L_p Sampling in Distributed Data Streams with Applications to Adversarial Robustness

Honghao Lin
Carnegie Mellon University
honghaol@andrew.cmu.edu

Zhao Song
UC Berkeley
magic.linuxkde@gmail.com

David P. Woodruff
Carnegie Mellon University
dwoodruf@andrew.cmu.edu

Shenghao Xie
Texas A&M University
xsh1302@gmail.com

Samson Zhou
Texas A&M University
samsonzhou@gmail.com

Abstract

In the distributed monitoring model, a data stream over a universe of size n is distributed over k servers, who must continuously provide certain statistics of the overall dataset, while minimizing communication with a central coordinator. In such settings, the ability to efficiently collect a random sample from the global stream is a powerful primitive, enabling a wide array of downstream tasks such as estimating frequency moments, detecting heavy hitters, or performing sparse recovery. Of particular interest is the task of producing a perfect L_p sample, which given a frequency vector $f \in \mathbb{R}^n$, outputs an index i with probability $\frac{f_i^p}{\|f\|_p^p} + \frac{1}{\text{poly}(n)}$. In this paper, we resolve the problem of perfect L_p sampling for all $p \geq 1$ in the distributed monitoring model. Specifically, our algorithm runs in $k^{p-1} \cdot \text{polylog}(n)$ bits of communication, which is optimal up to polylogarithmic factors.

Utilizing our perfect L_p sampler, we achieve adversarially-robust distributed monitoring protocols for the F_p moment estimation problem, where the goal is to provide a $(1+\varepsilon)$-approximation to $f_1^p + \ldots + f_n^p$. Our algorithm uses $\frac{k^{p-1}}{\varepsilon^2} \cdot \text{polylog}(n)$ bits of communication for all $p \geq 2$ and achieves optimal bounds up to polylogarithmic factors, matching lower bounds by Woodruff and Zhang (STOC 2012) in the non-robust setting. Finally, we apply our framework to achieve near-optimal adversarially robust distributed protocols for central problems such as counting, frequency estimation, heavy-hitters, and distinct element estimation.

1 Introduction

As modern applications often require evolving data to be collected and stored across multiple servers, the *distributed monitoring model* has emerged as an increasingly important setting. In this model, there exist k servers and a central coordinator, with two-way communication links between the coordinator and each server. The network iteratively receives updates so that at each time t, server $j \in [k]$ has a dataset $f^{(t)}(j)$. The goal of the coordinator is to aggregate statistics of the overall dataset, which comprises of the union of the k datasets $f^{(t)}(1), \ldots, f^{(t)}(k)$ at each time t, while minimizing the total amount of communication between the sites and the servers, e.g., due to power constraints in sensors networks [EGHK99, JOW+02, MFHH05], or bandwidth constraints in distributed systems [RG06a, RG06b]. Motivated by its applications to the increasing prevalence of distributed systems, the distributed monitoring model has recently received significant study [DR02, BO03, CGMR05, KCR06, ABC09, CMY11, TW11, CMYZ12, HYZ12, WZ12, YZ13, CZ17, WGZ20].

Perfect L_p sampler. Sampling is a powerful and widely used technique for analyzing large-scale datasets, particularly in streaming and distributed settings [Vit85, GLH08, CDK+11, CDK+14]. By selecting a small representative subset of the data, sampling enables efficient computation and communication, and thus serves as a fundamental tool in applications such as data summarization [Vit85, GLH08, CDK+11, CDK+14], network traffic monitoring [GKMS01, EV03, MCS+06, HNG+07, TLJ10], database management [HS92, LN95, GM98, Haa16, CG19], and data summarization [FKV04, DRVW06, ADK09, IMMM14, MIGR19, IMGR20, MRWZ20, CPW20]. In this paper, we consider the task of sampling in the distributed monitoring model, where a distributed data stream is processed across k servers. At each time step t, an update $i_t \in [n]$ arrives at some server $j_t \in [k]$. Each server j maintains a local frequency vector $f^{(t)}(j) \in \mathbb{R}^n$, where $f_i^{(t)}(j)$ is the number of times i has appeared in the data stream local to server j up to time t. These collectively induce the global frequency vector $f^{(t)} = \sum_{j \in [k]} f^{(t)}(j)$. The goal is to sample an index $i \in [n]$ with probability proportional to some fixed weight function $G(x_i)$ and possibly obtain an approximation of the value x_i. Among the most impactful and extensively studied choices for the function $G(\cdot)$ is the family of functions $G(z) = |z|^p$ for $p > 0$, referred to as an L_p *sampler*, which have been used in tasks such as identifying heavy hitters [WZ21b, JWZ24, WXZ25], low-rank approximation [DRVW06, DV06, MRWZ20], L_p norm and F_p moment estimation [WZ21b, JWZ24, WXZ25], finding duplicates [JST11], mini-batch sampling in stochastic gradient descent [MWZ22], cascaded norm approximation [JW09], and data summarization, projective clustering, and volume maximization [DRVW06, DV07, ÇM09, IMMM14, MIGR19, IMGR20, MRWZ20]. Hence, an actively growing body of work has studied L_p sampling in the streaming model and its variants [MW10, AKO11, JST11, BOZ12, JW18, CPW20, JWZ22, SWZ25, WXZ25]. We recall the definition of L_p sampler, formalized to the distributed monitoring setting as follows:

Definition 1.1 (Continuous L_p sampler)**.** *At each time $t \in [m]$, a continuous L_p sampler returns an index $\mathbf{i}^*$, such that for all $j \in [n]$, we have*

$$\mathbf{Pr}\left[\mathbf{i}^* = j\right] = (1 \pm \varepsilon) \cdot \frac{(f_i^{(t)})^p}{\|f^{(t)}\|_p^p} \pm n^{-C},$$

where $f^{(t)} \in \mathbb{R}^n$ is the frequency vector at each time t and $C > 0$ can be chosen to be any constant. When $\varepsilon = 0$, the sampler is said to be perfect*; otherwise, it is said to be* approximate*.*

Due to the distortions in the sampling probabilities, approximate L_p samplers can cause an ε-fractional bias, which may compound over the course of more intricate algorithms built upon these samplers, causing downstream inaccuracies. Perfect L_p samplers reduce these issues by ensuring small total variation distance from the desired joint distribution across even a polynomial number of samples.

This is especially important in privacy-preserving applications, where a sampled index $i \in [n]$ is exposed to an external adversary $\mathcal{A}$, with the goal of limiting leakage of global information about the dataset. It is possible for an approximate sampler to bias the sampling probabilities for a large set S of indices by $(1+\varepsilon)$ if a certain global property $\mathcal{P}$ holds, and instead bias the sampling probabilities of S by $(1-\varepsilon)$ if $\mathcal{P}$ does not hold. In this case, $\mathcal{A}$ can deduce $\mathcal{P}$ from $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ samples. By contrast, perfect L_p samplers can protect such global properties even with the release of a number of samples polynomial in n.

Surprisingly, despite a long line of research in the distributed monitoring model, previous works have only been able to achieve perfect L_p samplers in this setting for $p = 1$ [JSTW19]. However, their techniques rely on the additive nature of the L_1 norm and do not generalize to other regimes of p. In fact, to the best of our knowledge, there do not even exist constructions of *approximate* L_p samplers for $p \neq 1$ that do not use non-trivial communication. This gap raises a fundamental question:

> Is it possible to design continuous perfect L_p samplers using total communication sublinear in the length of the data stream?

1.1 Our Contributions

In this paper, we resolve the above question in the affirmative, introducing the first perfect L_p sampler in the distributed monitoring setting.

Theorem 1.2. *Given a distributed stream of length $m = \text{poly}(n)$ on a universe of size n across k servers, there exists a continuous perfect L_p sampler for $p \geq 1$ that uses $k^{\max(1,p-1)} \cdot \text{polylog}(n)$ bits of communication.*

We emphasize that by comparison, previous constructions were not even able to achieve *approximate* L_p samplers for $p \neq 1$ using $o(m)$ communication, whereas we are able to achieve perfect L_p samplers. Thus our work represents a substantial contribution across the total communication of the protocol, the distortion of the L_p sampling probabilities, and the regime of p. Moreover, our L_p sampler can be implemented to also output an unbiased estimate to $f_{\mathbf{i}^*}^{(t)}$ for the sampled index $\mathbf{i}^*$ at each time in the stream, with variance at most $(f_{\mathbf{i}^*}^{(t)})^2$. We give a lower bound showing that any perfect L_p sampler for a static dataset distributed across k servers must necessarily use $\Omega(k^{p-1})$ bits of communication.

Theorem 1.3. *Any perfect L_p sampler for $p \geq 1$ must use $\Omega(k^{\max(1,p-1)})$ bits of communication.*

Hence, Theorem 1.3 shows that our continuous perfect L_p sampler in Theorem 1.2 is optimal up to $\text{polylog}(n)$ factors. Given its role as a core algorithmic primitive, our result enables a range of new protocols in the distributed monitoring model, which we now describe.

Moment estimation and adversarial robustness. A particular application where L_p sampling is often used is F_p moment estimation, where the goal is to output a $(1+\varepsilon)$-approximation to $\|f^{(t)}\|_p^p := \sum_{i\in[n]}(f_i^{(t)})^p$. The F_p moment estimation problem has a large number of applications. For $p \in [1,2]$, it has been used for computer networking in the context of using entropy estimation to identify port scanning attacks [FKSV02, LCD05, XZB05, WZ21b] and to measure the entropy of origin-destination pairs [ZLO+07, HNO08]. For $p = 2$, it has been used to estimate join and self-join sizes [AGMS02] and to detect network anomalies [KSZC03, TZ04]. Furthermore, the frequency moment estimation problem is closely related to the norm estimation problem, where the goal is to compute a $(1+\varepsilon)$-approximation to $\|f\|_p$, as a $(1+\varepsilon)$-approximation algorithm to one of these problems can be used to obtain a $(1+\varepsilon)$-approximation to the other for constant $p > 0$, after an appropriate rescaling of ε. In particular, larger values of p provide a smooth interpolation toward L_∞, which captures the maximum frequency and is useful for identifying the most dominant elements in a dataset. Thus, beginning with the seminal work of [AMS99], there has been a rich body of literature studying F_p moment estimation in various sublinear algorithm models, e.g., [Ind00, CKS03, BJKS04, Woo04, IW05, Li08, KNW10, KNPW11, AKO11, CMY11, WZ12, KVW14, BDN17, WZ21a, WZ21b, EKM+24, JWZ24].

Recent works observed that in many big data applications, future inputs may not necessarily be independent of previous outputs, e.g., in repeated interactions with a database, future queries may depend on answers to previous queries. As a result, [BEJWY20] introduced the adversarially robust streaming model, which has garnered a significant amount of attention [BY20, HKM+20, ABD+21, BHM+21, KMNS21, WZ21b, ABJ+22, BKM+22, BEO22, BJWY22, CGS22, AMYZ19, ACGS23, ACSS23, CSW+23, DSWZ23, GLW+24, WZ24, GLW+25, AHKO25, CSS25, CXWZ25] and was recently adapted by [XZH23] to the distributed monitoring setting. The model can be captured by the following two-player game between a distributed monitoring protocol $\mathcal{P}$ and a source $\mathcal{A}$ of adaptive or adversarial input to $\mathcal{P}$. After a fixed query function $\mathcal{Q}$ is given, the game proceeds over m rounds, and in the t-th round:

(1) The source $\mathcal{A}$ computes a pair $s_t = (u_t, i_t)$, where $u_t \in [n]$ and $i_t \in [k]$, which increments coordinate u_t at server i_t.

(2) The protocol $\mathcal{P}$ processes (u_t, i_t) and outputs its current answer Z_t.

(3) The source $\mathcal{A}$ observes and records the response Z_t.

The goal of the protocol $\mathcal{P}$ is to produce a correct answer Z_t to the query function $\mathcal{Q}$ on the dataset induced by the adaptive distributed stream $\{s_1, \ldots, s_t\}$ induced by the source $\mathcal{A}$, across all times $t \in [m]$. [XZH23] observed that the existing analysis of many randomized distributed monitoring protocols fail because the randomness is assumed to be independent of the input, an assumption that no longer due to the adaptive nature of the input. In fact, [XZH23] was only able to achieve a robust protocol for distributed counting, using a novel formulation of partial differential privacy and significant technical analysis.

We use our continuous perfect L_p sampler to achieve an adversarially robust distributed monitoring protocol for F_p estimation:

Theorem 1.4. *Given a distributed stream of length $m = \mathrm{poly}(n)$ on a universe of size n across k servers, a parameter $p \geq 2$, and an accuracy parameter $\varepsilon \in (0,1)$, there exists an adversarially robust algorithm that outputs a $(1+\varepsilon)$-approximation to the F_p moment at all times, using $\frac{k^{p-1}}{\varepsilon^2} \cdot \mathrm{polylog}(n)$ bits of communication.*

Theorem 1.4 is the first result to achieve adversarial robustness for F_p moment estimation in the distributed monitoring setting. Moreover, we remark that in the one-shot setting, where each server has a static vector, it is known that $\Omega\left(\frac{k^{p-1}}{\varepsilon^2}\right)$ bits of communication is necessary [EKM+24]. Therefore, not only is Theorem 1.4 optimal up to polylogarithmic factors, but it shows that perhaps surprisingly, there is no overhead in either the $\frac{1}{\varepsilon}$ or the k factors to achieve adversarial robustness.

Crucially, we achieve Theorem 1.4 by using our perfect L_p sampler to design a difference estimator for F_p moment estimation. For the purposes of moment estimation, a difference estimator is an algorithmic primitive that achieves an estimate of $F_p(\mathbf{u}+\mathbf{v}) - F_p(\mathbf{u})$ with additive error $\varepsilon \cdot F_p(\mathbf{u})$, given two frequency vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, under the promise that $F_p(\mathbf{u}+\mathbf{v}) - F_p(\mathbf{u}) \leq \gamma \cdot F_p(\mathbf{u})$. Difference estimators are useful because, due to the upper bound on the difference, it is sometimes possible to achieve more efficient protocols based on the value of γ. However, their constructions and analysis are often intricate, e.g., the design of efficient constructions for F_p moment estimation in the streaming model for non-integer $p > 2$ currently remains to be an open question [WZ21b]. Thus, our techniques may be of independent interest.

Along the way, we develop a framework for integrating our difference estimator toward achieving adversarial robustness in the distributed monitoring model. In addition to F_p moment estimation, our framework can also be used to achieve robust protocols for the counting problem, i.e., $p = 1$, the distinct element estimation problem, i.e., $p = 0$, and the L_2-heavy hitters problem:

Theorem 1.5. *Given an accuracy parameter $\varepsilon \in (0,1)$, there exists an adversarially robust distributed monitoring protocol that achieves an $(1+\varepsilon)$-approximation to:*

(1) The distinct elements problem, i.e., F_0 estimation, using $\frac{k}{\varepsilon^2} \cdot \text{polylog}(n)$ bits of communication.

(2) The counting problem, i.e., F_1 estimation, using $\left(k + \frac{\sqrt{k}}{\varepsilon}\right) \cdot \text{polylog}(n)$ bits of communication.

(3) L_2-heavy hitters, using $\frac{k}{\varepsilon^2} \cdot \text{polylog}(n)$ bits of communication.

We remark that previous works have shown communication lower bounds of $\Omega\left(\frac{k}{\varepsilon^2}\right)$ bits for distinct element estimation [WZ12], $\Omega\left(k + \frac{\sqrt{k}}{\varepsilon}\log n\right)$ bits for counting [HYZ12], and $\Omega\left(\frac{k}{\varepsilon^2}\right)$ bits for L_2-heavy hitters. Thus our results in Theorem 1.5 conceptually show that there is no overhead to achieve adversarial robustness for these central problems, up to polylogarithmic factors. Our framework is quite general, utilizing difference estimators and existing non-robust distributed monitoring protocols in a black-box and intuitive manner, and thus may be amenable to a number of additional other problems. Previously, there were no known non-trivial robust distributed monitoring protocols for distinct element estimation and L_2-heavy hitters. For the problem of counting, our robust algorithm matches the communication cost of the protocol of [XZH23], though our algorithm and analysis may arguably be much simpler. We summarize these results in Figure 1.

Comparison with [HXZW25]. In concurrent and independent work, [HXZW25] recently also achieved a distributed monitoring protocol for F_p moment estimation using total communication $\frac{k^{p-1}}{\varepsilon^2} \cdot \text{polylog}(n)$ for $p \geq 2$, which matches our result in Theorem 1.4, but for *non-adaptive* distributed streams. Their protocol modifies a scheme previously used by [IW05, WZ12, BO13] by carefully tracking the frequencies of heavy-hitters from substreams induced by subsampling the universe $[n]$ at different levels using the notion of (α, ε)-covers. Both their algorithm and analysis are quite elegant and significantly simpler than our result in Theorem 1.3. However, as their approach fundamentally

Problem	LB	Prior non-robust UB	Our Robust UB
Distinct elements	$\tilde{\Omega}\left(\frac{k}{\varepsilon^2}\right)$ [WZ12, WZ14]	$\tilde{\mathcal{O}}\left(\frac{k}{\varepsilon^2}\right)$ [CMY11]	$\tilde{\mathcal{O}}\left(\frac{k}{\varepsilon^2}\right)$
Count-tracking	$\tilde{\Omega}\left(k+\frac{\sqrt{k}}{\varepsilon}\right)$ [HYZ12]	$\tilde{\mathcal{O}}\left(k+\frac{\sqrt{k}}{\varepsilon}\right)$ [HYZ12]	$\tilde{\mathcal{O}}\left(k+\frac{\sqrt{k}}{\varepsilon}\right)$
F_p estimation, $p \geq 2$	$\tilde{\Omega}\left(\frac{k^{p-1}}{\varepsilon^2}\right)$ [WZ12]	$\tilde{\mathcal{O}}\left(\frac{k^{p-1}}{\varepsilon^2}\right)$ [HXZW25]	$\tilde{\mathcal{O}}\left(\frac{k^{p-1}}{\varepsilon^2}\right)$
L_2 heavy-hitters	$\tilde{\Omega}\left(\frac{k}{\varepsilon^2}\right)$ [WZ12]	$\tilde{\mathcal{O}}\left(\frac{k}{\varepsilon^2}\right)$ [WZ12]	$\tilde{\mathcal{O}}\left(\frac{k}{\varepsilon^2}\right)$

Fig. 1: Comparison between non-robust and robust $(1+\varepsilon)$-approximation algorithms for distributed monitoring. Here, $\tilde{\mathcal{O}}$ and $\tilde{\Omega}$ hide the polylog(n/ε) factors. **UB** denotes the upper bound and **LB** denotes the lower bound. Our results are from Theorem 1.5.

relies on heavy-hitter detection, they do not address the design of perfect L_p samplers, which is the core contribution of our work in Theorem 1.2. Moreover, the ability to perform perfect L_p sampling is essential to our difference estimator, which enables adversarial robustness in Theorem 1.4. It is not clear whether their non-robust, heavy-hitter-based techniques can be extended to yield adversarial robustness for F_p moment estimation especially given that, prior to this work, no existing approach could achieve adversarial robustness for heavy-hitters in the distributed monitoring model. Thus, although our results overlap in achieving similar bounds for non-robust moment estimation, our work is fundamentally different, both technically and conceptually, due to its focus on perfect L_p sampling and its applications to adversarial robustness.

1.2 Technical Overview

In this section, we describe the intuition behind our algorithms and techniques, and how we overcome significant barriers for previous approaches. In Section 1.2.1, we introduce our perfect L_p sampler. In Section 1.2.2, we describe the difference estimator framework. Lastly in Section 1.2.3, we provide the F_p difference estimator, which is essentially based on our L_p sampler.

1.2.1 L_p Sampling

A natural starting point might be the perfect L_1 sampler from [JSTW19] in the distributed monitoring model. However, at a high level, sampling is significantly easier due to the additivity of L_1. Thus it is not clear how to extend this approach to $p \neq 1$. L_p sampling has also been well-studied in the streaming model, where the data stream is given in its entirety to an algorithm that is bounded in space and cannot store the entire dataset [MW10, AKO11, JST11, BOZ12, JW18, CPW20, JWZ22, SWZ25, WXZ25]. Unfortunately, these approaches do not directly extend to the distributed monitoring model, where informing the coordinator of each update incurs communication cost linear in the length of the stream, and moreover, only [WXZ25] achieves near-optimal perfect L_p sampling for $p \geq 2$. On the other hand, the universal template for L_p sampling on an input vector $f \in \mathbb{R}^n$ is to (1) perform some linear transformation on f to obtain a scaled vector g, (2) estimate each entry of g and find the maximum index i^* of the estimated vector, (3) run a statistical test to decide whether to report i^*. [AKO11, JW18, WXZ25] observed that if we define $g_i = \frac{f_i}{\mathbf{e}_i^{1/p}}$ for all $i \in [n]$, where each $\mathbf{e}_i$ is an independent exponential random variable with rate 1, then

the probability of i^* being the maximum index of g is precisely the L_p sampling probability (see Definition 1.1), i.e.,

$$\mathbf{Pr}\left[i^* = \operatorname*{argmax}_i g_i\right] = \frac{f_i^p}{\|f\|_p^p},$$

which follows the desired distribution. Therefore, it suffices to continuously track the largest index of g at all times in the distributed stream. Unfortunately, this turns out to be a difficult task in general; even for a universe of size $n = 2$, tracking the maximum frequency at all times on a stream of length m provably requires $\Omega(m)$ communication in the worst case. Circumventing this issue forms the majority of both our algorithmic novelty and technical analysis for our perfect L_p sampler.

Global L_1-of-L_p sampling. Due to distributional properties of exponential random variables, the largest index of g is a L_p-heavy-hitter. Then suppose that there is a procedure that identifies the set of L_p-heavy-hitters at all times, it is our hope that we can keep track of the items in this set with a higher accuracy, to recover or approximate the maximum. To begin with, we introduce the construction of this L_p-heavy-hitter algorithm.

For ease of notation, we fix $t \in [m]$ and simply write $g_i = g_{i,t}$. It is known that with high probability, the largest rescaled coordinate g_i is a $\mathcal{O}\left(\frac{1}{\log^{2/p} n}\right)$-$L_p$ heavy-hitter with respect to g, i.e., $\max_{i\in[n]} g_i^p \geq \mathcal{O}\left(\frac{1}{\log^2 n}\right) \cdot \|g\|_p^p$. Now, consider the following hard distribution. Let $g_n(j)$ be the value of i at server j, where $n = \operatorname{argmax}_{i\in[n]} g_i$ without loss of generality. Suppose $g_n(j) = 1$ for all $j \in [k]$ so that $g_n = k$. Suppose that for each server j, we have $g_{(j-1)\cdot k^{p-1}+1}(j) = \ldots = g_{j\cdot k^{p-1}}(j) = 1$, i.e., each server has a single instance of coordinate n, as well as k^{p-1} additional disjoint coordinates so that $\|g\|_p^p = k^p + k \cdot (k^{p-1}) = 2k^p$. Then n is a $\mathcal{O}(1)$-L_p heavy and could be the largest coordinate after a possible rescaling of f, so we must identify g_n in this case.

Observe that in this case, the maximizer is a $\frac{1}{k^{p-1}}$-L_p heavy-hitter at each site. Suppose that for each coordinate $i \in [n]$ and server $j \in [k]$, we sample $g_i(j)$ with the *global L_1 of L_p* probability distribution $\min\left(k^{p-1} \cdot \frac{(g_i(j))^p}{\sum_{a\in[k]}\sum_{b\in[n]}(g_b(a))^p}, 1\right)$. In the above example, each of the k servers will report the maximizer with probability roughly $\frac{1}{k}$ and thus the maximizer will be reported with probability roughly $1 - \left(1 - \frac{1}{k}\right)^k = \Omega(1)$ across the k servers. Moreover, under this sampling distribution, k^{p-1} samples would be returned in expectation and thus it can be shown that over the course of the stream, the total communication is at most $k^{p-1} \cdot \operatorname{polylog}(n)$, as desired.

In fact, after the sampling procedure, there exists a reconstruction algorithm that estimates the value of each scaled coordinate g_i, eliminating all but at most $\operatorname{polylog}(n)$ candidate coordinates from possibly being the maximizer $\operatorname{argmax}_{i\in[n]} g_i$. In other words, for a fixed instance t, we can identify a set $\mathbf{PL}_t \subset [n]$ (standing for "probably large") with at most $\operatorname{polylog}(n)$ coordinates such that $\operatorname{argmax}_{i\in[n]} g_{i,t} \in \mathbf{PL}_t$.

Statistical test and dependence on the maximum index. The task is then to extract the maximizer from $\mathbf{PL}_t$ at all times, which essentially requires continuously estimating the frequency of each item $g_i \in \mathbf{PL}_t$. A natural approach would be to apply the continuous counting algorithm of [HYZ12], which will provide an unbiased estimate to each $g_i \in \mathbf{PL}_t$ with variance $\mathcal{O}\left(g_i^2\right)$. However, the algorithm only gives a constant-multiplicative approximation to each g_i; and as we mentioned earlier, it is impossible to obtain the exact value at all times. Fortunately, due to distributional properties of the exponential variables, the maximum of the scaled vector g is at least twice the

second-maximum with constant probability. Given this observation, [JW18, WXZ25] performed a statistical test to accept the instances with this desired guarantee to provide an L_p sample.

Formally, let $\widehat{g_{D(1)}}$ and $\widehat{g_{D(2)}}$ be 1.01-approximations of the maximum and the second-maximum, we accept a sample if and only if $\widehat{g_{D(1)}} \geq 2 \cdot \widehat{g_{D(2)}}$. However, the algorithm of [HYZ12] maintains an approximate counter for each local count $g_i(j)$ at the coordinator, which sums up all approximate counters; and as a result, the estimate depends on how g_i is distributed across the k servers. For example, if all updates go to a single server, the error is expected to be significantly smaller than that in the case that the updates are uniformly distributed across the k servers. But then the statistical test is conducted on the estimated value, so the failure probability of the test depends on the distribution of the maximum coordinate across the k servers. Therefore, the probability that we fail a sampler depends on which index i^* achieves the max in the scaled vector, which destroys the perfect L_p sampling probability distribution because each index may have a different failure probability and thus, we cannot apply the continuous counting algorithm of [HYZ12]. Hence, we need to design a novel distributed counting algorithm, whose estimation error is independent of how the coordinate is distributed across the k servers.

Double exponential-scaling method. To that end, we instead scale each $g_i(j)$ by another independent exponential variable $\mathbf{e}(j)$ and define $h_i(j) := \frac{g_i(j)}{\mathbf{e}(j)} = \frac{f_i(j)}{\mathbf{e}_i^{1/p}\mathbf{e}(j)}$. Due to the max-stability property of the exponential variables, $\max_j h_i(j)$ has the same distribution as $\frac{g_i}{\mathbf{e}_i'}$, where $\mathbf{e}_i'$ is independent of all $\mathbf{e}(j)$. Thus, we run a separate counting algorithm at each server for each $h_i(j)$, such that we can use the estimate of a single approximate counter that reaches the max, instead of the sum of all approximate counters. In this way, the second exponential scaling washes away dependencies, so that the resulting estimate does not depend on the distribution of g_i across all servers. We remark that a subsequent work [CSS25] showed that random sampling also gives an efficient distributed counting algorithm, such that the estimate does not dependent on the distribution across the servers; it is possible that this approach also gives the functionality required from our subroutine.

Amortizing the communication by anti-concentration. It is unclear how we control the communication cost of the aforementioned counting algorithm, because we must extract the *actual* max across the k servers to ensure the distributional-free property. In particular, let X be the maximum and let X' be the second-maximum of the scaled vector $[h_i(1), \cdots, h_i(k)]$. If the ratio $\frac{X}{X'} = 1 + \alpha$, then we run a counting algorithm with accuracy $\frac{\alpha}{100}$ at each server j, which outputs a $(1 + \frac{\alpha}{100})$-approximation to $h_i(j)$ at each j, so we recover the server j that achieves the max. This might require roughly $\frac{k}{\alpha}$ communication if X and X' are always α-close, which is prohibitively large for our purpose. Fortunately, due to the anti-concentration property of exponential variables, we have $\mathbf{Pr}\left[\frac{X}{X'} < 1 + \alpha\right] = \mathcal{O}(\alpha)$. Then, since the counting algorithm requires $\mathcal{O}\left(\frac{1}{\alpha}\right)$ communication, we only need roughly $\mathcal{O}(\alpha) \cdot k \cdot \frac{1}{\alpha} = \mathcal{O}(k)$ communication in expectation. On the other hand, we run a separate counting algorithm at each server j that outputs a 1.01-approximation $\widehat{h_i(j)}$ at all times. If j^* is detected to be the maximum server, then we take $\widehat{h_i(j^*)}$ as the estimate of X. In summary, although finding the maximum value of a frequency vector $f \in \mathbb{R}^n$ defined by a distributed stream of length m requires $\Omega(m)$ communication in general, this can only happen when the largest and second largest coordinates of the frequency vector are quite close. Critically, the anti-concentration property shows this can only happen a small fraction of times, which allows us to amortize the

additional communication cost of dealing with such occurrences over the entirety of the stream.

Geometric mean estimator and truncated Taylor series. Now that we obtain an unbiased estimate of $X = \max_j h_i(j) = \frac{g_i}{\mathbf{e}_i'}$ with constant-multiplicative error. The remaining task is to estimate g_i using X. Unfortunately, both the expectation and the variance of X can be unbounded. This is because the cumulative distribution function of a random variable $Z \sim \frac{1}{\mathbf{e}^{1/p}}$ for an exponential random variable $\mathbf{e}$ is roughly $\mathbf{Pr}\left[Z \leq t\right] = 1 - \Theta\left(\frac{1}{t^p}\right)$ for $t \geq 1$, and so the expectation of Z is unbounded for $p \leq 1$ and the variance of Z is unbounded for $p \leq 2$. On the other hand, we have $\mathbf{Pr}\left[Z^{1/3} \geq t\right] = \Theta\left(\frac{1}{t^{3p}}\right)$, and so both the expectation and variance of $Z^{1/3}$ are finite for $p \geq 1$. Hence if we take the geometric mean $Y = \sqrt[3]{X_1 \cdot X_2 \cdot X_3}$ of three independent instances X_1, X_2, and X_3, then there exists a constant C such that $\mathbb{E}\left[C \cdot Y\right] = g_i$ and $\mathbb{E}\left[C^2 \cdot Y^2\right] \leq \mathcal{O}\left(g_i^2\right)$, which gives us an unbiased estimate of g_i. However, it is not instantly clear how to provide a valid estimator for $X_r^{1/3}$, because the fractional power of $\frac{1}{3}$ might bring up undesirable covariance terms. We use a truncated Taylor series to estimate each $X_r^{1/3}$, which requires $\mathcal{O}\left(\log n\right)$ independent unbiased estimates of X_r. Namely, we expand $X_r^{1/3}$ by $\sum_q \alpha_q X_r^q$, so that it is easier to obtain unbiased estimates of X_r^q for an integer q. Then, averaging $\operatorname{polylog} n$ instances of Y gives us a 1.01-approximation to g_i. Thus, we can track the value for each g_i in **PL** and report the sample if it passes the statistical test. Although geometric mean estimators have been used for p-stable random variables when $p < 2$ [Li08, WZ21b], their use to upper bound the variance of inverse exponential random variables appears to be new. We believe this technique is of independent interest and may find broader applications. In fact, the combination of geometric mean estimator and truncated Taylor estimator is applied again in our subroutine to give unbiased estimate of the F_p moment $\|f\|_p^p$ (see Section 1.2.3).

Core insight. The underlying input in the distributed monitoring model can be captured by an $n \times k$ matrix $\mathbf{A}$, where the rows of the matrix are the coordinates $i \in [n]$ and the columns are the servers $j \in [k]$. Then the perfect L_p sampling problems involves computing the p-th powers of the L_1 norm of the rows, similar to more complex cascaded norms such as the $L_{p,q}$ norm by

$$\|\mathbf{A}\|_{p,q} = \left(\sum_{i \in [n]} \left(\sum_{j \in [d]} A_{i,j}\right)^{p/q}\right)^{1/p},$$

i.e., the L_p norm of the vector whose entries are the L_q norms of the rows of $\mathbf{A}$. In this setting, our two-sided embedding can be viewed as computing the matrix $\mathbf{E}_1\mathbf{A}\mathbf{E}_2$, where $\mathbf{E}_1 \in \mathbb{R}^{n \times n}$ is a diagonal matrix so that each entry is $\frac{1}{\mathbf{e}_i^{1/p}}$ for an independent exponential random variable $\mathbf{e}_i$ and $\mathbf{E}_2 \in \mathbb{R}^{k \times k}$ is a diagonal matrix where each entry is $\frac{1}{\mathbf{e}(j)^{1/q}}$ for an independent exponential random variable $\mathbf{e}(j)$, with $q = 1$ in our setting. Thus our main idea can be viewed as a two-sided non-negative embedding that reduces these more complicated quantities to a two-level heavy-hitters problem through properties of exponential random variables. The heavy-hitters in each level can be maintained and approximated. Moreover, although finding the maximum coordinate is difficult in general, exponential random variables enjoy anti-concentration properties so that at most α fraction of the instances have their maximum and second maximum values separated by less than $(1 + \alpha)$, which allows us to amortize the communication. On the other hand, our two-sided embedding leads

to values with unbounded expectation but we can address this by averaging independent instances of geometric mean estimators. Given this perspective, we believe our approach could potentially extend to more general samplers based on cascaded norms.

1.2.2 Difference Estimator Framework and Adversarial Robustness

Previous work introduced several effective frameworks to achieve adversarially robust streaming algorithms in the streaming model, where efficient space complexity is prioritized. [BEJWY20] provided the well-known sketch switching framework, which initiates several sketches simultaneously, and switches to a new sketch whenever the estimate increases by an ε-fraction. This framework achieves adversarially robustness because, the algorithm fixes the output until it switches to a new sketch, and so the inner randomness of each sketch is hidden from the adversary before its revelation. However, this framework loses a $\frac{1}{\varepsilon}$ factor in the space complexity, so its extension to the distributed monitoring setting would also lose a $\frac{1}{\varepsilon}$ factor in the communication cost.

To achieve tight bound in both k and $\frac{1}{\varepsilon}$ factors, we adapt the difference estimator framework introduced by [WZ21b] to the context of distributed monitoring. Firstly, we run a "double detector" algorithm, which flags when the target function value F doubles and gives a $(1+\varepsilon)$-approximation to F whenever it flags. So, this divides the algorithm into $\mathcal{O}(\log n)$ rounds, and for a vector u when the double detector flags, our goal is to estimate $F(u+v)-F(u)$ for a subsequent vector v, where $F(u+v)\leq 2F(u)$.

The main idea is that if $F(u+v)-F(u)$ is small, say roughly $\varepsilon\cdot F(u)$, then we only need a $\mathcal{O}(1)$-approximation to $F(u+v)-F(u)$, which still ensures an additive error of $\varepsilon\cdot F(u)$, instead of losing $\frac{1}{\varepsilon}$ to obtain a $(1+\varepsilon)$-approximation. Similarly, if $F(u+v)-F(u)$ is roughly $2\varepsilon\cdot F(u)$, then we only need a $(1+\frac{1}{2})$-approximation to $F(u+v)-F(u)$; and more generally, if $F(u+v)-F(u)$ is roughly $\frac{1}{2^r}\cdot F(u)$, then we only need a $(1+2^r\varepsilon)$-approximation to $F(u+v)-F(u)$.

Now, for $\beta=\mathcal{O}\left(\log\frac{1}{\varepsilon}\right)$ and $F(u+v)\leq 2F(u)$, we decompose $F(u+v)-F(u)$ into the binary expression

$$F(u+v)-F(u)=\sum_{r=1}^{\beta}F(u+v_{\leq r})-F(u+v_{\leq r-1}),$$

where we define $v_{\leq t}:=\sum_{j=0}^{t}v_t$ and $v_1,\ldots,v_\beta$ arrives in sequential order, and each difference satisfies $F(u+v_{\leq r})-F(u+v_{\leq r-1})\approx\frac{1}{2^r}\cdot F(u)$. Thus, it suffices to give a $\left(1+\frac{2^r\varepsilon}{\beta}\right)$-approximation for each difference, which we define to be a level-r difference estimator informally. We implement a separate difference estimator when $F(u+v)$ grows to the corresponding level and sum up the estimation of all levels to give a $(1+\varepsilon)$-approximation. Since a level-r difference estimator is used each time the difference increases by $\frac{1}{2^r}\cdot F(u)$, it suffices to run roughly 2^r instances of the level-r difference estimator. Thus, if there exists a level-r difference estimator which uses $\frac{1}{2^r\varepsilon^2}G(n,k)$ bits of communication for each $r\in[\beta]$, then in total we use $\frac{1}{\varepsilon^2}G(n,k)$ bits of communication in total for each level. Note that $\beta=\mathcal{O}\left(\log\frac{1}{\varepsilon}\right)$, so we only lose a multiplicative $\mathcal{O}\left(\log\frac{1}{\varepsilon}\right)$ factor in the total communication cost.

We observe that the difference estimator framework also achieves adversarially robustness. This can be done by using fresh randomness for each difference estimator and not changing the output until the estimated value increases by $(1+\varepsilon)$ multiplicatively. Therefore, when the estimate is revealed to the adversary, we start a new difference estimator at level-β with hidden randomness, so the adaptive input stream from the adversary is independent of any internal randomness in the new

difference estimator, guaranteeing adversarial robustness. We refer the reader to Section 3.2 for a more detailed outline for the difference estimator framework.

1.2.3 Adversarially Robust F_p Estimation, $p \geq 2$

With the difference estimator framework, it remains to construct a F_p difference estimator, so that it can be lifted to an adversarially robust F_p estimation algorithm.

