FOMA: Flexible Overlay Multi-path Data Aggregation in Wireless Sensor Networks

Majid Ashouri, Hamed Yousefi, Ali Mohammad Afshin Hemmatyar, and Ali Movaghar
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
{ashuri,hyousefi}@ce.sharif.edu, {hemmatyar,movaghar}@sharif.edu

Abstract—**Data aggregation is an efficient method to conserve energy by reducing packet transmissions in WSNs. However, providing end-to-end data reliability is a major challenge when the network uses data aggregation. In this case, a packet loss can miss a complete subtree of values, thus greatly affecting the final results. Data transmission in multiple paths can tolerate this problem, but it may incur some computation errors for duplicate-sensitive aggregates. In this paper, we propose a Flexible Overlay Multi-path data Aggregation protocol (FOMA) which uses the available path redundancy to deliver a correct aggregate result to the sink with high reliability in an energy-efficient manner. It aggregates data in two layers, routing layer and data aggregation layer, while eliminating the computation errors by using a signature-based method. We implement FOMA in TinyOs 2.x and test it by the TOSSIM simulator. The results reveal that the proposed algorithm outperforms other existing ones in terms of energy consumption and data accuracy.**

Keywords-**Wireless Sensor Networks; Data Aggregation; Reliability; Multi-path Routing.**

I. Introduction

Wireless Sensor Networks (WSNs) consist of a large number of sensor nodes equipped with limited and irreplaceable batteries which make energy efficiency a major concern [1]. Data aggregation is a promised technique to filter redundancy and reduce communication overhead and, in turn, energy consumption. Many applications use data aggregation functions to summarize and reduce data that must be sent to the sink.

Besides, in WSNs, communication links are unreliable and often unpredictable [7]. Data aggregation reduces the inherent redundancy and total number of messages, so it can decrease the accuracy of the final results against communication errors. A packet loss over a link in a spanning-tree structure can result in the loss of all sensor readings or partial aggregation results from the complete subtree below the link. Thus, the network requires some reliability methods to overcome the faults and increase data delivery probability. Multi-path routing can compensate loss effects by duplicating and forwarding sensor data over available redundant paths. However, it may incur computation errors when aggregating individual sensor values multiple times. Specifically, some aggregation functions are duplicate-sensitive and may produce wrong results with a duplicate aggregation. Finally, aggregation accuracy in WSNs is affected by the fidelity of both communication and computation.

The previous works propose different ways to aggregate data in a reliable manner. SD [6], SKETCH [3], and TD [5] aggressively exploit multi-path routing to combat message losses. However, they suffer from the computation errors in final results due to using hash functions. OPAG [2] performs in-network data aggregation with no computation error and tolerates moderate message losses in WSNs. However, it has a major drawback: The aggregation nodes are usually too far from the source nodes. This problem creates some serious issues in data accuracy and energy consumption.

To achieve full benefits, we propose FOMA, a flexible overlay protocol performing in-network data aggregation in two layers. At routing layer, it creates an aggregation structure based on the link-quality information. When the link error rate is low, the structure is close to a spanning tree. We use multi-path routing advantages to forward redundant packets, so it can tolerate losses if the link error rate is high. At the aggregation layer, FOMA aggregates intermediate partial results as soon as possible on their way to the sink while incurring no computation error and sensor's data is aggregated at most once by exploiting a signature-based transmission method.

The rest of the paper is organized as follows. Section II discusses the related work. Section III explains the proposed scheme and presents its specifications. Experimental setup and simulation results are presented and discussed in Section IV. Finally, Section V concludes our work.

II. Related Work

Many aggregation algorithms have been developed in WSNs. TAG [4] proposes an efficient framework which uses a tree-based structure to collect and aggregates data. However, it suffers from packet loss and node failure. If we lose any aggregated data, all sensor data from the complete subtree below the failure position will be lost.

