Power-Conserving Computation of Order-Statistics over Sensor Networks

Michael B. Greenwald
Dept. of Comp. & Inf. Sci.
University of Pennsylvania
Philadelphia, PA 19104
greenwald@cis.upenn.edu

Sanjeev Khanna*
Dept. of Comp. & Inf. Sci.
University of Pennsylvania
Philadelphia, PA 19104
sanjeev@cis.upenn.edu

ABSTRACT

We study the problem of power-conserving computation of order statistics in sensor networks. Significant power-reducing optimizations have been devised for computing simple aggregate queries such as COUNT, AVERAGE, or MAX over sensor networks. In contrast, aggregate queries such as MEDIAN have seen little progress over the brute force approach of forwarding all data to a central server. Moreover, battery life of current sensors seems largely determined by communication costs — therefore we aim to minimize the number of bytes transmitted. Unoptimized aggregate queries typically impose extremely high power consumption on a subset of sensors located near the server. Metrics such as total communication cost underestimate the penalty of such imbalance: network lifetime may be dominated by the worst-case replacement time for depleted batteries.

In this paper, we design the first algorithms for computing order-statistics such that power consumption is balanced across the entire network. Our first main result is a distributed algorithm to compute an ϵ-approximate quantile summary of the sensor data such that each sensor transmits only $O(\log^2 n/\epsilon)$ data values, *irrespective of the network topology*, an improvement over the current worst-case behavior of $\Omega(n)$. Second, we show an improved result when the height, h, of the network is significantly smaller than n. Our third result is that we can exactly compute any order statistic (e.g., median) in a distributed manner such that each sensor needs to transmit $O(\log^3 n)$ values.

Further, we design the aggregates used by our algorithms to be *decomposable*. An aggregate Q over a set S is decomposable if there exists a function, f, such that for all $S = S_1 \cup S_2$, $Q(S) = f(Q(S_1), Q(S_2))$. We can thus directly apply existing optimizations to decomposable aggregates that increase error-resilience and reduce communication cost.

Finally, we validate our results empirically, through simulation. When we compute the median exactly, we show that, even for moderate size networks, the worst communication cost for any single node is several times smaller than the corresponding cost in prior median algorithms. We show similar cost reductions when computing approximate order-statistic summaries with guaranteed precision. In all cases, our total communication cost over the entire network is smaller than or equal to the total cost of prior algorithms.

*Supported in part by an Alfred P. Sloan Research Fellowship and by an NSF Career Award CCR-0093117.

PODS 2004 June 14-16, 2004, Paris, France.

1. INTRODUCTION

We consider the problem of computing complex aggregates over sensor networks in a manner that conserves power to maximize network lifetime. The dominant consumer of power in such networks is communication, and thus network life time is determined by the maximum communication load at any node. In this paper, we design the first algorithms for computing order-statistics such that power consumption is balanced across the entire network.

1.1 Motivation

Sensor networks are large ad-hoc networks of interconnected battery powered, wireless, sensors. Onboard general-purpose processors make these sensors "smart", and enable them to use their data communication links for self-configuration. Long-lived batteries and low power consumption provide independence and freedom from maintenance. This collection of features makes such networks easy to deploy and maintain in difficult environments. Sensor nets promise a qualitative change in the way we study the real world, the way we deploy warning systems to protect us from earthquakes and storms, and the way we will eventually build self-adapting physical systems.

Physical deployment is only one step towards achieving this promise. We need also understand how to manage and use such networks. The primary operation on sensor networks is extracting data from the sensors. As such, it has been argued persuasively [2, 13, 14] that a declarative interface, much like traditional database query languages, is well suited to data extraction from sensor networks. Commonly, queries request values that are not simply a list of individual sensor values, but *aggregates*, functions that summarize a collection of observations [13, 2, 11]. Examples of aggregates include MAX, MEDIAN, COUNT, and AVERAGE. A query defines the information to be extracted from the network, a query optimizer produces a plan of extraction, and then the sensors deliver a stream of values that converge toward a powered basestation at the root of the sensor network. Queries are directed at a single table representing the entire sensor network. The table does not "exist" in memory, but logically consists of a growing set of records, one record for each sensor at each instant in time. Each record contains one column for each type of sensor and/or attribute at each node. For example, records may contain a column for sensor-types such as light, sound, or temperature, perhaps as well as columns for location, or node-id, where possible and useful[1]. An aggregate query may request, for example, the median value of a single column.

A network of sensors differs from standard computing platforms along many axes. Nodes in the network operate unsupervised for long periods of time, they may intermittently lose connectivity, and have relatively limited bandwidth and computational power. Arguably, the most significant difference is that nodes operate under severe power constraints, requiring applications running on top of the sensor network to be power-aware. A central problem in deploying sensor networks is thus maintaining battery-life through power-conservation. Power consumption of current sensors seems dominated by the cost of transmitting and receiving messages; computation is comparatively cheap[2].

Network lifetime is determined, however, not simply by the aggregate power consumed by all of the nodes, but by the *maximum power consumption* at any node [14]. The node that uses the most power will exhaust its battery first. Typically, the subset of sensors closest to the root server(s) consume the most power. Unfortunately, these nodes are also the most critical to maintain network connectivity. If the nodes near the root are down, they partition the remainder of the network from the root, rendering the entire net useless. This implies that minimizing total communication cost is insufficient; we must minimize the *maximum power consumed* by, or alternately the maximum communication load at, any network node. We say the power consumption is *balanced* if the load on any two nodes in the network differ by at most a poly-log factor.

1.2 Our Results

An aggregate Q over a set S is decomposable if there exists a combining function, f, such that for all $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$, $Q(S) = f(Q(S_1), Q(S_2))$. In other words, we can compute a new aggregate over the union of two disjoint sets, by simply combining their aggregates, needing no access to the individual observations in each set. Some examples of decomposable[3] aggregates are MAX, COUNT, and AVERAGE. It is easy to see that these aggregates can be computed with a maximum communication load of $O(1)$ data values at each node. In contrast, general order statistic queries such as MEDIAN or the ith largest element, are not decomposable, and the best known previous implementations impose a worst-case communication load of $\Omega(n)$ data values (each node sends all data values in its subtree) [13]. Approximate algorithms with smaller space requirements are known for streaming data, but have not been adapted to the setting of sensor networks. In this paper, we design algorithms to compute arbitrary order-statistics, approximately and exactly, with a worst-case communication load of poly-log(n) data values at any network node.

Our first main result is a distributed algorithm to compute an ϵ-approximate quantile summary of the sensor data such that each sensor transmits only $O(\log^2 n/\epsilon)$ data values, *irrespective of the network topology*. An *ϵ-approximate quantile summary* of an n-element data set is a summary that allows us to answer *any* order-statistic query to within an additive rank error guaranteed to be at most ϵn. Second, we show an improved result when the height, h, of the network is significantly smaller than n. Our third result is that we can exactly compute any order statistic (e.g., median) in a distributed manner such that each sensor needs to transmit $O(\log^3 n)$ values. The fact that our algorithms perform well independent of topology is important in practice, because node placement in sensor networks results in highly irregular topologies [6].

Further, we design the aggregates used by our algorithms to behave like decomposable and *non-holistic* aggregates (in which the aggregate is smaller than the underlying set). Significant power-reducing optimizations have been devised for non-holistic decomposable aggregates [13]. The most important optimization for non-holistic decomposable aggregates is simply that aggregation can occur *in the network*, reducing the communication cost. Nodes combine their children's aggregates and forward a single aggregate, rather than forwarding messages for *every* descendant in their subtree. Other optimizations increase error-resilience by scaling the aggregate by $1/k$ and sending to each of k parents, others reduce communication cost by inhibiting communication if the aggregate of a subtree is "close" to a guess passed down from the root, and others increase coverage by using cached prior aggregates to recover from loss of subtrees. Thus, by designing our algorithms to use decomposable aggregates, we can directly benefit from all existing optimizations.

