Holistic Aggregates in a Networked World: Distributed Tracking of Approximate Quantiles

Graham Cormode
Bell Laboratories
cormode@bell-labs.com

Minos Garofalakis
Bell Laboratories
minos@bell-labs.com

S. Muthukrishnan*
Rutgers University
muthu@cs.rutgers.edu

Rajeev Rastogi
Bell Laboratories
rastogi@bell-labs.com

ABSTRACT

While traditional database systems optimize for performance on one-shot queries, emerging large-scale monitoring applications require continuous tracking of complex aggregates and data-distribution summaries over collections of physically-distributed streams. Thus, effective solutions have to be simultaneously space efficient (at each remote site), communication efficient (across the underlying communication network), and provide continuous, guaranteed-quality estimates. In this paper, we propose novel algorithmic solutions for the problem of continuously tracking complex holistic aggregates in such a distributed-streams setting — our primary focus is on approximate quantile summaries, but our approach is more broadly applicable and can handle other holistic-aggregate functions (e.g., "heavy-hitters" queries). We present the first known distributed-tracking schemes for maintaining accurate quantile estimates with provable approximation guarantees, while simultaneously optimizing the storage space at each remote site as well as the communication cost across the network. In a nutshell, our algorithms employ a combination of local tracking at remote sites and simple prediction models for local site behavior in order to produce highly communication- and space-efficient solutions. We perform extensive experiments with real and synthetic data to explore the various tradeoffs and understand the role of prediction models in our schemes. The results clearly validate our approach, revealing significant savings over naive solutions as well as our analytical worst-case guarantees.

1. INTRODUCTION

Traditional data-management applications such as managing sales records, transactions, inventory, or facilities typically require database support for a variety of *one-shot queries*, including lookups, sophisticated slice-and-dice operations, data mining tasks, and so on. One-shot means the data processing is essentially done once, in response to the posed query. This has led to an enormously successful industry of database engines optimized for supporting complex, one-shot SQL queries over large amounts of data.

Recent years, however, have witnessed the emergence of a new class of *large-scale event monitoring* applications that pose novel data-management challenges. In one class of applications, monitoring a large-scale system is an operational aspect of maintaining and running the system. As an example, consider the Network Operations Center (NOC) for the IP-backbone network of a large ISP (such as Sprint or AT&T). Such NOCs are typically impressive computing facilities, monitoring hundreds of routers, thousands of links and interfaces, and blisteringly-fast sets of events at different layers of the network infrastructure (ranging from fiber-cable utilizations to packet forwarding at routers, to VPNs and higher-level transport constructs). The NOC has to continuously track patterns of usage levels in order to detect and react to hot spots and floods, failures of links or protocols, intrusions, and attacks. A similar example is that of data centers and web-content companies (such as Akamai) that have to monitor access to the thousands of web-caching nodes and do sophisticated load balancing, not only for better performance but also to protect against failures. Similar issues arise for utility companies such as electricity suppliers that need to monitor the power grid and customer usage. A different class of applications is one in which monitoring is the goal in itself. For instance, consider a wireless network of seismic, acoustic, and physiological sensors that are deployed for habitat, environmental, and health monitoring. Here, the sensor systems monitor the distribution of measurements for trend analysis, detecting moving objects, intrusions, or other adverse events. Similar issues arise in sophisticated satellite-based systems that do atmospheric monitoring for weather patterns.

Examining these monitoring applications in detail allows us to abstract a number of common elements. Primarily, monitoring is *continuous*, that is, we need real-time tracking of measurements or events, not merely one-shot responses to sporadically posed queries. Second, monitoring is inherently *distributed*, that is, the underlying infrastructure comprises several remote sites (each with its own local data source) that can exchange information through a communication network. This also means that there typically are important *communication constraints* owing to network-capacity restrictions (e.g., in IP-network monitoring, where the collected utilization and traffic is voluminous [6]) or power and bandwidth restrictions (e.g., in wireless sensor networks, where communication overhead is the key factor in determining sensor battery life [18]). Furthermore, each remote site may see a *high-speed stream* of data and has its own local resource constraints, such as *storage-space* or *CPU-time* constraints. This is true for IP routers that cannot possibly store the log of all observed traffic due to the ultra-fast rates at which packets are forwarded. This is also true for the wireless sensor nodes, even though they may not observe large data volumes, since they typically have very small memory onboard.

* Work done while at AT&T Research.

SIGMOD 2005 June 14-16, 2005, Baltimore, Maryland, USA.

In addition, there are two key aspects of such large-scale monitoring problems. First, one needs a way to effectively monitor the *complete distribution of data* (e.g., IP traffic or sensor measurements) observed over the collection of remote sites. Having an accurate picture of the overall data distribution is crucial in understanding system behavior and characteristics, tracking important trends, and making informed judgments about measurement or utilization patterns. In other words, while hardwired outlier detection methods can be of use for certain applications (e.g., network anomaly detection), monitoring the entire data distribution gives us a much broader and more robust indicator of overall system behavior — such indicators are critical, for instance, in network-provisioning systems that try to provision routing paths with guaranteed Quality-of-Service parameters (e.g., delay or jitter) over an IP network (e.g., for a VoIP application). Second, answers that are precise to the last decimal are typically not needed when tracking statistical properties of large-scale systems; instead, *approximate estimates* (with reasonable guarantees on the approximation error) are often sufficient, since we are typically looking for indicators or patterns, rather than precisely-defined events. Obviously, this can work in our favor, allowing us to effectively tradeoff efficiency with approximation quality. To summarize, our focus is on large-scale monitoring problems that aim to continuously provide accurate summaries of the complete data distribution over a collection of remote data streams. Solutions for such monitoring problems have to work in a distributed setting (i.e., over a communication network), be real-time or *continuous*, and be space and communication efficient; furthermore, approximate, yet accurate, answers suffice.

Prior Work. Given the nature of large-scale monitoring applications, their importance for security as well as daily operations, and their general applicability, it is surprising that very little is known about solutions for many basic distributed-monitoring problems. The bulk of recent work on data-stream processing has focused on developing space-efficient, one-pass algorithms for performing a wide range of *centralized, one-shot computations* on massive data streams; examples include computing quantiles [14], estimating distinct values [11] and set-expression cardinalities [10], counting frequent elements (i.e., "heavy hitters") [5, 21], approximating large Haar-wavelet coefficients [12], and estimating join sizes and stream norms [1, 2]. As already mentioned, all the above methods work in a centralized, one-shot setting and, therefore, do not consider communication-efficiency issues. More recent work has proposed methods that carefully optimize site communication costs for approximating different queries in a distributed setting, including quantiles [15, 23] and heavy hitters [19]; however, the underlying assumption is that the computation is triggered either periodically or in response to a one-shot request. Thus, such techniques are not immediately applicable for a *continuous-monitoring* environment, where the goal is to continuously provide guaranteed-quality estimates over a distributed collection of streams.

It is important to realize that each of the dimensions of our problem (distributed, continuous, and space-constrained) induce specific technical bottlenecks. For instance, even efficient streaming solutions at individual sites can lead to constant updates on the distributed network and become highly communication-inefficient when they are directly used in distributed monitoring. Likewise, morphing one-shot solutions to continuous problems entails propagating each change and recomputing the solutions which is communication inefficient, or involves periodic updates and other heuristics that can no longer provide real-time estimation guarantees.

Closest in spirit to our work are the recent results of Olston et al. [3, 22] and Das et al. [8]. All three efforts consider the trade-off between accuracy and communication for monitoring a limited class of continuous queries (at a coordinator site) over distributed streams (at remote sites). More specifically, Olston et al. [22] consider aggregation queries that compute *simple, non-holistic aggregates* (e.g., `AVERAGE` or `MAX`) of dynamically-changing numeric values spread over multiple sources. Their approach relies on giving each site a tolerance interval such that the cumulative width of per-site intervals is upper-bounded by the application's total error tolerance. Das et al. [8] also employ similar ideas for the distributed monitoring of set-expression cardinalities; since their estimation problem relies on set semantics, they propose a scheme for effectively *charging* local changes against a site's error tolerance. Finally, Olston et al. [3] focus on monitoring the top-k (i.e., k most frequent) values over remote data streams; their techniques ensure the validity of the current top-k set (at the coordinator) by installing appropriate arithmetic constraints at each site. Once again, these earlier papers focus on specific distributed-monitoring queries (namely, simple aggregates, set-expression cardinalities, and top-k), and are not always applicable to more general settings (specifically, for monitoring summaries of the entire data distribution or more complex, *holistic aggregates* over the remote sites).

Our Contributions. In this paper, we address the fundamental problem of continuously tracking approximate, guaranteed-quality summaries of the complete data distribution over distributed data streams, in its full generality. More specifically, we focus on *approximate quantile summaries* of the overall data distribution (i.e., equi-width histograms). Quantiles are a very general form of *holistic aggregates* over the underlying distribution that, in fact, subsumes other useful holistic-aggregate functions, such as heavy hitters. That is, tracking the quantiles immediately allows us to track heavy hitters based on the same information (as discussed later in this paper). Our contributions are as follows:

• **Communication- and Space-Efficient Approximate Quantile Tracking.** We present the first known algorithms for tracking quantiles over a distributed collection of streams to specified accuracy, provably, at all times. In a nutshell, our algorithms achieve communication efficiency by requiring remote sites to exchange only concise, local-summary information over the communication network. Our tracking schemes also exploit the novel idea of simple, per-site *prediction models* for capturing the behavior of individual stream distributions at remote sites. As our analysis and results demonstrate, this intuitive idea is actually quite powerful, and allows our schemes to achieve a natural *stability* property that, essentially, renders communication unnecessary as long as the behavior of local distributions at remote sites remains reasonably stable. Furthermore, our analysis of the worst-case communication costs for simple cases of our distributed-tracking protocols shows that they are, in fact, comparable to that of one-shot approximate quantile computations. Finally, our schemes are also space-efficient, since they can be implemented using only slightly more space than that used by centralized, one-shot quantile estimation methods for data streams. To the best of our knowledge, our work is the first to provide principled models and analyses for the important problem of approximate quantile tracking over distributed streams.

