Optimal Tracking of Distributed Heavy Hitters and Quantiles*

Ke Yi†
HKUST
yike@cse.ust.hk

Qin Zhang‡
MADALGO§
Aarhus University
qinzhang@cs.au.dk

Abstract

We consider the the problem of tracking heavy hitters and quantiles in the distributed streaming model. The heavy hitters and quantiles are two important statistics for characterizing a data distribution. Let A be a multiset of elements, drawn from the universe $U = \{1, \ldots, u\}$. For a given $0 \leq \phi \leq 1$, the ϕ-heavy hitters are those elements of A whose frequency in A is at least $\phi|A|$; the ϕ-quantile of A is an element x of U such that at most $\phi|A|$ elements of A are smaller than A and at most $(1-\phi)|A|$ elements of A are greater than x. Suppose the elements of A are received at k remote *sites* over time, and each of the sites has a two-way communication channel to a designated *coordinator*, whose goal is to track the set of ϕ-heavy hitters and the ϕ-quantile of A approximately at all times with minimum communication. We give tracking algorithms with worst-case communication cost $O(k/\epsilon \cdot \log n)$ for both problems, where n is the total number of items in A, and ϵ is the approximation error. This substantially improves upon the previous known algorithms. We also give matching lower bounds on the communication costs for both problems, showing that our algorithms are optimal. We also consider a more general version of the problem where we simultaneously track the ϕ-quantiles for all $0 \leq \phi \leq 1$.

1 Introduction

Data streams have been studied in both the database and theory communities for more than a decade [1, 3]. In this model, data items arrive in an online fashion, and the goal is to maintain some function f over all the items that have already arrived using small space. Of particular interests to the database community are the frequent items (a.k.a. *heavy hitters*) [7, 14, 18, 17] and quantiles [12, 13]. After a long and somehow disorganized line of research, the heavy hitter problem is now completely understood with both space upper and lower bounds determined at $\Theta(1/\epsilon)$, where ϵ is the approximation error (formally defined later); please see the recent paper by Cormode and Hadjieleftheriou [7] for a comprehensive comparison of the existing algorithms for this problem. For maintaining quantiles, the best upper bound is due to a sketch structure by Greenwald and Khanna [13], using space $O(1/\epsilon \cdot \log(\epsilon n))$ where n is the number of items in the stream. It is still an open question whether this bound can be improved.

Recent years have witnessed an increasing popularity of another model more general than the streaming model, where multiple streams are considered. In this model, multiple streams are received at multiple distributed *sites*, and again we would like to continuously track some function f over the union of all the

*A preliminary version of this article was presented at the ACM Symposium on Principles of Database Systems (PODS), 2009.
†Supported in part by a DAG and an RPC grant from HKUST, and a Google Faculty Research Award.
‡Corresponding author. Most of this work was done while Qin Zhang was a Ph.D. student at HKUST.
§Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation.

items that have arrived across all the sites. Here the most important measure of complexity is the total communication cost incurred during the entire tracking period. This model, which is either referred to as the *distributed streaming model* or the *continuous communication model*, is a natural combination of the classical communication model [21] and the data stream model. Recall that the communication model studies the problem of computing some function f over distributed data using minimum communication. The data is predetermined and stored at a number of sites, which communicate with a central coordinator, and the goal is to do a *one-time* computation of the function f. Thus the distributed streaming model is more general as we need to maintain f continuously over time as items arrive in a distributed fashion.

The rising interest on the distributed streaming model is mainly due to its many applications in distributed databases, wireless sensor networks, and network monitoring. As a result, it has attracted a lot of attention lately in the database community, resulting in a flurry of research in this area [4, 5, 6, 10, 11, 15, 16, 19, 20]. Most works in this area are heuristic and empirical in nature, with a few exceptions to be mentioned shortly. For many fundamental problems in this model, our theoretical understandings are still premature. This is to be contrasted with the standard streaming model, where theory and practice nicely blend, and in fact many of the most practically efficient solutions are the direct products of our theoretical findings. In this paper, we take an important step towards an analytical study of the distributed streaming model, by considering the worst-case communication complexity of tracking heavy hitters and quantiles, arguably two of the most fundamental problems on data streams.

The distributed streaming model. We now formally define the distributed streaming model, which is the same as in most works in this area. Let $A = (a_1, \ldots, a_n)$ be a sequence of items, where each item is drawn from the universe $U = \{1, \ldots, u\}$. The sequence A is observed in order by $k \geq 2$ remote *sites* $S_1, \ldots, S_k$ collectively, i.e., item a_i is observed by exactly one of the sites. Let $A(t) = \{a_1, \ldots, a_t\}$ be the multiset of the first t items in A. Then the general goal is to continuously track $f(A(t))$ for some function f at all times t with minimum total communication among the sites. Note that in the classical communication model, the goal is to just compute $f(A(n))$; in the data stream model, the goal is to track $f(A(t))$ for all t but there is only one site ($k = 1$), and we are interested in the space complexity of the tracking algorithm, not communication. Thus, the distributed streaming model is a natural combination of the two, but is also significantly different from either.

We define the manner of communication more precisely as follows. There is a distinguished *coordinator* C, who will maintain (an approximate) $f(A(t))$ at all times. There is a two-way communication channel between the coordinator and each of the k sites, but there is no direct communication between any two sites (but up to a factor of 2, this is not a restriction). When site S_j receives the item a_i, based on its local status, S_j may choose to send a message to C, which in turn may trigger iterative communication with other sites. We assume that communication is instant. When all communication finishes, all the sites who have been involved may have new statuses, getting ready for the next item a_{i+1} to arrive. We will measure the communication cost in terms of words, and assume that each word consists of $\Theta(\log u) = \Theta(\log n)$ bits. Finally we assume that n is sufficiently large (compared with k and $1/\epsilon$); if n is too small, a naive solution that transmits every arrival to the coordinator would be the best.

Heavy hitters and quantiles. By taking different f's, we arrive at different continuous tracking problems. The notion of ϵ-approximation also differs for different functions. We adopt the following agreed definitions in the literature. In the sequel, we abbreviate $A(t)$ as A when there is no confusion.

For any $x \in U$, let $m_x(A)$ be the number of occurrences of x in A. For some user specified $0 \leq \phi \leq 1$, the set of *ϕ-heavy hitters* of A is $\mathcal{H}_\phi(A) = \{x \mid m_x(A) \geq \phi|A|\}$, where $|A|$ denotes the total number of items in A. If an ϵ-approximation is allowed, then the returned set of heavy hitters must contain $\mathcal{H}_\phi(A)$ and

cannot include any x such that $m_x(A) < (\phi-\epsilon)|A|$. If $(\phi-\epsilon)|A| \leq m_x(A) < \phi|A|$, then x may or may not be reported. In the *heavy hitter tracking* problem, the coordinator should always maintain an approximate $\mathcal{H}_\phi(A)$ at all times for a given ϕ.

