

Data Management Challenges for Big Data Analytics

Minos Garofalakis

**Technical University of Crete, Software Technology
and Network Applications Lab**

MODAP Summer School'2012, Leysin, Switzerland



The future continues to bring new data sources and new volumes of data



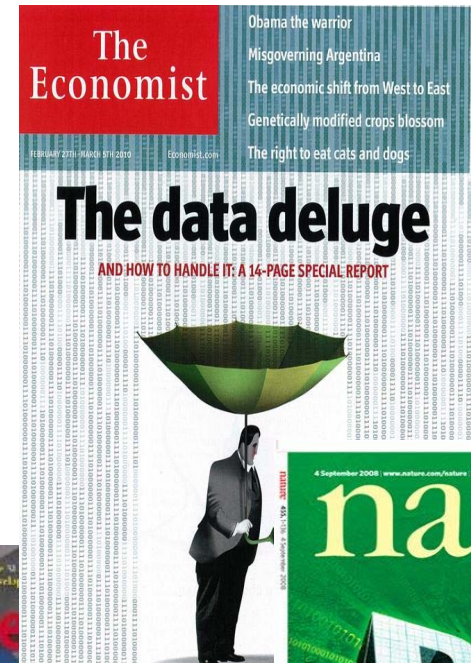
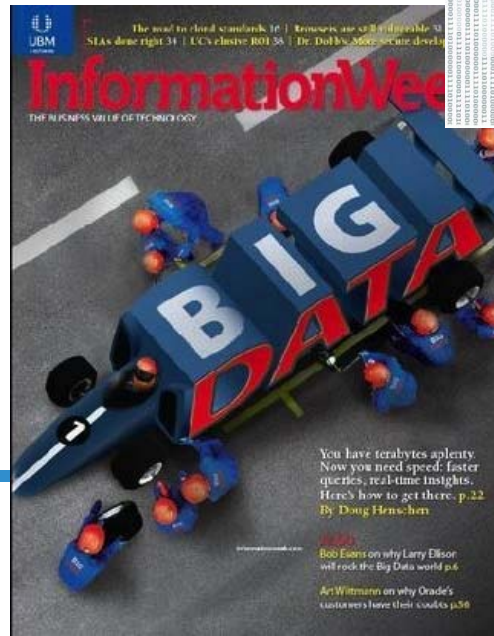
World Data Centre for Climate

- **220 Terabytes** of Web data
- **9 Petabytes** of additional data

**2 Billion** Internet
users by 2011Twitter processes
7 terabytes of
data every dayFacebook
processes
10 terabytes of data
every dayUPS supports largest
known peak database
workload of **1.1 Billion**
SQL statements per hourNew Sources of
InformationTraditional Sources of
Information**4.6 Billion**
Mobile Phones
World Wide**74 Terabytes**
UK Land Registry -
Largest known transaction
processing DB

Big Data is Big News (and Big Business...)

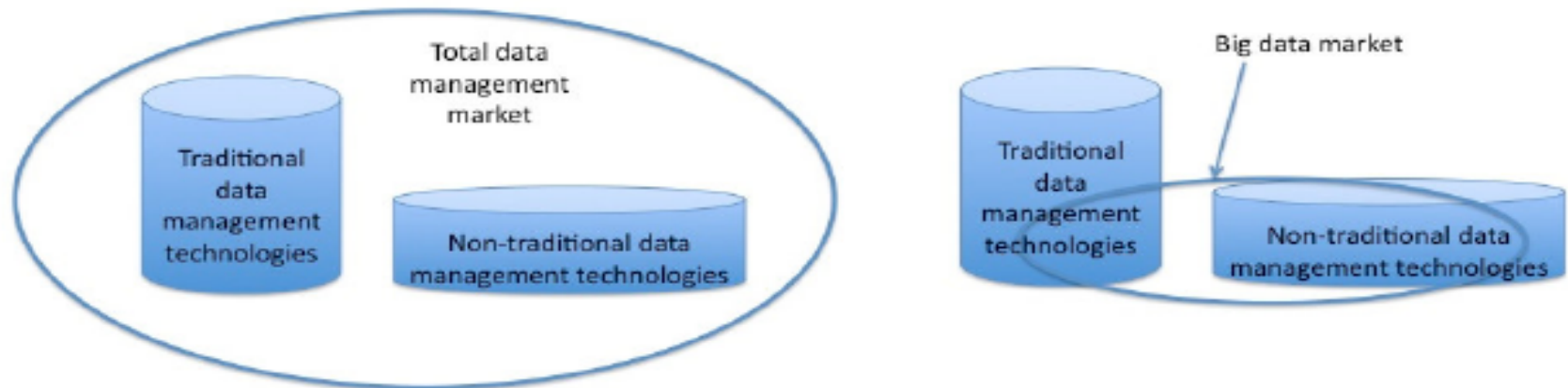
- Rapid growth due to several information-generating technologies, such as mobile computing, sensor networks, and social networks
- How can we cost-effectively manage and analyze all this data...?



What is “Big Data”? - 451 Group - Definition

“Big data is a term applied to data sets that are large, complex and dynamic (or a combination thereof) and for which there is a requirement to capture, manage and process the data set in its entirety, such that it is not possible to process the data using traditional software tools and analytic techniques within tolerable time frames .”

451 Group, Sizing the Big Data Problem



Big Data Challenges: The Four V's (and one D)...

- **Volume:** Scaling from Terabytes to Exa/Zettabytes
- **Velocity:** Processing massive amounts of streaming data
- **Variety:** Managing the complexity of multiple relational and non-relational data types and schemas
- **Veracity:** Handling the inherent uncertainty and noise in the data
- **Distribution:** Dealing with massively distributed information
- *Our focus here: Volume, Velocity, Distribution...*





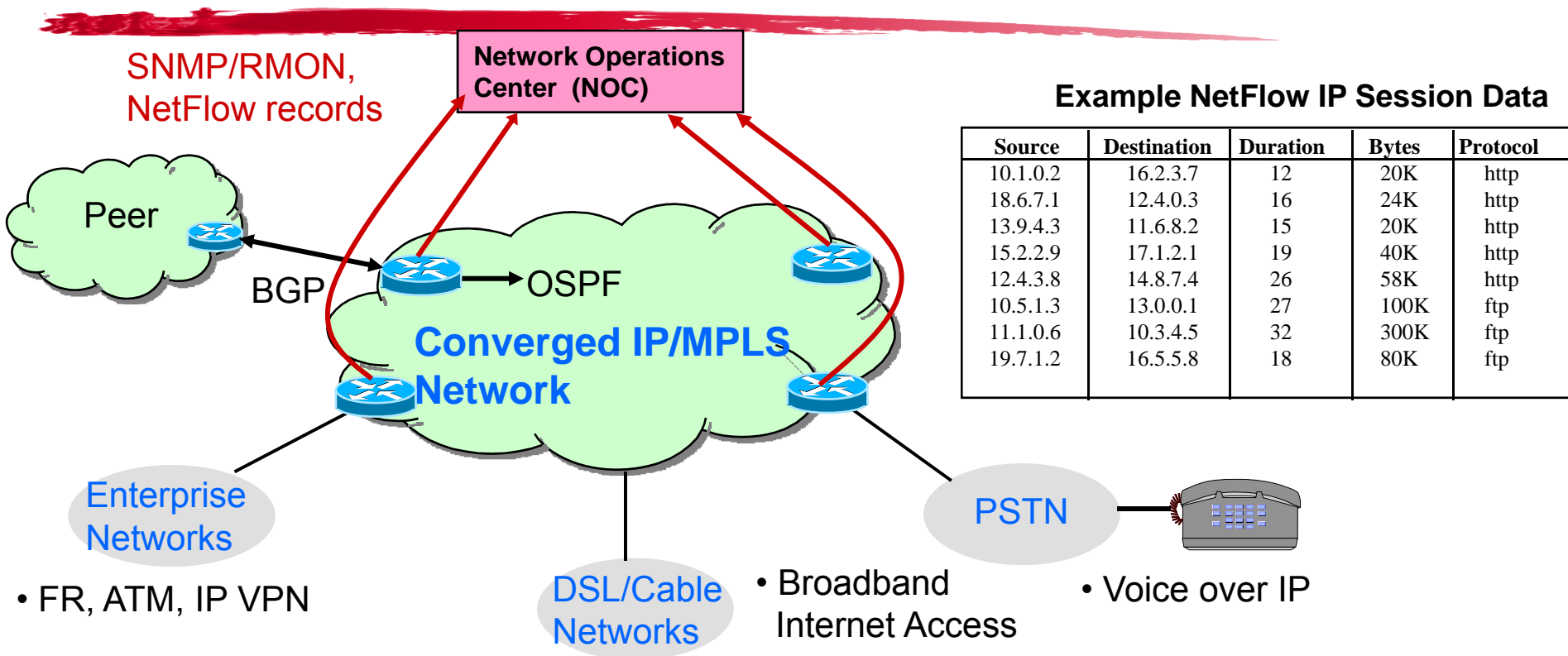
Part I: Centralized Data Streaming

Data-Stream Management



- **Traditional DBMS** - data stored in **finite, persistent data sets**
- **Data Streams** - distributed, continuous, unbounded, rapid, time varying, noisy, . . .
- **Data-Stream Management** - variety of modern applications
 - Network monitoring and traffic engineering
 - Telecom call-detail records
 - Network security
 - Financial applications
 - Sensor networks
 - Manufacturing processes
 - Web logs and clickstreams
 - Massive data sets

Networks Generate Massive Data Streams



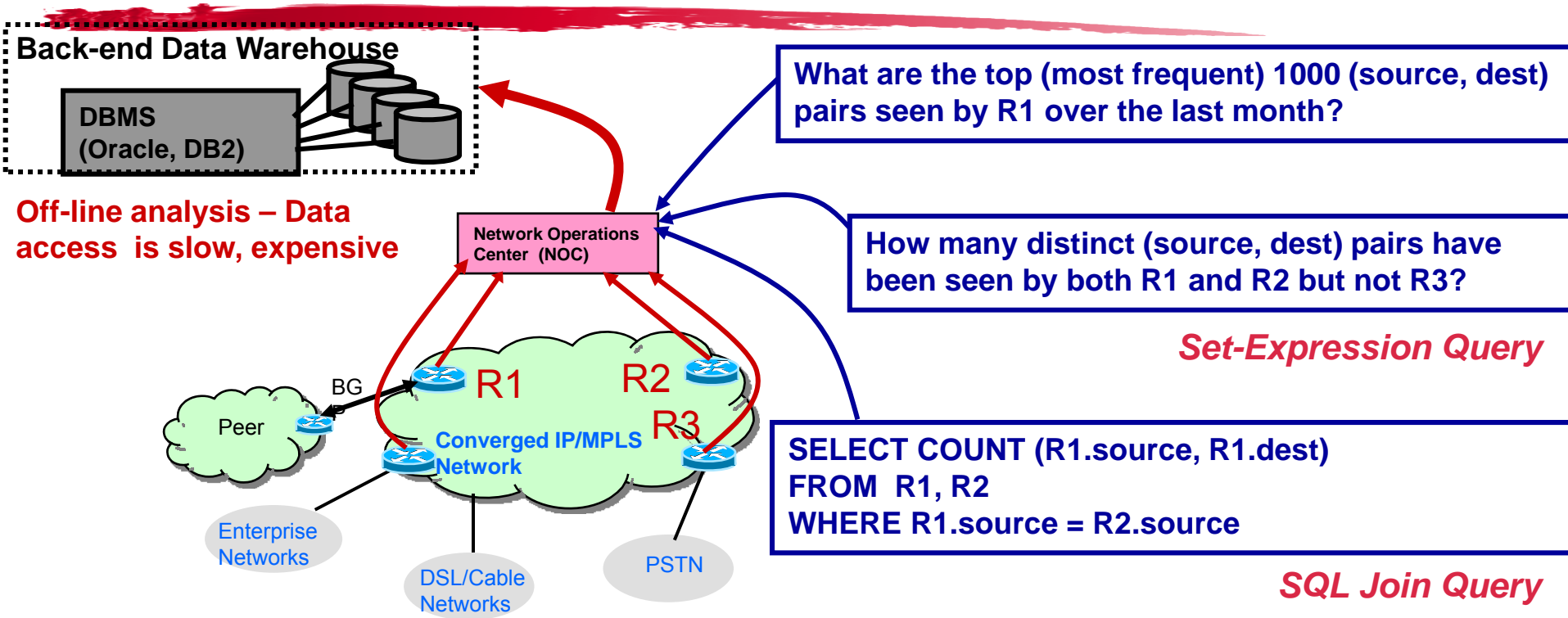
- SNMP/RMON/NetFlow data records arrive 24x7 from different parts of the network
- Truly massive streams arriving at rapid rates
 - AT&T collects **> 1 Terabyte** of NetFlow data each day!
- Typically shipped to a back-end data warehouse (off site) for off-line analysis

Packet-Level Data Streams



- Single 2Gb/sec link; say avg packet size is 50bytes
- Number of packets/sec = 5 million
- Time per packet = 0.2 microsec
- If we only capture **header information** per packet:
src/dest IP, time, no. of bytes, etc. - at least 10bytes.
 - Space per second is 50Mb
 - Space per day is 4.5Tb per link
 - ISPs typically have hundreds of links!
- Analyzing **packet content streams** - whole different ballgame!!

Real-Time Data-Stream Analysis



- Need ability to process/analyze network-data streams *in real-time*
 - As records stream in: look at records *only once in arrival order!*
 - Within resource (CPU, memory) limitations of the NOC
- Critical to important NM tasks
 - Detect and react to Fraud, Denial-of-Service attacks, SLA violations
 - Real-time traffic engineering to improve load-balancing and utilization

IP Network Data Processing



- 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?
- 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

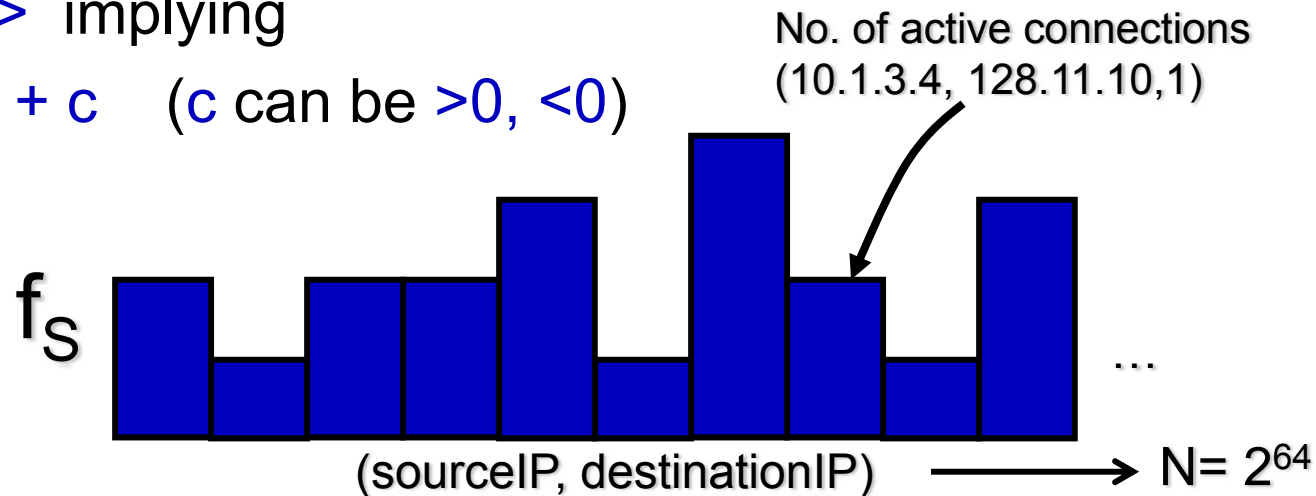
Outline – Part I



- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
 - Sampling
 - Linear-Projection (aka AMS) Sketches
 - *Applications:* Join/Multi-Join Queries, Wavelets
 - Hash (aka FM) Sketches
 - *Applications:* Distinct Values, Set Expressions
 - Another Example Sketch for Multisets: CountMin

Model of a Relational Stream

- Relation “signal”: Large array $f_s[1...N]$ with values $f_s[i]$ all initially zero
 - Frequency-distribution array of S
 - Multi-dimensional arrays as well (e.g., row-major)
- Relation implicitly rendered via a *stream of updates*
 - Update $\langle x, c \rangle$ implying
 - $f_s[x] := f_s[x] + c$ (c can be >0 , <0)



- Goal: Compute queries (functions) on such dynamic vectors in “small” space and time ($\ll N$)