• **Extensions to General Monitoring Hierarchies and other Holistic Aggregates.** Our basic problem formulation is set using a flat, single-coordinator distributed setting (as in [3, 22, 8]). Later in the paper, however, we show that our schemes and results can also be naturally extended to more complex, hierarchical-monitoring architectures, where the communication network is arranged as a *tree-structured* hierarchy of nodes (such as a sensornet *routing tree* [18]). We also demonstrate the broader applicability of the general

framework and tracking strategies proposed in this paper to other *holistic-aggregate* queries, including heavy hitters [5, 21] and large wavelet coefficients [12]. Further applications of our ideas to even more complex distributed queries (e.g., joins) is a challenging direction for future work.

• **Experimental Results Validating our Approach.** We perform a thorough set of experiments from IP-traffic level (SNMP logs from IP networks) to application level (server downloads from World Cup HTTP request) and to synthetic data. The experiments explore the space vs communication tradeoffs as well as the role of prediction models in detail, and discover more efficiencies than our worst case theoretical bounds indicate. They illustrate that even simple prediction models can have significant impact on saving communication: the data sent represents only a tiny fraction of the total number of updates and, as the updates increase, this fraction shrinks further.

Throughout, we have omitted many proofs due to space constraints; the complete details will appear in the full version of this paper.

2. PROBLEM FORMULATION

In this section, we describe the key elements of our distributed stream-processing architecture and formally define the distributed approximate quantile tracking problem addressed in this paper.

System Architecture. We consider a distributed-computing environment, comprising a collection of k *remote sites* and a designated *coordinator site*. Streams of data updates arrive continuously at remote sites, while the coordinator site is responsible for generating approximate answers to (possibly, continuous) user queries over the *union* of all remotely-observed streams. Our distributed stream-processing model is similar to that of Olston et al. [3, 22] and Das et al. [8] where no direct communication between remote sites is allowed; instead, as illustrated in Figure 1, a remote site exchanges messages only with the coordinator, providing it with state information on its (locally-observed) stream. Note that such a hierarchical processing model is, in fact, representative of a large class of applications, including network monitoring where a central Network Operations Center (NOC) is responsible for processing network traffic statistics (e.g., link bandwidth utilization, IP source-destination byte counts) collected at switches, routers, and/or Element Management Systems (EMSs) distributed across the network.

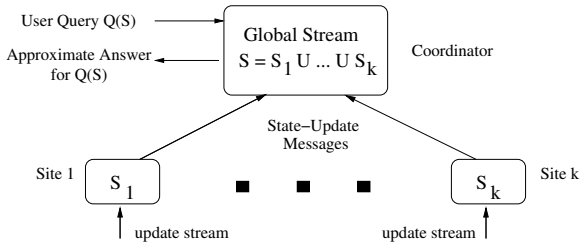

Figure 1: Distributed Stream Processing Architecture.

At each remote site $j \in \{1, \ldots, k\}$, the local update stream renders a multi-set S_j (or, in other words, a *frequency distribution*) over data elements from the integer domain $[U] = \{0, \ldots, U-1\}$. As an example, in the case of IP routers monitoring the number of connections between source and destination IP addresses, $[U]$ is the domain of 64-bit (source, destination) IP-address pairs, and S_j captures the frequency of specific (source, destination) pairs observed at router j. (We use S_j to denote both the update stream at site j as well as the underlying element multi-set/frequency distribution in what follows.) To simplify the exposition, we initially assume that each stream update at remote site j has the form $< +1, v >$, denoting the insertion of data element $v \in [U]$ in the S_j multi-set (i.e., an increase of $+1$ in v's net frequency in S_j); then, in Section 4, we demonstrate that our key ideas and results actually hold for streams S_j of *general updates* (i.e., elements insertions and deletions) of the form $< \pm 1, v >$ at the remote sites. (Note that handling delete operations substantially enriches our distributed streaming model; for instance, it allows us to effectively handle tracking over *sliding windows* of the streams by simply issuing implicit delete operations for expired stream items no longer in the window of interest at remote sites.) We also discuss the extension of our techniques to more complex distributed-tracking architecture, where the underlying communication network is structured as a *multi-level tree hierarchy* (such as the routing trees typically built over sensornet deployments [18]).

The Approximate Quantile Tracking Problem. Our focus in this paper is on the problem of effectively answering user queries on the frequency distribution of the *global collection of streams* $S = \cup_j S_j$ at the coordinator site. Rather than one-time query evaluation, we assume a continuous-querying environment which implies that the coordinator needs to *continuously maintain* (or, *track*) a picture of the global frequency distribution S as the local update streams S_j evolve at individual remote sites. More specifically, our primary focus is on continuously tracking the *quantiles* (i.e., order-statistics) of the global frequency distribution S at the coordinator.

The distributed nature of the local streams S_j comprising the global frequency distribution S makes this a very challenging problem. A naive scheme that accurately tracks the quantiles of S by forcing remote sites to ship every remote stream update to the coordinator is clearly impractical, since it not only imposes an inordinate burden on the underlying communication infrastructure (especially, for high-rate data streams and large numbers of remote sites), but also drastically limits the battery life of power-constrained remote devices (such as wireless sensor nodes) [9, 18]. Instead, to reduce communication overhead, we focus on the continuous tracking of *approximate* quantiles of S at the coordinator site with strong guarantees on the quality of the approximation. This allows our schemes to effectively trade-off communication efficiency and quantile-approximation accuracy in a precise, quantitative manner; in other words, larger error tolerances for the approximate quantiles at the coordinator imply smaller communication overheads to ensure continuous approximate tracking.

More formally, let $N = |S|$ denote the total size of the global data stream S. For a domain value $v \in [U]$, we use $r(v)$ and $q(v)$ to denote the *absolute rank* and *quantile (i.e., relative) rank* of v in S, respectively; in other words, $r(v) = |\{u \in S : u \leq v\}|$ and $q(v) = \frac{r(v)}{N}$. Given a prespecified error tolerance ϵ, our goal is to continuously maintain an *ϵ-approximate quantile summary* $\mathcal{Q}(S)$ of the global frequency distribution S at the coordinator while minimizing the overall amount of communication between the coordinator and the remote sites. By providing a continuous ϵ-approximation guarantee, this quantile summary $\mathcal{Q}(S)$ at the coordinator can, at any time instant, be employed to answer:
(1) *ϵ-approximate quantile-rank queries*, where, given a domain value $v \in [U]$, we seek to find an approximate quantile rank $\hat{q}(v) \in [0, 1]$ that is within ϵ of v's true quantile rank in S, i.e., find $q(v)$ such that $q(v) - \epsilon \leq \hat{q}(v) \leq q(v) + \epsilon$; and,

(2) ϵ-*approximate quantile-value queries*, where, given a quantile rank $q \in [0, 1]$, we seek a value $v = v(q) \in [U]$ whose quantile rank in S is within ϵ of q, i.e., find $v = v(q)$ so $q - \epsilon \leq q(v) \leq q + \epsilon$.
(The corresponding ϵ-approximate queries for *absolute rank* are also naturally defined in a similar manner—in the case of absolute rank, error tolerance for $r(v)$ is defined as $\pm\epsilon N$.) Thus, our notion of approximate quantile summaries for S is identical to that of all earlier research on approximate quantiles [14, 15, 13, 20]. Note that an approximate quantile-value query is essentially the *dual* of an approximate quantile-rank query, and can be easily answered with $O(\log U)$ quantile-rank queries using binary search to determine a value v that generates an approximate quantile rank in the desired range $[q - \epsilon,\ q + \epsilon]$ for the original query q. Symmetrically, one we could answer approximate quantile-rank queries with a bounded number of approximate quantile-value queries. Hence, our discussion focuses on answering quantile-rank queries with ϵ-approximation guarantees, knowing that this is sufficient to also answer quantile-value queries within the same error bounds.

3. OUR QUANTILE TRACKING SOLUTION

In this section, we discuss the details of our proposed scheme for the distributed tracking of approximate quantiles. We begin by providing an overview of our general approach and a discussion of important design desiderata for our distributed-tracking solution.

3.1 Overview

In a nutshell, our distributed quantile-tracking scheme is based on each individual remote site j continuously monitoring the quantiles of its *local* update stream S_j ($j = 1, \ldots, k$). When a certain amount of change is observed locally, then a site may communicate with the coordinator in order to update the coordinator with more recent information about its local update stream and, then, resumes monitoring its local updates. Our overall goal is to ensure strong, ϵ-approximation guarantees for quantile queries over $S = \cup_j S_j$ at the coordinator while minimizing the amount of communication with the remote sites. Besides the obvious goal of *correctness* (i.e., making sure that the coordinator maintains ϵ-approximate quantiles of S at all times), we can identify other important design desiderata that our solution should strive for.

- *Summary-based Information Exchange.* Rather than shipping the complete frequency distribution for their local streams S_j to the coordinator, remote sites only communicate *concise quantile summaries* $\mathcal{Q}(S_j)$ of their locally-observed updates (along with, perhaps, some additional summary information). The size of the $\mathcal{Q}(S_j)$ summary depends critically on the desired ϵ-approximation guarantees at the coordinator.
- *Stability.* Intuitively, the stability property means that, provided the local distributions at remote sites remain approximately the same, there is no need for communication between the remote sites and the coordinator. The interpretation of this property depends on our ability to *model* the similarity of the up-to-date local distributions S_j to their past behavior—as long as our models accurately capture the true behavior of local update streams, no communication between the remote sites and the coordinator is necessary.
- *Minimal Global Information Exchanges.* Even though remote sites communicate only summary information on their local streams, as the number of sites k increases, the communication penalty for interrogating all remote sites becomes inordinately high. Hence, we aim to avoid solutions that may require regular collection or broadcasting of information from/to every remote site in the system. For instance, a scheme that distributes information on the global quantiles over S to all remote sites would typically need to rebroadcast up-to-date global-quantile information to sites (either periodically or during some "global resolution" stage [3]) in order to ensure correctness. Our solutions are designed to explicitly avoid such expensive "global synchronization".

As already mentioned, our solution is based on remote sites continuously monitoring *local constraints* on the quantile distributions of their local update streams S_j, and contacting the coordinator with an appropriate quantile summary $\mathcal{Q}(S_j)$ (and, possibly, additional summary information) once these local quantile constraints are violated. Briefly, our tracking scheme splits the allowed error tolerance ϵ at the coordinator into two distinct components ϕ and θ; that is, $\epsilon = \phi + \theta$, where

- ϕ captures the error of local quantile summaries communicated to the coordinator; and,
- θ captures (an upper bound on) the deviation of local quantiles at each remote site based on locally-observed updates since the last communication with the coordinator.

