

Data Streams Algorithms

Phillip B. Gibbons
Intel Research Pittsburgh

Guest Lecture in 15-853
Algorithms in the Real World

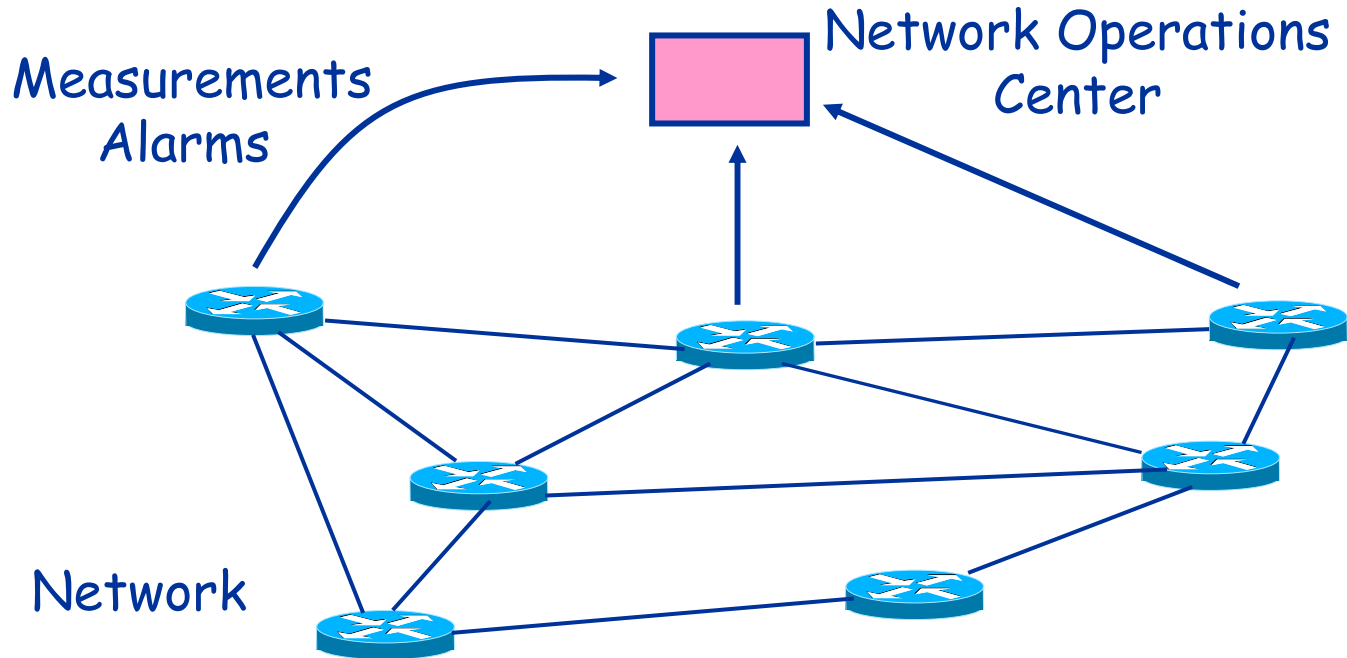
Outline

- Data Streams in the Real World
- Formal Model
- Important Basic Techniques: FM & AMS
- Distributed Streams Algorithms for Sliding Windows
- Synopsis Diffusion for Sensor Networks
- Further Results & Conclusions

Processing Data Streams

- Many applications generate streams of data
 - Performance measurements in network monitoring
 - Call detail records in telecommunications
 - Transactions in retail chains, ATM operations in banks
 - Log records generated by Web Servers
 - Sensor network data
- Application characteristics
 - Massive volumes of data (several terabytes)
 - Records arrive at a rapid rate
- Goal: Maintain aggregates / statistics on data streams in real-time

Example: Network Management



Massive amounts of rapidly-arriving data at each node

IP Network Measurement Data

- IP session data (collected using Cisco NetFlow)

Source	Destination	Duration	Bytes	Protocol
10.1.0.2	16.2.3.7	12	20K	http
18.6.7.1	12.4.0.3	16	24K	http
13.9.4.3	11.6.8.2	15	20K	http
15.2.2.9	17.1.2.1	19	40K	http
12.4.3.8	14.8.7.4	26	58K	http
10.5.1.3	13.0.0.1	27	100K	ftp
11.1.0.6	10.3.4.5	32	300K	ftp
19.7.1.2	16.5.5.8	18	80K	ftp

- AT&T collects 100 GBs of NetFlow data each day!

Network Data Processing (I)

- Traffic estimation
 - How many bytes were sent between a pair of IP addresses?
 - What fraction network IP addresses are active?
 - List the top 100 IP addresses in terms of traffic
- Traffic analysis
 - What is the average duration of an IP session?
 - What is the median of the number of bytes in each IP session?

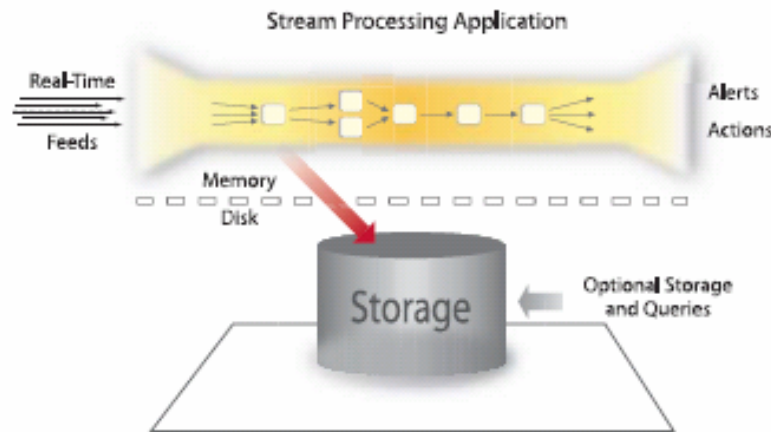
Network Data Processing (II)

- Fraud detection
 - List all sessions that transmitted more than 1000 bytes
 - Identify all sessions whose duration was more than twice the normal
- Security/Denial of Service
 - List all IP addresses that have witnessed a sudden spike in traffic
 - Identify IP addresses involved in more than 1000 sessions

\$150 BILLION MONEY MANAGER BUILDS REAL-TIME ANALYTICS

Managing institutional money, some of it in alpha strategies, makes significant demands on real-time market data and fast analytics. So Bridgewater Associates, a Westport, CT-based firm that managed \$150 billion in global investments for a wide array of institutional clients, used stream processing software from StreamBase to build some key components in its investment infrastructure – only a few of which it will discuss publicly.

Bridgewater uses StreamBase to process market data from multiple providers and check for errors, all in real-time, said Ed Thieberger, who leads the firm's trading technology group. The firm has also built a trading decision support application using the StreamBase technology.