Simple aggregate queries such as COUNT, AVERAGE, or MAX over sensor networks have constant size decomposable aggregates, and all the above optimizations can be applied with great effect. In contrast, aggregate queries such as MEDIAN have seen little progress over the brute force approach of forwarding all data to a central server, and then locally computing the median value. We present here a power-efficient algorithm for computing approximate quantile summaries of the whole data set. The summary algorithm is important because we can use it to convert *any* computation currently implemented by shipping all observations to the basestation to be more efficient and decomposable, at the price of approximate rather than exact results. However, our exact quantile query algorithm (e.g. median) demonstrates how one can use the approximate summary to get exact results while still gaining in efficiency.

Although the result of a query over a sensor network is typically a stream of values, we analyze the cost of sending only a single value back to the basestation. The optimizations described above, and others, can be applied to our algorithm to reduce the cost of subsequent values in the stream — however they cannot be applied to the conventional im-

[1] "Timestamp" can also be considered a column in this table, but because of physical resource constraints, records with non-current timestamps are unlikely to persist in the table for long.

[2] Representative power costs can be gleaned from [20, 4, 13, 14]: Including processor cost, each bit requires 4000-4500 nJ to send. A single instruction on a 5mW processor running at 4MHz only consumes 4-5nJ. So transmitting a single bit costs as much as 800-1000 instructions.

[3] "Decomposable" is a term used in the sensor-network community. The corresponding term in database literature (c.f. [8]) is *non-holistic*.

plementation of median and quantile summary algorithms. Thus, analyzing the cost of a single value is a conservative measure, adopting a worst-case scenario for our algorithm.

1.3 Sensor vs. Streaming Computation

Computing responses to queries over sensor streaming data bears obvious similarities to computing the same responses over data streams, an area that has received great attention in recent years. Both attempt to answer questions about ephemeral data that does not lie in memory, cannot be directly accessed, and may be delivered to processors in orders we cannot control. Motwani *et al* [18] provide an overview of issues and progress in managing data streams. Madden and Franklin [12] provide a good introduction to sensor streaming data.

Nevertheless significant differences separate the two, and results over data streams cannot, in general, be directly applied to streaming sensor data. First, when sensor values change slowly enough, sensor networks can use multiple passes over the data. Second, uniformly sampling values from a sensor network with unknown topology is complex: sampling nodes near the leaves are more expensive than sampling near the root, sampling rate should correspond to density — but density is unknown, and loss of a single message can eliminate the values of an entire subtree. Sensor networks are inherently parallel, and no single processor observes all of the values in the stream. To compare sets of values observed in disjoint subtrees of the network, one either pays both communication and memory costs, or else aggregates the values in one subtree before seeing the values in the other, thereby discarding information. It is easy to see that *the sensor net model is at least as hard as the streaming model.* Space-efficient computation of a quantile summary in the streaming model corresponds to a communication-efficient computation of quantile summary in the sensor net model when the communication topology is a linear chain. The message sent up by the ith node to its parent can be interpreted as the quantile summary after i observations and vice versa. Thus, any topology-independent algorithm in the sensor net model can be immediately applied to the streaming model, but the converse is not true.

1.4 Related work

Prior algorithms for in-network aggregation that balance power-consumption across the network do exist for sensor networks, but only for simple aggregates [13, 21]. However, there the balanced power property falls out naturally (and trivially) because most such aggregates are constant size (*e.g.* COUNT, AVERAGE, or MAX). Aggregates that balance power consumption are discussed even in sensor networks that are not addressed using database-like declarative queries [22], but here, too, balanced power consumption is a by-product of constant size aggregates.

The issue of power consumption of individual nodes as bottleneck is mentioned, but not specifically addressed, by Madden *et al* in [13]. In later work on TinyDB, Madden et al [14] address the issue directly, but not algorithmically. Rather, users can specify the desired lifetime of the network through the `QUERY LIFETIME` clause, and Tiny DB computes the sample interval necessary to guarantee that each node will last for that lifetime. The sample interval of the entire tree is limited by the sample rate of the nodes near the root. For non-trivial aggregates, algorithms with balanced power consumption could support finer sample granularity for a given lifetime.

There has been little work, to our knowledge, on balanced power-consumption algorithms for order statistics. [13] categorizes MEDIAN as holistic, and therefore unoptimizable. Similarly, Zhao *et al* [22] does not consider optimizing medians, because they are not decomposable. Nevertheless, these unoptimizable aggregates are considered desirable by users, who felt the simple summarizations like average and count were too limited [10]. For similar reasons, [15] proposes tackling median and other more generalized aggregation predicates in future work.

Hellerstein *et al* discusses some of these more general aggregates in [10]. In particular, they describe an implementation of a multi-resolution histograms that provide approximate summaries of the distribution of sensor readings, with successively finer resolution buckets at the cost of additional rounds of communication. This work is based on wavelet histograms[16], but unlike the quantile summary algorithms reported here, they do not provide any *guarantees* on accuracy, despite the existence of wavelet histograms with error guarantees for streaming data [7].

Dobra *et al* use sketches to approximately answer complex aggregate queries over streaming data [5]. [1] discusses using sampling to obtain a small set representing a larger database, and operating on the reduced set to obtain quick but approximate answers. Several implementations of quantile summaries exist for streaming data [9, 17, 3]. In this paper we chose to extend the currently most space-efficient deterministic implementation of quantile summaries, GK [9], that uses $O(\log(\epsilon n)/\epsilon)$ space in the streaming model. The individual quantiles summaries in the summary sets used in this paper share their representation (maintaining $r_{min}()$ and $r_{max}()$ for each stored observation). It may be equally possible to extend MRL summaries (Manku, Rajagopalan and Lindsay [17]) or CM sketches [3] to sensor networks — both have decomposable summaries, and algorithmically may be simpler to combine without loss of precision or blowup in space. CM seems easier to extend (presumably you need only ensure that each node chooses the same set of $\ln(1/\delta)$ hash functions, and the same set of $\log(n)$ dyadic sums, and trivially adding the corresponding terms when merging the summaries of each child). However, the resulting summary only has probabilistic accuracy guarantees, and the space requirements are worse by a factor of at least $\log(1/(\epsilon\delta))$. The extension to MRL is not straightforward: the order and number of COLLAPSE operations will be constrained by the topology of the network. The impact on the space and accuracy of the algorithm is unclear.

Finally, our multi-pass algorithm for computing exact order statistics (such as medians) using approximate summaries with narrower and narrower ranges builds on the foundational work of Munro and Paterson on multi-pass selection and sorting [19].

2. PRELIMINARIES

Throughout this paper we assume that the sensor network is a tree. In the first sensor networks, nodes attached themselves to the network as close to the root as possible, resulting in shallow routing trees. Today, more sophisticated algorithms attempt to self-organize the routing tree in a more balanced manner. Tomorrow, sensor network routing trees may also take underlying physical features into account. Consequently we make no specific assumptions about the network topology other than it is a tree.

We use n to denote the larger of the number of network nodes and the number of data points in our observation set. An order-statistic query over a data set S takes as input an integer $r \in [1..|S|]$ and outputs an element of rank r in S. We say that the order-statistic query is answered with ϵ-accuracy if the output element is guaranteed to have rank within $r \pm \epsilon n$.