Thus, in our solution, a local quantile summary $\mathcal{Q}(S_j)$ last communicated to the coordinator at time t carries an approximation error in the order of ϕ *with respect to the snapshot of local stream* S_j *at time* t, whereas θ bounds the deviation of local quantiles with respect to the snapshot-summary information sent to the coordinator (since time t). Intuitively, a larger θ value allows for larger local deviations since the last communication and, therefore, implies fewer communications to the coordinator but, since $\epsilon = \phi + \theta$, for a given error tolerance ϵ, the size of the ϕ-approximate summary $\mathcal{Q}(S_j)$ sent during each communication is larger. Our analysis provides rules for optimally dividing the allowed error tolerance ϵ in simple cases; in more complex scenarios, we give empirical guidelines for allocating ϕ and θ.

Each local quantile summary $\mathcal{Q}(S_j)$ communicated to the coordinator at time t gives a ($O(\phi)$-approximate) picture of the snapshot of the S_j stream at time t.[1] In order to achieve the *stability* property, a crucial component of our solutions is the concept of concise *prediction models* that may be communicated from remote sites to the coordinator (along with the local $\mathcal{Q}(S_j)$ quantile summaries) in an attempt to accurately capture the anticipated behavior of local streams. The idea is that the coordinator employs the prediction model for site j (in conjunction with its most recent $\mathcal{Q}(S_j)$ local snapshot summary) to predict the *current state* of the S_j stream when estimating the global, up-to-date quantiles for S; similarly, remote site j employs the same prediction model to check for the deviation of its local quantiles with respect to the corresponding predictions at the coordinator. Thus, as long the prediction models accurately capture the local update behavior at remote sites, no communication is needed. Note that a prediction model is *local* information for a specific remote site, and it can be computed either by the remote site itself or by the coordinator (and, of course, transmitted to the other party for the purposes of quantile computation or local monitoring). Since our prediction models are also part of the information exchanged between the remote sites and the coordinator, it is crucial to keep them simple and concise—we discuss several different options for such prediction models, ranging from naive "empty" models to more natural models based on locally-observed update rates.

[1] To simplify the exposition, we assume that communications with the coordinator are instantaneous. In the case of non-trivial delays in the underlying communication network, techniques based on timestamping and message serialization can be employed to ensure correctness, as in [22].

To simplify the exposition, we initially focus solely on minimizing the overall communication cost, and assume that each remote site accurately maintains the *full distribution* for its local update stream S_j (and, thus, can compute its local quantiles exactly). Subsequently, we generalize our approach to additionally bound the space/time requirements at remote sites through the use of *approximate local-quantile tracking techniques*; as we demonstrate, our solution can be extended to accommodate the use of most known streaming quantile summaries at remote sites, including, for example, deterministic Greenwald-Khanna summaries [14] or q-digests [23], as well as randomized subset-sum summaries [13] and Count-Min sketches [4].

3.2 The Basic Tracking Scheme

Fix a remote site j, and let $\mathcal{Q}(S_j)$ denote the collection of ϕ-quantile values of (the current snapshot of) its local update stream S_j; that is, $\mathcal{Q}(S_j)$ comprises $\lceil \frac{1}{\phi} \rceil + 1$ values $v_{0,j}, \ldots, v_{\lceil \frac{1}{\phi} \rceil, j}$ such that the quantile rank of the i^{th} value $v_{i,j}$ in the local stream S_j (denoted by $q_j(v_{i,j})$) is $q_j(v_{i,j}) = i\phi$ (or, equivalently, its absolute rank is $r_j(v_{i,j}) = i\phi N_j$), for $i = 0, 1, \ldots, \lceil \frac{1}{\phi} \rceil$.[2] In particular, note that $v_{0,j}$ and $v_{1/\phi,j}$ are the minimum and maximum values observed in stream S_j (respectively).

It is not difficult to see that the above-described collection of ϕ-quantile values $\mathcal{Q}(S_j) = \{v_{i,j} : i = 0, \ldots, 1/\phi\}$ is a $\frac{\phi}{2}$-approximate quantile summary (of size $O(\frac{1}{\phi} \log M)$) for stream S_j. This fact follows from the simple observation that, given a value $v \in [U]$, we can determine two consecutive values $v_{i,j}$ and $v_{i+1,j}$ in $\mathcal{Q}(S_j)$ such that $v \in (v_{i,j}, v_{i+1,j}]$, and estimate the approximate quantile rank of v in S_j as $\frac{q_j(v_{i,j}) + q_j(v_{i+1,j})}{2}$. Since, $q_j(v) \in [(q_j(v_{i,j}), q_j(v_{i+1,j})]$ and the quantile-rank difference between consecutive values is bounded by ϕ, this estimate is guaranteed to be a $\frac{\phi}{2}$-approximation of v's true quantile rank.

In our basic tracking scheme, remote sites can communicate their ϕ-quantile values summary $\mathcal{Q}(S_j) = \{v_{i,j} : i = 0, \ldots, 1/\phi\}$, along with (possibly) a concise prediction model for their local updates to the coordinator site. Let S_j denote the snapshot of the local stream last communicated (through $\mathcal{Q}(S_j)$) to the coordinator, and let $N_j = |S_j|$; also, let n_j denote the total number of element updates to the S_j multi-set since this last communication. Thus, the size of the up-to-date local stream at site j (denoted by $S_j \cup \Delta S_j$) is $N_j + n_j$, while the size of the up-to-date global stream $S = \cup_j (S_j \cup \Delta S_j)$ is $N = |S| = \sum_j N_j + n_j$. Obviously, when site j communicates with the coordinator, it sets $N_j \leftarrow N_j + n_j$ and resets $n_j \leftarrow 0$.

After shipping $\mathcal{Q}(S_j) = \{v_{i,j} : i = 0, \ldots, 1/\phi\}$ and (possibly) a corresponding prediction model to the coordinator, site j continuously monitors the state of its local quantile values $v_{i,j}$ in its up-to-date local stream. More specifically, for each local quantile value $v_{i,j}$, site j monitors both its *true absolute rank* $r_j(v_{i,j})$ in $S_j \cup \Delta S_j$, as well as its *predicted absolute rank* $r_j^{\mathrm{p}}(v_{i,j})$ based on the prediction model communicated to/from the coordinator. Clearly, the exact methodology for computing the predicted rank $r_j^{\mathrm{p}}(v_{i,j})$ depends on the specific prediction model being used and is discussed in detail in Section 3.3; for now, we just treat it as a value that can be computed by both the remote site and the coordinator. In our solution, a communication with the coordinator is triggered at site j only if, for some $v_{i,j}$, $|r_j^{\mathrm{p}}(v_{i,j}) - r_j(v_{i,j})| > \theta(N_j + n_j)$; that is, the predicted and true rank of the monitored quantile value in $S_j \cup \Delta S_j$ deviate by more than $\theta(N_j + n_j)$.[3]

[2] To simplify the notation, we typically omit the integral ceiling operators in what follows.

[3] In general, if the same value occurs many times in the stream, its absolute rank is actually a range, and we measure the deviation away from this range; for simplicity of presentation, we gloss over this detail, but our results still hold when it is taken into account.

Procedure SiteUpdate(j, ϕ, θ, v)
Input: Site index j; local summary-error and rank-deviation parameters ϕ, θ; inserted value $v \in [U]$.
1. Set $S_j := S_j \cup \{v\}$, $n_j := n_j + 1$, goodPredictions := **true**
2. **for** $i := 0$ **to** $1/\phi$ **do**
3. **if** ($v < v_{i,j}$) **then**
4. $r_j(v_{i,j}) := r_j(v_{i,j}) + 1$
5. **if** ($|r_j^{\mathrm{p}}(v_{i,j}) - r_j(v_{i,j})| > \theta(N_j + n_j)$) **then**
6. goodPredictions := **false**
7. **if** (**not** goodPredictions) **then**
8. Set $N_j := N_j + n_j$, $n_j := 0$
9. Compute new local ϕ-quantile value summary $\mathcal{Q}(S_j)$
10. Send $\{j, \mathcal{Q}(S_j), N_j,$ predictionModel(j) $\}$ to coordinator

Procedure AbsoluteRankQuery(v)
Input: Query value $v \in [U]$.
Output: ϵ-approximate absolute rank of v in the global update stream.
1. rank := 0
2. **for** $j := 1$ **to** k **do**
3. $i' := 0$
4. **for** $i := 0$ **to** $1/\phi$ **do**
5. **if** ($v_{i,j} < v$) **then** $i' := i$
6. rank := rank + $(r_j^{\mathrm{p}}(v_{i',j}) + r_j^{\mathrm{p}}(v_{i'+1,j}))/2$
7. **return**(rank)

Figure 2: Procedures for (a) Quantile Maintenance at Remote Sites, and (b) Approximate Rank-Query Answering at the Coordinator.

As our analysis demonstrates, this condition is sufficient to provide strong ϵ-approximation guarantees for rank and quantile estimates based on the quantile summaries $\mathcal{Q}(S_j)$ last communicated to the coordinator. The pseudo-code for processing streaming updates and monitoring local quantile shifts at site j is depicted at the top of Figure 2.

Estimating Absolute- and Quantile-Rank Queries at the Coordinator. Let $\mathcal{Q}(S) = \cup_j \mathcal{Q}(S_j) = \cup_j \{v_{i,j} : i = 0, \ldots, 1/\phi\}$, i.e., the collection of the most recent quantile summaries received from all remote sites at the coordinator — the global quantile summary used for approximate query answering at the coordinator is essentially a combination of $\mathcal{Q}(S)$ and the per-site prediction models (used in conjunction with $\mathcal{Q}(S_j)$'s to compute the predicted ranks of quantile values in $\mathcal{Q}(S)$). More specifically, define $\hat{N} = \sum_j r_j^{\mathrm{p}}(v_{1/\phi,j})$, that is, the sum of predicted maximum-element ranks across all S_j streams. Given a query value $v \in [U]$, the coordinator determines, for each site summary $\mathcal{Q}(S_j)$, the index $i' = \mathrm{argmax}_i \{v_{i,j} \in \mathcal{Q}(S_j) : v_{i,j} < v\}$, and defines the *bounding quantile value* for v as $v_{i',j}$. It then estimates the absolute and quantile rank of v using the formulas:

$$\hat{r}(v) = \sum_j \frac{r_j^{\mathrm{p}}(v_{i',j}) + r_j^{\mathrm{p}}(v_{i'+1,j})}{2} \quad \text{and} \quad \hat{q}(v) = \frac{\hat{r}(v)}{\hat{N}}.$$

A pseudo-code description of our approximate rank estimation procedure at the coordinator site is shown at the bottom of Figure 2. The following theorem demonstrates that as long as the monitored local rank deviations at remote sites remain bounded, and our predicted ranks satisfy a natural condition (that, as we show in Section 3.3, is satisfied by all our proposed prediction models), then we can provide strong approximation guarantees for the absolute-rank estimate at the coordinator.

