Turbocounting: Network Traffic Measurement using Sparse Graph Counters

Balaji Prabhakar
Stanford University

Joint work with:

Yi Lu, Andrea Montanari, Sarang Dharmapurikar and Abdul Kabbani

Overview

- Background
  - Approaches to network traffic measurement
    - Exact, per-flow accounting: Adversarial inputs
    - Approximate, large-flow accounting: Heavy-tailed flow sizes
- Our approach
  - The Counter Braid architecture
  - Optimality of the Maximum Likelihood Estimator
  - A simple, efficient Message Passing Estimator
- Performance and comparisons
- Further work

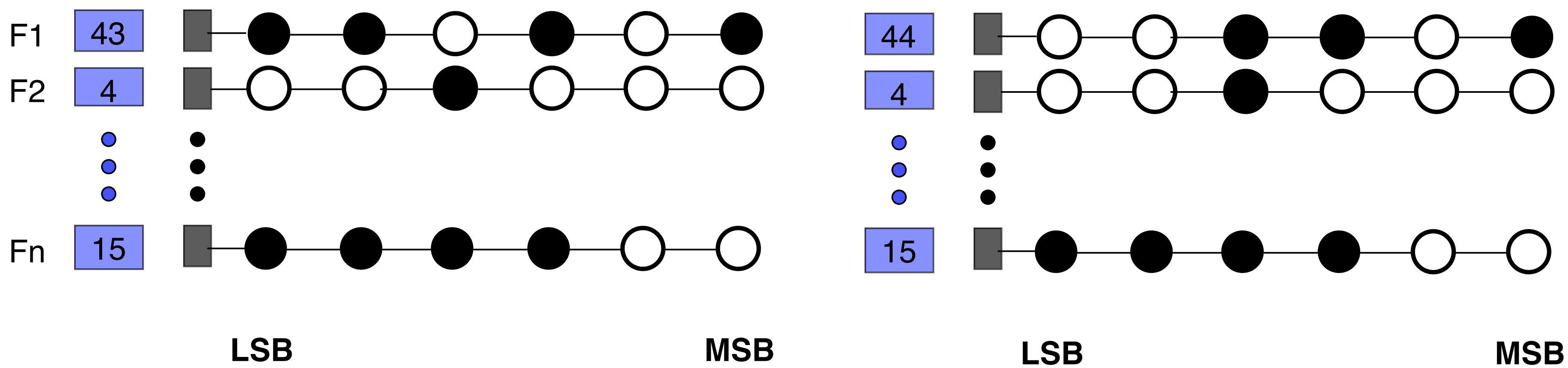
Traffic Statistics: Background

- Routers collect traffic statistics; useful for
  - Accounting/billing, traffic engineering, security/forensics
- Several products in this area, notably
  - Cisco Systems’ NetFlow, Juniper Networks’ cflowd, and Huawei Technology’s NetStream
- 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
  - This means we need *fast* and *large* memories for implementing counters
  - Memories are either fast (SRAM) or large (DRAM), need serious extra work to get both!
- This 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

- Definition of flow
  - Set of packets with common properties (e.g. same SA-DA, protocol, …)

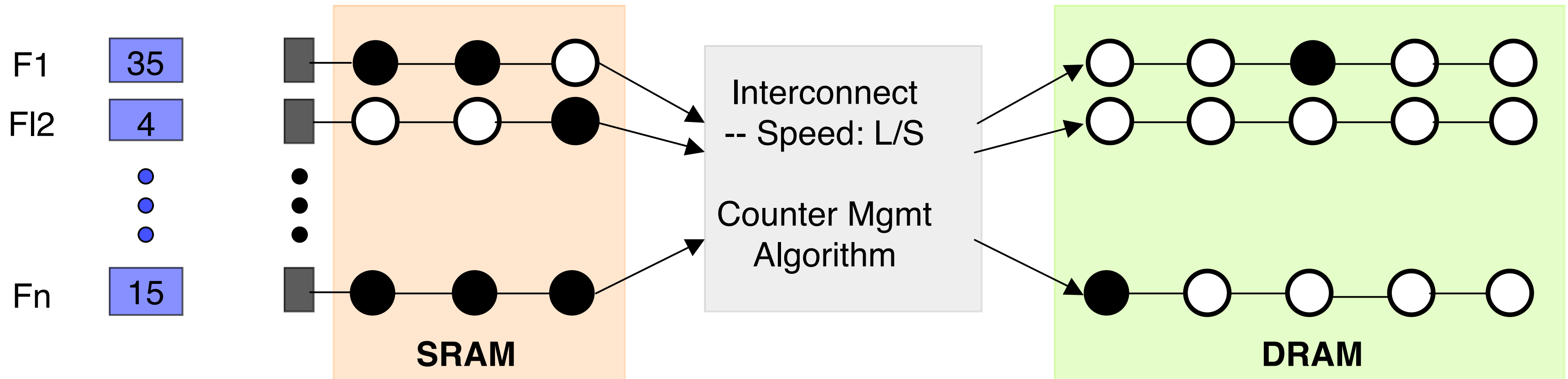
- Naïve approach to per-flow counting: one counter per flow



