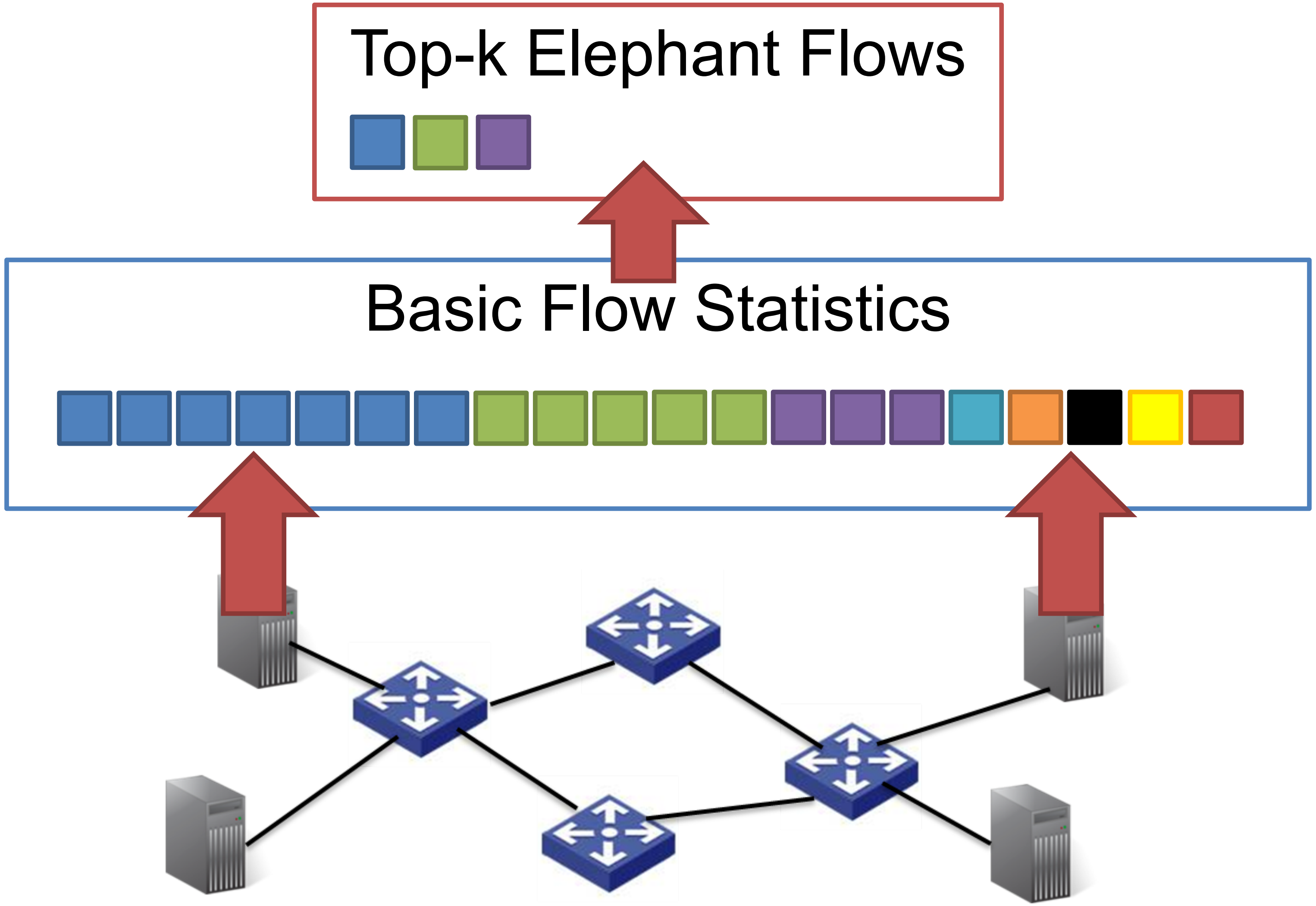
HeavyKeeper: An Accurate Algorithm for Finding Top-k Elephant Flows

Junzhi Gong[1], Tong Yang[1], Haowei Zhang[1], Hao Li[1], Steve Uhlig[2], Shigang Chen[3], Lorna Uden[4], Xiaoming Li[1]

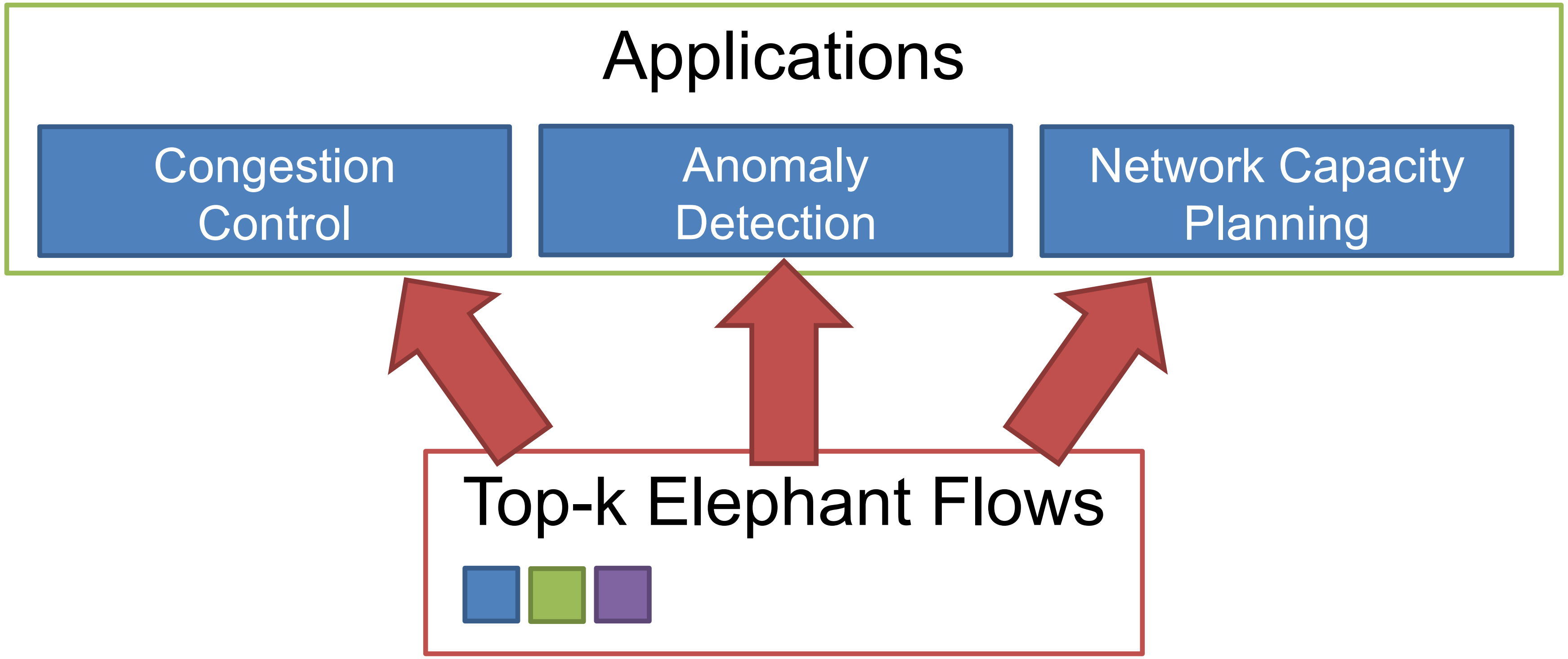
Peking University[1], Queen Mary University of London[2], University of Florida[3], Staffordshire University[4]

