# Distributed Monitoring of Election Winners

Arnold Filtser
Ben-Gurion University
Beer-Sheva, Israel
arnoldf@cs.bgu.ac.il

Nimrod Talmon
Weizmann Institute of Science
Rehovot, Israel
nimrodtalmon77@gmail.com

## ABSTRACT

We consider distributed elections, where there is a center and $k$ sites. In such distributed elections, each voter has preferences over some set of candidates, and each voter is assigned to exactly one site such that each site is aware only of the voters assigned to it. The center is able to directly communicate with each of the sites. We are interested in designing communication-efficient protocols, allowing the center to maintain (i.e., declare) a candidate which, with arbitrary high probability, is guaranteed to be a winner, or at least close to being a winner. We consider various single-winner voting rules, such as variants of Approval voting and scoring rules, tournament-based voting rules, and several round-based voting rules. For these voting rules, we show that, using communication which is logarithmic in the number of voters, it is possible for the center to maintain such approximate winners. We complement our protocols with lower bounds. Our results have implications in various scenarios, such as aggregating customer preferences in online shopping websites or supermarket chains and collecting votes from different polling stations of political elections.

## Keywords

Voting, Distributed streams, Sublinear algorithms

## 1. INTRODUCTION

Elections are used extensively to aggregate preferences of voters. Some elections are centralized, but others are carried out in distributed settings. Consider, e.g., a supermarket chain consisting of a large number of stores, each collecting data on the purchases made in it; the managers at the chain headquarters might want to aggregate this data, to identify, say, the most popular items being sold. One solution would be to have a central database, collecting all data from all stores, and to compute the most popular items on this centralized database. As the number of customers might be huge it might not be practical to do so. Further, as the communication between the stores and the headquarters might be expensive, a more efficient solution would be to have some computations being made at each store, and to develop a protocol for efficient communication between the stores and the headquarters, to allow the managers at the headquarters to know, at each point in time, what are the most popular items being sold throughout the chain.

A similar situation happens in online shopping websites, where buyers from all around the world make purchases. As the design of modern websites is based on data centers, aggregating the data of all the buyers involves communicating in a distributed setting. Specifically, in order to identify the current trends, and as communication between data centers might be expensive, it is of interest to develop protocols for those data centers to communicate with a central entity.

In this paper, we model such situations as follows. We are considering an election whose electorate is distributed into $k$ sites. Assuming some common axis of time, we have that at each point in time, a new voter arrives and votes, and her vote is assigned to one of those $k$ sites. There is some center which is able to directly communicate with each of the $k$ sites. With respect to a voting rule $\mathcal{R}$, the goal of the center is to maintain, at any point in time, a candidate which is either an $\mathcal{R}$-winner of the whole election, or is close to being an $\mathcal{R}$-winner of the whole election. That is, we are interested in designing communication-efficient protocols, allowing the center to declare a candidate which, with high probability, is guaranteed to be an $\mathcal{R}$-winner in an election which is similar to the original election, except for adding up to $\epsilon$-fraction of voters. A more formal description of our model and our notion of approximation is given in Section 2.

Our model also catches scenarios of political polls and elections. That is, in political elections and in TV polls, it is usually the case that there are several polling stations, spread around the country or the region. Then, in order to compute the results of the election (or the intermediate results during the day of the poll), the voters' preferences from all those polling stations are aggregated at some central station. For example, in the general political elections held in Brazil in 2014, there were roughly 500000 polling stations, with an average of 300 voters per station.

We concentrate on single-winner voting rules, and develop protocols for maintaining approximate winners in such distributed elections for various voting rules, ranging from approval-based rules and scoring rules, to tournament-based rules and round-based voting rules. We develop some general techniques for designing protocols for maintaining approximate winners in distributed elections, which might be applicable for other voting rules and settings as well. We show how to apply those techniques for the rules we consider. We discuss the effect which several parameters have on the communication complexity, namely the number $n$ of

**Appears in:** *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

voters, the number $m$ of candidates, the required approximation $\epsilon$, and the number $k$ of sites. We complement our communication-efficient protocols with a lower bound.

As a by-product of our lower bounds for maintaining approximate Plurality winner in distributed elections, we improve the state-of-the-art lower bound on the COUNT-TRACKING problem, which is a central problem in distributed streams. In it, the task is to maintain a value which approximates the number of items in a given distributed stream. In the regime where $k \geq 1/\epsilon^2$, we improve the lower bound for COUNT-TRACKING from $\Omega(k)$, proved by Huang et al. [20, Theorem 2.3], to $\Omega(k \log n / \log k)$ (see Remark 1).

**Related Work**. There are several papers which deal with identifying election winners using time or space which is sublinear in the number of voters. Several papers [16, 14, 6] study sampling algorithms for winner determination as well as winner determination in the streaming model. Dey and Narahari [15] study sampling algorithms for estimating the margin of victory. These works deal with centralized elections, while the current paper considers distributed elections. In a series of papers, Chevaleyre et al. [8, 7] and Xia and Conitzer [26] define and study the compilation complexity of various voting rules; in their model, the electorate is partitioned into two parts, and the general concern is the amount of communication which needs to be transmitted between the two parts, in order to determine an election winner. In compilation complexity there are no rounds of communication, as only one message is being passed between the two parts. This stands in contrast to our protocols, which use small amounts of communication due to their use of several rounds of communication between the center and the sites.

Conitzer and Sandholm [10] study communication complexity for various voting rules, but they are interested in finding exact winners, and do not consider approximations. Further, in their model, each voter acts as a site. In a related paper, Service and Adams [1] do consider approximations, but they also consider each voter as a site, concentrate on deterministic protocols, and their notion of approximation is different: a candidate is "$(1-\epsilon)$-approximation" if $sc(a)/sc(w) \geq 1-\epsilon$, where $w$ is the winner and $sc$ is the score function. Some papers deal with vote elicitation [17, 9, 23, 22], and provide algorithms for finding approximate winners under various voting rules, by elicitating the voters' preference orders.

The model of computation which we study in the current paper is generally called the *continuous distributed monitoring* model. There is a fairly recent survey about this model [11], as well as quite extensive line of work studying various problems in this model, such as sampling-based protocols [13, 25], protocols for approximating moments [12, 2], protocols for counting with deletions [24] (interestingly, that paper specifically mentions elections as a motivation, but do not study it explicitly), heuristic protocols for monitoring most-frequent items [3], and randomized protocols for counting the number of items in a distributed stream and finding frequent items [20].

Our notion of approximation resembles the problem of electoral control (see, e.g., [4] and [19]), where the goal is to change the election outcome by, e.g., adding voters; and the problem of possible winners (see, e.g., [5, 18, 21, 27]), where the goal is to strategically complete some incomplete profile.

# 2. PRELIMINARIES

We begin by providing preliminaries regarding elections and voting rules and continue by providing preliminaries regarding continuous monitoring of distributed streams. We use standard notions from computational complexity. For $n \in \mathbb{N}$, we denote the set $\{1, \ldots, n\}$ by $[n]$.

## 2.1 Elections and Voting Rules

An *election* $E = (C, V)$ consists of a set of *candidates* $C = \{c_1, \ldots, c_m\}$ and a collection of *voters* $V = (v_1, \ldots, v_n)$. We consider both approval-based elections and ranked-based elections. In *approval-based* elections, each voter is associated with her set of approved candidates, such that $v_i \subseteq C$. We say that $v_i$ *approves* candidate $c$ if $c \in v_i$. In *ranked-based election* each voter is a total order $\succ_{v_i}$ over $C$.

