NETWORKING SEMINAR

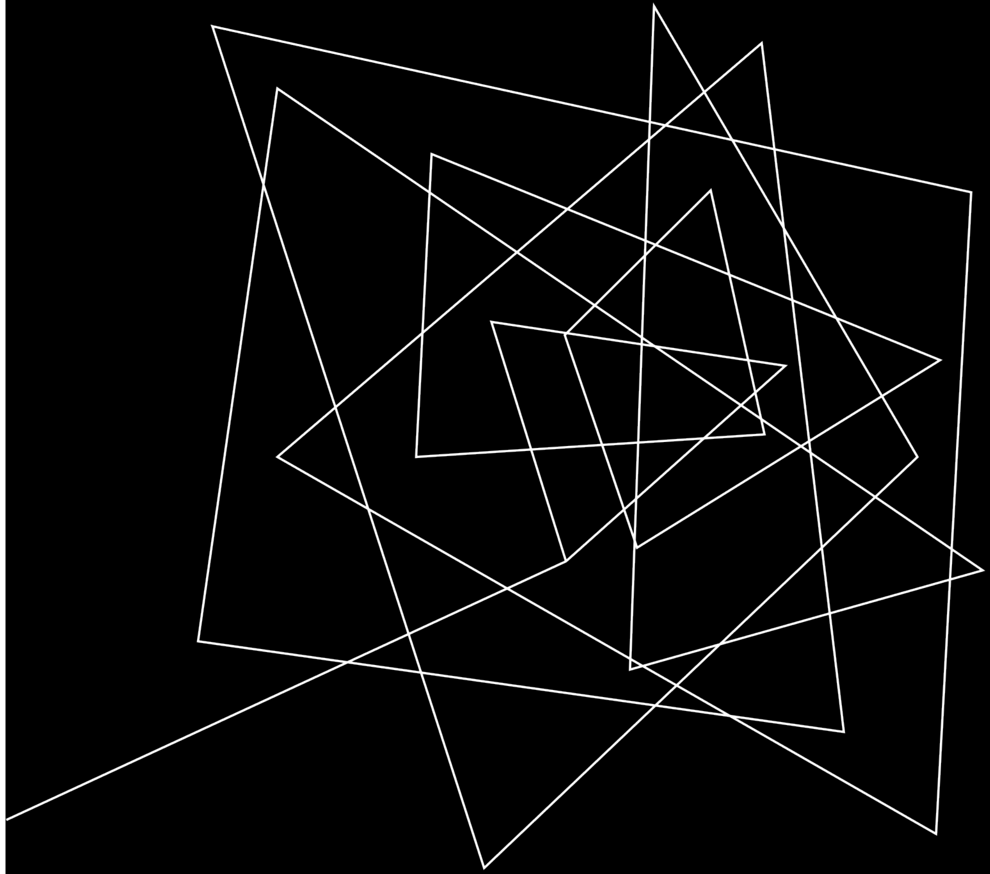
Emad Taghiye

Winter 2024

AGENDA

INTRODUCTION TO HEAVY HITTERS

The heavy hitter problem, a fundamental challenge in network traffic analysis, involves identifying the most frequently occurring elements in the network traffic. It plays a crucial role in network management and security by helping to detect and prioritize high-impact events or anomalies.



NETWORK-WIDE HEAVY HITTER DETECTION WITH COMMODITY SWITCHES

Rob Harrison, Qizhe Cai, Arpit Gupta, and Jennifer Rexford

NETWORK-WIDE HEAVY HITTER DETECTION

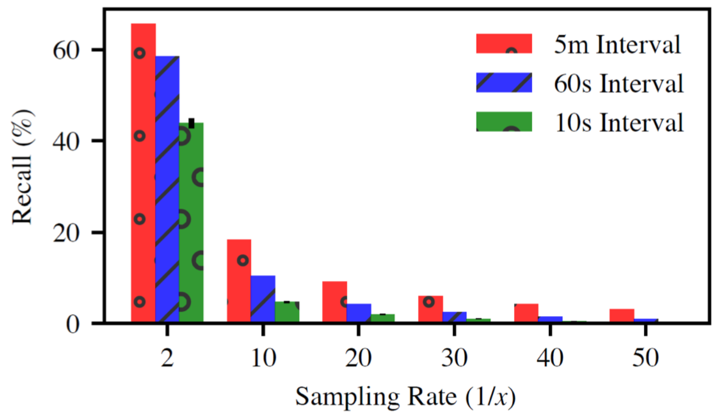
While prior work has focused on heavy-hitter detection at a single switch, network operators often need to track the network-wide heavy hitters

Detecting the heavy hitters separately at each switch and then combining the results is not sufficient.

