Group (flow size)	Unidentified flows / Average error		
	Sample and hold	Multistage filters	Sampled NetFlow
> 0.1%	0%/0.0%	0%/0.0%	0%/4.88%
0.1 . . . 0.01%	0%/0.002%	0%/0.001%	0.0%/15.3%
0.01 . . . 0.001%	0%/0.165%	0%/0.144%	5.7%/39.9%

- Flow Ids are (Src AS, Dst AS)
- Few large flows
- Algorithms still work well

17

Implementation issues

- Estimation of parameters
  - ADAPTTHRESHOLD algorithm
    - Utilize up to 90% of hash table
  - Leads to graceful degradation in case of over use, greater number of flow statistics in case of under use
  - Similar to dynamic adjustment of the window size: Control sampling rate of tuples to drop
- Do the magic values work only on traces, or are they generic?

18

Conclusions and discussions

- Complete solution: algorithms, analytical evaluation, experimental evaluation, implementation design
- Exploitation of specific facts about distribution can simplify the problem for certain applications

- Argument of cheap DRAM Vs expensive SRAM is not good
  - Not considered Netflow optimizations to reduce resource usage
  - Assume that per packet sampling can be done through SRAM, and it scales with line speeds. Verification?
- Examples in proofs do not tell anything!
- Too many magic values. Suggests the need for frequent fine tuning. Automation?

19