Hierarchical Heavy Hitter

TANG DING

Papers

- Mitzenmacher, M., Steinke, T., & Thaler, J. Hierarchical heavy hitters with the space saving algorithm. *(ALENEX 2012)*

- Ben Basat, R., Einziger, G., Friedman, R., Luizelli, M. C., & Waisbard, E. Constant time updates in hierarchical heavy hitters. *(SIGCOMM 2017)*

Application

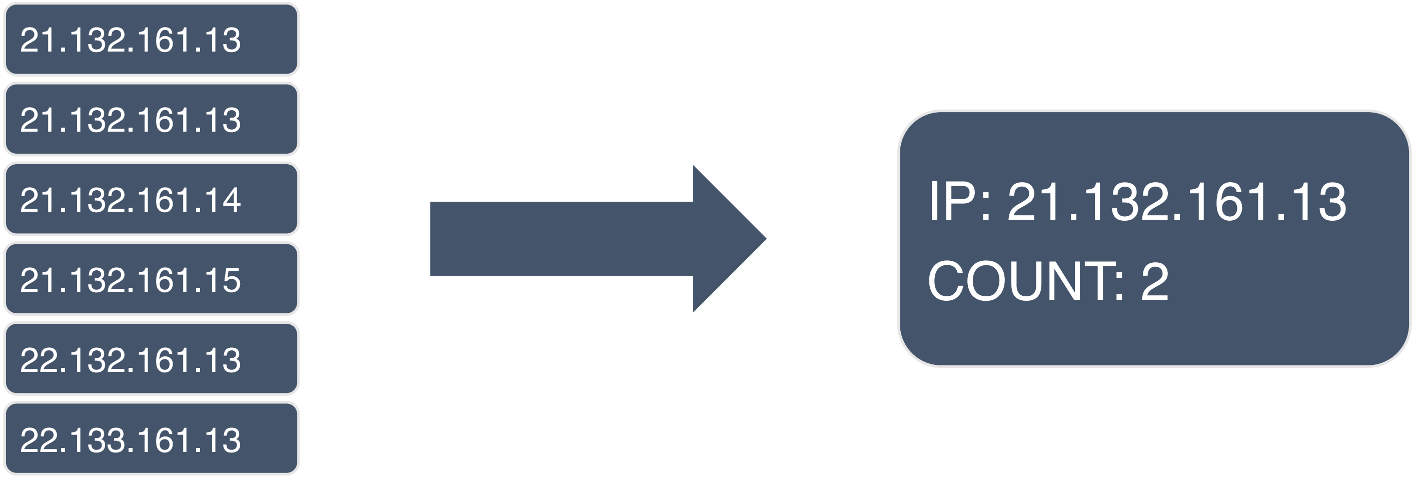
- Network traffic monitoring
- Anomaly detection
- DDoS detection.

Heavy Hitter

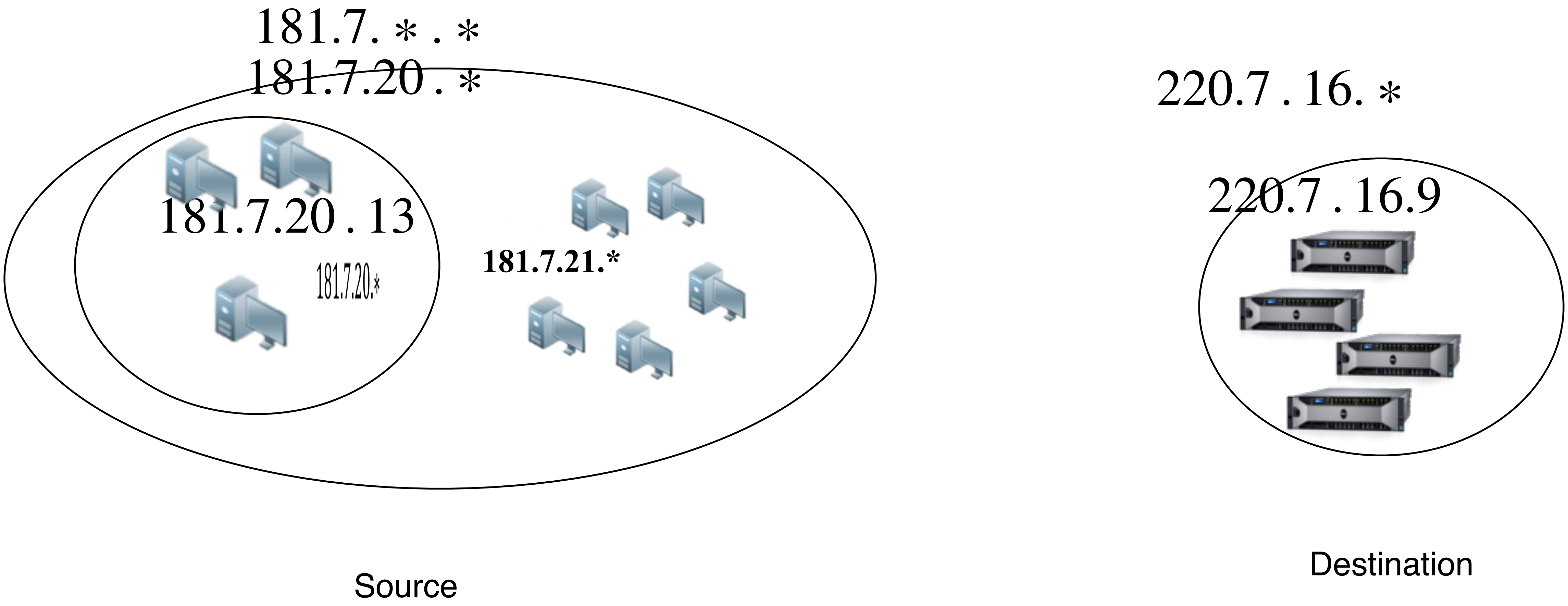
- Basically, equals to frequent items
- Identifies frequent:
  - Source.
  - Source-Destination pairs.

