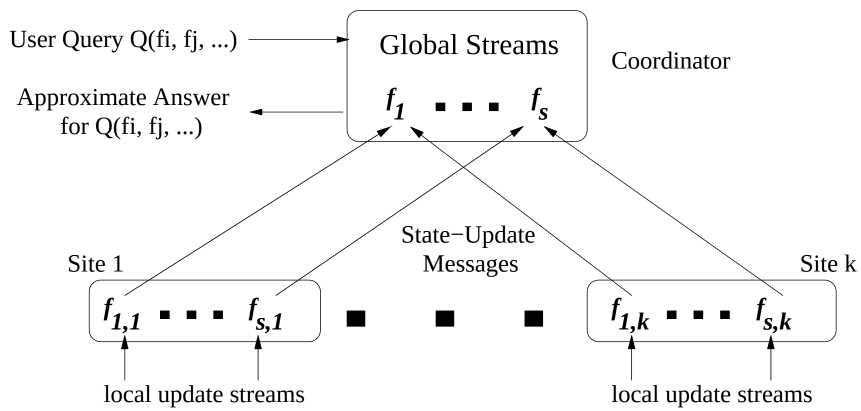
Managing *Distributed* Data Streams – I

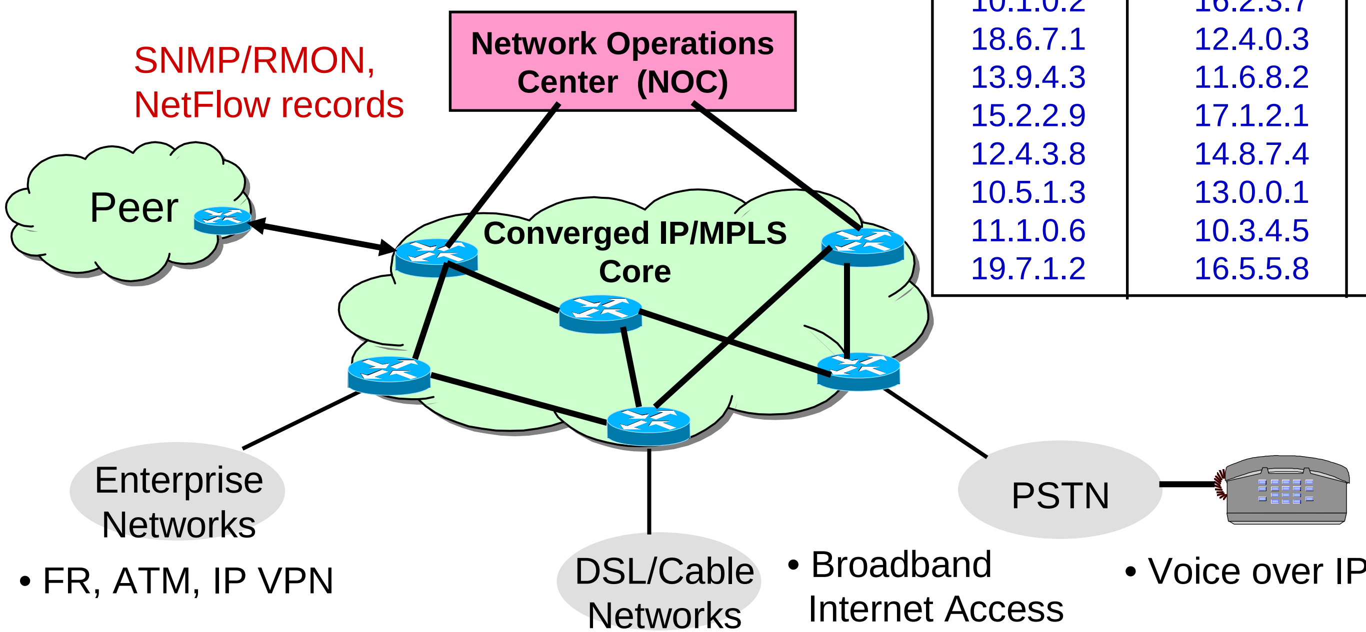


Slides based on the Cormode/Garofalakis VLDB'2006 tutorial

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

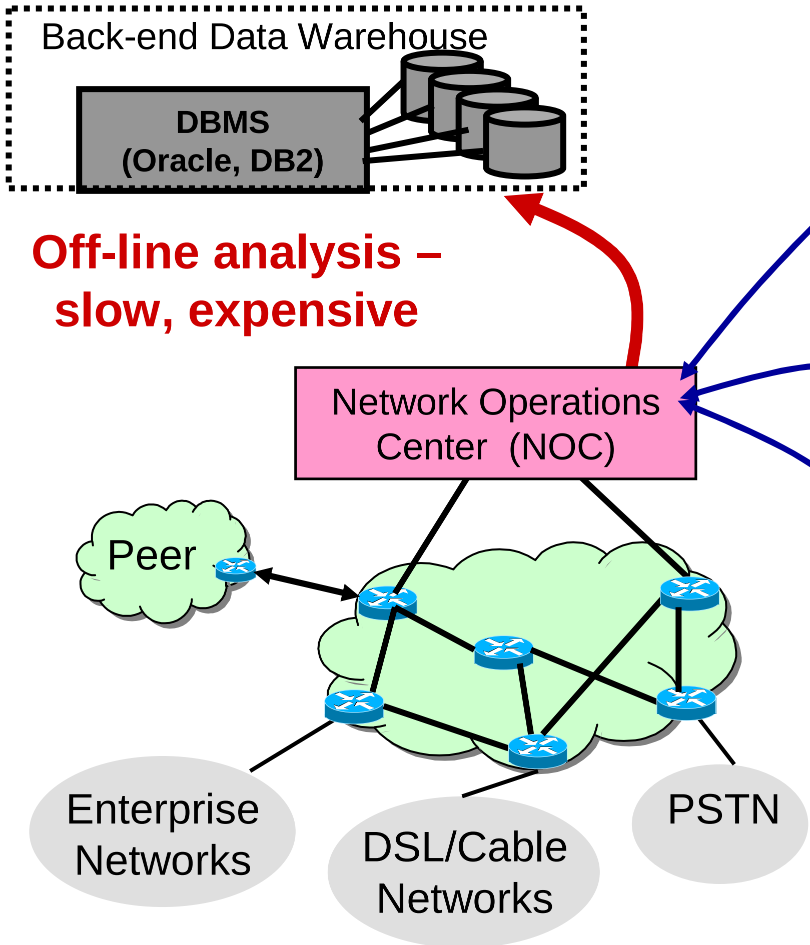


Source	Destination	Duration	Bytes	Protocol
10.1.0.2	16.2.3.7	12	20K	http
18.6.7.1	12.4.0.3	16	24K	http
13.9.4.3	11.6.8.2	15	20K	http
15.2.2.9	17.1.2.1	19	40K	http
12.4.3.8	14.8.7.4	26	58K	http
10.5.1.3	13.0.0.1	27	100K	ftp
11.1.0.6	10.3.4.5	32	300K	ftp
19.7.1.2	16.5.5.8	18	80K	ftp

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?

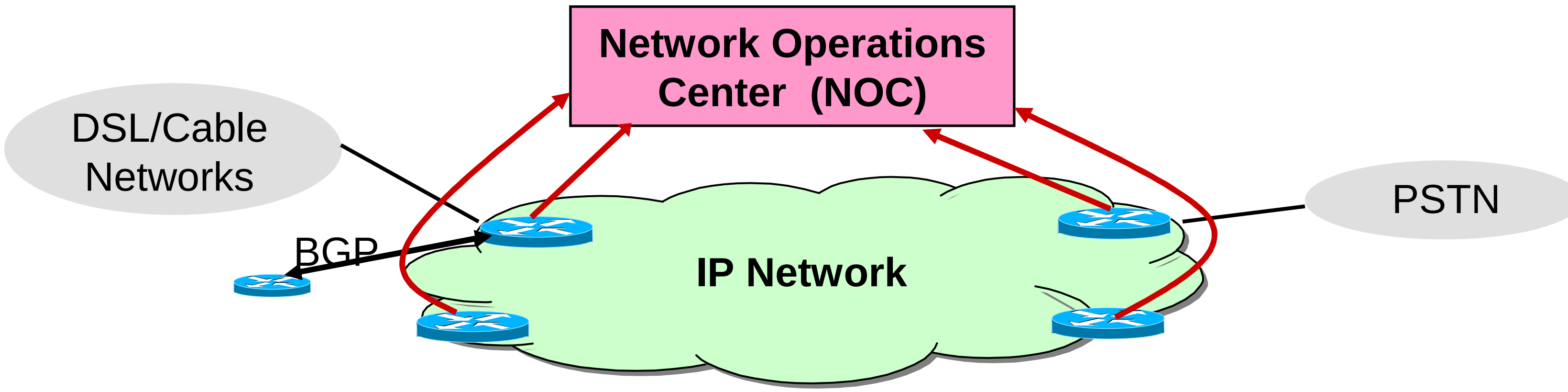
How many distinct (source, dest) pairs have been seen by both R1 and R2 but not R3?