- 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

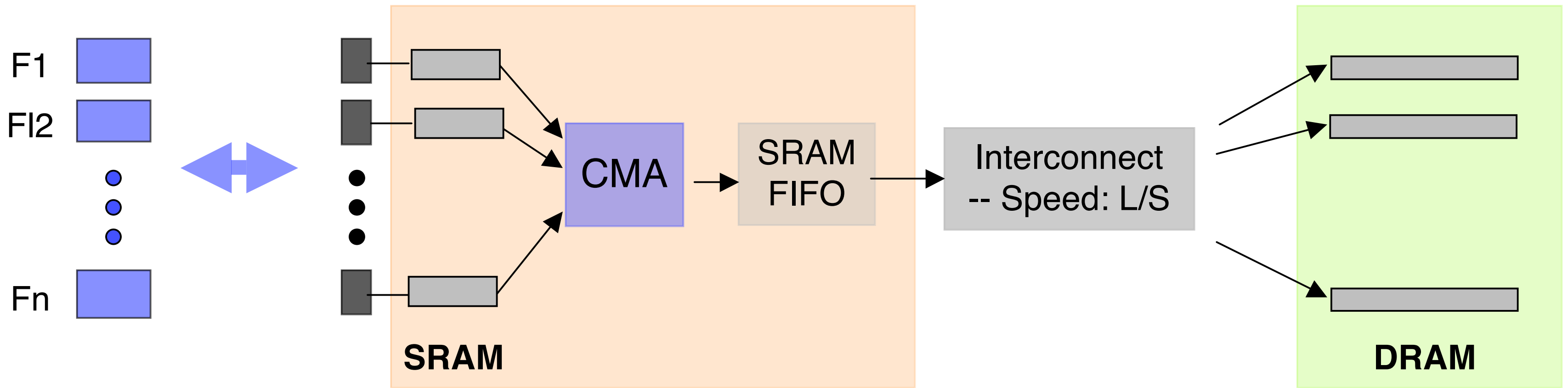


- The setup
  - Line speed = SRAM speed = L;   Interconnect speed = DRAM speed = L/S
  - Adversarial packet arrival process

- Results
  1. The counter management algorithm Longest Counter First is optimal
  2. Min. num. of bits for each SRAM counter: $\log\left(\frac{\log(SN)}{\log(S/S-1)}\right) \approx \log\log N$

Related work

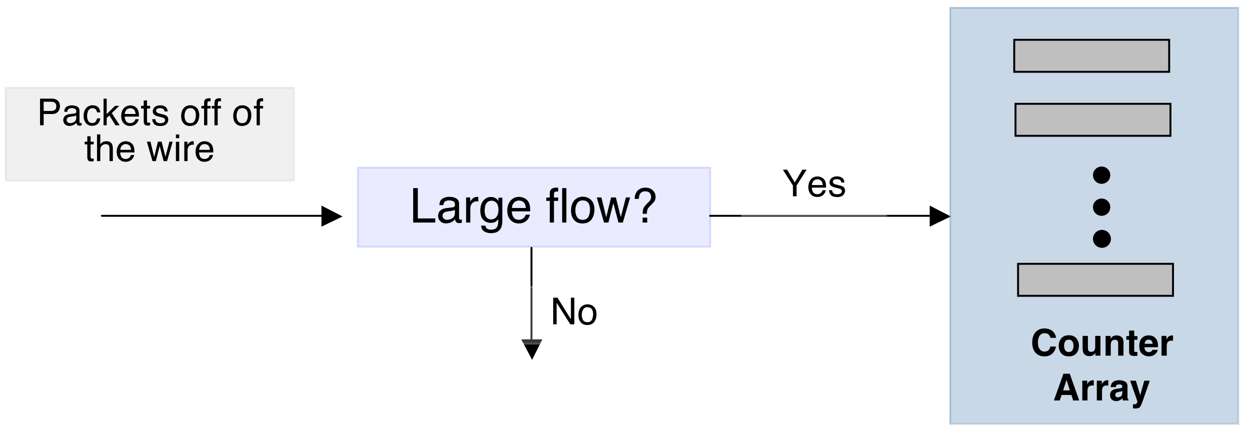
- Ramabhadran and Varghese (2003) obtained a simpler version of the LCF algorithm
- Zhao et al (2006) randomized the initial values in the SRAM counters to prevent the adversary from causing several counters to overflow closely



- 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 (often Pareto) distribution of network flow sizes
  - Roughly, 80% of data brought by the biggest 20% of the flows
  - So, it makes sense to quickly identify these big flows and count their packets
- Sample and hold: Estan et al (2004) propose sampling packets to catch the large “elephant” flows and then counting just their packets
  - Significantly simpler, but approximate