THEOREM 3.1. *Assume that, for each remote site j and local quantile value $v_{i,j} \in \mathcal{Q}(S_j)$, we have (1)* $|r_j^{\mathrm{p}}(v_{i,j}) - r_j(v_{i,j})| \leq$

$\theta(N_j + n_j)$, i.e., the rank of $v_{i,j}$ in $S_j \cup \Delta S_j$ is within $\theta(N_j + n_j)$ of its prediction; and, (2) $(r_j^p(v_{i+1,j}) - r_j^p(v_{i,j})) \leq 2\phi(N_j + n_j)$, i.e., the range between consecutive predictions is upper bounded by $2\phi(N_j + n_j)$. Then, for any value $v \in [U]$, the absolute-rank estimate $\hat{r}(v)$ at the coordinator is a $(\phi + \theta)$-approximation to v's true absolute rank $r(v)$ in S; that is,

$$r(v) - (\phi + \theta)N \;\leq\; \hat{r}(v) \;\leq\; r(v) + (\phi + \theta)N.$$

Proof: Note that the absolute ranks for v across different remote sites are clearly additive (i.e., the overall rank is the summation of the per-site ranks) and, further, by the definition of the bounding quantile values, we have $r_j(v) \in (r_j(v_{i',j}), r_j(v_{i'+1,j})]$; thus, we have:

$$\begin{aligned} r(v) = & \sum_j r_j(v) \geq \sum_j r_j(v_{i',j}) \\ \geq & \sum_j [r^{\mathrm{p}}(v_{i',j}) + (r_j(v_{i',j}) - r^{\mathrm{p}}(v_{i',j}))] \\ = & \sum_j \frac{r^{\mathrm{p}}(v_{i',j}) + r^{\mathrm{p}}(v_{i'+1,j})}{2} - \sum_j \frac{r^{\mathrm{p}}(v_{i'+1,j}) - r^{\mathrm{p}}(v_{i',j})}{2} \\ & + \sum_j (r_j(v_{i',j}) - r^{\mathrm{p}}(v_{i',j})). \end{aligned}$$

Based on the definition of our $\hat{r}(v)$ estimate and the assumptions of the theorem, this last inequality gives:

$$\begin{aligned} r(v) \geq & \hat{r}(v) - \sum_j \phi(N_j + n_j) + \sum_j (r^{\mathrm{p}}(v_{i',j}) - r_j(v_{i',j})) \\ \geq & \hat{r}(v) - \phi N - \sum_j |r_j(v_{i',j}) - r^{\mathrm{p}}(v_{i',j})| \\ \geq & \hat{r}(v) - \phi N - \sum_j \theta(N_j + n_j) \\ \geq & \hat{r}(v) - (\phi + \theta)N. \end{aligned}$$

The other direction is symmetric. ∎

We should note here that, in principle, it is possible to allocate a distinct total-error threshold $\epsilon_j = \phi_j + \theta_j$ to each remote site $j = 1, \ldots, k$. However, as can be seen from the proof of Theorem 3.1, the error guarantee ϵ at the coordinator is determined by the *maximum* per-site error, i.e., $\epsilon = \max_j\{\epsilon_j\}$. Thus, since the per-site communication cost is clearly monotonically decreasing in its error tolerance, our algorithms allocate the same error $\epsilon = \phi + \theta$ across all remote sites. [4]

Our scheme also provides similar approximation guarantees for the quantile-rank estimates $\hat{q}(v)$ at the coordinator, as demonstrated in the following corollary. The result employs a slightly tighter upper bound of $\phi(1+\theta)(N_j + n_j)$ on the range between consecutive rank predictions that is, in fact, satisfied by all our proposed prediction models (Section 3.3); it also uses the natural assumptions $\theta \leq 2\phi$ (i.e., no tracked quantile value $v_{i,j}$ can shift more than two quantile widths) and $\epsilon \leq \frac{3}{10}$ (typically, we would want to track errors much smaller than $0.3N$). The proof follows along similar lines as that of Theorem 3.1, and the observation that, based on our stipulated local-quantile deviation bounds, $\hat{N} \in (1 \pm \theta)N$.

COROLLARY 3.2. *Let $\theta \leq \phi$ and $\epsilon = \phi + \theta \leq \frac{3}{10}$. Also, assume that, for each remote site j and local quantile value $v_{i,j} \in \mathcal{Q}(S_j)$, we have (1) $|r_j^p(v_{i,j}) - r_j(v_{i,j})| \leq \theta(N_j + n_j)$, and (2) $(r_j^p(v_{i+1,j}) - r_j^p(v_{i,j})) \leq \phi(1+\theta)(N_j + n_j)$. Then, our quantile-rank estimates $\hat{q}(v)$ at the coordinator are a 2ϵ-approximation to v's true quantile rank $q(v)$ in S.*

Time and Space Complexity. In order to track the quantile values at each remote site, we can maintain an array with the current rank of each of the $v_{i,j}$ values, and update these with each update, as shown in Figure 2. Thus, the time and space complexity for this tracking procedure is only $O(1/\phi)$ per update, in addition to the cost of maintaining sufficient information (i.e., full or approximate local-distribution information) for the quantiles to be recalculated if the distribution changes. When communication is required, the site must recompute a set of quantiles, send these to the coordinator, and compute their ranks. The cost of this will depend on the method used to maintain quantiles, but is at least $\Omega(\frac{1}{\phi})$ per communication.

3.3 Prediction Models

We now describe possible choices for the prediction models employed to describe local update behaviors at the remote sites. Such models are part of the information exchanged between the remote sites and the coordinator so that both parties are "in-sync" with respect to predicted-rank computations; therefore, it is critical to keep prediction models concise and, yet, powerful enough to effectively capture *stability* properties in our distributed-tracking environment.[5]

Zero-Information Model. The simplest prediction model we consider is the "empty" model, namely both the site and the coordinator assume that there are no further local updates and, thus, the absolute ranks for the quantile values $v_{i,j}$ in the $\mathcal{Q}(S_j)$ snapshot summaries last communicated to the coordinator remains exactly the same. (Under such "empty" predictions, no additional information needs to be exchanged between remote sites and the coordinator.) More formally, assuming empty predictions, we have $r_j^{\mathrm{p}}(v_{i,j}) = i\phi N_j$, for each $i = 0, \ldots, 1/\phi$ and $j = 1, \ldots, k$.

Synchronous-Updates Model. A key drawback for the zero-information scheme is that it only achieves stability (i.e., a state of no communication between remote sites and coordinator) in an unrealistic scenario, namely when there is no update activity at remote sites. Intuitively, however, we should be able to achieve stability in more general situations. In particular, assume a scenario where remote sites read in new updates periodically (i.e., one update at each time step) and, furthermore, the quantile distribution observed at each remote site remains relatively stable; in such scenarios, there should be no need for communication to the coordinator, since the global quantile distribution remains about the same.

Our next prediction model works on the assumption of *synchronous updates* for remote sites; that is, at each time step, every remote site observes an update to its local distribution (e.g., by periodically polling a sensor node or a network switch every second). Furthermore, our synchronous-updates model assumes that the *quantile (i.e., relative)* ranks of the values $v_{i,j}$ in the $\mathcal{Q}(S_j)$ snapshot summaries last sent to the coordinator remain the same. In other words, letting t_j denote the number of time steps since the $\mathcal{Q}(S_j)$ summary was communicated to the coordinator, the predicted ranks for site j are defined as $r_j^{\mathrm{p}}(v_{i,j}) = i\phi(N_j + t_j)$, for each $i = 0, \ldots, 1/\phi$. Of course, since both the coordinator and all remote sites work under the same synchronicity assumption, no detailed prediction-model information needs to be exchanged in the synchronous-updates case.

Note that this scheme implicitly requires some notion of "global time", since the coordinator must be able to compute the predicted ranks $r_j^{\mathrm{p}}(v_{i,j})$ in order to answer quantile queries. This depends on t_j, the number of time steps since the last communication from site j, and so this must be something that the coordinator can compute without contacting the sites. In a periodic polling scenario

[4] It is possible to "break" this global error threshold ϵ into different (ϕ_j, θ_j) components for different sites, and even dynamically negotiate the per-site error distribution between ϕ_j and θ_j based on observed site behavior. To keep the analysis tractable and focus on the key features of our scheme, we assume the same (ϕ, θ) error components across all sites.

[5] Note that this is distinct from the notion and usage of the models used in [9]: there, models are used in a sensor network to optimize the cost of evaluating one-shot queries by polling certain sensors.

(i.e., reading a value every second or minute) this is straightforward. In such a setting, where the (different) distributions observed by remote sites remain reasonably stable (i.e., local site variations stay within the θ bounds), stability is achieved, and no communication is required. This may not be the case if the updates are *not* synchronous. Consider the situation when there are two sites, one of which sees a uniform distribution on values $[0, \ldots, 6]$, and the other sees a uniform distribution on values $[6, \ldots, 12]$. If updates arrive at equal rates, then we correctly compute the median as 6. However, suppose that for every update to the first distribution, we see three updates to the second distribution. Now the median is 8. Hence, in order to better predict the behavior we must incorporate the *local rate of updates* at each remote site into our model.