Example

We want to find Source IP whose frequency >= 2



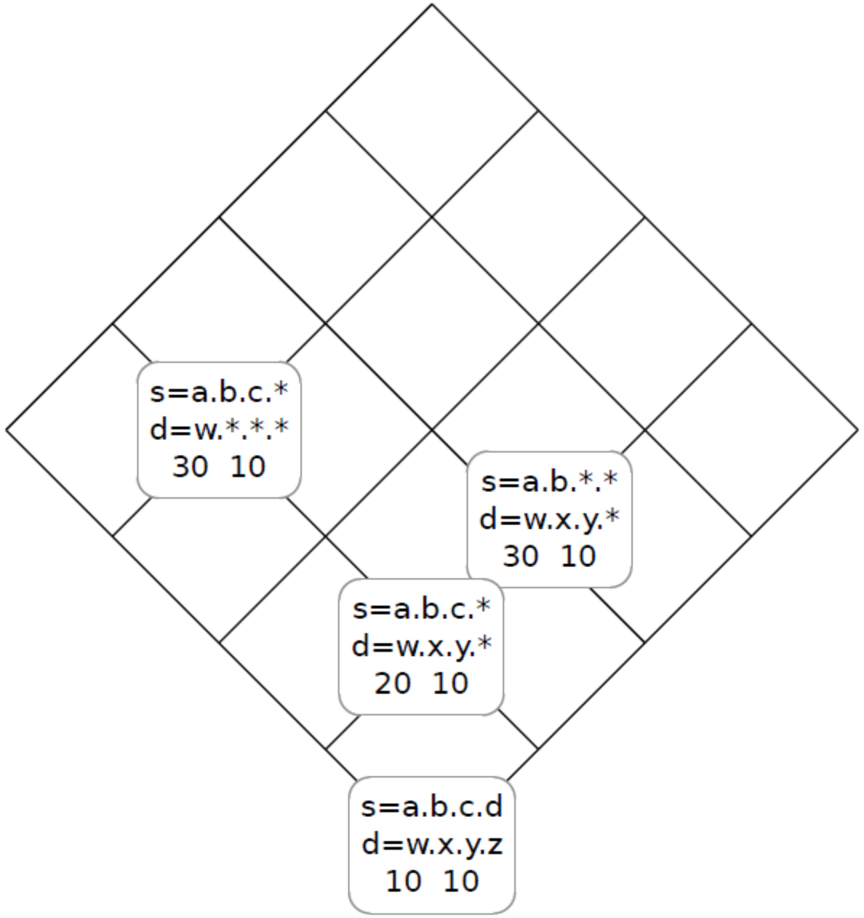
Hierarchical Heavy Hitter



1D Hierarchy example

- Level4: *.*.*.*
- Level3: 21.*.*.*
- Level2: 21.132.*.*
- Level1: 21.132.145.*, 21.132.146.*
- Level0: 21.132.145.13, 21.132.146.14

2D Hierarchy example



Example

We want to find HHH whose frequency >= 2

21.132.161.13
21.132.161.13
21.132.161.14
21.132.161.15
22.132.161.13
22.133.161.13

