Software-Defined Traffic Measurement with OpenSketch

Lavanya Jose
Stanford University

Joint work with Minlan Yu and Rui Miao at USC
Management is Control + Measurement

control

- Access Control
- Routing

measure

- DDoS
- Flow Size Distribution
Questions we want to ask

1. Who’s sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is someone doing a port scan?
4. Is someone being DDoS-ed?
5. Who’s getting traffic from blacklisted IPs?
6. How many people downloaded files from 10.0.2.1?
Switches are great at counting per flow bytes and packets

- NetFlow and sFlow sample packets
- NetFlow maintains per flow byte and packet counts
- Can find count of a particular flow, prefix or counts of heavy flows
Problem: NetFlow counts can’t answer my questions

Is someone doing a port scan?

NetFlow samples packets from heavy flows. Missed packets from small “port scanners”.

- Increase sampling rate --> inefficient
Streaming algorithms

+ Process efficiently at line rate
+ Accurate answers
- But each answers a specific question
What measurement architecture can answer all my questions?

1. Who’s sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is someone doing a port scan?
4. Is someone being DDoS-ed?
5. Who’s getting traffic from blacklisted IPs?
6. How many people downloaded files from 10.0.2.1?
SDN Model: Find Building Blocks

1. Who’s sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is someone doing a port scan? ...
Sketches as building blocks

- Sketch
  - Data structure
  - Support approx. computing some function of data
  - Much smaller than actual data
  - Streaming, small per-item processing cost
  - Provable space-accuracy tradeoffs
Sketches as building blocks

e.g., Count Min sketch
to store counts of frequent source IP addresses

Packet

<table>
<thead>
<tr>
<th>$h_1$</th>
<th>$h_2$</th>
<th>$h_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>23</th>
<th>0</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>22</td>
<td>5</td>
</tr>
</tbody>
</table>

(Cormode 2005)
Sketches as building blocks

e.g., Count Min sketch to store counts of frequent source IP addresses

Source IP address: 23.43.12.1

(Cormode 2005)
Sketches as building blocks

e.g., Count Min sketch to store counts of frequent source IP addresses

Source IP address: 23.43.12.1

h1

h2

h3

process

(Cormode 2005)
Sketches as building blocks

e.g., Count Min sketch
to store counts of frequent source IP addresses

Source IP address: 23.43.12.1

(Cormode 2005)
Sketches as building blocks

e.g., Count Min sketch
to store counts of frequent source IP addresses

Source IP address : 23.43.12.1

(Cormode 2005)
Sketches as building blocks

e.g., Count Min sketch
to store counts of frequent source IP addresses

\[
\begin{array}{cccccc}
\text{h1} & \text{h2} & \text{h3} \\
3 & 23 & 2 \\
2 & 4 & 3 \\
1 & 9 & 0 \\
24 & 3 & 4 \\
0 & 2 & 23 \\
4 & 1 & 5 \\
\end{array}
\]

(Cormode 2005)
Sketches as building blocks

e.g., Count Min sketch
to store counts of frequent source IP addresses

# packets from 23.43.12.1?

(Cormode 2005)
Sketches as building blocks
e.g., Count Min sketch
to store counts of frequent source IP addresses

# packets from 23.43.12.1?

query

estimate

(Cormode 2005)
Sketches as building blocks

e.g., Count Min sketch
to store counts of frequent source IP addresses

<table>
<thead>
<tr>
<th></th>
<th>h1</th>
<th>h2</th>
<th>h3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

# packets from 23.43.12.1?

query

estimate

pick min.

(Cormode 2005)

Friday, April 5, 13
Sketches as building blocks

e.g., Count Min sketch
to store counts of frequent source IP addresses

within $\varepsilon$ total packets with high probability

$\varepsilon = \frac{e}{\text{no. of counters}}$

$\Pr\{\text{error} > \varepsilon \text{ total packets}\} < e^{-\text{no. of hash functions}}$

+ Provable space-accuracy tradeoffs

(Cormode 2005)
Sketches as building blocks

Counting, storing statistics

Picking packets to measure

Identifying heavy “keys”

(Reversible Sketch: Schweller 2004)
...answer many questions

1. Who’s sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is someone doing a port scan?
4. Is someone being DDoS-ed?
5. Who’s getting traffic from blacklisted IPs?
6. How many people downloaded files from 10.0.2.1?

(Reversible Sketch: Schweller 2004)
But each sketch estimates only one function

- frequency count
- cardinality
- set membership
- heavy “keys”
3-stage pipeline

- frequency count
- cardinality
- set membership
- heavy “keys”
3-stage pipeline

- frequency count
- cardinality
- set membership
- heavy "keys"

Hash
Classify
Count
3-stage pipeline

Packet

Hash

Classify

Count

pick fields to hash

hash values

pick field to match

compute counter addresses

header fields

hash values

header fields

header fields

header fields
3-stage pipeline

- frequency count
- cardinality
- set membership
- heavy “keys”

Hash
Classify
Count
3-stage pipeline

- Identifying heavy “keys”
- Counting, storing statistics
- Picking packets to measure
3-stage pipeline

1. Who’s sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is someone doing a port scan?

Identifying heavy “keys”
Counting, storing statistics
Picking packets to measure

Hash
Classify
Count
OpenSketch Measurement Framework

1. Who’s sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is ..

<table>
<thead>
<tr>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifying heavy “keys”</td>
</tr>
<tr>
<td>Counting, storing statistics</td>
</tr>
<tr>
<td>Picking packets to measure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash</td>
</tr>
<tr>
<td>Classify</td>
</tr>
<tr>
<td>Count</td>
</tr>
</tbody>
</table>
OpenSketch Measurement Framework

1. Who’s sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is ...

Measurement Library

Controller

Data Plane

Hash  Classify  Count
1. Who's sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is ..
1. Who's sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?