Finding Top-k Elephant Flows



Finding Top-k Elephant Flows

Finding top-k elephant flows serves as a fundamental network management function.

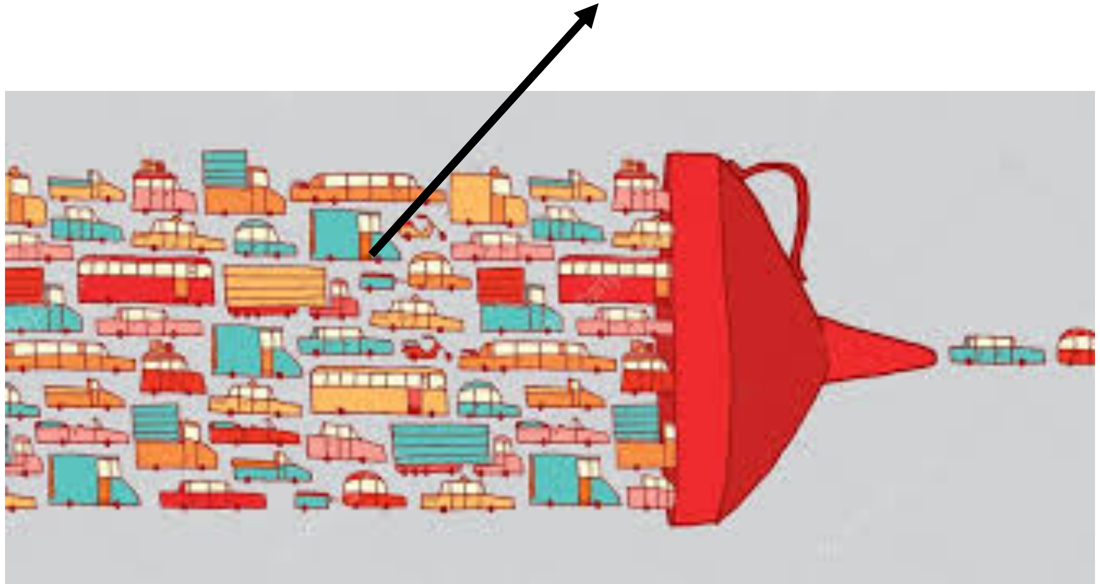


Flow Distribution of Real Traffic

- Highly Skewed
  - The majority are mouse flows
  - The minority are elephant flows
  - However, elephant flows are much more important

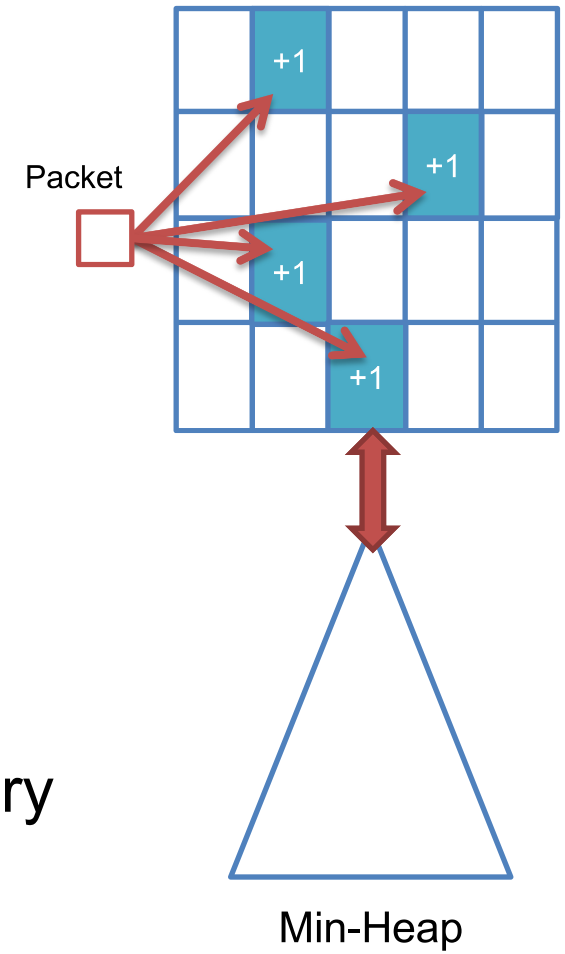
Challenges

- High Line Rate
  - Impossible to track information of all flows
  - Solution: approximate methods
- High Latency of Off-chip Memory
  - Force algorithm to use on-chip memory, like SRAM
  - The size of on-chip memory is small, e.g., several megabytes



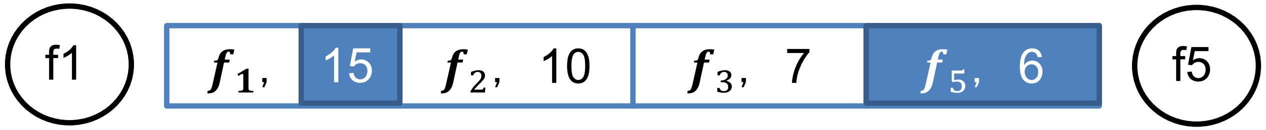
Existing Solution I: Count-All

- ➢ A sketch to approximately record all flow sizes
- ➢ And a min-heap to maintain top-k elephant flows
- ➢ Sketches are smaller than hash tables
  - But they still need to store all flows, which is not memory efficient



Existing Solution II: Admit-All-Count-Some

- Frequent, Lossy Counting, Space-Saving, CSS
- Process all flows, but only record a small part of them
- Example: Space-Saving



Limitation of two existing solutions

- Count-All
  - Spend too much memory and time on mouse flows
  - With total memory size small, the accuracy cannot be high
- Admit-All-Count-Some
  - Could drastically over-estimate flow sizes (Space-Saving, CSS)
  - Could make the size of elephant flows inaccurate (Frequent, Lossy Counting)
  - With limited memory size, many mouse flows can be mistreated as elephant flows.

