Research Article

An Adaptive Spanning Tree-Based Data Collection Scheme in Wireless Sensor Networks

Yi Zhang, 1,2 Juhua Pu, 1,2 Xingwu Liu, 3 and Zun Chen 1,2

1 The State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
2 Research Institute of Beihang University in Shenzhen, Shenzhen 518057, China
3 Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

Correspondence should be addressed to Juhua Pu; pujh@buaa.edu.cn

Received 26 September 2014; Accepted 8 December 2014

Copyright © 2015 Yi Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In-network data aggregation is a widely used method for collecting data efficiently in wireless sensor networks (WSNs). The authors focus on how to achieve high aggregation efficiency and prolonging networks’ lifetime. Firstly, this paper proposes an adaptive spanning tree algorithm (AST), which can adaptively build and adjust an aggregation spanning tree. Owing to the strategies of random waiting and alternative father nodes, AST can achieve a relatively balanced spanning tree and flexible tree adjustment. Then a redundant aggregation scheme (RAG) is illustrated. In RAG, interior nodes help to forward data for their sibling nodes and thus provide reliable data transmission for WSN. Finally, the simulations demonstrate that (1) AST can prolong the lifetime and (2) RAG makes a better trade-off between storage and aggregation ratio, comparing to other aggregation schemes.

1. Introduction

Wireless sensor networks, as a promising technology, have been attracting more and more interest from the research community for a broad range of applications. Typical WSN applications involve lots of distribute-deployed sensors and one or more base stations (each of which is called the sink). Each sensor can sense, collect, and transmit information to the sink periodically or in events-driven style.

However, WSN is challenged by its limited capacity of communication and computation, limited power supply, and high vulnerability to failures. Among all the challenges, power supply is a big issue, especially under the wretched and inaccessible network environments. Thus, it is critical to save energy and collect data efficiently and correctly. On the ground, it has attracted much attention to design communication protocols for topology management and in-network aggregation [1–3].

The design of in-network aggregation [4–8] is a well-studied energy efficient data collection method and has been applied in other cases [9]. This owes to the following two aspects. First, if sensors send their original readings directly to the sink without local process, it will usually occupy excessive bandwidth and consume too much energy because of the redundancy in these original readings. Second, what the applications need usually are not the original sensor readings but the processed data. These two aspects make in-network aggregation possible and necessary. The basic idea of in-network aggregation is that redundant and irrelevant data are discarded and the relevant data are fused into an aggregation result at interior nodes along the transmission paths. Thus, data size and traffic are reduced, which will save considerable energy.

2. Related Work

In WSNs, data aggregation is closely related to the network structure, such as tree-based [10–12], cluster-based, and hybrid structures [8]. In particular, the tree-based structure has attracted many researchers and different tree-based aggregation algorithms have been proposed to both reduce energy consumption and maximize the lifetime [13–15].

Paper [10] proposes an aggregation tree construction/reorganization algorithm that minimizes such desired metrics as energy consumption. By calculating and sending a small set of intuitive statistics, a father node may be substituted by one of its brother nodes based on attachment cost.
Both [11, 12] propose the so-called life-time preserving tree (LPT). While building the aggregation tree, LPT chooses nodes with higher residual energy as aggregating fathers. In E-span tree (energy-aware spanning tree) [12] algorithm, each node uses two measurement to choose its father. The principle one is the depth nodes located in the spanning tree, and another is the residual energy.

Different from E-span, EE-span (energy efficient spanning tree) algorithm in [13] uses nodes’ residual energy as the main parameter and the distance as the complementary. That is, the selected father node has the most energy and the distance is reasonable. Recall that the transmission power is directly related to the distance. So, EE-span in fact takes the average path's energy as a new parameter when choosing father nodes.

AEE-span, an automata-based energy efficient spanning tree proposed in [14], uses learning automata to dynamically compute the probability that a node can be selected as a father node according to environment information such as distance or residual energy. Then, the node with the biggest probability is selected as a father node.