For any $0 \leq \phi \leq 1$, the *ϕ-quantile* of A is some $x \in U$ such that at most $\phi|A|$ items of A are smaller than x and at most $(1-\phi)|A|$ items of A are greater than x. The quantiles are also called *order statistics* in the statistics literature. In particular, the $\frac{1}{2}$-quantile is also known as the *median* of A. If an ϵ-approximation is allowed, we can return any ϕ'-quantile of A such that $\phi - \epsilon \leq \phi' \leq \phi + \epsilon$. In the *$\phi$-quantile tracking* problem, the coordinator needs to keep an ϵ-approximate ϕ-quantile of A at all times for a given ϕ. We also consider a more general version of the problem, where we would like to keep track of all the quantiles approximately. More precisely, here the "function" f is a data structure from which an ϵ-approximate ϕ-quantile for any ϕ can be extracted. Note that such a structure is equivalent to an (approximate) equal-height histogram, which characterizes the entire distribution.

In particular, from an *all-quantile* structure, we can easily obtain the (2ϵ)-approximate ϕ-heavy hitters for any ϕ, as observed in [6]. Therefore, the all-quantile tracking problem is more general than either the ϕ-heavy hitter tracking problem or the ϕ-quantile tracking problem. In the rest of the paper, we omit the word "approximate" when referring to heavy hitters and quantiles when the context is clear.

Previous work. Various f's have been considered under the distributed tracking streaming model with theoretical bounds, including frequency moments [15, 8], entropy [2], samplings [9] and some non-linear functions [20]. The simplest case $f(A) = |A|$ just counts the total number of items received so far across all the sites. This problem can be easily solved with $O(k/\epsilon \cdot \log n)$ communication where each site simply reports to the coordinator whenever its local count increases by a $1+\epsilon$ factor [15]. The more general single-valued statistics are the frequency moments: $F_p(A) = \sum_x (m_x(A))^p$. F_0 is the number of distinct items, and can be tracked with cost $O(k/\epsilon^2 \cdot \log n \log \frac{n}{\delta})$ [8]; $F_1(A)$ is just $|A|$; F_2 is the self-join size, and can be tracked with cost $O((k^2/\epsilon^2 + k^{3/2}/\epsilon^4) \log n \log \frac{kn}{\epsilon\delta})$ [8].

Heavy hitters and quantiles can better capture the distribution of data than single-valued statistics like the frequency moments, and they have also been studied under the distributed streaming framework. Babcock and Olston [4] designed some heuristics for the top-k monitoring problem, where the goal is to track the k most frequent items (whose frequency may not be larger than $\phi|A|$). Their techniques can be adapted to tracking the heavy hitters [11], but the approach remains heuristic in nature. Manjhi et al. [16] also studied the heavy hitter tracking problem, but their communication model and the goal are different: They organize the sites in a tree structure and the goal is to minimize the communication only at the root node. The all-quantile tracking problem has been studied by Cormode et al. [6], who gave an algorithm with cost $O(k/\epsilon^2 \cdot \log n)$. As commented earlier, this also implies a heavy hitter tracking algorithm with the same cost. This remains the best communication upper bound for both problems to date. No lower bound is known.

Our results. Our main results in this paper are the matching upper and lower bounds on the communication cost for deterministic algorithms for both the heavy hitter tracking problem and the quantile tracking problem. Specifically, we show that for any ϕ, both the ϕ-heavy hitters (Section 2) and the ϕ-quantile (Section 3) can be tracked with total communication cost $O(k/\epsilon \cdot \log n)$. This improves upon the previous result of [6] by a $\Theta(1/\epsilon)$ factor. Note that in the classical communication model, we can easily do a one-shot computation of the ϕ-heavy hitters and the ϕ-quantile easily with cost $O(k/\epsilon)$, as observed in [6]. Interestingly, our results show that requiring the heavy hitters and quantiles to be tracked at all times indeed increases the communication complexity, but only by a $O(\log n)$ factor. In Section 4, we give an algorithm that tracks all quantiles with cost $O(k/\epsilon \cdot \log^2 \frac{1}{\epsilon} \log n)$.

We comment that all our algorithms only handle the arrival of items. The worst-case bounds are mean-

ingless if items are allowed to be deleted. In [22] competitive analysis is proposed to handle this issue, but only under the case that there is only one site and one coordinator. It is still unclear how to apply competitive analysis to multiple sites.

We also provide a matching lower bound of $\Omega(k/\epsilon \cdot \log n)$ for deterministic algorithms for both the ϕ-heavy hitters and the ϕ-quantile problem. Since the all-quantile problem is more difficult than the single-quantile problem, it has the same lower bound as the latter. Thus, our all-quantile tracking algorithm is also optimal up to a $\Theta(\text{polylog}\frac{1}{\epsilon})$ factor. Note that such a lower bound cannot hold for randomized algorithms that succeed with a constant probability (say), as these problems can be solved by drawing a random sample of $O(1/\epsilon^2)$. Cormode et al. [9] show how such a sample can be maintained continuously with communication $O((k + 1/\epsilon^2) \log n)$, breaking the deterministic lower bound when $k \gg 1/\epsilon$.

In the paper, we will focus on the communication cost (or simply the *cost*) while ignoring the memory requirements and running times at each site. In particular, we will assume that each site can remember the exact counts of all the items it has received so far. This is, however, mainly for convenience of presentation. All our algorithms can be in fact implemented both space- and time-efficiently. The observation is that, instead of remembering the exact counts, it is sufficient for a site to maintain the $\Theta(\epsilon)$-approximations of these counts. We will address this issue after describing the respective algorithms that assume exact counts.

2 Tracking the Heavy Hitters

2.1 The upper bound

The algorithm. Let m be the current size of A. First, the coordinator C always maintains $C.m$, an ϵ-approximation of m. This can be achieved by letting each site send its local count every time it has increased by a certain amount (to be specified shortly). Each site S_j maintains the exact frequency of each $x \in U$ at site S_j, denoted $m_{x,j}$, at all times. The overall frequency of x is $m_x = \sum_j m_{x,j}$. Of course, we cannot afford to keep track of m_x exactly. Instead, the coordinator C maintains an underestimate $C.m_{x,j}$ of $m_{x,j}$, and always sets $C.m_x = \sum_j C.m_{x,j}$ as an estimate of m_x. S_j will send its local value of $m_{x,j}$ to C, hence updating $C.m_{x,j}$, from time to time following certain rules to be specified shortly. In addition, each site S_j maintains $S_j.m$, an estimate of m, a counter $\Delta(m_j)$, denoting the increment of m_j since its last communication to C about m_j, as well as a counter $\Delta(m_{x,j})$ for each x, denoting the increment of $m_{x,j}$ since its last communication to C about $m_{x,j}$.