Our Contributions

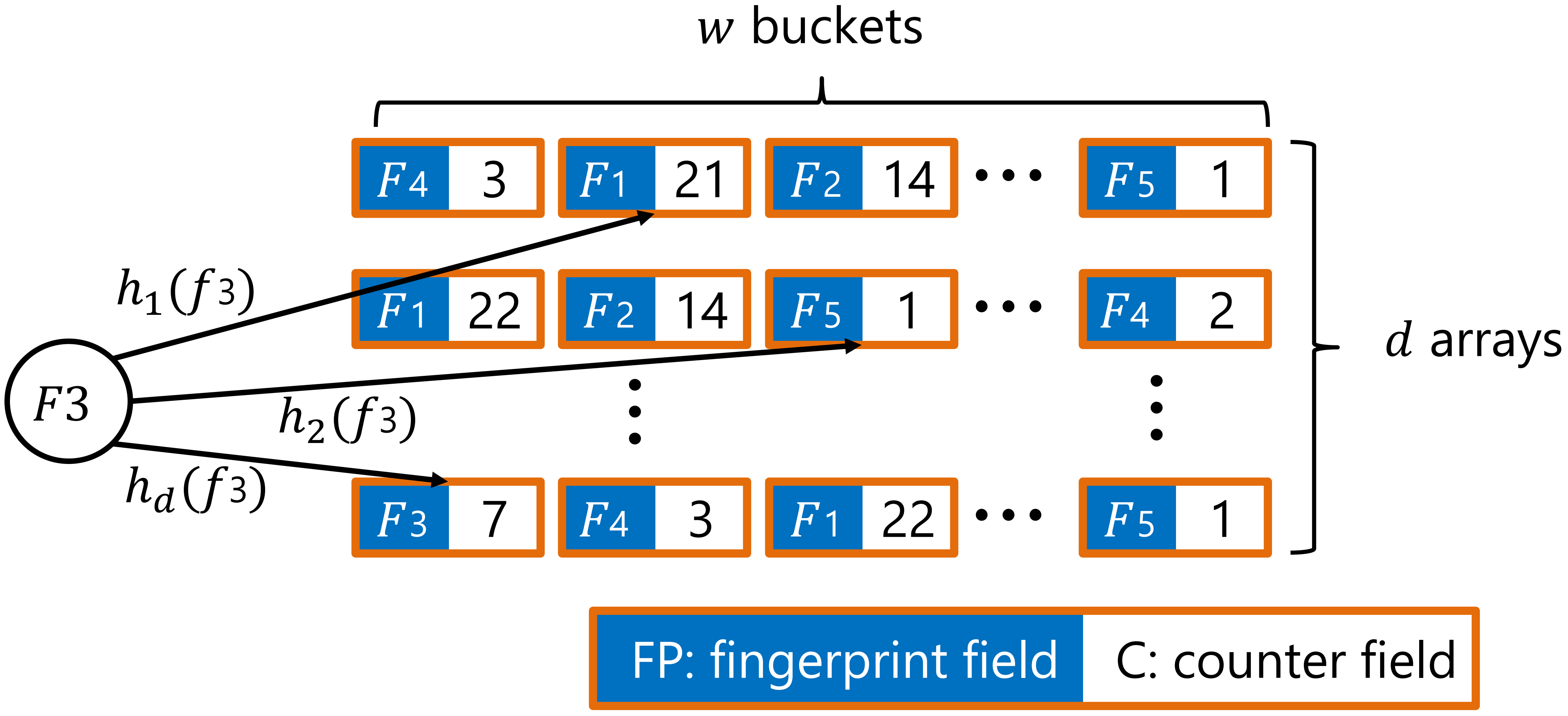
- A new data structure, named HeavyKeeper, which achieves high accuracy and high processing speed in finding top-k elephant flows.
- Experiments on real network traffic and synthetic datasets, showing the high performance of HeavyKeeper
- Deploy algorithms in Open vSwitch (OVS) platform

Our Solution

- HeavyKeeper
  - Strategy: count-with-exponential-decay
  - Keeps only elephant flows
  - Drastically reduce space wasted on mouse flows

- Exponential-weakening-decay
  - Mouse flows are easily be decayed and removed
  - Elephant flows can hardly be removed
  - Will not admit new flows unless one flow is ready to be removed