Update-Rates Model. Our third model explicitly brings in the notion of update rates for different local streams. We once again assume a notion of global time, and we assume that updates are observed at each site j at a uniform (local) rate, denoted by δ_j. This rate δ_j completely specifies the prediction model for site j and is exchanged between the coordinator and the site when communication takes place. The specific method of estimating the (current) update rate δ_j (at either the remote site or the coordinator) has no effect on the correctness of our tracking but, obviously, good δ_j estimates are important for reducing communication costs. For instance, δ_j can be defined either as a historic average over the entire history of updates at site j, or, more naturally, as an average update rate over a recent window of observed update behavior at the site; of course, other options (e.g., using a time-of-day-based value from recent-history table) are also possible. As in the synchronous-updates case, our update-rates model also assumes that the *quantile (i.e., relative)* ranks in the latest snapshot summaries $\mathcal{Q}(S_j)$ remain the same; that is, letting t_j denote the number of time steps since the last communication between the coordinator and site j, the predicted ranks $r_j^{\mathrm{p}}(v_{i,j})$ are defined as $r_j^{\mathrm{p}}(v_{i,j}) = i\phi(N_j + \delta_j t_j)$, for each $i = 0, \ldots, 1/\phi$. It is not difficult to see that, if the distributions (i.e., local quantile values) at each site remain (approximately) the same, and the local-update rates are reasonably stable, then, by Theorem 3.1 and Lemma 3.3 (shown later in this section), our update-rates model achieves stability making site-coordinator exchanges unnecessary in the long run.

The following table summarizes the key points for each of our three prediction models (namely, the model information exchanged between the sites and the coordinator, and the assumptions under which they achieve stability).

Model	Model Info.	Model Assumptions
Zero-Information	$\emptyset$	$r_j^{\mathrm{p}}(v_{i,j}) = i\phi N_j$ (Absolute ranks stable, $n_j = 0$)
Synchronous	$\emptyset$	$r_j^{\mathrm{p}}(v_{i,j}) = i\phi(N_j + t_j)$ (Quantile ranks stable, $n_j = t_j$)
Rate-based	Rate δ_j	$r_j^{\mathrm{p}}(v_{i,j}) = i\phi(N_j + \delta_j t_j)$ (Quantile ranks stable, $n_j = \delta_j t_j$)

Analysis. We now demonstrate that all three of the above-described prediction models satisfy the upper bounds on the range between consecutive predicted ranks assumed in Theorem 3.1 and Corollary 3.2, assuming local rank deviations are bounded. Note that our update-rates model trivially captures both the synchronous-updates case ($\delta_j = 1$ for all j) and the zero-information case ($\delta_j = 0$ for all j); hence, it suffices to demonstrate the result for the rate-based case.

LEMMA 3.3. *Under all three prediction models, maintaining the local conditions* $|r_j^{\mathrm{p}}(v_{i,j}) - r_j(v_{i,j})| \leq \theta(N_j + n_j)$ *implies that the range of predicted ranks* $(r_j^{\mathrm{p}}(v_{i+1,j}) - r_j^{\mathrm{p}}(v_{i,j}))$ *is upper-bounded by* $\phi(1+\theta)(N_j + n_j)$ *at all times.*

Proof: It suffices to show the bound for the case of general update rates δ_j, as mentioned above. Consider the case of the maximum element $v_{1/\phi,j}$ at site j. In particular, note that, if we force $v_{1/\phi,j} = \infty$ (i.e., we always track the rank of the maximum element seen thus far), then $r_j(v_{1/\phi,j}) = N_j + n_j$. Since, by our update-rates model predictions, $r_j^{\mathrm{p}}(v_{1/\phi,j}) = N_j + \delta_j t_j$, our maintained condition on local rank deviations means $|(N_j + \delta_j t_j) - (N_j + n_j)| \leq \theta(N_j + n_j)$, which gives $(N_j + \delta_j t_j) \leq (1+\theta)(N_j + n_j)$. Hence we can upper-bound the difference in our rate-based rank predictions:

$$\begin{aligned} r_j^{\mathrm{p}}(v_{i+1,j}) - r_j^{\mathrm{p}}(v_{i,j}) &= (i+1)\phi(N_j + \delta_j t_j) - i\phi(N_j + \delta_j t_j) \\ &= \phi(N_j + \delta_j t_j) \leq \phi(1+\theta)(N_j + n_j). \end{aligned}$$

∎

Thus, by Theorem 3.1, all three of our prediction models can guarantee $(\phi+\theta)$-approximate absolute-rank estimates $\hat{r}(v)$ at the coordinator, as long as the local rank $r_j(v_{i,j})$ of $v_{i,j}$ in the up-to-date local stream $S_j \cup \Delta S_j$ does not deviate from its prediction $r_j^{\mathrm{p}}(v_{i,j})$ by more than $\theta(N_j + n_j)$, for each i, j. We now proceed to analyze the communication cost associated with our tracking schemes.

LEMMA 3.4. *Assume the empty model, and let* $\epsilon = \phi + \theta$ *denote the error tolerance at the coordinator. Then, for appropriate settings of parameters* ϕ *and* θ *(specifically,* $\phi = \theta = \epsilon/2$*), the worst-case total communication cost is* $O(\frac{1}{\epsilon^2}\sum_j \ln N_j)$.

Proof: Observe that the largest value tracked is the one whose rank changes the most due to the n_j local updates. If all n_j updates at the site happen below the maximum quantile value $v_{1/\phi,j}$ its absolute rank increases by n_j (this is achieved if we force $v_{1/\phi,j} = \infty$ as in the proof of Lemma 3.3), Thus, to ensure $(\phi+\theta)$ absolute-rank estimates in the zero-information model, remote site j is forced to communicate its current local summary to the coordinator if $n_j > \theta(N_j + n_j)$, or, equivalently, $n_j > \frac{\theta}{1-\theta}N_j$; in other words, site j needs to send its summary to the coordinator *once the number of updates* n_j *exceeds a certain fraction of its earlier snapshot size* N_j. The total number of communications required to reach a local count of N_j is m, where $(1+\frac{\theta}{1-\theta})^m = (1-\theta)^{-m} \geq N_j$ and hence $m = O(\frac{1}{\theta}\ln N_j)$. Since each communication is of size $O(\frac{1}{\phi})$, the total communication cost is $O(\frac{1}{\phi\theta}\ln N_j)$ per site. To optimize this cost, note that minimizing $1/(\phi\theta)$ is equivalent to maximizing $\phi\theta$, and since $\epsilon = \phi + \theta$ this leads us to choose $\phi = \theta = \epsilon/2$. Hence, using these optimal settings for ϕ and θ, the worst-case total communication cost is $O(\frac{1}{\epsilon^2}\sum_j \ln N_j)$. ∎

There is a couple of interesting things to note here. First, according to the analysis in Lemma 3.4, in order to minimize the worst-case communication cost in a zero-information model for a given error tolerance ϵ, the optimal settings for the ϕ and θ parameters are $\phi = \theta = \epsilon/2$. In other words, we should equally distribute the specified error tolerance between the quantile-summary error and the local-deviation error. Second, the worst-case communication-cost analysis in Lemma 3.4 can actually be shown to hold even in the case of our synchronous and rate-based models, assuming that the underlying update-rate assumptions (i.e., $n_j = t_j$ or $n_j = \delta_j t_j$) are correct. Thus, dividing the specified error tolerance equally between ϕ and θ provides a reasonable heuristic even for our more sophisticated prediction models; in fact, our experimental results in Section 5 clearly validate this point.

Other Prediction Models. One could argue that even the rate-

based model of site variation (based on following the update rates of at each remote site) is quite simplistic. It is certainly possible to expand our prediction-model repertoire, for instance, by examining *second-order effects* and modeling the rate-of-change of local update rates (i.e., the "acceleration" as opposed to the "speed" of updates). The parameters of such prediction models can be learned locally (at the sites or the coordinator) using a variety of methods, including, for example, averaging over sliding windows, recent-history tables, or even more sophisticated machine-learning techniques (e.g., linear or higher-order regression). Thus, we can have an entire family of sophisticated prediction models for remote-site behavior — these models can all be easily incorporated into our framework by creating new rank predictions $r_j^p(v_{i,j})$ based on measurable parameters that can be shared between the remote site and the coordinator. Provided that our prediction models ensure that the predicted-rank ranges $(r_j^p(v_{i+1,j}) - r_j^p(v_{i,j}))$ remain bounded by $\phi(1+\theta)(N_j + n_j)$ (either by demonstrating that the condition always holds or by explicitly checking the condition at the site and forcing communication when it is violated), then we can once again apply Theorem 3.1 and Corollary 3.2 to guarantee the quality of the approximate answers given by the coordinator. To keep the exposition in this paper concise and simple, we do not consider any further prediction models beyond the three key models outlined in this section.

3.4 Using Approximate Local Summaries

Thus far, our discussion has assumed that each remote site maintains an accurate picture of the *full distribution* of its local update stream S_j, and uses that distribution to compute the exact quantile values for its local ϕ-quantile value summary $\mathcal{Q}(S_j)$. In several application scenarios, however, maintaining the complete local distribution is not feasible; for instance, when dealing with massive, rapid-rate streams, or remote sites with severe resource limitations (such as tiny sensor motes [18]). In these settings, remote sites can make use of one of the several recently-proposed space/time-efficient methods for tracking their local quantiles *approximately* over their streams. Such approximate methods are either deterministic (based on retaining carefully-chosen subsets of the full stream distribution) [14, 23], or based on randomized-sketching techniques [13, 4]. Furthermore, given a stream S_j (of size N_j) over $[U]$ and an error bound α, these methods are able to track quantile information and report any requested quantile with an error bounded by αN_j, while using space/time that is only logarithmic in N_j, U.

Note that, in our proposed distributed-tracking schemes, a remote site j needs to recompute its ϕ-quantile value summary $\mathcal{Q}(S_j)$ whenever one of its existing local quantiles breaks its predicted rank bound (Figure 2). If we make use of an approximate quantile-tracking algorithm with error α at the remote sites then, when computing the $v_{i,j}$ values in $\mathcal{Q}(S_j)$, we can no longer ensure that $r_j(v_{i,j}) = i\phi N_j$; instead, we can only provide the weaker guarantee that $r_j(v_{i,j}) \in [i\phi N_j \pm \alpha N_j]$. Examining Theorem 3.1, it is not difficult to see that, if we replace $r_j(v_{i,j})$ with $[r_j(v_{i,j}) \pm \alpha N_j]$, then we can once again bound the approximation error at the coordinator as shown in the following corollary.

