J Supercomput (2017) 73:5105–5124
DOI 10.1007/s11227-017-2072-0

Scalable structure-free data fusion on wireless sensor networks

Mahnaz Koupaee[1] · **Mohammad Reza Kangavari**[2] · **Mohammad Javad Amiri**[1]

Published online: 16 May 2017

Abstract Recent advancements in sensor technology, wireless networks and consequently wireless sensor networks and the increase in their applications in different fields have led to their great importance. One of the most important challenges of such networks is the distributed management of the huge amount of data produced by sensors in network to reduce data traffic in network and minimize the energy consumption. In this research, a distributed, dynamic fusion algorithm is introduced. Since the proposed method is dynamic, the number of neighbors sending data to a node is not known in advance. So in order to increase the chances of different data to meet, the node waiting time is calculated. By the end of waiting time, the node performs data fusion and sends the fused data to the best neighbor chosen by the proposed best neighbor algorithm. This procedure continues until data reaches the sink. The proposed algorithm, while being scalable and convergent, outperforms similar methods in terms of number of transmissions, traffic load and energy consumption.

Keywords Wireless sensor network · Data fusion · Network graph · Temporal and spatial convergence

1 Introduction

Wireless sensor networks cause a lot of challenges; according to the inherent limitations of sensors and the environments, they are deployed in [1,2]. One of the main challenges of these networks is the limitation of available energy due to the low

✉ Mahnaz Koupaee
koupaee@cs.ucsb.edu

1 Department of Computer Science, UC Santa Barbara, Santa Barbara, CA, USA

2 School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran

capacity of batteries embedded in sensors. Since the sensors are usually deployed in environments in which they cannot be accessed, the batteries cannot be replaced or recharged [3]. Therefore, the lifetime of a wireless sensor network is highly dependent on the lifetime of sensors; thus, saving energy of sensors can expand the lifetime of a network [5,6]. The main responsibilities of sensors are divided into three parts: sensing, processing and communication. The communication part, including sending and receiving the packets, consumes the largest portion of available energy in a sensor [3]. There have been a lot of attempts to reduce the energy consumption and prolong the network lifetime [7]. Meanwhile, the number of sensors deployed in an environment is increased because of possible errors in sensors and the necessity to cover an area both spatially and temporally [8,9]. This causes the scalability issue in wireless sensor networks. So a mechanism to reduce this amount of data generated by numerous sensors is needed [10,11].

In order to manage the huge amount of data and reduce the energy consumption, a lot of methods were intended to take advantage of the sensor computation capacity and perform the desired fusion algorithm while data are being routed toward the sink [12,13]. Data fusion can be perceived as a set of automated methods of combining the data that comes from many sensor nodes into a set of meaningful information [3].

Wireless sensor networks applications can be divided in three major groups: the first group, time-driven networks, is used to monitor environmental conditions. In these applications, sensors constantly send their data to the sink. The event-based networks are responsible to target an event and send information about it to the sink. The final group is responsible to answer user's queries which are applied to the network. The sensors which can answer the query send their data to the sink. This group is referred to as query-based networks [6]. The proposed methods in the literature can be categorized into two main structured groups: tree-based approaches and cluster-based approaches. These methods perform well in data gathering applications where the error rate of sensors is low, the traffic patterns and the topology of the network do not change due to sensor failures or adding new sensors to network. However, in event-based networks where source nodes change with the event and in dynamic scenarios where the nodes may be added or removed from the network, the construction and maintenance of a structure may be too expensive in terms of both energy and traffic load in network [14]. Figure 1 shows the poor performance of tree-based and cluster-based approaches in event-based networks. The black circles in the cluster-based network represent the cluster heads. If the head cannot reach the sink, it uses other nodes for transmission. Although the nodes sensing the event are spatially close to each other, they are not able to be fused before reaching the sink. There are also some applications in which there are different types of data. A static fusion algorithm cannot be optimal while a dynamic approach can fuse the same-type data as early as possible on its way to the sink. So the fusion algorithm is performed sooner and the number of transmissions is reduced. Wireless sensor networks studied in this paper are event-driven. They include high number of sensors with different types placed in a dense manner. Sensors wait until an event happens. As soon as an event occurs, sensors which are in the event area sense the data regarding the event.

In this paper, in order to overcome the problems mentioned for structured approaches, a method for fusing data in event-based networks is proposed. This

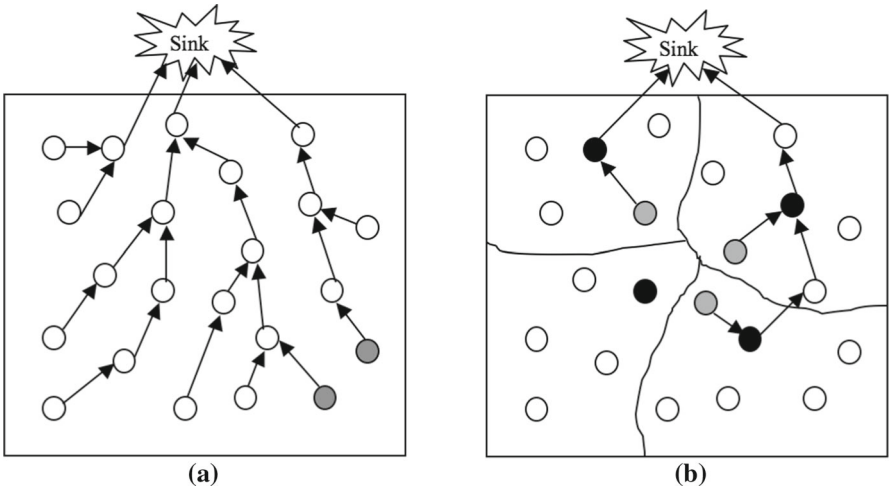

Fig. 1 Poor performance of structured approaches: **a** tree-based; **b** cluster-based

dynamic approach is intended to be scalable, that is, to be applicable to large number of sensors of order of thousands or millions [3] and adaptable to topology changes in order to reduce the network traffic and energy consumption and prolong network lifetime. The main contributions of this paper are described as follows:

- A waiting time is assigned to the sensors containing data. As nodes do not know how many neighbors send data to them, they cannot explicitly wait for the data from neighbors. Using this waiting time improves the chances of the same-type data meeting at intermediate nodes.
- An algorithm called best neighbor (BN) algorithm is proposed to dynamically choose the next hop to send data to. Introducing some attributes for each sensor can help the sensor choose the best neighbor.

