Counter Braids: A novel counter architecture

Balaji Prabhakar
Stanford University

Joint work with:
Yi Lu, Andrea Montanari, Sarang Dharmapurikar and Abdul Kabbani

Overview

- Counter Braids
  - Background: current approaches
    - Exact, per-flow accounting
    - Approximate, large-flow accounting
  - Our approach
    - The Counter Braid architecture
    - A simple, efficient message passing algorithm
  - Performance, comparisons and further work

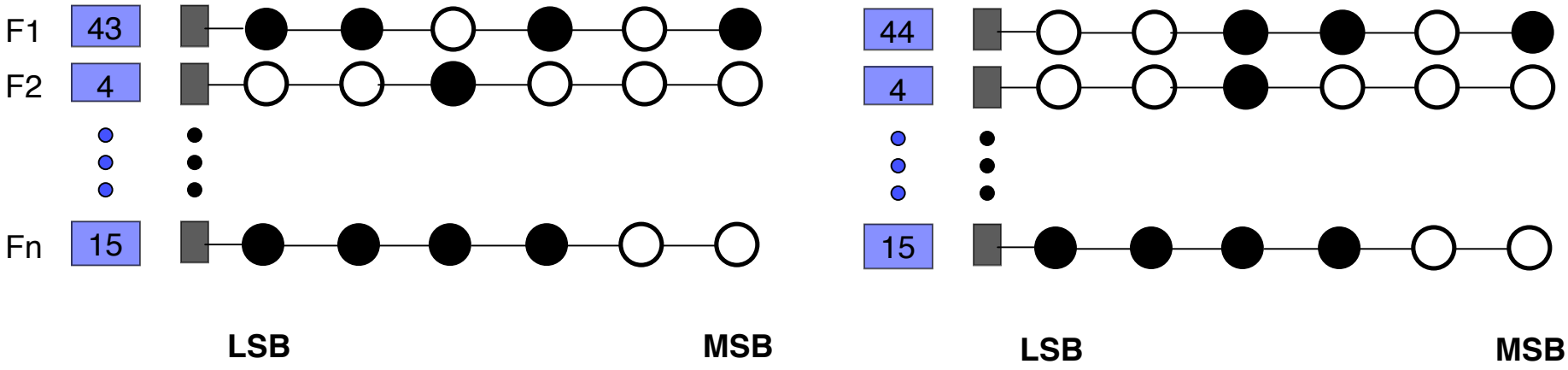
- Congestion notification in Ethernet
  - Overview of IEEE standards effort

Traffic Statistics: Background

- Routers collect traffic statistics; useful for
  - Accounting/billing, traffic engineering, security/forensics
  - Several products in this area; notably, Cisco's NetFlow, Juniper's cflowd, Huawei's NetStream
- Other areas
  - In databases: number and count of distinct items in streams
  - Web server logs
- Key problem: At high line rates, memory technology is a limiting factor
  - 500,000+ active flows, packets arrive once every 10 ns on 40 Gbps line
  - We need *fast* and *large* memories for implementing counters: v.expensive
- This has spawned two approaches
  - Exact, per-flow accounting: Use hybrid SRAM-DRAM architecture
  - Approximate, large-flow accounting: Use heavy-tailed nature of flow size distribution

Per-flow Accounting

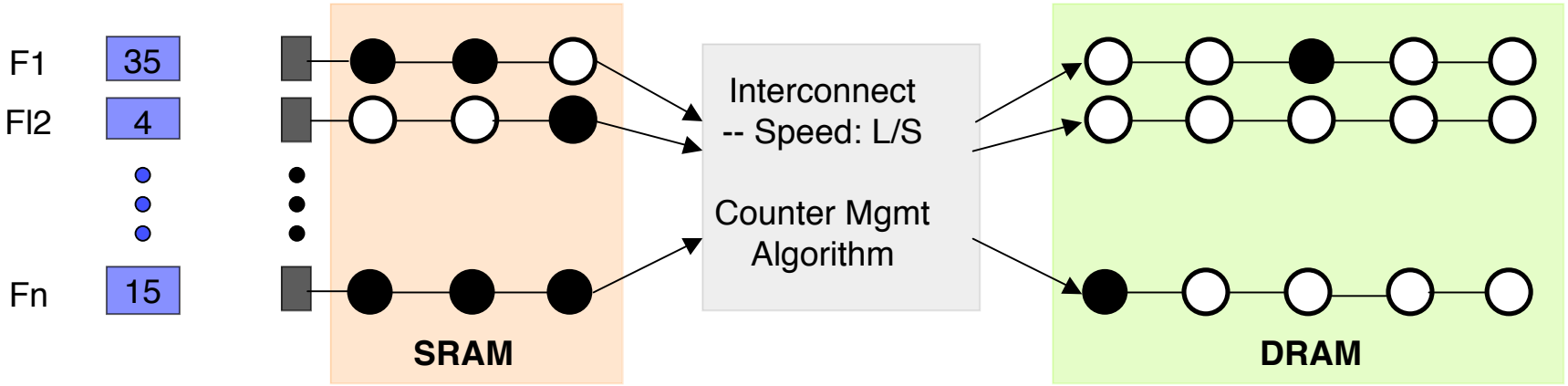
- Naïve approach: one counter per flow



- Problem: Need fast and large memories; infeasible

An initial approach
Shah, Iyer, Prabhakar, McKeown (2001)

- Hybrid SRAM-DRAM architecture
  - LSBs in SRAM: high-speed updates, on-chip
  - MSBs in DRAM: less frequent updates; can use slower speed, off-chip DRAMs