COROLLARY 3.5. *Assume that, for each local quantile value $v_{i,j} \in \mathcal{Q}(S_j)$, site j can compute only an approximate rank estimate $\hat{r_j}(v_{i,j}) \in [r_j(v_{i,j}) \pm \alpha N_j]$, and, furthermore, (1) $|r_j^p(v_{i,j}) - \hat{r_j}(v_{i,j})| \leq \theta(N_j + n_j)$, and (2) $(r_j^p(v_{i+1,j}) - r_j^p(v_{i,j})) \leq 2\phi(N_j + n_j)$. Then, for any value $v \in [U]$, the absolute-rank estimate $\hat{r}(v)$ at the coordinator is a $(\phi + \theta + \alpha)$-approximation to v's true absolute rank $r(v)$ in S; that is,*

$$r(v) - (\phi + \theta + \alpha)N \;\leq\; \hat{r}(v) \;\leq\; r(v) + (\phi + \theta + \alpha)N.$$

In other words, tracking local quantiles approximately (to within an α error) at each remote site implies an additional α error for the approximate global-rank estimates provided by our distributed-tracking schemes at the coordinator. Of course, these approximations also imply a drastic reduction in the local space requirements at each remote site. More specifically, in addition to the $O(1/\phi)$ space to store the local quantile values (and associated information) in $\mathcal{Q}(S_j)$, each site only needs to store a concise synopsis data structure for approximating its quantiles. Typical space bounds for such synopses are: (1) $O(\frac{1}{\alpha}\log(\alpha N_j))$ using the Greenwald-Khanna algorithm [14]; (2) $O(\frac{1}{\alpha}\log U)$ using the q-digest [23]; or, (3) $O(\frac{1}{\alpha}\log 1/p)$ (p is the probability of error) using the Count-Min sketch [4]. In practice, the space requirements of the above methods are all approximately $O(\frac{1}{\alpha})$. Given a fixed amount of space s available at each remote site and a desired global error tolerance ϵ, this raises an interesting optimization problem — namely, determine the settings for parameters ϕ, θ, and α that minimize overall communication subject to the constraints: $\phi + \theta + \alpha = \epsilon$ and $s \geq O(\frac{1}{\phi} + \frac{1}{\alpha})$. The following lemma summarizes our analysis of this problem in the simple case of a zero-information model.

LEMMA 3.6. *Assume a zero-information model, and let $\epsilon = \phi + \theta + \alpha$ denote the global error tolerance and s denote the space available at each remote site. Provided that s is at least $\Omega(\frac{1}{\epsilon})$, the worst-case total communication cost is upper bounded by $O(\frac{1}{\epsilon^2}\sum_j \ln N_j)$ for appropriate settings of parameters ϕ, θ, and α (specifically, $\phi = \theta = \frac{\epsilon}{2} - \Theta(\frac{1}{s})$ and $\alpha = \epsilon - \theta - \phi = \Theta(\frac{1}{s})$).*

The interpretation of Lemma 3.6 is that, given s and ϵ, we should try to make the local approximation error α as small as possible (consuming a constant fraction of the available space s), and divide the rest of the available error budget equally between ϕ and θ.

4. EXTENSIONS

Heavy-Hitter Tracking. The heavy hitters are those values that occur more than some fraction f of the time, i.e. their count is at least fN. Note that this is distinct from the top-k items, since although any heavy hitters are in the top-$1/f$ items, the inverse implication is not always true. Observe that our method to continuously track the ranks of items *immediately* allows us to find (approximate) heavy hitters: assume that $\hat{r}(v)$ gives an estimate of the maximum rank of item v, then $\hat{r}(v) - \hat{r}(v-1)$ gives an estimate of the *count* of v, with error at most $2\epsilon N$. Hence, the coordinator continuously maintains $O(\epsilon)$-approximate counts of each item (more generally, the approximate count of any range of contiguous items). From this, count queries can be answered, and with some simple additional data structures, the heavy-hitter item set can be continuously maintained. In the full version of this paper, we give more direct schemes to track heavy hitters, based on the same structure as our solution for quantiles: if every remote site tracks the heavy hitters from its stream, and only communicates to the coordinator when the count of an item differs by more than $\theta(N_j + n_j)$ from its *predicted* count. As before, one can plug in a variety of models to predict the counts of items in order to minimize communication to the coordinator. Recomputing heavy-hitter counts before a communication can be supported by keeping exact counts or using summary methods just as with quantiles. Other holistic summaries, such as the largest wavelet coefficients, may also be computed in this framework, since the computation can be decomposed by summing the components from the remote sites. We postpone the complete details to the full version of this paper.

Handling General Updates. Thus far our discussion has focused on the case of insertion-only streams. This model fits many data

stream scenarios but, in full generality, the observed update streams can consist of both element insertions (arrivals) and deletions (departures). For instance, in a network-monitoring application, sites tracking the distribution of open TCP connections must support deletions in order to purge inactive connections from their local S_j streams. We argue that our scheme naturally adapts to such general update streams, provided that the remote sites are capable of computing quantiles (either accurately or approximately) over a stream containing both insertions and deletions. (Randomized methods for approximate local quantile tracking [13, 4] are applicable for general updates.) The key observation here is that, in the presence of deletions, we can define n_j as the *net change* in the overall size of the local S_j stream since the last exchange between site j and the coordinator; this implies that n_j can now become negative. However, all of the proofs of our key approximate-tracking results (namely, Theorem 3.1, Lemma 3.3, and Corollaries 3.2 and 3.5) make no assumption on the sign of n_j. Thus, provided that the required local-deviation bounds (with respect to predicted ranks) are met, our scheme still guarantees accurate quantile predictions at the coordinator. Note though that some of our subsequent analyses of the communication cost implicitly assume insert-only streams. So these bounds no longer apply to general updates, and instead we can only provide weaker conservative worst-case bounds. Lastly, we observe that allowing deletions means that we can incorporate a sliding window approach, where only recent stream values are considered to contribute to the current distribution. We will give full description of the necessary changes to accomodate this in the full version of this paper.

Hierarchical Quantile Tracking. Consider a more complex distributed-tracking scenario where the communication network is arranged as a *tree-structured* hierarchy of nodes (i.e., sites) — our goal here is for the root node to effectively track approximate quantiles of the update streams observed at the leaf nodes of the hierarchy. (For simplicity, assume that internal nodes in the hierarchy do not observe any updates; we can add dummy leaf-child nodes to internal nodes to accept their corresponding streams.) This hierarchical-monitoring architecture generalizes the flat, single-level model discussed earlier in this paper (essentially, a one-level hierarchy with remote sites as leaves). Such monitoring hierarchies arise naturally, for instance, in the context of sensor networks, where sensor nodes are typically organized in a routing tree with a root base-station monitoring the sensornet operation [18]. In such scenarios, naively propagating quantile updates from the leaves to the root node is wasteful; quantile shifts at leaf nodes of a subtree may effectively "cancel out" at a higher node in the hierarchy rendering further communication unnecessary. For such multi-level hierarchies, our tracking scheme should be able to exploit stability properties *at any node of the hierarchy*.

Assume that our tracking hierarchy comprises $h+1$ levels, with the root node at level 0 and the leaf nodes at level h. We can compute quantiles over the union of streams observed at the leaf nodes by running our quantile-tracking scheme between each internal node and its children. That is, each internal node tracks the (approximate) quantiles of its children, and then passes up relevant information to its parent when its locally-observed quantiles (which summarize the quantiles of all streams in its subtree) violate a specified deviation bound with respect to their corresponding predicted values. Just as in the flat, single-level case it suffices to allocate the same error tolerance $\epsilon = \phi + \theta$ to every remote site (see Section 3.2), we argue that it suffices to allocate the same $\epsilon_l = \phi_l + \theta_l$ parameters to every node at level l in the hierarchy— essentially, this implies that each level-$(l-1)$ node gives all its children the maximum possible error tolerance (based on its own error bounds) in order to minimize communication cost across levels $l-1$ and l.

Consider a node u at level l in the tree hierarchy and let S_u denote the union of update streams in the subtree rooted at u. Generalizing from our approximate quantile tracking discussion in Section 3.4, define (1) α_l as the accuracy at which u tracks its local quantile values (for the S_u stream); (2) ϕ_l as the granularity of the quantile value summary that node u ships to its parent (i.e., $\mathcal{Q}(S_u)$ comprises the values at approximate quantile ranks $0, \ldots, 1/\phi_l$); and, (3) θ_l as the bound on the deviation of local quantiles (with respect to their predictions) at node u. The following corollary then follows easily from Corollary 3.5.

COROLLARY 4.1. *Let α_l, ϕ_l, and θ_l be as defined above. Then, the accuracy of local quantile estimates for nodes at level $l-1$ of the hierarchy is $\alpha_{l-1} = \alpha_l + \theta_l + \phi_l$.*

Corollary 4.1 essentially allows us to "cascade" our basic, single-level tracking scheme to the case of multi-level hierarchies. Specifically, if we let $\alpha_h = \alpha$ denote the error of the summaries used to track update streams at leaf nodes ($\alpha_h = 0$ if leaf nodes maintain the full stream distribution), then, by Corollary 4.1, summing across all levels, the total quantile-tracking error at the root node is $\epsilon = \alpha_0 = \alpha + \sum_{l=1}^{h}(\theta_l + \phi_l)$.

Fix the summary error α at leaf nodes. We now seek to optimize the settings for the θ_l and ϕ_l parameters for minimizing communication. We consider the worst-case bounds for the zero-information case, and two possible optimization objectives: (1) the *maximum transmission cost* for any node in the hierarchy (or, equivalently, the maximum load on any communication link), and (2) the *aggregate communication cost* over the entire communication hierarchy. Both of the above objectives are important in the sensornet context (e.g., for maximizing the lifetime of a sensor network) as well as more traditional distributed network-monitoring scenarios. Let k_l denote the number of hierarchy nodes at level l of the hierarchy. From our analysis in Section 3.3, the (worst-case) transmission cost for a node at level l is $O(\frac{1}{\phi_l \theta_l} \ln \frac{N}{k_l})$, which, for a given error $\epsilon_l = \phi_l + \theta_l$ at level l, is minimized for $\phi_l = \theta_l = \epsilon_l/2$.

Maximum Transmission Cost Minimization Problem: Determine ϵ_l's that minimize $\max_l\{\frac{4}{\epsilon_l^2} \ln \frac{N}{k_l}\}$ subject to $\sum_l \epsilon_l = \epsilon$.