To bridge the gap between line-rate monitoring and networkwide visibility, a distributed heavy-hitter detection scheme for networks modeled as one-big switch is presented.

LIMITATION OF TRADITIONAL NETWORKS

- Analyze packet samples
- Resulted in substantially reduced accuracy



CHALLENGES

- The solution should be:
  - Accurate
  - Real-time
  - Communication efficient
  - Memory efficient (Not the focus of this paper)

So, the objective is to minimize the communication cost between the observers and the coordinator, while continuously computing the heavy hitters in real time.

OPPORTUNITIES

Programmable switches open up new possibilities for aggregating traffic statistics and identifying large flows directly in the data plane

Spatial locality of network traffic provides opportunities to reduce the overhead of detecting heavy hitters

DISTRIBUTED, ADAPTIVE THRESHOLDS
INTRODUCTION

A solution that can accurately detect network-wide heavy hitters with up to 70% savings in communication overhead compared to an existing approach with a provable upper bound.

Summary of this approach:

- Each switch identifies which traffic to report to the coordinator, using different local thresholds for different monitored keys.
- The coordinator adapts these thresholds to the prevailing traffic to reduce the total number of reports.

DISTRIBUTED, ADAPTIVE THRESHOLDS
ALGORITHM

Variables:

- *N*: Total Number of switches
- $C_{i,k}$: Counter
- $T_{i,k}$: Local Threshold
- T_G: Global Threshold
- *i*: Switch Number
- *k*: Flow Key Number
- *a: Smoothing Factor*

Heavy hitters are detected based on a rolling time window.

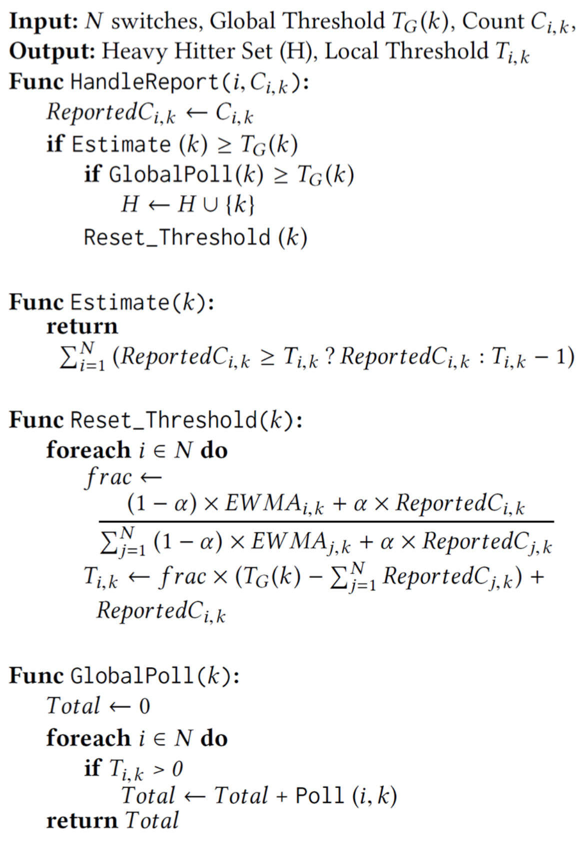
```
**Input:** $N$ switches, Global Threshold $T_G(k)$, Count $C_{i,k}$,
**Output:** Heavy Hitter Set (H), Local Threshold $T_{i,k}$
**Func** HandleReport($i, C_{i,k}$):
    $ReportedC_{i,k} \leftarrow C_{i,k}$
    **if** Estimate $(k) \geq T_G(k)$
        **if** GlobalPoll$(k) \geq T_G(k)$
            $H \leftarrow H \cup \{k\}$
        Reset_Threshold $(k)$

**Func** Estimate($k$):
    **return**
     $\sum_{i=1}^{N} (ReportedC_{i,k} \geq T_{i,k} \, ? \, ReportedC_{i,k} : T_{i,k} - 1)$

**Func** Reset_Threshold($k$):
    **foreach** $i \in N$ **do**
        $frac \leftarrow$
          $\frac{(1-\alpha) \times EWMA_{i,k} + \alpha \times ReportedC_{i,k}}{\sum_{j=1}^{N} (1-\alpha) \times EWMA_{j,k} + \alpha \times ReportedC_{j,k}}$
        $T_{i,k} \leftarrow frac \times (T_G(k) - \sum_{j=1}^{N} ReportedC_{j,k}) +$
         $ReportedC_{i,k}$

**Func** GlobalPoll($k$):
    $Total \leftarrow 0$
    **foreach** $i \in N$ **do**
        **if** $T_{i,k} > 0$
            $Total \leftarrow Total +$ Poll $(i, k)$
    **return** $Total$
```

