Continuous Distributed Counting for Non-monotonic Streams

Zhenming Liu*
zhliu@eecs.harvard.edu
Harvard School of Engineering and Applied Sciences

Božidar Radunović
bozidar@microsoft.com
Microsoft Research Cambridge

Milan Vojnović
milanv@microsoft.com
Microsoft Research Cambridge

ABSTRACT

We consider the continual count tracking problem in a distributed environment where the input is an aggregate stream that originates from k distinct sites and the updates are allowed to be non-monotonic, i.e. both increments and decrements are allowed. The goal is to continually track the count within a prescribed relative accuracy ϵ at the lowest possible communication cost. Specifically, we consider an adversarial setting where the input values are selected and assigned to sites by an adversary but the order is according to a random permutation or is a random i.i.d process. The input stream of values is allowed to be non-monotonic with an unknown drift $-1 \leq \mu \leq 1$ where the case $\mu = 1$ corresponds to the special case of a monotonic stream of only non-negative updates. We show that a randomized algorithm guarantees to track the count accurately with high probability and has the expected communication cost $\tilde{O}(\min\{\sqrt{k}/(|\mu|\epsilon), \sqrt{kn}/\epsilon, n\})$, for an input stream of length n, and establish matching lower bounds. This improves upon previously best known algorithm whose expected communication cost is $\tilde{\Theta}(\min\{\sqrt{k}/\epsilon, n\})$ that applies only to an important but more restrictive class of monotonic input streams, and our results are substantially more positive than the communication complexity of $\Omega(n)$ under fully adversarial input. We also show how our framework can also accommodate other types of random input streams, including fractional Brownian motion that has been widely used to model temporal long-range dependencies observed in many natural phenomena. Last but not least, we show how our non-monotonic counter can be applied to track the second frequency moment and to a Bayesian linear regression problem.

Categories and Subject Descriptors

F.2.2 [**Analysis of algorithms and problem complexity**]: Nonnumerical algorithms and problems; H.2.4 [**Database management**]: Systems-distributed databases

*Work performed while an intern with Microsoft Research Cambridge.

PODS'12, May 21–23, 2012, Scottsdale, Arizona, USA.

General Terms

Algorithms, theory

Keywords

Distributed counting, non-monotonic streams

1. INTRODUCTION

A continuous distributed tracking model was introduced in [7] to address the challenges of designing an effective strategy to constantly track statistics in a dynamic, distributed environment. In this model, data arrive in multiple streams to a number of sites. All the sites are connected to a coordinator, and the goal of the coordinator is to continuously track some function of the aggregate data, and update it as new data arrives. An exact tracking would require each data sample to be communicated to the coordinator, which would incur a prohibitively large communication cost - linear in the size of the input stream. Similarly, space and time processing requirements may be very large. However, for most applications it is satisfactory to provide an approximate tracking. Thus, a general formulation of a continuous distributed tracking problem is to design an algorithm that will minimize the space, time and/or communication complexity while providing approximation guarantees on the tracking accuracy. Continuous distributed tracking problems have recently gained much interest in the research community [23, 8, 1, 22].

One of the basic building blocks for many of the existing algorithms is a counter. The goal of the counter is to report, with a given relative accuracy, the sum of values of all elements that have arrived across the aggregate stream arriving from distributed sites. The main assumption in almost all the previous works is that the input stream being counted is monotonic non-decreasing and, surprisingly, there is very little work on continuous distributed non-monotonic counters. Similarly, most of the previous algorithms using counters are not guaranteed to work correctly under non-monotonic input stream.

However, many data streams do not satisfy the monotonicity property. A simple motivating example is a voting/ranking application. Suppose users' votes come in a distributed stream. The goal is to keep a continuous track of which of the two options has a higher number of votes, and approximately by which voting margin. Here, the votes for each option can essentially be seen as two separate data streams, but we are interested in continuously monitoring the *difference* of the two streams, which is clearly non-monotonic. The naive approach of estimating the count of each option separately and then taking the difference will not provide a relative error guarantee for the difference.

Non-monotonic streams are common in many situations when

dealing with instantaneous instead of cumulative phenomena, e.g. tracking a difference. One example we analyze in more detail is monitoring a process that exhibits long-range dependency, a phenomena that has been found to be prevalent in nature, e.g. network traffic [14]. Also, non-monotonic counters are useful as building blocks in more complex algorithms whose inputs are not necessarily monotonic. A source of non-monotonicity could be the use of random projections that transform an input data stream into a non-monotonic stream. Another example that we discuss is a streaming implementation of a Bayesian linear regression problem (c.f. [2]), which is useful in the context of machine learning platforms for processing of large-scale data (e.g. [16]).

In this work we are interested in designing a continuous non-monotonic distributed counter with optimal communication complexity. We will also discuss its applications in different scenarios. In the next section, we define the problem in more detail.

1.1 Problem Definition

Consider a standard distributed streaming model where k sites are connected to a coordinator. Each site is allowed to communicate with the coordinator but they cannot communicate with each other directly (and a broadcast message counts as k messages). Data items $a_1, \ldots, a_n$ arrive at sites $\psi(1), \ldots, \psi(n)$ respectively, at time instants $\tau_1 < \cdots < \tau_n$. We shall refer to the item a_t as the t-th update. In a general continuous distributed monitoring problem, the coordinator is responsible to maintain a value of a function $f(a_1, \ldots, a_t)$ at each time τ_t with a relative accuracy ϵ. We are interested in the *counter* problem, where the goal is to track a sum $S_t = \sum_{i \leq t} a_i$ of all the items that have arrived until time τ_t. The coordinator then needs to maintain an estimate that is between $(1-\epsilon)S_t$ and $(1+\epsilon)S_t$. Note that, by definition, the counter problem has low space and time complexity, and thus we focus on minimizing communication complexity.

A monotonic counter only allows for positive increments. In particular, a canonical example of a monotonic counter [12] implies $a_t = 1$ for all t, meaning that a counter is incremented by one whenever an update arrives. We relax the assumption that a_t is positive, and we call this a *non-monotonic counting* problem. In the streaming literature, this input model is usually called a general (non-strict) turnstile model [17].

To the best of our knowledge, the only research so far dealing with non-monotonic input streams is Arackaparambil et al. [1], who studied the tracking of frequency moments F_p, where deletion operations are allowed ($a_t = 1$ denotes an insertion and $a_t = -1$ denotes a deletion). There, a strong negative result is established for the adversary input case for both tracking counts and tracking F_p when deletion is allowed: the worst-case communication complexity is $\Omega(n)$ messages for an input stream of n elements. It is straightforward to construct a worst-case input for the counter problem: consider the case where there is only one site and the updates consist of alternations between an insertion and a deletion. In this case, the true global counter evolves as the sequence $0, 1, 0, 1, \ldots$. When one update is missed from the site, then the multiplicative error from the server becomes unbounded. Therefore, upon the arrival of each update, the site has to send a message to the server, which implies a communication lower bound of $\Omega(n)$ messages. While there is no way to circumvent this linear lower bound barrier for the worst-case input, it is natural to ask what the communication complexity is when the input is not fully adversarial and consider the following question:

Can we design a continuous, distributed tracking protocols for counter for non-monotonic updates that has a sublinear communication complexity when the input is randomized?

In particular, we are interested in a random permutation model. In this model, an adversary first decides the entire sequence of updates $a'_1, \ldots, a'_n$ for all sites. We only assume that the sequence is bounded. Then, the "nature" decides a random permutation π. The final input to the sites is $a_1 = a'_{\pi(1)}, a_2 = a'_{\pi(2)}, \ldots, a_n = a'_{\pi(n)}$. This model is very natural in large-scale settings (such as Internet scale, for example), where data is collected from a large number of individuals (e.g. Twitter or Facebook users). In such a model, a large amount of data is generated in short time intervals, and it is reasonable to assume that the order in which the individuals enter their inputs in the system is random, but the input itself can be arbitrary.

We are also interested if sublinear algorithms can be obtained for other types of random inputs that are well motivated by applications. For example, the use of random projections for computing sketches motivates to consider random i.i.d. updates. Another example is found in nature where many real-world phenomena exhibit self-similarity and long-range dependence (e.g. network traffic [14]) which where traditionally modeled by random processes such as fractional Brownian motion and found to be in good conformance with empirical data.

In all these data models, we shall assume that an adversary chooses the function $\psi(t)$ which defines how the stream is partitioned among the sites (an example is a load-balancing algorithm that can arbitrarily scatter inputs across sites). The adversary can only decide the function $\psi(t)$ based on the information observed up to a point in time. This means that when the input is a random permutation, $\psi(t)$ can depend on the content of the updates decided by the adversary, the prefix of the permutation observed so far, and the values $\psi(1), \ldots, \psi(t-1)$; while when the input is random, the function $\psi(t)$ can only depend on the values of $a_1, \ldots, a_{t-1}$ and the values of $\psi(1), \ldots, \psi(t-1)$; . We will also assume that the times $\tau_1, \ldots, \tau_n$ at which the inputs arrive are decided by an adversary. This essentially implies that the coordinator and the other sites have no knowledge of the time instants at which an input arrives to its corresponding site, and any communication can only be initiated by a site that has received an update.

Note that our model is a strict generalization of the standard monotonic stream model for counting (c.f. Huang et al. [12]), where the updates are fixed to $a_t = 1$, and the arrival times and sites are adversarial. In our case, we relax the assumption on the value of updates and allow for randomly permuted adversarial or entirely random values of updates, while still keeping the adversarial data partitioning and arrival times.

1.2 Our Contributions

Our main results in this paper are matching upper and lower bounds on the communication complexity for a continuous, distributed, non-monotonic counter (up to a poly-logarithmic factor), which are sublinear in the size of the input. While these bounds hold for different types of inputs, we give a single algorithm that is optimal for all the types of inputs considered, and whose communication cost also matches the corresponding lower bounds. The algorithm is lightweight in having only $\tilde{O}(1)$ space and update time complexity[1].

We first provide results for the case of Bernoulli i.i.d. input (Section 3) where we develop basic techniques that will be used in subsequent analysis. In the Bernoulli i.i.d. model, we assume that each update a_t is a Bernoulli random variable, with $\Pr[a_t = 1] =$

[1]We use $\tilde{O}(x) = x \log^{O(1)}(nk/\epsilon)$ notation to ignore log factors.