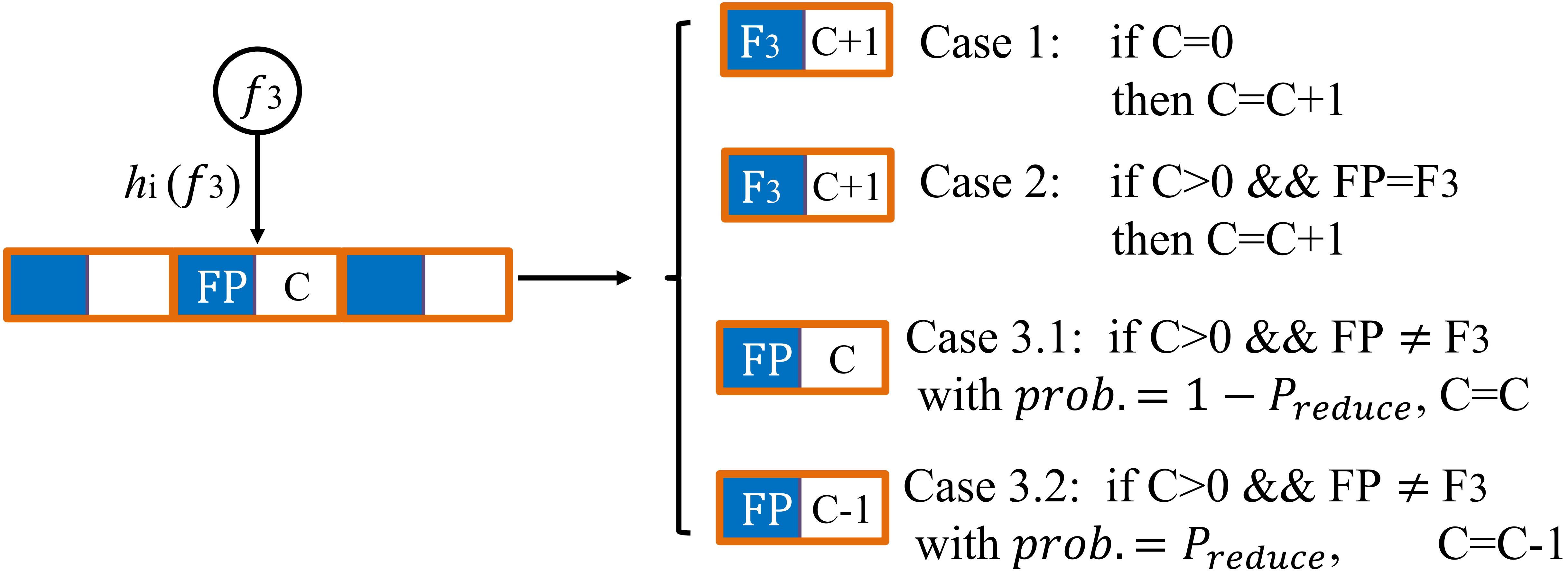
Data Structure



Insertion

- Map: each incoming flow is mapped to one bucket for each array by using the hash functions
- Insert: for each mapped bucket, different strategies are applied according to different cases.

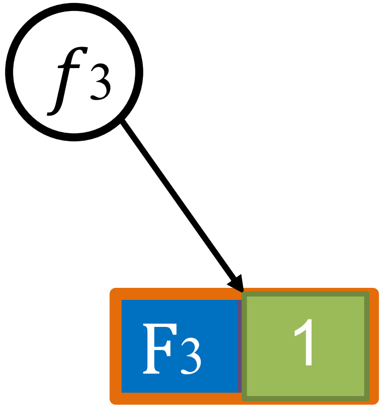
Insertion


f_3
$h_i(f_3)$
FP
C
F_3
C+1
Case 1: if C=0
then C=C+1
F_3
C+1
Case 2: if C>0 && FP=F3
then C=C+1
FP
C
Case 3.1: if C>0 && FP $\neq$ F3
with $prob. = 1 - P_{reduce}$, C=C
FP
C-1
Case 3.2: if C>0 && FP $\neq$ F3
with $prob. = P_{reduce}$, C=C-1

Case 1

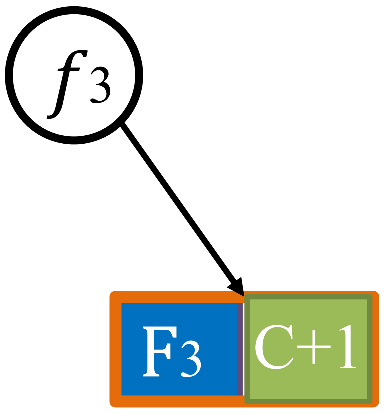
➢ The bucket is empty, i.e., C=0

- Simply set the fingerprint to FP(f_3), and set C=C+1=1



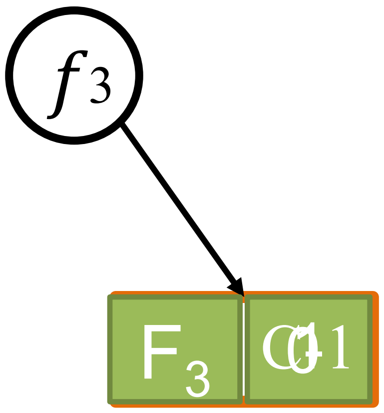
Case 2

- The bucket is not empty, i.e., $C>0$, and $FP=F_3$
  - Simply set $C=C+1$



Case 3

- The bucket is not empty, i.e., C>0, but $\mathrm{FP} \neq \mathrm{F3}$
  - With a probability P_{reduce}, decrease the counter C by 1.
  - If the counter is reduced to 0, we replace the original FP to the fingerprint of the new flow F_3



Query

- For incoming flow *f*
- Map: get *d* mapped buckets
- Filter: get those buckets whose fingerprint is FP*(f)*
- Answer: report the largest value among all filtered buckets (0 if no bucket left after filtering)

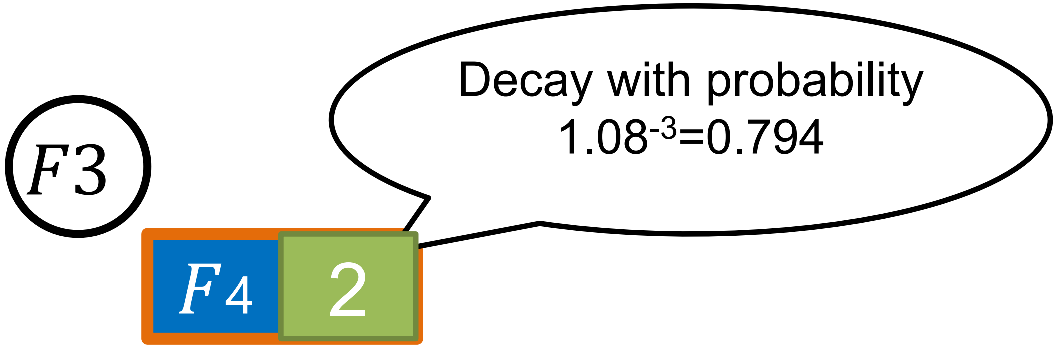
Decay Probability

- $P = b^{-C}$
- b is a predefined constant, e.g., $b = 1.08$
- C is the value in the current counter field (the value to be decayed)