- This approach spawned a lot of follow-on work
  - Given the cost of memory, it strikes an excellent trade-off
  - Moreover, the flow-to-counter association problem is manageable

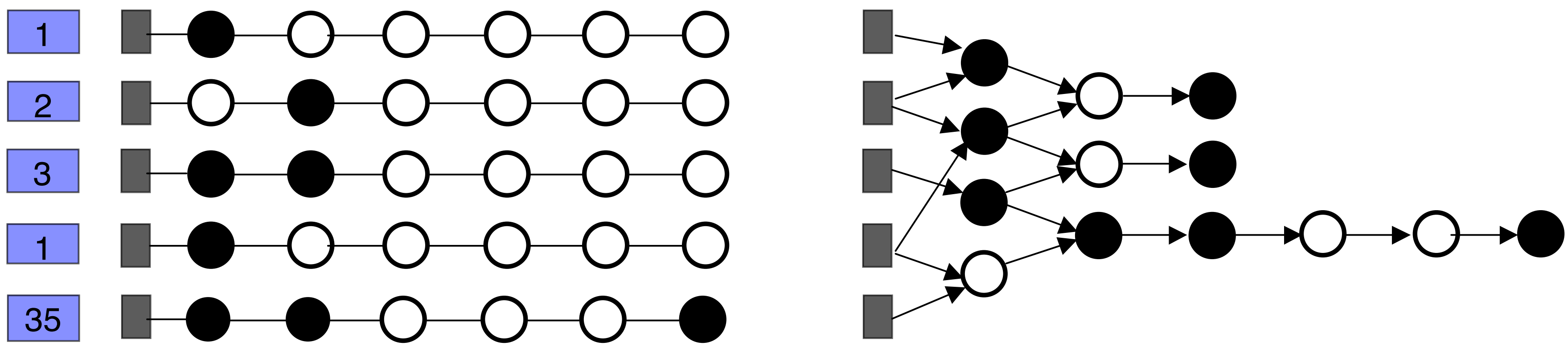
Summary

- Exact counting methods
  - Space intensive
  - Complex

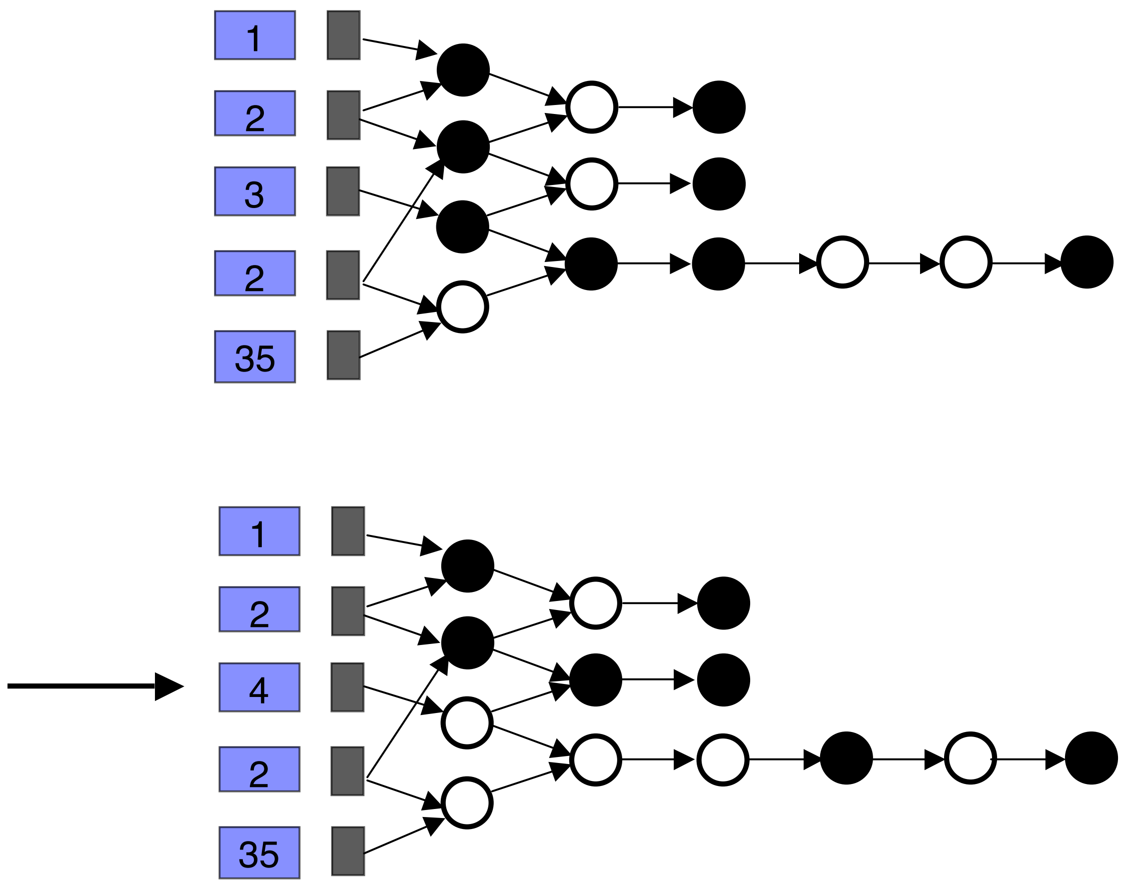
- Approximate methods
  - Focus on large flows
  - Not as accurate

