Approximate aggregation for tracking quantiles and range countings in wireless sensor networks

Zaobo He\textsuperscript{a,1}, Zhipeng Cai\textsuperscript{a,*}, Siyao Cheng\textsuperscript{b,2}, Xiaoming Wang\textsuperscript{c,3}

\textsuperscript{a} Department of Computing Science, Georgia State University, United States
\textsuperscript{b} School of Computer Science and Technology, Harbin Institute of Technology, China
\textsuperscript{c} School of Computer Science, Shaanxi Normal University, China

\begin{abstract}
We consider the problem of tracking quantiles and range countings in wireless sensor networks. The quantiles and range countings are two important aggregations to characterize a data distribution. Let $S(t) = (d_1, \ldots, d_s)$ denote the multi-set of sensory data that have arrived until time $t$, which is a sequence of data orderly collected by nodes $s_1, s_2, \ldots, s_k$. One of our goals is to continuously track $\epsilon$-approximate $\phi$-quantiles ($0 \leq \phi \leq 1$) of $S(t)$ for all $\phi$’s with efficient total communication cost and balanced individual communication cost. The other goal is to track $(\epsilon, \delta)$-approximate range countings satisfying the requirement of arbitrary precision specified by different users. In this paper, a deterministic tracking algorithm based on a dynamic binary tree is proposed to track $\epsilon$-approximate $\phi$-quantiles, whose total communication cost is $O(k/\epsilon \cdot \log n \cdot \log^2(1/\epsilon))$, where $k$ is the number of the nodes in a network, $n$ is the total number of the data, and $\epsilon$ is the user-specified approximation error. For range countings, a Bernoulli sampling based algorithm is proposed to track $(\epsilon, \delta)$-approximate range countings, whose total communication cost is $O\left(\frac{2}{\epsilon^2} \ln \frac{2}{1-\sqrt{1-\delta}} + n_c\right)$, where $\delta$ is the user-specified error probability, $n_c$ is the number of clusters.
\end{abstract}

\section{1. Introduction}

Wireless Sensor Networks (WSNs) consist of many nodes which interact with each other through wireless channels. They are now being widely employed to collect physical information, such as temperature, pressure, light intensity and so forth [13]. With the development of wireless communication technologies, the scale of a WSN can be very large [11]. However, the most severe constraint imposed on their extensive applications is the power supply as the on-board power is still the main power source which is limited in most cases. Compared with computation, communications among nodes consume more energy. According to [16], the energy consumption for sending one bit is enough for executing 1000 instructions for one sensor. Thus, how to extract significant information from a huge amount of sensory data with a reasonable communication cost becomes a crucial problem.

\* Corresponding author. Tel.: +1 (404) 413 5721.
E-mail address: zcai@gsu.edu (Z. Cai).
\footnote{1} Supported by NSF under grant No. CNS-1252292.
\footnote{2} Supported by the NSFC of China under grant No. 61370217.
\footnote{3} Supported by the NSFC of China under grants Nos. 61173094 and 61373083.

http://dx.doi.org/10.1016/j.tcs.2015.07.056
0304-3975/© 2015 Elsevier B.V. All rights reserved.
Much effort has been spent on studying various aggregations, including algebraic aggregations such as Sum, Count, and Average [4,9,10], or holistic aggregations such as Heavy hitters [12] and Quantiles [5,7,8,15]. In-network aggregation techniques are very meaningful for algebraic aggregations through computing partial results at intermediate nodes during the process of transmitting data to the sink [4,7,8,10,15]. By preventing nodes from forwarding all the data to the sink, in-network aggregation techniques significantly reduce energy consumption. They are meaningful for algebraic aggregations since these aggregations are decomposable [2], however, holistic aggregations are not decomposable so that the in-network aggregation techniques cannot be employed to track quantiles directly [5]. The ϕ-quantile (0 ≤ ϕ ≤ 1) of an ordered dataset S is the data x such that ϕ|S| elements of S are less than or equal to x and no more than (1 − ϕ)|S| elements are larger than x, particularly, the 1/2-quantile is the median of S. Another fundamental aggregation is range counting. For any query interval Q, the Q-range counting over an ordered dataset S is the value of |Q ∩ S|.

Since exact aggregations incur large communication costs and storage spaces, approximate results are usually expected. The work in [17] shows that a random sample of size Θ(1/ε^2) needs to be drawn from a dataset to compute ε-approximate quantiles with a constant probability. Moreover, practical applications involve various precision requirements so that aggregation results that can satisfy an arbitrary precision is expected. ε-approximate ϕ-quantiles and (ε, δ)-approximate Q-range countings are defined respectively as follows:

Definition 1 (ε-Approximate ϕ-quantiles). The ε-approximate ϕ-quantile of a dataset S is a set of elements satisfying (ϕ − ε)n ≤ r(x) ≤ (ϕ + ε)n where r(x) is the rank of x in S and n is the total number of elements in S.

Definition 2 ((ε, δ)-Approximate Q-range countings). Let Y and Y′ denote the exact and approximate Q-range countings respectively, i.e., Y = |Q ∩ S| and Y′ = |Q ∩ U| where U is a random sample set of S. An (ε, δ)-approximate Q-range counting is such Y′ that satisfies Pr(Y − ε|S| ≤ Y′ ≤ Y + ε|S|) ≥ 1 − δ, where Pr(X) is the probability of random event X.

Various data models have been studied in the database communities, such as the static model, single-stream model and multi-stream model. For the static model, data is predetermined and stored at nodes and aggregation function f is computed over the union of these multiple datasets. For the single-stream model, there is only one node, and data stream into it in an online fashion. Nowadays, the multi-distributed streaming model attracts a lot of attention since it is more general in the physical environment. In this model, data stream into each node in a distributed way and the tracking results are returned in a logical coordinator. Moreover, for the quantile tracking results, the querying mode can be divided into two classes: single ϕ-quantile and all ϕ-quantiles. For the single ϕ-quantile tracking, a certain summary is always maintained by a coordinator to compute a certain ϕ-quantile. Comparatively, the data structure or summary preserved by a coordinator for all ϕ-quantiles can be used to compute any ϕ simultaneously.

We want to track quantiles in a general manner, where the sensor nodes are organized into a spanning tree and sensory data stream into each node in an online fashion. S(t) = (d1, ..., dn) is the multi-set of items of the entire network that have arrived until time t, which is a sequence of data orderly collected by nodes s1, s2, ..., sk. The goal is to continuously track ε-approximate ϕ-quantiles (0 ≤ ϕ ≤ 1) of S(t) at the sink for all ϕ’s. Moreover, we design a Bernoulli sampling based algorithm for tracking range countings to satisfy arbitrary precision requirements specified by different users.

