Approximate Multiple Count in Wireless Sensor Networks

Xiaolin Fang*, Hong Gao*, Jianzhong Li*, and Yingshu Li*+

*School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
{xlforu, honggao, lijzh}@hit.edu.cn
+Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA
yli@cs.gsu.edu

***Abstract*—*COUNT* is a typical aggregation operation in Wireless Sensor Networks (WSNs). In such an operation, the total number of the items which are of the same kind is obtained and only one numerical value is returned as the result. This paper identifies the *multiple count* problem which counts items belonging to multiple categories. For each category, the total number of the items belonging to this category is calculated. Therefore, the returned result is a set of values instead of a single value. The multiple count problem is more challenging than the traditional count problem as the former incurs more communication overhead. This paper proposes a distributed approximate multiple count algorithm which can derive an error bounded result under a specified communication cost constraint for each node. The error of the derived result is $\frac{hN}{L}$, where h is the depth of the routing tree, N is the total number of all the items belonging to all the categories, and L is a representation of the communication cost constraint for each node. Furthermore, the weighted multiple count problem is investigated where different kinds of items to be counted have different weights. The proposed algorithms are evaluated through TOSSIM, a widely used simulation tool for WSNs. The theoretical analysis and simulation results both demonstrate the correctness and effectiveness of the proposed algorithms.**

I. Introduction

COUNT is a typical aggregation operation and has received much attention in Wireless Sensor Networks (WSNs) [1], [2]. In such an operation, the total number of the items which are of the same kind is obtained and only one numerical value is returned as the result. In this paper, we investigate the *multiple count* problem. It differs from the traditional count problem by counting items belonging to multiple categories. For each category, the total number of the items belonging to this category is calculated. Therefore, the returned result should be a set of values instead of a single value. The multiple count problem are involved in many applications. For example, in logistics monitoring systems, the numbers of different goods are desired to capture the sales status; in traffic monitoring systems, the numbers of different vehicles are needed to know the traffic conditions and analyze the automobile exhaust emissions; in wild animal monitoring systems, the numbers of different animals are required to understand the habits of the animals, observe the distribution of different animals, or analyze how the environment changes affect the animals' living conditions; and in war field monitoring systems, the numbers of different armed forces, such as tanks, vehicles and soldiers, are valuable information for making battle plans.

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

For a traditional count problem, the users specify a kind of items or an attribute to be counted, then the network aggregates the count result from all the sensor nodes and sends the single-value result back to the users. Usually, the counting process is carried out based on a routing tree. The counting process starts from the leaf nodes, each of which sends the detected count value to its parent. After a node receives the count values from all its children, it adds in its own count value and sends the sum to its parent. The base station eventually receives the final count result at the end of such a bottom-up aggregation process. As long as the collected information is correct and the communication links are reliable, the users can derive an accurate result, and the communication cost is little. To process a traditional count query, every node only needs to send one packet.

Different from the traditional count problem, the multiple count problem may result in large communication overhead as it involves many kinds of items to be counted rather than just one kind of items. Because the length of a packet is limited in WSNs, a node has to employ many packets to completely cover an entire count result so as to meet the packet length constraint. For instance, Telosb or Micaz motes can send a packet of 28 bytes at a time in default settings. Thus, a packet can only contain a few items' count information. An extreme example of the multiple count problem is whole-network data collection, where every node needs to send raw data to the base station. In this case multiple count incurs a large communication cost. In addition, the nodes near the base station undertake more traffic load, which results in unbalanced energy consumptions of nodes and reduced network lifetime.

As communication dominates the energy cost in WSNs where energy supply is limited, an energy-efficient way to process a multiple count query in WSNs is expected. This paper proposes a distributed approximate multiple count algorithm which can derive an error bounded result under a specified communication cost constraint for each node.

In the multiple count problem, different kinds of items may have different weights. For example, in logistics monitoring

systems, the prices of different goods are different; and in traffic monitoring systems, the exhaust emissions of different kinds of vehicles are different. In the weighted multiple count problem where different kinds of items have different weights, our algorithm can also obtain an error bounded result under the communication cost constraint.

The contributions of this paper are as follows.

- To the best of our knowledge, this is the first work to study the multiple count problem in WSNs.
- Given a routing tree and a communication cost constraint L for each node, a distributed approximate algorithm is proposed which can derive a result with error $\frac{hN}{L}$, where h is the depth of the routing tree and N is the total number of all the items belonging to all the categories.
- The weighted multiple count problem is also studied in this paper. In this problem, different kinds of items to be counted have different weights. The weighted multiple count problem can be solved with a modified version of the above mentioned algorithm. The error of this algorithm can also be bounded while satisfying the communication cost constraint.
- Extensive simulations were conducted to validate the correctness and effectiveness of the proposed algorithms.

The rest of this paper is organized as follows. Section II reviews the related works. Section III formally defines the multiple count problem. The proposed algorithm is introduced in Section IV and the corresponding approximation ratio is also analyzed. Section V investigates the weighted multiple count problem. The performance evaluation results are presented in Section VI and Section VII concludes this paper.

