Symposium on Theoretical Aspects of Computer Science year (city), pp. numbers
www.stacs-conf.org

CONTINUOUS MONITORING OF DISTRIBUTED DATA STREAMS OVER A TIME-BASED SLIDING WINDOW

HO-LEUNG CHAN [1] AND TAK-WAH LAM [1] AND LAP-KEI LEE [2] AND HING-FUNG TING [1]

[1] Department of Computer Science, University of Hong Kong, Hong Kong
E-mail address: {hlchan,twlam,hfting}@cs.hku.hk

[2] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
E-mail address: lklee@mpi-inf.mpg.de

Abstract. The past decade has witnessed many interesting algorithms for maintaining statistics over a data stream. This paper initiates a theoretical study of algorithms for monitoring distributed data streams over a time-based sliding window (which contains a variable number of items and possibly out-of-order items). The concern is how to minimize the communication between individual streams and the root, while allowing the root, at any time, to be able to report the global statistics of all streams within a given error bound. This paper presents communication-efficient algorithms for three classical statistics, namely, basic counting, frequent items and quantiles. The worst-case communication cost over a window is $O(\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$ bits for basic counting and $O(\frac{k}{\varepsilon}\log\frac{N}{k})$ words for the remainings, where k is the number of distributed data streams, N is the total number of items in the streams that arrive or expire in the window, and $\varepsilon < 1$ is the desired error bound. Matching and nearly matching lower bounds are also obtained.

1. Introduction

The problems studied in this paper are best illustrated by the following puzzle. John and Mary work in different laboratories and communicate by telephone only. In a forever-running experiment, John records which devices have an exceptional signal in every 10 seconds. To adjust her devices, Mary at any time needs to keep track of the number of exceptional signals generated by each device of John in the last one hour. John can call Mary every 10 seconds to report the exceptional signals, yet this requires too many calls in an hour and the total message size per hour is linear to the total number N of exceptional signals in an hour. Mary's devices actually allow some small error. Can the number of calls and message size be reduced to $o(N)$, or even poly-log N if a small error (say, 0.1%) is

1998 ACM Subject Classification: F.2.2 [Analysis of algorithms and problem complexity]: Nonnumerical algorithms and problems.

Key words and phrases: Algorithms, distributed data streams, communication efficiency, frequent items.

T.W. Lam is partially supported by the GRF Grant HKU-713909E; H.F. Ting is partially supported by the GRF Grant HKU-716307E.

SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

© H.L. Chan, T.W. Lam, L.K. Lee, and H.F. Ting
Creative Commons Attribution-NoDerivs License

allowed? It is important to note that the input is given online and Mary needs to know the answers continuously; this makes our problem different from those in other similar classical models, such as the Simultaneous Communication Complexity model [4], in which all inputs are given in advance and the parties need to compute an answer only once.

Motivation. The above problem appears in data stream applications, e.g., network monitoring or stock analysis. In the last decade, algorithms for continuous monitoring of a single massive data stream gained a lot of attention (see [1, 26] for a survey), and the main challenge has been how to represent the massive data using limited space, while allowing certain statistics (e.g., item counts, quantiles) to be computed with sufficient accuracy.

The space-accuracy tradeoff for representing a single stream has gradually been understood over the years (e.g., [2, 15, 18, 19]). Recently, motivated by large scale networks, the database community is enthusiastic about communication-efficient algorithms for continuous monitoring of multiple, distributed data streams. In such applications, we have $k \geq 1$ remote sites each monitoring a data stream, and there is a root (or coordinator) responsible for computing some global statistics. A remote site needs to maintain certain statistics itself, and has to communicate with the root often enough so that the root can compute, at any time, the statistics of the union of all data streams within a certain error. The objective is to minimize the communication. The communication aspects of data streams introduce several challenging theoretical questions such as what is the optimal communication-accuracy tradeoff for maintaining a particular statistic, and whether two-way communication is inherently more efficient than one-way communication.

Data stream models and ε-approximate queries. The data stream at each remote site is a sequence of items from a totally ordered set U. Each item is associated with an integral time-stamp recording its arrival time. Each remote site has limited space and hence it can only maintain the required statistics approximately. The statistics can be based on the whole data stream [2, 15, 18, 19] or only the recent items [3, 14, 22]. Recent items can be modeled by two types of sliding windows [5, 13]. Let W be the window size, which is a positive integer. The *count-based sliding window* includes the last W items in the data stream, while the *time-based sliding window* includes items whose time-stamps are within the last W time units. The latter assumes that zero or more items can arrive at a time. Items in a sliding window will expire and are more difficult to handle than in the whole data stream. For example, counting the frequency of a certain item in the whole stream can be done easily by maintaining a single counter, yet the same problem requires space $\Theta(\frac{1}{\varepsilon}\log^2(\varepsilon W))$ bits for a count-based sliding window even if we allow a relative error of at most ε [13, 16]. In fact, the whole data stream model can be viewed as a special case of the sliding window model with window size being infinite. Also, a count-based window is a special case of a time-based window in which exactly one item arrives at a time. This paper focuses on time-based window, and the algorithms are applicable to the other two models.

We study algorithms that enable the root to answer three types of classical ε-approximate queries, defined as follows. Let $0 < \varepsilon < 1$. For any stream σ, let $c_{j,\sigma}$ and c_σ be the count of item j and all items whose timestamps are in the current window, respectively. Denote $c_j = \sum_\sigma c_{j,\sigma}$ and $c = \sum_\sigma c_\sigma$ as the total count of item j and all items in all the data streams, respectively.

- *Basic Counting.* Return an estimate $\hat{c}$ on the total count c such that $|\hat{c} - c| \leq \varepsilon c$. (Note that this query can be generalized to count data items of a fixed subset $X \subseteq U$; the literature often refers to the special case with $U = \{0, 1\}$ and $X = \{1\}$.)

- *Frequent Items.* Given any $0 < \phi < 1$, return a set $F \subseteq U$ which includes all items j with $c_j \geq \phi c$ and possibly some items j' with $c_{j'} \geq \phi c - \varepsilon c$.
- *Quantiles.* Given any $0 < \phi < 1$, return an item whose rank is in $\big[\phi c - \varepsilon c,\ \phi c + \varepsilon c\big]$ among the c items in the current sliding window.