StreamBase conducts its analysis in real-time and offers the option of storing market data as well.

Bridgewater had built some real-time applications, said Thieberger, and then it had some applications that it wanted to build but the company didn't have the code it needed. So it turned to StreamBase.

itor on the system and found a vendor's system had five times the latency advertised. Another firm built a compliance engine in eight days, after failing to build one in Java in more than eight months.

"It runs very fast on a Windows Server," Hobbib said. "You can process 100,000 messages per second on a \$2,000 Dell."

At Wall Street on Demand, a Boulder, CO-based company that develops investor Web sites for many of the nation's largest brokerage firms, StreamBase helps handle all the information coming in from nearly 1,000 market data feeds. When clients want

to add new features to their retail investor sites, they want the innovations up and running fast, said John Leslie, chief technology officer at the company. In the past, adding a new market data stream could take be-

StreamBase (II)

- Algorithmic/Automated trading
 - Computing value-added analytics (e.g. VWAP, Black Scholes)
 - Filtering ticker data from multiple sources to capture time-critical events such as best bid/ask
 - Run sophisticated intraday trading algorithms
- Compliance Management and Market Surveillance
- Risk Management
 - Monitor positions and P&L across multiple trading operations
 - Track movement of all market variables for each risk model they are running.

StreamBase (III)

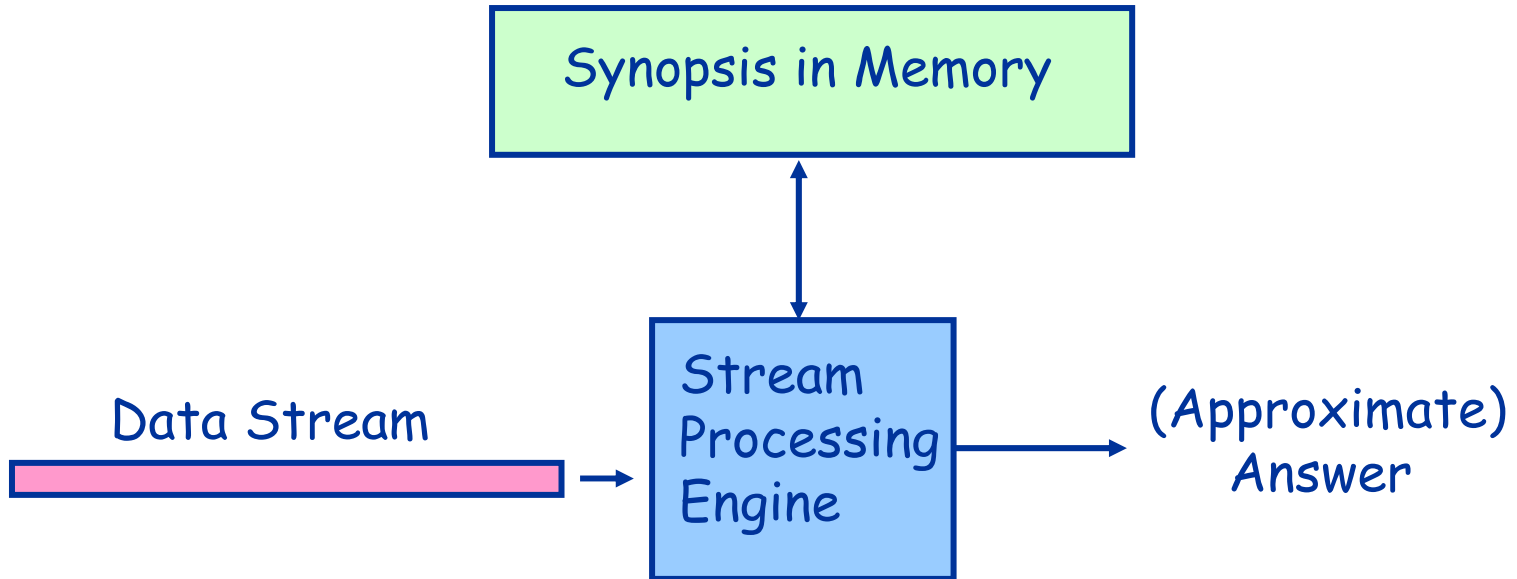
- Trial projects with NASA
 - Processing high-volume, disparate **oceanographic, atmospheric & meteorological data**
 - Real-time analysis of high-volume **sensor-generated data** from a variety of remote & in-situ sensing device to better identify scientific trends and perform predictive analysis

Outline

- Data Streams in the Real World
- **Formal Model**
- Important Basic Techniques: FM & AMS
- Distributed Streams Algorithms for Sliding Windows
- Synopsis Diffusion for Sensor Networks
- Further Results & Conclusions