- The larger C is, the harder its flow size is decayed

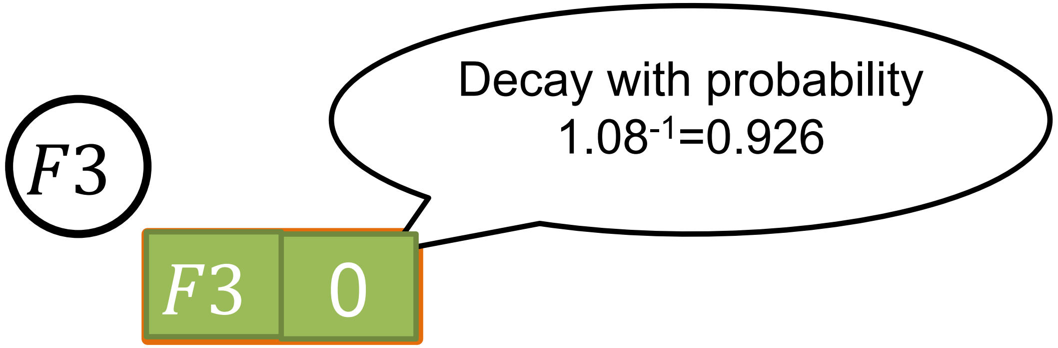
Decay Probability

- When the original bucket stores a mouse flow, it will be easily decayed



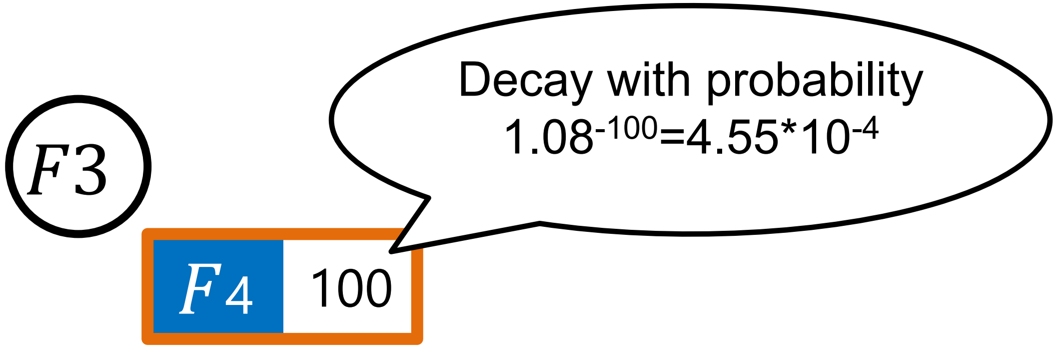
Decay Probability

- When the size of mouse flow is decayed to 0, the original flow will be replaced
  - Therefore, mouse flows can hardly stay in HeavyKeeper



Decay Probability

- When the original bucket stores an elephant flow, it can be hardly decayed
  - Therefore, elephant flows can be stably stored in HeavyKeeper
  - The estimated size of elephant flows will also be accurate

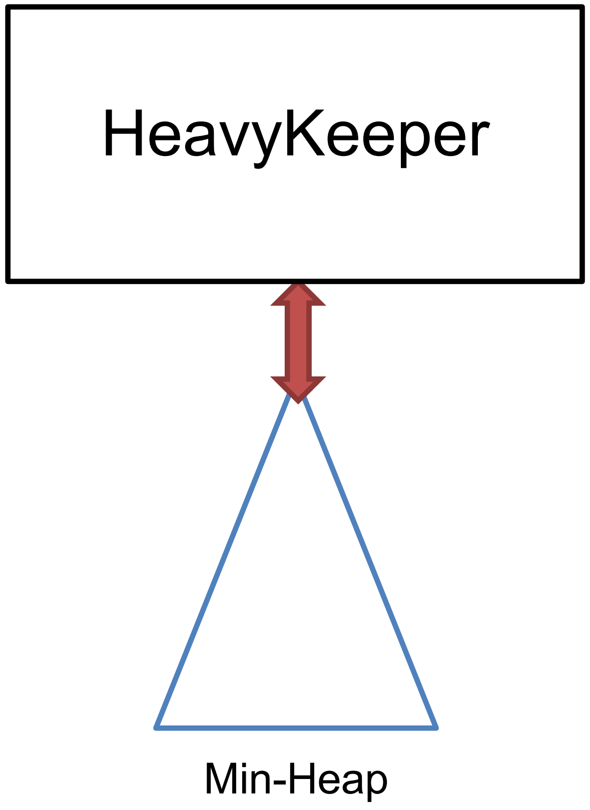


Analysis

- With exponential-weakening decay, HeavyKeeper will tend to store elephant flows and evict mouse flows.
- Most mouse flows are simply *passers-by* of HeavyKeeper.
- Elephant flows are easily stored, and their estimated flow sizes are also accurate.

Basic Algorithm

- To find top-k elephant flows, the basic version of our algorithm will simply use a min-heap to maintain the top-k elephant flows, like sketches.

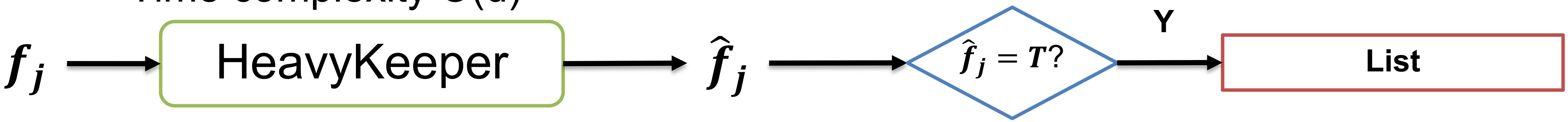


Analysis

- Complexity
  - Space: O(d*w).
    - Experiments will show HeavyKeeper achieves high accuracy with very small memory usage.
  - Time: O(k).
    - Updating the min-heap is time consuming. Even with the help of hash tables, the insertion of the min-heap will also consume O(logk) time complexity.

Optimizations

- Using the min-heap, improving accuracy
  - Too many details, skip
- Replacing the min-heap with a single list, improving speed
  - Ignoring fingerprint collisions, the flow size of each flow will grow 1 by 1.
  - Define a threshold T (e.g., T=1000)
  - Recording the incoming flow in the list if the estimated size of the flow is equal to T
  - Time complexity O(d)