Set-Expression Query

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

SQL Join Query

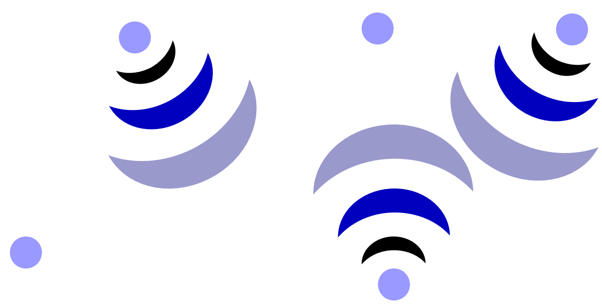
Real-Time Data-Stream Analysis



- 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

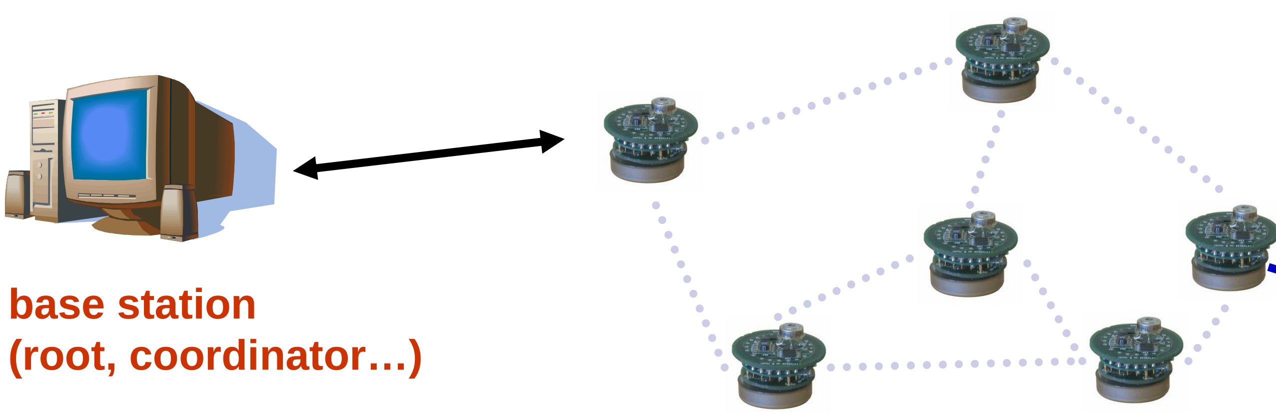
Sensor Networks

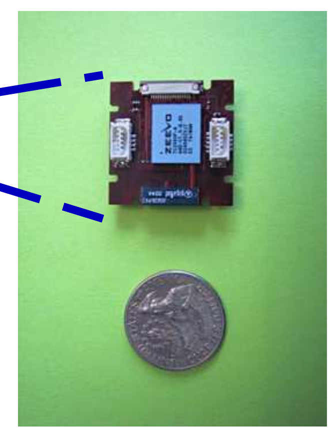
- Wireless sensor networks becoming ubiquitous in environmental monitoring, military applications, …
- Many (100s, 10^3, 10^6?) sensors scattered over terrain
- Sensors observe and process a local stream of readings:
  - Measure light, temperature, pressure…
  - Detect signals, movement, radiation…
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



Sensornet Querying Application

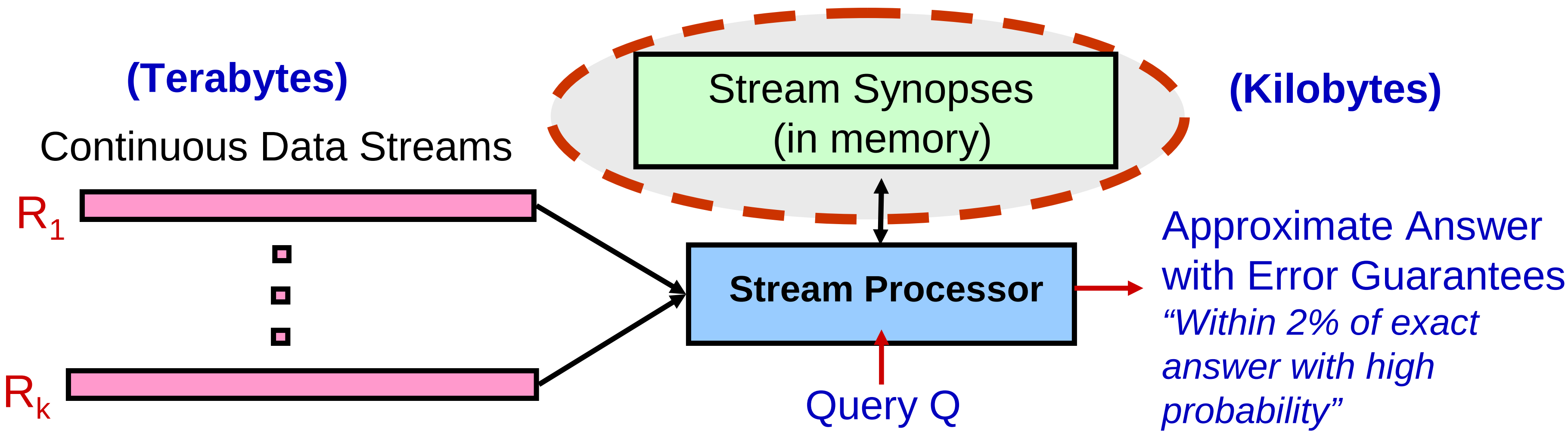
- Query sensornet through a (remote) *base station*
- Sensor nodes have severe resource constraints
  - Limited battery power, memory, processor, radio range…
  - *Communication* is the major source of battery drain
  - "transmitting a single bit of data is equivalent to 800 instructions"   [Madden et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





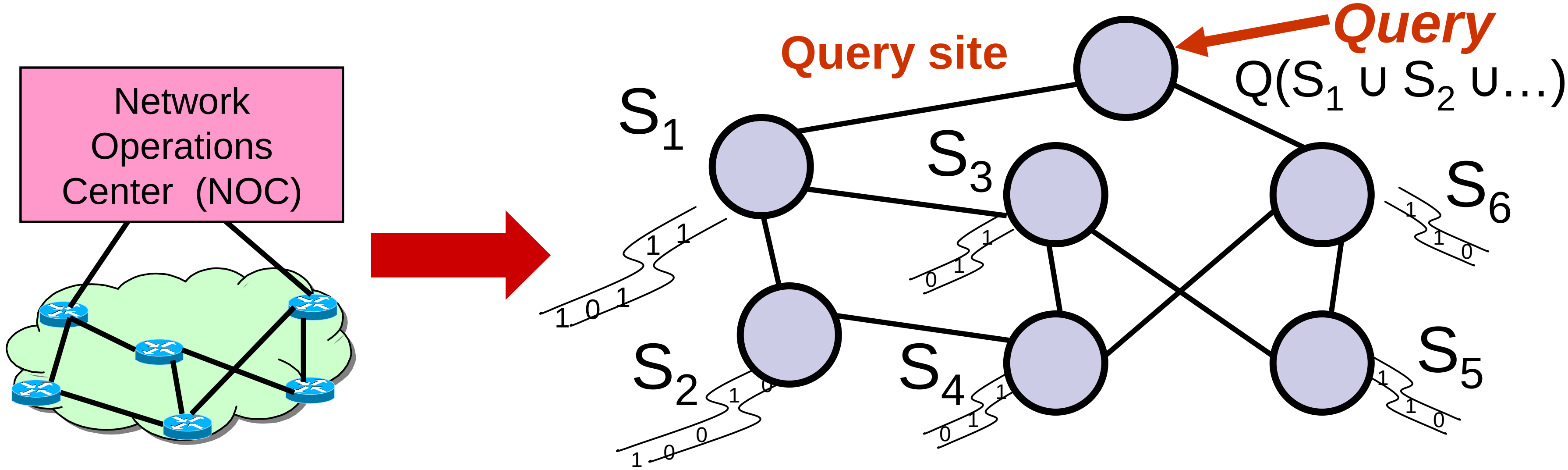
http://www.intel.com/research/exploratory/motes.htm