The rest of this paper is organized as follows. Section 2 reviews the related work in wireless sensor networks and data fusion. Section 3 explains the main elements of the proposed method. The evaluation of the proposed method is described in Sect. 4, and finally, Sect. 5 concludes the paper.

2 Related work

The main challenges in wireless sensor networks include fault tolerance, scalability, adaptability to topology changes and energy consumption [11]. In-network data fusion can be used to overcome the scalability issue and reduce network traffic in a distributed manner. Performing data fusion in this way can reduce the communication cost by aggregating same-type data and performing application dependent fusion before reaching the sink. It therefore reduces the energy consumption [15]. The most popular distributed paradigm in wireless sensor networks is in-network aggregation. The main idea of this method is to use the computational capacity of sensor nodes to

perform data fusion while routing data to the sink. This paradigm is also known as data-centric routing [4].

These methods are mainly divided into two groups: tree-based approaches and cluster-based approaches [16]. In the first group, a routing tree is constructed rooted at the sink. The data are forwarded from source nodes to the sink. The child nodes send their data to their parents. Each parent performs fusion after receiving all data from its children and then sends the result to its parent [15–17]. In the second group, the nodes are divided into clusters. A node acts as a cluster head; it receives data from cluster members, performs fusion and sends data to the sink.

2.1 Tree-based approaches

One of the early methods proposed is TAG framework [18]. In this method, users send the queries to the nodes. The nodes which can answer this query send their data through the routing tree rooted at the sink to the user. Fusion occurs at intermediate nodes while data are moving toward the sink. There is another method called EADAT in which sensors with higher residual energy are chosen as non-leaf nodes in order to prolong the network lifetime [19]. In another method called cascading timeout [20], the nodes in the routing tree choose their waiting time according to their distance to the sink; so, the nodes further from the sink consider lower waiting time. A tree-based data aggregation method using centralized method to manage data is introduced in [21].

2.2 Cluster-based approaches

One of the most famous protocols in this group is LEACH [22]. Cluster heads are randomly elected in order to distribute the energy consumption evenly among all nodes in the network. The cluster head is responsible to receive data from all cluster nodes and fuse it. The result is directly sent to the sink. LEACH-C is an improvement made on original LEACH to improve its efficiency [23]. In this protocol, the procedure of choosing cluster heads is done with the help of the sink. Another group of these methods are chain-based algorithms. PEGASIS [24] and Hierarchical PEGASIS [25] are the most famous ones. PEGASIS is based on the following assumptions: The sink is located further away from the sensors, and each node is able to perform perfect aggregation on its own data and the received data from its neighbor. This algorithm uses a greedy method to construct a chain starting from the furthest point in the network. A node is randomly chosen as the head. The head node is responsible to fuse the whole data received from nodes and send it to the sink. In order to balance the energy consumption, each node will perform as the head during network lifetime. Hierarchical PEGASIS is an improvement on PEGASIS. In this algorithm, there is the possibility of parallel transmission of data. The last node which receives data becomes the head node. There were also some other improvements on PEGASIS to reduce transmission delay and improve the chances for parallel transmission of data [26].

The algorithms in this group assume that all nodes in the network are possible to reach the sink in one hop. This assumption limits the size of network. Moreover, if there is no possibility to fuse data in the head node, it must send a lot of packets toward the sink. This may cause the head die soon because of the high amount of energy used for transmission [17].

Tree-based and cluster-based approaches perform well in stable environments in which the nodes work all the time. But in practical environments where the nodes may stop working or some nodes may be added because of application requirements, and in event-based networks, the cost of construction and maintenance of a structure is high. Therefore, constructing a structure cannot be beneficial.

2.3 Structure-free approaches

Based on the challenges issued for structured approaches, structure-free methods are intended to fuse data using local information of the nodes. The first attempt to propose a structure-free protocol is introduced in [17]. There are two mechanisms used in this protocol to improve chances of aggregation: randomized waiting time to improve temporal convergence and data-aware anycasting to make data more spatially convergent. The early aggregation only occurs when the nodes closer to sink choose longer delays. To overcome the issues of randomly assigning waiting times to the sensors, a dynamic mechanism to fuse data is introduced in [27]. To converge packets spatially, the potential field theory is used and cascading timeout policy [20] is used to delay packets at intermediate nodes to improve chances of same-type packets meeting at a node. However, this waiting time policy may cause long delays for sensors in large-scale networks. RAG is a real-time structure-free protocol introduced in [17]. Long delays are prevented by calculating a waiting time based on the location of a node and its distance to the sink. To improve chances of early fusion, waiting time is assigned to packets so that packets are delayed at intermediate nodes according to their distance to sink. When the waiting time is over, the packets are aggregated and forwarded to the next hop. Data-aware anycasting is also used to route packets forward. An attribute-aware aggregation method is introduced in [28] which can support data from different types and different applications. It also applies the potential field theory to spatially fuse packets. An adaptive timing algorithm is used to delay packets at intermediate nodes to make fusion process more effective by reducing the number of transmissions.

2.4 Structured approaches versus structure-free approaches

Table 1 compares these two approaches by naming their advantages and disadvantages. The first positive point of structured approaches is their speed to deliver fused data to the sink. Since the destination of each node is known beforehand, the data are sent immediately to the destination which is known in advance. For monitoring applications, having a static structure makes the fusion process faster. In these applications, sensors contain data all the time. So having pre-known routes to forward data will reduce the response time and traffic load by performing in-

Table 1 Structured approaches versus structure-free approaches

Disadvantages	Advantages
Structured approaches	
The overhead of construction and maintenance of a structure	High speed in transmitting data to the sink
Being static so expensive to adapt them to network changes	Suitable for monitoring applications and static networks or networks with low change rate
Not suitable for event-based networks	
High energy consumption for transmitting non-fused data	
Structure-free approaches	
Lower speed of transmission since the routes are chosen dynamically	Reducing the number of packets by early fusion of data while being forwarded to the sink
Higher delay time and response time	Low overhead and its adaptability to the changes
	Fault-tolerant
	Suitable for event-based networks and dynamic networks with high rate of changes
	Routing data using only local information without a centralized mechanism

network fusion. On the other hand, structured approaches intend to construct a structure such as a tree or clusters before network starts its mission. Therefore, if the change rate of network is high due to sensors failures or changes in their positions, the structure needs to be reconstructed which may increase the energy consumption. In event-based sensor networks, here are only some nodes sensing the event; so, having a static structure is not suitable as previously mentioned in Sect. 1.

