DREAM: Dynamic Resource Allocation for Software-defined Measurement

(SIGCOMM’14)

Masoud Moshref, Minlan Yu, Ramesh Govindan, Amin Vahdat
Measurement is Crucial for Network Management

Tenants:
- Netflix
- Expedia
- Reddit

Management:
- Accounting
- Failure Detection
- Traffic Engineering

Measurement:
- Heavy Hitter detection
- Change detection

Network:

Motivation | System | Algorithm | Evaluation
High Level Contribution: Flexible Measurement

Management:
Users **dynamically instantiate** complex measurements on network state

Measurement:
DREAM supports **the largest number** of measurement tasks while **maintaining measurement accuracy**, by dynamically leveraging tradeoffs between switch resource consumption and measurement accuracy

Network:
We leverage **unmodified hardware** and existing switch interfaces
Prior Work: Software Defined Measurement (SDM)

**Motivation**

- **Controller**
  - Heavy Hitter detection
  - Change detection
  - 3 Update rules
  - 2 Fetch counters

- **Source IP:** 10.0.1.130/31
- **Source IP:** 55.3.4.32/30
- **#Bytes=1M**
- **#Bytes=5M**
Our Focus: Measurement Using TCAMs

Existing OpenFlow switches use TCAMs which permit counting traffic for a prefix.

Focus on TCAMs enables immediate deployability.

Prior work has explored other primitives such as hash-based counters.
Challenge: Limited TCAM Memory

Find source IPs sending > 10Mbps

Controller

Heavy Hitter detection

Problem: Requires too many TCAMs
64K IPs to monitor a /16 prefix >> ~4K TCAMs at switches
Reducing TCAM Usage

Monitor internal nodes to reduce TCAM usage

- Monitoring 1* is enough because a node with size 5 cannot have leaves >10
Challenge: Loss of Accuracy

Fixed configuration misses heavy hitters as traffic changes

Motivation

- Fixed configuration misses heavy hitters as traffic changes.
Dynamic Configuration to Avoid Loss of Accuracy

Find leaves >10Mbps using 3 TCAMs

Monitor children to detect HHs but using 2 TCAMs

Monitor parent to save a TCAM
Reducing TCAM Usage: Temporal Multiplexing

**Motivation**

- **Task 1**
- **Task 2**

![Diagram showing Required TCAM changes over time](image-url)
Reducing TCAM Usage: Spatial Multiplexing

Required TCAMs varies across switches

Switch A
Switch B

Only needs more TCAMs at switch A
Reducing TCAM Usage: Diminishing Returns

Can accept an accuracy bound <100% to save TCAMs
Leverage **spatial and temporal multiplexing** and **diminishing returns** to dynamically adapt the **configuration and allocation** of TCAM entries per task to achieve **sufficient accuracy**
DREAM Contributions

System

Supports concurrent instances of three task types: Heavy Hitter, Hierarchical HH and Change Detection

Algorithm

Dynamically adapts tasks TCAM allocations and configuration over time and across switches, while maintaining sufficient accuracy

Evaluation

Significantly outperforms fixed allocation and scales well to larger networks
DREAM Tasks

**Management**
- Anomaly detection
- Traffic engineering
- Network provisioning
- Accounting
- Network visualization
- DDoS detection

**Measurement**
- Heavy Hitter detection
- Hierarchical HH detection
- Change detection

**Network**

Motivation  System  Algorithm  Evaluation
Algorithmic Challenges

Dynamically adapts tasks TCAM allocations and configuration over time and across switches, while maintaining sufficient accuracy.

How to allocate TCAMs for sufficient accuracy?

Which switches to allocate?

How to adapt TCAM configuration on multiple switches?

- Diminishing Return
- Temporal Multiplexing
- Spatial Multiplexing
Dynamic TCAM Allocation

Allocate TCAM  
Estimate accuracy

Measure

Enough TCAMs $\rightarrow$ High accuracy $\rightarrow$ Satisfied

Not enough TCAMs $\rightarrow$ Low accuracy $\rightarrow$ Unsatisfied
Dynamic TCAM Allocation

Allocate TCAM → Estimate accuracy

Why iterative approach?

