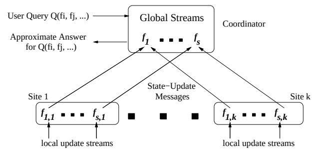
Streaming in a Connected World:
Querying and Tracking Distributed Data Streams



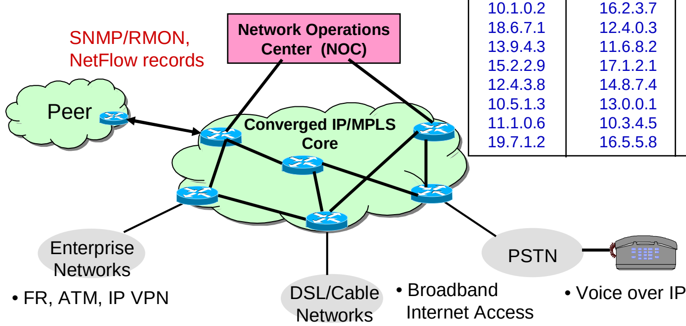
Graham Cormode
AT&T Labs - Research
graham@research.att.com

Minos Garofalakis
Yahoo! Research & UC Berkeley
minos@acm.org

Streams – A Brave New World

- 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
  - Sensor networks
  - Telecom call-detail records
  - Network security
  - Financial applications
  - Manufacturing processes
  - Web logs and clickstreams
  - Other massive data sets…

IP Network Monitoring Application

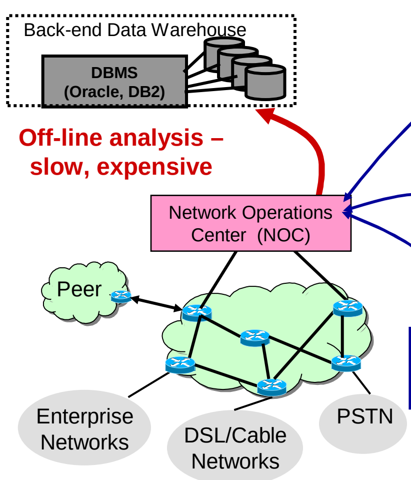


Source	Destination	Duration	Bytes	Protocol
10.1.0.2	16.2.3.7	12	20K	http
18.6.7.1	12.4.0.3	16	24K	http
13.9.4.3	11.6.8.2	15	20K	http
15.2.2.9	17.1.2.1	19	40K	http
12.4.3.8	14.8.7.4	26	58K	http
10.5.1.3	13.0.0.1	27	100K	ftp
11.1.0.6	10.3.4.5	32	300K	ftp
19.7.1.2	16.5.5.8	18	80K	ftp

Example NetFlow IP Session Data

- 24x7 IP packet/flow data-streams at network elements
- Truly massive streams arriving at rapid rates
  - AT&T collects 600-800 Gigabytes of NetFlow data each day.
- Often shipped off-site to data warehouse for off-line analysis

Network Monitoring Queries



What are the top (most frequent) 1000 (source, dest) pairs seen over the last month?

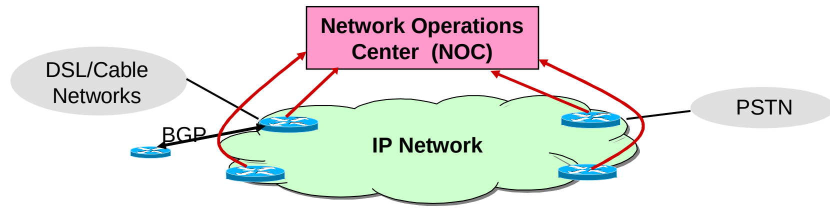
How many distinct (source, dest) pairs have been seen by both R1 and R2 but not R3?

Set-Expression Query

```
SELECT COUNT (R1.source, R2.dest)
FROM R1, R2
WHERE R1.dest = R2.source
```

SQL Join Query

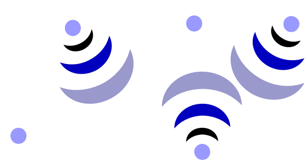
Real-Time Data-Stream Analysis



- Must process network streams in *real-time* and *one pass*
- Critical NM tasks: fraud, DoS attacks, SLA violations
  - Real-time traffic engineering to improve utilization
- Tradeoff communication and computation to reduce load
  - Make responses fast, minimize use of network resources
  - Secondarily, minimize space and processing cost at nodes

Sensor Networks

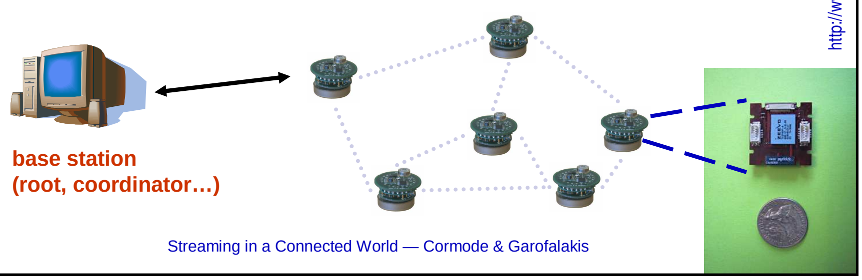
- Wireless sensor networks becoming ubiquitous in environmental monitoring, military applications, …
- Many (100s, 10^3, 10^6?) sensors scattered over terrain
- Sensors observe and process a local stream of readings:
  - Measure light, temperature, pressure…
  - Detect signals, movement, radiation…
  - Record audio, images, motion…



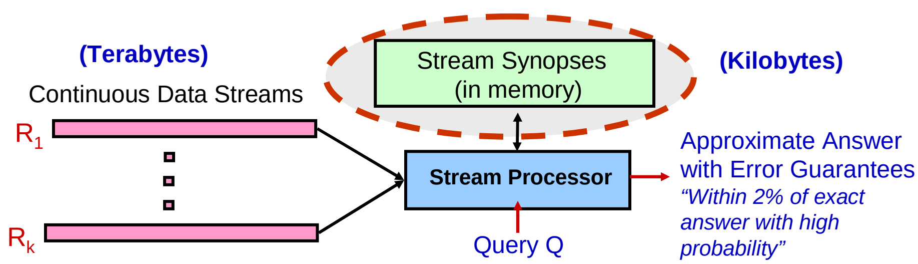
Sensornet Querying Application

- Query sensornet through a (remote) *base station*
- Sensor nodes have severe resource constraints
  - Limited battery power, memory, processor, radio range…
  - *Communication* is the major source of battery drain
  - "transmitting a single bit of data is equivalent to 800 instructions" [Madden et al.'02]

http://www.intel.com/research/exploratory/motes.htm

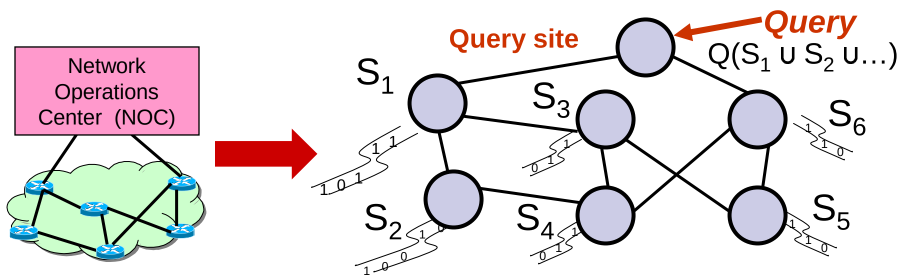


Data-Stream Algorithmics Model



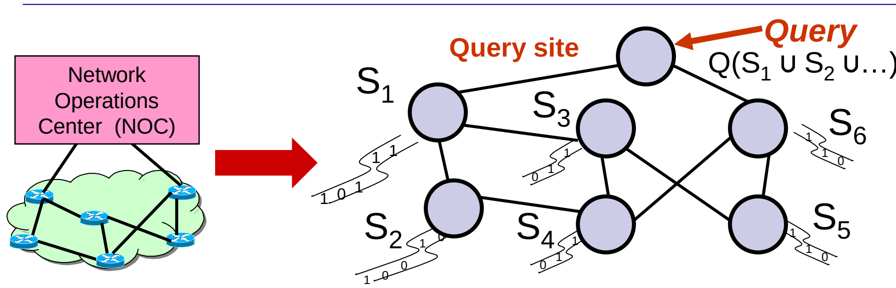
- *Approximate answers*– e.g. trend analysis, anomaly detection
- Requirements for stream synopses
  - *Single Pass:* Each record is examined at most once
  - *Small Space:* Log or polylog in data stream size
  - *Small-time:* Low per-record processing time (maintain synopses)
  - Also: *delete-proof,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

Distributed Streams Model



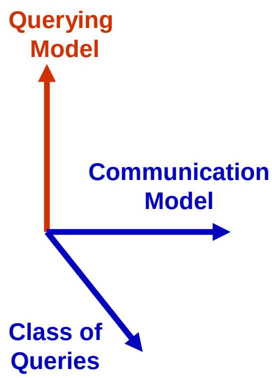
- Large-scale querying/monitoring: *Inherently distributed!*
  - Streams physically distributed across remote sites
    E.g., stream of UDP packets through subset of edge routers
- *Challenge is "holistic" querying/monitoring*
  - Queries over the *union of distributed streams* $Q(S_1 \cup S_2 \cup \ldots)$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

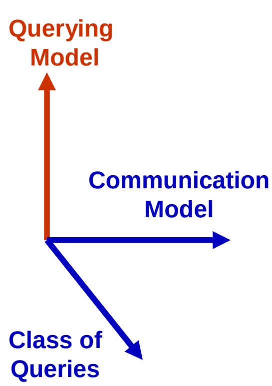
Distributed Stream Querying Space



"One-shot" vs. Continuous Querying

- One-shot queries: On-demand "pull" query answer from network
  - One or few rounds of communication
  - Nodes may prepare for a class of queries
- Continuous queries: *Track/monitor* answer at query site *at all times*
  - Detect anomalous/outlier behavior *in (near) real-time,* i.e., "Distributed triggers"
  - Challenge is to minimize communication
    Use "push-based" techniques
    May use one-shot algs as subroutines

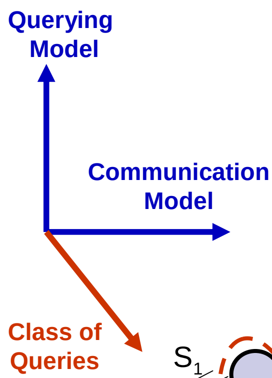
Distributed Stream Querying Space



Minimizing communication often needs approximation and randomization

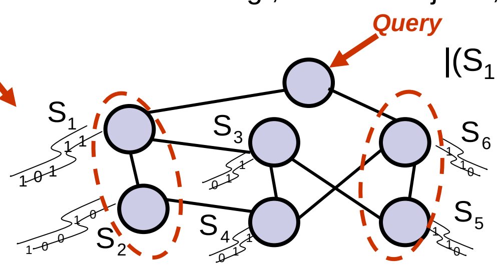
- E.g., Continuously monitor average value
  - Must send every change for exact answer
  - Only need 'significant' changes for approx (def. of "significant" specifies an algorithm)
- Probability sometimes vital to reduce communication
  - `count distinct` in one shot model needs randomness
  - Else **must** send complete data

Distributed Stream Querying Space



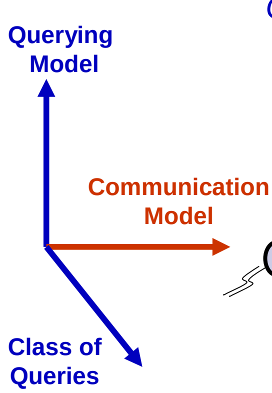
Class of Queries of Interest

- Simple algebraic vs. holistic aggregates
  - E.g., **`count/max`** vs. quantiles/top-k
- Duplicate-sensitive vs. duplicate-insensitive
  - "Bag" vs. "set" semantics
- Complex correlation queries
  - E.g., distributed joins, set expressions, …



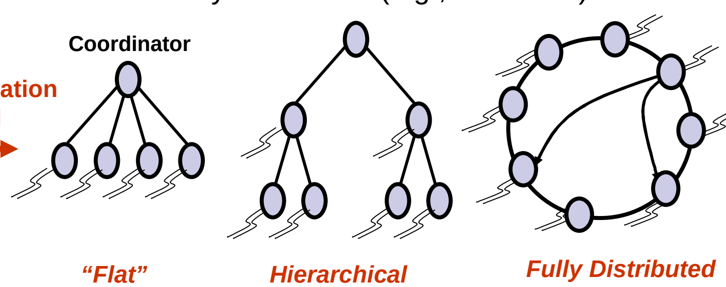
$|(S_1 \cup S_2) \bowtie (S_5 \cup S_6)|$