Following [9], a *quantile summary* for a set S is an ordered set $Q = \{q_1, q_2, ..., q_\ell\}$ along with two functions $\texttt{rmin}_Q$ and $\texttt{rmax}_Q$ such that

(i) $q_1 \leq q_2 ... \leq q_\ell$ and $q_i \in S$ for $1 \leq i \leq \ell$.

(ii) For $1 \leq i \leq \ell$, each q_i has rank at least $\texttt{rmin}_Q(q_i)$, and at most $\texttt{rmax}_Q(q_i)$ in S.

(iii) Finally, q_1 and q_ℓ are the smallest and the largest elements, respectively, in the set S, that is, $\texttt{rmin}_Q(q_1) = \texttt{rmax}_Q(q_1) = 1$, and $\texttt{rmin}_Q(q_\ell) = \texttt{rmax}_Q(q_\ell) = |S|$.

We say that a summary Q is an *ϵ-approximate quantile summary* for a set S if it can be used to answer any order statistic query over S with ϵ-accuracy, that is it can be used to compute the desired order-statistic within a rank error of at most $\epsilon|S|$. The proposition below describes a sufficient condition on the function $\texttt{rmin}_Q$ and $\texttt{rmax}_Q$ to ensure an ϵ-approximate summary.

Proposition 1. ([9]) *If a quantile summary satisfies the condition that for $1 \leq i < \ell$, $\texttt{rmax}_Q(q_{i+1}) - \texttt{rmin}_Q(q_i)$ is at most $2\epsilon|S|$, then it is an ϵ-approximate summary.*

In what follows, whenever we refer to a quantile summary as ϵ-approximate, we assume that it satisfies the conditions of Proposition 1.

3. APPROXIMATE ORDER STATISTICS

We now design an algorithm to compute ϵ-approximate quantile summaries for any $\epsilon > 0$, such that each node transmits $O(\log^2 n/\epsilon)$ data values irrespective of the underlying topology. We then present a more sophisticated variation of our main algorithm that exploits the height of the underlying topology, say h, to compute ϵ-approximate summaries with only $O(\log n \log(h/\epsilon)/\epsilon)$ data values transmitted by any node. Finally, we observe that a simplification of our main algorithm achieves ϵ-accuracy with only a transmission load of $O(h/\epsilon)$ at each node. This is of interest when the communication topology is a well-balanced tree, say, with height $O(\log n)$.

The high-level idea underlying our main algorithm is to compute quantile summaries in a bottom-up manner. Each node in the tree creates a set of $O(\log n)$ quantile summaries that collectively summarize the data contained in its subtree. It sends these summaries to its parent node, which then suitably merges together the quantile summaries from each of its children into its own set of local summaries, and sends merged summaries upwards to its own parent. The process continues until summaries arrive at the root which then merges together all received summaries into a single summary for the entire data set.

3.1 The Main Algorithm

We now describe in detail the structure of the sets of quantile summaries and the procedure for merging them. For ease of exposition, we will assume throughout that all data values are distinct. Our algorithms apply even when data values are not distinct.

3.1.1 Initial Quantile Summaries

Each node initially computes an $\epsilon/2$-approximate summary of any observations gathered by its sensor. This is done by simply sorting the set S of observations locally, and choosing, elements of rank $1, \epsilon|S|, 2\epsilon|S|, ..., |S|$. The resulting summary Q has at most $(1/\epsilon + 1)$ elements and satisfies the property that for any two consecutive elements q_i, q_{i+1} in the summary, $\texttt{rmax}_Q(q_{i+1}) - \texttt{rmin}_Q(q_i)$ is at most $\epsilon|S|$. By Proposition 1, it follows that Q is $\epsilon/2$-approximate.

3.1.2 Quantile Summaries at Intermediate Nodes

We describe here the structure of the set $\mathcal{Q}_v$ of summaries sent up by each node v of the tree. Let $\mathcal{Q}_v = \{Q_v^1, Q_v^2, ..., Q_v^k\}$ where each Q_v^i is a quantile summary covering a set of n_v^i data values. Let $\texttt{class}(Q_v^i) = \lfloor \log n_v^i \rfloor$, and let n_v denote the total number of data values covered by the subtree rooted at node v. The set Q_v satisfies the following three properties:

(a) $\sum_{i=1}^k n_v^i = n_v$.

(b) For any $1 \leq i < j \leq k$, the data values covered by Q_v^i and Q_v^j are disjoint.

(c) Finally, for $1 \leq i < j \leq k$, $\texttt{class}(Q_v^i) \neq \texttt{class}(Q_v^j)$.

An immediate consequence of the last property above is that $k \leq \log n$.

3.1.3 Merging Quantile Summaries at Intermediate Nodes

We now describe how a node u in the tree merges together quantile summaries received from its children nodes, say $v_1, v_2, ..., v_p$, along with quantile summary of data values locally gathered by it. At a high-level, the procedure works by repeatedly combining together quantile summaries of the same class and replacing them with a new quantile summary until no two summaries have the same class. This is refered to as the *combine* operation. Once combine is no longer applicable, we apply a *prune* operation to each resulting quantile summary so as to ensure that no summary contains more than $B + 1$ elements where we set $B = \log n/\epsilon$. The operation slightly reduces the accuracy – an ϵ'-approximate summary becomes $(\epsilon' + 1/(2B))$-accurate after we apply prune. After this phase, all summaries, at most one of each class, are forwarded by u to its parent node in the tree.

The Combine Operation Let $Q' = \{x_1, x_2, ..., x_a\}$ and $Q'' = \{y_1, y_2, ..., y_b\}$ be two quantile summaries. The operation $\texttt{combine}(Q', Q'')$ produces a new quantile summary $Q = \{z_1, z_2, ..., z_{a+b}\}$ by simply sorting the union of the elements in two summaries, and defining new rank functions for each element as follows. W.l.o.g. assume that z_i corresponds to some element x_r in Q'. Let y_s be the largest element in Q'' that is smaller than x_r (y_s is undefined if no such element), and let y_t be the smallest element in Q'' that is larger than x_r (y_t is undefined if no such element). Then

$$\texttt{rmin}_Q(z_i) = \begin{cases} \texttt{rmin}_{Q'}(x_r) & \text{if } y_s \text{ undefined} \\ \texttt{rmin}_{Q'}(x_r) + \texttt{rmin}_{Q''}(y_s) & \text{otherwise} \end{cases}$$

$$\texttt{rmax}_Q(z_i) = \begin{cases} \texttt{rmax}_{Q'}(x_r) + \texttt{rmax}_{Q''}(y_s) & \text{if } y_t \text{ undefined} \\ \texttt{rmax}_{Q'}(x_r) + \texttt{rmax}_{Q''}(y_t) - 1 & \text{otherwise} \end{cases}$$

Lemma 1. *Let Q' be an $\epsilon^{'}$-approximate quantile summary for a multiset S', and let $Q^{''}$ be an $\epsilon^{''}$-approximate quantile summary for a multiset $S^{''}$. Then* $\texttt{combine}(Q', Q^{''})$ *produces an $\overline{\epsilon}$-approximate quantile summary Q for the multiset $S = S' \cup S^{''}$ where $\overline{\epsilon} = \max\{\epsilon', \epsilon^{''}\}$.*

Proof. Let n' and $n^{''}$ respectively denote the number of observations covered by Q' and $Q^{''}$. Consider any two consecutive elements z_i, z_{i+1} in Q. By Proposition 1, it suffices to show that $\texttt{rmax}_Q(z_{i+1}) - \texttt{rmin}_Q(z_i) \le 2\overline{\epsilon}(n^{'} + n^{''})$. We analyze two cases. First, z_i, z_{i+1} both come from a single summary, say elements x_r, x_{r+1} in Q'. Let y_s be the largest element in $Q^{''}$ that is smaller than x_r and let y_t be the smallest element in $Q^{''}$ that is larger than x_{r+1}. Observe that if y_s and y_t are both defined, then they must be consecutive elements in $Q^{''}$.