$1 - \Pr[a_t = -1] = p$, for some unknown parameter $p \in [0, 1]$. The counter value S_t is then a Bernoulli random walk with a drift $\mu = 2p - 1$. In the case of a Bernoulli i.i.d. input without a drift ($\mu = 0$), we show that a count can be tracked with $\tilde{O}(\sqrt{kn}/\epsilon)$ communication cost. In case of a Bernoulli i.i.d. input with an unknown drift $\mu \in [-1, 1]$, the achievable communication cost is $\tilde{O}(\min\{\sqrt{kn}, \sqrt{k}/|\mu|\}/\epsilon)$. In both cases, our algorithm does not need to know the drift. We also give matching lower bounds for most important cases (Section 4), showing the optimality of our algorithm.

This result should be compared with the communication cost $\tilde{\Theta}(\sqrt{k}/\epsilon)$ for a monotonic counter (with $a_t = 1$) that was recently established in [12]. We show that the same bound holds for a more general choice of updates (any i.i.d. Bernoulli input), as long as the drift is a positive constant. This is perhaps not entirely surprising, as for the constant drift case we use the algorithm from [12] as one of the building blocks for our algorithm. The key novel insight is that the communication cost increases to $\tilde{O}(\sqrt{kn}/\epsilon)$ when the drift is $|\mu| = O(1/\sqrt{n})$. Thus, we demonstrate that we are still able to track the count with a sublinear communication cost, and we describe the parameter ranges in which the cost is polynomial vs. polylog in the input size.

We next turn to our main results for the permutation model. Here we show that tracking is achievable with $\tilde{O}(\sqrt{kn}/\epsilon)$ communication cost and we give a matching lower bound (Section 4). This is to be contrasted with $\Theta(n)$ lower bound for a non-monotonic counter in a fully adversarial setting [1]. We show that, in a setting where all other parameters are chosen by an adversary, randomly permuting an arbitrary non-monotonic input is enough to permit a tracking algorithm with a sublinear communication cost. This shows that a sublinear tracking of non-monotonic input is still possible in a large number of real-world scenarios.

We further show that our algorithm can track a fractional Brownian motion with Hurst parameter $H \in [1/2, 1)$, where $1 < \delta \leq 1/H$ is an arbitrary parameter (Section 3.4) with total expected communication cost of $\tilde{O}(k^{\frac{3-\delta}{2}} n^{1-H}/\epsilon)$ messages. For the case of independent increments ($H = 1/2$), we get the same bound as before. For the case of positively correlated increments ($1/2 < H < 1$), which is of most interest in applications, we get a smaller communication cost. This is intuitive in view of the facts that increments are positively correlated which makes the process more predictable and the variance is larger. This in turn implies smaller expected residence of the count in the region of small values where there is a higher sensitivity to relative errors. Interestingly, the algorithm does not require to know the exact value of the parameter H, but only needs to have an estimate $1/\delta$ such that $H \leq 1/\delta$.

Finally, we show how our counter can be used as a building block for some instances of distributed tracking problems (Section 5). First, we construct an algorithm to track the second frequency moment (F_2 tracking) with $\tilde{O}(\sqrt{kn}/\epsilon^2)$ communication complexity (Section 5.1) and then provide a $\Omega(\min\{\sqrt{kn}/\epsilon, n\})$ lower bound that is matching in both n and k. We also show how to use the non-monotonic counter as a building block for a Bayesian linear regression problem (Section 5.2), and show that the Bayesian linear regression can also be tracked with sublinear communication cost.

It is noteworthy that while the communication cost for non-monotonic random streams with subconstant drift is sublinear in the input size, this is significantly larger than for monotonic streams ($\tilde{O}(\sqrt{n})$ vs $\tilde{O}(1)$), which is because the problem is intrinsically more difficult. However, the fact that the communication cost is sublinear in the input size would still make the algorithm of appeal for practical applications. For example, Twitter users generate more than 10^8 tweets a day [21]. In this scenario, the communication cost of our algorithm for tracking a single counter would only be in the order of 10^4 messages per day, which is a significant reduction of the traffic load. Furthermore, our bounds are matching with the bounds for the monotonic counters in k and ϵ parameters.

Finally, we briefly discuss the main techniques used in this paper. As we are designing algorithms for random streams in a distributed environment, our solution naturally calls for an integration of different techniques from sampling theory, analysis of stochastic processes, classical streaming algorithms, and distributed algorithm design. The main ingredient in our algorithm to tackle an otherwise intractable problem in the adversarial setting is to make an optimal prediction on the evolution of the counter process using a scarce communication resource and adaptively changing the tracking strategy as we continuously update our predictions. Making the prediction requires us to understand the volatility structure of the counter process; in our specific case, this boils down to the analysis of first passage time of random walks and random permutations. Designing a communication efficient tracking algorithm requires us to construct a sampling based protocol that can judiciously cope with the volatile structure of the process. To prove our matching lower bounds, we needed to carefully decompose the entire tracking process into disjoint segments so that we can apply results from the communication complexity and sampling theory separately on each segment and reach a strong lower bound that is polynomial in n.

Due to space constraints, we omit some of the proofs. All the proofs can be found in the technical report [15].

1.3 Related Work

The research on functional monitoring in distributed systems has considered a variety of problems (e.g. [9, 10, 3, 6, 18]) including one-shot and continuous tracking query problems. To the best of our knowledge, Cormode et al. [7] is the first work that articulated the distributed computation model that we consider in the present paper. Substantial progress has recently been made on understanding various problems under this model, including drawing a sample with or without replacement (e.g. [8, 22]) and answering holistic queries such as tracking the rank of an item or computing a quantile (e.g. [12, 5, 23]).

The most closely related work to ours is the recent work of Huang et al. [12] and Arackaparambil et al. [1]. The work of Huang et al. examines the same counter problem as ours but assuming an important but more restrictive class of monotonic streams, where only positive increments are allowed. Our work relaxes this assumption on the input by allowing for non-monotonic streams where decrements are allowed (either i.i.d. or random permutation). Specifically, we assume that the rate of positive increments is $(1 + \mu)/2$, for some unknown drift parameter $-1 \leq \mu \leq 1$. For the special case of the drift parameter $\mu = 1$, our counter algorithm would solve the same counter problem as in [12] with the matching performance.

The work of Arackaparambil et al. considered non-monotonic functional monitoring in the adversarial setting, including the problem of continuously tracking F_2 that we study here. They established an $\Omega(n)$ lower bound for the 1-site case and an $\Omega(n/k)$ lower bound for the k-site case. For the random input stream that we study here, we establish a tight lower bound that is sublinear in n and grows with k, suggesting that our problem under random input may have a different structure than the one under fully adversarial input.

2. ALGORITHMS AND NOTATIONS

We now present our algorithm for continuous distributed counting for non-monotonic streams. The algorithm is applicable to all input models, subject to choosing appropriate constants. In the subsequent sections we will show how to choose the constants for each input model under consideration.

In what follows, we shall write X_i be the i-th input (because the input is stochastic) and let $\mu = \mathrm{E}[X_i]$ be the *drift rate* of the counter process, $\hat{S}_t$ be the coordinator's estimate of S_t. When the context is clear, we refer to a_t as both the t-th update and the item arrived at time t interchangeably though we shall be clear that the actual physical time is irrelevant in our algorithm. Also, we shall assume that each site always keeps track of the total number of updates arrived locally and maintain a local sum counter. Let us start with introducing the basic constructs we need for the design of our distributed algorithm.

2.1 Building Blocks

The key novel building block in our scheme is:

Sampling and Broadcasting (SBC)**.** In this protocol, the coordinator broadcasts its current estimate $\hat{S}_t$ to all the sites at the beginning. Each site maintains a common sampling rate $\approx 1/(\epsilon^2 \hat{S}_t^2)$ that depends only on the global estimate $\hat{S}_t$. Whenever a site receives a new update, it samples a Bernoulli random variable R_t with the above rate. If $R_t = 1$, the following actions will be carried out sequentially (invoking $\tilde{\Theta}(k)$ message exchanges):

1. The site signals the coordinator to sync all data.
2. The coordinator broadcasts a message to all the sites to collect their local counters.
3. Each site reports its local counter to the coordinator. The coordinator computes the new exact count and broadcasts this new count to all sites.

Upon receiving the new count $\hat{S}_t$, each site adjusts the sampling rate of the Bernoulli random variable to

$$\text{Sample-Prob}(\hat{S}_t, t) = \min\left\{\frac{\alpha \log^\beta n}{\epsilon^2 \hat{S}_t^2}, 1\right\} \tag{1}$$

where α and β are some appropriately chosen positive constants.

We will also use the following building blocks:

HYZ counter. In [12], a distributed counter is developed to track monotonic updates with a relative accuracy ϵ and error probability δ using $\tilde{O}(\frac{\sqrt{k}}{\epsilon}\log(1/\delta))$ communication when $k = O(1/\epsilon^2)$ and $\tilde{O}(k\log(1/\delta))$ communication when $k = \omega(1/\epsilon^2)$ (here, $\tilde{O}(\cdot)$ hides the poly-logarithmic dependencies on n). We shall refer this protocol as $\mathrm{HYZ}(\epsilon, \delta)$.

Geometric Progression Search for μ (GPSearch**).** The goal of this building block is to produce a reliable estimator of μ. It will report an estimate $\hat{\mu}$ only when sure w.h.p. that $\hat{\mu} \in [(1-\epsilon)\mu, (1+\epsilon)\mu]$, where ϵ is a given constant. It also guarantees that $\hat{\mu}$ is found before time $\Theta(\log n/\mu^2)$. We describe the GPSearch protocol in more details in [15].

Straightforward Synchronization (StraightSync**).** In this protocol, the coordinator pulls out the exact values of t and S_t from the sites in the beginning of the protocol. When a local site receives an update, it contacts the coordinator and executes the following steps sequentially:

1. The site sends both the total number of local updates and the local counter to the coordinator.
2. The coordinator updates the global count and the global number of updates.
3. The coordinator sends the updated global count and global number of updates to the site.

2.2 Algorithm Overview

Our algorithm, called **Non-monotonic Counter**, consists of two phases.

Phase 1: The first phase covers updates from $t = 0$ to $t = \tau$, where $\tau = c\log n/(\mu^2\epsilon)$ for some sufficiently large constant $c > 0$. During this phase, we have two communication patterns:

- When $(\epsilon\hat{S}_t)^2 \geq k$, we use the SBC protocol.
- When $(\epsilon\hat{S}_t)^2 < k$, we use the StraightSync protocol.

The coordinator shall make a broadcast when the algorithm makes a switch between SBC and StraightSync protocol.