II. Related works

Data aggregation has been widely studied in WSNs [1], [2], [3], [4]. The operations of data aggregation include MAX, MIN, SUM, AVERAGE, QUARTILE, COUNT, et al. The algorithms for data aggregation can be categorized into centralized algorithms and distributed algorithms. The centralized algorithms usually induce higher communication cost, and they may not be suitable for many applications in WSNs. Therefore, more effort has been spent on distributed algorithms. The distributed algorithms include cluster based algorithms, multipath based algorithms [5], [6], [7] and collection tree based algorithms [8]. LEACH [9] and COUGAR [10], [11] are two Cluster based algorithms; Synopsis Diffusion [12] is a multipath based algorithm; TAG [13], Directed Diffusion [14], PEGASIS [15], DB-MAC [16] and EADAT [17] are collection tree based algorithms. Collection tree based algorithms are most widely used as routing trees are easy to construct and maintain. The algorithm proposed in this paper is also a collection tree based algorithm.

The methods for the COUNT operation currently include sampling based methods [18], [19], compress sensing based methods [20], Monte Carlo based methods [21] and statistics based methods [22], [23]. The detection techniques used for different objectives may be different. For example, it may use video process techniques to count the number of cars in traffic monitoring systems [24]; it may use voice recognition techniques to count the number of birds in wild bird monitoring systems [25]; and it may use ultrasonic techniques to count the number of moving objects [26]. However, all these methods are designed for the traditional count problem where communication cost may not be a primary concern, because each sensor node only needs to send one packet to its parent in the collection tree. This paper studies the multiple count problem in WSNs.

The multiple count problem is similar to the frequency count problem in the data stream area. For example, data stream processing may need to count many different kinds of items [27]; sliding window processing may also need to count many different kinds of items [28]. However, these problems are different from the multiple count problem in WSNs identified in this paper. Data stream processing and sliding window processing are always carried out in centralized systems. They do not have to consider the energy consumption issue. Nevertheless, the multiple count problem in WSNs needs to satisfy the communication cost constraint. This paper proposes a distributed approximate multiple count algorithm which can derive an error bounded result under a specified communication cost constraint for each node.

III. Problem Definition

Collection tree based algorithms are most widely employed for data aggregation applications in WSNs. The algorithm proposed in this paper is also a collection tree based algorithm. In a collection tree, each node aggregates the data received from its children, and then sends the aggregated result to its parent. Data is aggregated layer by layer until reaching the base station. Following are some definitions.

Definition 1. *An item count is denoted as $[a_i, c_i]$, where a_i denotes a kind of items or an attribute, and c_i denotes the count of a_i.*

Definition 2. *The communication constraint P means each node can send at most P packets.*

Assume each packet can carry λ item counts, then each node can send λP item counts at most under the communication constraint. Let the item data received by node v be $\{[a_1, c_1], [a_2, c_2], \cdots, [a_m, c_m]\}$. If $m > \lambda P$, then v has to discard some item counts so as to satisfy the communication constraint. This means that error occurs when the communication cost for each node is limited.

Fortunately, the users can tolerate a certain degree of error in many applications. For example, in traffic monitoring systems, because of the complexity of the traffic condition, it is not very easy to count the numbers of different kinds of cars accurately. However, a rough count result is acceptable to roughly describe the traffic condition or the periodical change of the traffic condition. In battlefield monitoring systems, as another example, because of the complexity of the battlefield environment, it is not easy to count the exact numbers of soldiers and vehicles. However, a rough count result is acceptable to approximately describe the deployment of enemy

troops. Therefore, the approximate multiple count problem is investigated in this paper, which is defined as follows.

Definition 3. *Given a routing tree and a communication constraint P, the* approximate multiple count *problem is to find a count aggregation result of a number of items of different kinds, so that the error for each kind of items is minimized, i.e.* $\min\{c-\hat{c}\}$*, where $\hat{c}$ is the aggregation result, and c is the real value.*

IV. Error bounded approximation algorithm

An error bounded approximation algorithm is proposed in this section to solve the multiple count problem. It is a tree based algorithm, which aggregates item counts via an aggregation tree. There are many tree construction methods, any of which can be adopted.

Given an aggregation tree, our algorithm aggregates item counts from the leave nodes to the base station. Before presenting our algorithm, the definition of *block* is given first. For convenience, let $L = \frac{\lambda P - 1}{2}$, which will be frequently used in our algorithm and performance analysis later.

Definition 4. *A block is a set of item counts and the block size is no more than L.*

The block size is the total number of the items in the block. For example, let a block be $B = \{[a_1, c_1], [a_2, c_2], \cdots, [a_m, c_m]\}$, then $\sum_{i=1}^{m} c_i \leq L$.

Definition 5. *A complete block is a set of item counts and the block size is exactly L.*

Definition 6. *An uncomplete block is a set of item counts and the block size is less than L.*

A complete block or an uncomplete block is also a block. Let a complete block be $B = \{[a_1, c_1], [a_2, c_2], \cdots, [a_m, c_m]\}$, then $\sum_{i=1}^{m} c_i = L$. Let B be an uncomplete block, then $\sum_{i=1}^{m} c_i < L$.

The count data received by a node can be divided into multiple complete blocks and at most one uncomplete block. Let us take Fig.1 as an example. Suppose $L = 5$ and the count data is $D = \{[a_1, 4], [a_2, 3], [a_3, 2], [a_4, 1], [a_5, 1], [a_6, 1]\}$, then D can be divided into two complete blocks $B_1 = \{[a_1, 4], [a_2, 1]\}$ and $B_2 = \{[a_2, 2], [a_3, 2], [a_4, 1]\}$, and an uncomplete block $B_x = \{[a_5, 1], [a_6, 1]\}$.