As in most previous works, we need to answer the following type of ε-approximate queries in order to answer queries on frequent items.

- *Approximate Counting.* Given any item j, return an estimate $\hat{c}_j$ such that $|\hat{c}_j - c_j| \leq \varepsilon c$. (Note that this query gives estimate for any item, not just the frequent items. Also, the error bound is in term of c, which may be much larger than c_j.)

We need an algorithm to determine when and how the remote sites communicate with the root so that the root can answer the queries at any time. The objective is to minimize the worst-case communication cost within a window of W time units.

Previous works. Recently, the database literature has a flurry of results on continuous monitoring of distributed data streams, e.g. [6, 8, 9, 12, 17, 20, 24, 25, 27, 28]. The algorithms studied can be classified into two types: *one-way* algorithms only allow messages sent from each remote site to the root, and *two-way* algorithms allow bi-directional communication between the root and each site. One-way algorithms are often very simple as a remote site has little information and all it can do is to update the root when its local statistics deviate significantly from those previously sent. On the other hand, most two-way algorithms are complicated and often involve non-trivial heuristics. It is commonly believed in the database community that two-way algorithms are more efficient; however, for most existing two-way algorithms, their worst-case communication costs are still waiting for rigorous mathematical analysis, and existing works often rely on experimental results when evaluating the communication cost.

The literature contains several results on the mathematical analysis of the worst-case performance of one-way algorithms. They are all for the whole data stream setting. Keralapura et al. [21] studied the thresholded-count problem, which leads to an algorithm for basic counting with communication cost $O(\frac{k}{\varepsilon}\log\frac{N}{k})$ words, where k and N are the number of streams and the number of items in these streams, respectively. Cormode et al. [9] gave an algorithm for quantiles with communication cost $O(\frac{k}{\varepsilon^2}\log\frac{N}{k})$ words per stream. They also showed how to handle frequent items via a reduction to quantiles, so the communication cost remains the same. More recently, Yi and Zhang [29] have reduced the communication cost for frequent items to $O(\frac{k}{\varepsilon}\log\frac{N}{k})$ words, and quantile to $O(\frac{k}{\varepsilon}\log^2(\frac{1}{\varepsilon})\log\frac{N}{k})$ words, using some two-way algorithms; these are the only analyses for two-way algorithms so far.

There have been attempts to devise heuristics to extend some whole-data-stream algorithms to sliding windows, yet not much has been known about their worst-case performance. For example, Cormode et al. [9] have extended their algorithms for quantiles and frequent items to sliding windows. They believed that the communication cost would only have a mild increase, but no supporting analysis has been given. The analysis of sliding-window algorithms is more difficult because the expiry of items destroys some monotonic property that is important to the analysis for whole data stream. In fact, finding sliding-window algorithms with efficient worst-case communication has been posed as an open problem in the latest work of Yi and Zhang [29].

Our results. This paper gives the first mathematical analysis of the communication cost in the sliding window model. We derive lower bounds on the worst-case communication cost of any two-way algorithm (and hence any one-way algorithm) for answering the four

	Basic Counting (**bits**)	Approximate Counting/ Frequent items (**words**)	Quantiles (**words**)
Whole data stream	$O(\frac{k}{\varepsilon}\log\frac{N}{k})$ words [21]	$O(\frac{k}{\varepsilon}\log\frac{N}{k})$ [29]	$O(\frac{k}{\varepsilon}\log^2(\frac{1}{\varepsilon})\log\frac{N}{k})$ [29]
	$\Theta(\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$ bits	$\Omega(\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$ [29, 30]	
Sliding window	$\Theta(\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$	$O(\frac{k}{\varepsilon}\log\frac{N}{k})$	$O(\frac{k}{\varepsilon^2}\log\frac{N}{k})$
		$\Omega(\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$	
Sliding window & out-of-order	$O((\frac{W}{W-\tau})\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$	$O((\frac{W}{W-\tau})\frac{k}{\varepsilon}\log\frac{N}{k})$	$O((\frac{W}{W-\tau})\frac{k}{\varepsilon^2}\log\frac{N}{k})$
	$\Omega(\max\{\frac{W}{W-\tau}, \frac{k}{\varepsilon}\log\frac{\varepsilon N}{k}\})$	$\Omega(\max\{\frac{W}{W-\tau}, \frac{k}{\varepsilon}\log\frac{\varepsilon N}{k}\})$	

Table 1: Bounds on the communication costs. Note that the bounds are stated in bits for basic counting, and in words for the other problems.

types of ε-approximate queries. These lower bounds hold even when each remote site has unlimited space to maintain the local statistics exactly. More interestingly, we analyze some common-sense algorithms that use one-way communication only and prove that their communication costs match or nearly match the corresponding lower bounds. In our algorithms, each remote site only needs to maintain some $\Theta(\varepsilon)$-approximate statistics for its local data, which actually adds more complication to the problem. These results demonstrate optimal or near optimal communication-accuracy tradeoffs for supporting these queries over the sliding window. Our work reveals that two-way algorithms could not be much better than one-way algorithms in the worst case.

Below we state the lower and upper bounds precisely. Recall that there are k remote sites and the sliding window contains W time units. We prove that within any window, the root and the remote sites need to communicate, in the worst case, $\Omega(\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$ bits for basic counting and $\Omega(\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$ words for the other three queries, where N is the total number of items arriving or expiring within that window.[1] For upper bounds, our analysis shows that basic counting requires $O(\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$ bits within any window, and approximate counting $O(\frac{k}{\varepsilon}\log\frac{N}{k})$ words. The estimates given by approximate counting are sufficient to find frequent items, hence the latter problem has the same communication cost. For quantiles, it takes $O(\frac{k}{\varepsilon^2}\log\frac{N}{k})$ words. See the second row (sliding window) of Table 1 for a summary.