For this minimization problem the optimal point occurs when the per-node transmission costs at all levels are equal, giving the optimal per-level θ_l and ϕ_l settings

$$\theta_l \;=\; \phi_l \;=\; \frac{\epsilon\sqrt{\ln \frac{N}{k_l}}}{2\sum_j \sqrt{\ln \frac{N}{k_j}}}. \qquad \blacksquare$$

In the case of minimizing total communication the per-node transmission cost at level l is multiplied by a factor of k_l and so we have a sum objective function. This is a more complicated minimization problem, but we show a closed-form solution for the optimal θ_l, ϕ_l settings.

Total Communication Cost Minimization Problem. Determine ϵ_l's that minimize the sum $\sum_l \frac{4k_l}{\epsilon_l^2} \ln \frac{N}{k_l}$ subject to $\sum_l \epsilon_l = \epsilon$.

THEOREM 4.2. *The optimal θ_l and ϕ_l values for minimizing the (worst-case) total communication cost over a multi-level tracking hierarchy are given by*

$$\theta_l \;=\; \phi_l \;=\; \frac{\epsilon(k_l \ln \frac{N}{k_l})^{1/3}}{2\sum_j (k_j \ln \frac{N}{k_j})^{1/3}}.$$

Proof: Let $c_l = 4k_l \ln \frac{N}{k_l}$, for all levels l. Our proof uses *Hölder's inequality* [16], which states that, for any $x_l, y_l \geq 0$, and $p, q > 1$ such that $\frac{1}{p} + \frac{1}{q} = 1$:

$$\left(\sum_l x_l^p\right)^{1/p} \cdot \left(\sum_l y_l^q\right)^{1/q} \geq \sum_l x_l y_l, \quad (1)$$

with equality holding only if $y_l = \lambda \cdot x_l^{p-1}$ for all l.

Substituting $p = 3$, $q = 3/2$, $x_l = (\frac{c_l}{\epsilon_l^2})^{1/3}$, and $y_l = \epsilon_l^{2/3}$ in Hölder's Inequality, we get

$$\left(\sum_l \frac{c_l}{\epsilon_l^2}\right)^{1/3} \cdot \left(\sum_l \epsilon_l\right)^{2/3} \geq \sum_l c_l^{1/3},$$

or, equivalently (since $\sum_l \epsilon_l = \epsilon$), $\sum_l \frac{c_l}{\epsilon_l^2} \geq \frac{1}{\epsilon^2}\left(\sum_l c_l^{1/3}\right)^3$. Note that the left-hand side of this inequality is precisely our optimization objective, whereas the right-hand side is constant. Thus, the optimal (i.e., minimum) value for our objective occurs when equality holds in this instance of Hölder's inequality, or, equivalently, if $\epsilon_l^{2/3} = \lambda(\frac{c_l}{\epsilon_l^2})^{2/3}$, which after some simplification, gives $\epsilon_l = \lambda' c_l^{1/3}$ (where the new proportionality constant is $\lambda' = \lambda^{1/3}$). Coupled with the total error constraint $\sum_l \epsilon_l = \epsilon$, this directly implies that the optimal ϵ_l values are $\epsilon_l = \epsilon c_l^{1/3} / \sum_j c_j^{1/3}$. Since, $\epsilon_l = \phi_l + \theta_l$ and communication at level l is minimized for $\phi_l = \theta_l = \frac{\epsilon_l}{2}$, the result follows. ∎

5. EXPERIMENTAL STUDY

Our experimental study focused on the the main quantile tracking scheme, and the effect of the different prediction models and their parameters. We implemented a simulation of our tracking scheme in C, with experiments carried out on a single machine. We simulated each remote site, and measured the amount of communication in terms of the number of values sent to the coordinator. We compare this number to the cost of sending every update directly to the coordinator to see the savings of our tracking scheme. Overall, we found that our most sophisticated model performed the best over a variety of data sets. The cost of maintaining continuous quantiles is typically only a small percentage of the cost of communicating every update, and this fraction drops as the number of updates increases.

5.1 Data Sets and Methodology

We ran experiments on a variety of real-life and synthetic data sets, as follows:

World Cup HTTP request data, obtained from the Internet Traffic Archive [17]. We took two subsets of the data, 1 day of requests (15 million) and 8 days of requests (51 million requests total). Each request specified a timestamp, the server that handled the request, and the size of the object returned. In our data, there were 26 different servers (each one corresponding to one remote site), and they handled a varying number of requests, from a few thousand to many millions. The object sizes varied from a few bytes to several Megabytes in size, so computing quantiles over the transfer sizes is a non-trivial task.

SNMP network usage data, obtained from the Dartmouth Wireless-Network Trace Archive [7]. It comprises data from a total of 220 remote wireless access points, each of which recorded the total number of bytes received during roughly five-minute intervals over a four-month period. This gave a total of 6 million updates, where each access point handled up to 32,000 updates. We did not attempt to clean the data set, and so there are some missing values, some access points not recording values for long periods, and some disordered timestamps. Hence, we believe this to be a realistic challenge for our algorithms to deal with less than perfect data. We again compute quantiles over the number of bytes, which ranged from zero to tens of Megabytes per interval.

Synthetic data, representing a single site. We further investigate the power of our methods by examining synthetic data sets whose parameters we can control more directly. We simulated a single site, since we know that the communication cost of the methods in our scheme is just the sum of the costs over the different sites. We adopted simple distributions for the updates: the values are drawn from a Zipfian distribution with parameter z, and the delay between subsequent updates (with parameter δ) is computed as $\lceil \delta(1 + N(0,1)/3) \rceil$, where $N(0,1)$ denotes the usual normal distribution (hence, we allow a small probability of a negative delay, corresponding to out-of-order arrival of updates). We modeled the variability of the site with two parameters, p_r and p_d: with probability p_d the distribution changes, and we pick a new value of z uniformly from $[0 \ldots 2]$; with probability p_r the rate changes, and we pick a new value of δ uniformly from $[0 \ldots 100]$. Although this is a simplistic model, it allows us to understand better under what conditions our algorithm achieves stability, when no communication is necessary between remote site and coordinator.

For the rate-based prediction model, there is an issue of how the site should determine the rate when it communicates values to the coordinator. We implemented two methods. The "global historic" average rate is the total number of updates seen so far, N_j, divided by the time elapsed. This requires only a constant amount of space at the site. The "windowed" rate takes a sliding window of the most recent W updates, and computes W divided by the time between the first and last update in the window. This requires $O(W)$ space at the remote site, but no additional space at the coordinator, since the site merely has to inform the coordinator of the rate that it intends to use (i.e., δ_j), when they communicate.

A second issue to consider is that there is a "warm-up cost" for all our methods: when the number of items at a remote site, N_j, is small, then every update can trigger a communication of the current quantiles. So, instead, we can force remote sites to send all updates since the last communication if this is cheaper. This reduces the communication cost, but even without this adjustment we see significant savings.

5.2 Experimental Results

World Cup HTTP request data. Our first set of experiments on the World Cup HTTP requests are shown in Figure 3(a). On this data set, the synchronous model could not operate, since the rates fluctuated over the course of the data and between servers, so we focus on the rate-based model and the zero-information model. We first varied with the window size, W, to give the best results in computing the rate for the rate-based model. We found that the optimal window size was not very sensitive to the ratio of ϕ to θ, and the smallest cost for this data set was found using a window of size $W = 1500$. As W increases, the cost converges on the (higher) cost of the global rate method. This accords with intuition: too small a window, and the rate computation is too sensitive to recent minor fluctuations, but too large a window and rate changes take a long time to impact on the calculated rate. Our next experiment in Figure 3(b) fixes $\epsilon = 2\%$, and varies ϕ and θ so that $\phi + \theta = \epsilon$. We plot the results for the zero-information and rate-based models (with windowed rate computed over the last $W = 1500$ up-

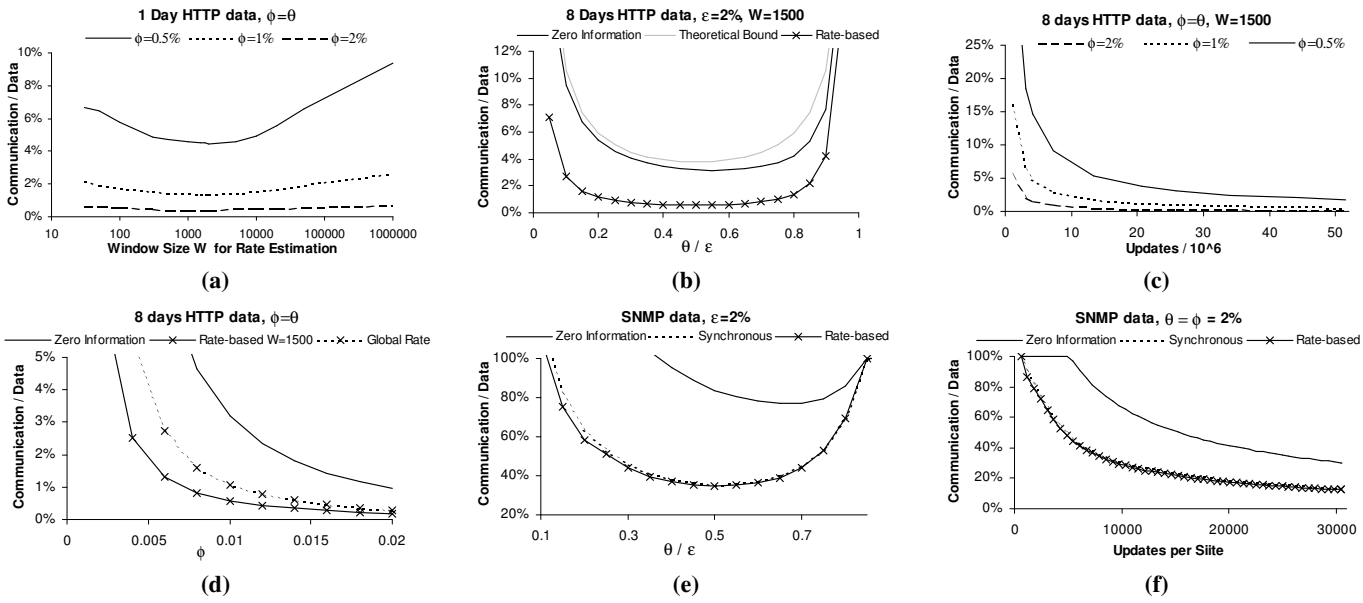

Figure 3: Experiments on real-life data: HTTP request data from World Cup '98: (a) Effect of different window sizes for rate computation; (b) Tradeoff between θ and ϕ; (c) Communication cost as number of updates increases; (d) Communication cost as a function of ϵ. SNMP data from Dartmouth Wireless Network: (e) Tradeoff between θ and ϕ; (f) Communication cost as updates increase.

dates). We observe that setting either ϕ or θ very small drives up the communication cost significantly. The best results are obtained for $\theta = \phi = \epsilon/2$, for both the zero-information model (as predicted by our analysis) and the rate-based model. Here, the communication cost is less than 1% of the cost of sending all the data for processing at the coordinator. We have also plotted a c/ϵ^2 curve to compare the behavior of the zero-information method to its predicted performance. There is some divergence due to our technique of sending updates to the coordinator when this is cheaper than sending quantiles: one can see that this gives an improvement in communication cost that is more pronounced for larger values of θ. Similar results were seen for other settings of ϵ and other parts of the data. Thus, we can conclude that, at least for this data set, choosing $\phi = \theta$ gives the best tradeoff.

