Algorithms for Distributed Functional Monitoring

Graham Cormode
AT&T Labs
Florham Park, NJ
graham@research.att.com

S. Muthukrishnan
Google Inc.
New York, NY
muthu@google.com

Ke Yi*
Hong Kong U.S.T.
Kowloon, Hong Kong
yike@cse.ust.hk

Abstract

We study what we call *functional monitoring* problems. We have k players each tracking their inputs, say player i tracking a multiset $A_i(t)$ up until time t, and communicating with a central coordinator. The coordinator's task is to monitor a given function f computed over the union of the inputs $\cup_i A_i(t)$, *continuously* at all times t. The goal is to minimize the number of bits communicated between the players and the coordinator. A simple example is when f is the sum, and the coordinator is required to alert when the sum of a distributed set of values exceeds a given threshold τ. Of interest is the approximate version where the coordinator outputs 1 if $f \geq \tau$ and 0 if $f \leq (1-\epsilon)\tau$. This defines the (k, f, τ, ϵ) distributed, functional monitoring problem.

Functional monitoring problems are fundamental in distributed systems, in particular sensor networks, where we must minimize communication; they also connect to problems in communication complexity, communication theory, and signal processing. Yet few formal bounds are known for functional monitoring.

We give upper and lower bounds for the (k, f, τ, ϵ) problem for some of the basic f's. In particular, we study frequency moments (F_0, F_1, F_2). For F_0 and F_1, we obtain continuously monitoring algorithms with costs almost the same as their one-shot computation algorithms. However, for F_2 the monitoring problem seems much harder. We give a carefully constructed multi-round algorithm that uses "sketch summaries" at multiple levels of detail and solves the (k, F_2, τ, ϵ) problem with communication $\tilde{O}(k^2/\epsilon + (\sqrt{k}/\epsilon)^3)$. Since frequency moment estimation is central to other problems, our results have immediate applications to histograms, wavelet computations, and others. Our algorithmic techniques are likely to be useful for other functional monitoring problems as well.

1 Introduction

We introduce *distributed, functional monitoring* with a basic problem, SUM. Suppose we have two observers, Alice and Bob, who each see arrivals of items over time. At time t, Alice has set $A(t)$ of items and Bob has set $B(t)$ of items. Both Alice and Bob have an individual two-way communication channel with Carole so that Carole can monitor $C(t) = |A(t)| + |B(t)|$. Our goal is to minimize the total number of communication with Carole; Alice and Bob do not communicate with each other, but up to factor of 2, that is not a limitation. Formally, let $b_A(t)$ be the total number of bits sent between Alice and Carole up to time t and let $b_B(t)$ be the same for Bob. We wish to design a communication protocol that minimizes $b(t) = b_A(t) + b_B(t)$ at time t while guaranteeing that Carole continually has the correct value of $C(t)$. As stated, it is easy to see that all Alice or Bob can do is to send a bit whenever they each see a new item, and hence, $b(t) = |A(t)| + |B(t)|$ trivially. Of more interest is a relaxed version of the problem where, given ϵ, Carole's new task is to output 0 whenever $C(t) \leq (1-\epsilon)\tau$ and must output 1 when $C(t) > \tau$ for a threshold τ. Now the problem is nontrivial. For example, here are some (randomized) communication procedures:

- [COIN TOSS] Alice and Bob each flip a coin (possibly biased) upon the arrival of an item and send Carole one bit whenever the coin turns up heads.
- [GLOBAL] Alice and Bob know a rough estimate of $\Delta = \tau - C(t')$ from some prior time t', and each send a bit whenever the number of items they have received exceeds $\Delta/2$. Carole updates Alice and Bob with estimates when she gets a bit update and the new value of Δ is computed and used.
- [LOCAL] Alice and Bob each create a model for arrival times of items and communicate the model parameters to Carole; they send bits to summarize differences when their current data significantly differs from their models. If the sources are compressible, this can yield savings.

What is the (expected) performance of these procedures, and what is the optimal bound on (expected) $b(t)$?

We study such functional monitoring problems more generally in which (a) there are $k \geq 2$ sites, (b) we wish to monitor $C(t) = f(A_1(t) \cup \cdots \cup A_k(t))$ where $A_i(t)$ is the multiset of items collected at site i by time t, and f is a monotonically nondecreasing function in time. There are two variants: *threshold monitoring* (determining when $C(t)$ exceeds a threshold τ) and *value monitoring* (providing a good approximation to $C(t)$ at all times t). Value

*Supported in part by Hong Kong Direct Allocation Grant (DAG07/08).

monitoring directly solves threshold monitoring, and running $O(\frac{1}{\epsilon}\log T)$ instances of a threshold monitoring algorithm for thresholds $\tau = 1, (1+\epsilon), (1+\epsilon)^2, \ldots, T$ solves value monitoring with relative error $1+\epsilon$. So the two variants differ by at most a factor of $O(\frac{1}{\epsilon}\log T)$. In many applications, the threshold version is more important, and so we focus on this case, and we call them (k, f, τ, ϵ) *distributed, functional monitoring problems*. Our interests in these problems come from both applied and foundational concerns.

Applied motivations. (k, f, τ, ϵ) functional monitoring problems arise immediately in a number of distributed monitoring systems, both traditional and modern.

In traditional sensor systems such as smart homes and elsewhere, security sensors are carefully laid out and configured, and there is a convenient power source. The straightforward way to monitor a phenomenon is to take measurements every few time instants, send them to a central site, and use back-end systems to analyze the *entire* data trace. In contrast, more interestingly, modern sensor networks are more ad hoc and mobile: they are distributed arbitrarily and work with battery power [17, 19]. They have to conserve their power for long use between charging periods. Further, these sensors have some memory and computing power. Hence the sensors can perform local computations and be more careful in usage of radio for communication, since radio use is the biggest source of battery drain. In this scenario, collecting *all* the data from sensors to correctly calculate f in the back-end is wasteful, and a direct approach is to design protocols which will trigger an alarm when a threshold is exceeded, and the emphasis is on minimizing the communication during the battery lifetime. This is modeled by (k, f, τ, ϵ) functional monitoring problems.

In this context, variations of (k, f, τ, ϵ) functional monitoring have been proposed as "reactive monitoring" (in networking [11]) and "distributed triggers" (in databases [16]). Prior work has considered many different functions f [2, 5, 7, 8, 10, 11, 14, 16, 22], and typically presents algorithms (often variants of GLOBAL or LOCAL described earlier) with correctness guarantees, but no nontrivial communication bounds. Some of the above work takes a *distributed streaming* approach where in addition to optimizing the bits communicated, the algorithms also optimize the space and time requirements of each of the sensors.

Foundational motivations. There are a number of research areas in computing, communication and signal processing that are related to the class of problems we study here.

In communication theory, there is the problem of collecting signals from multiple sources. The problem is typically formulated as that of collecting the entire set of signals and focus on using the fewest bits that captures the (unknown) complexity of the *stochastic* sources. An example is the classical Slepian-Wolf result that shows that two sources can encode with total cost proportional to the joint entropy without explicit coordination [9]. Extending these results to approximating some function f on the joint sources is an (untapped!) challenge. Further, our study is combinatorial, focusing on worst case signals.

In signal processing, the emerging area of compressed sensing [12] redefines the problem of signal acquisition as that of acquiring not the entire signal, but only the information needed to reconstruct the few salient coefficients using a suitable dictionary. These results can be extended to (k, f, τ, ϵ) problems where f yields the salient coefficients needed to reconstruct the entire signal [21]. Further, [21] extended compressed sensing to *functional compressed sensing* where we need to only acquire information to evaluate *specific* functions of the input signal. Except for preliminary results in [21] for quantiles, virtually no results are known for (k, f, τ, ϵ) problems. Some initial work in this direction uses graph coloring on the characteristic graph of the function f [13].