$$\begin{aligned} \texttt{rmax}_Q(z_{i+1}) - \texttt{rmin}_Q(z_i) \le & \\ & [\texttt{rmax}_{Q'}(x_{r+1}) + \texttt{rmax}_{Q''}(y_t) - 1] \\ & -[\texttt{rmin}_{Q'}(x_r) + \texttt{rmin}_{Q''}(y_s)] \\ \le & \ [\texttt{rmax}_{Q'}(x_{r+1}) - \texttt{rmin}_{Q'}(x_r)] + \\ & [\texttt{rmax}_{Q''}(y_t) - \texttt{rmin}_{Q''}(y_s) - 1] \\ \le & \ 2\overline{\epsilon}(n^{'} + n^{''}). \end{aligned}$$

Otherwise, if only y_s is defined, then it must be the largest element in $Q^{''}$; or if only y_t is defined, it must be the smallest element in $Q^{''}$. A similar analysis can be applied for both these cases as well.

Next we consider the case when z_i and z_{i+1} come from different summaries, say, z_i corresponds to x_r in Q' and z_{i+1} corresponds to y_t in $Q^{''}$. Then observe that x_r is the largest element smaller than y_t in Q' and that y_t is the smallest element larger than x_r in $Q^{''}$. Moreover, x_{r+1} is the smallest element in Q' that is larger than y_t, and y_{t-1} is the largest element in $Q^{''}$ that is smaller than x_r. Using these observations, we get

$$\begin{aligned} \texttt{rmax}_Q(z_{i+1}) - \texttt{rmin}_Q(z_i) \le & \\ & [\texttt{rmax}_{Q''}(y_t) + \texttt{rmax}_{Q'}(x_{r+1}) - 1] \\ & -[\texttt{rmin}_{Q'}(x_r) + \texttt{rmin}_{Q''}(y_{t-1})] \\ \le & \ [\texttt{rmax}_{Q''}(y_t) - \texttt{rmin}_{Q''}(y_{t-1})] + \\ & [\texttt{rmax}_{Q'}(x_{r+1}) - \texttt{rmin}_{Q'}(x_r) - 1] \\ \le & \ 2\overline{\epsilon}(n^{'} + n^{''}). \end{aligned}$$

□

Corollary 1. *Let Q be a quantile summary produced by repeatedly applying the* `combine` *operation to an initial set of summaries $\{Q_1, Q_2, ..., Q_q\}$ such that Q_i is an ϵ_i-approximate summary. Then regardless of the sequence in which* `combine` *operations are applied, the resulting summary Q is guaranteed to be $(\max_{i=1}^{q} \epsilon_i)$-approximate.*

Proof. By induction on q. The base case of $q = 2$ follows from Lemma 1. Otherwise, $q > 2$, and we can partition the set of indices $I = \{1, 2, ..., q\}$ into two disjoint sets I_1 and I_2 such that Q is a result of the `combine` operation applied to summary Q' resulting from a repeated application of `combine` to $\{Q_i | i \in I_1\}$, and summary $Q^{''}$ results from a repeated application of `combine` to $\{Q_i | i \in I_2\}$. By induction hypothesis, Q' is $\max_{i \in I_1} \epsilon_i$-approximate and $Q^{''}$ is $\max_{i \in I_2} \epsilon_i$-approximate. By Lemma 1, then Q must be $\max_{i \in I_1 \cup I_2} \epsilon_i = \max_{i \in I} \epsilon_i$-approximate. □

The Prune Operation The prune operation takes as input an ϵ'-approximate quantile summary Q' and a parameter B, and returns a new summary Q of size at most $B + 1$ such that Q is an $(\epsilon' + (1/(2B)))$-approximate quantile summary for S. Thus `prune` trades off slightly on accuracy for potentially much reduced space. We generate Q by querying Q' for elements of rank $1, |S|/B, 2|S|/B, ..., |S|$, and for each element $q_i \in Q$, we define $\texttt{rmin}_Q(q_i) = \texttt{rmin}_{Q'}(q_i)$, and $\texttt{rmax}_Q(q_i) = \texttt{rmax}_{Q'}(q_i)$.

Lemma 2. *Let Q' be an ϵ'-approximate quantile summary for a multiset S. Then* $\texttt{prune}(Q')$ *produces an $(\epsilon' + 1/(2B))$-approximate quantile summary Q for S containing at most $B + 1$ elements.*

Proof. For any pair of consecutive elements q_i, q_{i+1} in Q, $\texttt{rmax}_Q(q_{i+1}) - \texttt{rmin}_Q(q_i) \le (\frac{1}{B} + 2\epsilon')|S|$. By Proposition 1, it follows that Q must be $(\epsilon' + 1/(2B))$-approximate. □

3.1.4 Merging Quantile Summaries at the Root

Once the root receives quantile summaries from all its children, it `combine`s together all summaries, irrespective of their class, and then applies `prune` operation to the resulting summary.

3.1.5 Overall Analysis

The total amount of data transmitted by any node is bounded by $O(Bk)$ where k is the number of distinct classes at the node. As noted earlier, k is bounded by $\log n$. Thus if we set the parameter $B = \log n/\epsilon$, to get an overall upper bound of $\log^2 n/\epsilon$ on the maximum size of transmission by any node. Our goal now is to show that the resulting quantile summary at the root is guaranteed to be ϵ-approximate.

Lemma 3. *For any node u in the tree, a class i quantile summary in the set $\mathcal{Q}_u$ has error at most $\epsilon_i = \epsilon/2 + (i/(2B))$.*

Proof. We will prove by induction on the height that for any node u in the tree, a class i quantile summary in the set $\mathcal{Q}_u$ has error at most $\epsilon_i = \epsilon/2 + (i/(2B))$. The base case is easy to verify. Now suppose that the hypothesis holds for all nodes with height less than j. Consider a node u at height j with children $v_1, v_2, ..., v_p$. Let $Q_1, Q_2, ..., Q_q$ be the resulting summaries just before the `prune` operation becomes applicable at node u. Consider a summary Q_ℓ obtained by combining summaries $Q'_1, ..., Q'_d$ in some arbitrary (valid) order. If $d = 1$, then Q_ℓ is identical to one of the summaries at a child node of u and `prune` operation will not alter this summary. By induction hypothesis, the summary satisfies the claim since its class remains unchanged. If $d \ge 2$, then $\texttt{class}(Q_\ell) > \texttt{class}(Q'_a)$ for $1 \le a \le d$ since each application of the `combine` operation increases the class size by one. Prior to the prune operation, we know by Corollary 1 that the summary Q_ℓ must be $\epsilon' = \max_{1\le a\le d} \epsilon_a$-approximate. After we apply the `prune` operation, by Lemma 2, the new summary is $(\epsilon' + 1/(2B))$-approximate. □

LEMMA 4. *The quantile summary computed at the root is ϵ-approximate.*

PROOF. We consider two cases. Suppose the root receives a quantile summary of class $\log n$. In this case, by the definition of class, entire data set is represented in this single summary, and there are no other quantile summaries received by the root. By Lemma 3, a class $\log n$ summary is ϵ-approximate, and since incoming summaries are already `prune`d, this summary is unaltered at the root and the theorem holds. Otherwise, each incoming summary has class at most $\log n - 1$, and by Lemma 3, all incoming summaries are guaranteed to be $(\epsilon - 1/(2B))$-approximate. The quantile summary eventually generated by the repeated application of `combine` at the root is thus $(\epsilon - 1/(2B))$-approximate by Corollary 1. Finally, by Lemma 2, an application of `prune` increases the error by $1/(2B)$, resulting in an ϵ-approximate summary. □

Summarizing the above analysis, we get the following result.