We can assume that the system starts with $m = k/\epsilon$ items; before that we could simply send each item to the coordinator. So when the algorithm initiates, all the estimates are exact. We initialize $\Delta(m_j)$ and $\Delta(m_{x,j})$ for all x to be 0. The protocols of tracking the ϕ-heavy hitters are as follows.

1. *Each site S_j:* When a new item of x arrives, $\Delta(m_j)$ and $\Delta(m_{x,j})$ are incremented by 1. When $\Delta(m_j)$ (resp. $\Delta(m_{x,j})$) reaches $(\epsilon \cdot S_j.m)/3k$, site S_j sends a message $(all, \cdot)$ (resp. $(x, m_{x,j})$) to the coordinator, and resets $\Delta(m_j)$ (resp. $\Delta(m_{x,j})$) to 0.

2. *Coordinator C:* When C has received a message $(all, \cdot)$ or $(x, m_{x,j})$, it updates $C.m$ to $C.m + (\epsilon \cdot S_j.m)/3k$ or $C.m_{x,j}$ to $m_{x,j}$, respectively. Once C has received k signals in the forms of $(all, \cdot)$, it collects the local counts from each site to compute the exact value of m, sets $C.m = m$, and then broadcasts $C.m$ to all sites. Then each site S_j updates its $S_j.m$ to m. After getting a new $S_j.m$, S_j also resets $\Delta(m_j)$ to 0.

Finally, at any time, the coordinator C declares an item x to be a ϵ-approximate ϕ-heavy hitter if and only if

$$\frac{C.m_x}{C.m} \geq \phi + \frac{\epsilon}{2}. \tag{1}$$

Theorem 2.1 *For any $\epsilon \leq \phi \leq 1$, the deterministic algorithm above continuously tracks the ϕ-heavy hitters and incurs a total communication cost of $O(k/\epsilon \cdot \log n)$.*

Proof: We first show that our algorithm correctly tracks the ϕ-heavy hitters, and then analyze its communication complexity.

Correctness: To prove correctness we first establish the following invariants maintained by the algorithm.

$$m_x - \frac{\epsilon m}{3} + k \leq C.m_x \leq m_x, \tag{2}$$

$$m - \frac{\epsilon m}{3} + k \leq C.m \leq m. \tag{3}$$

The second inequalities of both (3) and (2) are obvious. The first inequality of (2) is valid since once a site S_j gets $(\epsilon \cdot S_j.m)/3k$ items of x, it sends a message to the coordinator and the coordinator updates $C.m_x$ accordingly. Thus the maximum error of $C.m_x$ in the coordinator is at most $\sum_{j=1}^{k}(\frac{\epsilon \cdot S_j.m}{3k} - 1) \leq \frac{\epsilon m}{3} - k$. The first inequality of (3) follows from a similar reason. Combining (2) and (3), we have

$$\frac{m_x}{m} - \frac{\epsilon}{3} < \frac{C.m_x}{C.m} < \frac{m_x}{m} \cdot \frac{1}{1 - \epsilon/3} < \frac{m_x}{m} + \frac{\epsilon}{2}, \tag{4}$$

which guarantees that the approximate ratio $\frac{C.m_x}{C.m}$ is within $\epsilon/2$ of $\frac{m_x}{m}$, thus classifying an item using (1) will not generate any false positives or false negatives.

Communication complexity: We divide the whole tracking period into rounds. A round starts at the time when the coordinator finishes a broadcast of $C.m$ to the time when it initiates the next broadcast. Since the coordinator initiates a broadcast after $C.m$ is increased by a factor of $1 + \sum_{i=1}^{k}(\epsilon/3k) = 1 + \epsilon/3$, the number of rounds is bounded by

$$\log_{1+\epsilon/3} n = O\left(\frac{\log n}{\epsilon}\right).$$

In each round, the number of messages in the form of $(all, \cdot)$ sent by all the sites is k by the definition of our protocol. Since there are $O(\log n/\epsilon)$ rounds in total, the number of messages in the form of $(all, \cdot)$ can be bounded by $O(k/\epsilon \cdot \log n)$. On the other hand, it is easy to see that total number of messages of the form $(x, m_{j,x})$ is no more than the total number of messages of the form $(all, \cdot)$. Therefore, the total cost of the whole system is bounded by $O(k/\epsilon \cdot \log n)$. □

2.2 Implementing with small space

In the algorithm described above, we have assumed that each site S_j maintains all of its local frequencies $m_{x,j}$ exactly. In this section we show that our algorithm still works if we replace these exact frequencies with a heavy hitter sketch, such as the *space-saving* sketch [18], which we describe here for completeness:

Algorithm space-saving. We maintain a list L of $1/\epsilon$ items together with their counters. For each $x \in L$, we use $f(x)$ denote its counter. We keep L in the sorted order of $f(x)$, and denote by `MIN` the smallest $f(x)$ in L. We let `MIN` $= 0$ if L has less than $1/\epsilon$ items. When a new item x arrives, we do the following:

1. If x is in L, increment $f(x)$ by 1.

2. If x is not in L, we add x to L with $f(x) = \texttt{MIN} + 1$. If L now has $1/\epsilon + 1$ items, delete the one with the smallest $f(x)$.

Now we will estimate the frequency of any $x \in L$ with $f(x)$, and any $x \notin L$ as 0. If we run this algorithm at each site S_j, then we have the following guarantees.

Lemma 2.2 ([18]) *The space-saving algorithm maintains an approximation of $m_{x,j}$ for any $x \in U$ with error at most $\epsilon|S_j|$, where $|S_j|$ denotes the current number of items received at S_j so far. The algorithm uses $O(1/\epsilon)$ space and spends amortized $O(1)$ time per item.*

We now run our tracking algorithm with the following changes.

1. We replace the error parameter ϵ in the tracking algorithm with $\epsilon_2 = \epsilon/2$.

2. At each site S_j, we run the space-saving algorithm with error parameter $\epsilon_1 = \epsilon/6$. Only for those items x in the list L do we maintain the increment counter $\Delta(m_{x,j})$. When a new item x is added to L, we set $\Delta(m_{x,j}) = 0$. And when an item is deleted, we delete its corresponding counter.

It is easy to see that the total space used by each site is $O(1/\epsilon_1) = O(1/\epsilon)$ and the processing time per item is $O(1)$. In the rest of the section we prove that the modified algorithm is still correct and the asymptotic communication cost remains the same.

For correctness, note that after the modifications, Equation (3) is not affected since we only decrease the error of tracking the total number of items. And Equation (2) also holds since the maximum underestimate of $C.m_x$ for the item x in the coordinator is at most $\sum_{j=1}^{k}((\frac{\epsilon_2 \cdot S_j.m}{3k} - 1) + \epsilon_1|S_j|) \leq \frac{\epsilon m}{3} - k$, where the second term on the left side of the inequality is the error introduced by the space-saving algorithm. For the communication complexity, the previous analysis can still go through except that we should replace ϵ by $\epsilon_2 = \epsilon/2$, which will not change the asymptotic results.