Whatever measurements are used, all strategies above have two things in common. Firstly, a node cannot determine its father until it collects all the environment information from all its neighbors and selects the one who fits the measurement best. This may lead to a big problem. The so-called best node is prone to be selected by lots of nodes in the same area, which makes this node a kind of bottleneck since it is responsible for data aggregation and transmission for all its huge children. Secondly, they will rebuild the aggregation tree periodically, which is energy-consuming and time-wasting [15].

Another important issue is how to aggregate sensors’ data via the constructed tree. One simple but popular aggregation operation is TAG illustrated in [2]. Firstly each leaf node in the tree reports data to its parent; then interior nodes aggregate data of their children and report the aggregated data to their parent. Obviously, TAG has only one single path for a node to transmit data for itself and its children if having some. Thus, if an interior node fails to communicate with its parent, all the data collected by it and its children cannot be sent to the root.

To sum up, while using tree-based structure to aggregate data, both the tree construction and updating and the aggregation scheme need improvement to achieve a robust and efficient data collection in WSNs. This is also the scope/aim of our paper. The main contribution of this paper lies in the following three aspects. (1) We let each node set a random waiting time to determine its father node via a specific criterion. According to this strategy, we can avoid that a favorable node has lots of child nodes and then becomes a bottleneck. (2) By the strategy of setting up alternative father nodes, any father node can be replaced by more occupied node asynchronously. Thus, no tree rebuilt is needed. Our tree construction and update strategies are called AST hereafter. (3) Enlightened by the multipath transmission idea of RING in [16], we propose RAG, a redundant aggregation scheme, to achieve flexible and efficient data aggregation and transmission in the tree structured network.

3. Network Models and Definitions

3.1. Network Models. In this paper, we assume that certain amounts of sensor nodes are uniformly deployed in an \( m \times n \) square area. Sensor nodes have the following characteristics: (1) in the network, all nodes are stationary after being deployed; (2) all nodes are of different amount of initial power and the sink has infinite power; (3) the sink can be deployed anywhere in the monitoring area; (4) all nodes have the same sensor radius, denoted as \( R \).

We adopt query-style aggregation strategy like that in [2]. The aggregation process includes two important phases, the query distribution phase and the data collection phase. During the query distribution phase, the sink generates and broadcasts tree construction message (msg\_GTD, as illustrated in Section 3.2) in which the query is included. Then, this msg\_GTD will flood through the network to construct the aggregation tree. In order to focus on aggregation operation, we simplify the model and assume that loose synchronized time and assigned aggregation interval are used as in [16], and these ensure each father node receives all children’s aggregation results and its original sensor readings. During data collection phase, each node’s aggregation interval is the same and is divided into 3 subintervals which will be illustrated in Section 5.

The collection phase starts from leaf nodes and the aggregation is carried out at each interior node along the aggregation tree interval by interval. Finally, an aggregation result is generated at the sink.

3.2. Related Definitions and Abbreviations. As shown in Table 1, we define a series of entities as functions, messages, or data structures necessary for accomplishing our algorithms. Prefixes data, msg, and func. in entity’s name are used to illustrate that the entity is a data structure or a message or a function, respectively. Each sensor node maintains not only its attributions but also the information about its parent or alternative parents. Nodes use the data structure entities to maintain all of its necessary information and use the function entities to process the data or maintain the tree structure and use the message entities to transfer messages in network.

A node has such attributions as ID, LEVEL, fID, fLEVEL, pos\_X, pos\_Y, fDIS, and so forth. ID and fID indicate the node and its father’s ID, respectively. LEVEL and fLEVEL indicate in which level the node and its father are located in the spanning tree. pos\_X and pos\_Y are the positions of this node. fDIS indicates the distance between this node and its father. Now, let us give the detailed structure of the message, msg\_GTD and msg\_DATA, which is very important in the procedure of tree construction. msg\_GTD is generated by a node and contains at least six important attributions of the node, that is, ID, LEVEL, fID, fLEVEL, pos\_X, and pos\_Y. The detailed definitions of all the functions are shown in Table 2.