We cannot know the curve for every traffic and task instance.
Thus we cannot formulate a one-shot optimization.
Dynamic TCAM Allocation

Why iterative approach?
We cannot know the curve for every traffic and task instance
Thus we cannot formulate a one-shot optimization

Why estimating accuracy?
We don’t have ground-truth
Thus we must estimate accuracy
Estimate Accuracy: Heavy Hitter Detection

**Precision** = $\frac{\text{True detected HH}}{\text{Detected HHs}}$

Is 1 because any detected HH is a true HH

**Recall** = $\frac{\text{True detected HH}}{\text{True detected + Missed HHs}}$

Estimate missed HHs
Estimate Recall for Heavy Hitter Detection

Recall = \frac{\text{True detected HH}}{\text{True detected} + \text{Missed HHs}}

Find an upper bound of missed HHs using size and level of internal nodes

With size 26: missed <= 2 HHs

Threshold = 10Mbps

At level 2: missed <= 2 HH

Motivation  System  Algorithm  Evaluation

22
Allocate TCAM

Goal: maintain high task *satisfaction*

Fraction of task’s lifetime with sufficient accuracy
Allocate TCAM

Goal: maintain high task satisfaction

How many TCAMs to exchange?

Small $\rightarrow$ Slow convergence

Large $\rightarrow$ Oscillations

Motivation | System | **Algorithm** | Evaluation
---|---|---|---

Avoid Overloading

Not enough TCAMs to satisfy all tasks

Solutions

Reject new tasks
Drop existing tasks
Algorithmic Challenges

Dynamically adapts tasks TCAM allocations and configuration over time and across switches, while maintaining sufficient accuracy.

How to allocate TCAMs for sufficient accuracy?

Which switches to allocate?

How to adapt TCAM configuration on multiple switches?

Diminishing Returns

Temporal Multiplexing

Spatial Multiplexing
A task can have traffic from multiple switches.
Allocate TCAM: Multiple Switches

A task can have traffic from multiple switches

Controller

Heavy Hitter detection

Global accuracy is important
If a task is globally satisfied, no need to increase A’s TCAMs
Allocate TCAM: Multiple Switches

A task can have traffic from multiple switches

Controller

Heavy Hitter detection

Local accuracy is important
If a task is globally unsatisfied, increasing B’s TCAMs is expensive (diminishing returns)
Allocate TCAM: Multiple Switches

A task can have traffic from multiple switches

Use both local and global accuracy
DREAM Modularity

Task Independent
- TCAM Configuration: Divide & Merge
- TCAM Allocation

Task Dependent
- Accuracy Estimation

DREAM
Evaluation: Accuracy and Overhead

Accuracy

Satisfaction of a task: Fraction of task’s lifetime with sufficient accuracy

% of rejected/dropped tasks

Overhead

How fast is the DREAM control loop?
Evaluation: Alternatives

Equal: divide TCAMs equally at each switch, no reject

Fixed: fixed fraction of TCAMs, reject extra tasks
Evaluation Setting

Prototype on 8 Open vSwitches
- 256 tasks (HH, HHH, CD, combination)
- 5 min tasks arriving in 20 mins
- Accuracy bound=80%
- 5 hours CAIDA trace
- Validate simulator using prototype

Large scale simulation (4096 tasks on 32 switches)
- accuracy bounds
- task loads (arrival rate, duration, switch size)
- tasks (task types, task parameters e.g., threshold)
- # switches per tasks
Prototype Results: Average Satisfaction

DREAM: High satisfaction of tasks at the expense of more rejection for small switches

Fixed: High rejection as over-provisions for small tasks
Prototype Results: 95\textsuperscript{th} Percentile Satisfaction

DREAM: High 95\textsuperscript{th} percentile satisfaction

Equal and Fixed only keep small tasks satisfied
Conclusion

Measurement is crucial for SDN management in a resource-constrained environment

Dynamic TCAM allocation across measurement tasks
• Diminishing returns in accuracy
• Spatial and temporal multiplexing

Future work
• More TCAM-based measurement tasks (quintiles for load balancing, entropy detection)
• Hash-based measurements

DREAM is available at
github.com/USC-NSL/DREAM