To sum up, our algorithm can be implemented in $O(1/\epsilon)$ space per site and amortized $O(1)$ time per item.

2.3 The lower bound

To give a lower bound on the total communication cost that any deterministic tracking algorithm must take, we first present a combinatorial result, that the number of changes that the set of heavy hitters could experience is $\Omega(\log n/\epsilon)$, where a *change* is defined to be the transition of the frequency of an item from above $\phi|A|$ to below $(\phi-\epsilon)|A|$, or the other way round. Then we show that to correctly detect each change, the sites must exchange at least $\Omega(k)$ messages, establishing an $\Omega(k/\epsilon \cdot \log n)$ communication lower bound.

Lower bound on the number of changes. The following lemma could be proved by construction.

Lemma 2.3 *For any $\phi > 3\epsilon$, there is a sequence of item arrivals in which the set of heavy hitters will have $\Omega(\log n/\epsilon)$ changes.*

Proof: Set $\epsilon' = 2\epsilon$. We construct two groups of $l = 1/(2\phi - \epsilon')$ items each: $\mathcal{S}_0 = \{1, 2, \ldots, l\}$ and $\mathcal{S}_1 = \{l+1, l+2, \ldots, 2l\}$. Since we only care about the total number of changes of the set of heavy hitters during the whole tracking period, we temporarily treat the whole system as one big site and items come one

by one. We will construct an input sequence under which the set of heavy hitters will undergo $\Omega(\log n/\epsilon)$ changes.

We still divide the whole tracking period to several rounds, and let m_i denote the total number of items when round i starts. The following invariant will be maintained throughout the construction:

> Let $b = i \bmod 2$. When round i starts, all items $a \in \mathcal{S}_b$ have frequency ϕm_i, and all items $a \in \mathcal{S}_{1-b}$ have frequency $(\phi - \epsilon')m_i$.

It can be verified that the total frequency of all items is indeed m_i. Note that from the start of round i to the end of round i, all the non-heavy hitters become heavy hitters, and all the heavy hitters become non-heavy hitters. In what follows we only care about the changes of the former type, which lower bounds the number of changes. To maintain the invariant for round $i+1$, we construct item arrivals as follows. Without loss of generality, suppose $\mathcal{S}_{1-b} = \{1, 2, \ldots, l\}$. Let $\beta = \frac{\epsilon'(2\phi-\epsilon')}{\phi-\epsilon'}$. We first generate βm_i copies of 1, and then βm_i copies of 2, ..., then βm_i copies of l, in sequence. After these items we end round i and start round $i+1$. At this turning point, the total number of items is

$$m_{i+1} = m_i + l \cdot \beta m_i = \frac{\phi}{\phi - \epsilon'} m_i.$$

Now the frequency of each item in the set $\mathcal{S}_{1-b}$ is

$$(\phi - \epsilon')m_i + \beta m_i = \phi \cdot \frac{\phi}{\phi - \epsilon'} m_i = \phi m_{i+1},$$

and the frequency of each item in $\mathcal{S}_b$ remains the same, that is, $\phi m_i = (\phi - \epsilon')m_{i+1}$. Now we have restored the invariant and can start round $i+1$.

Finally, we bound the number of rounds. Since the total number of items m_i increases by a $\phi/(\phi - \epsilon')$ factor in each round, the total number of rounds is $\Theta\left(\log_{\frac{\phi}{\phi-\epsilon'}} n\right)$. Consequently, the total number of changes in the set of heavy hitters (from non-heavy hitters to heavy hitters) is $l \cdot \Theta\left(\log_{\frac{\phi}{\phi-\epsilon'}} n\right) = \Omega\left(\frac{1}{\phi-\epsilon} \cdot \frac{\phi-\epsilon'}{\epsilon'} \log n\right) = \Omega(\log n/\epsilon)$. □

Communication lower bound. Now we go back to the distributed scenario and consider the cost of communication for "recognizing" each change. First we need to state our lower bound model more formally. The input consists of two sequences $A = (a_1, \ldots, a_n)$ and $J = (j_1, \ldots, j_n)$, which represent that the t-th item to arrive in the whole system is $a_t \in U$, which arrives at site S_{j_t}. Our model does not allow spontaneous communication, i.e., a site S_j can send out a message only when an item arrives at S_j, or in response to an incoming message from the coordinator; likewise, the coordinator can only send out messages to one or more sites in response to an incoming message. This is in fact not a restriction, because if communication happens between the arrival of two items a_t and a_{t+1}, we can push all this communication forward to the time instance when a_t arrives. Thus, communication only takes place at each of the n discrete time instances when the n items arrive respectively.

Let

$$P_t = \{j \mid S_j \text{ participates in the communication when } a_t \text{ arrives}\}.$$

For lower bound purposes, we simply define the total communication cost as $\sum_{t=1}^{n} |P_t|$. This ignores the message size, meaning that the site can send out its entire state with a communication cost of 1. Let

$A(t) = (a_1, \ldots, a_t)$ be the prefix of length t of the input A, and let $A(t, j)$ be the subsequence of $A(t)$ that arrive at site S_j. Let $M(t, j)$ be the message history between S_j and the coordinator before a_t arrives.

When a_t arrives at site S_{j_t}, S_{j_t} is the only site who can initiate communication. We assume that this decision (for a deterministic algorithm) only depends on $A(t, j_t)$ and $M(t, j_t)$, which is all the information available to S_{j_t} when a_t arrives. One may think that the current time (when a_t arrives) should also be taken into account. But since we allow the items to arrive at arbitrary times, the actual time instance at which a_t arrives conveys no more information than the fact that it is the latest item in $A(t, j_t)$.

It remains to specify what is considered as correct behavior for the algorithm. Because we allow some approximation when classifying heavy hitters and non-heavy hitters, the algorithm is allowed to report an item as a heavy hitter any time in the interval from the point when its frequency just passes $(\phi - \epsilon)|A|$ to the point when its frequency reaches $\phi|A|$. To make the model clean, we will only have the relaxed requirement that P_t should be nonempty for at least one t during this time interval.

With the model fully specified, we can now prove the following.

Lemma 2.4 *Consider the sequence of items A used in the proof of lemma 2.3. There is a sequence of arrival sites $J = (j_1, \ldots, j_n)$ such that, to correctly recognize each change, any deterministic algorithm has to incur a communication cost of $\Omega(k)$.*

Proof: Recall that in the i-the round of the construction of A, there are a batch βm_i copies of 1, followed by a batch βm_i of 2, and so on so forth, where each batch converts the respective item from a non-heavy hitter to a heavy hitter. Note that the algorithm is required to communicate at least once during the second half of the batch to report the item as a heavy hitter, and it should do so for *all* arrival pattern J. Below we will use an adversary argument to show that under this constraint, there must be one sequence J under which the communication cost is $\Omega(k)$ for every batch.