Sampling-based method. We denote the difference as $S = F_p(u+v) - F_p(u) = \|u+v\|_p^p - \|u\|_p^p$, with the guarantee $S < \gamma \cdot \|u\|_p^p$. The core idea of our difference estimator is to sample an index $i \in [n]$ with respect to its importance in the difference term S, and construct the estimator by rescaling the sampling probability and estimating the entry-wise difference $(u_i + v_i)^p - u_i^p$. Specifically, let p_i be our sampling probability. Then, we define our estimator to be

$$\widehat{s} = \frac{1}{p_i} \cdot ((u_i + v_i)^p - u_i^p).$$

It is obvious that $\widehat{s}$ is an unbiased estimation of S. Thus, our goal is to develop a sampling distribution with sufficiently small variance while bounding the total communication by $\frac{\gamma k^{p-1}}{\varepsilon^2}$ polylog(n) bits.

Mixed L_p sampling. Intuitively, we should give larger sampling probabilities to coordinates i that have large values of $(u_i + v_i)^p - u_i^p$, which implies that either u_i is large or v_i is large. Thus, a natural approach would be to split the indices into sets where $u_i > v_i$ and $u_i \leq v_i$, and L_p sample from u for indices in the first set, and L_p sample from v for indices in the second set. In practice, we could L_p sample some number of indices from both u and v and reject certain samples that are from the wrong set, e.g., if we acquire a coordinate $i \in [n]$ by L_p sampling from u and it turns out that $u_i \leq v_i$, or vice versa. However, we cannot afford to acquire the real value of v_i at all times. Rather, we can only run a distributed counting algorithm to estimate v_i with constant-multiplicative error, but then we could misclassify the index and incorrectly reject/accept some samples, which can give substantially large bias to the resulting estimator. Unfortunately, these simple misclassifications are known to generally require suboptimal dependencies to handle the resulting error, e.g., [IW05, WZ12, LSW18, BBC+17, WZ21a]. Additionally, another concern is that we could reject too many samples so that we need prohibitively large number of independent instances to ensure a sufficiently large number of successful samples.

Instead, we sample from a mixed L_p distribution. That is, with probability $\frac{1}{2}$, we obtain an L_p sample from a fixed vector u, otherwise we obtain an L_p sample from an evolving vector v, so that our sampling probability is $\frac{u_i^p}{2\|u\|_p^p} + \frac{v_i^p}{2\|v\|_p^p}$, thereby sampling the "important" indices from both u and v that contribute significantly to the difference. This construction bypasses the undesirable misclassification and rejection procedure and ensures a small variance via a careful analysis.

Bi-variate Taylor estimator. Then we need to give unbiased estimations for $\frac{1}{p_i}$ and $(u_i+v_i)^p - u_i^p$ with bounded variance. However, the main challenge with this sampling distribution is that we need to estimate

$$\frac{1}{p_i} = \left(\frac{u_i^p}{2\|u\|_p^p} + \frac{v_i^p}{2\|v\|_p^p} \right)^{-1} = \frac{2\|u\|_p^p \|v\|_p^p}{\|u\|_p^p v_i^p + \|v\|_p^p u_i^p},$$

up to high accuracy. We can query the servers collectively for the value of u_i, but it is particularly challenging to deal with the denominator, since $\|u\|_p^p$ and $\|v\|_p^p$ may require querying a large number of coordinates, and v_i^p may be evolving over the duration of the stream. To handle this, we define $a = \|u\|_p^p v_i^p$ and $b = \|v\|_p^p u_i^p$ and consider $\frac{1}{a+b}$. We first suppose that we have good estimators for both a and b; we shall subsequently discuss the construction of the estimators. Then, we utilize the bivariate Taylor expansion

$$f(a,b) = f(a_0,b_0) + \sum_{q=1}^{\infty} \sum_{\substack{d,e\geq 0 \\ d+e=q}} \frac{1}{d!\,e!} \frac{\partial^q f(x,y)}{\partial x^d \partial y^e}\Big|_{(x,y)=(a_0,b_0)} (x-a_0)^d (y-b_0)^e,$$

of the function $f(a,b) = \frac{1}{a+b}$ expanded at a_0, b_0, which are $\left(1+\frac{1}{100p}\right)$-estimations of a and b. Then, to achieve an approximation to $f(a,b)$ with $\frac{1}{\mathrm{poly}(n)}$-bias, it suffices to truncate the Taylor series after $Q = \mathcal{O}(\log n)$ terms and thus we define our truncated bivariate Taylor estimator to be

$$\sum_{q=0}^{Q} \frac{(-1)^q}{(a_0+b_0)^{q+1}} \cdot ((a-a_0)+(b-b_0))^q,$$

and we estimate the power $((a-a_0)+(b-b_0))^q$ in each term by multiplying q independent estimators. Here, we remark that the truncated bivariate Taylor estimator is valid because the partial derivatives of $f(a,b) = \frac{1}{a+b}$ with respect to a and b have the same expression; the tail error bound may not be clear if the partial derivatives are different. On the other hand, it is an 2-dimensional extension of the truncated Taylor estimator applied in our perfect L_p sampler.

Unbiased estimation of $\|u\|_p^p$. It remains to show the estimator for a and b. Here, the most challenging part is to give unbiased estimations of $\|u\|_p^p$ and $\|v\|_p^p$. We consider the simpler case of $\|u\|_p^p$ first. Notice that u is a fixed vector, so we can obtain the actual value of each u_i by querying each server, however, we still do not afford to retrieve the actual value of $\|u\|_p^p$. Applying the max-stability property of exponential variables stated in Section 1.2.1, we show that if we scale each coordinate in the original vector u by $g_i = \frac{u_i}{\mathbf{e}_i^{1/p}}$, then we have $\max_i g_i^p = \frac{\|u\|_p^p}{\mathbf{e}}$, where $\mathbf{e}$ is an independent exponential variable. Similar as our perfect L_p sampler, we still need to deal with the problem that $\frac{1}{\mathbf{e}}$ has an unbounded expectation, which fails to give an unbiased estimator. Again, we use the idea of geometric mean estimator defined by the geometric mean of three independent instances $X = \max_{i\in[n]} \frac{u_i}{\mathbf{e}_i^{1/p}}$:

$$Y = \sqrt[3]{X_1^p \cdot X_2^p \cdot X_3^p},$$

which provides an unbiased estimate to $\|u\|_p^p$. For the evolving vector v, we could also use the geometric mean estimator, but we are not able to recover the maximum coordinate at all times, instead we only have access to an unbiased estimate of X. This requires us to apply the truncated Taylor estimator to accurately approximate the geometric mean.

Estimation of the entry-wise difference $(u_i+v_i)^p - u_i^p$. For the estimation of $(u_i+v_i)^p - u_i^p$, it is natural to think of using binomial series to estimate $(u_i+v_i)^p$, e.g., [WZ21b] uses the finite binomial series to estimate the difference for integer p in the non-distributed streaming setting. However, a crucial drawback of binomial series is that we need to decide the dominant variable. This

means that, if $u_i > v_i$, then we can only transform $(u_i + v_i)^p$ to $u_i^p \cdot (1 + \frac{v_i}{u_i})^p$ and expand $(1 + \frac{v_i}{u_i})^p$. We cannot do it the other way or the binomial series would be divergent. As we mentioned before, we do not have access to the real value of v_i, which leads to misclassification error that could cause our algorithm to fail if we use the binomial series.

On the other hand, notice that we can compute the actual value of u_i, so instead it is intuitive to apply our truncated Taylor estimator again. Let $f(v_i) = (u_i + v_i)^p - u_i^p$, where we consider u_i as a known value. Then, we define the truncated Taylor estimator to be $((u_i + s_i)^p - u_i^p) + \sum_{q=1}^{Q} \binom{p}{q}(u_i + s_i)^{p-q}(v_i - s_i)^q$, with s_i being a $(1 + \frac{1}{100p})$-approximation of v_i.

With the above subroutines, we give unbiased estimates of the mixed L_p sampling-based estimator $\widehat{s} = \frac{1}{p_i}((u_i + v_i)^p - u_i^p$. Since our estimate has small variance, averaging across several independent instances provides a valid F_p difference estimator. Then, applying the difference estimator framework in Section 1.2.2, we achieve an adversarially robust algorithm for distributed F_p estimation.

In summary, our continuous perfect L_p sampler appears to be an essential procedure in constructing the F_p different estimator. The mixed-L_p sampler provides a reasonable distribution that captures the important coordinates in both u and v with respect to the difference. In addition, the formula of the sampling probability allows efficient unbiased estimation to its inverse, so that we can average across independent instances to reduce the variance.

1.3 Preliminaries

For an integer $n > 0$, we use the notation $[n]$ to denote the set $\{1, \ldots, n\}$. We use $\mathrm{poly}(n)$ to denote a fixed polynomial of n, whose degree can be determined by setting parameters in the algorithm accordingly. We use $\mathrm{polylog}(n)$ to denote $\mathrm{poly}(\log n)$. We use the notation $\tilde{\mathcal{O}}(f)$ for a function $f(\cdot, \ldots, \cdot)$ to denote $f \cdot \mathrm{polylog}(n/\varepsilon)$.

If an event occurs with probability $1 - 1/\mathrm{poly}(n)$, then we say the event occurs with high probability. For a vector $f \in \mathbb{R}^n$, we define

$$\|f\|_p = (f_1^p + \ldots + f_n^p)^{1/p}.$$

The coordinator model of distributed computing. We work in the asynchronous coordinator model, which was introduced by [DF92]. There are k servers, one of which acts as a special coordinator. Each server has access to private randomness and communicates with the coordinator over a private, bidirectional channel; no direct communication occurs between servers. Communication is asynchronous and message-based: messages on each channel must alternate between the coordinator and the server and must be self-delimiting, i.e., both parties must be able to determine when a message has ended. Each server $i \in [k]$ can determine from its transcript Π_i with the coordinator whether it is their turn to speak, since the previous message must have come from the coordinator. When it is the coordinator's turn, it may send messages to any subset of servers. At the end of the protocol, the coordinator must output an answer.

The communication cost of the protocol $\Pi = \{\Pi_1, \ldots, \Pi_k\}$ is the total number of bits sent across all channels coordinator outputs the answer. Thus we assume without loss of generality that the protocol is sequential and round-based, i.e., in each round the coordinator speaks to some number of players and await their responses before initiating the next round.

Our protocol actually works in the more restricted setting where each vector $f(j) \in \mathbb{R}^n$ at server $j \in [k]$ arrives in a data stream and each server must use space sublinear in n to store some sketch of $f(j)$. In this case, each element of the data stream consists of an update to a single coordinator

of a single server, which will be specified only to the appropriate server. After the data stream is completed, the servers run the protocol Π given only the sketches maintained by each server j, rather than given the entirety of $f(j)$.

Exponential random variables. Exponential random variables are a key part of our protocols. Recall the following definition for an exponential random variable:

Definition 1.6 (Exponential random variables)**.** *If* $\mathbf{e}$ *is a exponential random variable with scale parameter* $\lambda > 0$*, then the probability density function for* $\mathbf{e}$ *is*

$$p(x) = \lambda e^{-\lambda x}.$$

We say $\mathbf{e}$ *is a standard exponential random variable if* $\lambda = 1$*.*

We first state the following tail bounds for exponential random variables.

Fact 1.7. *Let* $\mathbf{e}$ *be a standard exponential random variable. Then for any* $a, b \geq 0$*,*

$$\Pr[\mathbf{e} \geq a \log n] = \frac{1}{n^a}, \quad \Pr[\mathbf{e} \leq b] \leq b.$$

We next recall that the largest scaled item is a polylogarithmic heavy-hitter with high probability.

Fact 1.8 (c.f., Lemma 2.1 in [EKM+24])**.** *Let* $\mathbf{e}_1, \ldots, \mathbf{e}_n$ *be standard exponential random variables, let* $\alpha_1, \ldots, \alpha_n \geq 0$*, and let* $C > 0$ *be a fixed constant. Then*

$$\Pr\Big[\frac{\max_{i\in[n]} \alpha_i/\mathbf{e}_i}{\sum_{i=1}^n \alpha_i/\mathbf{e}_i} \geq \frac{1}{C\log^2 n}\Big] \geq 1 - \frac{1}{\mathrm{poly}(n)}.$$

We next recall that scaling an exponential random variable inversely scales its rate parameter:

Fact 1.9 (Scaling of exponentials)**.** *Let* t *be exponentially distributed with rate* λ*, and let* $\alpha > 0$*. Then* αt *is exponentially distributed with rate* λ/α*.*

We now define the anti-rank vector, which is a vector consisting of the order statistics.

Definition 1.10 (Anti-rank vector)**.** *Let* $(t_1, \ldots, t_n)$ *be independent exponentials. For* $k = 1, 2, \ldots, n$*, we define the* k*-th anti-rank* $D(k) \in [n]$ *of* $(t_1, \ldots, t_n)$ *to be the values* $D(k)$ *such that* $t_{D(1)} \leq t_{D(2)} \leq \cdots \leq t_{D(n)}$*.*

Using the structure of the anti-rank vector, [Nag06] introduces a simple form for describing the distribution of $t_{D(k)}$ as a function of $(\lambda_1, \ldots, \lambda_n)$ and the anti-rank vector.

Fact 1.11 ([Nag06])**.** *For any* $i = 1, 2, \ldots, n$*, we have*

$$\Pr[D(1) = i] = \frac{\lambda_i}{\sum_{j=1}^n \lambda_j}$$

Fact 1.12 ([Nag06])**.** *Let* $(\mathbf{e}_1, \ldots, \mathbf{e}_n)$ *be independently distributed exponentials, where* $\mathbf{e}_i$ *has rate* $\lambda_i > 0$*. Then for any* $k = 1, 2, \ldots, n$*, we have*

$$\mathbf{e}_{D(k)} = \sum_{i=1}^{k} \frac{E_i}{\sum_{j=i}^{n} \lambda_{D(j)}},$$

where $E_1, E_2, \ldots, E_n$ *are i.i.d. exponential variables with mean* 1*, and are independent of the anti-rank vector* $(D(1), D(2), \ldots, D(n))$*.*

With the above results, we are able to describe the characterization to each entry in the scaled vector.

Corollary 1.13. *Let* $f \in \mathbb{R}^n$ *be a frequency vector. Let* $(\mathbf{e}_1, \ldots, \mathbf{e}_n)$ *be i.i.d. exponential random variables with rate* 1*. We define* $z_i = f_i / \mathbf{e}_i^{1/p}$*. Let* $(D(1), \ldots, D(n))$ *be the anti-rank vector of the vector* $\left(z_1^{-p}, \cdots z_n^{-p}\right)$*, we have*

$$\mathbf{Pr}\left[D(1) = i\right] = \frac{|f_i|^p}{\|f\|_p^p}$$

As a result, the probability that $|z_i| = \arg\max_j \{|z_j|\}$ *is precisely* $|f_i|^p / \|f\|_p^p$*, so for a perfect* L_p *sampler it suffices to return* $i \in [n]$ *with* $|z_i|$ *maximum. Moreover, we have*

$$z_{D(k)} = \left(\sum_{i=1}^{k} \frac{E_i}{\sum_{j=i}^{N} f_{D(j)}^p} \right)^{-1/p},$$

where E_i *'s are i.i.d. exponential random variables with mean* 1*, and are independent of the anti-rank vector* $(D(1), \ldots, D(n))$*. Specifically, we have*

$$\max_{i \in [n]} \frac{f_i}{\mathbf{e}_i} = z_{D(1)} = \frac{\|f\|_p}{E^{1/p}}.$$

Proof. By Fact 1.9, the variable $z_i^{-p} = \mathbf{e}_i / f_i^p$ is exponentially distributed with rate $\lambda_i = f_i^p$. By Fact 1.11, we have,

$$\mathbf{Pr}\left[D(1) = i\right] = \mathbf{Pr}\left[i = \arg\min \left\{ |z_1|^{-p}, \ldots, |z_n|^{-p} \right\}\right] = \frac{|f_i|^p}{\|f\|_p^p}.$$

Moreover, by Fact 1.12, we have,

$$z_{D(k)} = \left(\sum_{i=1}^{k} \frac{E_i}{\sum_{j=i}^{N} \lambda_{D(j)}} \right)^{-1/p} = \left(\sum_{i=1}^{k} \frac{E_i}{\sum_{j=i}^{N} f_{D(j)}^p} \right)^{-1/p},$$

where E_i 's are i.i.d. exponential random variables with mean 1, and are independent of the anti-rank vector $(D(1), \ldots, D(n))$. □

We state the anti-concentration property of exponential random variables, which means that there is a gap between the first max and the second max of the scaled vector.

Lemma 1.14 (Anti-concentration)**.** *Let $f \in \mathbb{R}^n$ be a frequency vector. Let $(\mathbf{e}_1, \ldots, \mathbf{e}_n)$ be i.i.d. exponential random variables with rate 1. We define $z_i = f_i/\mathbf{e}_i^{1/p}$. Let $(D(1), \ldots, D(n))$ be the anti-rank vector of the vector $\left(z_1^{-p}, \cdots z_n^{-p}\right)$. For parameter $\varepsilon \in (0, 1]$ we have*

$$\mathbf{Pr}\left[\frac{z_{D(1)}}{z_{D(2)}} \leq 1 + \varepsilon\right] = \mathcal{O}(\varepsilon).$$

Proof. By Corollary 1.13, let $F := \|f\|_p^p$ we have $z_{D(1)}^p = \frac{F}{\mathbf{e}_1}$ and $z_{D(2)}^p = \left(\frac{\mathbf{e}_1}{F} + \frac{\mathbf{e}_2}{F - f_{D(1)}^p}\right)^{-1}$, where $\mathbf{e}_1$ and $\mathbf{e}_2$ are independent variables. Then, we have

$$z_{D(2)}^p = \left(\frac{\mathbf{e}_1}{F} + \frac{\mathbf{e}_2}{F - f_{D(1)}^p}\right)^{-1} \leq \left(\frac{\mathbf{e}_1}{F} + \frac{\mathbf{e}_2}{F}\right)^{-1} = \frac{F}{\mathbf{e}_1 + \mathbf{e}_2}.$$

Thus, we have

$$\mathbf{Pr}\left[\frac{z_{D(1)}^p}{z_{D(2)}^p} \leq 1 + \varepsilon\right] \leq \mathbf{Pr}\left[\mathbf{e}_2 \leq \varepsilon \cdot \mathbf{e}_1\right].$$

Now, due to the pdf of exponential variables, we have

$$\begin{aligned}
\mathbf{Pr}\left[\mathbf{e}_2 \leq \varepsilon \cdot \mathbf{e}_1\right] &= \int_0^\infty e^{-s} \int_0^{\varepsilon s} e^{-t} dt ds \\
&= \int_0^\infty e^{-s}(1 - e^{-\varepsilon s}) ds \\
&= -e^{-s} \Big|_0^\infty + \frac{1}{1+\varepsilon} e^{-(1+\varepsilon)s} \Big|_0^\infty \leq \varepsilon.
\end{aligned}$$

Therefore, we have,

$$\mathbf{Pr}\left[\frac{z_{D(1)}^p}{z_{D(2)}^p} \leq 1 + \varepsilon\right] \leq \varepsilon.$$

For parameter $\varepsilon \in (0, 1]$ and constant p, we have

$$\mathbf{Pr}\left[\frac{z_{D(1)}}{z_{D(2)}} \leq 1 + \varepsilon\right] \leq \mathcal{O}(\varepsilon),$$

as desired. □

Next, we prove that the geometric mean of three exponential random variables has bounded variance, which will be used in our geometric mean estimator.

Lemma 1.15 (Geometric mean)**.** *Let $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ be independent exponential random variables with rate 1 and let $X_1 = \frac{1}{\mathbf{e}_1}$, $X_2 = \frac{1}{\mathbf{e}_2}$, and $X_3 = \frac{1}{\mathbf{e}_3}$. Let $X = \sqrt[3]{X_1 \cdot X_2 \cdot X_3}$. Then $\mathbb{E}[X] = C_{1.15}$ for some constant $C_{1.15} \in (0, 8]$ and $\mathbb{E}[X^2] \leq 27$.*

Proof. Since $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \geq 0$, then we trivially have $\mathbb{E}[X] \geq 0$ and $\mathbb{E}[X^2] \geq 0$. Since X_1, X_2, and X_3 are independent random variables, then we have

$$\mathbb{E}[X] = \mathbb{E}\left[X_1^{1/3}\right] \cdot \mathbb{E}\left[X_2^{1/3}\right] \cdot \mathbb{E}\left[X_3^{1/3}\right].$$

For $i \in \{1,2,3\}$, we have

$$\mathbb{E}\left[X_i^{1/3}\right] = \int_0^\infty \mathbf{Pr}\left[X_i > t^3\right]\, dt.$$

As $\mathbf{e}_i$ is an exponential random variable, then the probability density function of X_i is $p(t) = \frac{1}{t^2} \cdot e^{-\frac{1}{t}}$. Hence, we have $\mathbf{Pr}\left[X_i > t\right] \leq \frac{1}{t}$ so that

$$\begin{aligned}\mathbb{E}\left[X_i^{1/3}\right] &= \int_0^\infty \mathbf{Pr}\left[X > t^3\right]\, dt \\ &= \int_0^1 \mathbf{Pr}\left[X_i > t^3\right]\, dt + \int_1^\infty \mathbf{Pr}\left[X_i > t^3\right]\, dt\end{aligned}$$

Now, for the first term, it satisfies $\int_0^1 \mathbf{Pr}\left[X_i > t^3\right]\, dt \leq 1$. Moreover, for the second term, we have

$$\int_1^\infty \mathbf{Pr}\left[X_i > t^3\right]\, dt \leq \int_1^\infty \frac{1}{t^3}\, dt < 1,$$

and so $\mathbb{E}\left[X\right] \leq 8$. Similarly, we have

$$\mathbb{E}\left[X^2\right] = \mathbb{E}\left[X_1^{2/3}\right] \cdot \mathbb{E}\left[X_2^{2/3}\right] \cdot \mathbb{E}\left[X_3^{2/3}\right].$$

Moreover, for $i \in \{1,2,3\}$, we have

$$\begin{aligned}\mathbb{E}\left[X_i^{2/3}\right] &= \int_0^\infty \mathbf{Pr}\left[X_i > t^{3/2}\right]\, dt \\ &= \int_0^1 \mathbf{Pr}\left[X_i > t^{3/2}\right]\, dt + \int_1^\infty \mathbf{Pr}\left[X_i > t^{3/2}\right]\, dt\end{aligned}$$

Again, for the first term, it satisfies $\int_0^1 \mathbf{Pr}\left[X_i > t^3\right]\, dt \leq 1$. Moreover, for the second term, we have

$$\int_1^\infty \mathbf{Pr}\left[X_i > t^{3/2}\right]\, dt \leq \int_1^\infty \frac{1}{t^{3/2}}\, dt \leq 3,$$

and so $\mathbb{E}\left[X^2\right] \leq 27$. □

We next introduce an auxiliary lemma that bounds the expectation of the inverse exponential variable $\frac{1}{\mathbf{e}}$ after truncating its tail.

Lemma 1.16. *Let* $\mathbf{e}$ *be an exponential variable with rate* 1. *We define event* $\mathcal{E}$ *as* $\frac{1}{\mathrm{poly}(n)} < \mathbf{e} < \mathrm{poly}(n)$. *Then, we have*

$$\mathbb{E}\left[\frac{1}{\mathbf{e}_i} \mid \mathcal{E}\right] = \mathcal{O}\left(\log n\right).$$

Proof. Due to the probability density function of exponential variables, we have

$$\mathbb{E}\left[\frac{1}{\mathbf{e}_i} \mid \mathcal{E}\right] = \int_{\frac{1}{\mathrm{poly}(n)}}^{\mathrm{poly}(n)} \mathbf{Pr}\left[\frac{1}{\mathbf{e}_i} > t\right] \mathrm{d}t \leq \int_{\frac{1}{\mathrm{poly}(n)}}^{\mathrm{poly}(n)} \frac{1}{t} \mathrm{d}t = \mathcal{O}\left(\log n\right).$$

□

We note that the event $\mathcal{E}$ in the lemma statement of Lemma 1.16 happens with probability $\frac{1}{\mathrm{poly}(n)}$ due to the pdf of exponential variables. Since we generate at most $\mathrm{poly}(n)$ independent exponential variables in all our algorithms, in this paper we condition on that $\mathcal{E}$ holds for each exponential variable we generate, which still happens with high probability by a union bound.

2 Perfect L_p Sampler

In this section, we present our construction of the perfect L_p sampler in the distributed monitoring setting. Our L_p samplers works by first scaling the vector f by exponential random variables to obtain a scaled vector $g = [f_1/\mathbf{e}_1^{1/p}, \ldots, f_n/\mathbf{e}_n^{1/p}]$. We then aim to find the maximum index i^* of g. Section 2.1 introduces a crucial subroutine that identifies a small set of coordinates $\mathbf{PL}$ that are $\frac{1}{\text{polylog}(n)}$-heavy in g, so that we can track each item in thte distributed data stream. In Section 2.2, we produce a truncated Taylor estimator such that it obtains unbiased estimates of f_i^p, which is applied in our L_p sampler and our F_p estimation algorithms. Lastly, in Section 2.3, we show the algorithm of the sampler.

2.1 Identification of Candidate Maxima

In this section, we assume the input to be a scaled vector $g \in \mathbb{R}^n$ distributed across k servers. Specifically, we assume that for the index $i^* = \text{argmax}_{i\in[n]} g_i$, we have $g_{i^*}^p \geq \frac{1}{\text{polylog}(n)} \cdot \|g\|_p^p$ and our goal is to identify a small subset $\mathbf{PL} \subseteq [n]$ of indices that contain i^*, while using $k^{p-1} \cdot \text{polylog}(n)$ total communication. The main intuition is that since g_{i^*} is sufficiently large, then the extremal cases are either there exists a server $j \in [k]$ so that $g_{i^*}(j)$ is large or there exist many servers $j \in [k]$ with a non-trivial contribution to g_{i^*}. In the first case, we will sample the server j containing the large value of $g_{i^*}(j)$ and in the second case, we will sample a sufficient number of servers with a non-trivial contribution to g_{i^*} and then we can use these servers to estimate g_{i^*} using concentration. More generally, anywhere in the spectrum between the two extremes can occur, so we formalize this intuition through the notion of level sets. In particular, for each fixed index $i \in [n]$, we partition the servers $[k]$ based on their contribution $g_i(j)$ with respect to the total value g_i.

Definition 2.1. *For each index $i \in [n]$, server $j \in [k]$, we define the rescaled vector $g(j)$ to be $g_i(j) = \frac{f_i(j)}{\mathbf{e}_i^{1/p}}$. We define $U := \sum_{i\in[n]} \sum_{j\in[k]} g_i(j)^p$ and $G := \|g\|_p^p = \sum_{i\in[n]} g_i^p$. We partition the servers into groups Λ_b, where $b \geq 0$ is an integer, so that*

$$\Lambda_b = \left\{ j \in [k] \,:\, \frac{g_i(j)}{U^{1/p}} \in \left(\frac{1}{2^b}, \frac{1}{2^{b-1}} \right] \right\}.$$

Note that for each coordinate $i \in [n]$, we can decompose

$$g_i = \sum_{j\in[k]} \frac{f_i(j)}{e_i^{1/p}} = \sum_a \sum_{j\in\Lambda_a} \frac{f_i(j)}{e_i^{1/p}}.$$

Then we define the contribution $C_i(b)$ of the group Λ_b toward the frequency of an item $i \in [n]$ is

$$C_i(b) = \sum_{j\in\Lambda_b} g_i(j),$$

so that $g_i = \sum_b C_i(b)$.

We provide the algorithm construction in Algorithm 1 and Algorithm 2, which estimates the contribution of each level set in Definition 2.1.

Now, we analyze the correctness and communication complexity of our algorithm. We first show that the maximizer of the scaled vector is roughly $\frac{1}{\log^2 n}$-heavy, which is given by the maximum stability of exponential random variables.

Algorithm 1 Algorithm to estimate each scaled coordinate: Server

Input: Input scaled vector $g(j) \in \mathbb{R}^n$ that arrives at server j as a data stream
Output: Estimates of the frequencies of all coordinates

$R \leftarrow \mathcal{O}(\log n)$
Let C be the constant in the max-stability inequality ▷See Fact 1.8
Let $\gamma = C_{2.4} C^2 k^{p-1} \operatorname{poly}\left(\frac{\log n}{\varepsilon}\right)$ be the scaling factor for the sampling probability, which is a power of 2.
if the coordinator starts a new round **then**
    Receive an $(1 + \frac{1}{100 \cdot 2^p})$-approximation $\widehat{U}$ to $U = \sum_{j \in [k]} \sum_{i \in [n]} (g_i(j))^p$ from the coordinator
    Generate R uniform random variables $u_i^{(r)}(j) \in [0, 1]$ for each $g_i(j)$
for each time an item i arrives at server j **do**
    for $b \in [\mathcal{O}(\log n)]$, $r \in [R]$ **do**
        if $\frac{(g_i(j))^p}{\widehat{U}} \geq \frac{\varepsilon^2}{C_{2.4} \cdot C^2 k^{p-1} \operatorname{polylog}(n)}$ **then**
            Send $g_i(j)$ to the central server
            Update its value once it increases by a $\frac{\varepsilon}{100}$ fraction
        else if $\frac{(g_i(j))^p}{\widehat{U}} \cdot \gamma \in \left[\frac{1}{2^{bp}}, \frac{1}{2^{(b-1)p}}\right)$ **then**
            Send $g_i(j)$ to the central coordinator if $u_i^{(r)}(j) < \frac{1}{2^{bp}}$
            if $g_i(j)$ has been sent and it increases by a $\frac{\varepsilon}{100 \log n}$-fraction **then**
                Inform the coordinator to delete the sample
                Sample $g_i(j)$ with a newly-generated uniform variable $u_i^{(r)}(j)$

Lemma 2.2. *Let $G = \sum_{i \in [n]} g_i^p$, we have*

$$\mathbf{Pr}\left[\frac{G^{1/p}}{C \log^2 n} \leq \max_{i \in [n]} g_i\right] \geq 1 - \frac{1}{\operatorname{poly}(n)}.$$

Proof. Recall the max-stability property of exponential random variables (see Fact 1.8). Notice we define $G = \sum_{i \in [n]} g_i^p = \sum_{i \in [n]} \frac{f_i^p}{e_i}$, thus we have

$$\mathbf{Pr}\left[\frac{G}{C \log^2 n} \leq \max_{i \in [n]} g_i^p\right] \geq 1 - \frac{1}{\operatorname{poly}(n)}.$$

□

We next show that we indeed sample each coordinate $g_i(j)$ with the correct probability.

Lemma 2.3. *In Algorithm 1, for any fixed time t in the stream, all coordinates $g_i(j)$ that satisfy $\frac{(g_i(j))^p}{\widehat{U}} \cdot \gamma \in \left[\frac{1}{2^{bp}}, \frac{1}{2^{(b-1)p}}\right)$ at t are sampled with probability $\frac{1}{2^{bp}}$.*

Proof. Consider a fixed time t, let t' be the time we generate the last uniform variable $u_i^{(r)}(j)$. Let $g_{i,t}(j)$ denote the value of $g_i(j)$ at time t. The sampling probability of $g_i(j)$ at t equals to $\mathbf{Pr}\left[u_i^{(r)}(j) < \frac{1}{2^{b'p}}\right] = \frac{1}{2^{b'p}}$, where b' is the minimum positive integer such that $\frac{1}{2^{b'p}} \leq \max_{\tau \in [t', t]} \frac{(g_{i,\tau}(j))^p}{\widehat{U}} \cdot \gamma$. Notice that our stream is insertion-only, then we have

$$\max_{\tau \in [t', t]} \frac{(g_{i,\tau}(j))^p}{\widehat{U}} \cdot \gamma = \frac{(g_{i,t}(j))^p}{\widehat{U}} \cdot \gamma.$$

Algorithm 2 Algorithm to estimate the each scaled coordinate: Coordinator

Input: Input vectors $g(1), \ldots, g(k) \in \mathbb{R}^n$ that arrive as a data stream, accuracy parameter $\varepsilon \in (0,1)$
Output: Estimations of the frequencies of all coordinates

$R \leftarrow \mathcal{O}(\log n)$
Obtain an $(1 + \frac{1}{100 \cdot 2^p})$-approximation $\widehat{U}$ to $U = \sum_{j \in [k]} \sum_{i \in [n]} (g_i(j))^p$
Pick $\xi \in [\frac{1}{2}, 1)$ uniformly at random
if $\widehat{U}$ doubles **then**
    Delete all samples and send $\widehat{U}$ to all servers
$\widehat{C_i^{(r)}(b)} \leftarrow 0$ for all r, i, b
for $i \in [n]$ **do**
    if $\frac{(g_i(j))^p}{\widehat{U}} \geq \frac{\varepsilon^3}{C_{2.4} \cdot C^2 k^{p-1} \operatorname{polylog}(n)}$ **then**
        $\widehat{C_i(0)} \leftarrow \widehat{C_i(0)} + \widehat{g_i(j)}$
    for $b \in [\mathcal{O}(\log n)]$, $r \in [R]$ **do**
        if $\frac{(g_i^{*,(r)}(j))^p}{\widehat{U}} \cdot \gamma \in \left[\frac{\xi^p}{2^{bp}}, \frac{\xi^p}{2^{(b-1)p}}\right)$ **then** $\triangleright g_i^{*,(r)}(j)$ is the value of $g_i(j)$ when it is sampled in the r-th repetition
            if less than $\frac{C_{2.4} \cdot C^2 \cdot \log^5 n}{2^p \varepsilon^2}$ samples are in level b in the r-th repetition **then**
                Skip the level and set $\widehat{C_i(b)} \leftarrow 0$
            else
                $\widehat{C_i^{(r)}(b)} \leftarrow \widehat{C_i^{(r)}(b)} + 2^{bp} \cdot \widehat{g_i^{(r)}(j)}$
            $\widehat{C_i(b)} \leftarrow \operatorname{median}_{r \in [R]} \widehat{C_i^{(r)}(b)}$
    $\widehat{g_i} \leftarrow \frac{1}{R} \sum_{r \in [R]} \sum_{b \in \mathbb{Z}} \widehat{C_i^{(r)}(b)}$
return $\widehat{g_1}, \ldots, \widehat{g_n}$

Therefore, for $\frac{(g_{i,t}(j))^p}{\widehat{U}} \cdot \gamma \in \left[\frac{1}{2^{bp}}, \frac{1}{2^{(b-1)p}}\right)$, we have $b' = b$, which gives the correct sampling probability. □

After sampling a number of coordinates using Algorithm 1, the next step is for Algorithm 2 to use the samples to provide meaningful estimates to g_i. Namely, if g_i is large, then the estimate for g_i should be accurate. Conversely, if g_i is small, then the estimate for g_i does not need to be accurate, but it should not be large. The next statement reflects this property of our estimate using a level set argument. We remark that although the proof is involved, the techniques are rather standard, as similar level arguments have been commonly used in literature for sublinear algorithms, e.g., [IW05, WZ12, BBC+17, WZ21a, JWZ24].

We note that our Lemma 2.4 can also be obtained using an extension of the L_p-heavy-hitter algorithm from recent work [HXZW25] for $p \geq 2$. For $1 < p < 2$, an L_2-heavy-hitter algorithm with $\mathcal{O}(k)$ communication suffices to give a constant-factor approximation, which is sufficient in the construction of our perfect L_p-sampler. However, our result improves the dependence on k for ε-heavy-hitters. In addition, a modified version of our algorithm with refined analysis requires only $\mathcal{O}(k + sk^{p-1})$ communication to estimate s scaled vectors, rather than $\mathcal{O}(sk)$ using the L_2-heavy-hitter algorithm, e.g., as a subroutine to acquire s L_p-samples. This is by replacing our $\widehat{U}$ by a 2-approximation of the L_1-of-L_p sum of the original vector: $\sum_{j \in [k]} \sum_{i \in [n]} (f_i(j))^p$. Thus, we keep

the result here for completeness.

Lemma 2.4. *For the choice of constant $C_{2.4}$, the output of Algorithm 2 satisfies*

(1) Let $i = \text{argmax}_{\mathbf{i}\in[n]} g_{\mathbf{i}}$, the estimate $\widehat{g_i}$ satisfies $|\widehat{g_i} - g_i| \le \varepsilon \cdot g_i$.

(2) For all other $i' \neq i$, the estimate $\widehat{g_{i'}}$ satisfies $\widehat{g_{i'}} \le g_{i'} + \varepsilon g_i$.

*That is, taking $\varepsilon = \frac{1}{4}$, Algorithm 2 finds a set **PL** of coordinates that includes the maximizer, and only contains the items that are at least $\frac{1}{4}$-fraction of the maximizer.*

Proof. We fix a time t in the distributed data stream, and we consider a fix instance r.

Level 0. In the case where $\frac{(g_i(j))^p}{\widehat{U}} \ge \frac{\varepsilon^3}{C_{2.4}\cdot C^2 k^{p-1}\,\text{polylog}(n)}$, $g_i(j)$ is sent to the coordinator and is never deleted in a round. Since the server updates its value whenever it increases by a $\frac{\varepsilon}{100}s$-fraction, $\widehat{g_i(j)}$ is a $(1+\frac{\varepsilon}{100})$-approximation to $g_i(j)$. Thus, we have

$$\left|\widehat{C_i^{(r)}(0)} - C_i^{(r)}(0)\right| \le \varepsilon \cdot C_i^{(r)}(0).$$

Idealized setting. Now, we analyze our sampling and re-construction procedure We first consider an idealized setting where each of the estimates $\widehat{g_i^{(r)}(j)}$ are categorized into the correct level set. Let $\nu \in [1,2]$ be a parameter such that $\frac{\nu G^{1/p}}{\xi U^{1/p}} = 2^m$, where m is some integer. Notice that $G \ge U$, so we have $\frac{1}{2^m} = \mathcal{O}(1)$. For each integer $b \in \mathbb{Z}$, we define the level set

$$\Gamma_b = \left\{ j \in [k] \,:\, \frac{g_i(j)}{G^{1/p}} \in \left(\frac{\nu}{2^b}, \frac{\nu}{2^{b-1}}\right] \right\}.$$

Recall that we define the reconstruction level sets in Algorithm 2 as follows,

$$\Lambda_a = \left\{ j \in [k] \,:\, \frac{g_i(j)}{U^{1/p}} \in \left(\frac{\xi}{2^a}, \frac{\xi}{2^{a-1}}\right] \right\}.$$

Then, we have a one-to-one mapping between each pair of these level sets:

$$\Gamma_b = \Lambda_{b-m}.$$

Next, we consider casework on whether $\frac{(g_i(j))^p}{\widehat{U}} \cdot \gamma \ge \frac{C_{2.4}\cdot C^2 \cdot \text{polylog}(n)}{2^p\varepsilon^2 k}$ or whether $\frac{(g_i(j))^p}{\widehat{U}} \cdot \gamma < \frac{C_{2.4}\cdot C^2 \cdot \text{polylog}(n)}{2^p\varepsilon^2 k}$. In short, we show that in our algorithm, if a level has contribution less than a roughly $\frac{\varepsilon}{\log^2 n}$ fraction, then it is in the first case and we truncate those levels since they are negligible. Otherwise, the level is the second case 2 and the variance is small.