Considering the mentioned problems for structured approaches and the need for a fusion method regarding the special characteristics of event-based sensor networks, there are some efforts designing structure-free approaches. The structure needs to be dynamic since for each event, different set of nodes contains the data . Thus in these approaches, according to the local information of the sensors, its destination is determined without any centralized management. Being dynamic enables spatially convergent data to be fused close to the event. This causes reduction in network traffic. Since structure-free approaches are dynamic, they are flexible to changes; so, sensors failures or changes in network topology do not reduce the performance of the network and the maintenance can take place easier without high cost. On the other hand, lack of a structure and therefore not knowing the destinations in advance may cause longer delays because each time each sensor needs to choose its destination for fusion and transmission.

3 Proposed method

In this section, the method is described. The algorithm is intended to organize heterogeneous sensors and perform fusion on them under two conditions: First, the performance of network should not be influenced by sensor failures or by adding some sensors due to application requirements and second, by performing fusion algorithm in convenient time and place, it is able to reduce the volume of data toward the sink so the energy consumption is reduced and the network can exist longer.

3.1 Deploying sensors in the environment

Sensors will be deployed in the environment of study. The deployment can occur in different ways, but since the nodes are usually placed in fields where they cannot be reached easily, there is no pre-planned organization of sensors. Therefore, they are scattered randomly in the environment. The sink node is responsible to gather the network data, perform final processes and take the final decision before delivering it to the user. Sensor nodes are aware of the location of the sink.

3.2 Occurrence of events and sensors sensing it

Nodes in the network are aware of their one hop neighbors. Each node is aware of its neighbor location and its type. These attributes of a node are later used to find the best neighbor as destination.

3.3 Calculating the waiting time

One of the most important parts of the proposed algorithm is the calculation of the waiting time. In order to fuse data while being transferred to the sink, temporal and spatial convergences are the two necessary conditions. These conditions reduce the traffic and energy consumption by lowering the number of packets in the network. To meet these two conditions, structured approaches let the nodes transmit their data to their parents (spatial convergence) and each parent waits for all its children to receive their data (temporal convergence). In the proposed method, since the destination of a node and the number of nodes going to choose it as destination are not known in advance, we need other mechanisms to realize the conditions mentioned. In this section, temporal convergence is studied and spatial convergence is the topic of the next section. To fuse more packets in a node, the waiting time of a node is defined as Eq. (1).

$$\text{WaitingTime} = \frac{\left(\left(\text{TotalDelay} - \left(\left(\frac{\text{DistanceToSink}}{\text{OneHopDistance}}\right) * \text{HopTime}\right)\right) + \left(\left(\frac{\text{DistanceToFurthestPoint}}{\text{OneHopDistance}}\right) * \text{HopTime}\right)\right)}{2} \tag{1}$$

In this equation, the waiting time is assigned to nodes rather than packets. Each node performs fusion on its own data and all data received from its neighbors after its waiting time. Parameters used in the equation are described as follows:

- *TotalDelay* The first parameter is the total delay accepted by the application before the arrival of the event data. To calculate this delay, we make use of the time it takes to deliver data from the furthest point of the network to the sink. So, instead of sending data immediately, we propagate this delay through network nodes so the chances of fusion increase. To calculate the total delay, we estimate the number of hops needed to transmit data from the furthest point, and then multiply it by one hop transmission time. We can also have some added time accepted by the application. The result is the total delay for an event before being reported.
- *DistanceToSink* The next parameter is the distance of a node to the sink. Since each node is aware of its location and is able to calculate its location to the sink, the distance can be calculated easily using this information. Dividing this parameter by one hop distance, the average number of remained hops to sink is calculated.
- *HopTime* The time needed for a packet to be transmitted one hop is calculated using the network conditions. The parameters considered to calculate this waiting time are time needed to transmit one packet, time needed to receive it, time to analyze the packet and time to fuse the packet. After calculating the remained hop count to the sink and multiplying it by hop time, the time needed to transmit data to sink is calculated. By subtracting this amount from total delay, we will have the time a node can wait before its transmission.
- *DistanceToFurthestPoint* The first part of the equation is showing the importance of the distance of each node to sink. There is yet another parameter which can be effective in calculating the waiting time. This parameter is an indicator of how many packets are going to choose a node as their destination. So a portion of time is dedicated to the packets to be received from further nodes. This makes the second part of the equation.

This part may increase the waiting time a lot especially when the event occurs close to the sink and away from the furthest point. Meanwhile this gives the confidence that a node will wait enough for the data from lower nodes before sending its own data. Another point is that when sensors are deployed densely, the distance to sink is not sufficient to make the desired difference among waiting times of different sensors because sensors close to each other will have similar waiting times; thus, the chances of performing fusion on their data decrease.

There is another way to calculate waiting time so that the result is lower than the previous method. The equation is presented as Eq. (2).

$$\mathrm{WaitingTime} = \frac{\left(\left(\mathrm{TotalDelay} - \left(\left(\frac{\mathrm{DistanceToSink}}{\mathrm{OneHopDistance}}\right) * \mathrm{HopTime}\right)\right) + \left(\left(\frac{\mathrm{DistanceToFurthestPoint}}{\mathrm{OneHopDistance}}\right) * \mathrm{HopTime}\right)\right)}{2} \Big/ \left|\frac{\mathrm{DistanceToSink}}{\mathrm{OneHopDistance}}\right| \tag{2}$$

This equation is produced through dividing by the number of hops to the sink. So the resulted waiting time is reduced. So the end-to-end delay is also reduced. This equation works well in networks where the sensors are not too close. So the resulted waiting time is acceptable by the application.