Phase 2: The second phase covers from $t = \tau$ to $t = n$ (the second phase could be empty when $\tau \geq n$). In the second phase, the algorithm maintains a $\mathrm{HYZ}(\Theta(\epsilon\mu), \Theta(1/n^2))$ to track the total number of positive updates and another $\mathrm{HYZ}(\Theta(\epsilon\mu), \Theta(1/n^2))$ to track the total number of negative updates. The difference between the positive updates and the negative updates is the estimator maintained by the coordinator.

In addition, our GPSearch procedure is executed in the background, and it will be able to tell us a good estimate of μ, and decide when Phase 1 ends. When the algorithm changes from the first phase to the second phase, the coordinator shall make a broadcast to inform different sites of the phase change.

3. UPPER BOUNDS

In this section, we analyze *Non-monotonic Counter* for i.i.d. input, randomly ordered streams, and fractional Brownian motion. Because the input is of stochastic nature, we shall write the updates as $X_1, X_2, \ldots, X_n$ instead of $a_1, \ldots, a_n$ to emphasize the randomness. Our analysis starts with the simplest case, where $k = 1$ and the input is i.i.d. with $\mu = 0$. Then we move to the more complex scenarios, in which there are multiple sites and unknown μ. Finally, we generalize our algorithms and analysis to the randomly ordered stream and fractional Brownian motion case, where the updates are no longer independent. Along the way, we shall explain the reasons why we design the algorithm in such a way and gradually unfold the key techniques used in the analysis.

3.1 I.I.D. Input with Zero Drift

Recall the algorithm Non-monotonic Counter that is described in Section 2. To analyze the behavior of our algorithm we start by giving an upper bound for the single-site case ($k = 1$), and we then turn to the multi-site case. In the single-site case, we need to introduce a small modification to the algorithm. Since the site is aware of the exact counter value $\hat{S}_t = S_t$, there is no need for the straightforward stage and we assume that the algorithm is always in the broadcast stage. Also, there is no need for the coordinator to send messages back to the site.

We will use the sampling probability as defined earlier in (1) with $\alpha > 9/2$ and $\beta = 2$. The parameter α controls the tradeoff between communication complexity and the success probability. The choice of p_t is intuitive because the smaller S_t is, the more likely that a small change of S_t will cause large multiplicative change; therefore, the site should report the value to the coordinator with higher probability.

We have the following theorem:

THEOREM 3.1. *For the single-site case, the randomized algorithm Non-monotonic Counter with the sampling probability as in (1), with $\alpha > 9/2$ and $\beta = 2$, guarantees to track the count within the relative accuracy $\epsilon > 0$ with probability $1-O(1/n)$ and uses the total expected communication of $O(\min\{\sqrt{n}/\epsilon \cdot \log n, n\})$ messages.*

Proof is provided in [15]. Here, we comment on the intuition of using a sampling based algorithm and setting the sampling probability as specified in (1). We want the site to send messages to the coordinator as infrequently as possible. Suppose that at time t, we have $S_t = s$ and a message is sent to the server. We need to understand what the next appropriate time would be to send another message. Ideally, this shall happen at the time where S_t first passes through either $s/(1+\epsilon)$ or $s/(1-\epsilon)$. Implementing this strategy is feasible when there is only one site but it is unclear how it can scale up to k site case (because the challenge of distributed tracking algorithms exactly lies in the difficulties of *exactly* tracing the aggregate statistics and thus it is hard to spot the exact first passage time). It is therefore desirable to use a "smooth" strategy by a site, i.e. the algorithm does not critically rely on the knowledge on the time when S_t first passes through some pre-specified points. The sampling based algorithm possesses such a property. We also need to estimate the sampling rate of the algorithm. Intuitively, it takes an unbiased random walk approximately $(\epsilon s)^2$ time to travel for a distance of about length ϵs (to hit either $s/(1-\epsilon)$ or $s/(1+\epsilon)$). When ϵs becomes sufficiently large, we even have a concentration result, i.e. with high probability, the time it takes to hit either $s/(1-\epsilon)$ or $s/(1+\epsilon)$ is $\tilde{\Theta}((\epsilon s)^2)$. Therefore, sampling at rate $\tilde{\Theta}(1/(\epsilon s)^2)$ is not only sufficient to maintain high accuracy but also optimal.

We now extend to the multiple site case. Let us first go through the intuition why we want to distinguish the two stages of the algorithm, the straightforward ($\mathrm{StraightSync}$) stage and the broadcast (SBC) stage, described in Section 2. The main idea of our distributed algorithm is to simulate the behavior of the sampling algorithm for the single site case (Theorem 3.1). For that we require that each site has a good estimate $\hat{S}_t$ of the global count S_t. As $\hat{S}_t$ gets updated, the copies of the counter at all sites need to be updated, in order to maintain the correct local sampling rate. The only way to achieve so is to broadcast the counter, which would result in $\tilde{\Theta}(k)$ messages exchanged. The crucial observation here is that when $\hat{S}_t$ gets updated frequently (i.e., when S_t is sufficiently small), broadcasting messages after each update could be wasteful. It may be even worse than the trivial approach where a site synchronizes only with the coordinator whenever it receives an update (resulting in $\tilde{\Theta}(1)$ messages). This "trivial" approach is captured in our straightforward strategy, and we switch to it whenever it is less expensive. Note that we can use the estimator $\hat{S}_t$ instead of the actual value S_t to decide whether the broadcasting or the straightforward strategy has the smaller communication cost, because we guarantee a sufficiently high accuracy of the estimator.

We have the following theorem.

THEOREM 3.2. *The randomized algorithm Non-monotonic Counter with the sampling probability as in (1), with α large enough positive constant and $\beta = 2$, guarantees to track the count within the relative accuracy $\epsilon > 0$ with probability $1 - O(1/n)$ and uses the total expected communication of $O(\min\{\sqrt{kn}/\epsilon \cdot \log n, n\})$ messages.*

Proof is provided in [15]. It is based on a coupling argument that enables us to reuse the result of Theorem 3.1.

3.2 I.I.D. Input with Unknown Drift

In the previous section we have seen that the communication complexity in the case with no drift is $\tilde{O}(\sqrt{n})$. However, the monotonic counter from [12] is a special case of our model with $\mu = 1$, and its communication complexity is $\tilde{O}(1)$. Clearly, we conclude that a positive drift might help. The natural question then is whether this observation holds for an arbitrary drift $\mu \neq 0$, and how can we exploit it when the drift is unknown.

To gain an intuition on the input's behavior for an arbitrary drift, it is helpful to re-parameterize each input X_t as $X_t = \mu + Z_t$, where μ is the drift term and Z_t is a random variable representing the "noise" term, We shall intuitively view Z_t as noise that behaves similar to Gaussian noise. We want to identify which term contributes more to the estimation error. Suppose $S_t = s$. It takes the drifting term ϵt time units to reach $\pm\epsilon s$ while it takes the noise term $(\epsilon s)^2$ to do so. When $\epsilon t < (\epsilon s)^2$, the drifting term dominates the process, otherwise, the noise term dominates the process. Approximating s by its mean $s \approx t\mu$ and solving the equation, $\epsilon t < (\epsilon s)^2$, we get $t \approx 1/(\mu^2\epsilon)$. Therefore, the random walk S_t qualitatively behaves as follows: up to time $t = \Theta(1/(\epsilon\mu^2))$, the "noise" sum term dominates the process; after time $\Theta(1/(\epsilon\mu^2))$, the drifting term dominates the process. Therefore, intuitively, for $t \leq 1/(\epsilon\mu^2)$ we should use the algorithm that deals with the non-drift case, and for $t \geq 1/(\epsilon\mu^2)$ we might be able to use the monotonic counter HYZ.

Note that the algorithm does not know the actual value of the drift μ. We use an online estimator (the $\mathrm{GPSearch}$ algorithm, described in Section 2) to obtain an estimate $\hat{\mu}$. Our estimator is conservative in the sense that it does not report $\hat{\mu}$ until confident that it is within $[(1-\epsilon')\mu, (1+\epsilon')\mu]$ (the performance of the $\mathrm{GPSearch}$ estimator is discussed in [15]). Once the estimator $\hat{\mu}$ is reported, we can safely switch to the monotonic counter HYZ.

However, we need to guarantee correctness of the algorithm even before we have an estimate of $\hat{\mu}$. The monotonic counter HYZ essentially samples with sampling probability $\Theta(1/(\epsilon t))$. So to guarantee the correctness before we know whether we are in the no-drift phase or in the drift phase, we need to sample with the maximum of the sampling rate $\Theta(1/(\epsilon^2 s^2))$ of the no-drift phase and the sampling rate $\Theta(1/(\epsilon t))$ of the monotonic counter. We shall choose a slightly more conservative rate by tuning the constants in the sampling probability (1) so that Sample-Prob$(S_t, t) \geq \tilde{\Theta}(1/\epsilon^2 s^2 + 1/\epsilon t)$ for all $t < 1/(\mu^2\epsilon)$.

The crucial observation here is that this conservative way of sampling will not result in substantial increase in communication resource. Indeed, we have two types of unnecessary communication costs:

- Type 1: when $t \leq 1/(\epsilon\mu^2)$, the term $\tilde{\Theta}(1/\epsilon t)$ in the sampling rate is wasteful.
- Type 2: when $t > 1/(\epsilon\mu^2)$, the term $\tilde{\Theta}(1/(\epsilon s)^2)$ in the sampling rate is wasteful.

The total expected communication cost of type 1 is $O(\sum_{t\leq n}(1/t)) = O(\log n)$, which is acceptable. Computing the waste of type 2 is a bit more tedious, but we would be able to see that in expectation $\sum_{t\leq 1/(\epsilon\mu^2)} 1/(\epsilon^2 S_t^2) = \Omega\left(\sum_{t\leq n} 1/(\epsilon^2 S_t^2)\right)$. In other words, $\sum_{1/(\epsilon\mu^2)<t\leq n} 1/(\epsilon^2 S_t^2) = O\left(\sum_{t\leq 1/(\epsilon\mu^2)} 1/(\epsilon^2 S_t^2)\right)$, i.e the total wasted communication from the term $\tilde{\Theta}(1/(\epsilon s)^2)$ is bounded by the total "useful" communication from the same term. Therefore, the conservative sampling strategy is also optimal.

Finally, we can also split the no-drift phase into the the straightforward ($\mathrm{StraightSync}$) stage and the broadcast (SBC) stage, as

discussed in the previous section. We then have the following theorem.