Idealized setting, large b. First, we consider a level b where $g_i(j) \in \Gamma_b$ satisfies $\frac{(g_i(j))^p}{\widehat{U}} \cdot \gamma < \frac{C_{2.4}\cdot C^2 \cdot \log^5 n}{2^p\varepsilon^2 k}$. Since there are k servers in total, the number of samples at this level is at most $\frac{C_{2.4}\cdot C^2 \cdot \log^5 n}{2^p\varepsilon^2}$ with high probability. Then, by our algorithm construction, we set $\widehat{C_i(b)} = 0$, i.e., the variance is 0.

Idealized setting, small b. Second, we consider the case where $\frac{(g_i(j))^p}{\widehat{U}} \cdot \gamma \geq \frac{C_{2.4} \cdot C^2 \cdot \log^5 n}{2^p \varepsilon^2 k}$, that is, the sampling probability $p_b \geq \frac{C_{2.4} \cdot C^2 \cdot \log^5 n}{\varepsilon^2 k}$. Consider the level set $\Gamma_b = \Lambda_{b-m}$, where $\frac{(g_i(j))^p}{U} = \Theta(1) \cdot \frac{1}{2^{bp-mp}}$. In an idealized setting, since $\widehat{g_i(j)}$ is within $(1 + \frac{\varepsilon}{100})$-fraction of the actual value $g_i(j)$, we have

$$\mathbb{E}\left[\widehat{C_i^{(r)}(b)}\right] = \frac{1}{p_b} \cdot \sum_{j \in \Gamma_b} p_b \cdot \widehat{g_i^{(r)}(j)} = (1 \pm \varepsilon) \cdot C_i(b).$$

In addition, we have

$$\begin{aligned}
\operatorname{Var}\left[\widehat{C_i^{(r)}(b)}\right] &\leq \frac{1}{p_b^2} \cdot \sum_{j \in \Gamma_b} p_b \cdot \mathcal{O}(1) \cdot (g_i(j))^2 \\
&\leq \frac{1}{p_b} \cdot |\Gamma_b| \cdot \mathcal{O}(1) \cdot \left(\frac{G^{1/p}}{2^b}\right)^2,
\end{aligned}$$

Since we sample all items in Γ_b with probability $p_b = \Theta(1) \cdot \frac{(g_i(j))^p}{U} \cdot \gamma$ with $\gamma = \frac{C_{2.4} \cdot C^2 k^{p-1} \operatorname{polylog}(n)}{\varepsilon^2}$, we have

$$\operatorname{Var}\left[\widehat{C_i^{(r)}(b)}\right] \leq 2^{bp-mp} \cdot \frac{\varepsilon^2}{C_{2.4} \cdot C^2 k^{p-1} \log^5 n} \cdot |\Gamma_b| \cdot \mathcal{O}(1) \cdot \left(\frac{G^{1/p}}{2^b}\right)^2.$$

Notice that $\frac{1}{2^m} = \mathcal{O}(1)$ as mentioned earlier, so we have

$$\operatorname{Var}\left[\widehat{C_i^{(r)}(b)}\right] \leq 2^{bp} \cdot \frac{\varepsilon^2}{C_{2.4} \cdot C^2 k^{p-1} \log^5 n} \cdot |\Gamma_b| \cdot \mathcal{O}(1) \cdot \left(\frac{G^{1/p}}{2^b}\right)^2.$$

Since there are at most $\mathcal{O}\left(2^b\right)$ items in set Γ_b, so $|\Gamma_b| = \min\{\mathcal{O}\left(2^b\right), k\}$. For $\mathcal{O}\left(2^b\right) < k$, we have

$$\begin{aligned}
\operatorname{Var}\left[\widehat{C_i^{(r)}(b)}\right] &\leq 2^{bp} \cdot \frac{\varepsilon^2}{C_{2.4} \cdot C^2 k^{p-1} \log^5 n} \cdot 2^b \cdot \mathcal{O}(1) \cdot \left(\frac{G^{1/p}}{2^b}\right)^2 \\
&= 2^{b(p-1)} \cdot \frac{\varepsilon^2}{C_{2.4} \cdot C^2 k^{p-1} \log^5 n} \cdot \mathcal{O}(1) \cdot G^{2/p}.
\end{aligned}$$

Since $p > 1$, we have $2^{b(p-1)} < k^{p-1}$, so that

$$\begin{aligned}
\operatorname{Var}\left[\widehat{C_i^{(r)}(b)}\right] &\leq \mathcal{O}(1) \cdot k^{p-1} \cdot \frac{\varepsilon^2}{C_{2.4} \cdot C^2 k^{p-1} \log^5 n} \cdot G^{2/p} \\
&= \mathcal{O}\left(\varepsilon^2\right) \cdot \frac{G^{2/p}}{C_{2.4} \cdot C^2 \log^5 n}.
\end{aligned}$$

On the other hand, if $\mathcal{O}\left(2^b\right) \geq k$, since in this case, we have $\frac{1}{p_b} \leq \frac{\varepsilon^2 k}{C_{2.4} \cdot C^2 \cdot \log^5 n}$, so that

$$\operatorname{Var}\left[\widehat{C_i^{(r)}(b)}\right] \leq \frac{1}{p_b} \cdot |\Gamma_b| \cdot \mathcal{O}(1) \cdot \left(\frac{G^{1/p}}{2^b}\right)^2$$

$$\leq \frac{\varepsilon^2 \cdot k}{C_{2.4} \cdot C^2 \log^5 n} \cdot k \cdot \mathcal{O}(1) \cdot \frac{G^{2/p}}{k^2}$$
$$= \frac{\varepsilon^2 \cdot G^{2/p}}{C_{2.4} \cdot C^2 \log^5 n}.$$

Therefore, in all cases, we have

$$\mathrm{Var}\left[\widehat{C_i^{(r)}(b)}\right] \leq \mathcal{O}\left(\varepsilon^2\right) \cdot \frac{G^{2/p}}{C_{2.4} \cdot C^2 \log^5 n}.$$

Applying Lemma 2.2, with high probability we have,

$$\mathrm{Var}\left[\widehat{C_i^{(r)}(b)}\right] \leq \mathcal{O}\left(\frac{\varepsilon^2}{C_{2.4} \log n}\right) \cdot \left(\max_{i \in [n]} g_i\right)^2.$$

Then, by Chebyshev's inequality, we have

$$\mathbf{Pr}\left[\left|\widehat{C_i^{(r)}(b)} - C_i(b)\right| \leq \mathcal{O}\left(\frac{\varepsilon}{C_{2.4} \log n} \cdot \max_{i \in [n]} g_i\right)\right] \geq \frac{2}{3}.$$

Then, by taking the median of $R = \mathcal{O}(\log(n))$ instances, we have by standard Chernoff bounds

$$\mathbf{Pr}\left[\left|\widehat{C_i(b)} - C_i(b)\right| \leq \mathcal{O}\left(\frac{\varepsilon}{C_{2.4} \log n} \cdot \max_{i \in [n]} g_i\right)\right] \geq \frac{1}{\mathrm{poly}(n)}.$$

Then, by union bounding over the $\mathcal{O}(\log n)$ level sets Γ_b, we have that with high probability,

$$\left|\widehat{C_i(b)} - C_i(b)\right| \leq \mathcal{O}\left(\frac{\varepsilon}{C_{2.4} \log n} \cdot \max_{i \in [n]} g_i\right),$$

for all levels b in this case. Thus, summing up the estimate error for each $C_i(b)$ (both large b and small b), with high probability we have

$$\widehat{g_i} \leq g_i + \mathcal{O}\left(\frac{\varepsilon}{C_{2.4}}\right) \cdot g_i.$$

Idealized setting, bounding the bias The above analysis shows that, in the idealized setting, the estimate $\widehat{g_i}$ satisfies $\widehat{g_i} \leq g_i + \mathcal{O}\left(\frac{\varepsilon}{C_{2.4}}\right) \cdot g_i$ for all $i \in [n]$ with high probability. Next, let i be the maximum index, we show that the truncation of $g_i(j)$ samples with small values at most adds up a ε-additive estimate error.

First, we notice that the contribution of level b with $C_i(b) = \frac{\varepsilon \cdot G^{1/p}}{\log^3 n}$ is negligible, since there are at most $\mathcal{O}(\log n)$ levels, and the sum of such levels is at most $\frac{\varepsilon \cdot G^{1/p}}{\log^2 n}$. So, we only consider the case where $C_i(b) \geq \frac{\varepsilon \cdot G^{1/p}}{\log^3 n}$. Suppose that $2^{b+1} > \frac{k \log^3 n}{\varepsilon}$, then we have

$$C_i(b) = \sum_{j \in \Gamma_b} g_i(j) \leq k \cdot \frac{1}{2^{b+1}} \leq \frac{\varepsilon \cdot G^{1/p}}{\log^3 n},$$

which is negligible. Now, for $2^{b+1} \leq \frac{k \log^3 n}{\varepsilon}$, our sampling probability is at least

$$\begin{aligned}\frac{(g_i(j))^p}{\widehat{U}} \cdot \gamma &\geq \frac{(g_i(j))^p}{G} \cdot C_{2.4} C^2 k^{p-1} \operatorname{polylog}(n) \\ &\geq \mathcal{O}(1) \cdot \frac{C_{2.4} C^2 k^{p-1} \operatorname{poly}\left(\frac{\log n}{\varepsilon}\right)}{2^{bp}}.\end{aligned}$$

Then, since we have

$$2^{bp} = 2^{b(p-1)} \cdot 2^b \leq 2^b k^{p-1} \operatorname{poly}\left(\frac{\log n}{\varepsilon}\right),$$

the sampling probability is at least

$$\mathcal{O}\left(\frac{1}{2^b}\right) \cdot C_{2.4} C^2 \operatorname{poly}\left(\frac{\log n}{\varepsilon}\right).$$

Moreover, for a level b to be non-negligible, it at least contains $\mathcal{O}\left(\frac{\varepsilon 2^b}{\log^3 n}\right)$ elements. Then, with high probability, we have at least $\frac{C_{2.4} \cdot C^2 \cdot \log^5 n}{2^p \varepsilon^2}$ samples for sufficiently large exponents in $\operatorname{poly}\left(\frac{\log n}{\varepsilon}\right)$, so that the algorithm does not truncate those levels. Lastly, if some level set contains less than $\frac{C_{2.4} \cdot C^2 \cdot \log^5 n}{2^p \varepsilon^2}$ elements, e.g., $\mathcal{O}\left(\frac{\varepsilon 2^b}{\log^3 n}\right) \leq \frac{C_{2.4} \cdot C^2 \cdot \log^5 n}{2^p \varepsilon^2}$, then each of them satisfies $g_i(j) > \mathcal{O}\left(\frac{G^{1/p} \varepsilon^3}{C_{2.4} \cdot C^2 \log^8 n}\right)$, so our algorithm sends these elements to the coordinator.

Therefore, for all non-negligible level sets, the additive estimate error is at most $\mathcal{O}\left(\frac{\varepsilon}{C_{2.4} \log n}\right) \cdot g_i$ based on previous discussions, and so $|\widehat{g_i} - g_i| \leq \mathcal{O}\left(\frac{\varepsilon}{C_{2.4}}\right) \cdot g_i$ for the maximum index i.

Randomized boundaries. Unfortunately, we are not in an idealized setting, and thus there exist level sets b and servers j such that $\frac{g_i(j)}{G^{1/p}} \in \Gamma_b$, but $\widehat{\frac{g_i(j)}{G^{1/p}}} \notin \Gamma_b$. In this case, we say that $g_i(j)$ is misclassified. Due to our algorithm construction, we have $|\widehat{g_i(j)} - g_i(j)| \leq \frac{\varepsilon}{100} \cdot g_i(j)$ for all sampled items $g_i(j)$. Since $\xi \in \left[\frac{1}{2}, 1\right)$ is chosen uniformly at random, then the probability any coordinate is misclassified is $\mathcal{O}\left(\frac{\varepsilon}{100}\right)$. Observe that when an item is misclassified, its contribution is rescaled by the incorrect probability, but within a factor of two. Thus in expectation, the error due to the misclassification is at most $\mathcal{O}\left(\frac{\varepsilon}{100}\right) \cdot g_i$. Therefore, for those levels with $\widehat{c_i(b)} \neq 0$, by triangle inequality, we have that

$$\Pr\left[\left|\widehat{C_i^{(r)}(b)} - C_i(b)\right| \leq \mathcal{O}\left(\frac{\varepsilon}{C_{2.4}}\right) \cdot g_i\right] \geq \frac{2}{3}.$$

Using a similar analysis as previous sections, we have that

$$\widehat{g_i} \leq g_i + \mathcal{O}\left(\frac{\varepsilon}{C_{2.4}}\right) \cdot g_{\mathbf{i}},$$

for all $i \in [n]$, and

$$|\widehat{g_{\mathbf{i}}} - g_{\mathbf{i}}| \leq \mathcal{O}\left(\frac{\varepsilon}{C_{2.4}}\right) \cdot g_{\mathbf{i}},$$

for the maximum index $\mathbf{i}$, which proves the desired statement by choosing a suitable $C_{2.4}$. □

We next upper bound the total communication of Algorithm 2. To that end, we partition the analysis into rounds, where each round corresponds to an interval during which the quantity $U = \sum_{j\in[k]} \sum_{i\in[n]} (g_i(j))^p$ increases by at most a factor of two and then we upper bound the total number of rounds.

Lemma 2.5. *Define a round to be the time steps between which the quantity $U = \sum_{j\in[k]} \sum_{i\in[n]} (g_i(j))^p$ doubles. Then, there are at most* $\text{polylog}(n)$ *rounds in expectation.*

Proof. We define $V = \sum_{j\in[k]} \sum_{i\in[n]} f_i(j)^p$. Recall that $U = \sum_{j\in[k]} \sum_{i\in[n]} \frac{f_i(j)^p}{\mathbf{e}_i}$. Conditioned on $\mathbf{e}_i < \text{polylog}(n)$, which happens with high probability, we have $V \le U \cdot \text{polylog}(n)$. Moreover, condition on $\frac{1}{\text{poly}(n)} < \mathbf{e}_i < \text{poly}(n)$ for all $i \in [n]$, by Lemma 1.16, we have

$$\mathbb{E}[U] = \mathbb{E}\left[\sum_{j\in[k]} i \in [n] \frac{f_i(j)^p}{\mathbf{e}_i}\right] = V \cdot \mathcal{O}(\log n).$$

Then, consider a duration within the duration that V doubles, U at most increases by a $\text{polylog}(n)$-fraction in expectation: let V_1, V_2 be the value at the start and the end of this duration, $\mathbb{E}[U_2] = V_2 \cdot \mathcal{O}(\log n) = 2V_1 \cdot \mathcal{O}(\log n) \le U_1 \cdot \text{polylog}(n)$. This means that there are $\text{poly}(\log\log(n))$ number of rounds in expectation within this duration. Assuming that there is $m = \text{poly}(n)$ updates in the data stream, there can be at most $\mathcal{O}(\log n)$ times that V doubles. □

Then, it suffices to upper bound the total communication within each round, stated as follows.

Lemma 2.6. *With high probability, Algorithm 2 uses $(k + k^{p-1}) \cdot \text{poly}\left(\frac{\log n}{\varepsilon}\right)$ bits of communication in expectation.*

Proof. Consider a fixed round. Recall that $U = \sum_{j\in[k]} \sum_{i\in[n]} g_i(j)^p$, then by Lemma 2.3, we sample $\gamma = k^{p-1} \cdot \text{poly}\left(\frac{\log n}{\varepsilon}\right)$ elements with high probability. For each element, we delete it and re-sample it whenever it increases by a $\frac{\varepsilon}{100\log n}$-fraction, which each takes at most $\text{poly}\left(\frac{\log n}{\varepsilon}\right)$ bits of communication. In addition, we use $k \log n$ bits of communication in the beginning of each round to inform the new value of U, giving a total communication cost of $k^{p-1} \cdot \text{poly}\left(\frac{\log n}{\varepsilon}\right)$ bits. □

Combining Lemma 2.4 for $\varepsilon = \Theta(1)$ and Lemma 2.6, we have the following guarantees for Algorithm 1 and Algorithm 2, both showing that the set **PL** contains the maximizer and upper bounding the total communication.

Theorem 2.7. *Given a vector $f = \sum_{j\in[k]} f(j) \in \mathbb{R}^n$ defined by a distributed data stream S, n i.i.d. exponential random variable $\mathbf{e}_1, \ldots, \mathbf{e}_n$, and a scaled vector $g_i = \frac{f_i}{\mathbf{e}_i^{1/p}}$, there exists an algorithm that finds a set **PL** of coordinates that includes the maximizer of g and only contains the items that are at least $\frac{1}{4}$-fraction of the maximizer at all times. The algorithm succeeds with probability $1 - \frac{1}{\text{poly}(n)}$ and uses $(k + k^{p-1}) \cdot \text{polylog}(n)$ total bits of communication in expectation.*

2.2 Truncated Taylor Estimator

This section introduces another core technical tool in the L_p sampler as well as our robust F_p-estimation algorithm in following sections. We aim to retrieve unbiased estimates of f_i^p for $p > 0$.

However, we only have access to unbiased estimates of f_i, so there is no direct method to estimate f_i^p when p is not an integer. To handle this challenge, we use the Taylor series of f_i^p. Specifically, let s_i be a 1.01-approximation to f_i. Then

$$f_i^p = \sum_{q=0}^{\infty} \binom{p}{q} s_i^{p-q} (f_i - s_i)^q.$$

Since the tail of the Taylor series decays exponentially, we can truncate at $Q = \mathcal{O}(\log n)$ terms and only incur $\mathcal{O}(n^{-c})$ additive error. Then we for each $q \in [Q]$, we can estimate $(f_i - s_i)^q$ using the product of q independent unbiased estimate of f_i. We give the algorithm in full in Algorithm 3.

Algorithm 3 Truncated Taylor estimator

Input: $Q = \mathcal{O}(\log n)$ independent unbiased estimates $\{\widehat{f_i^{(l)}}\}_{l \in [Q]}$ of f_i satisfying $\mathrm{Var}\left[f_i^{(l)}\right] = \left(\frac{f_i}{100p}\right)^2$, $p > 0$

Output: Unbiased estimate of f_i^p

1: Let s_i be a $(1 + \frac{1}{100p})$-approximation of f_i
2: **for** $q \in [Q] \cup \{0\}$ **do**
3: $\quad \widehat{(f_i - s_i)^q} \leftarrow \prod_{l \in [q]} \left(\widehat{f_i^{(l)}} - s_i\right)$
4: $\quad H \leftarrow H + \binom{p}{q} s_i^{p-q} \widehat{(f_i - s_i)^q}$ ▷ H is the truncated Taylor estimator for f_i^p
5: **return** H

We first show that we can truncate the Taylor series of f_i^p after $Q = \mathcal{O}(\log n)$ terms and only incur additive error $\frac{1}{\mathrm{poly}(n)}$ from the tail of the Taylor series.

Lemma 2.8. *Let $Q = \mathcal{O}(\log n)$, $p > 0$, and $s_i \in \left[\frac{99 f_i}{100}, \frac{101 f_i}{100}\right]$. Then*

$$\left| f_i^p - \sum_{q=0}^{Q} \binom{p}{q} s_i^{p-q} (f_i - s_i)^q \right| \le n^{-c}.$$

Proof. Using the Taylor expansion we have

$$f_i^p = \sum_{q=0}^{\infty} \binom{p}{q} s_i^{p-q} (f_i - s_i)^q.$$

Note that $|f_i - s_i| \le \frac{f_i}{100}$, so for $q > p$ we have

$$\left| \binom{p}{q} s_i^{p-q} (f_i - s_i)^q \right| \le \mathcal{O}\left(e^p \cdot f_i^{p-q} \left(\frac{f_i}{100}\right)^q \right) = \mathcal{O}\left(f_i^p \cdot \left(\frac{1}{100}\right)^q \right),$$

where the first step is by $|\binom{p}{q}| \le e^p$. Hence, we have that for $Q = \mathcal{O}(\log n)$,

$$\sum_{q=Q+1}^{\infty} \binom{p}{q} s_i^{p-q} (f_i - s_i)^q \le \mathcal{O}(1) \cdot f_i^p \cdot \sum_{q=Q+1}^{\infty} \left(\frac{1}{100}\right)^q \le n^{-c},$$

since we assume the stream length to be $\mathrm{poly}(n)$. □

Now, we state an auxiliary lemma that will be used to upper bound the variance of the truncated Taylor estimator.

Lemma 2.9. *Let a be some constant and let $\widehat{a}$ be an estimate satisfying $\mathbb{E}[\widehat{a}] = a$ and $Var[\widehat{a}] = (\frac{a}{C})^2$, where C is some large constant. Let b be another constant satisfying $b = (1 \pm \frac{1}{C}) \cdot a$. Then we have $\mathbb{E}[\widehat{a} - b] = a - b$ and $\mathbb{E}[(\widehat{a} - b)^2] \leq 2 \cdot (\frac{a}{C})^2$.*

Proof. By the linearity of expectation, we have $\mathbb{E}[\widehat{a} - b] = a - b$. Now, we have

$$\begin{aligned}\mathbb{E}\left[(\widehat{a} - b)^2\right] &= \mathrm{Var}[\widehat{a} - b] + (\mathbb{E}[\widehat{a} - b])^2 \\ &= \mathrm{Var}[\widehat{a}] + (a - b)^2\end{aligned}$$

Then, by our assumption of $\mathrm{Var}[\widehat{a}] = (\frac{a}{C})^2$ and $b = (1 \pm \frac{1}{C}) \cdot a$, we have

$$\mathbb{E}\left[(\widehat{a} - b)^2\right] \leq 2 \cdot \left(\frac{a}{C}\right)^2$$

□

The following lemma shows the correctness of our truncated Taylor estimator.

Lemma 2.10. *Let s_i be a $(1 + \frac{1}{100p})$-approximation of f_i. For $Q = \mathcal{O}(\log n)$, let $\{\widehat{f_i^{(l)}}\}_{l \in [Q]}$ be independent estimates of f_i satisfying $Var\left[f_i^{(l)}\right] = \left(\frac{f_i}{100p}\right)^2$. Then, Algorithm 3 returns the following truncated Taylor estimator*

$$\widehat{f_i^p} := \sum_{q=0}^{Q} \left(\binom{p}{q} s_i^{p-q} \prod_{l \in [q]} \left(\widehat{f_i^{(l)}} - s_i \right) \right)$$

and it satisfies $\mathbb{E}\left[\widehat{f_i^p}\right] = f_i^p \pm n^{-c}$ and $\mathbb{E}\left[\left(\widehat{f_i^p}\right)^2\right] = \mathcal{O}\left(f_i^{2p} \cdot \log^2 n\right)$.

Proof. For $l \in [Q]$, the guarantee for each independent $\widehat{f_i^{(l)}}$ estimate is $\mathbb{E}\left[\widehat{f_i^{(l)}}\right] = f_i$ and $\mathrm{Var}\left[\widehat{f_i^{(l)}}\right] = \left(\frac{f_i}{100p}\right)^2$. Thus, applying Lemma 2.9 with $a = f_i$, $b = s_i$, and $C = \frac{1}{100p}$, we have $\mathbb{E}\left[\widehat{f_i^{(l)}} - s_i\right] = f_i - s_i$ and $\mathbb{E}\left[\left(\widehat{f_i^{(l)} - s_i}\right)^2\right] = \mathcal{O}\left(\left(\frac{f_i}{50}\right)^2\right)$. Then, we have

$$\mathbb{E}\left[\prod_{l \in [q]} \left(\widehat{f_i^{(l)}} - s_i \right)\right] = \prod_{l \in [q]} \mathbb{E}\left[\widehat{f_i^{(l)}} - s_i\right] = (f_i - s_i)^q.$$

Moreover, we have

$$\mathbb{E}\left[\left(\prod_{l \in [q]} \left(\widehat{f_i^{(l)}} - s_i \right)\right)^2\right] = \prod_{l \in [q]} \mathbb{E}\left[\left(\widehat{f_i^{(l)}} - s_i \right)^2\right] = \mathcal{O}\left(\left(\frac{f_i}{50p}\right)^{2q}\right).$$

Now, by Lemma 2.8 and $Q = \mathcal{O}(\log n)$, we have

$$\mathbb{E}\left[\widehat{f_i^p}\right] = \sum_{q=0}^{Q} \binom{p}{q} s_i^{p-q} \cdot \mathbb{E}\left[\prod_{l\in[q]} \left(\widehat{f_i^{(l)}} - s_i\right)\right] = \sum_{q=0}^{Q} \binom{p}{q} s_i^{p-q}(u_i - s_i)^q = f_i^p \pm n^{-c}.$$

Moreover, by the AM-GM inequality, we have

$$\begin{aligned}\mathbb{E}\left[\left(\widehat{f_i^p}\right)^2\right] &= \mathbb{E}\left[\left(\sum_{q=0}^{Q}\left(\binom{p}{q} s_i^{p-q} \prod_{l\in[q]}\left(\widehat{f_i^{(l)}} - s_i\right)\right)\right)^2\right] \\ &= \mathcal{O}(Q)\cdot \sum_{q=0}^{Q}\binom{p}{q}^2 s_i^{2p-2q}\mathbb{E}\left[\prod_{l\in[q]}\left(\widehat{f_i^{(l)}} - s_i\right)^2\right].\end{aligned}$$

Therefore,

$$\begin{aligned}\mathbb{E}\left[\left(\widehat{f_i^p}\right)^2\right] &= \mathcal{O}(Q)\cdot \sum_{q=0}^{Q}\binom{p}{q}^2 s_i^{2p-2q}\left(\frac{f_i}{50p}\right)^{2q} \\ &= \mathcal{O}(Q)\cdot \sum_{q=0}^{Q} s_i^{2p-2q}\left(\frac{p\cdot f_i}{50p}\right)^{2q},\end{aligned}$$

where the second step follows by $\binom{p}{q} < p^q$. Now, since $s_i > 0.99 f_i$

$$\mathbb{E}\left[\left(\widehat{f_i^p}\right)^2\right] = \mathcal{O}(Q)\cdot f_i^{2p}\sum_{q=0}^{Q}(0.99 f_i)^{-2q}\left(\frac{f_i}{50}\right)^{2q} = \mathcal{O}\left(Q^2\right)\cdot f_i^{2p}.$$

By $Q = \mathcal{O}(\log n)$, we have our desired result. □

2.3 Sampler Construction

In this section, we construct the L_p perfect sampler for the distributed monitoring setting. We first apply the subroutine introduced in Section 2.1 to find a set **PL** that contains the maximum index with high probability. Then, we obtain a $\left(1+\frac{1}{100p}\right)$-approximation for each g_i in **PL**. Due to the error in our estimate for g_i, we may incorrectly identify the wrong index as the maximum index. To avoid this happening, we would like to do a statistical test on whether $G_{D(1)} > 2G_{D(2)}$ to fail the invalid samplers. However, it is not clear that the failure probability of the statistical test is independent of which index obtains the maximum. For instance, let the original vector be $f = (100n, 1, \ldots, 1) \in \mathbb{R}^n$. Then we would expect that the first index is still the maximum in the scaled vector. And if we condition on the second index achieving the max, we would expect the max and the second max to not have a large gap, and we fail the test with a large probability. In this case, the failure probability differs by an additive constant when conditioning on the first or the second index achieving the max, which leads to a wrong sampling distribution. Thus, we apply the method of duplication introduced by [JW18], that is, we duplicate each item n^c times in the original vector and scale them by different exponential variables. Formally, for a fixed $c > 0$, let F denote a duplicated vector, so that $F_{i,l} := f_i$ for all $i \in [n], l \in [n^c]$, but F can be flattened to be a

vector with dimension n^{c+1}. Notice that $\|F\|_p^p = n^c \cdot \|f\|_p^p$. We use G to denote the scaled vector, so that $\frac{F_i}{e_i^{1/p}}$ for an independent exponential random variable e_i, for all $i \in [n^{c+1}]$.

Unfortunately, as we mentioned in Section 1.2.1, this still does not suffice, because the distribution of the output of optimal approximate counting procedures is affected by the distribution of the coordinate across the servers. Namely, the estimate for g_i may depend on the distribution of g_i across each server j and thus the failure probability for the statistical test is still not independent of the index i. To solve this problem, we induce a two-layer exponential scaling $h_i(j) = \frac{g_i(j)}{\mathbf{e}(j)}$ and apply the geometric mean estimator, which is unbiased and has bounded variance:

$$\widehat{g_i} = \frac{\sqrt[3]{h_i(j_{*,1}) \cdot h_i(j_{*,2}) \cdot h_i(j_{*,3})}}{C_{1.15}}.$$

Here, $j_{*,r}$ is the maximum index of $h_i^{(r)}(j)$ across all servers j for the r-th instance. Now we can run the approximate counting algorithm introduced by [HYZ12] to track each $h_i(j)$, where the communication is restricted to the coordinator and a single server j. The procedure gives a $(1+\varepsilon)$-estimate to the count of $h_i(j)$ using roughly $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ bits of communication. We use Counting to refer to this procedure in the following sections. Due to the max-stability of exponential variables, $\max_j h_i(j) = \frac{g_i}{\mathbf{e}_i'}$, where $\mathbf{e}_i'$ is an independent exponential variable. Thus, the estimate of the max only depends on the value of $\frac{g_i}{\mathbf{e}_i'}$, but not on how g_i is distributed across the k servers. However, this approach requires us to identify the actual maximum index, e.g., when the first max and the second max is $(1+\alpha)$-fractional close to each, we need a separate counting algorithm with accuracy $\frac{\alpha}{100}$ to correctly distinguish between them. Due to the anti-concentration of exponential variables, the first max and the second max of the scaled vector is $(1+\varepsilon)$-fractional close with probability at most $\mathcal{O}(\varepsilon)$, allowing us to bound the communication cost. We show our distributed monitoring L_p sampler in Algorithm 4.

Correctness. We first show that our algorithm has the correct output distribution. By Theorem 2.7, we know that **PL** contains the maximum index at all times with high probability. Therefore, we condition on these events in the following analysis.

We begin by bounding the failure probability of our proposed L_p sampler. Here, we remark that our algorithm only shows an instance of the L_p sampler. We show that it reports some index from the correct L_p distribution with constant probability in the following proof. Hence, we run $\mathcal{O}(\log n)$ such instances simultaneously and take the output of the first successful sampler. By standard concentration inequalities, it reports some index at all times with high probability, while only losing $\mathcal{O}(\log n)$ factors in communication.

Algorithm 4 Distributed monitoring L_p sampler

Input: Input vector $f = \sum_{j\in[k]} f(j) \in \mathbb{R}^n$ from a distributed data stream
Output: Perfect L_p sample from f

All servers generate n^{c+1} i.i.d. exponential variables $\{\mathbf{e}_{i,l}\}_{i\in[n],l\in[n^c]}$ using public randomness. Let $R = \text{polylog}(n)$ be sufficiently large. Each server j generates $3R$ i.i.d. exponential variables $\{\mathbf{e}^{(r)}(j)\}_{j\in[k],r\in[3R]}$

Let $F(j), G(j), H^{(r)}(j) \in \mathbb{R}^{n^{c+1}}$ be the duplicated vectors: $F_{i,l}(j) = f_i(j)$, $G_{i,l}(j) = \frac{f_i(j)}{\mathbf{e}_{i,l}^{1/p}}$, and $H_{i,l}(j) = \frac{f_i(j)}{\mathbf{e}_{i,l}^{1/p}\mathbf{e}(j)}$ for all $i \in [n], l \in [n^c], j \in [k], r \in [3R]$

Maintain a 1.01-approximation to $\|G\|_p^p$ (c.f. [WZ12]). Start a new round with fresh randomness when $\|G\|_p^p$ doubles

For each update, obtain a set **PL** containing the maximum index ▷See Theorem 2.7

if a new element $G_{i,l}$ is added to **PL** **then** ▷Do not delete

  $\widetilde{H_{i,l}^{(r)}}(j) \leftarrow H_{i,l}^{(r)}(j)$ for all $r \in [3R]$ and $j \in [k]$ ▷Query all servers

for $(i,l) \in$ **PL** **do**

  for $r \in [3R]$ **do**

    Let $\widetilde{H_{i,l}^{(r)}}(D(1))$ and $\widetilde{H_{i,l}^{(r)}}(D(2))$ denote the first max and the second max of vector $[\widetilde{H_{i,l}^{(r)}}(1), \ldots, \widetilde{H_{i,l}^{(r)}}(k)]$

    Let **PL**$_i$ be the set of servers with $\widetilde{H_{i,l}^{(r)}}(j) \geq \frac{\widetilde{H_{i,l}^{(r)}}(D(1))}{200}$ ▷**PL**$_i$ contains the servers with heavy values with respect to index i

    $\widehat{H_{i,l}^{(r)}}(j) \leftarrow \textsc{Counting}(H_{i,l}^{(r)}(j), \frac{1}{100p})$ for all $j \in [k]\backslash$**PL**$_i$

    if $\frac{\widetilde{H_{i,l}^{(r)}}(D(1))}{\widetilde{H_{i,l}^{(r)}}(D(2))} = 1+\alpha$ **then**

      $\widehat{H_{i,l}^{(r)}}(j) \leftarrow \textsc{Counting}(H_{i,l}^{(r)}(j), \frac{\alpha}{100p})$ for all $j \in$ **PL**$_i$ ▷Counting is ran on a single server

      Inform all servers in **PL**$_i$ to change the accuracy if α changes by a 2-fraction

  $j_{*,r} \leftarrow \text{argmax}_{j\in[k]} \widetilde{H_{i,l}^{(r)}}(j)$ ▷Obtain the maximum index across servers

  $\widehat{H_{i,l,q}^{(r)}}(j) \leftarrow \textsc{Counting}(H_{i,l}^{(r)}(j), \frac{1}{100p})$ for all $j \in [k]$ and $q \in [Q]$, where $Q = \mathcal{O}(\log n)$ ▷Run a separate counting algorithm to estimate $G_{i,l}$

  $\left(\widehat{H_{i,l}^{(r)}}(j)\right)^{1/3} \leftarrow \textsc{TaylorEst}(\{\widehat{H_{i,l,q}^{(r)}}(j)\}_{q\in[Q]})$ ▷See Lemma 2.10

  $\widehat{G_{i,l}} \leftarrow \frac{1}{R}\sum_{r=1}^{R} \frac{\left(\widehat{H_{i,l}^{(3r-2)}}(j_{*,3r-2})\right)^{1/3}\left(\widehat{H_{i,l}^{(3r-1)}}(j_{*,3r-1})\right)^{1/3}\left(\widehat{H_{i,l}^{(3r)}}(j_{*,3r})\right)^{1/3}}{C_{1.15}}$

$\mu \leftarrow \mathcal{U}(0.99, 1.01)$

Let $\widehat{G_{D(1)}}$ and $\widehat{G_{D(1)}}$ be the max and the second max of all tue estimates $\widehat{G_{i,l}}$

return $i^* = \text{argmax}\,\widehat{G_{i,l}}$ if $\widehat{G_{D(1)}} > 2\mu\widehat{G_{D(2)}}$; otherwise FAIL

First, we bound the error of our estimate $\widehat{G_{i,l}}$.

Lemma 2.11. *For all $(i,l) \in \boldsymbol{PL}$, $\widehat{G_{i,l}}$ is a 1.01-approximation to its true value $G_{i,l}$.*

Proof. Notice that $H_{i,l}^{(r)}(j) = \frac{F_{i,l}(j)}{\mathbf{e}_{i,l}^{1/p}\mathbf{e}^{(r)}(j)} = \frac{G_{i,l}(j)}{\mathbf{e}^{(r)}(j)}$ for all $r \in [3R]$. By the max-stability of the scaled vector (c.f. Corollary 1.13), we have $\max_{j\in[k]} H_{i,l}^{(r)}(j) = \frac{F_{i,l}}{\mathbf{e}_{i,l}^{1/p}\mathbf{e}^{*,(r)}}$, where $\mathbf{e}^{*,(r)}$ is an exponential variable independent of $\{\mathbf{e}_{i,l}^{(r)}(j)\}_{j\in[k]}$ and $F_{i,l} = \sum_{j\in[k]} F_{i,l}(j)$. Now, let $j_{*,r} = \max_{j\in[k]} H_{i,l}^{(r)}(j)$ and we define the geometric mean estimator as

$$G_{i,l}^{(r)} = \frac{\sqrt[3]{H_{i,l}^{(3r-2)}(j_{*,3r-2})H_{i,l}^{(3r-1)}(j_{*,3r-1})H_{i,l}^{(3r)}(j_{*,3r})}}{C_{1.15}}.$$

Then, we have

$$G_{i,l}^{(r)} = \frac{G_{i,l}}{C_{1.15}\cdot\sqrt[3]{\mathbf{e}^{*,(3r-2)}\mathbf{e}^{*,(3r-1)}\mathbf{e}^{*,(3r)}}}.$$

Applying Lemma 1.15, we have $\mathbb{E}\left[G_{i,l}^{(r)}\right] = G_{i,l}$ and $\mathbb{E}\left[\left(G_{i,l}^{(r)}\right)^2\right] = \mathcal{O}\left(G_{i,l}^2\right)$.