Our approach

- The two problems of exact counting methods solved as follows
  1. Large counter space
     - By "braiding" the counters
  2. Flow-to-counter association problem
     - By using multiple hash functions and a "decoder"
- Braiding



Incrementing



Flow-to-counter association

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

- Main issues
  - A simple decoding algorithm
  - It's performance: how much space? what decoding accuracy?

Decoder 1: The MLE

- Consider a single stage of counters and multiple (random) hash functions
  - Let F be the vector of flow sizes
  - Then C = MF is the vector of counter values; where M is the (random) adjacency matrix of dimensions $m \times n$; $m < n$
  - Let $\{f_i\}$ be IID, and let H(F) be the entropy of the flow-size vector
  - Clearly, the total amount of counter space needed cannot be less than H(F)

- We get the following result: C = MF 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
  - It is not a priori obvious that F can be losslessly recovered from C
  - However, the decoder is very complex: it is the Maximum Likelihood Estimator (or MLE)

Decoder 1: The MLE

- Since $m < n$, the linear equation $C = MF$ is under-determined

- For an instance of the problem, let $F^1, \ldots, F^k$ be the list of all solutions

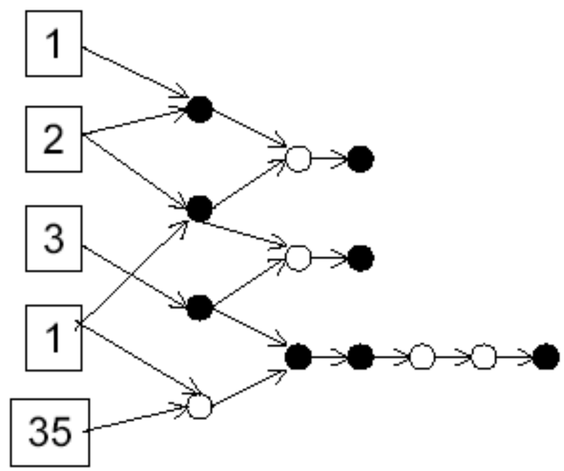
- The optimal solution F^{MLE} is the one which is most probable according the prior, P_{flow}, of the flow size distribution:

$$P_{flow}(F^{MLE}) >= P_{flow}(F^j) \quad \text{for all } j$$

  – More precisely, we use the Kullback-Leibler (or relative entropy) distance between the empirical distribution of F^j and P_{flow}

Optimality of MLE

- Counter braids: (G, q), $q \geq 2$
  - G: graph with input nodes, I, registers (counters), R
  - Each register is q-ary, reset on saturation
- Storage: $\mathsf{F} : \mathbb{N}^I \longrightarrow \mathbb{Z}_q^R$
- Estimation: $\hat{\mathsf{F}} : \mathbb{Z}_q^R \longrightarrow \mathbb{N}^I$



Optimality

- Admissible flow size distribution:
  - Power-law tails: $P(X_i \geq x) \leq Ax^{-\alpha}$
  - Decreasing digit entropy: If $\sum_{a \geq 0} X_i(a) q^a$, then entropy of $X_i(l)$ is monotonically decreasing in l
- Typical set decoder (or MLE):

$$x \in \mathsf{T}_n(p*) \text{ if its type } \theta_x \text{ satisfies } D(\theta_x \| p*) \leq n^{-\gamma},\ \gamma \in (0,1)$$

$$\widehat{\mathsf{F}}(y) = \begin{cases} \widehat{x} & \text{if } \mathsf{T}_n(p_*; y) = \{\widehat{x}\}, \\ * & \text{if } |\mathsf{T}_n(p_*; y)| \neq 1. \end{cases}$$

- Theorem: The typical set decoder is asymptotically optimal; i.e. the number of bits needed equals the entropy lower bound.