As discussed before, multi-path transmission reduces communication errors dramatically, but produces duplication concern. To cope with this problem, SD [6] and SKETCH [3] suggest hash-based data aggregation schemes over a ring structure. However, the final value is an approximation of actual value (computation error). TD [5] is a hybrid work that tries to solve the above problems by combining SD and TAG. The nodes that are near to the sink behave similar to SD's nodes (multi-path nodes), but other nodes build a spanning tree and aggregate like TAG in lower levels (tree nodes). However, it still suffers from the computation errors.

To eliminate the computation errors, OPAG [2] proposes an overlay network that separates routing from the aggregation

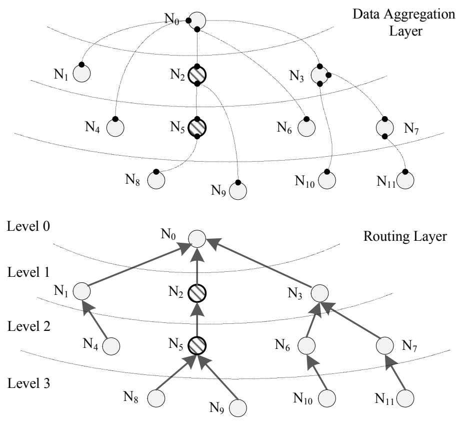

Fig. 1. Data aggregation layer and Routing layer.

layer. It is a highly reliable protocol without any computation error. However, it has a major drawback: The aggregation nodes are usually too far from the source nodes. This problem creates some serious issues in accuracy and energy consumption. Actually, a data message has limitations on length and, in turn, only a few numbers of partial results can be added into it. Therefore, especially in dense networks, OPAG has to drop some partials and it may decrease data aggregation accuracy. Moreover, if the aggregation nodes are far from the source nodes, partials are flooded through the available paths, so a large amount of energy is consumed in the forwarder nodes.

III. FOMA Protocol

FOMA performs data aggregation in two layers: data aggregation layer and routing layer. At the first layer, only the parents and Data Aggregation Nodes (DANs) can aggregate data along an overlay spanning tree while incurring no computation error. We try to compute the aggregation results in the parent nodes to reduce data transmissions; otherwise, DANs are responsible for data aggregation. Underneath the routing layer, network nodes may use the available path redundancy to deliver a correct aggregate result to the sink via a multi-path routing. FOMA has four phases to collect data from sources:

- Creating a routing tree structure: any routing protocol can be used in this phase.
- Finding proper DANs: a DAN can be one or more level lower than source nodes.
- Creating signatures: a signature-based method is used to avoid the duplication problem.
- Data Collection.

We now explain an example with two scenarios to present the basic idea. Let's suppose a routing protocol has built parent-child relations and specified the level of each node in Figure 1 where the root has level 0. The Nodes forward data only to the others at one level lower not to the same or higher level, and each node has one DAN. Here, we want to aggregate N_9's data.

First scenario: Suppose that N_9 and N_5 select N_2 as their DAN. N_9 broadcasts its data packet and N_4, N_5, and N_6 receive that. Because N_5 is the parent of N_9 and has the same DAN, it aggregates N_9's data with its own (and maybe other nodes') and updates the child signature. Besides N_5, N_4 and N_6 also receive N_9's data, so if they have a path to N_2, they add it to their partial results. Suppose N_6 can hear N_2's packets but N_4 cannot. Thus, if N_6's data packet has enough space, it will merge N_9's data into the message and broadcast it, but N_4 drops this partial result because it has no path to N_2. When N_2 receives a data packet from N_5, it looks at N_5's child signature first, and for every bit that has been set, it adds the related node (in this example N_9) to the aggregated node table and aggregates N_5's data. N_6's data packet also contains a partial result from N_9; however, this partial result will be dropped by N_2 because it has aggregated N_9's data before. If the transmitted packet from N_5 is lost in the middle, N_2 adds N_9's partial result that has received from N_6 to the partial table and finally, aggregates it.

