Approximate Aggregation for Tracking Quantiles in Wireless Sensor Networks

Zaobo He[1], Zhipeng Cai[1(✉)], Siyao Cheng[1,2], and Xiaoming Wang[3]

[1] Department of Computing Science, Georgia State University, Atlanta, USA
zcai@gsu.edu
[2] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
[3] School of Computer Science, Shaanxi Normal University, Xi'an, China

Abstract. We consider the problem of tracking quantiles in wireless sensor networks with efficient communication cost. Compared with the algebraic aggregations such as Sum, Count, or Average, holistic aggregations such as quantiles can better characterize data distribution. Let $S(t) = (d_1, \ldots, d_n)$ be the multi-set of sensory data that have arrived until time t in the entire network, which is a sequence of data orderly collected by nodes $s_1, s_2, \ldots, s_k$. The goal is to continuously track ϵ-approximate ϕ-quantiles ($0 \leq \phi \leq 1$) of $S(t)$ at the sink for all ϕ's with efficient total communication cost and balanced individual communication cost. In this paper, a deterministic tracking algorithm based on a dynamic binary tree is proposed to track ϵ-approximate ϕ-quantiles ($0 \leq \phi \leq 1$) in wireless sensor networks, 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 items, and ϵ is the required approximation error.

1 Introduction

Wireless Sensor Networks (WSNs) consist of many nodes which interact with each other through wireless channel. They are now being widely deployed to monitor physical information, such as temperature, pressure, light intensity and so forth [1,2,6–8,10,18,21]. With the development of technologies, the scale of a WSN can be very large [19]. However, the most severe constraint imposed on the extensive applications is the limited power supply as the on-board power is still the main power source which is not rechargeable in most cases. Compared with data computation or storage control, communications among nodes consume more energy. According to [25], the energy consumption for sending one bit data is equal to that for executing 1000 instructions for one sensor. Thus, how to extract information from a huge amount of sensor data with efficient communication cost becomes a crucial problem.

Much effort has been spent on studying various aggregation operations (denoted by function f), including algebraic aggregations such as Sum, Count, or Average [5,16,17], holistic aggregations such as Heavy hitters [20], Quantiles

Z. Zhang et al. (Eds.): COCOA 2014, LNCS 8881, pp. 161–172, 2014.
DOI: 10.1007/978-3-319-12691-3_13

[9,13,15,24], and complex correlation queries in the database area such as Distributed joins [22]. Quantile allows one to extract the order statistics information from the dataset which is widely used in network monitoring [4,29] and database query optimization [11], so that data distribution can be much better characterized. In-network aggregation algorithms are efficient techniques to track algebraic aggregations through computing partial results at intermediate nodes during the process of routing data to the sink [5,13,15,17,24]. By preventing nodes from forwarding all the data to the sink, in-network aggregation algorithms significantly reduce energy consumption. In-network aggregation algorithms can be conducted efficiently for algebraic functions due to the decomposable property of these aggregations [3]. Unfortunately, unlike Sum, Count, or Average, quantiles are not decomposable so that the traditional in-network aggregation algorithms do not work well to track quantiles [9]. The ϕ-quantile ($0 \leq \phi \leq 1$) of an ordered dataset S is the data x such that $\phi|S|$ elements of S are less than or equal to x and no more than $(1-\phi)|S|$ elements are larger than x, particularly, the $\frac{1}{2}$-quantile is the median of S.

Since exact results always require huge storage space and large communication cost in WSNs, approximate results are generally expected. The work in [27] shows that a random sample of size $\Theta(1/\epsilon^2)$ is needed to be drawn from a dataset to compute ϵ-approximate quantiles with a constant probability. Moreover, in many applications, the approximate results, rather than exact ones, are good enough for users to perform analysis and make decisions, such as trend analysis [9], anomaly detection [26], and so on. Based on these reasons, an ϵ-approximate ϕ-quantile is expected which can be formally defined as follows:

Definition 1. ϵ-approximate ϕ-quantiles: The ϵ-approximate ϕ-quantiles are those elements in dataset S such as element x that satisfies $(\phi-\epsilon)n \leq r(x) \leq (\phi+\epsilon)n$ where $r(x)$ is the rank of x in S and n is the total number of the data items.