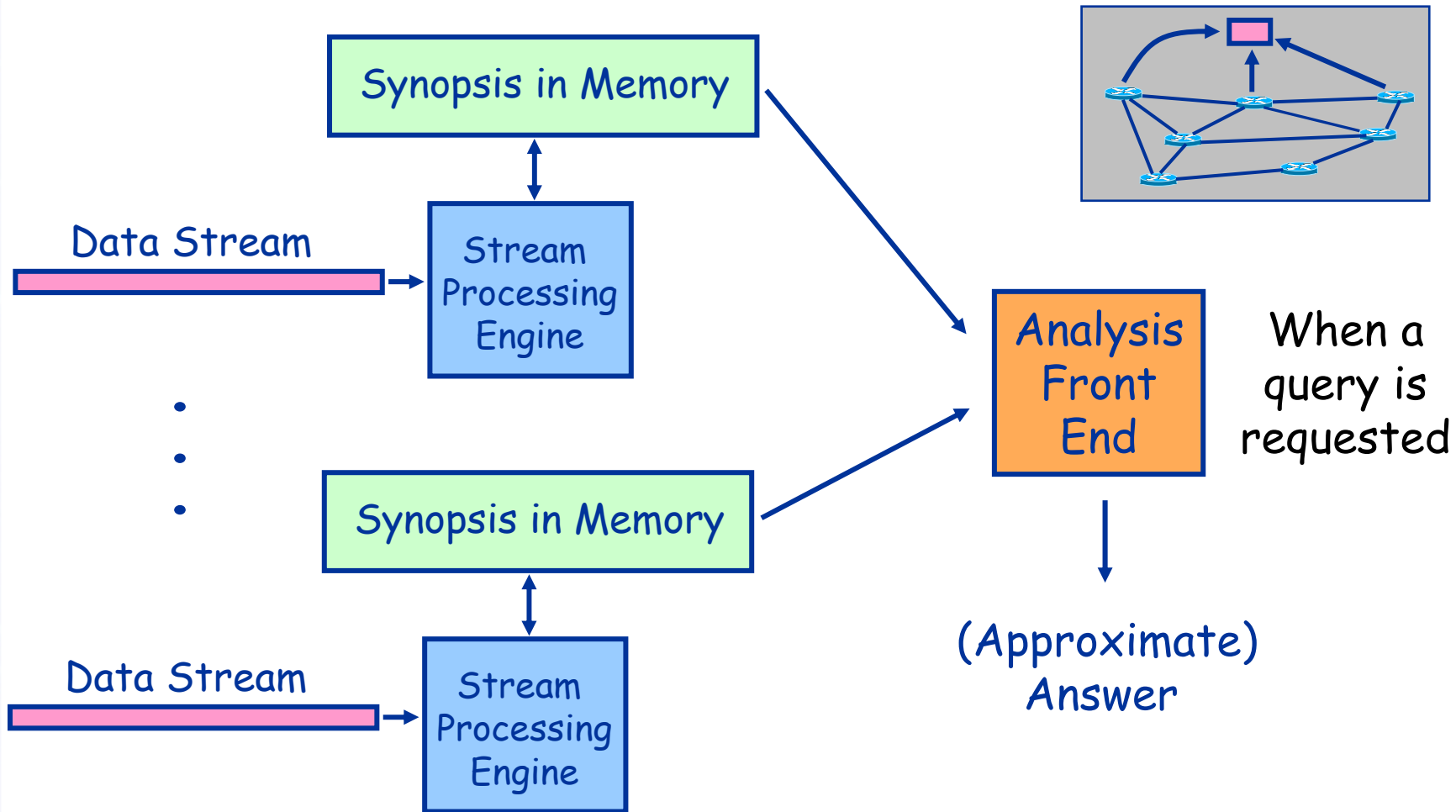
Data Stream Model

- A data stream is a sequence of elements: e_1, \dots, e_n



- Stream processing goals
 - Limited **memory** for storing synopsis, e.g., $O(\sqrt{n})$, $O(\log n)$
 - Fast synopsis **update time** (per element), e.g., $O(1)$
 - Fast **query time**, e.g., $O(\sqrt{n})$, $O(\log n)$

Distributed Data Streams Model



[G, Tirthapura, SPAA'01]

+ Avoids sending streams to Analysis Front End

Adversarial Stream Inputs

- Adversary controls input values and order
 - No distributional assumptions on the inputs
 - Past may not be representative of the future
 - Typically, do know the input domain
- Randomized algorithms
 - Have oracle for uniformly random numbers
 - Would like to minimize the number of oracle calls
 - Adversary does not observe/adapt to the random numbers

Coping With Memory Limitations

- Many queries cannot be answered exactly over streams, due to the memory limitations
 - E.g., see proofs in [Arasu et al, PODS'02]
 - Communication complexity arguments
- However, often a detailed, exact answer over streams is not interesting:
 - Prefer summarized data (aggregates)
 - Prefer to focus only on recent data
 - Suffices to get the leading digits of aggregates correct

=> Keys to staying within the memory limitations

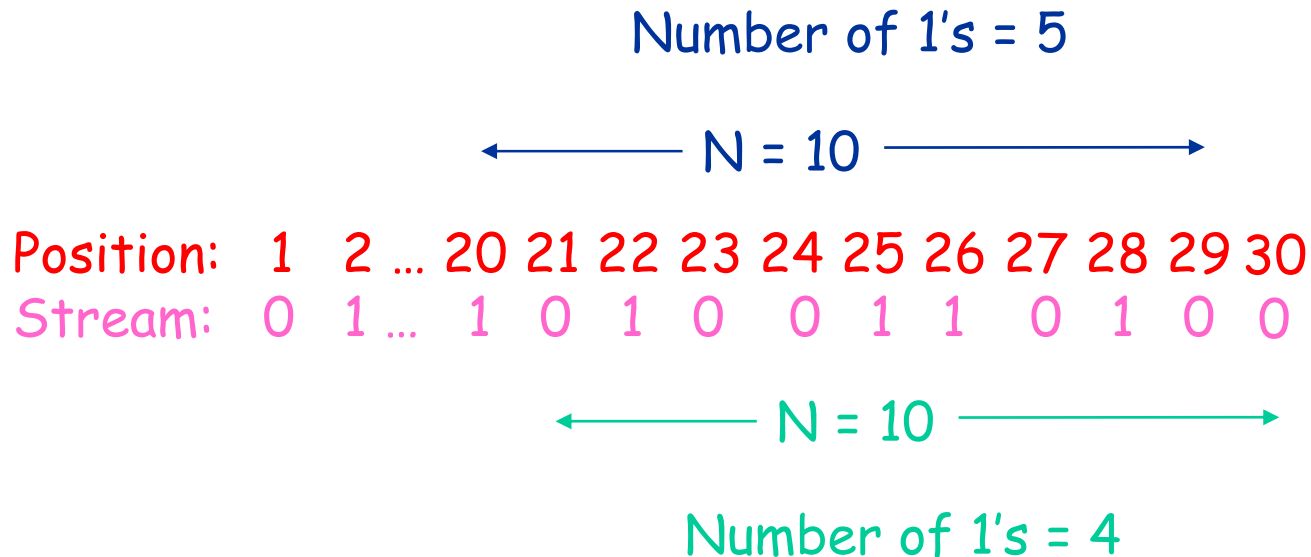
Goal: Approximation Schemes

- **ϵ -approximation scheme:** Given any positive $\epsilon < 1$, compute an estimate with worst case relative error $\leq \epsilon$
- **(ϵ, δ) -approximation scheme:** Given any positive $\epsilon < 1$ and $\delta < 1$, compute an estimate that, with probability $\geq 1 - \delta$, is within relative error $\leq \epsilon$

Nothing for free:
Time and Space bounds depend on ϵ and δ

Sliding Window

- Maintain the aggregate / statistic over a sliding window of the N most recent stream elements
 - Motivation: Only the most recent data is important



[Datar, Gionis, Indyk, Motwani, SODA'02]

Outline

- Data Streams in the Real World
- Formal Model
- **Important Basic Techniques: FM & AMS**
- Distributed Streams Algorithms for Sliding Windows
- Synopsis Diffusion for Sensor Networks
- Further Results & Conclusions

How Many Distinct Values in Stream?

Stream: 3, 1, 4, 1, 5, 9, 2, 6, 5, 4

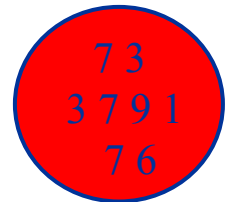
7 distinct values

Number of distinct values may be linear in the length of the stream, so can't afford space for each distinct value

Sampling-based approaches:

1. Collect and store a uniform random sample S of the data
2. At query time, estimate based on a function of the frequency distribution in S

10% sample

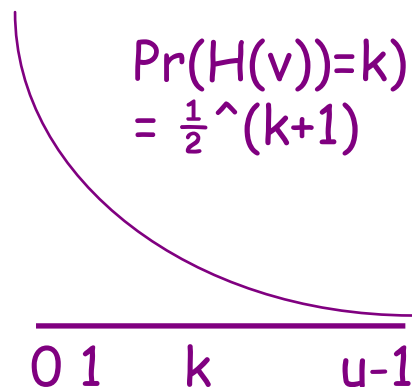


5 distinct?
50 distinct?