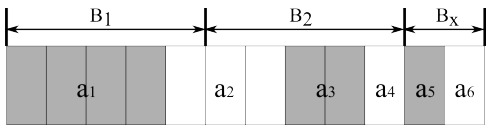

Fig. 1: Illustration of Block division.

The count data a node needs to process includes two parts: the data detected by the node, and the data received from its children. The next two subsections describe how to process the two parts of count data respectively.

A. Processing detected count data

The way a node processes the count data detected by itself is shown in Algorithm 1. This algorithm includes two steps. First, a node devides the count data into blocks. Second, those items whose counts are less than k in the k complete blocks are removed.

Let the detected count data be $D = \{[a_1, c_1], [a_2, c_2], \cdots, [a_n, c_n]\}$, then D can be divided into $k = \lfloor \frac{\sum_{i=1}^{n} c_i}{L} \rfloor$ complete blocks and at most one uncomplete block. In Algorithm 1, lines 4-11 collect the items in an uncomplete block, and put them into set B_x. The remaining counts constitute the k complete blocks. Note that, the blocks are virtual, and Algorithm 1 does not put the item counts into individual k complete blocks and one uncomplete block. Lines 12-14 remove the items whose counts are less than k from the union of the k complete blocks.

Take Fig.1 as an example. The count data can be divided into two complete blocks B_1 and B_2 (i.e. $k = 2$), and an uncomplete block B_x. In $B_1 \cup B_2 = \{[a_1, 4], [a_2, 3], [a_3, 2], [a_4, 1]\}$, the count of a_4 is less than 2, thus, a_4 is removed. After the removing process, it derives ($B_r = \{[a_1, 4], [a_2, 3], [a_3, 2]\}, k = 2, B_x = \{[a_5, 1], [a_6, 1]\}$).

Algorithm 1: DivideRemove(D)

```
Input: D = {[a_1, c_1], [a_2, c_2], ..., [a_n, c_n]}
Output: (B_r, k, B_x)
 1: k = (Σ_{i=1}^{n} c_i)/L;
 2: c = (Σ_{i=1}^{n} c_i)%L;
 3: B_r = ∅; B_x = ∅; i = n;
 4: for i = n to 1 do
 5:     if c − c_i > 0 then
 6:         B_x = B_x ∪ [a_i, c_i];
 7:         c = c − c_i;
 8:     else
 9:         B_x = B_x ∪ [a_i, c];
10:         c_i = c_i − c;
11:         beak;
12: for i = i to 1 do
13:     if c_i ≥ k then
14:         B_r = B_r ∪ [a_i, c_i];
15: return (B_r, k, B_x);
```

Theorem 1. *Let the set of item counts in the complete blocks after the removing process be B_r, then the number of different kinds of items in B_r is less than L, i.e. $|B_r| \leq L$.*

Proof: Those items whose counts are less than k are removed from the complete blocks. Therefore, the counts of the remaining items are all larger than k. Because $k = \lfloor \frac{\sum_{i=1}^{n} c_i}{L} \rfloor$, The total count of all the items of all the complete blocks is no more than kL. Therefore, $|B_r| \leq \frac{kL}{k} = L$. ■

Theorem 1 shows that there are at most L different kinds of items in B_r, i.e. $|B_r| \leq L$. Let the set of items in the uncomplete block be B_x, it is obvious that there are at most L different kinds of items in B_x, i.e. $|B_x| \leq L$. Therefore, the number of different kinds of items in B_r and B_x is less than $2L$.

B. Processing received count data

Two operations $Merge$ and $Remove$ are introduced before processing the received count data. As shown in Algorithm 2, the $Merge$ operation merges the same items in two data sets. Algorithm 3 removes those items whose counts are less than k from D.

Algorithm 2: MERGE(S_1, S_2)

Input: Two count sets S_1 and S_2.
Output: The merged result.

```
1: for [a_i, c_i] in S_1 do
2:     for [a_j, c_j] in S_2 do
3:         if a_i = a_j then
4:             c_i = c_i + c_j;
5:             S_2 = S_2 − [a_j, c_j];
6: return S_1 ∪ S_2;
```

Algorithm 3: REMOVE(D, k)

Input: $D = \{[a_1, c_1], [a_2, c_2], \cdots, [a_n, c_n]\}$
Output: The items whose counts are no less than k.

```
1: for i = n to 1 do
2:     if c_i < k then
3:         D = D − [a_i, c_i];
4: return D;
```

The aggregation starts from the leaf nodes of the tree. The leaf nodes process the count data detected by themselves and send the results to their parents. The parents aggregate the data detected by themselves and received from their children, and then send the aggregation result to their parents. Let a processed result of the data detected by a parent be (B_r, k, B_x), and the data it receives from its children be (B'_r, k', B'_x). Then the parent aggregates the data in two steps. First, as shown in lines 1-2 in Algorithm 4, the parent merges the two sets B_x and B'_x of the items from the uncomplete blocks, and processes the merged set using Algorithm 1. Let the processing result be (B''_r, k'', B''_x). Second, it merges the three sets B_r, B'_r and B''_r, and removes the items whose counts are less than $k+k'+k''$, as shown in lines 3-8 in Algorithm 4.

Algorithm 4 is executed every time when a parent receives data from its children. When the parent aggregates the data from all its children, it sends the aggregation result to its parent, and its parent also runs Algorithm 4 when receives this result.