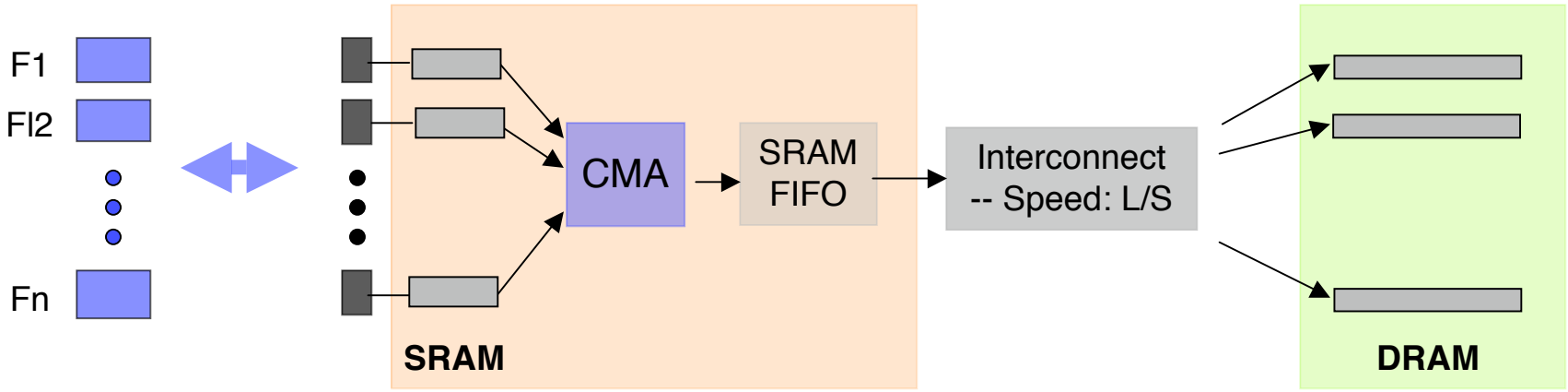


- Result: Under adversarial inputs, the minimum number of bits for each SRAM counter:

$$\log\left(\frac{\log(SN)}{\log(S/S-1)}\right) \approx \log\log N$$

Related work

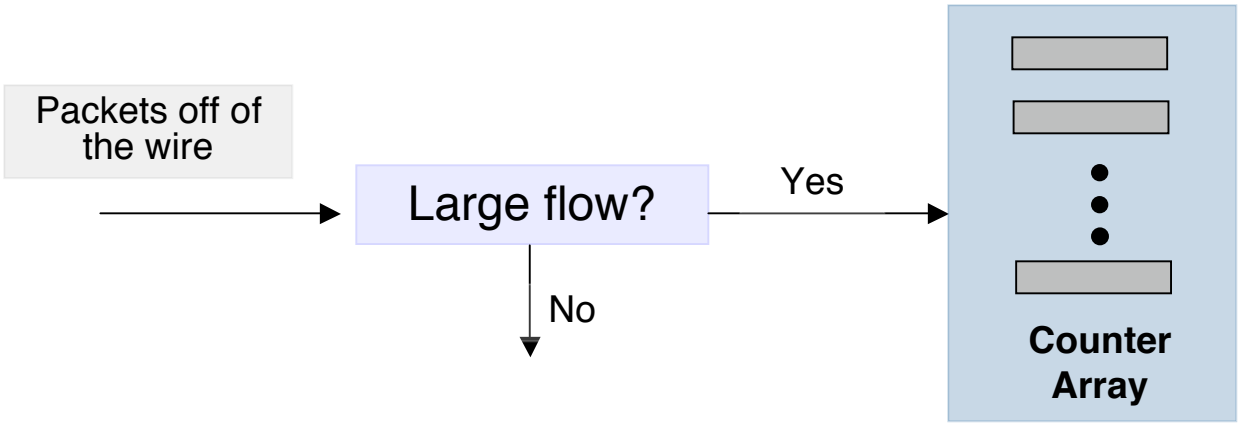
- Ramabhadran and Varghese (2003) obtained a simpler version of the LCF algorithm
- Zhao et al (2006) randomized the initial values in the SRAM counters to prevent the adversary from causing several counters to overflow closely



- Main problem of exact methods
  - Can't fit counters into single SRAM
  - Need to know the flow-counter association
    - Need perfect hash function; or, fully associative memory (e.g. CAM)

Approximate counting

- Statistical in nature
  - Use heavy-tailed (Pareto) distribution of network flow sizes
    - 80% of data brought by the biggest 20% of the flows
  - So, quickly identify these big flows and count their packets
- Sample and hold: Estan et al (2004)