Example IP Network Signals

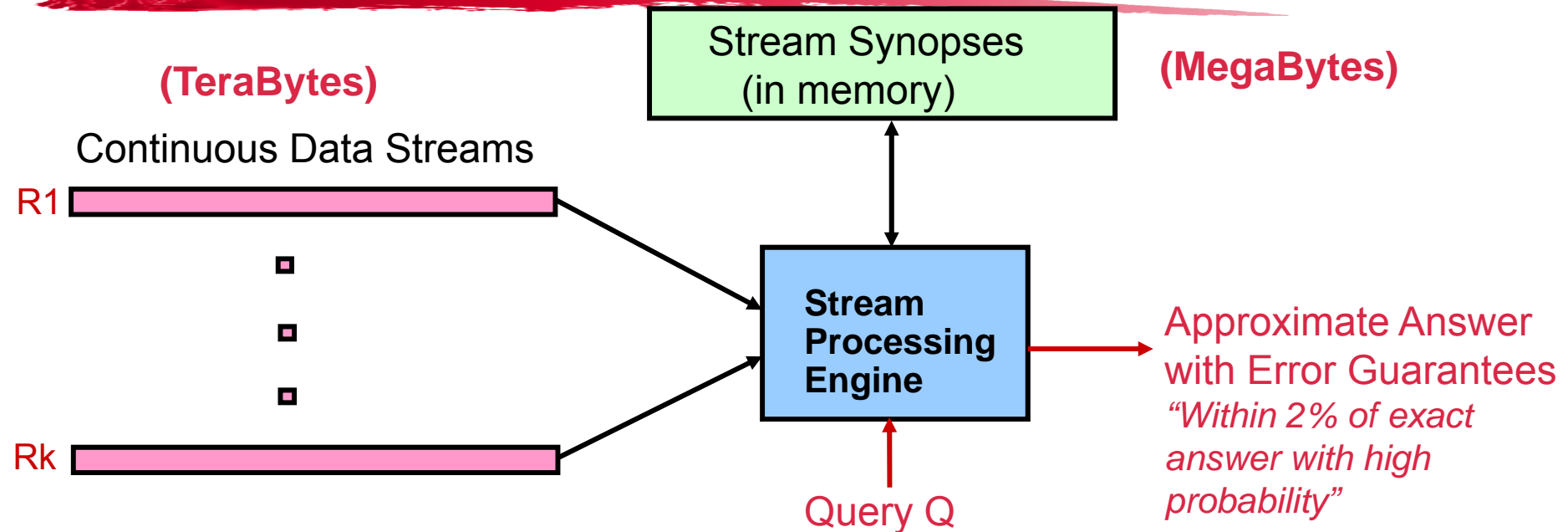


- Number of bytes (packets) sent by a source IP address during the day
 - 2^{32} sized one-d array; increment only
- Number of flows between a source-IP, destination-IP address pair during the day
 - 2^{64} sized two-d array; increment only, aggregate packets into flows
- Number of **active** flows per source-IP address
 - 2^{32} sized one-d array; increment and decrement

Streaming Model: Special Cases

- **Time-Series Model**
 - Only j -th update updates $f[j]$ (i.e., $f[j] := c[j]$)
- **Cash-Register Model**
 - $c[j]$ is always ≥ 0 (i.e., increment-only)
 - Typically, $c[j]=1$, so we see a multi-set of items in one pass
- **Turnstile Model**
 - Most general streaming model
 - $c[j]$ can be >0 or <0 (i.e., increment or decrement)
- *Problem difficulty varies depending on the model*
 - E.g., MIN/MAX in Time-Series vs. Turnstile!

Data-Stream Processing Model



- Approximate answers often suffice, e.g., trend analysis, anomaly detection
- Requirements for stream synopses
 - *Single Pass*: Each record is examined at most once, in (fixed) arrival order
 - *Small Space*: Log or polylog in data stream size
 - *Real-time*: Per-record processing time (to maintain synopses) must be low
 - *Delete-Proof*: Can handle record deletions as well as insertions
 - *Composable*: Built in a *distributed fashion* and combined later

Data Stream Processing Algorithms

- Generally, algorithms compute approximate answers
 - Provably difficult to compute answers accurately with limited memory
- Approximate answers - Deterministic bounds
 - Algorithms only compute an approximate answer, but bounds on error
- Approximate answers - Probabilistic bounds
 - Algorithms compute an approximate answer with high probability
 - With probability at least $1 - \delta$, the computed answer is within a factor ε of the actual answer
- Single-pass algorithms for processing streams also applicable to (massive) databases!

Probabilistic Guarantees

- Example: Actual answer is within 5 ± 1 with prob ≥ 0.9
- **Randomized algorithms:** Answer returned is a specially-built random variable
- User-tunable **(ϵ, δ) -approximations**
 - Estimate is within a relative error of ϵ with probability $\geq 1 - \delta$
- Use **Tail Inequalities** to give probabilistic bounds on returned answer
 - **Markov Inequality**
 - **Chebyshev's Inequality**
 - **Chernoff Bound**
 - **Hoeffding Bound**

Overview – Part I



- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
 - Sampling
 - Linear-Projection (aka AMS) Sketches
 - *Applications:* Join/Multi-Join Queries, Wavelets
 - Hash (aka FM) Sketches
 - *Applications:* Distinct Values, Set Expressions
 - Another Example Sketch for Multisets: CountMin

Sampling: Basics

- Idea: A small random sample S of the data often well-represents all the data

- For a fast approx answer, apply “modified” query to S

- Example: select agg from R where $R.e$ is odd

Data stream:

9	3	5	2	7	1	6	5	8	4	9	1
---	---	---	---	---	---	---	---	---	---	---	---

 ($n=12$)

Sample S :

9	5	1	8
---	---	---	---

- If agg is **avg**, return average of odd elements in S

answer: 5

- If agg is **count**, return average over all elements e in S of

- n if e is odd

answer: $12 \cdot 3/4 = 9$
--

- 0 if e is even

Unbiased: For expressions involving count, sum, avg: the estimator is **unbiased**, i.e., the expected value of the answer **is** the actual answer

Computing Stream Sample

- Reservoir Sampling [Vit85]: Maintains a sample S of a fixed-size M
 - Add each new element to S with probability M/n , where n is the current number of stream elements
 - If add an element, evict a random element from S
 - Instead of flipping a coin for each element, determine the number of elements to skip before the next to be added to S
- Concise sampling [GM98]: Duplicates in sample S stored as $\langle \text{value}, \text{count} \rangle$ pairs (thus, potentially boosting actual sample size)
 - Add each new element to S with probability $1/T$ (simply increment count if element already in S)
 - If sample size exceeds M
 - Select new threshold $T' > T$
 - Evict each element (decrement count) from S with probability $1 - T/T'$
 - Add subsequent elements to S with probability $1/T'$

Outline – Part I



- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
 - Sampling
 - Linear-Projection (aka AMS) Sketches
 - *Applications: Join/Multi-Join Queries, Wavelets*
 - Hash (aka FM) Sketches
 - *Applications: Distinct Values, Set Expressions*
 - Another Example Sketch for Multisets: CountMin

Synopses for Relational Streams



- Conventional data summaries fall short
 - Quantiles and 1-d histograms [MRL98,99], [GK01], [GKMS02]
 - Cannot capture attribute correlations
 - Little support for approximation guarantees
 - Samples (e.g., using Reservoir Sampling)
 - Perform poorly for joins [AGMS99] or distinct values [CCMN00]
 - Cannot handle deletion of records
 - Multi-d histograms/wavelets
 - Construction requires multiple passes over the data
- Different approach: *Pseudo-random sketch synopses*
 - Only logarithmic space
 - *Probabilistic guarantees* on the quality of the approximate answer
 - Support insertion as well as deletion of records (*i.e.*, *Turnstile model*)

Linear-Projection (aka AMS) Sketch Synopses

- Goal: Build small-space summary for distribution vector $f(i)$ ($i=1, \dots, N$) seen as a stream of i -values



- Basic Construct: *Randomized Linear Projection of $f()$* = project onto inner/dot product of f -vector

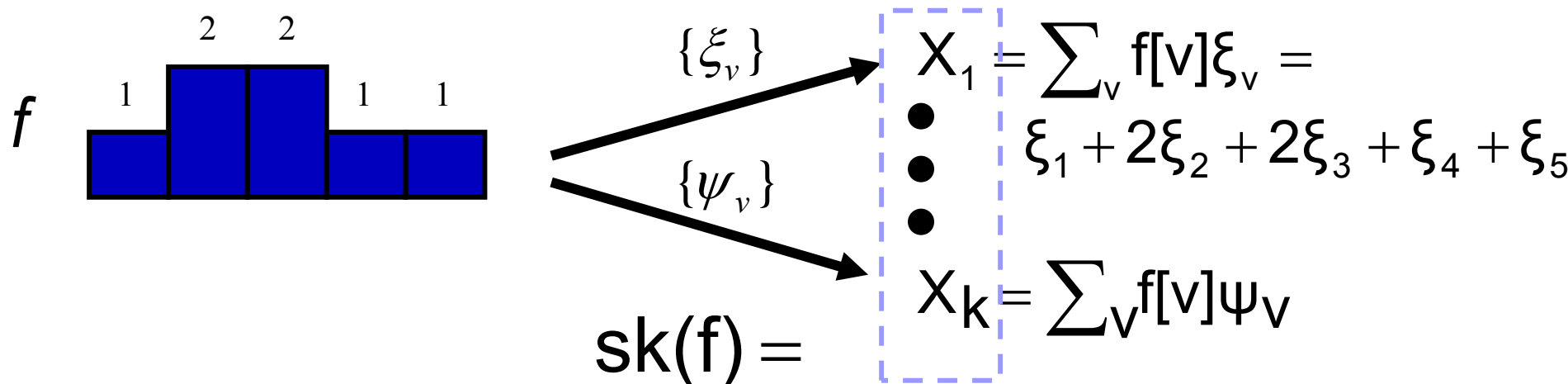
$$\langle f, \xi \rangle = \sum f(i) \xi_i \quad \text{where } \xi = \text{vector of random values from an appropriate distribution}$$

- Simple to compute over the stream: Add ξ_i whenever the i -th value is seen

Data stream: 3, 1, 2, 4, 2, 3, 5, ... $\longrightarrow \xi_1 + 2\xi_2 + 2\xi_3 + \xi_4 + \xi_5$

- Generate ξ_i 's in small ($\log N$) space using pseudo-random generators
- *Tunable probabilistic guarantees* on approximation error
- *Delete-Proof:* Just subtract ξ_i to delete an i -th value occurrence
- *Composable:* Simply *add* independently-built projections

AMS Sketching 101



- Simple randomized linear projections of data distribution
 - Easily computed over stream using logarithmic space
 - *Linear*: Compose through simple addition

- *Theorem[AGMS]*: Given sketches of size $k = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$

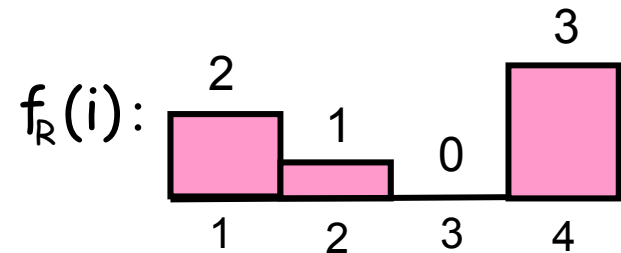
$$\text{sk}(f_R) \cdot \text{sk}(f_S) \in f_R \cdot f_S \pm \epsilon \|f_R\|_2 \|f_S\|_2$$

Example: Binary-Join COUNT Query

- Problem: Compute answer for the query $\text{COUNT}(R \bowtie_A S)$
- Example:

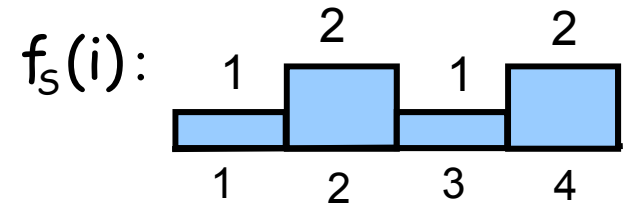
Data stream R.A:

4	1	2	4	1	4
---	---	---	---	---	---



Data stream S.A:

3	1	2	4	2	4
---	---	---	---	---	---



$$\begin{aligned}\text{COUNT}(R \bowtie_A S) &= \sum_i f_R(i) \cdot f_S(i) \\ &= 10 \quad (2 + 2 + 0 + 6)\end{aligned}$$

- Exact solution: too expensive, requires $O(N)$ space!
 - $N = \text{sizeof}(\text{domain}(A))$

Basic AMS Sketching Technique [AMS96]

- Key Intuition: Use randomized linear projections of $f()$ to define random variable X such that

- X is easily computed over the stream (in small space)

- $E[X] = \text{COUNT}(R \bowtie_A S)$

- $\text{Var}[X]$ is small



Probabilistic error guarantees

(e.g., actual answer is 10 ± 1 with probability 0.9)

- Basic Idea:

- Define a family of 4-wise independent $\{-1, +1\}$ random variables

$$\{\xi_i : i = 1, \dots, N\}$$

- $\Pr[\xi_i = +1] = \Pr[\xi_i = -1] = 1/2$

- Expected value of each ξ_i , $E[\xi_i] = 0$

- Variables ξ_i are 4-wise independent

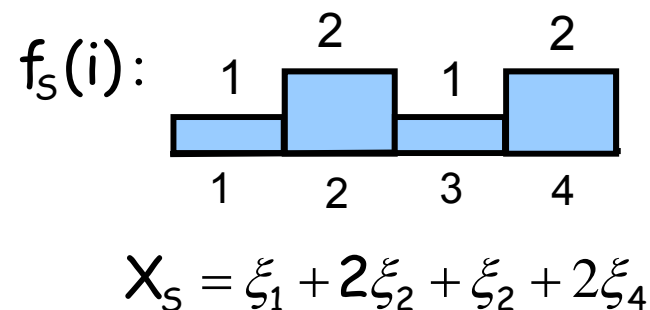
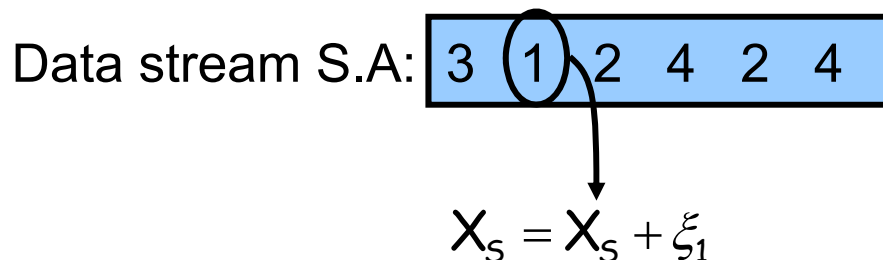
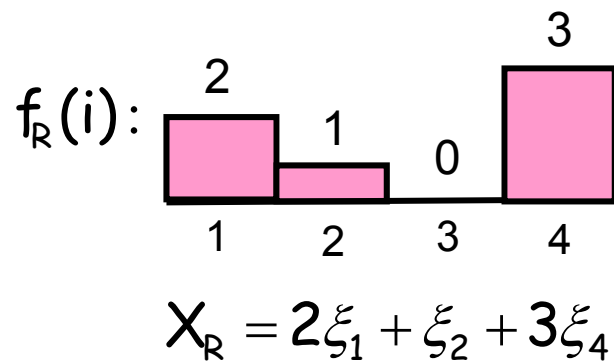
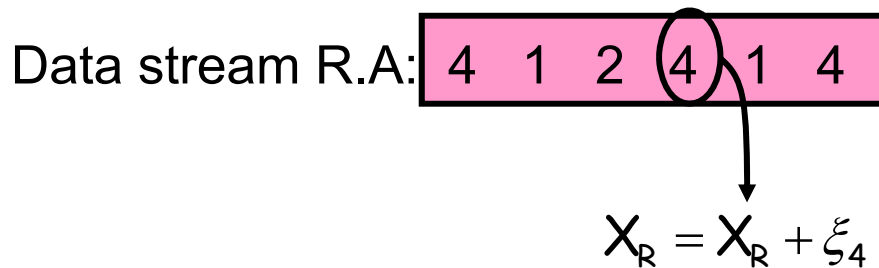
- Expected value of product of 4 distinct $\xi_i = 0$

- Variables ξ_i can be generated using pseudo-random generator using only $O(\log N)$ space (for seeding)!