Related Work

- Compressed sensing
  - Storing sparse (binary) vectors using random linear transformations
    - Candes and Tao, Donoho, Indyk, Muthukrishnan, Wainwright, and many others
  - Linear transformations need not sparse: lots of updating, space
  - LP decoding: worst-case cubic complexity

- Noiseless data compression with LDPC codes
  - Use regular graphs (i.e. not hash-based)
  - Typical pairs decoding:
    - Caire, Shamai, Verdu
    - Aji, Jin, Khandekar, MacKay, McEliece

Decoder 2: The MP estimator

- The MLE is NP-hard, in general; need something very simple for our application

- We develop a message passing algorithm, inspired by iterative decoding techniques

Message Passing Algorithm for Solving C = MF

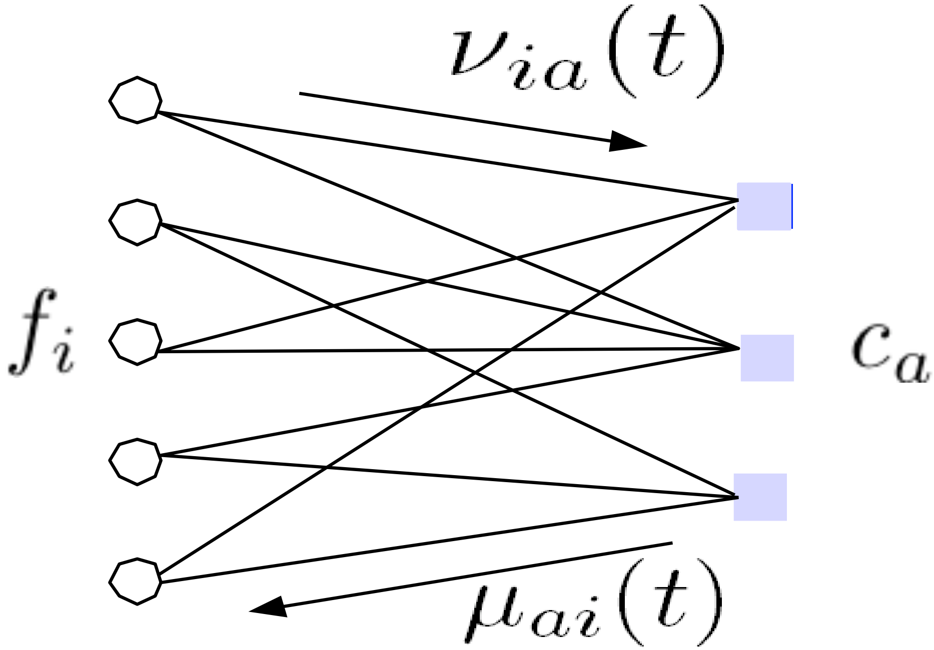
- Number of flows: n
- Number of counters: m

- Consider a single layer of the counter braid architecture

- Need to solve the equation: C = MF, where
  - C is vector of counter values
  - F is vector of flow values
  - M is the random incidence matrix of flows hashing into counters

- When n > m, we get a system of under-determined equations
  - Multiple solutions exist for the equation C = MF
  - The ML decoder chooses that solution which is most likely
  - But, this is too complex for implementation
  - We will now see a simple, suboptimal message-passing algorithm

Message Passing Algorithm for Solving C = MF



$$
\begin{aligned}
\mu_{ai}(t) &= \max\left\{\left(c_a - \sum_{j\neq i} \nu_{ja}(t-1)\right), 0\right\} \\
\nu_{ia}(t) &= \min_{b\neq a}\{\mu_{bi}(t)\} \\
\hat{f}_i(t) &= \min_a\{\mu_{ai}(t)\},
\end{aligned}
$$

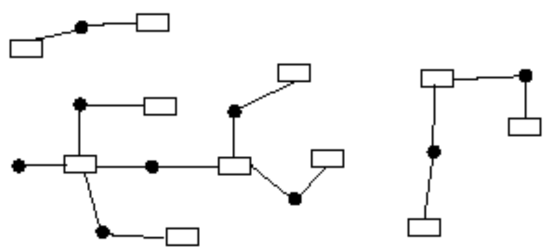
Properties of the MP Algorithm

- Sandwich property:

  LEMMA 1. *If* $\nu_{ia}(t-1) \le f_i$ *for every* i *and* a, *then* $\nu_{ia}(t) \ge f_i$. *Conversely, if* $\nu_{ia}(t-1) \ge f_i$ *for every* i *and* a, *then* $\nu_{ia}(t) \le f_i$.