[Flajolet, Martin '83], [Alon, Matias, Szegedy '96]:
 $O(\log(1/\delta))$ words for $\epsilon > 1$, can do distributed streams

Flajolet & Martin's 1983 Algorithm

1. Select a hash function H mapping each domain value to a bit position according to an exponential distribution
2. Scan data set, and for each item value v , set bit position $H(v)$
3. Estimate based on the least-significant position j with 0-bit



E.g., 0000101111 $j = 5$ (positions start at 0)

Estimate: $d' = (2^j)/.77351$

Alon, Matias, Szegedy '96/'99

Estimate: 2^r , where r is most-significant position with 1-bit ($r=6$)

+ Explicit construction of H (with only pairwise independence)

+ Stronger guarantees

Performance Bounds of AMS96

1. Select a hash function H mapping each domain value to a bit position according to an exponential distribution
2. Scan data set, and for each attr value v , set bit position $H(v)$
3. Estimate is 2^r , the most-significant position r with 1-bit

Ratio-Error(d') ≤ 5 with probability $> 3/5$

Can improve accuracy by taking averages

Can boost confidence to $1-\delta$ by taking the median of $O(\log(1/\delta))$ estimators (with independent H 's)

Synopsis size: $O((\log u) * (\log(1/\delta)))$, for domain $[1..u]$

Per-element update time: $O(\log(1/\delta))$ finite field operations

Query time: $O(\log(1/\delta))$

AMS Sketching Technique

- [Alon, Matias, Szegedy '96, '99] frequency moments
- **Key Subproblem - Self-Join Size Estimation**
 - Stream of values from $D = \{1, 2, \dots, N\}$
 - Let f_i = frequency of value i
 - Self-join size $S = \sum f_i^2$
 - **Question** - estimating S in small space?

Self-Join Size Estimation

- **AMS Technique (randomized sketches)**
 - **Given** (f_1, f_2, \dots, f_N)
 - $Z_i = \text{random}\{-1, 1\}$
 - $X = \sum f_i Z_i$ (X incrementally computable)
- **Theorem:** $\text{Exp}[X^2] = \sum f_i^2$
 - Cross-terms $f_i Z_i f_j Z_j$ have 0 expectation
 - Square-terms $f_i Z_i f_i Z_i = f_i^2$
- **Space** = $\log(N + \sum f_i)$
- **Independent samples** X_k reduce variance

Sample Run of AMS

frequencies

$$F = \begin{array}{|c|c|c|c|c|} \hline 3 & 11 & 2 & 5 & 7 \\ \hline \end{array}$$

$$Z_1 = \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & -1 & 1 & -1 \\ \hline \end{array}$$

$$Z_2 = \begin{array}{|c|c|c|c|c|} \hline -1 & 1 & -1 & 1 & 1 \\ \hline \end{array}$$

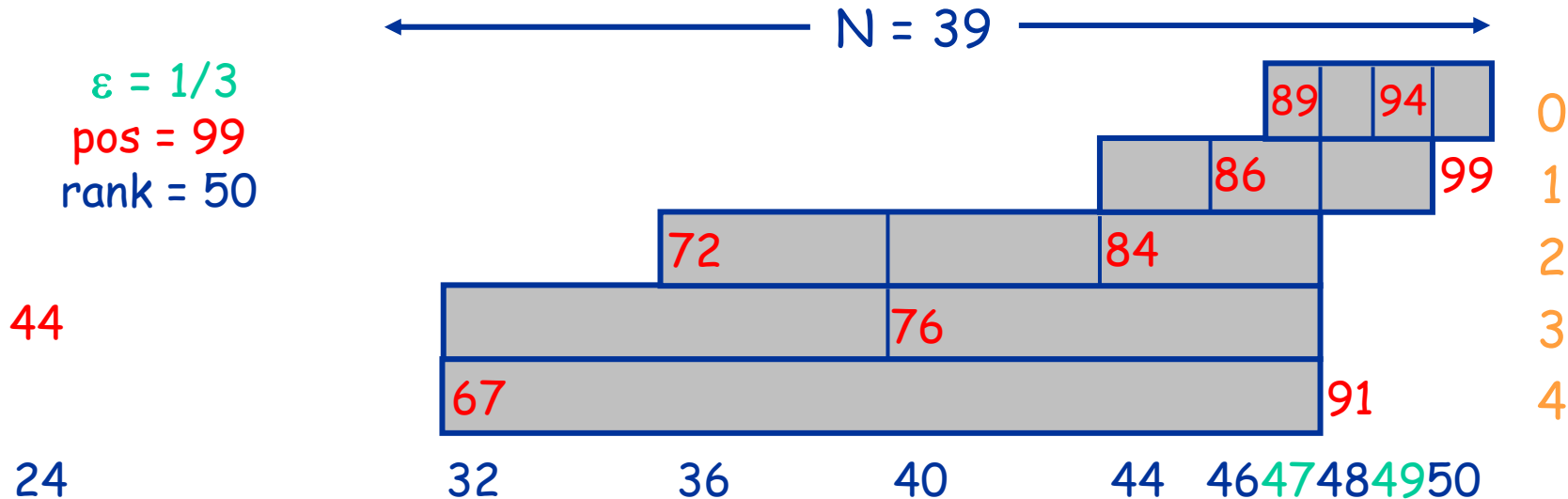
$$\Sigma f_i^2 = 208 \quad X_1 = 10, X_1^2 = 100 \quad X_2 = 18, X_2^2 = 324 \quad \text{Est} = 212$$

"Tug-of-War"

Outline

- Data Streams in the Real World
- Formal Model
- Important Basic Techniques: FM & AMS
- Distributed Streams Algorithms for Sliding Windows
 - Deterministic Wave for Single Stream
 - Randomized Wave for Distributed Streams
- Synopsis Diffusion for Sensor Networks
- Further Results & Conclusions

Improved Wave



- Discard all **positions** in wave outside of window
 - Retain largest discarded **position** (44) & **rank** (24)
- Store each **position** only at its maximum **level**
 - Give $O(1)$ time alg for computing max power of 2 dividing rank
- Maintain **positions** in wave in doubly-linked sorted list

Randomized Wave (for # of 1's)