4. Adaptive Spanning Tree (AST)

4.1. Tree Construction. To build an adaptive spanning tree, we also let the sink initiate the construction process and flood
the msg_GTD among the network. Different from others, every node uses asynchronous random wait mechanism to determine the ultimate father node. Also, we will collect all the alternative father nodes and store them in the set of alternative father nodes (data_FS) for every node. Algorithm 1 shows the detail of AST.

Here goes the exploration of the asynchronous random wait mechanism. When a node, saying \(N_i\), hears the msg_GTD for the first time, it selects the sender of this msg_GTD as its undetermined father and sets its LEVEL according to the level information in the msg_GTD. Rather than rebroadcasting the msg_GTD immediately, \(N_i\) will set a random time interval to wait. During this time interval, \(N_i\) will collect all msg_GTD messages from its neighbors if any and select a more favorable one. Specifically speaking, if \(N_i\) receives an msg_GTD from another node (we call this node \(N_i\)'s new possible father), it will replace the current undetermined father by the new possible father only when it satisfies the following two conditions: (1) the distance to the new possible father is shorter than that to the undetermined father; (2) the distance to the new possible father is less than half the sensor radius. When the random waiting time is out, \(N_i\) sets the current undetermined father as its father and updates and forwards the msg_GTD. In this process, \(N_i\) will update its alternative father nodes set, that is, data_FS, whenever it receives an msg_GTD which will be illustrated more specifically later.

Given the connectivity diagram in Figure 1, we illustrate the proposed AST algorithm and compare the differences between different algorithms. There are \(N_1\) to \(N_8\), and their residual energy is 10 J, 8 J, 6 J, 3 J, 4 J, 7 J, 9 J, and 8 J. Suppose \(N_1\) broadcasts msg_GTD and \(N_6\) is going to determine its father node. Now, let us explore how to construct spanning trees by different algorithms.

The TAG constructs the simplest tree (as in Figure 2(a)). When \(N_2\), \(N_3\), \(N_4\), and \(N_5\), neighbors of \(N_1\), receive this msg_GTD and select \(N_1\) as their father, then \(N_6\) receives an msg_GTD from \(N_3\). So, \(N_6\) chooses \(N_3\) as its father immediately. E-span tends to find a node that is nearer to sink as
father node and EE-span tends to find a node that is of the most average path's energy as father. E-span and EE-span continuously compare the nodes from which they receive msg_GTD to decide the most qualified father during the whole tree building period. So, \( N_6 \) compares \( N_2, N_3, \) and \( N_7 \) to decide which node is its father according to the respective criterion. The final spanning tree of E-span and EE-span can be seen in Figures 2(b) and 2(c). Using AST, \( N_6 \) firstly receives an msg_GTD from \( N_2 \) and \( N_6 \) waits a random time interval. During this interval, \( N_6 \) receives an msg_GTD from \( N_3 \), because \( D_{36} > D_{36} \) and \( D_{36} < R/2 \) (here, \( D_{ij} \) refers to the distance between \( N_i \) and \( N_j \); \( R \) is the sensing radius of each node). \( N_5 \) is selected as father and the random waiting of \( N_6 \) is over. Figure 2(d) shows the spanning tree constructed by AST.

4.2. Tree Maintenance. Another critical feather of AST is the local adjustment strategy that is carried out when finding a failure father. Recall that each node, say \( N_j \), sets up its alternative father nodes and stores them in its data_FS during the tree building period. During the random waiting time, if \( N_j \) hears an msg_GTD from the nodes with the same level as \( N_j \)'s father or \( N_j \) itself, \( N_j \) will add the msg_GTD's sender to its data_FS. We constrain \( N_j \)'s alternative father nodes level to avoid circles in the tree. Moreover, since these nodes are relatively closer to the sink node than those in the child level, the times for transmitting data of \( N_j \) will be smaller if they are set as a father node. This data_FS will be maintained during the whole lifecycle of WSN whenever the local adjustment strategy is used to update the spanning tree.