A *single-winner voting rule* $\mathcal{R}$ is a function that gets an election $E$ and returns a set $\mathcal{R}(E) \subseteq C$ of (tied) co-winners of that elections, such that $c$ is a winner of the election $E$ under $\mathcal{R}$ if $c \in \mathcal{R}(E)$. Since we will be interested in designing protocols where the center cannot see the full election, it will not be possible for our protocols to guarantee to find an exact winner; therefore, we will be satisfied with protocols which guarantee to find an approximate winner, specifically, an $\epsilon$-winner, as defined next; we view an $\epsilon$-winner as a candidate which is not far from being the winner of the election.

DEFINITION 1 ($\epsilon$-WINNER). *A candidate $c$ is an* $\epsilon$-winner *in an election $E$ under some voting rule $\mathcal{R}$ if it can become a winner under $\mathcal{R}$ by adding at most $\epsilon n$ additional voters to $E$. That is, if $c \in \mathcal{R}(E')$ where $E'$ is similar to $E$ except for up to $\epsilon n$ additional voters.*

Next we define our voting rules of interest. We ignore issues of tie-breaking; specifically, candidate $c$ is a winner if there is some fixed tie-breaking that makes him a winner.

**Approval-based rules and scoring rules**. Under *Approval*, each voter approves a subset of the candidates (that is, it is held in approval-based elections), and the score of a candidate is the number of voters approving him. The candidates with the highest score tie as co-winners. *$t$-Approval* is similar to Approval, but with the restriction that each voter shall approve exactly $t$ candidates (that is, $|v_i| = t$). *Plurality* is a synonym for 1-Approval. Under *Borda*, a voter ranking a candidate in position $j$ is giving it $m - j$ points, and the candidate with the highest score wins.

**Tournament-based voting rules**. The *Copeland score* of a candidate $c$ is the number of other candidates $c' \neq c$ for which a majority of voters prefer $c$ to $c'$. *Copeland* selects a candidate with the highest score as a winner. A *Condorcet winner* is a candidate with Copeland score $m - 1$. Under *Condorcet*, a Condorcet winner is selected as a winner if it exists; otherwise, all candidates tie as co-winners.

**Round-based voting rules**. *Plurality with run-off* proceeds in two rounds. In the first round, it selects the two candidates with the highest Plurality scores, where the Plurality score of a candidate is defined as the number of voters ranking him first. In the second round, it considers only those two candidates and selects as a winner the one which is preferred to the other by the larger number of voters. *Bucklin* also proceeds in rounds. In round $i \in [m]$, it computes, for each candidate $c$, the number of voters ranking $c$ among their top $i$ choices. Then, if there is a candidate

with a strict majority of the voters ranking him among their top $i$ choices, then such a candidate is selected as a winner; otherwise, a new round begins.

## 2.2 Computational Model

In our computational model we have one center and $k$ sites, arranged in a star-shaped network, centered at the center, such that the center has a direct communication link to all sites but two sites cannot communicate directly.

We assume some axis of time, $t_1, \ldots, t_n$, and a stream of voters $v_1, \ldots, v_n$, such that voter $v_i$ comes at time $t_i$. Each voter is assigned to exactly one site, such that each site is aware only of the subset of voters which are assigned to it. We stress that the time is not known to either the center or the sites. Such a stream is called a *distributed stream*.

We are interested in designing communication-efficient protocols, whose goal is to allow the center to declare, at any point in time, a candidate $c$ which is, with constant probability (say, 0.9), an $\epsilon$-winner (see Section 4 for a discussion on higher probabilities).

A protocol is defined via the messages which the center and the sites send to each other, and can consist of several rounds. The protocol shall be correct not only at the end of the stream (which is usually the case in streaming algorithms), but shall be correct at any point in time. As it is the custom in protocols operating on distributed streams, we describe our upper bounds in terms of words of communication, where we assume that a word contains $\log n$ bits.

# 3. ALGORITHMIC TECHNIQUES

The naive protocol, where each site sends to the center a message for every voter which arrives to it, clearly solves our problem, however it uses communication which is linear in the number of voters. Specifically, it communicates $O(nm \log m)$ bits, since $m \log m$ bits are sufficient for sending a single vote (indeed, for Plurality, e.g., $\log m$ bits are sufficient). In this paper we are interested in protocols which use significantly less communication, namely communication which is polylogarithmic in the number of voters.

In this section we provide high level descriptions of three algorithmic techniques which are useful for developing protocols for maintaining approximate winners in distributed vote streams. Accordingly, in Section 4 we demonstrate how to realize and instantiate those algorithmic techniques as concrete protocols for maintaining approximate winners.

**Protocols Based on Counting Frequencies**. In the FREQUENCY-TRACKING problem, we are given a distributed stream where, instead of voters, the items of the stream come from a known universe of items. The goal is for the center to maintain, for each item type in the distributed stream, a value which approximates the frequency of that item type. More formally, let us denote the items of the stream by $v_1, \ldots, v_n$ and consider $m$ different item types, such that item $i$ (for $i \in [n]$) is of type $j$ (for $j \in [m]$) if $v_i = j$. Let us denote the frequency of item type $j$ by $f(j) = |\{i : v_i = j\}|$. A protocol solving the FREQUENCY-TRACKING problem guarantees that with constant probability, simultaneously for every item type $j$, the center can maintain a value $f'(j)$ such that $f'(j) \in f(j) \pm \epsilon n$. There is a protocol for the FREQUENCY-TRACKING problem in distributed streams whose communication complexity is $O((\epsilon^{-1}\sqrt{k} + k) \log n)$ [20]. As some voting rules operate by counting points for candidates, it is sometimes possible to reduce the problem of maintaining an $\epsilon$-winner under such voting rules to the problem of maintaining approximate frequencies.

**Protocols Based on Sampling**. Instead of sending all voters to the center, as the naive protocol does, it is natural to let each site send only some of the voters arriving to it. Specifically, we would like the center to have a uniform sample of the voters. Cormode et al. [13] describe a protocol for maintaining a sample of $s$ items chosen uniformly at random from a distributed stream; it's communication complexity is $O((k + s) \log n)$. Since we are sampling voters, we need to take into account the communication needed to send each of the sampled voters. Specifically, in approval-based elections (ranked-based elections) we need $\log 2^m$ (resp., $\log m!$) bits per voter. Since we count the communication complexity in words, each of which contains $\log n$ bits, we need $\lceil \log 2^m / \log n \rceil \leq 1 + m/\log n$ (resp., $\lceil \log m! / \log n \rceil \leq 1 + m \log m / \log n$) words per voter.

As Bhattacharyya and Dey [14] show, it is indeed possible to sample only a small number of voters from a given election in order to identify an $\epsilon$-winner of that election, for various voting rules. Notice that, unlike Bhattacharyya and Dey [14], which guarantee that the winner of the sampled election is a winner in the original election, where we are allowed to *change* the votes of at most $\epsilon n$ voters, we allow to *add* at most $\epsilon n$ voters. We can, however, transfer between these two notions: sometimes the transformation is immediate, and for the less obvious cases, we describe the specific transformations within our proofs, in Section 4.