Second scenario: Suppose that N_9 chooses one of the nodes at the lower level separated by one hop (N_4, N_5, or N_6) as its DAN. In this condition, no signature modification would be required and the data will be aggregated at the DAN with no overhead. Next, we will present the different phases of the FOMA protocol.

A. DAN Selection

In the previous section, we showed how a node aggregates other node's data or forwards it to the sink by an example. We now explain how a node chooses its DAN. Choosing a DAN in FOMA is close to that of OPAG, but with some differences. Each node tries to select its DAN at one level lower if there is a node providing minimum user reliability; otherwise, the DAN will be selected at lower levels. Suppose that R is the minimum reliability that a user needs, so a node in one-hop distance that has higher reliability than R is selected as its DAN. If no node can satisfy this condition, a node with maximum reliability will be selected at lower levels.

The DAN selection mechanism starts at the root. Each node sends an announcement that could have a few entries (its own and lower level nodes whose announcement it can hear). An entry that is related to node j, contains two reliability parameters. First, the successful transmission probability of node j to the sink (p) and second, the successful transmission probability of node j to node i (pf). For example, suppose N_1 in Figure 1 has received an announcement message from the root whose both reliability parameters are 1. It would broadcast a message with two entries. An entry contains its own announcement with $pf = 1$ and p will be computed as in (1); another contains N_0's announcement with $p = 1$ and pf will be computed as (2). Each node repeats this process until all nodes have a DAN. In (1), we show how a node calculates its successful transmission probability to the sink.

$$p = pf \times p(D) \tag{1}$$

$p(D)$ is the successful transmission probability from a DAN to the sink (it is obtained from the announcement message) and pf is the successful transmission probability to the DAN calculated as follows:

id	*Did*	*Dlevel*	*payload*

Fig. 2. Partial result structure

$$pf = 1 - \prod_{i \in forwarers} (1 - pf_i(D) \times l(i)) \qquad (2)$$

where i belongs to the forwarder list of D that announces it, and $l(i)$ is the link reliability to i. Finally, if a node receives no announcement message or $p(D)$ is lower than a specified threshold, it chooses the parent as its DAN.

B. Signature Construction

According to the previous discussions, data aggregation could be performed in parents or DANs. Each node creates a child signature and broadcasts the child ID to avoid the duplication problem. We now explain child signature creation in this section.

After DAN selection phase was done by all nodes, they must produce a signature for each child. In contrast to the DAN selection phase, signature creation starts at the leaf nodes. Each node sends its own signature announcement message based on the TAG scheduling algorithm [4]. The message contains $parent-id$, $DAN-id$, $child-no$ and some Cid/Did pairs. The $parent-id$ and $DAN-id$ represent the parent and DAN of the sender node, respectively. $Child-no$ specifies the number of Cid/Did pairs. Each Cid/Did shows a child ID and its DAN.

Suppose that node i sends a child signature message to node j. when node j receives the message, it checks the parent field. If j is the parent of i and they both have the same DAN, j adds i and its $DAN-id$ to the child list. Then the algorithm verifies Cid/Did pairs. If j has been selected as a DAN by the announced pairs of Cid/Did, j adds them to the list of nodes that have selected j as their DAN, known as aggregation table. Besides Cid, j obtains the signature of Cid in i and adds it to Cid entry. We create a Cid signature in j according to its position in the message.

C. Data Collection

Here, we will illustrate how a node aggregates data and forwards it through the network. Data collection is initiated just after the signature construction because each node must have a child signature to aggregate data. Data message contains an aggregation signature and the level fields with one or more partial result data structures. The structure of a partial result is shown in Figure 2. The *id* field refers to the address of the node to which this partial belongs. Its DAN address and the DAN level are shown with *Did* and *Dlevel*. Clearly, the *payload* field refers to data that must be delivered to the sink.