Distributed Stream Querying Space



Communication Network Characteristics

Topology: "Flat" vs. Hierarchical vs. Fully-distributed (e.g., P2P DHT)



Other network characteristics:

- Unicast (traditional wired), multicast, broadcast (radio nets)
- Node failures, loss, intermittent connectivity, …

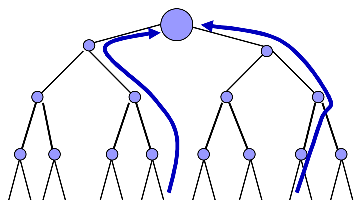
Some Disclaimers…

- We focus on aspects of *physical distribution* of streams
  - Several earlier surveys of (centralized) streaming algorithms and systems
    [Babcock et al.'02; Garofalakis et al.'02; Koudas, Srivastava '03; Muthukrishnan '03] …
- Fairly broad coverage, but still biased view of distributed data-streaming world
  - Revolve around personal biases (line of work and interests)
  - Main focus on key algorithmic concepts, tools, and results
    - Only minimal discussion of systems/prototypes
  - A lot more out there, esp. on related world of sensornets
    [Madden '06]

Tutorial Outline

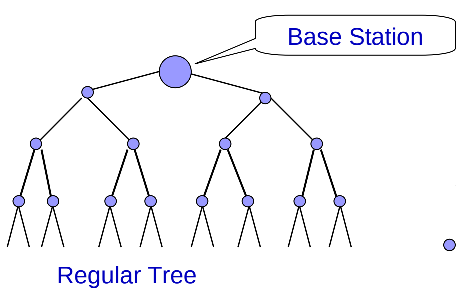
- Introduction, Motivation, Problem Setup
- One-Shot Distributed-Stream Querying
  - Tree Based Aggregation
  - Robustness and Loss
  - Decentralized Computation and Gossiping
- Continuous Distributed-Stream Tracking
- Probabilistic Distributed Data Acquisition
- Future Directions & Open Problems
- Conclusions

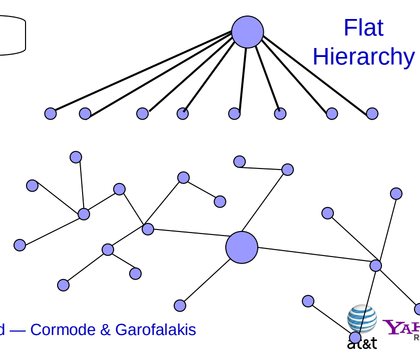
Tree Based Aggregation



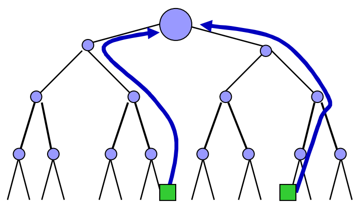
Network Trees

- Tree structured networks are a basic primitive
  - Much work in e.g. sensor nets on building communication trees
  - We assume that tree has been built, focus on issues with a fixed tree





Computation in Trees



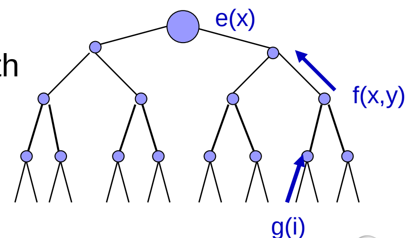
- Goal is for root to compute a function of data at leaves
- Trivial solution: push all data up tree and compute at base station
  - Strains nodes near root: batteries drain, disconnecting network
  - Very wasteful: no attempt at saving communication
- Can do much better by "In-network" query processing
  - Simple example: computing **`max`**
  - Each node hears from all children, computes max and sends to parent (each node sends only one item)

Efficient In-network Computation

- What are aggregates of interest?
  - SQL Primitives: **`min, max, sum, count, avg`**
  - More complex: **`count distinct`**, point & range queries, quantiles, wavelets, histograms, sample
  - Data mining: association rules, clusterings etc.
- Some aggregates are easy – e.g., SQL primitives
- Can set up a formal framework for in network aggregation

Generate, Fuse, Evaluate Framework

- Abstract in-network aggregation. Define functions:
  - **Generate**, g(i): take input, produce summary (at leaves)
  - **Fusion**, f(x,y): merge two summaries (at internal nodes)
  - **Evaluate**, e(x): output result (at root)
- E.g. **`max`**: g(i) = i   f(x,y) = max(x,y)   e(x) = x
- E.g. **`avg`**: g(i) = (i,1)   f((i,j),(k,l)) = (i+k,j+l)   e(i,j) = i/j
- Can specify any function with g(i) ={i}, f(x,y) = x ∪ y
  Want to bound |f(x,y)|



Classification of Aggregates

- Different properties of aggregates
  (from TAG paper [Madden et al '02])
  - Duplicate sensitive – is answer same if multiple identical values are reported?
  - Example or summary – is result some value from input (**`max`**) or a small summary over the input (**`sum`**)
  - Monotonicity – is F(X ∪ Y) monotonic compared to F(X) and F(Y) (affects push down of selections)
  - Partial state – are |g(x)|, |f(x,y)| constant size, or growing? Is the aggregate *algebraic*, or *holistic*?

Classification of some aggregates

	Duplicate Sensitive	Example or summary	Monotonic	Partial State
`min, max`	No	Example	Yes	algebraic
`sum, count`	Yes	Summary	Yes	algebraic
`average`	Yes	Summary	No	algebraic
median, quantiles	Yes	Example	No	holistic
count distinct	No	Summary	Yes	holistic
sample	Yes	Example(s)	No	algebraic?
histogram	Yes	Summary	No	holistic

adapted from [Madden et al.'02]

Cost of Different Aggregates

Slide adapted from http://db.lcs.mit.edu/madden/html/jobtalk3.ppt

Simulation Results

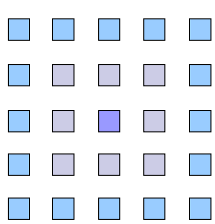
2500 Nodes

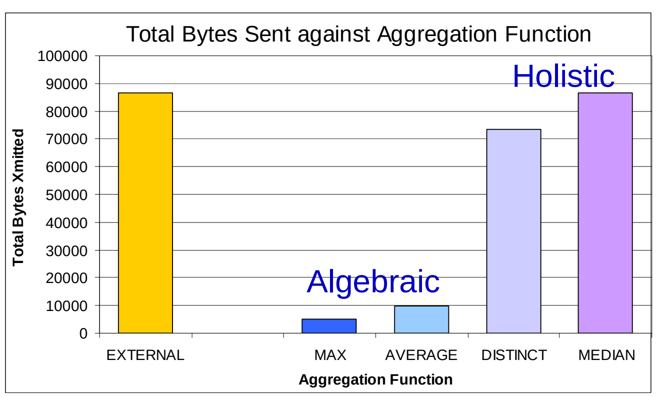
50x50 Grid

Depth = ~10

Neighbors = ~20

Uniform Dist.



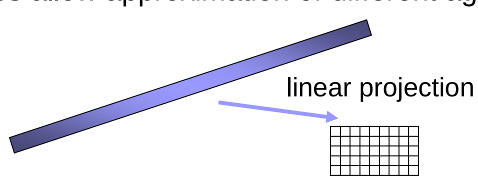


Holistic Aggregates

- Holistic aggregates need the whole input to compute (no summary suffices)
  - E.g., **`count distinct`**, need to remember all distinct items to tell if new item is distinct or not
- So focus on approximating aggregates to limit data sent
  - Adopt ideas from sampling, data reduction, streams etc.
- Many techniques for in-network aggregate approximation:
  - Sketch summaries
  - Other mergable summaries
  - Building uniform samples, etc…

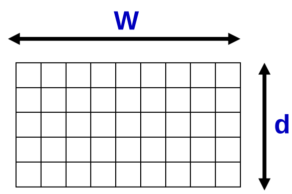
Sketch Summaries

- Sketch summaries are typically pseudo-random linear projections of data. Fits generate/fuse/evaluate model:
  - Suppose input is vectors x_i and aggregate is $F(\sum_i x_i)$
  - Sketch of x_i, $g(x_i)$, is a matrix product Mx_i
  - Combination of two sketches is their summation: $f(g(x_i),g(x_j)) = M(x_i + x_j) = Mx_i + Mx_j = g(x_i) + g(x_j)$
  - Extraction function $e()$ depends on sketch, different sketches allow approximation of different aggregates.

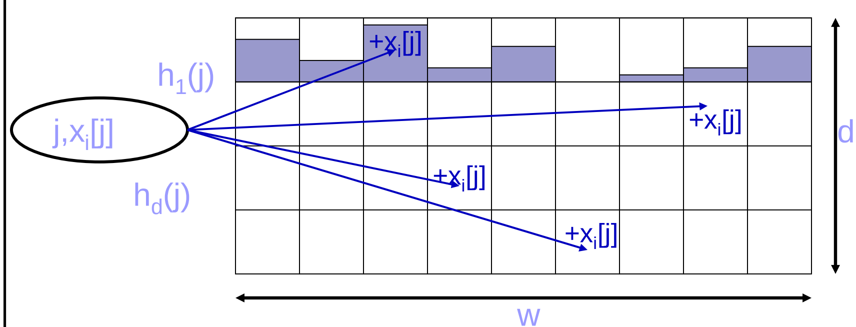


CM Sketch

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



CM Sketch Structure



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

[Cormode, Muthukrishnan '04]

Sketch Summary

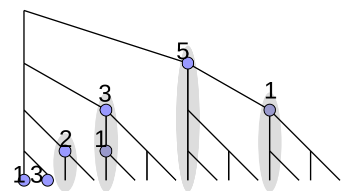
- CM sketch guarantees approximation error on point queries less than $\varepsilon||x||_1$ in size $O(1/\varepsilon \log 1/\delta)$
  - Probability of more error is less than $1-\delta$
  - Similar guarantees for range queries, quantiles, join size
- AMS sketches approximate self-join and join size with error less than $\varepsilon||x||_2\ ||y||_2$ in size $O(1/\varepsilon^2 \log 1/\delta)$
  - [Alon, Matias, Szegedy '96, Alon, Gibbons, Matias, Szegedy '99]
- FM sketches approximate number of distinct items ($||x||_0$) with error less than $\varepsilon||x||_0$ in size $O(1/\varepsilon^2 \log 1/\delta)$
  - FM sketch in more detail later [Flajolet, Martin '83]
- Bloom filters: compactly encode sets in sketch like fashion

Other approaches: Careful Merging

- **Approach 1. Careful merging of summaries**
  - Small summaries of a large amount of data at each site
  - E.g., Greenwald-Khanna algorithm (GK) keeps a small data structure to allow quantile queries to be answered
  - Can sometimes carefully merge summaries up the tree Problem: if not done properly, the merged summaries can grow very large as they approach root
  - Balance final quality of answer against number of merges by decreasing approximation quality (*precision gradient*)
  - See [Greenwald, Khanna '04; Manjhi et al.'05; Manjhi, Nath, Gibbons '05]

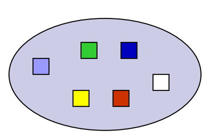
Other approaches: Domain Aware

- **Approach 2. Domain-aware Summaries**
  - Each site sees information drawn from discrete domain [1…U] – e.g. IP addresses, $U = 2^{32}$
  - Build summaries by imposing tree-structure on domain and keeping counts of nodes representing subtrees
  - [Agrawal et al '04] show $O(1/\varepsilon \log U)$ size summary for quantiles and range & point queries
  - Can merge repeatedly without increasing error or summary size



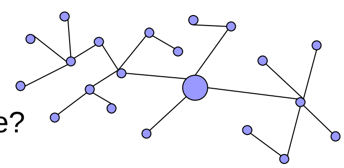
Other approaches: Random Samples

- **Approach 3. Uniform random samples**
  - As in centralized databases, a uniform random sample of size $O(1/\varepsilon^2 \log 1/\delta)$ answers many queries
  - Can collect a random sample of data from each node, and merge up the tree (will show algorithms later)
  - Works for frequent items, quantile queries, histograms
  - No good for count distinct, min, max, wavelets…



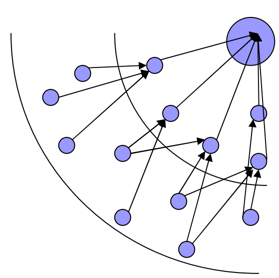
Thoughts on Tree Aggregation

- Some methods too heavyweight for today's sensor nets, but as technology improves may soon be appropriate
- Most are well suited for, e.g., wired network monitoring
  - Trees in wired networks often treated as flat, i.e. send directly to root without modification along the way
- Techniques are fairly well-developed owing to work on data reduction/summarization and streams
- Open problems and challenges:
  - Improve size of larger summaries
  - Avoid randomized methods?
    Or use randomness to reduce size?