In computer science, there are communication complexity bounds [24] that minimize the bits needed to compute a given function f of inputs *at any particular time* over k parties. But they do not minimize the bits needed *over* the entire time, continuously. We call them *one-shot* problems. The central issue in the continuous problems that we study here is how often and when to repeat parts of such protocols over time to minimize the overall number of bits.

The streaming model [1] has received much attention in recent years. There are many functions f that can be computed up to $1 \pm \epsilon$ accuracy in streaming model, using $\mathrm{poly}(1/\epsilon, \log n)$ space: this includes streaming algorithms for problems such as estimating frequency moments, clustering, heavy hitters, and so on [20]. There have been several works in the database community that consider the streaming model under the distributed setting, which is essentially the same as the model we study here. Subsequently several functional monitoring problems have been considered in this distributed streaming model [5, 6, 8, 18], but the devised solutions typically are heuristics-based, the worst-case bounds are usually large and far from optimal. In this paper, we give much improved upper bounds for some basic functional monitoring problems, as well as the first lower bounds for these problems.

Our main results and overview. In this paper, we focus on the frequency moments, i.e., $F_p = \sum_i m_i^p$ where m_i is the frequency of item i from all sites. Estimating the frequency moments has become the keystone problem in streaming algorithms since the seminal paper of Alon et al. [1]. In particular, the first three frequency moments ($p = 0, 1, 2$) have received the most attention. F_1 is the simple SUM problem above, F_0 corresponds to the number of distinct elements, and F_2 has found many applications such as surprise index, join sizes, etc.

Moment	Continuous		One-shot	
	Lower bound	Upper bound	Lower bound	Upper bound
F_0, randomized	$\Omega(k)$	$\tilde{O}(\frac{k}{\epsilon^2})$	$\Omega(k)$	$\tilde{O}(\frac{k}{\epsilon^2})$
F_1, deterministic	$\Omega(k \log \frac{1}{\epsilon k})$	$O(k \log \frac{1}{\epsilon})$	$\Omega(k \log \frac{1}{\epsilon k})$	$O(k \log \frac{1}{\epsilon})$
F_1, randomized	$\Omega(\min\{k, \frac{1}{\epsilon}\})$	$O(\min\{k \log \frac{1}{\epsilon}, \frac{1}{\epsilon^2} \log \frac{1}{\delta}\})$	$\Omega(k)$	$O(k \log \frac{1}{\epsilon\sqrt{k}})$
F_2, randomized	$\Omega(k)$	$\tilde{O}(k^2/\epsilon + (\sqrt{k}/\epsilon)^3)$	$\Omega(k)$	$\tilde{O}(\frac{k}{\epsilon^2})$

Table 1: Summary of the communication complexity for one-shot and continuous threshold monitoring of different frequency moments. The "randomized" bounds are expected communication bounds for randomized algorithms with failure probability $\delta < 1/2$.

- For the (k, F_1, τ, ϵ) problem, we show deterministic bounds of $O(k \log 1/\epsilon)$ and $\Omega(k \log \frac{1}{\epsilon k})$[1]; and randomized bounds of $\Omega(\min\{k, \frac{1}{\epsilon}\})$ and $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$, independent of k, where δ is the algorithm's probability of failure. Hence, randomization can give significant asymptotic improvement, and curiously, k is not an inherent factor. These bounds improve the previous result of $O(k/\epsilon \log \tau/k)$ in [18].
- For the (k, F_0, τ, ϵ) problem, we give a (randomized) upper bound of[2] $\tilde{O}(k/\epsilon^2)$, which improves upon the previous result of $O(k^2/\epsilon^3 \log n \log \frac{1}{\delta})$ in [7]. We also give a lower bound of $\Omega(k)$.
- Our main results are for the (k, F_2, τ, ϵ) problem: we present an upper bound of $\tilde{O}(k^2/\epsilon + (\sqrt{k}/\epsilon)^3)$ improving the previous result of $\tilde{O}(k^2/\epsilon^4)$ [5]. We also give an $\Omega(k)$ lower bound. The algorithm is a sophisticated variation of GLOBAL above, with multiple rounds, using different "sketch summaries" at multiple levels of accuracy.

Table 1 summarizes our results. For comparison, we also include the one-shot costs: observe that for F_0 and F_1, the cost of continuous monitoring is no higher than the one-shot computation and close to the lower bounds; only for F_2 is there a clear gap to address.

In addition to the specific results above which are interesting in their own right, they also imply communication-efficient solution to (k, f, τ, ϵ) problems for a number of others f's including histograms, wavelets, clustering, geometric problems, and others. In addition, we believe that the algorithmic approaches behind both the results above will prove to be useful for other (k, f, τ, ϵ) problems.

In this paper, we are mainly interested in the communication cost of the algorithms, and our lower bounds hold even assuming that the remote sites have infinite computing power. Nevertheless, all our algorithms can be implemented with low memory and computing costs at the remote sites and the coordinator.

[1] We use the notation $\log x = \max\{\log_2 x, 1\}$ throughout the paper.

[2] The $\tilde{O}$ notation suppresses logarithmic factors in $n, k, m, \tau, 1/\epsilon, 1/\delta$.

2 Problem Formulation

Let $A = (a_1, \ldots, a_m)$ be a sequence of elements, where $a_i \in \{1, \ldots, n\}$. Let $m_i = |\{j : a_j = i\}|$ be the number of occurrences of i in A, and define the *p-th frequency moment* of A as $F_p(A) = \sum_{i=1}^{n} m_i^p$ for each $p \geq 0$. In the distributed setting, the sequence A is observed in order by $k \geq 2$ remote *sites* $S_1, \ldots, S_k$ collectively, i.e., the element a_i is observed by exactly one of the sites at time instance i. There is a designated *coordinator* that is responsible for deciding if $F_p(A) \geq \tau$ for some given threshold τ. Determining this at a single time instant t yields the class of *one-shot* queries, but here we are more interested in *continuous-monitoring* (k, f, τ, ϵ) queries, where the coordinator must correctly answer over the collection of elements observed thus far $(A(t))$, for all time instants t.

We focus on the approximate version of these problems. For some parameter $0 < \epsilon \leq 1/4$, the coordinator should output 1 to raise an alert if $F_p(A(t)) \geq \tau$; output 0 if $F_p(A(t)) \leq (1 - \epsilon)\tau$; and is allowed either answer in-between. Since the frequency moments never decrease as elements are received, the continuous-monitoring problem can also be interpreted as the problem of deciding a time instance t, at which point we raise an alarm, such that $t_1 \leq t \leq t_2$, where $t_1 = \arg\min_t\{F_p(A(t)) > (1 - \epsilon)\tau\}$ and $t_2 = \arg\min_t\{F_p(A(t)) \geq \tau\}$. The continuous algorithm terminates when such a t is determined.

We assume that the remote sites know the values of τ, ϵ, and n in advance, but not m. The cost of an algorithm is measured by the number of bits that are communicated. We assume that the threshold τ is sufficiently large to simplify analysis and the bounds. Dealing with small τ's is mainly technical: we just need to carefully choose when to use the naïve algorithm that simply sends every single element to the coordinator.

The following simple observation implies that the continuous-monitoring problem is almost always as hard as the corresponding one-shot problem.

PROPOSITION 2.1. *For any monotone function f, an algorithm for (k, f, τ, ϵ) functional monitoring that communicates $g(k, n, m, \tau, \epsilon)$ bits implies a one-shot algorithm that communicates $g(k, n, m, \tau, \epsilon) + O(k)$ bits.*

Proof: The site S_1 first starts running the continuous-

monitoring algorithm on its local stream, while the rest pretend that none of their elements have arrived. When S_1 finishes, it sends a special message to the coordinator, which then signals S_2 to start. We continue this process until all k sites have finished, or an alarm is raised (output changes to 1) in the middle of the process. □

3 General Algorithm for F_p, $p \geq 1$

We first present a general algorithm based on each site monitoring only local updates. This gives initial upper bounds, which we improve for specific cases in subsequent sections.

