

ProgME: Towards Programmable Network MEasurement



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

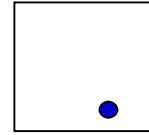
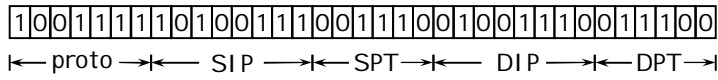
1

Outline

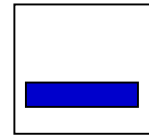
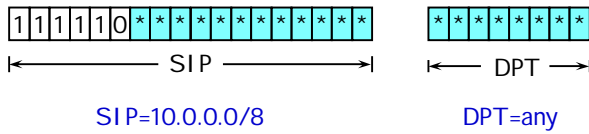
- Measurement Architectures and Challenges
- Flowset – A new measurement abstraction
- Programmable MEasurement Architecture
 - 1 Program Engine
 - 2 Query Answering Engine
 - 3 Adaptive Engine (Heavy-hitter identification)
- Evaluation & Discussion

2

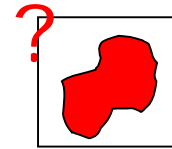
Existing Measurement Abstractions



Flow: Every bits significant, specifies a dot

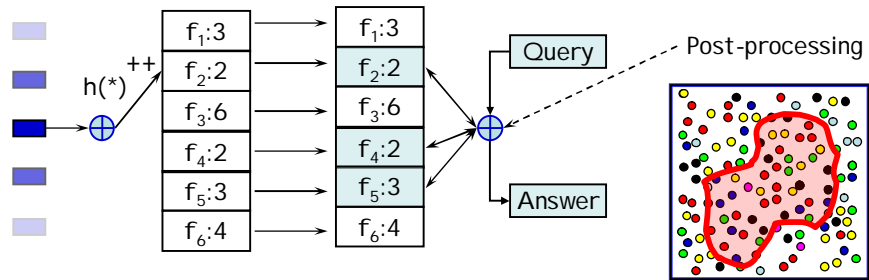


Superflow: "regular expression" of flows, specifies a regular shape



3

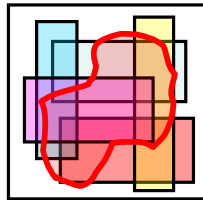
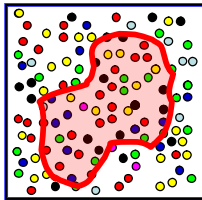
Flow-based Measurement



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

4

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?

5

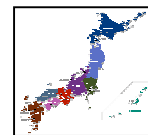
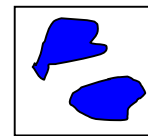
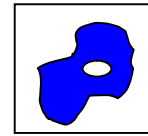
Outline

- 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

6

Flowset – A New Abstraction

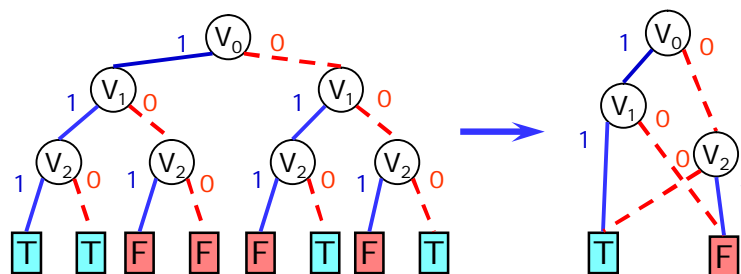
- **Flowset:** Arbitrary set of flows, define any shape, continuous or not
- Native mapping to any user queries
 - “the volume of DDoS traffic” or
 - “the volume of traffic going to ISP A”
 - Flexibly defined by users
- Scale to #queries vs. #flows
 - Thousands vs. millions
 - Human vs. computer



7

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 (\setminus ; $;$; $;$; n ; $!$)