By our construction in Algorithm 4, the accuracy of the first counting algorithm is $\frac{\alpha}{100p}$ if the ratio between the max and the second max is $1+\alpha$. So, we detect the actual max from the estimated scaled vector, i.e., $j_{*,r}$ in line 15 is the maximum index of vector $[H_{i,l}^{(r)}(1),\ldots,H_{i,l}^{(r)}(k)]$ for all $r \in [3R]$. Additionally, by Lemma 2.10, the truncated Taylor estimator for each $\left(H_{i,l}^{(r)}(j)\right)^{1/3}$ satisfies $\mathbb{E}\left[\widehat{\left(H_{i,l}^{(r)}(j)\right)^{1/3}}\right] = \left(H_{i,l}^{(r)}(j)\right)^{1/3} \pm n^{-c}$ and $\mathbb{E}\left[\left(\widehat{\left(H_{i,l}^{(r)}(j)\right)^{1/3}}\right)^2\right] = \mathcal{O}\left(\left(H_{i,l}^{(r)}(j)\right)^{2/3}\cdot\log^2 n\right)$. We define our estimate of $G_{i,l}^{(r)}$ as

$$\widehat{G_{i,l}^{(r)}} = \frac{\widehat{\left(H_{i,l}^{(3r-2)}(j_{*,3r-2})\right)^{1/3}}\widehat{\left(H_{i,l}^{(3r-2)}(j_{*,3r-2})\right)^{1/3}}\widehat{\left(H_{i,l}^{(3r-2)}(j_{*,3r-2})\right)^{1/3}}}{C_{1.15}}.$$

Then, we have $\mathbb{E}\left[\widehat{G_{i,l}^{(r)}}\right] = G_{i,l} \pm n^{-c}$ and $\mathbb{E}\left[\widehat{\left(G_{i,l}^{(r)}\right)^2}\right] = \mathcal{O}\left(G_{i,l}^2\cdot\log^2 n\right)$. Thus, averaging $R = \text{polylog}(n)$ estimates $\widehat{G_{i,l}^{(r)}}$ gives as a 1.01-approximation to $g_{i,l}$. □

We now show that with high probability, our perfect L_p sampler successfully produces a sample at each time, i.e., some instance does not fail the statistical test at each time.

Lemma 2.12. *With probability $1 - \frac{1}{\text{poly}(n)}$, our L_p sampler succeeds at all times.*

Proof. Consider a fixed time t. We accept the output from the L_p sampler if its estimates satisfy $\widehat{G_{D(1)}} > 2\mu\widehat{G_{D(2)}}$. From Lemma 2.11 we have that $\widehat{G_{D(1)}}$ and $\widehat{G_{D(2)}}$ are both 1.01-approximations. Also, we have $\mu \in [0.99, 1.01]$. Therefore, we accept the sample if $G_{D(1)} > 3G_{D(2)}$. Thus, it suffices to bound the probability that $G_{D(1)} > 3G_{D(2)}$ holds at time t. By Corollary 1.13, we have

$G_{D(1)} = \|F\|_p / E_1^{1/p}$ and $G_{D(2)} = (E_1/\|F\|_p^p + E_2/\|F_{-D(1)}\|_p^p)^{-1/p}$, where E_1 and E_2 are independent exponential variables. Now, since we duplicate each term by n^c times, we have

$$G_{D(2)} = \left(\frac{E_1}{\|F\|_p^p} + \frac{E_2}{\|F\|_p^p \cdot (1 \pm n^c)}\right)^{-1/p} = \frac{\|F\|_p}{(E_1 + E_2 \cdot (1 \pm n^c))^{1/p}}.$$

Therefore, due to the probability density function of exponential variables,

$$\mathbf{Pr}\left[G_{D(1)} > 3G_{D(2)}\right] > \Omega(1).$$

Then, since we run $\mathcal{O}(\log n)$ parallel samplers with different exponential variables, with probability $1 - n^{-c}$, at least one of them satisfies $G_{D(1)} > 3G_{D(2)}$. By a union bound across all times, we have our desired result. □

Next, we show that conditioning on the identity of an index that achieves the max, the failure probability of a sampler only shifts by $\frac{1}{\text{poly}(n)}$ after duplication. This is important because it shows that the failure condition cannot overly depend on the maximizer itself, which would otherwise destroy the perfect sampler.

First, we recall the following statement by [JW18], which states that each entry of the duplicated scaled vector can be decomposed as the sum of an independent term and a dependent term with magnitude n^{-c}.

Lemma 2.13. *[JW18] Let $D = (D(1) \cdots, D(n^{c+1}))$ be the anti-rank vector of G. Let N be size of the support of F. For every $i < N - n^{9c/10}$ we have*

$$G_{D(i)} = U_{D(i)}^{1/p}\left(1 \pm \mathcal{O}\left(\frac{1}{p} n^{-(c+1)/10}\right)\right),$$

where $U_{D(i)} = \left(\sum_{\tau=1}^{a} \frac{E_\tau}{\mathbb{E}\left[\sum_{j=\tau}^{N} |F_{D(j)}|^p\right]}\right)^{-1}$ is totally determined by a and the hidden exponentials E_i, and thus, independent of the anti-rank vector D.

Next, we introduce the following result, which states that if we decompose a variable Z into X and Y, where X is independent of some event E and Y has small magnitude, then the conditional probability of Z falling within an interval I is close to the unconditional probability of Z falling into I.

Lemma 2.14. *[JW18] Let $X, Y \in \mathbb{R}^d$ be random variables where $Z = X + Y$. Suppose X is independent of some event E, and let $M > 0$ be such that for every $i \in [d]$ and every $a < b$ we have $\Pr[a \le X_i \le b] \le M(b - a)$. Suppose further that $|Y|_\infty \le \varepsilon$. Then if $I = I_1 \times I_2 \times \cdots \times I_d \subset \mathbb{R}^n$, where each $I_j = [a_j, b_j] \subset \mathbb{R}, -\infty \le a_j < b_j \le \infty$ is a (possibly unbounded) interval, then*

$$\Pr[Z \in I \mid E] = \Pr[Z \in I] + \mathcal{O}(\varepsilon d M).$$

Utilizing the above lemma, we decompose the estimates of $G_{D(1)}$ and $G_{D(2)}$ into an independent term and a dependent disturbance with small magnitude, so that we eliminate the dependence of the failure probability on the anti-ranks.

Lemma 2.15. *Let FAIL denote the event that we reject the sample given by Algorithm 4. We have* $\mathbf{Pr}\left[\neg\text{FAIL} \mid D(1)\right] = \mathbf{Pr}\left[\neg\text{FAIL}\right] \pm n^{-c}$ *at all times, where* $D = (D(1), \cdots, D(n))$ *is the anti-rank vector.*

Proof. First, note that the subroutine that maintains $\mathbf{PL}$ succeeds with high probability by Theorem 2.7. This still holds after conditioning on which index achieves the max. To see this, let $\mathcal{E}$ be the event that it fails, we have

$$\mathbf{Pr}\left[\mathcal{E}\right] = \sum_{i\in[n^{c+1}]} \mathbf{Pr}\left[\mathcal{E} \mid D(1) = i\right] \cdot \mathbf{Pr}\left[D(1) = i\right] = \sum_{i\in[n^{c+1}]} \mathbf{Pr}\left[\mathcal{E} \mid D(1) = i\right] \cdot \frac{|F_i|^p}{\|F\|_p^p}.$$

We assume that $F_i \geq 1$ and $\|F\|_p = \text{poly}(n)$, then we have for all i,

$$\mathbf{Pr}\left[\mathcal{E} \mid D(1) = i\right] \cdot \frac{1}{n^\alpha} \leq \mathbf{Pr}\left[\mathcal{E}\right] = n^{-\beta}.$$

Therefore, we have $\mathbf{Pr}\left[\mathcal{E} \mid D(1) = i\right] \leq n^{-\beta+\alpha}$ which is $\frac{1}{\text{poly}(n)}$ by setting a large enough β. We union bound across all times in the stream, and hence we show that the dependency only causes a $\frac{1}{\text{poly}(n)}$ perturbation on the failure probability. Next, we condition on the event $\mathcal{E}$, which means that $\mathbf{PL}$ contains the maximum index at all times. Based on the same argument, we can condition on all (distributed) approximate counters succeed in the algorithm. Then, it suffices to bound the influence of the dependency on the estimate of $G_{D(1)}$ and $G_{D(2)}$.

Now, we briefly describe the distributed counting algorithm introduced by [HYZ12], which gives a $(1+\varepsilon)$-approximation to the total count n of the number of updates in a distributed data stream (or the count of a particular item). Since we only run the counting algorithm on a single server, we consider a communication protocol between a server and a coordinator, and our task is to maintain an estimate at the coordinator of the count of the server. The server sends the first $\Theta\left(\frac{1}{\varepsilon}\right)$ items, then it sets a probability $p = \Theta\left(\frac{1}{\varepsilon\cdot n}\right)$. When the server receives an item, it sends its current local count to the coordinator with probability p. Let $\bar{n}$ be the last update received by the coordinator, the estimate is $\bar{n} - 1 + 1/p$. The algorithm starts a new round with updated sampling probability whenever the estimate of n doubles.

Recall that $H_{i,l}^{(r)}(j) = \frac{F_{i,l}(j)}{\mathbf{e}_{i,l}^{1/p}\mathbf{e}^{(r)}(j)}$ for all $r \in [3R]$. By the max-stability of the scaled vector (c.f. Corollary 1.13), we have $\max_{j\in[k]} H_{i,l}^{(r)}(j) = \frac{F_{i,l}}{\mathbf{e}_{i,l}^{1/p}\mathbf{e}^{*,(r)}}$, where $\mathbf{e}^{*,(r)}$ is an exponential variable independent of $\{\mathbf{e}^{(r)}(j)\}_{j\in[k]}$, and $F_{i,l} = \sum_{j\in[k]} F_{i,l}(j)$. This implies that the distribution of $\max_{j\in[k]} H_{i,l}^{(r)}(j)$ is independent of how $F_{i,l}$ is distributed across the server j.

Due to our algorithm construction, $j_{*,r}$ in line 15 of Algorithm 4 is the maximum index of vector $[H_{i,l}^{(r)}(1), \ldots, H_{i,l}^{(r)}(k)]$ for all $r \in [3R]$. Additionally, we run a separate counting algorithm on each server j to obtain a 1.01-approximation $\widehat{H_{i,l}^{(r)}(j)}$, which is independent of the counting algorithm to detect the max $j_{*,r}$. We use the notation $\widehat{H_{i,l}^{(r)}(j_{*,r})}$ in the following estimation steps. Notice that in the counting algorithm, the sampling probability is defined solely by the count value. Therefore, the estimate of $\widehat{H_{i,l}^{(r)}(j_{*,r})}$ only depends on the value of $\max_{j\in[k]} H_{i,l}^{(r)}(j)$, which is $\frac{F_{i,l}}{\mathbf{e}_{i,l}^{1/p}\mathbf{e}^{*,(r)}}$.

Let $D(i)$ be the anti-ranks of vector G. Using Lemma 2.13, we have the following decomposition

$$G_{D(i)} = U_{D(i)}^{1/p}\left(1 \pm \mathcal{O}\left(n^{-(c+1)/10}\right)\right),$$

where $U_{D(i)}$ does not depend on the anti-ranks. Moreover, since each $\mathbf{e}^{*,(r)}$ is an independent exponential variable that does not depend on the anti-ranks. Thus, we can write $\mathbf{e}^{*,(r)}$ inside $U_{D(i)}^{1/p}$:

$$\frac{F_{D(i)}}{\mathbf{e}_{D(i)}^{1/p}\mathbf{e}^{*,(r)}} = U_{D(i)}^{1/p}\left(1 \pm \mathcal{O}\left(n^{-(c+1)/10}\right)\right).$$

Since the sampling probability is defined as $\mathcal{O}\left(\frac{\mathbf{e}_{i,l}^{1/p}\mathbf{e}^{*,(r)}}{F_{i,l}}\right)$ for the counting algorithm, we can condition on that the $\mathcal{O}\left(n^{-(c+1)/10}\right)$ disturbance caused by the dependency does not affect the time that the server update its count to the coordinator. It follows that the estimate of the counting algorithm has a similar decomposition (here we abuse the notation of $U_{D(i)}$),

$$\widehat{H_{i,l}^{(r)}(j_{*,r})} = \widehat{U_{D(i)}} \cdot \left(1 \pm \mathcal{O}\left(n^{-(c+1)/10}\right)\right).$$

Then, the truncated Taylor estimator in Lemma 2.10 is decomposed into

$$\left(\widehat{H_{i,l}^{(r)}(j_{*,r})}\right)^{1/3} = \widehat{U_{D(i)}} \cdot \left(1 \pm \mathcal{O}\left(\log n \cdot n^{-(c+1)/10}\right)\right).$$

Consequently, the geometric mean estimator in Line 18 of Algorithm 4 is decomposed into

$$\widehat{G_{D(i)}} = \widehat{U_{D(i)}} \cdot \left(1 \pm \operatorname{polylog} n \cdot n^{-(c+1)/10}\right).$$

We abort a sampler if $\widehat{G_{D(1)}} < 2\mu\widehat{G_{D(2)}}$. So, we decompose $\widehat{G_{D(1)}} - 2\mu\widehat{G_{D(2)}}$ into $\widehat{U_{D(1)}} - 2\mu\widehat{U_{D(2)}} + \widehat{V_{D(1)}} - 2\mu\widehat{V_{D(2)}}$ where $\widehat{U_{D(1)}} - 2\mu\widehat{U_{D(2)}}$ is independent of the anti-ranks and $\widehat{V_{D(1)}} - 2\mu\widehat{V_{D(2)}}$ is some random variable upper bounded by $|\widehat{V_{D(1)}} - 2\mu\widehat{V_{D(2)}}| \le \mathcal{O}\left(n^{-(c+1)/20}\right)$. Notice that for any interval $I \in \mathbb{R}$, we have

$$\mathbf{Pr}\left[\widehat{U_{D(1)}} - 2\mu\widehat{U_{D(2)}} \in I\right] = \mathbf{Pr}\left[\mu \in I'/2\widehat{U_{D(2)}}\right] = \mathcal{O}\left(|I|/2\widehat{U_{D(2)}}\right)$$

where I' is the shifted interval of I which is independent of μ_1. Therefore, applying Lemma 2.14 with $X = \widehat{U_{D(1)}} - 2\mu\widehat{U_{D(2)}}$, $Y = \widehat{V_{D(1)}} - 2\mu\widehat{V_{D(2)}}$, and $E = D(1)$ we have

$$\mathbf{Pr}\left[\widehat{G_{D(1)}} - 2\mu\widehat{G_{D(2)}} \le 0 \mid D(1)\right] = \mathbf{Pr}\left[\widehat{G_{D(1)}} - 2\mu\widehat{G_{D(2)}}\right] \pm \frac{1}{\operatorname{poly}(n)},$$

which proves our desired result. □

Finally, we show that our L_p sampler has the correct sampling distribution.

Theorem 2.16. *Algorithm 4 returns an index $\mathbf{i} \in [n]$ at all times, such that for any $i \in [n]$, we have*

$$\mathbf{Pr}\left[\mathbf{i} = i\right] = \frac{f_i^p}{\|f\|_p^p} \pm \frac{1}{\operatorname{poly}(n)}.$$

Moreover, if i is sampled, it outputs an estimation $\widehat{G_i}$ such that $\mathbb{E}\left[\widehat{G_i}\right] = G_i$ and $\mathbb{E}\left[\left(\widehat{G_i}\right)^2\right] = \mathcal{O}\left(G_i^2\right)$.

Proof. Algorithm 4 reports the index if it satisfies that $\widehat{G_{D(1)}} > 2\mu\widehat{G_{D(2)}}$. Since our algorithm gives a 1.01-approximation for each item in **PL** at all times with probability $1 - n^{-c}$ (c.f. Lemma 2.11), $G_{D(1)}$ is strictly larger than $G_{D(2)}$. Hence, we recover the true maximum index of the scaled vector with probability $1 - n^{-c}$. Let q be the success probability of Algorithm 4. Then, by Lemma 2.15, the probability that Algorithm 4 outputs $\mathbf{i}$ is

$$\sum_{j\in[n^{c+1}]} \Pr\left[\neg\text{FAIL} \mid \mathbf{i}_j = \arg\max_{i',j'}\left\{\left|G_{i'_{j'}}\right|\right\}\right]\Pr\left[i_j = \arg\max_{i',j'}\left\{\left|G_{i_{j'}}\right|\right\}\right] = \sum_{j\in[n^c]}\frac{|f_i|^p}{\|F\|_p^p}\left(q \pm \frac{1}{\text{poly}(n)}\right)$$
$$= \frac{|f_i|^p}{\|f\|_p^p}\left(q \pm \frac{1}{\text{poly}(n)}\right)$$

Note that the additive $\frac{1}{\text{poly}(n)}$ term due to dependencies on the maximizer can be union bounded across all samplers and all time. Moreover, by Lemma 2.12, some index is reported at all times with probability $1 - n^{-c}$, since we run $\mathcal{O}(\log n)$ instances of Algorithm 4. Therefore, the probability of our L_p sampler outputting i is

$$\frac{f_i^p}{\|f\|_p^p} + \frac{1}{\text{poly}(n)}.$$

Additionally, our guarantee for the estimation of $\widehat{G_i}$ follows from the analysis of Lemma 2.11. □

Communication cost. We analyze the communication cost of our algorithm. Let $\mathcal{E}$ denote the event that $\frac{1}{\text{poly}(n)} < \mathbf{e} < \text{poly}(n)$ and let $\mathcal{B}$ denote the event that $\mathbf{e} < \text{polylog}(n)$. From the pdf of exponential variables, we have $\mathbf{Pr}[\mathcal{E}\cap\mathcal{B}] = 1 - \frac{1}{\text{poly}(n)}$. Therefore, we condition on $\mathcal{E}$ and $\mathcal{B}$ for all exponentials in this section, which still has high probability by a union bound. We use restriction $\mathbb{E}[X] = \mathbb{E}[X \mid \mathcal{E}\cap\mathcal{B}]$ for all expectation computations in this section.

We define a round to be the times between which $\|G\|_p^p$ doubles. Within a round, for a parameter X, we use X' to denote this term at the end of this round, and we use X to denote this term at the beginning of this round. For simplicity of notation, we use F_i with $i \in [n^{c+1}]$ to denote a coordinate in the duplicated vector F, instead of $F_{i,l}$ in previous sections. The next lemma upper bounds the expected number of rounds in the data stream.

Lemma 2.17. *Let a round be the times between which $\|G\|_p^p$ doubles, i.e., between 2^i and 2^{i+1} for a fixed integer i. Then there are at most* $\text{polylog}(n)$ *rounds in expectation.*

Proof. Conditioned on $\mathcal{E}$ and $\mathcal{B}$, we have

$$\mathbb{E}\left[\|G\|_p^p\right] = \mathbb{E}\left[\sum_{i\in[n^{c+1}]}\frac{(F_i)^p}{\mathbf{e}_i}\right] = \mathcal{O}(\log n)\cdot\|F\|_p^p = \mathcal{O}(\log n)\cdot\|F\|_p^p,$$

where the second step follows from Lemma 1.16. Since we conditioned on $\mathcal{B}$, we have $(F_i)^p/\mathbf{e}_i \geq (F_i)^p/\text{polylog}(n)$ for all $i \in [n^{c+1}]$, which implies that $\|F\|_p^p \leq \|G\|_p^p \cdot \text{polylog}(n)$. Then, consider a duration within the duration that $\|F\|_p^p$ doubles, $\|G\|_p^p$ at most increases by a $\text{polylog}(n)$-fraction in expectation:

$$\mathbb{E}\left[\|G'\|_p^p\right] = \|F'\|_p^p \cdot \mathcal{O}(\log n) = 2\|F\|_p^p \cdot \mathcal{O}(\log n) \leq \|G\|_p^p \cdot \text{polylog}(n).$$

This means that there are $\text{poly}(\log\log(n))$ number of rounds in expectation within this duration. Assuming that there is $m = \text{poly}(n)$ updates in the data stream, there can be at most $\mathcal{O}(\log n)$ times that $\|F\|_p^p$ doubles, and so there are at most $\text{polylog}(n)$ rounds in expectation. □

Given the above result, it suffices to upper bound the communication cost within one round. To upper bound the communication in one round, we first upper bound the size of **PL** in one round. In our algorithm, once an index is added to **PL**, we never delete it. Then it suffices to upper bound the number of additions. The key observation is that all elements in **PL** are roughly $\frac{1}{\log^2 n}$-heavy with respect to the scaled vector G at the beginning. Therefore, it is still a heavy-hitter with respect to G at the end, up to some $\mathrm{polylog}(n)$ factors. Thus, there are at most $\mathrm{polylog}(n)$ such elements. We formalize this argument in the following lemma.

Lemma 2.18. *With high probability, there are at most* $\mathrm{polylog}(n)$ *elements added to* ***PL*** *during a round.*

Proof. Let $G_{D(1)}$ be the max of G. Then by Fact 1.8, we have $G_{D(1)}^p \geq \frac{\|G\|_p^p}{\mathrm{polylog}\, n}$ with high probability, so we condition on this event for the remainder of the proof. Note that our data stream is insertion-only. Thus by the guarantee of PLRecovery (c.f. Theorem 2.7), **PL** only contains those elements that are at least $\frac{1}{4}$-fraction of the maximum coordinate $G_{D(1)}$. At the end of this round, with high probability, we have

$$\|G'\|_p^p = 2\|G\|_p^p \leq \frac{G_{D(1)}^p}{\mathrm{polylog}(n)}.$$

Therefore, there are at most $\mathrm{polylog}(n)$ elements in **PL** at the end of the round that are at least $\frac{1}{4}$-fraction of $G_{D(1)}$, and so there are at most $\mathrm{polylog}(n)$ elements in **PL**. □

Next, we upper bound the communication used to estimate G_i. Recall that we scaled the local count $G_i(j)$ by an independent exponential and defined $H_i(j) = \frac{G_i(j)}{\mathbf{e}(j)}$. Due to the dependence on the anti-ranks, we need to identify the actual maximum candidate among the k servers, which means that if the ratio between the max and the second max is $1+\alpha$, we need $\mathcal{O}\left(\frac{k}{\alpha}\right)$ bits of communication to run the counting algorithm with accuracy $\frac{\alpha}{100p}$ at each server. In addition, if the ratio is reduced from $1+\alpha$ to $1+\frac{\alpha}{2}$, we need to inform each server to run a counting algorithm with a higher accuracy. So, we need to upper bound the expected times of changing to a new anti-concentration ratio.

To solve this problem, we define an auxiliary variable $H_i = \sum_{j\in[k]} H_i(j) = \sum_{j\in[k]} \frac{G_i(j)}{\mathbf{e}(j)}$. First, we show that H_i at most increases by $G_i \cdot \mathrm{polylog}(n)$ in expectation during the whole round. Next, we show that whenever the ratio is reduced from $1+\alpha$ to $1+\frac{\alpha}{2}$, H_i increases by at least $\alpha \cdot \frac{G_i}{\mathrm{polylog}(n)}$ with high probability. This implies that we expected to witness at most $\frac{1}{\alpha} \cdot \mathrm{polylog}(n)$ times of such changes in a round, which does not give our desired communication bound. Fortunately, due to the anti-concentration property of the exponential variables, the ratio between the max and the second max is $1+\alpha$ with probability at most $\mathcal{O}(\alpha)$. Thus, we can amortize the communication to change accuracy by this property and prove an expected $k \cdot \mathrm{polylog}(n)$ communication in total. We formalize the above argument in the following lemmas. In the following statement, we show that each item in **PL** only increases by a $\mathrm{polylog}(n)$ fraction in each round.

Lemma 2.19. *With high probability, we have* $G_i' \leq G_i \cdot \mathrm{polylog}(n)$ *for all* $i \in$ ***PL***.

Proof. For a coordinate $i \in$ **PL**, from the analysis in Lemma 2.18, we know that $G_i^p \geq \frac{\|G\|_p^p}{\mathrm{polylog}(n)}$ with high probability. From Lemma 2.18, we have $|\mathbf{PL}| = \mathrm{polylog}(n)$, so by a union bound, we have $(G_i)^p \leq G_i^p \cdot \mathrm{polylog}(n)$ for all $i \in$ **PL** with high probability. Conditioned on these events, with high probability we have

$$(G_i')^p \leq \|G'\|_p^p \leq 2 \cdot \|G\|_p^p \leq G_i^p \cdot \mathrm{polylog}(n).$$

Note that in the above proof, we assume that G_i is added to $\mathbf{PL}$ at the beginning of the round. This assumption is without loss of generality, since otherwise, we suppose that i is added to $\mathbf{PL}$ at time t. Then we can consider t as the start time of this round for coordinate i, e.g., let $G_i = G_i^{(t)}$ and $H_i = H_i^{(t)}$, so the analysis remains the same since our data stream is insertion-only. In addition, when i is not added to $\mathbf{PL}$, we only run a PLRecovery on G_i to give a rough estimate, so we do not incur additional communication to identify the max of $H_i(j)$. □

Next, we upper bound how much H_i changes over the course of a round.

Lemma 2.20. *Define H_i to be $\sum_{j\in[k]} H_i(j)$. Then with high probability, we have $\mathbb{E}[H_i'] \le G_i \cdot \mathrm{polylog}(n)$.*

Proof. From the definition of H_i, we have

$$\mathbb{E}[H_i'] = \mathbb{E}\left[\sum_{i\in k} \frac{G_i'(j)}{\mathbf{e}_i(j)}\right] = \mathcal{O}(\log n)\cdot \sum_{i\in k} G_i'(j) = \mathcal{O}(G_i'\cdot \log n),$$

where the second step follows from Lemma 1.16 and the third step follows from Lemma 2.19. Then, since $G_i' \le G_i \cdot \mathrm{polylog}(n)$, c.f. Lemma 2.19, we have

$$\mathbb{E}[H_i'] = \mathcal{O}(G_i'\cdot \log n) \le G_i \cdot \mathrm{polylog}(n).$$

□

Recall that in our algorithm, for each heavy coordinate $i \in \mathbf{PL}$, we select a list of servers that obtains a heavy value, denoted as $\mathbf{PL}_i$, and we only keep track of these servers with higher accuracy. The next lemma shows that this set contains $\mathrm{polylog}(n)$ servers in expectation, so that we only inform those servers whenever we change the accuracy.

Lemma 2.21. *With high probability, there are at most $\mathrm{polylog}(n)$ elements added to $\mathbf{PL}_i$ in expectation during a round for all $i \in \mathbf{PL}$.*

Proof. By Lemma 2.20, we have $\mathbb{E}[H_i'] \le G_i \cdot \mathrm{polylog}(n)$. Recall that in the analysis of Lemma 2.23, we have shown $H_i(D(1)) \ge \frac{G_i}{\mathrm{polylog}(n)}$ with high probability. Note that we add j to $\mathbf{PL}_i$ only if $H_i^{(t)}(j) \ge \frac{H_i(D(1))}{300}$ at some time t. Since our stream is insertion-only, we have $H_i'(j) \ge \frac{H_i(D(1))}{300} \ge \frac{G_i}{\mathrm{polylog}(n)}$. Then, there are at most $\mathrm{polylog}(n)$ items in $\mathbf{PL}_i$ in expectation. □

Next, we lower bound the increment of G_i by that of H_i.

Lemma 2.22. *We define H_i as $\sum_{j\in[k]} H_i(j)$. With probability at least $1 - \frac{\eta}{\mathrm{polylog}(n)}$, for all $i \in \mathbf{PL}$, we have that if H_i increases by Δ, then G_i at least increases by $\frac{\eta\Delta}{k\,\mathrm{polylog}(n)}$.*

Proof. Consider a fixed i, due to the distribution of exponential variables, we have for each $\mathbf{e}_i(j)$, it satisfies $\mathbf{e}_i(j) \ge \frac{\eta}{k\,\mathrm{polylog}(n)}$ with probability at least $1 - \frac{\eta}{k\,\mathrm{polylog}(n)}$. By a union bound across the k servers and the $\mathrm{polylog}(n)$ items in $\mathbf{PL}$, we have with probability at least $1 - \frac{\eta}{\mathrm{polylog}(n)}$, $\mathbf{e}_i(j) \ge \frac{\eta}{k\,\mathrm{polylog}(n)}$ for all $i \in \mathbf{PL}$ and $j \in [k]$. Let $\bar{H}_i$ be the value of H_i at some later time, we have

$$\bar{H}_i - H_i = \sum_{j\in[k]} \frac{\bar{G_i}(j) - G_i(j)}{\mathbf{e}_i(j)} < \frac{k\,\mathrm{polylog}(n)}{\eta}\cdot(\bar{G}_i - G_i).$$

where the second step follows by our conditioning on all $\mathbf{e}_i(j)$ larger than $\frac{1}{k\,\mathrm{polylog}(n)}$. Thus, we have our desired result. □

Now we show that H_i increases by at least $\frac{\alpha \cdot G_i}{\mathrm{polylog}(n)}$ when we change the accuracy.

Lemma 2.23. *Let $H_i(D(1))$ and $H_i(D(2))$ denote the first max and the second max of H_i. Suppose that the ratio $\frac{H_i(D(1))}{H_i(D(2))}$ changes from $1+\alpha$ to $1+2\alpha$ or $1+\frac{\alpha}{2}$, then H_i increases by at least $\frac{\alpha \cdot G_i}{\mathrm{polylog}(n)}$ in this duration.*

Proof. If the ratio is reduced from $1+\alpha$ to $1+\alpha/2$, then since our data stream is insertion-only, at least one element $H_i(j)$ increases by $\frac{\alpha}{4} \cdot H_i(D(1))$. Thus, their sum H_i increases by at least $\frac{\alpha}{4} \cdot H_i(D(1))$. Recall that $H_i(j) = \frac{G_i(j)}{\mathbf{e}(j)}$. Then, due to the max-stability property of the exponential variables (c.f. Fact 1.12), we have $H_i(D(1)) = \frac{\sum_{j \in [k]} G_i(j)}{\mathbf{e}} = \frac{G_i}{\mathbf{e}}$, where $\mathbf{e}$ is an independent exponential variable. With our conditioning on the event $\mathcal{B}$, we have $H_i(D(1)) \geq \frac{G_i}{\mathrm{polylog}(n)}$. Therefore, H_i increases by at least $\alpha \cdot \frac{G_i}{\mathrm{polylog}(n)}$. Using the same argument, if the ratio is increased from $1+\alpha$ to $1+2\alpha$, H_i increases by at least $\alpha \cdot \frac{G_i}{\mathrm{polylog}(n)}$ as well. □

Next, we upper bound the communication used to reset the accuracy for the counting algorithm.

Lemma 2.24. *With probability at least $1 - \frac{\eta}{\mathrm{polylog}(n)}$, we use $\frac{k}{\eta} \cdot \mathrm{polylog}(n)$ bits of communication to broadcast the new accuracy of the counting algorithm.*

Proof. In this proof, we condition on the success of Lemma 2.19 and Lemma 2.23, which holds with high probability, by union bounding across the $\mathrm{polylog}(n)$ items in $\mathbf{PL}$, c.f. Lemma 2.18. We also condition on the success of Lemma 2.22, which holds with probability at least $1 - \frac{\eta}{\mathrm{polylog}(n)}$. From Lemma 2.21, we know that there are at most $\mathrm{polylog}(n)$ elements in $\mathbf{PL}_i$. Thus, we only pay $\mathrm{polylog}(n)$ bits of communication to inform all servers in $\mathbf{PL}_i$ each time we change the accuracy. Then it suffices to upper bound the number of times we switch to a new accuracy for a single instance $i \in \mathbf{PL}$.

Now, we consider a fixed instance $i \in \mathbf{PL}$. Let D_l denote the number of times we switch to a new accuracy in cases where, at the moment we last set the current accuracy, the anti-concentration ratio $\frac{H_i(D(1))}{H_i(D(2))}$ lies in the interval $\left[1 + \frac{1}{2^{l+1}}, 1 + \frac{1}{2^l}\right)$. Notice that by the anti-concentration property of exponentials, c.f. Lemma 1.14, the ratio is at least $\frac{1}{\mathrm{poly}(n)}$ at all times with high probability. Moreover, when the ratio is larger than 2, we do not need to change the accuracy since a constant accuracy suffices. Note that the total number of times we change the accuracy is $D = \sum_{l \in [\mathcal{O}(\log n)]} D_l$. Thus, it remains to fix an index $l \in [\mathcal{O}(\log n)]$ and upper bound D_l.

We partition a round into blocks where G_i increases by $\frac{\eta G_i}{10k \cdot 2^l\,\mathrm{polylog}(n)}$. Now we consider a block such that the anti-concentration ratio $\frac{H_i(D(1))}{H_i(D(2))}$ lies in the interval $\left[1 + \frac{1}{2^{l+1}}, 1 + \frac{1}{2^l}\right)$ at some time during this block. Let this ratio be $1+\alpha$. By Lemma 2.23, H_i increases by at least $\frac{G_i}{2^{l+1}\,\mathrm{polylog}(n)}$ if the ratio changes to $1+2\alpha$ or $1+\frac{\alpha}{2}$. Then by Lemma 2.22, we have that G_i increases by at least $\frac{\eta G_i}{k \cdot 2^{l+1}\,\mathrm{polylog}(n)}$ in the duration when H_i increases by at least $\frac{G_i}{2^{l+1}\,\mathrm{polylog}(n)}$. However, G_i increases only by $\frac{\eta G_i}{10k \cdot 2^l\,\mathrm{polylog}(n)}$ at the end of this block. Thus, if the ratio is $1+\alpha$ where $\alpha \in \left[1 + \frac{1}{2^{l+1}}, 1 + \frac{1}{2^l}\right)$ at time t of some block and we broadcast a new accuracy at t, the next switch of accuracy does not

happen in the same block. Therefore, D_l is upper bounded by the number of blocks such that the ratio ever falls to $\left[1+\frac{1}{2^{l+1}}, 1+\frac{1}{2^l}\right)$.

Recall that by Lemma 2.19, we have that G_i increases by at most $G_i \cdot \text{polylog}(n)$ in a round. Then, since G_i increases by $\frac{\eta G_i}{10k \cdot 2^l \text{ polylog}(n)}$ in each block, there are at most $\frac{k}{\eta} \cdot 2^l \text{ polylog}(n)$ blocks in a round. Moreover, if the ratio lies in the interval $\left[1+\frac{1}{2^{l+1}}, 1+\frac{1}{2^l}\right)$ at time t in some block, then it is at most $1+\frac{1}{2^{l-1}}$ at the beginning of this block. Otherwise, H_i increases by at least $\frac{G_i}{2^{l-1} \text{ polylog}(n)}$, which implies that G_i increases by at least $\frac{\eta G_i}{k 2^{l-1} \text{ polylog}(n)}$, so we are in another block at t.

Now, we consider the list of start times $T = \{t_1, \ldots t_B\}$ of each block. Consider a fixed time $t_b \in T$, due to the anti-concentration property of exponentials (c.f. Lemma 1.14), the ratio $\frac{H_i(D(1))}{H_i(D(2))}$ is smaller than $1+\frac{1}{2^{l-1}}$ with probability at most $\mathcal{O}\left(\frac{1}{2^{l-1}}\right)$. Here, we remark that since we define the start times of each block by the increment of G_i, which is independent of the scaled variables $\mathbf{e}(j)$, the anti-concentration property of $\mathbf{e}(j)$ does not depend on t_b. Then, let X_b be an indicator variable that is 1 if the ratio is smaller than $1+\frac{1}{2^{l-1}}$ at t_b, and 0 otherwise, and let $X = \sum_{b\in[B]} X_b$. Then, X gives an upper bound on the number of blocks such that the ratio ever falls to $\left[1+\frac{1}{2^{l+1}}, 1+\frac{1}{2^l}\right)$. Since $\mathbb{E}[X_b] \leq \mathcal{O}\left(\frac{1}{2^{l-1}}\right)$, we have

$$\mathbb{E}[X] = \sum_{b\in[B]} \mathbb{E}[X_b] \leq B \cdot \mathcal{O}\left(\frac{1}{2^{l-1}}\right).$$

Recall that the total number of blocks is $B \leq \frac{k}{\eta} \cdot 2^l \text{ polylog}(n)$, then we have

$$\mathbb{E}[X] = \frac{k}{\eta} \cdot 2^l \text{ polylog}(n) \cdot \mathcal{O}\left(\frac{1}{2^{l-1}}\right) = \frac{k}{\eta} \cdot \text{polylog}(n).$$

Therefore, by Markov's inequality $D_l \leq X \leq \frac{k}{\eta} \cdot \text{polylog}(n)$ with probability at least $1 - \frac{\eta}{\text{polylog}(n)}$. Summing up all $\mathcal{O}(\log n)$ levels, $D = \sum_{l\in[\mathcal{O}(\log n)]} D_l \leq \frac{k}{\eta} \cdot \text{polylog}(n)$, showing our final statement. We note that we can union bound the failure probability across the $\text{polylog}(n)$ elements in **PL**. □

Given the above bounds, we can upper bound the total communication used by the approximate counting algorithms and the corresponding procedures used to maintain the correct accuracies for the approximate counting algorithms.

Lemma 2.25. *We use at most $k \cdot \text{polylog}(n)$ bits of communication to run the counting algorithm in expectation.*

Proof. Consider coordinate i in **PL**. By our algorithm construction, when the anti-concentration ratio $\frac{H_i(D(1))}{H_i(D(2))}$ lies in the interval $\left[1+\frac{1}{2^{l+1}}, 1+\frac{1}{2^l}\right)$, we track all servers $j \in \mathbf{PL}_j$ using a counting algorithm with accuracy $\frac{1}{100p \cdot 2^{l+1}}$. In the counting algorithm with accuracy α, each update is sent with probability $\mathcal{O}\left(\frac{1}{\alpha n}\right)$, where n is the total count, which is $H_i(j)$ in our case. Recall that from the analysis in Lemma 2.24, we have the ratio is smaller than $1+\frac{1}{2^l}$ for at most $\mathcal{O}\left(\frac{1}{2^l}\right)$-fraction of blocks during a round in expectation. Then, the expected number of updates sent by the counting algorithm is smaller than

$$\sum_{l\in[\mathcal{O}(\log n)]} \mathcal{O}\left(\frac{1}{2^l}\right) \cdot H_i(j) \cdot \mathcal{O}\left(\frac{100p \cdot 2^l}{H_i(j)}\right) = \mathcal{O}(\log n).$$

By Lemma 2.21, there is at most $\text{polylog}(n)$ servers in $\mathbf{PL}_i$, giving us $\text{polylog}(n)$ in total.