The algorithm proceeds in multiple rounds, based on the generalized GLOBAL idea. Let u_i be the frequency vector $(m_1, \ldots, m_n)$ at the beginning of round i. In round i, every site keeps a copy of u_i and a threshold t_i. Let v_{ij} be the frequency vector of recent updates received at site j during round i. Whenever the impact of v_{ij} causes the F_p moment locally to increase by more than t_i (or multiples thereof), the site informs the coordinator. After the coordinator has received more than k such indications, it ends the round, collects information about all k vectors v_{ij} from sites, computes a new global state u_{i+1} and distributes it to all sites.

More precisely, we proceed as follows. Define the round threshold $t_i = \frac{1}{2}(\tau - \|u_i\|_p^p)k^{-p}$, chosen to divide the current "slack" uniformly between sites. Each site j receives a set of updates during round i, which we represent as a vector v_{ij}. During round i, whenever $\lfloor \|u_i + v_{ij}\|_p^p / t_i \rfloor$ increases, site j sends a bit to indicate this (if this quantity increases by more than one, the site sends one bit for each increase). After the coordinator has received k bits in total, it ends round i and collects v_{ij} (or some compact summary of v_{ij}) from each site. It computes $u_{i+1} = u_i + \sum_{j=1}^k v_{ij}$, and hence t_{i+1}, and sends these to all sites, beginning round $i+1$. The coordinator changes its output to 1 when $\|u_i\|_p^p \geq (1-\epsilon/2)\tau$, and the algorithm terminates.

THEOREM 3.1. *At the end of round i, we have $\|u_i\|_p^p + kt_i \leq \|u_{i+1}\|_p^p \leq 2k^p t_i + \|u_i\|_p^p$. There can be at most $O(k^{p-1} \log \frac{1}{\epsilon})$ rounds.*

Proof: We first define the function $\psi(x, y) = \|x + y\|_p^p - \|x\|_p^p$. ψ is convex in both its arguments for all $p \geq 1$, in the range where x and y are non-negative (have no negative components). The left hand side is straightforward: each site sends an indication whenever its local F_p moment increases by t_i, i.e. we monitor $\psi(u_i, v_{ij})$. Observe that providing all vectors are non-negative, we have that $\psi(u_i, \sum_{j=1}^k v_{ij}) \geq \sum_{j=1}^k \psi(u_i, v_{ij})$ (this can be seen by analyzing each dimension of each vector in turn). Thus, we have that

$$\|u_{i+1}\|_p^p - \|u_i\|_p^p = \|u_i + \sum_{j=1}^k v_{ij}\|_p^p - \|u_i\|_p^p \geq kt_i.$$

For the right hand side, we have (by Jensen's inequality on the second argument of ψ, and monotonicity on the first argument):

$$\|u_i + \sum_{j=1}^k v_{ij}\|_p^p - \|u_i\|_p^p = \psi(u_i, \sum_{j=1}^k v_{ij})$$
$$\leq \frac{1}{k} \sum_{j=1}^k \psi(ku_i, kv_{ij}) = k^{p-1} \sum_{j=1}^k \psi(u_i, v_{ij})$$
$$= k^{p-1} \sum_{j=1}^k (\|u_i + v_{ij}\|_p^p - \|u_i\|_p^p) < 2k^p t_i.$$

The last bound follows by observing that we see k messages from sites whenever $\|u_i + v_{ij}\|_p^p - \|u_i\|_p^p$ increases by t_i, so the largest this can be is $2kt_i$ (kt_i from changes that have been notified, and up to t_i at each of $k-1$ sites apart from the one that triggers the end of the round).

By our choice of t_i, we ensure that this upper bound on the current global value of F_p never exceeds τ during a round, and we terminate the procedure as soon as it exceeds $(1-\epsilon/2)\tau$. Analyzing the number of rounds, from the lower bound above, we have

$$t_{i+1} = \frac{1}{2}(\tau - \|u_{i+1}\|_p^p)k^{-p} \leq \frac{1}{2}(\tau - \|u_i\|_p^p - kt_i)k^{-p}$$
$$= \frac{1}{2}(2k^{p-1} - 1)t_i k^{1-p}$$

So $t_{i+1}/t_i \leq 1 - k^{1-p}/2 \leq (1 - k^{1-p}/2)^i t_0$. Since $t_0 = \tau k^{-p}/2$, and we terminate when $t_i < \epsilon\tau k^{-p}/4$, it is clear that there can be at most $O(k^{p-1} \log 1/\epsilon)$ rounds before this occurs. □

We now consider various special cases of (k, F_p, τ, ϵ) monitoring depending on the choice of p:

Case 1: $p = 1$. For the case $p = 1$, the above immediate implies a bound of $O(k \log 1/\epsilon)$ messages of counts being exchanged. In fact, we can give a tighter bound: the coordinator can omit the step of collecting the current v_{ij}'s from each site, and instead just sends a message to advance to the next stage. The value of t_i is computed simply as $2^{-1-i}\tau/k$, and the coordinator has to send only a constant number of bits to each site to signal the end of round i. Thus, we obtain a bound of $O(k \log 1/\epsilon)$ *bits*, compared to the $O(k/\epsilon \log \tau/k)$ scheme presented in [18].

Case 2: $p = 2$. When $p = 2$, in order to concisely convey information about the vectors v_{ij} we make use of *sketch* summaries of vectors [1]. These sketches have the property that (with probability at least $1 - \delta$) they allow F_2 of the summarized vector to be estimated with relative error ϵ, in $O(\frac{1}{\epsilon^2} \log \tau \log \frac{1}{\delta})$ bits. We can apply these sketches in the above protocol for $p = 2$, by replacing each instance

of u_i and v_{ij} with a sketch of the corresponding vector. Note that we can easily perform the necessary arithmetic to form a sketch of $u_i + v_{ij}$ and hence find (an estimate of) $\|u_i + v_{ij}\|_2^2$. In order to account for the inaccuracy introduced by the approximate sketches, we must carefully set the error parameter ϵ' of the sketches. Since we compare the change in $\|u_i + v_{ij}\|_2^2$ to t_i, we need the error given by the sketch—which is $\epsilon'\|u_i+v_{ij}\|_2^2$ —to be at most a constant fraction of t_i, which can be as small as $\frac{\epsilon\tau}{2}$. Thus we need to set $\epsilon' = O(\frac{\epsilon}{k^2})$. Putting this all together gives the total communication cost of $\tilde{O}(k^6/\epsilon^2)$.

Case 3: $p > 2$**.** For larger values of p, we can again use sketch-like summaries. This time, we can make use of the data summary structures of Ganguly et al. [4], since these have the necessary summability properties. We omit full details for brevity; the analysis is similar to the $p = 2$ case.

4 Bounds for F_1

To get improved bounds, we start with the easiest case, of monitoring F_1, which is simply the total number of elements observed, i.e., SUM. The analysis in the above section yields a deterministic algorithm for F_1 which communicates $O(k \log \frac{1}{\epsilon})$ bits. This is almost optimal for deterministic algorithms, as indicated by the following lower bound, which actually follows from a reduction from the one-shot case. The proof appears in the full version of the paper.

THEOREM 4.1. *Any deterministic algorithm that solves (k, F_1, τ, ϵ) functional monitoring has to communicate $\Omega(k \log \frac{1}{\epsilon k})$ bits.*

If we allow randomized protocols that may err with certain probability δ, we can design a sampling based algorithm whose complexity is independent of k. This is to be contrasted with the one-shot case, where there is an $\Omega(k)$ lower bound even for randomized algorithms.