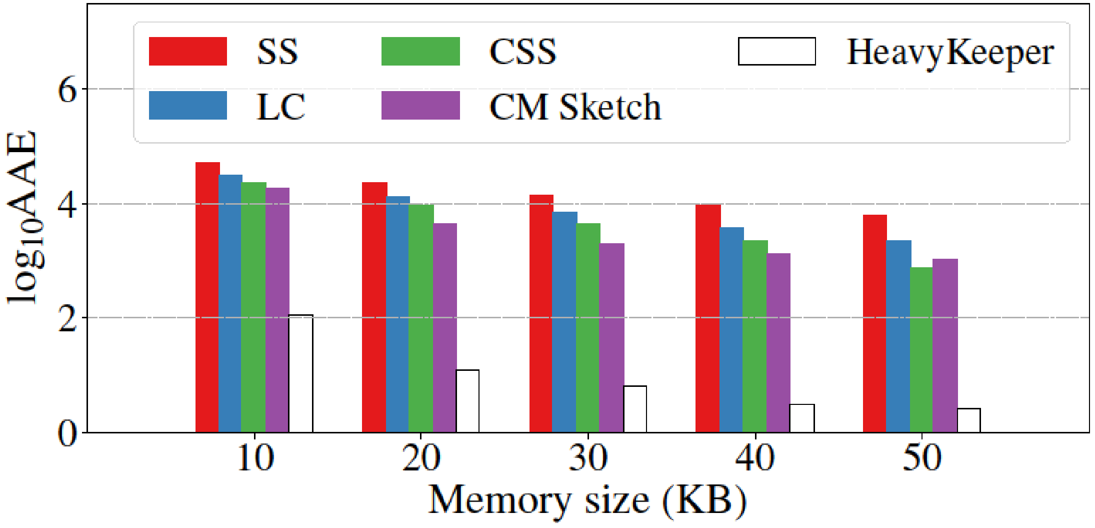
Experimental Setup

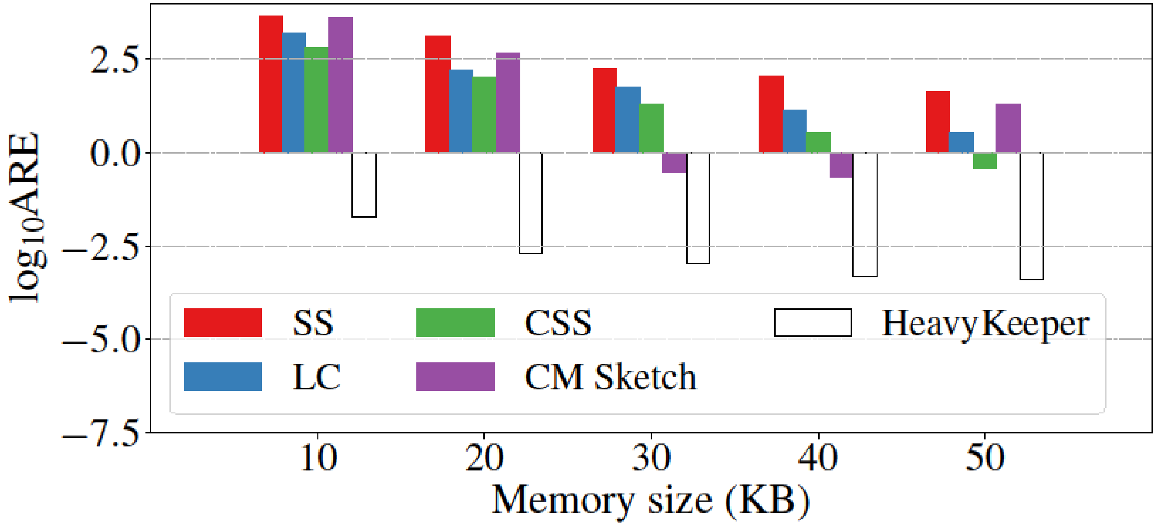
- Datasets
  - Campus network traffic
  - CAIDA
  - Synthetic skewed datasets, Zipf
- Implementation
  - $d = 2$, enough for HeavyKeeper
  - Fixing k and changing memory size, estimate accuracy
  - Fixing memory size and changing k, estimate accuracy
  - Speed evaluation

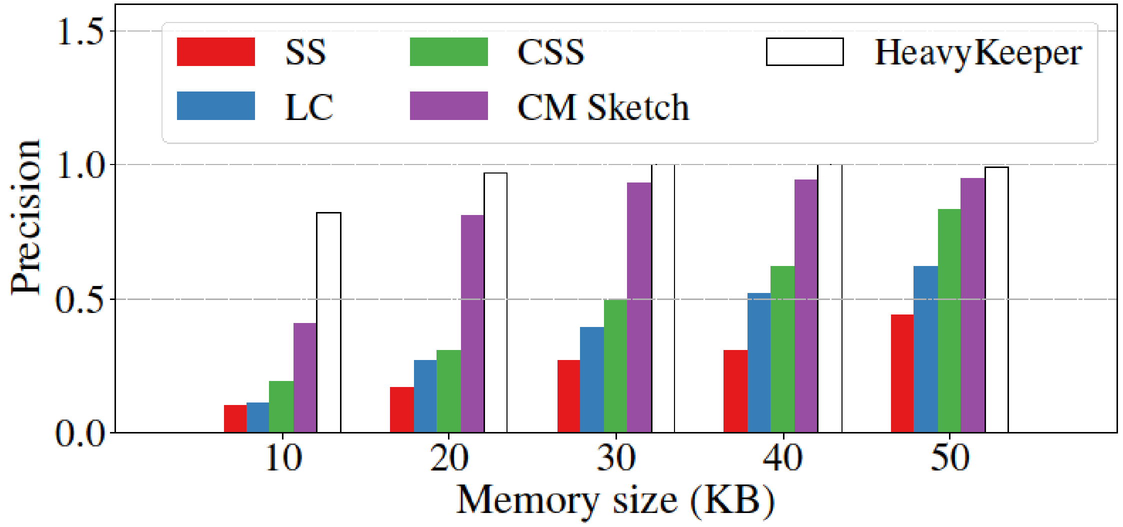
Experimental Setup

- Metrics
  - Average absolute error (AAE). Absolute error is defined as $|n_j - \hat{n}_j|$.
  - Average relative error (ARE). Relative error is defined as $\frac{|n_j - \hat{n}_j|}{n_j}$.
  - Precision. Among the k reported flows, how many flows are real top-k elephant flows. 0%~100%.