**Protocols Based on Checkpoints**. While protocols based on counting frequencies or sampling are randomized (even though protocols based on counting frequencies can be made deterministic; see Section 6), protocols based on checkpoints are deterministic in nature. The general idea is as follows. Assume that the center knows an $\epsilon/2$-winner $c$ of the election containing the first $n'$ voters. Then, until the number of voters reaches $(1 + \epsilon/2)n'$, the center can usually still declare $c$ as an $\epsilon$-winner. (This is true for voting rules where, by adding at most $\epsilon n$ new voters, we can "correct" the at-most-$\epsilon n$ voters which arrived after the last checkpoint.)

This observation suggests protocols where the center only updates its declared candidate whenever the number of voters multiplies by an $(1 + \delta)$ fraction, for a frequency parameter $\delta = q \cdot \epsilon$ for some constant $q < 1$. We refer to those points in time with $1, (1 + \delta), (1 + \delta)^2, \ldots$ voters, which are the points in time where the center needs to update its declared $\epsilon$-winner, as *checkpoints*. Notice that the number of checkpoints is $\log_{1+\delta} n = O(\delta^{-1} \log n) = (\epsilon^{-1} \log n)$. In each checkpoint, the center shall initiate a (non-dynamic) subprotocol to identify an $\epsilon$-winner of the election so far.

Importantly, in order to identify those checkpoints, the center shall be able to count the number of voters arriving so far. Fortunately, there is an efficient deterministic protocol which uses $O(\epsilon^{-1} k \log n)$ words [28] for solving the COUNT-TRACKING problem, where the center maintains a value $n'$ such that $n' \in n \pm \epsilon n$, where $n$ is the actual number of items. Notice that for protocols based on checkpoints, it is enough to feed the COUNT-TRACKING protocol with $\epsilon' = \Omega(\epsilon)$, and, correspondingly, compute an $O(\epsilon)$-winner in each checkpoint. The communication complexity would remain asymptotically unchanged. Assuming that it is possible to compute an $\epsilon$-winner using $O(z)$ bits of communi-

| Voting rule | Frequencies | Checkpoints | Sampling |
|---|---|---|---|
| **Plurality** | $O((\epsilon^{-1}\sqrt{k}+k)\log n)$ | | |
| **$t$-Approval** | $O((\epsilon^{-1}\sqrt{kt}+k)\log tn)$ | $O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log n)\right)$ | $O((\epsilon^{-2}\log t+k)(m+\log n))$ |
| **Approval** | $O((\epsilon^{-1}\sqrt{km}+k)\log mn)$ | $O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log n)\right)$ | $O((\epsilon^{-2}\log m+k)(m+\log n))$ |
| **Borda** | $O((\epsilon^{-1}\sqrt{km}+k)\log mn)$ | $O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log n)\right)$ | $O((\epsilon^{-2}\log m+k)(m\log m+\log n))$ |
| **Condorcet** | $O((\epsilon^{-1}\sqrt{km^2}+k)\log mn)$ | $O\left(\frac{k}{\epsilon}(m^2\log\frac{k}{\epsilon}+\log n)\right)$ | $O((\epsilon^{-2}\log^3 m+k)(m\log m+\log n))$ |
| **Copeland** | $O((\epsilon^{-1}\sqrt{km^2}+k)\log mn)$ | $O\left(\frac{k}{\epsilon}(m^2\log\frac{k}{\epsilon}+\log n)\right)$ | $O((\epsilon^{-2}\log^3 m+k)(m\log m+\log n))$ |
| **Run Off** | $O((\epsilon^{-1}\sqrt{km^2}+k)\log mn)$ | $O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log n)\right)$ | $O((\epsilon^{-2}+k)(m\log m+\log n))$ |
| **Bucklin** | $O((\epsilon^{-1}\sqrt{km}\log^2 m+k)\log mn)$ | $O(\frac{k\log m}{\epsilon}(m\log\frac{k}{\epsilon}+\log n))$ | $O((\epsilon^{-2}\log m+k)(m\log m+\log n))$ |

**Table 1: Overview of our results. The results in this table are for randomized protocols. $\epsilon$ is the required approximation, $k$ is the number of sites, $m$ is the number of candidates, and $n$ is the number of voters. There are three columns of upper bounds, where the first is for protocols based on counting frequencies, the second is for protocols based on checkpoints, and the third is for sampling-based protocols. For Plurality with run-off there is a fourth protocol, based on sampling and checkpoints, which uses $O\left((\epsilon^{-2}+k\epsilon^{-1})\log n\right)$ words.**

cation, a protocol based on checkpoints would then need $O((k+z)\epsilon^{-1}\log n)$ bits of communication. As $z$ will be usually greater than $O(k)$, we would get $O(z\cdot\epsilon^{-1}\log n)$.[1]

Note that as the COUNT-TRACKING is done approximately, instead of having a checkpoint precisely after $(1+\delta)^i$ voters arrived, we will have a small multiplicative shift of, say, $1\pm\frac{\delta}{10}$. This can be easily handled by having a checkpoint when approximately $(1+\frac{\delta}{10})^i$ voters arrived. Indeed, we will have more checkpoints, but between the arriving of $(1+\frac{\delta}{10})^i$ voters to the arrival of $(1+\frac{\delta}{10})^{i+1}$ voters we will have a checkpoint for sure, which is enough. For the sake of clarity, throughout the paper we will ignore this issue and assume that the COUNT-TRACKING is precise.

# 4. RESULTS

Our upper bounds are summarized in Table 1. We begin with approval-based rules and scoring rules, continue with tournament-based rules, and finish with round-based rules.

Notice that we state our results for protocols which are correct with some constant probability, say 0.9. One can always achieve arbitrary high probability $\delta$, as follows: For protocols based on counting frequencies, one shall run $O(\log(1/\delta))$ instances of the FREQUENCY COUNT protocol and take the median; protocols based on checkpoints are deterministic anyhow; and the increase in communication for protocols based on sampling is usually quite small [14]. Due to space restrictions, we omit some proof details.

## 4.1 Approval-based Rules and Scoring Rules

A distributed stream for Plurality contains $m$ item types (one item type for each candidate). Given an approximate frequency for each type (that is, an approximate number of voters voting for each candidate), the center can safely declare the candidate with the highest approximate frequency.

THEOREM 1. *There is a protocol for* PLURALITY-WINNER-TRACKING *which uses* $O((\epsilon^{-1}\sqrt{k}+k)\log n)$ *words.*

PROOF. We use the efficient protocol for FREQUENCY-TRACKING [20] with $\epsilon'=\epsilon/2$. This allows the center to maintain, for each candidate $c$, a value which is guaranteed to be at most $\frac{\epsilon}{2}n$-far from the real number of voters voting for $c$. The center would declare the candidate $c$ for which the approximate frequency is the highest.

Let us denote the real frequency of a candidate $c$ by $f(c)$ (which equals its Plurality score), and its approximate frequency computed by the FREQUENCY-TRACKING protocol by $f'(c)$. For each $c'\neq c$, it holds that $f(c')\leq f'(c')+\frac{\epsilon}{2}n\leq f'(c)+\frac{\epsilon}{2}n\leq f(c)+\epsilon n$ where the first and third inequalities follows from the $\epsilon/2$-approximation and the second from our choice of $c$. Therefore, we conclude that $c$ is an $\epsilon$-winner. □

We go on to consider $t$-Approval, where each voter specifies $t$ candidates which she approves.