We assume that all sensors in the event area sense it simultaneously, so they tend to calculate waiting time immediately after the event occurs. During waiting time, sensors receive data from their neighbors and if possible perform fusion on the data. If a node does not have a neighbor which its distance from sink is more than the node itself, the waiting time is equal to zero. So the data are immediately transmitted to the next node. The reason for assigning zero to the waiting time is that there is no node to choose this node as destination. While an event occurs, only those nodes in the event area calculate waiting time. Others stay idle until they receive data from one of their neighbors. If the selected destination of a node is not in the event area, the node also transmits its own waiting time to the destination. The node that receives data calculates its waiting time and subtract from it the received waiting time. This procedure continues for all nodes chosen as destination but is not in the event area. The next part is performing fusion on data.

3.4 Performing fusion on data

Any sensor containing data tends to perform fusion after its waiting time. The first step to fuse data is to aggregate same-type packets. If there exists more than one packet with the same-type data, according to the application type, an aggregation function, such as MIN, MAX, AVG, COUNT or any other possible function, aggregates the data packets. There may also be the possibility to perform fusion decision to reach an intermediate representation. If sensor nodes are not able to perform high-order computations due to lack of computational power, or if it is not possible to fuse data in a representational form according to the application specifications, fusion only occurs on same-type data. Final fusion will occur at the sink.

3.5 Choosing the best neighbor as the next hop

After performing fusion on the available data in the node, the destination will be known using the local information of neighbors so that the data will finally reach the sink.

After the occurrence of an event, nodes in the event area are activated. When the waiting time is over, and after performing fusion, each node should choose its best neighbor as the destination. Since the proposed method is intended to be convergent while moving toward the sink, that is, data are intended to move closer to sink not further, nodes choose the destination among the neighbors which are closer to sink than the node itself. The waiting time is also defined in a way that nodes closer to sink have longer waiting times in comparison with nodes further away from the sink. This is achieved by subtracting the time needed for a packet to be transmitted to sink from current node, from possible total delay of network as stated in Eqs. (1) and (2). Nodes further send their data sooner. Now the next step is to choose the best node among the set of neighbors closer to sink. Network is modeled as a graph in which sensors

represent nodes and the connections between them are represented by edges. Each node has some attributes. The attributes include distance to sink and data type. Nodes choose their destination using the information provided by these features as follows:

Among all neighbors of a node, those which are closer to the sink than the node itself are the possible options. To find this set, since the node knows the location of its neighbors and the location of the sink, the euclidean distance can be calculated. Then neighbors are sorted in an ascending way according to their distance to the sink. Now it is time to check the data types. We start checking the neighbors for one which includes the same-type data. If there is one, we send the data to that node. If there is no node with the same type, we send data to the closest node containing data from any type. Finally, if none of the neighbors include data, the data are transmitted to the closest neighbor to the sink. The algorithm for choosing the next hop is Algorithm 1.

Algorithm 1 Choose Best Neighbor Destination Algorithm

Input: node n with data d and set V of its neighbors
Output: destination of node n

```
DataSent = false
Let V' ⊆ V be the set of neighbors that are closer than n to the Sink node.
Let v'_1, v'_2, ...., v'_k be an enumeration of V' such that v'_1 has the least and v'_n has the most distance to the Sink, respectively
Let v.type and v.data show the type and data of node v
for i ← 0 to k do                          ▷ nodes in V' which are sorted based on the distance to the Sink
    if v'_i.type = n.type AND v'_i.data ≠ null then
        send data d to v'_i
        DataSent = true
        Break                              ▷ destination is found, algorithm is done
    end if
end for
if DataSent = false then                   ▷ none of the neighbors in V' has same data type
    for i ← 0 to k do
        if v'_i.data ≠ null then
            send data d to v'_i
            DataSent = true
            Break                          ▷ destination is found, algorithm is done
        end if
    end for
end if
if DataSent = false then                   ▷ none of the neighbors in V' has data
    send data d to v'_1                    ▷ data are sent to closest node to the Sink
    DataSent = true
end if
```

The algorithm has three main steps. After it sorts all the neighbors of node n which are closer than n to the *Sink*, it checks all the neighbors based on the order; if the type of neighbor node is the same as the type of n and the neighbor has some data, algorithm sends the data to the neighbor and the job is done. In case no node is found in the first step, in the second step, algorithm checks for the node that have any data. Again it checks neighbor nodes in order and when a node with such property has been found, it sends the data to the node and the algorithm is finished. The third case happens when

Table 2 Structured approaches versus structure-free approaches

C	Data type	Spatial convergence	Assignment of WT	Temporal convergence
DAA+RW [22]				
–	Homogeneous sensors same data types	Data-aware anycast at MAC layer	Packet	Randomized waiting time
DASDR [23]				
✓	Homogeneous sensors same data types	Potential field theory	Packet	–
RAG [24]				
–	Heterogeneous sensors different data types	Data-aware anycast at MAC layer	Packet	Judiciously waiting time policy
ADA [25]				
✓	Heterogeneous sensors same data types	Potential field theory	Packet	Packet-driven adaptive timing scheme
Proposed method				
✓	Heterogeneous sensors different data types	Graph summarization using sensors attributes	Node	Waiting time according to the distance to the sink and furthest point of the network

C convergent, *WT* waiting time

none of the neighbors has data. In this situation, the algorithm sends the data to the closest neighbor to the *Sink*.

3.6 Qualitative comparison of the proposed method and related work

In this subsection, the proposed method is being compared with other similar methods using some parameters in wireless sensor networks. Due to the problems and insufficiencies of structured approaches for event-based applications, the proposed method is intended to fuse and route data in a dynamic way. There have been some attempts trying to solve the mentioned challenges. Table 2 compares the method in this paper and some of the famous similar methods. All methods in this group try to realize two conditions in order to perform fusion. Those two conditions are temporal convergence and spatial convergence. In order to perform fusion, packets containing data need to be met at the same type at the same place.