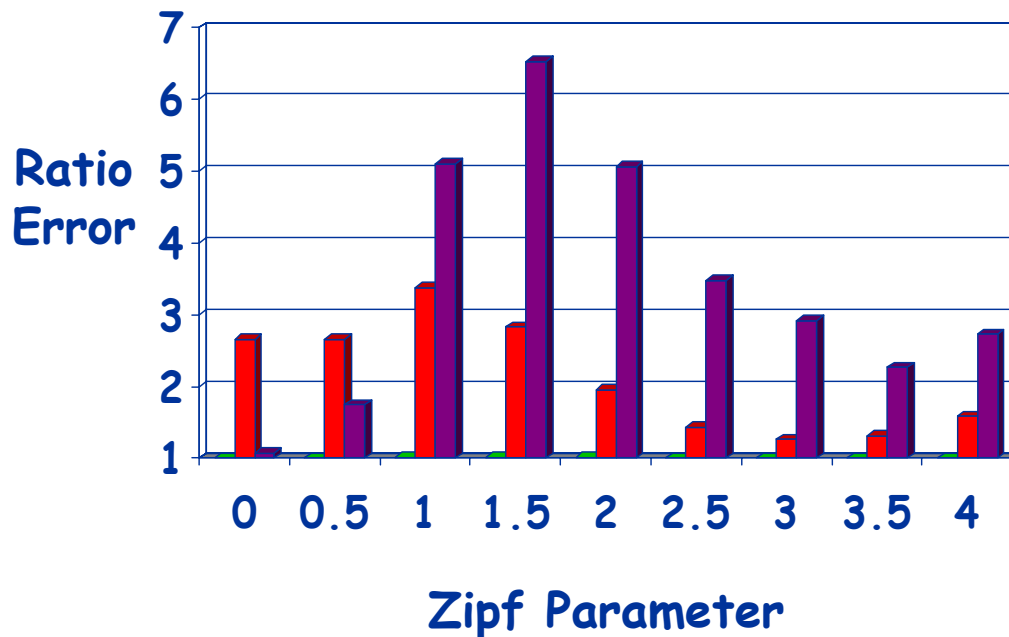
- Each level j contains the most recently selected positions of the 1-bits
- Deterministic Wave: Every (2^j) th 1-bit
- Randomized Wave: Each 1-bit with probability $1/2^j$
- Store $O(1/\epsilon^2)$ per level
- Use pairwise independent, exponentially distributed hash function h --- Store at all levels $O..h(\text{pos})$

Distinct Values, Distributed Streams

- [G, Tirthapura, SPAA'01] $O(\log(1/\delta) / \epsilon^2)$ words per stream, any $\epsilon < 1$, but no sliding window
- Challenge: Accounting for elements that **drop out** of the window
- Idea: Use randomized wave; hash the values (not pos); retain only most recent copy of any selected value; some additional data structures needed
 - Use $O(\log(1/\delta))$ waves with independent hash functions & Take the median of their estimates
 - $O(\log N \log(1/\delta) / \epsilon^2)$ memory words per stream
 - [Datar et al] has $\Omega((\log N)/\epsilon)$ word lower bound

Accuracy vs. Data Skew

Adapted from
[G, VLDB'01]



Data set size = 1M
Sample sizes = 1%

■ Rand. Waves
■ GEE
■ AE

Over the entire range of skew :

- Randomized Wave has 1.00-1.02 ratio error (1.00=no error)
- At least 25 times smaller relative error than best approaches based on uniform sampling (GEE & AE)

Summary of Contributions (I)

- Introduce "Waves" synopsis data structure
- Number of 1's, Sliding window, Single stream
 - First optimal ϵ -approximation scheme
 - $O((\log N)/\epsilon)$ words, $O(1)$ worst case update & query times
- Sum of integers in $[0,R]$, Sliding window, Single stream
 - First optimal ϵ -approximation scheme
 - $O((\log N + \log R)/\epsilon)$ words, $O(1)$ worst case updates & queries

Previous best:

[Datar et al] $O(1)$ amortized, $O(\log N)$ worst case query time

Summary of Contributions (II)

0	1	0	1
1	0	0	1

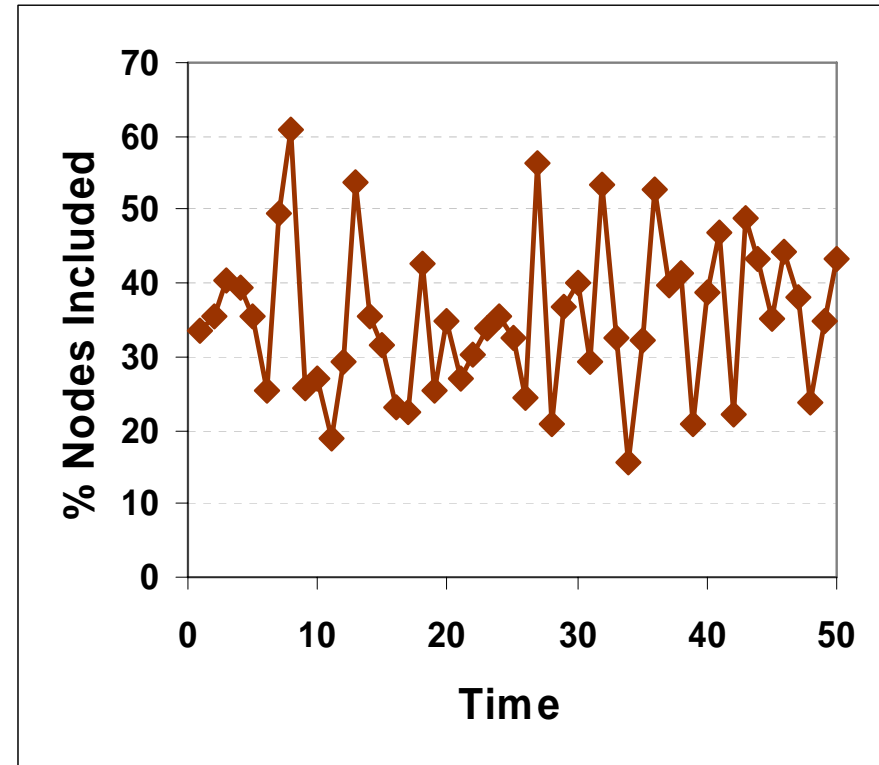
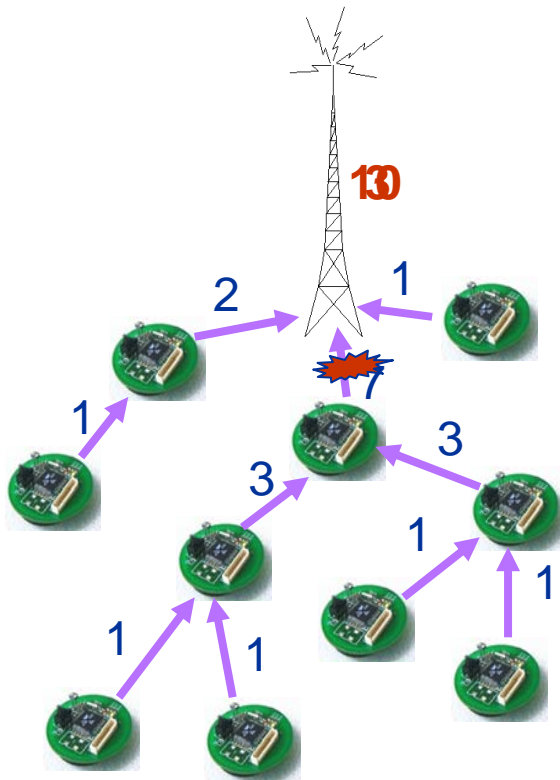
- **Number of 1's, Union of distributed streams**
 - Lower Bound: Any deterministic approx scheme for $\epsilon \leq 1/64$ requires $\Omega(N)$ space, even for 2 streams
 - First randomized (ϵ, δ) -approx scheme for sliding window using only logarithmic memory words per stream
- **Number of distinct values, Distributed streams, Sliding window**
 - First randomized (ϵ, δ) -approx scheme using only logarithmic memory words per stream
 - $O(\log N \log(1/\delta) / \epsilon^2)$ words per stream,
 $O(\log(1/\delta))$ expected update time