For $j \notin \mathbf{PL}_i$, it suffices to run a counting algorithm with accuracy 1.01 on those servers, which uses $k\,\text{polylog}(n)$ bits of communication in total. In addition, we run $\text{polylog}(n)$ separate instances of the counting algorithm for each $i \in \mathbf{PL}$ to estimate G_i, which uses an extra $k \cdot \text{polylog}(n)$ bits of communication. □

Combining the results above, we can upper bound the communication cost of our L_p sampler.

Lemma 2.26. *Algorithm 4 uses $(k + k^{p-1}) \cdot \text{polylog}(n)$ bits of communication in expectation.*

Proof. The result follows from the combination of Theorem 2.7, Lemma 2.24 and Lemma 2.25.

To sum up, Theorem 2.7 shows that we use $(k + k^{p-1}) \cdot \text{polylog}(n)$ bits of communication in expectation to track the set $\mathbf{PL}$. Lemma 2.24 shows that with probability at least $1 - \frac{\eta}{\text{polylog}(n)}$, we use $\frac{k}{\eta} \cdot \text{polylog}(n)$ bits of communication to broadcast the new accuracy of the counting algorithm. Lemma 2.25 shows that we use $k \cdot \text{polylog}(n)$ bits of communication in expectation to run the counting algorithm at each server. Let X be the random variable denoting the communication for Lemma 2.24. Conditioned on $X < \text{poly}(n)$, which happens with high probability, we have

$$\begin{aligned}
\mathbb{E}[X] &\leq \sum_{b \in [\log n]} \mathcal{O}\left(2^b\right) \cdot \mathbf{Pr}\left[X \in \left[2^b, 2^{b+1}\right]\right] \\
&\leq \sum_{b \in [\log n]} \mathcal{O}\left(2^b\right) \cdot \mathbf{Pr}\left[X \geq 2^b\right] \\
&\leq \sum_{b \in [\log n]} \mathcal{O}\left(2^b\right) \cdot \frac{k\,\text{polylog}(n)}{2^b} = k\,\text{polylog}(n).
\end{aligned}$$

Thus, we need $(k + k^{p-1}) \cdot \text{polylog}(n)$ bits of communication in expectation in total. □

We conclude the result of our continuous perfect L_p sampler, given the correctness of Theorem 2.16 and Lemma 2.26.

Theorem 2.27. *Let vector $f \in \mathbb{R}^n$ be induced by a distributed stream S, let k denote the total number of servers, and let $p \geq 1$. Then, there is an algorithm LpSampler that gives a perfect L_p sampler of f, using at most $(k + k^{p-1}) \cdot \text{polylog}(n)$ bits of communication in expectation.*

2.4 Lower Bounds for Perfect Sampling

In this section, we show that any perfect L_p sampler requires $\Omega(k^{p-1})$ bits of communication. We first recall the multi-party set disjointness, where k servers hold subsets $S_1, \ldots, S_k \subseteq [n]$. In the YES case, the subsets are all pairwise disjoint, so that there is no coordinate shared among any two servers. In the NO case, there exists a single coordinate that is shared across all k servers, i.e., there exists $i \in [n]$ such that $i \in S_1 \cap \ldots \cap S_k$. It is known that the multi-party set disjointness communication problem requires $\Omega\left(\frac{n}{k}\right)$ communication:

Theorem 2.28. *[Jay09] Any protocol that solves the multi-party set disjointness problem with probability at least $\frac{3}{4}$ requires $\Omega\left(\frac{n}{k}\right)$ bits of communication.*

By taking a constant number of perfect L_p samples, we can identify any item shared across k servers, for a universe of size $n = k^p$. This translates to a lower bound of $\Omega(k^{p-1})$ for perfect L_p sampling.

Theorem 1.3. *Any perfect L_p sampler for $p \geq 1$ must use $\Omega(k^{\max(1,p-1)})$ bits of communication.*

Proof. Given sets $S_1, \ldots, S_k \subseteq [k^p]$, let $f(1), \ldots, f(k) \in \mathbb{R}^n$ denote the indicator vectors so that for all $i \in [n]$ and $j \in [k]$, we have $f_i(j) = 1$ if and only if $i \in S_j$. Let $f = f(1) + \ldots + f(k)$. Now, if there exists an item $i \in [k^p]$ such that $i \in S_1, \ldots, S_k$, then $f_i = k$. Moreover, the other $k^p - 1$ coordinates have magnitude at most 1. Thus in this case, any perfect L_p sampler on f selects i with probability at least $\frac{k^p}{k^p + k^p - 1} \geq \frac{1}{3}$. Hence, with 4 independent instantiations of perfect L_p samplers, we will sample i with probability at least $1 - \left(1 - \frac{1}{3}\right)^4 = \frac{65}{81} \geq \frac{3}{4}$. We can then query all k servers for the values of the sampled indices, which takes an additional $\mathcal{O}(k)$ communication, and check whether any of the sampled indices appears across all k servers. Note that this also handles the case where all servers have disjoint pairwise support. Thus, we can solve multi-party set disjointness across k servers on a universe of size k^p with probability at least $\frac{3}{4}$ using a constant number of perfect L_p servers. By Theorem 2.28, it therefore follows that any perfect L_p sampler requires $\Omega(k^{p-1})$ bits of communication. □

3 Framework for Adversarially Robust Distributed Monitoring Algorithms

In this section, we describe two frameworks that achieve adversarially robust distributed monitoring protocols. In Section 3.1, we first describe a simple framework that transforms non-robust distributed monitoring protocols into robust distributed monitoring protocols, at the cost of sub-optimal communication. In Section 3.2, we then describe a more sophisticated framework that achieves near-optimal communication for adversarially robust distributed monitoring, but requires more algorithmic ingredients.

3.1 Warm-up: Framework by Sketch Switching

We start by describing the sketch switching framework. In this approach, we initialize several independent sketches to track the value of F. To prevent the adversary from affecting the output of the algorithm, the estimate of the i-th sketch is not revealed until the value of F becomes at least $(1+\varepsilon)^i$. Then, the algorithm repeatedly outputs this value until the $(i+1)$-th subroutine has value at least $(1+\varepsilon)^{i+1}$. Hence, the algorithm will always give a $(1+\varepsilon)$-approximation to F, and the adversary knows nothing about the internal randomness of a subroutine until it is revealed. The framework appears in full in Algorithm 5.

To formally analyze our algorithm, we first recall the definition of strong tracker:

Definition 3.1 (Strong tracker, Definition 1.11 in [WZ21b])**.** *Given a distributed data stream S, a fixed time t_0, a splitting time t_1, a frequency vector u induced by updates from time t_0 to t_1, a ε-strong tracker gives a $(1+\varepsilon)$-approximation of $F(u)$ at all times with probability at least $1 - \frac{1}{\text{poly}(n)}$. We use $\mathcal{F}(t_0, t_1, \varepsilon)$ to denote a ε-strong tracker.*

In this section, we shall generally set t_0 to 0, so that the strong tracker will simply estimate the value of F on the frequency vector defined by the entire stream at each time t_1.

Now, we define the flip number of a function F, which controls the time it changes proportionally in the data stream.

Definition 3.2 (Flip number, Definition 3.1 of [BEJWY20]). *Let $\varepsilon \geq 0$. Let $x = (x_0, \ldots, x_m)$ for some $m \in \mathbb{N}$ be a sequence of real numbers. We define the (ε, m)-flip number $\lambda_{\varepsilon,m}(x)$ of x to be the maximum $k \in \mathbb{N}$ such that there exist $0 \leq i_1 < \cdots < i_k \leq m$ so that $x_{i_j - 1} \notin x_i \cdot (1 \pm \varepsilon)$ for all $j = 2, 3, \ldots k$.*

Fix a function $F : \mathbb{R}^n \to \mathbb{R}$ and a class $\mathcal{C} \subseteq ([n] \times \mathbb{Z})^m$ of distributed stream updates, which has the form of $((u_1, i_1), \ldots, (u_m, i_m))$. Here, u_j is the local server that receives the item, i_j is the index in the induced vector. We define the (ε, m)-flip number $\lambda_{\varepsilon,m}(F)$ of F over $\mathcal{C}$ to be the maximum, over all sequences , of the ε-flip number of the sequence $y = (y_0, y_1, \ldots, y_m)$ defined by $y_j = F(1, j)$ for all $j \in [m]$, where $(1, j)$ denotes the frequency vector induced by the distributed stream updates $(u_1, i_1), \ldots, (u_j, i_j)$ (and $f^{(0)}$ is the n-dimensional zeros vector).

This lemma shows that we can approximate a vector by another vector with a small number of unique coordinates, which is applied to define our algorithmic output.

Lemma 3.3 (Lemma 3.2 of [BEJWY20]). *Let $\varepsilon \in (0, 1)$. Suppose that $u = (u_0, \ldots, u_m)$, $v = (v_0, \ldots, v_m)$, and $w = (w_0, \ldots, w_m)$, such that (1) $v_i = u_i \cdot (1 \pm \frac{\varepsilon}{8})$ for all $i \in [m]$, (2) $w_0 = v_0$, and if $w_{i-1} = v_i \cdot (1 \pm \varepsilon/2)$ then $w_i = v_{i-1}$, otherwise $w_i = v_i$. Then $w_i = u_i \cdot (1 \pm \varepsilon)$ for all $i \in [m]$, and moreover, $\lambda_0(w) \leq \lambda_{\varepsilon/8,m}(u)$. In particular, if $u_j = F(1, j)$ where $(1, j)$ denotes the frequency vector induced by the distributed stream updates $(u_1, i_1), \ldots, (u_j, i_j)$ for all $j \in [m]$, then $\lambda_0(w) \leq \lambda_{\varepsilon/8,m}(u)$.*

Algorithm 5 Framework for Robust Distributed Streaming Algorithms via Sketch Switching

Input: Updates $(u_1, i_1), \cdots, (u_m, i_m) \in \mathbb{R}^{[n] \times [k]}$ for a distributed data stream, i_t denotes the index of the insertion, accuracy $\varepsilon \in (0, 1)$, ε-strong tracker $\mathcal{F}$ of F

Output: Adversarially robust $(1 + \varepsilon)$-approximation of F

$\lambda \leftarrow \lambda_{\varepsilon/8,m}(F)$, start λ independent instances $\mathcal{F}_1, \cdots, \mathcal{F}_\lambda$ of $\frac{\varepsilon}{8}$-strong tracker of F
$\rho \leftarrow 1$, $Z \leftarrow 0$
for each update (u_t, i_t) **do**
    Update $\mathcal{F}_1, \cdots, \mathcal{F}_\lambda$ with (u_t, i_t)
    $y \leftarrow$ current output of $\mathcal{F}_\rho$
    if $Z \notin y \cdot (1 \pm \frac{\varepsilon}{2})$ **then**
        $Z \leftarrow y$
        $\rho \leftarrow \rho + 1$ ▷Switch the sketch
    Return Z

We now describe the full guarantees of the sketch switching framework.

Theorem 3.4 (Sketch switching framework, analogous to Theorem 3.5 in [BEJWY20]). *Let n be the cardinality of the universe, let k be the number of the servers, let m be the stream length where $\log m = \mathcal{O}(\log n)$. Suppose there exists a ε-strong tracker $\mathcal{F}$ of F that uses $H(n, k, \varepsilon)$ bits of communication, where the function H depends on the problem F. Then, there exists an adversarially robust streaming algorithm that outputs a $(1 + \varepsilon)$-approximation to F at all times with probability $1 - \frac{1}{\text{poly}(n)}$. Let $\lambda = \lambda_{\varepsilon/8,m}(F)$. The total communication cost of the algorithm is $\mathcal{O}(\lambda \cdot H(n, k, \varepsilon))$ bits.*

Proof. Without loss of generality, we can assume the adversary against a fixed randomized Algorithm 5 is deterministic. This is because given a randomized adversary and algorithm, if the

adversary succeeds with probability greater than δ in fooling the algorithm, there must exist a protocol with deterministic inputs of the adversary which fools $\mathcal{A}$ with probability greater than δ over the randomness of Algorithm 5. Also, conditioned on a fixing of the randomness for both the algorithm and adversary, the entire stream and behavior of both parties is fixed. Thus, let Z_t is the output of the streaming algorithm at time t, given $Z_1, Z_2, \ldots, Z_k$ and the stream updates $(u_1, i_1), \ldots, (u_k, i_k)$ so far, the next stream update (u_{k+1}, i_{k+1}) is deterministically fixed.

Now, consider the first instance $\mathcal{F}_1$. Let t_1 be the time that $Z \notin y \cdot (1 \pm \frac{\varepsilon}{2})$, where y is the estimation of $\mathcal{F}_1$. Then it satisfies that $Z \notin y \cdot (1 \pm \frac{\varepsilon}{2})$ when $t \in [1, t_1 - 1]$. Moreover, since $\mathcal{F}_1$ is a $\frac{\varepsilon}{8}$-strong tracker, it gives a $(1 + \frac{\varepsilon}{8})$ to F at time t_1 when we reveal $\mathcal{F}_1$ and set the output Z to be $\mathcal{F}_1(1, t)$. Observe that when we change the output Z to $\mathcal{F}_1(1, t)$, we deactivate $\mathcal{F}_1$ and activate $\mathcal{F}_2$ to trace the following distributed stream. Since $\mathcal{F}_2$ is not revealed before, its randomness remains hidden from the adversary. Then, by an inductive argument we can show that at time t_{l-1} when the $l-1$-th sketch is revealed and the l-th sketch is activate. We have that $Z \notin y \cdot (1 \pm \frac{\varepsilon}{2})$ when $t \in [t_{l-1}, t_l]$ and $\mathcal{F}_l$ gives a $(1 + \frac{\varepsilon}{8})$ to F at time t_l. Notice that we can union bound the failure probability of all sketches. Therefore, applying Lemma 3.3 with $u = (F(1,1), F(1,2), \ldots, F(1,m))$, $v = (F(1,1), \mathcal{F}_1(1,2), \ldots, \mathcal{F}_1(1,t_1), \mathcal{F}_2(1,t_1+1), \ldots, \mathcal{F}_1(1,t_2), \ldots)$, and w being the output of Algorithm 5, we have that the algorithm returns a $(1+\varepsilon)$-approximation to the function value F at all times with high probability. Also, the flip number of w satisfies $\lambda_0(w) \le \lambda_{\varepsilon/8,m}(F)$. Therefore, it is easy to see it suffices to implement $\lambda_{\varepsilon/8,m}(F)$ sketches in total so that our communication cost is $\mathcal{O}\left(\lambda_{\varepsilon/8,m}(F) \cdot H(n,k,\varepsilon)\right)$ in bits. □

3.2 Framework by Difference Estimator

In this section, we present our framework for adversarially robust distributed monitoring through the usage of difference estimators, which yields optimal results. We first recall the following definition of a difference estimator introduced by [WZ21b]:

Definition 3.5 (Difference estimator)**.** *Suppose that we are given a distributed data stream S, a splitting time t_0, a frequency vector u induced by updates from time 0 to t_0, a frequency vector v induced by updates from time t_0 to t, an accuracy parameter $\varepsilon \in (0,1)$, and a ratio parameter $\gamma \in (0,1]$. Suppose $F(u+v) - F(u) \le \gamma \cdot F(u)$ and $F(v) \le \gamma F(u)$. Then a (γ, ε) difference estimator for a function F outputs an additive $\varepsilon \cdot F(u)$ approximation to $F(u+v) - F(u)$ with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$.*

We use continuous difference estimator to refer to a distributed functional monitoring protocol where the frequency vector v evolves over time and we use static difference estimator to refer to a protocol for the one-shot setting where v is fixed. We do not show the existence of difference estimators for various problems of interest; for these, we defer to Section 4. We also introduce the following concept of a distributed algorithm that flags each time the value of F roughly doubles:

Definition 3.6 (Double detector)**.** *Given a distributed data stream S, a starting time t_0, a splitting time t_1, an evolving frequency vector u induced by updates from time t_0 to t_1, an ε-double detector flags at some time t^* where $F(u)$ satisfies $0.99 \cdot 2^a \le F \le 1.01 \cdot 2^a$ for $a \in \mathbb{N}$. Moreover, it outputs a $(1+\varepsilon)$-approximation of $F(u)$ at t^* with failure probability $1 - \frac{1}{\mathrm{poly}(n)}$.*

Our framework progresses in a number of rounds, switching to a new round whenever the double detector discovers that F has increased by roughly a factor of 2. In this paper, we assume that F at

most doubles $\mathcal{O}(\log n)$ times through out the entire stream with length $m = \text{poly}(n)$. We use the counter a to count the number of rounds.

Consider a fixed round, let u be the vector at its initialization and let $u + v$ be the underlying frequency vector for the full data stream at some later time. The key intuition of the difference estimator framework is that, when $F(u+v) - F(u)$ increases by roughly $2^j \varepsilon \cdot F(u)$, it suffices to keep a $(1 + \frac{1}{2^j})$-approximation to the increment, i.e., an additive $\varepsilon \cdot F(u)$ error in total. Let $v_{\le j} = \sum_{l=1}^{j} v_l$ where $v_1 \dots v_\beta$ arrive in subsequential order. Then, we can decompose

$$F(u+v) = F(u) + \sum_{j=1}^{\beta} \Big(F(u + v_{\le j}) - F(u + v_{\le j-1}) \Big),$$

where we adopt the convention that $u + v_{\le 0} = u$ and $u + v_{\le \beta} = u + v$. Moreover, suppose that $F(u + v_{\le j}) - F(u + v_{\le j-1}) \le 2^j \varepsilon F(u)$ for all j. Then, we can estimate $F(u + v_{\le j}) - F(u + v_{\le j-1})$ by a difference estimator with accuracy $\frac{1}{2^j}$.

In our algorithmic implementation, we use the counter b to count the number of blocks in one round, where a block is defined as an interval in which F increases by roughly $\varepsilon \cdot F(u)$. Suppose that t_a is the most recent time that the double detector $\mathcal{M}$ reported. We use the counter Z to stitch together the sketches of the difference estimator. In particular, let the binary expression of $b + 1 = \sum_{j=1}^{r} 2^{z_j}$. Then, to estimate $F(1, t) - F(1, t_a)$ with block size b, the algorithm stitches together estimate $Z_{a,j}$ for $F(1, t_{a,j}) - F(1, t_{a,j-1})$ with the corresponding granularity. Specifically, for the first $r - 1$ difference $F(1, t_{a,j}) - F(1, t_{a,j-1})$, we use a static estimators as these difference estimators have already been frozen and for the last part we shall use a continuous difference estimator with granularity 0. Whenever the value of b increases, we update the start time and end time of the corresponding difference estimators.

When $b + 1$ is even, we slightly modify the binary expression to ensure that the ongoing sketch is at granularity 0. This is beneficial because we only need a constant approximation at granularity 0, whose communication cost avoids $\frac{1}{\varepsilon}$ factors. In particular, the resulting estimate to the true value of the function has error roughly $\mathcal{O}(\varepsilon) \cdot F(1, t_a)$. Thus when the estimated value increases by roughly $\varepsilon \cdot F(1, t_a)$, we increase the value of b, freeze the current sketch, and set up the start time of the new sketches.

However, at this point, we may need a difference estimator with higher granularity to avoid the error accumulating. Notice that the centralized coordinator sends the stopping time each to the servers whenever a sketch is frozen, so the subsequent vector is fixed and can be retrieved from each server. Thus, we can apply an (ε, γ)-static difference estimator. The framework appears in full in Algorithm 6.

Now, we analyze the correctness of our framework. The proof is analogous to that of [WZ21b]. For simplicity, in the proof, we use "difference estimator" to refer to both the continuous version and the static version since they have the same accuracy. In our algorithm, we use $\mathcal{B}(t_0, t, \gamma, \varepsilon)$ to denote a difference estimator, where t_0 is the split time of the prefix vector u, and t is the current time of the data stream. The following lemma shows that during the activated duration of a (γ, ε)-difference estimator, F at most increases by $\frac{\gamma}{2} F_{\text{start}}$ additively, where F_{start} is the value of F at the start of the round, so that the difference estimator is always valid.

Lemma 3.7 (Success of difference estimators)**.** *Suppose all subroutines $\mathcal{B}$ and $\mathcal{M}$ succeed. Consider a fixed round a, let t_a denote its start time, let $F(1, t_a)$ be the function value at time t_a. For each integer $r \ge 0$, let $t_{a,r}$ and $w_{a,r}$ be the start time and end time of difference estimator $Z_{a,r}$. We have $F(1, w_{a,r}) - F(1, t_{a,r}) \le \frac{2^r \varepsilon}{2} \cdot F(1, t_a)$, and all difference estimators give a valid estimation.*

Algorithm 6 Framework for Robust Distributed Algorithms via Difference Estimator

Input: Updates $(u_1, i_1), \cdots, (u_m, i_m) \in \mathbb{R}^{[n]\times[k]}$ arriving in a distributed data stream, accuracy $\varepsilon \in (0,1)$, $(\varepsilon,\varepsilon)$-continuous estimator $\mathcal{B}_C$, (γ,ε)-static estimator $\mathcal{B}_S$, ε-double detector $\mathcal{M}$
Output: Adversarially robust $(1+\varepsilon)$-approximation of F

$\beta \leftarrow \lceil \log \frac{8}{\varepsilon} \rceil$, $\varepsilon' \leftarrow \frac{\varepsilon}{64\beta}$, $a \leftarrow 0$, $\widehat{F} \leftarrow 0$
For $j \in [\beta]$, $\gamma_j \leftarrow 2^j\varepsilon$ ▷Accuracy of each difference estimator
for each update (u_t, i_t) **do**
  Update $\mathcal{M}_{a+1}(1, t, \frac{\varepsilon}{100})$ with (u_t, i_t) ▷Double detector
  if $\mathcal{M}_{a+1}(1, t, \frac{\varepsilon}{100})$ returns **FLAG then** ▷Switch to a new round when F duplicates
    $\widehat{F} \leftarrow \mathcal{M}_{a+1}(1, t, \frac{\varepsilon}{100})$ ▷Run the double detector to give a $(1+\varepsilon)$-approximation
    $a \leftarrow a+1$, $b \leftarrow 0$, $Z_a \leftarrow \widehat{F}$, $t_{a,j} \leftarrow t$ for $j \in [\beta]$
  if $b+1$ is odd **then**
    $r \leftarrow \text{numbits}(b+1), z_i \leftarrow \text{lsb}(b+1, r+1-i)$ for all $i \in [r]$ ▷$z_1 > \cdots > z_r$ are the nonzero bits in the binary representation of b
  else
    $r \leftarrow \text{numbits}(b)+1, z_i \leftarrow \text{lsb}(b, r+1-i)$ for all $i \in [r-1]$, $z_r \leftarrow 0$ ▷Make sure the ongoing sketch is a difference estimator at granularity 0, i.e., $\gamma = \varepsilon$
  $\widehat{F} \leftarrow Z_a$
  for $0 \le j \le r-1$ **do** ▷Sum up previous frozen sketches to estimate F
    $\widehat{F} \leftarrow \widehat{F} + Z_{a,z_j}$
  $j \leftarrow b+1$
  $\widehat{F} \leftarrow \widehat{F} + \mathcal{B}_{C_j}(t_{a,z_r}, t, \varepsilon', \varepsilon')$ ▷Use continuous difference estimator for the ongoing sketch
  if $\widehat{F} > (1+\frac{(b+1)\varepsilon}{8}) \cdot Z_a$ **then** ▷Switch to the next block in the same round
    $b \leftarrow b+1$, $r \leftarrow \text{numbits}(b), z_r \leftarrow \text{lsb}(b,1)$, $j \leftarrow \lfloor \frac{b}{2^{z_r}} \rfloor$
    $Z_{a,r} \leftarrow \mathcal{B}_{S_j}(t_{a,z_r}, t, \gamma_{z_r}, \varepsilon')$ ▷Use static difference estimator for the frozen sketch
    $t_{a,j} \leftarrow t$ for $j \in [z_r]$ ▷Update difference estimator times
  return $(1+\frac{b\varepsilon}{8}) \cdot Z_a$

Proof. Fix a block b. Consider the binary representation of b:

$$b = \sum_{i=1}^{r} 2^{z_i}, \text{ where } z_1 > \cdots > z_r > 0, r = \text{numbits}(b),$$

such that the difference estimator $\mathcal{B}_{a,z_r}(t_{a,z_r}, w_{a,z_r}, \gamma_{z_r}, \varepsilon')$ is running in block b to estimate $F(1, w_{a,z_r}) - F(1, t_{a,z_r})$. Now, suppose by way of contradiction we have $F(1, w_{a,z_r}) - F(1, t_{a,z_r}) > \frac{2^r\varepsilon}{2} \cdot F(1, t_a)$. Then, at some time $t < w_{a,z_r}$, it satisfies that

$$F(1,t) - F(1, t_{a,z_r}) < \frac{2^r\varepsilon}{2} F(1, t_a) \text{ and } F(1, t+1) - F(1, t_{a,z_r}) \ge \frac{2^r\varepsilon}{2} F(1, t_a).$$

Since we only receive one update at time t, so we have $F(1,t) - F(1, t_{a,z_r}) > \frac{2^r\varepsilon}{4} F(1, t_a)$. Then, because we suppose all procedures $\mathcal{B}$ succeed and the actual difference has not passed $\frac{2^r\varepsilon}{2} F(1, t_a)$, by definition our difference estimator $\mathcal{B}_{a,z_r}(t_{a,z_r}, w_{a,z_r}, \gamma_{z_r}, \varepsilon')$ gives an estimation with additive error

$\varepsilon' \cdot F(1, t_a)$. So, we have

$$Z_{a,z_r} \geq F(1, w_{a,z_r}) - F(1, t_{a,z_r}) - \varepsilon' \cdot F(1, t_a) \geq \frac{2^r \varepsilon}{4} F(1, t_a).$$

Notice that our criterion for switching to a new difference estimator is the estimated value increases by $\frac{(b-b')\varepsilon}{8} \cdot Z_a = \frac{2^{z_r}\varepsilon}{8} \cdot Z_a$. Since Z_a is a $(1 + \frac{\varepsilon}{100})$-approximation to $F(1, t_a)$, $Z_{a,z_r} > \frac{2^r \varepsilon}{5} Z_a$ already passes the threshold. Thus, $\mathcal{B}_{a,z_r}$ should have been switched to a new difference estimator, which is a contradiction. Therefore, $F(1, w_{a,r}) - F(1, t_{a,r}) \leq \frac{2^r \varepsilon}{2} \cdot F(1, t_a)$, and all difference estimators give a valid estimation. □

The following lemma shows that for each difference estimator with granularity r, the value F increases by roughly a $2^r\varepsilon$-fraction after it completes, which further upper bounds the number of difference estimators that are used in each round. This statement will eventually be used to bound the estimation error and the communication cost of the framework.

Lemma 3.8 (Upper bound on the number of difference estimators)**.** *Suppose all subroutines $\mathcal{B}$ and $\mathcal{M}$ succeed. Consider a fixed round a, let t_a denote its start time, let $F(1, t_a)$ be the function value at time t_a. For each integer $r \geq 0$, let $t_{a,r}$ and $w_{a,r}$ be the start time and end time of difference estimator $Z_{a,r}$. We have $F(1, w_{a,r}) - F(1, t_{a,r}) \geq \frac{2^r \varepsilon}{4} \cdot F(1, t_a)$.*

Proof. As before, we fix a block b. Consider the binary representation of b:

$$b = \sum_{i=1}^{r} 2^{z_i}, \text{ where } z_1 > \cdots > z_r > 0, r = \text{numbits}(b),$$

such that the difference estimator $\mathcal{B}_{a,z_r}(t_{a,z_r}, w_{a,z_r}, \gamma_{z_r}, \varepsilon')$ is running in block b to estimate $F(1, w_{a,z_r}) - F(1, t_{a,z_r})$. By Lemma 3.7, our difference estimator $\mathcal{B}_{a,z_r}(t_{a,z_r}, w_{a,z_r}, \gamma_{z_r}, \varepsilon')$ outputs an estimation with additive error $\varepsilon' \cdot F(1, t_i)$. Let $b' = \sum_{i=1}^{r-1} 2^{z_i}$ such that $\mathcal{B}_{a,z_r}(t_{a,z_r}, w_{a,z_r}, \gamma_{z_r}, \varepsilon')$ starts at the end of block b'. Note that our criterion for switching to a new difference estimator is the estimated value increases by $\frac{(b-b')\varepsilon}{8} \cdot Z_a = \frac{2^{z_r}\varepsilon}{8} \cdot Z_a$. Then, since $\mathcal{B}_{a,z_r}(t_{a,z_r}, w_{a,z_r}, \gamma_{z_r}, \varepsilon')$ is frozen at time w_{a,z_r}, we have

$$\frac{2^{z_r}\varepsilon}{8} \cdot Z_a \leq Z_{a,z_r} \leq F(1, w_{a,z_r}) - F(1, t_{a,z_r}) + \varepsilon' \cdot F(1, t_i).$$

Therefore, since Z_a is a $(1 + \frac{\varepsilon}{100})$-approximation to $F(1, t_a)$, we have

$$\frac{2^{z_r}\varepsilon}{4} \cdot F(1, t_a) \leq F(1, w_{a,z_r}) - F(1, t_{a,z_r}).$$

□

The following lemma shows the correctness of our framework on non-adaptive streams. Thus, it remains to argue that the inner randomness of each difference estimator is hidden from the adversary.

Lemma 3.9 (Correctness on non-adaptive streams)**.** *Suppose all subroutines $\mathcal{B}$ and $\mathcal{M}$ succeed. Then Algorithm 6 gives an $(1 + \varepsilon)$-approximation of the function F at all times.*

Proof. We fix a time t in round a and block b. Consider the binary representation of b:

$$b = \sum_{i=1}^{r} 2^{z_i}, \text{ where } z_1 > \cdots > z_r > 0, r = \text{numbits}(b).$$

Let $t_{a,i}$ denote the time when the counter b is first set to i. Let $t_{a,0}$ denote the start time of round a. Let $\widehat{F}$ denote the current estimation. Notice that our output $(1 + \frac{b\varepsilon}{8}) \cdot Z_a$ is within one block-size from $\bar{F}$. Therefore, we have

$$\left|\left(1 + \frac{b\varepsilon}{8}\right) \cdot Z_a - F\right| \leq \left|\left(1 + \frac{b\varepsilon}{8}\right) \cdot Z_a - \widehat{F}\right| + |\widehat{F} - F| \leq \frac{\varepsilon}{8} \cdot Z_a + |\widehat{F} - F| \tag{1}$$

Now, we analyze the second term. Recall that $\widehat{F}$ is the sum of Z_a and each difference estimator, so by triangle inequality we have

$$|\widehat{F} - F| \leq |Z_a - F(1, t_{a,0})| + \sum_{i=1}^{r} \left|\left(\widehat{F(1, t_{a,r}) - F}(1, t_{a,r-1})\right) - \left(F(1, t_{a,r}) - F(1, t_{a,r-1})\right)\right|.$$

For the first term, our double detector $\mathcal{M}_a(1, t_{a,0}, \frac{\varepsilon}{100})$ outputs a $(1 + \frac{\varepsilon}{100})$-approximation, hence the error is within $\frac{\varepsilon}{100} \cdot F(1, t_{a,0})$. For the second term, by Lemma 3.7 our difference estimator $\mathcal{B}_{a,r}(1, t_{a,r}, t, \gamma_r, \varepsilon')$ gives an estimation with additive error $\varepsilon' \cdot F(1, t_{a,0})$. Then, we have

$$|\widehat{F} - F| \leq \frac{\varepsilon}{100} \cdot F(1, t_{a,0}) + r\varepsilon' \cdot F(1, t_{a,0}).$$

Now, by Lemma 3.8, we know

$$\sum_{i=1}^{r} F(1, t_{a,r}) - F(1, t_{a,r-1}) \geq \sum_{i=1}^{r} \frac{2^r \varepsilon}{4} \cdot F(1, t_{a,0}).$$

By our double detector, F at most increases by a factor of 3 in one round. So, we have

$$\sum_{i=1}^{r} \frac{2^r \varepsilon}{4} \cdot F(1, t_{a,0}) \leq 2F(1, t_{a,0}),$$

which implies $r \leq \log \frac{8}{\varepsilon}$. Therefore, by our choice of ε', we have $r\varepsilon' \leq \frac{\varepsilon}{8}$, so we have $|\widehat{F} - F| \leq \frac{\varepsilon}{4} \cdot F(1, t_{a,0})$. This implies that

$$\left|\left(1 + \frac{b\varepsilon}{8}\right) \cdot Z_a - F\right| \leq \varepsilon \cdot F(1, t_{a,0}) \leq \varepsilon F.$$

□

Now, we give the formal statement of our framework.

Framework 3.10. *Let n be the cardinality of the universe, let k be the number of the servers, let m be the stream length where $\log m = \mathcal{O}(\log n)$. Suppose the flip number of F satisfies $\lambda_{1,m}(F) = \mathcal{O}(\log n)$ Suppose there exists a $(\varepsilon, \varepsilon)$-continuous difference estimator $\mathcal{B}$ of F that takes communication cost*

$$\frac{1}{\varepsilon} \cdot G_1(n, k, \varepsilon) + G_2(n, k, \varepsilon),$$

a (γ, ε)-static difference estimator $\mathcal{B}$ of F that takes communication cost

$$\frac{\gamma}{\varepsilon^2} \cdot G_1(n, k, \varepsilon) + G_2(n, k, \varepsilon),$$

and a ε-double detector $\mathcal{M}$ of F that takes communication cost

$$\frac{1}{\varepsilon^2} \cdot H_1(n, k, \varepsilon) + H_2(n, k, \varepsilon),$$

where functions G_1, G_2, H_1, H_2 depend on the problem F. Then, there exists an adversarially robust streaming algorithm that outputs a $(1+\varepsilon)$-approximation to F at all times with probability $1 - n^{-c}$. The total communication cost of the algorithm is

$$\mathcal{O}\left(\frac{1}{\varepsilon^2}\log n \cdot H_1\left(n, k, \frac{\varepsilon}{100}\right) + \log n \cdot H_2\left(n, k, \frac{\varepsilon}{100}\right)\right) +$$
$$\mathcal{O}\left(\frac{1}{\varepsilon^2}\log n \log^3\frac{1}{\varepsilon} \cdot G_1(n, k, \varepsilon') + \frac{1}{\varepsilon}\log n \log\frac{1}{\varepsilon} \cdot G_2(n, k, \varepsilon')\right) + \mathcal{O}\left(\frac{k}{\varepsilon}\log^2 n \log\frac{1}{\varepsilon}\right).$$

where $\varepsilon' = \mathcal{O}\left(\frac{\varepsilon}{\log\frac{1}{\varepsilon}}\right)$.

Proof. The correctness of the estimation is shown by Lemma 3.9. Observe that each time either counter a or b increase by 1, Algorithm 6 switch to new subroutines $\mathcal{M}$ or $\mathcal{B}$ that have not been previously revealed to the adversary. Thus, the adversarial input is independent of the new subroutines, which gives us adversarially robustness.

Now, we bound the communication cost. Notice that we have $\mathcal{O}(\log n)$ rounds since F at most doubles $\mathcal{O}(\log n)$ times in the stream. In each round, we first run a $\frac{\varepsilon}{100}$-double detector $\mathcal{M}$ of F that takes communication cost

$$\frac{1}{\varepsilon^2} \cdot H_1\left(n, k, \frac{\varepsilon}{100}\right) + H_2\left(n, k, \frac{\varepsilon}{100}\right)$$

At the end time t of each round, the binary expression of counter b is $b = \sum_{i=1}^{r} 2^{z_i}$ where $r = \mathcal{O}\left(\log\frac{1}{\varepsilon}\right)$ by the analysis of Lemma 3.9. Now, let us suppose $b = 2^{r+1}$, which always gives an upper bound. Due to the upper bound on the number of different estimators implied by Lemma 3.8, for blocks at granularity 0, we run $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ separate $(\varepsilon', \varepsilon')$-continuous difference estimators totally in one round; moreover, for blocks at granularity $z_i > 0$, we run $\mathcal{O}\left(\frac{1}{2^{z_i}\varepsilon}\right)$ separate $(\gamma_{z_i}, \varepsilon')$-static difference estimators in total for the round, where $\gamma_{z_i} = 2^{z_i}\varepsilon$. Thus, for our choice of $\varepsilon' = \mathcal{O}\left(\frac{\varepsilon}{\log\frac{1}{\varepsilon}}\right)$, the communication cost is

$$\begin{aligned}
&\mathcal{O}\left(\frac{1}{\varepsilon}\right)\left(\frac{\varepsilon}{\varepsilon'^2} \cdot G_1(n, k, \varepsilon', \delta') + G_2(n, k, \varepsilon', \delta')\right) \\
&+ \sum_{i=1}^{\mathcal{O}\left(\log\frac{1}{\varepsilon}\right)} \mathcal{O}\left(\frac{1}{2^{z_i}\varepsilon}\right)\left(\frac{2^{z_i}\varepsilon}{\varepsilon'^2} \cdot G_1(n, k, \varepsilon', \delta') + G_2(n, k, \varepsilon', \delta')\right) \\
&= \mathcal{O}\left(\frac{1}{\varepsilon^2}\log^3\frac{1}{\varepsilon} \cdot G_1(n, k, \varepsilon', \delta') + \frac{1}{\varepsilon}\log\frac{1}{\varepsilon} \cdot G_2(n, k, \varepsilon', \delta')\right).
\end{aligned}$$

Notice that the central coordinator needs to inform all servers the frozen time for each difference estimator at the bottom level (e.g. difference estimators with constant accuracy). In each round, we run $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ such different estimators. Therefore, this gives us additional $\mathcal{O}\left(\frac{k}{\varepsilon}\log^2 n \log\frac{1}{\varepsilon}\right)$ bits of communications. Combining them together gives us the result. □

4 Adversarially Robust Distributed Streaming Algorithms

In this section, we show several applications of our difference estimator framework presented in Section 3.2. The framework requires both the construction of a double detector and a difference estimator. We propose these subroutines for the counting problem (F_1 estimation problem) in Section 4.1, for the distinct elements estimation problem (F_0 estimation problem) in Section 4.2, for the F_2 estimation problem in Section 4.3, for the F_p estimation problem for $p \geq 2$ in Section 4.4, and for the L_2-heavy-hitters problem in Section 4.5.