As mentioned before, sliding-window algorithms can be applied to handle the special case of whole data streams in which the window size W is infinite and N is the total number of arrived items. The first row of Table 1 shows the results on whole data streams. Our work has improved the communication cost for basic counting from $O(\frac{k}{\varepsilon}\log\frac{N}{k})$ words [21] to $O(\frac{k}{\varepsilon}\log\frac{\varepsilon N}{k})$ bits. For approximate counting and frequent items, our work implies a one-way algorithm with communication cost of $O(\frac{k}{\varepsilon}\log\frac{N}{k})$ words; this matches the performance of the two-way algorithm by Yi and Zhang [29]. In their algorithm, the root regularly updates every remote site about the global count of all items. In contrast, we use the idea that

[1]Note that the number of items arriving or expiring within window $[t-W+1, t]$ is no greater than the number of items arriving within $[t-2W+1, t]$.

items with small count could be "turned off" for further updating. As a remark, our upper bound on quantiles is $O(\frac{k}{\varepsilon^2}\log\frac{N}{k})$ words which is weaker than that of [29].

Our algorithms can be readily applied to out-of-order streams [7, 10]. In an out-of-order stream, each item is associated with an integral time-stamp recording its creation time, which may be different from its arrival time. We say that the stream has *tardiness* τ if any item with time-stamp t must arrive within τ time units from t, i.e., at any time in $[t, t+\tau]$. Without loss of generality, we assume that $\tau \in \{0, 1, 2, \ldots, W-1\}$ (if an item time-stamped at t arrives after $t+W-1$, it has already expired and can be ignored). Note that for any data stream with tardiness greater than zero, the items may not be arriving in non-decreasing order of their time-stamps. Our previous discussion of data streams assumes tardiness equal to 0, and such data streams are called *in-order* data streams. The previous lower bounds for in-order streams are all valid in the out-of-order setting. In addition, we obtain lower bounds related to τ, namely, $\Omega(\frac{W}{W-\tau})$ bits for basic counting and $\Omega(\frac{W}{W-\tau})$ words for the other three problems. Regarding upper bounds, our algorithms when applied to out-of-order streams with tardiness τ will just increase the communication cost by a factor of $\frac{W}{W-\tau}$. The results are summarized in the last row of Table 1.

The idea for basic counting is relatively simple. As the root does not require an exact total count, each data stream can communicate to the root only when its local count increases or decreases by a certain ratio $\varepsilon > 0$; we call such a communication step an *up* or *down* event, respectively. To answer the total count of all streams, the root simply sums up all the individual counts it has received. It is easy to prove that this answer is within some desired error bound. If each count is over the whole stream (i.e., window size $= \infty$ and N is the total number of arrived items), the count is increasing and there is no down event. A stream would have at most $O(\log_{1+\varepsilon} N)$ up events and the communication cost is at most that many words. However, the analysis becomes non-trivial in a sliding time window. Now items can expire and down events can occur. An up event may be followed by some down events and the count is no longer increasing. The tricky part is to find a new measure of progress. We identify a "characteristic set" of each up event such that each up event must increase the size of this set by a factor of at least $1+\varepsilon$, hence bounding the number of up events to be $O(\log_{1+\varepsilon} N)$. Down events are bounded using another characteristic set. Due to space limitation, the details can only be given in the full paper.

Approximate counting of all possible items is much more complicated, which will be covered in details in the rest of this paper. Assuming in-order streams, we derive and analyze two algorithms for approximate counting in Section 2. In Section 3, we discuss frequent items, quantiles, and finally out-of-order streams. The lower bound results are relatively simple and omitted due to space limitation.

2. Approximate Counting of all items

This section presents algorithms for the streams to communicate to the root so that the root at any time can approximate the count of each item. As a warm-up, we first consider the simple algorithm in which a stream will inform the root whenever its count of an item increases or decreases by a certain fraction of its total item count. We show in Section 2.1 that within any window of W time units, each data stream σ_i $(1 \le i \le k)$ needs to send at most $O((\Delta + \frac{1}{\varepsilon})\log n_i)$ words to the root, where Δ is the number of distinct items and n_i is the number of items of σ_i that arrive or expire within the window. Then, the total communication cost within this window is $\sum_{1\le i\le k}(\Delta + \frac{1}{\varepsilon})\log n_i$, which, by

Jensen's inequality, is no greater than $(\Delta + \frac{1}{\varepsilon})k \log(\sum_{1\le i\le k} n_i)/k = (\Delta + \frac{1}{\varepsilon})k \log \frac{N}{k}$ where $N = \sum_{1\le i\le k} n_i$. We then modify the algorithm so that a stream can "turn off" items whose counts are too small, and we give a more complicated analysis to deal with the case when many such items increase their counts rapidly (Section 2.2). The communication cost is reduced to $O(\frac{k}{\varepsilon} \log \frac{N}{k})$ words, independent of Δ.

2.1. A simple algorithm

Consider any stream σ. At any time t, let $c(t)$ and $c_j(t)$ be the number of all items and item j arriving at σ in $[t - W + 1, t]$, respectively. Let $\lambda < 1/11$ be a positive constant (which will be set to $\varepsilon/11$). We maintain two λ-approximate data structures [13, 23] at σ locally, which can report estimates $\hat{c}(t)$ and $\hat{c}_j(t)$ for $c(t)$ and $c_j(t)$, respectively, such that [2]

$$(1 - \lambda/6)c(t) \le \hat{c}(t) \le (1 + \lambda/6)c(t); \quad \text{and} \quad c_j(t) - \lambda c(t) \le \hat{c}_j(t) \le c_j(t) + \lambda c(t).$$

Simple algorithm. At any time t, for any item j, let $p < t$ be the last time $\hat{c}_j(p)$ is sent to the root. The stream sends the estimate $\langle j, \hat{c}_j(t)\rangle$ to the root if the following event occurs.

- *Up:* $\hat{c}_j(t) > \hat{c}_j(p) + 9\lambda\hat{c}(t)$.
- *Down:* $\hat{c}_j(t) < \hat{c}_j(p) - 9\lambda\hat{c}(t)$.