AMS Sketch Construction

- Compute random variables: $X_R = \sum_i f_R(i) \xi_i$ and $X_S = \sum_i f_S(i) \xi_i$
 - Simply add ξ_i to $X_R(X_S)$ whenever the i -th value is observed in the R.A (S.A) stream
- Define $X = X_R X_S$ to be estimate of COUNT query

- Example:



Binary-Join AMS Sketching Analysis

- Expected value of $X = \text{COUNT}(R \bowtie_A S)$

$$E[X] = E[X_R \cdot X_S]$$

$$= E\left[\sum_i f_R(i) \xi_i \cdot \sum_i f_S(i) \xi_i\right]$$

$$= E\left[\sum_i f_R(i) \cdot f_S(i) \xi_i^2\right] + E\left[\sum_{i \neq i'} f_R(i) \cdot f_S(i') \xi_i \xi_{i'}\right]$$

$$= \sum_i f_R(i) \cdot f_S(i)$$

1

0

- Using 4-wise independence, possible to show that

$$\text{Var}[X] \leq 2 \cdot \text{SJ}(R) \cdot \text{SJ}(S)$$


- $\text{SJ}(R) = \sum_i f_R(i)^2$ is self-join size of R

Boosting Accuracy

- Chebyshev's Inequality:

$$\Pr(|X - E[X]| \geq \varepsilon E[X]) \leq \frac{\text{Var}[X]}{\varepsilon^2 E[X]^2}$$

- Boost accuracy to ε by averaging over several independent copies of X (reduces variance)



$$s = \frac{8 \cdot (2 \cdot \text{SJ}(R) \cdot \text{SJ}(S))}{\varepsilon^2 \text{COUNT}^2}$$

copies

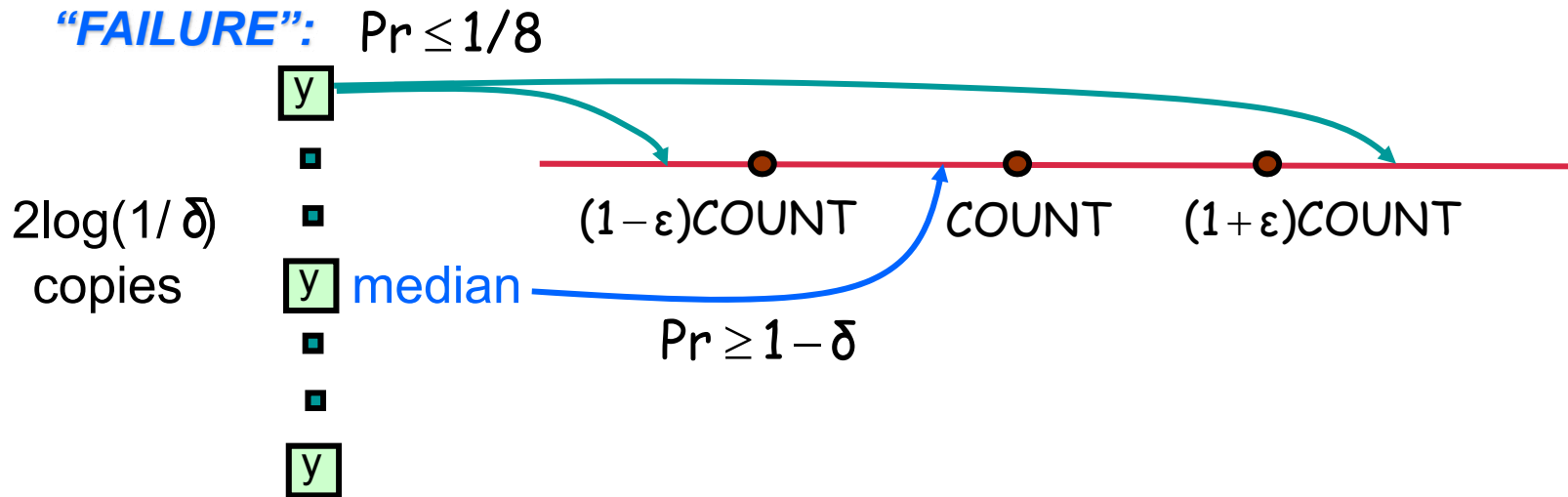
$$E[Y] = E[X] = \text{COUNT}(R \bowtie S)$$

- By Chebyshev: $\text{Var}[Y] = \frac{\text{Var}[X]}{s} \leq \frac{\varepsilon^2 \text{COUNT}^2}{8}$

$$\Pr(|Y - \text{COUNT}| \geq \varepsilon \cdot \text{COUNT}) \leq \frac{\text{Var}[Y]}{\varepsilon^2 \text{COUNT}^2} \leq \frac{1}{8}$$

Boosting Confidence

- Boost confidence to $1 - \delta$ by taking median of $2\log(1/\delta)$ independent copies of Y
- Each $Y = \text{Bernoulli Trial}$



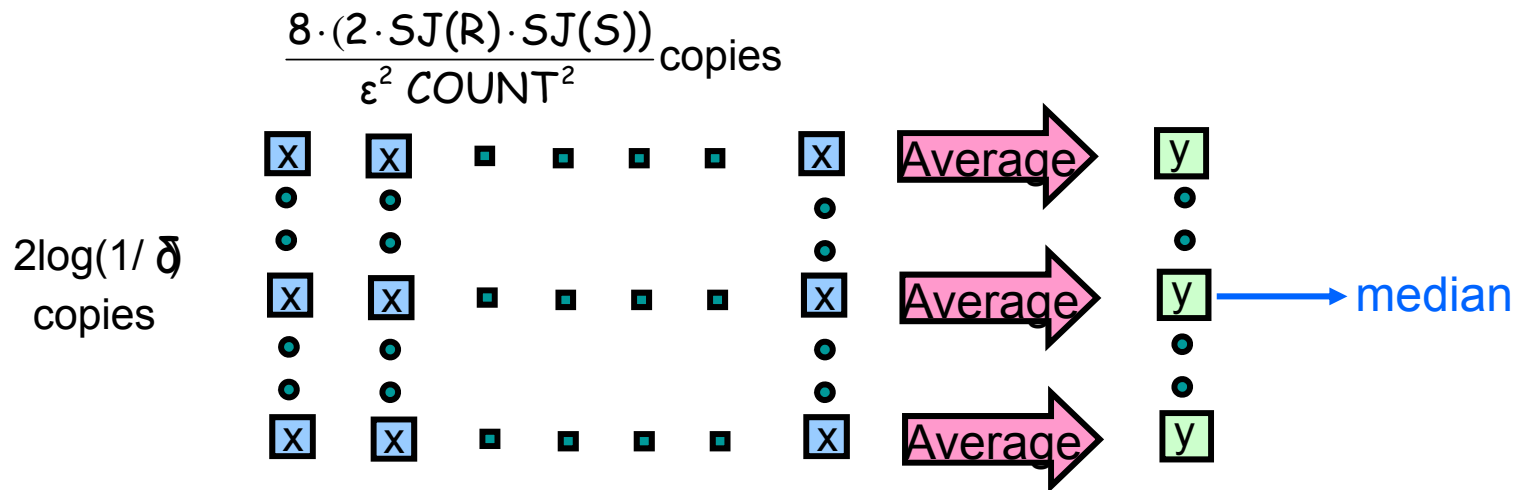
$$\Pr[|\text{median}(Y) - \text{COUNT}| \geq \epsilon \cdot \text{COUNT}]$$

$$= \Pr[\text{\# failures in } 2\log(1/\delta) \text{ trials} \geq \log(1/\delta)]$$

$$\leq \delta \quad (\text{By Chernoff Bound})$$

Summary of Binary-Join AMS Sketching

- Step 1: Compute random variables: $X_R = \sum_i f_R(i) \xi_i$ and $X_S = \sum_i f_S(i) \xi_i$
- Step 2: Define $X = X_R X_S$
- Steps 3 & 4: Average independent copies of X ; Return median of averages



- Main Theorem (AGMS99): Sketching approximates COUNT to within a relative error of ϵ with probability $\geq 1 - \delta$ using space

$$O\left(\frac{SJ(R) \cdot SJ(S) \cdot \log(1/\delta) \cdot \log N}{\epsilon^2 \text{COUNT}^2}\right)$$

- Remember: $O(\log N)$ space for "seeding" the construction of each X

A Special Case: Self-join Size

- Estimate $\text{COUNT}(R \bowtie_A R) = \sum_i f_R^2(i)$ *(original AMS paper)*
- Second (L2) moment of data distribution, Gini index of heterogeneity, measure of skew in the data

In this case, $\text{COUNT} = \text{SJ}(R)$, so we get an (ϵ, δ) -estimate using space only

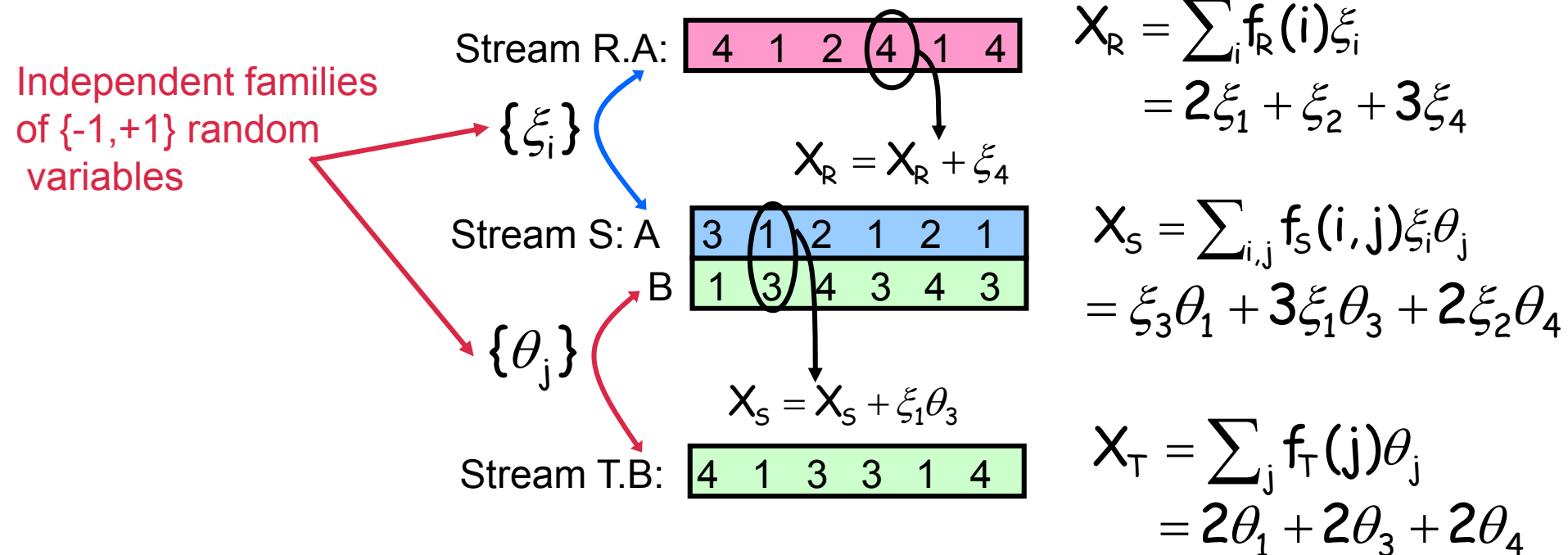
$$O\left(\frac{\log(1/\delta) \cdot \log N}{\epsilon^2}\right)$$

Best-case for AMS streaming join-size estimation

What's the worst case??

AMS Sketching for Multi-Join Aggregates [DGGR02]

- Problem: Compute answer for $\text{COUNT}(R \bowtie_A S \bowtie_B T) = \sum_{i,j} f_R(i) f_S(i,j) f_T(j)$
- Sketch-based solution
 - Compute random variables X_R , X_S and X_T

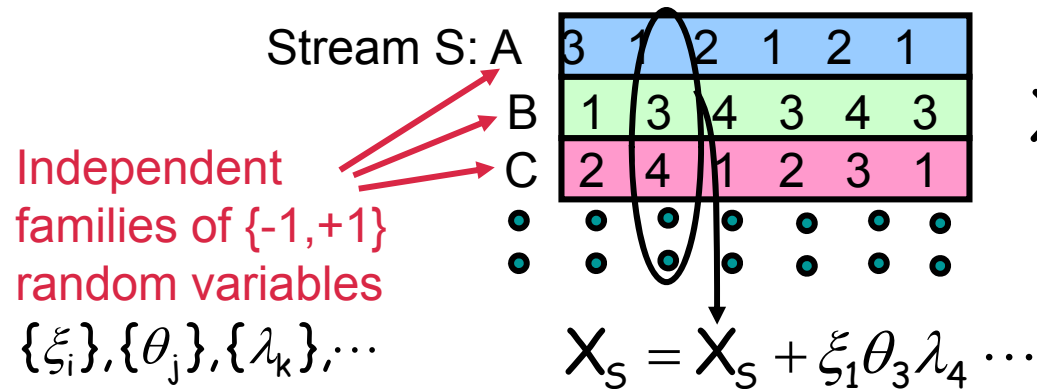


- Return $X = X_R X_S X_T$ ($E[X] = \text{COUNT}(R \bowtie_A S \bowtie_B T)$)

$$E[f_R(i) \cdot f_S(i', j) \cdot f_T(j') \xi_i \xi_{i'} \theta_j \theta_{j'}] = 0 \text{ if } i \neq i' \text{ or } j \neq j'$$

AMS Sketching for Multi-Join Aggregates

- Sketches can be used to compute answers for general multi-join COUNT queries (over streams R, S, T,)
- For each pair of attributes in equality join constraint, use independent family of $\{-1, +1\}$ random variables
- Compute random variables X_R, X_S, X_T, \dots



$$X_S = \sum_{i,j,k,\dots} f_S(i,j,k,\dots) \xi_i \theta_j \lambda_k \dots$$


- Return $X = X_R X_S X_T \dots$ ($E[X] = \text{COUNT}(R \bowtie S \bowtie T \bowtie \dots)$)

$$\text{Var}[X] \leq 2^{2m} \cdot \text{SJ}(R) \cdot \text{SJ}(S) \cdot \text{SJ}(T) \dots$$

- Explosive increase with the number of joins!

Boosting Accuracy by Sketch Partitioning: Basic Idea

- For ϵ error, need $\text{Var}[Y] \leq \frac{\epsilon^2 \text{COUNT}^2}{8}$

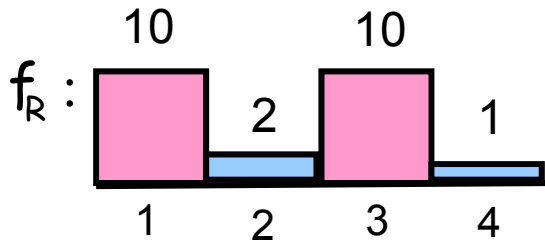

$$s = \frac{8 \cdot (2^{2m} \text{SJ}(R) \cdot \text{SJ}(S) \cdots)}{\epsilon^2 \text{COUNT}^2} \quad \text{copies}$$

$$\text{Var}[Y] = \frac{\text{Var}[X]}{s} \leq \frac{\epsilon^2 \text{COUNT}^2}{8}$$

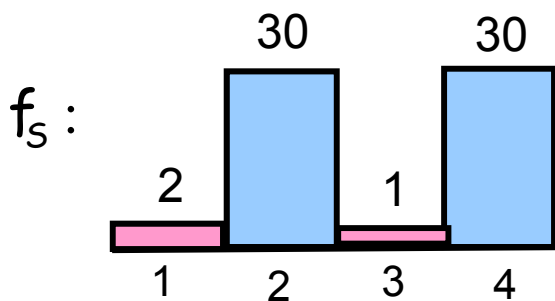
- Key Observation: Product of self-join sizes for **partitions** of streams can be *much smaller* than product of self-join sizes for streams
 - Reduce space requirements by partitioning join attribute domains
 - Overall join size = **sum of join size estimates for partitions**
 - Exploit coarse statistics (e.g., histograms) based on historical data or collected in an initial pass, to compute the *best partitioning*

Sketch Partitioning Example: Binary-Join COUNT Query

Without Partitioning



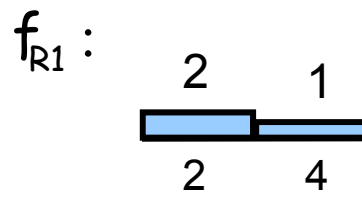
$SJ(R)=205$



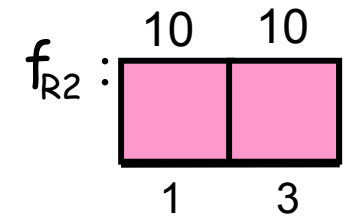
$SJ(S)=1805$

$$VAR[X] \approx 2 \cdot SJ(R) \cdot SJ(S) \approx \text{720K}$$

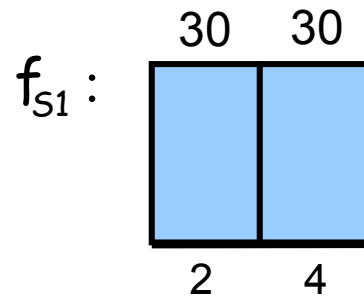
With Partitioning ($P1=\{2,4\}$, $P2=\{1,3\}$)