We remark that we construct (γ, ε)-continuous difference estimator for all granularities γ in some of the above problems, which generalize static estimators since they can be initiated when a sub-stream begins and finalized when it freezes. In particular, we achieve (γ, ε)-continuous difference estimators for F_1 estimation in Section 4.1, F_0-estimation in Section 4.2, and L_2-heavy-hitters problem in Section 4.5. In our framework in Section 3.2, however, we use static estimators because constructing continuous F_p-difference estimators for $\gamma > \mathcal{O}(\varepsilon)$ is significantly harder.

4.1 Adversarially Robust F_1 Estimation

In this section, we construct an adversarially robust algorithm for the counting problem, i.e., F_1 approximation. We begin by describing the counting algorithm in the classical non-robust distributed setting introduced by [HYZ12]. Let p be a parameter to be determined later. In their algorithm, each server j sends the local count $n(j)$ to the coordinator with probability p whenever it receives a new update. The coordinator maintains an estimator $\widehat{n(j)}$ for each local count $n(j)$. Here, $\widehat{n(j)}$ is defined as $\widetilde{n(j)} - 1 + \frac{1}{p}$ where $\widetilde{n(j)}$ is the last updated value from the server j, it is 0 if $\widetilde{n(j)}$ does not exist. The coordinator outputs $\widehat{n} = \sum_{j=1}^{k} \widehat{n(j)}$ as the estimate of the total count. The next lemma states the guarantee of this estimator.

Lemma 4.1 ([HYZ12]). *For each j, the estimator satisfies $\mathbb{E}\left[\widehat{n(j)}\right] = n(j)$ and $Var\left[\widehat{n(j)}\right] = \frac{1}{p^2}$.*

Taking $p = \frac{\sqrt{k}}{\varepsilon n}$ gives $\text{Var}\left[\widehat{n(j)}\right] = \frac{\varepsilon^2 n^2}{k}$, hence the total variance is $\varepsilon^2 n^2$ summing across all the k estimates. This algorithm is not naturally robust because its correctness requires that the error of each estimate $\widehat{n(j)}$ be independent. If an estimate $\widehat{n(j)}$ is exposed to the adversary, the inputs afterward are not independent of $\widehat{n(j)}$, which implies that the errors of other estimates are not independent of $\widehat{n(j)}$. Then, the variance of the output $\widehat{n} = \sum_{j=1}^{k} \widehat{n(j)}$ may include covariance terms, which does not guarantee correctness.

We show that our difference estimator framework solves this problem while preserving the optimal communication bound $\frac{\sqrt{k}}{\varepsilon} + k$. Let u be the prefix distributed stream, and let v be the subsequent evolving distributed stream, we abuse the notation of u and v to also denote their total count. Recall that we decompose the evolving stream v into a sequence of sub-streams $v_1, \ldots, v_\beta$ with geometrically decreasing count: $u + v = u + \sum_{\ell=1}^{\beta} v_\ell$, where each v_ℓ is roughly $2^\ell \varepsilon u$ and $\beta = \mathcal{O}\left(\log \frac{1}{\varepsilon}\right)$. First, we observe that if we run the algorithm from [HYZ12] on each sub-stream v_ℓ with sampling probability $p = \frac{\sqrt{k}}{\varepsilon u}$, then the variance of the estimator $\widehat{v_\ell(j)}$ on each server is $\frac{\varepsilon^2 u^2}{k}$ by Lemma 4.1, which suffices to give an estimate of the difference v_ℓ with additive error εu. Notice that for each difference estimator, its output is hidden from the adversary before the end of its block, when the coordinator reveals the estimate to the adversary. So, the estimates of the local differences $\widehat{v_\ell(j)}$ are independent of each other, implying the robust result. However, in our standard difference

estimator framework introduced in Section 3.2, the algorithm requires that each server knows the start time of each v_ℓ, which would use $\frac{k}{\varepsilon}$ bits of communication since there are roughly $\frac{1}{\varepsilon}$ difference estimators in one round, and it is prohibitively large for our communication bound.

Instead, for our F_1 estimation algorithm, it is unnecessary to update the start time of each difference estimators to all the k servers. Instead, we apply a lazy update scheme to reduce communication. Recall that in the difference estimator framework, we divide the stream into $\mathcal{O}(\log n)$ rounds where the count doubles in each round. Then, at the beginning of the round, the coordinator sends the count of the prefix stream u, and each server update the sampling probability as $p = \frac{\sqrt{k}}{\varepsilon u}$. Within a round, the coordinator sends the start time of v_ℓ to a server j only if j initiates the communication. That is, if j sends an update to the coordinator in the duration of v_ℓ, then the coordinator informs j of the start time of v_ℓ, and j returns the actual value of $v_\ell(j)$. This only loses constant factors to the communication cost. In addition, this does not change the distribution of the estimator of $v_\ell(j)$, since each server define the sampling probability with the correct count of u, and thus ensures the correctness under the adversarially robust setting. We next present the formal theorem statement and analysis of the above descriptions.

Theorem 4.2 (Adversarially robust distributed streaming counting)**.** *Let n be the total count of items in a distributed stream S, let k be the number of servers, and let $\varepsilon \in (0,1)$ be the accuracy parameter satisfying $\frac{1}{\varepsilon^2} \geq k$. Then, there exists an adversarially robust algorithm that outputs a $(1+\varepsilon)$-approximation to n at all times with probability $1 - \frac{1}{\mathrm{poly}(n)}$. This algorithm uses $\mathcal{O}\left((k + \frac{\sqrt{k}}{\varepsilon}) \log^2 n \log \frac{1}{\varepsilon}\right)$ bits of communication.*

Proof. First, we remark that Lemma 4.1 implies the existence of a ε-double detector that communicates $\mathcal{O}\left((k + \frac{\sqrt{k}}{\varepsilon}) \log n\right)$ bits (c.f. Theorem 2.1 in [HYZ12]).

Next, we consider a fixed difference estimator block. Our lazy update sketch guarantees that in each round, each server samples an update to the coordinator with probability $p = \frac{\sqrt{k}}{\varepsilon u}$, and the coordinator receives the true value of v_j if it receives a sample from j. Then, the distribution of the communication transcript between the coordinator and the servers is the same as running [HYZ12] on v with $p = \frac{\sqrt{k}}{\varepsilon u}$. Hence, inside each separate difference estimator block, the error of each estimate $\widehat{v_j}$ is independent, as the estimates are hidden from the adversary. Thus, the variance of each estimator $\widehat{v_j}$ is $\frac{\varepsilon^2 u^2}{k}$, and the variance of the estimator $\widehat{v}$ is $\varepsilon^2 u^2$, which follows from the analyze framework in Lemma 4.1 and [HYZ12]. Since $\widehat{v}$ is unbiased, by standard concentration inequalities, averaging $\mathcal{O}(\log n)$ instances gives an estimate of the difference v with additive error εu with high probability. Therefore, our algorithm provides a (γ, ε)-continuous difference estimator, then by Famework 3.10, there is an robust algorithm that gives a $(1+\varepsilon)$-approximation of the total count.

Now, we compute the communication cost of a (γ, ε)-difference estimator. With our assumption on $v \leq \gamma u$, we sample each update with probability $\frac{\sqrt{k}}{\varepsilon u} \leq \frac{\gamma\sqrt{k}}{\varepsilon v}$. Since there are v updates in total and we run $\mathcal{O}(\log n)$ instances, the communication cost of our (γ, ε)-continuous difference estimator is $\mathcal{O}\left(\frac{\gamma\sqrt{k}}{\varepsilon} \log n\right)$. Notice that all $\frac{1}{\varepsilon}$ difference estimators in the same round are implemented with the same sampling probability $\frac{\sqrt{k}}{\varepsilon u}$, where u is the count at the beginning of the round. So, we only need to inform each server of the sampling probability once a round, which saves the $\mathcal{O}\left(\frac{k}{\varepsilon} \log n\right)$ factor in our difference estimator framework in Famework 3.10. Therefore, following the calculation in Famework 3.10, the total communication of the adversarially robust algorithm is $\mathcal{O}\left((k + \frac{\sqrt{k}}{\varepsilon}) \log^2 n \log \frac{1}{\varepsilon}\right)$ bits. □

4.2 Adversarially Robust F_0 Estimation

This section provides an adversarially robust algorithm for F_0 estimation. We first describe the subsampling scheme to count the distinct elements in [CMY11]. The algorithm maintains roughly $\frac{1}{\varepsilon^2}$ samples from the distinct elements. When there are more than $\frac{1}{\varepsilon^2}$ samples, it discards each item with probability $\frac{1}{2}$. We iteratively run this procedure until the stream ends so that the number of samples does not pass the threshold. Then at the end of the stream, we retrieve the estimate by rescaling the number of surviving samples by 2^t, where t is the number of times we subsample. To implement the subsampling scheme, each server hashes the incoming elements to $[n]$ by a unique hashing function $f : [n] \to [n]$, and it computes the largest coordinate q in its binary expression. The centralized coordinator keeps a counter t to count the rounds of the subsample procedure. Whenever the algorithm increases a subsampling layer, t increases by 1 and we discard all the elements with $q < t$. We state the formal theorem for the non-robust distributed algorithm to estimate the F_0 norm in a data stream as follows.

Theorem 4.3 ([CMY11])**.** *Let n be the number of items in the universe, let k be the total number of servers, and let $\varepsilon \in (0, 1)$ be the accuracy parameter. Then, there exists an algorithm that, with high probability, outputs a $(1+\varepsilon)$-approximation to the F_0 estimation problem at all times t, using $\mathcal{O}\left(\frac{k}{\varepsilon^2}\log^2 n + \frac{1}{\varepsilon^2}\log n \log\frac{1}{\varepsilon}\right)$ bits of communication.*

We extend the above algorithm to obtain a difference estimator, which is analogous to the construction introduced by [WZ21b] in the centralized streaming setting. The protocol first runs a subsampling procedure on the static vector u to some level t with $\Theta(\frac{\gamma}{\varepsilon^2})$ survivors. Then, it runs the same subsampling procedure on $v - u$ and only counts the additional elements surviving at level t. Recall that our assumption is $F_0(v) - F_0(u) = \gamma F_0(u)$, then in expectation there are $\Theta(\frac{\gamma^2}{\varepsilon^2})$ distinct elements in v but not in u in the t-th subsampling layer. Therefore, rescaling by 2^t gives us a $(1 + \frac{\gamma}{\varepsilon})$-approximation to $F_0(v) - F_0(u)$, which implies an additive error $\varepsilon \cdot F_0(u)$ as we desired. We now present the formal analysis of this algorithm. We first show the correctness of the protocol in the following statement. The proof is analogous to the analysis in Lemma 6.2 in [WZ21b].

Lemma 4.4. *Suppose that $F_0(u + v) - F_0(u) \le \gamma F_0(u)$, then the difference estimator gives an estimation to $F_0(u + v) - F_0(u)$ within additive error $\varepsilon F_0(u)$ with probability $1 - \frac{1}{\mathrm{poly}(n)}$.*

Proof. Consider the deepest layer t where we subsample each item with probability $\frac{1}{2^t}$, we have $\frac{F_0(u)}{2^t} = C \cdot \frac{\gamma}{\varepsilon^2}$. By our assumption on $F_0(v) - F_0(u) \le \gamma F_0(u)$, in expectation we have $\frac{F_0(v)-F_0(u)}{2^t} = C \cdot \frac{\gamma^2}{\varepsilon^2}$. Hence, let X denote the survivors from $F_0(v) - F_0(u)$ at t, we have

$$\mathbb{E}\left[2^t \cdot X\right] = F_0(v) - F_0(u)$$
$$\mathrm{Var}(2^t \cdot X) = 2^t \cdot (F_0(v) - F_0(u)) \le \frac{\varepsilon^2 \cdot F_0(u)}{C\gamma} \cdot (F_0(v) - F_0(u)) \le \frac{\varepsilon^2}{C} \cdot (F_0(u))^2.$$

Therefore, for a sufficiently large C, by Chebyshev's inequality the additive error of our output is $\varepsilon F_0(u)$ with probability at least 0.5. Averaging $\mathcal{O}(\log n)$ repetitions achieves high probability. □

Next, we upper bound the communication cost of this protocol.

Lemma 4.5. *The communication cost of the difference estimator is $\mathcal{O}\left(\frac{k\gamma}{\varepsilon^2}\log^2 n\right)$ bits.*

Proof. It takes $\mathcal{O}(k \log n)$ communications to send the hashing function to each site. In each subsample layer, we sample $\Theta(\frac{\gamma}{\varepsilon^2})$ elements from the stream. Since we have $\mathcal{O}(\log n)$ such layers and k servers, the communication cost is $\mathcal{O}\left(\frac{\gamma}{\varepsilon^2} \cdot k \log n + k \log n\right)$ bits in total. To reach a high probability, we run $\mathcal{O}(\log n)$ repetitions, which gives an additional logarithmic factor. □

Combining Lemma 4.4 and Lemma 4.5, we have the following statement of our difference estimator.

Theorem 4.6. *Let n be the number of items in the universe, let k be the number of servers, and let $\varepsilon \in (0,1)$ be the accuracy parameter. Then, there exists a distributed (γ, ε)-continuous difference estimator for the distinct element problem that uses $\mathcal{O}\left(\frac{k\gamma}{\varepsilon^2} \log^2 n\right)$ bits of communication.*

Given the non-robust distributed F_0 tracker and our difference estimator, we show the following theorem for adversarially robust F_0-estimation.

Theorem 4.7. *Let S be a distributed stream, let n be the number of items in the universe, let k be the number of servers, and let $\varepsilon \in (0,1)$ be the accuracy parameter. Then, there exist an adversarially robust distributed monitoring algorithm that outputs a $(1+\varepsilon)$-approximation to the distinct element problem in stream S. The algorithm uses $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log^3 n \log^3 \frac{1}{\varepsilon}\right)$ bits of communication.*

Proof. By Theorem 4.3, we have a ε-double detector that communicates $\mathcal{O}\left(k \log^2 n + \frac{1}{\varepsilon^2} \log n \log \frac{1}{\varepsilon}\right)$ bits. By Theorem 4.6, we have a (γ, ε)-continuous difference estimator that communicates $\mathcal{O}\left(\frac{k\gamma}{\varepsilon^2} \log^3 n\right)$ bits. Therefore, by Famework 3.10, there is an adversarially robust algorithm that uses $\mathcal{O}\left(\frac{k}{\varepsilon^2} \log^3 n \log^3 \frac{1}{\varepsilon}\right)$ bits of communication. □

4.3 Adversarially Robust F_2 Estimation

In this section, we provide an adversarially robust algorithm for F_2 estimation. We note that the result of F_2 estimation is a direct corollary from the F_p estimation algorithm in Section 4.4. Here, we present a simpler difference estimator using the inner product of u and v. Observe that we can decompose the difference as the sum of the inner product $\langle u, v \rangle$ and the F_2-moment of v:

$$\|u + v\|_2^2 - \|u\|_2^2 = 2\langle u, v \rangle + \|v\|_2^2.$$

Then, to estimate the inner product, we simply obtain L_2 samples from u and compute $u_i \cdot v_i \cdot \frac{\|u\|_2^2}{u_i^2}$. This gives an unbiased estimate with variance bounded by $\gamma \cdot \|u\|_2^4$, and therefore, averaging $\mathcal{O}\left(\frac{\gamma}{\varepsilon^2}\right)$ such estimators would give an estimate with additive error $\varepsilon \cdot \|u\|_2^2$. It may seem unclear how to obtain the estimation of $\|v\|_2^2$ due to the lack of a continuous monitoring algorithm for F_2-estimation with optimal $\frac{1}{\varepsilon}$ factor in the communication cost. However, recall that in our difference estimator framework, we only need a continuous difference estimator for an evolving v at granularity 0, where a constant-multiplicative estimate of $\|v\|_2^2$ is sufficient. For the static difference estimator at a higher granularity, we can simply run the one-shot F_2-estimation algorithm in [EKM+24] for v with a higher accuracy.

Now, we state another result of distributed counting algorithm that gives a constant-multiplicative estimation of the count, which is frequently used in our difference estimator. We remark that this result is a corollary of the estimator in Lemma 4.1.

Theorem 4.8 ([HYZ12])**.** *Let n be the total count, let k be the number of servers. There is a procedure* Counting *such that, at any given time t, it gives an unbiased estimate of the current count, and its variance is bounded by $\mathcal{O}\left(\left(\frac{n}{100}\right)^2\right)$. Moreover, the estimate is a $(1+\frac{1}{100})$-approximation to the current count, with probability at least $1-\frac{1}{\mathrm{poly}(n)}$. The total communication of the algorithm is $\mathcal{O}\left(k\cdot\log^2 n\right)$ bits.*

We next state the following result of perfect L_2 sampler, which is a direct corollary of the perfect L_p sampler, in Theorem 2.27.

Theorem 4.9 (Perfect L_2 sampler, c.f. Theorem 2.27)**.** *Let vector $f\in\mathbb{R}^n$ be defined by a distributed data stream S, and let k denote the number of servers. Then, there is an algorithm that gives a perfect L_2 sampler of f, using $k\cdot\mathrm{polylog}(n)$ bits of communication in expectation.*

Next, we present a subrountine that produces unbiased F_2 estimates, which is essential to our difference estimator. We defer its analysis to Section 4.4.

Theorem 4.10 (Unbiased F_2 estimation, c.f. Theorem 4.23 and Theorem 4.20)**.** *Let vector $f\in\mathbb{R}^n$ be defined by a distributed data stream S, and let k denote the number of servers. Then, there is an algorithm that gives an estimate of $\widehat{F}$ to $\|f\|_2^2$, which satisfies $\mathbb{E}\left[\widehat{F}\right]=(1+\mathcal{O}(\varepsilon))\cdot\|f\|_2^2$ and $\mathbb{E}\left[\left(\widehat{F}\right)^2\right]=\mathcal{O}\left(\|f\|_2^4\right)$. The algorithm uses $(k+\frac{\sqrt{k}}{\varepsilon})\cdot\mathrm{polylog}(n)$ bits of communication. Moreover, if f is a static vector, the algorithm gives an estimate of $\widehat{F}$ to $\|f\|_2^2$, which satisfies $\mathbb{E}\left[\widehat{F}\right]=\|f\|_2^2$ and $\mathbb{E}\left[\left(\widehat{F}\right)^2\right]=\mathcal{O}\left(\|f\|_2^4\right)$. This algorithm uses $k\,\mathrm{polylog}(n)$ bits of communication.*

The following statement shows the F_2-estimation algorithm with constant-factor approximation.

Theorem 4.11 (Constant-multiplicative F_2-estimation, c.f. Theorem 4.34 and Theorem 4.35)**.** *Let vector $f\in\mathbb{R}^n$ be defined by a distributed data stream S, and let k denote the number of servers. There is an algorithm that gives a 1.01-approximation to the F_2-moment of f at all times in the stream using $k\cdot\mathrm{polylog}(n)$ bits of communication. In addition, if f is static, there is an algorithm that gives a $(1+\varepsilon)$-approximation to the F_2-moment of f at all times in the stream using $\frac{k^{p-1}}{\varepsilon^2}\cdot\mathrm{polylog}(n)$ bits of communication.*

We describe our difference estimator for robust F_2 moment estimation in full in Algorithm 7.

The following lemma shows that our estimator W_s in Algorithm 7 for the inner product $\langle v,u\rangle$ is unbiased, up to a small additive term.

Lemma 4.12. $\mathbb{E}[W_s]=\langle v,u\rangle+\frac{1}{\mathrm{poly}(n)}$

Proof. By the correctness of the L_2 sampler, we have that

$$\mathbf{Pr}[w_s=i]=\frac{u_i^2}{\|u_i\|_2^2}+\frac{1}{\mathrm{poly}(n)}.$$

Therefore, since the Counting algorithm gives an unbiased estimation to v_{w_s}, we have

$$\mathbb{E}[W_s]=\sum_{i\in[n]}u_iv_i\cdot\frac{\mathbb{E}\left[\widehat{U}^2\right]}{u_i^2}\cdot\left(\frac{u_i^2}{\|u\|_2^2}+\frac{1}{\mathrm{poly}(n)}\right).$$

Algorithm 7 Difference estimator for F_2-moment

Input: Accuracy $\varepsilon \in (0,1)$, parameter $\gamma > 0$, fixed vector u, subsequent vector v that arrives in a distributed data stream

Output: Difference estimator for $\|u+v\|_2^2 - \|u\|_2^2$

1: $S \leftarrow \mathcal{O}\left(\frac{\gamma}{\varepsilon^2}\right)$
2: Let $\widehat{U}$ be an unbiased estimate of $\|u\|_2^2$ with variance $\mathcal{O}\left(\|u\|_2^4\right)$ ▷See Theorem 4.10
3: Let $\widehat{V}$ be an $\mathcal{O}\left(\frac{\varepsilon}{\gamma}\right)$-approximation of $\|v\|_2^2$ ▷See Theorem 4.11
4: **for** $s \in [S]$ **do**
5: $\quad L_2$ sample a coordinate w_s from u ▷See Theorem 4.9
6: $\quad \widehat{v_{w_s}} \leftarrow \textsc{Counting}(v_{w_s})$ ▷See Theorem 4.8
7: $\quad W_s \leftarrow \langle \widehat{v_{w_s}}, u_{w_s} \rangle \cdot \frac{(\widehat{U})^2}{u_{w_s}^2}$
8: **return** $2 \cdot \frac{1}{S} \sum_{s \in [S]} W_s + \widehat{V}$

Since $\mathbb{E}\left[\widehat{U}^2\right] = \|u\|_2^2$, then

$$\mathbb{E}\left[W_s\right] = \sum_{i \in [n]} u_i v_i + \frac{1}{\text{poly}(n)} = \langle v, u \rangle + \frac{1}{\text{poly}(n)},$$

as desired. □

We next upper bound the variance of our estimator W_s.

Lemma 4.13. $\mathbb{E}\left[W_s^2\right] \leq \mathcal{O}\left(\gamma \cdot \|u\|_2^4\right) + \frac{1}{\text{poly}(n)}$

Proof. By the correctness of the L_2 sampler, we have that

$$\mathbf{Pr}\left[w_s = i\right] = \frac{u_i^2}{\|u_i\|_2^2} + \frac{1}{\text{poly}(n)}.$$

Therefore, since the estimation $\widehat{v_{w_s}}$ by Counting algorithm has second moment $\mathcal{O}\left(v_{w_s}^2\right)$, c.f. Theorem 4.8, we have

$$\mathbb{E}\left[W_s^2\right] = \mathcal{O}\left(1\right) \cdot \sum_{i \in [n]} (u_i v_i)^2 \cdot \frac{(\widehat{U})^4}{u_i^4} \cdot \left(\frac{u_i^2}{\|u\|_2^2} + \frac{1}{\text{poly}(n)}\right).$$

Since $\mathbb{E}\left[\widehat{U}^4\right] \leq \mathcal{O}\left(\|u\|_2^4\right)$, then

$$\mathbb{E}\left[W_s^2\right] \leq \mathcal{O}\left(\|u\|_2^2 \cdot \sum_{i \in [n]} v_i^2\right) + \frac{1}{\text{poly}(n)} \leq \mathcal{O}\left(\gamma \cdot \|u\|_2^4\right) + \frac{1}{\text{poly}(n)},$$

since $\|v\|_2^2 \leq \gamma \cdot \|u\|_2^2$. □

We now prove the correctness of our difference estimator for F_2 estimation.

Lemma 4.14. *With probability* 0.99*, Algorithm 7 outputs an additive* $\varepsilon \cdot \|u\|_2^2$ *approximation to* $\|u+v\|_2^2 - \|u\|_2^2$*. In addition, taking the median of* $\mathcal{O}(\log n)$ *repetitions would give a correct result with high probability.*

Proof. We have

$$\|u+v\|_2^2 - \|u\|_2^2 = 2\langle u, v\rangle + \|v\|_2^2.$$

By Lemma 4.12 and Lemma 4.13, we have $\mathbb{E}[W_s] = \langle v, u\rangle + \frac{1}{\text{poly}(nm)}$ and $\mathbb{E}[W_s^2] \le \mathcal{O}(\gamma \cdot \|u\|_2^4) + \frac{1}{\text{poly}(nm)}$. Let $W = 2 \cdot \frac{1}{S}\sum_{s\in[S]} W_s$. Then we have

$$\mathbb{E}[W] = \frac{2}{S}\sum_{s\in[S]} \mathbb{E}[W_s] = 2\langle v, u\rangle + \frac{1}{\text{poly}(nm)}$$

and

$$\mathbb{E}\left[W^2\right] \le \frac{4}{S^2}\sum_{s\in[S]} \mathbb{E}\left[W_s^2\right] \le \frac{2}{S}\cdot\mathcal{O}\left(\gamma\cdot\|u\|_2^4\right) + \frac{1}{\text{poly}(n)} \le \mathcal{O}\left(\varepsilon\cdot\|u\|_2^4\right) + \frac{1}{\text{poly}(n)},$$

for $S = \mathcal{O}\left(\frac{\gamma}{\varepsilon^2}\right)$. By Chevyshev's inequality, it follows that with probability at least 0.99,

$$|W - 2\langle u, v\rangle| \le \frac{\varepsilon}{2}\cdot\|u\|_2^2 + \frac{1}{\text{poly}(n)}.$$

The desired claim then follows from the fact that $\widehat{V}$ is an $\mathcal{O}\left(\frac{\varepsilon}{\gamma}\right)$-approximation of $\|v\|_p^p$, which itself is at most $\gamma\cdot\|u\|_p^p$, i.e., the additive error is $\mathcal{O}\left(\frac{\varepsilon}{\gamma}\|v\|_p^p\right) = \mathcal{O}\left(\varepsilon\|u\|_p^p\right)$. □

We next upper bound the communication cost of our difference estimator for F_2 estimation.

Theorem 4.15. *Let* n *be the number of items in the universe,* k *denote the total number of servers, and* $\varepsilon \in (0,1)$ *denote some accuracy parameter. Let* u *be a fixed vector and* v *be a subsequent vector in the stream. There is a distributed* $(\varepsilon,\varepsilon)$*-continuous* F_2 *difference estimator which uses* $\frac{1}{\varepsilon}\cdot k\cdot\text{polylog}(n)$ *bits of communication in expectation. Also, there is a distributed* (γ,ε)*-static* F_2 *difference estimator which uses* $\frac{\gamma}{\varepsilon^2}\cdot k\cdot\text{polylog}(n)$ *bits of communication in expectation.*

Proof. Given the correctness result in Lemma 4.14, it suffices to show the communication cost of Algorithm 7. For each instance $s\in S$, where $S = \mathcal{O}\left(\frac{\gamma}{\varepsilon^2}\right)$, we use $k\,\text{polylog}(n)$ bits in expectation to run the F_2-sampler (see Theorem 2.27), and we use $\mathcal{O}\left(k\log^2 n\right)$ to run the COUNTING algorithm (see Theorem 4.8). This sums up to $\frac{\gamma}{\varepsilon^2}\cdot k\cdot\text{polylog}(n)$ bits of communication. In addition, we use $k\cdot\text{polylog}(n)$ bits to estimate $\|u\|_2^2$ (see Theorem 4.10). Now, we consider the estimate of $\|v\|_2^2$. For the continuous version, we have $\gamma = \varepsilon$, so we only need a constant approximation to $\|v\|_2^2$. Thus, we use $\frac{1}{\varepsilon}\cdot k\cdot\text{polylog}(n)$ bits in total. For the static version, we use $\frac{\gamma^2}{\varepsilon^2}\cdot k\cdot\text{polylog}(n)$ to estimate $\|v\|_2^2$, which is dominated by other terms. Thus, we use $\frac{\gamma}{\varepsilon^2}\cdot k\cdot\text{polylog}(n)$ bits in total. □

Given the oblivious strong F_2 tracker and our difference estimator, we can give the following guarantees for adversarially robust F_2-estimation.

Theorem 4.16. *Let S be a distributed stream, n be the number of items in the universe, k denote the total number of servers, and $\varepsilon \in (0,1)$ denote some accuracy parameter. There is an adversarially robust distributed streaming algorithm that gives a $(1+\varepsilon)$-approximation to the F_2-moment of the vector induced by stream S, while using $\frac{k}{\varepsilon^2}\operatorname{polylog}\left(n, \frac{1}{\varepsilon}\right)$ bits of communication with probability* 0.99.

Proof. By Theorem 4.11, we have an algorithm that flags whenever the F_2-moment doubles. When the algorithm flags, we can retrieve a $(1+\varepsilon)$-approximation by running the static algorithm in [EKM+24]. This gives an ε-double detector that communicates $\frac{k}{\varepsilon^2} \cdot \operatorname{polylog}\left(n, \frac{1}{\varepsilon}\right)$ bits. By Theorem 4.15, there is a $(\varepsilon, \varepsilon)$-continuous difference estimator that uses $\frac{1}{\varepsilon} \cdot k \operatorname{polylog}(n)$ bits of communication. Also, there is a (γ, ε)-static difference estimator that uses $\frac{\gamma}{\varepsilon^2} \cdot k \cdot \operatorname{polylog}(n)$ bits of communication. Therefore, by Famework 3.10, there is an adversarially robust algorithm that communicates in $\frac{k}{\varepsilon^2} \cdot \operatorname{polylog}\left(n, \frac{1}{\varepsilon}\right)$ bits. □

4.4 Adversarially Robust F_p Estimation, $p \geq 2$

In this section, we present an adversarially robust distributed monitoring protocol for F_p moment estimation, with $p \geq 2$. We first describe how we obtain the (γ, ε) difference estimator for $F_p(u+v) - F_p(u)$. We obtain a mixed L_p sample $i \in [n]$ with probability $\frac{u_i^p}{2\|u\|_p^p} + \frac{v_i^p}{2\|v\|_p^p}$. Then, we construct our estimator for the difference as

$$s = \frac{1}{p_i}[(u_i + v_i)^p - u_i^p].$$

Suppose that we have an unbiased estimate for s with variance $\mathcal{O}(s^2)$. Then averaging $\frac{\gamma}{\varepsilon^2}$ instances of the estimator gives us an estimate for the difference with additive error $\varepsilon \cdot F_p(u)$. In the following sections, we formalize the framework for the F_p difference estimator, and we provide algorithms to estimate $\frac{1}{p_i}$ and $(u_i + v_i)^p - u_i^p$ with near-optimal communication.

We again emphasize that we only need a continuous algorithm for difference estimators at granularity 0, i.e., corresponding to $F_p(u+v) - F_p(u) \leq \varepsilon \cdot F_p(u)$. So, we introduce the $(\varepsilon, \varepsilon)$-continuous F_p difference estimator and the (γ, ε)-static F_p difference estimator.

4.4.1 Outline for the F_p Difference Estimator

We begin by outlining our difference estimator. We first consider the mixed L_p sampling subroutine. Intuitively, it detects the important coordinates for both u and v, and thereby has a small variance. Now, we state the correctness of several important subroutines, whose analysis is deferred to the following sections. We remark that we show two versions of the algorithm for estimating $\frac{1}{p_i}$. We present a result for evolving v with constant-fractional bias, which is used to construct the $(\varepsilon, \varepsilon)$-continuous difference estimator, where a constant-multiplicative approximation suffices. We also present an unbiased estimator for static v, which is used to construct the (γ, ε)-static difference estimator.

Theorem 4.17. *Let there be k servers, each server $j \in [k]$ with an n-dimensional vector $f(j) = [f_1(j), \cdots, f_n(j)]$ that arrives in an insertion-only stream, and let $f = \sum_{j \in [k]} f(j)$. Then for $p \geq 2$, we have the following algorithms*

- *There is an algorithm* LpSampler *that outputs $i \in [n]$ so that for any $j \in [n]$,*

$$\mathbf{Pr}\left[i = j\right] = \frac{f_j^p}{\|f\|_p^p} \pm n^{-c},$$

at all times. This protocol uses $k^{p-1} \cdot \mathrm{polylog}(n)$ bits of communication in expectation.

- *Fix a time t. Let u be the vector at time t and let v be the subsequent vector. Let $p_i = \frac{u_i^p}{2\|u\|_p^p} + \frac{v_i^p}{2\|v\|_p^p}$ denote the sampling probability. Fix an $i \in [n]$. Then there is an algorithm* InverseProbEst *that gives an estimation $\widehat{\frac{1}{p_i}}$ to $\frac{1}{p_i}$ such that $\mathbb{E}\left[\widehat{\frac{1}{p_i}}\right] = \frac{1}{p_i}\left(1 \pm \mathcal{O}\left(\frac{1.01}{C_{4.27}}\right)\right)$ and $Var\left[\frac{1}{p_i}\right] = \frac{1}{p_i^2} \cdot \mathrm{polylog}(n)$ at all times, with probability $1 - \frac{1}{\mathrm{poly}(n)}$. This protocol uses $k^{p-1} \cdot \mathrm{polylog}(n)$ bits of communication. For static v, the algorithm uses the same amount of communication, and the estimate satisfies $\mathbb{E}\left[\widehat{\frac{1}{p_i}}\right] = \frac{1}{p_i} \pm n^{-c}$ and $Var\left[\frac{1}{p_i}\right] = \frac{1}{p_i^2} \cdot \mathrm{polylog}(n)$.*

- *Fix a time t. Let u be the vector at time t and let v be the subsequent vector. Fix an index $i \in [n]$. There is an algorithm* DifferenceEst *that gives an estimation $\widehat{(u_i + v_i)^p} - u_i^p$ to $(u_i+v_i)^p - u_i^p$ such that $\mathbb{E}\left[\widehat{(u_i + v_i)^p} - u_i^p\right] = (u_i+v_i)^p - u_i^p \pm n^{-c}$ and $Var\left[\widehat{(u_i + v_i)^p} - u_i^p\right] = \mathcal{O}\left((u_i + v_i)^{2p} \cdot \log^2 n\right)$ at all times with probability $1 - \frac{1}{\mathrm{poly}(n)}$. Moreover, if $u_i > v_i$, we have $Var\left[\widehat{(u_i + v_i)^p} - u_i^p\right] = \mathcal{O}\left(((u_i + v_i)^p - u_i^p)^2 \cdot \log n\right)$. This protocol uses $k \cdot \mathrm{polylog}(n)$ bits of communication.*

Proof. See Theorem 2.27 for the LpSampler. See Theorem 4.23 for the LpEstUnbiased. See Lemma 4.27, Lemma 4.28, and Corollary 4.29 for InverseProbEst. See Lemma 4.32 and Lemma 4.33 for DifferenceEst. □

In light of the subroutines in Theorem 4.17, we present our difference estimator for F_p moment estimation for $p \geq 2$ in Algorithm 8. We now upper bound the error of our difference estimator.

Lemma 4.18. *For $\|v\|_p^p \leq \gamma \|u\|_p^p$ and $\|u+v\|_p^p - \|u\|_p^p \leq \gamma \|u\|_p^p$, Algorithm 8 gives an estimation to $\|u+v\|_p^p - \|u\|_p^p$ with additive error $\varepsilon \cdot \|u\|_p^p$, with probability $1 - \frac{1}{\mathrm{poly}(n)}$.*

Proof. Let p_i denote the sampling probability of index i. Let $S := \|u+v\|_p^p - \|u\|_p^p = \sum_{i\in[n]}(u_i + v_i)^p - u_i^p$. Notice that our estimator is defined as

$$s_r = \widehat{\frac{1}{p_i}} \cdot \left(\widehat{(u_i + v_i)^p} - u_i^p\right).$$

Therefore, for the continuous version, we have

$$\begin{aligned}
\mathbb{E}\left[s_r\right] &= \sum_{i\in[n]} (p_i \pm n^{-c}) \cdot \left(\frac{1}{p_i}\right) \cdot \left(1 \pm \mathcal{O}\left(\frac{1.01}{C_{4.27}}\right)\right) \cdot ((u_i + v_i)^p - u_i^p) \\
&= \sum_{i\in[n]} p_i \cdot \frac{1}{p_i} \cdot ((u_i + v_i)^p - u_i^p) \cdot \left(1 \pm \mathcal{O}\left(\frac{1.01}{C_{4.27}}\right)\right) = S \cdot \left(1 \pm \mathcal{O}\left(\frac{1.01}{C_{4.27}}\right)\right)
\end{aligned}$$

Algorithm 8 Difference estimator for F_p moment estimation

Input: Accuracy $\varepsilon \in (0,1)$, parameter $\gamma > 0$, fixed vector u, subsequent vector v that arrives in a distributed data stream

Output: Difference estimator for $\|u+v\|_p^p - \|u\|_p^p$

$R \leftarrow \frac{\gamma}{\varepsilon^2} \cdot \text{polylog}(n)$
for $r \in [R]$ **do**
  $s^r \leftarrow 0$, $U \leftarrow \mathcal{U}(0,1)$
  if $U < \frac{1}{2}$ **then**
    $i \leftarrow \textsc{LpSampler}(u)$ ▷Run the static version.
    $\widehat{\frac{1}{p_i}} \leftarrow \textsc{InverseProbEst}(u, v, i, u_i, \varepsilon)$ ▷Query the actual value of u_i
    $\widehat{(u_i + v_i)^p} - u_i^p \leftarrow \textsc{DifferenceEst}(u, v, i, u_i)$
    $s^r \leftarrow s^r + \widehat{\frac{1}{p_i}} \cdot \left(\widehat{(u_i + v_i)^p} - u_i^p\right)$
  if $U > \frac{1}{2}$ **then**
    $j \leftarrow \textsc{LpSampler}(v)$
    $\widehat{\frac{1}{p_j}} \leftarrow \textsc{InverseProbEst}(u, v, j, u_j, \varepsilon)$
    $\widehat{(u_j + v_j)^p} - u_j^p \leftarrow \textsc{DifferenceEst}(u, v, j, u_j)$
    $s^r \leftarrow s^r + \widehat{\frac{1}{p_j}} \cdot \left(\widehat{(u_j + v_j)^p} - u_j^p\right)$
return $\frac{1}{R}\sum_{r\in[R]} s^r$

Note that we only run continuous difference estimator at granularity 0, where we have $\gamma = \varepsilon$. Taking a sufficiently large constant $C_{4.27}$, by our assumption on $S \le \gamma\|u\|_p^p = \varepsilon\|u\|_p^p$, we have

$$\mathbb{E}[s_r] = S \cdot \left(1 \pm \frac{1}{2}\right) = S \pm \frac{\varepsilon}{2} \cdot \|u\|_p^p.$$

For the static version, the same equation holds since the bias is bounded by $\mathcal{O}(n^{-c})$.