THEOREM 2. *There are three protocols for* $t$-APPROVAL-WINNER-TRACKING*, for* $t\leq m/2$*. Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{kt}+k)\log tn)$*,* $O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log n)\right)$*, and* $(\epsilon^{-2}\log t+k)(m+\log n)$ *words of communication.*

PROOF. For the first protocol, we reduce $t$-Approval to Plurality, as follows. Each site, upon receiving a voter $v$ which approves $t$ candidates, instead of considering the voter $v$, creates and considers $t$ voters, $v_1,\ldots v_t$, such that voter $v_i$ (for $i\in[t]$) is set to approve the $i$th approved candidate of $v$. For example, a voter approving $\{a,b,d\}$ would be reduced to three voters, approving $a$, $b$, and $d$, respectively.

The reduced election has $n'=nt$ voters, and will be executed with precision parameter $\epsilon'=\epsilon/2t$. Consider a candidate $c$ which is an $\epsilon'$-winner in the reduced election; we argue that $c$ is an $\epsilon$-winner in the original election. Indeed, we can add $\epsilon n$ voters, each approving $c$, while for each other candidate $c'$, at most $\epsilon n/2$ of them approve $c'$ (as $t\leq m/2$); thus, the relative score of $c$ increases by $\epsilon n/2=\epsilon' n'$. As $c$ is $\epsilon'$-winner in the reduced election, this is sufficient. By Theorem 1, the communication used is $O((\epsilon'^{-1}\sqrt{k}+k)\log n')=O((\epsilon^{-1}\sqrt{kt}+k)\log tn)$.

The second protocol is based on checkpoints, where we consider $O(\epsilon^{-1}\log n)$ checkpoints, and in each checkpoint we execute a subprotocol for identifying an $\epsilon/4$-winner. In the subprotocol, the center initiates communication with all sites, asking from each site to send an approximate score for each candidate. That is, each site, for each candidate $c$, sends the number of voters approving $c$, rounded to the closest multiplication of $\epsilon n/(4k)$. Such rounding is enough, since, summing up the possible errors from all $k$ sites, the center would have a value which is at most $\epsilon n/8$-far from the real score. Thus, the candidate $c$ with the highest score

[1] Huang et al. [20] provide a randomized protocol for COUNT-TRACKING which uses $O(\sqrt{k}\epsilon^{-1}\log n)$ bits of communication. As $z$ will be greater than $O(k)$, using randomization will not reduce the total asymptotic communication.

will indeed be an $\epsilon/4$-winner. Each site should communicate $\log(\frac{4k}{\epsilon})$ bits for each candidate. Thus, the total communication in each subprotocol is bounded by $k\lceil\frac{m\log\frac{4k}{\epsilon}}{\log n}\rceil \leq O(k(1+\frac{m\log\frac{k}{\epsilon}}{\log n}))$. The bound follows.

For the third protocol, we sample $O(\epsilon^{-2}\log t)$ voters, chosen uniformly at random, which are sufficient for identifying an $\epsilon$-winner under $t$-Approval [14, Theorem 7]. The result then follows from the discussion in Section 3. □

For Approval, we proceed similarly to $t$-Approval. Naturally, we have $m$ factors instead of $t$ factors in our bounds.

THEOREM 3. *There are three protocols for* APPROVAL-WINNER-TRACKING. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m + k)\log mn)$, $O\left(\frac{k}{\epsilon}(m\log\frac{k}{\epsilon}+\log n)\right)$, *and* $O((\epsilon^{-2}\log m + k)(m+\log n))$ *words of communication.*

We go on to consider Borda (which is defined over ranked-based elections), for which we describe three protocols.

THEOREM 4. *There are three protocols for* BORDA-*winner-tracking. Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m+k)\log mn)$, $O(\epsilon^{-1}k(m\log(k/\epsilon)+\log n))$, *and* $O((\epsilon^{-2}\log m+k)\ (m\log m+\log n))$ *words of communication.*

PROOF. We first discuss the impact of adding voters. For an arbitrary candidate $c$, consider two voters where one voter is ranking $c$ first and then the other candidates in an arbitrary order, and another voter is ranking $c$ first and then the other candidates in reverse order. Adding these two voters increases the score of $c$ by $2(m-1)$ while the score of all other candidates increases by $m-2$. Thus, by adding $\epsilon n$ voters, we can increase the relative score of $c$ by $\epsilon nm/2$.

The first protocol is based on reducing Borda to Plurality, similarly to the first protocol stated in Theorem 2. Specifically, each site, upon receiving a voter $v$ with preference order $c_1 \succ \ldots \succ c_m$, instead of considering the voter $v$, creates and considers $\sum_{j\in[m]} m-j < m^2$ voters, such that for $j\in[m]$, it creates $m-j$ voters, each approving $c_j$. For example, a voter $v : a \succ b \succ d$ would be transformed into three voters, approving $a$, $a$, $b$, respectively. In the reduced election we have $n' < m^2 n$ voters, where $n$ is the number of voters in the original election. We use the protocol for Plurality described in Theorem 1 with $\epsilon' = \epsilon/(4m)$. Let us denote the real frequency of a candidate $c$ in the reduced election by $f(c)$ and its computed approximate frequency by $f'(c)$. The error is bounded by $|f'(c)-f(c)| \leq \epsilon' n' < \frac{\epsilon}{4m}\cdot nm^2 = \frac{\epsilon nm}{4}$. Since by adding $\epsilon n$ voters we can increase the relative score of the chosen candidate by $\epsilon nm/2$, we are done.

The second protocol is based on checkpoints, where in each checkpoint we use a subprotocol to compute the Borda score of each candidate with accuracy of $\epsilon nm/8$. Similarly to the second protocol in Theorem 2 , the subprotocol uses $O(k(1+(m\log\frac{k}{\epsilon})/(\log n)))$ words, thus the total communication is as claimed. Correctness follows since adding $\epsilon n$ voters can increase the score of our candidate by $\epsilon nm/2$, which is more than the rounding error and the impact of the voters arriving after the last checkpoint.

The third bound follows by a protocol based on sampling, since, as shown by Bhattacharyya and Dey [14, Theorem 6], a sample of $s = O(\epsilon^{-2}\log m)$ voters chosen uniformly at random is sufficient to identify an $\epsilon$-Borda-winner with constant probability. The result then follows from the discussion in Section 3. □

## 4.2 Tournament-based Rules

In this section we consider Condorcet winners and the Copeland voting rule. Both rules are built upon the tournament defined over the election by considering head-to-head contests between all pairs of candidates.

THEOREM 5. *There are three protocols for* COPELAND-WINNER-TRACKING. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m^2+k)\log mn)$, $O\left(\frac{k}{\epsilon}(m^2\log\frac{k}{\epsilon}+\log n)\right)$, *and* $O((\epsilon^{-2}\log^3 m+k)(m\log m+\log n))$ *words.*

PROOF. For the first protocol, we reduce each voter, corresponding to a total order over the candidates, to $O(m^2)$ items; specifically, the reduced distributed stream will contain items of $O(m^2)$ item types, where for each pair of candidates $c_1$ and $c_2$ we have a different type, denoted by $(c_1,c_2)$. The reduction proceeds as follows. Each site, upon receiving a voter $v$ which specifies a linear order, instead of considering the voter $v$, creates and considers $\binom{m}{2}$ items, such that if $v$ prefers $c_1$ to $c_2$, then we create an item $(c_1,c_2)$ (notice that this is an ordered tuple). For example, a voter $v : a \succ b \succ d$ would be transformed into three items, $(a,b)$, $(a,d)$, and $(b,d)$. The reduced distributed stream has $n' = O(m^2 n)$ items, and $O(m^2)$ types of items. For two candidates $c_1$ and $c_2$, let $N(c_1,c_2)$ denote the number of voters preferring $c_1$ to $c_2$. Now we can use a protocol based on counting frequencies (see Section 3), with $\epsilon' = \epsilon/(4m^2)$, to let the center maintain, for each pair of candidates $c_1$ and $c_2$, a value $N'(c_1,c_2)$ such that $N'(c_1,c_2) \in N(c_1,c_2) \pm \epsilon' n' = N(c_1,c_2) \pm \epsilon n/4$.