Root's perspective. At any time t, let $r_{j,\sigma}(t)$ be the last estimate received from a stream σ for item j (at or before t). The root can estimate the total count of item j over all streams by summing all $r_{j,\sigma}(t)$ received. More precisely, for any $0 < \varepsilon < 1$, we set $\lambda = \varepsilon/11$ and let each stream use the simple algorithm. Then for each stream σ, the approximate data structures for $\hat{c}_j(t)$ and $\hat{c}(t)$ together with the simple algorithm guarantee that $c_j(t) - 11\lambda c(t) \le r_{j,\sigma}(t) \le c_j(t) + 11\lambda c(t)$. Summing $r_{j,\sigma}(t)$ over all streams would give the root an estimate of the total count of item j within an error of ε of the total count of all items.

Communication Complexity. At any time t, we denote the reference window as $[t_o, t]$, where $t_o = t - W + 1$. Let n be the number of items of σ that arrive or expire in $[t_o, t]$. Assume that there are at most Δ distinct items. We first show that a stream σ encounters $O((\frac{1}{\lambda} + \Delta) \log n)$ up events and sends $O((\frac{1}{\lambda} + \Delta) \log n)$ words within $[t_o, t]$. The analysis of down events is similar and will be detailed later. For any time $t_1 \le t_2$, it is useful to define $\sigma_{[t_1,t_2]}$ (resp. $\sigma_{j,[t_1,t_2]}$) as the multi-set of all items (resp. item j only) arriving at σ within $[t_1, t_2]$, and $|\sigma_{[t_1,t_2]}|$ as the size of this multi-set.

Consider an up event U_j of some item j that occurs at time $v \in [t_o, t]$. Define the *previous event* of U_j to be the latest event (up or down) of item j that occurs at time $p < v$. We call p the *previous-event time* of U_j. The number of up events with previous-event time before t_o is at most Δ. To upper bound the number of up events with previous-event time $p \ge t_o$ is, however, non-trivial; below we call such an up event a *follow-up* (event). Intuitively, a follow-up can be triggered by frequent arrivals of an item, or mainly the relative decrease of the total count. This motivates us to classify follow-ups into two types and analyze them differently. A follow-up U_j is said to be *absolute* if $c(p) \le \frac{6}{5}c(v)$, and *relative* otherwise. Define *Recent-items*(U_j) to be the multi-set of item j's that arrive after the previous event of U_j, i.e., *Recent-items*$(U_j) = \sigma_{j,[p+1,v]}$.

[2] The constant 6 in the inequality is arbitrary. It can be replaced with any number provided that other constants in the algorithm and analysis (e.g., the constant 9 in definition of up events) are adjusted accordingly.

Absolute follow-ups. To obtain a tight bound of absolute follow-ups, we need a characteristic-set argument that can consider the growth of different items together. Let $t_1, t_2, ..., t_k$ be the times in $[t_o, t]$ when some absolute follow-ups (of one or more items) occur. Let x_i be the number of items having an absolute follow-up at t_i. Note that for all i, $x_i \leq \min\{1/(7\lambda), \Delta\}$,[3] and $\sum_{i=1}^{k} x_i$ is the number of absolute follow-ups in $[t_o, t]$. We define the characteristic set S_i at each t_i as follows:

S_i = the union of *Recent-items*(U_j) over all absolute follow-ups U_j occurring at $t_1, t_2, \ldots, t_i$.

Recall that n is the number of items of σ that arrive or expire in $[t - W + 1, t]$.

Lemma 2.1. **(i)** *For any* $2 \leq i \leq k$, $|S_i| > (1 + 6x_i\lambda)|S_{i-1}|$. **(ii)** *There are* $\sum_{i=1}^{k} x_i = O(\frac{1}{\lambda}\log n)$ *absolute follow-ups within* $[t_o, t]$.

Proof. For (i), consider an absolute follow-up U_j of an item j, occurring at time t_i with previous-event time p_i. Note that the increase in the count of item j from p_i to t_i must be due to the recent items. We have

$$\begin{aligned}
|\textit{Recent-items}(U_j)| &\geq c_j(t_i) - c_j(p_i) \\
&\geq \hat{c}_j(t_i) - \hat{c}_j(p_i) - \lambda c(t_i) - \lambda c(p_i) && \text{(by } \sigma\text{'s local data structures)} \\
&> 9\lambda\hat{c}(t_i) - \lambda c(t_i) - \lambda c(p_i) && \text{(definition of an up event)} \\
&\geq (9\lambda(1 - \tfrac{\lambda}{6}) - \lambda - \tfrac{6}{5}\lambda)c(t_i) \geq 6\lambda c(t_i) && (U_j \text{ is absolute})
\end{aligned}$$

There are x_i absolute follow-ups at t_i, so $|S_i| > |S_{i-1}| + x_i\,(6\lambda c(t_i))$. Since $S_i \subseteq \sigma_{[t_o,t_i]}$, $c(t_i) \geq |S_i| \geq |S_{i-1}|$. Therefore, we have $|S_i| > |S_{i-1}| + 6x_i\lambda|S_i| \geq (1 + 6x_i\lambda)|S_{i-1}|$.

For (ii), we note that $n \geq |S_k| > \prod_{i=2}^{k}(1 + 6x_i\lambda)|S_1|$, and $|S_1| \geq 1$. Thus, $\prod_{i=2}^{k}(1 + 6x_i\lambda) < n$, or equivalently, $\ln n > \sum_{i=2}^{k}\ln(1 + 6x_i\lambda)$. The latter is at least $\sum_{i=2}^{k}\frac{6x_i\lambda}{1+6x_i\lambda} \geq \lambda\sum_{i=2}^{k} x_i$. The last inequality follows from that $x_i \leq 1/(7\lambda)$ for all i. Thus, $\sum_{i=1}^{k} x_i \leq x_1 + \frac{1}{\lambda}\ln n = O(\frac{1}{\lambda}\log n)$. ∎

Relative follow-ups. A relative follow-up occurs only when a lot of items expire, and relative follow-ups of the same item cannot occur too frequently. Below we define $O(\log n)$ time intervals and argue that no item can have two relative follow-ups within an interval. For an item with time-stamp t_1, we define the *first expiry time* to be $t_1 + W$. At any time u in $[t_o, t]$, define H_u to be the set of all items whose first expiry time is within $[u + 1, t]$, i.e., $H_u = \sigma_{[u-W+1,t_o-1]}$. $|H_u|$ is non-increasing as u increases. Consider the times $t_o = u_0 < u_1 < u_2 < \cdots < u_\ell \leq t$ such that for $i \geq 1$, u_i is the first time such that $|H_{u_i}| < \frac{5}{6}|H_{u_{i-1}}|$. For convenience, let $u_{\ell+1} = t + 1$. Note that $|H_{u_0}| \leq n$ and $\ell = O(\log n)$.