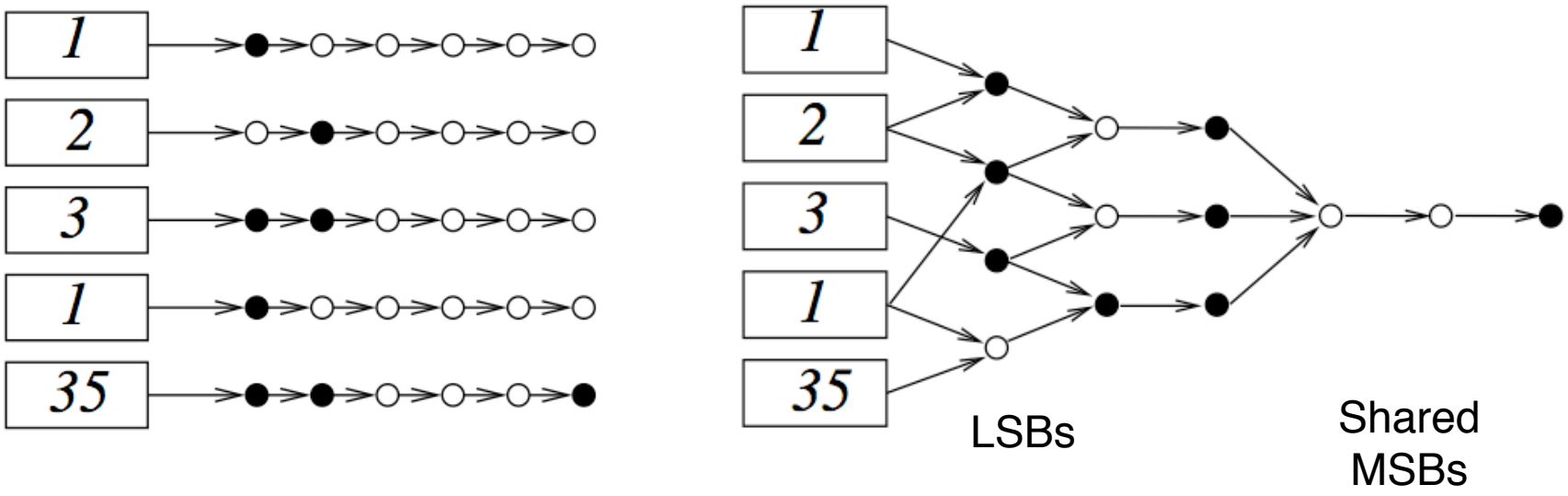
- Given the cost of memory, it strikes an good trade-off
  - Moreover, the flow-to-counter association problem is manageable
  - But, the counts are very approximate

Summary

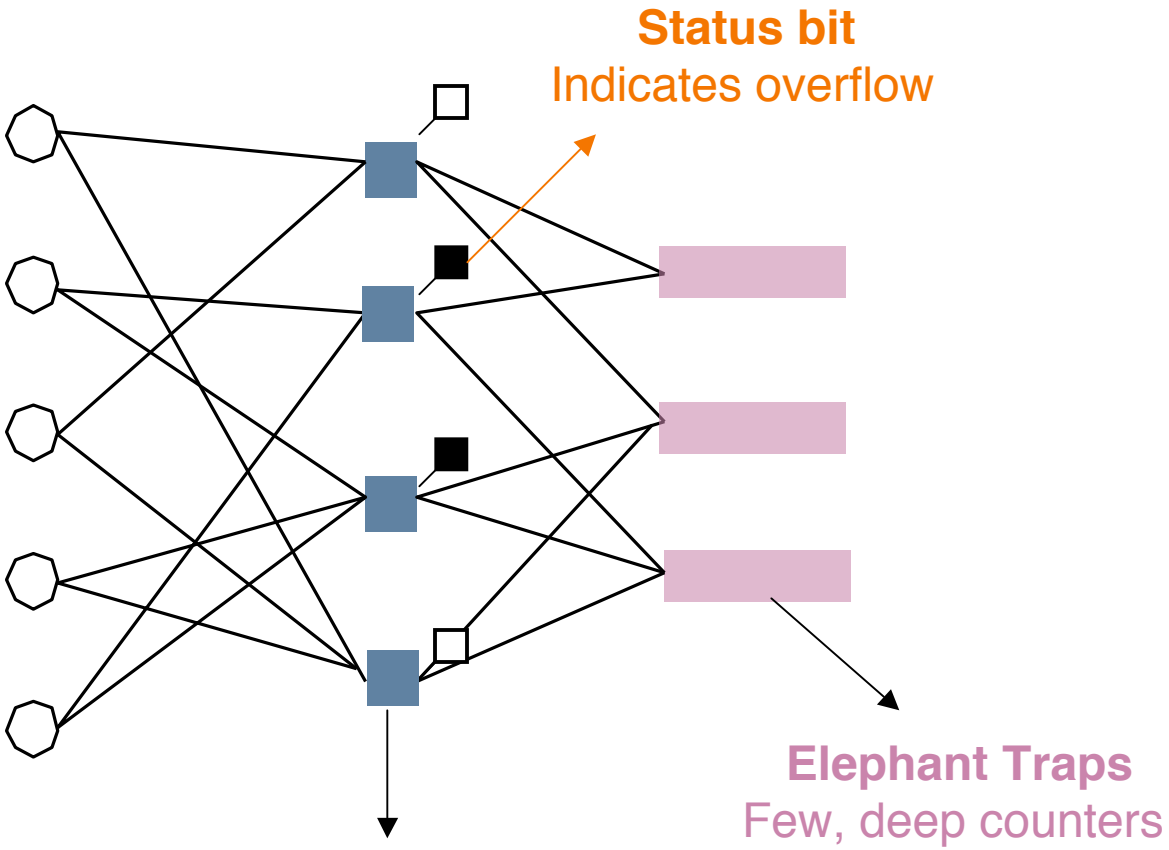
- Exact counting methods
  - Space intensive, complex
- Approximate methods
  - Focus on large flows, inaccurate
- Problems to address
  - Save space
  - Get rid of flow-to-counter association problem