For quantile-tracking objective, the data model can be divided into three classes: static model, single-stream model and multi-stream model. For the static model, data is predetermined and stored at nodes and f is computed over the union of these multiple datasets. For the single-stream model, there is only one node and data arrives at it in an online fashion. The goal is to track f over the items that have arrived with the minimum storage space or communication cost. Nowadays, the multi-distributed streaming model attracts a lot of attention since it is more general in the physical environment. In this model, data streams into each node in a distributed way and the tracking results are returned in a logical coordinator. If all the nodes are connected to one coordinator directly, it is called a flat model. The nodes in a WSN is organized into a spanning tree and the tracking results are returned at the sink. Moreover, for tracking results, the querying mode can be divided into two classes: single ϕ-quantile and all ϕ-quantile. For the single ϕ-quantile tracking mode, a certain summary always is maintained by a coordinator to compute a certain ϕ-quantile. Comparatively, the data structure or summary preserved by a coordinator for all the ϕ-quantiles

can be used to compute any ϕ simultaneously. The bottleneck of single ϕ-quantile is that frequent tracking operations, such as multiple sampling, are needed to satisfy different user-defined ϕ's.

The aforementioned reasons motivate us to track quantiles in WSNs in a general way, which can be described as follows. The sensor nodes are organized into a spanning tree and sensory data streams into each node in an online fashion. The intermediate nodes not only need to relay data of its descendants, but also hold a local dataset for itself. $S(t)$ is the multi-set of items of the entire network that have arrived until time t. $S(t) = (d_1, \ldots, d_n)$ is a sequence of data that is collected orderly by nodes $s_1, s_2, \ldots, s_k$. The goal is to continuously track ϵ-approximate ϕ-quantiles ($0 \leq \phi \leq 1$) of $S(t)$ at the sink for all ϕ's.

The main contribution 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.

Thus, our tracking operation is conducted on a platform that combines multi-steam and all ϕ-quantile computations, but it is also significantly complex either. Finally, our algorithm can continuously track the ϕ-quantiles over dataset $S(t)$ for all ϕ's and has a total communication cost of $O\left(\frac{k}{\epsilon}\log n \log^2 \frac{1}{\epsilon}\right)$, where k is the number of the sensor nodes in the network, n is the total number of the data items, and ϵ is the required approximation error.

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 are to return the exact ϕ-quantile result to users. According to [23], the space complexity for computing the exact median with p passes is $\Omega(n^{1/p})$. Clearly, the space complexity of the exact algorithms is high, especially when the number of the passes p is small.

To further reduce the time and space complexities during tracking quantiles, the deterministic algorithms are proposed, such as the recent works [12,15,24,28]. Unlike the exact algorithms, the deterministic algorithms return an ϵ-approximate ϕ-quantiles of a dataset. Since the deterministic algorithms just require approximate results, they have lower space and communication complexities. In 2005, Cormode *et al.* [9] proposed an all-tracking algorithm with the cost of $O\left(\frac{k}{\epsilon^2}\log n\right)$. The work in [28] 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 [28] 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 by users in most cases, the complexity of tracking the quantile

can be further reduced. Thus, a group of probabilistic algorithms [5,15,17] were proposed. Different from the above two types of the algorithms, the probabilistic algorithms require that the ϵ-approximate ϕ-quantile result is guaranteed with a probability. For example, the work in [15] proposes a quantile estimator and partitions the routing tree to compute ϵ-approximate quantiles within constant probability with the total communication cost of $O(\sqrt{kH}/\epsilon)$ where H is the hight 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.

3 Problem Definition

Without loss of generality, we assume that there are k sensor nodes in a WSN, denoted by $\{s_1, s_2, ..., s_k\}$. Meanwhile, we assume that each node samples a sensory value from the monitored environment at each time slot and it holds a small dataset before the algorithm is initiated. Δt is used to denote the length of the interval between two adjacent time slots and $s_i(t)$ denotes the sensory dataset sampled by node i $(1 \leq i \leq k)$ until time t. Therefore, a sensory dataset can be obtained at node i for any given time $t \in [0, +\infty)$, and $S(t) = s_1(t) \cup s_2(t) \cup ... \cup s_i(t)$ denotes the entire sensory dataset in the network at time $t \in [0, +\infty)$. We assume at the initial time, each node s_i preserves an initial dataset $s_i(0)$.