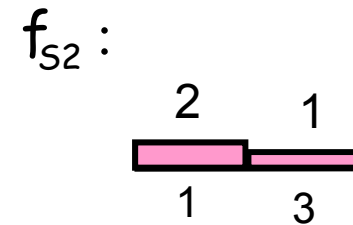
$SJ(R1)=5$



$SJ(R2)=200$



$SJ(S1)=1800$



$SJ(S2)=5$

$$VAR[X1] \approx 2 \cdot SJ(R1) \cdot SJ(S1) \approx \text{18K} \quad VAR[X2] \approx 2 \cdot SJ(R2) \cdot SJ(S2) \approx \text{2K}$$

$$X = X1 + X2, \quad E[X] = \text{COUNT}(R \bowtie S)$$

$$VAR[X] = VAR[X1] + VAR[X2] \approx \text{20K}$$

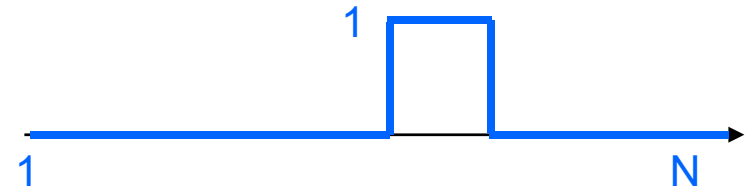
Other Applications of AMS Stream Sketching

- Key Observation: $|R1 \bowtie R2| = \sum f_1(i) f_2(i) = \langle f_1, f_2 \rangle = \text{inner product!}$
- *General result:* Streaming (ϵ, δ) estimation of "large" inner products using AMS sketching

- Other streaming inner products of interest

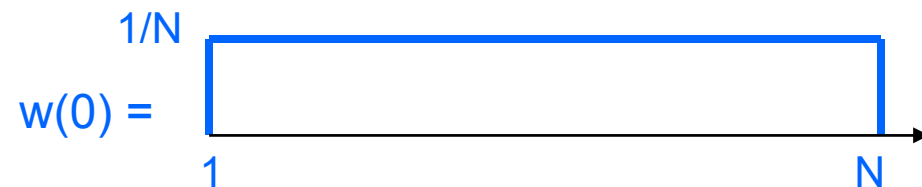
- Top-k frequencies [CCF02]

- Item frequency = $\langle f, \text{"unit_pulse"} \rangle$

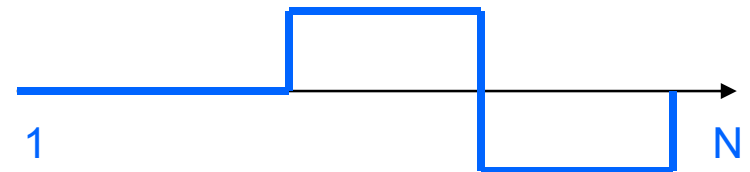


- Large wavelet coefficients [GKMS01, CGS06]

- $\text{Coeff}(i) = \langle f, w(i) \rangle$, where $w(i)$ = i -th wavelet basis vector



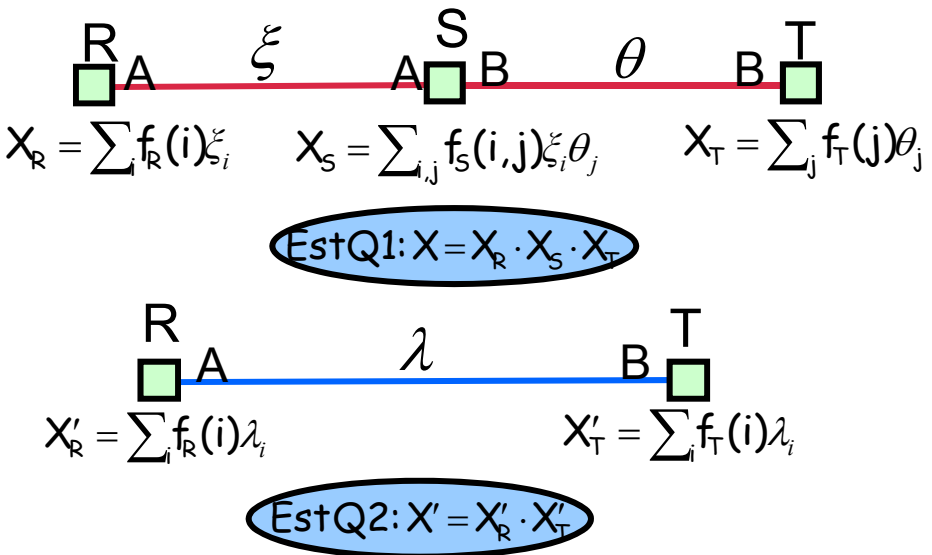
$w(i) =$



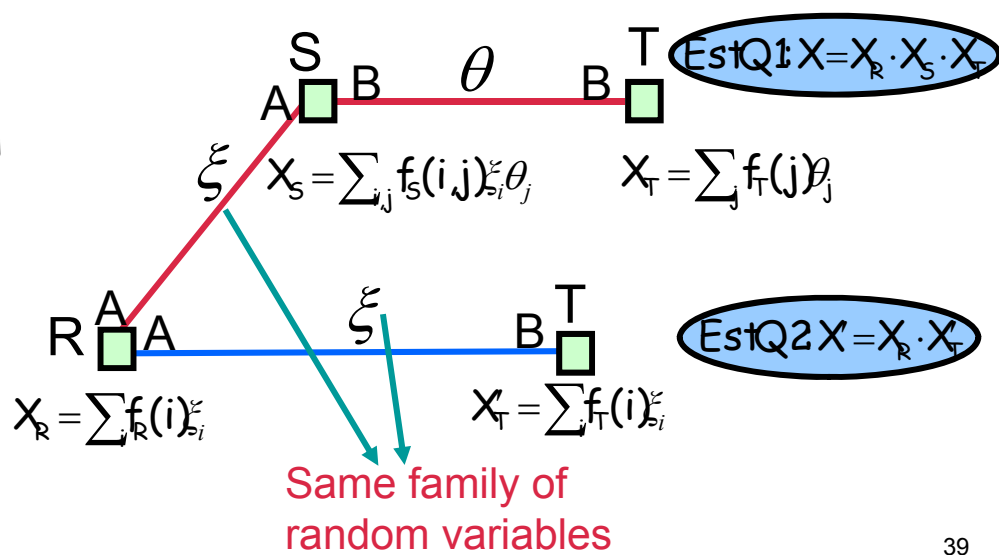
More Results on Sketches for Stream Joins

- Better accuracy using “skimmed sketches” [GGR04]
 - “Skim” dense items (i.e., large frequencies) from the AMS sketches
 - Use the “skimmed” sketch only for sparse element representation
 - Stronger worst-case guarantees, and much better in practice
 - Same effect as sketch partitioning with *no apriori knowledge!*
- Sharing sketch space/computation among *multiple queries* [DGGR04]

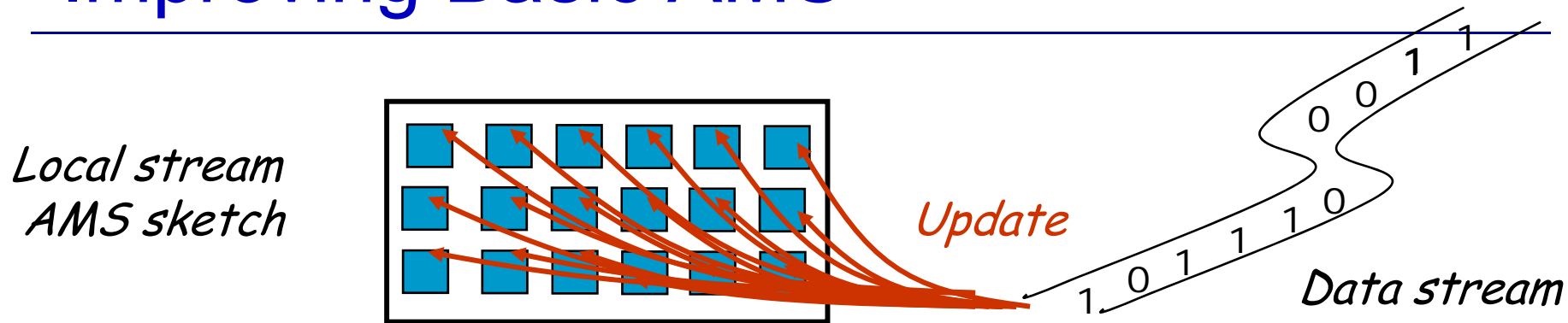
Naive



Sharing

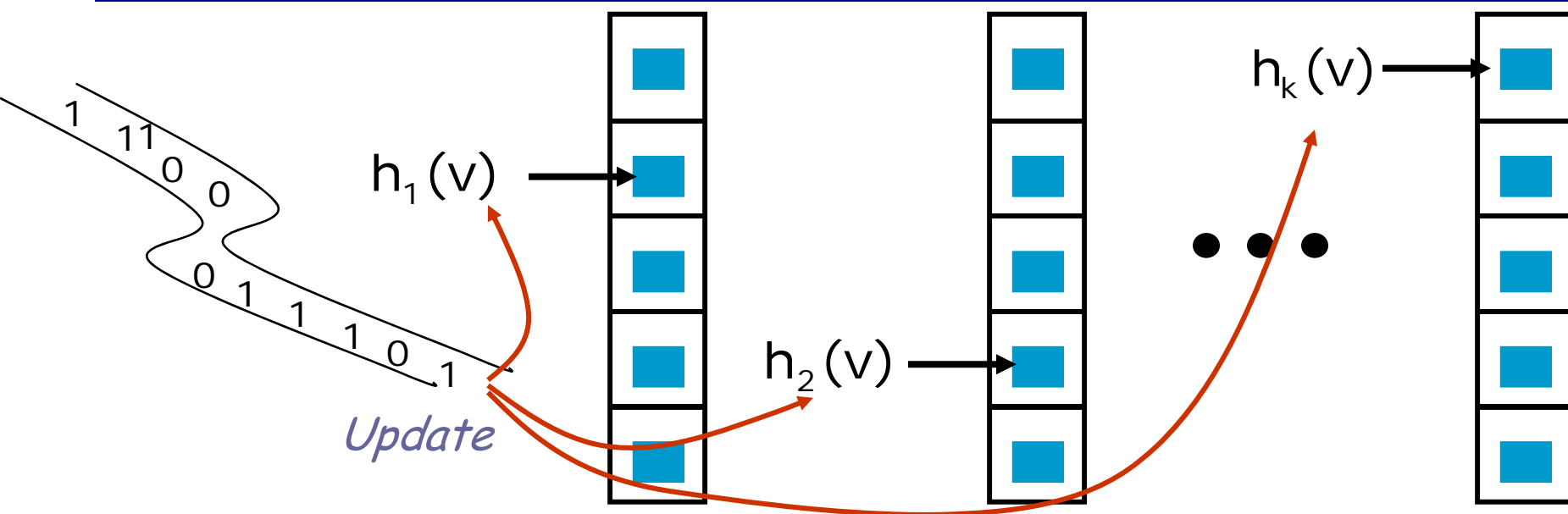


Improving Basic AMS



- Update time for basic AMS sketch is $\Omega(|\text{sketch}|)$
- *BUT...*
 - Sketches can get large – cannot afford to touch every counter for rapid-rate streams!
 - Complex queries, stringent error guarantees, ...
 - Sketch size may not be the limiting factor (PCs with GBs of RAM)

Key Idea [MG+05'08]: The Fast AMS Sketch



- **Fast AMS Sketch:** Organize the atomic AMS counters into hash-table buckets
 - Each update touches only a few counters (one per table)
 - Same space/accuracy tradeoff as basic AMS (in fact, better☺)
 - *BUT, guaranteed logarithmic update times (regardless of sketch size)!!*

Outline – Part I



- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
 - Sampling
 - Linear-Projection (aka AMS) Sketches
 - *Applications:* Join/Multi-Join Queries, Wavelets
 - Hash (aka FM) Sketches
 - *Applications:* Distinct Values, Distinct sampling
 - Another Example Sketch for Multisets: CountMin

Distinct Value Estimation

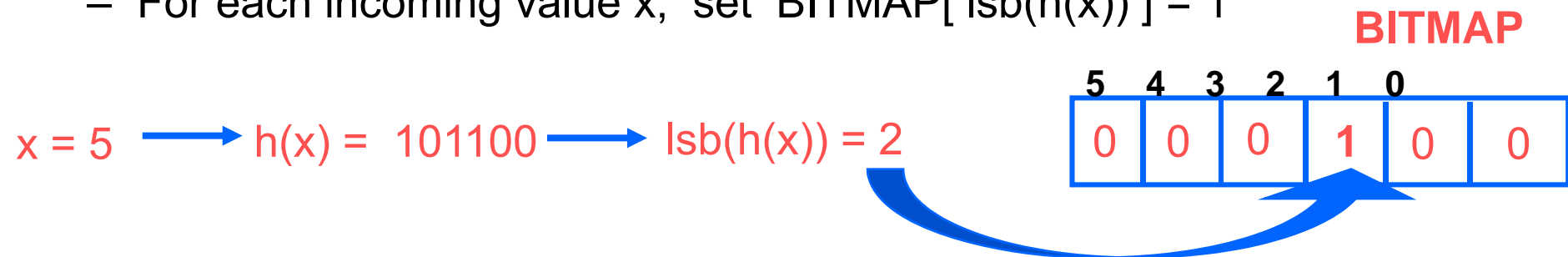
- Problem: Find the *number of distinct values* in a stream of values with domain $[0, \dots, N-1]$
 - Zeroth frequency moment F_0 , L0 (Hamming) stream norm
 - Statistics: number of *species or classes* in a population
 - Important for query optimizers
 - *Network monitoring*: distinct destination IP addresses, source/destination pairs, requested URLs, etc.
- Example (N=64) Data stream:

3	0	5	3	0	1	7	5	1	0	3	7
---	---	---	---	---	---	---	---	---	---	---	---

Number of distinct values: 5
- Hard problem for random sampling! [CCMN00]
 - Must sample almost the entire table to guarantee the estimate is within a factor of 10 with probability $> 1/2$, regardless of the estimator used!

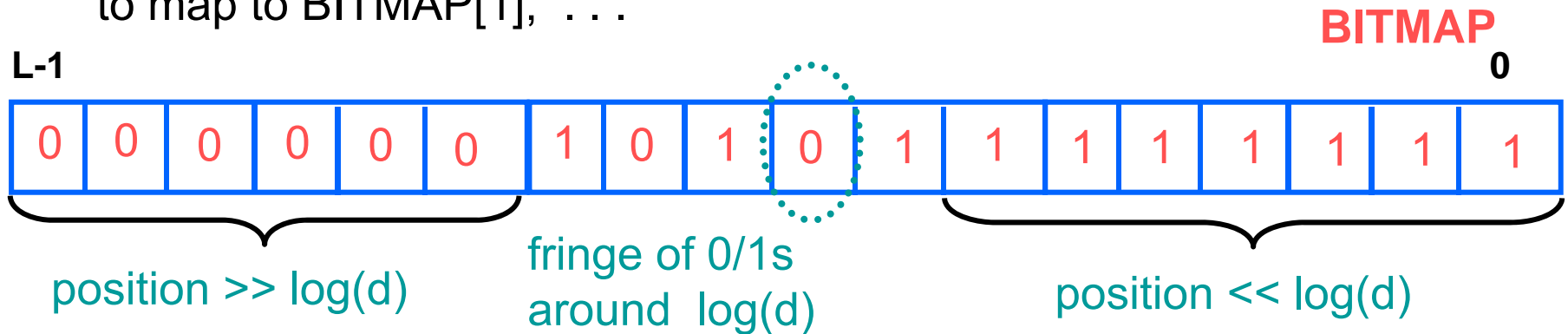
Hash (aka FM) Sketches for Distinct Value Estimation [FM85]

- Assume a hash function $h(x)$ that maps incoming values x in $[0, \dots, N-1]$ uniformly across $[0, \dots, 2^L-1]$, where $L = O(\log N)$
- Let $\text{lsb}(y)$ denote the position of the least-significant 1 bit in the binary representation of y
 - A value x is mapped to $\text{lsb}(h(x))$
- Maintain *Hash Sketch* = BITMAP array of L bits, initialized to 0
 - For each incoming value x , set $\text{BITMAP}[\text{lsb}(h(x))] = 1$



Hash (aka FM) Sketches for Distinct Value Estimation [FM85]

- By uniformity through $h(x)$: $\text{Prob}[\text{BITMAP}[k]=1] = \text{Prob}[10^k] = \frac{1}{2^{k+1}}$
 - Assuming d distinct values: expect $d/2$ to map to $\text{BITMAP}[0]$, $d/4$ to map to $\text{BITMAP}[1]$, ...