Compress Space via Braiding

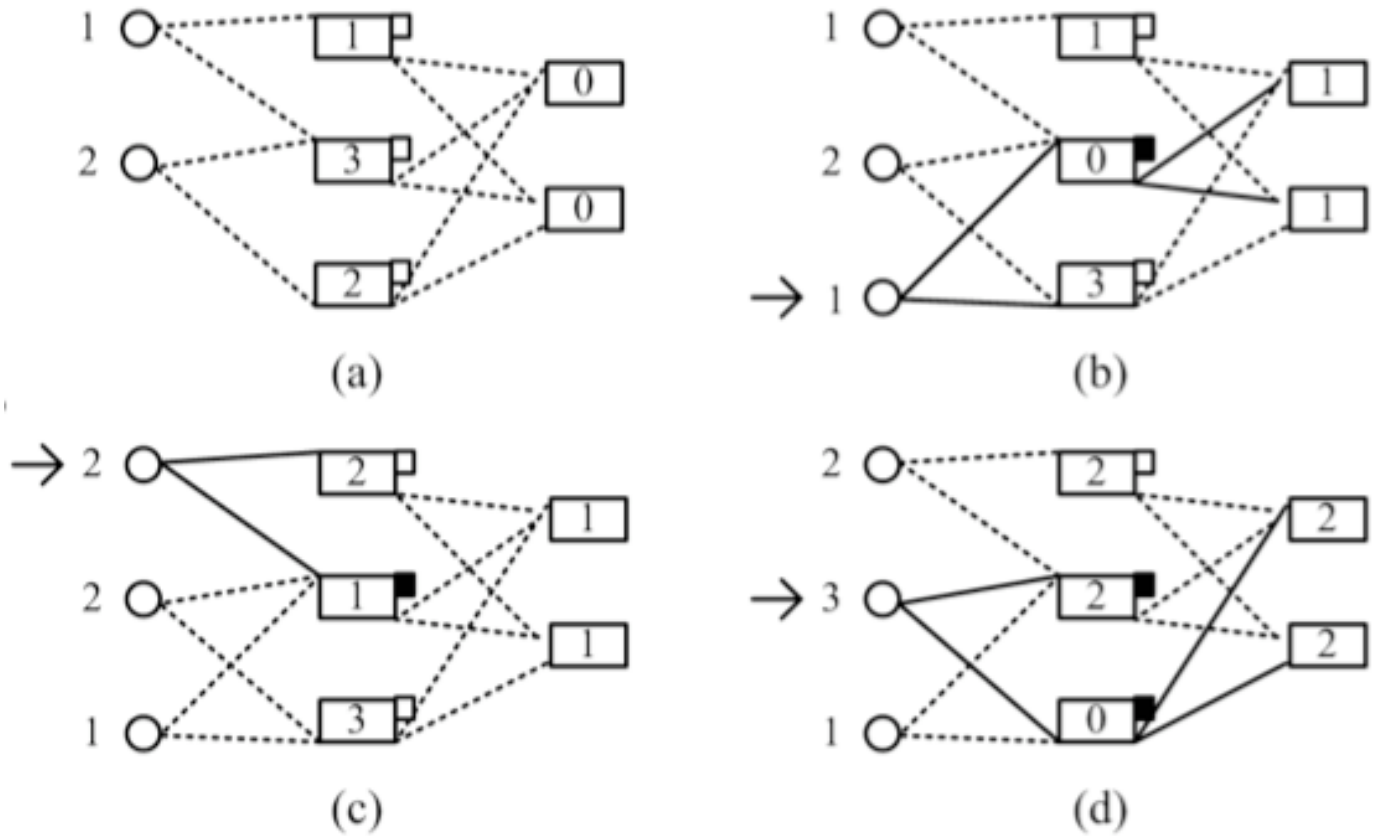
- Save counter space by “braiding” counters
  - Give nearly exclusive LSBs, share MSBs



Counter Braids for Measurement
(in anticipation)

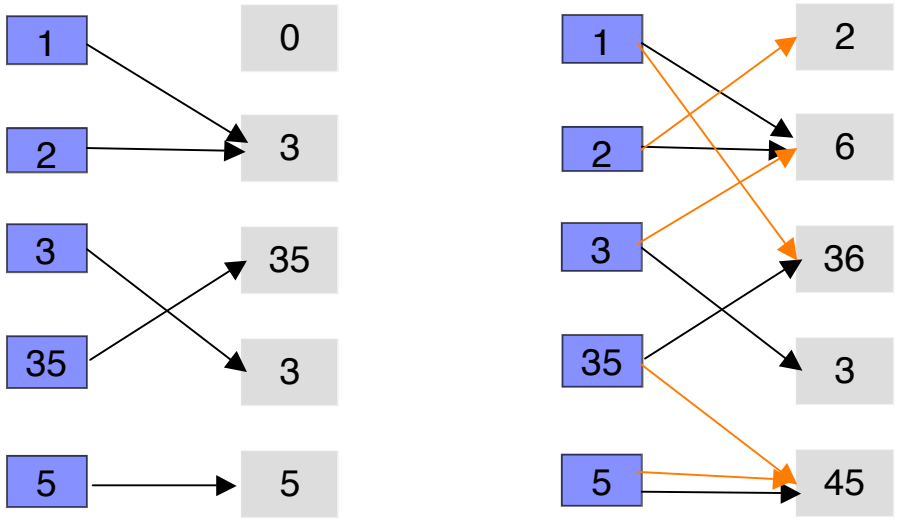


Counting with CBs



Multiple hashes to get rid of flow-to-counter association problem

- Multiple hash functions
  - Single hash function leads to collisions
  - However, one can use *two or more* hash functions and use the redundancy to recover the flow size



- Need efficient decoding algorithm for solving C = MF
  - Invert C --> F

Decoder 1: The MLE

- Consider a single stage of counters and multiple (random) hash functions
  - Let F be the vector of flow sizes, and C = MF be the vector of counter values; where M is the (random) adjacency matrix of dimensions m x n; m < n
  - Let $\{f_i\}$ be IID, and let H(F) be the entropy of the flow-size vector

- The MLE decoder
  - For an instance of the problem, let $F^1, \ldots, F^k$ be the list of all solutions
  - F^{MLE} is that solution which is most likely; i.e. if P_{flow} is the flow size distribution, then

$$F^{MLE} = \text{argmin}_i \{ D(F^i || P_{FLOW}) \}$$

- Theorem (Lu, Montanari, P): The MLE decoder is optimal; that is, the space needed asymptotically equals H(F)
  - This is interesting because C is a *linear, incremental* function of the data, F