THEOREM 4.2. *There is a randomized algorithm for (k, F_1, τ, ϵ) functional monitoring with error probability at most δ that communicates $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ bits.*

Proof: We present a randomized algorithm derived from a careful implementation of COIN TOSS, with error probability $1/3$. By running $O(\log \frac{1}{\delta})$ independent instances and raising an alarm when at least half of the instances have raised alarms, we amplify to success probability $1 - \delta$, as required. Every time a site has received $\epsilon^2\tau/(ck)$ elements, where c is some constant to be determined later, it sends a signal to the coordinator with probability $1/k$. The server raises an alarm as soon as it has received $c/\epsilon^2 - c/(2\epsilon)$ such signals, and terminates the algorithm. The communication bound is immediate. For correctness, it is sufficient to prove the following: On any sequence A', the algorithm fails to output 0 with probability at most 1/6 if $F_1(A') \leq (1 - \epsilon)\tau$, and fails to output 1 with probability at most 1/6 if $F_1(A') \geq \tau$. Then for the given input sequence A, applying this statement on A_{t_1-1} and A_{t_2} proves the theorem (where t_1 and t_2 are as defined in Section 2).

Let X be the number of signals received by the coordinator. Its expectation is at most $\mathrm{E}[X] \leq 1/k \cdot F_1/(\epsilon^2\tau/(ck)) = cF_1/(\epsilon^2\tau)$, and at least $\mathrm{E}[X] \geq 1/k \cdot (F_1 - \epsilon^2\tau)/(\epsilon^2\tau/(ck)) = cF_1/(\epsilon^2\tau) - c$. Its variance is $\mathrm{Var}[X] \leq (ckF_1)/(\epsilon^2\tau) \cdot (1/k - 1/k^2) \leq cF_1/(\epsilon^2\tau)$.

If $F_1 \leq (1 - \epsilon)\tau$, then the probability that the coordinator outputs 1 is (by Chebyshev inequality)

$$\begin{aligned}\Pr[X \geq c/\epsilon^2 - c/(2\epsilon)] &\leq \Pr[X - \mathrm{E}[X] \geq c/(2\epsilon)] \\ &\leq \frac{c(1/\epsilon^2 - 1/\epsilon)}{(c/(2\epsilon))^2} \leq \frac{4}{c}.\end{aligned}$$

Similarly, if $F_1 \geq \tau$, then the probability that the coordinator does not output 1 is

$$\begin{aligned}\Pr[X \leq c/\epsilon^2 - c/(2\epsilon)] &\leq \Pr[X - \mathrm{E}[X] \leq -c/(2\epsilon) + c] \\ &\leq \frac{c/\epsilon^2}{(-c/(2\epsilon) + c)^2} \leq \frac{1}{c(1/2 - \epsilon)^2} \leq \frac{16}{c}.\end{aligned}$$

Choosing $c = 96$ makes both probabilities at most 1/6, as desired. □

Therefore, the randomized algorithm is better than the deterministic algorithm for large enough ϵ. Combined with the deterministic bound, we obtain the bound in Table 1. In addition, we also have the following lower bound (proof appears in the full version of the paper):

THEOREM 4.3. *For any $\epsilon < 1/4$, any probabilistic protocol for (k, F_1, τ, ϵ) functional monitoring that errs with probability smaller than 1/2 has to communicate $\Omega(\min\{k, 1/\epsilon\})$ bits in expectation.*

5 Bounds for F_0

We know that the F_1 problem can be solved deterministically and exactly (by setting $\epsilon = 1/\tau$) by communicating $O(k \log \tau)$ bits. For any $p \neq 1$, the same arguments of Proposition 3.7 and 3.8 in [1] apply to show that both randomness (Monte Carlo) and approximation are necessary for the F_p problem in order to get solutions with communication cost better than $\Omega(n)$ for any $k \geq 2$. So for the rest of the paper we only consider probabilistic protocols that err with some probability δ.

For monitoring F_0, we can generalize the sketch of [3] in a distributed fashion, leading to the following result, which improves upon the previous bound of $O(k^2/\epsilon^3 \log n \log \frac{1}{\delta})$ in [7]. The basic idea is that, since the F_0 sketch changes "monotonically", i.e., once an entry is added, it will never be removed, we can communicate to the coordinator every addition to all the sketches maintained by the individual sites.

THEOREM 5.1. *There is a randomized algorithm for the* (k, F_0, τ, ϵ) *functional monitoring problem with error probability at most* δ *that communicates* $O(k(\log n + \frac{1}{\epsilon^2}\log\frac{1}{\epsilon})\log\frac{1}{\delta})$ *bits.*

Proof: Below we present an algorithm with error probability $1/3$. Again, it can be driven down to δ by running $O(\log\frac{1}{\delta})$ independent copies of the algorithm.

Define t as the integer such that $48/\epsilon^2 \leq \tau/2^t < 96/\epsilon^2$. The coordinator first picks two random pairwise independent hash functions $f : [n] \to [n]$ and $g : [n] \to [6 \cdot (96/\epsilon^2)^2]$, and send them to all the remote sites. This incurs a communication cost of $O(k(\log n + \log\frac{1}{\epsilon})) = O(k \log n)$ bits. Next, each of the remote sites evaluates $f(a_i)$ for every incoming element a_i, and tests if the last t bits of $f(a_i)$ are all zeros. If so it evaluates $g(a_i)$. There is a local buffer that contains all the $g()$ values for such elements. If $g(a_i)$ is not in the buffer, we add $g(a_i)$ into the buffer, and then send it to the coordinator. The coordinator also keeps a buffer of all the unique $g()$ values it has received, and outputs 1 whenever the number of elements in the buffer exceeds $(1-\epsilon/2)\tau/2^t$. Since each $g()$ value takes $O(\log\frac{1}{\epsilon})$ bits, the bound in the theorem easily follows. We prove the correctness of the algorithm below.

It is sufficient to prove the following: On any sequence A', the algorithm outputs 1 with probability at most 1/6 if $F_0(A') \leq (1-\epsilon)\tau$, and outputs 0 with probability at most 1/6 if $F_0(A') \geq \tau$.

One source of error is g having collisions. Since g is evaluated on at most $96/\epsilon^2$ elements, the probability that g has collisions is at most $1/12$. From now on we assume that g has no collisions, and will add $1/12$ to the final error probability.

Let X be the number of distinct elements in A' that have zeros in their last t bits of the $f()$ value. We know [3] that $\mathrm{E}[X] = F_0/2^t$ and $\mathrm{Var}[X] \leq F_0/2^t$.

If $F_0 \leq (1-\epsilon)\tau$, then the algorithm outputs 1 with probability

$$\begin{aligned}\Pr[X > (1-\epsilon/2)\tau/2^t] &\leq \Pr[X - \mathrm{E}[X] > \epsilon\tau/2^{t+1}] \\ &\leq \frac{4\cdot\mathrm{Var}[X]}{(\epsilon\tau/2^t)^2} \leq \frac{4F_0/2^t}{(\epsilon\tau/2^t)^2} \leq \frac{4F_0}{\epsilon^2\tau\cdot 48/\epsilon^2} \leq \frac{1}{12}.\end{aligned}$$

When F_0 reaches τ, the probability of outputting 0 is

$$\begin{aligned}\Pr[X \leq (1-\epsilon/2)\tau/2^t] &\leq \Pr[X - \mathrm{E}[X] \leq -\epsilon\tau/2^{t+1}] \\ &\leq \frac{4\cdot\mathrm{Var}[X]}{(\epsilon\tau/2^t)^2} \leq \frac{1}{12}.\end{aligned}$$