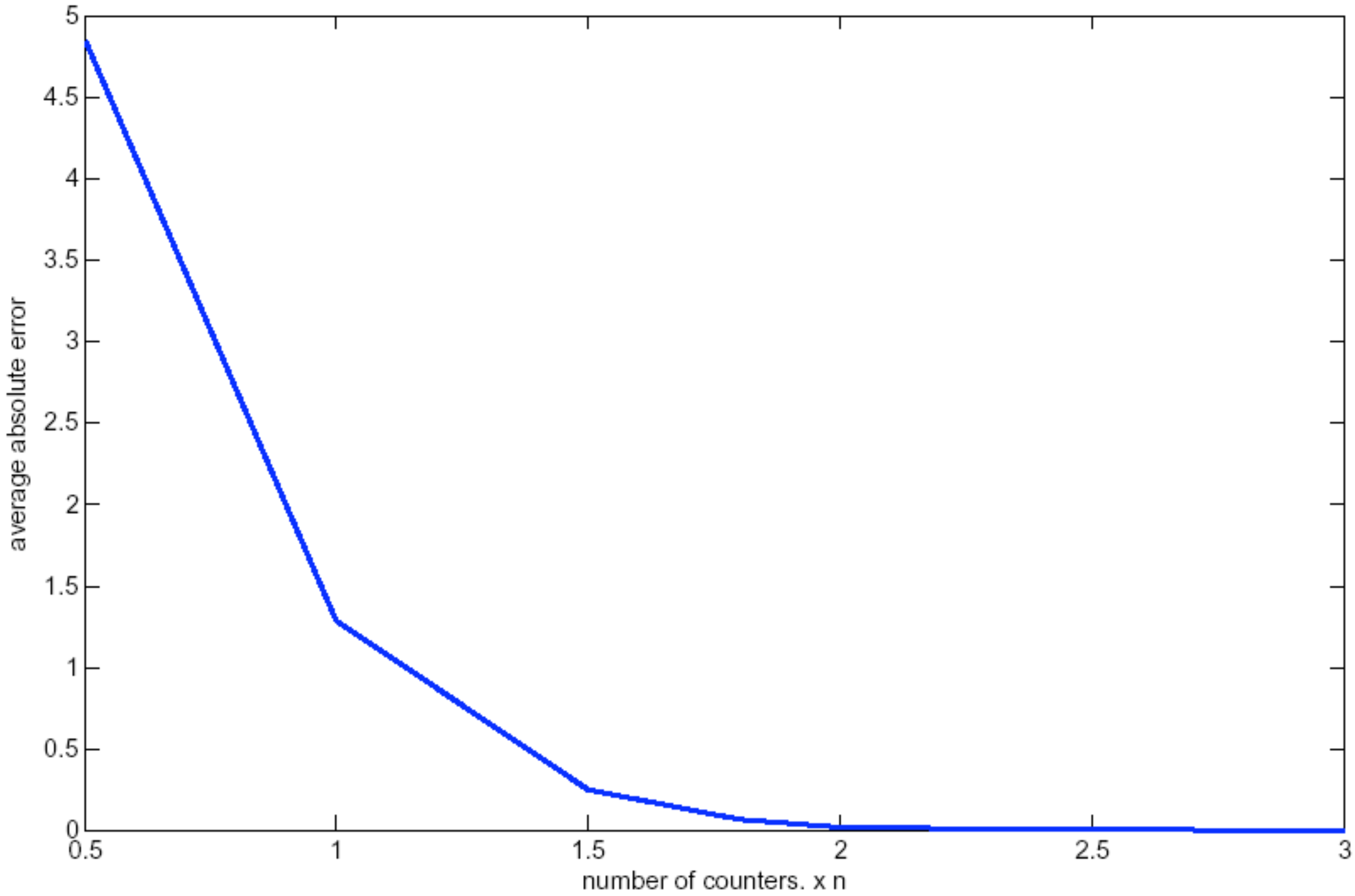
  – Therefore, if we start with all $F_e(0) = 0$, then subsequent estimates alternately upper- and lower-bound the true value, F. We have convergence if the sandwich closes.

- The MP algorithm is exact on trees.
  – If there are a large number of counters relative to flows (more precisely, if m > 2n), then the graph becomes a forest
  – But, we are interested in m << n

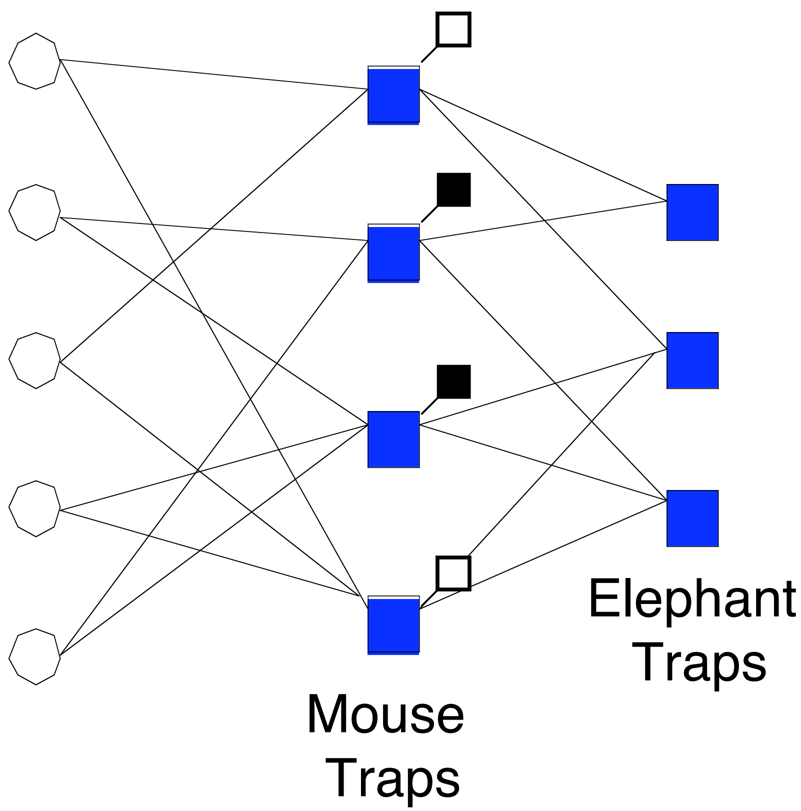


The Correct Number of Counters

- We see empirically that if m = 1.2 n, then we get good decoding with the message-passing algorithm