Data-Stream Algorithmics Model



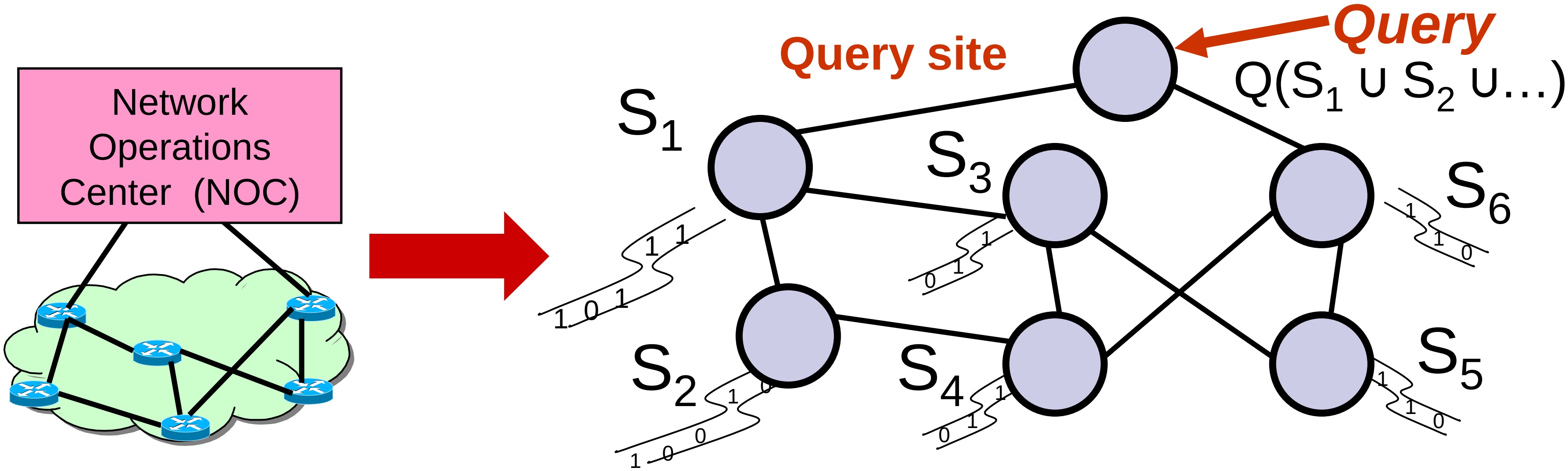
- *Approximate answers*– e.g. trend analysis, anomaly detection
- Requirements for stream synopses
  - *Single Pass:* Each record is examined at most once
  - *Small Space:* Log or polylog in data stream size
  - *Small-time:* Low per-record processing time (maintain synopses)
  - Also: *delete-proof,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

Distributed Streams Model



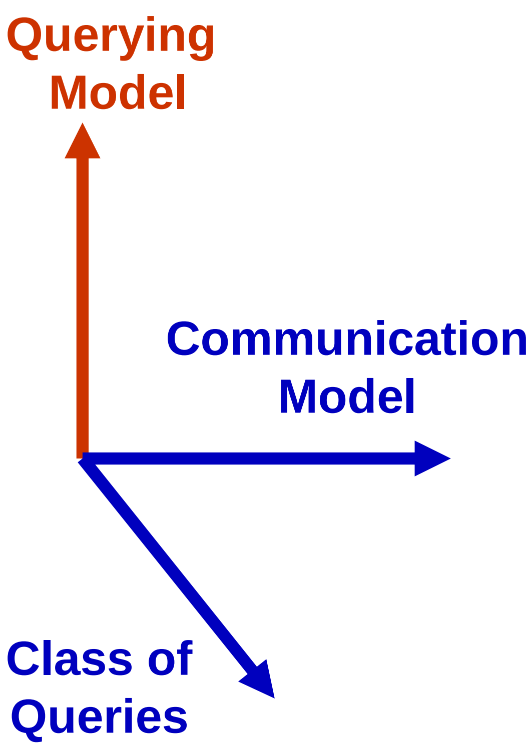
- Large-scale querying/monitoring: *Inherently distributed!*
  - Streams physically distributed across remote sites
    E.g., stream of UDP packets through subset of edge routers
- *Challenge is "holistic" querying/monitoring*
  - Queries over the *union of distributed streams* $Q(S_1 \cup S_2 \cup \ldots)$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