The main contributions of this work can be summarized as follows. First, quantiles can be tracked over the arrived data at any time t rather than through a one-time computation over a predetermined dataset. Second, quantiles are computed based on an arbitrary topological spanning tree rather than the centralized flat model. Third, a data structure can be maintained in the tree from which all the ϕ-quantiles can be tracked simultaneously rather than for just a specific ϕ. Fourth, the proposed range countings tracking algorithm can satisfy arbitrary precision requirements.

2. Related works

The previous quantile tracking techniques can be divided into three categories, which are the exact algorithms, deterministic algorithms and probabilistic algorithms. For a given ϕ, the exact algorithms return the exact ϕ-quantile result. According to [14], the space complexity for computing the exact median with p passes is Ω(n^{1/p}). Clearly, the space complexity of the exact algorithms is high, especially when the number of the passes p is small.

To further reduce time and space complexities during tracking quantiles, some deterministic algorithms are proposed, such as the recent works in [6,8,15,18]. Unlike the exact algorithms, the deterministic ones return ε-approximate ϕ-quantiles of a dataset. Since the deterministic algorithms just require approximate results, they have smaller space and communication complexities. In 2005, Cormode et al. [5] proposed an all-tracking algorithm with the cost of $O\left(\frac{k}{\epsilon^2} \log n\right)$. The work in [18] improves this result by a $\Theta(\frac{1}{\epsilon})$ factor, whose result has an upper bound $O\left(\frac{k}{\epsilon} \log n\right)$. Note that the work in [18] discusses the all ϕ-quantiles tracking problem under the flat model. However, it is unclear how to track quantiles in the tree model.

Considering that the approximate quantile with a probability guarantee can be accepted in most cases, the complexity of tracking the quantile can be further reduced. Thus, a group of probabilistic algorithms [4,8,10] were proposed. Different from the above two types of algorithms, the probabilistic algorithms require that the ε-approximate ϕ-quantile result is
guaranteed with a probability. For example, the work in [8] proposes a quantile estimator and partitions the routing tree to compute $\epsilon$-approximate quantiles within a constant probability with the total communication cost of $O(\sqrt{kH}/\epsilon)$ where $H$ is the height of the routing tree. However, this work just carries out one-time computation over the predetermined dataset, so it is not clear whether it works well for the data stream model.

Meanwhile, sampling algorithms also draw much attention. The work in [17] proposes a sampling algorithm to track $\epsilon$-approximate range countings with sampling probability $\Theta(\epsilon^{-2}\log 1/\epsilon)$. A deterministic algorithm is proposed by [3] to track range counting with sample size $O(\epsilon^{-2}\log^{1/2})$. However, these algorithms cannot satisfy the arbitrary precision requirements specified by different users.

3. Problem definition

Let $S = (d_1, \ldots, d_n)$ be a sequence of items, which is disorderly collected by nodes $s_1, \ldots, s_k$. Each node samples a sensory value from the monitored environment in each time slot. $S(t)$ is the multi-set of items of the entire network that have arrived until time $t$. Each user specify a time interval $(0, t_{final}]$, then, the general goal is to continuously track all $\phi$ quantiles at all times $t$ with efficient communication cost, $t \in (0, t_{final}]$.

For the quantile tracking problem, a binary tree is constructed to initialize the tracking algorithm, which is based on a small dataset collected by each node. Thus, the algorithm takes the initial dataset $S(0) = S_1(0) \cup S_2(0) \cup \ldots \cup S_k(0)$ as input. Meanwhile, to describe the network feature, we introduce $\Delta_t$ to denote the length of the interval between two adjacent time slots and $n$ to denote the total number of the data items in the entire tracking process, $n = |S(t_{final})|$. Last, for any user-specified $\epsilon$ and $\phi$, the algorithm returns those elements $x$ in dataset $S(t)$ with rank $r(x)$, $(\phi - \epsilon)n \leq r(x) \leq (\phi + \epsilon)n$, as defined in Definition 1. Thus, for any user-specified $(\epsilon, \delta)$ and $Q$, the output of the algorithm is $\Upsilon'$ that satisfies $\Pr[\Upsilon' - \epsilon|S| \leq \Upsilon' \leq \Upsilon + \epsilon|S|] \geq 1 - \delta$, as defined in Definition 2.

For clarity, the investigated problem in this paper for tracking $\epsilon$-approximate $\phi$-quantiles and $(\epsilon, \delta)$-approximate $Q$-range countings can be defined as follows:

**$\epsilon$-Approximate $\phi$-quantiles**

**Input:**
1) $\epsilon (\epsilon \geq 0)$ and $\phi (0 \leq \phi \leq 1)$.
2) $n$ and $\Delta_t$.
3) $S(t)$ $\forall i = 1, 2, \ldots, k$.

**Output:**
Data $x$ that satisfies $\{x|x \in S(t), (\phi - \epsilon)n \leq r(x) \leq (\phi + \epsilon)n\}$, as defined in Definition 1.

**$\epsilon, \delta$-Approximate $Q$-range countings**

**Input:**
1) $S(t) = S_1(t) \cup S_2(t) \cup \ldots \cup S_k(t)$.
2) $\epsilon (\epsilon \geq 0)$, $\delta (0 \leq \delta \leq 1)$.
3) user-defined interval $Q$.

**Output:**
$\Upsilon'$ that satisfies $\Pr[\Upsilon' - \epsilon|S| \leq \Upsilon' \leq \Upsilon + \epsilon|S|] \geq 1 - \delta$, as defined in Definition 2.

For clarity, the meanings of the symbols are summarized in Table 1.

4. Tracking the quantiles

In order to efficiently track quantiles in WSNs, the whole network is organized by a spanning tree rooted at the sink. The nodes in the spanning tree can be distinguished as the leaf nodes and the intermediate nodes, where the communication cost of the intermediate nodes is large since they not only maintain local datasets but also relay data of its descendants. Therefore, one key problem of tracking quantiles in WSNs is to reduce the communication cost of the intermediate nodes. To achieve this goal, we develop a global data structure over the routing tree and maintain it dynamically with a bounded communication cost.