Thus, the total error probability in either case is at most 1/6, as desired. □

Unlike the F_1 case where there is a randomized algorithm whose communication complexity is independent of k, we show below that this is not the case for F_0. To obtain a lower bound for randomized algorithms we invoke Yao's Minimax Principle [23], which requires us to construct a probability distribution on the inputs, and show that any deterministic algorithm has to communicate a certain number of bits in expectation (w.r.t the distribution of the inputs). For this purpose we cast any deterministic continuous-monitoring algorithm in the following model. Each remote site S_i maintains a set of an arbitrary number of *triggering conditions*. Each triggering condition is a frequency vector $(m_1, \ldots, m_n) \in [m]^n$. The site S_i will conduct some communication when and only when the frequency vector of the elements it has received so far is one triggering condition. The communication may in turn lead to communication between the coordinator and other remote sites. After all the communication is completed, those sites that have communicated with the coordinator are allowed to change their sets of triggering conditions arbitrarily. This is a powerful model, as the communication is arbitrary when a triggering condition is met. However, note that on the other hand, no communication is allowed if none of the triggering conditions is reached. We will use this fact to show that the constructed inputs will trigger communication at least $\Omega(k)$ times. Another implicit assumption in this model is that only the current state matters but not *how* the state is reached. For instance if $(1, 1, 0, \ldots, 0)$ is a trigger condition, the site will trigger communication no matter if a "1" is observed before a "2" and vice versa. However, this assumption is not an issue in our proof, as in our construction of the inputs, there is at most one way to reach any state vector.

THEOREM 5.2. *For any* $\epsilon \leq 1/4$, $n \geq k^2$, *any probabilistic protocol for* (k, F_0, τ, ϵ) *functional monitoring that errs with probability smaller than 1/2 has to communicate* $\Omega(k)$ *bits in expectation.*

Proof: Following the Minimax Principle [23], it suffices to demonstrate a probability distribution on the inputs, and show that any deterministic algorithm that errs with probability at most 1/8 has to communicate expected $\Omega(k)$ bits.

For simplicity, we will use $\tau = k$ in the proof. Similar constructions work for larger τ's. The inputs are constructed as follows. We first pick an integer r between 1 and $k/2$ uniformly at random. We then proceed in r rounds. In the first round, we randomly pick an element from $\{1, \ldots, k\}$ and send it to all the sites; the order is irrelevant (for concreteness, say in the order $S_1, \ldots, S_k$). In the second round, we do the same thing except that the element is now chosen from $\{k+1, \ldots, 2k\}$. We continue this process until in the r-the round, we uniformly randomly send a different element from $\{(r-1)k+1, \ldots, rk\}$ to each of the k sites. We denote by I_r the set of inputs that end in r rounds. It can be easily verified that for any input in I_r, the algorithm can correctly terminate during and only during the

r-th round. It is helpful to think of the input construction as follows. At first, with probability $p = \frac{1}{k/2}$, we (a) pick a different element randomly and send it to each of the k sites; otherwise, we (b) pick one random element and send it to all the sites. In case (a) we terminate the construction, and in case (b), we proceed to the next round. In the second round, we do the same except that the probability of choosing case (a) is $p = \frac{1}{k/2-1}$. We continue this process in this fashion for a maximum of $k/2$ rounds, using $p = \frac{1}{k/2-i+1}$ in the i-th round.

Since the algorithm is correct with probability at least 7/8, there are $s \geq k/4$ values of r: $r_1 \leq r_2 \leq \cdots \leq r_s$, such that the algorithm is correct with probability at least 3/4 within I_{r_j} for each of $j = 1, \ldots, s$. Note that for any deterministic algorithm, these r_j's are fixed. For any $1 \leq j \leq s-1$, consider the triggering conditions just before the r_j'th round. Note that these triggering conditions may depend on the elements received in the first $r_j - 1$ rounds. So let us consider a particular history H of the first $r_j - 1$ rounds in which case (b) is always chosen. There are k^{r_j-1} such histories, and each happens with equal probability. Let $z_{i,\ell} = 1$ if S_i will trigger communication when the next element it receives is ℓ, and $z_{i,\ell} = 0$ otherwise. We claim that for at least half of these histories, the following condition must hold.

$$\sum_{i=1}^{k} \sum_{\ell=(r_j-1)k+1}^{r_j k} z_{i,\ell} \geq \frac{k}{2}. \tag{5.1}$$

Indeed, we will show in the following that if (5.1) does not hold for a history H, then conditioned on the input being in I_{r_j} and having H as its history, the probability that the algorithm errs is at least 1/2. If this were the case for more than half of the histories, then the error probability would be more than 1/4 for I_{r_j}, contradicting the previous assumption.

To prove that if (5.1) does not hold for H, the algorithm is very likely to fail in the next round if $r = r_j$, consider a random input in I_{r_j} with history H. Recall that a randomly selected element from $\{(r_j - 1)k + 1, \ldots, r_j k\}$ is given to each of the k sites. The coordinator can output 1 only if some site triggers communication, whose probability is at most (by the union bound)

$$\sum_{i=1}^{k} \left(\frac{\sum_{\ell=(r_j-1)k+1}^{r_j k} z_{i,\ell}}{k} \right) = \frac{1}{2}.$$

Therefore we conclude that for any r_j, (5.1) must hold for at least half of its histories. Now consider the case that the input π belongs to some I_r such that $r > r_j$. This happens with probability $1 - r_j/(k/2)$. We next compute the expected number of messages that π triggers in the r_j-th round. Suppose that (5.1) holds and π sends ℓ to all the sites. Note that $\sum_{i=1}^{k} z_{i,\ell}$ sites will be triggered, unless they receive a message from the coordinator telling them to change their triggering conditions. So at least $\sum_{i=1}^{k} z_{i,\ell}$ messages need to be transmitted. Thus, the expected number of messages that π triggers in the r_j-th round is

$$\frac{1}{2} \cdot \sum_{\ell=(r_j-1)k+1}^{r_j k} \left(\frac{1}{k} \cdot \sum_{i=1}^{k} z_{i,\ell} \right) \geq \frac{1}{4}. \tag{5.2}$$

Summing up (5.2) over all r_j, the total expected number of messages is at least $\sum_{j=1}^{s} \left(1 - \frac{r_j}{k/2}\right) \cdot \frac{1}{4} = \Omega(k)$. □

6 Bounds for F_2

In the following, we present an F_2 monitoring algorithm that combines the multi-round framework of our general monitoring algorithm and the AMS sketch [1], giving a total communication cost of $\tilde{O}(k^2/\epsilon + k^{3/2}/\epsilon^3)$. This strictly improves the bound which follows from prior work, of $O(k^2/\epsilon^4)$ [5]. Our algorithm consists of two phases. At the end of the first phase, we make sure that the F_2 is between $\frac{3}{4}\tau$ and τ; while in the second phase, we more carefully monitor F_2 until it is in the range $((1-\epsilon)\tau, \tau)$. Each phase is divided into multiple rounds. In the second phase, each round is further divided into multiple sub-rounds to allow for more careful monitoring with minimal communication. We use sketches such that with probability at least $1 - \delta$, they estimate F_2 of the sketched vector within $1 \pm \epsilon$ using $O(\frac{1}{\epsilon^2} \log n \log \frac{1}{\delta})$ bits [1]. For now, we assume that all sketch estimates are within their approximation guarantees; later we discuss how to set δ to ensure small probability of failure over the entire computation.

Algorithm. We proceed in multiple rounds, which are in turn divided into subrounds. Let u_i be the frequency vector of the union of the streams at the beginning of the ith round, and $\hat{u}_i^2$ be an approximation of u_i^2. In round i, we use a local threshold $t_i = \frac{(\tau - \hat{u}_i^2)^2}{64k^2\tau}$. Let $v_{ij\ell}$ be the local frequency vector of updates received at site j during subround ℓ of round i, and let $w_{i\ell} = \sum_{j=1}^{k} v_{ij\ell}$ be the total increment of the frequency vectors in subround ℓ of round i. During each (sub)round, each site j continuously monitors its $v_{ij\ell}^2$, and sends a bit to the server whenever $\lfloor v_{ij\ell}^2/t_i \rfloor$ increases.