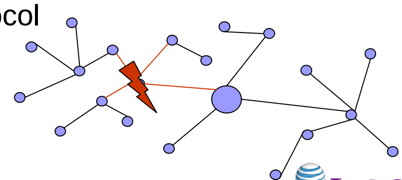
33 Streaming in a Connected World — Cormode & Garofalakis

Robustness and Loss



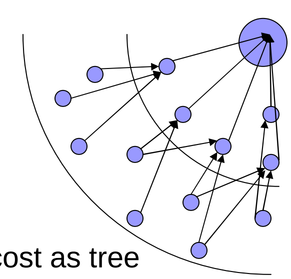
Unreliability

- Tree aggregation techniques assumed a reliable network
  - we assumed no node failure, nor loss of any message
- Failure can dramatically affect the computation
  - E.g., `sum` – if a node near the root fails, then a whole subtree may be lost
- Clearly a particular problem in sensor networks
  - If messages are lost, maybe can detect and resend
  - If a node fails, may need to rebuild the whole tree and re-run protocol
  - Need to detect the failure, could cause high uncertainty



Sensor Network Issues

- Sensor nets typically based on radio communication
  - So broadcast (within range) cost the same as unicast
  - Use multi-path routing: improved reliability, reduced impact of failures, less need to repeat messages
- E.g., computation of `max`
  - structure network into rings of nodes in equal hop count from root
  - listen to all messages from ring below, then send max of all values heard
  - converges quickly, high path diversity
  - each node sends only once, so same cost as tree

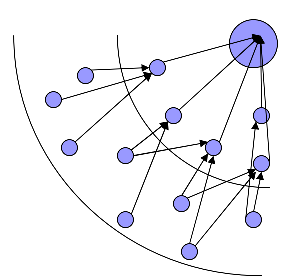


Order and Duplicate Insensitivity

- It works because **max** is Order and Duplicate Insensitive (ODI) [Nath et al.'04]
- Make use of the same e(), f(), g() framework as before
- Can prove correct if e(), f(), g() satisfy properties:
  - g gives same output for duplicates: $i=j \Rightarrow g(i) = g(j)$
  - f is associative and commutative:
    $f(x,y) = f(y,x)$; $f(x,f(y,z)) = f(f(x,y),z)$
  - f is same-synopsis idempotent: $f(x,x) = x$
- Easy to check **min**, **max** satisfy these requirements, **sum** does not

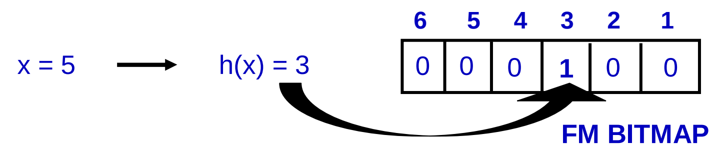
Applying ODI idea

- Only **max** and **min** seem to be "naturally" ODI
- How to make ODI summaries for other aggregates?
- Will make use of duplicate insensitive primitives:
  - Flajolet-Martin Sketch (FM)
  - Min-wise hashing
  - Random labeling
  - Bloom Filter



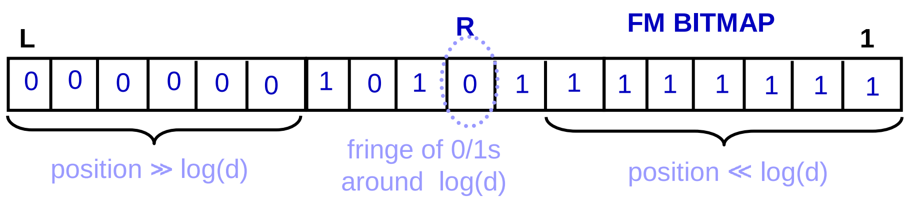
FM Sketch

- Estimates number of distinct inputs (**`count distinct`**)
- Uses hash function mapping input items to i with prob 2^{-i}
  - i.e. Pr[h(x) = 1] = ½, Pr[h(x) = 2] = ¼, Pr[h(x)=3] = 1/8 …
  - Easy to construct h() from a uniform hash function by counting trailing zeros
- Maintain FM Sketch = bitmap array of L = log U bits
  - Initialize bitmap to all 0s
  - For each incoming value x, set FM[h(x)] = 1



FM Analysis

- If d distinct values, expect d/2 map to FM[1], d/4 to FM[2]…



  - Let R = position of rightmost zero in FM, indicator of log(d)
  - Basic estimate $d = c2^R$ for scaling constant $c \approx 1.3$
  - Average many copies (different hash fns) improves accuracy

FM Sketch – ODI Properties

6	5	4	3	2	1		6	5	4	3	2	1		6	5	4	3	2	1
0	0	1	0	1	1	+	0	1	1	0	0	1	=	0	1	1	0	1	1

- Fits into the Generate, Fuse, Evaluate framework.
  - Can fuse multiple FM summaries (with same hash h()): take bitwise-OR of the summaries
- With $O(1/\varepsilon^2 \log 1/\delta)$ copies, get $(1\pm\varepsilon)$ accuracy with probability at least $1-\delta$
  - 10 copies gets ≈ 30% error, 100 copies < 10% error
  - Can pack FM into eg. 32 bits. Assume h() is known to all.
- Similar ideas used in [Gibbons, Tirthapura '01]
  - improves time cost to create summary, simplifies analysis

FM within ODI

- What if we want to count, not count distinct?
  - E.g., each site i has a count c_i, we want $\sum_i c_i$
  - Tag each item with site ID, write in unary: $(i,1), (i,2)\ldots (i,c_i)$
  - Run FM on the modified input, and run ODI protocol
- What if counts are large?
  - Writing in unary might be too slow, need to make efficient
  - [Considine et al.'05]: simulate a random variable that tells which entries in sketch are set
  - [Aduri, Tirthapura '05]: allow range updates, treat (i,c_i) as range.

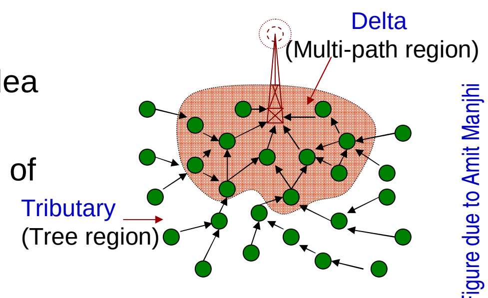
Other applications of FM in ODI

- Can take sketches and other summaries and make them ODI by replacing counters with FM sketches
  - CM sketch + FM sketch = CMFM, ODI point queries etc. [Cormode, Muthukrishnan '05]
  - Q-digest + FM sketch = ODI quantiles [Hadjieleftheriou, Byers, Kollios '05]
  - Counts and sums [Nath et al.'04, Considine et al.'05]

6	5	4	3	2	1
0	1	1	0	1	1

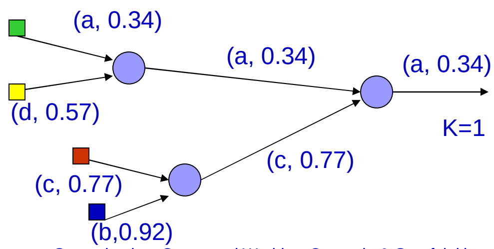
Combining ODI and Tree

- *Tributaries and Deltas* idea [Manjhi, Nath, Gibbons '05]
- Combine small synopsis of tree-based aggregation with reliability of ODI
  - Run tree synopsis at edge of network, where connectivity is limited (tributary)
  - Convert to ODI summary in dense core of network (delta)
  - Adjust crossover point adaptively



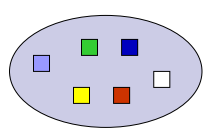
Random Samples

- Suppose each node has a (multi)set of items.
- How to find a random sample of the union of all sets?
- Use a “random tagging” trick [Nath et al.’05]:
  - For each item, attach a random label in range [0…1]
  - Pick the items with the K smallest labels to send
  - Merge all received items, and pick K smallest labels



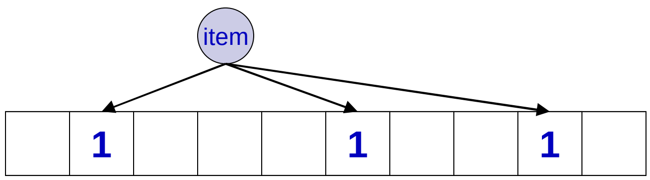
Uniform random samples

- Result at the coordinator:
  - A sample of size K items from the input
  - Can show that the sample is chosen uniformly at random without replacement (could make “with replacement”)
- Related to min-wise hashing
  - Suppose we want to sample from distinct items
  - Then replace random tag with hash value on item name
  - Result: uniform sample from set of present items
- Sample can be used for quantiles, frequent items etc.



Bloom Filters

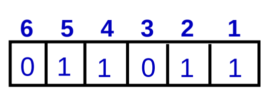
- Bloom filters compactly encode set membership
  - k hash functions map items to bit vector k times
  - Set all k entries to **1** to indicate item is present
  - Can lookup items, store set of size n in ~ 2n bits



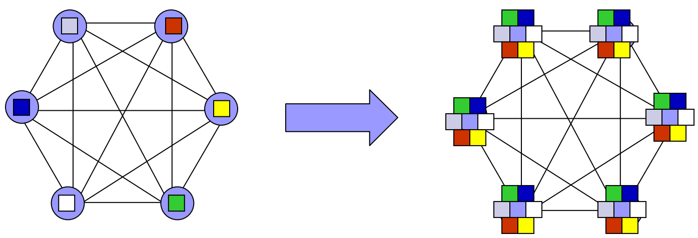
- Bloom filters are ODI, and merge like FM sketches

Open Questions and Extensions

- Characterize all queries – can everything be made ODI with small summaries?
- How practical for different sensor systems?
  - Few FM sketches are very small (10s of bytes)
  - Sketch with FMs for counters grow large (100s of KBs)
  - What about the computational cost for sensors?
- Amount of randomness required, and implicit coordination needed to agree hash functions etc.?
- Other implicit requirements: unique sensor IDs?

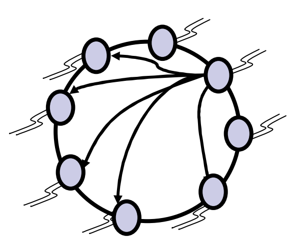


Decentralized Computation and Gossiping



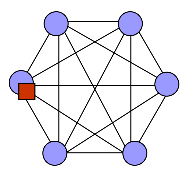
Decentralized Computations

- All methods so far have a single point of failure: if the base station (root) dies, everything collapses
- An alternative is Decentralized Computation
  - Everyone participates in computation, all get the result
  - Somewhat resilient to failures / departures
- Initially, assume anyone can talk to anyone else directly



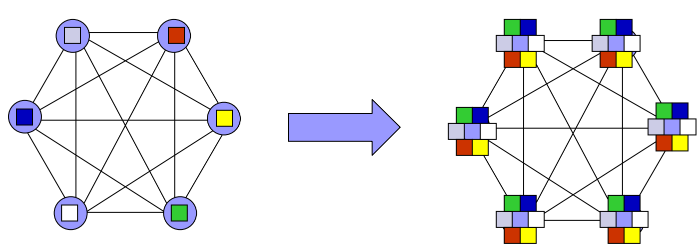
Gossiping

- "Uniform Gossiping" is a well-studied protocol for spreading information
  - I know a secret, I tell two friends, who tell two friends …
  - Formally, each round, everyone who knows the data sends it to one of the n participants chosen at random
  - After O(log n) rounds, all n participants know the information (with high probability) [Pittel 1987]



Aggregate Computation via Gossip

- Naïve approach: use uniform gossip to share all the data, then everyone can compute the result.
  - Slightly different situation: gossiping to exchange n secrets
  - Need to store all results so far to avoid double counting
  - Messages grow large: end up sending whole input around

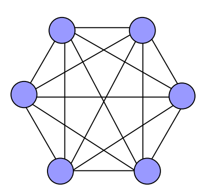


ODI Gossiping

- If we have an ODI summary, we can gossip with this.
  - When new summary received, merge with current summary
  - ODI properties ensure repeated merging stays accurate
- Number of messages required is same as uniform gossip
  - After O(log n) rounds everyone knows the merged summary
  - Message size and storage space is a single summary
  - O(n log n) messages in total
  - So works for FM, FM-based sketches, samples etc.

