Robust Aggregation in Sensor Networks

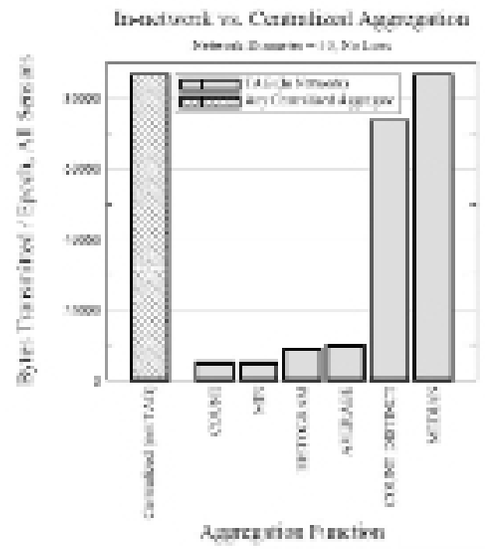
Jie Gao
Computer Science Department
Stony Brook University

STONY BROOK

Papers

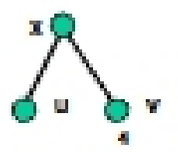
- [Shrivastava04] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divy Agrawal, Subhash Suri, Medians and Beyond: New Aggregation Techniques for Sensor Networks, ACM SenSys '04, Nov. 3-5, Baltimore, MD.
- [Nath04] Suman Nath, Phillip B. Gibbons, Zachary Anderson, and Srinivasan Seshan, Synopsis Diffusion for Robust Aggregation in Sensor Networks". In proceedings of ACM SenSys'04.
- [Considine04] Jeffrey Considine, Feifei Li, George Kollios, and John Byers, Approximate Aggregation Techniques for Sensor Databases, Proc. ICDE, 2004.
- [Przydatek03] Bartosz Przydatek, Dawn Song, Adrian Perrig, SIA: Secure Information Aggregation in Sensor Networks, Sensys'03.

Problem I: median



Problem I: median

- Computing average is simple on an aggregation tree.
  - Each node x stores the average a(x) and the number of nodes in its subtree n(x).
  - The average of a node x can be computed from its children u, v. n(x)=n(u)+n(v). a(x)=(a(u)n(u)+a(v)n(v))/n(x).
- Computing the median with a fixed amount of message is hard.
  - We do not know the rank of u's median in v's dataset.
  - We resort to approximations.



Median and random sampling

- Problem: compute the median a of n unsorted elements $\{a_i\}$.
- Take a random sample of k elements. Compute the median x.
- Claim: x has rank within $(\frac{1}{2}+\varepsilon)n$ and $(\frac{1}{2}-\varepsilon)n$ with probability at least $1-2/\exp\{2k\varepsilon^2\}$. (Proof left as an exercise.)
- Choose $k=\ln(2/\delta)/(2\varepsilon^2)$, then x is an approximate median with probability $1-\delta$.
- A deterministic algorithm?
- How about approximate histogram?
- What if a sensor generates a list of values?

Quantile digest (q-digest)

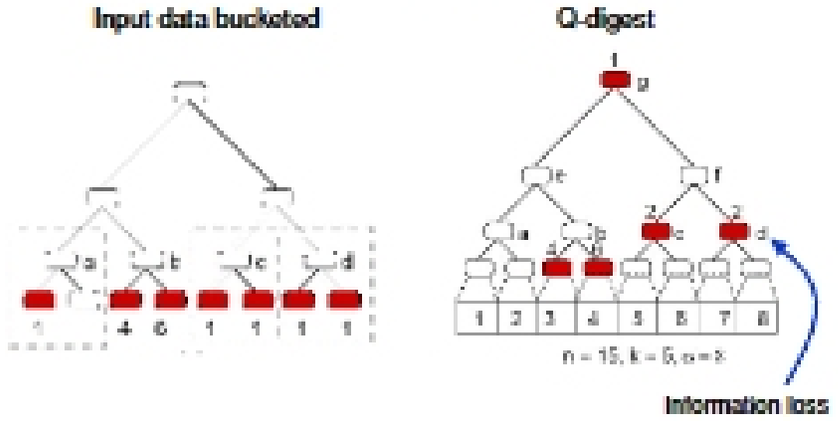
- A data structure that answers
  - Approximate quantile query: median, the kth largest reading.
  - Range queries: the kth to lth largest readings.
  - Most frequent items.
  - Histograms.
- Properties:
  - Deterministic algorithm.
  - Error-memory trade-off.
  - Confidence factor.
  - Support multiple queries.