Focus on a particular batch of βm_i copies of item a. For each site S_j, imagine that we feed all of the βm_i copies of a to S_j one by one. Suppose that after n_j copies have arrived, S_j decides to communicate for the first time (note that the other sites must stay silent before S_j initiates communication). We call n_j the *triggering threshold* for S_j. Since the n_j's are a property of the deterministic algorithm, we may assume that they are known to the adversary.

We argue it must be the case that

$$\sum_{j=1}^{k} (n_j - 1) \leq \beta m_i. \tag{5}$$

Suppose not, then we can send $n_j - 1$ copies of a to S_j for $j = 1, \ldots, k$. From the definition of the triggering thresholds, this will not trigger any communication, and thus will make the algorithm miss to report the change. From (5), there must an S_j such that $n_j \leq \beta m_j/k + 1$. Then the adversary sends the first $\beta m_j/k + 1$ copies of a to S_j. This will cause S_j to communicate at least once. Then we apply the argument above again. Note that the n_j's may have changed (if a site has received a message from the coordinator), but (5) still must hold (in fact, a stronger version holds where the RHS is replaced by $\beta m_i - (\beta m_i/k + 1)$). Thus, we can repeat this process until the first half of the batch has been sent, i.e., this argument can be repeated for $\frac{\beta m_i/2}{\beta m_i/k+1} = \Omega(k)$ times, triggering at least $\Omega(k)$ messages. □

Combining Lemma 2.3 and Lemma 2.4, the overall over bound immediately follows.

Theorem 2.5 *Any deterministic algorithm that continuously tracks the ϕ-heavy hitters has to incur a total communication cost of $\Omega(k/\epsilon \cdot \log n)$, for any $\phi > 3\epsilon$.*

3 Tracking the Median

In this section we first present an algorithm to track any ϕ-quantile for $0 \leq \phi \leq 1$. For ease of presentation we describe how to track the median (the $1/2$-quantile); the generalization to any ϕ-quantile with $\epsilon < \phi/3$ is straightforward. Then we give a matching lower bound. For the upper bound, we will again first assume that each site has unlimited space and processing power, and then address the issue of how to implement the protocols efficiently.

3.1 The upper bound

For simplicity we assume that all the items in A are distinct; this assumption can be removed easily with any tie-breaking rule. As in Section 2 we can track (an approximation) of $|A|$, using which we divide the whole tracking period into $O(\log n)$ rounds; whenever $|A|$ has increased by a constant factor, we start a new round. In the following we focus on one round, and show that our median-tracking algorithm has a communication cost of $O(k/\epsilon)$. We set m to be the cardinality of A at the beginning of the round. m is fixed throughout the round and we always have $m \leq |A|$.

The main idea of our algorithm is to maintain a dynamic set of disjoint intervals in the coordinator (by maintaining a set of separating items), such that each interval contains between $\frac{\epsilon}{8}m$ and $\frac{\epsilon}{4}m$ items. We first show that if we have such a set of intervals, the median can be tracked efficiently. Afterward we discuss how to maintain these intervals.

Let M denote the approximate median that is kept at the coordinator. We maintain two counters $C.\Delta(L)$ and $C.\Delta(R)$, counting the number of items that have been received at all sites to the left and the right of M, respectively. These two counters are maintained as underestimates with an absolute error at most $\frac{\epsilon}{8}m$, by asking each site to send in an update whenever it has received $\frac{\epsilon}{8k}m$ items to the left or right of M. So the cost of maintaining them is $O(k/\epsilon)$.

Whenever $|C.\Delta(L) - C.\Delta(R)| \geq \frac{\epsilon}{2}m$, we update M as follows.

1. Compute $C.L$ and $C.R$ as the total number of items to the left and the right of M. W.l.o.g., suppose $C.L > C.R$ and let $d = (C.L - C.R)/2$.
2. Compute a new median M' such that $|(r(M) - d) - r(M')| \leq \frac{\epsilon}{4}m$ where $r(M)$ is the rank of M in A. Update M to M'.
3. Reset $C.\Delta(L)$ and $C.\Delta(R)$ to 0.

Lemma 3.1 *Suppose a set of intervals with lengths between $\frac{\epsilon}{8}m$ and $\frac{\epsilon}{4}m$ can be maintained, the algorithm above continuously tracks the ϵ-approximation median in one round and incurs a communication cost $O(k/\epsilon)$.*

Proof: For the correctness of the algorithm, we can show that our tracking algorithm always maintains an approximate median that is at most $\frac{\epsilon}{4}m + \frac{3\epsilon}{4}m = \epsilon m$ items away from the exact median. The first term $\frac{\epsilon}{4}m$ is due to the fact that whenever we update M, M is within an error of at most $\frac{\epsilon}{4}m$ to the exact median. This is because at the Step 2 of the update, $(r(M) - d)$ is the rank of the true median. The second term $\frac{3\epsilon}{4}m$ accounts for the error introduced by the triggering condition $|C.\Delta(L) - C.\Delta(R)|$ monitored in the coordinator. Note that we keep both $C.\Delta(L)$ and $C.\Delta(R)$ within an additive error of at most $\frac{\epsilon}{8}m$ and whenever $|C.\Delta(L) - C.\Delta(R)| \geq \frac{\epsilon}{2}m$, we initiate an update. Therefore, the total error introduced is at most $2 \cdot \frac{\epsilon}{8}m + \frac{\epsilon}{2}m = \frac{3\epsilon}{4}m$.

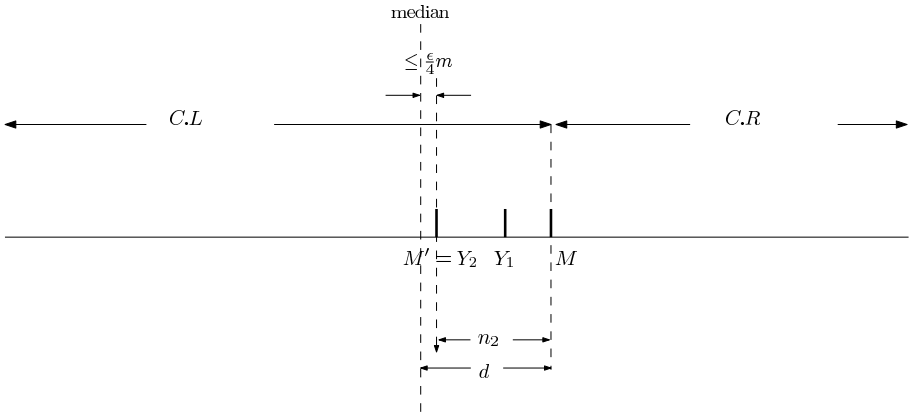

Figure 1: Update the approximate median M.