The proposed method assigns waiting time to the node rather than packets. This makes all nodes, even those which are not in the event area, have the possibility to fuse data rather than just forwarding separate packets.

The calculation of the waiting time depends on two factors: distance to the sink and distance to the furthest point of the network. These two factors will make the desired difference between the waiting times of different nodes which is especially suitable for dense wireless sensor networks. Data types can be different so we can have a network

of heterogeneous sensors. For each event and after the expiration of the waiting time, nodes are chosen to be fused according to the value of these attributes, so that the volume of the data is reduced and the redundant data are not forwarded to the sink. The spatial and temporal mechanisms which are used in this method make the whole fusion convergent; thus, data of different nodes are met and fused before reaching the sink.

4 Evaluation

In order to evaluate the proposed method, it is implemented using VisualSense Simulator. To compare its performance, three other methods are implemented as well. These methods are evaluated showing the effect of different parameters.
Methods The methods that are compared are as follows:

1. *Shortest path (SP) tree*
   In this method, nodes transmit their packets through shortest paths to the sink. Packets are only fused if they meet at the same time at the same node. It is assumed that the topology of the network in known in advance and all nodes are aware of the network conditions.
2. *Shortest path tree and waiting time (SP+WT)*
   In this method, nodes spend some time waiting according to Eq. (1) and then transmit their data using shortest path.
3. *The proposed method: best neighbor and waiting time (BN+WT)*
   First the waiting time is calculated. After the expiration of the waiting time, the best neighbor algorithm is performed and the data are routed.
4. *Data-aware anycast and randomized waiting time (DAA+RW)* In this method, which is proposed in [17], a random waiting time is assigned to the nodes. Then the best neighbor is chosen by sending RTS packets and receiving CTS responses.
5. *Real-time data AGgregation (RAG)* This protocol was designed using two mechanisms for spatial and temporal convergences. This method is intended to answer real-time constraints of wireless sensor networks.

Evaluation criteria The criteria with which different methods are compared are as follows:

1. *Number of total transmissions in network per event*
   This is an important factor since it is an indicator of traffic reduction using fusion. Lower number of transmissions in network will lower the network traffic and consequently energy consumption and response time.
2. *Normalized number of transmissions per event*
   This factor is the division of number of total transmissions in network by number of nodes producing data per event. The number of fusion in the network cannot show the performance of an algorithm alone, since data may be transmitted a lot of hops before being fused. On the other hand, the number of total transmissions can be high because of high miss ratio, not because of the good performance of the algorithm. So this is a good indicator of the methods performance [17].

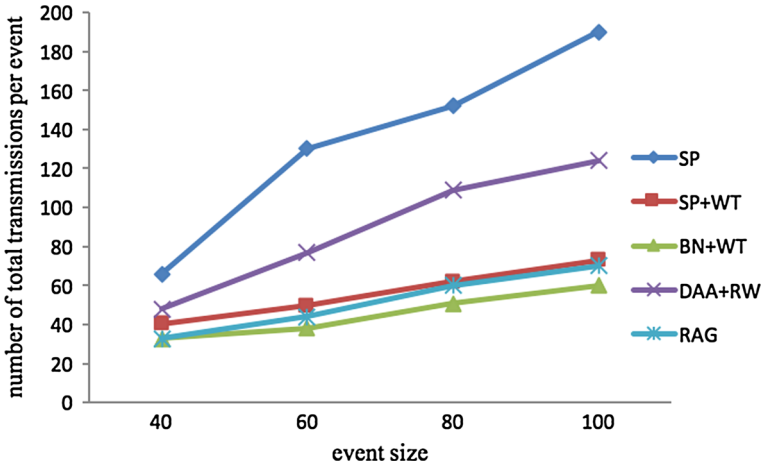

Fig. 2 Comparison of different methods for different event sizes

3. *Distance of the first node containing the whole data to the sink*
   Data-centric routing protocols intend to fuse the whole data before reaching the sink so that fewer packets are transmitted to the sink. The further from the sink the nodes fuse their data, the number of transmissions is reduced and the performance is improved.
4. *Distance of the first node containing the whole data to the event*
   If the fusion can happen close to the event, the number of transmissions is reduced and the performance is improved. An efficient algorithm is able to perform fusion closer to the event.

Implementation assumptions Most of the setup details for evaluation is based on the literature [17,24]. The network under study is a $200 * 200$ m environment. The sink is placed at the center of the network on the top. The communication range is about 25 m. Each sensor has at most 8 neighbors. The reason for this assumption is to make sure the whole area under study is covered by sensors based on the number of deployed sensors. However, our method can be used regardless of number of neighbors per node.

Evaluation scenarios and their results In order to evaluate the performance of the method, different scenarios are implemented. Each scenario is described and the results are discussed. Different algorithms are run multiple times, each time changing the location of the event, and the results are the average values over the multiple runs.

1. *First scenario: Event size effect*

One of the effective parameters on the performance of the methods is the event size. The bigger the event is, more nodes will sense it, and so, more data are available on the network. The method performs well if the increase in event size does not increase the number of transmissions sharply. In the first scenario, based on the network assumptions, we increase the event size by starting from an event of size $40 * 40$ and

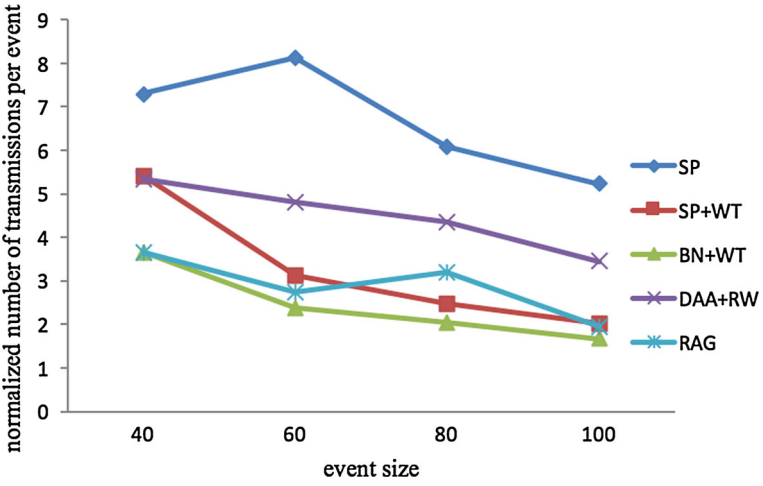

Fig. 3 Effect of event size on normalized number of transmissions

then increasing it to $60 * 60$, $80 * 80$ and $100 * 100$ m. The first criterion studied is the total number of transmissions. The results are shown in Fig. 2.