Algorithm 4: AGGREGATE(B'_r, k', B'_x)

Input: Processed result (B_r, k, B_x) of the parent,
processed result (B'_r, k', B'_x) received from a child.
Output: Aggregation result.

```
1: D=MERGE(B_x, B'_x);
2: (B''_r, k'', B''_x)=DIVIDEREMOVE(D);
3: B_r =MERGE(B_r, B'_r);
4: B_r =MERGE(B_r, B''_r);
5: B_r =REMOVE(B_r, k + k' + k'');
6: k = k + k' + k'';
7: B_x = B''_x;
8: (B_r, k, B_x) is the aggregation result;
```

Theorem 2. *The count error of every kind of item satisfies $c \le \hat{c} + \frac{hN}{L}$, where c is the real count, $\hat{c}$ is the aggregation result, h is the depth of the given tree, and N is the total number of all the kinds of items, i.e. $N = \sum c_i$.*

Proof: The theorem is proved by induction. Assume the depth of the tree is 1 (i.e. there is only one node in the tree). It is obvious that the error is less than $k = \frac{N}{L}$.

Assume that the count error of every kind of items is less than $\frac{hN}{L}$ when the depth of the tree is h.

We now prove the count error of every kind of items is less than $\frac{(h+1)N}{L}$ when the depth of the tree is $h+1$ as follows.

Assume the root has m children v_1, v_2, $\cdots$, and v_m. Let the subtrees rooted at these children be T_1, T_2, $\cdots$, and T_m. Let the depth of these subtrees be h_1, h_2, $\cdots$, and h_m ($h_i \le h$, $i = 1, 2, \cdots, m$). Suppose N_1, N_2, $\cdots$, and N_m are the total numbers of all the kinds of items in these subtrees. For simplity, assume the root does not detect any data, thus we have $N = \sum_{i=1}^{m} N_i$. Then the count errors of every kind of items in these subtrees are less than $\frac{h_1 N_1}{L}$, $\frac{h_2 N_2}{L}$, $\cdots$, and $\frac{h_m N_m}{L}$ respectively. Without loss of generality, we consider the aggregation of a kind of items a. The count of a aggregated in these subtrees is at most $C = \sum_{i=1}^{m} \frac{h_i N_i}{L}$ less than the actual count according to the inductive assumption. There are two cases when aggregating the count of a.

1) If a does not appear in the aggregation result, then the aggregation count of a received from v_1, v_2, $\cdots$, and v_m must be less than $\sum_{i=1}^{m} \frac{N_i}{L}$. In this situation, the count of a aggregated at the root is at most $\sum_{i=1}^{m} \frac{N_i}{L}$ less. Therefore, the count of a aggregated in the tree may be $C + \sum_{i=1}^{m} \frac{N_i}{L} = \sum_{i=1}^{m} \frac{(h_i+1)N_i}{L} \le \frac{(h+1)N}{L}$ less than the actual count.

2) If a appears in the aggregation result, then the aggregation count of a received from $v_1, v_2, \cdots$, and v_m must be more than $\sum_{i=1}^{m} \frac{N_i}{L}$. However, the count of a may be received from only one child, while the counts of a in other children are removed. Without loss of generality, let the child not removing the count of a be v_1, then the counts of a in other children are less than $\frac{h_2 N_2}{L}$, $\frac{h_3 N_3}{L}$, $\cdots$, and $\frac{h_m N_m}{L}$ respectively. In this situation, the count of a aggregated at the root is at most $\sum_{i=2}^{m} \frac{N_i}{L}$ less than the actual count. Therefore, the count of a aggregated in the tree is at most $C + \sum_{i=2}^{m} \frac{N_i}{L} = \sum_{i=1}^{m} \frac{h_i N_i}{L} + \sum_{i=2}^{m} \frac{N_i}{L} \le \sum_{i=1}^{m} \frac{(h_i+1)N_i}{L} \le \frac{(h+1)N}{L}$ less than the actual count.

Thus, the theorem is proved. ■

It is easy to get the following corollaries from Theorem 2.

Corollary 1. *The count error of every kind of items appearing in the aggregation result satisfies $c \le \hat{c} + \frac{hN}{L}$.*

Corollary 2. *The count of every kind of items not appearing in the aggregation result is less than $\frac{hN}{L}$.*

Theorem 3. *In an aggregation process, every node satisfies the communication constraint, i.e. every node can send at most P packets.*

Proof: The data every node sends is (B_r, k, B_x), where $|B_r| \le L$ and $|B_x| \le L$. Therefore, the number of different kinds of items every node sends is at most $|B_r| + |B_x| + 1 \le$

$2L+1$, which requires at most $\frac{2L+1}{\lambda} = P$ packets. ■

Theorem 3 shows that the number of packets a node sends is at most P. Therefore, the communication cost of each node is limited. The communication cost constraint can improve the lifetime of a sensor node, meanwhile, it can also alleviate the unbalance problem in an aggregation process so as to improve network lifetime.

Because memory is limited in a resource constrained sensor node, efficient methods are also required to reduce the memory usage. As shown in Theorem 4, our algorithm provides a solution whose memory usage is limited.

Theorem 4. *In an aggregation process, the memory size required by every node is $O(L)$.*

The primary memory usage is the space to cache (B_r, k, B_x). Similar to the communication cost analysis, the memory usage of each node is $O(L)$.

V. Weighted frequency count problem

In many systems, the importance of different kinds of items are different. In these systems, different kinds of items have different weights. This section discusses the weighted multiple count problem.