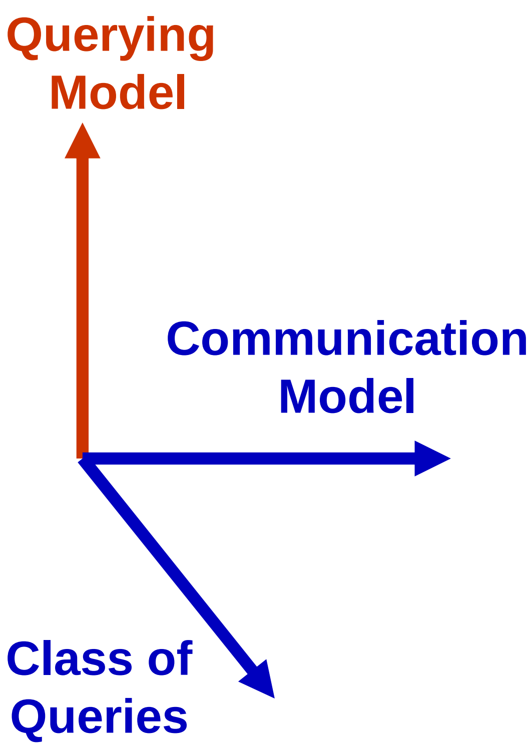
Distributed Stream Querying Space



"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

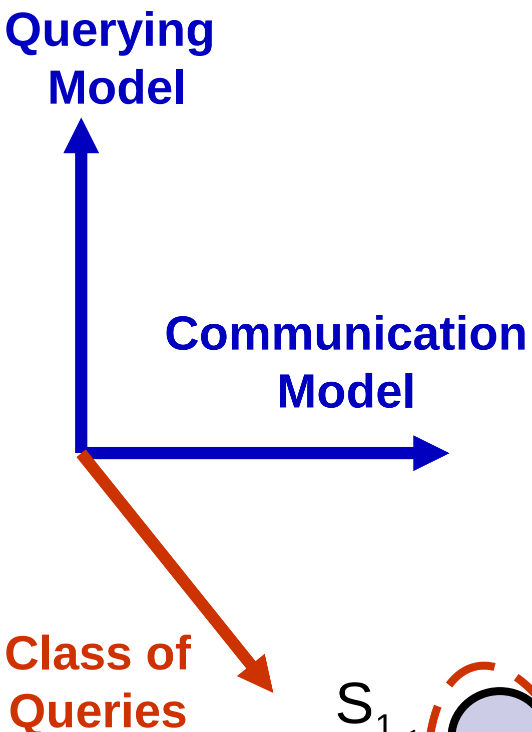
Distributed Stream Querying Space



Minimizing communication often needs approximation and randomization

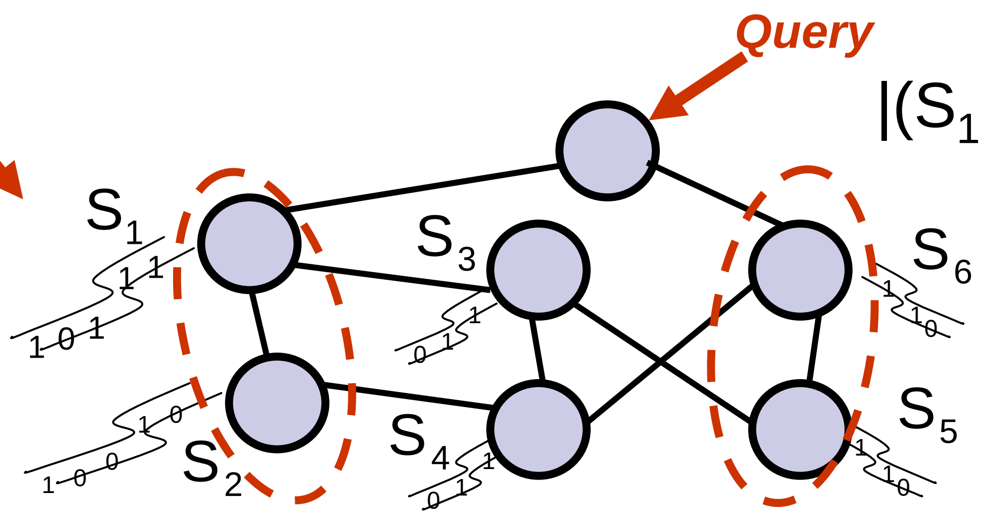
- E.g., Continuously monitor average value
  - Must send every change for exact answer
  - Only need 'significant' changes for approx (def. of "significant" specifies an algorithm)
- Probability sometimes vital to reduce communication
  - `count distinct` in one shot model needs randomness
  - Else **must**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

Distributed Stream Querying Space



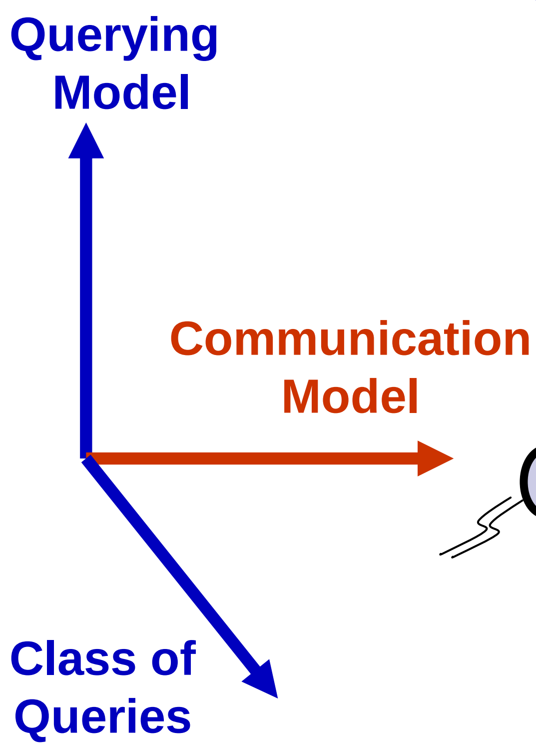
Class of Queries of Interest

- 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, …



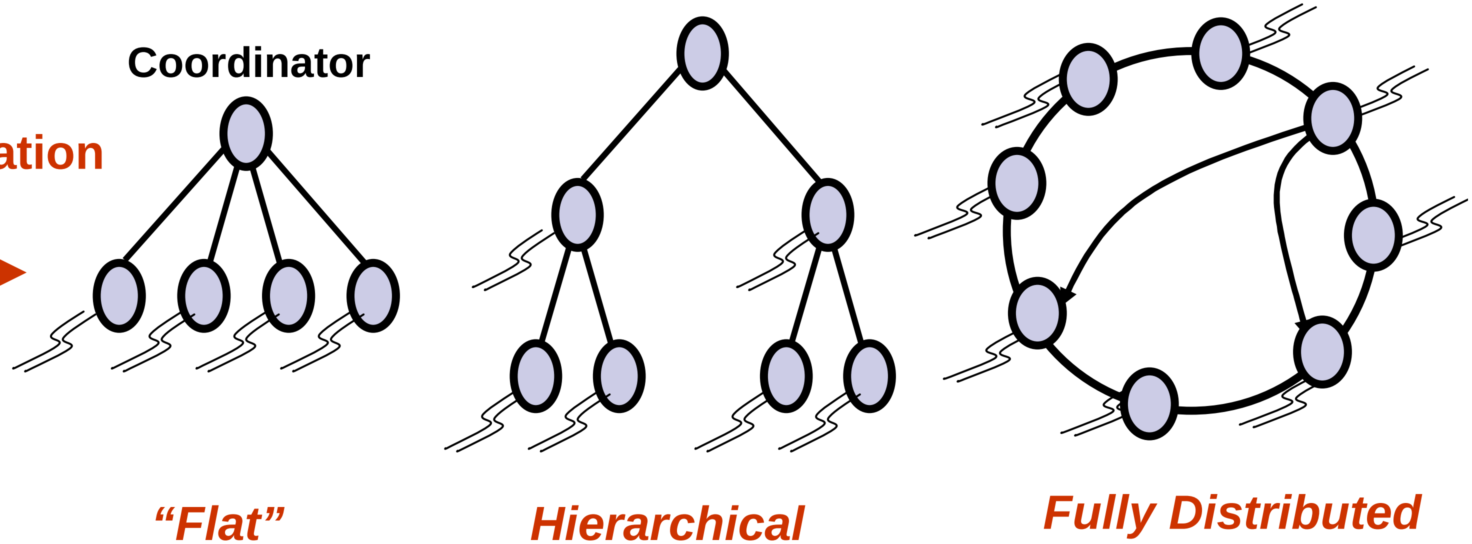
$$|(S_1 \cup S_2) \bowtie (S_5 \cup S_6)|$$

Distributed Stream Querying Space



Communication Network Characteristics

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



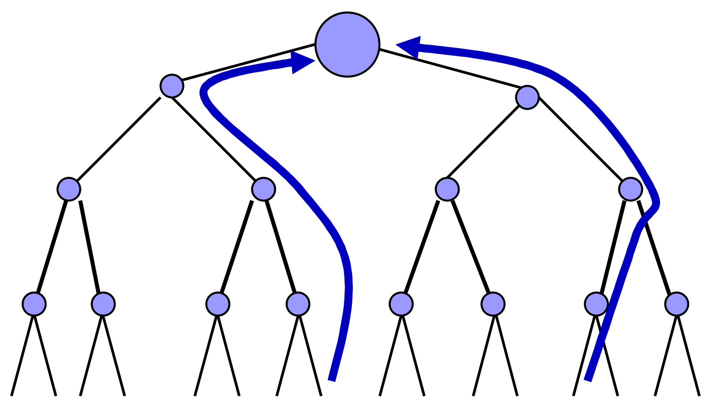
Other network characteristics:

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

Outline

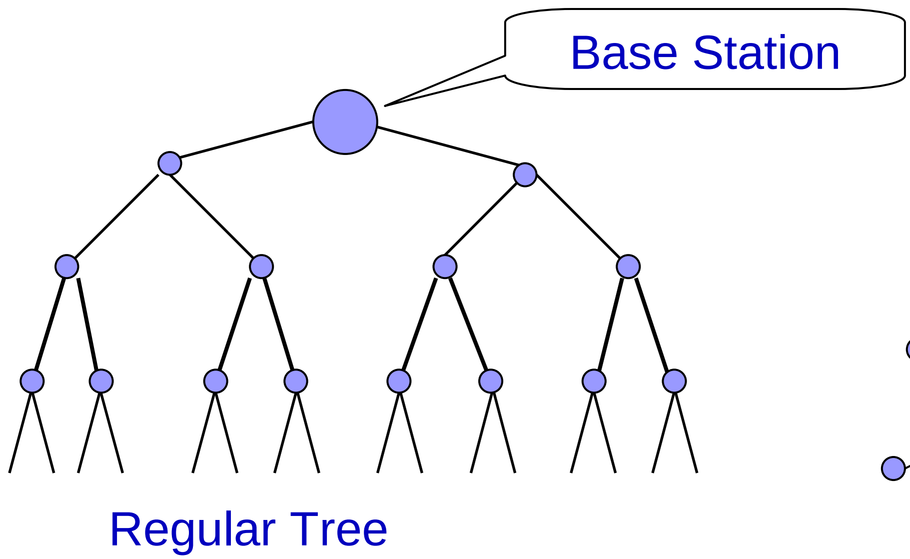
- Introduction, Motivation, Problem Setup
- One-Shot Distributed-Stream Querying
  - Tree Based Aggregation
  - Robustness and Loss
  - *Decentralized Computation and Gossiping*
- Continuous Distributed-Stream Tracking
- Probabilistic Distributed Data Acquisition
- Conclusions