Then, for a pair of candidates $c_1$ and $c_2$, the center defines that $c_1$ wins over $c_2$ in a head-to-head contest if $N'(c_1,c_2) \geq n/2 - \epsilon n/4$. Finally, the center declares as an $\epsilon$-winner a candidate $c$ for which the center defines that he wins over the largest number of other candidates. Correctness follows since a candidate $c$ which is declared as an $\epsilon$-winner is guaranteed to have at least $N(c,c') > n/2 - \epsilon n/2$ for the largest number of $c' \neq c$. Then, by adding $\epsilon n/2$ voters which rank $c$ on top and then the other candidates in arbitrary order, and another $\epsilon n/2$ voters which rank $c$ on top and then the other candidates in reverse order, we causes a sufficient increase of $N(c,c')$ for each $c'$, while not increasing the number of wins of any other candidate. The communication complexity follows by the discussion given in Section 3, the size of the reduced distributed stream, and our choice of $\epsilon'$.

The second protocol is based on checkpoints. In each checkpoint, we use a subprotocol that for every two candidates $c$ and $c'$, computes the number of voters preferring $c$ over $c'$ within a (rounding) error of $\epsilon n/4$. As there are $m^2$ quantities to estimate, following the analysis of the second protocol in Theorem 2, the subprotocol uses $k\lceil\frac{m^2\log\frac{2k}{\epsilon}}{\log n}\rceil$ words. The total communication follows. In each checkpoint, a candidate achieving estimated score higher that $\frac{n}{2} - \frac{\epsilon n}{4}$ for the maximal number of times is declared a winner. Correctness follows as additional $\epsilon n$ voters can increase the relative values of $N(c,c')$ for our candidate by $\epsilon n$, while not increasing the relative score of the other candidates.

The third protocol is a sampling-based protocol, based on the fact that a sample of size $O(\epsilon^{-2}\log^3 m)$ is sufficient for Copeland [14, Theorem 9][2] □

[2]Recall that Bhattacharyya and Dey [14] allow to change (rather then add) $\epsilon n$ voters. Notice that, for Copeland, we can reduce changing $\epsilon n$ voters to adding $2\epsilon n$ voters. Specifically, the relative impact of changing voter $v$ to $v'$ can be imitated by adding the voter $v'$ and the "reverse" voter of $v$.

Since Copeland can be seen as a relaxation of Condorcet, we can use the protocols for Copeland as protocols for Condorcet.

THEOREM 6. *There are three protocols for* CONDORCET-WINNER-TRACKING. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m^2 + k)\log mn)$, $O\left(\frac{k}{\epsilon}(m^2 \log \frac{k}{\epsilon} + \log n)\right)$, *and* $O((\epsilon^{-2}\log^3 m + k)(m \log m + \log n))$ *words.*

## 4.3 Round-based Rules

Next we consider two round-based voting rules. We begin with Plurality with run-off and then continue to Bucklin.

THEOREM 7. *There are four protocols for* PLURALITY-WITH-RUN-OFF-WINNER-TRACKING. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m^2 + k)\log mn)$, $O(\epsilon^{-1}k(m\log(k/\epsilon) + \log n))$, $O((\epsilon^{-2} + k)(m\log m + \log n))$, *and* $O\left((\epsilon^{-2} + k\epsilon^{-1})\log n\right)$ *words.*

PROOF. The first protocol is based on counting frequencies. We combine the protocol for Plurality, described in the proof of Theorem 1, with the protocol for Condorcet, described in the proof of Theorem 5. Specifically, the center can identify two candidates with the highest Plurality scores using the protocol for Plurality and use the results of the protocol for Condorcet to decide which of those two candidates it shall declare as an $\epsilon$-winner. We will use both protocols with parameter $\epsilon' = \epsilon/3$. Assuming that the frequency counting protocol for Plurality identifies $c_1$ and $c_2$ as the candidates with the highest Plurality scores and the frequency counting protocol for Condorcet decides that $c_1$ wins over $c_2$, we can add $2\epsilon n/3$ voters ranking $c_1$ on top and another $\epsilon n/3$ voters ranking $c_2$ on top; this guarantees that $c_1$ and $c_2$ have the highest Plurality score and a majority of the voters prefer $c_1$ over $c_2$.

The second protocol is based on checkpoints with frequency parameter $\epsilon' = \epsilon/8$ (see Section 3). In each checkpoint we execute a subprotocol which operates in two rounds. In the first round, we compute the Plurality score of all the candidates, within $\epsilon n/16$ error. This allows the center to identify and broadcast two candidates with the approximate highest Plurality scores, say $c_1$ and $c_2$. In the second round, each site sends the exact number of voters preferring $c_1$ to $c_2$. Finally, the center declares $c_1$ ($c_2$) as an $\epsilon$-winner if the there are at least $n/2$ voters preferring $c_1$ to $c_2$ ($c_2$ to $c_1$). In the first round, each site sends at most $1 + m\log\frac{8k}{\epsilon}/\log n$ words, while in the second round each site sends at most 1 word. The claimed communication complexity then follows.

Next we show that the declared winner $c$ is indeed an $\epsilon$-winner at any point until the next checkpoint. Let $c'$ be the second Plurality winner. Given that $z \leq \epsilon' n$ voters arrived after the last checkpoint, we can add $4z$ voters which rank $c$ on top and $3z$ voters ranking $c'$ on top. As a result, $c$ and $c'$ will be the two candidates with the highest Plurality score, and $c$ will win over $c'$ in a head-to-head contest.

The third, sampling-based protocol (see Section 3 ), proceeds by sampling $O(\epsilon^{-2})$ voters uniformly at random, which are sufficient for Plurality with run-off [14, Theorem 11].

The fourth protocol is a "hybrid" protocol which combines checkpoints and sampling. During the protocol we maintain a sample of $O(\epsilon^{-2})$ voters, chosen uniformly at random, specifically storing only their top candidates.

Then, at each checkpoint we use that sample to identify two candidates $c$ and $c'$ with the highest (approximated) Plurality score[3]. Given $c$ and $c'$, the center collects from all sites the number of voters preferring $c$ over $c'$, and declares as a winner the one which is preferred by more voters. The protocol uses $O\left((\epsilon^{-2} + k)\log n\right) + O\left(k\epsilon^{-1}\log n\right) = O\left((\epsilon^{-2} + k\epsilon^{-1})\log n\right)$ words. Correctness follows similarly to the checkpoints protocol described above. □

THEOREM 8. *There are three protocols for* BUCKLIN-WINNER-TRACKING. *Respectively, the protocols use* $O((\epsilon^{-1}\sqrt{k}m\log^2 m + k)\log mn)$, $O(\epsilon^{-1}km\log m\log n)$, *and* $O((\epsilon^{-2}\log m + k)(m\log m + \log n))$ *words.*

PROOF. For simplicity we assume that $m$ is even (otherwise we can add one dummy candidate). By averaging arguments, a Bucklin winner is necessarily found within the first $m/2$ rounds. Observe that adding a voter and its reverse does not change anything (i.e., any candidate $c$ at any level $j$ has a majority after the addition if and only if it had a majority before the addition). Let $c$ be an arbitrary candidate and consider adding two voters, each ranking $c$ on top, and ranking the other candidates in reverse orders. As a result, the score of $c$ increases by 2 for each level $j \leq m/2$, while the status of each candidate $c' \neq c$ is only weaker (thus, if $c'$ does not have a majority at level $j$ before the addition, then it will also not have a majority after the addition).

