ProgME: Towards Programmable Network Measurement

Lihua Yuan, Chen-Nee Chuah, Prasant Mohapatra
University of California, Davis

Outline

• Measurement Architectures and Challenges

• Flowset – A new measurement abstraction

• Programmable Measurement Architecture
  1 Program Engine
  2 Query Answering Engine
  3 Adaptive Engine (Heavy-hitter identification)

• Evaluation & Discussion
Existing Measurement Abstractions

Flow: Every bits significant, specifies a dot

Superflow: “regular expression” of flows, specifies a regular shape

Flow-based Measurement

- Per-flow counters + Post-processing
- Scalability issues
  - Expensive faster SRAM vs. cheap slower DRAM
  - Choose flows to ignore (biased statistics)
Our Observation

- Flow and superflow are not expressive/flexible to map to real world complexities
- We use descriptive abstractions
  - Population of Kyoto vs. population of 50km²
- Defined flexibly to suit our purpose
  - Any country/province in rectangular shape?

Outline

- Measurement Architectures and Challenges
- Flowset – A new measurement abstraction
- Programmable MEasurement Architecture
  1. Program Engine
  2. Query Answering Engine
  3. Adaptive Engine (Heavy-hitter identification)
- Evaluation & Discussion
Flowset – A New Abstraction

• **Flowset**: Arbitrary set of flows, define any shape, continuous or not

• Native mapping to any user queries
  - “the volume of DDoS traffic” or
  - “the volume of traffic going to ISP A”
  - Flexibly defined by users

• Scale to #queries vs. #flows
  - Thousands vs. millions
  - Human vs. computer

Flowset – Underlying Data Structure

• Encoded using Binary Decision Diagram (BDD)
  - Bit field in IP header -> bit variable in BDD
  - Canonical representation of binary decision tree
  - Easy set operations ( \ ; [ ; ; ; ] ; ! )
Flowset-based Query Answering Engine (FQAE)

1. Query interpretation
2. Map packets to a flowset and count
3. Dynamic re-program FQAE
Flowset Composition Language

- Use regular superflow as the primitive
  - Popular among network administrators
- Use set algebra to compose any flowset

\[
Q_1 = S_1 \setminus S_2 \setminus S_3
\]
\[
Q_2 = P_1 \cap P_2
\]

Disentangle Queries

- Set algebra to compute independent sub-queries
- Goal: Ensure packets map to exactly one sub-query
- Motivation: help FQAE (next 4 slides)
- Worst case complexity is \( O(2^n) \)
- Optimistic based on study on router/firewall configs
  - Small percentage of rules correlates to each other
  - Ave. correlation size is 3~4

\[
D_1 = Q_1 \cap Q_2
\]
\[
D_2 = Q_1 \cap Q_2
\]
\[
D_3 = Q_2 \cap Q_1
\]
\[
D_4 = : Q_1 : Q_2
\]
Flowset Packet Matching

- Sequential mapping
  1. Build a flowset representation for the packet
  2. Check if set implication holds
  3. Iterate until a matching flowset is found

- Improvement from sequential mapping
  - **Disentangle**: ensures exactly one matching flowset
  - Stop immediately once a matching is found
  - **HashReduce**: reduce the number of candidates
  - **TrafficSort**: traffic-based dynamic optimization
HashReduce

- Hash(): choose some bits as hash key
- Find candidates for each key to build TMC
- Choose good hash function
  - Avoid bits unused in all flowsets
  - Compare several hash functions offline

TrafficSort

- Re-order matching candidates based on traffic

   - Trivial to sort with independent sub-queries
   - NP-complete if queries have dependencies
**ProgME Architecture**

1. Query interpretation
2. Flowset-based Query Answering Engine
   - Map packets to a flowset and count
3. Dynamic re-program FQAE

**Adaptive Engine**

- Dynamically re-program FQAE
  - Learn from previous results
  - Adapt to traffic condition
- A sample application: heavy hitter identification
  - Relative weight threshold \( f_w > \mu \)
  - Largest-K, rate threshold
- HHI need to co-exist with general static measurement
- Ignoring mice produce biased statistics
  - Loss of accuracy on queries dominated by mice
  - ICMP, DNS, DDoS
Multi-Resolution Tiling

- Simple inference, binary decision
- Zoom in until a flow is identified
- Max flowsets to track: $\frac{1}{\mu}$
- Rectangles just for illustration!

Zoom Strategy

- Zoom factor
  - Tradeoff between memory & speed
- Variable zoom
  - search for elephants with an emphasis
Outline

• Measurement Architectures and Challenges

• Flowset – A new measurement abstraction

• Programmable MEasurement Architecture

  1. Program Engine
  2. Query Answering Engine
  3. Adaptive Engine (Heavy-hitter identification)

• Evaluation & Discussion

FQAE: Discussion on Accuracy

• Per-query accuracy matters more than per-flow accuracy

• Ignoring mice sacrifices worst-case accuracy to improve average accuracy

• Queries dominated by mice have practical importance
  - DNS, ICMP traffic
  - Below-the-radar traffic

ProgME uses pre-aggregation and has equal accuracy to ideal per-flow measurement
Scalability - #Counters

<table>
<thead>
<tr>
<th>sip</th>
<th>dip</th>
<th>sip, dip</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>53,191</td>
<td>214,411</td>
</tr>
<tr>
<td>#2</td>
<td>53,762</td>
<td>127,543</td>
</tr>
</tbody>
</table>

CAIDA OC-48 Traces

<table>
<thead>
<tr>
<th>Log (Orig/Disj)</th>
<th>All (Orig/Disj)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 19/22</td>
<td>40/55</td>
</tr>
<tr>
<td>#2 0/0</td>
<td>35/38</td>
</tr>
<tr>
<td>#3 0/0</td>
<td>800/845</td>
</tr>
</tbody>
</table>

Real Firewall Rules

- We infer the usage pattern of ProgME from real firewall/router rules
- \#flowsets << \#flows

Case Study - Measure Traffic Demand to an ISP

- [sigcomm'00] per-flow + post-processing
- Found 84132 prefixes in BGP table
- Union them into a single flowset
- Encoding cost peaked ~70K nodes (1.4MB)
Conclusion

• Flowset
  - Flexible abstraction
  - Flowset composition using set algebra
  - BDD-based data structure
• Static program
  - Users defines what to measure
  - Scalable, accurate
  - Disentangle, HashReduce, TrafficSort
• Adaptive program (Heavy hitter identification)
  - Multi-resolution zoom
  - Sequential hypothesis test

Thanks for your attention!
&
Questions?