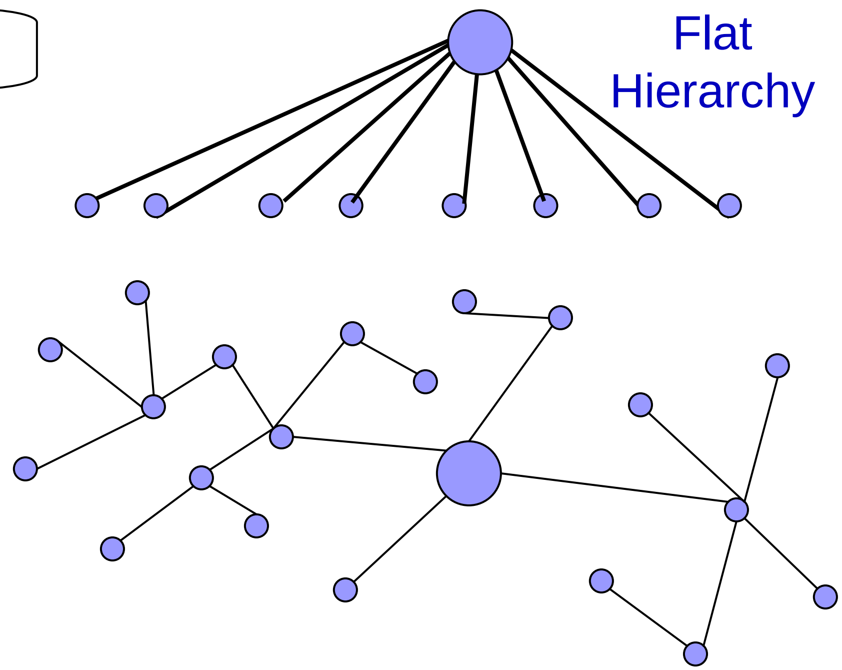
Tree Based Aggregation



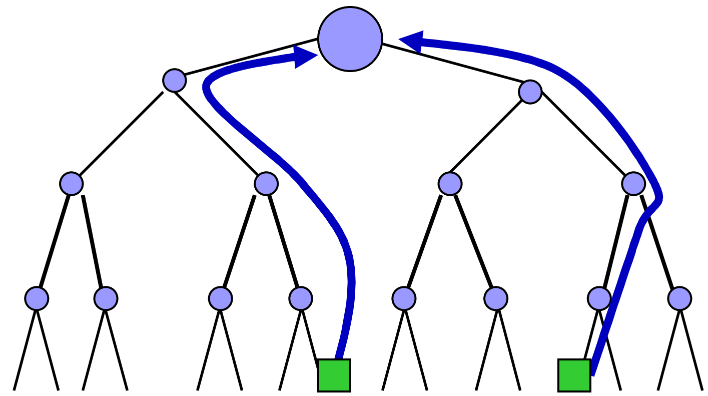
Network Trees

- 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



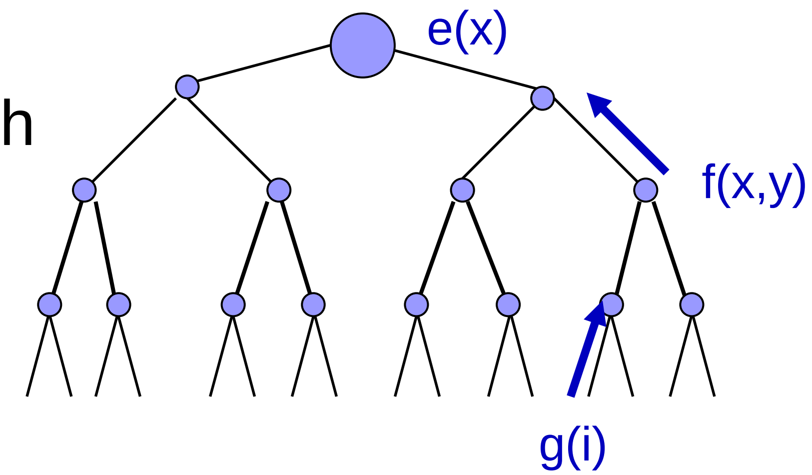
- Goal is for root to compute a function of data at leaves
- Trivial solution: push all data up tree and compute at base station
  - Strains nodes near root: batteries drain, disconnecting network
  - Very wasteful: no attempt at saving communication
- Can do much better by *"In-network" query processing*
  - 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 $F(X \cup Y)$ 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

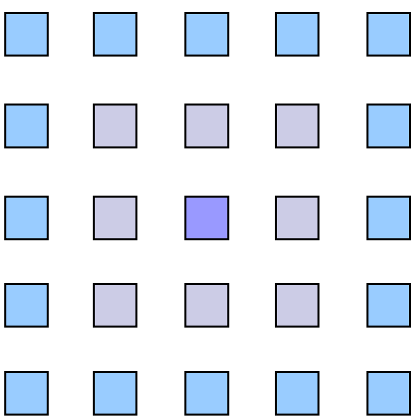
2500 Nodes

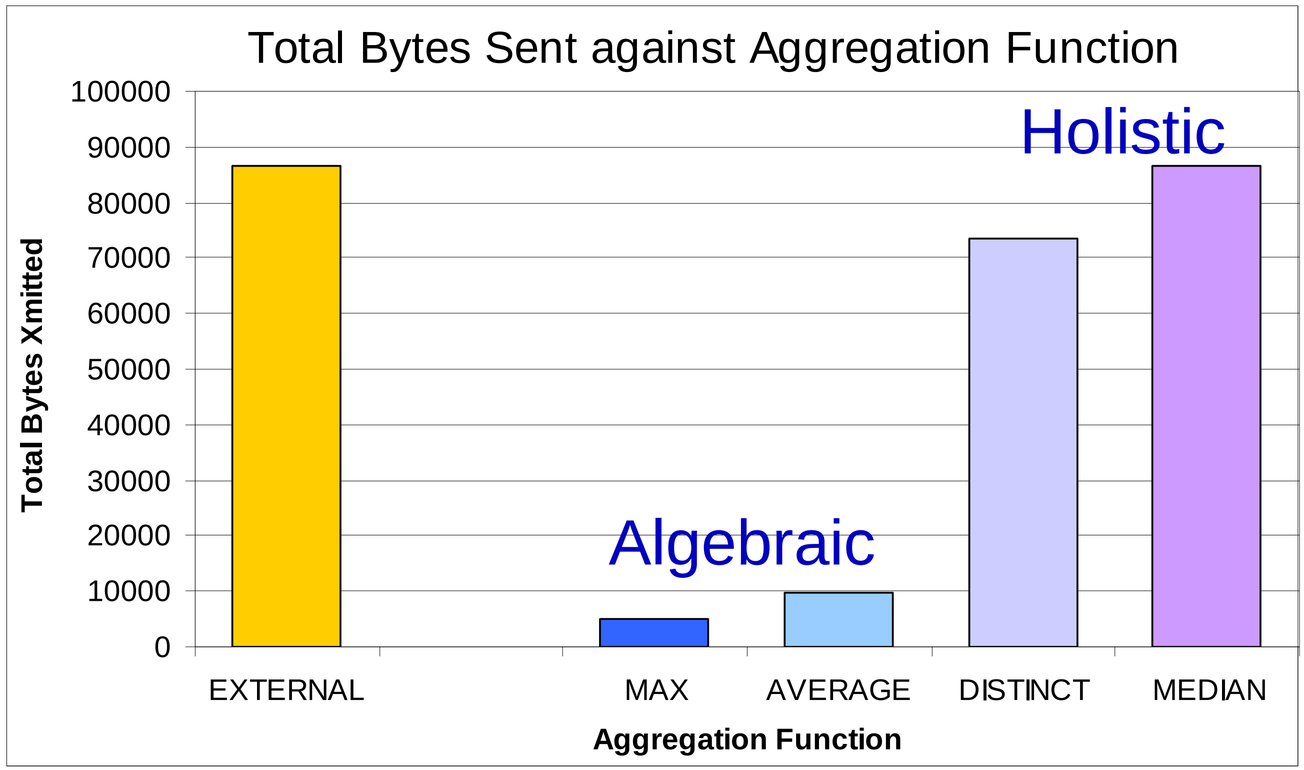
50x50 Grid

Depth = ~10

Neighbors = ~20

Uniform Dist.