Lemma 2.2. **(i)** *Every item* j *has at most one relative follow-up* U_j *within each interval* $[u_i, u_{i+1} - 1]$. **(ii)** *There are at most* $O(\Delta\log n)$ *relative follow-ups within* $[t_o, t]$.

Proof. For (i), assume U_j occurs at time v in $[u_i, u_{i+1} - 1]$, and its previous event occurs at time p. By definition, $c(p) > \frac{6}{5}c(v)$. Thus,

$$|H_p| - |H_v| = |\sigma_{[p-W+1,v-W]}| \geq c(p) - c(v) > \tfrac{1}{5}c(v) \geq \tfrac{1}{5}|\sigma_{[v-W+1,t_o-1]}| = \tfrac{1}{5}|H_v| \ ,$$

and $|H_v| < \frac{5}{6}|H_p|$. Since $v < u_{i+1}$ and $|H_v| \geq \frac{5}{6}|H_{u_i}|$, we have $|H_p| > |H_{u_i}|$ and $p < u_i$. For (ii), there are Δ distinct items, so there are at most Δ relative follow-ups within each interval $[u_i, u_{i+1} - 1]$, and at most $O(\Delta\log n)$ relative follow-ups within $[t_o, t]$. ∎

[3] If an up event of an item j occurs at time t_i, then $c_j(t_i) \geq \hat{c}_j(t_i) - \lambda c(t_i) > 9\lambda\hat{c}(t_i) - \lambda c(t_i) \geq 7\lambda c(t_i)$. Thus the number of up events at time t_i is at most $c(t_i)/(7\lambda c(t_i)) = 1/(7\lambda)$.

Down events. The analysis is symmetric to that of up events. The only non-trivial thing is the definition of the characteristic set for bounding the absolute follow-downs D_j, which is defined in an opposite sense: Assume D_j occurs at time v and its previous event occurs at $p \geq t_o$. D_j is said to be *absolute* if $c(p) \leq \frac{6}{5}c(v)$. Let *Expire*$(D_j)$ be the multi-set of item j's whose first expiry time is within $[p+1, v]$. I.e., *Expire*$(D_j) = \sigma_{j,[p-W+1,v-W]}$.

It is perhaps a bit tricky that instead of defining the characteristic set of absolute follow-downs at the time they occur, we consider the times of the corresponding *previous events* of these follow-downs. Let $p_1, p_2, ..., p_k$ be the times in $[t_o, t]$ such that there is at least one event E_j (up or down) at p_i which is the previous event of an absolute follow-down D_j occurring after p_i. Let y_i be the number of such previous events at p_i, and let $AD(p_i)$ be the set of corresponding absolute follow-downs. Note that y_i (unlike x_i) only admits a trivial upper bound of Δ. We define the characteristic set T_i for each p_i as follows:

$$T_i = \text{the union of } \mathit{Expire}(D_j) \text{ over all } D_j \in AD(p_i), AD(p_{i+1}), \ldots, AD(p_k).$$

Similar to Lemma 2.1, we can show that $|T_i| > (1 + 5y_i\lambda)|T_{i+1}|$. Owing to a weaker bound of individual y_i, the number of absolute follow-downs, which equals $\sum_{i=1}^{k} y_i$, is shown to be $O((\frac{1}{\lambda} + \Delta)\log n)$.

Combining the analyses on up and down events, and let $\lambda = \varepsilon/11$, we have the following.

Theorem 2.3. *The simple algorithm sends at most* $O((\frac{1}{\varepsilon} + \Delta)\log n)$ *words to the root during window* $[t - W + 1, t]$.

2.2. The full algorithm

In this section, we extend the previous algorithm and give a new characteristic-set analysis that is based on future events (instead of the past events) to show that each stream's communication cost per window can be reduced to $O(\frac{1}{\varepsilon}\log n)$ words. Then, by Jensen's inequality again, we conclude that the total communication cost per window is $O(\frac{k}{\varepsilon}\log\frac{N}{k})$. Intuitively, when the estimate $\hat{c}_j(t)$ of an item j is too small, say, less than $3\lambda\hat{c}(t)$, the algorithm treats this estimate as 0 and set the off_j flag of j to be true. This restricts the number of items with a positive estimate to $O(\frac{1}{\lambda})$. Initially, the off_j flag is true for all items j. Given $0 < \lambda < \varepsilon/11$, the stream communicates with the root as follows.

Algorithm AC. At any time t, for any item j, let $p < t$ be the time the last estimate of j, i.e., $\hat{c}_j(p)$, is sent to the root. The stream sends the estimate of j to the root if the following event occurs.

- *Up:* If $\hat{c}_j(t) > \hat{c}_j(p) + 9\lambda\hat{c}(t)$, send $\langle j, \hat{c}_j(t)\rangle$ and set $\mathit{off}_j = \mathit{false}$.
- *Off:* If $\mathit{off}_j = \mathit{false}$ and $\hat{c}_j(t) < 3\lambda\hat{c}(t)$, reset $\hat{c}_j(t)$ to 0, send $\langle j, \hat{c}_j(t)\rangle$ and set $\mathit{off}_j = \mathit{true}$.
- *Down:* If $\mathit{off}_j = \mathit{false}$ and $\hat{c}_j(t) < \hat{c}_j(p) - 9\lambda\hat{c}(t)$, send $\langle j, \hat{c}_j(t)\rangle$.

It is straightforward to check that the root can answer the approximate counting query for any item. We analyze the communication complexity of different events as follows.