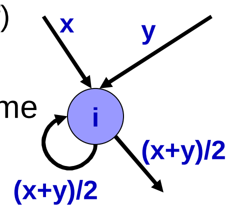
Aggregate Gossiping

- ODI gossiping doesn't always work
  - May be too heavyweight for really restricted devices
  - Summaries may be too large in some cases
- An alternate approach due to [Kempe et al. '03]
  - A novel way to avoid double counting: split up the counts and use "conservation of mass".

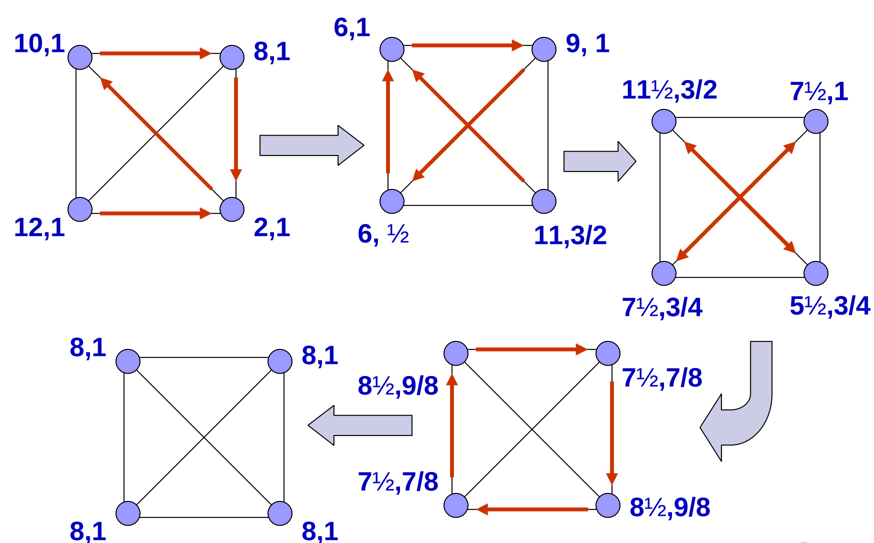


Push-Sum

- Setting: all n participants have a value, want to compute average
- Define "**Push-Sum**" protocol
  - In round t, node i receives set of $(sum_j^{t-1}, count_j^{t-1})$ pairs
  - Compute $sum_i^t = \sum_j sum_j^{t-1}$, $count_i^t = \sum_j count_j$
  - Pick k uniformly from other nodes
  - Send $(\frac{1}{2} sum_i^t, \frac{1}{2} count_i^t)$ to k and to i (self)
- Round zero: send (value,1) to self
- Conservation of counts: $\sum_i sum_i^t$ stays same
- Estimate **avg** $= sum_i^t / count_i^t$



Push-Sum Convergence

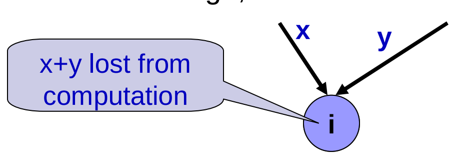


Convergence Speed

- Can show that after $O(\log n + \log 1/\varepsilon + \log 1/\delta)$ rounds, the protocol converges within ε
  - n = number of nodes
  - ε = (relative) error
  - δ = failure probability
- Correctness due in large part to conservation of counts
  - Sum of values remains constant throughout
  - (Assuming no loss or failure)

Resilience to Loss and Failures

- Some resilience comes for "free"
  - If node detects message was not delivered, delay 1 round then choose a different target
  - Can show that this only increases number of rounds by a small constant factor, even with many losses
  - Deals with message loss, and "dead" nodes without error
- If a node fails during the protocol, some "mass" is lost, and count conservation does not hold
  - If the mass lost is not too large, error is bounded…



Gossip on Vectors

- Can run **Push-Sum** independently on each entry of vector
- More strongly, generalize to **Push-Vector**:
  - Sum incoming vectors
  - Split sum: half for self, half for randomly chosen target
  - Can prove same conservation and convergence properties
- Generalize to sketches: a sketch is just a vector
  - But ε error on a sketch may have different impact on result
  - Require $O(\log n + \log 1/\varepsilon + \log 1/\delta)$ rounds as before
  - Only store $O(1)$ sketches per site, send 1 per round

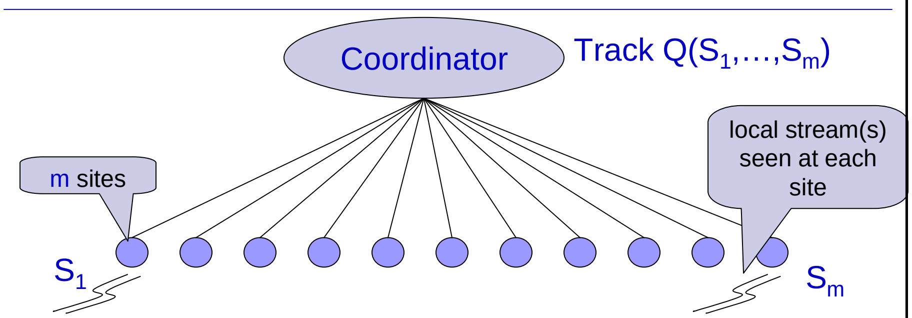
Thoughts and Extensions

- How realistic is complete connectivity assumption?
  - In sensor nets, nodes only see a local subset
  - Variations: spatial gossip ensures nodes hear about local events with high probability [Kempe, Kleinberg, Demers '01]
- Can do better with more structured gossip, but impact of failure is higher [Kashyap et al.'06]
- Is it possible to do better when only a subset of nodes have relevant data and want to know the answer?

Tutorial Outline

- Introduction, Motivation, Problem Setup
- One-Shot Distributed-Stream Querying
- Continuous Distributed-Stream Tracking
  - Adaptive Slack Allocation
  - Predictive Local-Stream Models
  - Distributed Triggers
- Probabilistic Distributed Data Acquisition
- Future Directions & Open Problems
- Conclusions

Continuous Distributed Model

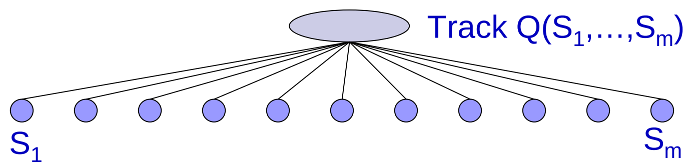


- Other structures possible (e.g., hierarchical)
- Could allow site-site communication, but mostly unneeded

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!

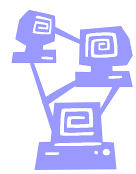


Challenges

- 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

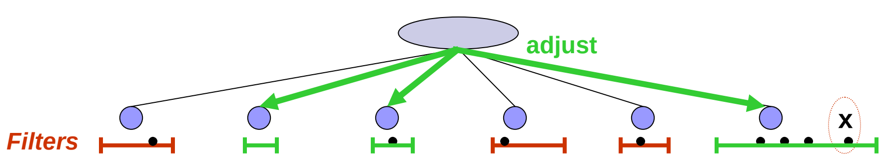
How about Periodic Polling?

- 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

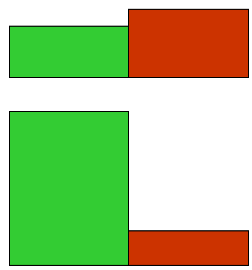


Communication-Efficient Monitoring

- Exact answers are not needed
  - Approximations with accuracy guarantees suffice
  - Tradeoff *accuracy* and *communication/ processing cost*
- Key Insight: *"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
  - Coordinator sets/adjusts local filters to guarantee accuracy



Adaptive Slack Allocation



Slack Allocation

- A key idea is Slack Allocation
- Because we allow approximation, there is slack: the tolerance for error between computed answer and truth
  - May be absolute: $|Y - \hat{Y}| \leq \varepsilon$: slack is ε
  - Or relative: $\hat{Y} / Y \leq (1 \pm \varepsilon)$: slack is εY
- For a given aggregate, show that the slack can be divided between sites
- Will see different slack division heuristics

Top-k Monitoring

- Influential work on monitoring [Babcock, Olston'03]
  - Introduces some basic heuristics for dividing slack
  - Use local offset parameters so that all local distributions look like the global distribution
  - Attempt to fix local slack violations by negotiation with coordinator before a global readjustment
  - Showed that message delay does not affect correctness

Billboard
Top 100

Images from http://www.billboard.com

Top-k Scenario

- Each site monitors n objects with local counts $V_{i,j}$
  - item $i \in [n]$
  - site $j \in [m]$
- Values change over time with updates seen at site j
- Global count $V_i = \sum_j V_{i,j}$
- Want to find topk, an ε-approximation to true top-k set:
  - OK provided $i \in$ topk, $l \notin$ topk, $V_i + \varepsilon \geq V_l$

gives a little "wiggle room"

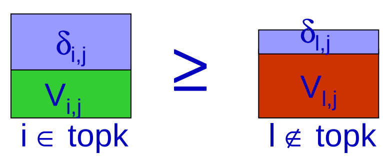
Adjustment Factors

- Define a set of 'adjustment factors', $\delta_{i,j}$
  - Make top-k of $V_{i,j} + \delta_{i,j}$ same as top-k of V_i

- Maintain invariants:
  1. For item i, adjustment factors sum to zero
  2. $\delta_{l,0}$ of non-topk item $l \leq \delta_{i,0} + \varepsilon$ of topk item i
  - Invariants and local conditions used to prove correctness

Local Conditions and Resolution

Local Conditions:
At each site j check adjusted topk counts dominate non-topk

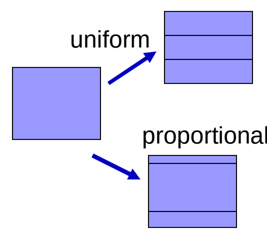


If any local condition violated at site j, resolution is triggered

- Local resolution: site j and coordinator only try to fix
  - Try to "borrow" from $\delta_{i,0}$ and $\delta_{l,0}$ to restore condition
- Global resolution: if local resolution fails, contact all sites
  - Collect all affected $V_{i,j}$s, ie. topk plus violated counts
  - Compute slacks for each count, and reallocate (next)
  - Send new adjustment factors $\delta'_{i,j}$, continue

Slack Division Strategies

- Define "slack" based on current counts and adjustments
- What fraction of slack to keep back for coordinator?
  - $\delta_{i,0} = 0$: No slack left to fix local violations
  - $\delta_{i,0}$ = 100% of slack: Next violation will be soon
  - Empirical setting: $\delta_{i,0}$ = 50% of slack when ε very small
    $\delta_{i,0} = 0$ when ε is large ($\varepsilon > V_i/1000$)
- How to divide remainder of slack?
  - Uniform: $1/m$ fraction to each site
  - Proportional: $V_{i,j}/V_i$ fraction to site j for i



Pros and Cons

- Result has many advantages:
  - Guaranteed correctness within approximation bounds
  - Can show convergence to correct results even with delays
  - Communication reduced by 1 order magnitude
    (compared to sending $V_{i,j}$ whenever it changes by ε/m)
- Disadvantages:
  - Reallocation gets complex: must check $O(km)$ conditions
  - Need $O(n)$ space at each site, $O(mn)$ at coordinator
  - Large ($\approx O(k)$) messages
  - Global resyncs are expensive: m messages to k sites

Other Problems: Aggregate Values

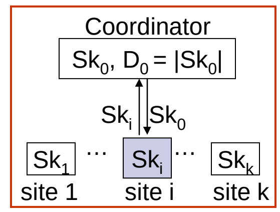
- **Problem 1: Single value tracking**
  Each site has one value v_i, want to compute f(v), e.g., **sum**
- Allow small bound of uncertainty in answer
  - Divide uncertainty (slack) between sites
  - If new value is outside bounds, re-center on new value
- Naïve solution: allocate equal bounds to all sites
  - Values change at different rates; queries may overlap
- Adaptive filters approach [Olston, Jiang, Widom '03]
  - Shrink all bounds and selectively grow others:
    moves slack from stable values to unstable ones
  - Base growth on frequency of bounds violation, optimize

Other Problems: Set Expressions

- **Problem 2: Set Expression Tracking**
  A ∪ (B ∩ C) where A, B, C defined by distributed streams
- Key ideas [Das et al.'04]:
  - Use semantics of set expression: if b arrives in set B, but b already in set A, no need to send
  - Use cardinalities: if many copies of b seen already, no need to send if new copy of b arrives or a copy is deleted
  - Combine these to create a *charging scheme* for each update: if sum of charges is small, no need to send.
  - Optimizing charging is NP-hard, heuristics work well.