Take Figure 3(a) as an example. \( N_2, N_3, \) and \( N_6 \) are in the same level, while \( N_5 \) and \( N_7 \) are from \( N_2 \)'s father level, and \( N_7 \) is \( N_5 \)'s father. \( N_2 \) adds \( N_3, N_5, \) and \( N_6 \) to its data_FS when constructing the AST. That is, \( N_2 \)'s alternative father nodes include \( N_3, N_5, \) and \( N_6 \).

In WSN, message sending behavior of a node is a kind of its heartbeat which can be heard passively by its neighbors. Thus, during the lifecycle of the network, AST uses this passive heartbeat detecting strategy to update nodes' data_FS and detect father node's status. For a node \( N_j \), if it cannot hear a passive heartbeat from its alternative father, say \( N_j \), for a period of time such as \( l \) successive aggregation intervals, \( N_j \) may think \( N_j \) is a failure and remove it from the alternative

Figure 2: Aggregation trees of different algorithms.

Figure 3: Alternative father nodes illustration.
father set $data_{FS}$. Similarly, if $N_i$ finds that its father node is a failure, it will be replaced by selecting a favorable alternative node from its $data_{FS}$ (using function $func_{SFA}$) according to the residual energy.

Now, we will use an example to illustrate this strategy. As in Figure 3(a), if $N_2$ can hear the passive heartbeat from $N_3$ during an aggregation interval, $N_3$ records that, at this aggregation interval, $N_7$ is in a normal status; else, $N_7$ records that $N_7$ is abnormal. $N_2$ will record the latest $n$ statuses of each alternative father node. If spare father node’s latest $n$ statuses are all abnormal, they indicate that this alternative father node is broken. It is deleted from the $data_{FS}$. If $N_2$, $N_3$’s current father node, is thought to be a failure, apart from deleting it from $data_{FS}$, $N_2$ will also select a new father from its $data_{FS}$ based on the residual energy. That is, the node in $N_2$’s $data_{FS}$ with the biggest residual energy will be set as the new father of $N_2$. We assume that the residual energy of $N_3$, $N_6$, and $N_7$ is 5 J, 6 J, and 10 J, respectively. Then, $N_3$ will be selected to be the new father. We can see that even if $N_1$ dies its child node $N_9$ and $N_2$ can locally find a replacement, the connectivity of network is maintained (Figure 3(b)). This strategy is asynchronous and distributed for sensor nodes. Unlike other current spanning tree construction algorithms, periodical reconstruction of aggregation tree is not needed in AST. So, it is more flexible and energy-saving.

5. Redundant Aggregation Scheme

In this section, we propose redundant aggregation scheme for tree structure. This scheme changes the single-path transmission of the traditional tree aggregation scheme and allows the nodes to transmit data for their siblings. In our aggregation scheme, when a nondestination node receives a message, it firstly checks if this message is from its siblings. If so, it will store and forward this message to its father node. This forms our redundant aggregation scheme RAG. In RAG, nodes’ tasks in their data aggregation interval are illustrated as follows.

(1) Receiving Subinterval. In this subinterval, interior nodes have two tasks. Firstly, nodes receive the uploaded aggregation results from their children. Secondly, interior nodes detect and delete duplicated data to build the local results. For the leaf nodes, they do not have this subinterval.

(2) Sensing and Processing Subinterval. In this subinterval, interior nodes read their sensing data and aggregate it with the local results built in receiving subinterval and thus make the aggregation results ready for sending. For the leaf nodes, there is no aggregation process during this subinterval.

(3) Delivery Subinterval. This interval is divided into sending slot and forwarding slot. In the sending slot, nodes send their own aggregation results to their father nodes and receive the aggregation results from siblings, if they have, and store them for forwarding. Then, they forward the data for their siblings, if they have, in forwarding slot. The pseudocode of redundant aggregation is illustrated in Algorithm 2.