IP: 21.132.161.13
COUNT: 2

IP: 21.132.161.*
COUNT: 2

IP: 22.*.*.*
COUNT: 2

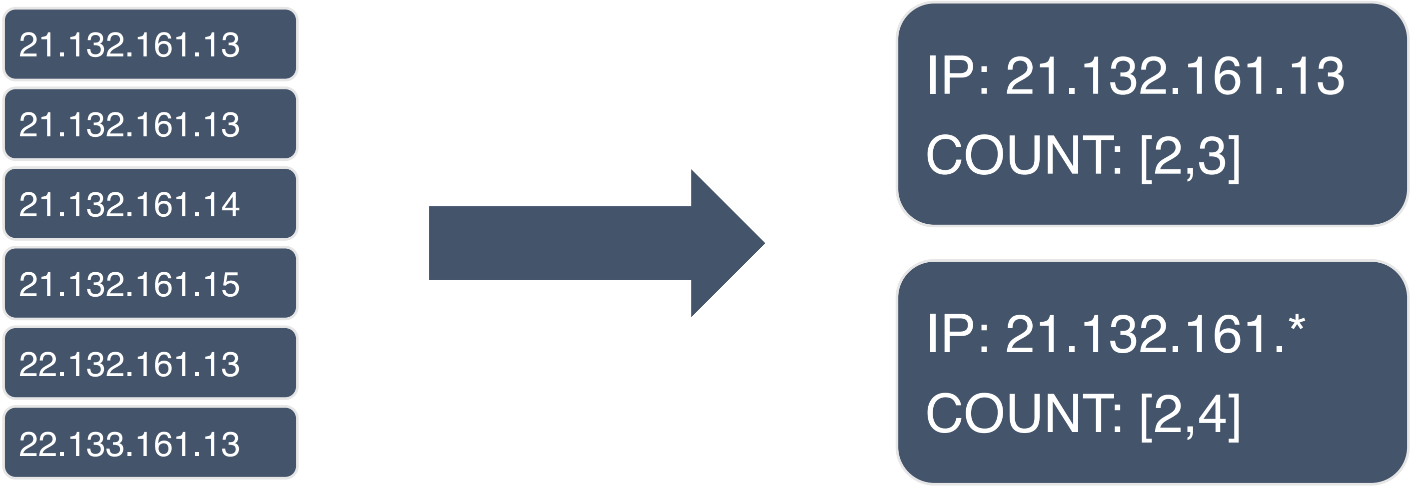
Why count of 21.132.161.* is 2?

Problems

- Often, the large volume of network traffic makes it infeasible to store the relevant data in memory.
- Space-saving streaming algorithm.

Example

We want to find HHH whose frequency >= 2



Output confidence interval instead of actual frequency

Target

- Accuracy: Limited Errors
- Coverage: No False Negatives

Target

- Given parameter , we want to output a set of items , and lower and upper bounds of frequency and , such that they satisfy two properties, as follows:
  - 1. Accuracy. , and for all .
  - 2. Coverage. For set , . Define the conditioned count of with respect to to be . We require for all prefixes

Algorithm

- Two main procedures:
  - Update
  - Output

Update

Time Step	Update	Counter 1	Counter 2	Counter 3
0		***unused***	unused	unused
1	(a,+2)	(a,2)	***unused***	unused
2	(b,+6)	(a,2)	(b,6)	***unused***
3	(c,+4)	***(a,2)***	(b,6)	(c,4)
4	(a,+3)	(a,5)	(b,6)	***(c,4)***
5	(d,+4)	***(a,5)***	(b,6)	(d,8)
6	(e,+4)	(e,9)	***(b,6)***	(d,8)

For any item C being tracked, the actual frequency f(C) is in (Nc – Nmin, Nc),

where Nc is the count of that item, and Nmin is the minimum count in all counters

Update

Time T	Counter 1	Counter 2	Counter 3
Level 1	(a.*.*.*, 20)	***(h.*.*.*, 12)***	(q.*.*.*, 16)
Level 2	(a.b.*.*, 18)	***(h.i.*.*, 12)***	(q.r.*.*, 16)
Level 3	(a.b.c.*, 18)	***(q.r.s.*, 9)***	(h.i.j.*, 14)
Level 4	(a.b.c.d, 10)	(h.i.m.n, 15)	***(a.b.c.e, 8)***

(w.x.y.z, +3)

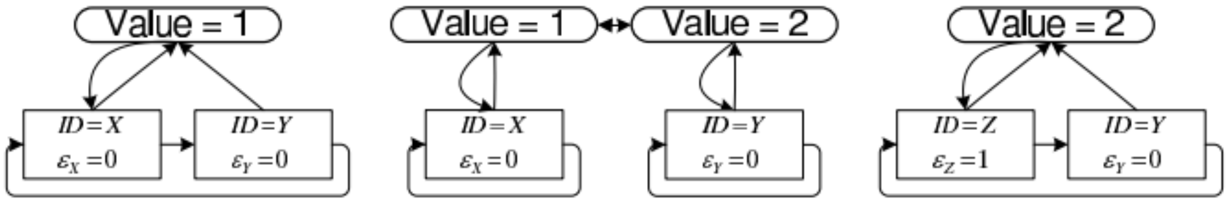
Time T+1	Counter 1	Counter 2	Counter 3
Level 1	(a.*.*.*, 20)	***(w.*.*.*, 15)***	(q.*.*.*, 16)
Level 2	(a.b.*.*, 18)	***(w.x.*.*, 15)***	(q.r.*.*, 16)
Level 3	(a.b.c.*, 18)	***(w.x.y.*, 12)***	(h.i.j.*, 14)
Level 4	***(a.b.c.d, 10)***	(h.i.m.n, 15)	(w.x.y.z, 11)