For the rest of our results with the World Cup HTTP request data, we focus further on the behavior of the rate-based model. We set $\theta = \phi$ and $W = 1500$, and plot the communication cost as the number of updates increases in Figure 3(c). We see that after a sufficient number of updates, the model has "learned" the distribution and the update rate sufficiently well that very little communication is needed as the number of updates increases—the ratio of total communication to total updates becomes progressively smaller as the system is mostly stable. We see that the cost increases for smaller values of ϵ, in Figure 3(d). Fitting a trend line to the rate-based model, we see (empirically) the cost rising in proportion to $\epsilon^{-1.6}$, asymptotically better than the prediction of ϵ^{-2} for the empty model.

SNMP network usage data. We now consider the SNMP data set from Dartmouth College. This data was collected by periodic polling, so we can apply the synchronous model to this case and compare to the rate-based method which has to discover the rate. This data set turned out to be more challenging for our methods, because the number of updates per site is much smaller than on the World Cup data: at most 32,000 compared to many millions of web requests. Still, the savings over sending all updates to the coordinator are quite significant. This is seen in Figure 3(e), where for $\epsilon = 2\%$, the largest savings from our methods represent nearly two thirds of the cost of sending all updates directly. Our results show that, for this data, using the synchronous method (assuming that there is one update per polling period) does very well, but using the rate-based method with a small sliding window $W = 50$ can do a little better (5% improvement for $\theta = \epsilon/5$, but only 0.5% for $\theta = \phi = \epsilon/2$). This is because there are some fluctuations in the rate, due to polling problems, drops and delays etc., which can force communications in the synchronous method. Using a small sliding window to compute the rate allows these fluctuations to be adjusted for with fewer communications to the coordinator. The cost saving with zero-information is rather small, and skewed by our warm-up heuristic, whereas the best performance for both rate-based and synchronous comes at $\theta = \phi = \epsilon/2$. As the number of updates increases, the ratio of the communication per site to the number of updates seen so far decreases. Figure 3(f) shows that, while the empty model needs an extended warm-up period of approximately 5,000 updates per site, the cost for our methods drops off steadily, ending up close to 20% of the number of updates.

Synthetic data. The synthetic data depends on two parameters, p_r, the probability of the update rate changing, and p_d, the probability of the distribution changing. In our first experiment, we fixed the distribution and varied the rate with probabilities $10^{-3}, 10^{-4}$ and 10^{-5}. [6] We took one million updates to a single site, and plotted the amount of communication as a fraction of the total number of updates (i.e., a million) in Figure 4(a). This lets us see under what conditions stability is achieved for our rate-based model. Note that zero-information cost continues to climb throughout the whole trial, whereas using our method is stable for long stretches.

[6] Although these may seem low, in experiments with higher probabilities (e.g., $10^{-2}, 10^{-1}$), we saw even lower overall cost: when the nature of the stream changes with high frequency (say, once every ten or hundred updates), then this becomes a "mixed" distribution that the model can fit itself to.

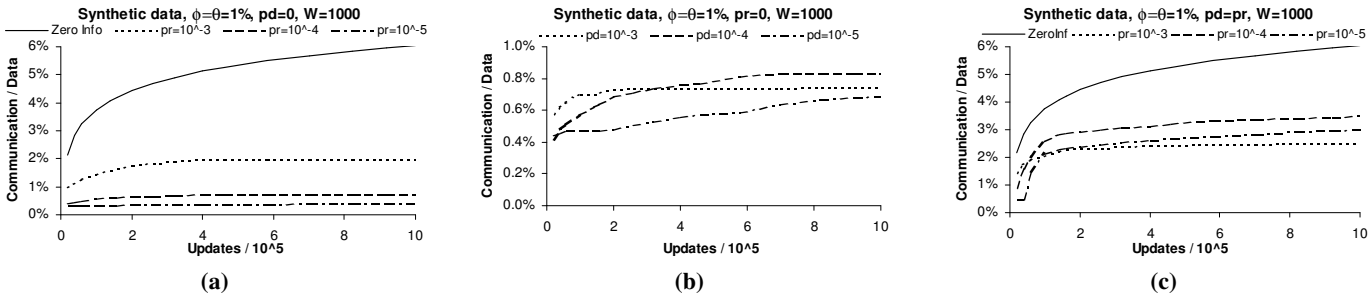

Figure 4: Synthetic data: (a) Fixed distribution, varying rate; (b) Fixed rate, varying distribution; (c) Varying distribution and rate.

The overall cost depends on the frequency of rate changes, since each time a rate change occurs, some communication may be necessary when this causes the predicted ranks to be invalid. We next fixed $p_r = 0$, and varied p_d, as shown in Figure 4(b). When p_d is highest, at 10^{-3}, then stability persists after an initial period of communication. This can be attributed to the fact that the distribution itself becomes stable: with sufficiently frequent changes of parameter, we get a uniform mixture of Zipfians with parameter z ranging from 0 to 2. By contrast, for very low probability of changing distribution, 10^{-5}, we see that stability is reached initially, but then, when the first jump in distribution occurs (in this case, after about 200,000 updates), then a period of communication is entered as the model has to be adjusted for the new distribution. Several subsequent distribution shifts prolong this communication, but as the number of updates increases, the cost levels off. Lastly, we allowed both rate and distributional changes, in Figure 4(c), by setting $p_r = p_d$ for a different probabilities. Here, the overall cost for rate-based methods is higher than before, but still around half as much as for the empty model, and only 3% of the cost of sending every update.

6. CONCLUDING REMARKS

We have presented principled, distributed, continuous tracking algorithms for quantile summaries of data that work under space constraints at local sites processing high-speed streams, work under communication constraints between sites, and maintain provably accurate quantile summaries at all times. Our algorithms employ a combination of local tracking of quantiles and simple prediction models to guarantee communication efficiency, as demonstrated in our experiments with real-life and synthetic data sets. Our approach and results apply immediately to tracking problems for other valuable statistics, such as heavy hitters and heavy wavelet coefficients, since they can be viewed as special cases of quantile tracking. Our results are general and comprehensive for distributed continuous monitoring applications which are of growing interest for large-scale systems.

7. REFERENCES

[1] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. "Tracking Join and Self-Join Sizes in Limited Storage". In *Proceedings of ACM PODS*, 1999.

[2] N. Alon, Y. Matias, and M. Szegedy. "The Space Complexity of Approximating the Frequency Moments". In *Proceedings of ACM STOC*, 1996.

[3] B. Babcock and C. Olston. "Distributed Top-K Monitoring". In *Proceedings of ACM SIGMOD*, 2003.

[4] G. Cormode and S. Muthukrishnan. "An improved data stream summary: The count-min sketch and its applications". In *Proceedings of Latin American Informatics*, 2004.

[5] G. Cormode and S. Muthukrishnan. "What's Hot and What's Not: Tracking Most Frequent Items Dynamically". In *Proceedings of ACM PODS*, 2003.

[6] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. "Gigascope: A Stream Database for Network Applications". In *Proceedings of ACM SIGMOD*, 2003.

[7] Dartmouth wireless traces. (`http://cmc.cs.dartmouth.edu/data/dartmouth.html`).

[8] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. "Distributed Set-Expression Cardinality Estimation". In *Proceedings of VLDB*, 2004.

[9] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. "Model-Driven Data Acquisition in Sensor Networks". In *Proceedings of VLDB*, 2004.

[10] S. Ganguly, M. Garofalakis, and R. Rastogi. "Processing Set Expressions over Continuous Update Streams". In *Proceedings of ACM SIGMOD*, 2003.

[11] P. B. Gibbons. "Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports". In *Proceedings of VLDB*, 2001.

[12] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. "Surfing Wavelets on Streams: One-pass Summaries for Approximate Aggregate Queries". In *Proceedings of VLDB*, 2001.

[13] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. "How to Summarize the Universe: Dynamic Maintenance of Quantiles". In *Proceedings of VLDB*, 2002.

[14] M. B. Greenwald and S. Khanna. "Space-Efficient Online Computation of Quantile Summaries". In *Proceedings of ACM SIGMOD* , 2001.

[15] M. B. Greenwald and S. Khanna. "Power-Conserving Computation of Order-Statistics over Sensor Networks". In *Proceedings of ACM PODS*, 2004.

[16] G.H. Hardy, J.E. Littlewood, and G. Pólya. *"Inequalities"*. Cambridge University Press, 1988. (Second Edition).

[17] Internet traffic archive. (`http://ita.ee.lbl.gov/`).

[18] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and H. Wei. "The Design of an Acquisitional Query Processor for Sensor Networks". In *Proceedings of ACM SIGMOD*, 2003.

[19] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. "Finding (Recently) Frequent Items in Distributed Data Streams". In *Proceedings of IEEE ICDE*, 2005.

[20] G. S. Manku, S. Rajagopalan, and B. G. Lindsey. "Approximate medians and other quantiles in one pass and with limited memory". In *Proceedings of ACM SIGMOD*, 1998.

[21] G. S. Manku and R. Motwani. "Approximate Frequency Counts over Data Streams". In *Proceedings of VLDB*, 2002.

[22] C. Olston, J. Jiang, and J. Widom. "Adaptive Filters for Continuous Queries over Distributed Data Streams". In *Proceedings of ACM SIGMOD*, 2003.

[23] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. "Medians and beyond: New aggregation techniques for sensor networks". In *Proceedings of ACM SenSys*, 2004.