Now, we use an example to demonstrate the detail of RAG as shown in Figures 4 and 5. Before we explore RAG, we firstly illustrate the problems that the traditional tree aggregation schemes face. As can be seen from Figure 4(a), $N_1$, $N_2$, and $N_6$ are $N_4$’s children. $N_1$’s message is heard by $N_5$ and $N_6$, but $N_5$ and $N_6$ discard it because they are not $N_4$’s father node. Thus, there is only one single link between $N_1$ and $N_4$. If this unique link is broken, $N_1$’s message cannot be transmitted to $N_4$, which leads to the subtree leading by $N_1$ isolated from the network. Luckily, the messages deleted by brothers can be leveraged to form our RAG. When $N_5$ and $N_6$ receive the message whose destination is $N_4$, their father node, they do not delete this message but store it. Then, they forward it to $N_4$. So, in Figure 4(b), there are 3 different links that connect $N_1$ with $N_4$. Thus, even if the direct link between $N_1$ and $N_4$ is broken, there are two extra paths. Figure 5 illustrates RAG in more detail by time slot. To avoid collision, father allocates time slot for each child during sending slot and forwarding slot.

6. Simulation

6.1. Simulation Setup. Here, we evaluate the performances of AST and RAG separately. We compare AST with another three popular tree structures, TAG, E-span, and EE-span, by simulation. For the aggregation schemes, we compare RAG with TAG and RING.

In all our simulations, we uniformly deploy a number of sensor nodes whose sensing radius is 4 m on a 25 m × 25 m square area and a sink node in the center of the simulation area. We vary the number of nodes from 100 to 500, with...
Sink node
(a) A traditional tree aggregation scheme
(b) Redundant aggregation scheme

Figure 4: Illustration of redundant aggregation.

Figure 5: Redundant aggregation on each time slot.

1 2 3 4 5 6
Sink node

The increment step of 50. When setting up the simulation environment, we initialize every node with energy randomly selected from 800 J to 1000 J and also suppose the sink has infinite energy. We suppose that the energy consumption for nodes to process data is negligible, and the energy consumption for message receiving and transmitting satisfies formulas (1) and (2) in [13] with $C_1$ and $C_2$ set to 1 and 5, respectively, in our simulations.

To make the analysis more reasonable, we conduct 500 independent simulations to get an average result in all our analysis.

6.2. Simulation-Based Analysis of AST. We compare the network lifetime, average levels, and number of alive nodes among AST, E-span, EE-span, and TAG. As can be seen from previous illustration, in the tree-based aggregation WSN, the system runs round by round. A round is one complete cycle for collecting sensor readings, aggregating data, and transmitting the aggregated data to sink. Namely, a round ends when the sink receives aggregation results from all its alive child nodes. We also define that the network is failure if 5% nodes fall failure as in [13]. Thus, the number of rounds before the network is a failure indicates the network lifetime.

We define the level of node which means the depth a node locates in the spanning tree. The level of node indicates the number of nodes along the path between the sink and the node itself. We further define average level as the mean of all levels of all nodes in the network. It is obvious that a less average level means that the average degree [17] of the spanning tree is bigger and that interior nodes averagely have more children.

In Figure 6(a), the average levels of AST and EE-span are crossed, and their levels are higher than the levels of TAG and E-span. So, the average degrees of AST and EE-span are smaller, which indicates that they averagely have less child nodes per father node than TAG and E-span. Any interior node with less child nodes has less communication requirement per round. Recall that communication is the major factor of energy consumption. Thus, the network lifetime of AST and EE-span are more likely to live longer than TAG or E-span; hence, the entire network lifetime of them is bigger, which coincides with Figure 7.

Figure 6(b) shows the trend of the network lifetime. The lifetime of AST can be prolonged when adding nodes to the network, while E-span and EE-span change a little. As illustrated in Sections 1 and 4, unlike AST which only uses physical distance among nodes as the most important issue, EE-span and E-span take the residual energy as an important issue while constructing the spanning tree. Thus, EE-span and E-span may select the most powerful nodes as a father node. But the physical distance between father and its child node may be large which hence consume more energy during the collection phase. That is to say, AST relies more on the physical distance among nodes. While adding nodes to the network, the average distances among nodes decrease, and the average energy consumption for nodes in AST will decrease significantly. The decrease in energy consumption further leads to an improvement of the network lifetime. In TAG, nodes determine their father nodes immediately when they receive an msg_GTD. This simple strategy in TAG causes the nodes near the sink to carry more and more child nodes while adding nodes to the network, which make these nodes a kind of bottleneck. Thus, while adding nodes to the network, the lifetime of TAG decreases contrarily. To help understand the lifetime trend, we also show how the number of alive nodes varies during the network lifetime in Figure 6(c).