When node j receives a data message from node i, it runs the *Receive* function shown in Figure 3. First, It checks the aggregation signature and performs the bitwise *AND* operation between the aggregation and child signature. If the result is not equal to zero, it adds the related node to the aggregated node list. Then j executes the *Process* function for every partial result in the message. It checks the child list, if i is in the

```
1) Periodically sense the environment
2) Receive(msg){
3)   FindAggregatedNodes(msg.sig)
4)   for each partial call Process(i, msg.partial)
5) }
6) Process(i, partial){
7)   if (isChild(i)){
8)     Aggregate(partial.payload)
9)     UpdateSignature(i)}
10)   if (partial.Did = my.id)
11)     AggregaeBeforeSending(i, partial.payload)
12)   if (partial.Dlevel >= my.level)
13)     Drop(partial)
14)   if (!isChild(i) and partial.Dlevel < my.level)
15)     AddToPartialList()
16) }
```

Fig. 3. Partials processing algorithm

child list, it aggregates data payload with its own data and updates the aggregation signature. For this purpose j performs the bitwise *OR* operation between the aggregation signature and the child signature. If a partial result in node i's data message has selected j as its DAN, node j adds *id* of the partial result and its data to the list of data that may be aggregated before sending the task. Finally, if j did not meet the above conditions and its level is lower than *Dlevel*, it adds its partial result to the partial list. The protocol drops partials that have a bigger level than the receiver.

IV. Performance Evaluation

A. Experimental Setup

To evaluate performance, we implemented FOMA as TinyOS modules and tested on the TOSSIM simulator. The results are compared with TAG [4] and OPAG [2] protocols to show the efficiency of FOMA for reliable data aggregation in WSNs. The simulated network consists of 36 nodes located on a 6×6 grid. One of the nodes is configured as the sink and all the other ones sense the environment and forward aggregated data to the sink. Each partial result contains a 8-byte data payload. TinyOS has a limitation on the size of messages (payload has at most a 120-byte length), so a limited number of partials can be added to a data message. A maximum hop distance between a source and its DAN is set to 2 for decreasing control messages and eliminating the affect of redundant paths. We use five minutes to boot sensors and build the network structure. Then each source sends its data every two minutes according to a scheduling algorithm like TAG.

B. Evaluation

Here, we compare FOMA with OPAG and TAG for the performance evaluation. One of the most important parameters of a WSN is energy consumption. Considering that the message sending task consumes the major energy, we use the number of transmitted bytes to show the efficiency of the proposed methods in terms of energy consumption. Aggregation accuracy is another important parameter to be evaluated for data aggregation applications. This parameter

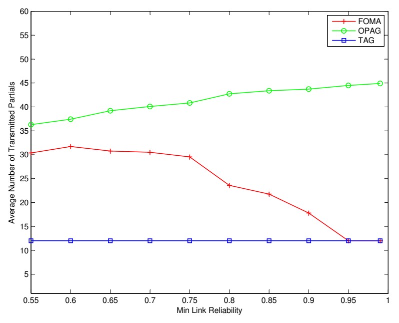

Fig. 4. The impact of link reliability on partial results transmissions

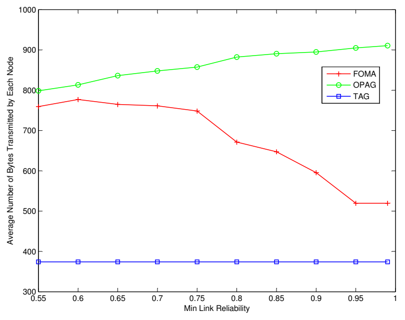

Fig. 5. Total number of transmitted bytes vs. Min Link Reliability

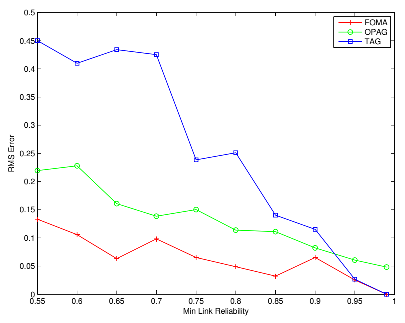

Fig. 6. The impact of link reliability on RMS error

compares the actual value with the aggregated value in the sink. We use the Root Mean Square (RMS) error to normalize the inaccuracy of the final results.