Fact 1. At any time v, the number of items j with $\mathit{off}_j = \mathit{false}$ is at most $\frac{1}{\lambda}$.[4]

[4]For any item j, if $\mathit{off}_j = \mathit{false}$, then $\hat{c}_j(v) \geq 3\lambda\hat{c}(v)$ and $c_j(v) \geq \hat{c}_j(v) - \lambda c(v) \geq (3\lambda(1-\lambda) - \lambda)c(v) \geq \lambda c(v)$. Thus the number of items j with $\mathit{off}_j = \mathit{false}$ is at most $c(v)/\lambda c(v) = \frac{1}{\lambda}$.

Off events. Recall that we are considering the window $[t_o, t]$, and n is the number of items arriving or expiring within $[t_o, t]$. By Fact 1, just before t_o, there are at most $\frac{1}{\lambda}$ items with $\mathit{off}_j = \mathit{false}$. Within $[t_o, t]$, only an up event can set the *off* flag to false. Thus the number of off events within $[t_o, t]$ is bounded by $\frac{1}{\lambda}$ plus the number of up events.

Up and Down events. The assumption of Δ gives a trivial bound on those events involving items with very small counts and in particular, those up events immediately following the off events. Such up events are called *poor-up* events or simply *poor-ups*. Using the *off* flag, we can easily adapt the analysis of the simple algorithm to bound all the down and up events of the full algorithm, but except the poor-ups. The following simple observations, derived from Fact 1, allow us to replace Δ with $1/\lambda$ in the previous analysis to obtain a tighter upper bound of $O(\frac{1}{\lambda}\log n)$. Let v be any time in $[t_o, t]$.

- There are at most $1/\lambda$ items whose first event after v is a down event.
- There are at most $1/\lambda$ non-poor-up events after v whose previous event is before v.

It remains to analyze the poor-ups. Consider a poor-up U_j at time v in $[t_o, t]$. By definition, $\mathit{off}_j = \mathit{false}$ at time v. The trick of analyzing U_j's is to consider when the corresponding items will be "off" again instead of what items constitute the up events. Then a characteristic set argument can be formulated easily. Specifically, we first observe that, by Fact 1, there are at most $\frac{1}{\lambda}$ poor-ups whose *off* flags remain false up to time t. Then it remains to consider those U_j whose *off* flags will be set to true at some time $d \le t$. Below we refer to d as the *first off time* of U_j.

Poor-up with early off. Consider a poor-up U_j that occurs at time v in $[t_o, t]$ and has its first off time at d in $[v+1, t]$. Let *F-Expire*(U_j) be all the item j whose first expiry time is within $[v+1, d]$. I.e., *F-Expire*$(U_j) = \sigma_{j,[v+1-W,d-W]}$. As an early off can be due to the expiry of many copies of item j or the arrival of a lot of items, it is natural to divide the poor-ups into two types: with an *absolute* off if $c(d) \le \frac{6}{5}c(v)$, and *relative* off otherwise. For the case with absolute off, we consider the distinct times $t_1, t_2, \ldots, t_k$ in $[t_o, t]$ when such poor-ups occur. Let x_i be the number of such poor-ups at time t_i. Note that $x_i \le 1/(7\lambda)$. For each time t_i, we define the characteristic set

$$F_i = \text{the union of } \textit{F-Expire}(U_j) \text{ over all } U_j \text{ occurring at } t_i, t_{i+1}, \ldots, t_k.$$

Lemma 2.4. **(i)** *For any* $1 \le i \le k-1$, $|F_i| > (1 + x_i\lambda)|F_{i+1}|$. **(ii)** *Within* $[t_o, t]$, *there are* $\sum_{i=1}^{k} x_i = O(\frac{1}{\lambda}\log n)$ *poor-ups each with an absolute off.*

Proof. For (i), consider an item j and a poor-up U_j with an absolute off that occurs at time t_i and has its first off at time d_i. The decrease in c_j must be due to expiry of item j.

$$\begin{aligned}
|\textit{F-Expire}(U_j)| &\ge c_j(t_i) - c_j(d_i) \ge \hat{c}_j(t_i) - \hat{c}_j(d_i) - \lambda c(t_i) - \lambda c(d_i) \\
&> 9\lambda\hat{c}(t_i) - 3\lambda\hat{c}(d_i) - \lambda c(t_i) - \lambda c(d_i) && \text{(definition of up and off)} \\
&\ge (9\lambda(1 - \tfrac{\lambda}{6}) - \lambda)c(t_i) - (3\lambda(1 + \tfrac{\lambda}{6}) + \lambda)c(d_i) \ge 7\lambda c(t_i) - 5\lambda c(d_i) \\
&\ge (7 - 5(\tfrac{6}{5}))\lambda c(t_i) = \lambda c(t_i) && \text{(definition of absolute off)}
\end{aligned}$$

Thus, $|F_i| > |F_{i+1}| + x_i\,(\lambda c(t_i))$. Since $F_i \subseteq \sigma_{[t_i-W+1,t-W]}$, $|F_i| \le c(t_i)$. Therefore, $|F_i| > |F_{i+1}| + x_i\lambda|F_i| > (1 + x_i\lambda)|F_{i+1}|$. By (i), we can prove (ii) similarly to Lemma 2.1 (ii). ∎

Analyzing poor-ups with a relative off is again based on an isolating argument. We divide $[t_o, t]$ into $O(\log n)$ intervals according to how fast the total item count starting from t_o grow; specifically, we want two consecutive time boundaries u_{i-1} and u_i to satisfy

$|\sigma_{[t_o,u_i]}| > \frac{6}{5}|\sigma_{[t_o,u_{i-1}]}|$. Then we show that for any poor-up within $[u_{i-1}, u_i - 1]$, its relative off, if exists, occurs at or after u_i. Thus there are at most $\frac{1}{\lambda}$ such poor-ups within each interval and a total of $O(\frac{1}{\lambda}\log n)$ within $[t_o, t]$.