DISTRIBUTED, ADAPTIVE THRESHOLDS

ALGORITHM

Steps:

1. *When the local count for a key reaches or exceeds its local threshold, the switch sends the coordinator a report with the key and the count*
2. *The coordinator combines the counts for the same key across reports from multiple switches*
3. *The coordinator makes a conservative estimate of the global count assuming a count equal to one less than the local threshold for the counters which did not send reports for the particular key.*
4. *If the estimated total equals or exceeds the global threshold for the key (TG (k)), the coordinator polls all of the switches whose local thresholds are greater than zero to learn their current counts.*
5. *The coordinator adapts the per-key local thresholds based on past reports*

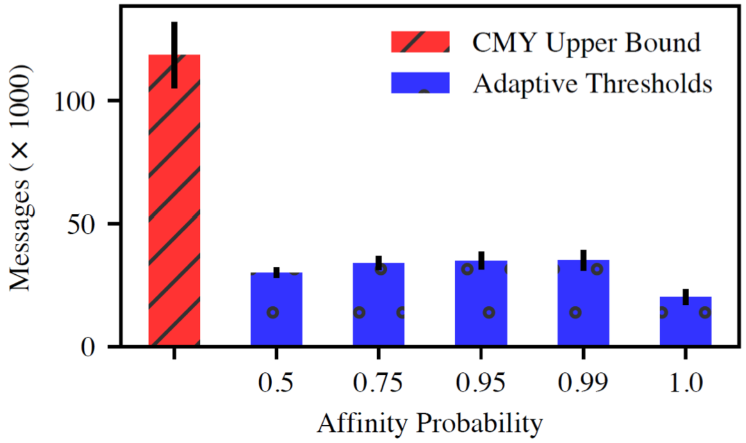


```
Input: N switches, Global Threshold $T_G(k)$, Count $C_{i,k}$,
Output: Heavy Hitter Set (H), Local Threshold $T_{i,k}$
Func HandleReport($i, C_{i,k}$):
    $ReportedC_{i,k} \leftarrow C_{i,k}$
    if Estimate $(k) \geq T_G(k)$
        if GlobalPoll$(k) \geq T_G(k)$
            $H \leftarrow H \cup \{k\}$
        Reset_Threshold $(k)$

Func Estimate($k$):
    return
     $\sum_{i=1}^{N} (ReportedC_{i,k} \geq T_{i,k}\ ?\ ReportedC_{i,k} : T_{i,k} - 1)$

Func Reset_Threshold($k$):
    foreach $i \in N$ do
        $frac \leftarrow$
          $\frac{(1-\alpha) \times EWMA_{i,k} + \alpha \times ReportedC_{i,k}}{\sum_{j=1}^{N} (1-\alpha) \times EWMA_{j,k} + \alpha \times ReportedC_{j,k}}$
        $T_{i,k} \leftarrow frac \times (T_G(k) - \sum_{j=1}^{N} ReportedC_{j,k}) +$
         $ReportedC_{i,k}$

Func GlobalPoll($k$):
    $Total \leftarrow 0$
    foreach $i \in N$ do
        if $T_{i,k} > 0$
            $Total \leftarrow Total +$ Poll $(i, k)$
    return $Total$
```

EVALUATION

- Compared with:
  - CMY (Cormode–Muthukrishnan–Yi) upper bound
- Experimental setup:
  - Dataset: CAIDA’s anonymized Internet traces from 2016
  - Tofino Switches
  - W = 6s
  - n = 4, l = 2, p = 0.95, a = 0.8, and TG = 600 unless otherwise specified
- Metrics:
  - Affinity probability
  - Threshold
  - Number of sites
  - Smoothing factor

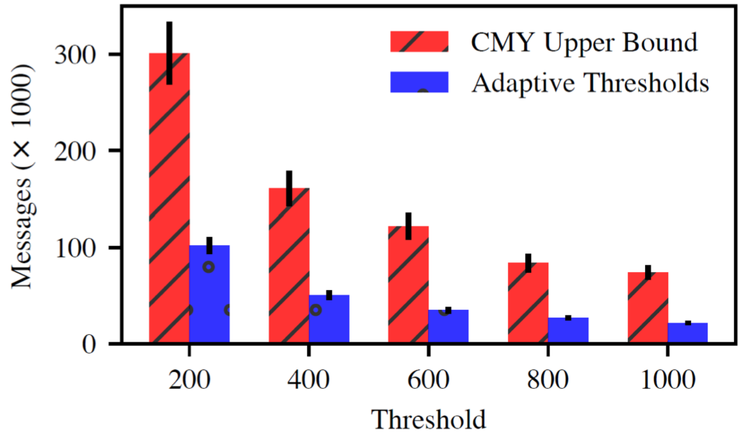
EVALUATION
SENSITIVITY TO SITE AFFINITY



The number of messages exchanged between the sites and the controller are substantially reduced regardless of the sources' affinity for a particular ingress switch.