Q-digest

- Exact data: frequency of data value $\{f_1, f_2, \ldots f_\sigma\}$.
- Compress the data:
  - detailed information concerning frequent data are preserved;
  - less frequently occurring values are lumped into larger buckets resulting in information loss.
- Buckets: the nodes in a binary partition of the range $[1, \sigma]$. Each bucket v has range [v.min, v.max].
- Only store non-zero buckets.
- Digest property:
  - Count(v) ≤ n/k. (except leaf)
  - Count(v) + Count(p) + Count(s) > n/k. (except root)

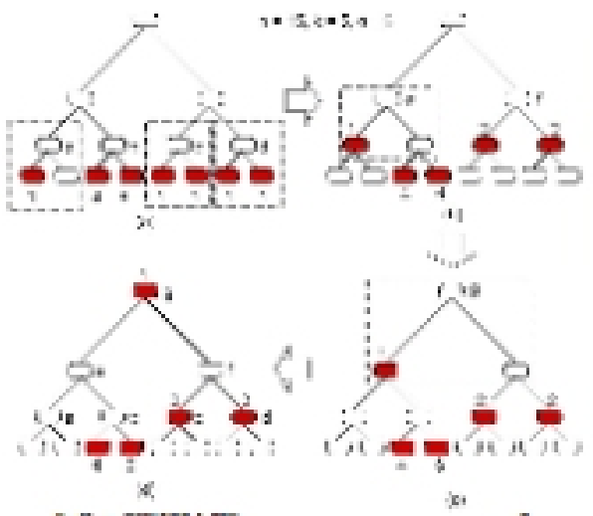
parent sibling

Example



Construct a q-digest

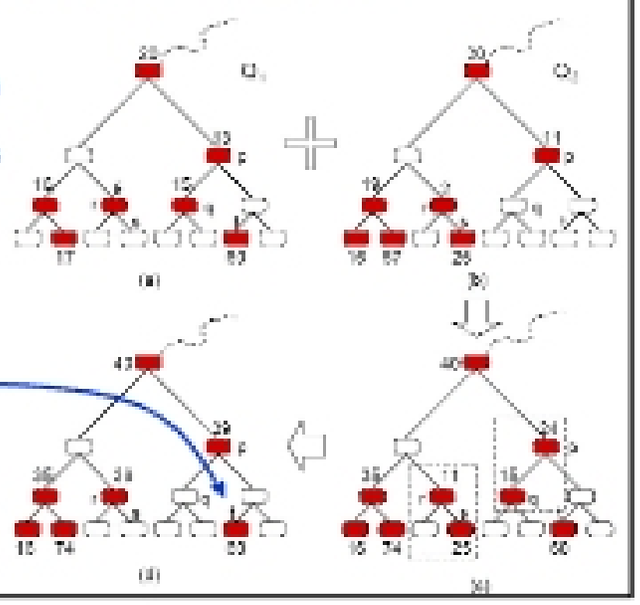
- Each sensor constructs a q-digest based on its value.
- Check the digest property bottom up: two "small" children's count are added up and moved to the parent.



Merging two q-digests

- Merge q-digests from two children
- Add up the values in buckets
- Re-evaluate the digest property bottom up.

Information loss: it undercounts since some of its value appears on ancestors.



Space complexity and error bound

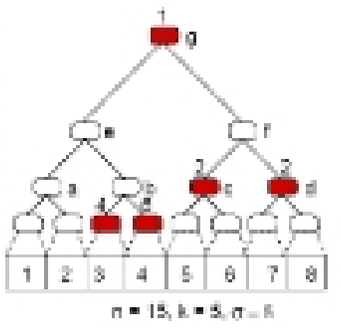
1. A q-digest with compression parameter k has at most 3k buckets.
- By property 2, for buckets Q,
  - $\Sigma_{v \in Q}$ [Count(v) + Count(p) + Count(s)] > |Q| n/k.
  - $\Sigma_{v \in Q}$ [Count(v) + Count(p) + Count(s)] ≤ $3\Sigma_{v \in Q}$Count(v)=3n.
  - |Q|<3k.

- Any value that should be counted in v can be present in one of the ancestors.