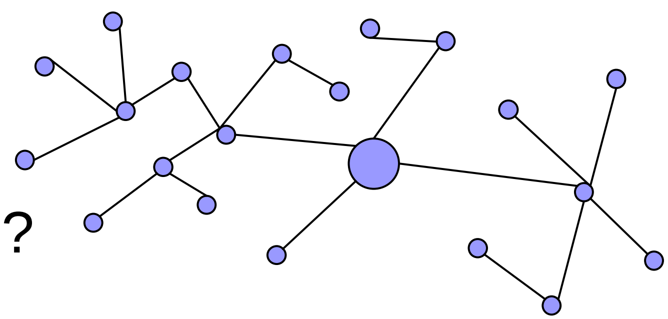


Holistic Aggregates

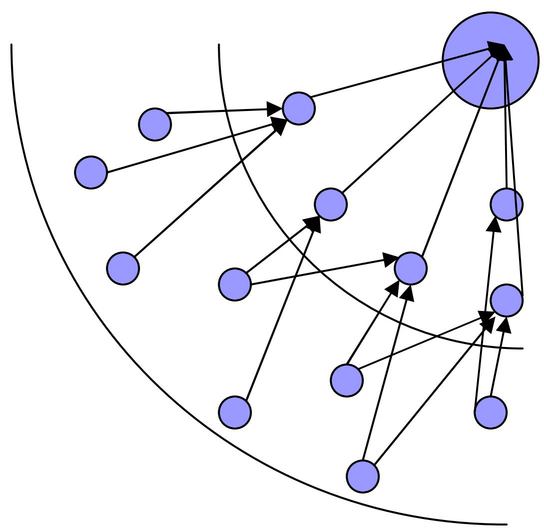
- Holistic aggregates need the whole input to compute (no summary suffices)
  - E.g., **`count distinct`**, need to remember all distinct items to tell if new item is distinct or not
- So focus on approximating aggregates to limit data sent
  - Adopt ideas from sampling, data reduction, streams etc.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

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?

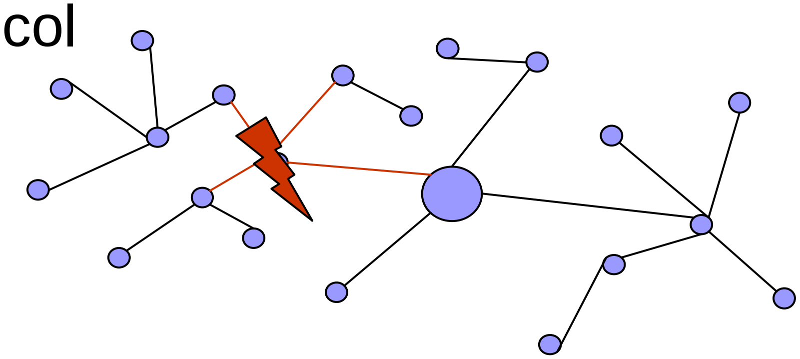


Robustness and Loss



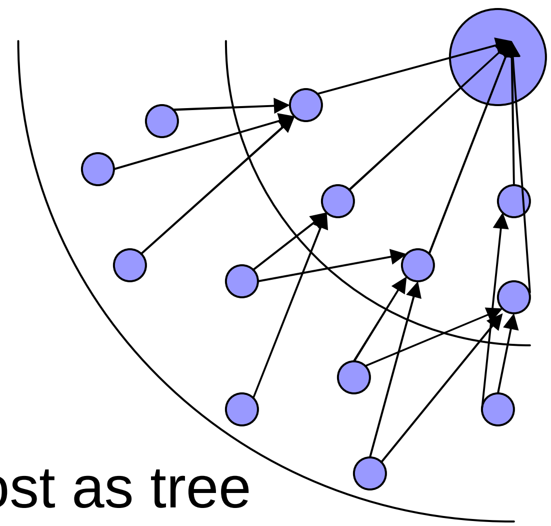
Unreliability

- 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

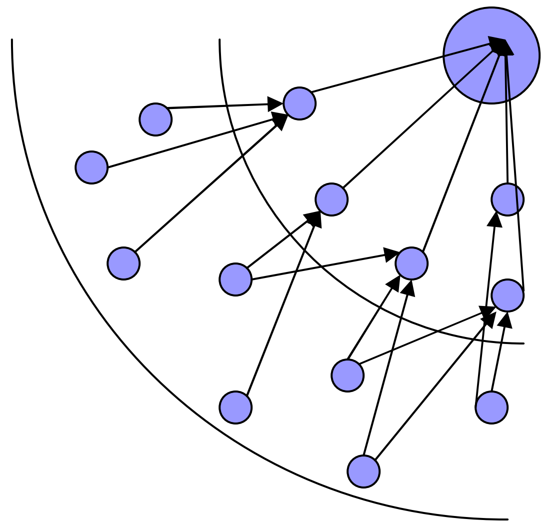


Order and Duplicate Insensitivity

- 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

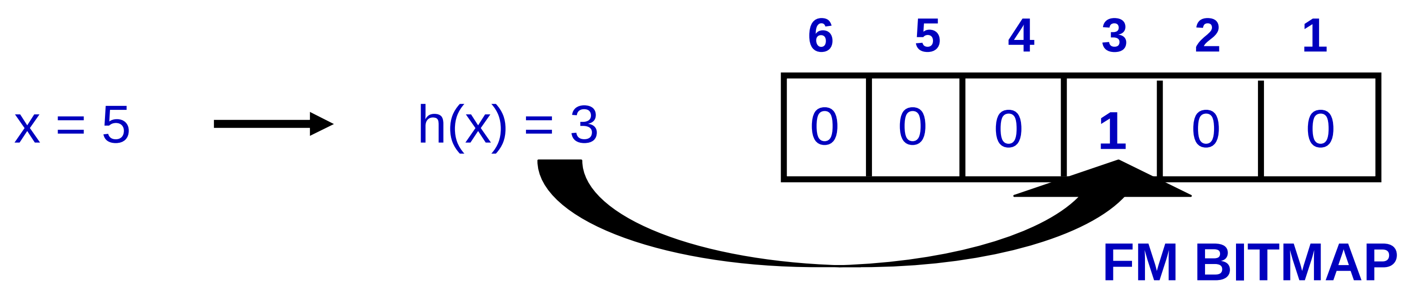
Applying ODI idea

- 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



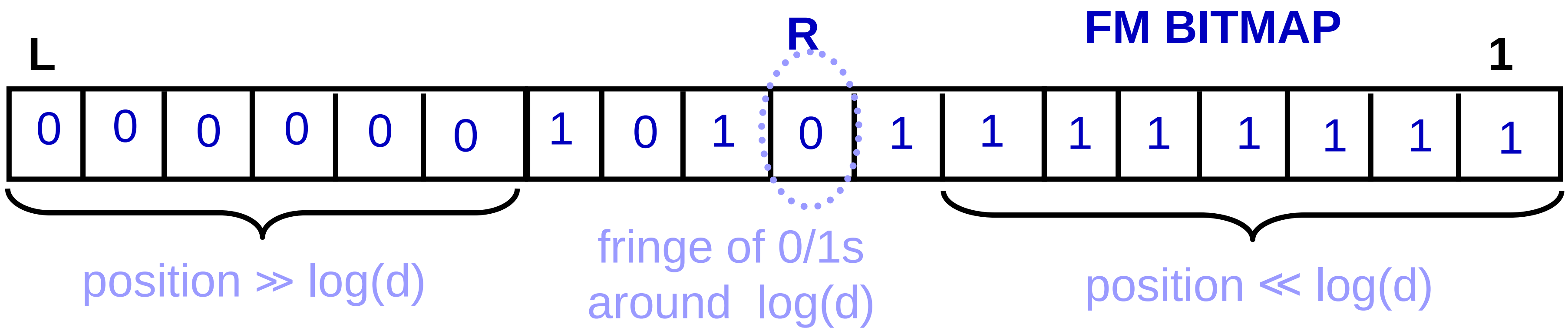
FM Sketch

- 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
0	0	1	0	1	1

+

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

=

6	5	4	3	2	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.