The 2-stage Architecture: Counter Braids



-- First stage: Lots of shallow counters; mouse traps

-- Second stage: V.few deep counters; elephant traps

-- 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

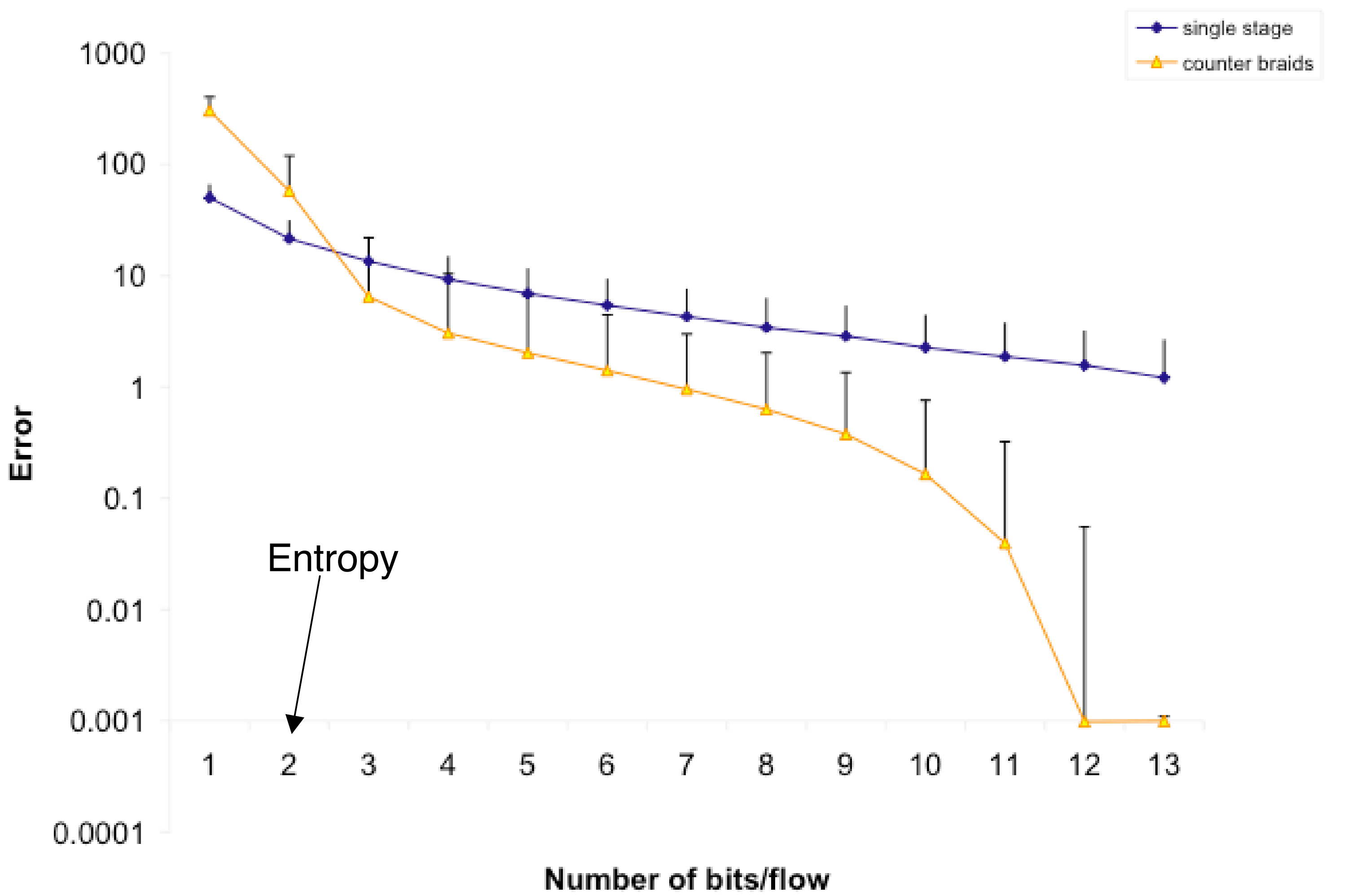
-- If a first stage counter overflows, it resets and counts again; second stage counters track most significant bits