As it is shown in Fig. 2, the proposed method (BN+WT) has the best performance among all methods. By increasing the event size, the number of transmissions in our method is increased slightly. This shows that this method is less sensitive to the event size increase in comparison with other methods.

In Fig. 3, the effect of event size on the normalized number of transmissions is shown.

As it is shown in Fig. 3, all the methods studied in this paper perform well by the increase in event size. Since more nodes produce data as the size increases, the chances of data being fused at intermediate modes increase. This causes the reduction in normalized number of transmissions per event.

The third factor examined here is the distance of the first node containing the whole data to the sink. This shows the convergence of the proposed method. The further the fusion takes place, the more efficient the algorithm is, since it can reduce the number of transmissions. Figure 4 shows this distance for different methods.

Shortest path and data-aware anycast with randomized waiting time methods are convergent at the sink as shown in Fig. 4. Two other methods, the proposed method and the shortest path with waiting time can be convergent somewhere before the sink. RAG, except the first case, is also convergent at the sink. This factor is important since it can reduce the traffic in the network, so the end-to-end delay is reduced and the response time decreases.

The last factor which is studied is the distance of the first node containing the whole data to the event. The lower this factor is the more efficient the method is, since the fusion takes place sooner. Figure 5 shows different values of this factor for different methods.

As shown in Fig. 5, the proposed method performs the best since it can fuse the data closer to the event than other methods. Early fusing of data closer to event will

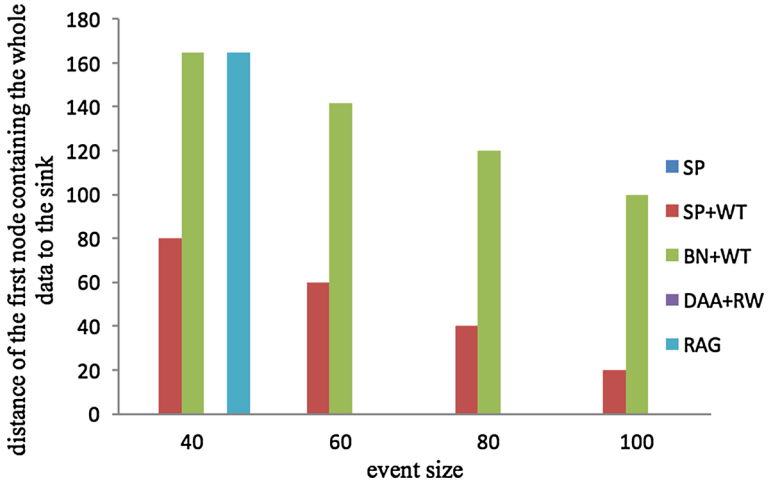

Fig. 4 Distance of the first node containing the whole data to the sink for different methods

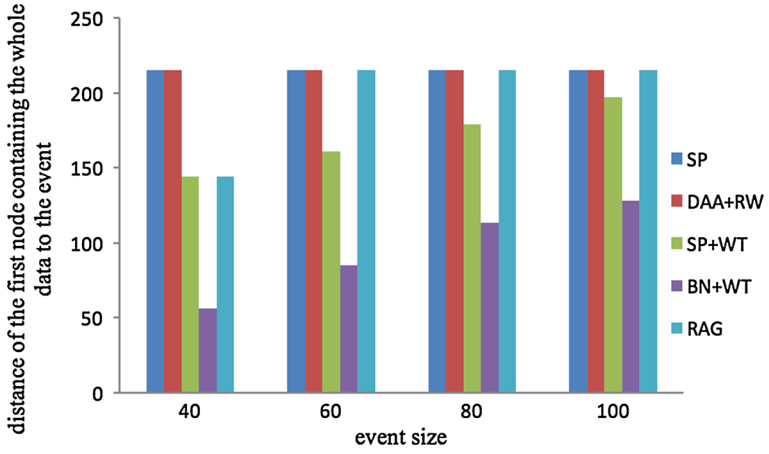

Fig. 5 Distance of the first node containing the whole data to the event for different methods

decrease the number of packets needed to be sent to the sink. Due to spatial and temporal convergence of our method, the fusion is done a lot earlier in comparison with other methods. Shortest path, data-aware anycast with randomized waiting time and RAG have the longest distance because final fusion occurs at the sink.

2. *Second scenario: Distance to sink effect*

The other effective parameter on the performance of different methods is the distance of the event to the sink. If a method is able to reduce the data volume for events happening further from the sink, it can reduce the energy consumption by performing well enough in those situations.

To study this scenario, according to the network assumptions, the location of event is moved. The event size is considered to be $40 * 40$. We start our analysis from the

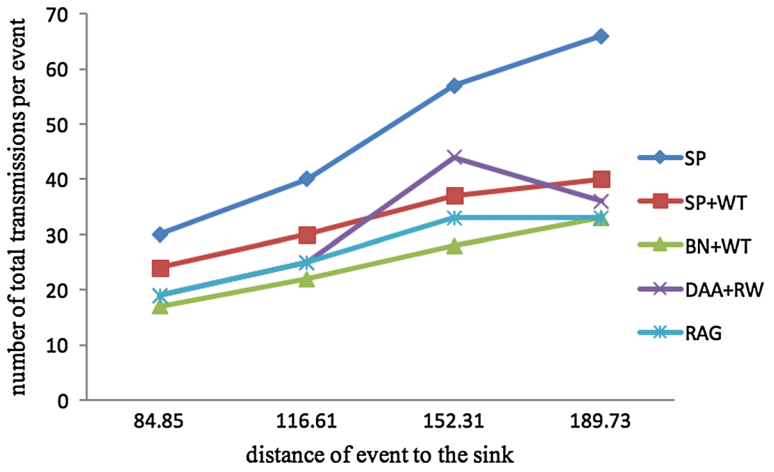

Fig. 6 Comparison of different methods based on the distance of the event to the sink