Other Problems: ODI Aggregates

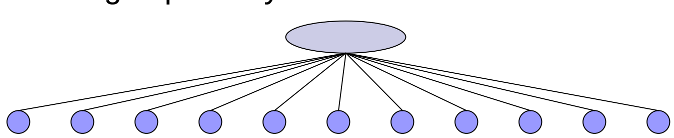
- **Problem 3: ODI aggregates**
  e.g., **`count distinct`** in continuous distributed model
- Two important parameters emerge:
  - How to divide the slack
  - What the site sends to coordinator



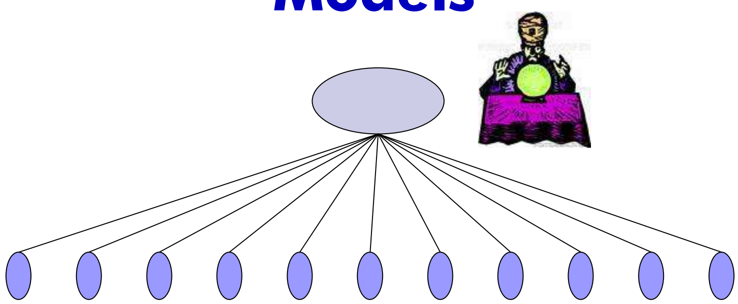
- In [Cormode et al.'06]:
  - Share slack evenly: hard to do otherwise for this aggregate
  - Sharing sketch of global distribution saves communication
  - Better to be lazy: send sketch in reply, don't broadcast

General Lessons

- Break a global (holistic) aggregate into *"safe"* local conditions, so local conditions ⇒ global correctness
- Set local parameters to help the tracking
- Use the approximation to define slack, divide slack between sites (and the coordinator)
- Avoid global reconciliation as much as possible, try to patch things up locally

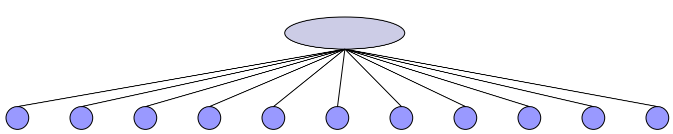


Predictive Local-Stream Models

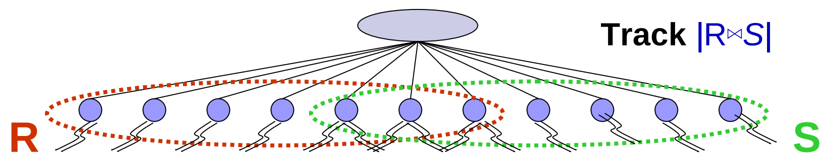


More Sophisticated Local Predictors

- Slack allocation methods use simple *"static"* prediction
  - Site value implicitly assumed constant since last update
  - No update from site ⇒ last update ("predicted" value) is within required slack bounds ⇒ global error bound
- *Dynamic, more sophisticated prediction models* for local site behavior?
  - Model complex stream patterns, reduce number of updates to coordinator
  - But... more complex to maintain and communicate (to coordinator)



Tracking Complex Aggregate Queries



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

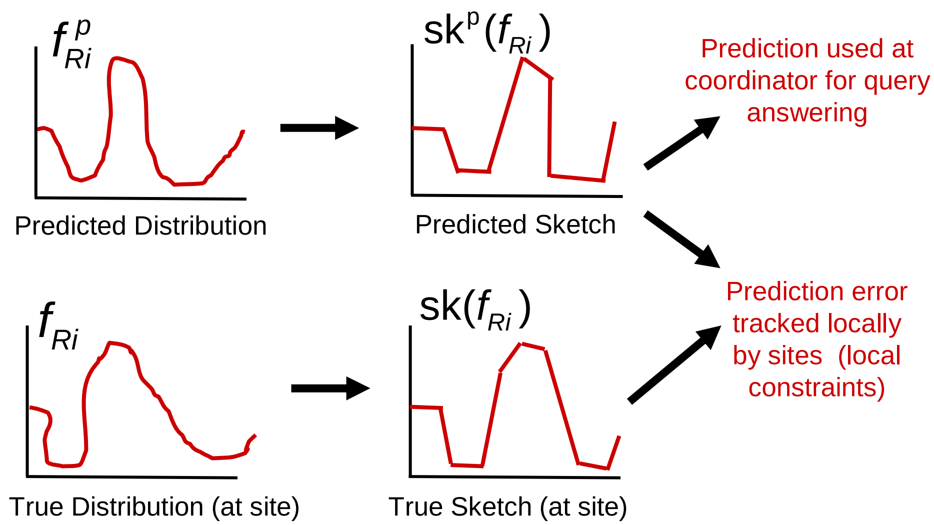
  $|R \bowtie S| = f_R \cdot f_S = \sum_v f_R[v]\, f_S[v]$ $\quad (\pm\, \varepsilon\, ||f_R||_2\, ||f_S||_2)$

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

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
- 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 Scheme

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

– Track local deviation between stream and prediction:

$$\| sk(f_{R,i}) - sk^p(f_{R,i})\|_2 \le \theta/\sqrt{k}\, \| sk(f_{R,i}) \|_2$$

– Send current sketch (and other info) if violated

Querying. At coordinator, query error $\le (\varepsilon + 2\theta)\|f_R\|_2 \, \|f_S\|_2$

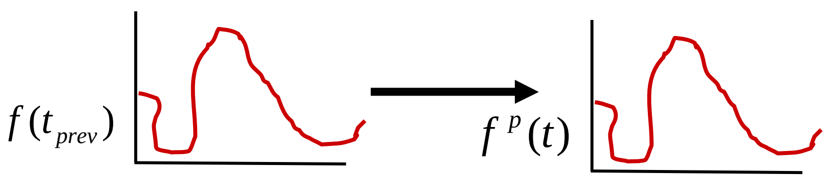
– ε = local-sketch summarization error (at remote sites)

– θ = upper bound on local-stream deviation from prediction ("Lag" between remote-site and coordinator view)

- ***Key Insight:*** *With local deviations bounded, the predicted sketches at coordinator are guaranteed accurate*

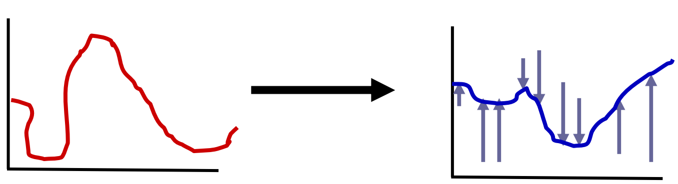
Sketch-Prediction Models

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



$f(t_{prev})$ $\qquad$ $f^p(t) = f(t_{prev}) + \Delta t \cdot v$

Sketch-Prediction Models

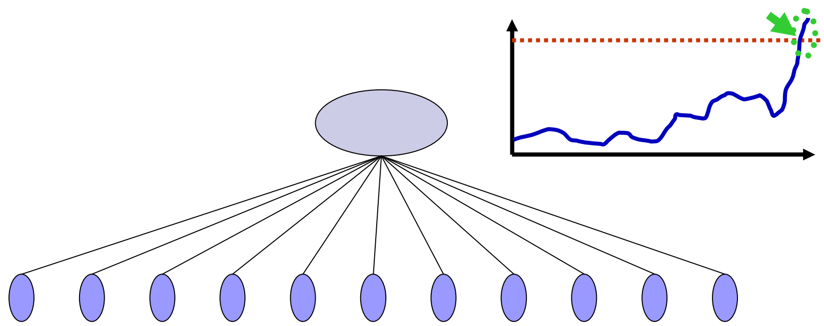
Model	Info	Predicted Sketch
Static	∅	$\text{sk}^{p}(f(t)) = \text{sk}(f(t_{prev}))$
Velocity	sk(v)	$\text{sk}^{p}(f(t)) = \text{sk}(f(t_{prev})) + \Delta t \cdot \text{sk}(v)$

- 1 – 2 orders of magnitude savings over sending all data

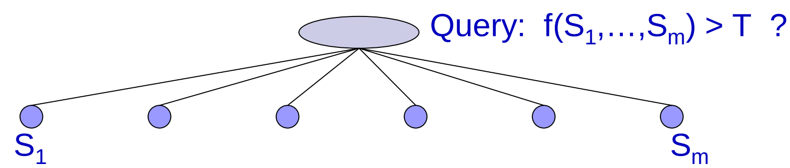
Lessons, Thoughts, and Extensions

- Dynamic prediction models are a natural choice for continuous in-network processing
  - Can capture complex temporal (and spatial) patterns to reduce communication
- Many model choices possible
  - Need to carefully balance power & conciseness
  - Principled way for model selection?
- General-purpose solution (generality of AMS sketch)
  - Better solutions for special queries
    E.g., continuous quantiles [Cormode et al.'05]

Distributed Triggers

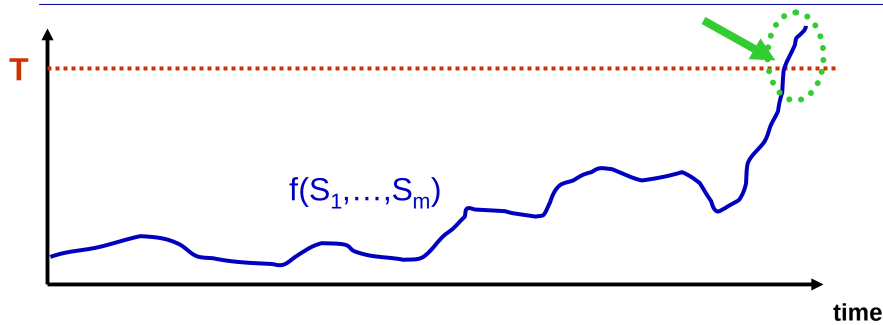


Tracking Distributed Triggers



- Only interested in values of the “global query” above a certain threshold T
  - Network anomaly detection (e.g., DDoS attacks)
    - Total number of connections to a destination, “fire” when it exceeds a threshold
  - Air / water quality monitoring, total number of cars on highway
    - Fire when count/average exceeds a certain amount
- Introduced in HotNets paper [Jain, Hellerstein et al.’04]

Tracking Distributed Triggers



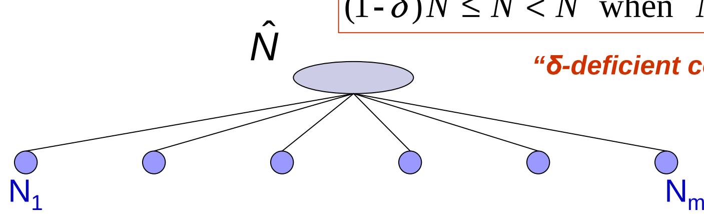
- Problem “easier” than approximate query tracking
  - Only want accurate f() values when they’re close to threshold
  - *Exploit threshold for intelligent slack allocation to sites*
- Push-based in-network operation even more relevant
  - Optimize operation for “common case”

Tracking Thresholded Counts

- Monitor a distributed aggregate count
- Guarantee a user-specified accuracy δ *only if the count exceeds a pre-specified threshold* T [Kerlapura et al.’06]
  - E.g., N_i = number of observed connections to 128.105.7.31 and $N = \sum_i N_i$

$$0 \le \hat{N} < T \quad \text{when} \quad N < T$$
$$(1-\delta)N \le \hat{N} < N \quad \text{when} \quad N \ge T$$

“δ-deficient counts”

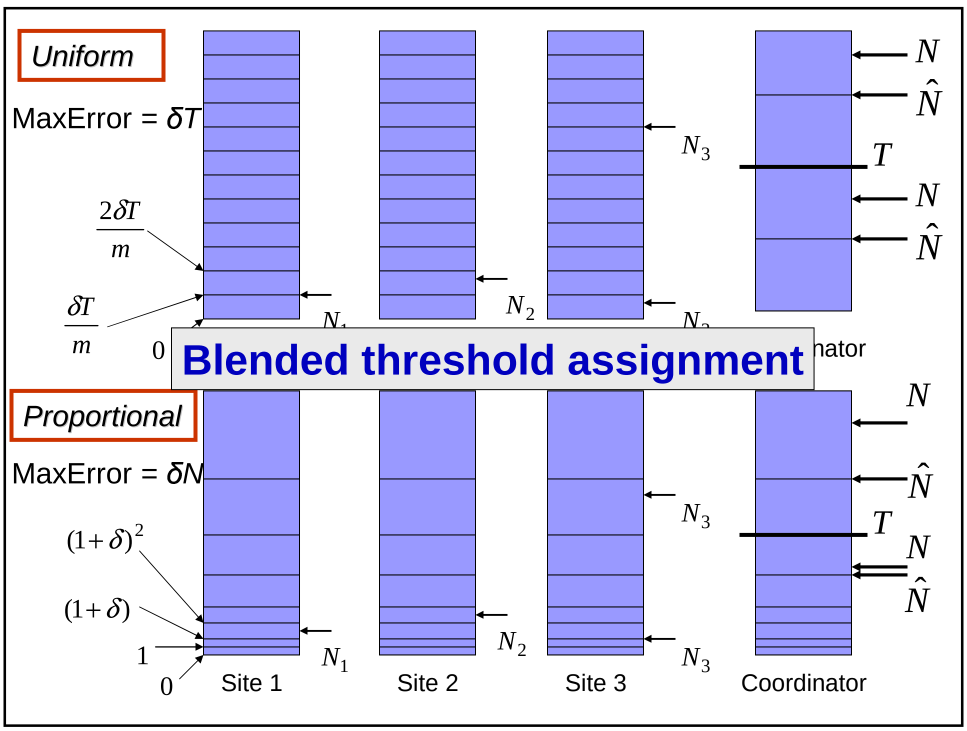


Thresholded Counts Approach

- Site i maintains a set of local thresholds $t_{i,j}$, j= 0, 1, 2, …
- Local filter at site i: $t_{i,f(i)} \leq N_i < t_{i,f(i)+1}$
  - Local count between adjacent thresholds
  - Contact coordinator with new "level" f(i) when violated
- Global estimate at coordinator $\hat{N} = \sum_i t_{i,f(i)}$
- For δ-deficient estimate, choose local threshold sequences $t_{i,j}$ such that

$$\sum_i (t_{i,f(i)+1} - t_{i,f(i)}) < \delta \sum_i t_{i,f(i)} \quad \text{whenever} \quad \sum_i t_{i,f(i)+1} > T$$

"large" to minimize communication!
"small" to ensure global error bound!