OpenSketch Measurement Framework

Controller

Measurement Programs

Measurement Library

Data Plane

Hash
Classify
Count

# counters, size, update type, addr. calculation

#, field, range

(match, action)
OpenSketch Measurement Framework

Controller

Measurement Programs

Measurement Library

Data Plane

Hash

Classify

Count

Identifying heavy "keys"

Counting, storing statistics

Picking packets to measure

Measurement Library

Measurement Programs

Controller

Data Plane

Hash

Classify

Count

1. Who's sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is ..
Details

• Implementing sketches with the Pipeline
• Configuring the Pipeline
• Evaluation and NetFPGA prototype
Count Min Sketch with the Pipeline to store counts of frequent source IP addresses.

Source IP address: 23.43.12.1

Hash

process

Count

<table>
<thead>
<tr>
<th>h1</th>
<th>h2</th>
<th>h3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>23</td>
<td>5</td>
</tr>
</tbody>
</table>
Bitmap Sketch with the Pipeline

to store number of different destination port numbers

Packet

process

Hash

Count

h

00100100010

Counting, storing statistics

Friday, April 5, 13
Bitmap Sketch with the Pipeline

to store number of different destination port numbers

Destination port number: 5596

Hash: 0010010010

Counting, storing statistics

process

Hash

Count
Bitmap Sketch with the Pipeline

to store number of different destination port numbers

Destination port number : 5596

\[ h \rightarrow \begin{array}{cccccccc}
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
\end{array} \]

process

Hash

Count
Bitmap Sketch with the Pipeline
to store number of different destination port numbers

Counting, storing statistics

# different destination port numbers?

query

1 0 1 0 0 1 0 0 1 0

(Whang 1990)
Bitmap Sketch with the Pipeline to store number of different destination port numbers.

Counting, storing statistics

# different destination port numbers?

query

estimate

Six counters out of ten are 0.

estimate

\[ N = -10 \ln(6/10) = 5 \]

(Whang 1990)

\[ \begin{array}{cccccccc}
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
\end{array} \]
Other Sketches

- K-ary Sketch for heavy changes
- Bloom Filter Sketch to check set membership
- PCSA sketch to count distinct values

(Schweller 2004; Goel 2010; Flajolet 1985)
Efficient implementation of 3-stage pipeline

- **Hash**: hash in parallel
- **Classify**: TCAM rules
- **Count**: cheap fast memory MBs of SRAM
Similar functions, diverse configurations

- Hash: ?? hash functions
- Classify: ?? TCAM entries for classify rules
- Count: ?? MBs of SRAM for tables of counters
Similar functions, diverse configurations

- Count Min: 3
- Bloom Filters: 7-8
- Fixed size reversible sketch: 5
- Can share hash functions

4-8 simple hash functions per question
Similar functions, diverse configurations

Classify

30-40 TCAM entries per question maximum

- Match a prefix/ value: 1 rule
- Match a set of values: Bloom Filters
Similar functions, diverse configurations

Count

From simulation and worst case bounds for different tasks

up to 8MB SRAM
Similar functions, diverse configurations

Count

up to 8MB SRAM

Change detection

2MB for 1.7% err

relative error (%)

SRAM (MB)
Similar functions, diverse configurations

**Count**

up to 8MB SRAM

Change detection

7.5MB for 0.5% err
Measurement tasks

1. Who’s sending a lot to 10.0.2.0/16? (Heavy Hitters)
2. How are flow sizes distributed?
3. Is someone doing a port scan?
4. Is someone being DDoS-ed?
5. Who’s getting traffic from blacklisted IPs?
6. How many people downloaded files from 10.0.2.1?

More efficient than NetFlow (Heavy Hitters)

Efficient - needs 1/4th as much memory as NetFlow for 4% f.p.
More efficient than NetFlow (Heavy Hitters)

Accurate - with 600KB, OpenSketch has less than 0.05% f.p. NetFlow has around 3%

![Graph showing comparison between NetFlow and OpenSketch for False Positives vs Switch memory size (KB)]
OpenSketch NetFPGA Prototype

• 3-stage meas. pipeline parallel to forwarding
• Full throughput 1Gbps @ 4 ports
• Measurement pipeline takes fewer cycles than forwarding
Conclusion

• Current switches good for flow statistics
• But they don’t answer basic measurement questions
• Like identify heavy hitters, detect DDoS attacks, port scans, traffic from blacklisted IP address etc.
Takeaway

• Hash, classify and count pipeline in the Data Plane
• And sketch based building blocks in the Control Plane
• Make measurement in switches efficient and easy