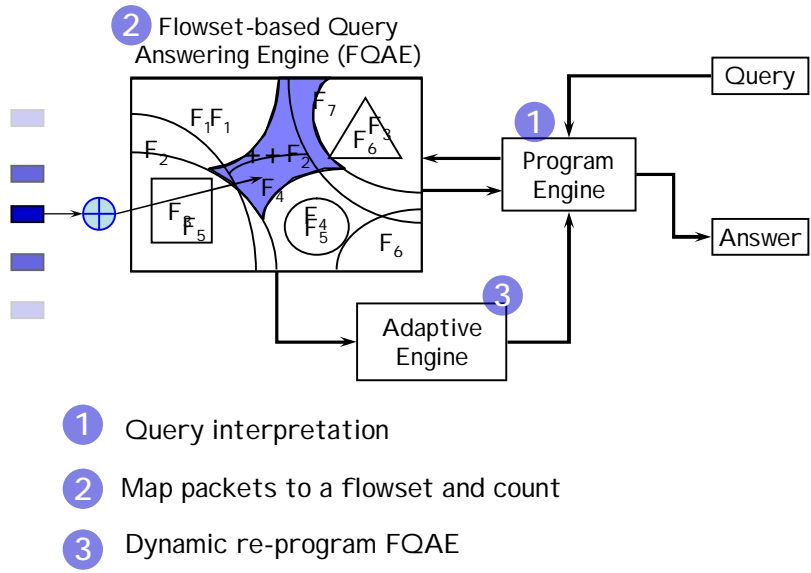


Binary Decision Tree

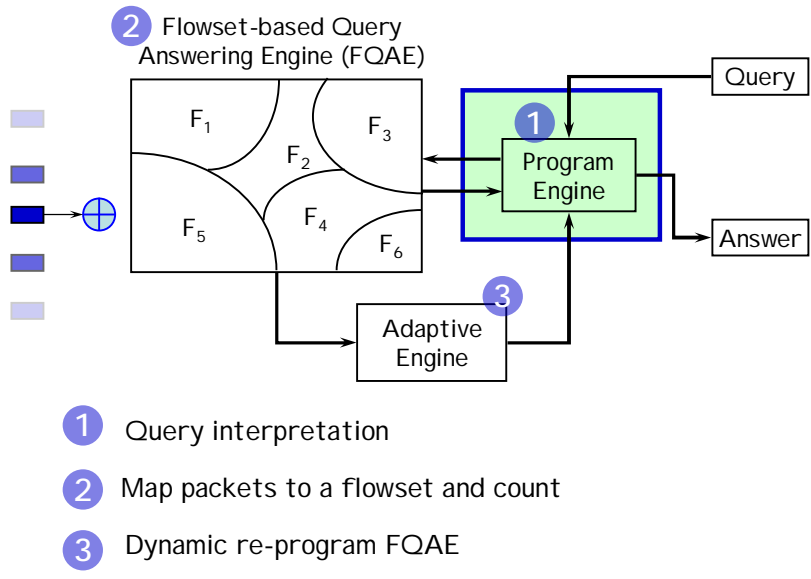
Binary Decision Diagram

8

ProgME Architecture



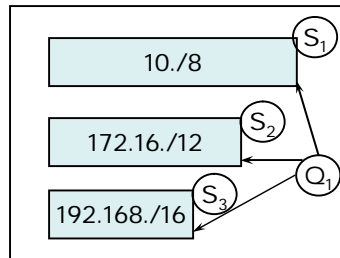
ProgME Architecture



1

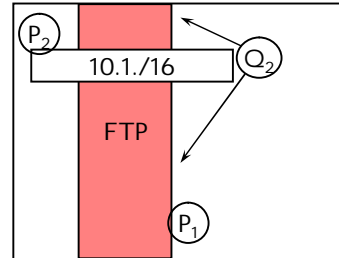
Flowset Composition Language

Q_1 : Packets from private IP



$$Q_1 = S_1 \cup S_2 \cup S_3$$

Q_2 : FTP packets except 10.1./16



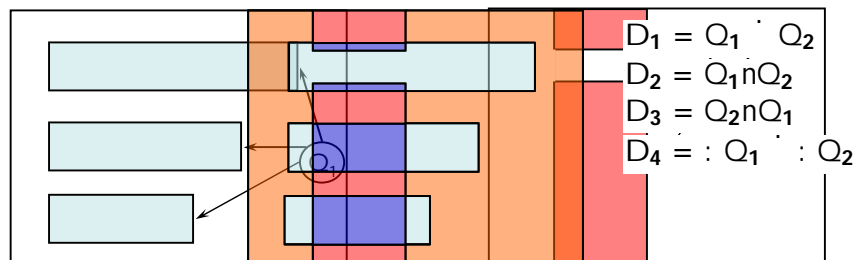
$$Q_2 = P_1 \cap P_2$$

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

11

1

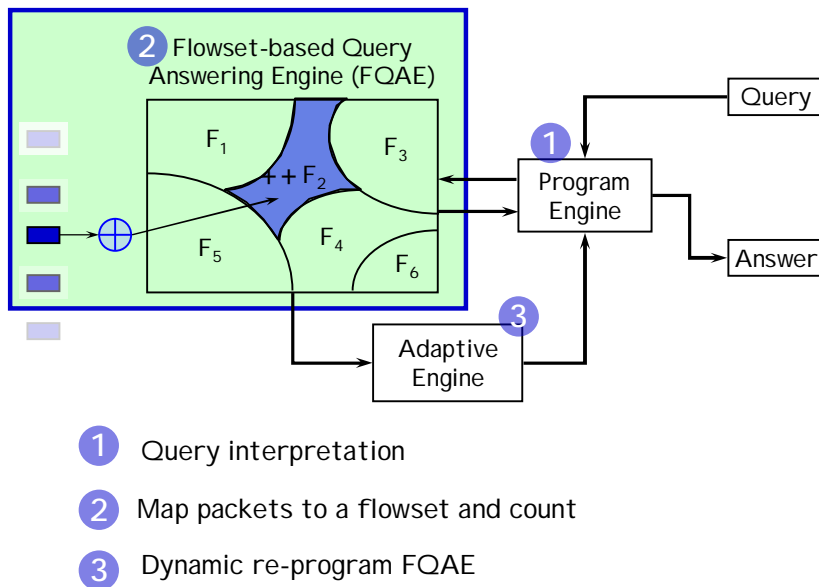
Disentangle Queries



- Set algebra to compute **independent sub-queries**
- Goal: Ensure packets map to exactly one sub-query
- Motivation: help FOAE (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