Definition 7. *Given an aggregation tree, a communication constraint P, and a weight p_i for each kind of items a_i, the weighted multiple count problem is to find a count aggregation result of a number of kinds of items, so that the weighted error for each kind of items is minimized, i.e. $\min\{p_i c_i - p_i \hat{c_i}\}$, where $\hat{c_i}$ is the aggregation result and c_i is the real value.*

The weighted multiple count problem can be solved by a slight modification of Algorithm 4. In the non-weighted multiple count problem, an item count is denoted as $[a_i, c_i]$. In the weighted version, we only need to treat an item count as $[a_i, p_i c_i]$, then the weighted multiple count problem is the same as the non-weighted one. Algorithm 4 can return the result so that the weighted error satisfies $|p_i c_i - p_i \hat{c_i}| \leq \frac{hN}{L}$, where $\hat{c_i}$ is the aggregation result, c_i is the real value, h is the depth of the tree, N is the total weighted count of all kinds of items, i.e. $N = \sum p_i c_i$. The algorithm satisfies the communication cost constraint, and it can also guarantee that the memory usage is $O(L)$.

VI. Performance evaluation

The simulations are conducted on TOSSIM, which is widely used for WSN simulations. A programm running on TOSSIM can be easily implemented on real test-beds. The network is generated by deploying the nodes into a 1000×1000 area. The communication distance is set to 80. We use two test datasets to evaluate the performance of our algorithm. The first dataset is the carbon dioxide emissions in the world from 2008 [29], and the second is the traffic count infomation from UK in 2000-2009 [30]. We use the first dataset to evaluate the performance of the unweighted multiple count problem while the second dataset to test the effectiveness of the weighted multiple count problem.

A. Evaluation with the carbon dioxide emission dataset

This dataset is employed to evaluate the performance of the unweighted multiple count problem. The dataset includes carbon dioxide emissions of 216 countries. The carbon dioxide emissions are uniformly and randomly distributed to the nodes in the network. Note that the carbon dioxide emissions of one country may be distributed into multiple nodes. The simulations with different settings of L and different aggregation trees are conducted to demonstrate the effectiveness and efficiency of our algorithm.

1) Impact of L: This group of simulations study the impact of L on count error, communication cost and memory usage. In this group of simulations, the number of nodes in the network is 400. The aggregation tree is a BFS tree rooted at the base station. L is set to 50, 100, 150, 200 respectively.

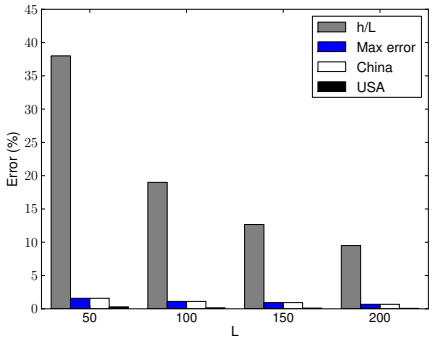

Fig. 2: Relative error with different L settings.

Fig.2 illustrates the relative error of carbon dioxide emissions for different L settings. In this group of simulations, the depth of the constructed aggregation tree is $h = 19$. The relative error is defined as $\frac{c-\hat{c}}{N}$, where c is the real value, $\hat{c}$ is the aggregation count, and N is the total carbon dioxide emissions of all the countries. As shown in Fig.2, the relative error decreases as L increases. This is because more counts can be kept in the pacekts as L increases, thus, fewer counts are dropped in the aggregation process. In Fig.2, the white bars are the relative errors of carbon dioxide emissions of China, and the black bars are the relative errors of carbon dioxide emissions of USA. As L increases, both the relative errors of carbon dioxide emissions of China and USA decrease. The blue bars are the maximum relative error of carbon dioxide emissions among all the countries. Fig.2 shows that the maximum error is far less than the upper bound $\frac{h}{L}$.

Fig.3 demonstrates the data amount a node transmits. The data amount in this group of simulations is the number of item counts a node sends plus one, i.e. the number of items in B_r and B_x, and an additional number k. By Theorem 3, the data amount a node sends is at most $2L+1$. However, in our simulations, the data amount a node sends is much far less than the upper bound. As L increases, the difference between the upper bound and the actual data amount is more significant. In Fig.3, the gray bars are the upper bound $2L+1$, the white bars

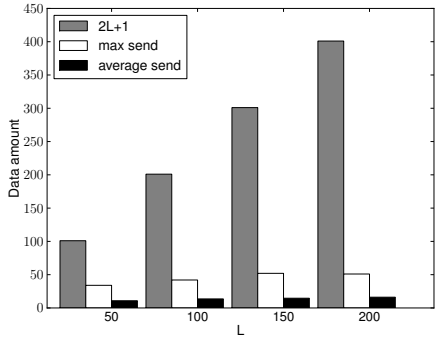

Fig. 3: Data amount for different L settings.

are the maximum data amount a node sends, and the black bars are the average data amount a node sends. As can be seen in Fig.3, the maximum data amount and the average data amount a node sends vary slightly as L increases. In some cases, the data amount a node sends even decreases as L increases. For example, as shown in Fig.3, when L increases from 150 to 200 the maximum data amount a node sends slightly decreases. Applications can take full advantage of this observation. The communication cost can be sharply reduced by loading item counts into fewer packets.