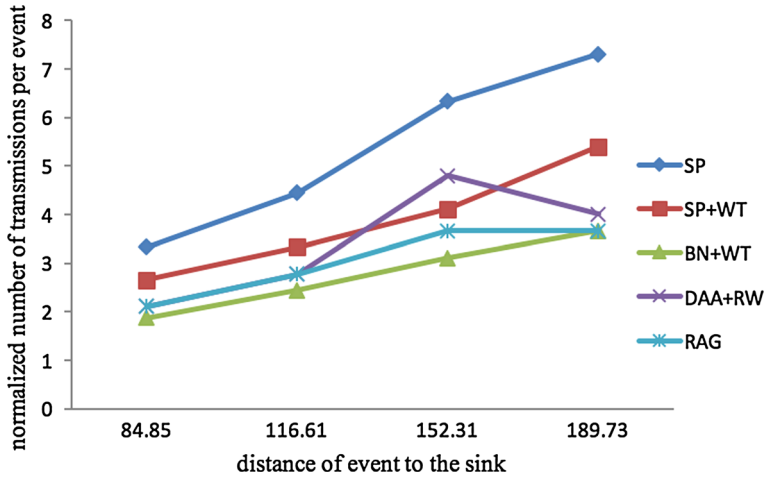

Fig. 7 Comparison of different methods based on the distance of the event to the sink

furthest part of the network and reduce this distance to get closer to the sink. The first criterion examined is number of total transmissions in the network as shown in Fig. 6.

In all methods, the number of transmissions is increased since data take a longer journey to reach the sink. But the proposed method still has the best performance.

The point which is obvious in this graph is the break in the data-aware anycast and randomized waiting time. Since the waiting time assigned to the packets is assigned randomly, it is possible that packets get further away from the sink so more packets remain in the network and the number of transmissions increases. Figure 7 shows the normalized number of transmissions to the distance of event to the sink.

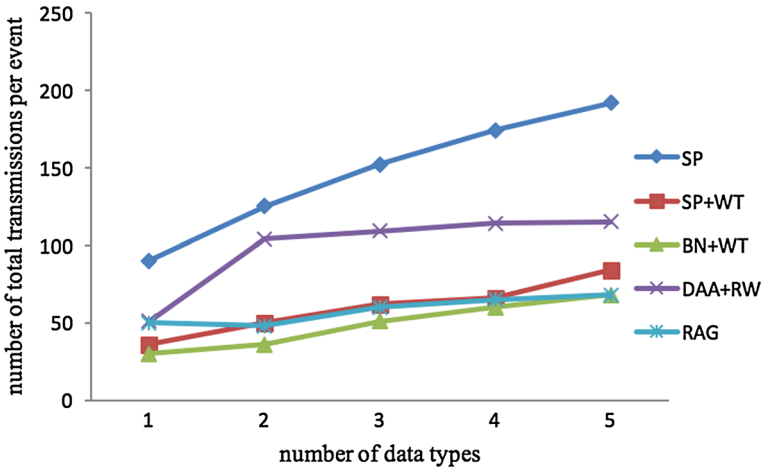

Fig. 8 Number of total transmissions for different data types

As shown in Fig. 7 the value of normalized number of transmissions is also increased by this parameter. This means that by the increase in the distance of the event from the sink, the number of transmissions increases too.

3. *Third scenario: Data type effect*

Another important factor to evaluate the performance of different methods, is the number of different data types in the network. Based on the application, different sensors providing different types of data can be deployed in the environment. This may happen because an event has different aspects so different sensors are needed. The more different the data types are, the lower the chances of aggregation would be. Therefore, more packets remain in the network. If the increase in transmissions is not sharp with the increase in data types, the method performs better. For this scenario an $80 * 80$ network is chosen. The reason for selecting this size of network is that the heterogeneity increases and the chances of aggregation of same-type data decrease. We intend to show that even based on these conditions, the proposed method performs better.

Different approaches are compared according to different number of data types. In the first case, data are chosen to be homogeneous. We increase the data types from one to five and compare the performance of different methods. The first metric studied in this scenario is the number of total transmissions. The results are shown in Fig. 8.

As it is shown in Fig. 8, in all methods the number of transmissions increases by the increase in data types available in the network. The reason is that the more different the packets are in the network, the less they have the chance to be fused. Therefore, more packets remain in the network to be transmitted. The proposed method performs the best among the other methods since if different data cannot be fused in the direct neighboring nodes, they can be fused later in the next hops according to the waiting time defined. Another point in this figure is that data-aware anycast and randomized waiting time approach are the least sensitive one to the data types in comparison with the other approaches, since in this approach the effect of waiting time in fusion is more than the effect of the mechanism to choose the next hop. RAG also performs better by

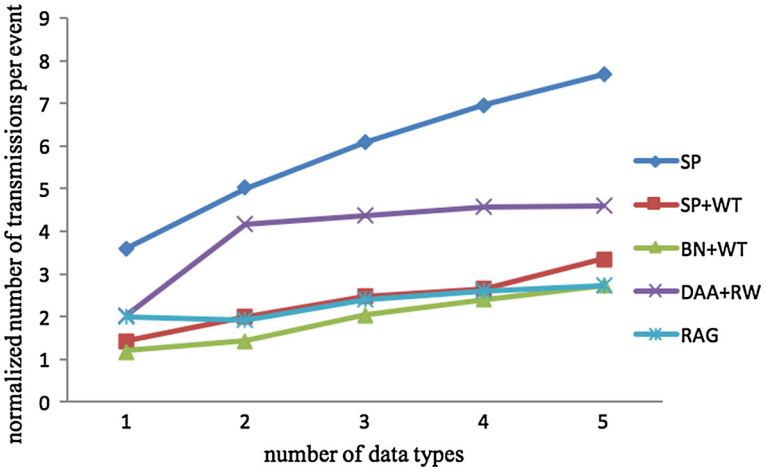

Fig. 9 Number of total transmissions for different data types

the increase in data types. The reason is that by increasing the number of data types, the chances of fusion become less even though the data may be met at intermediate nodes. So the proposed method and RAG will perform similarly.