Blended Threshold Assignment

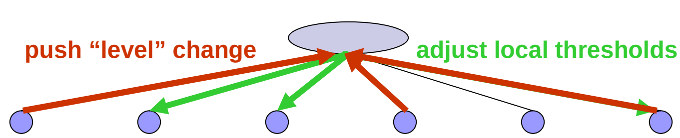
- Uniform: overly tight filters when $N > T$
- Proportional: overly tight filters when $N \ll T$
- **Blended Assignment**: combines best features of both:

  $t_{i,j+1} = (1+\alpha\delta)\cdot t_{i,j} + (1-\alpha)\cdot\delta T/m$ where $\alpha \in [0,1]$

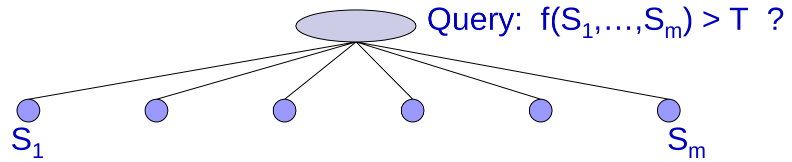
  - $\alpha = 0 \Rightarrow$ *Uniform assignment*
  - $\alpha = 1 \Rightarrow$ *Proportional assignment*
- Optimal value of α exists for given N (expected or distribution)
  - Determined through, e.g., gradient descent

Adaptive Thresholding

- So far, *static* threshold sequences
  - Every site only has "local" view and just pushes updates to coordinator
- Coordinator has global view of current count estimate
  - Can *adaptively* adjust the local site thresholds (based on estimate and T)
  - E.g., dynamically switch from *uniform* to *proportional* growth strategy as estimate approaches/exceeds T



What about *Non-Linear* Functions?



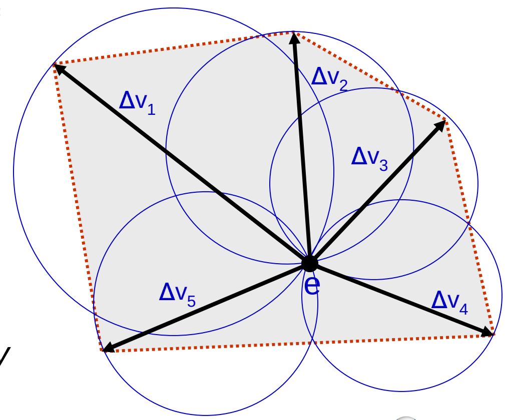
- For *general, non-linear* f(), the problem becomes a lot harder!
  - E.g., information gain or entropy over global data distribution
  - *Non-trivial* to decompose the global threshold into "safe" local site constraints
    - E.g., consider $N=(N_1+N_2)/2$ and $f(N) = 6N - N^2 > 1$
      Impossible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

Monitoring General Threshold Functions

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

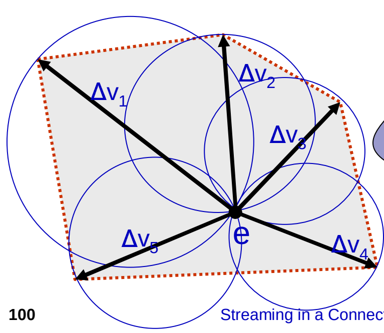
Monitoring General Threshold Functions

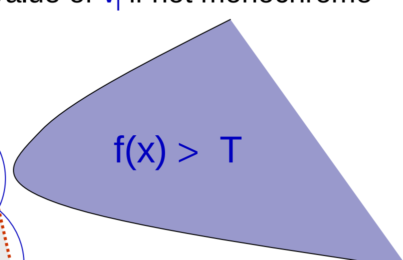
- Key observation: $v = \sum_i \lambda_i \cdot (e + \Delta v_i)$
  (a *convex combination* of “translated” local drifts)
- v lies in the *convex hull* of the $(e + \Delta v_i)$ vectors
- Convex hull is completely covered by the *balls* with radii $||\Delta v_i/2||_2$ centered at $e + \Delta v_i/2$
-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



Monitoring General Threshold Functions

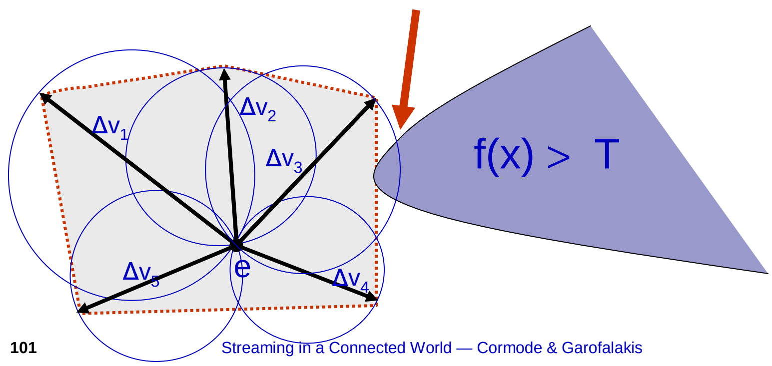
- *Monochromatic Region*: For all points x in the region $f(x)$ is on the same side of the threshold ($f(x) > T$ or $f(x) \leq T$)
- Each site independently checks its ball is monochromatic
  - Find max and min for f() in local ball region (may be costly)
  - Broadcast updated value of v_i if not monochrome





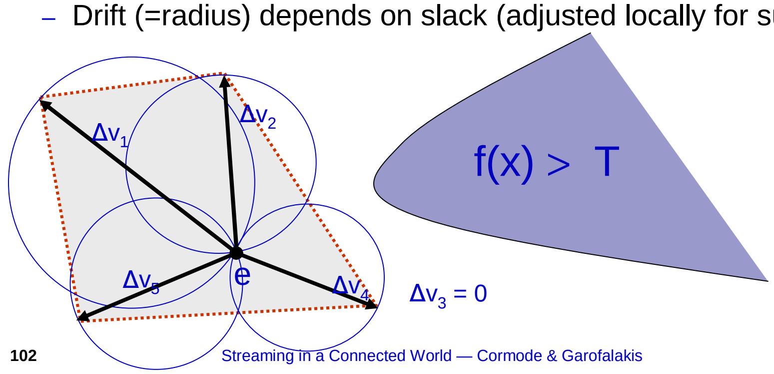
Monitoring General Threshold Functions

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

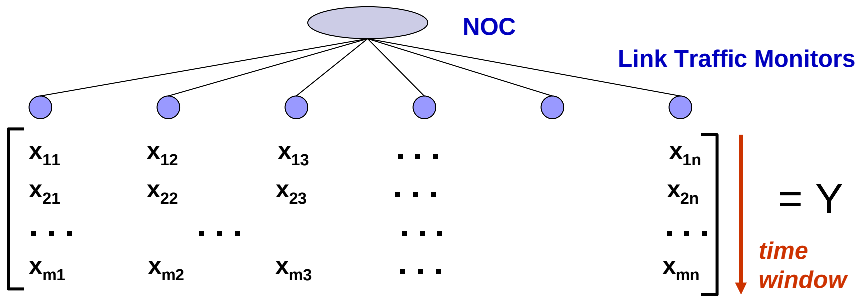


Monitoring General Threshold Functions

- 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 case:* Can allocate local slack vectors to sites to enable "localized" resolutions
  - Drift (=radius) depends on slack (adjusted locally for subsets)



Extension: Filtering for PCA Tracking



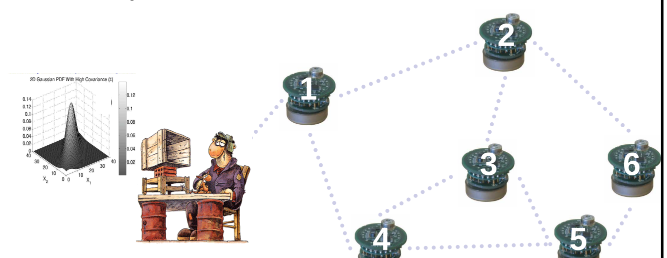
- Threshold total energy of the low PCA coefficients of Y = Robust indicator of network-wide anomalies [Lakhina et al.'04]
  - Non-linear matrix operator over combined time-series
- Can combine local filtering ideas with *stochastic matrix perturbation theory* [Huang et al.'06]

Lessons, Thoughts and Extensions

- Key idea in *trigger tracking*: The threshold is your friend!
  - Exploit for more intelligent (looser, *yet "safe"*) local filtering
- Also, optimize for the common case!
  - Threshold violations are typically "outside the norm"
  - "Push-based" model makes even more sense here
  - Local filters eliminate most/all of the "normal" traffic
- Use richer, dynamic prediction models for triggers?
  - Perhaps adapt depending on distance from threshold?
- More realistic network models?
- Geometric ideas for approximate query tracking?
  - Connections to approximate join-tracking scheme?

Tutorial Outline

- Introduction, Motivation, Problem Setup
- One-Shot Distributed-Stream Querying
- Continuous Distributed-Stream Tracking
- Probabilistic Distributed Data Acquisition
- Future Directions & Open Problems
- Conclusions



Model-Driven Data Acquisition

- *Not only aggregates* – Approximate, bounded-error acquisition of individual sensor values [Deshpande et al. '04]
  - (ε,δ)–approximate acquisition: $|Y - \hat{Y}| \leq \varepsilon$ with prob. $> 1-\delta$
- Regular readings entails large amounts of data, noisy or incomplete data, inefficient, low battery life, …
- *Intuition:* Sensors give (noisy, incomplete) samples of real-world processes
- Use *dynamic probabilistic model* of real-world process to
  - Robustly complement & interpret obtained readings
  - Drive efficient acquisitional query processing

Query Processing in TinyDB

Declarative Query
```
select nodeID, temp
where nodeID in {1..6}
```

USER

Query Results
1, 22.73,
…
6, 22.1.