Phase one. In phase one, there is only one subround per round. At the beginning of round i, the server computes a $\frac{5}{4}$-overestimate $\hat{u}_i^2$ of the current u_i^2, i.e., $u_i^2 \leq \hat{u}_i^2 \leq \frac{5}{4}u_i^2$. This can be done by collecting sketches from all sites with a communication cost of $O(k \log n)$. Initially $\hat{u}_1^2 = u_1^2 = 0$. When the server has received k bits in total from sites, it ends the round by computing a new estimate $\hat{u}_{i+1}^2$ for u_{i+1}^2. If $\hat{u}_{i+1}^2 \geq \frac{15}{16}\tau$, then we must have $u_{i+1}^2 \geq \hat{u}_{i+1}^2/\frac{5}{4} \geq \frac{3}{4}\tau$, so we proceed to the second phase. Otherwise the server computes the new t_{i+1}, broadcasts it to all sites, and proceeds to the next round of phase one.

Analysis of phase one. The following lemma guarantees that the algorithm will never need to terminate during phase one.

LEMMA 6.1. *At the end of round i in phase one, $u_{i+1}^2 < \tau$.*

Proof: Assuming pessimistically that all sites are just below the threshold of sending the next bit, once the server has received k bits, by the Cauchy-Schwartz inequality, we have $w_{i\ell}^2 = (\sum_{j=1}^k v_{ij\ell})^2 \le k\sum_{j=1}^k v_{ij\ell}^2 < 2k^2 t_i$. Therefore,

$$\begin{aligned} u_{i+1}^2 &= (u_i + w_{i\ell})^2 = u_i^2 + 2u_i w_{i\ell} + w_{i\ell}^2 \\ &\le u_i^2 + 2\|u_i\| \cdot \|w_{i\ell}\| + w_{i\ell}^2 \\ &< u_i^2 + 2\|u_i\|\sqrt{2k^2 t_i} + 2k^2 t_i \\ &\le u_i^2 + \frac{\sqrt{2}}{4}\|u_i\|\frac{\tau - \hat{u}_i^2}{\sqrt{\tau}} + \frac{(\tau - \hat{u}_i^2)^2}{32\tau} \\ &\le u_i^2 + \frac{\sqrt{2}}{4}\|u_i\|\frac{\tau - u_i^2}{\sqrt{\tau}} + \frac{(\tau - u_i^2)^2}{32\tau} \\ &= u_i^2 + \left(\frac{\sqrt{2}\|u_i\|}{4\sqrt{\tau}} + \frac{1}{32} - \frac{u_i^2}{32\tau}\right)(\tau - u_i^2). \end{aligned}$$

Since $\frac{\|u_i\|}{\sqrt{\tau}} \le 1$, $\left(-\frac{u_i^2}{32\tau} + \frac{\sqrt{2}\|u_i\|}{4\sqrt{\tau}} + \frac{1}{32}\right)$ is always less than 1, and we have $u_{i+1}^2 < \tau$. □

The communication cost in each round is $O(k \log n)$ bits, and we bound the number of rounds:

LEMMA 6.2. *There are $O(k)$ rounds in phase one.*

Proof: We can bound the number of rounds by showing that sufficient progress can be made in each round. In each round, we know $w_{i\ell}^2 = (\sum_{j=1}^k v_{ij\ell})^2 \ge \sum_{j=1}^k v_{ij\ell}^2 \ge kt_i$, thus

$$\begin{aligned} u_{i+1}^2 = (u_i + w_{i\ell})^2 &\ge u_i^2 + w_{i\ell}^2 \ge u_i^2 + kt_i \\ &= u_i^2 + \frac{(\tau - \hat{u}_i)^2}{64k\tau} \ge u_i^2 + \frac{(\tau - \frac{15}{16}\tau)^2}{64k\tau} \\ &= u_i^2 + \Theta(\tau/k). \end{aligned}$$

So the total number of rounds in this phase is $O(k)$. □

The communication cost of phase one is thus bound by $O(k^2 \log n)$. It would be possible to continue the first phase by using more accurate estimates $\hat{u}_i^2$ until u_i^2 reaches $(1-\epsilon)\tau$, but this would result in a communication cost of $\tilde{O}(k^2/\epsilon^3)$. Instead, the use of subrounds in the second phase gives an improved bound.

Phase two. In the second phase, the server computes a $(1+\epsilon/3)$-overestimate $\hat{u}_i^2$ at the start of each round by collecting sketches from the sites with a communication cost of $O(k/\epsilon^2 \log n)$. The server keeps an upper bound $\hat{u}_{i,\ell}^2$ on $u_{i,\ell}^2$, the frequency vector at the beginning of the ℓ-th sub-round in round i.

As above, during each sub-round, each site j continuously monitors its $v_{ij\ell}^2$, and sends a bit to the server whenever $\lfloor v_{ij\ell}^2/t_i \rfloor$ increases. When the server has collected k bits in total, it ends the sub-round. Then, it asks each site j to send a $(1 \pm \frac{1}{2})$-approximate sketch for $v_{ij\ell}^2$. The server computes an estimate $\tilde{w}_{i\ell}^2$ for $w_{i\ell}^2$ by combining these sketches. Note that $\tilde{w}_{i\ell}^2 \in (1 \pm \frac{1}{2})w_{i\ell}^2$. The server computes the new upper bound $\hat{u}_{i,\ell+1}^2$ for $u_{i,\ell+1}^2$ as

$$\hat{u}_{i,\ell+1}^2 = \hat{u}_{i,\ell}^2 + 2\sqrt{2}\|\hat{u}_{i,\ell}\| \cdot \|\tilde{w}_{i\ell}\| + 2\tilde{w}_{i\ell}^2. \tag{6.3}$$

Indeed, since

$$u_{i,\ell+1}^2 = (u_{i,\ell} + w_{i\ell})^2 \le u_{i,\ell}^2 + 2\|u_{i,\ell}\| \cdot \|w_{i\ell}\| + w_{i\ell}^2,$$

and $u_{i,\ell}^2 \le \hat{u}_{i,\ell}^2$, $w_{i\ell}^2 \le 2\tilde{w}_{i\ell}^2$, we have $u_{i,\ell+1}^2 \le \hat{u}_{i,\ell+1}^2$. Then the server checks if

$$\hat{u}_{i,\ell+1}^2 + 3k\|\hat{u}_{i,\ell+1}\|\sqrt{t_i} < \tau. \tag{6.4}$$

If (6.4) holds, the server starts sub-round $\ell+1$. The local threshold t_i remains the same. If (6.4) does not hold, the whole round ends, and the server computes a new $\hat{u}_{i+1}^2$ for u_{i+1}^2. If $\hat{u}_{i+1}^2 \ge (1 - \frac{2}{3}\epsilon)\tau$, the server changes its output to 1 and terminates the algorithm. Otherwise, it computes the new t_{i+1}, sends it to all sites, and starts the next round.

Analysis of phase two. Below we assume $\epsilon < \frac{1}{4}$. We first prove correctness. The second phase of the algorithm never raises a false alarm, since if $\hat{u}_{i+1}^2 \ge (1 - \frac{2}{3}\epsilon)\tau$, then $u_{i+1}^2 \ge \hat{u}_{i+1}^2/(1+\epsilon/3) > (1-\epsilon)\tau$. The following lemma implies that the algorithm will never miss an alarm either.