-- Apply MP algorithm recursively

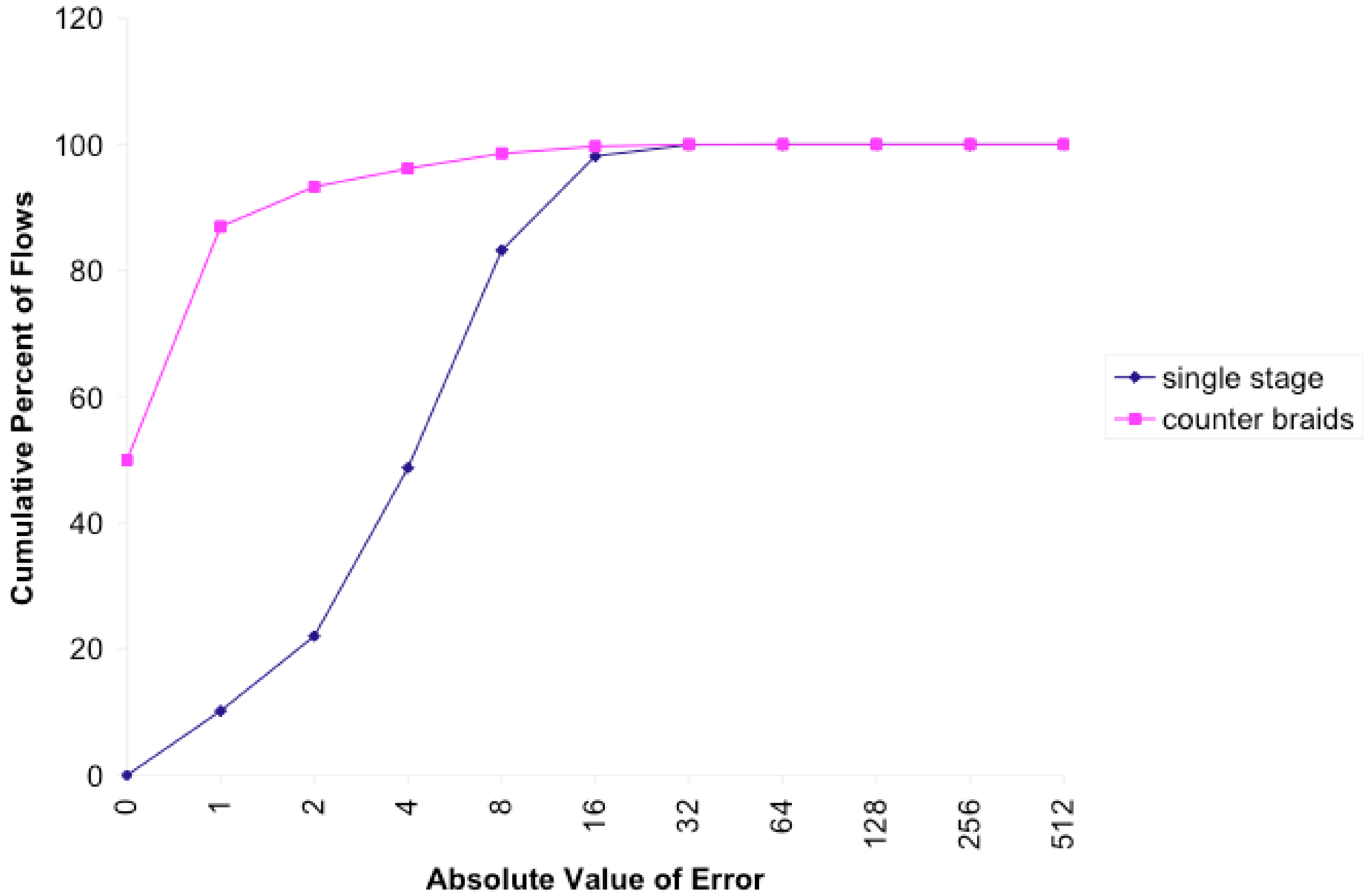
Performance of the MP Algorithm

- Interested in absolute error as a function of flow size
  - Pareto flow sizes
  - Entropy = 1.96 bits
  - Max flow size = 7364
  - Number of flows = 100,000

Counter Braids vs. the Single-stage Architecture



It's much better…
(comparison for 3xEntropy, incl. status bit!)



Features

- The counter braid architecture is
  - Optimal: Asymptotic number of bits needed matches entropy lower bound
  - An incremental compressor: As packets arrive, flow sizes not known in advance
  - Sparse, random hash-based: Fast updates, easy decoding
  - Multi-layered: Easy to pipeline

Conclusions

- Cheap and accurate solution to the network traffic measurement problem
  - Message Passing Decoder
  - Counter Braids
- Initial results showed that the performance was quite good
- Further work
  - Multi-stage generalization of Counter Braids
  - Analyze MP algorithm
  - Multi-router solution: same flow passes through many routers