- Let R = position of rightmost zero in BITMAP
 - Use as indicator of $\log(d)$
- [FM85] prove that $E[R] = \log(\phi d)$ where $\phi = .7735$
 - Estimate $d = 2^R / \phi$
 - Average several iid instances (different hash functions) to reduce estimator variance

Hash Sketches for Distinct Value Estimation

- [FM85] assume “ideal” hash functions $h(x)$ (N-wise independence)
 - [AMS96]: pairwise independence is sufficient
 - $h(x) = (a \cdot x + b) \bmod N$, where a, b are random binary vectors in $[0, \dots, 2^L - 1]$
 - Small-space (ε, δ) estimates for distinct values proposed based on FM ideas
- *Delete-Proof*: Just use counters instead of bits in the sketch locations
 - +1 for inserts, -1 for deletes
- *Composable*: Component-wise OR/add distributed sketches together
 - Estimate $|S_1 \cup S_2 \cup \dots \cup S_k| = \text{set-union cardinality}$

Generalization: Distinct Values Queries



- SELECT COUNT(DISTINCT target-attr)
- FROM relation
- WHERE predicate

Template

- SELECT COUNT(DISTINCT o_custkey)
- FROM orders
- WHERE o_orderdate >= '2012-01-01'

TPC-H example

- "How many distinct customers have placed orders this year?"
 - Predicate not necessarily only on the DISTINCT target attribute
-
- *Approximate answers with error guarantees over a stream of tuples?*

Distinct Sampling [Gib01]



Key Ideas

- Use FM-like technique to collect a specially-tailored sample over the *distinct values in the stream*
 - ***Use hash function mapping to sample values from the data domain!!***
 - Uniform random sample of the distinct values
 - Very different from traditional random sample: each distinct value is chosen uniformly regardless of its frequency
 - DISTINCT query answers: simply scale up sample answer by sampling rate
- To handle additional predicates
 - *Reservoir sampling* of tuples for each distinct value in the sample
 - Use reservoir sample to evaluate predicates

Building a Distinct Sample [Gib01]

- Use FM-like hash function $h()$ for each streaming value x
 - $\text{Prob}[h(x) = k] = \frac{1}{2^{k+1}}$
- **Key Invariant:** “All values with $h(x) \geq \text{level}$ (and only these) are in the distinct sample”

DistinctSampling(B , r)

// B = space bound, r = tuple-reservoir size for each distinct value

level = 0; S = ϕ

for each new tuple t do

 let x = value of DISTINCT target attribute in t

 if $h(x) \geq \text{level}$ then // x belongs in the distinct sample

 use t to update the reservoir sample of tuples for x

 if $|S| \geq B$ then // out of space

 evict from S all tuples with $h(\text{target-attribute-value}) = \text{level}$

 set level = level + 1

Using the Distinct Sample [Gib01]

- If $\text{level} = l$ for our sample, then we have selected all distinct values x such that $h(x) \geq l$
 - $\text{Prob}[h(x) \geq l] = \frac{1}{2^l}$
 - By $h()$'s randomizing properties, we have uniformly sampled a 2^{-l} fraction of the distinct values in our stream

Our sampling rate!
- *Query Answering:* Run distinct-values query on the distinct sample and scale the result up by 2^l
- *Distinct-value estimation:* Guarantee ε relative error with probability $1 - \delta$ using $O(\log(1/\delta)/\varepsilon^2)$ space
 - For $q\%$ selectivity predicates the space goes up inversely with q
- *Experimental results:* 0-10% error vs. 50-250% error for previous best approaches, using 0.2% to 10% synopses

Distinct Sampling Example

- $B=3, N=8$ ($r = 0$ to simplify example)

Data stream:

3 0 5 3 0 1 7 5 1 0 3 7

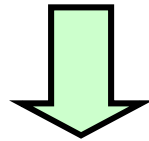
hash:

0	1	3	5	7
0	1	0	1	0

Data stream:

1 7 5 1 0 3 7

$S=\{3,0,5\}, \text{ level} = 0$



$S=\{1,5\}, \text{ level} = 1$

- Computed value: 4

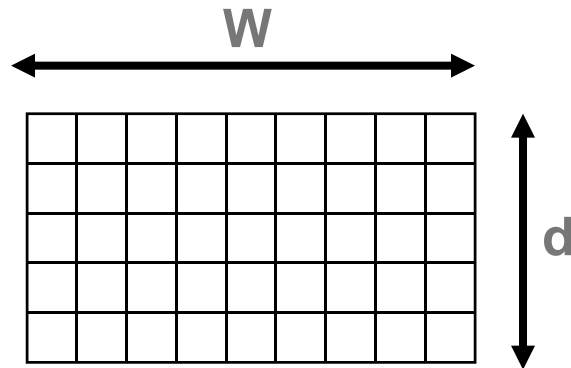
Outline – Part I



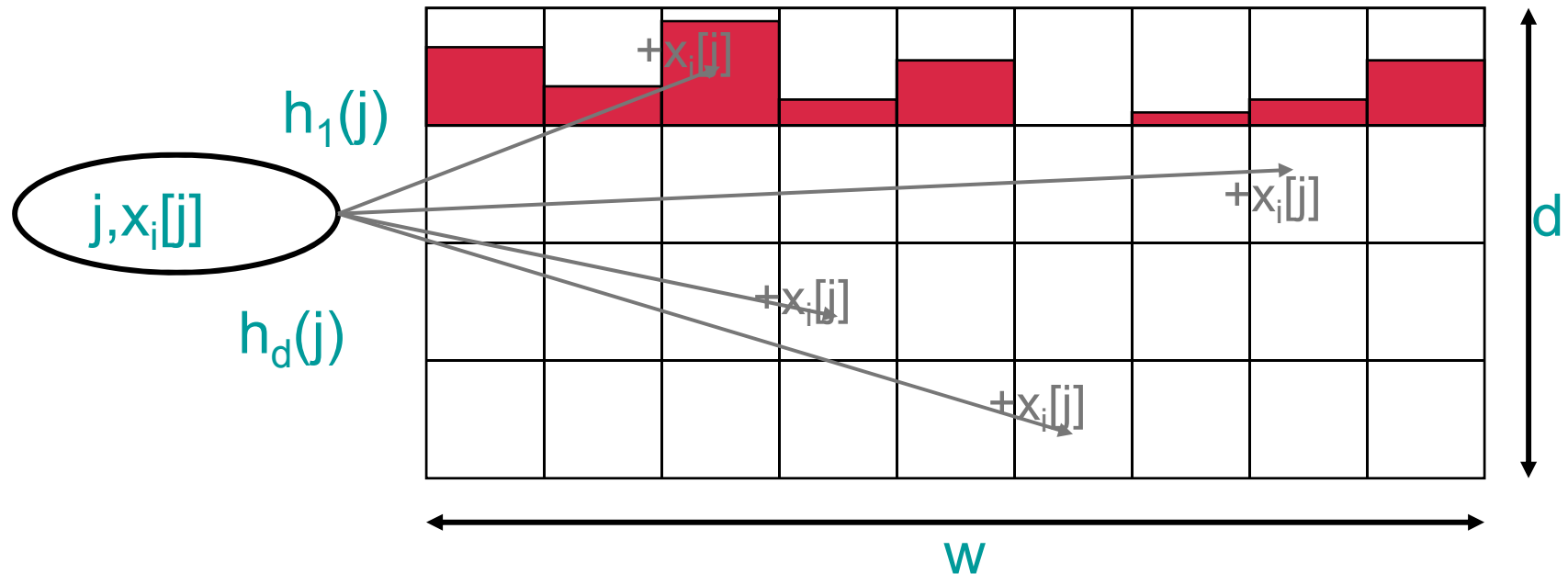
- Introduction & Motivation
- Data Streaming Models & Basic Mathematical Tools
- Summarization/Sketching Tools for Streams
 - Sampling
 - Linear-Projection (aka AMS) Sketches
 - *Applications:* Join/Multi-Join Queries, Wavelets
 - Hash (aka FM) Sketches
 - *Applications:* Distinct Values, Distinct sampling
 - Another Example Sketch for Multisets: CountMin

The CountMin (CM) Sketch

- Simple sketch idea, can be used for point queries, range queries, quantiles, join size estimation
- Model input at each node as a vector A of dimension N , where N is large
- Creates a small summary as an array of $w \times d$ in size
- Use d hash functions to map vector entries to $[1..w]$



CM Sketch Structure



- Each entry in vector A is mapped to one bucket per row
- Merge two sketches by entry-wise summation
- Estimate $A[j]$ by taking $\min_k \text{sketch}[k, h_k(j)]$

CM Sketch Summary

- CM sketch guarantees approximation error on point queries less than $\varepsilon ||A||_1$ in size $O(1/\varepsilon \log 1/\delta)$
 - Probability of more error is less than $1-\delta$
 - Similar guarantees for range queries, quantiles, join size
- Hints
 - Counts are *biased!* Can you limit the expected amount of extra "mass" at each bucket? *(Use Markov)*
 - *Use Chernoff* to boost the confidence for the min{} estimate
- *Food for thought:* How do the CM sketch guarantees compare to AMS??

Conclusions – Part I

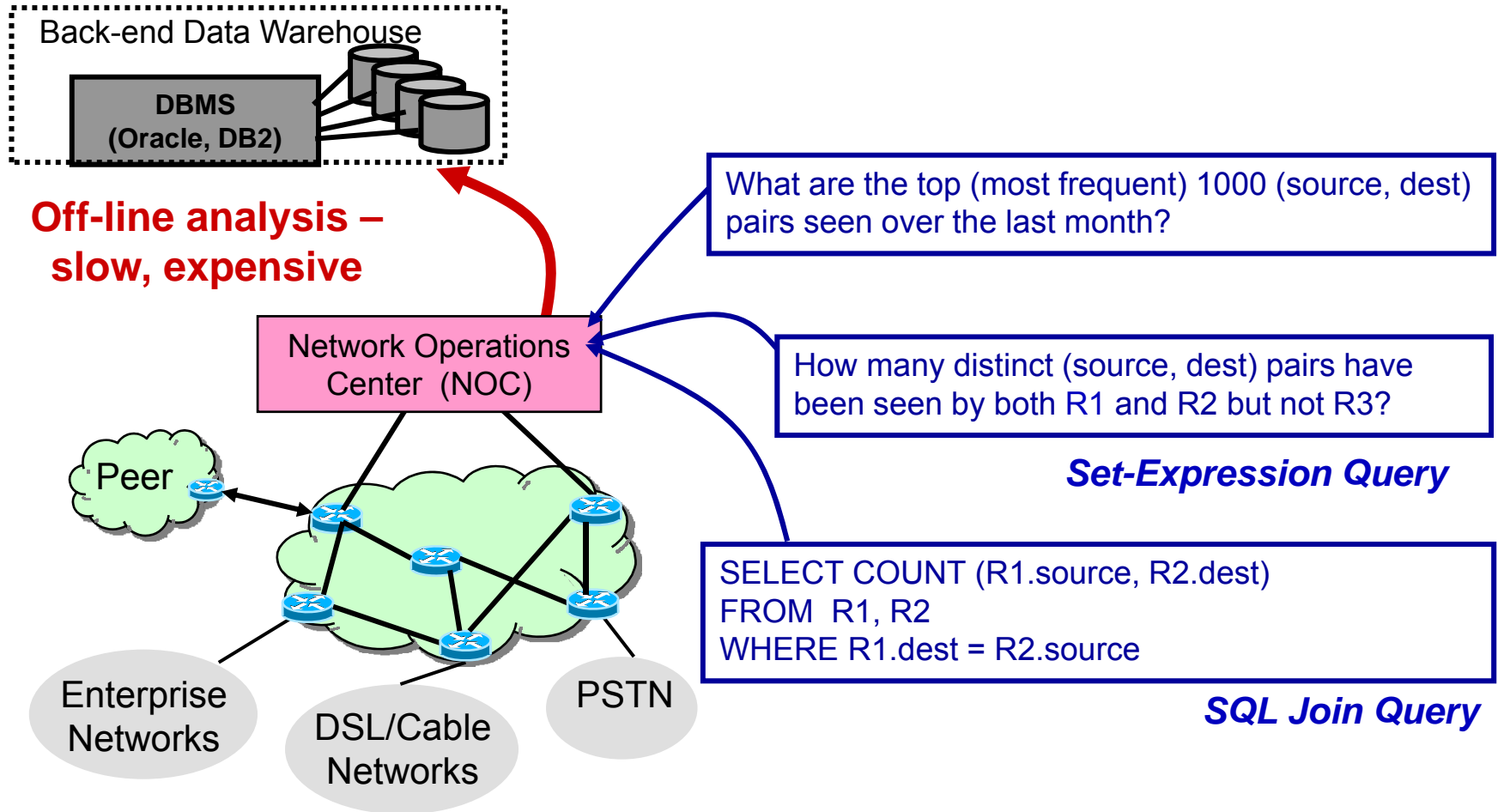


- Querying and finding patterns in massive streams is a real problem with many real-world applications
- Fundamentally rethink data-management issues under stringent constraints
 - Single-pass algorithms with limited memory resources
- A lot of progress in the last few years
 - Algorithms, system models & architectures
 - GigaScope (AT&T), Aurora (Brandeis/Brown/MIT), Niagara (Wisconsin), STREAM (Stanford), Telegraph (Berkeley)
- Commercial acceptance still lagging, but will most probably grow in coming years
 - Specialized systems (e.g., fraud detection, network monitoring), but still far from “DSMSs”

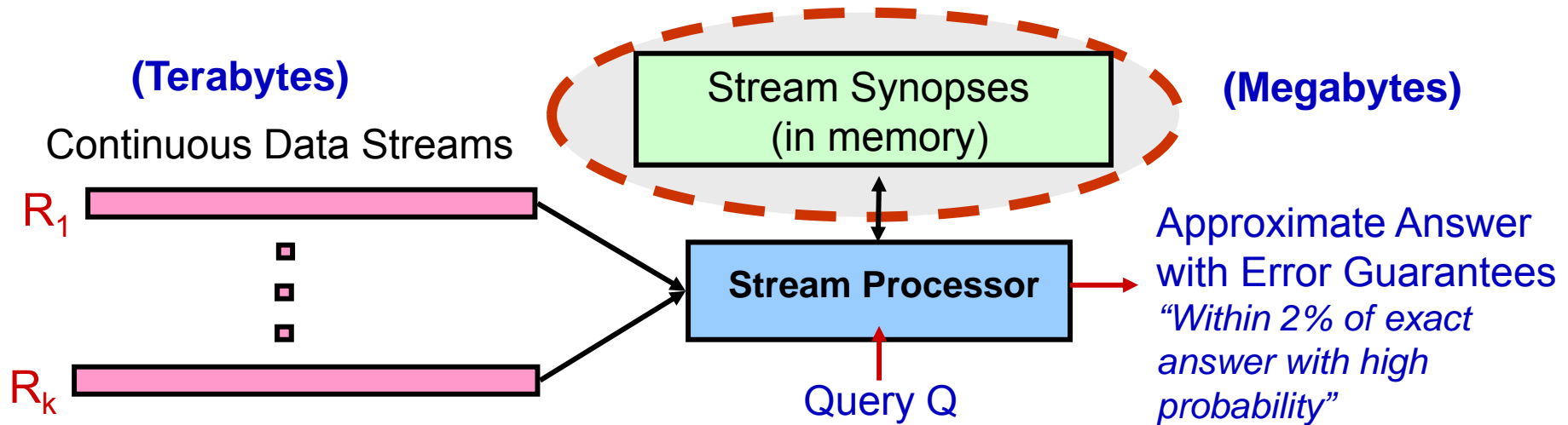


Part II: Distributed Data Streaming

Network Monitoring Queries



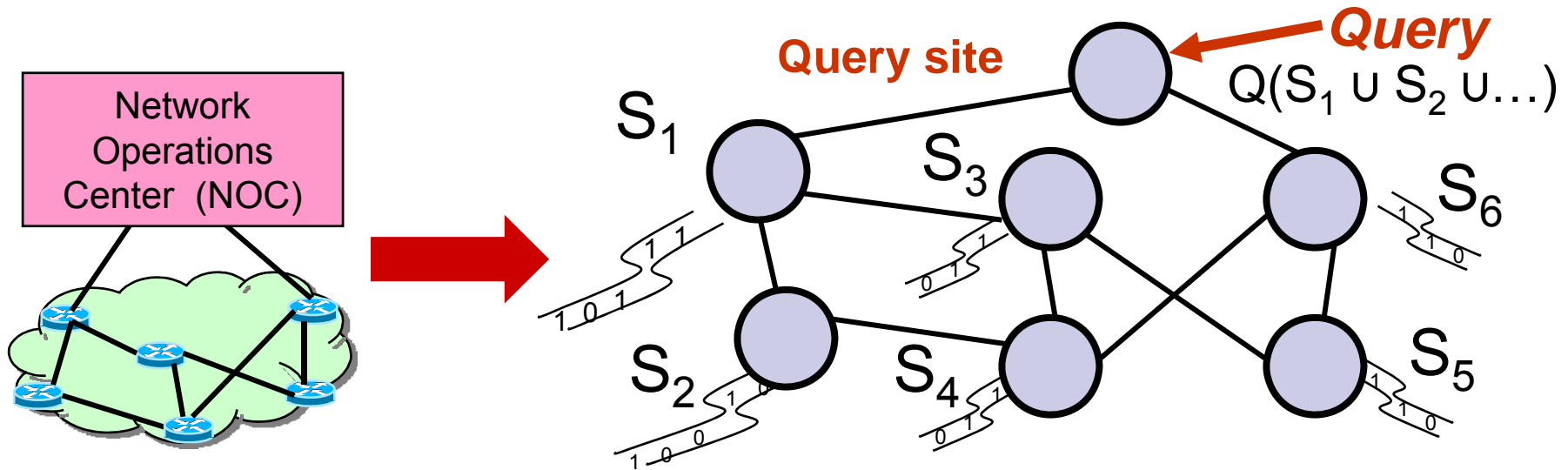
Data-Stream Algorithmics Model



- *Approximate answers*— e.g. trend analysis, anomaly detection
- Requirements for stream synopses
 - *Single Pass*: Each record is examined at most once
 - *Small Space*: Log or polylog in data stream size
 - *Small-time*: Low per-record processing time (maintain synopses)
 - Also: *delete-proof*, *composable*, ...



