



# Continuous Monitoring of Distributed Data Streams over a Time-based Sliding Window

## Motivation

Suppose a set of routers are monitoring a network. To allow an early detection of a Distributed Denial-of-service (DDoS) attack, we need to answer *at any time*:

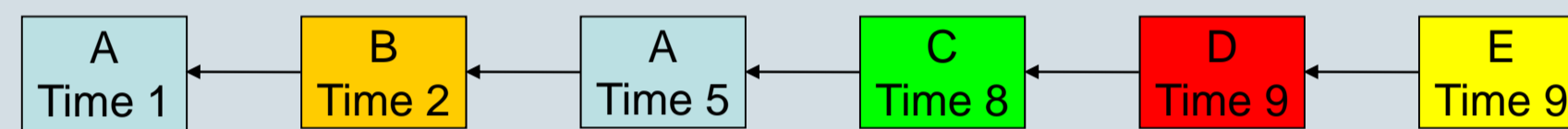
In the IP packets received by all the routers over the last hour, does there exist any frequent destination address?

The problem is to *minimize the communication overhead* to maintain such statistics.

## Distributed Data Stream Model

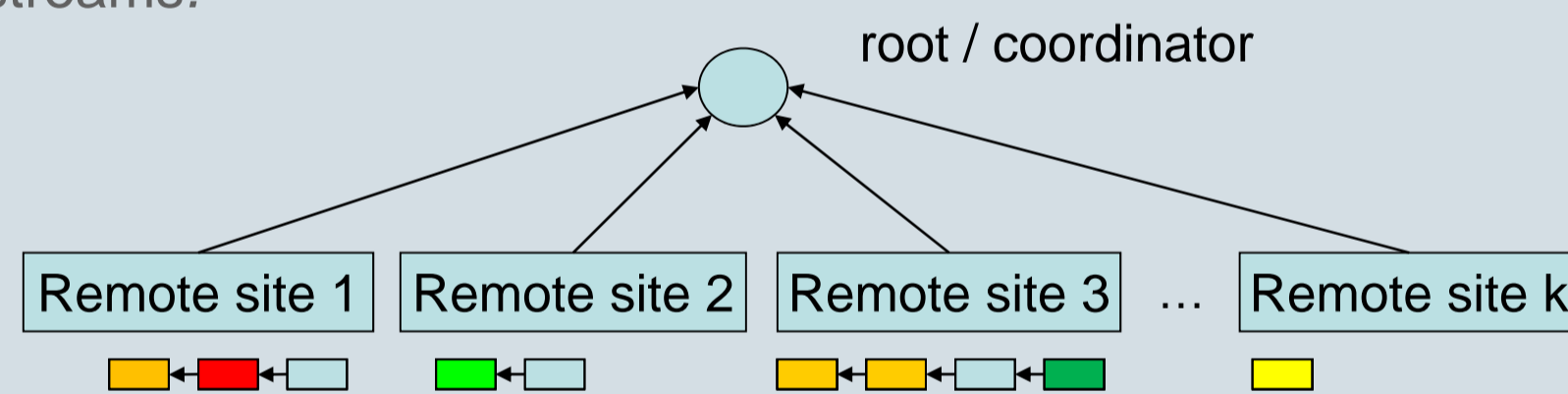
We have  $k \geq 1$  remote sites and a single root (or coordinator) distant apart.

- Each remote site is monitoring a *stream of items*, where each item contains an item label and a timestamp.



As the stream contains a massive volume of data items, the remote site cannot store the whole stream for processing, and hence each remote site can only maintain an *approximation of some stream statistics* on its own stream.

- The root is required to compute the (approximate) *global statistics* on the union of the  $k$  streams.



- Only communication between the root and each remote site is allowed; remote sites cannot communicate with each other. This restriction is practical; e.g., in a sensor network, the sensors are cheap devices with limited processing power and memory, and they cannot communicate with each other.

Algorithms in this model are *communication protocols* for the root and remote sites. They can be classified into two types:

- Two-way algorithms:** bi-directional communication between the root and remote sites are allowed.
- One-way algorithms:** only the remote sites are allowed to communicate with the root and the root cannot send message to any remote site.

One-way algorithms are usually simple and thus easy to implement, as each remote site has only local information about its own stream; the best the remote site can do is to update the root if its local statistics deviate too much from the one it previously sent to the root.

Two-way communication allows more sophisticated algorithm. Thus, *it is believed that two-way algorithms are more communication-efficient than one-way algorithms.*

## Stream Statistics

We study algorithms that enable the root to answer the following classical  $\epsilon$ -approximate queries, where  $0 < \epsilon \leq 1$  is a user-specified error bound. Let  $c$  be the total count of all items in the stream.

**Basic Counting:** Estimate the total count  $c$  with absolute error  $\epsilon c$ .

**Frequent Items:** Let  $0 < \phi \leq 1$  be a user-specified threshold. The query asks for a *set of items*, which contains

- all frequent items appearing at least  $\phi c$  times.
- some items appearing at least  $(\phi - \epsilon) c$  times.

It is well-known that to return frequent items, it suffices to answer **Approximate Counting:** Estimate the count of any item with absolute error  $\epsilon c$ .

**Quantiles:** Given any  $0 < \phi \leq 1$ , in the sorted order of all items in the stream, return an item with rank in  $[(\phi - \epsilon) c, (\phi + \epsilon) c]$ .