THEOREM 1. *There is a distributed algorithm to compute an ϵ-approximate quantile summary with a maximum transmission load of $O(\log^2 n/\epsilon)$ at each node.*

Sensor networks generally have irregular topologies, and we need to resort to `class` in order to bound the number of `prune` operations. However, if the height of the tree, h, is *very small* then a simpler and more efficient algorithm is possible.

THEOREM 2. *There is a distributed algorithm to compute an ϵ-approximate quantile summary with a maximum transmission load of $O(h/\epsilon)$ at each node.*

PROOF. At each node we `combine` all the data into a single summary Q', which we `prune` to h/ϵ elements. Each time we apply `prune` (at most h times) we increase ϵ' by $\epsilon/2h$. Thus, if summaries at the leaves have precision $\epsilon/2$, we are guaranteed that summaries are ϵ-approximate when they reach the root, with a maximum communication cost of only $O(h/\epsilon)$. □

3.2 An Improved Algorithm

We now refine the main algorithm above to exploit information about the communication topology, namely, its height h. If the height of the topology is $o(n)$, as in the case of many realistic topologies, the new algorithm gives stronger performance guarantees. However, if h is *much* smaller than n, e.g. $O(\log n)$ as is the case for balanced trees), then the $O(h/\epsilon)$ algorithm described in Theorem 2 is the best algorithm to use.

The idea is to introduce an additional operation, called `reduce` which ensures that no node in the tree sends up more than $O(\log(h/\epsilon))$ quantile summaries. The reduce operation is applied after the `combine` and the `prune` operations, just before a node is ready to forward its summary set to its parent. It increases the error in one of the summaries by $\epsilon/2h$. Since the operation is applied at most once at each node along the path to the root, the cumulative error as a result of this operation is at most $\epsilon/2$.

The input to `reduce` is a set $\mathcal{Q}_v = \{Q_1, Q_2, ..., Q_k\}$ of quantile summaries such that for $1 \le i < k$, $\texttt{class}(Q_i) < \texttt{class}(Q_{i+1})$. Output of reduce is a new set $\mathcal{Q}'_v$ of quantile summaries where $\mathcal{Q}'_v = Q_v$ if $k \le \log(8h/\epsilon)$. Otherwise, $\mathcal{Q}'_v = \{Q_j, ..., Q_{k-1}, Q'_k\}$ where j is the smallest index such that $\texttt{class}(Q_j) > \texttt{class}(Q_k) - \log(8h/\epsilon)$, and Q'_k is a new summary. In other words, we absorb the first $j-1$ summaries into Q_k. We now describe how this absorption is done:

(a) Each summary Q_i such that $1 \le i < j$, is first replaced by a new 2-element summary, containing the smallest and the largest elements represented by it. The function `rmin` and `rmax` remain unchanged for these elements.

(b) We then combine these modified summaries into Q_k just as in a `combine` operation even though the summaries have different classes. This produces a new summary Q'_k containing at most $B + 2(j-1) \le 3B$ elements, for any choice of $B \ge \log(n)$.

To analyze the error introduced by this step, suppose the class of the summary Q_k is α, that is, Q_k represents at least 2^α observations. Then total size of observations represented by $Q_1, ..., Q_{j-1}$ is at most $\epsilon 2^\alpha/2h$. As a result of the operations (a) and (b) above, we can introduce at most an error of $\epsilon/2h$ in Q_k (for instance, if all observations represented by $Q_1, Q_2, ..., Q_{j-1}$ fall between 2 consecutive elements in Q_k).

THEOREM 3. *There is a distributed algorithm to compute an ϵ-approximate quantile summary with a maximum transmission load of $O((\log n \log(\frac{h}{\epsilon}))/\epsilon)$ at each node.*

PROOF. We run the main algorithm presented in Section 3.1 along with an application of `reduce` before each broadcast of quantile summaries by a node to its parent. We slightly change some parameters of the algorithm: initial summaries are computed to be $\epsilon/4$-approximate and B is chosen to be $\log n/4\epsilon$. Following the analysis in Subsection 3.1.5, we know that the summary computed at the root is $\epsilon/2$-approximate with this setting of parameters if there is no `reduce` operation along the way. But each application of reduce increases the error by $\epsilon/2h$, bounding the cumulative error due to `reduce` to be $\epsilon/2$. Thus the final summary at the root is guaranteed to be ϵ-approximate. Finally, the bound on the maximum transmission load immediately follows from our choice of B and that the `reduce` operation ensures that no node sends more than $O(\log(\frac{h}{\epsilon}))$ summaries, with size of each summary bounded by $3B$. □

4. EXACT ORDER STATISTICS

We now build on the approximate quantile summary algorithm of the preceding section to design a multi-pass distributed algorithm that can exactly compute any particular order statistic. The new algorithm is exact rather than approximate, computes a specific quantile rather than summarizing all quantiles, requires multiple passes rather than a single pass, and has an overall transmission load of $O(\log^3 n)$ data values.

The basic idea of the algorithm is quite simple. Suppose we wish to find an element of rank r. In the first pass, the algorithm computes an $\epsilon/4$-approximate summary of the entire data set. We then query the summary for elements of

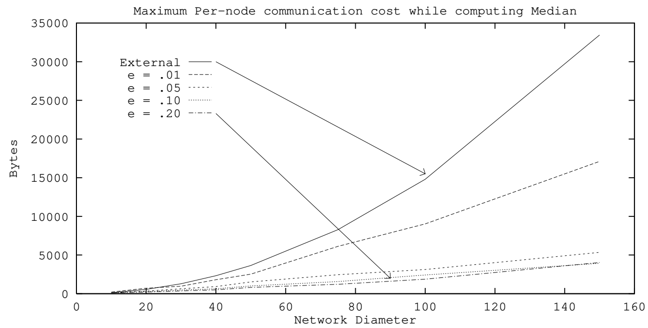

Figure 1: Total communication cost expended by worst-case node in the entire network in multiple passes, while computing the median. ϵ refers to the precision of the summaries used in each pass to get the exact result. Finer granularity ϵ increases the space used in each pass, but reduces the number of passes. The x-axis is the network diameter; there are x^2 nodes in the network.