We divide the entire tracking period into $O(\log n)$ rounds, denoted by $m_i$ ($i = 1, 2, \ldots, \log n$). Whenever $|S(t)|$ has increased by a constant factor, e.g., $|S(t)|$ is doubled, a new round is started. Assuming at time $t''$, round $m_i$ is launched. We use $M_i$ to denote the set of data at the beginning of round $m_i$, i.e., $M_i = S(t'')$. $M_i$ is fixed throughout round $m_i$, i.e., $M_i \subseteq |S(t)|$ and $t'' \leq t$. It is always true that $|S(t)| = \Theta(|C \cdot S(t)|)$ for constant $C$ and $|M_i| = C \cdot |S(t)|$ is ensured in one round. Thus, in round $m_i$, we have $|S(t)| = \Theta(|C \cdot M_i|)$. Based on this reason, our goal can be described in another way: along with data streaming into network continuously, the goal is to maintain a data structure over $S(t)$ based on which the rank of any data item $x (x \in S(t))$ can be extracted with error $O(\epsilon M_i)$, where $M_i \subseteq S(t) \subseteq M_{i+1}$ and $1 \leq i \leq \log n$.

Since the operations of initialization, maintenance and tracking are similar in each round, we first focus on one round and then obtain the total cost for all rounds naturally. For simplicity, we assume that all the data values are distinct. In summary, the algorithm for tracking quantiles in WSNs is described as follows. First, initialize a binary tree $T$ based on
Table 1
Major symbols.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k)</td>
<td>The number of the sensor nodes</td>
</tr>
<tr>
<td>(s_i)</td>
<td>Sensor node (s_i) ((1 \leq i \leq k))</td>
</tr>
<tr>
<td>(r(x))</td>
<td>The rank of data (x)</td>
</tr>
<tr>
<td>(\Delta t)</td>
<td>The length of the interval between two adjacent time slots</td>
</tr>
<tr>
<td>(s_i(t))</td>
<td>The sensory dataset sampled by sensor node (i) ((1 \leq i \leq k)) until time (t)</td>
</tr>
<tr>
<td>(s_i(0))</td>
<td>The initial dataset preserved by sensor node (i) ((1 \leq i \leq k)) at the initial time</td>
</tr>
<tr>
<td>(S(t))</td>
<td>The whole sensory dataset in the network at time (t)</td>
</tr>
<tr>
<td>(n)</td>
<td>The total number of the data items in the entire tracking process</td>
</tr>
<tr>
<td>(m_i)</td>
<td>The (i)-th tracking round</td>
</tr>
<tr>
<td>(M_i)</td>
<td>The set of data at the beginning of round (m_i)</td>
</tr>
<tr>
<td>(h)</td>
<td>The height of the binary tree</td>
</tr>
<tr>
<td>(</td>
<td>b)</td>
</tr>
<tr>
<td>(</td>
<td>A_b</td>
</tr>
<tr>
<td>(\mu)</td>
<td>The deviation between (</td>
</tr>
<tr>
<td>(t_r)</td>
<td>The spanning tree rooted at sensor node (a)</td>
</tr>
<tr>
<td>(c_a)</td>
<td>The number of child nodes of node (a)</td>
</tr>
<tr>
<td>(k_{tr})</td>
<td>The number of the sensor nodes in (t_r)</td>
</tr>
<tr>
<td>(T_a)</td>
<td>The binary tree built by sensor node (a)</td>
</tr>
<tr>
<td>(T_{ur})</td>
<td>The binary tree built for spanning tree (t_r)</td>
</tr>
<tr>
<td>(\rho)</td>
<td>The number of the leaf nodes in the spanning tree</td>
</tr>
<tr>
<td>(q)</td>
<td>The number of the intermediate nodes in the spanning tree</td>
</tr>
<tr>
<td>(\delta_i)</td>
<td>The threshold for the number of the data items blocked by sensor node (s_i)</td>
</tr>
</tbody>
</table>

\(|s_i(0)| \leq \frac{1}{k}\). The detailed structure of \(T\) is provided in Section 4.1 and the specific steps of the initialization algorithm are presented in Section 4.2. Second, an accumulatively updating algorithm given in Section 4.3 is carried out to reduce the transmission cost when new data arrives. Third, we need to maintain the binary tree \(T\) so that the height and the leaf nodes of \(T\) satisfy some requirements. The detailed binary tree maintenance algorithm is given in Section 4.4. Fourth, the sink computes the rank of \(x\) \((x \in S(t))\), i.e., \(r(x)\), based on the binary tree \(T\). Finally, the above four steps are executed iteratively until the number of the execution times reaches \(\log n\), where \(n\) is the total number of the sensory values in the network.

### 4.1. The data structure

The data structure for querying is a binary tree \(T\) initialized at the beginning of each round and maintained throughout one round. We take a specific round \(m_i\) as an example to describe the data structure. \(T\) is constructed in the following way. The root of \(T\) is the approximate median of \(M_i\), which divides \(M_i\) into two subsets. Each subset is recursively split by selecting its approximate median as the root of the subtree. The splitting process is iteratively executed until the number of the items in each subset is no more than \(\epsilon|M|/\beta\), where \(\beta\) is a constant satisfying \(\beta > 1\) and \(\epsilon\) is a user-defined error parameter. After the splitting, the intermediate nodes and the root contain only one number (i.e., the median of its subset), and the leaves contain more than one numbers (the upper bound of the number is \(\epsilon|M|/\beta\)). Thus, the data structure is a binary tree with \(\Theta(\beta/\epsilon)\) as the error parameter.

Obviously, \(T\) has \(\Theta(\beta/\epsilon)\) nodes in total and the height of \(T\) is \(h = \Theta(\log \beta/\epsilon)\). Meanwhile, each node in \(T\), denoted by \(b\), corresponds to an interval \(I_b = [l_b, u_b]\), where \(l_b\) and \(u_b\) are the smallest and largest value in the subtree rooted at \(b\) respectively.

The exact results can be obtained if we update \(l_b\) whenever a new sensory value arrives. However, it incurs a large communication cost if both the size of the network and sampling frequency of each sensor are large. In practice, each node \(b\) just needs to correspond to an approximate interval \(A_b\) so that the corresponding interval does not need to be updated every time, where \(A_b\) satisfies \(|l| - \mu \leq |A_b| \leq |l|, \mu\) satisfies \(h\mu + \epsilon|M|/\beta = \epsilon|M|\), and \(h\) is the height of binary tree \(T\).