Now, let $S_1 := \{i, u_i \ge v_i\}$, $S_2 := \{i, u_i < v_i\}$, we have

$$\begin{aligned}
\mathbb{E}\left[s_r^2\right] &= \sum_{i\in S_1} (p_i \pm n^{-c}) \cdot \mathbb{E}\left[\widehat{\frac{1}{p_i}}^2\right] \cdot \mathbb{E}\left[\left(\widehat{(u_i + v_i)^p} - u_i^p\right)^2\right] \\
&\quad + \sum_{i\in S_2} (p_i \pm n^{-c}) \cdot \mathbb{E}\left[\widehat{\frac{1}{p_i}}^2\right] \cdot \mathbb{E}\left[\left(\widehat{(u_i + v_i)^p} - u_i^p\right)^2\right] \\
&:= S_1 + S_2.
\end{aligned}$$

For the first term, we have

$$\begin{aligned}
S_1 &= \sum_{i\in S_1} p_i \cdot \frac{1}{p_i^2} \cdot ((u_i + v_i)^p - u_i^p)^2 \cdot \text{polylog}(n) \\
&= \sum_{i\in S_1} \frac{1}{p_i} \cdot ((u_i + v_i)^p - u_i^p)^2 \cdot \text{polylog}(n).
\end{aligned}$$

Then,

$$S_1 = \sum_{i \in S_1} \frac{\|u\|_p^p \|v\|_p^p}{\|u\|_p^p v_i^p + \|v\|_p^p u_i^p} \cdot ((u_i + v_i)^p - u_i^p)^2 \cdot \text{polylog}(n)$$
$$\leq \sum_{i \in S_1} \frac{\|u\|_p^p}{u_i^p} \cdot ((u_i + v_i)^p - u_i^p)^2 \cdot \text{polylog}(n).$$

Moreover, since $u_i \geq v_i$ for $i \in S_1$, we have $(u_i + v_i)^p - u_i^p = \mathcal{O}(u_i^p)$. Then, we have

$$S_1 \leq \|u\|_p^p \cdot \sum_{i \in S_1} ((u_i + v_i)^p - u_i^p) \cdot \text{polylog}(n)$$
$$\leq \gamma \|u\|_p^{2p} \cdot \text{polylog}(n),$$

where the second step is by our assumption of $\|u + v\|_p^p - \|u\|_p^p \leq \gamma \|u\|_p^p$. For the second term, similarly we have

$$S_2 = \sum_{i \in S_2} \frac{\|u\|_p^p \|v\|_p^p}{\|u\|_p^p v_i^p + \|v\|_p^p u_i^p} \cdot (u_i + v_i)^{2p} \cdot \text{polylog}(n)$$
$$\leq \sum_{i \in S_2} \frac{\|v\|_p^p}{v_i^p} \cdot (u_i + v_i)^{2p} \cdot \text{polylog}(n).$$

Now, since $u_i < v_i$ for $i \in S_2$, we have $(u_i + v_i)^p = \mathcal{O}(v_i^p)$. Thus, we have

$$S_2 = \sum_{i \in S_2} \frac{\|v\|_p^p}{v_i^p} \cdot v_i^{2p} \cdot \text{polylog}(n) \leq \|v\|_p^{2p} \cdot \text{polylog}(n) \leq \gamma \cdot \|u\|_p^{2p},$$

where the last step is by our assumption of $\|v\|_p^p \leq \gamma \|u\|_p^p$. Therefore, $\mathbb{E}[s_r^2] = \gamma \|u\|_p^{2p} \cdot \text{polylog}(n)$. Recall that the expectation of our estimator satisfies $\mathbb{E}[s_r] = S \pm \frac{\varepsilon}{2} \cdot \|u\|_p^p$. By Chebyshev's inequality, taking the mean of our $\frac{\gamma}{\varepsilon^2} \cdot \text{polylog}(n)$ instances of estimators s_r gives us an estimation to S with additive error $\varepsilon \cdot \|u\|_p^p$. Last, union bounding across all instances of estimators will give us the desired failure probability. □

Now, we give the full guarantees of our difference estimator.

Theorem 4.19. *Let k denote the total number of servers, $p \geq 2$, and $\varepsilon \in (0, 1)$ denote some accuracy parameter. Given a vector f with universe size n, there exists a distributed $(\varepsilon, \varepsilon)$-continuous F_p difference estimator which uses $\frac{1}{\varepsilon} \cdot k^{p-1} \text{polylog}(n)$ bits of communication in expectation. Also, there is a distributed (γ, ε)-static F_p difference estimator which uses $\frac{\gamma}{\varepsilon^2} \cdot k^{p-1} \cdot \text{polylog}(n)$ bits of communication in expectation.*

Proof. Given the correctness result shown in Lemma 4.18, it remains to bound the communication of Algorithm 8. For the continuous difference estimator, since we run $\frac{\gamma}{\varepsilon^2} \cdot \text{polylog}(n) = \frac{1}{\varepsilon} \cdot \text{polylog}(n)$ instances of estimators and each of them has communication cost $k^{p-1} \cdot \text{polylog}(n)$ in expectation, the total communication cost is $\frac{1}{\varepsilon} \cdot k^{p-1} \text{polylog}(n)$. For the static version, we run $\frac{\gamma}{\varepsilon^2} \cdot \text{polylog}(n)$ instances of estimators and each of them has communication cost $k^{p-1} \cdot \text{polylog}(n)$, the total communication cost is $\frac{\gamma}{\varepsilon^2} \cdot k^{p-1} \text{polylog}(n)$. □

4.4.2 Unbiased F_p Estimator

In this section, we provide the unbiased F_p estimation scheme, which is an important subroutine in our difference estimator. We use the unbiased estimator given by the geometric mean of three independent instances $X = \max_{i\in[n]} \frac{f_i}{\mathbf{e}_i^{1/p}}$:

$$Y = \sqrt[3]{X_1^p \cdot X_2^p \cdot X_3^p},$$

which provides an unbiased estimate to $\|f\|_p^p$.

Static algorithm. We first consider the static version of F_p-estimation. Notice that our L_p sampler reports the maximum index i^* of the scaled vector, then we can query all servers to attain the real value of the $\frac{f_{i^*}}{\mathbf{e}_{i^*}^{1/p}}$. We present the algorithm as follows.

Algorithm 9 Static unbiased F_p estimation

Input: Vector $f = \sum_{j\in[k]} f(j) \in \mathbb{R}^n$
Output: Unbiased estimation of $\|f\|_p^p$

1: **for** $r \in [3]$ **do**
2: Scaled f by i.i.d. exponential variables to get vector g_r where $g_{i,r} = \frac{f_i}{e_{i,r}^{1/p}}$
3: $i \leftarrow \operatorname{argmax}_{j\in[n]} g_{j,r}$ ▷Run the static version of LpSampler (See Theorem 2.27)
4: $H_r \leftarrow g_{i,r}^p$
5: **return** $\frac{1}{C_{1.15}} \cdot \sqrt[3]{H_1 \cdot H_2 \cdot H_3}$

Theorem 4.20. *With probability* $1 - \frac{1}{\mathrm{poly}(n)}$, *Algorithm 9 outputs an estimation* $\widehat{\|f\|_p^p}$ *to* $\|f\|_p^p$ *such that* $\mathbb{E}\left[\widehat{\|f\|_p^p}\right] = \|f\|_p^p$ *and* $\mathbb{E}\left[\left(\widehat{\|f\|_p^p}\right)^2\right] = \mathcal{O}\left(\|f\|_p^{2p}\right)$. *It uses* $k^{p-1}\,\mathrm{polylog}(n)$ *bits of communication.*

Proof. By Corollary 1.13, we have

$$\max_{j\in[n]} g_j^p = \frac{\|f\|_p^p}{E},$$

where E is an independent exponential variable with rate 1. Then, we have

$$\sqrt[3]{H_1 \cdot H_2 \cdot H_3} = \frac{\|f\|_p^p}{\sqrt[3]{E_1 \cdot E_2 \cdot E_3}},$$

where E_1, E_2, E_3 are independent. Now, by Lemma 1.15, we have

$$\mathbb{E}\left[\frac{1}{C_{1.15}} \cdot \sqrt[3]{H_1 \cdot H_2 \cdot H_3}\right] = \frac{\|f\|_p^p}{C_{1.15}} \cdot \mathbb{E}\left[\frac{1}{\sqrt[3]{E_1 \cdot E_2 \cdot E_3}}\right] = \|f\|_p^p.$$

Thus, our estimator is unbiased. Moreover, using Lemma 1.15 again yields

$$\mathbb{E}\left[\left(\frac{1}{C_{1.15}} \cdot \sqrt[3]{H_1 \cdot H_2 \cdot H_3}\right)^2\right] = \frac{\|f\|_p^{2p}}{C_{1.15}^2} \cdot \mathbb{E}\left[\left(\frac{1}{\sqrt[3]{E_1 \cdot E_2 \cdot E_3}}\right)^2\right] = \mathcal{O}\left(\|f\|_p^{2p}\right).$$

Therefore, we have bounded the second moment. In addition, its communication bound directly follows from the communication bound of our L_p sampler Theorem 2.27. □

Continuous algorithm. Now, we consider an evolving vector f arriving in a data stream. Applying the L_p sampler in Algorithm 4 identifies the maximum index i of the scaled vector g in the continuous monitoring setting, which constructs the geometric mean estimator. Recall that we conduct a statistical test to ensure that we select the actual maximum, given the estimation error. However, the correctness of our geometric mean estimator relies on the max-stability of exponential variables: $\frac{f_i^p}{\mathbf{e}_i} \sim \frac{\|f\|_p^p}{\mathbf{e}}$. It is unclear whether this property holds when some instances are rejected due to the statistical test. To address this issue, we run the COUNTING algorithm with $\mathcal{O}(1)$ accuracy, ensuring that we incur at most an $\mathcal{O}(1)$-fractional bias. This modification results in an additional communications cost of $\mathcal{O}\left(\sqrt{k}\right)$ bits, which is acceptable because, ultimately, we only use the continuous difference estimator at granularity 0, where constant accuracy suffices, and there is roughly $\frac{1}{\varepsilon}$ such continuous different estimators.

Similar to the static algorithm, we calculate the geometric mean estimator. Since we do not have access to the actual value of g_i in the continuous monitoring setting, we make a slight modification to the static algorithm to ensure an unbiased estimate. Instead of taking the geometric mean of three estimators G_i^p under different scaling, we retrieve three estimators $g_i^{p/3}$ and then multiply them together. Hence, our goal is to give an unbiased estimation of $G_i^{p/3}$ with bounded variance. This is done by applying the truncated Taylor estimator mentioned introduced in Section 2.2:

$$\widehat{g_i^{(q)}} = \sum_{q=0}^{Q} \binom{p/3}{q} s_i^{p/3-q} (g_i - s_i)^q,$$

where s_i is a $(1 + \frac{1}{100p})$-approximation to g_i. Truncating at $\mathcal{O}(\log n)$ terms, the estimation only has a $\frac{1}{\text{poly}(n)}$ additive bias. We describe the algorithms in full in Algorithm 10 and Algorithm 11.

Algorithm 10 Distributed monitoring F_p estimation: Truncated Taylor estimator

Input: Input vector $f = \sum_{j\in[k]} f(j) \in \mathbb{R}^n$ from a distributed data stream, index i

Output: Unbiased estimate of $g_i^{p/3}$

1: The central coordinator sends the random seeds to each server to generate n i.i.d. exponential random variables with rate 1
2: For each update, obtain a set **PL** containing the maximum index ▷See Theorem 2.7
3: **for** $i \in$ **PL do**
4: $\quad \widehat{g_i} \leftarrow \text{COUNTING}(g_i, \frac{1}{100p})$ ▷Give a $(1 + \frac{1}{100p})$-approximation to g_i
5: $i \leftarrow \text{argmax}_{\mathbf{i}}\, \widehat{g_{\mathbf{i}}}$
6: $\widehat{g_i^{(q)}} \leftarrow \text{COUNTING}(g_i, \frac{1}{100p})$ for all $q \in [\mathcal{O}(\log n)]$
7: $\widehat{g_i^{p/3}} \leftarrow \text{TAYLOREST}(\{\widehat{g_i^{(q)}}\}_{q\in[\mathcal{O}(\log n)]})$ ▷See Lemma 2.10
8: **return** $\widehat{g_i^{p/3}}$

We now show the correctness of our algorithm.

Lemma 4.21. *With probability* $1 - \frac{1}{\text{poly}(n)}$, *Algorithm 11 outputs an estimation* $\widehat{\|f\|_p^p}$ *to* $\|f\|_p^p$ *such that* $\mathbb{E}\left[\widehat{\|f\|_p^p}\right] = \|f\|_p^p \cdot (1 \pm \mathcal{O}(\varepsilon))$ *and* $\mathbb{E}\left[\left(\widehat{\|f\|_p^p}\right)^2\right] = \mathcal{O}\left(\|f\|_p^{2p} \cdot \log^6 n\right)$.

Algorithm 11 Distributed monitoring F_p estimation: geometric mean estimator

Input: Input vector $f = \sum_{j\in[k]} f(j) \in \mathbb{R}^n$ from a distributed data stream
Output: Nearly unbiased estimate for $\|f\|_p^p$

1: **for** $r \in [3]$ **do**
2: Run the distributed F_p estimation algorithm and collect the estimate H_r ▷Algorithm 10
3: **return** $\frac{1}{C_{1.15}} \cdot H_1 \cdot H_2 \cdot H_3$

Proof. Consider an instance r and a fixed time t. Let i be the reported max index, by Lemma 2.10, our estimator H_r satisfies $\mathbb{E}[H_r] = g_i^{p/3} \pm n^{-c}$ and $\mathbb{E}[H_r^2] = \mathcal{O}\left(g_i^{p/3} \cdot \log^2 n\right)$, where the expectation is taken w.r.t. the Counting procedure. Now, since we have a $(1+\varepsilon)$-approximation by Counting for each item in **PL**, g_i is within $(1 \pm \mathcal{O}(\varepsilon))$ fraction from the real max, denoted by $g_{D(1)}$. Therefore, we have $\mathbb{E}[H_r] = g_{D(1)}^{p/3} \cdot (1 \pm \mathcal{O}(\varepsilon))$ and $\mathbb{E}[H_r^2] = \mathcal{O}\left(g_{D(1)}^{p/3} \cdot \log^2 n\right)$. By Corollary 1.13, we have

$$g_i^{p/3} = \max_{j\in[n^{c+1}]} g_j^{p/3} = \frac{\|f\|_p^{p/3}}{E^{1/3}},$$

where E is an independent exponential variable with rate 1. Therefore, let $g_{D(1),r}$ denote the real max in round r, we have

$$\mathbb{E}\left[\frac{1}{C_{1.15} \cdot n^c} \cdot H_1 \cdot H_2 \cdot H_3\right] = \frac{1}{C_{1.15} \cdot n^c} \cdot \mathbb{E}\left[g_{D(1),1}^{p/3} \cdot g_{D(1),2}^{p/3} \cdot g_{D(1),3}^{p/3}\right] \cdot (1 \pm \mathcal{O}(\varepsilon))$$

Now, by Lemma 1.15, we have

$$\mathbb{E}\left[\frac{1}{C_{1.15}} \cdot H_1 \cdot H_2 \cdot H_3\right] = \frac{\|f\|_p^p}{C_{1.15}} \cdot \mathbb{E}\left[\frac{1}{\sqrt[3]{E_1 \cdot E_2 \cdot E_3}}\right] \cdot (1 \pm \mathcal{O}(\varepsilon)) = \|f\|_p^p (1 \pm \mathcal{O}(\varepsilon)).$$

Moreover, using Lemma 1.15 again yields

$$\mathbb{E}\left[\left(\frac{1}{C_{1.15}} \cdot H_1 \cdot H_2 \cdot H_3\right)^2\right] = \mathcal{O}\left(\log^6 n\right) \cdot \frac{\|f\|_p^{2p}}{C_{1.15}^2} \cdot \mathbb{E}\left[\left(\frac{1}{\sqrt[3]{E_1 \cdot E_2 \cdot E_3}}\right)^2\right] = \mathcal{O}\left(\|f\|_p^{2p} \cdot \log^6 n\right).$$

Therefore, we have bounded the second moment. □

We now upper bound the total communication cost of our protocol.

Lemma 4.22. *With probability* $1 - \frac{1}{\text{poly}(n)}$, *Algorithm 11 uses* $k^{p-1} \cdot \text{polylog}(n) + \frac{\sqrt{k}}{\varepsilon} \cdot \text{polylog}(n)$ *bits of communication.*

Proof. First, we need $k^{p-1} \cdot \text{polylog}(n)$ bits of to run LpSampler. Second, we run Counting with accuracy ε to track each item in **PL**, which uses $\mathcal{O}\left(\frac{\sqrt{k}}{\varepsilon} \log n\right)$ bits of communication each. Recall that in Lemma 2.18, we showed that only $\text{polylog}(n)$ items are added to **PL** whenever $\|f\|_p^p$ doubles. Therefore, the total communication cost is at most $k^{p-1} \cdot \text{polylog}(n) + \frac{\sqrt{k}}{\varepsilon} \cdot \text{polylog}(n)$ bits. □

Finally, we give the full guarantees of our continuous F_p estimation protocol, given that Lemma 4.21 and Lemma 4.22 hold.

Theorem 4.23. *Given a vector $f \in \mathbb{R}^n$ arriving in the distributed streaming model, let k denote the total number of servers. Let $p \geq 2$. Then, there is a algorithm* LpEstUnbiased *that gives an estimation $\widehat{F}$ to $\|f\|_p^p$, which satisfies $\mathbb{E}\left[\widehat{F}\right] = (1 \pm \mathcal{O}(\varepsilon)) \cdot \|f\|_p^p$ and $\mathbb{E}\left[\left(\widehat{F}\right)^2\right] = \|f\|_p^{2p} \cdot \mathrm{polylog}(n)$. The algorithm uses $k^{p-1} \cdot \mathrm{polylog}(n) + \frac{\sqrt{k}}{\varepsilon} \cdot \mathrm{polylog}(n)$ bits of communication. The algorithm succeeds with probability $1 - \frac{1}{\mathrm{poly}(n)}$.*

4.4.3 Estimation of the Inverse Sampling Probability

This section provides an estimator for the inverse of the mixed sampling probability $\frac{1}{p_i}$, which is

$$\frac{1}{p_i} = \frac{2}{u_i^p/\|u\|_p^p + v_i^p/\|v\|_p^p} = \frac{2\|u\|_p^p\|v\|_p^p}{\|u\|_p^p v_i^p + \|v\|_p^p u_i^p}.$$

So, we need to construct an algorithm to estimate $\frac{1}{\|u\|_p^p v_i^p + \|v\|_p^p u_i^p}$. For convenience, we denote $a = \|u\|_p^p v_i^p$ and $b = \|v\|_p^p u_i^p$. Our approach is to express the function $f(a,b) = \frac{1}{a+b}$ via a bivariate Taylor expansion. The expansion has bounded tail because the partial derivatives of $f(a,b)$ with respect to both a and b have the same form. Then, we utilize the subroutines introduced in previous sections to give a good estimation to each term in the Taylor expansion, such as $\|u\|_p^p$, $\|v\|_p^p$, and v_i^p. We remark that, since each server stores the vector u upon its arrival, we can obtain the actual value of u_i by querying each server for each sampled index i.

Again, for static different estimator, it suffices to use the static subroutines in Section 4.4.2 to give (nearly)-unbiased estimates to $\|u\|_p^p$ and $\|v\|_p^p$. We describe our algorithm in full in Algorithm 12.

We show the tail bound of the bivariate Taylor expasion of $f(a,b)$.

Lemma 4.24. *Let $Q = \mathcal{O}(\log n)$, $a_0 \in [0.99a, 1.01a]$ and $b_0 \in [0.99b, 1.01b]$. Suppose we have $a \geq 1$ and $b \geq 1$. Then,*

$$\left| \frac{1}{a+b} - \sum_{q=0}^{Q} \frac{(-1)^q}{q!} \cdot \frac{1}{(a_0+b_0)^{q+1}} \cdot ((a-a_0)+(b-b_0))^q \right| < n^{-c}.$$

Proof. First, we have the bivariate Taylor series of $\frac{1}{a+b}$ to be

$$\frac{1}{a+b} = \sum_{q=0}^{\infty} \frac{(-1)^q}{q!} \cdot \frac{1}{(a_0+b_0)^{q+1}} \cdot ((a-a_0)+(b-b_0))^q.$$

Now, by our assumption, we have $a \geq 1$ and $b \geq 1$. Since $a_0 \in [0.99a, 1.01a]$ and $b_0 \in [0.99b, 1.01b]$, we have

$$\left| \frac{(-1)^q}{q!} \cdot \frac{1}{(a_0+b_0)^{q+1}} \cdot ((a-a_0)+(b-b_0))^q \right| \leq \frac{1}{a+b} \cdot \left(\frac{1}{40}\right)^q.$$

Hence, we have that for $Q = \mathcal{O}(\log n)$,

$$\sum_{q=Q}^{\infty} \frac{(-1)^q}{q!} \cdot \frac{1}{(a_0+b_0)^{q+1}} \cdot ((a-a_0)+(b-b_0))^q \leq \frac{1}{a+b} \cdot \sum_{q=Q}^{\infty} \left(\frac{1}{40}\right)^q \leq n^{-c}.$$

□

Algorithm 12 Continuous estimation of the inverse of the sampling probability

Input: Underlying vector u defined at time t, sampled index i, u_i

Output: Estimates of $1/p_i$, accuracy ε

$B \leftarrow \text{polylog}(n)$, $Q \leftarrow \mathcal{O}(\log n)$, $D \leftarrow \mathcal{O}(\log n)$ ▷B is the number of replications of each estimator, Q is the number of truncated terms of $1/p_i$, D is the truncated terms of v_i^p

for $q \in [Q]$ **do**

  for $b \in [B]$ **do**

    $x_{q,b} \leftarrow \textsc{LpEstUnbiasedStat}(u)$ ▷Static F_p-estimation

    $y_{q,b} \leftarrow \textsc{LpEstUnbiased}(v, \frac{\varepsilon}{C_{4.27} \log n})$

  $x_q \leftarrow \frac{1}{B} \sum_{b \in [B]} x_{q,b}$ ▷Estimator for $\|u\|_p^p$

  $y_q \leftarrow \frac{1}{B} \sum_{b \in [B]} y_{q,b}$ ▷Estimator for $\|v\|_p^p$

  for $b \in [B]$ **do**

    Let $s_i^{(b)}$ be a $(1 + \frac{1}{100p})$-approximation of v_i

    for $l \in [D]$ **do**

      $\widehat{v_i^{(q,b,l)}} \leftarrow \textsc{Counting}(v_i)$

    for $d \in [D]$ **do**

      $w_{q,b,d} \leftarrow \binom{p}{d} s_i^{(b)^{p-d}} \prod_{l \in [d]} \left(\widehat{v_i^{(q,b,l)}} - s_i^{(b)} \right)$

    $w_{q,b} \leftarrow \sum_{d \in [D]} w_{q,b,d}$

  $w_q \leftarrow \frac{1}{B} \sum_{b \in [B]} w_{q,b}$ ▷Estimator for v_i^p

  $a_q \leftarrow x_q \cdot w_q$ ▷Estimator for $\|u\|_p^p \cdot v_i^p$

  $b_q \leftarrow y_q \cdot u_i^p$ ▷Estimator for $\|v\|_p^p \cdot u_i^p$

Let r be a $(1 + \frac{1}{100p})$-approximation of $\|u\|_p^p$ ▷Obtain by F_p-estimation of u

Let o be a $(1 + \frac{1}{100p})$-approximation for $\|v\|_p^p$ ▷Obtain by F_p-estimation of v

Let h_i be a $(1 + \frac{1}{100p})$-approximation of v_i

$a_0 \leftarrow r \cdot h_i$

$b_0 \leftarrow o \cdot u_i^p$

$z \leftarrow \sum_{q=0}^{Q} \frac{(-1)^q}{q!} \cdot \frac{1}{(a_0+b_0)^{q+1}} \cdot \prod_{l \in [q]} ((a_l - a_0) + (b_l - b_0))$

$x \leftarrow \textsc{LpEstUnbiased}(u)$

$y \leftarrow \textsc{LpEstUnbiased}(v)$

return $2xyz$

Next, we state the guarantee of our truncated bivariate Taylor estimator for $f(a, b)$, given that we have good approximations to both a and b.

Lemma 4.25. *Let $Q = \mathcal{O}(\log n)$, $a_0 \in [0.99a, 1.01a]$, and $b_0 \in [0.99b, 1.01b]$. Suppose we have Q independent estimates for a and b satisfying $\mathbb{E}\left[\widehat{a^{(l)}}\right] = a \cdot (1 \pm \varepsilon)$, $\mathbb{E}\left[\widehat{b^{(l)}}\right] = b \cdot (1 \pm \varepsilon)$ and $Var\left[\widehat{a^{(l)}}\right] \le a^2/5$, $Var\left[\widehat{b^{(l)}}\right] \le b^2/5$. We define the Taylor estimator by*

$$z = \sum_{q=0}^{Q} \frac{(-1)^q}{q!} \cdot \frac{1}{(a_0 + b_0)^{q+1}} \cdot \prod_{l \in [q]} \left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0) \right).$$

Then, we have $\mathbb{E}[z] = \frac{1}{a+b} \cdot (1 \pm \mathcal{O}(\varepsilon \log n))$ *and* $\mathbb{E}[z] = \left(\frac{1}{a+b}\right)^2 \cdot \log^2 n$.

Proof. First, since we have independent estimators such that $\mathbb{E}\left[\widehat{a^{(l)}}\right] = a \cdot (1 \pm \varepsilon)$, $\mathbb{E}\left[\widehat{b^{(l)}}\right] = b \cdot (1 \pm \varepsilon)$, so it holds that

$$\begin{aligned}\mathbb{E}\left[\prod_{l\in[q]}\left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right)\right] &= \prod_{l\in[q]} \mathbb{E}\left[(\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right] \\ &= ((a - a_0) + (b - b_0)^q \cdot (1 \pm \varepsilon)^q \\ &= ((a - a_0) + (b - b_0))^q \cdot (1 \pm \mathcal{O}(q\varepsilon)),\end{aligned}$$

where $q \leq \mathcal{O}(\log n)$. Then, summing up the bias in each term of the truncated Taylor series z, we have $\mathbb{E}[z] = \frac{1}{a+b} \cdot (1 \pm \mathcal{O}(\varepsilon \log n))$ by Lemma 4.24. Moreover, by Lemma 2.9 we have

$$\begin{aligned}\mathbb{E}\left[\left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right)^2\right] &= \mathrm{Var}\left[\widehat{a^{(l)}} + \widehat{b^{(l)}}\right] + \mathbb{E}\left[(\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right]^2 \\ &= \mathrm{Var}\left[\widehat{a^{(l)}} + \widehat{b^{(l)}}\right] + ((a - a_0) + (b - b_0))^2.\end{aligned}$$

Now, since our estimates for a and b are independent, we have

$$\begin{aligned}\mathbb{E}\left[\left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right)^2\right] &= \mathrm{Var}\left[\widehat{a^{(l)}}\right] + \mathrm{Var}\left[\widehat{b^{(l)}}\right] + ((a - a_0) + (b - b_0))^2 \\ &= \frac{a^2}{5} + \frac{b^2}{5} + ((a - a_0) + (b - b_0))^2.\end{aligned}$$

Now, since $a_0 \in [0.99a, 1.01a]$ and $b_0 \in [0.99b, 1.01b]$, we have

$$\mathbb{E}\left[\left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right)^2\right] \leq \left(\frac{a+b}{2}\right)^2.$$

Now, we have

$$\mathbb{E}\left[\left(\prod_{l\in[q]}\left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right)\right)^2\right] = \prod_{l\in[q]} \mathbb{E}\left[\left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right)^2\right] \leq \left(\frac{a+b}{2}\right)^{2q}.$$

Notice that $2(a_0 + b_0) > a + b$. Therefore, we have

$$\begin{aligned}&\mathbb{E}\left[\left(\frac{(-1)^q}{q!} \cdot \frac{1}{(a_0+b_0)^{q+1}} \cdot \prod_{l\in[q]}\left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right)\right)^2\right] \\ \leq &\frac{1}{(q!)^2} \cdot \left(\frac{1}{a_0+b_0}\right)^{2q+2} \cdot \left(\frac{a+b}{2}\right)^{2q} \leq \left(\frac{1}{a+b}\right)^2.\end{aligned}$$

Last, by the AM-GM inequalities, we have

$$\mathbb{E}\left[o^2\right] = \mathbb{E}\left[\left(\sum_{q=0}^{Q} \frac{(-1)^q}{q!} \cdot \frac{1}{(a_0+b_0)^{q+1}} \cdot \prod_{l\in[q]}\left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right)\right)^2\right]$$

$$\leq \mathcal{O}(Q) \cdot \sum_{q=0}^{Q} \mathbb{E}\left[\left(\frac{(-1)^q}{q!} \cdot \frac{1}{(a_0+b_0)^{q+1}} \cdot \prod_{l\in[q]} \left((\widehat{a^{(l)}} - a_0) + (\widehat{b^{(l)}} - b_0)\right)\right)^2\right]$$
$$\leq \mathcal{O}\left(Q^2\right)\left(\frac{1}{a+b}\right)^2.$$

which proves our statement as $Q = \mathcal{O}(\log n)$. □

The following lemma proves that we can reduce the variance by an arithmetic mean estimator. This will be used to upper bound the variance of our truncated bivariate Taylor estimator.

Lemma 4.26. *Let $\widehat{a}$ be an estimation of a with $\mathbb{E}[\widehat{a}] = a \cdot (1 \pm \varepsilon)$ and $Var[\widehat{a}] = C$. Then if we take the mean of M independent estimations, the resulting quantity has $(1+\varepsilon)$-multiplicative error in expectation and its variance will be C/M.*

Proof. First, we have the expectation to be

$$\mathbb{E}\left[\frac{1}{M}\sum_{l=1}^{M}\widehat{a^{(l)}}\right] = \frac{1}{M} \cdot M \cdot a \cdot (1 \pm \varepsilon) = a \cdot (1 \pm \varepsilon).$$

Moreover, since they are independent, we have

$$\mathrm{Var}\left[\frac{1}{M}\sum_{l=1}^{M}\widehat{a^{(l)}}\right] = \frac{1}{M^2}\sum_{l=1}^{M}\mathrm{Var}\left[\widehat{a^{(l)}}\right] = \frac{C}{M}.$$

□

Applying the above statement with $M = \mathrm{polylog}(n)$, we prove the correctness of our algorithm:

Lemma 4.27. *With probability $1 - \frac{1}{\mathrm{poly}(n)}$, Algorithm 12 gives an estimation $\widehat{\frac{1}{p_i}}$ to $\frac{1}{p_i}$ satisfying $\mathbb{E}\left[\widehat{\frac{1}{p_i}}\right] = \frac{1}{p_i} \cdot \left(1 \pm \mathcal{O}\left(\frac{\varepsilon}{C_{4.27}}\right)\right)$ and $\mathbb{E}\left[\widehat{\frac{1}{p_i}}^2\right] = \frac{1}{p_i^2} \cdot \mathrm{polylog}(n)$.*

Proof. First, by Lemma 2.10, Theorem 4.20 and Lemma 4.21, we have the following guarantees of our estimators. For all $q \in [Q]$ and $b \in [B]$, we have

$$\mathbb{E}[x_{q,b}] = \|u\|_p^p \quad \text{and} \quad \mathbb{E}\left[x_{q,b}^2\right] = \|u\|_p^{2p}$$
$$\mathbb{E}[y_{q,b}] = \|v\|_p^p \left(1 \pm \mathcal{O}\left(\frac{\varepsilon}{C_{4.27}\log n}\right)\right) \quad \text{and} \quad \mathbb{E}\left[y_{q,b}^2\right] = \|v\|_p^{2p} \cdot \mathrm{polylog}(n)$$
$$\mathbb{E}[w_{q,b}] = v_i^p \cdot \left(1 \pm \mathcal{O}\left(\frac{\varepsilon}{C_{4.27}\log n}\right)\right) \quad \text{and} \quad \mathbb{E}\left[w_{q,b}^2\right] = v_i^{2p} \cdot \mathrm{polylog}(n).$$

Then, by Lemma 4.26, averaging $B = \mathcal{O}(\mathrm{polylog}(n))$ such estimators give us:

$$\mathbb{E}[x_q] = \|u\|_p^p \quad \text{and} \quad \mathrm{Var}[x_q] = \|u\|_p^{2p}/5$$
$$\mathbb{E}[y_q] = \|v\|_p^p \cdot \left(1 \pm \mathcal{O}\left(\frac{\varepsilon}{C_{4.27}\log n}\right)\right) \quad \text{and} \quad \mathrm{Var}[y_q] = \|v\|_p^{2p}/5$$
$$\mathbb{E}[w_q] = v_i^p \cdot \left(1 \pm \mathcal{O}\left(\frac{\varepsilon}{C_{4.27}\log n}\right)\right) \quad \text{and} \quad \mathrm{Var}[w_q] = v_i^{2p}/5.$$

Therefore, defining $a = \|u\|_p^p v_i^p$ and $b = \|v\|_p^p u_i^p$, for all $q \in [Q]$, we have

$$\mathbb{E}[a_q] = a \cdot \left(1 \pm \mathcal{O}\left(\frac{\varepsilon}{C_{4.27} \log n}\right)\right) \quad \text{and} \quad \mathrm{Var}[x_q] \le a^2/5$$
$$\mathbb{E}[b_q] = b \cdot \left(1 \pm \mathcal{O}\left(\frac{\varepsilon}{C_{4.27} \log n}\right)\right) \quad \text{and} \quad \mathrm{Var}[b_q] \le b^2/5.$$

Moreover, we have $a_0 = a \cdot (1 \pm \frac{1}{100p})$ and $b_0 = b \cdot (1 \pm \frac{1}{100p})$. Note that our estimator for $1/p_i$ is

$$z = \sum_{q=0}^{Q} \frac{(-1)^q}{q!} \cdot \frac{1}{(a_0 + b_0)^{q+1}} \cdot \prod_{l \in [q]} ((a_l - a_0) + (b_l - b_0)).$$

Then, by Lemma 4.25, we have

$$\mathbb{E}[z] = \frac{1}{a+b} \cdot \left(1 \pm \mathcal{O}\left(\frac{\varepsilon}{C_{4.27}}\right)\right) \quad \text{and} \quad \mathbb{E}\left[z^2\right] \le \left(\frac{1}{a+b}\right)^2 \cdot \mathrm{polylog}(n).$$

Note that $\frac{1}{a+b} = \frac{1}{p_i}$, we have our desired result. Moreover, the failure probability follows by union bounding the failure probability of each estimator. □

The following lemma upper bounds the communication cost.

Lemma 4.28. *With probability $1 - \frac{1}{\mathrm{poly}(n)}$, Algorithm 12 uses $\left(k^{p-1} + \frac{\sqrt{k}}{\varepsilon}\right) \cdot \mathrm{polylog}(n)$ bits of communication. In addition, if v is static, Algorithm 12 uses $k^{p-1}\,\mathrm{polylog}(n)$ bits of communication and gets unbiased estimation (up to $\frac{1}{\mathrm{poly}(n)}$ factors).*

Proof. We run $\mathrm{polylog}(n)$ instances of LpEstUnbiased in Algorithm 12, whose communication cost follows from Theorem 4.23. Moreover, it takes $\mathrm{polylog}(n)$ instances of Counting to retrieve the $\mathrm{polylog}(n)$ estimations for v_i we needed in Algorithm 12. Hence, we uses $k^{p-1} \cdot \mathrm{polylog}(n)$ bits of communication in total.

If v is static, we can implement the static version of L_p estimation algorithm to estimate $\|v\|_p^p$. Then, it only uses $k^{p-1}\,\mathrm{polylog}(n)$ bits of communication due to Theorem 4.20. Since the bias only comes from the error in the continuous Counting algorithm, we have an unbiased estimation in the static version. □

Notice that we only use the continuous difference estimator at granularity 0, where a constant-factor approximation suffices. Therefore, we state the following corollary showing a constant-fractional bias result for the continuous monitoring setting, and an unbiased result for the static setting, given Lemma 4.27 and Lemma 4.28.

Corollary 4.29. *Algorithm 12 gives an estimation $\widehat{\frac{1}{p_i}}$ to $\frac{1}{p_i}$ such that $\mathbb{E}\left[\widehat{\frac{1}{p_i}}\right] = \frac{1}{p_i}\left(1 \pm \mathcal{O}\left(\frac{1.01}{C_{4.27}}\right)\right)$ and $Var\left[\frac{1}{p_i}\right] = \frac{1}{p_i^2} \cdot \mathrm{polylog}(n)$ at all times with probability $1 - \frac{1}{\mathrm{poly}(n)}$. This protocol uses $k^{p-1} \cdot \mathrm{polylog}(n)$ bits of communication. For static v, it uses $k^{p-1} \cdot \mathrm{polylog}(n)$ bits of communication, and the estimate satisfies $\mathbb{E}\left[\widehat{\frac{1}{p_i}}\right] = \frac{1}{p_i} \pm n^{-c}$ and $Var\left[\frac{1}{p_i}\right] = \frac{1}{p_i^2} \cdot \mathrm{polylog}(n)$.*

4.4.4 Estimation of $(u_i + v_i)^p - u_i^p$

This section provides the algorithm for estimating $(u_i + v_i)^p - u_i^p$. Since we can obtain the actual value of u_i, we consider the difference as a function of v_i, i.e., we define $f(x) = (u_i + x)^p - u_i^p$. We expand it at $x = \widehat{v_i}$ by Taylor expansion. Similarly, it is sufficient to truncate at the $\mathcal{O}(\log n)$-th term in the Taylor series, as shown in Algorithm 13.