1. Count(v) has max error $\log\sigma$·n/k.
   - Error(v) ≤ $\Sigma_{\text{ancestor } p}$ Count(p) ≤ $\Sigma_{\text{ancestor } p}$ n/k ≤ $\log\sigma$ n/k.
2. MERGE maintains the same relative error.
   - Error(v) ≤ Σ_i Error(v_i) ≤ Σ_i $\log\sigma$ n_i/k ≤ $\log\sigma$ n/k.

Median and quantile query

- Given q∈(0, 1), find the value whose rank is qn.
- Relative error ε=|r-qn|/n, where r is the true rank.
- Post-order traversal on Q, sum the counts of all nodes visited before a node v, which is the lower bound on the # of values less than v.max. Report it when it is first time larger than qn.
- Error bound: $\log\sigma$ /k = $3\log\sigma$ /m, where m=3k is the storage bound for each sensor.

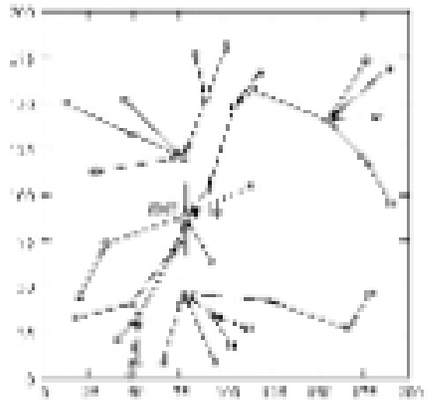


Other queries

- Inverse quantile: given a value, determine its rank.
  - Traverse the tree in post-order, report the sum of counts v for which x<v.max, which is within [rank(x), rank(x)+εn]
- Range query: find # values in range [l,h].
  - Perform two inverse quantile queries and take the difference. Error bound is 2εn.
- Frequent items: given s∈ (0, 1), find all values reported by more than sn sensors.
  - Count the leaf buckets whose counts are more than sn.
  - Small false positive: values with count between (s-ε)n and sn may also be reported as frequent.

Simulation setup

- A typical aggregation tree (BFS tree) on 40 nodes in a 200 by 200 area. In the simulation they use 4000~8000 nodes.

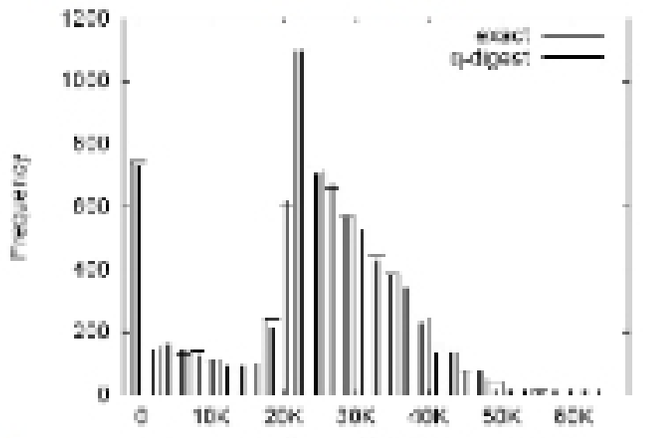


Simulation setup

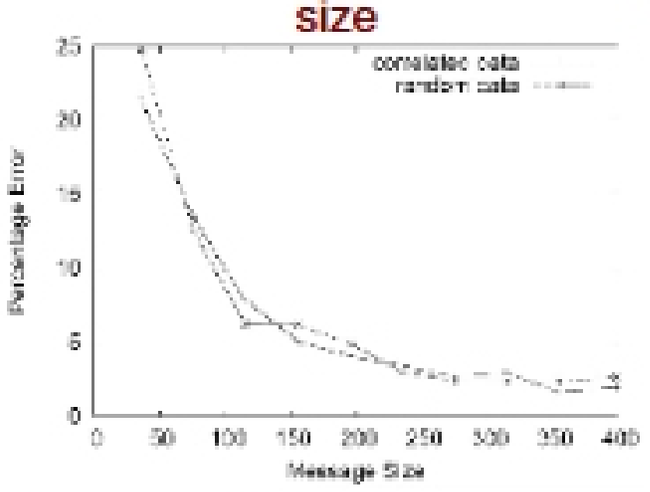
- Random data;
- Correlated data:3D elevation value from Death Valley.

Histogram v.s. q-digest

- Comparison of histogram and q-digest.



Tradeoff between error and msg size



Saving on message size