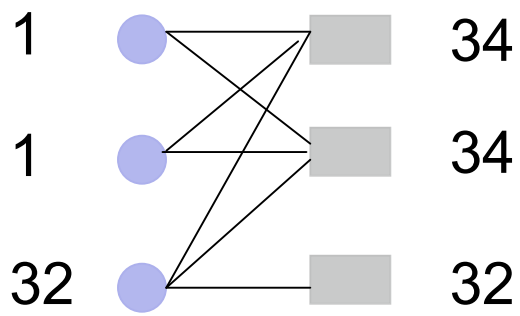
Related Work

- Compressed sensing
  - Storing sparse vectors using random linear transformations
    - Candes and Tao, Donoho, Indyk, Muthukrishnan, Wainwright, et al
- Problem statement

  minimize $\|F\|_1$ subject to $C = MF$
  - Main result of CS: reconstruction is exact if F is *sparse*
- But, for us
  - Linear transformations not necessarily sparse: lot of updating
  - LP decoding: worst-case cubic complexity
- Noiseless data compression with LDPC codes
  - Use regular graphs (i.e. not hash-based)
    - Caire, Shamai, Verdu, and Aji, Jin, Khandekar, MacKay, McEliece

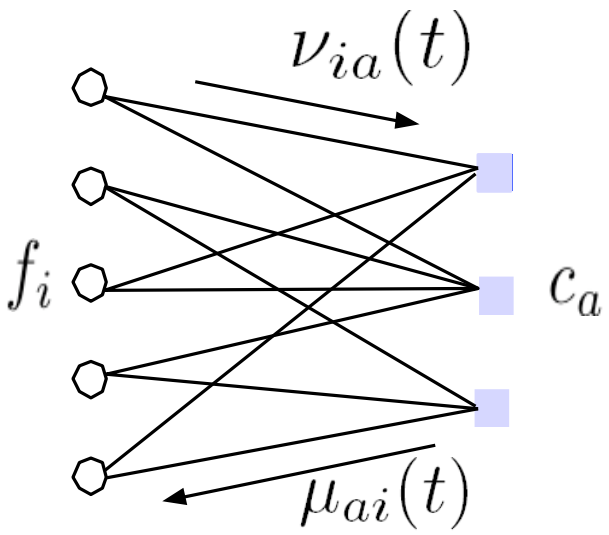
Practical algorithms: The Count-Min Algorithm

- This algorithm is due to Cormode and Muthukrishnan
  - Algorithm: Estimate flow j's size as the minimum counter it hits
  - The flow sizes for the example below would be estimated as: 34, 34, 32



- Major drawbacks
  - Need lots of counters for accurate estimation
  - Don't know how much the error is; in fact, don't know ***if*** there is an error
- We shall see that applying the "Turbo-principle" to this algorithm gives terrific results

The Turbo-principle



1: **Initialize**

2: $min =$ minimum possible flow size;

3: $\nu_{ia}(0) = min \; \forall i \in I$ and $a \in R$;

4: $c_a = a^{th}$ counter value

5: **Iterations**

6: for iteration number $t = 1$ to $niter$

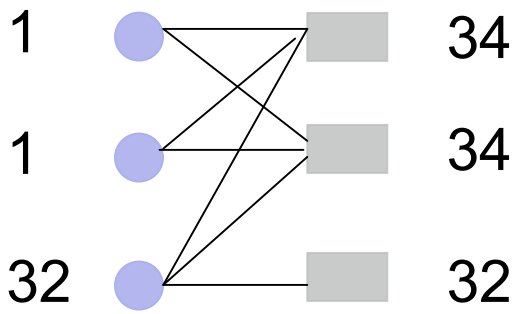
7: $$\mu_{ai}(t) = \max\left\{\left(c_a - \sum_{j \neq i} \nu_{ja}(t-1)\right), min\right\};$$

8: $$\nu_{ia}(t) = \begin{cases} \min_{b \neq a} \mu_{bi}(t) & \text{if } t \text{ is odd,} \\ \max_{b \neq a} \mu_{bi}(t) & \text{if } t \text{ is even.} \end{cases}$$

9: **Final Estimate**

10: $$\widehat{f}_i(t) = \begin{cases} \min_a \{\mu_{ai}(niter)\} & \text{if } t \text{ is odd,} \\ \max_a \{\mu_{ai}(niter)\} & \text{if } t \text{ is even.} \end{cases}$$

Example



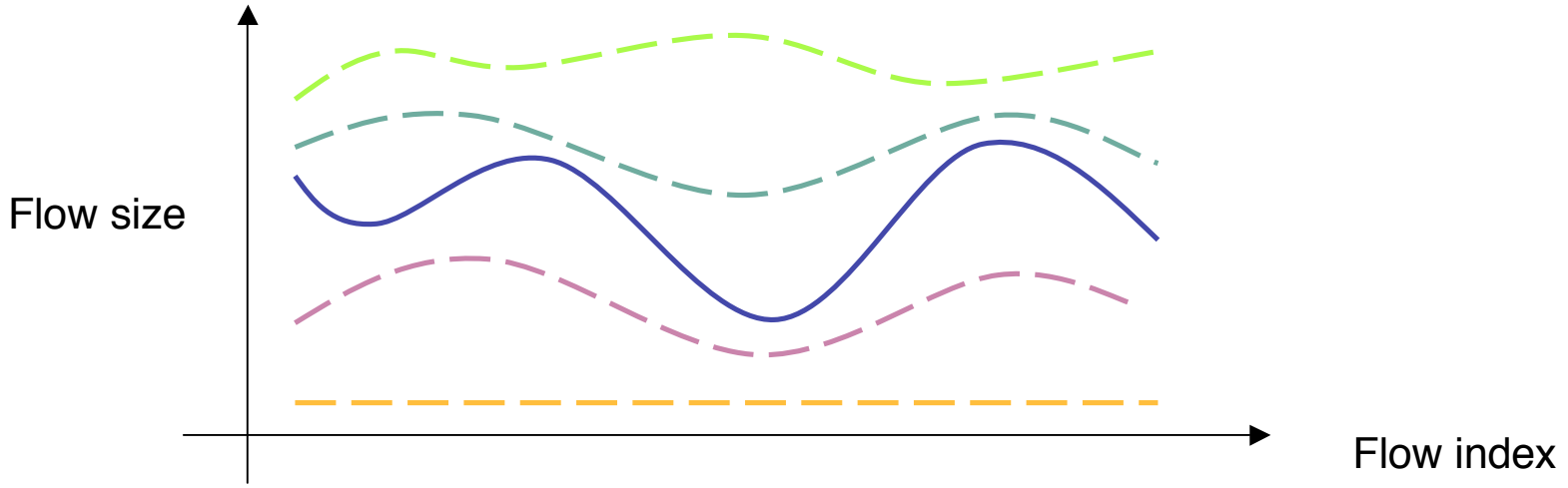
Iter 0	Iter 1	Iter 2	Iter 3	Iter 4
0	34	1	1	1
0	34	1	1	1
0	32	1	32	32