The tracking process for \(r(x)\) is a traversal process over \(T\) from the root to leaf node \(v\) such that \(x \in A_v\). For each root-to-leaf path, whenever following a right child, the approximate interval size of its left sibling is summed up. Since there are at most \(h\) such intervals, the total error introduced by the traversal process is at most \(h\mu\). Finally, since the interval size corresponded by a leaf node is less than \(\epsilon|M|/\beta\), one can query \(r(x)\) in \(S(t)\) with absolute error \(O(h\mu + \epsilon|M|/\beta)\). If \(h\mu + \epsilon|M|/\beta = \epsilon|M|\) with appropriate parameters \(\mu\) and \(\beta\), the error of \(r(x)\) is \(O(\epsilon|M|)\).

### 4.2. Initialization of the binary tree

The initialization algorithm is launched at the beginning of each round to build a global binary tree \(T\). For any node \(a\) in the spanning tree, \(t_r\) denotes the tree rooted at \(a\) and \(c_a\) denotes the number of children of node \(a\). \(k_{tr}\) is used to denote the number of the nodes in \(t_r\). \(|t_r(0)|\) denotes the total number of the data items preserved by the nodes within \(t_r\) at
the initial time. \( r \) is the sink of the network. \( p \) and \( q \) denote the number of the leaf nodes and the intermediate nodes in the spanning tree respectively, where the leaf node set is denoted by \( \{v_i | i = 1, 2, \ldots, p\} \) and the intermediate node set is denoted by \( \{u_i | i = 1, 2, \ldots, q\} \). The binary tree built by node \( v \) is denoted by \( T_v \) and the binary tree built for spanning tree \( T_{\text{initialization}} \) is denoted by \( T_{\text{initialization}} \).

The initialization algorithm in one round has the following 5 steps.

**Step 1.** Based on its initial dataset, each node \( v_i (1 \leq i \leq k) \) builds its own approximate balanced binary tree \( T_{v_i} \) with \( \epsilon / \beta \) as the error parameter. Now querying any \( r(x) \) in \( s_i(0) \) has an error \( \epsilon |s_i(0)| / \beta \).

**Step 2.** Each leaf node \( v_i (1 \leq i \leq p) \) transmits its binary tree \( T_{v_i} \) to its parent node in the spanning tree.

**Step 3.** Assume \( v_i (1 \leq i \leq u_i) \) is the child node of node \( u_j (1 \leq j \leq q) \). Based on \( T_{v_i} (1 \leq i \leq c_u) \) and \( T_u \), \( u_j \) can compute any \( r(x) \) within \( T_{v_i} \) with error \( \sum_{j=1}^{c_u} \epsilon |s_i(0)| / \beta = \epsilon |T_{v_i}(0)| / \beta \), which is enough for \( u_j (1 \leq j \leq q) \) to build a binary tree \( T_{u_j} \) with \( \epsilon / \beta \) as the error parameter. Finally, \( T_{u_j} \) is transmitted to the parent node of \( u_j \) in the spanning tree.

**Step 4.** Step 3 is iteratively executed until sink \( r \) is reached. Then \( r \) broadcasts \( T \) to the network through the spanning tree.

**Step 5.** After receiving \( T \), each node \( s_i (1 \leq i \leq k) \) computes the exact number of items in each interval and transmits to its parent node, where the total number of intervals corresponded by the global binary tree \( T \) is \( \Theta(\beta / \epsilon) \).

It is easy to derive that the communication cost of the initialization in one round is \( O(k\beta/\epsilon) \).

### 4.3. Updating the interval size accumulatively

#### 4.3.1. Algorithm description

After initialization, each node \( s_i (1 \leq i \leq k) \) preserves a global binary tree \( T \). The naive method of updating \( T \) is to report all sensory values sampled by the nodes to the sink leading to the communication cost of \( O(nH) \), where \( H \) is the height of the spanning tree. Obviously, the cost is very huge since \( n \) is generally much larger than \( k \) and \( 1/\epsilon \), otherwise, we just need to send each sensory data to the sink. Thus, an accumulative updating algorithm is proposed to reduce the communication cost in the updating phase with an accumulative report strategy. Although the error is generated during the quantile tracking process, the proposed algorithm dramatically reduces the communication cost for updating the global binary tree \( T \).

\( Int \) is used to denote an arbitrary interval of \( T \). As described in Section 4.1, each \( Int \) has an exact interval size and an approximated interval size denoted by \( |I| \) and \( |A| \) respectively. Meanwhile, each node keeps a set of counters for counting the size of each \( Int \).

The accumulative updating algorithm includes the following two steps:

**Step 1.** With new sensory data continuously streaming into \( s_i (1 \leq i \leq k) \), \( s_i \) monitors \( Int \) continuously.

**Step 2.** If the local count of \( Int \) at \( s_i (1 \leq i \leq k) \) has increased by a threshold since its last communication to its parent about the local count of \( Int \), \( s_i \) must report an updated local count for \( Int \) to its parent. Then, each \( s_i (1 \leq i \leq k) \) resets the counter to 0 and continuously monitors \( Int \).

Among the steps of the accumulative updating algorithm, determining the threshold is very important since it affects the communication cost and accuracy of the algorithm. Fortunately, this problem can be solved by Lemma 4.1 and Theorem 4.2.

**Lemma 4.1.** To satisfy that any \( r(x) (x \in S(t)) \) can be extracted with error \( O(\epsilon |M_j|) \), where \( M_j \subseteq S(t) \subseteq M_{j+1} \) and \( 1 \leq j \leq \log n \), the condition \( |I| - \mu \leq |A| \leq |I| \) should be ensured, where \( \mu = (1 - 1/\beta) \cdot \epsilon / h \cdot |M_j| \).

**Proof.** As presented in Section 4.1, the querying process for \( r(x) \) in \( T \) is a traversal process over \( T \). Assuming for each interval \( Int \) of \( T \), there exists a parameter \( \mu \) that satisfies \( |I| - \mu \leq |A| \leq |I| \). Thus, the total error introduced by the traversal process from the sink to the leaf node is \( O(h \mu) \) where \( h \) is the height of \( T \). Thus, the total error of querying any \( r(x) \) in \( T \) is \( O(h \mu + \epsilon |M_j| / \beta) \), where \( \epsilon |M_j| / \beta \) resulted from the interval size corresponded by each leaf node of \( T \) is less than \( \epsilon |M_j| / \beta \). Letting \( h \mu + \epsilon |M_j| / \beta = \epsilon |M_j| \), we can derive the value of \( \mu \).