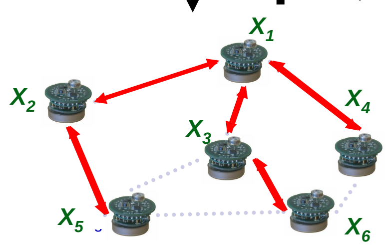
Query Processor

Virtual Table seen by the User

nodeID	Time	temp
1	10am	21
2	10am	22
…	…	…

Observation Plan
{[temp, 1], [temp, 2],
… , [temp, 6]}

Data
1, ten
…
6, ten



SENSOR NETWORK

107

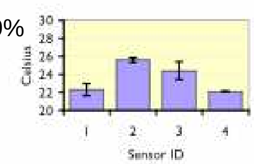
at&t YAHOO! RESEARCH

Model-Based Data Acquisition: BBQ

Declarative Query
```
Select nodeID,
temp ± .1C, conf(.95)
where nodeID in {1..6}
```

USER

Query Results
1, 22.73, 100%
…
6, 22.1, 99%



Probabilistic Model

Query Processor

Observation Plan
{[temp, 1],

Data
1, temp = 22.73,

A *dynamic probabilistic model* of how the data (or the underlying physical process) behaves
- Models the evolution over time
- Captures inter-attribute correlations
- Domain-dependent

X_5 X_6

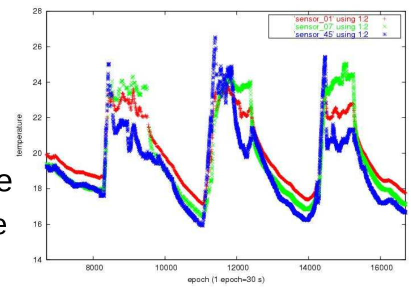
108

at&t YAHOO! RESEARCH

BBQ Details

Probabilistic model captures the joint pdf $p(X_1, \ldots, X_n)$

- *Spatial/temporal correlations*
  - Sensor-to-sensor
  - Attribute-to-attribute
    E.g., voltage & temperature
- *Dynamic:* pdf evolves over time
  - BBQ: Time-varying multivariate Gaussians



- Given user query Q and accuracy guarantees (ε, δ)
  - Try to answer Q directly from the current model
  - If not possible, use model to find efficient *observation plan*
  - Observations update the model & generate (ε,δ) answer

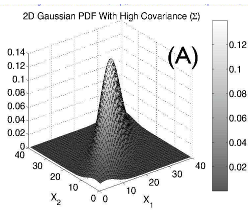
BBQ Probabilistic Queries

- Classes of probabilistic queries
  - *Range predicates:* Is $X_i \in [a_i, b_i]$ with prob. $> 1-\delta$
  - *Value estimates:* Find X'_i such that $\Pr[\,|X_i - X'_i| < \varepsilon] > 1 - \delta$
  - *Aggregate estimates:* (ε,δ)-estimate **avg**/**sum**$(X_{i1}, X_{i2} \ldots X_{ik})$
- Acquire readings if model cannot answer Q at δ conf. level
- Key model operations are
  - *Marginalization:* $p(X_i) = \int p(X_1,\ldots,X_n)\, d\mathbf{x}$
  - *Conditioning:* $p(X_1,\ldots, X_n \mid \text{observations})$
  - *Integration:* $\int_a^b p(X_1,\ldots,X_n)\, d\mathbf{x}$, also expectation $X'_i = E[X_i]$

All significantly simplified for Gaussians!

BBQ Query Processing

Joint pdf at time=t
$p(X^t_1, \dots, X^t_n)$



Probabilistic query
Value of $X_2 \pm \varepsilon$
with prob. > 1-δ

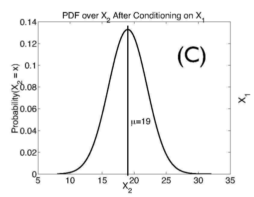
Is
$$P(X_2 \in [\mu_2 - \epsilon, \mu_2 + \epsilon]) = \int_{\mu_2-\epsilon}^{\mu_2+\epsilon} P(x_2)\,dx_2$$
below 1-δ?

Yes / **No**

Return μ_2 / **Must sense more data**

Example: Observe X_1=18
Incorporate into model

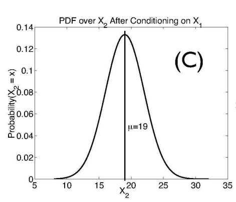
$P(X_2|X_1=18)$



Higher prob., can now answer query

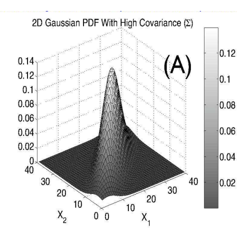
Evolving the Model over Time

Joint pdf at time=t
$p(X^t_1, \dots, X^t_n \mid X^t_1=18)$



Use a (Markov) *Transition Model*
$$P(\mathbf{X}^{t+1} \mid \mathbf{X}^t)$$

Joint pdf at time=t
$p(X^{t+1}_1, \dots, X^{t+1}_n \mid X^t_1=18)$



- In general, a two-step process:

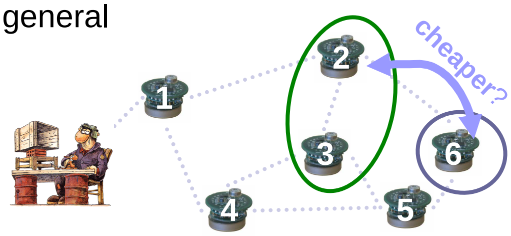
$$p(X^t \mid obs^{1\ldots t}) \xrightarrow{\text{Trans. Model}} p(X^{t+1} \mid obs^{1\ldots t}) \xrightarrow{\text{Condition}} p(X^{t+1} \mid obs^{1\ldots t+1})$$

- *Bayesian filtering* (for Gaussians this yields *Kalman filters*)

Optimizing Data Acquisition

- Energy/communication-efficient observation plans
  - Non-uniform data acquisition costs and network communication costs
  - Exploit data correlations and knowledge of topology
- Minimize Cost(*obs*) over all *obs* $\subseteq$ {1,…, n} so expected confidence in query answer given *obs* (from model) $> 1-\delta$
- **NP-hard** to optimize in general

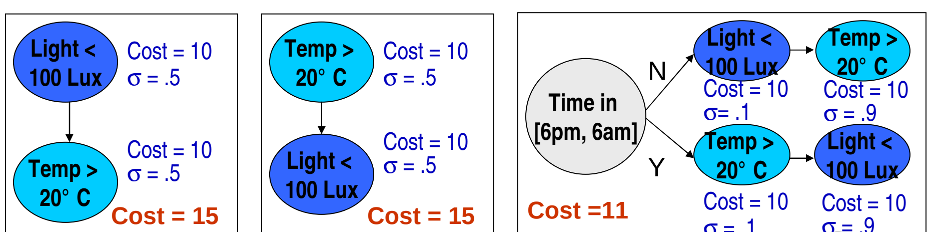
Sensor	Energy per sample (mJ)
Solar Radiation	.525
Barometric Pressure	0.003
Humidity and Temp.	0.5
Voltage	0.00009



Conditional Plans for Data Acquisition

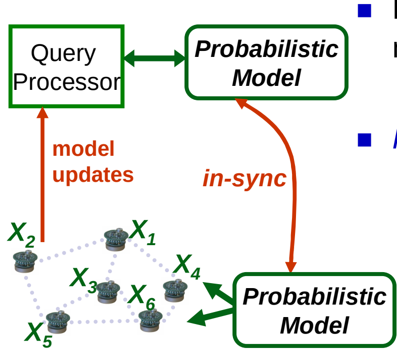
- Observation plans ignore the attribute values observed
  - Attribute subset chosen is observed in its entirety
  - The observed attribute values give a lot more information
- *Conditional* observation plans (outlined in [Deshpande et al.'05])
  - Change the plan depending on observed attribute values (not necessarily in the query)
  - Not yet explored for *probabilistic* query answers

```
SELECT * FROM sensors WHERE light<100Lux and temp>20°C
```



Continuous Model-Driven Acquisition

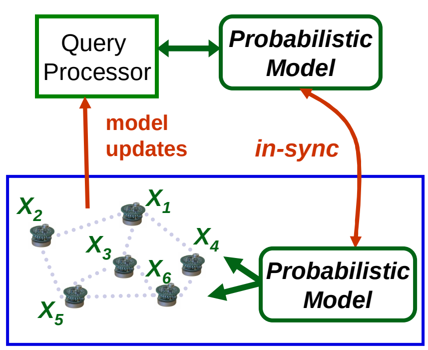
```
select nodeID,
temp ± .1C, conf(.95)
where nodeID in {1..6}
epoch 2 min
```



Dynamic *Replicated* Prob Models *(Ken)* [Chu et al.'06]

- Model *shared and sync'd* across base-station and sensornet
- Nodes continuously check & maintain model accuracy based on ground truth
  - Push vs. Pull (BBQ)
- *Problem: In-network model maintenance*
  - Exploit *spatial data correlations*
  - Model updates decided in-network and sent to base-station
  - Always keep model (ε,δ)-approximate

In-Network Model Maintenance



- Mapping model maintenance onto network topology
  - *At each step*, nodes check (ε,δ) accuracy, send updates to base
- Choice of model drastically affects communication cost
  - Must centralize correlated data for model check/update
  - Can be expensive!
- Effect of *degree of spatial correlations:*

Single-node models $\Pi\, p(X_i)$
No spatial correlations
Cheap – check is local!

⟷

Full-network model $p(X_1,\ldots,X_n)$
Full spatial correlations
Expensive – centralize all data!

In-Network Model Maintenance

Single-node models $\prod p(X_i)$
No spatial correlations
Cheap – check is local!

Single-node Kalman filters
[Jain et al.'04]

⟷

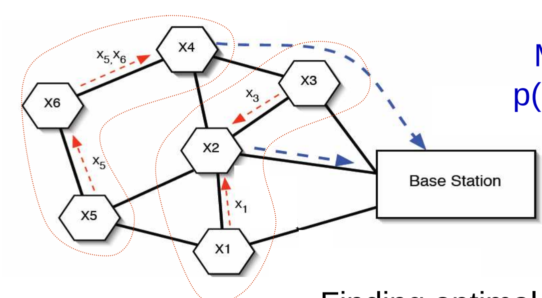
Full-network model $p(X_1,\ldots,X_n)$
Full spatial correlations
Expensive – centralize all data!

BBQ
[Deshpande et al. '04]

- **Problem:** Find dynamic probabilistic model and in-network maintenance schedule to minimize overall communication
  - Map maintenance/update operations to network topology
- Key idea for "practical" in-network models
  - Exploit *limited-radius* spatial correlations of measurements
  - Localize model checks/updates to small regions

Disjoint-Cliques Models

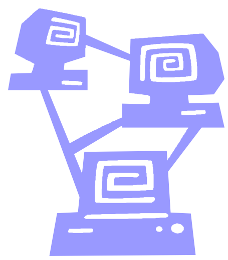
- *Idea:* Partition joint pdf into a set of small, localized "cliques" of random variables
  - Each clique maintained and updated *independently* at "clique root" nodes



Model $p(X_1,\ldots,X_6) = p(X_1,X_2,X_3) \cdot p(X_4,X_5,X_6)$

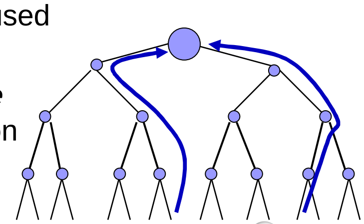
- Finding optimal DC model is NP-hard
  - Natural analogy to *Facility Location*

Distributed Data Stream Systems/Prototypes



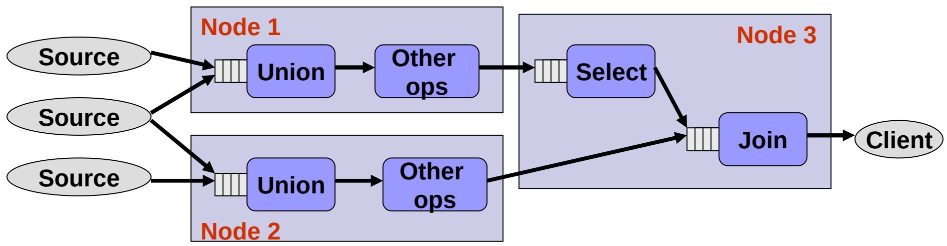
Current Systems/Prototypes

- Main algorithmic idea in the tutorial: Trade-off space/time *and communication* with approximation quality
- Unfortunately, approximate query processing tools are still not widely adopted in current Stream Processing engines
  – Despite obvious relevance, *especially for streaming data*
- In the sensornet context
  – Simple in-network aggregation techniques (e.g., for **`average`**, **`count`**, etc.) are widely used
  E.g., TAG/TinyDB [Madden et al '02]
  – More complex tools for *approximate* in-network data processing/collection have yet to gain wider acceptance



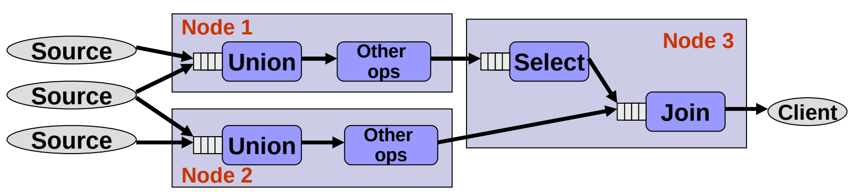
Distributed SP Engine Prototypes

- Telegraph/TelegraphCQ [Chandrasekaran et al.'03] , Borealis/Medusa [Balazinska et al.'05], P2 [Loo et al.'06]
- Query processing typically viewed as a *large dataflow*
  - Network of connected, pipelined query operators



  - Schedule a large dataflow over a distributed system
    - Objectives: Load-balancing, availability, early results, …