$$RMS = \frac{1}{V}\sqrt{\sum_{t=1}^{N}(V_t - V)^2/N} \tag{3}$$

where V is the actual value and V_t is the aggregated value in the sink at time t. All the active links are uniformly assigned different values of link reliability in the interval of *Min Link Reliability* and 1.

C. Results

We evaluate FOMA in two aspects: energy consumption and data accuracy. The values have been obtained in 30 minutes.

1) Energy Consumption: We compare FOMA's energy consumption with TAG and OPAG in Figure 4. It shows the effect of Minimum Link Reliability on the number of transmitted partial results. Source nodes in TAG send one partial result at each epoch, so it has a fixed value for different link reliabilities. OPAG selects a node with the highest reliability as its DAN and almost chooses the same DAN in most cases; energy consumption increases when the link reliability is high because in this condition most of the partials can be received and forwarded by the forwarder nodes. Unlike OPAG, FOMA has more flexibility in different situations and decreases sending partials in good conditions due to the fact that it tries to choose closer nodes as DANs, if they satisfies the user reliability threshold (user threshold is considered to be 0.9 in all scenarios). FOMA also uses a signature structure to decrease the number of partial results, thus it always sends fewer partial results than OPAG.

Figure 5 shows the total number of transmitted bytes (including the protocol overheads). Although our protocol has some extra control message overhead, it does not have a considerable effect on the total number of the transmitted bytes.

2) Accuracy: Aggregation accuracy is shown in Figure 6. TAG transmits few partials but it has high RMS error, especially when the link error rate is high, because it does not send any additional information to provide end-to-end reliability. FOMA has lower RMS error than OPAG in different conditions. Although our proposed method wastes the length of messages by using the signatures, OPAG fills the message space with partial results which are not aggregated at higher levels quickly, so it loses some partials due to the limited size of the message and in the consequence, it has a higher error in all scenarios. Definitely, this problem becomes more obvious if we increase the network density.

V. Conclusion

In this paper, we presented an overlay protocol for duplicate-sensitive aggregation functions that aggregates partial results with no computation error in a highly energy-efficient manner. When the link error rate is low, FOMA behaves like a spanning-tree and consumes a small amount of energy; otherwise, it uses multi-path to transmit data to the sink for a reliable data aggregation. The results evaluated in simulation demonstrated a significant performance improvement in terms of energy consumption and data accuracy.

Acknowledgment

This work has been supported by Iran Telecommunication Research Center (ITRC).

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[2] Z. Chen and K. G. Shin, "OPAG: Opportunistic data aggregation in wireless sensor networks," in *Proceedings of Real-Time Systems Symposium (RTSS'08)*, Dec. 2008, pp. 345–354.

[3] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proceedings of 20th International Conference on Data Engineering*, April 2004, pp. 449–460.

[4] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: A tiny aggregation service for ad-hoc sensor networks," in *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, 2002.

[5] A. Manjhi, S. Nath, and P. B. Gibbons, "Tributaries and Deltas: efficient and robust aggregation in sensor network streams," in *Proceedings of ACM SIGMOD international conference on Management of data (SIGMOD'05)*, 2005, pp. 287–298.

[6] 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)*, 2004, pp. 250–262.

[7] H. Yousefi, K. Mizanian, and A. H. Jahangir, "Modeling and evaluating the reliability of cluster-based wireless sensor networks," in *Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications (AINA'10)*, April 2010, pp. 827–834.