For any given ϕ $(0 \leq \phi \leq 1)$ and integer n $(0 \leq n \leq 1)$, if there exists $t' \in [0, +\infty]$ satisfying $n = |S(t')|$, then the proposed algorithm is to return the ϵ-approximate ϕ-quantiles at t $(\forall t \in [0, t'])$. Note that the definition of ϵ-approximate ϕ-quantile is given in Definition 1. Specifically, the problem studied in this paper is defined as follows.

Input:
1) ϵ $(\epsilon \geq 0)$ and ϕ $(0 \leq \phi \leq 1)$.
2) n and Δt.
3) $\{s_i(0) \mid i = 1, 2, \ldots, k\}$.
Output:
ϵ-approximate ϕ-quantile for any $t \in [0, t']$, where t' satisfies that $n = |S(t')|$.

4 The Proposed Algorithm

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 need to maintain local datasets but also need to 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.*, $|M_i| \leq |S(t)|$ and $t'' \leq t$. It is always true that $\epsilon|S(t)| = \Theta(\epsilon|C \cdot S(t)|)$ for constant C and $|M_i| = C \cdot |S(t)|$ is ensured in one round. Thus, in round m_i, we have $\epsilon|S(t)| = \Theta(\epsilon|M_i|)$.

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 $\{s_i(0) \mid 1 \leq i \leq k\}$. The detailed structure of T is provided in Sect. 4.1 and the specific steps of the initialization algorithm are presented in Sect. 4.2.

Second, an accumulatively updating algorithm given in Sect. 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 detail of the binary tree maintenance algorithm is given in Sect. 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 as given in Sect. 3.

4.1 The Data Structure

The data structure for querying is a binary tree T that is 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 their 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 $\epsilon|M_i|/\beta$, where β is a constant satisfying $\beta > 1$ and ϵ is a user-defined error parameter as shown in Sect. 3. Thus, the data structure is a binary tree with ϵ/β 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 values in the subtree rooted at b respectively.

The exact results can be obtained if we update I_b whenever a new sensory value arrives. However, it incurs a huge 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 $|I_b| - \mu \leq |A_b| \leq |I_b|$, μ satisfies $h\mu + \epsilon|M_i|/\beta = \epsilon|M_i|$, 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 $\epsilon|M_i|/\beta$, one can query $r(x)$ in $S(t)$ with absolute error of $O(h\mu + \epsilon|M_i|/\beta)$. If let $h\mu + \epsilon|M_i|/\beta = \epsilon|M_i|$ with corresponding parameters μ and β, the error of $r(x)$ is $O(\epsilon|M_i|)$.

4.2 Initialization of the Binary Tree

Algorithm Description. The initialization algorithm is initiated at the beginning of each round to build a global binary tree T. For any node a in the spanning tree, tr_a denotes the tree rooted at a and c_a denotes the number of children for node a. k_{tr_a} is used to denote the number of the nodes in tr_a. $|tr_a(0)|$ denotes the total number of the data items preserved by the nodes within tr_a 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 $\{v_i \mid i = 1, 2, \ldots, p\}$ and the intermediate node set is denoted by $\{u_i \mid i = 1, 2, \ldots, q\}$. The binary tree built by node a is denoted by T_a and the binary tree built for spanning tree tr_a is denoted by T_{tr_a}.

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

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

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 $(1 \leq i \leq c_{u_j})$ is the child node of node u_j $(1 \leq j \leq q)$. Based on T_{v_i} $(1 \leq i \leq c_{u_j})$ and T_{u_j}, u_j can compute any $r(x)$ within tr_{u_j} with an error of $\sum_{i=1}^{c_{u_j}} \epsilon|s_i(0)|/\beta = \epsilon|tr_{u_j}(0)|/\beta$, which is enough for u_j $(1 \leq j \leq q)$ to build a binary tree $T_{tr_{u_j}}$ with ϵ/β 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 clear that each interval size of T is exact at the initial time.