Distributed SP Engine Prototypes



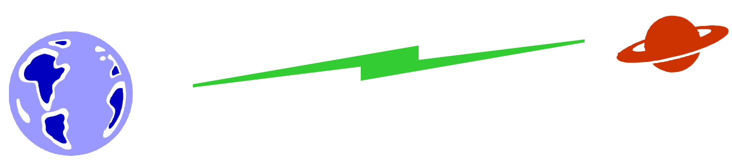
- Approximate answers and error guarantees not considered
  - General relational queries, push/pull-ing tuples through the query network
  - Load-shedding techniques to manage overload
    - No hard error guarantees
- Network costs (bandwidth/latency) considered in some recent work [Pietzuch et al.'06]

Other Systems & Prototypes

- **PIER** – Scaling to large, dynamic site populations using DHTs [Huebsch et al.'03]
  - See also the *Seaweed* paper [Narayanan et al.'06]
- **Gigascope** – Streaming DB engine for large-scale network/ application monitoring
  - Optimized for high-rate data streams ("line speeds")
  - Exploits approximate query processing tools (sampling, sketches, …) for tracking streams at endpoints
  - Distribution issues not addressed (yet…)

Tutorial Outline

- Introduction, Motivation, Problem Setup
- One-Shot Distributed-Stream Querying
- Continuous Distributed-Stream Tracking
- Probabilistic Distributed Data Acquisition
- Future Directions & Open Problems
- Conclusions

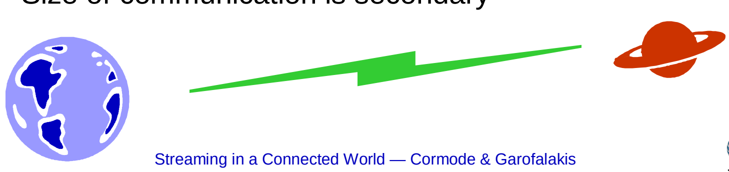


Extensions for P2P Networks

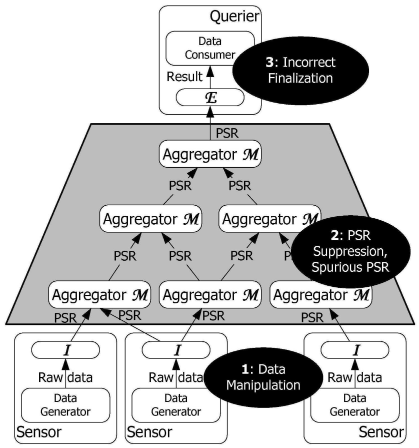
- Much work focused on specifics of sensor and wired nets
- P2P and Grid computing present alternate models
  - Structure of multi-hop overlay networks
  - "Controlled failure" model: nodes explicitly leave and join
- Allows us to think beyond model of "highly resource constrained" sensors.
- Implementations such as OpenDHT over PlanetLab [Rhea et al.'05]

Delay-Tolerant Networks

- How to cope when connectivity is *intermittent* ?
  - Roaming devices, exploring outer and inner space, network infrastructure for emerging regions (e.g., rural India), …
  - Round trip times may be very long and varying
    - Radio to Mars is many minutes
    - Connectivity to remote villages varies [Jain, Fall, Patra '05]
- Goal is to minimize the number of communications and maximize timeliness
  - Size of communication is secondary



Authenticated Stream Aggregation



- 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 in [Garofalakis et al.'06]

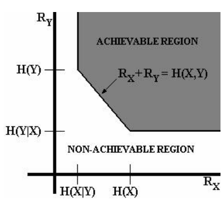
Other Classes of Queries

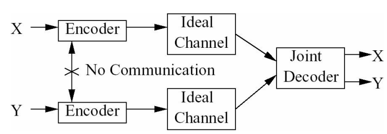
- Mostly talked about specific, well-defined aggregates
- What about *set-valued* query answers?
  - No principled, "universal" approximation error metric
- A general distributed query language (dist-streamSQL?)
  - Define a language so a query optimizer can find a plan that guarantees good performance, small communication?
- Other tasks, e.g., data mining, machine learning, over distributed streams?
  - ML/AI communities are already starting to consider communication-efficient distributed learning

Theoretical Foundations

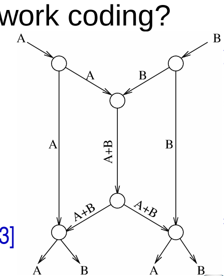
"Communication complexity" studies lower bounds of distributed **one-shot** computations

- Gives lower bounds for various problems, e.g., **`count distinct`** (via reduction to abstract problems)
- Need new theory for continuous computations
  - Based on info. theory and models of how streams evolve?
  - Link to distributed source coding or network coding?





Slepian-Wolf theorem [Slepian Wolf 1973]



http://www.networkcoding.info/

https://buffy.eecs.berkeley.edu/PHP/resabs/resabs.php?f_year=2005&f_submit=chapgrp&f_chapter=1

Richer Prediction models

- 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]

Conclusions

- Many new problems posed by developing technologies
- Common features of *distributed streams* allow for general techniques/principles instead of "point" solutions
  - In-network query processing
    Local filtering at sites, trading-off approximation with processing/network costs, …
  - Models of "normal" operation
    Static, dynamic ("predictive"), probabilistic, …
  - Exploiting network locality and avoiding global resyncs
- Many new directions unstudied, more will emerge as new technologies arise
- *Lots of exciting research to be done*! ☺

References (1)

[Aduri, Tirthapura '05] P. Aduri and S. Tirthapura. Range-efficient Counting of F_0 over Massive Data Streams. In IEEE International Conference on Data Engineering, 2005

[Agrawal et al. '04] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: New aggregation techniques for sensor networks. In ACM SenSys, 2004

[Alon, Gibbons, Matias, Szegedy '99] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In Proceedings of ACM Symposium on Principles of Database Systems, pages 10–20, 1999.

[Alon, Matias, Szegedy '96] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In Proceedings of the ACM Symposium on Theory of Computing, pages 20–29, 1996. Journal version in Journal of Computer and System Sciences, 58:137–147, 1999.

[Babcock et al. '02] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems In ACM Principles of Database Systems, 2002

[Chu et al'06] D. Chu, A. Deshpande, J. M. Hellerstein, W. Hong. Approximate Data Collection in Sensor Networks using Probabilistic Models. IEEE International Conference on Data Engineering 2006, p48

[Considine, Kollios, Li, Byers '05] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In IEEE International Conference on Data Engineering, 2004.

[Cormode, Garofalakis '05] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In Proceedings of the International Conference on Very Large Data Bases, 2005.

[Cormode et al.'05] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In Proceedings of ACM SIGMOD International Conference on Management of Data, 2005.

References (2)

[Cormode, Muthukrishnan '04] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. Journal of Algorithms, 55(1):58–75, 2005.

[Cormode, Muthukrishnan '05] G. Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In Proceedings of ACM Principles of Database Systems, 2005.

[Das et al.'04] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. Distributed Set-Expression Cardinality Estimation. In Proceedings of VLDB, 2004.

[Deshpande et al'04] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, W. Hong. Model-Driven Data Acquisition in Sensor Networks. In VLDB 2004, p 588-599

[Deshpande et al'05] A. Deshpande, C. Guestrin, W. Hong, S. Madden. Exploiting Correlated Attributes in Acquisitional Query Processing. In IEEE International Conference on Data Engineering 2005, p143-154

[Dilman, Raz '01] M. Dilman, D. Raz. Efficient Reactive Monitoring. In IEEE Infocom, 2001.

[Flajolet, Martin '83] P. Flajolet and G. N. Martin. Probabilistic counting. In IEEE Conference on Foundations of Computer Science, pages 76–82, 1983. Journal version in Journal of Computer and System Sciences, 31:182–209, 1985.

[Garofalakis et al. '02] M. Garofalakis, J. Gehrke, R. Rastogi. Querying and Mining Data Streams: You Only Get One Look. Tutorial in ACM SIGMOD International Conference on Management of Data, 2002.

[Garofalakis et al.'06] M. Garofalakis, J. Hellerstein, and P. Maniatis. Proof Sketches: Verifiable Multi-Party Aggregation. UC-Berkeley EECS Tech. Report, 2006.

[Gibbons, Tirthapura '01] P. Gibbons, S. Tirthapura. Estimating simple functions on the union of data streams. In ACM Symposium on Parallel Algorithms and Architectures, 2001.

[Greenwald, Khanna '01] M. Greenwald, S. Khanna. Space-efficient online computation of quantile summaries. In Proceedings of ACM SIGMOD International Conference on Management of Data, 2001.

References (3)

[Greenwald, Khanna '04] M. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In Proceedings of ACM Principles of Database Systems, pages 275–285, 2004.

[Hadjieleftheriou, Byers, Kollios '05] M. Hadjieleftheriou, J. W. Byers, and G. Kollios. Robust sketching and aggregation of distributed data streams. Technical Report 2005-11, Boston University Computer Science Department, 2005.

[Huang et al.'06] L. Huang, X. Nguyen, M. Garofalakis, M. Jordan, A. Joseph, and N. Taft. Distributed PCA and Network Anomaly Detection. In NIPS, 2006.

[Huebsch et al.'03] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, I. Stoica. Querying the Internet with PIER. In VLDB, 2003.

[Jain et al'04] A. Jain, E. Y. Chang, Y-F. Wang. Adaptive stream resource management using Kalman Filters. In ACM SIGMOD International Conference on Management of Data, 2004.

[Jain, Fall, Patra '05] S. Jain, K. Fall, R. Patra, Routing in a Delay Tolerant Network, In IEEE Infocom, 2005

[Jain, Hellerstein et al'04] A. Jain, J.M.Hellerstein, S. Ratnasamy, D. Wetherall. A Wakeup Call for Internet Monitoring Systems: The Case for Distributed Triggers. In Proceedings of HotNets-III, 2004.

[Johnson et al.'05] T. Johnson, S. Muthukrishnan, V. Shkapenyuk, and O. Spateschek. A heartbeat mechanism and its application in Gigascope. In VLDB, 2005.

[Kashyap et al. '06] S. Kashyap, S. Deb, K.V.M. Naidu, R. Rastogi, A. Srinivasan. Efficient Gossip-Based Aggregate Computation. In ACM Principles of Database Systems, 2006.

[Kempe, Dobra, Gehrke '03] D. Kempe, A. Dobra, and J. Gehrke. Computing aggregates using gossip. In IEEE Conference on Foundations of Computer Science, 2003.

[Kempe, Kleinberg, Demers '01] D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. In Proceedings of the ACM Symposium on Theory of Computing, 2001.

References (4)

[Kerlapura et al.'06] R. Kerlapura, G. Cormode, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In ACM SIGMOD, 2006.

[Koudas, Srivastava '03] N. Koudas and D. Srivastava. Data stream query processing: A tutorial. In VLDB, 2003.

[Madden '06] S. Madden. Data management in sensor networks. In Proceedings of European Workshop on Sensor Networks, 2006.

[Madden et al. '02] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. In Proceedings of Symposium on Operating System Design and Implementation, 2002.

[Manjhi, Nath, Gibbons '05] A. Manjhi, S. Nath, and P. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In Proceedings of ACM SIGMOD International Conference on Management of Data, 2005.

[Manjhi et al.'05] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In IEEE International Conference on Data Engineering, pages 767–778, 2005.

[Muthukirshnan '03] S. Muthukrishnan. Data streams: algorithms and applications. In ACM-SIAM Symposium on Discrete Algorithms, 2003.

[Narayanan et al.'06] D. Narayanan, A. Donnelly, R. Mortier, and A. Rowstron. Delay-aware querying with Seaweed. In VLDB, 2006.

[Nath et al.'04] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggrgation in sensor networks. In ACM SenSys, 2004.

References (5)

[Olston, Jiang, Widom '03] C. Olston, J. Jiang, J. Widom. Adaptive Filters for Continuous Queries over Distributed Data Streams. In ACM SIGMOD, 2003.

[Pietzuch et al.'06] P. R. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, M. I. Seltzer. Network-Aware Operator Placement for Stream-Processing Systems. In IEEE ICDE, 2006.

[Pittel '87] B. Pittel On Spreading a Rumor. In SIAM Journal of Applied Mathematics, 47(1) 213-223, 1987

[Rhea et al. '05] S. Rhea, G. Brighten, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, Y. Harlan. OpenDHT: A public DHT service and its uses. In ACM SIGCOMM, 2005

[Rissanen '78] J. Rissanen. Modeling by shortest data description. Automatica, 14:465-471, 1978.

[Sharfman et al.'06] Izchak Sharfman, Assaf Schuster, Daniel Keren: A geometric approach to monitoring threshold functions over distributed data streams. SIGMOD Conference 2006: 301-312

[Slepian, Wolf '73] D. Slepian, J. Wolf. Noiseless coding of correlated information sources. IEEE Transactions on Information Theory, 19(4):471-480, July 1973.