Stream statistics can be computed over

- whole data stream
- a **sliding window** of recent items: all items with timestamps in the *most recent*  $W$  time units, where  $W$  is the window size.

Sliding window is more difficult to handle as items will expire.

## Previous and Our Results

All previous work focuses on the *whole data stream* setting. Let  $N$  be the number of items in all the  $k$  streams.

- Basic Counting:  $O(\frac{k}{\epsilon} \log \frac{N}{k})$  words (*one-way*) [2]
- Frequent Items / Approximate Counting:  $O(\frac{k}{\epsilon^2} \log \frac{N}{k})$  words (*one-way*) [1];  $O(\frac{k}{\epsilon} \log \frac{N}{k})$  words (*two-way*) [4]
- Quantiles:  $O(\frac{k}{\epsilon^2} \log \frac{N}{k})$  words (*one-way*) [1];  $O(\frac{k}{\epsilon} \log^2(\frac{1}{\epsilon}) \log \frac{N}{k})$  words (*two-way*) [4]

Our results [3] extend the above study to the *sliding window* setting. Let  $N$  be the number of items in all the  $k$  streams that *arrive or expire* within the current sliding window of  $W$  time units. We show *upper bounds of one-way algorithms* and *lower bounds for any two-way algorithms.*

- Basic Counting:  $\Theta(\frac{k}{\epsilon} \log \frac{\epsilon N}{k})$  bits (this result also improves that in [2])
- Frequent Items / Approximate Counting:  $O(\frac{k}{\epsilon} \log \frac{N}{k})$  words  $\Omega(\frac{k}{\epsilon} \log \frac{\epsilon N}{k})$  words
- Quantiles:  $O(\frac{k}{\epsilon^2} \log \frac{N}{k})$  words  $\Omega(\frac{k}{\epsilon} \log \frac{\epsilon N}{k})$  words

Our upper bounds match or nearly match the lower bounds, which reveals that *for these statistics, two-way algorithms could not be much better than one-way algorithms in the worst case.*

## Basic Counting

**Algorithm.** Let  $\lambda = \epsilon/9$ .

Each remote site: keeps an  $\lambda$ -approximate local count. Let  $c_{cur}$  and  $c_{old}$  be current estimate and the previously sent estimate. At any time, it updates the root about  $c_{cur}$  if the following event occurs

- Up event:  $c_{cur} - c_{old} > 4\lambda c_{old}$
- Down event:  $c_{old} - c_{cur} > 4\lambda c_{old}$

The root: upon a query, returns the sum of all the  $k$  estimates received.

**Analysis techniques.** In a remote site, let  $n$  be the number of items arriving or expiring in the current sliding window  $[t_1, t_2]$ . If items do not expire, only Up's occur. When an Up occurs,  $n$  increases by a fraction of  $(1 + \Theta(\epsilon))$ , so no. of Up's is  $O(\log_{1+\epsilon} \epsilon n) = O(\frac{1}{\epsilon} \log \epsilon n)$ . Expiry of items destroys the monotonic property of  $n$ . Our idea is to define a *characteristic set* of items as a new measure of progress.

- Up event: The set is the items arriving from  $t_1$  up to the time the Up occurs. When an Up occurs, the size of this set increases by a fraction of  $(1 + \Theta(\epsilon))$ , and thus the no. of Up's is  $O(\frac{1}{\epsilon} \log \epsilon n)$ .
- Down event: The set is the items expiring from  $t_1$  up to the time the Down occurs. Similarly, the no. of Down's is  $O(\frac{1}{\epsilon} \log \epsilon n)$ .

The total no. of events is  $O(\frac{k}{\epsilon} \log \frac{\epsilon N}{k})$ . To reduce message size, we *restrict the estimates to a predefined set*, giving the required upper bound.

## Approximate Counting

**Algorithm.** Let  $\lambda = \epsilon/11$ .

Each remote site: keeps an  $\lambda$ -approximate item count  $c_{cur}(j)$  for each item  $j$ , and an  $\lambda/6$ -approximate total count  $c_{cur}$ . At any time, for each item  $j$ , it updates  $c_{cur}(j)$  if the following event occurs

- Up event:  $c_{cur}(j) - c_{old}(j) > 9\lambda c_{cur}$
- Down event:  $c_{old}(j) - c_{cur}(j) > 9\lambda c_{cur}$

The root: upon a query on item  $j$ , returns the total estimate of  $j$  received.

**Analysis techniques.** Up's can be caused by a *big increase of item  $j$* , or a *small increase of  $j$  but significant drop of  $c_{cur}$* , which requires different characteristic set analysis. Furthermore, the communication cost would depend on the no. of possible items. We observe that if  $c_{cur}(j) < 3\lambda c_{cur}$ , we can treat  $c_{cur}(j) = 0$  and stop updating  $j$  until  $c_{cur}(j) \geq 3\lambda c_{cur}$ . This keeps  $O(\frac{1}{\epsilon})$  'active' items at any time, leading to the required upper bound.

## References

- Cormode, Garofalakis, Muthukrishnan, Rastogi. *Holistic aggregates in a networked world: distributed tracking of approximate quantiles.* SIGMOD 2005.
- Keralapura, Cormode, Ramamirtham. *Communication-efficient distributed monitoring of thresholded counts.* SIGMOD 2006.
- Chan, Lam, Lee, Ting. *Continuous monitoring of distributed data streams over a time-based sliding window.* STACS 2010.
- Yi, Zhang. *Optimal tracking of distributed heavy hitters and quantiles.* PODS 2009.