Communication Cost. The communication cost of the initialization algorithm in one round is analyzed as follows. In Step 2, the communication cost of the network is $O(p\beta/\epsilon)$ since each leaf node v_i $(1 \leq i \leq p)$ needs to transmit the binary tree T_{v_i} to its parent node and the size of T_{v_i} is $\Theta(\beta/\epsilon)$. Similarly, the communication cost generated in Step 3 is $O(q\beta/\epsilon)$ since the intermediated nodes also need to report the binary tree to their parents. In Step 4, the communication cost is $O(k\beta/\epsilon)$ since the global binary tree T needs to be broadcasted to k nodes and the size of T is $\Theta(\beta/\epsilon)$. Finally, each node needs to transmit the number of the items in each interval to its parent along the spanning tree, and the communication cost is $O(k\beta/\epsilon)$. In summary, the communication cost of the initialization algorithm in one round is $O(k\beta/\epsilon)$.

4.3 Updating the Interval Size Accumulatively

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 far larger than k and $1/\epsilon$, otherwise, we just need to send each sensory data to the sink. Thus, an accumulatively 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 Sect. 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 accumulatively 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 accumulatively 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 Theorems 1 and 2.

Theorem 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 $\mu = (1 - 1/\beta) \cdot \epsilon/h \cdot |M_j|$.*

The proof of Theorem 1 is given in our technique report [14].

Theorem 2. *Assuming s_i $(1 \leq i \leq k)$ is a node located at layer l_i $(0 \leq l_i \leq H)$ in the spanning tree, its ancestor node set is $\{P_f \mid 0 \leq f \leq l_i - 1, P_f$ 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 for Int to its parent when the local count of sensory data in Int at s_i has increased by a threshold δ_i, where $\delta_i = (\prod_{f=1}^{l_i} c_{P_f})^{-1} \cdot \mu$, querying any $r(x)$ in T has error $O(\epsilon|M_j|)$.*

The proof of Theorem 2 is given in our technique report [14].

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 δ_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 shipping process, the average cost incurred by an item is $O(h/\delta_i)$. After substituting the expression of δ_i, one can obtain $O(h/\delta_i) = O((1-1/\beta)^{-1} \cdot h^2/\epsilon \cdot |M_j|^{-1} \cdot \prod_{f=1}^{l_i} c_{P_f})$. 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 $O(h|M_j(i)|/\delta_i) = O((1-1/\beta)^{-1} \cdot h^2/\epsilon \cdot |M_j(i)|/|M_j| \cdot \prod_{f=1}^{l_i} c_{P_f})$. Thus, combing the condition $h = \Theta(\log(\beta/\epsilon))$, the total cost of updating the binary tree T in one round is $O((1-1/\beta)^{-1} \cdot 1/\epsilon \cdot \log^2(\beta/\epsilon) \cdot \sum_{i=1}^{k}(|M_j(i)|/|M_j| \prod_{f=1}^{l_i} c_{P_f}))$. 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 $O((1-1/\beta)^{-1} \cdot 1/\epsilon \cdot \log^2(\beta/\epsilon) \cdot 1/k \cdot \sum_{i=1}^{k}(\prod_{f=1}^{l_i} c_{P_f}))$. It is easy to derive that $\sum_{i=1}^{k}(\prod_{f=1}^{l_i} c_{P_f}) \leq k^2$, then the communication cost for the accumulative updating process in one round is $O((1-1/\beta)^{-1} \cdot k/\epsilon \cdot \log^2(\beta/\epsilon))$.

4.4 Maintaining the Binary Tree

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 $\epsilon|M_i|/\beta$ in round m_i $(1 \leq i \leq \log n)$. Thus, our goal is to maintain the height of the binary tree 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 1 and Theorem 3.

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

The proof of Lemma 1 is given in our technique report [14].

Theorem 3. *For each $|A_u|, |A_v|$ and $|A_w|$, if condition*

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

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.

The proof of Theorem 3 is given in our technique report [14].