A substantial drop between p = 0.99 to p = 1.0 is visible because when p = 1.0, a given source always enters the network at the same location.

EVALUATION
SENSITIVITY TO THRESHOLD

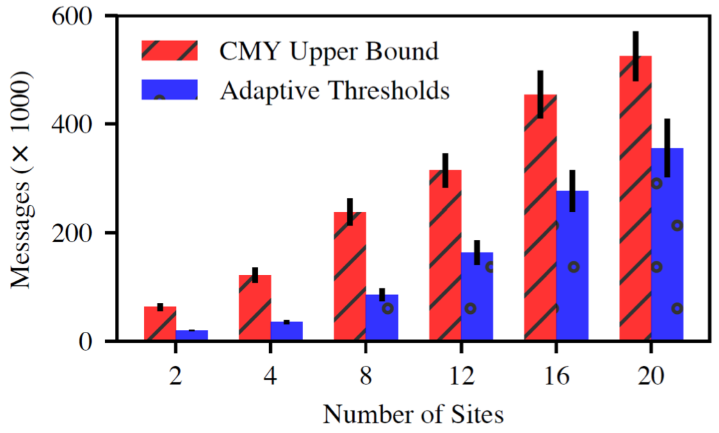


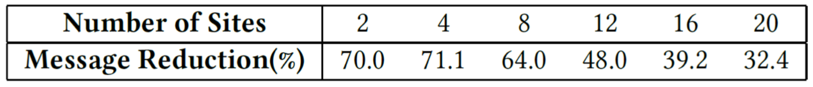
Thresholds	200	400	600	800	1000
Message Reduction (%)	66.1	68.7	71.3	67.7	70.8

As the global threshold increases, the fraction of overall traffic that constitute heavy hitters decreases, reducing the communication overhead.

Communication reduction over the CMY upper bound is not much affected as the threshold increases.

EVALUATION
SENSITIVITY TO NUMBER OF SITES



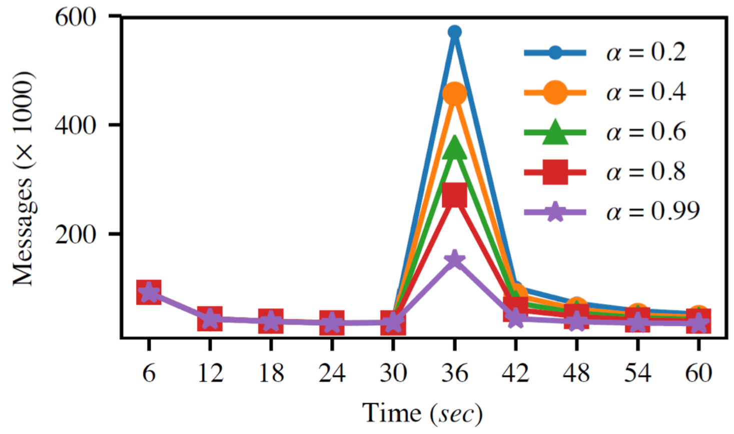


Number of Sites	2	4	8	12	16	20
Message Reduction(%)	70.0	71.1	64.0	48.0	39.2	32.4

As the number of sites increases, the communication overhead increases due to the additional nodes communicating with the coordinator.

Communication reduction over the CMY upper bound lessens as the number of sites increases.

EVALUATION
SENSITIVITY TO SITE AFFINITY



As alpha increases, the algorithm is able to more quickly adapt to changes in the traffic distribution which results in lower communication overhead.

The single, abrupt change in this experiment fails to account for the variety of traffic dynamics one might experience in a production network and selecting the best value of α depends on those specific conditions.

CONCLUSION

- An efficient and accurate network-wide heavy hitter algorithm for the reduction of the communication overhead between the switches and the coordinator is proposed.
- Evaluations are done on real-world traffic traces
- evaluation demonstrates that by dynamically adapting per-key thresholds, we can reduce the communication overhead required to detect network-wide heavy hitters without compromising accuracy

QUESTIONS?

THANK YOU