THEOREM 3.3. *There exists a choice of constants α and $\beta > 0$ for the randomized algorithm Non-monotonic Counter, for the k-site count problem with unknown drift, to guarantee the continuous tracking within a prescribed relative accuracy ϵ with high probability and the following communication cost in expectation:*

- $\tilde{O}\left(\min\left\{\frac{\sqrt{k}}{|\mu|\epsilon}, \frac{\sqrt{kn}}{\epsilon}, n\right\} + \frac{\sqrt{k}}{\epsilon}\right)$, *if* $k = O(1/(\mu\epsilon)^2)$, *and*
- $\tilde{O}\left(\min\left\{\frac{\sqrt{k}}{|\mu|\epsilon}, \frac{\sqrt{kn}}{\epsilon}, n\right\} + k\right)$, *if* $k = \omega(1/(\mu\epsilon)^2)$.

Notice that our algorithm's communication cost has two types of asymptotic behaviors for different k because the HYZ counter uses different strategies for different k. Proof of Theorem 3.3 is provided in [15].

3.3 Randomly Ordered Data Streams

We now move to the random permutation case. We use the same tracking algorithm described in Section 2 to solve this problem by using the sampling rate defined in (1) with $\beta = 2$ and sufficiently large $\alpha > 0$.

THEOREM 3.4. *Let $a_1, \ldots, a_n$ be an arbitrary, randomly permuted, sequence of bounded real values. The randomized algorithm Non-monotonic Counter with the sampling probability in (1) for $\beta = 2$ and sufficiently large constant $\alpha > 0$ guarantees to track the count within the relative accuracy ϵ with probability $1 - O(1/n)$ and uses the total expected communication of $O(\sqrt{kn}/\epsilon \cdot \log n + \log^3 n)$ messages.*

Note that here, because of the adversarial choice of the input sequence, we cannot exploit the drift. We remark that when the update is a fractional number from $[-1, 1]$ rather than $\{-1, 1\}$, our Non-monotonic Counter algorithm still holds. The key difference between the analysis for Theorem 3.4 and the one for i.i.d. input is that the updates are correlated when the content of the stream is decided in advance. This difference boils down to a modified analysis for the first passage time of the partial sums. In the Bernoulli i.i.d. case, a straightforward application of Hoeffding's inequality suffices to give a bound on the first passage time. While here Hoeffding's inequality is no longer applicable, we are able to use tail inequalities for sampling without replacement [11, 20] to circumvent this problem. Technical report [15] gives a detailed analysis.

3.4 Fractional Brownian Motion

In this section we consider the counting process S_t evolving as a fractional Brownian motion with parameters $\sigma > 0$ and $0 < H < 1$ where we extend the counting process to continuous time in a natural manner. We briefly discuss some of the basic properties of fractional Brownian motion (more details can be found, e.g. in [19]). Fractional Brownian motion is a process with stationary increments whose finite dimensional distributions are Gaussian. Specifically, for a fractional Brownian motion S_t, we have $\mathrm{E}[S_t] = 0$, for every $t \geq 0$ and the covariance of the process is defined as

$$\mathrm{E}[S_t S_u] = \frac{\sigma^2}{2}(|t|^{2H} + |u|^{2H} - |u - t|^{2H}).$$

Thus, the variance of the process is $\mathrm{E}[S_t^2] = \sigma^2 |t|^{2H}$, for every $t \geq 0$. The parameter H is known as the Hurst parameter. For $H = 1/2$, the process corresponds to a Brownian motion whose increments are independent. For $0 < H < 1/2$, the variance of S_t grows sublinearly with t and the process has a negative autocorrelation while for $1/2 < H < 1$, the variance of S_t grows superlinearly with t. The process is self-similar, meaning that random variables S_{at} and $a^H S_t$ have the same distribution. To simplify notation, in the remainder, we will assume $\sigma^2 = 1$. Notice that this is without loss of generality as it amounts only to rescaling of the time units. It is noteworthy that the fractional Brownian motion is one of standard statistical models that captures some of the salient properties of temporal statistical dependencies that were observed in many natural phenomena, including self-similarity and long-range dependency (see, e.g. [19]).

We present an algorithm that requires only an upper bound on the Hurst parameter H and guarantees continual tracking within prescribed relative accuracy with high probability for the range $H \in [1/2, 1)$. Note that this is the range of particular interest in practice since typical values of the Hurst parameter observed in nature fall precisely in this region. For the purpose of deriving an upper bound on the communication complexity, we will write the sampling probability in the following form, for $1 < \delta \leq 2$,

$$\text{Sample-Prob}(S_t, t) = \min\left\{\frac{\alpha_\delta \log^{1+\delta/2} n}{(\epsilon|S_t|)^\delta}, 1\right\} \qquad (2)$$

where $\alpha_\delta = c(2(c+1))^{\delta/2}$, for any $c > 3/2$.

As before, we start with the single site (k=1) case. We have the following theorem (proof in Technical report [15]).

THEOREM 3.5. *For the single site ($k = 1$) case, the randomized algorithm Non-monotonic Counter with the sampling probability as in (2) guarantees to track the count within the relative accuracy $\epsilon > 0$ with probability $1 - 1/n$ for every $1/2 \leq H \leq 1/\delta$, where $1 < \delta \leq 2$, with the total expected communication of $O(n^{1-H}/\epsilon \cdot \log^{1/2+1/\delta} n)$ messages.*

We observe that for standard Brownian motion, which we may interpret as a continuous-time analog of a random walk, we have $H = 1/\delta = 1/2$, and in this case, the sampling probability and the result of the last theorem matches that of Theorem 3.1. For values of the Hurst parameter H in $(1/2, 1)$, the communication complexity of the algorithm is sublinear in n, with the upper bound increasing with n as a polynomial with the exponent decreasing with H as $1 - H$ (up to a poly-logarithmic factor). Note that this is inline with the intuition as a larger value of the parameter H means a larger variance and thus less of a concentration around value zero where the relative error tolerance is the most stringent.

Multiple sites. Finally, we look at the case with multiple sites. Let the sampling probability be as in (2) but with constant $\gamma_{\alpha,\delta}$ redefined as follows $\alpha_\delta = 9 \cdot 2^{\delta/2}(c+1)^{1+\delta/2}$, for any $c > 3/2$. Using the same coupling argument as in Theorem 3.3, we have the following corollary (proof in [15].)

COROLLARY 3.6. *The randomized algorithm Non-monotonic Counter, with the sampling probability given in (1), guarantees to track the count across k sites within the relative accuracy $\epsilon > 0$ with probability $1 - O(1/n)$ for every $1/2 \leq H \leq 1/\delta$, where $1 < \delta \leq 2$, with the total expected communication of $\tilde{O}(n^{1-H} k^{\frac{3-\delta}{2}}/\epsilon)$ messages*

4. LOWER BOUNDS

In this section, we establish matching lower bounds for the two cases of inputs: i.i.d. Bernoulli and random permutation. Recall that we denote with M_n the number of messages exchanged

over an input of size n. We are interested in the lower bounds on the expected number of messages $\mathrm{E}[M_n]$ that is necessary to track the value over an interval of n updates within ϵ relative accuracy with high probability. We use sample-path arguments to prove the results.

We start by presenting lower bounds for the single site case, first without and then with a drift. We then provide our main results that provides a lower bound parameterized with the number of sites k for the case without drift. We conclude by giving a lower bound for the case with random permutation input stream.

THEOREM 4.1. *Consider the single site ($k = 1$) continual count-tracking problem for an input of n random i.i.d. updates without a drift ($\Pr[X_i = 1] = \Pr[X_i = -1] = 1/2$) within relative accuracy $\epsilon > 0$ with probability at least $1 - O(1/n)$. Then the expected number of messages exchanged is $\Omega(\min\{\sqrt{n}/\epsilon, n\})$.*

Proof is provided in Technical report [15] . The key idea of the proof is the observation that whenever the value of the counter is in $\mathcal{E} = \{s \in \mathbb{Z} : |s| \leq 1/\epsilon\}$, the site must report the value to the coordinator as otherwise an error would occur with a constant probability. The proof then follows by noting that $\sum_{t \leq n} \Pr[S_t \in \mathcal{E}] = \Omega(\sqrt{n}/\epsilon)$.

The lower bound in Theorem 4.1 is established by counting the average number of visits to the set $\mathcal{E}$, and we can use the same argument to establish a lower bound for the general case of Bernoulli updates with with an arbitrary drift $-1 < \mu < 1$ (that is $0 < p < 1$). Intuitively, the no drift case should be the worst case with respect to communication complexity as observed in Section 3. Also, for any constant $\mu > 0$ we expect to have the lower bound similar to the bound from [12] for a monotonic counter $\mathrm{E}[M_n] = \Omega(1/\epsilon)$. It is thus of interest to ask what the lower bound would be for small but non-zero drift $\mu = o(1)$. We have the following result (proof in Technical report [15]).

THEOREM 4.2. *Consider the single site ($k = 1$) continual count-tracking with Bernoulli random walk updates with drift $\mu = o(1)$ and relative accuracy parameter $\epsilon > 0$. Suppose $\epsilon = \omega(1/\sqrt{n})$ and $|\mu| = O(\epsilon)$. Then, for the tracking to succeed with probability at least $1 - O(1/n)$, the expected number of messages is $\Omega\left(\min\left\{\sqrt{n}, \frac{1}{|\mu|}\right\} \cdot \frac{1}{\epsilon}\right)$.*

The result is in line with the intuition that any non-zero drift may only reduce the communication complexity, and it matches the bound in [12] for large enough μ. Our lower bound matches the corresponding upper bound result (presented in Technical report [15]) up to poly-logarithmic factors.

We now move to the main result of this section which provides a lower bound that is parameterized with the number of sites k. We consider only the non-drift case, as this is used to establish the lower bound for the permutation model. While the proof for $k = 1$ case essentially only needs to exploit the structure of a simple random walk, here we need to carefully integrate techniques from communication complexity theory with the structure of random walks. The main step is a reduction to a query problem (Lemma 4.4) that at a time instance asks whether the sum of updates over all k sites is larger than or equal to a $\Theta(\sqrt{k})$ threshold, which requires $\Omega(k)$ communication to guarantee a sufficiently low probability of error; otherwise, the overall error probability does not satisfy the requirements.

We start our analysis by introducing a building block for communication complexity.

DEFINITION 4.3 (TRACKING k INPUTS). *Let c be a constant. Consider the following functional monitoring problem: let X_1, X_2, ..., X_k be i.i.d. variables from $\{-1, 1\}$ such that $\Pr[X_i = -1] = \Pr[X_i = 1] = 1/2$ that arrive uniformly to each of the sites (i.e. each site receives exactly one update). Upon the arrival of the last update, we require the coordinator to*

- *be able to tell whether the sum is positive or negative if $|\sum_{i \leq k} X_i| \geq c\sqrt{k}$.*
- *do anything (i.e. no requirement) when $|\sum_{i \leq k} X_i| < c\sqrt{k}$.*

We have the following lemma.