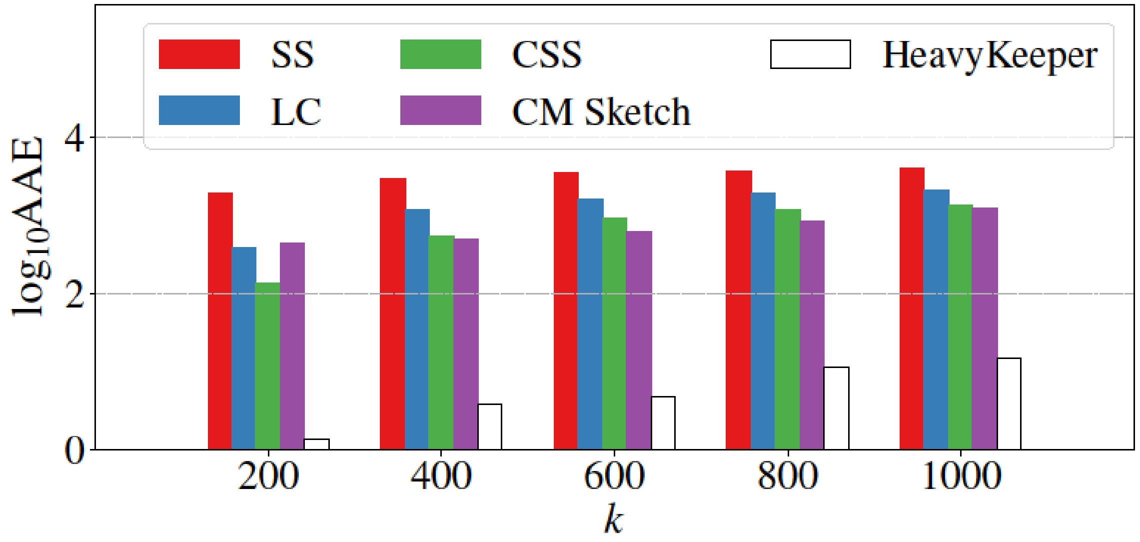
Changing Memory Size



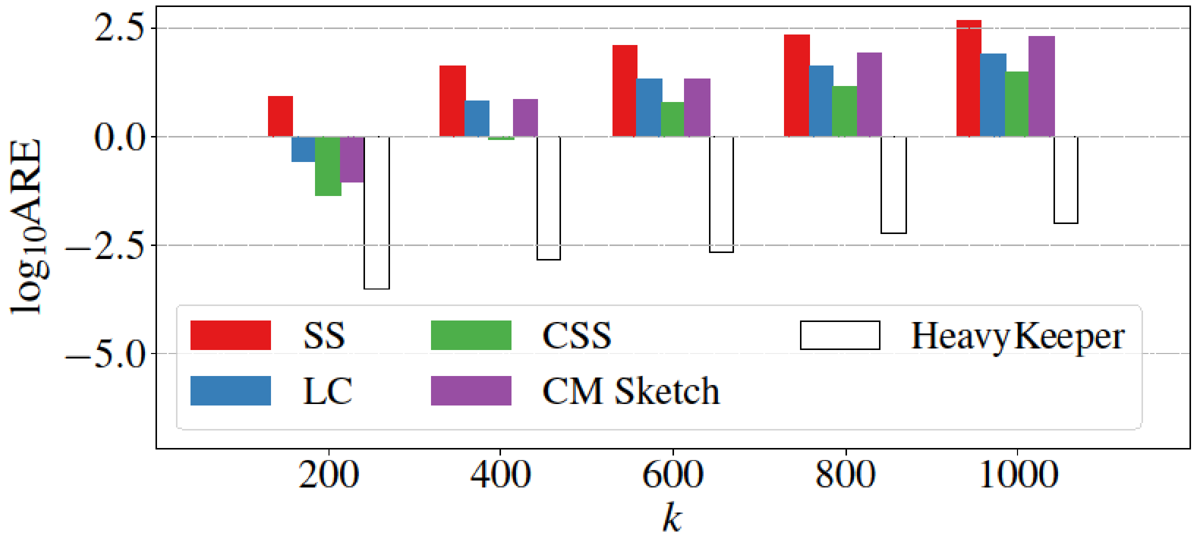




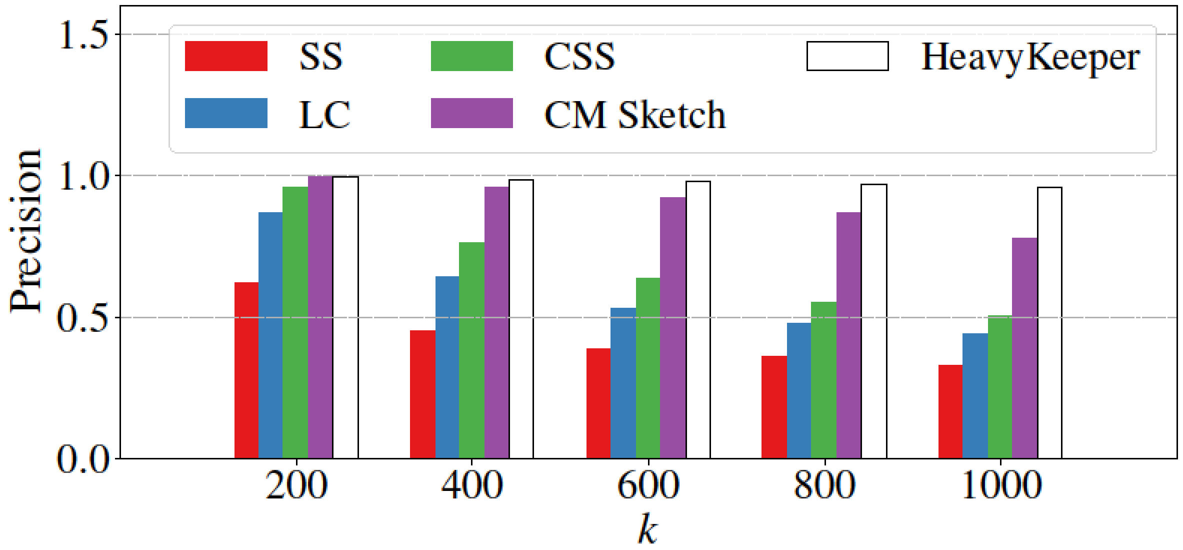
Changing K



SS
CSS
HeavyKeeper
LC
CM Sketch
$\log_{10}$AAE
0
2
4
200
400
600
800
1000
k

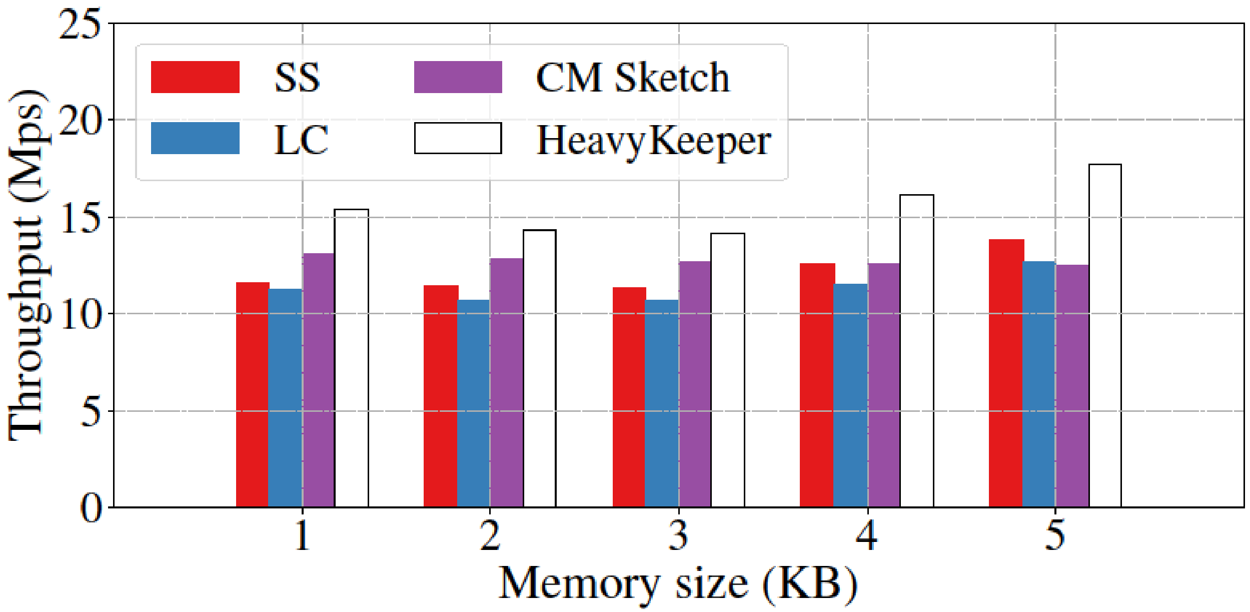


2.5
0.0
−2.5
−5.0
$\log_{10}$ARE
SS
CSS
HeavyKeeper
LC
CM Sketch
200
400
600
800
1000
k



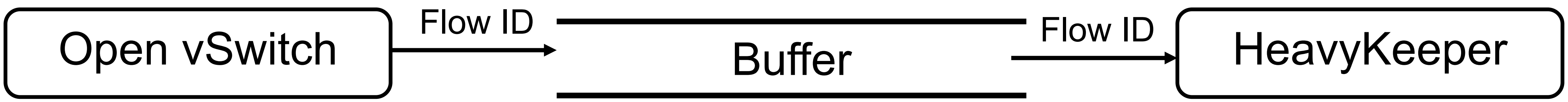
1.5
1.0
0.5
0.0
Precision
SS
CSS
HeavyKeeper