LEMMA 6.3. *For any round i, at the end of the ℓ-th sub-round, $u_{i,\ell+1}^2 < \tau$.*

Proof: Since the algorithm did not terminate at the end of the $(\ell-1)$-th sub-round, by the condition of (6.4) we have $\hat{u}_{i,\ell}^2 + 3k\|\hat{u}_{i,\ell}\|\sqrt{t_i} < \tau$. At the end of the ℓ-th sub-round when the server has collected k bits, assuming pessimistically that all sites are just below the threshold of sending the next bit, by the Cauchy-Schwartz inequality, we have $w_{i\ell}^2 = (\sum_{j=1}^k v_{ij\ell})^2 \le k\sum_{j=1}^k v_{ij\ell}^2 \le 2k^2 t_i$. Since

$$2k^2 t_i = \frac{2(\tau - \hat{u}_i^2)^2}{64\tau} \le \frac{1}{128}(\tau - \hat{u}_i^2),$$

$$\text{and} \quad k\|u_{i,\ell}\|\sqrt{t_i} = \|u_{i,\ell}\|\frac{\tau - \hat{u}_i^2}{8\sqrt{\tau}} \ge \frac{\sqrt{3}}{16}(\tau - \hat{u}_i^2),$$

we have $2k^2 t_i \le \frac{1}{8\sqrt{3}}k\|u_{i,\ell}\|\sqrt{t_i}$. Thus,

$$\begin{aligned} u_{i,\ell+1}^2 = (u_{i,\ell} + w_{i\ell})^2 &\le u_{i,\ell}^2 + 2\|u_{i,\ell}\| \cdot \|w_{i\ell}\| + w_{i\ell}^2 \\ &\le u_{i,\ell}^2 + 2\|u_{i,\ell}\|\sqrt{2k^2 t_i} + 2k^2 t_i \\ &\le u_{i,\ell}^2 + (2\sqrt{2} + \frac{1}{8\sqrt{3}})k\|u_{i,\ell}\|\sqrt{t_i} \\ &< \hat{u}_{i,\ell}^2 + 3k\|\hat{u}_{i,\ell}\|\sqrt{t_i} < \tau. \end{aligned}$$

□

Now we proceed to the analysis of the algorithm's communication complexity. It is clear that the cost of a sub-round is $O(k \log n)$ bits, since each $(1 \pm \frac{1}{2})$-approximate sketch for $v_{ij\ell}$ has $O(\log n)$ bits. Apart from the sub-round communication cost, each round has an additional $O(\frac{k}{\epsilon^2} \log n)$ cost to compute $\hat{u}_i$. All the other costs, e.g., the bits signaling the start and end of a sub-round, broadcasting t_i, etc., are asymptotically dominated by these costs. Therefore, the problem reduces to bounding the number of rounds and sub-rounds.

LEMMA 6.4. *In any round, the number of sub-rounds is* $O(\sqrt{k})$.

Proof: At the end of the ℓ-th sub-round, the server has received k bits, so $w_{i\ell}^2 \geq \sum_{j=1}^{k} v_{ij\ell}^2 \geq kt_i$. Since $\tilde{w}_{i\ell}^2$ is a $(1 \pm \frac{1}{2})$-estimate of $w_{i\ell}^2$, we have $\tilde{w}_{i\ell}^2 \geq \frac{1}{2} w_{i\ell}^2 \geq kt_i/2$. According to (6.3),

$$\begin{aligned}
\hat{u}_{i,\ell+1}^2 &\geq \hat{u}_{i,\ell}^2 + 2\sqrt{2}\|\hat{u}_{i,\ell}\| \cdot \|\tilde{w}\| \\
&\geq \hat{u}_{i,\ell}^2 + 2\sqrt{2}\|\hat{u}_{i,\ell}\| \cdot \sqrt{\frac{kt_i}{2}} = \hat{u}_{i,\ell}^2 + \frac{1}{4}\|\hat{u}_{i,\ell}\| \cdot \frac{\tau - \hat{u}_i^2}{\sqrt{k\tau}} \\
&\geq \hat{u}_{i,\ell}^2 + \frac{1}{4} \cdot \frac{\sqrt{3}}{2} \cdot \frac{1}{\sqrt{k}}(\tau - \hat{u}_i^2) = \hat{u}_{i,\ell}^2 + \frac{\sqrt{3}}{8\sqrt{k}}(\tau - \hat{u}_i^2).
\end{aligned}$$

For any ℓ, if the ℓ-th sub-round starts, by (6.4) we have $\hat{u}_{i,\ell}^2 + 3k\|\hat{u}_{i,\ell}\|\sqrt{t_i} < \tau$, or

$$\tau > \hat{u}_{i,\ell}^2 + \frac{3}{8} \cdot \|\hat{u}_{i,\ell}\| \frac{\tau - \hat{u}_i^2}{\sqrt{\tau}} > \hat{u}_{i,\ell}^2 + \frac{3}{8} \cdot \frac{\sqrt{3}}{2}(\tau - \hat{u}_i^2).$$

$$\text{Rearranging,} \quad \hat{u}_{i,\ell}^2 < \tau - \frac{3\sqrt{3}}{16}(\tau - \hat{u}_i^2).$$

As $\hat{u}_{i,1}^2 = \hat{u}_i^2$, there are at most $\left(\tau - \frac{3\sqrt{3}}{16}(\tau - \hat{u}_i^2) - \hat{u}_i^2\right) / \left(\frac{\sqrt{3}}{8\sqrt{k}} \cdot (\tau - \hat{u}_i^2)\right) < 4\sqrt{k}$ sub-rounds in phase two. □

LEMMA 6.5. *The total number of rounds is* $O(\sqrt{k}/\epsilon)$.

Proof: Focus on one round, say round i. Suppose there are $s < 4\sqrt{k}$ sub-rounds in this round. For any ℓ, we have $w_{i\ell}^2 < \tau/4$; else the subround would have ended earlier. So $\tilde{w}_{i\ell}^2 < 3\tau/8$. We first show how the upper bound $\hat{u}_{i,\ell}$ increases in each sub-round. From (6.3), $\hat{u}_{i,\ell+1}^2$ is at most

$$\hat{u}_{i,\ell}^2 + 2\sqrt{2}\sqrt{\tau} \cdot \|\tilde{w}_{i\ell}\| + 2\sqrt{3\tau/8} \cdot \|\tilde{w}_{i\ell}\| < \hat{u}_{i,\ell}^2 + 5\sqrt{\tau} \cdot \|\tilde{w}_{i\ell}\|,$$

$$\text{so} \quad \hat{u}_{i,s+1}^2 \leq \hat{u}_{i,1}^2 + 5\sqrt{\tau} \textstyle\sum_{\ell=1}^{s} \|\tilde{w}_{i\ell}\|. \tag{6.5}$$

We know that $\hat{u}_{i,s+1}$ violates (6.4), so

$$\begin{aligned}
\tau &\leq \hat{u}_{i,s+1}^2 + 3k\|\hat{u}_{i,s+1}\|\sqrt{t_i} \leq \hat{u}_{i,s+1}^2 + \frac{3}{8}\|\hat{u}_{i,s+1}\| \frac{\tau - \hat{u}_i^2}{\tau} \\
&< \hat{u}_{i,s+1}^2 + \frac{3}{8}(\tau - \hat{u}_i^2).
\end{aligned}$$

Substituting into (6.5), together with $\hat{u}_{i,1} \leq (1 + \epsilon/3)u_i^2$, we have

$$\textstyle\sum_{\ell=1}^{s} \|\tilde{w}_{i\ell}\| > \frac{\tau - \frac{3}{8}(\tau - u_i^2) - (1 + \frac{1}{3}\epsilon)u_i^2}{5\sqrt{\tau}} = \frac{1}{8} \cdot \frac{\tau - (1 + \frac{8}{15}\epsilon)u_i^2}{\sqrt{\tau}}. \tag{6.6}$$