Algorithm 13 Estimator of $(u_i + v_i)^p - u_i^p$

Input: Underlying vector u defined at time t, subsequent vector v, u_i, index i
Output: Unbiased estimator of $(u_i + v_i)^p - u_i^p$

1: $s \leftarrow 0$, $Q \leftarrow \mathcal{O}(\log n)$
2: **for** $q \in [Q]$ **do**
3: $\quad \widehat{v_i^{(q)}} \leftarrow \textsc{Counting}(v_i)$
4: Let s_i be a $(1 + \frac{1}{100p})$-approximation of v_i
5: **for** $q \in [Q]$ **do**
6: $\quad \widehat{(v_i - s_i)^q} \leftarrow \prod_{l \in [q]} \left(\widehat{v_j^{(l)}} - s_i \right)$
7: $\quad s \leftarrow s + \binom{p}{q} (u_i + s_i)^{p-q} \widehat{(v_i - s_i)^q}$
8: **return** $s + [(u_i + s_i)^p - u_i^p]$

The following statement upper bounds the tail error of the truncated Taylor series.

Lemma 4.30. *Let $Q = \mathcal{O}(\log n)$ and $s_i \in [0.99 v_i, 1.01 v_i]$. Then*

$$\left| [(u_i + v_i)^p - u_i^p] - [(u_i + s_i)^p - u_i^p] - \sum_{q=1}^{Q} \binom{p}{q} \left[(u_i + s_i)^{p-q} \right] (v_i - s_i)^q \right| \leq n^{-c}.$$

Proof. Using the Taylor expansion at $\widehat{v_i}$ for the function $f(x) = (u_i + x)^p - u_i^p$ at s_i, we have

$$[(u_i + v_i)^p - u_i^p] = [(u_i + s_i)^p - u_i^p] + \sum_{q=1}^{\infty} \binom{p}{q} \left[(u_i + s_i)^{p-q} \right] (v_i - s_i)^q.$$

For $q > p$, we have $|v_i - s_i| \leq \frac{v_i}{100}$, then

$$\binom{p}{q} (u_i + s_i)^{p-q} (v_i - s_i)^q \leq (2e)^{p/2} (u_i + s_i)^p \left(\frac{1}{100} \right)^q.$$

Therefore, we have that for $Q = \mathcal{O}(\log n)$,

$$\sum_{q=Q+1}^{\infty} \binom{p}{q} \left[(u_i + s_i)^{p-q} \right] (v_i - s_i)^q \leq (u_i + s_i)^p \cdot \sum_{q=Q+1}^{\infty} (2e)^{p/2} \left(\frac{1}{100} \right)^q \leq n^{-c}.$$

Hence,

$$\left| [(u_i + v_i)^p - u_i^p] - [(u_i + s_i)^p - u_i^p] - \sum_{q=1}^{Q} \binom{p}{q} \left[(u_i + s_i)^{p-q} \right] (v_i - s_i)^q \right| \leq n^{-c}.$$

□

Now, we give an auxiliary result that is used to upper bound the variance.

Lemma 4.31. *Let $c > 1$ be some constant. Then we have*

$$(u_i + cv_i)^p - u_i^p \leq c^p \left((u_i + v_i)^p - u_i^p\right).$$

Proof. We define an auxiliary function as follows

$$f(x) = c^p \left((u_i + x)^p - u_i^p\right) - \left((u_i + cx)^p - u_i^p\right).$$

The derivative of the auxiliary function satisfies

$$f'(x) = c^p p(u_i + x)^{p-1} - cp(u_i + cx)^{p-1} = cp(cu_i + cx)^{p-1} - cp(u_i + cx)^{p-1} \geq 0,$$

for $x > 0$, $c > 1$, and $p \geq 2$. Therefore, we have $f(v_i) \geq f(0) = 0$, which gives our result. □

The following lemma proves the correctness of our algorithm.

Lemma 4.32. *With probability $1 - \frac{1}{\mathrm{poly}(n)}$, Algorithm 13 outputs an estimation s of $(u_i + v_i)^p - u_i^p$ satisfying $\mathbb{E}[s] = (u_i + v_i)^p - u_i^p \pm n^{-c}$ and $\mathbb{E}[s^2] = \mathcal{O}\left((u_i + v_i)^p \cdot \log^2 n\right)$. Moreover, if $u_i > v_i$, we have $\mathbb{E}[s^2] = \mathcal{O}\left(((u_i + v_i)^p - u_i^p)^2 \cdot \log n\right)$*

Proof. For $l \in [Q]$, by the guarantee of counting (see Theorem 4.8), we have $\mathbb{E}\left[\widehat{v_i^{(l)}}\right] = v_i$ and $\mathrm{Var}\left[\widehat{v_i^{(l)}}\right] = \left(\frac{v_i}{100p}\right)^2$. Thus, applying Lemma 2.9 with $a = v_i$, $b = s_i$, and $C = \frac{1}{100p}$, we have $\mathbb{E}\left[\widehat{v_i^{(l)}} - s_i\right] = v_i - s_i$ and $\mathbb{E}\left[\left(\widehat{v_i^{(l)} - s_i}\right)^2\right] = \mathcal{O}\left(\left(\frac{v_i}{50}\right)^2\right)$. Then, since we obtain Q independent estimation of v_i, we have

$$\mathbb{E}\left[\prod_{l\in[q]}\left(\widehat{v_i^{(l)}} - s_i\right)\right] = \prod_{l\in[q]}\mathbb{E}\left[\widehat{v_i^{(l)}} - s_i\right] = (v_i - s_i)^q.$$

Moreover, we have

$$\mathbb{E}\left[\left(\prod_{l\in[q]}\left(\widehat{v_i^{(l)}} - s_i\right)\right)^2\right] = \prod_{l\in[q]}\mathbb{E}\left[\left(\widehat{v_i^{(l)}} - s_i\right)^2\right] = \mathcal{O}\left(\left(\frac{v_i}{50p}\right)^{2q}\right).$$

Hence, by Lemma 4.30 and $Q = \mathcal{O}(\log n)$, we have

$$\begin{aligned}
\mathbb{E}[s] &= (u_i + s_i)^p - u_i^p + \sum_{q\in[Q]\setminus\{0\}}\binom{p}{q}(u_i + s_i)^{p-q} \cdot \mathbb{E}\left[\prod_{l\in[q]}\left(\widehat{v_i^{(l)}} - s_i\right)\right] \\
&= (u_i + s_i)^p - u_i^p + \sum_{q\in[Q]\setminus\{0\}}\binom{p}{q}(u_i + s_i)^{p-q}(v_i - s_i)^q = G_i^p \pm n^{-c}.
\end{aligned}$$

Next, applying the AM-GM inequalities, we have

$$\mathbb{E}\left[s^2\right] = \mathbb{E}\left[\left((u_i+s_i)^p - u_i^p + \sum_{q\in[Q]\setminus\{0\}}\left(\binom{p}{q}(u_i+s_i)^{p-q}\prod_{l\in[q]}\left(\widehat{v_i^{(l)}} - s_i\right)\right)\right)^2\right]$$
$$= \mathcal{O}(Q)\cdot\left(((u_i+s_i)^p - u_i^p)^2 + \sum_{q\in[Q]\setminus\{0\}}\binom{p}{q}^2(u_i+s_i)^{2p-2q}\mathbb{E}\left[\prod_{l\in[q]}\left(\widehat{v_i^{(l)}} - s_i\right)^2\right]\right).$$

Now, we have

$$\mathbb{E}\left[s^2\right] = \mathcal{O}(Q)\cdot\left(((u_i+s_i)^p - u_i^p)^2 + \sum_{q\in[Q]\setminus\{0\}}\binom{p}{q}^2(u_i+s_i)^{2p-2q}\left(\frac{v_i}{50p}\right)^{2q}\right).$$

Then, based on the same bound as in the analysis of Lemma 2.10, we have $\mathbb{E}\left[s^2\right] = \mathcal{O}\left((u_i+v_i)^p\cdot\log n\right)$. Now, we suppose $u_i > v_i$. Consider two adjacent terms in the finite sum, since $s_i = v_i\cdot(1\pm\frac{1}{100p})$, we have

$$\frac{\binom{p}{q}^2(u_i+s_i)^{2p-2q}\left(\frac{v_i}{50p}\right)^{2q}}{\binom{p}{q+1}^2(u_i+s_i)^{2p-2q-2}\left(\frac{v_i}{50p}\right)^{2q+2}} = \frac{(q+1)^2(u_i+s_i)^2(50p)^2}{(q-p+1)^2v_i^2} > 2500.$$

Then, the sum is upper bounded by a decreasing geometric sequence. So, we have

$$\sum_{q\in[Q]\setminus\{0\}}\binom{p}{q}^2(u_i+s_i)^{2p-2q}\left(\frac{v_i}{50p}\right)^{2q} \le \binom{p}{1}^2(u_i+s_i)^{2p-2}\left(\frac{v_i}{50p}\right)^2\cdot\frac{1}{1-1/2500}.$$

Then, we have

$$\mathbb{E}\left[s^2\right] = \mathcal{O}(Q)\cdot\left(((u_i+s_i)^p - u_i^p)^2 + \binom{p}{1}^2(u_i+s_i)^{2p-2}\left(\frac{v_i}{50p}\right)^2\right)$$

Now, we look at the Taylor expansion of $f(x) = (u_i+x)^p - u_i^p$ evaluated at $s_i + \frac{v_i}{50p}$:

$$f(s_i+\frac{v_i}{50p}) = (u_i+s_i)^p - u_i^p + \sum_{q=1}^{\infty}\binom{p}{q}(u_i+s_i)^{p-q}\left(\frac{v_i}{50p}\right)^q.$$

Notice that, for $q\ge p$, each pair of adjacent terms consists of one positive term and one negative term. Then, as we mentioned earlier, the absolute value of each term is upper bounded by a decreasing geometric series, so the negative term is always dominated by the positive term. This means that

$$f(s_i+\frac{v_i}{50p}) \ge (u_i+s_i)^p - u_i^p + \binom{p}{1}(u_i+s_i)^{p-1}\frac{v_i}{50p}.$$

Therefore,

$$\mathbb{E}\left[s^2\right] = \mathcal{O}(Q)\cdot f(s_i+\frac{v_i}{50p})^2 = \mathcal{O}(Q)\cdot((u_i+s_i+\frac{v_i}{50p})^p - u_i^p)^2.$$

Moreover, since $s_i = v_i \cdot (1 \pm \frac{1}{100p})$, we have $(u_i + s_i + \frac{v_i}{50p})^p < \left(u_i + \left(1 + \frac{1}{25p}\right) \cdot v_i\right)^p$. Then, by Lemma 4.31, we have

$$\left(u_i + \left(1 + \frac{1}{25p}\right) \cdot v_i\right)^p - u_i^p \leq \left(1 + \frac{1}{25p}\right)^p \cdot ((u_i + v_i)^p - u_i^p) = \mathcal{O}(1) \cdot ((u_i + v_i)^p - u_i^p)$$

Therefore, we have

$$\mathbb{E}\left[s^2\right] = \mathcal{O}(Q) \cdot ((u_i + v_i)^p - u_i^p)^2 = \mathcal{O}\left(((u_i + v_i)^p - u_i^p)^2 \cdot \log n\right)$$

□

The following lemma bounds the communication cost.

Lemma 4.33. *With probability $1 - \frac{1}{\mathrm{poly}(n)}$, Algorithm 13 uses $k \cdot \mathrm{polylog}(n)$ bits of communication.*

Proof. We need $\mathcal{O}(\log n)$ instances of Counting to estimate v_i. By Theorem 4.8, this needs $k \cdot \mathrm{polylog}(n)$ bits of communication. □

4.4.5 Putting it All Together

With the F_p difference estimator introduced in previous sections, we are able to apply the framework to achieve an adversarially robust algorithm for estimating F_p.

First, we state an algorithm that provides a 1.01-approximation to the F_p-moment at all times, which is used in the double detector.

Theorem 4.34 ([WZ12])**.** *Let S be a distributed stream. Let n be the size of the universe. Let k be the number of servers. Let $p \geq 1$. There is an algorithm that gives a 1.01-approximation to the F_p-moment of vector f induced by stream S at all times with probability $1 - \frac{1}{\mathrm{poly}(n)}$. This algorithm implements in $k^{p-1} \cdot \mathrm{polylog}(n)$ bits of communications.*

We then state an algorithm that provides a static $(1+\varepsilon)$-approximation to the F_p-moment.

Theorem 4.35 ([EKM+24])**.** *Let n be the size of the universe. Let k be the number of servers. Each server holds a vector $f(j) \in \mathbb{R}^n$. Let $p \geq 2$. There is an algorithm that gives a $(1+\varepsilon)$-approximation to the F_p-moment with probability $1 - \frac{1}{\mathrm{poly}(n)}$. This algorithm runs in $k^{p-1} \cdot \mathrm{polylog}(n)$ bits of communications.*

Combining the above subroutines gives a ε-double detector.

Corollary 4.36. *There is a ε-double detector for F_p-estimation problem which runs in $k^{p-1} \cdot \mathrm{polylog}(n)$ bits of communications.*

Proof. We run the algorithm in Theorem 4.34 at all times. It flags when F satisfies $0.99 \cdot 2^a \leq F \leq 1.01 \cdot 2^a$. When it flags, we run the algorithm in Theorem 4.35 to give a $(1+\varepsilon)$-approximation with probability $1 - \frac{1}{\mathrm{poly}(n)}$. Since there are only $\mathcal{O}(\log n)$ rounds, we only need to run the algorithm in Theorem 4.35 for $\mathcal{O}(\log n)$ times, which gives the communication cost. □

Given the ε-double detector and our difference estimator, we have an adversarially robust algorithm for F_p-estimation:

Theorem 4.37. *Let S be a distributed stream. Let n be the number of items in the universe. Let k denote the total number of servers. Let $\varepsilon \in (0,1)$ denote some accuracy parameter. There is an adversarially robust distributed streaming algorithm which gives a $(1+\varepsilon)$-approximation to the F_p-moment of the vector induced by stream S. This algorithm uses $\frac{k^{p-1}}{\varepsilon^2} \cdot \text{polylog}(n, \frac{1}{\varepsilon})$ bits of communication with probability 0.99.*

Proof. By Corollary 4.36, we have a ε-double detector that communicates $\frac{k^{p-1}}{\varepsilon^2} \cdot \text{polylog}(n)$ bits. By Theorem 4.19, there is a distributed $\mathcal{B}_C(u, v, \varepsilon, \varepsilon)$-continuous difference estimator which uses $\frac{1}{\varepsilon} \cdot k^{p-1} \text{polylog}(n)$ bits of communication. Also, there is a distributed $\mathcal{B}_S(u, v, \gamma, \varepsilon)$-static difference estimator which uses $\frac{\gamma}{\varepsilon^2} \cdot k^{p-1} \cdot \text{polylog}(n)$ bits of communication. Therefore, by Famework 3.10, there is an adversarially robust algorithm that communicates in $\frac{k^{p-1}}{\varepsilon^2} \cdot \text{polylog}(n, \frac{1}{\varepsilon})$ bits. □

4.5 Adversarially Robust L_2 Heavy-Hitters

In this section, we describe our L_2 heavy-hitter algorithm for the adversarially robust distributed monitoring model. We state the definition for the L_2 heavy-hitter problem as follows.

Definition 4.38 (L_2 heavy-hitter problem)**.** *Given a frequency vector $f \in \mathbb{R}^n$ and an accuracy parameter $\varepsilon \in (0,1)$, the goal is to output a list $\mathcal{L}$ that contains all $i \in [n]$ such that $f_i \geq \varepsilon \cdot \|f\|_2$, but contains no such $j \in [n]$ with $f_j \leq \frac{\varepsilon}{2} \cdot \|f\|_2$. Moreover, for each $i \in \mathcal{L}$, we require an estimate to f_i with additive error $\frac{\varepsilon}{4} \cdot \|f\|_2$.*

We introduce the result from [HXZW25], which solves the L_2 heavy-hitter problem in the non-robust distributed setting.

Theorem 4.39 ([HXZW25])**.** *There is a distributed monitoring algorithm that solves the L_2 heavy-hitter problem using $\frac{k}{\varepsilon^2} \cdot \text{polylog}(n)$ bits of communication.*

Next, we introduce an algorithm in the adversarially robust setting. Recall that in our difference estimator framework, we divide the algorithm into $\mathcal{O}(\log n)$ rounds, where in each round, F_2 doubles. Within each round, we further decompose the stream into difference estimators with different granularities. Consider a round a, suppose that there is some coordinate i that is a ε-heavy-hitter at the end of the round, i.e., $(f_i)^2 \geq \varepsilon^2 F_2(1, t_{a+1})$. Then, it must be either roughly ε^2-heavy at the beginning of the round or roughly $2^r\varepsilon^2$-heavy with respect to a duration $(t_{a,r-1}, t_{a,r})$ within this round, in which F_2 increases by roughly $2^r \varepsilon F_2(1, t_{a+1})$. Therefore, we define our difference estimator at granularity r in the L_2-heavy-hitter problem as the implementation of the distributed monitoring L_2-heavy-hitter algorithm with accuracy $2^r \cdot \varepsilon^2$. By applying the analysis from our difference estimator framework, we establish the correctness of our adversarially robust L_2-heavy-hitter algorithm.

Now, we analyze our adversarially robust algorithm for the distributed monitoring setting.

Theorem 4.40. *Let S be a distributed stream, n be the number of items in the universe, k denote the total number of servers, and $\varepsilon \in (0,1)$ denote some accuracy parameter. There is an adversarially robust distributed streaming algorithm that solves the L_2 heavy-hitter problem on S, which succeeds with high probability and uses $\frac{k}{\varepsilon^2} \cdot \text{polylog}(n, \frac{1}{\varepsilon})$ bits of communication.*

Proof. Recall that in Algorithm 6, we have a counter a to count the number of rounds. We have a parameter $r \in [\beta]$, where $\beta = \mathcal{O}\left(\log \frac{1}{\varepsilon}\right)$, denoting the granularity of the difference estimators.

Suppose there exists $i \in [n]$ such that $(f_i)^2 \geq \varepsilon^2 F_2(1, t_{a+1})$. Then, the coordinate i must either be $\frac{\varepsilon^2}{32^2}$-heavy with respect to $F_2(1, t_a)$ or $\frac{2^r \cdot \varepsilon^2}{32^2 \beta^2}$-heavy with respect to $F_2(t_{a,r-1}, t_{a,r})$ for some integer $r \in [\beta]$, where we use the convention $t_{a,0} = t_a$. Hence, running the algorithm in Theorem 4.39 with accuracy $\frac{2^r \cdot \varepsilon^2}{32^2 \beta^2}$ with respect to $F_2(t_{a,r-1}, t_{a,r})$ detects that f_i is a heavy-hitter, and it uses $\frac{k\beta^2}{2^r \cdot \varepsilon^2}$ bits of communications. Since this is the same accuracy parameter as the difference estimator $\mathcal{B}_{a,r}$, by the same analysis as in Algorithm 6, the communication cost follows.

Additionally, note that if a coordinate i is reported as heavy in an algorithm corresponding to the difference estimator at granularity r, then $\mathcal{O}(\varepsilon^2) \cdot F_2$ of the contribution of coordinate i towards the overall F_2 still appears after i is reported as a heavy-hitter. Thus, keeping track of the frequency of i after it is reported using the COUNTING algorithm, we obtain an estimate of the frequency of i up to an additive $\mathcal{O}(\varepsilon) \cdot F_2$, which solves the L_2-heavy-hitter problem. □

Acknowledgments

Honghao Lin is supported in part by a Simons Investigator Award and Office of Naval Research award number N000142112647. David P. Woodruff is supported in part by a Simons Investigator Award, Office of Naval Research award number N000142112647, and NSF CCF-2335412. Shenghao Xie is supported in part by NSF CCF-2335411. Samson Zhou is supported in part by NSF CCF-2335411. Samson Zhou gratefully acknowledges funding provided by the Oak Ridge Associated Universities (ORAU) Ralph E. Powe Junior Faculty Enhancement Award. The work was conducted in part while David P. Woodruff and Samson Zhou were visiting the Simons Institute for the Theory of Computing as part of the Sublinear Algorithms program.

References

[ABC09] Chrisil Arackaparambil, Joshua Brody, and Amit Chakrabarti. Functional monitoring without monotonicity. In *Automata, Languages and Programming, 36th International Colloquium, ICALP, Proceedings, Part I*, pages 95–106, 2009. 1

[ABD+21] Noga Alon, Omri Ben-Eliezer, Yuval Dagan, Shay Moran, Moni Naor, and Eylon Yogev. Adversarial laws of large numbers and optimal regret in online classification. In *STOC: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 447–455, 2021. 3

[ABJ+22] Miklós Ajtai, Vladimir Braverman, T. S. Jayram, Sandeep Silwal, Alec Sun, David P. Woodruff, and Samson Zhou. The white-box adversarial data stream model. In *PODS '22: International Conference on Management of Data*, pages 15–27, 2022. 3

[ACGS23] Sepehr Assadi, Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Coloring in graph streams via deterministic and adversarially robust algorithms. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, pages 141–153, 2023. 3

[ACSS23] Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. A framework for adversarial streaming via differential privacy and difference estimators. In *14th Innovations in Theoretical Computer Science Conference, ITCS*, pages 8:1–8:19, 2023. 3

[ADK09] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX and 13th International Workshop, RANDOM. Proceedings*, pages 15–28, 2009. 1

[AGMS02] Noga Alon, Phillip B. Gibbons, Yossi Matias, and Mario Szegedy. Tracking join and self-join sizes in limited storage. *J. Comput. Syst. Sci.*, 64(3):719–747, 2002. 3

[AHKO25] Alexandr Andoni, Themistoklis Haris, Esty Kelman, and Krzysztof Onak. From search to decision: A framework for adversarially robust approximate nearest neighbor search, 2025. 3

[AKO11] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 363–372, 2011. 1, 3, 5

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999. 3

[AMYZ19] Dmitrii Avdiukhin, Slobodan Mitrovic, Grigory Yaroslavtsev, and Samson Zhou. Adversarially robust submodular maximization under knapsack constraints. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD*, pages 148–156. ACM, 2019. 3

[BBC+17] Jaroslaw Blasiok, Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, and Lin F. Yang. Streaming symmetric norms via measure concentration. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 716–729, 2017. 10, 19

[BDN17] Jaroslaw Blasiok, Jian Ding, and Jelani Nelson. Continuous monitoring of ℓ_p norms in data streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 32:1–32:13, 2017. 3

[BEJWY20] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems PODS20*, 2020. 3, 9, 41

[BEO22] Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. Adversarially robust streaming via dense-sparse trade-offs. In *5th Symposium on Simplicity in Algorithms, SOSA@SODA*, pages 214–227, 2022. 3

[BHM+21] Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. Adversarial robustness of streaming algorithms through importance sampling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 3544–3557, 2021. 3

[BJKS04] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004. 3

[BJWY22] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *J. ACM*, 69(2):17:1–17:33, 2022. 3

[BKM+22] Amos Beimel, Haim Kaplan, Yishay Mansour, Kobbi Nissim, Thatchaphol Saranurak, and Uri Stemmer. Dynamic algorithms against an adaptive adversary: generic constructions and lower bounds. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1671–1684, 2022. 3

[BO03] Brian Babcock and Chris Olston. Distributed top-k monitoring. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 28–39, 2003. 1

[BO13] Vladimir Braverman and Rafail Ostrovsky. Generalizing the layering method of indyk and woodruff: Recursive sketches for frequency-based vectors on streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX, and 17th International Workshop, RANDOM. Proceedings*, pages 58–70, 2013. 4

[BOZ12] Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.*, 78(1):260–272, 2012. 1, 5

[BY20] Omri Ben-Eliezer and Eylon Yogev. The adversarial robustness of sampling. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, pages 49–62, 2020. 3

[CDK+11] Edith Cohen, Nick G. Duffield, Haim Kaplan, Carsten Lund, and Mikkel Thorup. Efficient stream sampling for variance-optimal estimation of subset sums. *SIAM J. Comput.*, 40(5):1402–1431, 2011. 1

[CDK+14] Edith Cohen, Nick G. Duffield, Haim Kaplan, Carsten Lund, and Mikkel Thorup. Algorithms and estimators for summarization of unaggregated data streams. *J. Comput. Syst. Sci.*, 80(7):1214–1244, 2014. 1

[CG19] Edith Cohen and Ofir Geri. Sampling sketches for concave sublinear functions of frequencies. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 1361–1371, 2019. 1

[CGMR05] Graham Cormode, Minos N. Garofalakis, S. Muthukrishnan, and Rajeev Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 25–36, 2005. 1

[CGS22] Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Adversarially robust coloring for graph streams. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, pages 37:1–37:23, 2022. 3

[CKS03] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th Annual IEEE Conference on Computational Complexity*, pages 107–117, 2003. 3

[ÇM09] Ali Çivril and Malik Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theor. Comput. Sci.*, 410(47-49):4801–4811, 2009. 1

[CMY11] Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. *ACM Trans. Algorithms*, 7(2):21:1–21:20, 2011. 1, 3, 5, 50

[CMYZ12] Graham Cormode, S. Muthukrishnan, Ke Yi, and Qin Zhang. Continuous sampling from distributed streams. *J. ACM*, 59(2):10:1–10:25, 2012. 1

[CPW20] Edith Cohen, Rasmus Pagh, and David P. Woodruff. WOR and p's: Sketches for ℓ_p-sampling without replacement. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020. 1, 5

[CSS25] Edith Cohen, Moshe Shechner, and Uri Stemmer. A simple and robust protocol for distributed counting. *CoRR*, abs/2509.05870, 2025. 3, 7

[CSW+23] Yeshwanth Cherapanamjeri, Sandeep Silwal, David P. Woodruff, Fred Zhang, Qiuyi Zhang, and Samson Zhou. Robust algorithms on adaptive inputs from bounded adversaries. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023. 3

[CXWZ25] Vincent Cohen-Addad, Shenghao Xie, David P. Woodruff, and Samson Zhou. Nearly space-optimal graph and hypergraph sparsification in insertion-only data streams. *CoRR*, abs/2510.18180, 2025. 3

[CZ17] Jiecao Chen and Qin Zhang. Improved algorithms for distributed entropy monitoring. *Algorithmica*, 78(3):1041–1066, 2017. 1

[DF92] Danny Dolev and Tomás Feder. Determinism vs. nondeterminism in multiparty communication complexity. *SIAM J. Comput.*, 21(5):889–895, 1992. 12

[DR02] Mark Dilman and Danny Raz. Efficient reactive monitoring. *IEEE J. Sel. Areas Commun.*, 20(4):668–676, 2002. 1

[DRVW06] Amit Deshpande, Luis Rademacher, Santosh S. Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(12):225–247, 2006. 1

[DSWZ23] Itai Dinur, Uri Stemmer, David P. Woodruff, and Samson Zhou. On differential privacy and adaptive data analysis with bounded space. In *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part III*, pages 35–65, 2023. 3

[DV06] Amit Deshpande and Santosh S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization.*

Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX and 10th International Workshop on Randomization and Computation, RANDOM, Proceedings, pages 292–303, 2006. 1

[DV07] Amit Deshpande and Kasturi R. Varadarajan. Sampling-based dimension reduction for subspace approximation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 641–650, 2007. 1

[EGHK99] Deborah Estrin, Ramesh Govindan, John S. Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *MOBICOM '99, The Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 263–270, 1999. 1

[EKM+24] Hossein Esfandiari, Praneeth Kacham, Vahab Mirrokni, David P. Woodruff, and Peilin Zhong. Optimal communication bounds for classic functions in the coordinator model and beyond. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC*, pages 1911–1922, 2024. 3, 4, 13, 51, 55, 70

[EV03] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.*, 21(3):270–313, 2003. 1

[FKSV02] Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. An approximate l1-difference algorithm for massive data streams. *SIAM J. Comput.*, 32(1):131–151, 2002. 3

[FKV04] Alan M. Frieze, Ravi Kannan, and Santosh S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004. 1

[GKMS01] Anna C Gilbert, Yannis Kotidis, S Muthukrishnan, and Martin Strauss. Quicksand: Quick summary and analysis of network data, 2001. Technical report. 1

[GLH08] Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. Maintaining bounded-size sample synopses of evolving datasets. *VLDB J.*, 17(2):173–202, 2008. 1

[GLW+24] Elena Gribelyuk, Honghao Lin, David P. Woodruff, Huacheng Yu, and Samson Zhou. A strong separation for adversarially robust l_0 estimation for linear sketches. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 2318–2343, 2024. 3

[GLW+25] Elena Gribelyuk, Honghao Lin, David P. Woodruff, Huacheng Yu, and Samson Zhou. Lifting linear sketches: Optimal bounds and adversarial robustness. In *Proceedings of the 57th ACM Symposium on Theory of Computing, STOC*, 2025. (to appear). 3

[GM98] Phillip B. Gibbons and Yossi Matias. New sampling-based summary statistics for improving approximate query answers. In *SIGMOD, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 331–342, 1998. 1

[Haa16] Peter J. Haas. Data-stream sampling: Basic techniques and results. In *Data Stream Management - Processing High-Speed Data Streams*, Data-Centric Systems and Applications, pages 13–44. Springer, 2016. 1

[HKM+20] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020. 3

[HNG+07] Ling Huang, XuanLong Nguyen, Minos N. Garofalakis, Joseph M. Hellerstein, Michael I. Jordan, Anthony D. Joseph, and Nina Taft. Communication-efficient online detection of network-wide anomalies. In *INFOCOM. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, pages 134–142, 2007. 1

[HNO08] Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 489–498, 2008. 3

[HS92] Peter J. Haas and Arun N. Swami. Sequential sampling procedures for query size estimation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 341–350, 1992. 1

[HXZW25] Zengfeng Huang, Zhongzheng Xiong, Xiaoyi Zhu, and Zhewei Wei. Simple and optimal algorithms for heavy hitters and frequency moments in distributed models. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 371–382. ACM, 2025. 4, 5, 19, 71

[HYZ12] Zengfeng Huang, Ke Yi, and Qin Zhang. Randomized algorithms for tracking distributed count, frequencies, and ranks. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems PODS12*, 2012. 1, 4, 5, 6, 7, 28, 32, 48, 49, 52

[IMGR20] Piotr Indyk, Sepideh Mahabadi, Shayan Oveis Gharan, and Alireza Rezaei. Composable core-sets for determinant maximization problems via spectral spanners. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1675–1694, 2020. 1

[IMMM14] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 100–108. ACM, 2014. 1

[Ind00] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS*, pages 189–197, 2000. 3

[IW05] Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 202–208, 2005. 3, 4, 10, 19

[Jay09] T. S. Jayram. Hellinger strikes back: A note on the multi-party information complexity of AND. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX, and 13th International Workshop, RANDOM. Proceedings*, pages 562–573, 2009. 39

[JOW+02] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 96–107, 2002. 1

[JST11] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *In Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems PODS11*, pages 49 – 58, 2011. 1, 5

[JSTW19] Rajesh Jayaram, Gokarna Sharma, Srikanta Tirthapura, and David P. Woodruff. Weighted reservoir sampling from distributed streams. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, pages 218–235, 2019. 2, 5

[JW09] T. S. Jayram and David P. Woodruff. The data stream space complexity of cascaded norms. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 765–774, 2009. 1

[JW18] Rajesh Jayaram and David P. Woodruff. Perfect l_p sampling in a data stream. In *Proceedings of the IEEE 59th Annual Symposium on Foundations of Computer Science, FOCS*, pages 544–555, 2018. 1, 5, 7, 27, 31

[JWZ22] Rajesh Jayaram, David P. Woodruff, and Samson Zhou. Truly perfect samplers for data streams and sliding windows. In *PODS '22: International Conference on Management of Data*, pages 29–40, 2022. 1, 5

[JWZ24] Rajesh Jayaram, David P. Woodruff, and Samson Zhou. Streaming algorithms with few state changes. In *PODS: International Conference on Management of Data*, 2024. 1, 3, 19

[KCR06] Ram Keralapura, Graham Cormode, and Jeyashankher Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 289–300, 2006. 1

[KMNS21] Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. Separating adaptive streaming from oblivious streaming using the bounded storage model. In *Advances in Cryptology - CRYPTO - 41st Annual International Cryptology Conference, CRYPTO Proceedings, Part III*, pages 94–121, 2021. 3

[KNPW11] Daniel M. Kane, Jelani Nelson, Ely Porat, and David P. Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 745–754, 2011. 3

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1161–1178, 2010. 3

[KSZC03] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM Internet Measurement Conference, IMC*, pages 234–247, 2003. 3

[KVW14] Ravi Kannan, Santosh S. Vempala, and David P. Woodruff. Principal component analysis and higher correlations for distributed data. In *Proceedings of The 27th Conference on Learning Theory, COLT*, pages 1040–1057, 2014. 3

[LCD05] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *Proceedings of the ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 217–228, 2005. 3

[Li08] Ping Li. Estimators and tail bounds for dimension reduction in l_α $(0 < \alpha \leq 2)$ using stable random projections. In Shang-Hua Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 10–19, 2008. 3, 8

[LN95] Richard J. Lipton and Jeffrey F. Naughton. Query size estimation by adaptive sampling. *J. Comput. Syst. Sci.*, 51(1):18–25, 1995. 1

[LSW18] Roie Levin, Anish Prasad Sevekari, and David P. Woodruff. Robust subspace approximation in a stream. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2018. 10

[MCS+06] Jianning Mai, Chen-Nee Chuah, Ashwin Sridharan, Tao Ye, and Hui Zang. Is sampled data sufficient for anomaly detection? In *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC*, pages 165–176, 2006. 1

[MFHH05] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on database systems (TODS)*, 30(1):122–173, 2005. 1

[MIGR19] Sepideh Mahabadi, Piotr Indyk, Shayan Oveis Gharan, and Alireza Rezaei. Composable core-sets for determinant maximization: A simple near-optimal algorithm. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97, pages 4254–4263, 2019. 1

[MRWZ20] Sepideh Mahabadi, Ilya P. Razenshteyn, David P. Woodruff, and Samson Zhou. Non-adaptive adaptive sampling on turnstile streams. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1251–1264, 2020. 1

[MW10] Morteza Monemizadeh and David P. Woodruff. 1-pass relative-error L_p-sampling with applications. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1143–1160, 2010. 1, 5

[MWZ22] Sepideh Mahabadi, David P. Woodruff, and Samson Zhou. Adaptive sketches for robust regression with importance sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM)*, pages 31:1–31:21, 2022. 1

[Nag06] HN Nagaraja. Order statistics from independent exponential random variables and the sum of the top order statistics. In *Advances in Distribution Theory, Order Statistics, and Inference*, pages 173–185, 2006. 13, 14

[RG06a] Alejandro Ribeiro and Georgios B Giannakis. Bandwidth-constrained distributed estimation for wireless sensor networks-part i: Gaussian case. *IEEE transactions on signal processing*, 54(3):1131–1143, 2006. 1

[RG06b] Alejandro Ribeiro and Georgios B Giannakis. Bandwidth-constrained distributed estimation for wireless sensor networks-part ii: Unknown probability density function. *IEEE Transactions on Signal Processing*, 54(7):2784–2796, 2006. 1

[SWZ25] William Swartworth, David P. Woodruff, and Samson Zhou. Perfect l_p sampling with polylogarithmic update time. In *66th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2025. 1, 5

[TLJ10] Marina Thottan, Guanglei Liu, and Chuanyi Ji. Anomaly detection approaches for communication networks. In *Algorithms for Next Generation Networks*, Computer Communications and Networks, pages 239–261. Springer, 2010. 1

[TW11] Srikanta Tirthapura and David P. Woodruff. Optimal random sampling from distributed streams revisited. In *Distributed Computing - 25th International Symposium, DISC. Proceedings*, pages 283–297, 2011. 1

[TZ04] Mikkel Thorup and Yin Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 615–624, 2004. 3

[Vit85] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985. 1

[WGZ20] Hao Wu, Junhao Gan, and Rui Zhang. Learning based distributed tracking. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2040–2050, 2020. 1

[Woo04] David P. Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 167–175, 2004. 3

[WXZ25] David P. Woodruff, Shenghao Xie, and Samson Zhou. Perfect sampling in turnstile streams beyond small moments. *Proc. ACM Manag. Data*, 3(2):106:1–106:27, 2025. 1, 5, 7

[WZ12] David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC*, pages 941–960. ACM, 2012. 1, 3, 4, 5, 10, 19, 29, 70

[WZ14] David P. Woodruff and Qin Zhang. An optimal lower bound for distinct elements in the message passing model. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 718–733, 2014. 5

[WZ21a] David P. Woodruff and Samson Zhou. Separations for estimating large frequency moments on data streams. In *48th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 112:1–112:21, 2021. 3, 10, 19

[WZ21b] David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1183–1196, 2021. 1, 3, 4, 8, 9, 11, 40, 42, 43, 50

[WZ24] David P. Woodruff and Samson Zhou. Adversarially robust dense-sparse tradeoffs via heavy-hitters. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS*, 2024. 3

[XZB05] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *Proceedings of the ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 169–180, 2005. 3

[XZH23] Zhongzheng Xiong, Xiaoyi Zhu, and Zengfeng Huang. Adversarially robust distributed count tracking via partial differential privacy. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS*, 2023. 3, 4

[YZ13] Ke Yi and Qin Zhang. Optimal tracking of distributed heavy hitters and quantiles. *Algorithmica*, 65(1):206–223, 2013. 1

[ZLO+07] Haiquan (Chuck) Zhao, Ashwin Lall, Mitsunori Ogihara, Oliver Spatscheck, Jia Wang, and Jun (Jim) Xu. A data streaming algorithm for estimating entropies of od flows. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC*, pages 279–290, 2007. 3