To illustrate the distribution of the number of alive nodes, we deploy 500 nodes in simulation area and record how many alive nodes exist in each round. Figure 6(c) shows an average result of 500 independent experiments which demonstrates the superiority of AST.

6.3. Simulation-Based Analysis of RAG. RAG and TAG and RING are aggregation schemes to aggregate and transmit data
The contribution ratio, a decimal fraction in interval $[0,1]$, is the ratio of the number of nodes contributing to the aggregation to that of all alive nodes under different package loss. The bigger the contribution ratio is, the more effective the aggregation process is. Communication overload means the total communication cost needed to complete data aggregations per round, which is represented by the energy consumption. Storage overload indicates the average storage occupancy per node per round during the aggregation process, which is calculated by dividing the number of packages into the number of nodes alive in the network. The lower the communication or storage overload is, the better the aggregation scheme performs. We try to study how these three parameters vary with the package loss which indicates the network condition. To make the results more reasonable, we simulate 500 times independently to get the average, and the number of nodes in all these simulations is 500.

As shown in Figures 7(a) to 7(c), these three parameters of all the schemes decrease while the package loss increases. That is because a bigger package loss indicates worse network status which means more nodes failure to fulfill the aggregation. Thus, there are less data transmissions in the network.
which further need less communication requirement and storage occupancy.

Now, let us explore the differences among different schemes under the same network status, namely, the same package loss here. Figure 7(a) indicates that the aggregation contribution ratio of RING is the highest while TAG's is the lowest. This owes to the multitransmissions of the aggregation result for most nodes in RING and RAG. Between RING and RAG, the former has real multipaths from every node to the sink, while the latter only allows the multiple forwards between siblings. That is why RING's aggregation contribution ratio is higher than RAG's. Despite achieving the highest aggregation contribution ratio, RING pays a high cost of the communication requirement and storage occupancy, as shown in Figures 7(b) and 7(c). Data transmission in TAG is along the aggregation tree, only each pair of father and child node send or receive, store, and forward aggregation results between each other. Thus, the average communication cost and storage requirement of TAG's is the minimal, comparing to RING and RAG. To sum up, RAG makes a great balance among network lifetime and aggregation efficiency, which is supposed to be more promising.

7. Conclusion

In this paper, we focus on tree-based aggregation and make improvements in terms of aggregation tree structure and the aggregation scheme itself to achieve an efficient but
flexible data collection for WSNs. Firstly, we illustrate the AST algorithm, which can adaptively construct and adjust an aggregation spanning tree. By adopting random waiting and alternative father nodes strategies, the spanning tree built by AST is relatively balanced and can be adjusted locally. The lifetime of AST is longer than other tree structures according to simulation.

Secondly, we propose a redundant aggregation scheme, RAG. RAG combines the advantages of traditional tree-based and RING’s aggregation scheme. Allowing interior nodes to help forwarding data for their siblings, RAG assures a reliable data transmission for WSNs. In fact, RAG can be applied to any tree structures, for example, E-span and LPT. Simulation shows that the RAG owns better average performance in terms of communication overload, storage overload, and aggregation contribution ratio, comparing with RING and TAG.

In the future, by using the probability distribution of distance between nodes in 2-dimensional space and the studies of node degree mentioned in [17], we will further explore the relationship between network lifetime and nodes distance and quantitatively investigate node degree’s influence on network lifetime in in-network aggregation field. Meanwhile, it is also promising to employ matrix analyzing techniques such as collaborative filtering and matrix factorization [17,18] to analyze the obtained distance matrix for more detailed information regarding the structure of the given network.

Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this work.