Lemma 2.5. **(i)** *Consider a poor-up U_j with a relative off. Suppose it occurs at time v in $[t_o, t]$, and its first off time is at d in $[v+1, t]$. Then $|\sigma_{[t_o,d]}| > \frac{6}{5}|\sigma_{[t_o,v]}|$.* **(ii)** *Within $[t_o, t]$, there are at most $O(\frac{1}{\lambda}\log n)$ poor-ups each with a relative off.*

Proof. For (i), by the definition of a relative off, $c(d) > \frac{6}{5}c(v)$. Thus, $|\sigma_{[t_o,d]}| - |\sigma_{[t_o,v]}| = |\sigma_{[v+1,d]}| \geq c(d) - c(v) > \frac{1}{6}c(d) \geq \frac{1}{6}|\sigma_{[t_o,d]}|$. This implies $|\sigma_{[t_o,d]}| > \frac{6}{5}|\sigma_{[t_o,v]}|$.

For (ii), consider the times $t_o = u_0 < u_1 < u_2 < \cdots < u_\ell \leq t$ such that for $i \geq 1$, u_i is the first time such that $|\sigma_{[t_o,u_i]}| > \frac{6}{5}|\sigma_{[t_o,u_{i-1}]}|$. For convenience, let $u_{\ell+1} = t+1$. Note that $|\sigma_{[t_o,t]}| \leq n$ and $\ell = O(\log n)$. Furthermore, for any time $v \in [u_{i-1}, u_i - 1]$, $|\sigma_{[t_o,v]}| \leq \frac{6}{5}|\sigma_{[t_o,u_{i-1}]}|$. Therefore, by (i), for any poor-up of an item j within $[u_{i-1}, u_i - 1]$, its relative off, if exists, occurs at or after u_i, which implies at time $u_i - 1$, $c_j(u_i - 1) \geq \lambda c(u_i - 1)$. Then within each interval $[u_{i-1}, u_i - 1]$, the number of such j as well as the number of poor-ups with a relative off are at most $\frac{1}{\lambda}$. Within $[t_o, t]$, there are $\ell = O(\log n)$ intervals and hence $O(\frac{1}{\lambda}\log n)$ poor-ups each with a relative off. ∎

Theorem 2.6. *For approximate counting, each individual stream can use the algorithm AC with $\lambda = \varepsilon/11$ and it sends at most $O(\frac{1}{\varepsilon}\log n)$ words to the root within a window.*

Memory usage of each remote site. Recall that we use two λ-approximate data structures [13, 23] for the total item count and individual item counts, which respectively require $O(\frac{1}{\lambda}\log^2(\lambda n))$ bits and $O(\frac{1}{\lambda})$ words. Note that $O(\frac{1}{\lambda}\log^2(\lambda n))$ bits is equivalent to $O(\frac{1}{\lambda}\log(\lambda n))$ words. Furthermore, at any time, we only need to keep track of the last estimate sent to the root of all item j with $\mathit{off}_j = \mathit{false}$, which by Fact 1, requires $O(\frac{1}{\lambda})$ words. By setting $\lambda = \varepsilon/11$ (see Theorem 2.6), the total memory usage of a remote site is $O(\frac{1}{\lambda}\log(\lambda n)) = O(\frac{1}{\varepsilon}\log(\varepsilon n))$ words.

3. Extensions

We extend the previous techniques to solve the problems of frequent items and quantiles and handle out-of-order streams. Below BC refers to our algorithm for basic counting.

Frequent items. Using the algorithms BC and AC, the root can answer the ε-approximate frequent items as follows. Each stream σ communicates with the root using BC with error parameter $\varepsilon/24$ and AC with error parameter $11\varepsilon/24$. At any time t, let $r_\sigma(t)$ and $r_{j,\sigma}(t)$ be the latest estimates of the numbers of all items and item j, respectively, received by the root from σ. To answer a query of frequent items with threshold $\phi \in (0, 1]$ at time t, the root can return all items j with $\sum_\sigma r_{j,\sigma}(t) \geq (\phi - \frac{\varepsilon}{2})\sum_\sigma r_\sigma(t)$ as the set of frequent items.

To see the correctness, let $c_\sigma(t)$ and $c_{j,\sigma}(t)$ be the number of all items and item j in σ at time t, respectively. Algorithm BC guarantees $|r_\sigma(t) - c_\sigma(t)| \leq \frac{\varepsilon}{24}c_\sigma(t)$, and algorithm AC guarantees $|r_{j,\sigma}(t) - c_{j,\sigma}(t)| \leq \frac{11\varepsilon}{24}c_\sigma(t)$. Therefore, if an item j is returned by the root, then $\sum_\sigma c_{j,\sigma}(t) \geq \sum_\sigma r_{j,\sigma}(t) - \frac{11\varepsilon}{24}\sum_\sigma c_\sigma(t) \geq (\phi - \frac{\varepsilon}{2})\sum_\sigma r_\sigma(t) - \frac{11\varepsilon}{24}\sum_\sigma c_\sigma(t) \geq (\phi - \frac{\varepsilon}{2})(1 - \frac{\varepsilon}{24})\sum_\sigma c_\sigma(t) - \frac{11\varepsilon}{24}\sum_\sigma c_\sigma(t) \geq (\phi - \frac{\varepsilon}{2} - \phi\frac{\varepsilon}{24} - \frac{11\varepsilon}{24})\sum_\sigma c_\sigma(t)$ where the second inequality comes from the definition of the algorithm. The last term above is at least

$(\phi - \varepsilon)\sum_\sigma c_\sigma(t)$, so j is a frequent item. If an item j is not returned by the root, then $\sum_\sigma r_{j,\sigma}(t) < (\phi - \frac{\varepsilon}{2})\sum_\sigma r_\sigma(t)$ and we can show similarly that $\sum_\sigma c_{j,\sigma}(t) < \phi \sum_\sigma c_\sigma(t)$.