**Theorem 4.2.** Let \( H \) denote the height of the spanning tree. Assuming \( s_i (1 \leq i \leq k) \) is a node on layer \( l_i \) \((0 \leq l_i \leq H)\) in the spanning tree, and its ancestor node set is \( \{P_f | 0 \leq f \leq l_i - 1\} \), \( P_f \) is the parent of \( P_{f+1} \) and \( P_{l_i - 1} \) is the parent of \( s_i \). \( c_{P_f} \) is the number of children of \( P_f \). If \( s_i (1 \leq i \leq k) \) reports an updated local count each \( Int \) to its parent when the local count of sensory data in \( Int \) at \( s_i \) has increased by a threshold \( \delta_i \), where \( \delta_i = (\prod_{f=1}^{l_i} c_{P_f})^{-1} \cdot \mu \), querying any \( r(x) \) in \( T \) has error \( O(\epsilon |M_j|) \).

**Proof.** Assume that \( u_i \) \((1 \leq i \leq c_{P_f}) \) is the child node of sink \( r \). Lemma 4.1 shows that to achieve the purpose that querying any \( r(x) \) in \( T \) has error \( O(\epsilon |M_j|) \), each \( u_i \) \((1 \leq i \leq c_{P_f}) \) only needs to satisfy the condition \( |I| - \mu / c_f \leq |A| \leq |I| \), where \( |A| \) and \( |I| \) are the approximate and exact size of \( Int \) monitored by \( u_i \). To satisfy this condition, \( u_i \) \((1 \leq i \leq c_{P_f}) \) only needs to query its child nodes \( v_j \) \((1 \leq j \leq c_{u_i}) \) to satisfy \( |I'| - \mu / c_{u_i} \leq |A'| \leq |I'| \), where \( |A'| \) and \( |I'| \) are the approximate and exact size of \( Int \) monitored by \( v_j \) \((1 \leq j \leq c_{u_i}) \). Using mathematical induction, it is easy to derive the updating threshold \( \delta_i \) for \( s_i \) \((1 \leq i \leq k) \).

---

The text is extracted from the paper titled "Z. He et al. / Theoretical Computer Science 607 (2015) 381–390" which appears to be focused on algorithms for updating binary trees in a network context.
4.3.2. Communication cost

Now we analyze the communication cost for the accumulative updating process in one round. Note that the cost for one time communication is regarded as one unit. When $s_i$ ($1 \leq i \leq k$) sends an updated count message for $|A|$ to its parent, the communication cost incurred by the accumulated $\delta_i$ sensory data is viewed as one unit so that the average cost of one item is $O(1/\delta_i)$. Since each item may incur $h$ times of such a message transmitting process, the average cost incurred by an item is $O(h/\delta_i)$. After substituting the expression of $\delta_i$, one can obtain $O(h/\delta_i) = O((1 - 1/\delta)^{-1} \cdot h^2/\epsilon \cdot |M_j| \cdot \prod_{i=1}^{t} c_{P_i})$. Since in one round, the number of the data items streaming into $s_i$ ($1 \leq i \leq k$) is $\Theta(|M_j(i)|)$, the cost of updating the local count of $|A|$ at $s_i$ ($1 \leq i \leq k$) is $O((1 - 1/\delta)^{-1} \cdot h^2/\epsilon \cdot |M_j(i)| \cdot |M_j| \cdot \prod_{i=1}^{t} c_{P_i})$. Thus, combining the condition $h = \Theta(\log(\beta/\epsilon))$, the total cost of updating the binary tree $T$ in one round is $O((1 - 1/\delta)^{-1} \cdot 1/\epsilon \cdot \log^2(\beta/\epsilon) \cdot \sum_{i=1}^{k} |M_j(i)| \cdot |M_j| / \prod_{i=1}^{t} c_{P_i})$. Assuming data streams into each node with a similar speed i.e., $|M_j(i)| / |M_j| = 1/k$, the above expression can be rewritten as $O((1 - 1/\delta)^{-1} \cdot 1/\epsilon \cdot \log^2(\beta/\epsilon) \cdot 1/k \cdot \sum_{i=1}^{k} (\prod_{j=1}^{t} c_{P_j}))$. It is easy to derive that

$$\sum_{i=1}^{k} (\prod_{j=1}^{t} c_{P_j}) \leq k^2,$$

then the communication cost for the accumulative updating process in one round is $O((1 - 1/\delta)^{-1} \cdot k/\epsilon \cdot \log^2(\beta/\epsilon))$.

4.4. Maintaining the binary tree

4.4.1. Algorithm description

The global binary tree $T$ may become unbalanced with new items arriving in the data stream, which leads to a high communication cost for tracking quantiles. Thus, the approximate median as a splitting element should not deviate from the exact median too much. Meanwhile, the data structure requires that the interval size corresponded by the leaf nodes of $T$ should not be beyond $\epsilon |M_j| / \beta$ in round $m_i$ ($1 \leq i \leq \log n$). Thus, our goal is to maintain the height of $T$ as $h = \Theta(\log(\beta/\epsilon))$ and the interval size corresponded by the leaf nodes of $T$. Before presenting the specific maintenance algorithm, we first introduce Lemma 4.3 and Theorem 4.4.

For any intermediate node $u$ in binary tree $T$, let $v$ and $w$ be the left and right child of $u$ respectively. $|A_u|$, $|A_v|$ and $|A_w|$ denote the approximate interval size of $T$ corresponded by node $u$, $v$ and $w$ respectively. Meanwhile, $|I_u|$, $|I_v|$ and $|I_w|$ denote the exact interval size of $T$ corresponded by node $u$, $v$ and $w$ respectively.

Lemma 4.3. For any intermediate node $u$ with left child $v$ and right child $w$ in $T$ and parameter $\lambda$, ($0 < \lambda < 1/2$), if $\lambda |I_u| \leq |I_v| \leq (1 - \lambda)|I_u|$, it is always true that $h = \Theta(\log(\beta/\epsilon))$.