Update

- For each level, set counters
- Three procedures
    - Find counter for certain IP (pairs)
    - Find counter with minimum count
    - Update counter found (item / count)
- Update Complexity
    - Arbitrary Update: Using heaps –
    - Unitary Update:
    - is the number of levels

Unitary Update

- Data Structure
  - HashTable
  - Item{HashEntry, GroupEntry , NextItemInGroup, PreviousItemInGroup}
  - Group {Items, Count, PreviousGroup, NextGroup}
    - Sorted link list with entries of same count combined



(a) *Stream-Summary*, $S = X, Y$

(b) *Stream-Summary*, $S = X, Y, Y$

(c) *Stream-Summary*, $S = X, Y, Y, Z$

Output (for 1D)

OutputHHH1D(threshold ϕ)

/* par(e) is parent of e*/

Let $s_e = 0$ for all e

/*s_e conservatively estimates the difference between unconditioned and conditioned counts of e*/

for each e in postorder

  $(f_{\min}(e), f_{\max}(e)) = \text{GetEstimateSS}(SS(n), e)$

  if $f_{\max}(e) - s_e \geq \phi N$

    print$(e,\ f_{\min}(e),\ f_{\max}(e))$

    $s_{\text{par}(e)} += f_{\min}(e)$

  else $s_{\text{par}(e)} += s_e$

Proof

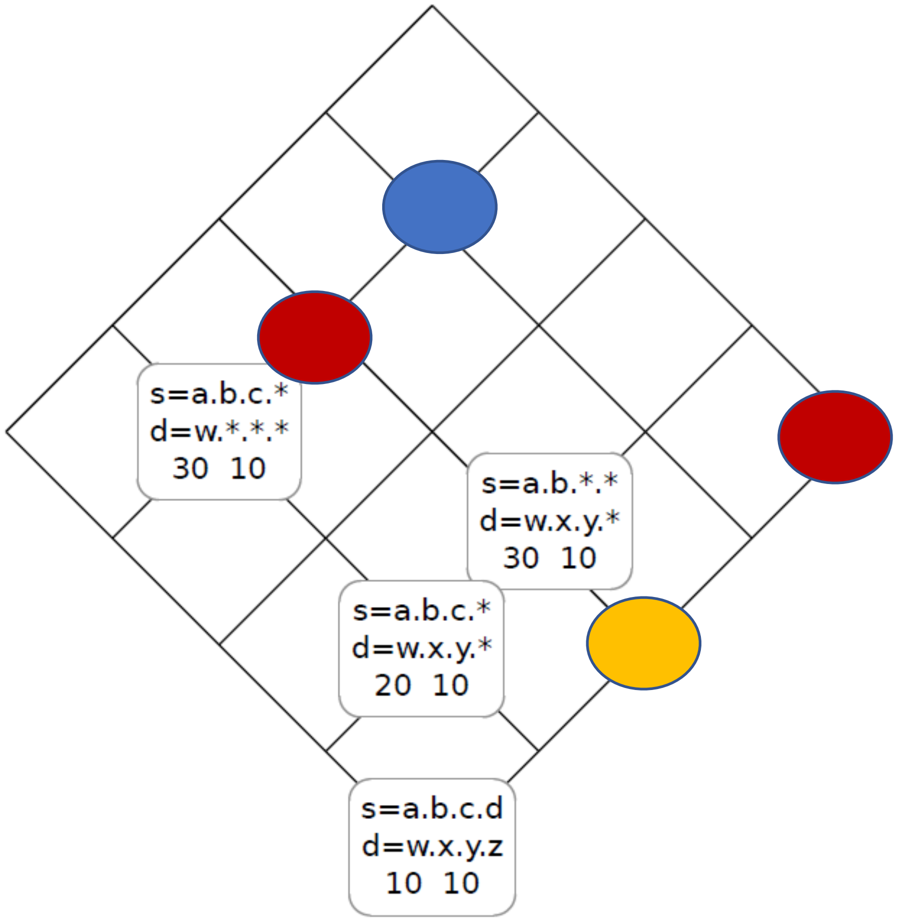
- By using space (counters in each level), we can achieve our target in both accuracy and coverage
  - Accuracy:
    - if ,
    - =
  - Coverage