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)… (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.

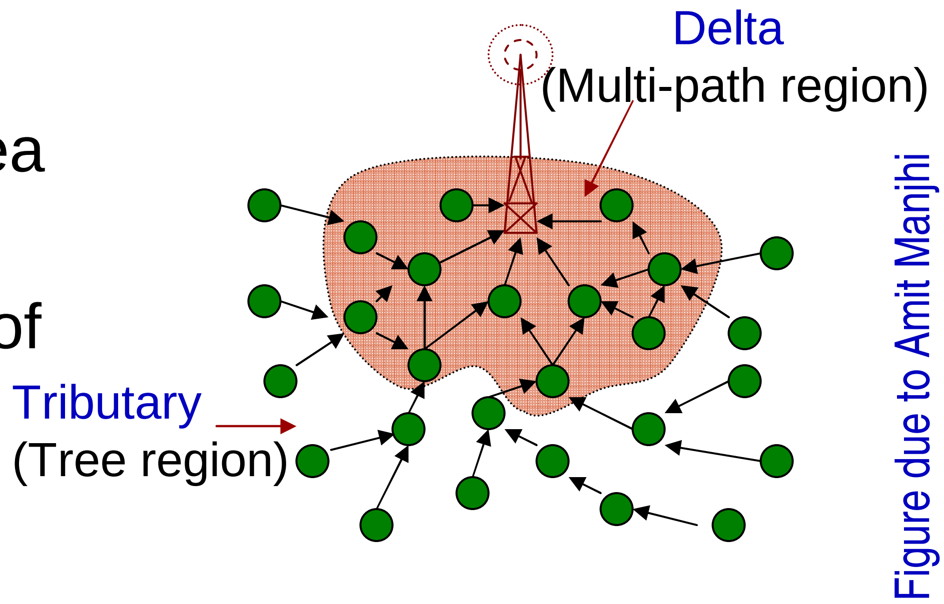
Other applications of FM in ODI

- 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

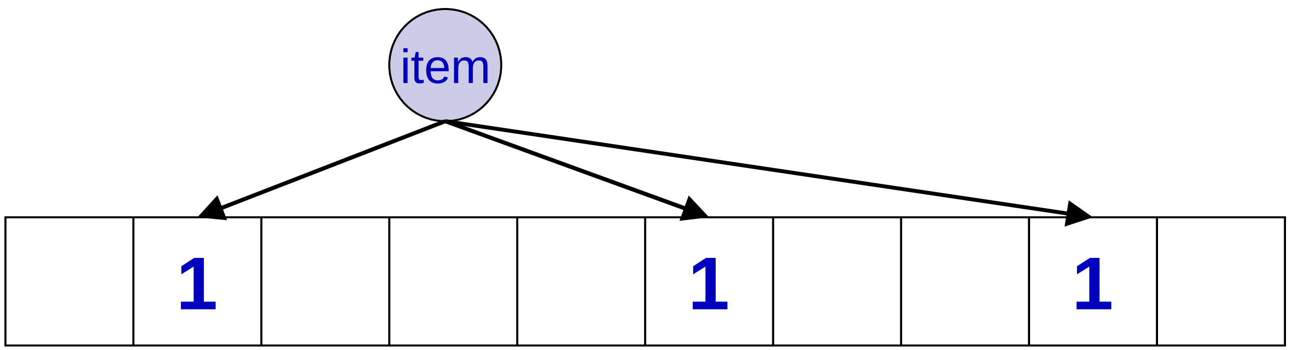
Combining ODI and Tree

- *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



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



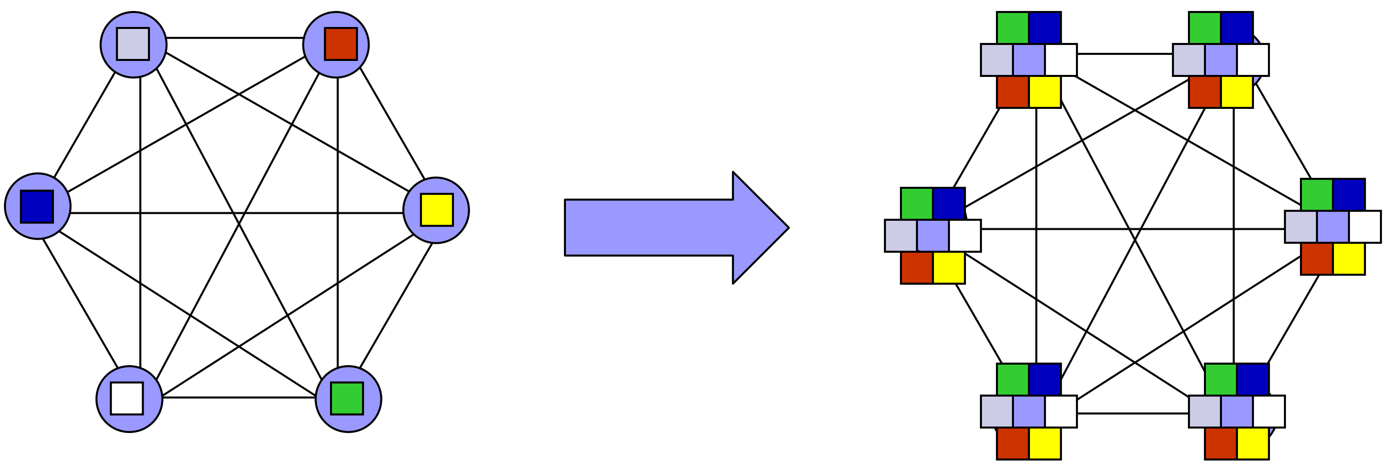
- Bloom filters are ODI, and merge like FM sketches

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.?