Proof. In the extreme case, if each intermediate node $u$ of $T$ has left child $v$ and right child $w$, then $|A_v| \geq (1 - \lambda)|A_u|$ and $|A_w| \leq \lambda |A_u|$. Now $T$ has the highest height contributed by each left subtree of $T$. Since the interval size of $T$ corresponded by each leaf node is of at most $\epsilon |M_j| / \beta$, it is easy to derive that in this extreme case, the height of $T$ is $h = \log_{1-\lambda} (\epsilon / \beta)$, so that we have $\epsilon / \beta = (1 - \lambda)^h$. Evaluating the logarithm to $\epsilon / \beta = (1 - \lambda)^h$, one can obtain $\log(\beta/\epsilon) = -h \log(1 - \lambda)$. From this equation, one can derive $h = -\log(\beta/\epsilon) / \log(1 - \lambda) = \Theta(\log(\beta/\epsilon))$. Thus, condition $\lambda |I_u| \leq |I_v| \leq (1 - \lambda)|I_u|$ always bounds the height of $T$ by $h = \Theta(\log(\beta/\epsilon))$. $\square$

The above proof shows that $h = -\frac{1}{\log(1 - \lambda)} \log(\beta/\epsilon)$. Further, we can obtain that $\lim_{\lambda \to 0} \frac{1}{\log(1 - \lambda)} = -\infty$, $\lim_{\lambda \to 1} \frac{1}{\log(1 - \lambda)} = C$ ($C$ is a constant which is determined by the base of the logarithmic function), $\lim_{\lambda \to 1} \frac{1}{\log(1 - \lambda)} = 0$. We find that $h = -\log(\beta/\epsilon) / \log(1 - \lambda) = \Theta(\log(\beta/\epsilon))$ is hold only on the condition that $\lambda$ converges to $1/2$. Thus, $\lambda(0 < \lambda < 1/2)$ determines the constant $C$ in $h = C \log(\beta/\epsilon)$. In fact, $\lambda(0 < \lambda < 1/2)$ is a specific network parameter which is different in different applications.

Theorem 4.4. For each $|A_u|$, $|A_v|$ and $|A_w|$, if condition

$$\eta |A_u| \leq |A_v| \leq (1 - \eta) |A_u| \tag{4.1}$$

is always satisfied, the height of $T$ is bounded by $h = \Theta(\log(\beta/\epsilon))$, where $\eta = (\lambda h_i + 1) / (h_i - 1)$, $0 < \lambda < 1/2$ and $h_i$ is the height of $T$ at the beginning of a round.

Proof. With parameter $\eta$ ($0 \leq \eta \leq 1$) which always satisfies $\eta |A_u| \leq |A_v|$, based on condition $|I_u| - \mu \leq |A_u| \leq |I_u|$, we have

$$\eta (|I_u| - \mu) \leq \eta |A_u| \leq |A_v| \leq |I_v| \leq |I_v| + \mu. \tag{4.2}$$

From Lemma 4.1, we know that $\mu$ should be set as $\mu = (1 - 1/\beta) \cdot (\epsilon / h_i)|M_j|$ to compute $r(x)$ with error $O(\epsilon |M_j|)$. Substituting this value into the left part of (4.2), we have

$$\eta (|I_u| - (1 - 1/\beta) \frac{\epsilon}{h_i} |M_j|) \leq |I_v| + \mu. \tag{4.3}$$
Since the interval size corresponded by the leaf node of T is at most $\epsilon|\mathcal{M}_l|/\beta$ and $I_v \subseteq I_u$, for each intermediate node $u$, it is easy to derive that $(1 - 1/\beta)\epsilon|\mathcal{M}_l| \leq \epsilon|\mathcal{M}_l|/\beta \leq |I_u|$. Substituting this condition into the left part of (4.3), one can obtain

$$\eta \left(1 - \frac{1}{h_1}\right)|I_u| \leq |I_v| + \mu. \quad (4.4)$$

Moreover, from $(1 - 1/\beta)\epsilon|\mathcal{M}_l| \leq |I_u|$, one can obtain $\mu \leq |I_u|/h_1$. Substituting this condition into (4.4) and rearranging it, we have $\eta (1 + \eta)/h_1)|I_u| \leq |I_v|$. Similarly, we can derive $|I_u| \leq (1 - \eta + (1 + \eta)/h_1)|I_u|$. In summary, there exists a $\eta$ which satisfies

$$\left(\eta - \frac{1 + \eta}{h_1}\right)|I_u| \leq |I_v| \leq \left(1 - \frac{1 + \eta}{h_1}\right)|I_u|. \quad (4.5)$$

Let $\lambda = \eta - (1 + \eta)/h_1$ so that (4.5) can be described as $\lambda |I_u| \leq |I_v| \leq (1 - \lambda)|I_u|$. According to Lemma 4.3, condition 4.1 ensures that the height of $T$ is bounded by $h = \Theta(\log(\beta/\epsilon))$. \qed

Thus, Theorem 4.4 provides the critical condition whether binary tree $T$ is unbalanced or not. Next, we will introduce the maintenance algorithm to let the height of $T$ always satisfy $h = \Theta(\log(\beta/\epsilon))$.

When the critical condition 4.1 is violated, i.e., one of the two conditions $|\mathcal{A}_v| < |\mathcal{A}_u|$ or $|\mathcal{A}_v| > (1 - \eta)|\mathcal{A}_u|$ is satisfied for the binary tree rooted at $u$, a partial rebuilding is needed to restore condition 4.1 for the partial binary tree rooted at $u$. If one of these two conditions is satisfied at several nodes in $T$ simultaneously, we rebuild the highest tree rooted at one of these nodes. The operation of rebuilding the binary tree rooted at $u$ needs to initialize the binary tree rooted at $u$. The initialization algorithm is shown in Section 4.2.

Meanwhile, as data items arrive, we need to make sure that the interval size of $T$ corresponded by each leaf node of $T$ is not larger than $\epsilon|\mathcal{M}_l|/\beta$, i.e., $|\mathcal{A}_v| \leq \epsilon|\mathcal{M}_l|/\beta$, where $v$ is a leaf node of $T$. In each round, each approximate interval $|\mathcal{A}_v|$ will be monitored and $v$ will be split by adding two children for $v$ as new leaves whenever $|\mathcal{A}_v| > \epsilon|\mathcal{M}_l|/\beta - \mu$. Because $A_v$ has error of at most $\mu$, $|\mathcal{A}_v| > \epsilon|\mathcal{M}_l|/\beta - \mu$ will ensure that $|\mathcal{A}_v| \leq \epsilon|\mathcal{M}_l|/\beta$. The splitting process is also to initialize the interval of $T$ corresponded by $v$.