Output (for 2D)

OutputHHH2D(threshold ϕ)

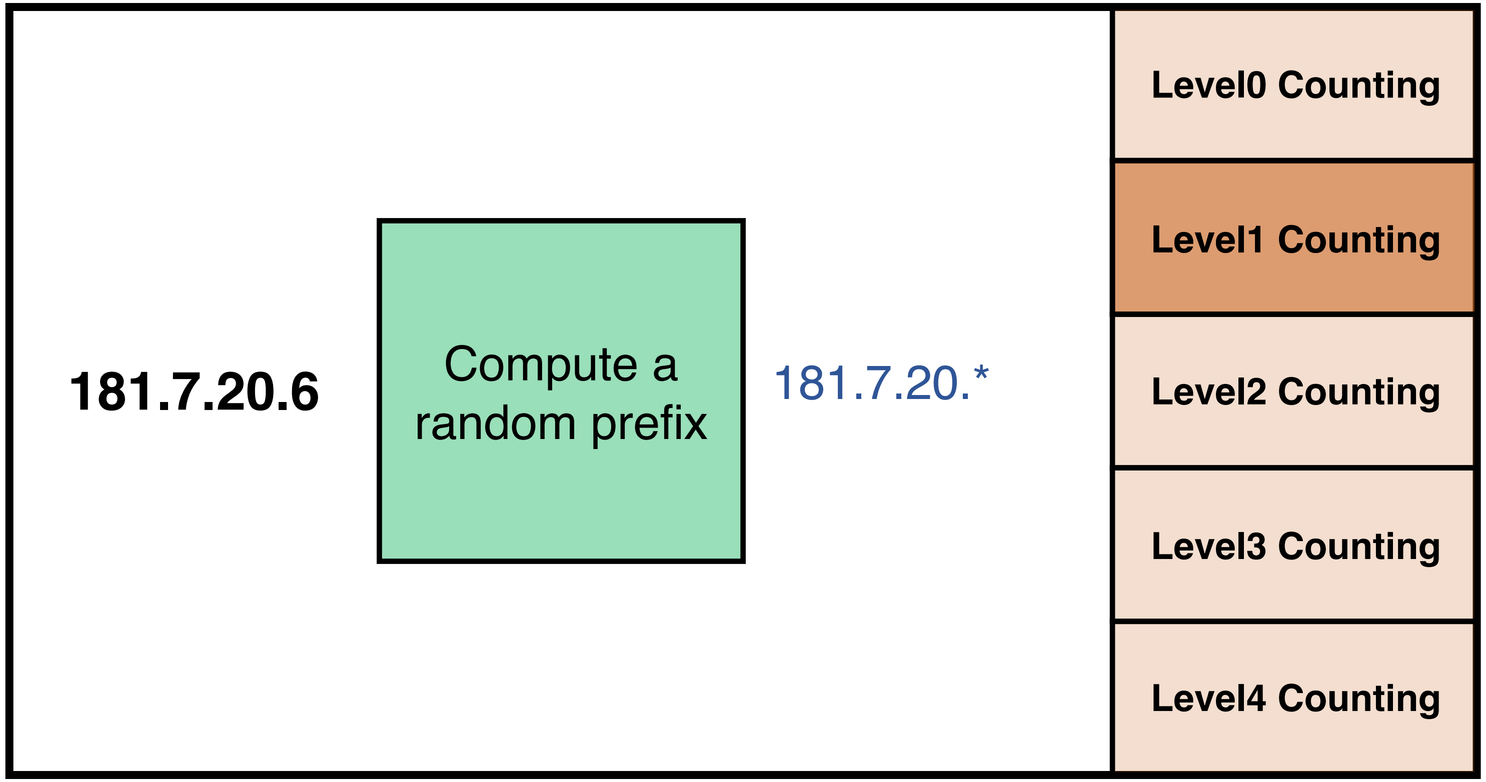
$P = \emptyset$
for *level* l=L **downto** 0
    for each item p at level l
        Let n be the lattice node that p belongs to
        $(f_{\min}(p), f_{\max}(p))$=GetEstimateSS($SS(n)$, p)
        $F'_p = f_{\max}(p)$
        $H_p = \{h \in P \text{ such that } \nexists h' \in P : h \prec h' \prec p\}$
        for each $h \in H_p$
            $F'_p = F'_p - f_{\min}(h)$
        for each pair of distinct elements h, h' in H_p
            $q = \mathrm{glb}(h, h')$
            if $\nexists h_3 \neq h, h'$ in H_p s.t. $q \preceq h_3$
                $F'_p = F'_p + f_{\max}(q)$
        if $F'_p \geq \phi N$
            $P = P \cup \{p\}$
            print(p, $f_{\min}(p)$, $f_{\max}(p)$)