LEMMA 4.4. *Solving the* tracking k inputs *problem with probability $1 - c_0$ (w.r.t. both the protocol and the input) for some constant c_0 requires $\Theta(k)$ communication.*

We provide a proof in Technical report [15] that is based on the same main ideas as in the proof of Lemma 2.2 in [12] with some minor technical differences to account for particularities of our setting.

We are now ready to present our main lower bound theorem for the k-site case.

THEOREM 4.5. *For the case of $k < n$ sites, the expected amount of communicated messages to guarantee relative accuracy $\epsilon > 0$ with probability at least $1 - O(1/n)$ is $\Omega(\min\{\sqrt{kn}/\epsilon, n\})$.*

Proof is provided in Technical report [15] . Here, again our lower bound matches the corresponding upper bound presented in Theorem 3.3. The intuition behind the result is as follows. We chop the stream into phases of k updates each, where each site gets exactly one update per phase. If the counter value S_t is in between $-\sqrt{k}/\epsilon$ and $\sqrt{k}/\epsilon$, we show that our problem is equivalent to the tracking of k input problems, and $\Theta(k)$ messages need to be sent to guarantee correctness. Summing the expected number of visits of the counter to these states, we obtain the lower bound.

Random Permutation. Finally, we have the following corollary providing a lower bound on the communication complexity for randomly permuted input stream (proof in [15]).

COROLLARY 4.6. *The expected total communication in presence of a randomly permuted adversarial input, with $-1 \leq a_t \leq 1$ for all $1 \leq t \leq n$, is at least $\Omega(\sqrt{kn}/\epsilon)$ messages.*

5. APPLICATIONS

5.1 Tracking the Second Frequency Moment

We now apply our distributed counter algorithms for continuously tracking second frequency moment of the randomly ordered stream. Let us recall the F_2 problem. The input stream consists of $a_1, a_2, ..., a_n$, where $a_t = (\alpha_t, z_t)$, α_t are all items from the universe $[m]$, and $z_t \in \{-1, 1\}$ for all $t \leq n$. Denote with $m_i(t) = \sum_{s \leq t : \alpha_s = i} z_s$ the sum of elements of type i in the stream at time t. Here, we allow the count $m_i(t)$ to be non-monotonic in t (i.e. allow decrements of the counts). Our goal is to continuously track the second moment of the stream, i.e. $F_2(t) = \sum_{i \leq m} m_i^2(t)$, at the coordinator. We shall refer to this problem as *monitoring $F_2(t)$ with decrements*.

Next, we review the fast AMS sketching algorithm for this problem (see [4] and references therein). Consider a set of counters $(S_{i,j})_{i \leq I, j \leq J}$ whose values at the t-th update are $S_{ij}(t)$, and set $S_{ij}(0) = 0$. Let $g_j : [m] \to \{-1, 1\}$ and $h_j : [m] \to J$ (for $j \leq I$) be two sets of 4-wise independent hash functions. Upon receiving the t-th item (α_t, z_t), we add $z_t \cdot g_j(\alpha_t)$ to all

$(S_{j,h_j(\alpha_t)}(t))_{j\le I}$. When $I = O(\log 1/\delta)$ and $J = O(1/\epsilon^2)$, we are able to recover F_2 with ϵ-relative guarantee with prob. $1-\delta$ for a specific t. We can then execute $\log n$ copies of the fast AMS sketches in parallel to make sure our estimator is correct for all updates.

An important property of fast AMS sketching is that it updates only $O(\log(1/\delta)+\log n) = \tilde{O}(1)$ counters after each update. Let $n_i = \sum_{s\le n} \mathbb{1}\{\alpha_s = i\}$ be the number of occurencies of item i in the stream until time t. Clearly, $\sum_{i\in[m]} n_i = n$. Further, let $\tilde{N}_{ij} = \{k \in [m] : h_j(k) = i\}$ be the set of items that map to the counter S_{ij}. Tracking counter S_{ij} in randomly ordered stream takes $\tilde{O}(\sqrt{k\sum_{k\in\tilde{N}_{ij}} n_k}/\epsilon)$ communication. Summing over all counter, and using Jensen inequality, we get that the expected total number of communicated messages is $\tilde{O}(\sqrt{kn}/\epsilon^2)$. We also remark that the lower bound $\Omega(\sqrt{kn}/\epsilon)$ for counter in randomly ordered stream is also a lower bound for $F_2(\cdot)$ for randomly ordered streams. We may summarize the upper bound and the lower bound results in the following corollary.

COROLLARY 5.1. *The communication lower bound for tracking the frequency moment $F_2(t)$ with decrements in randomly ordered stream is $\Omega(\min\{\sqrt{kn}/\epsilon, n\})$. There exists a randomized algorithm for tracking $F_2(t)$ using the total expected communication of $\tilde{O}(\sqrt{kn}/\epsilon^2)$ messages.*

5.2 Bayesian Linear Regression

We next describe another application of a distributed non-monotonic counter in tracking the posterior of the coefficients in a *Bayesian linear regression*. Recall the Bayesian linear regression problem (c.f. [2]): assume we are given a set of training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i$ is a row-vector $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We are interested in carrying out a linear regression over this data, i.e. finding a $\mathbf{w} \in \mathbb{R}^d$ such that $y = \mathbf{w}^T \cdot \mathbf{x}$ best fits the training data $\{(\mathbf{x}_t, y_t)\}_{t\le n}$. Furthermore, we impose an initial prior knowledge over the vector $\mathbf{w}_0$, and in particular we assume that it follows a multivariate Gaussian distribution $\mathbf{w}_0 \sim \mathcal{N}(\mathbf{m}_0, S_0)$. Our goal is to maintain a posterior belief over $\mathbf{w}_t$, as the training data $\{(\mathbf{x}_t, y_t)\}_{t\le n}$ arrives.

In the distributed functional monitoring setting, the training data $\{(\mathbf{x}_t, y_t)\}_{t\le n}$ arrives at different sites in a streaming fashion and the coordinator has to continuously track an approximate estimate of the mean $\mathbf{m}_t$ and the variance S_t of $\mathbf{w}_t$. We assume that the training data is an arbitrary bounded sequence selected by an adversary, and randomly permuted, as in the random permutation model.

We next describe how we may use $O(d^2)$ counters to track the posterior belief. Let A_t be an $t\times d$ matrix so that the i-th column of A_t is $\mathbf{x}_i$. Also, denote with $\mathbf{y}_t \in \mathbb{R}^t$ a vector whose i-th component is y_i. Furthermore, let β be the inverse of the variance of the noise variable in the model, i.e., $y_t = \mathbf{w}^T \cdot A_t + \mathcal{N}(0, \beta^{-1})$. The value of β is usually assumed to be a known parameter (see Bishop [2] for a detailed description of the model). It turns out that the posterior of $\mathbf{w}$ is also a Gaussian distribution with mean $\mathbf{m}_t$ and variance S_t, where

$$\begin{array}{rcl} \mathbf{m}_t & = & S_t(S_0^{-1}\mathbf{m}_0 + \beta A_t^T \mathbf{y}_t) \\ S_t^{-1} & = & S_0^{-1} + \beta A_t^T A_t. \end{array} \quad (3)$$

The inverse of S_t^{-1} at time t is also referred as the *precision matrix*. Observe that tracking the precision matrix S_t^{-1} as well as the vector $A_t^T\mathbf{y}$ suffices to recover the posterior structure of $\mathbf{w}$. Our specific goal here is to continuously track S_t^{-1} and $A_t^T\mathbf{y}_t$ by using our counter algorithm.

Upon the arrival of the $t+1$-st update, we have

$$S_{t+1}^{-1} = S_t^{-1} + \beta \underbrace{x_{t+1}^T x_{t+1}}_{\text{outer product of } x_{t+1}}$$

and $A_{t+1}^T\mathbf{y}_{t+1} = A_t^T\mathbf{y}_t + (y_{t+1}\times(x_{t+1})_1, y_{t+1}\times(x_{t+1})_2, \ldots, y_{t+1}\times(x_{t+1})_d)^T$.

Therefore, to track $S_t^{-1} \in \mathbb{R}^{d\times d}$, it suffices to keep d^2 counters $\{C_{i,j}\}_{i,j\le d}$ such that upon the arrival of the t-th training data, $C_{i,j} \leftarrow C_{i,j} + \beta(x_t)_i(x_t)_j$. Similarly, we may keep another d copies of counters $\{D_i\}_{i\le d}$ to track $A_t^T\mathbf{y}_t$, where $D_i \leftarrow D_i + y_t \times (x_t)_i$ at the t-th update. Notice that here our algorithm can guarantee each entry in S_t^{-1} and $A_t^T\mathbf{y}$ has at most ϵ-relative error. The actual error of our estimate for $\mathbf{m}_t$, however, also depends on how sensitive of the precision matrix's inverse is when it is perturbed.

The total communication complexity using this algorithm thus is $\tilde{O}(\sqrt{kn}d^2/\epsilon)$, being sublinear in the size of training data for a wide range of parameters.

Acknowledgments

We thank Zengfeng Huang, Ke Yi and Qin Zhang for useful discussions regarding the relation of Lemma 2.2 in [12] and the lower bound for the sampling problem in Lemma 4.4. We thank Graham Cormode for pointing out how to improve the bound for F_2 tracking using the fast AMS sketching.

6. REFERENCES

[1] C. Arackaparambil, J. Brody, and A. Chakrabarti. Functional monitoring without monotonicity. In *Proc. of ICALP*, 2009.

[2] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *Proc. of International Conference on Very Large Databases*, 2005.

[4] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 2011.

[5] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *Proc. of SIGMOD*, June 2005.

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

[7] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. In *Proc. of SODA*, 2008.

[8] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang. Optimal sampling from distributed streams. In *Proc. of PODS*, June 2010.

[9] M. B. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proc. of SIGMOD*, pages 58–66, 2001.

[10] M. B. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *Proc. of PODS*, 2004.

[11] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, pages 13–30, March 1963.

[12] Z. Huang, K. Yi, and Q. Zhang. Randomized algorithms for tracking distributed count, frequencies, and ranks. In *arXiv:1108.3413v1*, Aug 2011.

[13] T. Konstantopoulos. *Markov Chains and Random Walks*. Lecture notes, 2009.

[14] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[15] Z. Liu, B. Radunović, and R. Vojnović. Continuous distributed counting for non-monotonic streams. In *Technical Report MSR-TR-2011-128*, 2011.