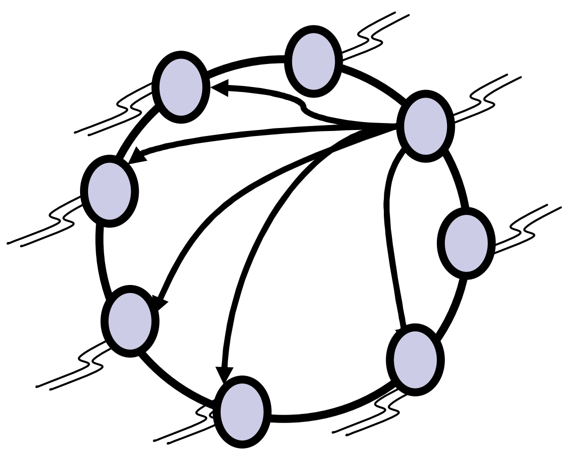
6	5	4	3	2	1
0	1	1	0	1	1

Decentralized Computation and Gossiping



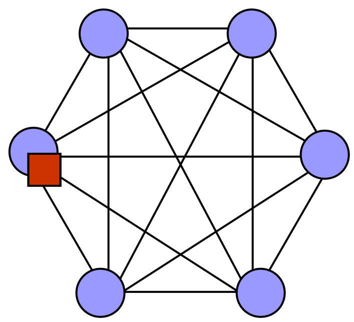
Decentralized Computations

- 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



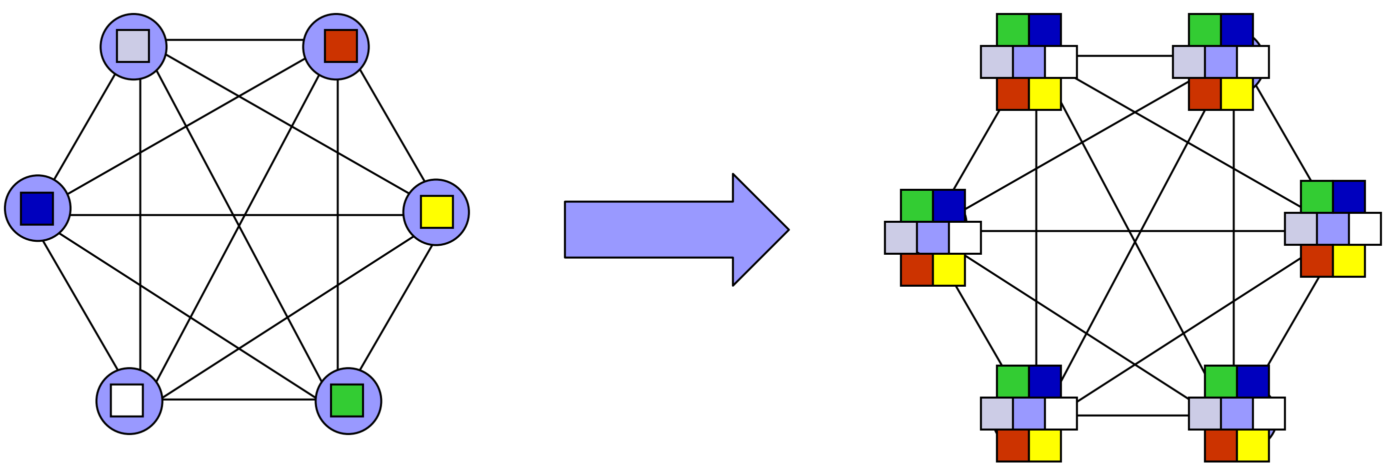
Gossiping

- "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

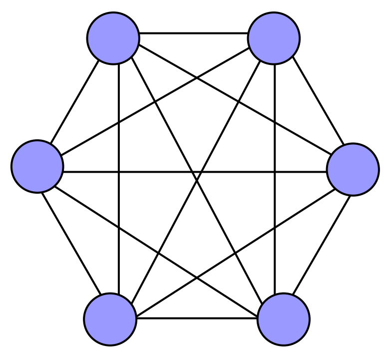


ODI Gossiping

- 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.

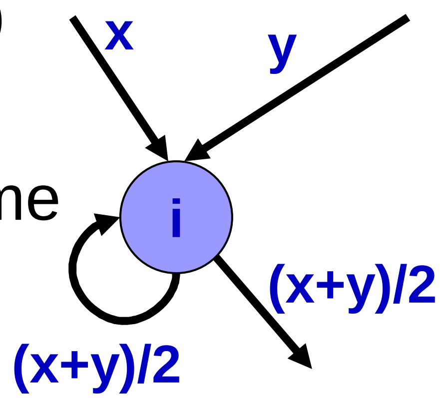
Aggregate Gossiping

- 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".

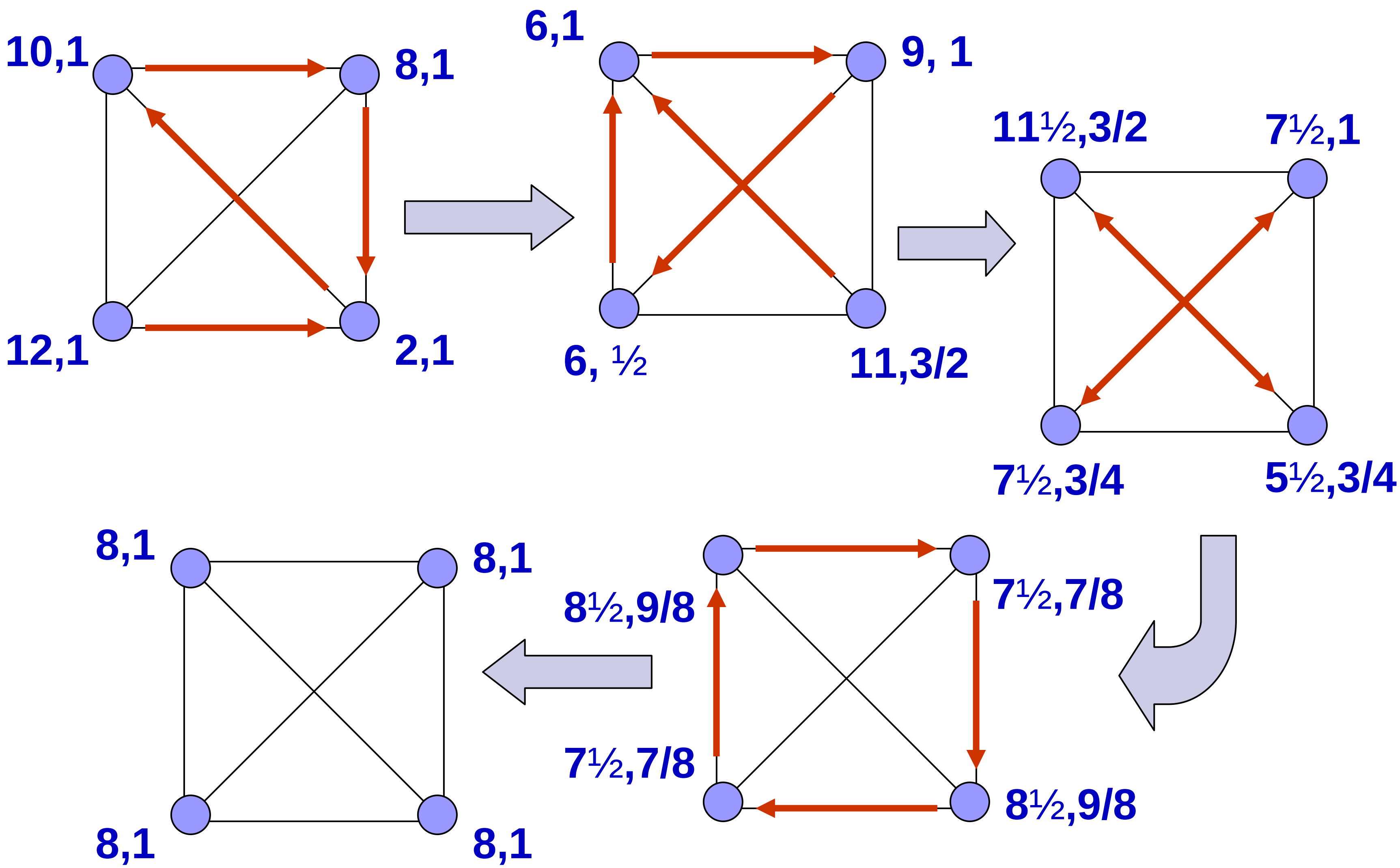


Push-Sum

- 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 $(½\ sum_i^t, ½ count_i^t)$ 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

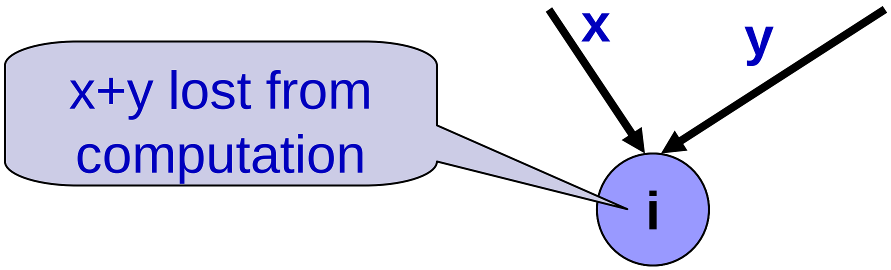


Convergence Speed

- 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?