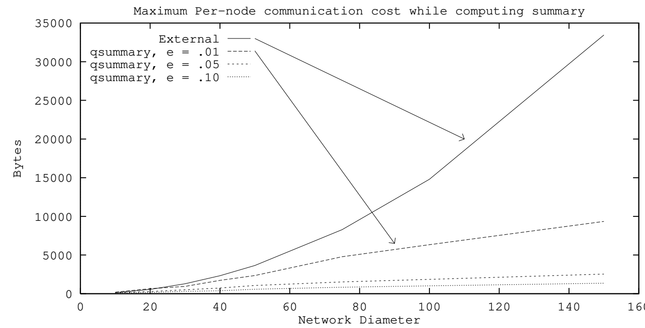

Figure 2: Number of bytes transmitted by worst-case node in the sensor network, while computing an ϵ-approximate summary of all the sensor readings. ϵ represents the precision of the final summary. Smaller ϵs produce more accurate summaries, but require more communication.

rank $r - (\epsilon n)/4$ and $r + (\epsilon n)/4$, respectively. Let x and y be the elements returned by the summary. Since the summary is $\epsilon/4$-approximate, we know that $\texttt{rank}(x) \leq r \leq \texttt{rank}(y)$, and $\texttt{rank}(y) \leq \texttt{rank}(x) + \epsilon n$. In the second pass, the algorithm broadcasts the elements x and y to all sensors, and now computes an $\epsilon/4$-approximate summary of only those data values that are in the interval $[x, y]$. Along the way, we also maintain a count of number of observations that are less than x, thus computing the exact rank of x. We now reset the rank value r to be $r - \texttt{rank}(x)$, and repeat the entire process again. After p passes, the length of the interval containing the element of rank r is at most $\epsilon^p n$. In at most $p = \log_{1/\epsilon}(n)$ passes, we are guaranteed that the interval containing element of rank r has length 1, giving us the element of rank r itself. Any fixed value of $\epsilon \in (0, 1)$ suffices to bound the total number of passes by $O(\log n)$. Smaller values of ϵ increase the maximum communication load in each pass while larger values of ϵ increase the total number of passes, giving us a tradeoff between the number of passes and the transmission load in each pass. We also observe that the above algorithm can be terminated after i passes to get an interval of length $\epsilon^i n$ containing the element of rank r. Combining with Theorem 1, we get the following theorem.

THEOREM 4. *There is a distributed algorithm to compute any order statistic exactly with an overall transmission load of $O(\log^3 n)$ data values at each node. Furthermore, we can answer any fixed order-statistic query to within an accuracy of δ, with an overall transmission load of $O(\log^2 n \log(1/\delta))$ data values.*

5. PRACTICAL CONCERNS

In terms of impact on real sensor network implementations, our work makes two contributions. First, our balanced power algorithms extend network lifetime by reducing maximum per-node power consumption. Second, cost can be further reduced because our approximate and exact order-statistic algorithms are amenable to the general optimization techniques introduced in earlier works (*e.g.* [13] and [22]) on aggregate queries over sensor networks, such as child caching and hypothesis testing.

In order to compare our balanced-power algorithm with earlier aggregation algorithms, we implemented our ϵ-approximate quantile summary and exact median computations on the sensor network simulator of Madden and Stanek (used in [13, 14, 10]). The median algorithm on that simulator, similar to all we are aware of, ships all observations to a central server and computes the median there. (Sampling implementations have been attempted, but are reported to have extremely inaccurate results).

We simulated sensor networks with a square grid topology, varying the diameter of the network. We expect our algorithm to perform well asymptotically, but it was unclear whether it would perform well at a small scale. Our measurements show that, after applying some simple optimizations (described in Appendix A), we always outperform centralized algorithms in both total and max per-node power. (Even without the optimizations, our maximum power consumption per-node is lower than the maximum in the centralized algorithms.)

Our measurements are summarized in Appendix B, but we briefly show some results here. We chose parameters that were, in some sense, pessimal for our algorithm. We chose a grid topology, with a relatively high branching factor near the root, because this gave the centralized algorithm the benefit from splitting the incoming observations into as many different paths as possible (our algorithm is indifferent to topology). We chose to simulate perfectly reliable radios, because our summaries can easily recover from losses (through child caching of summaries), while centralized algorithms cannot, without using space proportional to the size of the tree. Our computation of exact order statistics requires multiple passes. We allowed the sensor readings to change arbitrarily between passes (although the readings were drawn from the original distribution), rather than change slowly or with any relation to their previous value. The centralized algorithm is single-pass, and hence unaffected, but our exact median algorithm requires extra passes if it finds that it converged to a range that does not include the median. When measuring the per-node cost in our multi-pass computation, we capture some worst-case possibilities by post-facto combining the maximum communication cost in each pass to the *same* node, even if, in fact, the node consuming the most power differed between passes. In our simulation we had every sensor generate a unique value (although in Appendix B we show how our cost is much reduced when the number of distinct values obtained by sensor readings is smaller than the number of sensors).

Figures 1 and 2 show that even in our pessimistic simulation, the algorithms described here consume less power than our worst-case analysis and far better than prior implementations.

6. CONCLUSIONS

We presented the first algorithms for computing exact and approximate order statistics that balance power consumption over an entire sensor network. In contrast to earlier approaches that lead to a worst-case communication load of $\Omega(n)$ data values on some of the network nodes, our algorithms ensure that no node sends more than poly-log(n) data values. Our main results hold without regard (or even a priori knowledge of) the network topology; an important requirement in sensor networks which have irregular and dynamically changing topology. We also observe that the problem of computing communication-efficient approximate quantile summaries over sensor networks is at least as hard (and possibly harder) as the problem of computing space-efficient approximate quantile summaries in the data stream model. Specifically, the latter problem corresponds to the special case of the sensor net problem when the communication topology is a linear chain. We also note here that our algorithms use decomposable aggregates at each node, so the computations are amenable to standard in-network optimizations. Finally, we have shown through simulation that in practice the algorithms perform substantially better than prior, centralized, algorithms even over moderate size networks.

Acknowledgments

We thank the anonymous referees whose thorough reading and suggestions have helped improve our presentation.

7. REFERENCES

[1] Daniel Barbara, William DuMouchel, Christos Faloutsos, Peter J. Haas, Joseph M. Hellerstein, Yannis Ioannidis, H.V. Jagadish, Theodore Johnson, Raymond Ng, Viswanath Poosala, Kenneth A. Ross, and Kenneth C. Sevcik. The New Jersey Data Reduction Report. *Data Engineering Bulletin*, 20(4):3–45, December 1997.

[2] Phillipe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. In *Proceedings of Conference on Mobile Data Management*, January 2001.

[3] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *to appear in Proceedings of Latin American Theoretical Informatics (LATIN 2004)*, April 2004. Buenos Aires, Argentina (also available as DIMACS Tech Report TR-2003-20).

[4] Crossbow Technology, Inc. Wireless sensor networks: MICA, MICA2, and MICA2DOT. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

[5] Alin Dobra, Minos Garofalakis, Johannes Gehrke, and Rajeev Rastoghi. Processing complex aggregate queries over data stream. In *Proceedings of the 2002 ACM SIGMOD Intl. Conference on Management of Data*, pages 61–72, June 4-6 2002. Madison, WI.

[6] Deepak Ganesan, Sylvia Ratnasamy, Hanbiao Wang, and Deborah Estrin. Coping with irregular spatio-temporal sampling in sensor networks. In *Proceedings of Second Annual Workshop on Hot Topics in Networking (HOTNETS-II)*, November 20-21 2003. Cambridge, MA.

[7] Minos Garofalakis and Phillip B. Gibbons. Wavelet synopses with error guarantees. In *Proceedings of the 2002 ACM SIGMOD Intl. Conference on Management of Data*, pages 476–487, June 4-6 2002. Madison, WI.

[8] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. Data Cube: A relational operator generalizing group-by, cross-tabl and sub-totals. In *Proceedings of the 12th International Conference on Data Engineering (ICDE '96)*, pages 152–159, 1996.

[9] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the 2001 ACM SIGMOD Intl. Conference on Management of Data*, pages 58–66, May 2001.

[10] Joseph M. Hellerstein, Wei Hong, Samuel Madden, and Kyle Stanek. Beyond average: Toward sophisticated sensing with queries. In *Proceedings of 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, pages 63–79. (Lecture Notes in Computer Science 2634), Springer Verlag, April 2003. Palo Alto, CA.

[11] Chalermek Intanagonwiwat, Ramesh Govndan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual ACM/IEEE Conference on Mobile Computing and Networking (MobiCOM '00)*, pages 56–67, August 2000. Boston, MA.