[16] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. Hellerstein. Graphlab: A new framework for parallel machine learning. In *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

[17] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Computer Science*, 2005.

[18] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2003.

[19] G. Samorodnitsky and M. S. Taqqu. *Stable non-Gaussian random processes*. Chapman & Hall, 1994.

[20] R. J. Serfling. Probability inequalities for the sum in sampling without replacement. *Ann. Statist*, 2(1):39–48, 1974.

[21] T. S. Team. The engineering behind twitter's new search experience, 2011.

[22] S. Trithapura and D. P. Woodruff. Optimal random sampling from distributed streams revisited. In *Proc. of DISC*, Roma, Italy, Sep 2011.

[23] K. Yi and Q. Zhang. Optimal tracking of distributed heavy hitters and quantiles. In *Proc. of PODS*, June 2009.

APPENDIX

A. ADDITIONAL NOTATIONS

We will repeatedly use some notation in the rest of the appendices which we summarize in the following. We will denote the sampling probability in SBC for the t-th update with $p_t = \text{Sample-Prob}(\hat{S}_t, t)$. For an algorithm, we define E_n to be the number of errors observed over an input of size n. We will be interested in algorithms such that $\Pr[E_n > 0] = O(1/n)$. We define M_n to be the number of messages transmitted over an input of size n. We note that it is sufficient to limit the size of a message to $O(\log n)$ bits to convey any possible counter value. Thus the number of bits transmitted over an input of size n is $\tilde{\Theta}(M_n)$. We define R_t to be 1 if a message is sent to the coordinator and otherwise $R_t = 0$. We further denote with U_t the time until next message is sent to the coordinator as observed at time t. Similarly, we define V_t to be the time until the count process exits the ball $\mathbf{B}_\epsilon(S_t) = \{s \in \mathbb{Z} : |x - S_t| \leq \epsilon S_t\}$.

For the purpose of exposition, we will first start with the most fundamental case with Bernoulli i.i.d. increments. Recall that in this case $\Pr[X_t = 1] = p$ and $\Pr[X_t = -1] = 1 - p$. The expected increment in each step is then $\mu \triangleq p - (1-p) = 2p - 1$. We shall refer to μ as the *drift* of the problem. We will first treat the case without drift ($\mu = 0$ and $p = 1/2$) and then the general case with an unknown drift. The analysis for other distributions heavily utilizes the idea developed for these simple cases.

B. ANALYSIS FOR I.I.D. INPUT WITH ZERO DRIFT

B.1 Single Site Case

In this subsection, we prove Theorem 3.1.

Communication cost. We first show that the expected number of communicated messages is bounded as asserted. Let $\vartheta = \sqrt{\alpha}/\epsilon \cdot \log n$ and note

$$\mathrm{E}[R_t] = \Pr[|S_t| \leq \vartheta] + \vartheta^2 \mathrm{E}[\frac{1}{S_t^2} I(|S_t| > \vartheta)].$$

Since $\Pr[|S_t| \leq \vartheta] = \Theta(\vartheta/\sqrt{t})$ and $\mathrm{E}[\frac{1}{S_t^2} I(|S_t| > \vartheta)] = \Theta(1/(\vartheta\sqrt{t}))$, it follows

$$\mathrm{E}[R_t] = \Theta(\vartheta/\sqrt{t}).$$

Hence, the expected number of transmitted messages is

$$\sum_{t \leq n} \mathrm{E}[R_t] = O(\vartheta\sqrt{n}) = O(\sqrt{n}/\epsilon \cdot \log n).$$

Correctness. We next establish the asserted bound on the probability of error. Let us write $\mathcal{F}_t$ to be the σ-algebra generated by $X_1, \ldots, X_t$ and $R_1, \ldots, R_t$, i.e. all the information available up to the t-th update is measurable by $\mathcal{F}_t$. Define the indicator variable R_t that sets to 1 if and only if at the t-th update the site sends a message to the coordinator. Notice that our algorithm guarantees that $R_1 = 1$. Let U_t be the number of updates until the next report is sent to the coordinator as observed at the t-th update, i.e. $U_t = \min\{\tau > 0 : R_{t+\tau} = 1\}$. We remark that U_t depends on a future event and thus, it is not measurable by $\mathcal{F}_t$. Next, let $V_t = \min\{\tau > 0 : S_{t+\tau} \notin \mathbf{B}_\epsilon(S_t)\}$ be the number of updates until the first instance at which the coordinator fails to track the counter within the relative accuracy ϵ, and let E_n be the number of such update instances. Notice that a necessary and sufficient condition that at least one error happens is that there exists at least one $t \leq n$ such that $R_t = 1$ and $V_t < U_t$. We thus have

$$\Pr[E_n > 0] = \Pr[R_t = 1 \text{ and } V_t < U_t, \text{ for some } 1 \leq t \leq n],$$

where $I(\cdot)$ is an indicator function that sets to 1 if and only if its parameter is true. By using the union bound, we have

$$\Pr[E_n > 0] \leq \sum_{t \leq n} \mathrm{E}[R_t \cdot I(V_t < U_t)]. \tag{4}$$

Using the fact that R_t is measurable by $\mathcal{F}_t$, we have

$$\begin{aligned}
\mathrm{E}[R_t \cdot I(V_t < U_t)] &= \mathrm{E}_{\mathcal{F}_t}[\mathrm{E}[R_t I(V_t < U_t) \mid \mathcal{F}_t]] \\
&= \mathrm{E}_{\mathcal{F}_t}[R_t \mathrm{E}[I(V_t < U_t) \mid \mathcal{F}_t]] \\
&= \mathrm{E}_{\mathcal{F}_t}[R_t \Pr[V_t < U_t \mid S_t]] \\
&\leq \mathrm{E}_{\mathcal{F}_t}\left[R_t \cdot \max_s \Pr[V_t < U_t \mid S_t = s]\right] \\
&= \mathrm{E}_{\mathcal{F}_t}[R_t] \cdot \max_s \Pr[V_t < U_t \mid S_t = s].
\end{aligned}$$

We next proceed to give an upper bound for $\Pr[V_t < U_t \mid S_t = s]$. Note that for every $r \geq 0$, it holds

$$\begin{aligned}
& \Pr[V_t < U_t \mid S_t = s] \\
= \; & \Pr[V_t < U_t, V_t > r \mid S_t = s] \\
& + \Pr[V_t < U_t, V_t \leq r \mid S_t = s] \\
\leq \; & \Pr[r < U_t \mid S_t = s, V_t > r] + \Pr[V_t \leq r \mid S_t = s]
\end{aligned} \tag{5}$$

We start by giving a bound on $\Pr[V_t \leq r \mid S_t = s]$. Notice that under $S_t = s$ the distribution of V_t is equal to the distribution

of the first passage time of either value $\lceil s/(1-\epsilon)\rceil - s$ or value $\lfloor s/(1+\epsilon)\rfloor - s$ for a symmetric random walk started at the origin. The following lemma follows by standard results from the theory of random walks (c.f. [13]) and the Hoeffding bound:

LEMMA B.1. *For every $r \geq 0$, it holds*

$$\Pr[V_t \leq r \mid S_t = s] \leq 2\exp(-\frac{(\frac{\epsilon}{1-\epsilon})^2 s^2}{2r}). \tag{6}$$

PROOF. Let V_t^+ denote the number of steps until the random walk up-crosses the value $\lceil \frac{s}{1-\epsilon}\rceil$, starting from value s. Similarly, we define V_t^- to be the number of steps until the random walk down-crosses the value $\lfloor \frac{s}{1+\epsilon}\rfloor$ starting from value s. Then,

$$\begin{aligned}\Pr[V_t \leq r | S_t = s] &\leq \Pr[V_t^+ \leq r | S_t = s] + \Pr[V_t^- \leq r | S_t = s] \\ &\leq 2\Pr[V_t^+ \leq r | S_t = s].\end{aligned}$$

Now, let $b = \lceil \frac{s}{1-\epsilon}\rceil - s$ and note that by the reflection principle of random walks, we have

$$\begin{aligned}\Pr[V_t^+ \leq r | S_t = s] &= \Pr[X_1 + X_2 + \cdots + X_r = b] \\ &\quad +2\Pr[X_1 + X_2 + \cdots + X_r > b] \\ &\leq 2\Pr[X_1 + X_2 + \cdots + X_r \geq b].\end{aligned}$$

By applying the Hoeffding's inequality, we bound the the probability in the right-hand side with $\exp(-\frac{b^2}{2r})$ which yields the asserted result. □

From (6), we observe that $\Pr[V_t \leq r \mid S_t = s] \leq 2/n^c$ for given $c > 0$, iff it holds

$$\text{(C1):} \quad r \leq \frac{1}{2c\log n}\left(\frac{\epsilon}{1-\epsilon}\right)^2 s^2.$$

We next note

$$\Pr[r < U_t | S_t = s, V_t > r] \leq (1 - \rho_\epsilon(s))^r \tag{7}$$

where $\rho_\epsilon(s) = \text{Sample-Prob}(s/(1-\epsilon), t)$. Requiring that the right-hand side in the above inequality is less than or equal to $1/n^c$, we obtain

$$\text{(C2):} \quad \rho_\epsilon(s) \geq 1 - \exp(-\frac{c\log n}{(1-\epsilon)^2 r}).$$

Indeed, both conditions (C1) and (C2) hold true by taking $r = \frac{1}{2c\log n}\left(\frac{\epsilon}{1-\epsilon}\right)^2 s^2$ and $\rho_\epsilon(s) = \min\{\frac{2c^2\log^2 n}{(\epsilon s)^2}, 1\}$. The latter choice is a sufficient condition for (C2) in view of the fact that $\min\{x, 1\} \geq 1 - e^{-x}$, for $x \geq 0$. Therefore, we showed that for $\Pr[V_t < U_t | S_t = s] \leq 3/n^c$ to hold, it suffices that the sampling probability satisfies

$$p_t \geq \min\{\frac{2c^2\log n}{\epsilon^2 S_t^2}, 1\}. \tag{8}$$

Combining with (4), we have

$$\begin{aligned}\Pr[E_n > 0] &\leq \sum_{t\leq n} \mathrm{E}[R_t]\cdot O(1/n^c) \\ &= \Theta(\vartheta n^{1/2-c}) = \Theta(n^{1/2-c}\log n).\end{aligned}$$

From the last inequality, we note that no error occurs with high probability provided that $c > 3/2$. Hence, in view of the inequality (8), it suffices to to choose the sampling probability as in (1) with $\alpha = 2c^2 > 9/2$ and $\beta = 2$. This completes the proof.