4.4.2. Communication cost

If the sink detects that the binary tree rooted at $u$ is unbalanced, it just needs to initialize the partial binary tree rooted at $u$, so that the cost of this partial initialization operation is $O\left(\frac{k^2}{\epsilon} \cdot \frac{1}{|\mathcal{M}_l|}\right)$. Since $I_u \subseteq I_v$, it means that a new rebuilding operation for the binary tree rooted at $u$ is needed iff $|I_u|$ has increased by a constant factor. Thus, the average cost for an item to rebuild the tree rooted at $u$ once is $O\left(\frac{k^2}{\epsilon} \cdot \frac{1}{|\mathcal{M}_l|}\right)$. Since each item is contained in $O(h)$ intervals, i.e., one item may incur $O(h)$ times of rebuilding, the average cost for an item to rebuild the tree rooted at $u$ is $O\left(\frac{k^2}{\epsilon} \cdot \frac{h}{|\mathcal{M}_l|}\right)$. Thus, the communication cost for maintaining the balance of $T$ in one round is $O\left(\frac{k^2}{\epsilon} \cdot \frac{h}{|\mathcal{M}_l|} \cdot |\mathcal{M}_l|\right) = O\left(\frac{k^2}{\epsilon} \cdot h\right) = O\left(\frac{k^2}{\epsilon} \cdot \log(\beta/\epsilon)\right)$.

To split the leaf node $v$ of $T$, the sink launches the initialization algorithm for the interval corresponded by $v$. Since the initialization operation is conducted on the interval corresponded by leaf $v$, it incurs a cost of $O\left(\frac{k^2}{\epsilon} \cdot \frac{1}{|\mathcal{M}_l|}\right)$. Since the interval size corresponded by $v$ is less than $\epsilon|\mathcal{M}_l|/\beta$, i.e., $|I_v| \leq \epsilon|\mathcal{M}_l|/\beta$, one can derive $O\left(\frac{k^2}{\epsilon} \cdot \frac{1}{|\mathcal{M}_l|}\right) = O(k)$. Since $T$ has at most $\beta/\epsilon$ leaf nodes, the cost for the splitting operation in one round is $O(k\beta/\epsilon)$.

In summary, the communication cost of the maintenance algorithm in one round is $O\left(\frac{k^2}{\epsilon} \cdot \log(\beta/\epsilon)\right)$.

4.5. Total communication cost

The above analysis shows that the communication cost for the accumulative updating algorithm is dominant. Since $\beta$ is a constant factor, we can obtain the following conclusion:

**Proposition.** There is a deterministic algorithm that can continuously track $\epsilon$-approximate $\phi$-quantiles in WSNs for all $\phi$ ($0 \leq \phi \leq 1$) with a communication cost of $O\left(\frac{k}{\epsilon} \cdot \log n \cdot \log^2 \left(\frac{1}{\epsilon}\right)\right)$.

5. Tracking the range countings

Many of the existing algorithms for tracking range countings have fixed error bounds, which cannot satisfy the arbitrary precision requirement specified by different users. In this section, we propose a Bernoulli sampling based algorithm that can track $(\epsilon, \delta)$-approximate range countings with efficient communication cost.

5.1. Determination of sampling probability

Since our algorithm is sampling-based, how to decide the sampling probability to ensure the user-specified tracking precision is the key problem. Generally speaking, the larger the sampling probability, the higher precision of the tracking
result, however, a larger sample size results in a higher communication cost. To find the optimal sampling probability, we first present the definition of \((\epsilon, \delta)\)-approximate \(\phi\)-quantiles:

**Definition 3** ((\(\epsilon, \delta\))-Approximate \(\phi\)-quantiles). Let \(U\) be the sample set of \(S\) and \(x\) be the \(\phi\)-quantile of \(U\). \(x\) is an \((\epsilon, \delta)\)-approximate \(\phi\)-quantile of \(S\) iff \(\Pr[(\phi - \epsilon)m \leq r_S(x) \leq (\phi + \epsilon)n] \geq 1 - \delta\) where \(r_S(x)\) is the rank of \(x\) in \(S\) and \(n\) is the total number of the data items (i.e., \(n = |S|\)).

Then, **Lemma 5.1** gives the optimal sample size for tracking \((\epsilon, \delta)\)-approximate \(\phi\)-quantiles.

**Lemma 5.1.** If a sample set \(U\) with \(|U| = m\) satisfies that

\[ m \geq \frac{\ln \frac{2}{\delta}}{2\epsilon^2}, \]

then the \(\phi\)-quantile of \(U\) is an \((\epsilon, \delta)\)-approximate \(\phi\)-quantile for dataset \(S\).

**Proof.** According to **Definition 3**, the \(\phi\)-quantile of \(U\) is not an \((\epsilon, \delta)\)-approximate \(\phi\)-quantile for dataset \(S\) iff the following two cases appear:

**Case 1:** There exist more than \(\phi m\) data in \(U\), whose rank is less than \((\phi - \epsilon)|S|\) in dataset \(S\).

**Case 2:** There exist more than \((1 - \phi)m\) data in \(U\), whose rank is larger than \((\phi + \epsilon)|S|\) in dataset \(S\).

Since our method is Bernoulli sampling-based, the distribution of the data in \(U\) is independent. Therefore, the sample collection process can be described in another way: randomly sampling one item from a box containing \(|S|\) data. We introduce a \(0-1\) variable \(X_i\) for each sampled data \(x_i\), \(1 \leq i \leq m\).

Firstly, if the rank of an arbitrary \(x_i\) is less than \((\phi - \epsilon)|S|\), \(X_i\) is set to "1", otherwise, "0". The above situation can be described as follows:

\[ X_i = \begin{cases} 
1 & \text{if } r_S(x_i) < (\phi - \epsilon)|S| \\
0 & \text{else}
\end{cases} \]

where \(r_S(x_i)\) denotes the rank of \(x_i\) in \(S\). Assume \(X = \sum_{i=1}^{m} X_i\) denotes the sum of the number of data in \(U\) whose rank is less than \((\phi - \epsilon)|S|\) in \(S\). It is easy to derive that \(E(X_i) = \phi - \epsilon\), where \(E(X_i)\) is the expectation of variable \(X_i\). Then, we can obtain

\[ E[X] = E[\sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} E[X_i] = m(\phi - \epsilon) = m(\phi - \epsilon). \]

According to Hoeffding Inequality [1], we have

\[ \Pr[X > \phi m] = \Pr[X - E[X] > \phi m - E[X]] = \Pr[X - E[X] > \epsilon m] \leq e^{-2\epsilon^2 m}. \]

Secondly, if the rank of an arbitrary \(x_i\) is larger than \((\phi + \epsilon)|S|\), \(X_i\) is set to "1", otherwise, "0". The above situation can be described as follows:

\[ X_i = \begin{cases} 
1 & \text{if } r_S(x_i) > (\phi + \epsilon)|S| \\
0 & \text{else}
\end{cases} \]