Acknowledgments

This work was supported by China’s Natural Science Foundation (61173009), Science Foundation of Shenzhen City in China (JCYJ2014050915097445), Project of the State Key Laboratory of Software Development Environment (SKLSDE-2014KF-01), and the Fundamental Research Funds for the Central Universities.

References

[1] C. Hua and T.-S. P. Yum, “Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks,” IEEE/ACM Transactions on Networking, vol. 16, no. 4, pp. 892–903, 2008.
[2] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “TAG: a tiny aggregation service for Ad-hoc sensor networks,” in Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI ’02), pp. 131–146, Boston, Mass, USA, December 2002.
[3] A. A. Abbasi and M. Younis, “A survey on clustering algorithms for wireless sensor networks,” Computer Communications, vol. 30, no. 14-15, pp. 2826–2841, 2007.
[4] R. Rajagopalan and P. K. Varshney, “Data-aggregation techniques in sensor networks: a survey,” IEEE Communications Surveys and Tutorials, vol. 8, no. 4, pp. 48–63, 2006.
[5] J. Zhang, X. Jia, and G. Xing, “Real-time data aggregation in contention-based wireless sensor networks,” ACM Transactions on Sensor Networks, vol. 7, no. 1, article 2, 2010.
[6] L. Krishnamachari, D. Estrin, and S. Wicker, “The impact of data aggregation in wireless sensor networks,” in Proceedings of the 22nd International Conference on Distributed Computing Systems, pp. 575–578, Vienna, Austria, July 2002.
[7] X. Tang and J. Xu, “Optimizing lifetime for continuous data aggregation with precision guarantees in wireless sensor networks,” IEEE/ACM Transactions on Networking, vol. 16, no. 4, pp. 904–917, 2008.
[8] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, “In-network aggregation techniques for wireless sensor networks: a survey,” IEEE Wireless Communications, vol. 14, no. 2, pp. 70–87, 2007.
[9] H. Jiang, J. Cheng, D. Wang, C. Wang, and G. Tan, “Continuous multi-dimensional top-k query processing in sensor networks,” in Proceedings of the 30th IEEE INFOCOM, Shanghai, China, April 2011.
[10] A. Deligiannakis, Y. Kotidis, V. Stoumpos, and A. Delis, “Building efficient aggregation trees for sensor network event-monitoring queries,” in Proceedings of the 3rd International Conference on GeoSensor Networks, pp. 63–76, Oxford, UK, July 2009.
[11] W. M. Lee and V. M. S. Wong, “LPT for data aggregation in wireless sensor networks,” in Proceedings of the Global Telecommunications Conference, vol. 2, Louis, Mo, USA, December 2005.
[12] W. M. Lee and V. W. S. Wong, “E-Span and LPT for data aggregation in wireless sensor networks,” Computer Communications, vol. 29, pp. 2506–2520, 2006.
[13] Z. Eskandari, M. H. Yaghmaee, and A. H. Mohajerzadeh, “Energy efficient spanning tree for data aggregation in wireless sensor networks,” in Proceedings of the 17th International Conference on Computer Communications and Network, St. Thomas, Virgin Islands, USA, August 2008.
[14] Z. Eskandari, M. H. Yaghmaee, and A. Mohajerzadeh, “Automata based energy efficient spanning tree for data aggregation in wireless sensor networks,” in Proceedings of the 11th IEEE Singapore International Conference on Communication Systems (ICCS ’08), pp. 943–947, Guangzhou, China, November 2008.
[15] P. Sharma and P. S. Mandal, “Reconstruction of aggregation tree in spite of faulty nodes in wireless sensor networks,” in Proceedings of the IEEE 6th International Conference on Wireless Communication and Sensor Networks, Allahabad, India, December 2010.
[16] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, “Synopsis diffusion for robust aggregation in sensor networks,” in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys ’04), pp. 250–262, Baltimore, Md, USA, November 2004.
[17] X. Luo, Y. Xia, Q. Zhu, and Y. Li, “Boosting the K-Nearest-Neighborhood based incremental collaborative filtering,” Knowledge-Based Systems, vol. 53, pp. 90–99, 2013.
[18] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, “An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems,” IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1273–1284, 2014.