[12] Samuel Madden and Michael J. Franklin. Fjording the stream: An architechture for queries over streaming sensor data. In *Proceedings of the ICDE*, 2002. San Jose, CA.

[13] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI '02)*, pages 131–146, December 9-11 2002. Boston, MA.

[14] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD International Conference on the Management of Data*, pages 491–502, June 9-12 2003. San Diego, CA.

[15] Samuel Madden, Robert Szewczyk, Michael J. Franklin, and David Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *In Proceedings of Workshop on Mobile Computing and Systems Applications*, 2002.

[16] Stephane Mallat. *A Wavelet tour of signal processing*. Academic Press, 2nd edition, 1999. London, UK.

[17] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 27(2):426–435, June 1998. Seattle, WA.

[18] Rajeev Motwani, Jennifer Widom, Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Gurmeet Singh Manku, Chris Olston, Justin Rosenstein, and Rohit Varma. Query processing, approximation, and resource management in a data stream management system. In *CIDR*, 2003.

[19] J. I. Munro and M.S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12:315–323, 1980.

[20] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[21] Yong Yao and Johannes Gehrke. Query processing for sensor networks. In *CIDR*, 2003.

[22] Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Computing aggregates for monitoring wireless sensor networks. In *Proceedings of the 1st IEEE International Workshop on Sensor Net Protocols and Applications (SNPA 2003)*, May 11 2003. Anchorage, AK.

APPENDIX

A. OPTIMIZING OUR IMPLEMENTATION

We here note that several simple optimizations are possible that are specific to the implementation of our algorithm. These optimizations improve performance in many cases, and never affect the worst-case analysis. We describe four optimizations used in our measurements: summary lists, value merging, conservative pruning, and early combining.

Summary lists: The quantile summaries used in this paper require 3 integers per stored observation: the value, rmin, and rmax. If summarizing the set of observations would require storing more than one third of the observations, then we both gain in precision and reduce space by storing the entire set of observations. This optimization reduces space mainly in small summaries, but there are many small summaries so the total space reduction is significant.

Value merging: If q_i is equal to q_{i+1} in summary Q, then we can remove q_i from Q without losing any information. We must adjust $\mathtt{rmin}_Q(q_{i+1})$ to $\mathtt{rmin}_Q(q_i)$, but can maintain $\mathtt{rmax}_Q(q_{i+1})$ without modification. This optimization reduces space only in large summaries, on average, where the count of covered observations is large compared to the number of distinct possible values.

Conservative Pruning: The `prune` operation takes as input a summary Q' and is required to produce as output a new summary, Q, over the same data set, but with size $B+1$ and $\epsilon \leq \epsilon' + 1/(2B)$. We can meet the precision requirement of ϵ but possibly reduce B substantially by means of a simple optimization. We scan the stored observations in Q and remove any observation q_i if $\mathtt{rmin}_Q(q_{i-1}) + 2\epsilon \geq \mathtt{rmax}_Q(q_{i+1})$.

Early Combining: Each `prune` operation increases ϵ by a fixed amount, and we have only allocated enough spare precision to `prune` each observation $\log n$ times. We specify that our algorithm combines and prunes only summaries of equal `class`, thus guaranteeing that any observation will participate in a `prune` operation at most $\log n$ times. However, it is equally safe to combine *any* set of summaries, provided only that we have enough small summaries to increase the `class` of the largest summary. If we combine k summaries together, then this can reduce the space requirement because we now represent the k summaries by a single summary of at most $B+1$ observations.

It is still the case that summaries will only be pruned when their `class` increases. It is clear that the `class` can only increase at most $\log n$ times (by definition, $\mathtt{class}(Q_v^i) = \lfloor \log n_v^i \rfloor$). Therefore we still satisfy both the precision and space requirements in the worst case, but can gain some improvements when a node has a large number of children.

A further optimization can be used as part of early combining: *relaxed epsilon*. We know that if the count of observations covered by a summary Q_i is n_i, then $\mathtt{class}(Q_i)$ can only be increased at most $\log(n) - \log(n_i)$ more times. Therefore we can safely increase ϵ to $\frac{\epsilon}{2}(1 + \frac{\log(n_i)}{\log(n)})$. This relaxation of epsilon to increase a number of steps at once allows conservative pruning to prune more observations during early combining.

B. PERFORMANCE MEASUREMENTS

The optimizations described in Appendix A can be divided into two classes. Value merging and summary lists are two optimizations that lose no information — they *maintain precision* of the summaries. All of the other optimizations listed above are *precision-reducing*. They discard information and deliver summaries with lower precision and lower cost. Precision-reducing summaries make sense if at any point in time we notice that the obtained precision is higher than necessary to meet the pre-specified precision we have guaranteed the user. In such cases we can shed *unnecessary* precision without violating our contract, thereby reducing communication costs. Without the precision-reducing optimizations, when users specify a given precision they may receive either a summary with the requested precision, or a more accurate but more costly summary. Enabling these precision-reducing optimizations makes the former far more likely.

We measure the relative cost reductions obtained by each of the loss-free optimizations, while enabling all of the precision-reducing optimizations.

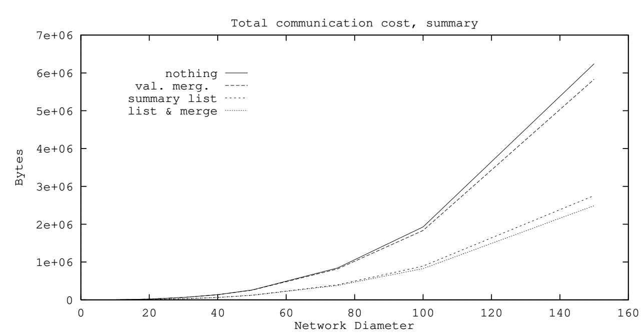

Figure 3: Total communication cost over entire network, while computing a .01-approximate summary, comparing effect of value merging and summary list optimizations.

Figure 3 shows the relative benefits of summary lists and value merging on the total communication costs over the entire network while computing a single summary with .01 precision. (In this measurement we limit the resolution of the sensors to produce only 1000 distinct values.) We compute total costs by summing the per-node costs over all nodes. Note that the number of nodes in the network grows with the square of the diameter. There is at least a constant per-packet overhead transmitted by every node, thus we expect an $O(n)$ term in the total network computation cost measured in Figure 3. We see this manifested by the quadratic-like shape of the curves in the figure.

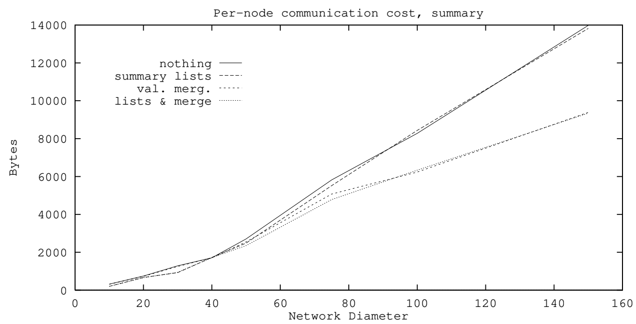

Figure 4: Per-node communication cost of most expensive node, while computing .01-approximate summary, comparing effect of value merging and summary list optimizations.

When we consider the communication costs summed over the entire network as a whole, then the cost generated by nodes sending only small summaries dominate, and therefore the summary list optimization has significant effect. Although value merging has a significant effect on reducing the maximum communication cost for the busiest nodes in the network (as we can see from Figure 4), it has little impact on the total cost. However, we are more concerned with bounding the worst-case per-node cost.