Count-Min

Properties of the MP Algorithm

- Anti-monotonicity: With initial estimates of 1 for the flow sizes,

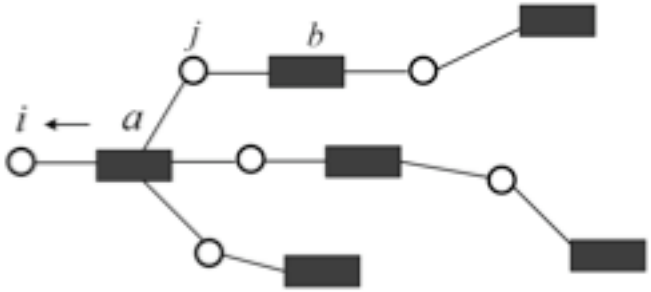
$$\hat{f}_i(2t) \leq \hat{f}_i(2t+2) \leq \ldots \leq f_i \leq \ldots \leq \hat{f}_i(2t+3) \leq \hat{f}_i(2t+1)$$



- Note: Because of this property, estimation errors are both detectable and have a bound!

When does the sandwich close?

- Answer 1: No assumption on flow size distribution.
  - Suppose we use k hash functions. Then, if m > k(k-1)n, the counters--flows graph becomes a tree and decoding is ***exact.***

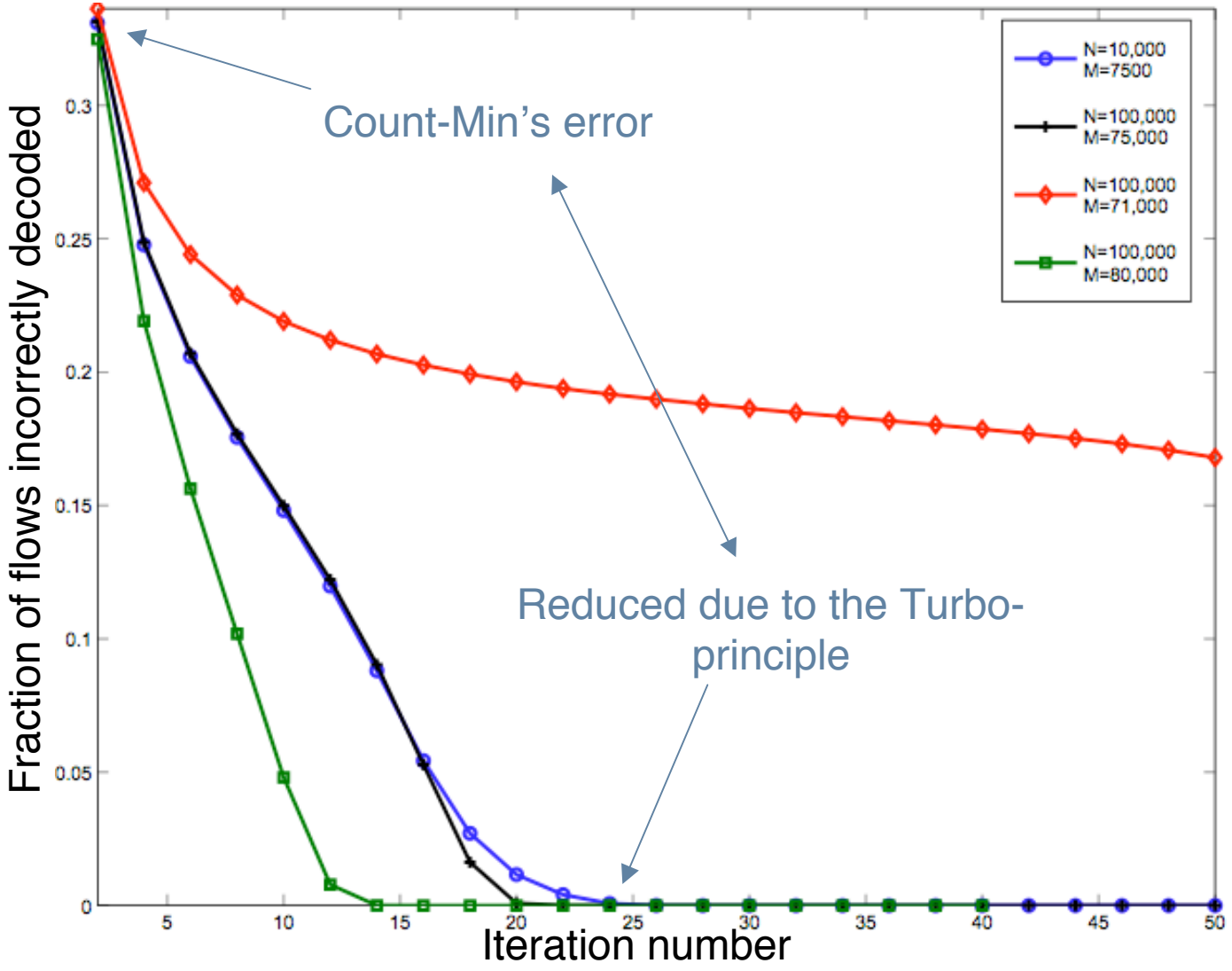


- Answer 2: Given the flow size distrubution.
  - Using the "density evolution" technique of Coding Theory, one can show that it suffices for m > c*n, where