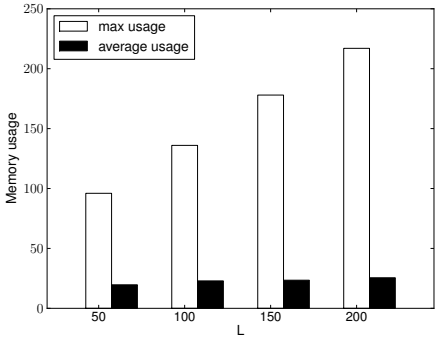

Fig. 4: Memory usage for different L settings.

The next group of simulations is to evaluate how L affects the memory usage of every node. In the aggregation process, every node needs to store the information of (B_r, k, B_x). The memory usage is bounded by Theorem 4. In Fig.4, the white bars are the maximum memory usage, and the black bars are the average memory usage. In order to simplify programming, the memory is statically allocated rather than dynamically allocated in our simulations. Thus, the memory usage is the actual memory that is used within the statically allocated memory. One can use the dynamic allocation component to improve the memory usage. The memory usage of a node includes the memory used to store (B_r, k, B_x) of it is own counted data, the buffer used to cache (B'_r, k', B'_x) received from the children, and the buffer used to merge two item sets. In Fig.4, the average memory usage shows a good performance. It does not vary much as L increases. This observation indicates that the number of item counts that most nodes send is limited. The nodes near the base station receive more item counts, therefore, the memory usage of these nodes is larger than others. The node which uses the maximum memory is always the base station in our simulations. The maximum memory usage increases as L increases.

2) Impact of the depth of tree: The next sets of simulations evaluate the impact of the depth of the aggregation tree on the count error, communication cost and memory usage. In these sets of simulations, L is set to 200, and the number of nodes in the network is 400. The trees with different depths are constructed by selecting different nodes as the base station in the same network. The depths of the constructed trees are 16, 19, 22 and 25 respectively.

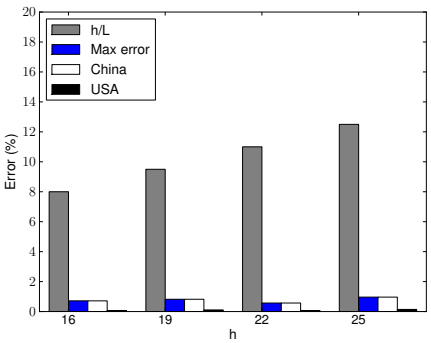

Fig. 5: Relative error for different h settings.

Fig.5 presents the count error for different h settings. The relative error is defined as $\frac{c-\hat{c}}{N}$, where c is the real value, $\hat{c}$ is the aggregation count, and N is the total carbon dioxide emissions of all the countries. Theorem 2 indicates that the error bound is proportional to h. However, the count error in our simulations is far less than the upper bound, and it seems to have no relation with h. In Fig.5, the white bars are the relative errors of carbon dioxide emissions of China, the black bars are the relative errors of carbon dioxide emissions of USA, and the blue bars are the maximum relative error of carbon dioxide emissions among all the countries. Fig.5 shows that count error stays in a low level.

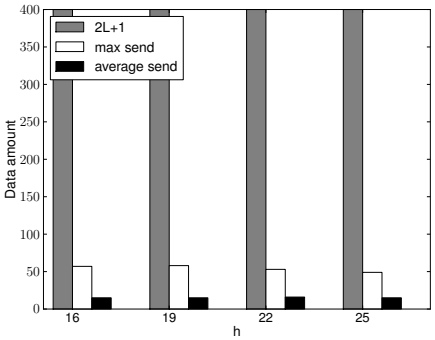

Fig. 6: Data amount for different h settings.

Fig.6 shows the data amount a node transmits for different h settings. The data amount in this group of simulations is the number of item counts in B_r and B_x plus one. In our simulations, the data amount a node sends is far less than the upper bound $2L+1$. In Fig.6, the gray bars are the upper bound $2L+1$, the white bars are the maximum data amount a node sends, and the black bars are the average data amount a node sends. As can be observed from Fig.6, both the maximum data amount and the average data amount a node sends vary slightly as h increases. Applications can reduce communication cost by loading item counts into fewer packets.

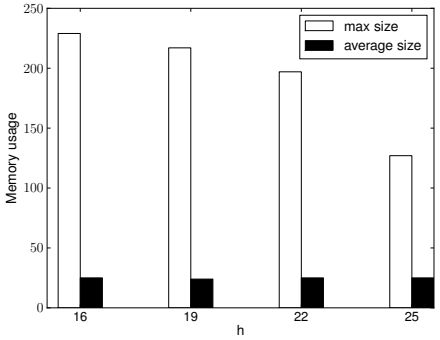

Fig. 7: Memory usage for different h settings.

Fig.7 presents the memory usage for different h settings. The white bars in Fig.7 are the maximum memory usage, and the black bars are the average memory usage. The memory usage of a node includes the memory used to store (B_r, k, B_x) of its own counted data, the buffer used to cache (B'_r, k', B'_x) received from the children, and the buffer used to merge two item sets. In Fig.7, the average memory usage shows a good performance, which stays in a low level. This observation indicates that the number of item counts that most nodes send is limited. The nodes near the base station receive more item counts, therefore, the memory usage of these nodes is larger than others. The node which uses the maximum memory is always the base station in our simulations. The maximum memory usage decreases as h increases. This is because as h increases, the aggregation is processed in more layers. Thus, the number of children of a parent is reduced. The memory used to aggregate fewer children's results is probably less than that to aggregate more children's results.