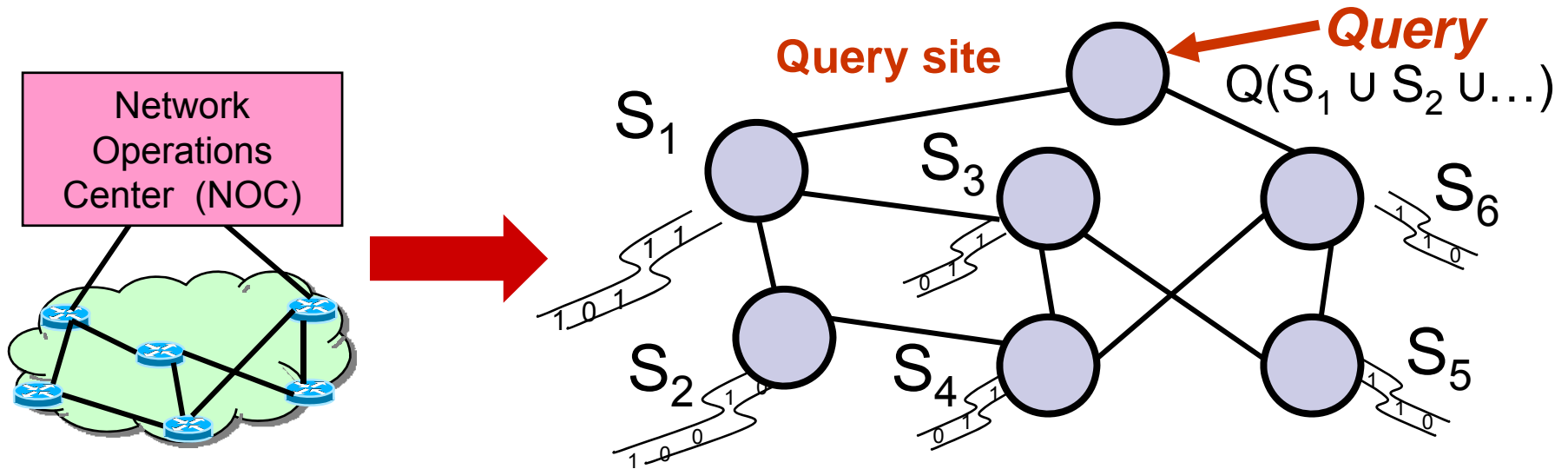
Distributed Streams Model



- Large-scale querying/monitoring: *Inherently distributed!*
 - Streams physically distributed across remote sites
E.g., stream of UDP packets through subset of edge routers
- Challenge is “holistic” querying/monitoring
 - Queries over the *union of distributed streams* $Q(S_1 \cup S_2 \cup \dots)$
 - Streaming data is spread throughout the network



Distributed Streams Model



- Need timely, accurate, and efficient query answers
- Additional complexity over centralized data streaming!
- Need space/time- *and communication-efficient* solutions
 - Minimize network overhead
 - Maximize network lifetime (e.g., sensor battery life)
 - Cannot afford to “centralize” all streaming data

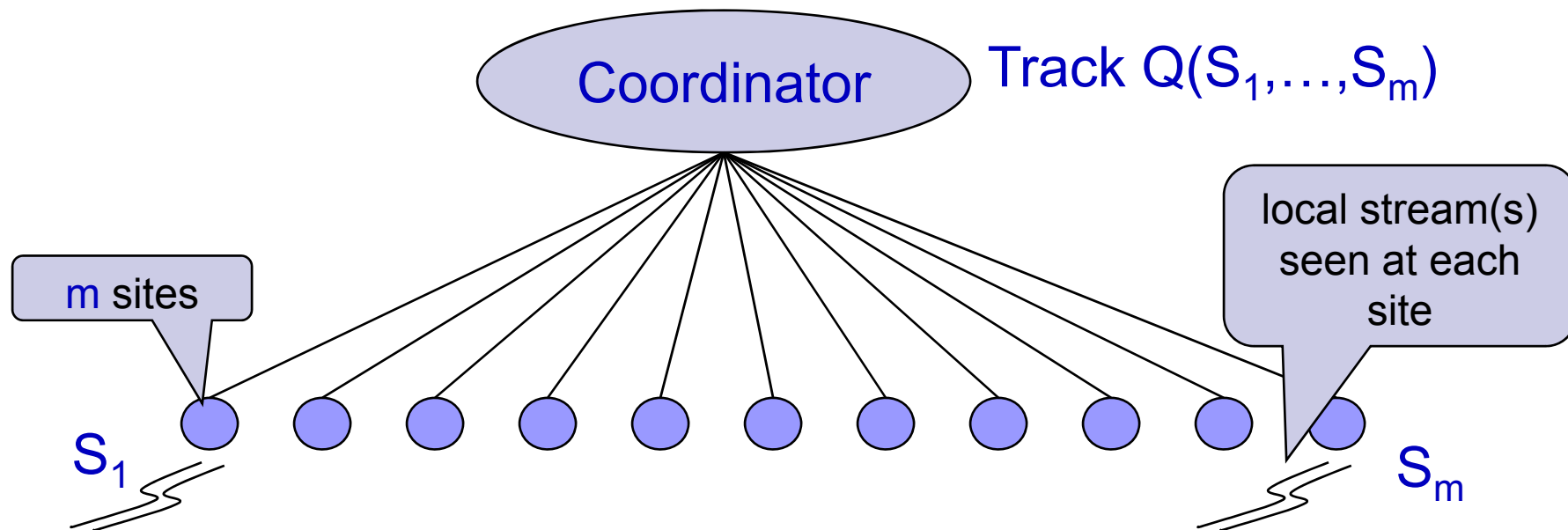


Outline - Part II

- Introduction and Motivation
- Continuous Distributed Monitoring (CDM) of Streams
 - Problem Setup
 - CDM of Complex Aggregates using AMS Sketches
 - Geometric Monitoring of Distributed Streams
- Extensions



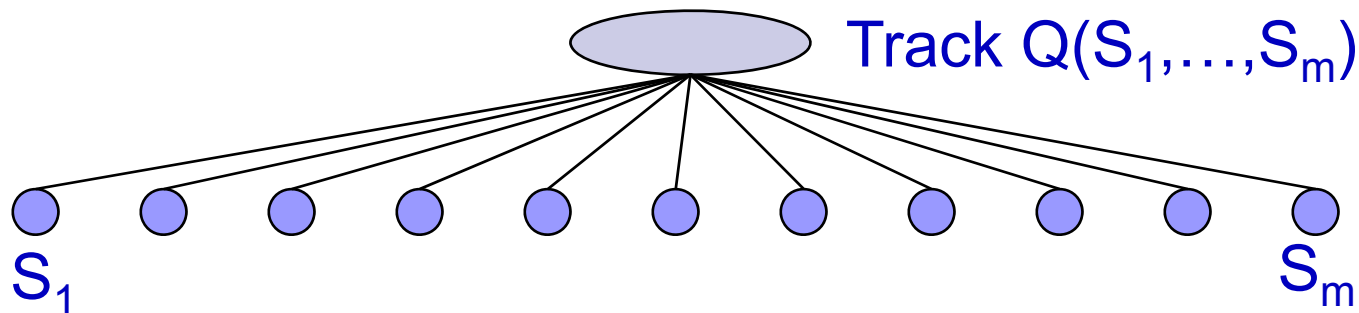
Continuous Distributed Monitoring



- Other structures possible (e.g., hierarchical)
 - Could allow site-site communication, but mostly unneeded
- Goal:** *Continuously track* (global) query over streams at the coordinator
- Large-scale network-event monitoring, real-time anomaly/DDoS attack detection, power grid monitoring, ...

Continuous Distributed Streams

- But... local site streams continuously change!
 - E.g., new readings are made, new data arrives
 - *Assumption:* Changes are somewhat smooth and gradual
- Need to guarantee an answer at the coordinator that is always correct, within some guaranteed accuracy bound
- Naïve solutions must *continuously* centralize all data
 - Enormous communication overhead!

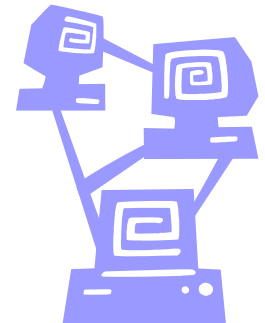


Challenges

- Monitoring is **Continuous...**
 - Real-time tracking, rather than one-shot query/response
- **...Distributed...**
 - Each remote site only observes part of the global stream(s)
 - *Communication constraints*: must minimize monitoring burden
- **...Streaming...**
 - Each site sees a high-speed local data stream and can be resource (CPU/memory) constrained
- **...Holistic...**
 - Challenge is to monitor the *complete global data distribution*
 - Simple aggregates (e.g., aggregate traffic) are easier

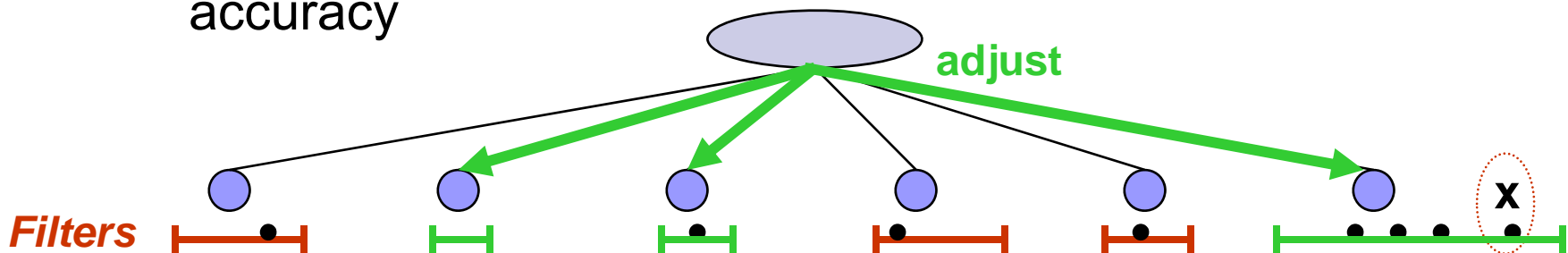
How about Periodic Polling?

- Sometimes **periodic polling** suffices for simple tasks
 - E.g., SNMP polls total traffic at coarse granularity
- Still need to deal with holistic nature of aggregates
- Must balance polling frequency against communication
 - Very frequent polling causes high communication, excess battery use in sensor networks
 - Infrequent polling means delays in observing events
- Need techniques to reduce communication while guaranteeing rapid response to events



Communication-Efficient Monitoring

- Exact answers are not needed
 - Approximations with accuracy guarantees suffice
 - Tradeoff *accuracy* and *communication/ processing cost*
- **Key Idea:** “Push-based” in-network processing
 - Local filters installed at sites process local streaming updates
 - Offer bounds on local-stream behavior (at coordinator)
 - “Push” information to coordinator only when filter is violated
 - “**Safe**”! Coordinator sets/adjusts local filters to guarantee accuracy



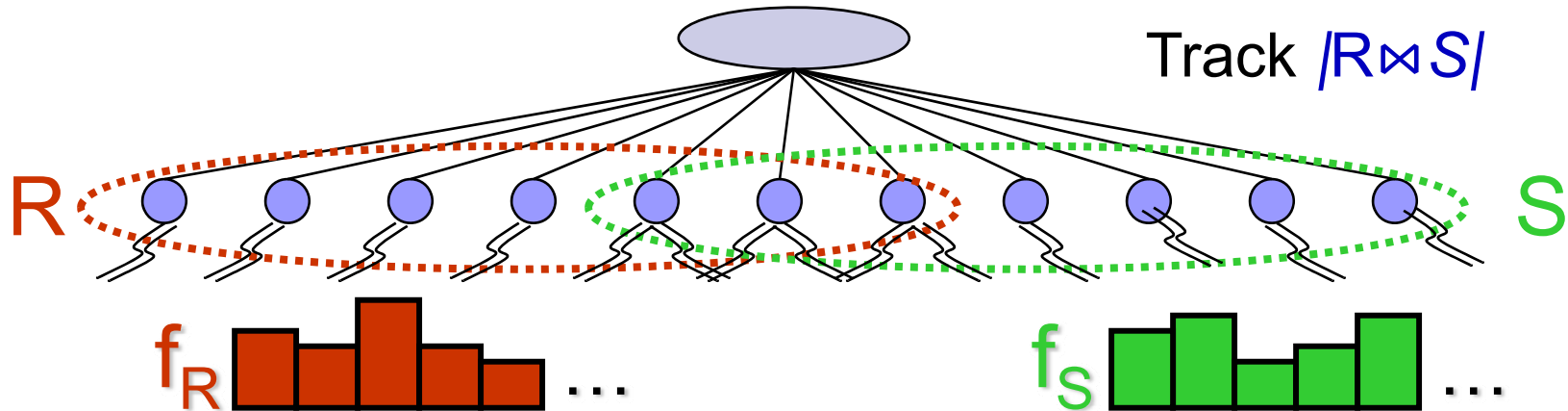
Local Slack/Filter Allocation

- Problem is relatively straightforward for tracking ***linear*** functions/queries
 - By *additivity*, simply divide the total slack across sites
 - Several early papers on tracking distributed counts, sums, top-k frequencies, etc.
- Things get much more interesting for ***non-linear*** functions and when approximate representations (sketches) are employed

Outline – Part II

- Introduction and Motivation
- Continuous Distributed Monitoring (CDM) of Streams
 - Problem Setup
 - CDM of Complex Aggregated using AMS Sketches
 - Geometric Monitoring of Distributed Streams
- Extensions

Tracking Complex Aggregate Queries



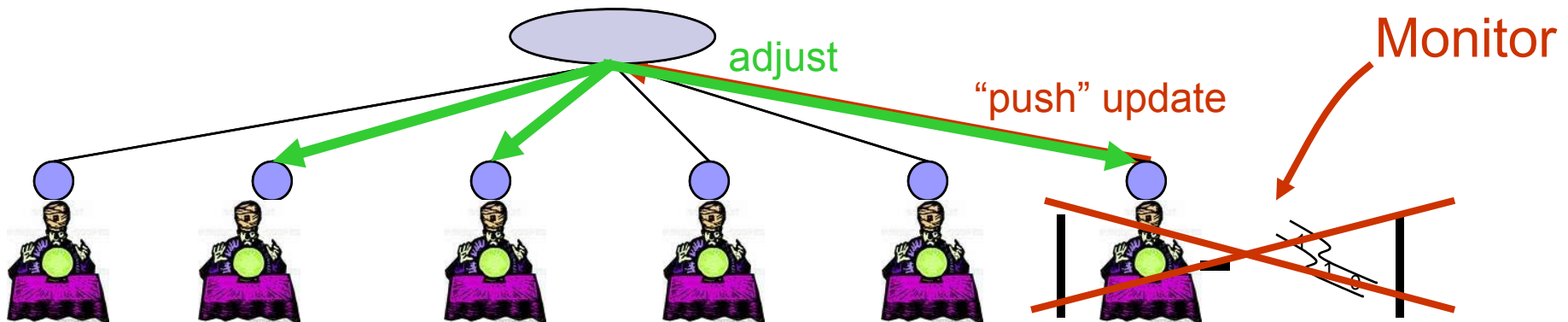
- Continuous distributed tracking of complex aggregate queries using AMS sketches and local prediction models
[Cormode, MG'05,'08]
- *Class of queries:* Generalized inner products of streams

$$|R \bowtie S| = f_R \cdot f_S = \sum_v f_R[v] f_S[v] \quad (\pm \varepsilon \|f_R\|_2 \|f_S\|_2)$$

- Join/multi-join aggregates, range queries, heavy hitters, histograms, wavelets, ...