Figure 9 is representing the effect of data types on the normalized number of transmissions.

Figure 9 shows that the increase in data types leads to the increase in normalized number of transmissions as well, because having different data types decreases the chances of the same-type data to be met and fused.

5 Conclusion

In this paper, we proposed a data fusion method. In order to realize temporal and spatial convergence conditions, two mechanisms are introduced. For the temporal convergence, a waiting time is assigned to each node to increase the chances of fusion at intermediate nodes before reaching the sink. The best neighbor algorithm is used to realize the spatial convergence condition. Simulation results show that the proposed method outperforms similar methods in terms of number of transmissions in the network and the speed of convergence. The best neighbor algorithm can be modified to take into account some other attributes such as the remained energy of each node to further enhance the fusion process. Designing a fusion scheme for mobile wireless sensor networks can be another issue to be addressed in future work.

References

1. Hammoudeh M, Newman R (2013) Adaptive routing in wireless sensor networks: QoS optimization for enhanced application performance. J Inf Fusion 22:3–15
2. Yick J, Mukherjee B, Ghosal D (2008) Wireless sensor network survey. J Comput Netw 52(12):2292–2330

3. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E (2002) Wireless sensor networks: a survey. Science 38(4):393–422
4. Mikami S, Aonishi T, Yoshino H, Ohta C, Kawaguchi H, Yoshimoto M (2006) Aggregation efficiency-aware greedy incremental tree routing for wireless sensor networks. IEICE Trans 89(10):2741–2751
5. Zhang K, Li C, Zhang W (2013) Wireless sensor data fusion algorithm based on the sensor scheduling and batch estimate. Int J Future Comput Commun 2(4):333
6. Li Q, Li C, Li J (2010) Data aggregation algorithm based on grid and adaptive genetic algorithm for wireless senor networks with a mobile sink. In: Intelligent Systems and Applications (ISA), pp 1–4
7. Maraiya K, Kant K, Gupta N (2011) Study of data fusion in wireless sensor network. In: International Conference on Advanced Computing and Communication Technologies, pp 535–539
8. Nakamura EF, Loureiro AAF, Frery AC (2007) Information fusion for wireless sensor networks: methods, models, and classifications. ACM Comput Surv 39(3):9
9. Zhu Y, Vedantham R, Park S-J, Sivakumar R (2008) A scalable correlation aware aggregation strategy for wireless sensor networks. Inf Fusion 9(3):354–369
10. Younis O, Fahmy S (2004) Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach. In: Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies
11. Wenz M, Wom H (2006) Event-based production rules for data aggregation in wireless sensor networks. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Heidelberg, Germany
12. Du W, Deng J, Han YS, Varshney PK (2003) A witness-based approach for data fusion assurance in wireless sensor networks. In: GLOBECOM, vol 3, pp 1435–1439
13. Dai X, Xia F, Wang Z, Sun Y (2005) A survey of intelligent information processing in wireless sensor network. In: International Conference on Mobile Ad-Hoc and Sensor Networks. Springer, Berlin, pp 123–132
14. Yousefi H, Yeganeh MH, Alinaghipour N, Movaghar A (2012) Structure-free real-time data aggregation in wireless sensor networks. Comput Commun 35(9):1132–1140
15. Heinzelman WR, Chandrakasan A, Balakrishnan H (2000) Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Hawaii International Conference on System Sciences, p 10
16. Chao C-M, Hsiao T-Y (2009) Design of structure-free and energy-balanced data aggregation in wireless sensor networks. In: 11th IEEE International Conference on High Performance Computing and Communications, pp 222–229
17. Fan K-W, Liu S, Sinha P (2007) Structure-free data aggregation in sensor networks. IEEE Trans Mob Comput 6(8):929–942
18. Madden S, Franklin MJ, Hellerstein J, Hong W (2002) TAG: a tiny aggregation service for ad-hoc sensor networks. In: Proceedings of the Fifth Symposium on Operating Systems Design and Implementation
19. Ding M, Cheng X, Xue G (2003) Aggregation tree construction in sensor networks. In: Proceedings of the 58th IEEE Vehicular Technology Conference, pp 2168–2172
20. Solis I, Obraczka K (2004) The impact of timing in data aggregation for sensor networks. In: Proceedings of the IEEE International Conference on Communications (ICC04), pp 3640–3645
21. Du H, Hu X, Jia X (2006) Energy efficient routing and scheduling for real-time data aggregation in WSNS. Comput Commun 29:3527–3535
22. Heinzelman WR, Chandrakasan A, Balakrishnan H (2000) Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Hawaii International Conference on System Sciences, p 10
23. Heinzelman WR, Chandrakasan A, Balakrishnan H (2002) An application-specific protocol architecture for wireless microsensor networks. IEEE Trans Wirel Commun 1(4):660–670
24. Lindsey S, Raghavendra C (2002) PEGASIS: power-efficient gathering in sensor information systems. In: Proceedings of IEEE Aerospace Conference, vol 3
25. Lindsey S, Raghavendra CS, Sivalingam KM (2001) Data gathering in sensor networks using the energy delay metric. In: Proceedings 15th International Parallel and Distributed Processing Symposium, vol 188, pp 2001–2008
26. Lindsey S, Raghavendra C, Sivalingam KM (2002) Data gathering algorithms in sensor networks using energy metrics. IEEE Trans Parallel Distrib Syst 13(5):924–935

27. Zhang J, Wu Q, Ren F, He T, Lin C (2010) Effective data aggregation supported by dynamic routing in wireless sensor networks. In: Proceedings of the IEEE International Conference on Communications (ICC10), p 16
28. Ren F, Zhang J, Wu Y, He T, Chen C, Lin C (2013) Attribute-aware data aggregation using potential-based dynamic routing in wireless sensor networks. IEEE Trans Parallel Distrib Syst 24(5):881–892