The first protocol is based on counting frequencies. It begins by reducing the distributed vote stream into a different distributed stream. Specifically, each site, upon receiving a voter $v$, instead of considering the voter, creates for each candidate $c_l$ ($l \in [m]$), the items $(c_l, i, j)$ for each $i \in [0, \log m - 1]$ and for each $j \in [0, m/2^i - 1]$ for which it holds that $v$ ranks $c_l$ between the $(j \cdot 2^i + 1)$'th position and the $((j+1)\cdot 2^i)$'th position. The idea is that we can recover the approximate number of voters ranking each $c$ at the first $j$ positions using $\log m$ approximate counters of these items.

The protocol initiates a FREQUENCY-TRACKING protocol on the reduced distributed stream with $\epsilon' = \epsilon/(2m\log^2 m)$. This will give us approximate values on the number of items of each type in our reduced distributed stream. Let us denote, for a candidate $c_i$ and position $j$ (for $i \in [m]$ and $j \in [m]$), the number of voters ranking $c_i$ at any position $j' \leq j$ by $N(c_i, j)$. Then, we can approximate each of the values $N(c_i, j)$ by adding $\log m$ different approximated frequencies, computed by the FREQUENCY-TRACKING protocol (on the reduced stream).

Using these approximations of $N(c_i, j)$, denoted by $N'(c_i, j)$, we are now able to simulate Bucklin; specifically, the center finds the minimum $j$ for which there is at least one $c_i$ for which $N'(c_i, j) \geq \frac{n}{2} - \frac{\epsilon n}{2}$. Next we show correctness. The size of the reduced distributed stream is $n' = nm\log m$, since each voter is transformed into $m\log m$ items, specifically $\log m$ per each candidate. To approximate the value $N(c_i, j)$ we add up $\log m$ approximate frequencies, each of which can be wrong by at most $\epsilon' n' = \epsilon n/2\log m$; thus, the value of $N'(c_i, j)$ can be wrong by at most $\epsilon n/2$. Therefore, in each level $j' < j$ where we do not find a winner, there is indeed no candidate with a majority. Finally, $\epsilon n$ additional voters can indeed make our chosen candidate a winner.

The second protocol is based on checkpoints with frequency parameter $\epsilon' = \epsilon/4$. Each checkpoint contains $\log m$

[3]Note that, while Bhattacharyya and Dey [14] concentrate on identifying an $\epsilon$-winner, they estimate the Plurality scores of all candidates within an $\epsilon n$ error.

rounds, where in those $\log m$ rounds, the center is performing an approximate binary search to find the first $j$ for which there is at least one candidate $c_i$ for which the approximation for $N(c_i, j)$ is greater than $\frac{n}{2} - \frac{\epsilon n}{8}$, and declares this $c_i$ as an $\epsilon$-winner; when considering an index $j$, we approximate $N(c_i, j)$ for every $i$ within $\epsilon n/8$ error. The binary search is done in the standard order. Note that if we stopped at level $j$, then necessarily for every $j' < j$ there was no candidate with $n/2$ votes.

The communication bound follows since in each subprotocol we use $k\lceil \frac{m \log 4k/\epsilon}{\log n} \rceil$ words. For correctness, let $c$ be the declared winner and assume that additional $\delta n < \epsilon' n$ voters arrived since the last checkpoint. Let us assume that all of them ranked $c$ last, as this is the hardest. By adding $\delta n$ reversed voters to those which arrived, and additional $\epsilon n/2$ voters that rank $c$ first (and such that every $c' \neq c$ is ranked among the first $m/2$ places at most half of the time) we can assure that $c$ becomes a winner.

The third bound follows by sampling $O(\epsilon^{-2} \log m)$ voters uniformly at random, which is sufficient for Bucklin [14, Theorem 10]. □

# 5. LOWER BOUND

In this section we prove an almost tight lower bound (up to a factor of $\log k$) for PLURALITY-WINNER-TRACKING. The lower bound holds already for Plurality with 2 candidates. This lower bound applies as well to all other voting rules we consider, via the following reduction. Assuming a protocol for a voting rule $\mathcal{R}$, we can use it as a black-box for solving Plurality with 2 candidates: for each Plurality voter which arrives and approves some candidate, say $c$, we create a voter ranking $c$ on top and the remaining candidate after him [4].

Notice that in our lower bound, we assume, as it is usual when studying distributed streams, that there is no spontaneous communication; that is, the center can initiate communication only as a result of receiving a message from the sites, and each site can initiate communication only as a result of receiving a stream item or a message from the center.

Now we are ready to state our lower bound; the proof of the corresponding theorem (that is, Theorem 9) appears at the end of the section, and is based on Lemma 1 and Lemma 2. Recall that for PLURALITY-WINNER-TRACKING, Theorem 1 provides an upper bound of $O((\epsilon^{-1}\sqrt{k}+k)\log n)$.

THEOREM 9. *Any randomized protocol for* PLURALITY-WINNER-TRACKING *uses at least* $\Omega((\epsilon^1\sqrt{k}+k)\log n/\log k)$ *words of communication, even when* $m = 2$.

The next lemma shows a lower bound when $k < \epsilon^{-2}$.

LEMMA 1. *If* $k < \epsilon^{-2}$, *then any randomized protocol for* PLURALITY-WINNER-TRACKING *uses at least* $\Omega(\epsilon^{-1}\sqrt{k}\log n)$ *words of communication, even when* $m = 2$.

PROOF. We reduce COUNT-TRACKING to PLURALITY-WINNER-TRACKING. To this end, we assume, towards a contradiction, that there is a protocol for PLURALITY-WINNER-TRACKING with $o(\epsilon^{-1}\sqrt{k}\log n)$ communication complexity, and describe a protocol with the same communication complexity for COUNT-TRACKING. For $k < \epsilon^{-2}$ this leads to a contradiction, since there is a lower bound of $\Omega(\epsilon^{-1}\sqrt{k}\log n)$ for COUNT-TRACKING where $k < \epsilon^{-2}$ [20, Theorem 2.4].

The distributed stream for COUNT-TRACKING contains items of only one type, and a protocol for COUNT-TRACKING maintains a value $n'$ such that $n' \in n \pm \epsilon n$, where $n$ is the number of items in the distributed stream. Let us treat those items as voters, each of which is approving the candidate $c_1$.

The general idea of the reduction is for the center to simulate another site, called a *ghost site* (since it is not a real site, just simulated by the center), to which the center will send *ghost voters* (again, not real voters, but only simulated by the center). The center will simulate a protocol for Plurality with voters approving $c_1$ going to the $k$ "real" sites, and simulated voters approving $c_2$ going to the ghost site. Specifically, the center has three parts. The first part is a center for a PLURALITY-WINNER-TRACKING protocol operating on $k+1$ sites. The second part is a site in a PLURALITY-WINNER-TRACKING protocol; this is the ghost site. The third part is for the center to inspect the PLURALITY-WINNER-TRACKING protocol from above, and (using knowledge about the current winner) to send voters approving the candidate $c_2$ to the ghost site.