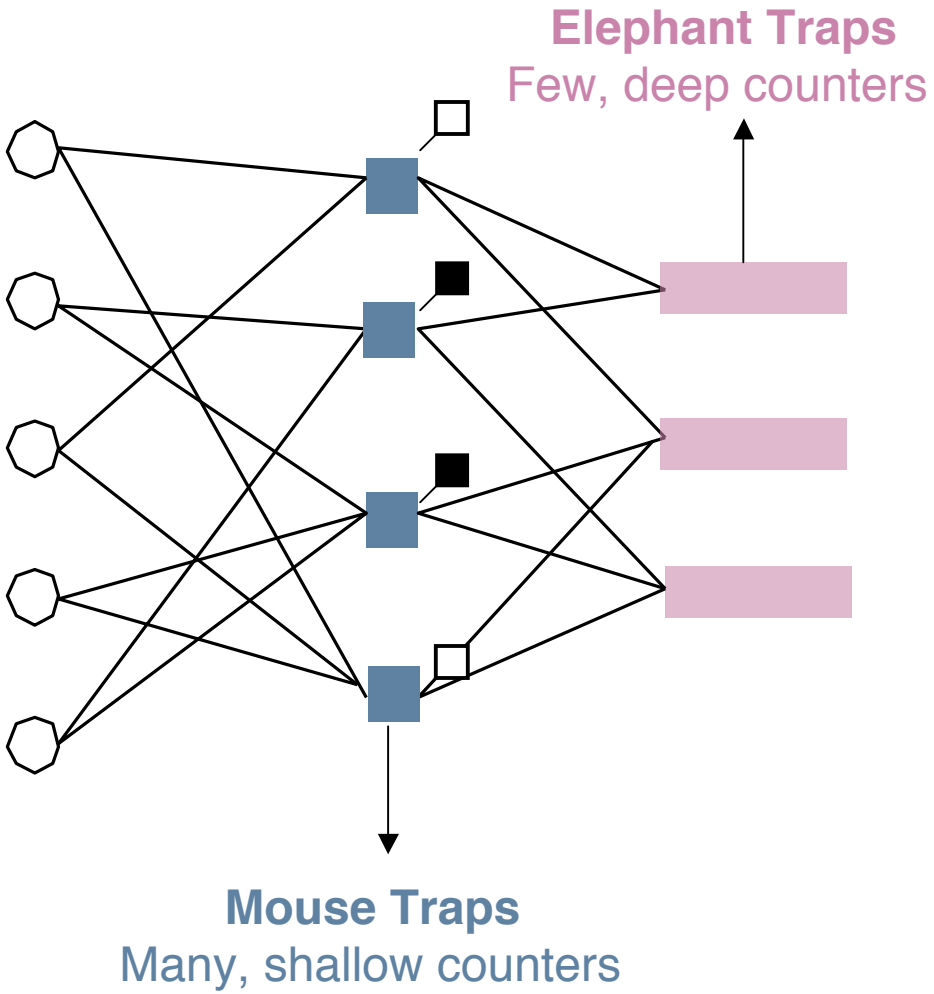
$$c^* = \sqrt{P(f > min)}$$

  - This means for heavy-tailed flow sizes, where there are approximately 35% 1-packet flows, c* is roughly 0.8