Thus, Theorem 3 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 (1) in Theorem 3 is violated, *i.e.*, one of the two conditions $|A_v| < \eta|A_u|$ or $|A_v| > (1-\eta)|A_u|$ is satisfied for the binary tree rooted at u, a partial rebuilding is needed to restore condition (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. Operation of rebuilding the binary tree rooted at u needs to initialize the binary tree rooted at u. The initialization algorithm is shown in Sect. 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 $\epsilon|M_i|/\beta$, *i.e.*, $|A_v| \leq \epsilon|M_i|/\beta$, where v is a leaf node of T. In each round, each approximate interval $|A_v|$ will be monitored and v will be split by adding two children for v as new leaves whenever $|A_v| > \epsilon|M_i|/\beta - \mu$. Because A_v has error of at most μ, $|A_v| > \epsilon|M_i|/\beta - \mu$ will ensure that $|A_v| \leq \epsilon|M_i|/\beta$. The splitting process is also to initialize the interval of T corresponded by v.

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 $O\left(\frac{k\beta}{\epsilon} \cdot \frac{|I_u|}{|M_i|}\right)$. Since $I_v \subseteq I_u$, 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 $O(\frac{k\beta}{\epsilon} \cdot \frac{1}{|M_i|})$. 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 $O(\frac{k\beta}{\epsilon} \cdot \frac{h}{|M_i|})$. Thus, the communication cost for maintaining the balance of T in one round is $O(\frac{k\beta}{\epsilon} \cdot \frac{h}{|M_i|} \cdot |M_i|) = O(\frac{k\beta}{\epsilon} \cdot h) = O(\frac{k\beta}{\epsilon} \cdot \log(\beta/\epsilon))$.

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 $O\left(\frac{k\beta}{\epsilon} \cdot \frac{|I_v|}{|M_i|}\right)$.

Since the interval size corresponded by v is less than $\epsilon|M_i|/\beta$, *i.e.*, $|I_v| \leq \epsilon|M_i|/\beta$, one can derive $O\left(\frac{k\beta}{\epsilon} \cdot \frac{|I_v|}{|M_i|}\right) = O(k)$. Since T has at most β/ϵ 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 $O(\frac{k\beta}{\epsilon} \cdot \log(\beta/\epsilon))$.

4.5 Total Communication Cost

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

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

5 Conclusion

This paper studies the problem of tracking quantiles in WSNs. A binary tree based data structure is proposed to achieve continuous tracking of ϵ-approximate ϕ-quantiles $(0 \leq \phi \leq 1)$ over the arrived sensory data for all ϕ's. The communication cost of the proposed algorithm is $O\left(k/\epsilon \cdot \log n \cdot \log^2 \frac{1}{\epsilon}\right)$. Compared with the previous works, the proposed algorithm can (1) track quantiles over distributed data stream; (2) obtain quantiles over an arbitrary topological spanning tree; and (3) track all ϕ-quantiles simultaneously. Therefore, the proposed algorithm can better satisfy the requirements of quantile computation for the distributed stream data model with efficient a communication cost.

References

1. Cai, Z., Chen, Z.-Z., Lin, G.: A 3.4713-approximation algorithm for the capacitated multicast tree routing problem. Theoret. Comput. Sci. **410**(52), 5415–5424 (2008)
2. Cai, Z., Lin, G., Xue, G.: Improved approximation algorithms for the capacitated multicast routing problem. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 136–145. Springer, Heidelberg (2005)
3. Calvo, T., Mayor, G., Mesiar, R. (eds.): Aggregation Operators: New Trends and Applications. Physica-Verlag GmbH, Heidelberg (2002)
4. Cao, J., Li, L.E., Chen, A., Bu, T.: Incremental tracking of multiple quantiles for network monitoring in cellular networks
5. Cheng, S., Li, J., Cai, J.: O(ϵ)-approximation to physical world by sensor networks. In: INFOCOM, pp. 3084–3092 (2013)
6. Cheng, X., Du, D., Baogang, X.: Relay sensor placement in wireless sensor networks. Wireless Netw. **14**(3), 347–355 (2008)
7. Cheng, X., Huang, X., Li, D., Weili, W., Du, D.: A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. Networks **42**(4), 202–208 (2003)