Output

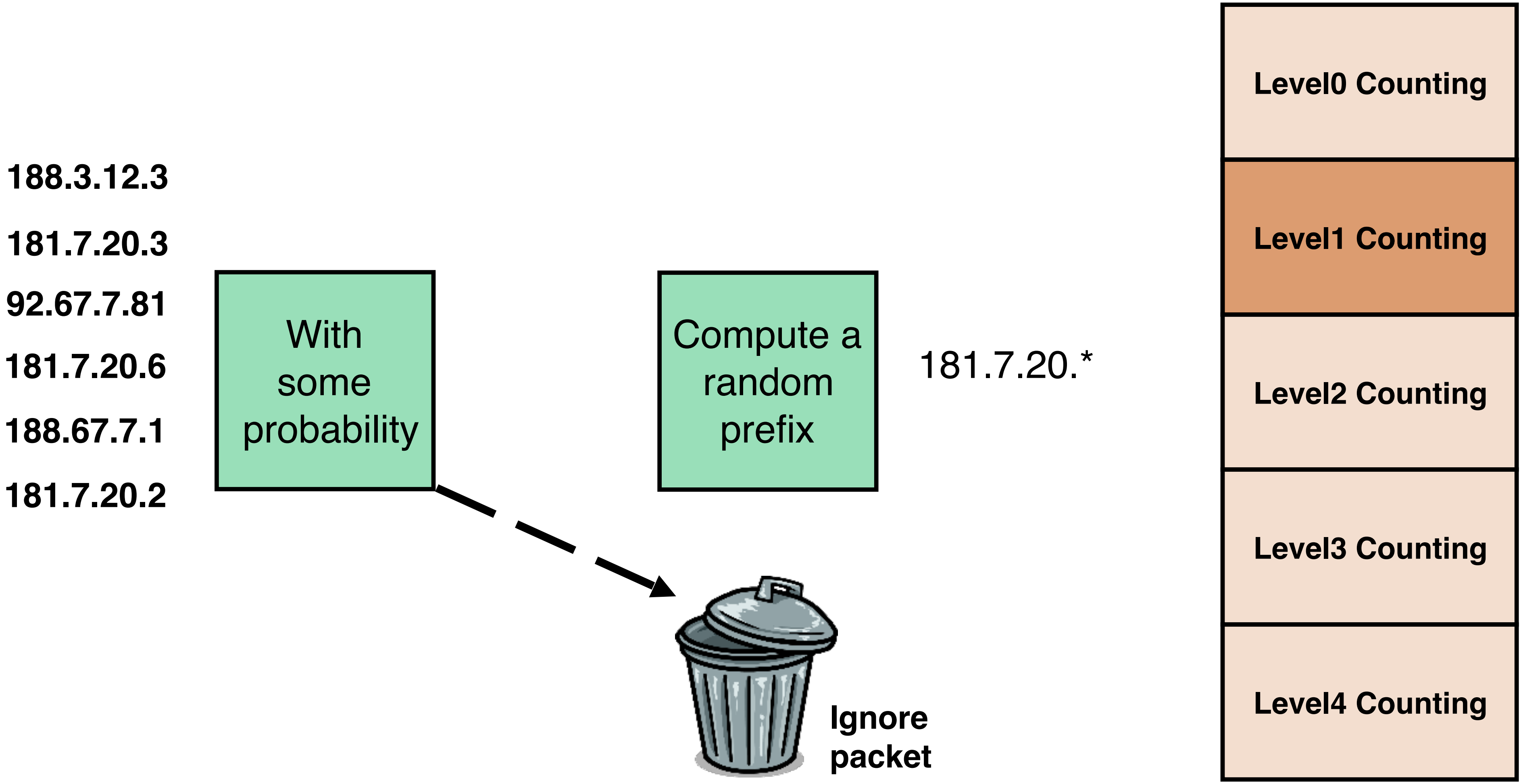


Randomized

Select a prefix at random and count it

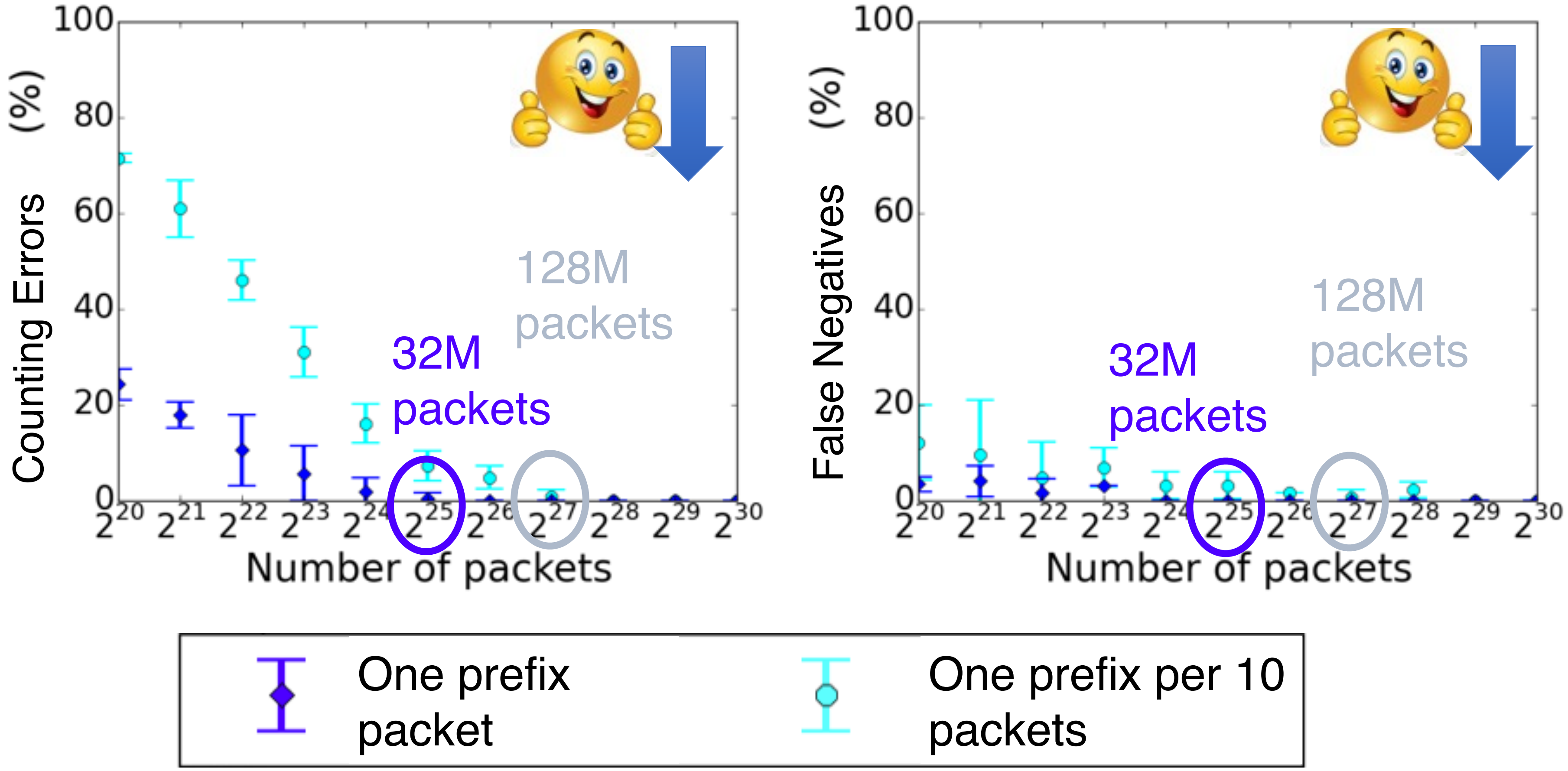


Additional Speedup