Threshold, c*= 0.72



The 2-stage Architecture: Counter Braids



-- First stage: Lots of shallow counters

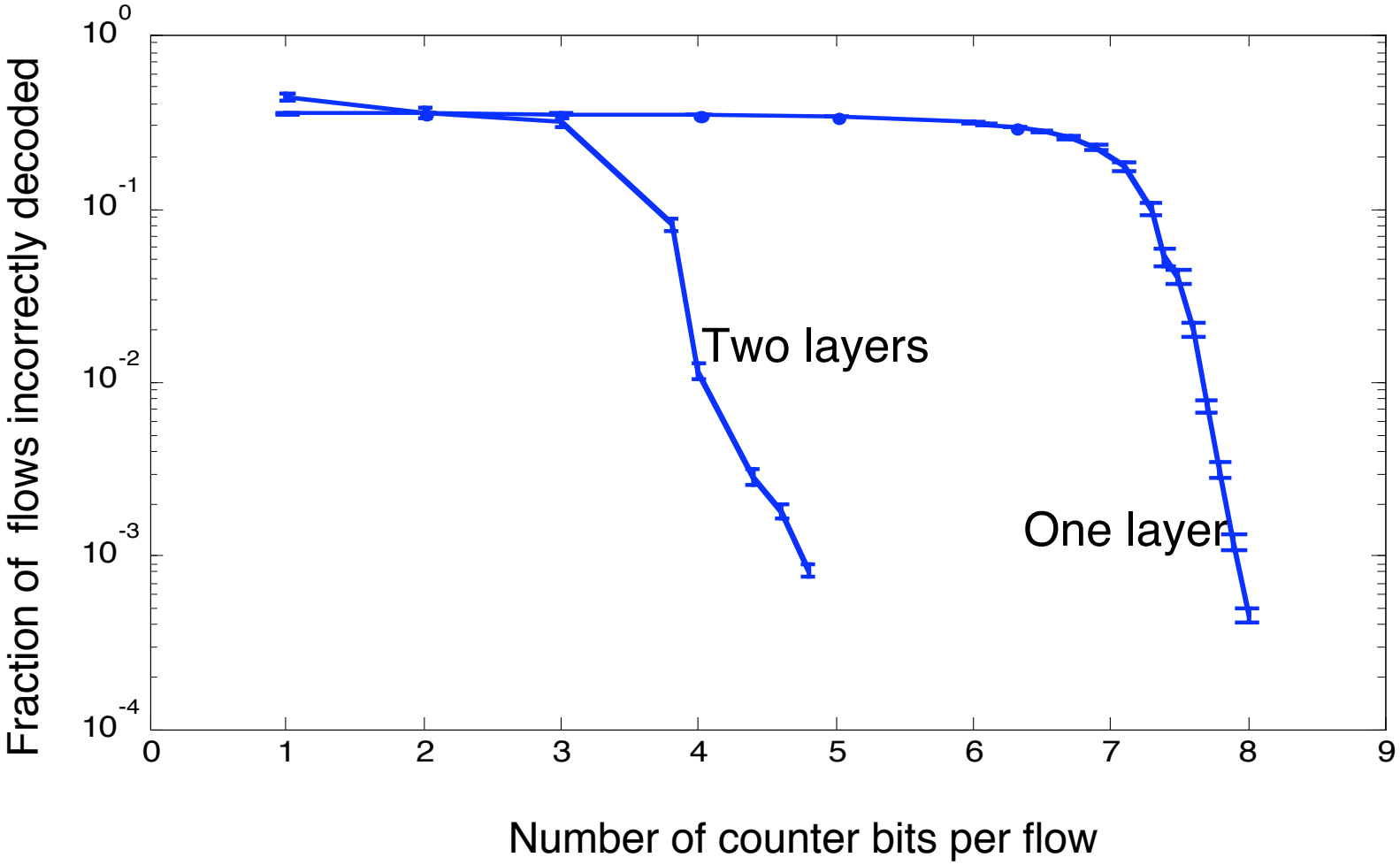
-- Second stage: V.few deep counters

-- First stage counters hash into the second stage; an “overflow” status bit on first stage counters indicates if the counter has overflowed to the second stage

-- If a first stage counter overflows, it resets and counts again; second stage counters track most significant bits

-- Apply MP algorithm recursively

Counter Braids vs. the One-layer Architecture



Internet Trace Simulations

- Used two OC-48 (2.5 Gbps) one-hour contiguous traces collected by CAIDA at a San Jose router.
- Divided traces into 12 5-minute segments.
  - Trace 1: 0.9 million flows and 20 million packets per segment
  - Trace 2: 0.7 million flows and 9 million packets per segment
- We used total counter space of 1.28 MB.
- We ran 50 experiments, each with different hash functions. There were a total of 1200 runs. ***No error*** was observed.

Comparison

	Hybrid	Sample-and-Hold	Count-Min	Counter Braids
Purpose	All flow sizes. Exact.	Elephant flows. Approximate.	All flow sizes. Approximate.	All flow sizes. Exact.
Number of flows	900,000	98,000	900,000	900,000
Memory Size (SRAM) counters	4.5 Mbit (31.5 Mbit in DRAM + counter-management)	1 Mbit	10 Mbit	10 Mbit
Memory Size (SRAM) flow-to-counter association	> 25 Mbit	1.6 Mbit	Not needed	Not needed
Error	Exact	Fractional Large: 0.03745% Medium: 1.090% Small: 43.87%	Pe ~ 1 avg abs error = 24.7	Lossless recovery.

Conclusions for Counter Braids

- Cheap and accurate solution to the network traffic measurement problem
  - Good initial results

- Further work
  - Lossy compression
  - Multi-router solution: same flow passes through many routers

Congestion Notification in Ethernet: Part of the IEEE 802.1 Data Center Bridging standardization effort

Berk Atikoglu, Abdul Kabbani, Balaji Prabhakar
Stanford University

Rong Pan
Cisco Systems

Mick Seaman

Background

- Switches and routers send congestion signals to end-systems to regulate the amount of network traffic.
  - Two types of congestion.
    - ***Transient:*** Caused by random fluctuations in the arrival rate of packets, and effectively dealt with using buffers and link-level pausing (or dropping packets in the case of the Internet).
    - ***Oversubscription:*** Caused by an increase in the applied load either because existing flows send more traffic, or (more likely) because new flows have arrived.
  - We've been developing QCN (for Quantized Congestion Notification), an algorithm which is being studied as a part of the IEEE 802.1 Data Center Bridging group for deployment in Ethernet

Switched Ethernet vs Internet

- Some significant differences …
  1. There is no end-to-end signaling in the Ethernet *a la* per-packet acks in the Internet
     - So congestion must be signaled to the source by switches
     - Not possible to know round trip time!
     - Algorithm not automatically self-clocked (like TCP)
  2. Links can be paused; i.e. packets may not be dropped
  3. No sequence numbering of L2 packets
  4. Sources do not start transmission gently (like TCP slow-start); they can potentially come on at the full line rate of 10Gbps
  5. Ethernet switch buffers are much smaller than router buffers (100s of KBs vs 100s of MBs)
  6. Most importantly, algorithm should be simple enough to be implemented completely in hardware

- An interesting environment to develop a congestion control algorithm
  - QCN derived from the earlier BCN algorithm
  - Closest Internet relatives: BIC TCP at source, REM/PI controller at switch