8. Cheng, X., Thaeler, A., Xue, G., Chen, D.: Tps: A time-based positioning scheme for outdoor wireless sensor networks. In: IEEE INFOCOM 2004, pp. 2685–2696, Hong Kong, China, 7–11 March 2004
9. Cormode, G., Garofalakis, M.: Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In: SIGMOD, pp. 25–36 (2005)
10. Ding, M., Chen, D., Xing, K., Cheng, X.: Localized fault-tolerant event boundary detection in sensor networks. In: IEEE INFOCOM 2005, pp. 902–913, Miami, USA, 13–17 March 2005
11. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.J.: Domain-driven data synopses for dynamic quantiles. IEEE Trans. Knowl. Data Eng. **17**(7), 927–938 (2005)
12. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. In: SIGMOD '01, pp. 58–66. ACM, New York (2001)
13. Greenwald, M.B., Khanna, S.: Power-conserving computation of order-statistics over sensor networks. In: PODS '04, pp. 275–285. ACM, New York (2004)
14. He, Z., Cai, Z., Cheng, S., Wang, X.: Appendix: Approximate aggregation for tracking quantiles in wireless sensor networks. http://www.cs.gsu.edu/zcai/reports/2014/COCOAAppendix.pdf
15. Huang, Z., Wang, L., Yi, K., Liu, Y.: Sampling based algorithms for quantile computation in sensor networks. In: SIGMOD '11, pp. 745–756. ACM, New York (2011)
16. Keralapura, R., Cormode, G., Ramamirtham, J.: Communication-efficient distributed monitoring of thresholded counts. In: SIGMOD '06, pp. 289–300. ACM, New York (2006)
17. Li, J., Cheng, S.: (ϵ, δ)-approximate aggregation algorithms in dynamic sensor networks. IEEE Trans. Parallel Distrib. Syst. **23**(3), 385–396 (2012)
18. Li, J., Cheng, S., Gao, H., Cai, Z.: Approximate physical world reconstruction algorithms in sensor networks. IEEE Trans. Parallel Distrib. Syst. (2014)
19. Liu, Y., He, Y., Li, M., Wang, J., Liu, K., Mo, L., Dong, W., Yang, Z., Xi, M., Zhao, J., Li, X.-Y.: Does wireless sensor network scale? a measurement study on greenorbs. In: 2011 Proceedings IEEE INFOCOM, pp. 873–881, April 2011
20. Metwally, A., Agrawal, D., El Abbadi, A.: An integrated efficient solution for computing frequent and top-k elements in data streams. ACM Trans. Database Syst. **31**(3), 1095–1133 (2006)
21. Mo, L., He, Y., Liu, Y., Zhao, J., Tang, S.-J., Li, X.-Y., Dai, G.: Canopy closure estimates with greenorbs: Sustainable sensing in the forest. In: SenSys '09, pp. 99–112. ACM, New York (2009)
22. Moon, B., Fernando Vega Lopez, I., Immanuel, V.: Efficient algorithms for large-scale temporal aggregation. IEEE Trans. Knowl. Data Eng. **15**(3), 744 (2003)
23. Munro, J.I., Paterson, M.S.: Selection and sorting with limited storage. In: SFCS '78, pp. 253–258. IEEE Computer Society, Washington, DC (1978)
24. Shrivastava, N., Buragohain, C., Agrawal, D., Suri, S.: Medians and beyond: New aggregation techniques for sensor networks. In: SenSys '04, pp. 239–249. ACM, New York (2004)
25. Siew, Z.W., Wong, C.H., Kiring, A., Chin, R.K.Y., Teo, K.T.K.: Fuzzy logic based energy efficient protocol in wireless sensor networks. ICTACT J. Commun. Technol. (IJCT) **3**(4), 639–645 (2012)
26. Thatte, G., Mitra, U., Heidemann, J.: Parametric methods for anomaly detection in aggregate traffic. IEEE/ACM Trans. Networking **19**(2), 512–525 (2011)
27. Vapnik, V., Chervonenkis, A.: On the uniform convergence of relative frequencies of events to their probabilities. Theory Probab. Its Appl. **16**(2), 264–280 (1971)

28. Yi, K., Zhang, Q.: Optimal tracking of distributed heavy hitters and quantiles. Algorithmica **65**(1), 206–223 (2013)
29. Yu, B.: Comment: Monitoring networked applications with incremental quantile estimation. Stat. Sci. **21**(4), 483–484 (2006)