B.2 Multiple Sites

In this subsection, we prove Theorem 3.2. We need again to show that the algorithm is correct and the communication complexity is as described. We start with showing the correctness part.

Correctness. We will invoke a coupling argument that will allow us to reuse the results of Theorem 3.1. We couple the proposed multiple sites algorithm with the single site sampling algorithm with a different set of error parameters over the same set of input. Specifically, we also execute a single site algorithm with relative accuracy $\epsilon/3$ and success rate $1 - O(1/n^2)$[2], in parallel to the multiple sites algorithm over the same input sequence. We shall couple the random tosses in these two algorithms and show that when the single site algorithm makes no error, our multiple sites algorithm will also make no error.

We need a few more notations. Let $p_{s,i}$ be the sampling rate for the single site algorithm and $R_{s,i}$ be its corresponding Bernoulli random variable. Let $p_{m,i}$ be the sampling rate for the multiple sites algorithm and $R_{m,i}$ be its corresponding Bernoulli random variable. When we are in the straightforward stage, we shall assume $p_{m,i} = 1$. Finally, let $\hat{S}_{s,t}$ be the estimator of the single site algorithm at time t and $\hat{S}_{m,t}$ be the estimator for the multiple sites algorithm.

We couple the Bernoulli random variable in the following way.

- When $p_{s,i} > p_{m,i}$: the two Bernoulli variables are sampled independently.
- When $p_{s,i} \leq p_{m,i}$: if $R_{s,i} = 1$, then we set $R_{m,i} = 1$; otherwise, we set $R_{m,i} = 1$ with probability $(p_{m,i} - p_{s,i})/(1 - p_{s,i})$ and $R_{m,i} = 0$ otherwise. One may see that we still have $\Pr[R_{m,i} = 1] = p_{m,i}$.

Now using the above coupling rules, we show that when the single site makes no error, our multiple sites algorithm also makes no error. Suppose on the contrary that at time t the multiple sites algorithm makes the first error. Then our algorithm ensures that for every $\tau < t$, it holds $p_{m,\tau} \geq p_{s,\tau}$ (by our choice of sampling probabilities), i.e. the multiple sites algorithm samples more frequently than the single site algorithm. Therefore, our coupling rule gives us $\hat{S}_{m,t} = S_{t_1}$ and $\hat{S}_{s,t} = S_{t_2}$, where $t_1 > t_2$, i.e. the multiple sites algorithm is holding a more recent value of the count. We can now get a contradiction using the following arguments,

1. At time t_2, the single site algorithm's estimator is S_{t_1} and is correct. Therefore, $S_{t_1} \in \mathbf{B}_{\epsilon_3}(S_{t_2})$, i.e.
$$|S_{t_2} - S_{t_1}| \leq \frac{\epsilon}{3}|S_{t_2}|. \tag{9}$$
2. At time t, the multiple site algorithm is wrong. Therefore, $S_{t_1} \notin \mathbf{B}_\epsilon(S_t)$, i.e.
$$|S_{t_1} - S_t| > \epsilon|S_t|. \tag{10}$$
3. At time t, the single site algorithm is correct, i.e. $S_{t_2} \in \mathbf{B}_\epsilon(S_t)$. We have $|S_{t_2} - S_t| \leq \epsilon|S_t|$. We can use $\epsilon \leq 1$ to relax this inequality and get
$$|S_{t_2}| \leq 2|S_t|. \tag{11}$$

Using (9) and (10) and a triangle inequality, we have

$$|S_{t_2} - S_t| > \epsilon|S_t| - \frac{\epsilon}{3}|S_{t_2}| \geq \epsilon|S_t| - \frac{2\epsilon}{3}|S_t| \geq \frac{\epsilon}{3}|S_t|. \tag{12}$$

[2]To boost the success rate, we need to use a larger constant in the sampling parameter, i.e. $\text{Sample-Prob}(S_t, t) = \min\left\{\frac{2(1+c)^2\log^2 n}{\epsilon^2\hat{S}_t^2}, 1\right\}$, any $c > 3/2$

The second inequality holds because of (11). (12) implies that the single site algorithm errs at time t, which contradicts with our assumption.

Communication cost. We have the following types of communications,

1. At the straightforward stage, whenever there is an update, $O(1)$ messages are exchanged.
2. At the broadcast stage, whenever there is an update, $O(k)$ messages are exchanged.
3. At the beginning and the end of the broadcasting stage, the coordinator needs to make a broadcast to signal the stage change, which takes $\Theta(k)$ messages.

Notice that in order to change from the broadcast stage to straightforward stage, type 2 messages are sent for at least once. Therefore, the total complexity of the type 3 messages is asymptotically smaller than type 2 messages. We need to only focus on the communication complexity for the first two type of messages.

Let C_t be the communication cost associated with the t-th update and let $R_{m,t}$ indicates the event that a message is sent to the communicator after the t-th update ($R_{m,t}$ shall correspond with R_t in Theorem 3.1). Therefore, when $(\epsilon\hat{S}_t)^2 < k$, $C_t = 1$; otherwise, $\mathrm{E}[C_t] = k\mathrm{E}[R_{m,t}]$. We estimate C_t using the following rule:

- If $(1-\epsilon)(\epsilon S_t)^2 \leq k$, we set $C_t = 1$;
- If $(1+\epsilon)(\epsilon S_t)^2 > k$, we set $C_t = k\mathrm{E}[R_{m,t}]$.

This rule intuitively gives a conservative guess on which stage we are in (conditioned on the estimator being correct). Notice that when $(1-\epsilon)(\epsilon S_t)^2 < k < (1+\epsilon)(\epsilon S_t)^2$, in this case, we can set $C_t = 1 + k\mathrm{E}[R_{m,t}]$ without impacting the asymptotic behavior. The case where our estimator makes an error (and thus the above rules may not give an overestimate of C_t) is an asymptotically smaller term.

We next proceed with computing the expectation of C_t using our overestimation rule,

$$\begin{aligned}\mathrm{E}[C_t] \leq\ & \underbrace{\Pr[S_t \leq \frac{\sqrt{k}}{\epsilon\sqrt{1-\epsilon}}]}_{\text{straightforward stage}} \\ & \underbrace{+ k\mathrm{E}[R_{m,t} I(S_t \geq \frac{\sqrt{k}}{\epsilon\sqrt{1+\epsilon}})]}_{\text{broadcast stage}} \\ & + \underbrace{O(1/n^2)}_{\text{estimator fails}} \\ =\ & O(\tfrac{\sqrt{k}\cdot\log n}{\epsilon\sqrt{t}}).\end{aligned} \tag{13}$$

We can compute the above terms using Theorem 3.1. Thus, the total communication cost in expectation is $O(\sqrt{nk}/\epsilon \cdot \log^2 n)$.

C. COMMUNICATION COMPLEXITY LOWER BOUNDS

In this section we provide proofs for our results on lower bounds on the communication complexity of the continuous distributed counting problem with non-monotonic input stream.

C.1 Proof of Theorem 4.1

Let $\mathcal{E} = \{s \in \mathbb{Z} : |s| \leq 1/\epsilon\}$. Our crucial observation here is that whenever S_t walks inside the region $\mathcal{E}$ we have $\epsilon|S_t| < 1$ and no errors are allowed. Specifically, let I_t be the indicator random variable that sets to 1 if and only if $S_t \in \mathcal{E}$. Notice that $\mathrm{E}[I_t] = \Pr[S_t \in \mathcal{E}] = \Omega(|\mathcal{E}|/\sqrt{t}) = \Omega(1/(\sqrt{t}\epsilon))$ and $\mathrm{E}[\sum_{t\leq n} I_t] = \Theta(\min\{\sqrt{n}/\epsilon, n\})$. On the other hand, our error requirement gives us $\Pr[M_n \geq \sum_{t\leq n} I_t] \geq 1 - 1/n$. We can then derive $\mathrm{E}[M_n]$ from $\mathrm{E}[\sum_{t\leq n} I_t]$ using the following argument. Let $\mathcal{A}$ be the subset of the probability space where $M_n \geq \sum_{t\leq n} I_t$ and let $\neg\mathcal{A}$ be the subset where this does not hold. We have

$$\begin{aligned}\mathrm{E}[M_n] &\geq \int_{\mathcal{A}} M_n \mathrm{d}F \geq \int_{\mathcal{A}} \sum_{t\leq n} I_t \mathrm{d}F \\ &= \mathrm{E}[\sum_{t\leq n} I_t] - \int_{\neg\mathcal{A}} \sum_{t\leq n} I_t \mathrm{d}F \geq \mathrm{E}[\sum_{t\leq n} I_t] - 1\end{aligned}$$

where the last equality follows from the facts that $\sum_{t\leq n} I_t \leq n$ by construction, and that $\int_{\neg\mathcal{A}} \mathrm{d}F \leq 1/n$.

C.2 Proof of Theorem 4.2

The proof is by direct analysis of the probability of event $S_t \in \mathcal{E} = \{s \in Z : |s| \leq 1/\epsilon\}$, where the distribution of S_t is given by

$$\Pr[S_t = s] = \binom{t}{\frac{t+s}{2}} p^{\frac{t+s}{2}} (1-p)^{\frac{t-s}{2}}.$$

We remark that in the proof it is implicitly assumed that p, μ and ϵ are sequences indexed with n, but we omit to make this explicit in the notation for simplicity of presentation.

For convenience, we introduce the notation $\sigma^2 = \mathrm{Var}[X_1] = 4p(1-p)$ and let $\rho = \sqrt{\frac{p}{1-p}}$. We then have

$$\Pr[S_t = s] = \sigma^t \frac{1}{2^t} \binom{t}{\frac{t+s}{2}} \rho^s.$$

Since $\frac{1}{2^t}\binom{t}{\frac{t+s}{2}} = \sqrt{\frac{2}{\pi}}\frac{1}{\sqrt{t}}$ for $s = o(\sqrt{t})$, we have

$$\Pr[S_t = s] = \sqrt{\frac{2}{\pi}} \frac{1}{\sqrt{t}} \sigma^t \rho^s \cdot [1 + o(1)], \text{ for } s = o(\sqrt{t}).$$

In order to simplify the notation and with a slight abuse in the remainder of the proof we omit to write the factor $[1 + o(1)]$.