12

ProgME Architecture



13

Flowset Packet Matching

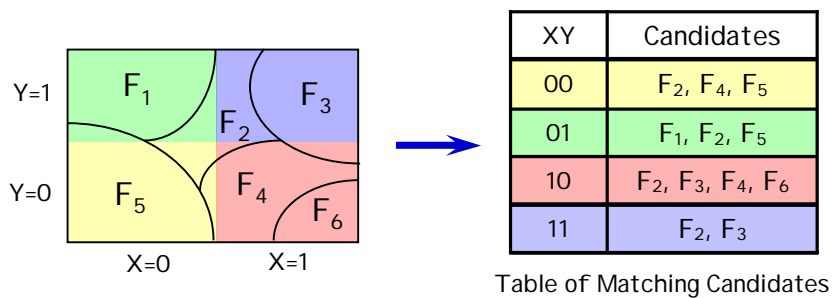
- Sequential mapping
 1. Build a flowset representation for the packet
 2. Check if set implication holds
 3. Iterate until a matching flowset is found
- Improvement from sequential mapping
 - **Disentangle**: ensures exactly one matching flowset
 - § Stop immediately once a matching is found
 - **HashReduce**: reduce the number of candidates
 - **TrafficSort**: traffic-based dynamic optimization

14

2

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



15

2

TrafficSort

- Re-order matching candidates based on traffic

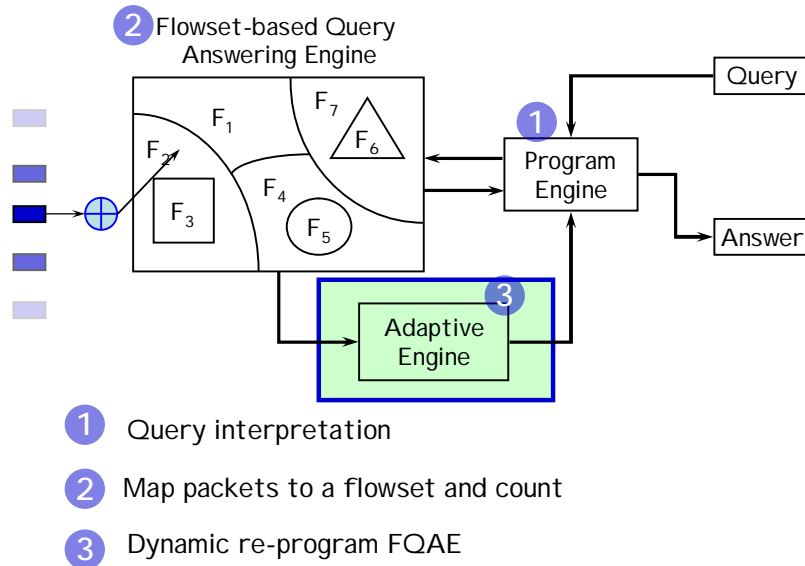
$$\text{00} \rightarrow \boxed{F_2:1} \boxed{F_4:2} \boxed{F_5:99} \quad 99 \cdot 3 + 2 \cdot 2 + 1 = 302$$

$$\text{00} \rightarrow \boxed{F_4:2} \quad 99 \cdot 1 + 2 \cdot 2 + 1 \cdot 3 = 106$$

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

16

ProgME Architecture



17

3

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

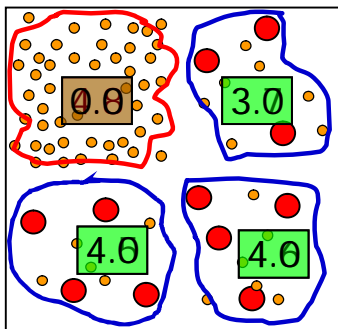
18

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

21

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

22

Scalability – #Counters

	sip	dip	sip, dip
#1	53,191	214,411	336,463
#2	53,762	127,543	293,519

CAI DA OC-48 Traces

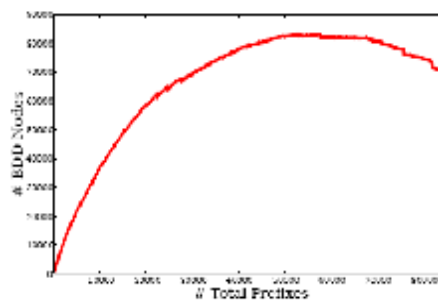
	Log (Orig/Disj)	All (Orig/Disj)
#1	19/22	40/55
#2	0/0	35/38
#3	0/0	800/845

Real Firewall Rules

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

23

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 \sim 70K nodes (1.4MB)

24

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](#)

25

Thanks for
your attention!
&
Questions?

26