Let us denote the number of voters voting for $c_1$ ($c_2$) by $s(c_1)$ (respectively, $s(c_2)$). Set $\delta = \epsilon/10$. The protocol for PLURALITY-WINNER-TRACKING will work with respect to approximation $\delta$, and will consist of $k+1$ sites.

Next we describe the logic of the third part of the center. The estimation for COUNT-TRACKING will be est $= (1+3\delta)s(c_2)$ (note that only the ghost site receives voters approving $c_2$, hence the center knows $s(c_2)$ exactly).

Before there is any communication from the (real) sites to the center, we set $s(c_2) = 0$. Then, at some point in time there will be some communication from the sites to the center indicating that some voters approving $c_1$ arrived; specifically, the first part of the center would declare $c_1$ as the winner of the election. More generally, our protocol works in phases, where a phase starts when the center "flip"s its estimation; that is, the (first part of the) center changes the estimation for the Plurality winner from $c_2$ to $c_1$. When such a flip occurs, the center sends some ghost voters (approving $c_2$) to the ghost site until $s(c_2) = (1+3\delta)^i$ (for some $i$) and a flip (from $c_1$ back to $c_2$) occurs. (That is, we send ghost voters until a flip occurs and then send some additional voters until we reach a power of $1+3\delta$; reaching this power of $1+3\delta$ is actually not needed, but it does not affect the communication complexity and it makes the analysis cleaner.) We assume, as is usually done in distributed streams, that communication and internal computation happens instantly. Thus, we have that $c_2$ is always the winner of the PLURALITY-WINNER-TRACKING protocol. This finishes the description of the reduction.

Next we argue that our estimation (for COUNT-TRACKING) is accurate. Specifically, we will show that $s(c_1) \leq$ est $\leq (1+\epsilon)s(c_1)$. As $c_2$ is always the winner, it always holds that $s(c_2) + \delta(s(c_1) + s(c_2)) \geq s(c_1)$. Hence, since $\delta < 1/10$ (as $\epsilon < 1$), it holds that:

$$s(c_1) \leq \frac{1+\delta}{1-\delta} \cdot s(c_2) < (1+3\delta) \cdot s(c_2) = \text{est} \ .$$

[4]The t-Approval voting rule requires different reduction. We will have two special candidates, $c_1$ and $c_2$, which correspond to the candidates in the Plurality instance. For each arriving Plurality voter (which vote for, say, $c_1$), we create two voters, both approving $c_1$; one of them approving $c_2$; and any other candidate $c'$ is approved by at most one of them (recall that $t \leq m/2$).

Fix $s(c_2) = (1+3\delta)^i$. Note that when $s(c_2)$ was equal to $(1+3\delta)^{i-1}$, the protocol for PLURALITY-WINNER-TRACKING considered $c_1$ as the winner. Hence $s(c_1) + \delta(s(c_1) + (1+3\delta)^{i-1}) \geq (1+3\delta)^{i-1}$, therefore

$$s(c_1) \geq \frac{1-\delta}{1+\delta}(1+3\delta)^{i-1} \geq \frac{(1+3\delta)}{(1+3\delta)^3}(1+3\delta)^i \geq \frac{\text{est}}{1+\epsilon} .$$

Note that until the next flip, $s(c_1)$ can only grow, while our estimation remains unchanged. Hence, it will still hold that est $\leq (1+\epsilon)s(c_1)$. Finally, we have that the communication of our protocol is bounded by $o(\delta^{-1}\sqrt{k+1}\log(s(c_1) + s(c_2))) = o(\epsilon^{-1}\sqrt{k}\log n)$, which contradicts the lower bound for COUNT-TRACKING discussed above. □

The next lemma is especially interesting for $k \geq \epsilon^{-2}$.

LEMMA 2. *Any randomized protocol for* PLURALITY-WINNER-TRACKING *uses at least* $\Omega(k \log n / \log k)$ *words of communication, even when* $m = 2$.

PROOF. We assume that $\epsilon < \frac{1}{3}$. Consider a protocol for PLURALITY-WINNER-TRACKING which is correct with constant probability on every input. Next we describe a distributed stream of voters which come to the sites. Specifically, the stream consists of $s$ phases. Let $x_1 = 1$, $y_1 = 1$, $x_i = (1+3\epsilon) \cdot k \cdot y_{i-1}$ and $y_i = y_{i-1} + x_i$. During the $i$'s phase, $x_i$ voters will go to each site and vote for the candidate $c_{(i \bmod 2)}$. Note that after the $i$'s phase, exactly $y_i$ voters voted at each site. The total number of votes for $c_{(i \bmod 2)}$ is at least $k \cdot x_i$, while the total number of votes for $c_{(i-1 \bmod 2)}$ is at most $k \cdot y_{i-1}$. In particular, $c_{(i \bmod 2)}$ is a unique $\epsilon$-winner. By induction it holds that $y_i = y_{i-1} + (1+3\epsilon) \cdot k \cdot y_{i-1} < 3k \cdot y_{i-1} \leq (3k)^{i-1}$; hence, the total number of voters is $n = k \cdot y_s < (3k)^s$. In particular, $s = \Omega(\frac{\log n}{\log k})$.

Next consider the $j$'s site $S_j$ during the phase $i$. Let $Y_{i,j}$ be the event that some communication between the center and $S_j$ occurs. Let $Z_{i,j}$ be the event that the center initiates communication with $S_j$. Let $X_{i,j}$ be the event that $S_j$ initiates communication with the center, conditioned on the event that the center does not initiate communication with $S_j$ (that is, $Y_{i,j}$ conditioned on $\overline{Z_{i,j}}$). We argue that $\mathbb{E}[X_{i,j}] = \Omega(1)$. Before the $i$'s phases starts, $c_{(i-1 \bmod 2)}$ is the unique $\epsilon$-winner. Consider an alternative scenario where, after the end of the $i-1$'s phase, $x_i$ voters come to $S_j$ (and vote for $c_{(i \bmod 2)}$), while no additional voters arrive. In this alternative scenario the center will not initiate communication with $S_j$, as from its point of view nothing have changed since the end of the $(i-1)$'s phase (since it did not receive any new messages). Note also that in the alternative scenario, $c_{(i \bmod 2)}$ is the unique $\epsilon$-winner. This is since

$$\begin{aligned} k \cdot y_{i-1} + \epsilon(k \cdot y_{i-1} + x_i) &= k \cdot y_{i-1}\left(1 + \epsilon(1 + (1+3\epsilon))\right) \\ &= k \cdot y_{i-1}\left(1 + 2\epsilon + 3\epsilon^2\right) < x_i. \end{aligned}$$

Thus, if $S_j$ will not initiate communication with the center, then, in the alternative scenario, the center would not hold the right estimation both at the end of the $i-1$'s phase and at the end of the $i$'s phase. This is so since it will have the same estimation, while there are different unique $\epsilon$-winners at those times. Therefore, the probability that the center is right in both of these times is bounded by $\Pr[X_{i,j}]$. As the center has constant probability to have the right estimation twice, we conclude that $\mathbb{E}[X_{i,j}] = \Omega(1)$.