Outline

- Data Streams in the Real World
- Formal Model
- Important Basic Techniques: FM & AMS
- Distributed Streams Algorithms for Sliding Windows
- Synopsis Diffusion for Sensor Networks
- Further Results & Conclusions

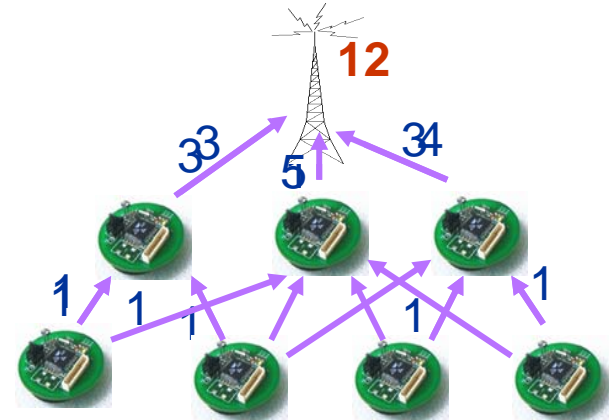
Wireless Sensor Network Aggregation

- Done in-network on a spanning tree
- Problem: Not robust against node- and link-failures



The Problem and the Goal

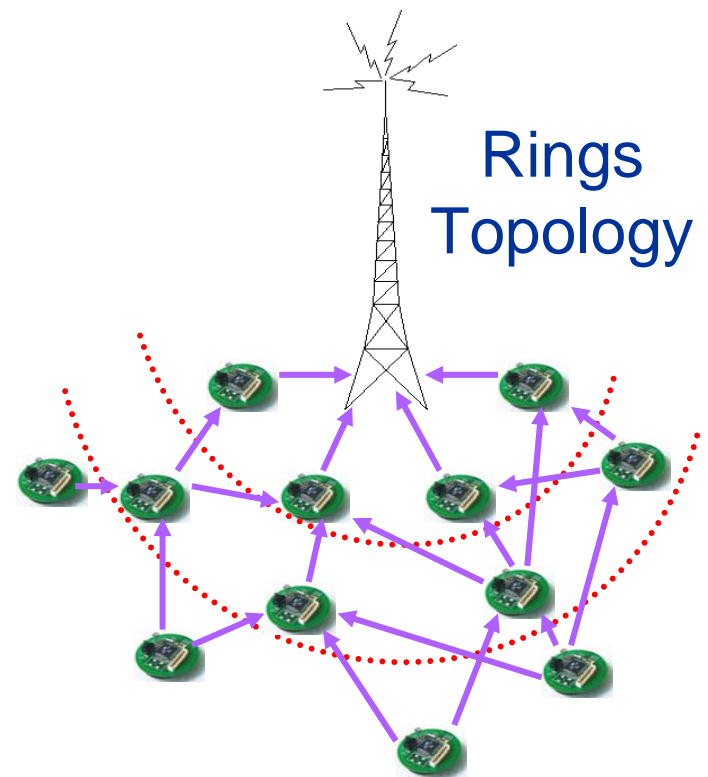
- Tree topology used to avoid **double-counting**
- **Aggregation** and **routing** are tightly coupled



- **Our goal:** decouple the two components
 - They can be independently optimized
 - Robust multi-path routing can be used
 - Can exploit the broadcast medium

Synopsis Diffusion (on Rings)

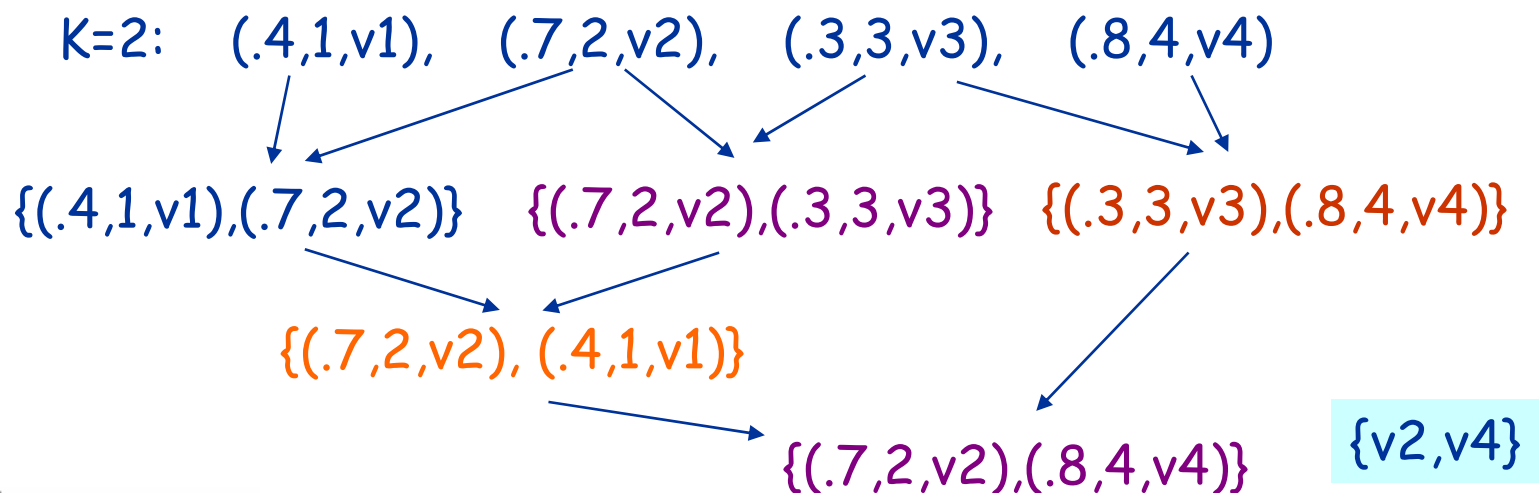
- Each node generates a small **synopsis** of its readings (SG)
- Starting with outer ring, each node **broadcasts** its synopsis
- **Synopsis Fusion (SF)**: Each node in next ring combines all synopses it hears into its own synopsis
- SF must be **order- and duplicate- insensitive (ODI)**