Now we analyze the communication cost. Step 1 could be done by exchanging $O(k)$ messages. For step 2, first note that $d \leq \epsilon m$ since by the reasoning above, M is still an ϵ-approximate median. Next, we can find M' quickly with the help of the set of intervals. We start by finding the first separating item Y_1 of the intervals to the left of M, and then collect information from all sites to compute the number of items in the interval $[Y_1, M]$, say n_1. If $|n_1 - d| \leq \frac{\epsilon}{4}m$, we are done; otherwise we go on to pick the second separating item Y_2 to the left of M, and check if $|n_2 - d| \leq \frac{\epsilon}{4}m$, where n_2 is the number of items in the interval $[Y_2, M]$. See Figure 1. It is easy to see that after at most $O(1)$ such probes, we can find an item Y such that the rank difference between Y and the exact median is no more than $\frac{\epsilon}{4}m$. Note that the cost of each probe is $O(k)$ thus the total cost of step 2 is $O(k)$. Finally, we update M at most $O(1/\epsilon)$ times within a single round, since each update increases $|A|$ by at least a factor of $1 + \frac{\epsilon}{2}$. To sum up, the total cost of the algorithm within a round is $O(k/\epsilon)$ provided that the dynamic set of intervals are maintained. □

Maintaining the set of intervals. When a new round starts, we initialize the set of intervals as follows: Each site S_j $(1 \leq j \leq k)$ computes a set of intervals, each containing $\frac{\epsilon|A_j|}{32}$ items, where A_j stands for the set of items S_j has received, and then sends the set of intervals to the coordinator (by sending those separating items). Then the coordinator can compute the rank of any $x \in U$ with an error of at most $\sum_{j=1}^{k} \frac{\epsilon}{32}|A_j| = \frac{\epsilon}{32}m$, therefore it can compute a set of intervals, each of which contains at least $\frac{\epsilon}{8}m$ and at most $\frac{\epsilon}{4}m$ items. After the coordinator has built the set of intervals, it broadcasts them to all the k sites, and then computes the exact number of items in each interval. The cost of each rebuilding is $O(k/\epsilon)$.

During each round, each site S_j maintains a counter for each interval as new items arrive. And whenever the local counter of items in some interval I has increased by $\frac{\epsilon}{4k}m$, it sends a message to the coordinator and the coordinator updates the count for interval I accordingly. Whenever the count of some interval in the coordinator C reaches $\frac{\epsilon}{4}m$, the coordinator splits the interval into two intervals, each of which containing at least $\frac{\epsilon}{8}m$ and at most $\frac{\epsilon}{4}m$ items. To perform such a split, we can again call the rebuilding algorithm above, except that the rebuilding is only applied to the interval I, so the cost is only $O(k)$.

The correctness of algorithm is obvious. The total communication cost of interval splits is $O(k/\epsilon)$ in each round, since there are at most $O(1/\epsilon)$ splits and each split incurs a communication cost $O(k)$.
Now we conclude with the following theorem.

Theorem 3.2 *There is a deterministic algorithm that continuously tracks the ϵ-approximate median (and*

generally, any ϕ-quantile $(0 \leq \phi \leq 1)$) and incurs a total communication cost of $O(k/\epsilon \cdot \log n)$.

Implementing with small space. Similar to our heavy hitter tracking algorithm, instead of maintaining the intervals exactly at each site, we can again deploy a sketch that maintains the approximate ϵ'-quantiles for some $\epsilon' = \Theta(\epsilon)$ to maintain these intervals approximately.

We use the following fact from Greenwald-Khanna [13].

Fact 3.1 *There exists a sketching algorithm (the GK algorithm) that maintains the local ϵ_1-approximate quantile with $O(1/\epsilon_1 \cdot \log(\epsilon_1 n))$ space per site and amortized $O(\log n)$ time per item.*

Now we set $\epsilon_1 = \frac{\epsilon}{32}$, and run the GK algorithm locally at each site. At the time of each rebuild we can guarantee that the rank of any $x \in U$ with an error of at most $\frac{\epsilon}{32}m + \sum_{j=1}^{k} \frac{\epsilon}{32}|A_j| = \frac{\epsilon}{16}m$. The first term of the left side of the equality is from the original analysis; and the second term is due to the extra imprecision introduced by the GK algorithm at each site. Now based on this information the coordinator can still compute a set of intervals, each of which contains at least $\frac{\epsilon}{8}m$ and at most $\frac{\epsilon}{4}m$ items.

To sum up, our algorithm can be implemented in $O(1/\epsilon \cdot \log(\epsilon n))$ space per site and amortized $O(\log n)$ time per item.

3.2 The lower bound

The proof of the lower bound for tracking the median is very similar to that for the heavy hitters. We construct a sequence of item arrivals with the following properties.

1. The median will change at least $\Omega(\log n/\epsilon)$ times.
2. To correctly recognize each change, any deterministic algorithm has to incur a communication cost of $\Omega(k)$.

Consider the following construction. The universe consists of only two items 0 and 1. We divide the whole tracking period to $O(\log n)$ rounds and let m_i be the number of items at the beginning of round i. We maintain the following invariant: When round i starts, the frequency of item b is $(0.5 - 2\epsilon)m_i$ and the frequency of item $1-b$ is $(0.5+2\epsilon)m_i$, where $b = i \bmod 2$. This could be done by inserting $\frac{4\epsilon}{0.5-2\epsilon}m_i$ copies of b during round i and then start a new round. It is easy to see that there will be at least $\Omega(\log n/\epsilon)$ rounds and the median will change at least once during each round, therefore the total number of changes of the median is $\Omega(\log n/\epsilon)$. For the second property, we can invoke the same arguments as that for Lemma 2.4. Combining the two properties, we have the following.

Theorem 3.3 *Any deterministic algorithm that continuously tracks the approximate median has to incur a total communication cost of $\Omega(k/\epsilon \cdot \log n)$.*

4 Tracking All Quantiles

In this section, we give a tracking algorithm so that the coordinator C always tracks the ϵ-approximate ϕ-quantiles for all $0 \leq \phi \leq 1$ simultaneously. We will solve the following equivalent problem: The coordinator is required to maintain a data structure from which we can extract the rank $r(x)$ for any $x \in U$ in A with an additive error at most $\epsilon|A|$. We still assume that all items in A are distinct.

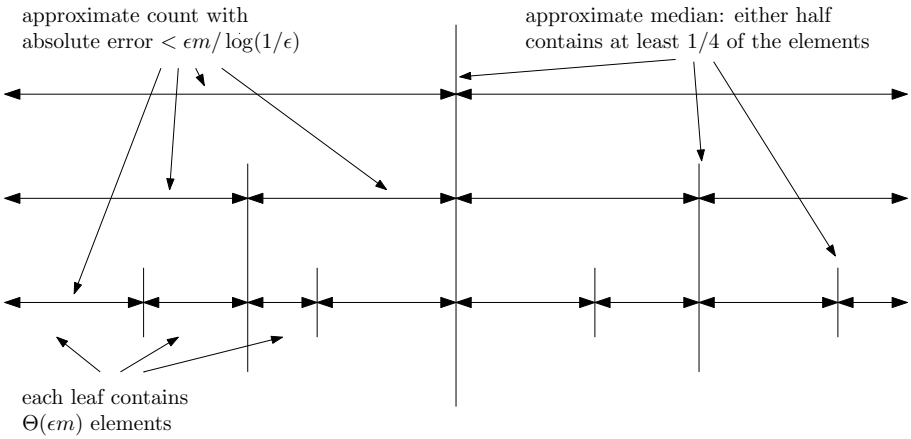

Figure 2: The data structure that can be used to extract the rank of any x with absolute error $< \epsilon m$.