LC
CM Sketch
200
400
600
800
1000
k

Speed Evaluation

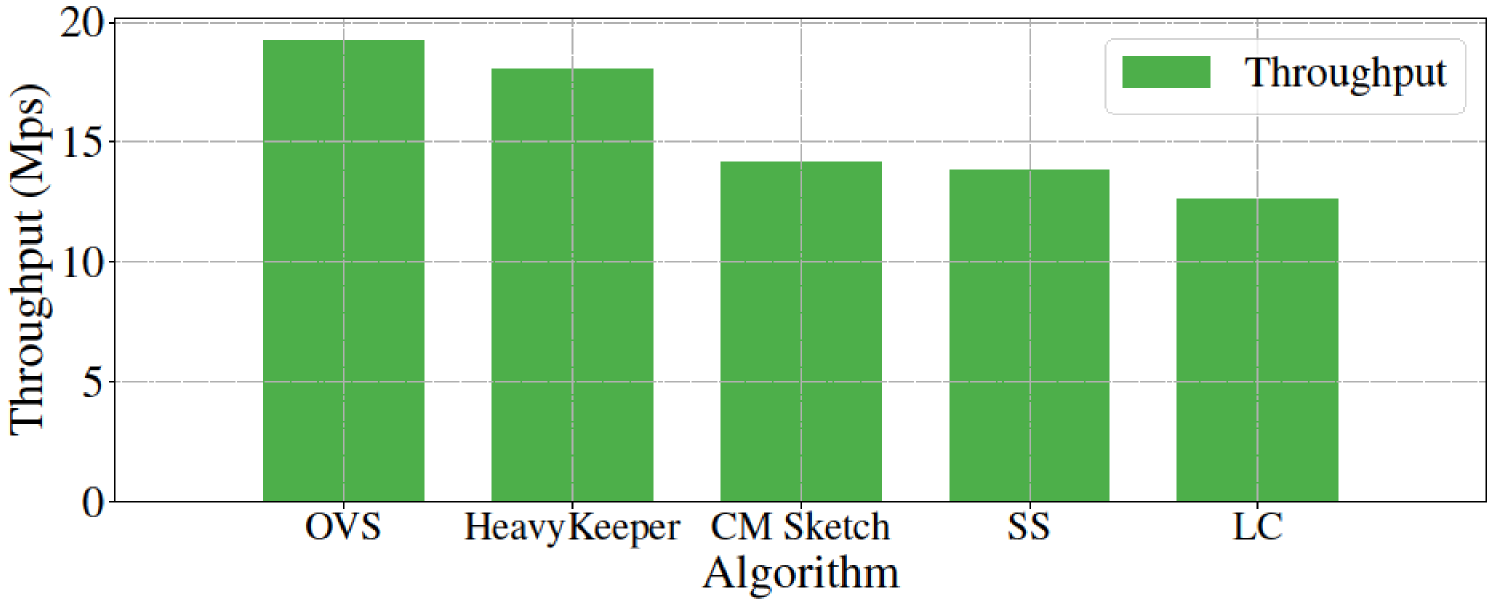


Open vSwitch Deployment

- Create a shared buffer between processes to store flow IDs.
- Modify OVS datapath to report each incoming flow ID to the shared buffer.
- A user-space program of HeavyKeeper to process those flow IDs from the buffer.



Evaluation



Throughput (Mps)
20
15
10
5
0
OVS
HeavyKeeper
CM Sketch
SS
LC
Algorithm
Throughput

Thank you!