One More Idea: Shared Prediction Models

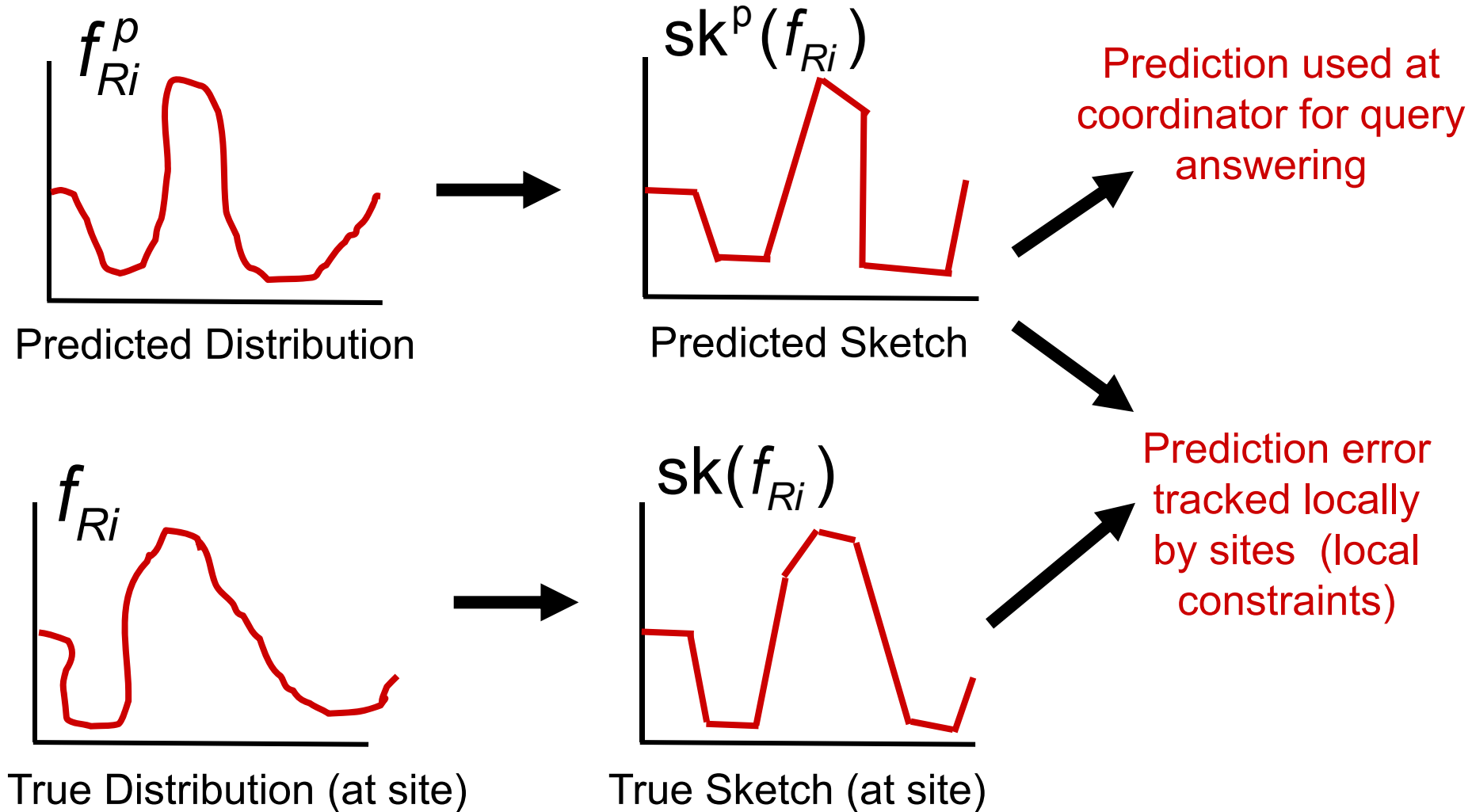
- Naïve “*static*” prediction: Local stream assumed “unchanged” since last update
 - No update from site \Rightarrow last update (“predicted” value) is within required filter bounds \Rightarrow global error bound
- *Dynamic, sophisticated prediction models* of site behavior
 - Built locally at sites and *shared* with coordinator
 - Model complex stream patterns, reduce number of updates to coordinator
 - *But...* more complex to maintain and communicate



Local Sketches and Sketch Prediction

- Use AMS sketches to summarize local site distributions
 - Synopsis=small collection of random linear projections $sk(f_{R,i})$
 - *Linear transform*: Simply add to get global stream sketch
 - Combine with “*push-based*” processing
- Minimize updates to coordinator through *Sketch Prediction*
 - Try to predict how local-stream distributions (and their sketches) will evolve over time
 - Concise *sketch-prediction models*, built locally at remote sites and communicated to coordinator
 - *Shared knowledge* on expected stream behavior over time: Achieve “stability”

Sketch Prediction



Query Tracking Protocol

Tracking. At site i keep sketch of stream so far, $\text{sk}(f_{R,i})$

- Track local deviation between stream and prediction:

$$\| \text{sk}(f_{R,i}) - \text{sk}^p(f_{R,i}) \|_2 \leq \theta / \sqrt{k_i} \| \text{sk}(f_{R,i}) \|_2 \quad (*)$$

- Send current sketch (and other info) if violated

Theorem [MG+'05'08]. If $(*)$ holds at sites, then, at coordinator

$$\text{sk}^p(f_R) \cdot \text{sk}^p(f_S) \in f_R \cdot f_S \pm (\varepsilon + 2\theta) \|f_R\|_2 \|f_S\|_2$$

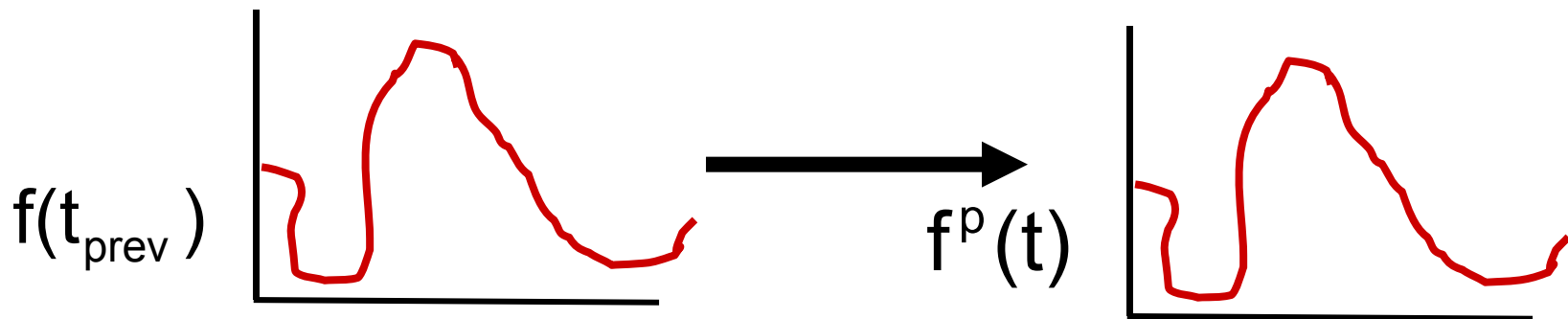
- ε = local-sketch summarization error (at remote sites)
- θ = upper bound on local-stream deviation from prediction (“Lag” between site and coordinator view)

■ **Key Insight:** *With local deviations bounded, predicted sketches at coordinator are **guaranteed** $(\varepsilon + 2\theta)$ -accurate*

- *Generalizes to other queries*

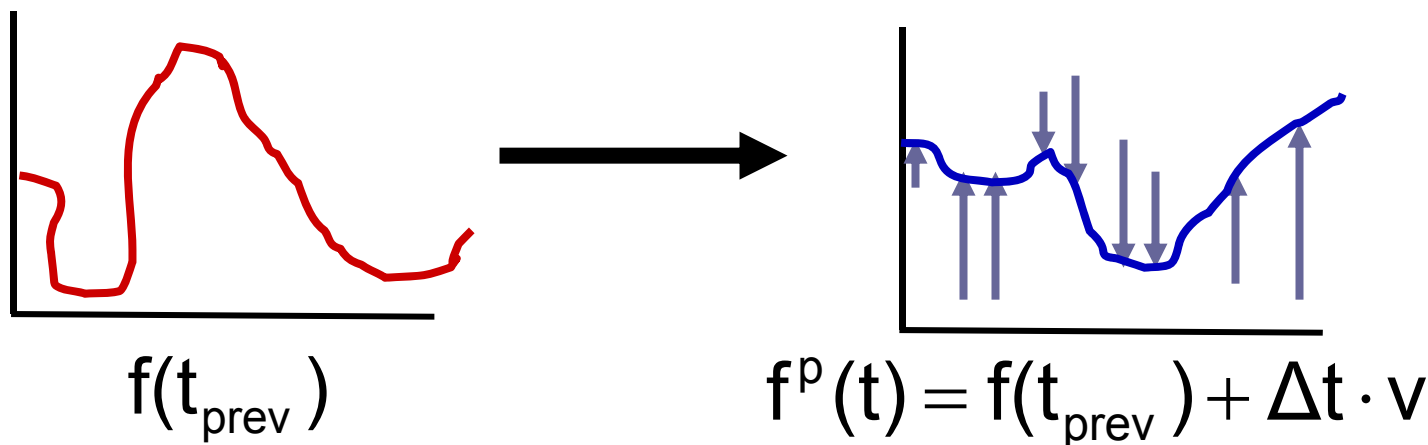
Sketch-Prediction Models

- Simple, concise models of local-stream behavior
 - Sent to coordinator to keep site/coordinator “in-sync”
 - Many possible alternatives
- Static model: No change in distribution since last update
 - Naïve, “no change” assumption:
 - No model info sent to coordinator, $sk^p(f(t)) = sk(f(t_{prev}))$



Sketch-Prediction Models

- **Velocity/Acceleration model:** Predict change through “velocity” vectors from recent local history (simple linear model)
 - Velocity model: $f^p(t) = f(t_{\text{prev}}) + (t - t_{\text{prev}}) \cdot v$
 - By sketch linearity, $\text{sk}^p(f(t)) = \text{sk}(f(t_{\text{prev}})) + (t - t_{\text{prev}}) \cdot \text{sk}(v)$
 - Just need to communicate one extra sketch
 - Can extend with acceleration component



Sketch-Prediction Models

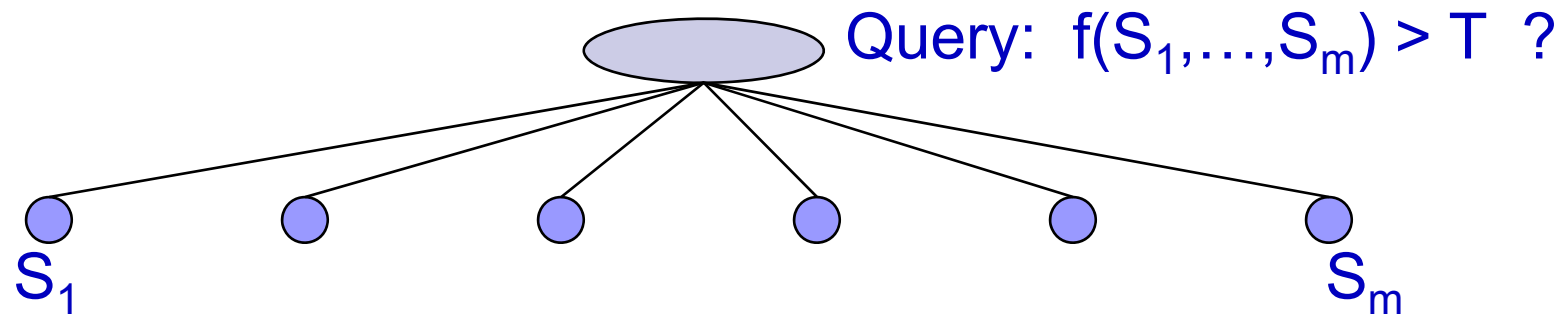
Model	Info	Predicted Sketch
Static	\emptyset	$sk^p(f(t)) = sk(f(t_{prev}))$
Velocity/ Acceleration	$sk(v)$	$sk^p(f(t)) = sk(f(t_{prev})) + \Delta t \cdot sk(v)$

- Communication cost analysis: comparable to *one-shot* sketch computation
- Order of magnitude improvements in communication
- Many other models possible – not the focus here...
 - Need to carefully balance power & conciseness

Outline – Part II

- Introduction and Motivation
- Continuous Distributed Monitoring (CDM) of Streams
 - Problem Setup
 - CDM of Complex Aggregated using AMS Sketches
 - Geometric Monitoring of Distributed Streams
- Extensions

Thresholding *Non-Linear* Functions?



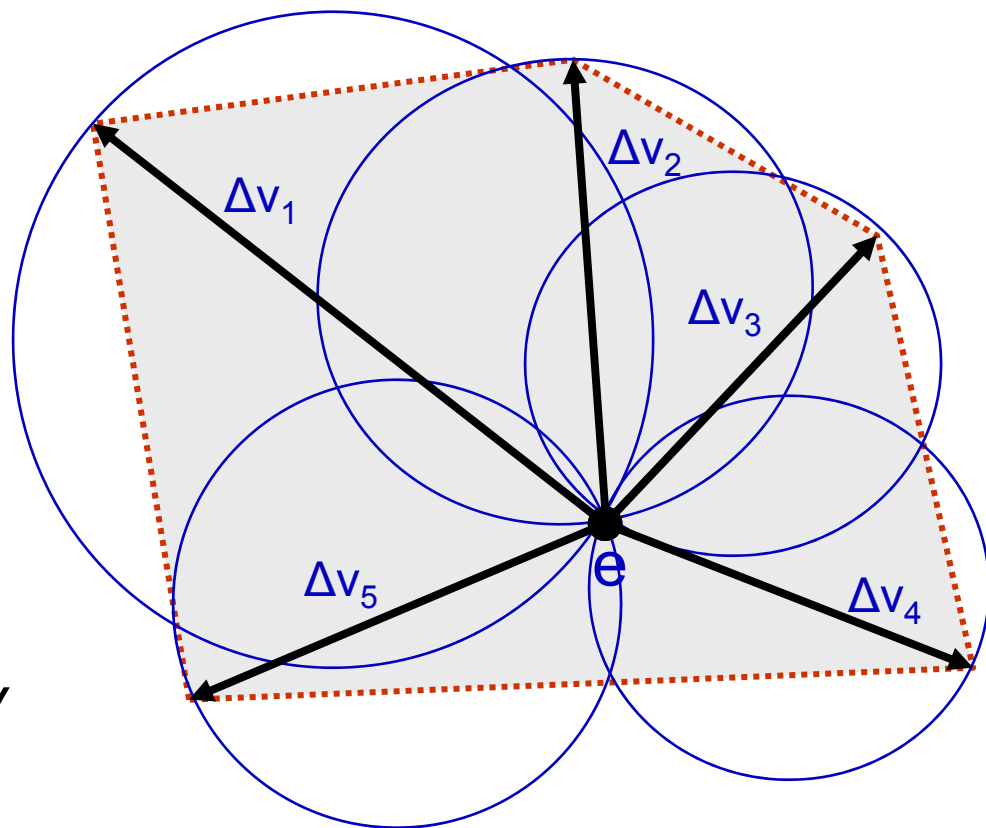
- For *general, non-linear* $f()$, the problem becomes a lot harder!
 - E.g., information gain or entropy over global data distribution
 - *Non-trivial* to decompose the global threshold into “safe” local site constraints
 - E.g., consider $N=(N_1+N_2)/2$ and $f(N) = 6N - N^2 > 1$
Impossible to break into thresholds for $f(N_1)$ and $f(N_2)$

Monitoring General Threshold Functions

- Interesting *geometric* approach [Scharfman et al.'06]
 - *Provided the basis for LIFT!*
- Each site tracks a *local statistics vector* \mathbf{v}_i (e.g., data distribution)
- Global condition is $f(\mathbf{v}) > T$, where $\mathbf{v} = \sum_i \lambda_i \mathbf{v}_i$ ($\sum_i \lambda_i = 1$)
 - \mathbf{v} = convex combination of local statistics vectors
- All sites have an estimate $\mathbf{e} = \sum_i \lambda_i \mathbf{v}_i'$ of \mathbf{v} based on latest update \mathbf{v}_i' from site i
- Each site i continuously tracks its *drift* from its most recent update $\Delta \mathbf{v}_i = \mathbf{v}_i - \mathbf{v}_i'$