B. Evaluation with the traffic count dataset

This dataset is adopted to evaluate the effectiveness of the weighted multiple count problem. The dataset records the traffic counts of different kinds of vehicles in different roads from UK in 2000-2009. Different kinds of vehicles includes cars, taxis, motorcycles, buses, coaches, light vans and heavy goods vehicles with two axles, three axles, four or more axles and so on. The traffic counts are randomly and uniformly distributed to the nodes in network. We assign a random weight within $(0, 1]$ to each kind of vehicle of each city. The result will return a set of traffic counts of different kinds of vehicles of a number of cities.

Becaue the results are similar with the unweighted multiple count problem which is evaluated with the carbon dioxide emission dataset, we only evaluate the impact of L on count error, communication cost and memory usage. In this group of simulations, the number of nodes in the network is 1000. The aggregation tree is a BFS tree rooted at the base station. The depth of the tree is 19.

Fig.8 illustrates the relative error of traffic counts for different L settings. The relative error is defined as $\frac{pc-p\hat{c}}{N}$, where p is the weight, c is the real value, $\hat{c}$ is the aggregation count, and $N = \sum p_i c_i$ is the total traffic counts of all kinds of vehicles. As shown in Fig.8, the relative error decreases as L increases. The white bars are the error bound and the black bars are the maximum relative error of traffic counts. Fig.8 shows that the maximum error is far lower than $\frac{h}{L}$.

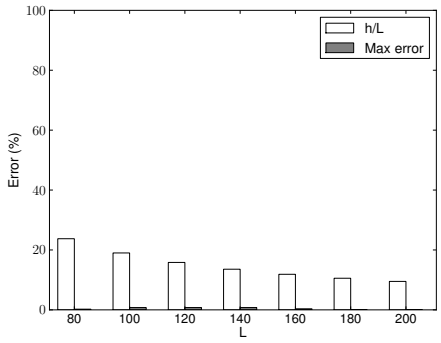

Fig. 8: Relative error for different L settings.

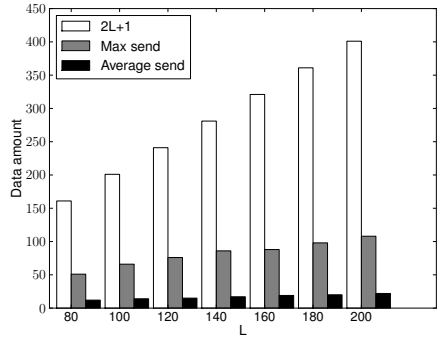

Fig. 9: Data amount for different L settings.

Fig.9 demonstrates the data amount a node transmits. The data amount in this group of simulations is the number of item counts a node sends plus one. By Theorem 3, the data amount a node sends is at most $2L + 1$. However, in our simulations, the data amount a node sends is less than the upper bound. In Fig.9, the white bars are the upper bound $2L+1$, the gray bars are the maximum data amount a node sends, and the black bars are the average data amount a node sends. As can be seen in Fig.9, the maximum data amount and the average data amount

a node sends vary slightly as L increases. It can be found that most of the nodes only need to send a little amount of data. The basestation is always the node that sends the maximum amount of data.

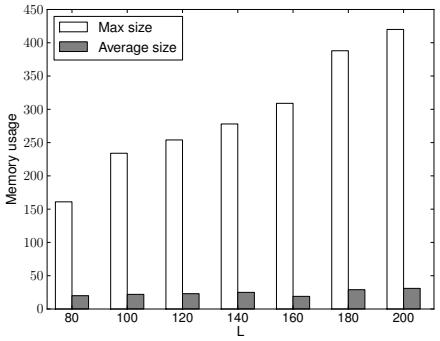

Fig. 10: Memroy usage for different L settings.

The next group of simulations is to evaluate how L affects the memory usage of every node. The memory usage is bounded by Theorem 4. In Fig.10, the white bars are the maximum memory usage, and the black bars are the average memory usage. Again, the memory is statically allocated rather than dynamically allocated in our simulations. Thus, the memory usage is the actual memory that is used within the statically allocated memory. The memory usage of a node includes the memory used to store (B_r, k, B_x) of it is own counted data, the buffer used to cache (B'_r, k', B'_x) received from the children, and the buffer used to merge two item sets. In Fig.10, the average memory usage is little. This observation indicates that the number of item counts that most nodes send is limited. The nodes near the base station receive more item counts, therefore, the memory usage of these nodes is larger than others. The node which uses the maximum memory is always the base station in our simulations.

VII. Conclusion

This paper studies the multiple count problem in wireless sensor networks. Because the count aggregation of many kinds of items increases traffic load and results in load unbalance in a network, an approximation count algorithm aiming at reducing the count error while satisfying the communication cost constraint is proposed in this paper. The error bound of our algorithm is $\frac{hN}{L}$, where h is the depth of the constructed aggregation tree, N is the total count of all kinds of items, and L is a representation of the communication cost constraint. We also prove that the communication cost and the memory usage of our algorithm can also be bounded. We show that the proposed algorithm can also be used to solve the weighted multiple count problem where each kind of item has a weight. Extensive simulations are conducted to evaluate the performance of our algorithm. The simulations consider the impact of communication constraint L, the depth of the tree h and the network scale. The simulation results show that both the error and the communication cost are much lower than the upper bounds. The maximum memory usage is bounded in our analysis, and the average memory usage of our algorithm stays in a low level.