Next, we lower bound $u_{i+1}^2 = u_{i,s+1}^2$, to show that we must have made progress by the end of this round. Since $u_{i,\ell+1}^2 = (u_{i,\ell} + w_{i\ell})^2 \geq u_{i,\ell}^2 + w_{i\ell}^2$, we have

$$\begin{aligned}
u_{i+1}^2 &\geq u_i^2 + \sum_{\ell=1}^{s} w_{i\ell}^2 \\
&\geq u_i^2 + \frac{1}{2}\sum_{\ell=1}^{s} \tilde{w}_{i\ell}^2 \geq u_i^2 + \frac{1}{2s}(\sum_{\ell=1}^{s} \|\tilde{w}_{i\ell}\|)^2 && \text{(C-S ineq)} \\
&> u_i^2 + \frac{1}{64s\tau}(\tau - (1 + \frac{8}{15}\epsilon)u_i^2)^2 && \text{(by (6.6))} \\
&> u_i^2 + \frac{1}{256\sqrt{k} \cdot \tau}(\tau - (1 + \frac{8}{15}\epsilon)u_i^2)^2 && \text{(Lemma 6.4)}
\end{aligned}$$

Initially we have $u_1^2 \geq \frac{3}{4}\tau$, and the algorithm terminates as soon as u_i^2 exceeds $(1 - \frac{2}{3}\epsilon)\tau$. For $a = 3, 4, \ldots, \log \frac{3}{2\epsilon}$ (assuming w.l.o.g. that $\frac{3}{2\epsilon}$ is a power of 2), we bound the number of rounds for u_i^2 to increase from $(1 - 2^{-a+1})\tau$ to $(1 - 2^{-a})\tau$, as:

$$\begin{aligned}
&\frac{\tau(1 - 2^{-a} - (1 - 2^{-a+1}))}{\frac{1}{256\sqrt{k} \cdot \tau}(\tau - (1 + \frac{8}{15}\epsilon)(1 - 2^{-a})\tau)} + 1 \\
&< \frac{2^{-a}\tau}{\frac{1}{256\sqrt{k} \cdot \tau}(\frac{2^{-a}}{5}\tau)^2} + 1 = 2^a 25^2 \cdot 256\sqrt{k} + 1.
\end{aligned}$$

Summing over all a, we obtain that the total number of rounds is $O(\sqrt{k}/\epsilon)$. □

Combining Lemma 6.4 and 6.5, we know that there are a total of $O(k/\epsilon)$ sub-rounds and $O(\sqrt{k}/\epsilon)$ rounds. Thus phase two incurs a communication of $O((k^2/\epsilon + k^{3/2}/\epsilon^3) \log n)$. Recall that the cost of phase one is $O(k^2 \log n)$. So far we have assumed that all the estimates are always within the claimed approximation ranges. Since we have in total computed $O(\text{poly}(k/\epsilon))$ estimates, by running $O(\log \frac{k}{\epsilon\delta})$ independent repetitions and taking the median for each estimate, we can guarantee an overall error probability of no more than δ by the union bound. Thus, we conclude

THEOREM 6.1. *The* (k, F_2, τ, ϵ) *functional monitoring problem can be solved by an algorithm with a communication cost of* $O((k^2/\epsilon + k^{3/2}/\epsilon^3) \log n \log \frac{k}{\epsilon\delta})$ *bits and succeeds with probability at least* $1 - \delta$.

F_2 lower bound. Similar to the F_0 case, we prove an $\Omega(k)$ lower bound for continuously monitoring F_2 (proof given in the full version of the paper):

THEOREM 6.2. *For any $\epsilon \leq 1/4$, $n \geq k^2$, any probabilistic protocol for (k, F_2, τ, ϵ) functional monitoring that errs with probability smaller than 1/2 has to communicate $\Omega(k)$ bits in expectation.*

7 Conclusion and Open Problems

For functional monitoring problems (k, f, τ, ϵ), we observe the surprising results that for some functions, the communication cost is close to or the same as the cost for one-time computation of f, and that the cost can be less than the number of participants, k. Our results for F_2 make careful use of compact sketch summaries, switching between different levels of approximation quality to minimize the overall cost. These algorithms are more generally useful, since they immediately apply to monitoring L_2 and L_2^2 of arbitrary non-negative vectors, which is at the heart of many practical computations such as join size, wavelet and histogram representations, geometric problems and so on [5, 15]. Likewise, our F_1 techniques are applicable to continuously track quantiles and heavy hitters of time-varying distributions [6].

It remains to close the gap in the F_2 case: can a better lower bound than $\Omega(k)$ be shown, or do there exist $\tilde{O}(k \cdot \mathrm{poly}(1/\epsilon))$ solutions? For other functions, including non-linear functions such as entropy, rolling average, information gain and variance [22], one can progress on each function in turn, but it will be more rewarding to find more general techniques for showing bounds for appropriate classes of functions, based on the techniques shown here. Designing (k, f, τ, ϵ) functional monitoring algorithms for non-monotonic fucntions require new performance measures in order to give meaningful analytic communication bounds. Model variants need to be understood, for example, the difference between one-way and two-way communication from sites to coordinators, and the power of having a broadcast channel between coordinator and sites. Ultimately, this study may lead to a new theory of *continuous* communication complexity.

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] B. Babcock and C. Olston. Distributed top-k monitoring. In *ACM SIGMOD Intl. Conf. Management of Data*, 2003.

[3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *RANDOM*, 2002.

[4] L. Bhuvanagiri, S. Ganguly, D. Kesh, and C. Saha. Simpler algorithm for estimating frequency moments of data streams. In *ACM-SIAM Symp. on Discrete Algorithms*, 2006.

[5] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *Intl. Conf. Very Large Data Bases*, 2005.

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

[7] G. Cormode, S. Muthukrishnan, and W. Zhuang. What's different: Distributed, continuous monitoring of duplicate resilient aggregates on data streams. In *Intl. Conf. on Data Engineering*, 2006.

[8] G. Cormode, S. Muthukrishnan, and W. Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *Intl. Conf. on Data Engineering*, 2007.

[9] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[10] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. Distributed set-expression cardinality estimation. In *Intl. Conf. Very Large Data Bases*, 2004.

[11] M. Dilman and D. Raz. Efficient reactive monitoring. In *IEEE Infocom*, 2001.

[12] D. Donoho. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, April 2006.

[13] V. Doshi, D. Shah, M. Médard, and S. Jaggi. Distributed functional compression through graph coloring. In *IEEE Data Compression Conf.*, 2007.

[14] L. Huang, X. Nguyen, M. Garofalakis, J. Hellerstein, A. D. Joseph, M. Jordan, and N. Taft. Communication-efficient online detection of network-wide anomalies. In *IEEE Infocom*, 2007.

[15] P. Indyk. Algorithms for dynamic geometric problems over data streams. In *ACM Symp. Theory of Computing*, 2004.

[16] A. Jain, J. Hellerstein, S. Ratnasamy, and D. Wetherall. A wakeup call for internet monitoring systems: The case for distributed triggers. In *Proceedings of the 3rd Workshop on Hot Topics in Networks (Hotnets)*, 2004.

[17] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiments with zebranet. In *ASPLOS-X*, 2002.

[18] R. Keralapura, G. Cormode, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *ACM SIGMOD Intl. Conf. Management of Data*, 2006.

[19] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Systems*, 30(1):122–173, 2005.

[20] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Now Publishers, 2005.

[21] S. Muthukrishnan. Some algorithmic problems and results in compressed sensing. In *Allerton Conference*, 2006.

[22] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distribtuted data streams. In *ACM SIGMOD Intl. Conf. Management of Data*, 2006.

[23] A. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *IEEE Symp. Foundations of Computer Science*, 1977.

[24] A. C. Yao. Some complexity questions related to distributive computing. In *ACM Symp. Theory of Computing*, 1979.