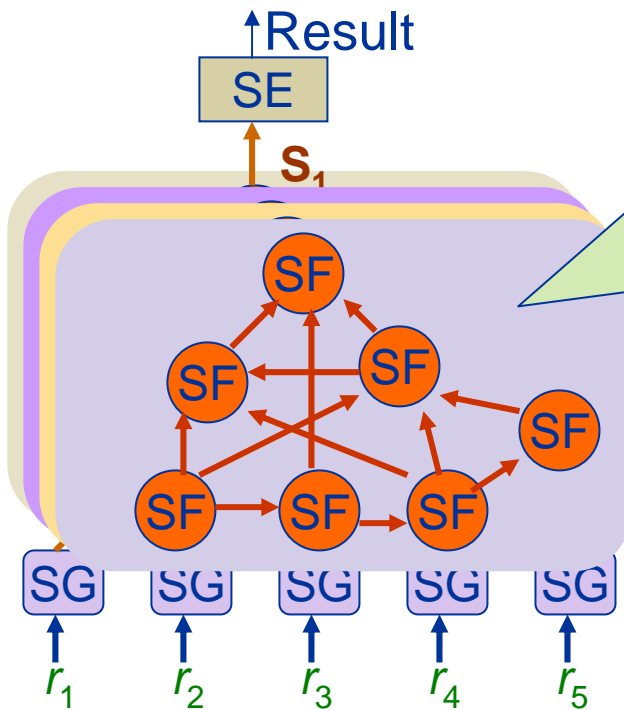
[SenSys'04]

SD Example: Uniform Sample of Size K

- $SG()$: Each node selects a random number r in $[0,1]$, and creates a synopsis (r, id, val)
- $SF(s,s')$: Output the K (r,id,val) pairs from the union of s and s' with maximum r -values
- $SE(s)$: Output the K val 's in s



Key Challenge & A Solution



Aggregation Topology

Potentially large unknown set of combinations!

Key Result:
Give 4 simple, locally testable properties for ODI correctness (necessary & sufficient)

Makes topology independence tractable

ODI Goal: S_1 is always the same

Order- & Duplicate-Insensitive Synopses

- Key Result: We show that the following simply-checked properties are necessary & sufficient
 1. SF is commutative
 2. SF is associative
 3. SF is same-synopsis idempotent: $SF(s,s) = s$
 4. If readings r and r' are duplicates, then $SG(r) = SG(r')$

E.g., suppose use $SF(s_1,s_2) = (s_1+s_2)/2$,
which of P1-P3 fails?

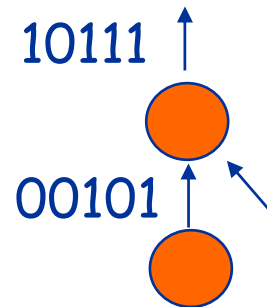
$$P2: SF(2,SF(6,30)) = 10 \quad \text{but} \quad SF(SF(2,6),30) = 17$$

However, our Uniform Sampling SF and SG satisfies P1-P4

Implications

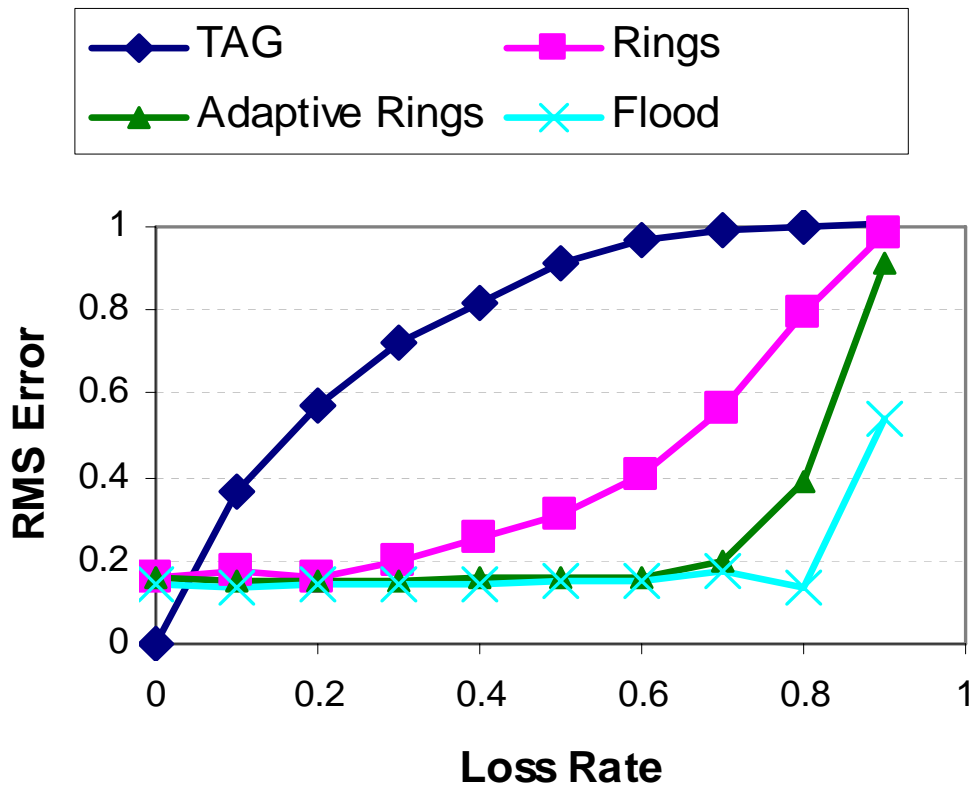
- SF forms a semi-lattice
- Lattice property => can tell if another ODI synopsis has taken my ODI synopsis into account

E.g., SF is bitwise-OR



- => Implicit acks (Listen to what parent sends to know if your message was received)
- => Efficient adaptation to dynamic message loss, even when asymmetric links
- => More robust routing => More accurate answers

Synopsis Diffusion on Rings



More robust than TAG

Scheme	Energy
Tree (TAG)	41.8mj
A. Rings	42.1mj
Flood	685mj

Almost as energy efficient as TAG

Outline

- Data Streams in the Real World
- Formal Model
- Important Basic Techniques: FM & AMS
- Distributed Streams Algorithms for Sliding Windows
- Synopsis Diffusion for Sensor Networks
- Further Results & Conclusions