Acknowledgment

This work was supported in part by the Major Program of National Natural Science Foundation of China under grant No. 61190115, the National Basic Research Program of China (973 Program) under grant No. 2012CB316200, and the National Natural Science Foundation of China under grants No. 61033015, No. 60933001, No. 61100030, No. 61370217 and No. 61133002.

References

[1] David Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, SASN '04, pages 78–87, New York, NY, USA, 2004. ACM.

[2] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 239–249, New York, NY, USA, 2004. ACM.

[3] Jianzhong Li, Siyao Cheng, Hong Gao, and Zhipeng Cai. Approximate physical world reconstruction algorithms in sensor networks. *IEEE/ACM Transactions on Parallel and Distributed Systems*, 2014.

[4] Siyao Cheng, Jianzhong Li, and Zhipeng Cai. O(ϵ)-approximation to physical world by sensor networks. In *INFOCOM*, pages 3084–3092, 2013.

[5] Zhipeng Cai, Guohui Lin, and Guoliang Xue. Improved approximation algorithms for the capacitated multicast routing problem. In *COCOON*, pages 136–145, 2005.

[6] Zhipeng Cai, Zhizhong Chen, Guohui Lin, and Lusheng Wang. An improved approximation algorithm for the capacitated multicast tree routing problem. In *COCOA*, pages 286–295, 2008.

[7] Zhipeng Cai, Zhizhong Chen, and Guohui Lin. A 3.4713-approximation algorithm for the capacitated multicast tree routing problem. *Theoretical Computer Science*, 410(52):5415–5424, 2009.

[8] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-network aggregation techniques for wireless sensor networks: a survey. *Wireless Communications, IEEE*, 14(2):70–87, 2007.

[9] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, 2002.

[10] Yong Yao and Johannes Gehrke. Query processing for sensor networks. In *Proceedings of the 1st ACM Conference on Innovative Data Systems Research*, CIDR 2003, Asilomar, CA, US, 2003. ACM.

[11] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, September 2002.

[12] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 250–262, New York, NY, USA, 2004. ACM.

[13] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, December 2002.

[14] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, February 2003.

[15] Stephanie Lindsey, Cauligi Raghavendra, and Krishna M. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Trans. Parallel Distrib. Syst.*, 13(9):924–935, September 2002.

[16] Guido Di Bacco, Tommaso Melodia, and Francesca Cuomo. A mac protocol for delay-bounded applications in wireless sensor networks. In *Proceedings of The Third Annual Mediterranean Ad Hoc Networking Workshop*, pages 208–220, Bodrum, Turkey, June 2004.

[17] M. Ding, Xiuzhen Cheng, and Guoliang Xue. Aggregation tree construction in sensor networks. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 4, pages 2168–2172 Vol.4, 2003.

[18] Siyao Cheng, Jianzhong Li, Qianqian Ren, and Lei Yu. Bernoulli sampling based (ϵ;, δ;)-approximate aggregation in large-scale sensor networks. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 1181–1189, Piscataway, NJ, USA, 2010. IEEE Press.

[19] Siyao Cheng and Jianzhong Li. Sampling based (epsilon, delta)-approximate aggregation algorithm in sensor networks. In *Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*, ICDCS '09, pages 273–280, Washington, DC, USA, 2009. IEEE Computer Society.

[20] Bowu Zhang, Xiuzhen Cheng, Nan Zhang, Yong Cui, Yingshu Li, and Qilian Liang. Sparse target counting and localization in sensor networks based on compressive sensing. In *INFOCOM, 2011 Proceedings IEEE*, pages 2255–2263, 2011.

[21] Dengyuan Wu, Xiuzhen Cheng, Dechang Chen, Wei Cheng, Biao Chen, and Wei Zhao. A monte carlo method for mobile target counting. In *2011 International Conference on Distributed Computing Systems, ICDCS 2011, Minneapolis, Minnesota, USA, June 20-24, 2011*, pages 750–759. IEEE Computer Society, 2011.

[22] Dengyuan Wu, Dechang Chen, Kai Xing, and Xiuzhen Cheng. A statistical approach for target counting in sensor-based surveillance systems. In *INFOCOM*, pages 226–234, 2012.

[23] Y. M. Baryshnikov, E. G. Coffman, K. J. Kwak, and Bill Moran. Stochastic counting in sensor networks, or: Noise is good. In *Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems*, DCOSS '08, pages 32–45, Berlin, Heidelberg, 2008. Springer-Verlag.

[24] K. Cheung and P. Varaiya. Traffic surveillance by wireless sensor network: Final report. Technical Report UCBITS-PRR-2007-4, University of California, Berkeley, Richmond, CA, USA, 2007.

[25] Harry Gros-Desormeaux, Philippe Hunel, Nicolas Vidot, and Erick Stattner. Acoustic counting algorithms for wireless sensor networks. In *Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, PE-WASUN '09, pages 79–84, New York, NY, USA, 2009. ACM.

[26] Quanbin Chen, Min Gao, Jian Ma, Dian Zhang, Lionel M. Ni, and Yunhao Liu. Mocus: moving object counting using ultrasonic sensor networks. *Int. J. Sen. Netw.*, 3(1):55–65, December 2008.

[27] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 346–357. VLDB Endowment, 2002.

[28] Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '04, pages 286–296, New York, NY, USA, 2004. ACM.

[29] Wikipedia. List of countries by carbon dioxide emissions. http://en.wikipedia.org/wiki/List_of_countries_by_carbon_dioxide_emissions.

[30] UK Department for Transport. Traffic counts. http://www.dft.gov.uk/traffic-counts/.