We divide the whole tracking period into $O(\log n)$ rounds. In each round $|A|$ roughly doubles. We will show that the algorithm's cost in each round is $O(k/\epsilon \cdot \log^2 \frac{1}{\epsilon})$. The algorithm restarts itself at the beginning of each round, therefore the total communication of the algorithm will be $O(k/\epsilon \cdot \log n \log^2 \frac{1}{\epsilon})$.

The data structure. Let m be the cardinality of A at the beginning of a round. The data structure is a binary tree $\mathcal{T}$ with $\Theta(1/\epsilon)$ leaves. The root r of $\mathcal{T}$ corresponds to the entire A. It stores a splitting element x_r which is an approximate median of A, i.e., it divides A into two parts, either of which contains at least $(\frac{1}{2} - \alpha)|A|$ and at most $(\frac{1}{2} + \alpha)|A|$ items, for some constant $0 < \alpha < \frac{1}{2}$. Then we recursively build r's left and right subtrees on these two parts respectively, until there are no more than $\epsilon m/2$ items left. It is clear that $\mathcal{T}$ has $\Theta(1/\epsilon)$ nodes in total, and has height at most $h = \log_{\frac{1}{2}+\alpha} \frac{\epsilon}{2} = \Theta(\log \frac{1}{\epsilon})$, though it is not necessarily balanced. Each node in $\mathcal{T}$ is naturally associated with an interval. Let I_u be the interval associated with u. Then I_r is the entire U; suppose v and w are u's children, then I_u is divided into I_v and I_w by x_u. Set $\theta = \frac{\epsilon}{2h}$. Each node u of $\mathcal{T}$ is in addition associated with s_u, which is an underestimate of $|A \cap I_u|$ with an absolute error of at most θm, i.e., $|A \cap I_u| - \theta m \leq s_u \leq |A \cap I_u|$. Please see Figure 2 for an illustration of the data structure.

Lemma 4.1 *If the coordinator can maintain the above data structure, then it can compute the rank of x with an absolute error of at most ϵm.*

Proof: For a given x, we first search down the binary tree and locate the leaf v such that $x \in I_v$. As we go along the root-to-leaf path, whenever we follow a right child, we add up the s_u of its left sibling. In the end we add up h such partial sums, each contributing an error of at most θm, totaling $\theta m \cdot h = \epsilon m/2$. Finally, since $|A \cap I_v| < \epsilon m/2$, the sum of all the s_u's for the preceding intervals of x is off by at most ϵm from the actual rank of x. □

Now we show how to efficiently maintain such a data structure at the coordinator side.

Initialization. At the beginning of each round, we initialize the data structure similarly as in Section 3. Suppose the set of items at S_j is A_j. Each site S_j builds its own structure $S_j.\mathcal{T}$, but with $\epsilon/32$ as the error parameter, and ships to C. This costs a communication of $O(k/\epsilon)$. Note that $S_j.\mathcal{T}$ allows one to extract the rank of any x within A_j with an error of $\epsilon/32 \cdot |A_j|$. By querying each $S_j.\mathcal{T}$, the coordinator can compute

the rank of any x with an error of $\sum_{i=1}^{k} \frac{\epsilon}{32}|A_i| = \frac{\epsilon}{32}m$, which is enough for the coordinator to build its own $C.\mathcal{T}$. In particular, all the splitting elements can be chosen to be within a distance of $\frac{\epsilon}{32}m$ to the real median. After building $C.\mathcal{T}$, the coordinator broadcasts it to all the sites, costing communication $O(k/\epsilon)$. Now each site S_j knows how U is subdivided into those $\Theta(1/\epsilon)$ intervals represented by the binary tree $\mathcal{T}$. Then for each interval I_u, it computes $|A_j \cap I_u|$ and sends the count to C, so that the coordinator has all the exact partial sums s_u to start with. It is easy to see that the total communication cost for initializing the data structure is $O(k/\epsilon)$.

Maintaining the partial sums. As items arrive, each site S_j monitors all the intervals I_u in $\mathcal{T}$. For each I_u, every time the local count of items in I_u at S_j has increased by $\theta m/k$, it sends an updated local count to C. Thus in the worst case, each site is holding $(\theta m/k - 1)$ items that have not been reported, leading to a total error of at most θm. The cost of these messages can be bounded as follows. When S_j sends a new count for some interval I_u, we charge the cost to the $\theta m/k$ new items that have arrived since the last message for I_u, $O(k/(\theta m))$ each. Since each item contributes to the counts of at most h intervals, it is charged $O(h)$ times, so the total cost charged to one item is $O(\frac{kh}{\theta m})$. There are a total of $O(m)$ items in a single round, so the overall cost is $O(kh/\theta) = O(k/\epsilon \cdot \log^2 \frac{1}{\epsilon})$.

Maintaining the splitting elements. The maintenance algorithm above ensures that all the s_u are within the desired error bound. We still need to take care of all the splitting elements, making sure that they do not deviate from the real medians too much. Specifically, when we build $\mathcal{T}$, for any u with children v and w, we ensure that

$$\frac{3}{8}|A \cap I_u| \leq |A \cap I_v| \leq \frac{5}{8}|A \cap I_u|. \tag{6}$$

This property can be easily established during initialization, since $|A \cap I_u| > \frac{\epsilon}{2}m$ for any internal node u of $\mathcal{T}$, and we can estimate $|A \cap I_v|$ with an error of $\frac{\epsilon}{32}m$. In the middle of the round, we maintain the following condition:

$$\frac{1}{4}s_u \leq s_v \leq \frac{3}{4}s_u. \tag{7}$$

Recall that s_u (resp. s_v) is an estimate of $|A \cap I_u|$ (resp. $|A \cap I_v|$) with an error of at most θm. As long as (7) holds, we have

$$\frac{1}{4}(|A \cap I_u| - \theta m) \leq \frac{1}{4}s_u \leq s_v \leq |A \cap I_v| + \theta m.$$

Rearranging,

$$|A \cap I_v| \geq \frac{1}{4}|A \cap I_u| - \frac{5}{4} \cdot \frac{\epsilon}{2h}m \geq \frac{1}{4}|A \cap I_u| - \frac{5}{4} \cdot \frac{1}{h}|A \cap I_u| \geq \frac{3}{32}|A \cap I_u|,$$

for $h \geq 8$. (Note that assuming h larger than any constant does not affect our asymptotic results.) Similarly, we also have $|A \cap I_v| \leq \frac{29}{32}|A \cap I_u|$. Thus condition (7) ensures that the height of $\mathcal{T}$ is bounded by $h = \Theta(\log \frac{1}{\epsilon})$.