Let us go back to our original scenario. Set $\Pr[Z_{i,j}] = \alpha$. Then, we have that:

$$\begin{aligned} \mathbb{E}[Y_{i,j}] &= \mathbb{E}[Z_{i,j}] + \Pr\left[\overline{Z_{i,j}}\right]\mathbb{E}[X_{i,j}] \\ &= \alpha + (1-\alpha) \cdot \Omega(1) = \Omega(1) . \end{aligned}$$

The total communication during the whole protocol is lower bounded by $\sum_{i=1}^{s}\sum_{j=1}^{k}\mathbb{E}[Y_{i,j}] = \Omega(sk) = \Omega\left(\frac{k \log n}{\log k}\right)$. □

We are ready to prove Theorem 9.

PROOF OF THEOREM 9 . If $k < \epsilon^{-2}$, then Lemma 1 provides us with a lower bound of $\Omega(\frac{\sqrt{k}}{\epsilon}\log n) = \Omega\left((\frac{\sqrt{k}}{\epsilon} + k)\frac{\log n}{\log k}\right)$ .

Otherwise ($k \geq \epsilon^{-2}$), using Lemma 2 we get a lower bound of $\Omega(\frac{k \log n}{\log k}) = \Omega\left((\frac{\sqrt{k}}{\epsilon} + k)\frac{\log n}{\log k}\right)$ . □

REMARK 1. *For the* COUNT-TRACKING *problem in the regime where* $k \geq \epsilon^{-2}$*, Huang et al. [20, Theorem 2.3] give a lower bound of* $\Omega(k)$*. As there is a straightforward reduction from* PLURALITY-WINNER-TRACKING *with two candidates to* COUNT-TRACKING*, Lemma 2 implies a* $\Omega(\frac{k \log n}{\log k})$ *lower bound for the* COUNT-TRACKING *problem.*

# 6. DISCUSSION AND OUTLOOK

We begin this section with a discussion on deterministic protocols, followed by a brief discussion on the choice of which protocol to use at which scenario. Then, we mention some directions for future research.

**Deterministic protocols**. While in this paper we concentrated on randomized protocols, it turns out that there are efficient deterministic protocols as well (this is arguably quite surprising; for example, there are usually no efficient deterministic algorithms operating on centralized streams). Indeed, while there are no deterministic equivalents to our sampling-based protocols, our other protocols can generally be made deterministic. Specifically, protocols based on checkpoints are already deterministic; protocols based on counting frequencies can use a deterministic protocol for FREQUENCY COUNT which uses $O(\epsilon^{-1}k \log n)$ words of communication [28]. Correspondingly, the increase in the communication complexity is by at most a factor of $\sqrt{k}$. Notice that the corresponding deterministic protocols still maintain only approximate solutions.

**Choice of protocol**. A closer look at our upper bounds reveals that the choice of which protocol to use for which voting rule crucially depends on the relationships between the various parameters; specifically, as a rule of thumb, it looks as if the choice of which protocol to use depends on the relation between $k$ and $1/\epsilon^{-2}$; specifically, if $k < 1/\epsilon^{-2}$, then protocols based on counting frequencies or on checkpoints shall be used, while if $k \geq 1/\epsilon^{-2}$, then sampling-based protocols achieve better communication complexity. We believe that both cases make sense; for example, in a supermarket chain with 4000 stores, requiring approximation of $\epsilon = 1/100$ would put us in the first case, while requiring $\epsilon = 1/10$ would put us in the second case.

**Future Directions**. As future directions, we mention (1) improving our (upper and lower) bounds, (2) considering further (single-winner and multiwinner) voting rules, (3) considering limited time and space for the sites, and (4) considering domain restrictions.

## REFERENCES


[1] J. A. Adams and T. C. Service. Communication complexity of approximating voting rules. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent (AAMAS '12)*, pages 593–602, 2012.

[2] C. Arackaparambil, J. Brody, and A. Chakrabarti. Functional monitoring without monotonicity. In *Automata, Languages and Programming*, pages 95–106. 2009.

[3] B. Babcock and C. Olston. Distributed top-$k$ monitoring. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (CDM '03)*, pages 28–39, 2003.

[4] J. Bartholdi, III, C. Tovey, and M. Trick. How hard is it to control an election? *Mathematical and Computer Modeling*, 16(8/9):27–40, 1992.

[5] N. Betzler, S. Hemmann, and R. Niedermeier. A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*, pages 53–58, July 2009.

[6] A. Bhattacharyya and P. Dey. Fishing out winners from vote streams. *arXiv preprint arXiv:1508.04522*, 2015.

[7] Y. Chevaleyre, J. Lang, N. Maudet, and J. Monnot. Compilation and communication protocols for voting rules with a dynamic set of candidates. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK '11)*, pages 153–160, 2011.

[8] Y. Chevaleyre, J. Lang, N. Maudet, and G. Ravilly-Abadie. Compiling the votes of a subelectorate. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI '09)*, pages 97–102, 2009.

[9] V. Conitzer and T. Sandholm. Vote elicitation: Complexity and strategy-proofness. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02)*, pages 392–397, 2002.

[10] V. Conitzer and T. Sandholm. Communication complexity of common voting rules. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC' 05)*, pages 78–87, 2005.

[11] G. Cormode. The continuous distributed monitoring model. *ACM SIGMOD Record*, 42(1):5–14, 2013.

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

[13] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang. Continuous sampling from distributed streams. *Journal of the ACM (JACM)*, 59(2):10, 2012.

[14] P. Dey and A. Bhattacharyya. Sample complexity for winner prediction in elections. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*, pages 1421–1430, 2015.

[15] P. Dey and Y. Narahari. Estimating the margin of victory of an election using sampling. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI '15)*, pages 1120–1126, 2015.

[16] P. Dey, N. Talmon, and O. van Handel. Proportional representation in vote streams. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '17)*, 2017. To appear.

[17] S. Dhamal and Y. Narahari. Scalable preference aggregation in social networks. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing (HCOMP '13)*, 2013.

[18] E. Elkind, P. Faliszewski, and A. Slinko. Cloning in elections: Finding the possible winners. *Journal of Artificial Intelligence Research*, 42:529–573, 2011.

[19] P. Faliszewski and J. Rothe. Control and bribery in voting. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 7. Cambridge University Press, 2015.

[20] Z. Huang, K. Yi, and Q. 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 (PODS '12)*, pages 295–306, 2012.

[21] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proceedins of the Multidisciplinary IJCAI-05 Worshop on Advances in Preference Handling*, pages 124–129, July/August 2005.

[22] D. T. Lee. Efficient, private, and $\epsilon$-strategyproof elicitation of tournament voting rules. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI '15)*, 2015.

[23] D. T. Lee, A. Goel, T. Aitamurto, and H. Landemore. Crowdsourcing for participatory democracies: Efficient elicitation of social choice functions. In *Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing (HCOMP' 14)*, 2014.

[24] Z. Liu, B. Radunovic, and M. Vojnovic. Continuous distributed counting for non-monotonous streams. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '12)*, pages 307–318, 2012.

[25] S. Tirthapura and D. P. Woodruff. Optimal random sampling from distributed streams revisited. In *Proceeding of the 25th international conference on Distributed computing (DISC '11)*, pages 283–297, 2011.

[26] L. Xia and V. Conitzer. Compilation complexity of common voting rules. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI '10)*, pages 915–920, 2010.

[27] L. Xia and V. Conitzer. Determining possible and necessary winners given partial orders. *Journal of Artificial Intelligence Research*, 41:25–67, 2011.

[28] K. Yi and Q. Zhang. Optimal tracking of distributed heavy hitters and quantiles. *Algorithmica*, 65(1):206–223, 2013.