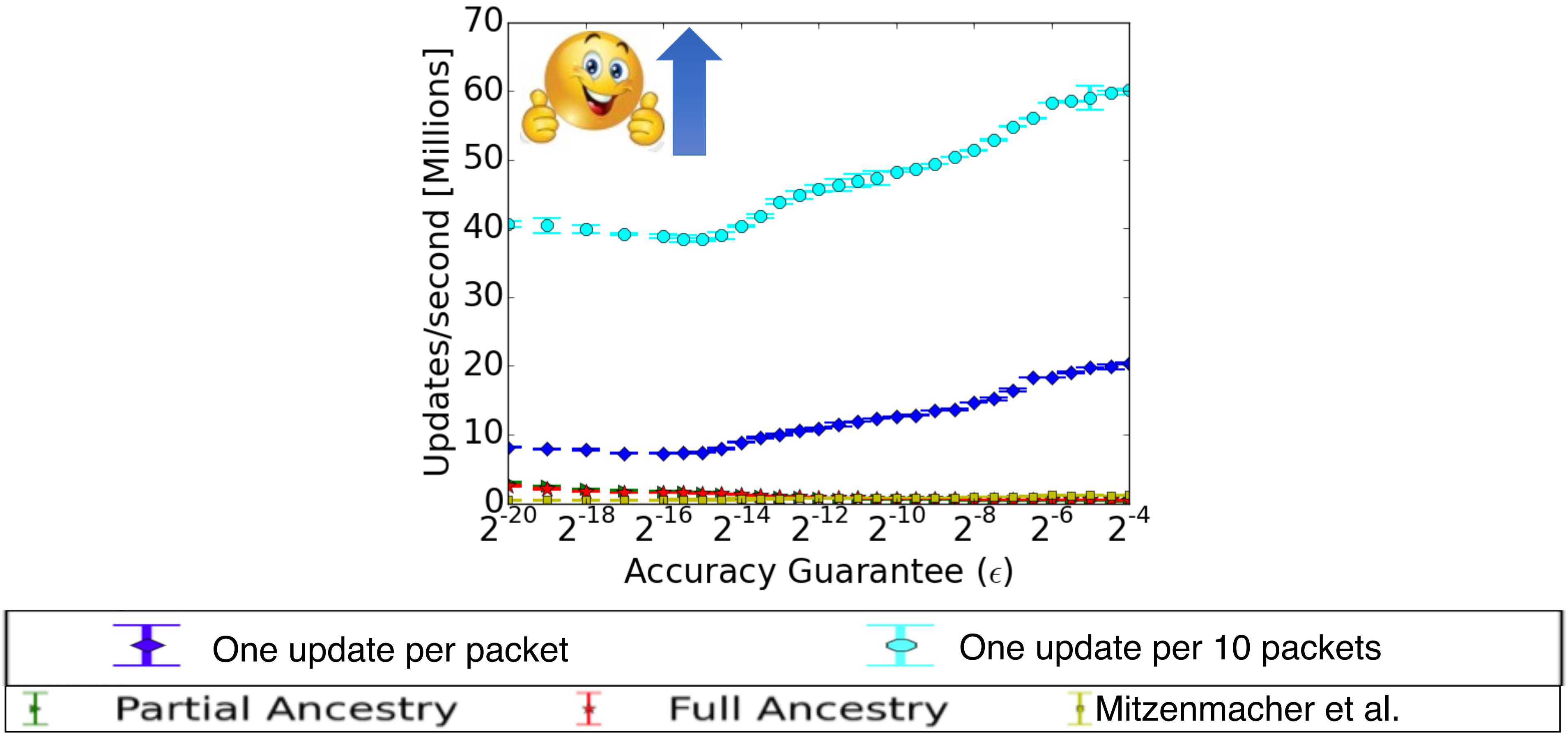
Accuracy

"Accuracy improves with the number of packets"

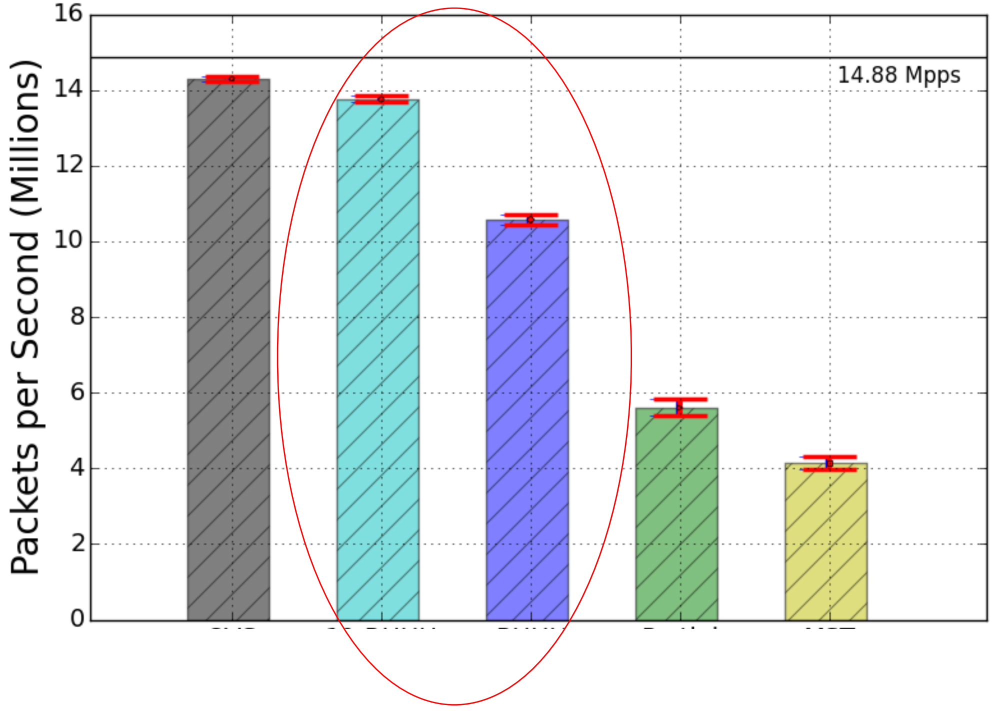


Performance

Up to X62 speedup!



Performance



Highlights:

- Only -4% overheads for HHH in the Open vSwitch data plane!
- +250% throughput improvement compared to previous work.