Figure 4 compares the two optimization's effect on worst-case per-node cost. Value merging has more significant effect as network size gets large; for small networks summary lists have a larger relative impact — but the network size is a far more significant factor than the choice of optimization, so the differences are obscured on this graph for small networks. There is no noticeable penalty for applying both optimizations, a choice that improves performance for both small and large networks.

Choice of precision has little effect on total cost, as can be seen in Figure 5, because many nodes send fewer than $1/\epsilon$ values, even for small ϵ, that they behave the same regardless of precision.

However, the effect of precision on maximum per-node cost can be significant, as can be seen in Figure 6, because lower precision summaries can be compressed significantly more as the number of observations covered by the summary gets large. As expected, for fixed network size, the maximum per-node cost is proportional to $1/\epsilon$. (These two graphs were constructed using both value merging and summary lists).

We would expect the impact of optimizations on median computations to be very similar to the impact on quantile summaries, as median is computed by repeatedly computing quantile summaries over narrower and narrower ranges. To a large degree this is true as can be seen in Figures 7 and 8.

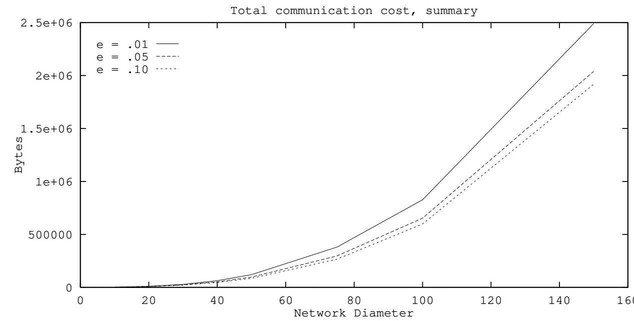

Figure 5: Total communication cost summed over entire network, while computing a single quantile summary of all the sensor readings. We compare the costs of different requested precision. ϵ denotes the accuracy of the resulting summary.

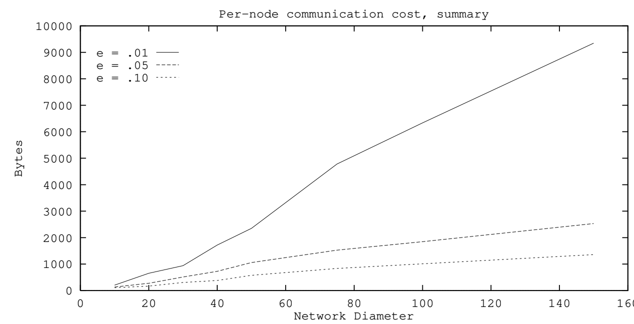

Figure 6: Per-node communication cost of most expensive node, while computing an ϵ-approximate summary of all the sensor readings. We compare costs for different values of ϵ.

However, in each successive pass the number of values (as well as the number of distinct values) is reduced by approximately a factor of ϵ, and consequently both optimizations have a correspondingly larger effect.

However, when we consider the effect of precision on the median computation we find a significant difference from the effect on computing individual quantile summaries. When we compute exact order statistics, including our computation of median, precision has no effect on the accuracy of the final result: it is always exact (that is, assuming the underlying data doesn't change significantly during the computation). Precision only controls communication costs: increased precision increases the cost of each pass in our multi-pass algorithm, but reduces the number of passes.

We observe that in Figure 9 the maximum per-node cost increases with precision. Increased precision increases message size, but reduces the number of passes. Therefore, in this case, the per-node cost is dominated by the message sizes in the earlier passes. This, in turn, implies that maximum message size diminishes sufficiently so that we can ignore the later passes.

However, Figure 10 shows that when considering total communication cost summed over the entire network, increasing precision reduces, rather than increases, cost. (Note that the order in the legend is reversed from Figure 9.) This is because, as we saw in earlier figures, the summed cost is fairly insensitive to precision. Therefore the number of

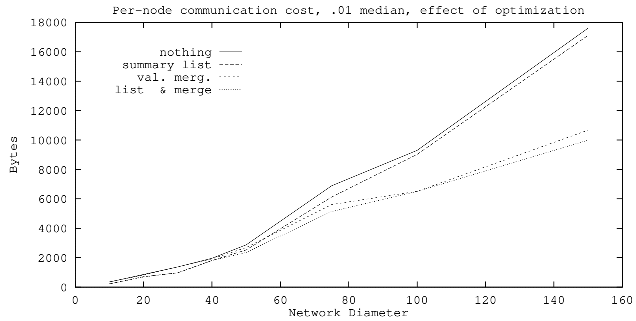

Figure 7: Total communication cost per-node, median, effect of optimization.

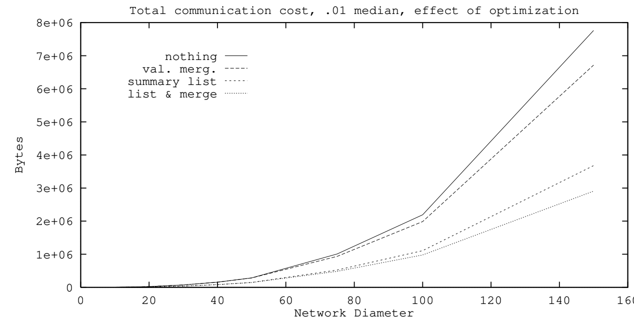

Figure 8: Total communication cost total, median, effect of optimization.

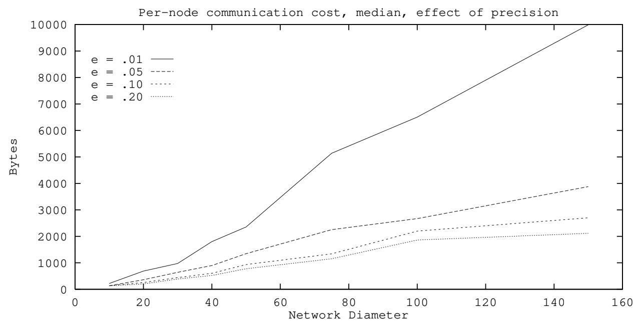

Figure 9: Total communication cost per-node, median, effect of precision.

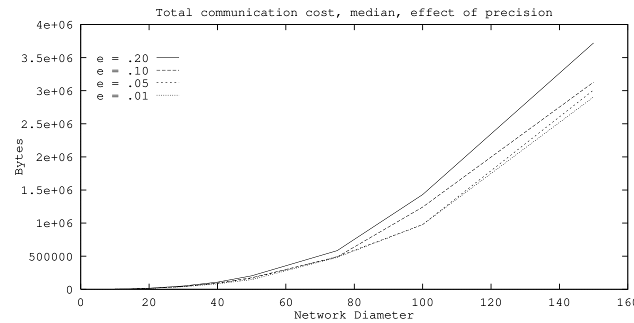

Figure 10: Total communication cost total, median, effect of precision.

passes is the dominant component in the aggregate communication cost over the whole network. In this case, *increasing* precision *reduces* the number of passes, and therefore reduces cost. This observation holds within the precision range of .01 to .20, because increasing precision from .05 to .01 reduces the mean number of required passes from 3.2 to 2.2. However, finer precision never[4] reduces the number of passes below 2, so at some point increasing the precision no longer reduces the number of passes significantly enough to offset the increased overhead of the first passes.

Given that we are generally more concerned with bounding per-node costs, it is likely that we will always prefer to choose lower precision intermediate summaries. Nevertheless, it is instructive to understand the interaction between specifying precision and the resulting number of passes. In practice, many optimizations, for example the ability to suppress values from nodes, and/or power-aware routing within the network, may make this interaction more relevant.

[4]Unless we choose ϵ such that $1/\epsilon > n$.