Assume \(X = \sum_{i=1}^{m} X_i\) denotes the sum of the number of data in \(U\) whose rank is larger than \((\phi + \epsilon)|S|\) in \(S\). It is easy to derive that \(\Pr[X_i = 1] = 1 - \phi - \epsilon\) and \(E(X) = m(1 - \phi - \epsilon)\). According to Hoeffding Inequality [1], we have

\[ \Pr[X > (1 - \phi)m] = \Pr[X - E[X] > (1 - \phi)m - E[X]] = \Pr[X - E[X] > \epsilon m] \leq e^{-2\epsilon^2 m}. \]

According to **Definition 3**, \(\phi\)-quantile of \(U\) is an \((\epsilon, \delta)\)-approximate \(\phi\)-quantile of \(S\) iff \(1 - 2e^{-2\epsilon^2 m} \geq 1 - \delta\). Thus, we have

\[ m \geq \frac{\ln \frac{2}{\delta}}{2\epsilon^2}. \]

Then, the sample size that can track \((\epsilon, \delta)\)-approximate range countings is determined by **Theorem 5.2**.
Theorem 5.2. If sample set $U$ has size $m$

$$m \geq \frac{2}{\epsilon^2} \ln \frac{2}{1 - \sqrt{1 - \delta}},$$

(5.1)

then $\Upsilon'$ is an $(\epsilon, \delta)$-approximate $Q$-range counting for dataset $S$.

Proof. If $m \geq \frac{2}{\epsilon^2} \ln \frac{2}{1 - \sqrt{1 - \delta}}$ according to Lemma 5.1, the $\phi$-quantiles of sample set $U$ is an $(\epsilon, \sqrt{1 - \delta})$-approximate $\phi$-quantiles of $S$. Thus, for an arbitrary data $s_i \in S$, $1 \leq i \leq m$, we have

$$\Pr((\phi - \frac{\epsilon}{2}) \leq r(s_i) \leq (\phi + \frac{\epsilon}{2}) | S| \geq \sqrt{1 - \delta}).$$

Then, for two arbitrary data $s_i, s_j$ in sample set $U$, the above formula can be rewritten in other ways respectively:

$$\Pr(|_S(r_S(s_i)) - \frac{\epsilon}{2} | S| \leq r_U(s_i) \leq r_S(s_i) + \frac{\epsilon}{2} | S| \geq \sqrt{1 - \delta}}$$

$$\Pr(|_S(r_S(s_j)) - \frac{\epsilon}{2} | S| \leq r_U(s_j) \leq r_S(s_j) + \frac{\epsilon}{2} | S| \geq \sqrt{1 - \delta})$$

Assuming $s_j > s_i$, we have

$$\Pr(|_S(r_S(s_j) - r_S(s_i)) - \epsilon | S| \leq r_U(s_j) - r_U(s_i)$$

$$\leq (r_S(s_j) - r_S(s_i)) + \epsilon | S| \geq 1 - \delta)$$

According to Definition 2, we have $\Upsilon = |Q \cap S| = r_S(s_j) - r_S(s_i)$ and $\Upsilon' = r_U(s_j) - r_U(s_i)$. Thus, the above formula can be rewritten as

$$\Pr(\Upsilon - \epsilon | S| \leq \Upsilon' \leq \Upsilon + \epsilon | S| \geq 1 - \delta) \quad \square$$

Then, it is easy to derive the sampling probability $q$ for $(\epsilon, \delta)$-approximate range counting

$$q \geq \frac{2}{\epsilon^2 n} \ln \frac{2}{1 - \sqrt{1 - \delta}},$$

(5.2)

5.2. Bernoulli sampling

To reduce the communication cost, the intermediate nodes compute the partial range countings based on the data received rather than transmitting the raw data to the base station. The network is divided into $n_c$ clusters and all the cluster heads are organized as a spanning tree rooted at the sink. Then, the Bernoulli sampling process can be described with the following three steps:

**Step 1.** The sink transmits the sampling probability $q$ to each cluster head using the spanning tree-based protocol.

**Step 2.** Each cluster head broadcasts $q$ among its cluster after receiving $q$. Each node sends its data to the cluster head with probability $q$.

**Step 3.** After receiving the data sent from the nodes, each cluster head computes the partial range counting. Then, each cluster head sends its intermediate result to the sink using the spanning tree-based protocol.

5.3. Communication cost

Let $n_c$ denote the number of clusters for the whole network. In Step 1, the communication cost for transmitting the sampling probability from the sink to the cluster heads network is $O(n_c)$. In Step 2, the communication and computation cost for collecting the data from the $i$th cluster to cluster head and computing the partial results is $O(|B_i|)$, where $1 \leq i \leq n_c$ and $B_i$ is the Bernoulli sample of Cluster $i$. In Step 3, the communication cost for transmitting the partial range counting from each cluster head to the sink is $O(n_c)$. In summary, the total communication cost of tracking the approximate range counting is $O(|B| + n_c)$, where $B$ is the sample set for the entire network. According to Theorem 5.2, the optimal sample size is

$$\frac{2}{\epsilon^2} \ln \frac{2}{1 - \sqrt{1 - \delta}}.$$ 

Thus, the total communication cost of tracking the range counting is $O(\frac{2}{\epsilon^2} \ln \frac{2}{1 - \sqrt{1 - \delta}} + n_c)$.

6. Conclusion

This paper studies the problem of tracking quantiles and range countings in WSNs. Firstly, a binary tree based data structure is proposed to achieve continuous tracking of $\epsilon$-approximate $\phi$-quantiles $(0 \leq \phi \leq 1)$ over the arrived sensory data for all $\phi$’s. The communication cost of the proposed algorithm is $O(k/\epsilon \cdot \log n \cdot \log^2 \frac{1}{\epsilon})$. Compared with the previous works, the proposed algorithm can 1) track quantiles over distributed data streams; 2) obtain quantiles over an arbitrary
topological spanning tree; and 3) track all \( \phi \)-quantiles simultaneously. Meanwhile, a Bernoulli sampling based algorithm is proposed to track the \((\epsilon, \delta)\)-approximate range counting with communication cost of \(O\left(\frac{2}{\epsilon^2} \ln \frac{2}{1-\sqrt{1-\delta}} + n_T\right)\), comparatively, it can satisfy the arbitrary precision requirement specified by different users.

References