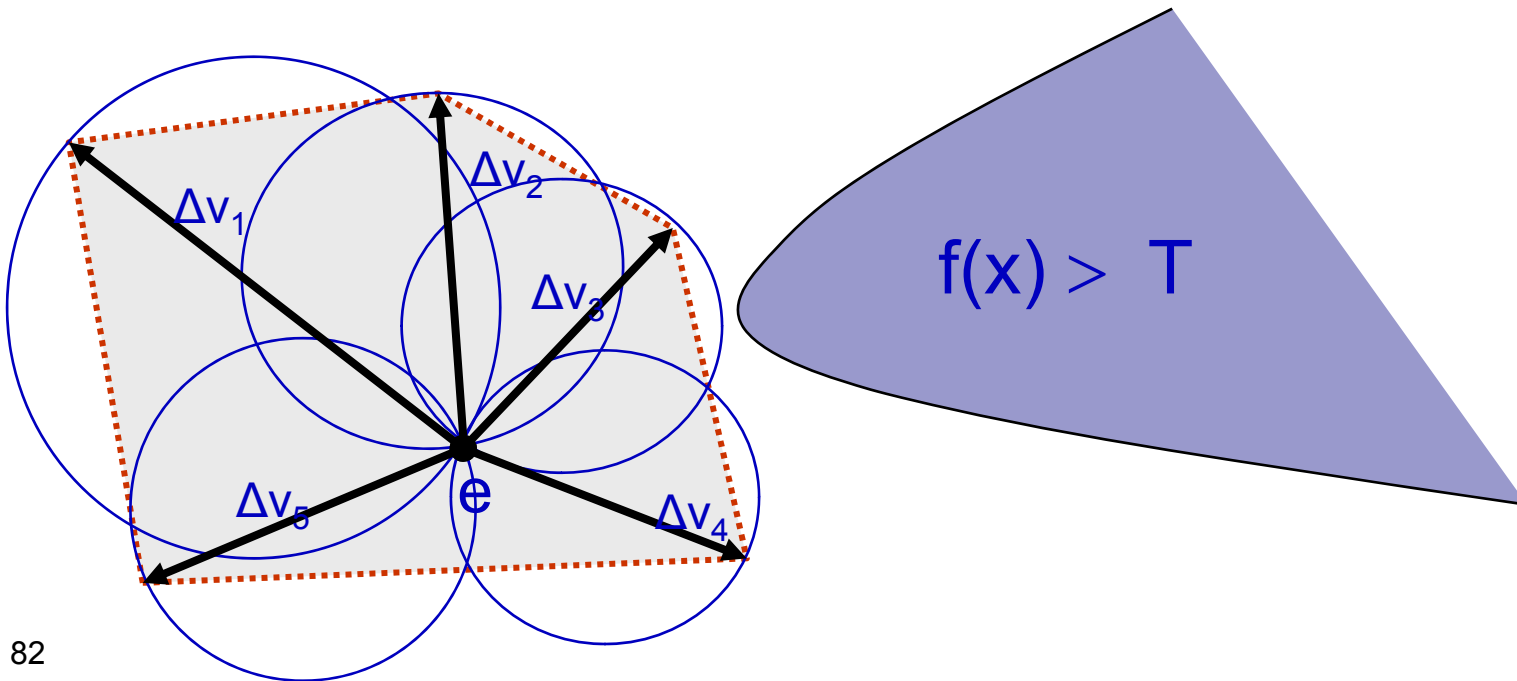
Monitoring General Threshold Functions

- Key observation: $\mathbf{v} = \sum_i \lambda_i \cdot (\mathbf{e} + \Delta \mathbf{v}_i)$
(a *convex combination* of “translated” local drifts)
- \mathbf{v} lies in the *convex hull* of the $(\mathbf{e} + \Delta \mathbf{v}_i)$ vectors
- Convex hull is completely covered by the *balls* with radii $\|\Delta \mathbf{v}_i / 2\|_2$ centered at $\mathbf{e} + \Delta \mathbf{v}_i / 2$
- Each such ball can be constructed *independently*



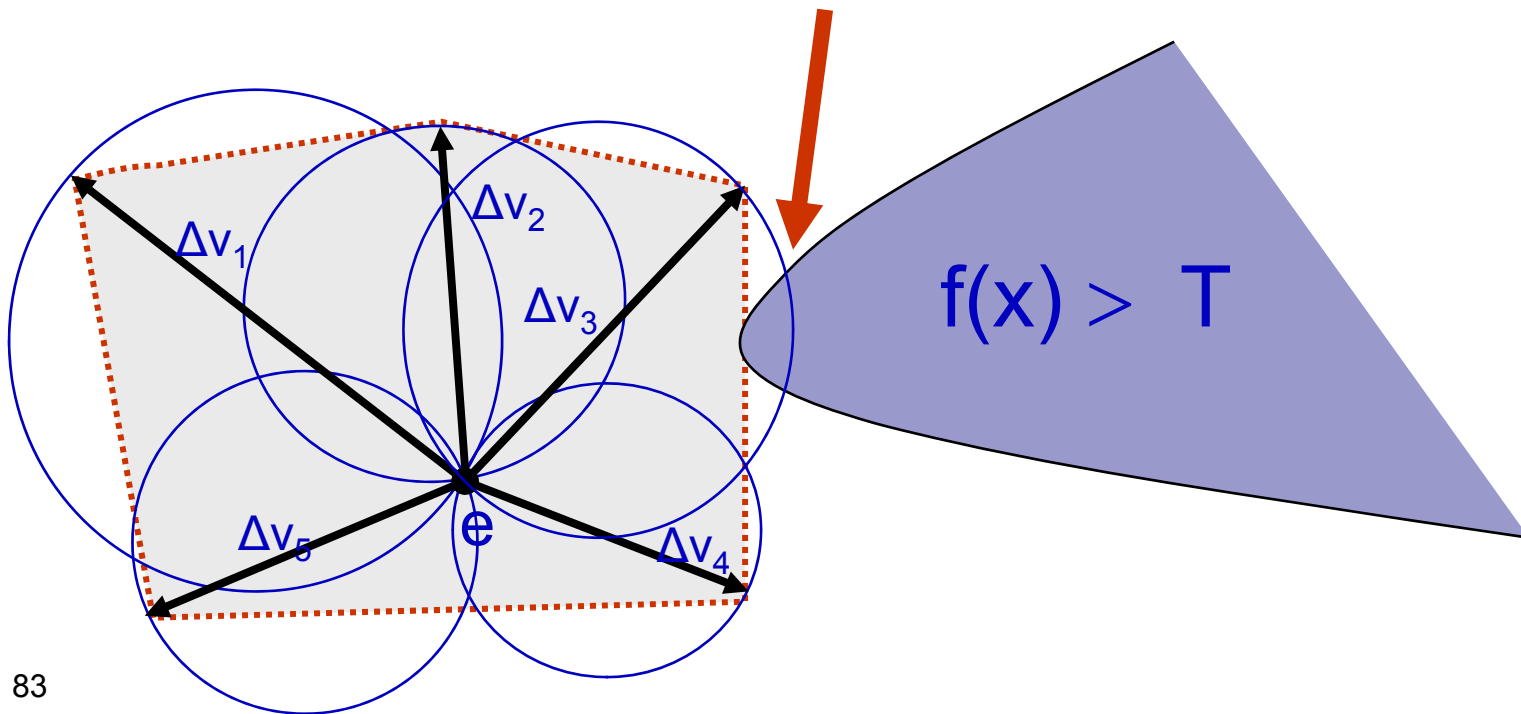
Monitoring General Threshold Functions

- *Monochromatic Region*: For all points x in the region $f(x)$ is on the same side of the threshold ($f(x) > T$ or $f(x) \leq T$)
- Each site *independently* checks its ball is monochromatic
 - Find **max** and **min** for $f()$ in local ball region (may be costly)
 - Broadcast updated value of v_i if not monochrome



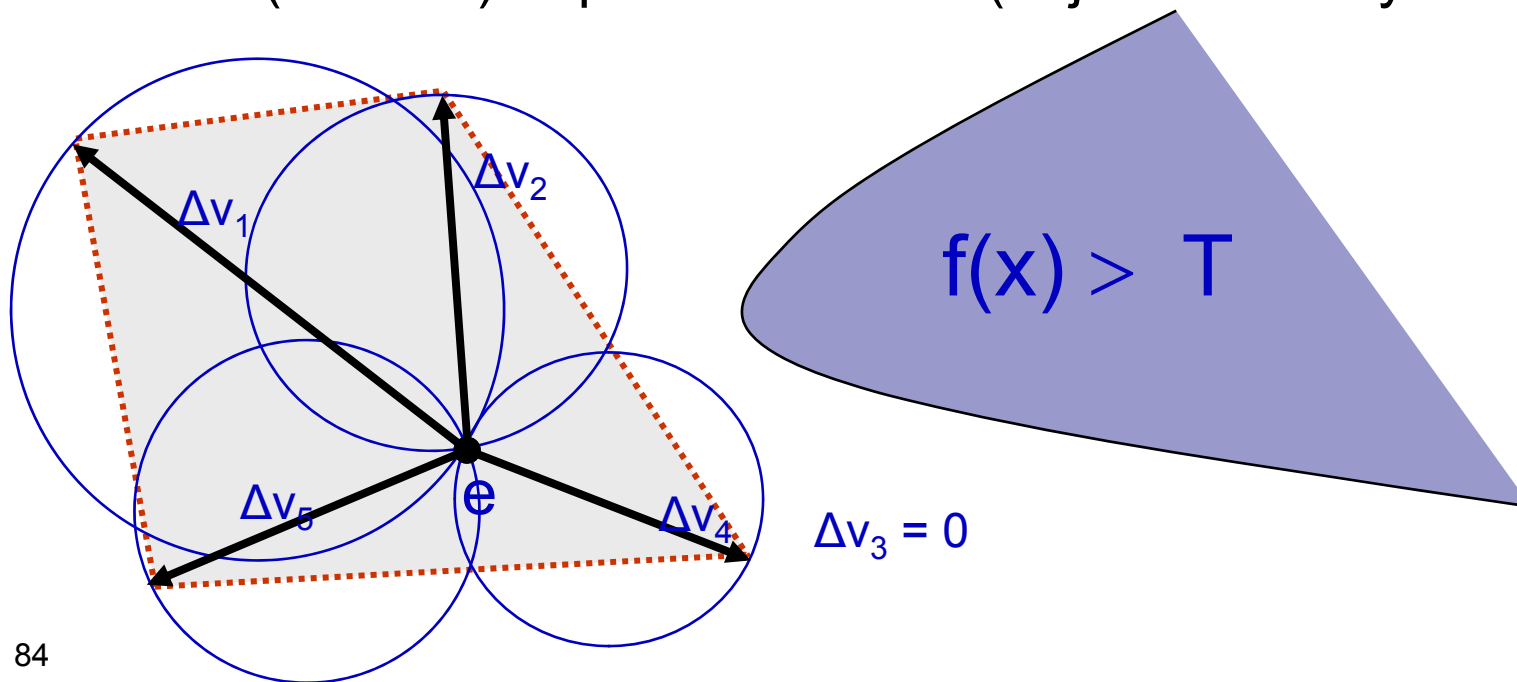
Monitoring General Threshold Functions

- After broadcast, $\|\Delta v_i\|_2 = 0 \Rightarrow$ Ball at i is monochromatic



Monitoring General Threshold Functions

- After broadcast, $\|\Delta v_i\|_2 = 0 \Rightarrow$ Ball at i is monochromatic
 - Global estimate e is updated, which may cause more site update broadcasts
- Coordinator can allocate local slack vectors to sites to enable “localized” resolutions
 - Drift (=radius) depends on slack (adjusted locally for subsets)



Extensions I

- Extending the Geometric Method to incorporate
 - Prediction Models [SIGMOD'12]
 - Sketches [In Submission]
- Showing how various sensornet monitoring and distributed query processing tasks can be expressed in GM [VLDB'11, ICDE'12]
- More general theory of “Safe Zones” for CDM
 - Arbitrary convex subsets of the “admissible region” [TKDE'12]
- Lots of ongoing work on applications to data mining/
distributed learning, complex queries, impact on privacy, ...

Extensions II

- Lots of research interest around CDM
- **Theory and Algorithms:** Lower bounds for CDM, (Near) Optimal algorithms for several problems (e.g., sampling, counts, ranks)
- **Systems:** Interactions between CDM and traditional “big data” processing engines (e.g., Hadoop, S4)
- See proceedings of recent *NII Shonan Workshop* in Japan (December'2011)



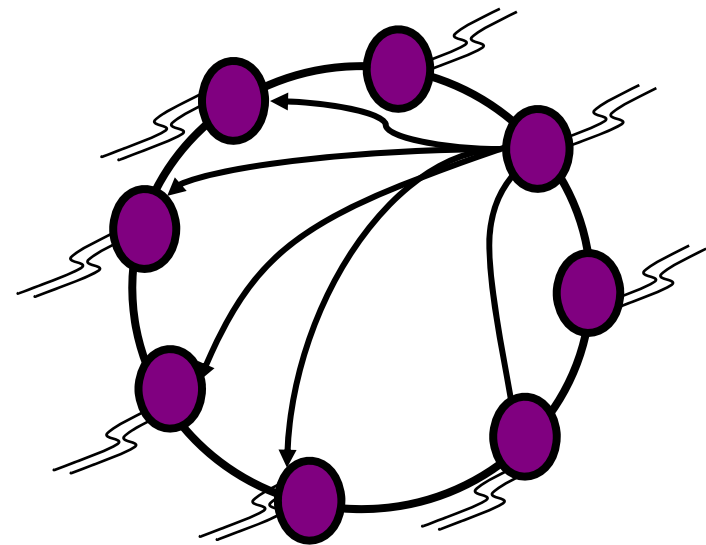
Future Directions

Problem I: Exploring the Prediction Model Space

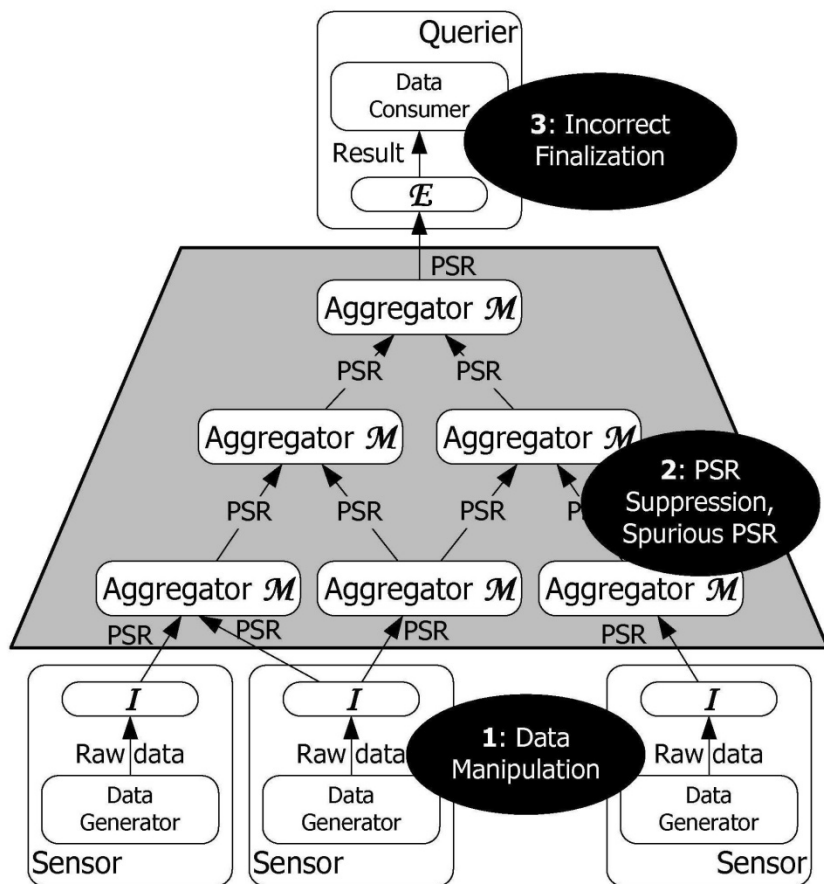
- The better we can capture and anticipate future stream direction, the less communication is needed
- So far, only look at predicting each stream alone
- Correlation/anti-correlation across streams should help?
 - But then, checking validity of model is expensive!
- Explore tradeoff-between power (expressiveness) of model and complexity (number of parameters)
 - Optimization via Minimum Description Length (MDL)?
[Rissanen 1978]

Problem II: CDM in Scalable Network Architectures

- E.g., DHT-based P2P networks
- Single query point
 - “Unfolding” the network gives a hierarchy
 - But, single point of failure (i.e., root)
- Decentralized monitoring
 - Everyone participates in computation, all get the result
 - Exploit epidemics? Latency might be problematic...



Problem III: Secure/Private CDM



- Wide-area query processing
 - Possible *malicious aggregators*
 - Can suppress or add spurious information
- Authenticate query results at the querier?
 - Perhaps, to within some approximation error
- Initial steps for secure aggregation in [Garofalakis et al. ICDE'07, Papadopoulos et al. ICDE'12]

Problem Laundry List

- Lower bounds for general function monitoring using the GM or SZs?
- Can GM/SZs be extended for non-L2 spaces (e.g., Hamming)?
- Apply/extend approach and ideas to general (not necessarily aggregate) queries?

Conclusions

- Many new “big data” problems posed by developing technologies
- Common features of *distributed streams* allow for general techniques/principles instead of “point” solutions
 - Space-, Time-efficient Sketching
 - In-network query processing
 - Models of “normal” operation
 - Exploiting network locality and avoiding global resyncs
- Many new (theoretical and practical) directions unstudied, more will emerge as new technologies arise
- Interactions with “big data” engines (e.g., Hadoop, S4) is being explored
- *Lots of exciting research to be done!* ☺

Thank you!



<http://www.lift-eu.org/>
<http://www.softnet.tuc.gr/~minos/>