Quantiles. We give an algorithm for ε-approximate quantiles queries. Let $\lambda = \varepsilon/20$. For each stream, we keep track of the λ-approximate ϕ-quantiles for $\phi = 5\lambda, 10\lambda, 15\lambda, \ldots, 1$. We update the root for all these ϕ-quantiles when one of the following two events occurs: (i) for any k, the value of the $(5k\lambda)$-quantile is larger than the value of the $(5(k+1)\lambda)$-quantile last reported to the root, or (ii) for any k, the value of the $(5k\lambda)$-quantile is smaller than the value of the $(5(k-1)\lambda)$-quantile last reported to the root. The stream also communicates with the root using BC with error parameter λ. In the root's perspective, at any query time t, let $\phi \in (0, 1]$ be the query given and let $r_\sigma(t)$ be the last estimate sent by σ for the number of all items. The root sorts the quantiles last reported by all streams and for each stream σ, gives a weight of $5\lambda r_\sigma(t)$ to each quantile of σ. Then the root returns the smallest item j in the sorted sequence such that the sum of weights for all items no greater than j is at least $\lceil \phi \sum_\sigma r_\sigma(t) \rceil$. Careful counting can show that j is an ε-approximate ϕ-quantile. To bound the communication cost, let n be the number of items of σ arriving or expiring during the window $[t - W + 1, t]$. We observe that when an event occurs, many items have either arrived or expired after the previous event. Using similar analysis as before, we can show that within a window, there are at most $O(\frac{1}{\varepsilon}\log n)$ such events and thus each stream sends $O(\frac{1}{\varepsilon^2}\log n)$ words. By Jensen's inequality again, our algorithm's total communication cost per window is $O(\frac{k}{\varepsilon^2}\log\frac{N}{k})$ where N is the number of items of the k streams that arrive or expire within the window. Note that the lower bound of $O(\frac{1}{\varepsilon}\log(\varepsilon n))$ words for approximate frequent items carries to approximate quantiles, as we can answer approximate frequent items using approximate quantiles as follows. The root poses ε-approximate ϕ-quantile queries for $\phi = \varepsilon, 2\varepsilon, \ldots, 1$. Given the threshold ϕ' for frequent items, the root returns all items that repeatedly occur as $\frac{\phi'}{\varepsilon} - 2$ (or more) consecutive quantiles, and these items are (4ε)-approximate frequent items.

Out-of-order streams. All our algorithms can be extended to out-of-order stream with a communication cost increased by a factor of $\frac{W}{W-\tau}$, as follows. Each stream uses the data structures for out-of-order streams (e.g., [7, 10]) to maintain the local estimates. Then each stream uses our communication algorithms for in-order streams. It is obvious the root can answer the corresponding queries. For the communication cost, consider any time interval $P = [t-(W-\tau)+1, t]$ of size $W-\tau$. Items arriving in P must have time-stamps in $[t - W + 1, t]$. Using the same arguments as before, we can show the same communication cost of each algorithm, but only for a window of size $W - \tau$ instead of W. Equivalently, in any window of size W, the communication cost is increased by a factor of $O(\frac{W}{W-\tau})$.

References

[1] C. Aggarwal. *Data streams: models and algorithms.* Springer, 2006.

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

[3] A. Arasu and G. Manku. Approximate counts and quantiles over sliding windows. In *Proc. PODS*, pages 286–296, 2004.

[4] L. Babai, A. Gal, P. Kimmel, and S. Lokam. Communication compleixty of simultaneous messages. *SIAM Journal on Computing*, 33(1):137–166, 2004.

[5] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proc. SODA*, pages 633–634, 2002.

[6] B. Babcock and C. Olston. Distributed top-k monitoring. In *Proc. SIGMOD*, pages 28–39, 2003.
[7] C. Busch and S. Tirthapua. A deterministic algorithm for summarizing asynchronous streams over a sliding window. In *STACS*, 2007.
[8] G. Cormode and M. Garofalakis. Sketching streams through the net: distributed approximate query tracking. In *Proc. VLDB*, pages 13–24, 2005.
[9] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In *Proc. SIGMOD*, 25–36, 2005.
[10] G. Cormode, F. Korn, and S. Tirthapura. Time-decaying aggregates in out-of-order streams. In *Proc. PODS*, pages 89–98, 2008.
[11] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. In *Proc. SODA*, pages 1076–1085, 2008.
[12] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. Distributed set-expression cardinality estimation. In *Proc. VLDB*, pages 312–323, 2004.
[13] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002.
[14] M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. In *Proc. ESA*, pages 323–334, 2002.
[15] E. Demaine, A. Lopez-Ortiz, and J. Munro. Frequency estimation of internet packet streams with limited space. In *Proc. ESA*, pages 348–360, 2002.
[16] P. Gibbons and S. Tirthapura. Distributed streams algorithms for sliding windows. In *Proc. SPAA*, pages 63–72, 2002.
[17] M. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *Proc. PODS*, pages 275–285, 2004.
[18] S. Guha, N. Koudas, and K. Shim. Data-streams and histograms. In *Proc. STOC*, pages 471–475, 2001.
[19] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proc. FOCS*, pages 148–155, 2000.
[20] N. Jain, P. Yalagandula, M. Dahlin, and Y. Zhang. Insight: A distributed monitoring system for tracking continuous queries. In *Proc. SOSP*, pages 1–7, 2005.
[21] R. Keralapura, G. Cormode, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *Proc. SIGMOD*, pages 289–300, 2006.
[22] L. K. Lee and H. F. Ting. Maintaining significant stream statistics over sliding windows. In *Proc. SODA*, pages 724–732, 2006.
[23] L. K. Lee and H. F. Ting. A simpler and more efficient deterministic scheme for finding frequent items over sliding windows. In *Proc. PODS*, pages 290–297, 2006.
[24] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *Proc. ICDE*, pages 767–778, 2005.
[25] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In *Proc. SIGMOD*, pages 635–646, 2006.
[26] S. Muthukrishnan. *Data streams: algorithms and applications*. Now Publisher Inc., 2005.
[27] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proc. SIGMOD*, pages 563–574, 2003.
[28] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM TODS*, 32(4), 2007.
[29] K. Yi and Q. Zhang. Optimal tracking of distributed heavy hitters and quantiles. In *Proc. PODS*, pages 167–174, 2009.
[30] K. Yi and Q. Zhang. Private communication.

This work is licensed under the Creative Commons Attribution-NoDerivs License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/3.0/.