Let $\theta_0 \geq 0$ and $\theta_1 \geq 0$ be such that $|\theta_0| = o(\sqrt{t})$ and $\theta_1 = o(\sqrt{t})$ and consider $\Pr[S_t \in [-\theta_0, \theta_1]]$, for $t = 1, 2, \ldots, n$. For $1/2 < p < 1$ and $s = o(\sqrt{t})$, we have

$$\begin{aligned}\Pr[S_t \in [-\theta_0, \theta_1]] &= \sqrt{\frac{2}{\pi}} \frac{1}{\sqrt{t}} \sigma^t \sum_{s=-\theta_0}^{\theta_1} \rho^s \\ &= \sqrt{\frac{2}{\pi}} \frac{1}{\sqrt{t}} \sigma^t \left(\frac{\rho^{\theta_1+1} - 1 + \rho^{\theta_0+1} - 1}{\rho - 1} - 1 \right).\end{aligned}$$

Let $E_n[-\theta_0, \theta_1]$ denote the number of visits of the set $[-\theta_0, \theta_1]$ by the counter S_t and let $\tau_n = \omega(\max\{\theta_0, \theta_1\})$. Then, note

$$\begin{aligned}\mathrm{E}[E_n[-\theta_0, \theta_1]] &\geq \sum_{t=\tau_n}^{n} \Pr[S_t \in [-\theta_0, \theta_1]] \\ &= \left(\frac{\rho^{\theta_1+1} - 1 + \rho^{\theta_0+1} - 1}{\rho - 1} - 1 \right) \cdot \sqrt{\frac{2}{\pi}} \sum_{t=\tau_n}^{n} \frac{1}{\sqrt{t}} \sigma^t.\end{aligned}$$

Notice that for every $c > 0$,

$$\begin{aligned}\sum_{t=\tau_n}^{n} \frac{1}{\sqrt{t}} e^{-ct} &= \int_{\tau_n}^{n} \frac{1}{\sqrt{t}} e^{-ct} dt \geq \int_{2\sqrt{\tau_n}}^{2\sqrt{n}} e^{-\frac{c}{4}u^2} du \\ &= 2\sqrt{\frac{\pi}{c}} [\Phi(\sqrt{2cn}) - \Phi(\sqrt{2c\tau_n})]\end{aligned}$$

where Φ is the distribution of a standard normal random variable.

Therefore,

$$\mathrm{E}[E_n[-\theta_0,\theta_1]] \geq \frac{4a_nb_n}{\log^{1/2}(\frac{1}{\sigma^2})} \tag{14}$$

where

$$\begin{aligned} a_n &= \frac{\rho^{\theta_1+1}-1+\rho^{\theta_0+1}-1}{\rho-1}-1 \\ b_n &= \Phi(\log^{1/2}(\frac{1}{\sigma^2})\sqrt{n})-\Phi(\log^{1/2}(\frac{1}{\sigma^2})\sqrt{\tau_n}). \end{aligned}$$

Now, we consider the case of a small but non-zero drift $\mu = p-(1-p) = o(1)$ and $\theta_0 = \theta_1 = 1/\epsilon$ where $1/\epsilon$ is a positive integer. We will assume that $\tau_n = o(n)$ and $\tau_n = \omega(1/\epsilon^2)$, thus $\epsilon = \omega(1/\sqrt{n})$.

It is straightforward to show that the following asymptotes hold:

$$\begin{aligned} \rho &= 1+\mu+O(\mu^2) \\ \rho-1 &= \mu+O(\mu^2) \\ \sigma^2 &= 1-\mu^2 \\ \log(\frac{1}{\sigma^2}) &= \mu^2+O(\mu^3) \end{aligned}$$

For the term a_n, it holds

$$a_n = 2\frac{\rho^{\frac{1}{\epsilon}+1}-1}{\rho-1}-1 = \frac{2}{\mu}(e^{\frac{\mu}{\epsilon}}-1)\cdot[1+o(1)].$$

Hence, $a_n = \Theta(1/\mu)$, for $\mu = O(\epsilon)$. Notice that for the case $\epsilon = o(\mu)$, a_n grows as $\Omega(e^{\mu/\epsilon})$.

For the term b_n, we observe

$$\begin{aligned} b_n &= \Phi(\log^{1/2}(\frac{1}{\sigma^2})\sqrt{n})-\Phi(\log^{1/2}(\frac{1}{\sigma^2})\sqrt{\tau_n}) \\ &= [\Phi(\mu\sqrt{n})-\Phi(\mu\sqrt{\tau_n})]\cdot[1+o(1)] \end{aligned}$$

and is easy to derive that $b_n = \Theta(1)$, for $\mu = O(1/\sqrt{n})$ and $b_n = \Theta(\mu\sqrt{n})$ for $\mu = o(1/\sqrt{n})$. Indeed, these are easy to derive from the above asymptotes and the facts $\Phi(\mu\sqrt{n})-\Phi(\mu\sqrt{\tau_n}) = 1-1/2 = 1/2$ for $\mu = \omega(1/\sqrt{n})$ and $\Phi(\mu\sqrt{n})-\Phi(\mu\sqrt{\tau_n}) \geq \frac{1}{\sqrt{2\pi}}e^{-\frac{\mu^2 n}{2}}\mu(\sqrt{n}-\sqrt{\tau_n})$.

The assertion of the theorem follows by plugging the derived asymptotes for a_n, b_n and $\log^{1/2}(1/\sigma^2)) = \mu[1+o(1)]$ into (14).

C.3 Proof of Lemma 4.4

The proof in this section follows that of Lemma 2.2 in [12]. Here, we only need to argue that the communication lower bound still holds for a two round *deterministic* protocol such that

- in the first round, a subset of sites report their individual values to the coordinator;
- in the second round, the coordinator probes a subset of sites to make the decision.

The lemma follows in view of the fact that a randomized protocol can be seen as a distribution over a set of deterministic algorithms. It suffices to consider the case where $o(k)$ messages are sent in the first round, as otherwise we are done with the proof. This essentially reduces the communication complexity problem to a known sampling complexity problem [12]. The only remaining obstacle here is that the input distribution under our consideration is not exactly the same as the one studied in [12]. Therefore, we need to re-establish the sampling lower bound in our setting, which is provided in the remainder of this section.

Let $k' = \Theta(k)$ be the number of sites that have not sent any messages in the first round, and without loss of generality, assume that these sites are $1,2,\ldots,k'$. Since the number of messages sent in the first round is $o(k)$, in the second round, we need to solve a problem that is at least as hard as the following one:

- answer whether the sum $\sum_{i\leq k} X_i$ is positive or negative, if $|\sum_{i\leq k'} X_i| \geq c\sqrt{k'}$;
- do anything (i.e. no requirement) when $|\sum_{i\leq k'} X_i| < c\sqrt{k'}$

where c is a positive constant.

Let us denote with z the number of sites that are sampled by the coordinator in the second round, and without loss of generality, let us assume that these sites are $1,2,\ldots,z$. To contradict, let us suppose $z = o(k')$. Let $N = \sum_{i\leq z} X_i$ be the cumulative update value of the sampled sites and $U = \sum_{z<i\leq k'} X_i$ be the cumulative update value of the unsampled sites. Clearly, the optimal detection algorithm for the sampling problem is to declare $\sum_{i\leq k'} X_i > c\sqrt{k'}$ if $N > 0$, to declare $\sum_{i\leq k'} X_i < -c\sqrt{k'}$ if $N < 0$ and to declare either (with probability $1/2$) if $N = 0$. The probability of error is then

$$\begin{aligned} \Pr[\text{error}] &\geq \Pr[N<0, N+U \geq c\sqrt{k'}] \\ &\geq \Pr[-c\sqrt{z} \leq N < 0]\Pr[U \geq c(\sqrt{k'}+\sqrt{z})]. \end{aligned}$$

Since N is a sum of independent and identically distributed random variables of mean zero and variance 1, we have $\mathrm{E}[N] = 0$ and $\mathrm{Var}[N] = z$, and thus $\Pr[-c\sqrt{z} \leq N < 0] = \Theta(1)$. Similarly, since U is a sum of independent and identically random variables of mean zero and variance 1, we have $\mathrm{E}[U] = 0$ and $\mathrm{Var}[U] = k'-z$, and under our assumption $z = o(k')$, it holds $c(\sqrt{k'}+\sqrt{z}) = c\sqrt{k'-z}\cdot[1+o(1)] = c\mathrm{Var}[U]\cdot[1+o(1)]$, and thus $\Pr[U \geq c(\sqrt{k'}+\sqrt{z})] = \Theta(1)$. Therefore, the probability of error is $\Omega(1)$ which contradicts the error requirement, for sufficiently small constant c_0 in the statement of the lemma.

C.4 Proof of Theorem 4.5

We partition the updates into n/k phases, each of which consists of k updates. In each phase, the k updates are randomly matched to k sites (so that each site receives exactly one update). Let I_j be an indicator random variable that sets to 1 when, at the beginning of the jth phase, the sum is in the interval $[-a_{j,k,\epsilon}, a_{j,k,\epsilon}]$ where $a_{j,k,\epsilon} \triangleq \min\{\sqrt{k}/\epsilon, \sqrt{jk}\}$. Notice that when the sum is in the interval $[-a_{j,k,\epsilon}, a_{j,k,\epsilon}]$, the additive error we can tolerate is at most $\epsilon\sqrt{k}/\epsilon = \sqrt{k}$. Therefore, at the end of the jth stage, the tracking algorithm has to be able to tell whether the absolute value of j-th phase's sum is below $-\sqrt{k}$, above $\sqrt{k}$, or in between the two. This is at least as difficult as the *tracking k inputs* problem we studied above, with $\Theta(k)$ communication lower bound.

Let M_n be the total number of messages exchanged between the coordinator and the sites. Our correctness requirement gives us $\Pr[M_n \geq \Omega(k\sum_{i\leq n/k} I_i)] \geq 1-1/n$. Using the fact that $\mathrm{E}[\sum_i I_i] = \min\{\sqrt{n/(\epsilon k)}, n/k\}$, and following similar arguments as in the proof of Theorem 4.1, we get $\Omega(\min\{\sqrt{kn}/\epsilon, n\})$.

C.5 Proof of Corollary 4.6

Consider an adversary that select each input a'_t randomly such that $\Pr[a'_t = 1] = \Pr[a'_t = -1] = 1/2$. Then the process (a_t) obtained by randomly permuting (a'_t) is also a sequence of Bernoulli variables, and from Theorem 4.5 we know that $\mathrm{E}[M_n] \geq \Omega(\sqrt{kn}/\epsilon)$. Clearly, using an averaging argument, there is at least one deterministic sequence a'_t that, randomly permuted, requires on average $\Omega(\sqrt{kn}/\epsilon)$ messages. This proves the claim.