Time-Decaying Aggregates

Cohen-Strauss
PODS'03

- Random Early Detection (RED)
 - Weighted average of previous queue lengths used to determine what fraction of packets to discard
- Holding-time policies for ATM virtual circuits
 - Time-decaying weighted averages of previous idle times used to determine which circuits to close
- Internet gateway selection products
 - Time-decaying average of previous reliability measurements used in path selection
- Telecom usage patterns

Decay Functions

- Focused on the Decayed Count Problem:
 - $C(T) = \sum_{(t < T \text{ s.t. } v(t) = 1)} g(T - t)$
- Exponential: $g(x) = \exp(-\lambda x)$, $\lambda > 0$
 - $C := v + \exp(-\lambda) * C$
- Sliding window of size W : $g(x) = 1$ for $x \leq W$ and otherwise $g(x) = 0$
- Polynomial: $g(x) = (1/x)^\alpha$
- Also: Polyexponential, Chordal, Polygonal

Cascaded Sliding Windows

- Often, algorithm for a sliding window of size W can produce an estimate for **all windows** of size $\leq W$
- **Linear combination** of its estimates results in an estimate for **ANY decay function** $g(x)$, that is within ϵ relative error

Example: $g(0)=8, g(1)=5, g(2)=3, g(3)=2, \& g(x)=0$ for $x > 3$

$$\begin{aligned} \text{Decayed count is } & 8*v(T) + 5*v(T-1) + 3*v(T-2) + 2*v(T-3) \\ & = (8-5)*v(T) + (5-3)*v[T-1..T] + (3-2)*v[T-2..T] + 2*v[T-3..T] \end{aligned}$$

Plug in sliding window estimate for each $v[t..T]$ above

Counting Samples [G, Matias 98]

- Effective for identifying the most popular items
 - Without keeping track of all items
 - With an adaptive threshold for what's popular

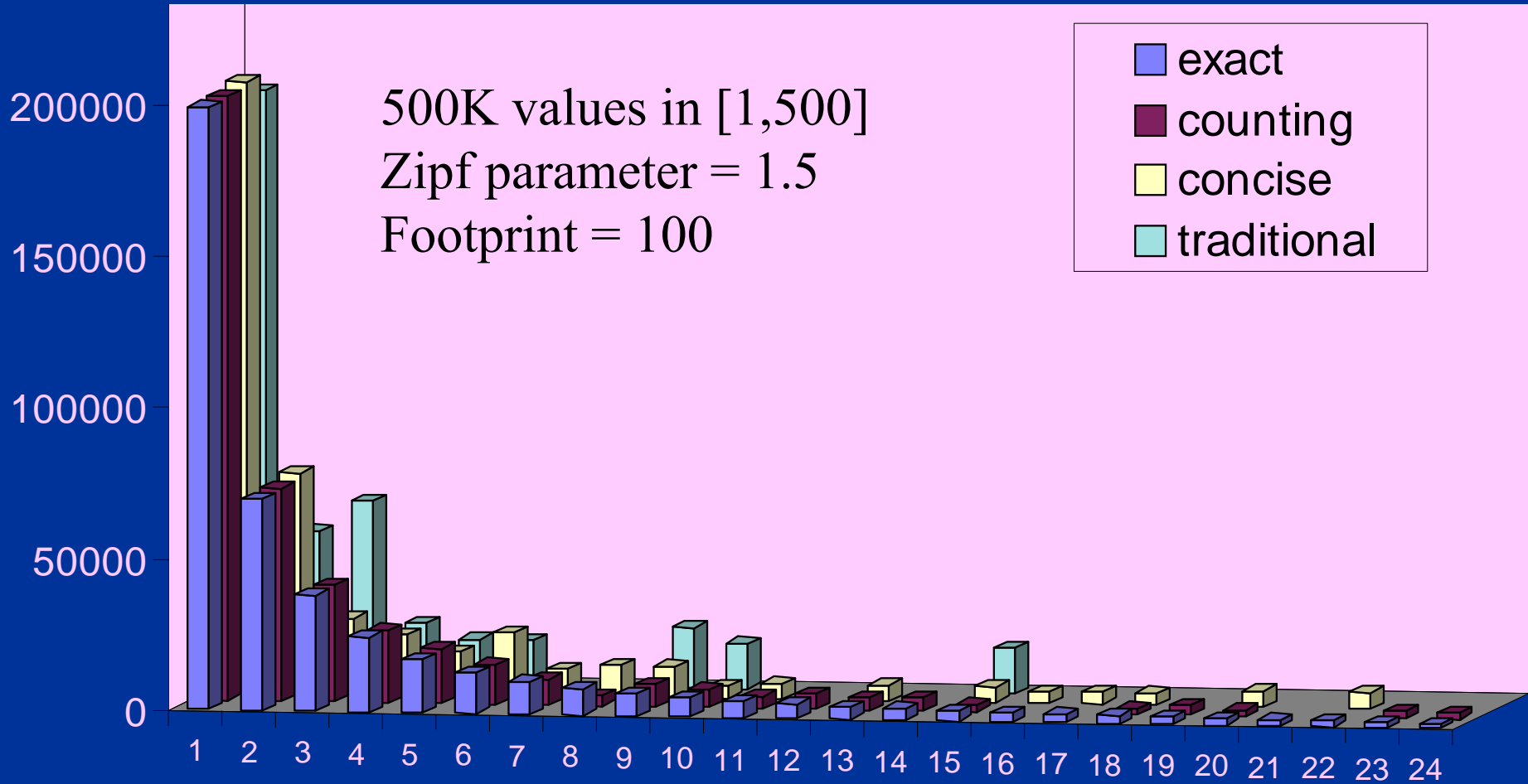
Sample S is a set of $\langle \text{value}, \text{count} \rangle$ pairs

- For each new stream element
 - If element value in S , increment its count
 - Otherwise, add to S with probability $1/T$

Counting Samples [G, Matias 98]

- If size of sample S exceeds M , select new threshold $T' > T$
- For each value (with count C) in S , decrement count in repeated tries until C tries or a try in which count is not decremented
 - First try, decrement count with prob $1 - T/T'$
 - Next tries, decrement count with prob $1 - 1/T'$
- Subject each subsequent stream element to higher threshold T'
- Estimate of frequency for value in S : count in $S + 0.418 * T$

Comparison of Hot List Algorithms



Many New Stream Algorithms:

- **Histograms**
 - Equi-Width Histograms (Quantiles)
 - Most popular items, V-Opt Histograms
 - Wavelets
- **Data Mining**
 - Stream Clustering (e.g. k -medians)
 - Decision Trees
- **Frequency moments, L_p Norms of two streams**
- **Relational DB operators**
 - Join size estimation

Also
Lower
Bounds

Papers in STOC, FOCS, SODA, SIGMOD, VLDB, etc

Questions