Whenever (7) is violated, we do a partial rebuilding of the subtree rooted at u to restore this condition. If multiple conditions are violated at the same time, we rebuild at the highest such node. To rebuild the subtree rooted at u, we apply our initialization algorithm, but only for the range I_u. This incurs a cost of $O(k\frac{|A \cap I_u|}{\epsilon m})$, since we are essentially building a new data structure on $|A \cap I_u|$ elements with error parameter $\epsilon' = \epsilon m/|A \cap I_u|$. After rebuilding, we have restored (6) for u and all its descendants.

Lemma 4.2 *The total cost of all the partial rebuildings in a round is $O(k/\epsilon \cdot \log \frac{1}{\epsilon})$.*

Proof: Similarly as before, we show that when (7) is violated, we must have

$$|A \cap I_v| < \frac{21}{64}|A \cap I_u|, \tag{8}$$

or

$$|A \cap I_v| > \frac{43}{64}|A \cap I_u|, \tag{9}$$

assuming $h \geq 16$. Note that both $|A \cap I_v|$ and $|A \cap I_u|$ may increase. From (6) to (8), $|A \cap I_u|$ must increase by $\Omega(|A \cap I_v|) = \Omega(|A \cap I_u|)$; from (6) to (9), $|A \cap I_v|$ must increase by $\Omega(|A \cap I_u|)$, which implies that $|A \cap I_u|$ must also increase by $\Omega(|A \cap I_u|)$ since $I_v \subset I_u$. This means that between two partial rebuildings of u, $|A \cap I_u|$ must have increased by a constant factor. Thus, we can charge the rebuilding cost of u to the $\Omega(|A \cap I_u|)$ new items that have arrived since the last rebuilding, $O(k/(\epsilon m))$ each. Since each item is contained in the intervals of $O(h)$ nodes, it is charged a cost of $O(hk/(\epsilon m))$ in total. Therefore, the total cost of all the partial rebuildings in a round is $O(hk/\epsilon) = O(k/\epsilon \cdot \log \frac{1}{\epsilon})$. □

Maintaining the leaves. Finally, we need to make sure that $|A \cap I_v| \leq \frac{\epsilon}{2}m$ for each leaf v as required by the data structure. During initialization, we can easily ensure that $\frac{1}{8}\epsilon m \leq |A \cap I_v| \leq \frac{3}{8}\epsilon m$. During the round, the coordinator monitors s_v, and will split v by adding two new leaves below v whenever $s_v > (\frac{\epsilon}{2} - \theta)m$. Since s_v has error at most θm, this splitting condition will ensure that $|A \cap I_v| \leq \frac{\epsilon}{2}m$. To split v, we again call our initialization algorithm on the interval I_v, incurring a cost of $O(k\frac{|A \cap I_v|}{\epsilon m}) = O(k)$. Since we create at most $O(1/\epsilon)$ leaves in this entire round, the total cost for all the splittings is $O(k/\epsilon)$.

Putting everything together, we obtain the following result.

Theorem 4.3 *There is a deterministic algorithm that continuously tracks the ϕ-quantiles for all $0 \leq \phi \leq 1$ simultaneously and incurs a total communication cost of $O(k/\epsilon \cdot \log n \log^2 \frac{1}{\epsilon})$.*

Implementing with small space. Similar as the median case, instead of maintaining the counts in the intervals associated with $\mathcal{T}$ exactly at each site, we can again deploy a sketch that maintains the approximate ϵ_1-quantiles for some $\epsilon_1 = \theta/c$ (c is some sufficiently small constant) to maintain these intervals approximately. Suppose we use the Greenwald-Khanna sketch [13], then we can implement our all-quantile tracking algorithm with $O(1/\theta \cdot \log(\theta n)) = O(1/\epsilon \cdot \log \frac{1}{\epsilon} \log(\epsilon n))$ space per site and amortized $O(\log n)$ time per item.

5 Open Problems

We have restricted ourselves to deterministic algorithms in the paper. If randomization is allowed, simple random sampling can be used to achieve a cost of $O((k + 1/\epsilon^2) \cdot \text{polylog}(n, k, 1/\epsilon))$ for tracking both the heavy hitters and the quantiles. This observation has been well exploited in maintaining the heavy hitters and quantiles for a single stream when both insertions and deletions are present (see e.g. [12]). This breaks the deterministic lower bound for $\epsilon = \omega(1/k)$. It is not known if randomization can still help for smaller ϵ. Another possible direction is to design algorithms to track the heavy hitters and quantiles within a sliding window in the distributed streaming model or even allowing arbitrary insertion and deletion of items.

References

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58:137–147, 1999.

[2] C. Arackaparambil, J. Brody, and A. Chakrabarti. Functional monitoring without monotonicity. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*, 2009.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of the ACM Symposium on Principles of Database Systems*, 2002.

[4] B. Babcock and C. Olston. Distributed top-k monitoring. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

[5] G. Cormode and M. Garofalakis. Approximate continuous querying over distributed streams. *ACM Transactions on Database Systems*, 33(2), Article 9, 2008.

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

[7] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. In *Proceedings of the International Conference on Very Large Databases*, 2008.

[8] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. *ACM Transactions on Algorithms*, 7(2), Article 21, 2011.

[9] G. Cormode, S. Muthukrishnan, K. Yi and Q. Zhang. Optimal sampling from distributed streams. In *Proceedings of the ACM Symposium on Principles of Database Systems*, 2010.

[10] G. Cormode, S. Muthukrishnan, and W. Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *Proceedings of the IEEE International Conference on Data Engineering*, 2006.

[11] R. Fuller and M. Kantardzic. FIDS: Monitoring frequent items over distributed data streams. In *Proceedings of the International Conference on Machine Learning and Data Mining*, 2007.

[12] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *Proceedings of the International Conference on Very Large Databases*, 2002.

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

[14] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems*, 28(1):51–55, 2003.

[15] R. Keralapura, G. Cormode, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2006.

[16] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *Proceedings of the IEEE International Conference on Data Engineering*, 2005.

[17] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the International Conference on Very Large Databases*, 2002.

[18] A. Metwally, D. Agrawal, and A. E. Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems*, 31(3):1095-1133, 2006.

[19] C. Olston and J. Widom. Efficient monitoring and querying of distributed, dynamic data via approximate replication. *IEEE Data Engineering Bulletin*, 2005.

[20] I. Sharfman, A. Schuster, and D. Keren. Shape sensitive geometric monitoring. In *Proceedings of the ACM Symposium on Principles of Database Systems*, 2008.

[21] A. C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the ACM Symposium on Theory of Computation*, 1979.

[22] K. Yi and Q. Zhang. Multi-Dimensional online tracking. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2008.