FOMA: Flexible Overlay Multipath Data Aggregation for Wireless Sensor Networks

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

Abstract—**In wireless sensor networks source data must be routed by intermediate nodes to reach the sink. Data aggregation is an efficient method to decrease energy consumption in these networks. However, providing end to end data reliability is a major challenge when the network uses data aggregation, in this case, a packet loss can greatly affect final results. End to end reliability can be increased by data transmission in multiple paths; but energy consumption and computational error are major constraints.**

In this paper we propose FOMA (Flexible Overlay Multipath Data Aggregation) and FOMA-G (FOMA with gossiping) protocols for aggregation of sensor's data with high reliability and low energy consumption in networks with large number of sensors. The proposed protocol aggregates data in two layers, *routing* and *data aggregation,* and eliminates computational error by using signature based structure. FOMA-G is extended version of FOMA that is introduced to reduce energy consumption. We implement two version of FOMA in TinyOs 2.x and test them by TOSSIM simulator. Finally, in order to evaluate FOMA, we compare it with previous works. The results show that our protocols consumes much less energy and FOMA has lower RMS (Root Mean Square) error in all cases.

Keywords; Wireless Sensor Network, Data Aggregation, Reliability, Multipath, Overlay Network

I. INTRODUCTION

Recent advances in sensor and wireless networking have made it possible to deploy large number of sensor devices in Wireless Sensor Networks (WSNs). These networks consist of distributed nodes that monitor environment conditions. We need to process the sensor's data at particular nodes that are called sink. However, sensor nodes are equipped with small batteries, (limited power) and make energy efficiency a major concern. Because most of the energy consumption occurs due to data transmission [1], it is highly critical to reduce the amount of transmitted data and data aggregation is an efficient way to achieve this goal.

Many applications use data aggregation functions such as SUM, AVG, MIN and MAX that have been categorized in [2], to summarize and reduce the data that must be sent to the sink. Though data aggregation reduces total number of messages, it can cause final result to be less accurate, due to communicational or computational error. Computation of aggregation results may be exact [2,3] or approximate[4,5]. Besides If we lose one packet from node *A* due to communicational error, *A*'s data and all of the aggregated data in *A* and in prior nodes will be missed. So, the network requires some reliability method that can cover these faults and increase data delivery probability in sink. Multipath routing can compensate data loss by duplicating and forwarding sensor data over multipath. Using aggregation functions such as MIN and MAX involve no complexity but other functions such as COUNT, SUM and AVG are duplicate-sensitive and duplication may produce wrong results. Some of the previous works like Synopsis Diffusion (SD) [4], SKETCH [5], TD [6], Ride Sharing [7] and OPAG [3] propose different ways to solve this problem.

Using hash functions and broadcasting in SD and SKETCH make them robust to packet loss but they have computational error. TD tries to decrease computational error in previous works, but it also suffers from this problem too. There is no computational error in OPAG but it consumes lots of energy and could not work well in dense environments.

In FOMA we propose an overlay data aggregation protocol that creates aggregation structure based on link error rate. Our protocol structure is very flexible. When link error rate is low the structure can be close to spanning tree. We use multipath routing advantages to forward packets if link error rate would be high. Using multipath can cause before-mentioned duplication problem, but we will use very low overhead signature to overcome this problem. FOMA does not have any computational error and Sensor's data is aggregated at most once, during transmission to the sink. We attained 100% delivery ratio when link error rate isn't high.

Reducing energy consumption was another improvement achieved by two-layer data aggregation in FOMA which will be explained in section III. Our designed protocol also can work well in dense environments. FOMA-G is an extended version of FOMA that is suitable for networks with strict limitations on energy consumption. It reduces energy consumption by gossiping mechanism.

FOMA properties and contributions are listed below:

- Tow layer data aggregation to increase reliability with low energy consumption.
- Flexible data aggregation structure based on link error rate.

- Suitable for dense networks with large number of sensors.
- Providing high reliability in presence of packet loss without any computational error.

The rest of the paper is organized as follows. Section II describes the related works that is basis for this paper. Section III presents basic idea, protocol phases and details of FOMA protocol. We introduce FOMA-G in section IV. Our experimental setup, evaluation and simulation results are illustrated in section V and finally we conclude our work in section VI.

II. RELATED WORKS

In this section, We'd review related works on reliable data aggregation by using multipath. Typically, data aggregation and broadcasting are very fundamental and useful in wireless sensor networks and they are used in many applications to build a network with low energy consumption and high reliability. If each sensor transmits its data to sink directly, the network consumes large amount of energy. TAG [2,8] proposes a framework to reduce energy consumption by data aggregation. It uses a tree based structure to aggregates and collect data. Though, this structure is simple and reduces the number of data transmission, it suffers from packet loss and node failure. If we lose an aggregated data, actually we will lose all sensor data that are in sub tree below the failure.

As discussed before, multipath transmission reduces communicational error dramatically, but produces duplication problem. To cope with this problem, SD [4] and SKETCH [5] suggest hash based data aggregation schema. They convert duplicate sensitive data aggregation functions to insensitive operation like as OR. This operation maps *sensor ids* to array of bits and the protocol adds it to the data packet. Using broadcasting property of wireless environment and ring structure make them high loss tolerant. However they have two major problems, first: the final value is approximation of actual value (computational error), second: they have extra overhead to attach bit vector to the data message.

TD [6] is a hybrid work that tried to solve above problems by combining SD and TAG. The nodes that are near to the sink behave similar to SD's nodes (*multipath nodes*), other nodes build a spanning tree and aggregate like as TAG in lower levels (*tree nodes*). In low error rate conditions, tree nodes comprise majority of network without any overhead and when loss rate is high TD labels the most of nodes as multipath node. Though this protocol has reduced overhead and increased accuracy of results, it still suffers from computational error.

To eliminate computational error, OPAG [3] proposes an overlay network that separates routing from aggregation layer. Each node maintains a data aggregation candidate list and chooses a node with highest reliability as *aggregation node* from this list. A partial data is aggregated only in aggregation node and intermediate nodes add and forward partial results. OPAG is a highly reliable protocol without any computational error. It also provides accurate result in well distributed networks. But it has following drawbacks. (1) it consumes lots of energy for adding partial results to the packet until the packet has free space; (2) the aggregation node is the best possible reliable node and it may be too far from the source node (3) OPAG has problem with dense networks because the sensors have limitation in packet length and a few number of partial results can be added to the packet.

III. FOMA PROTOCOL

FOMA data aggregation is performed in two layers, data aggregation and routing. Only parent and Data Aggregation Node (DAN) can aggregate data. We use data aggregation operation in parent node to reduce data transmission and if it couldn't do that, DAN is responsible for aggregation of data.

FOMA has four phases to collect data from sources:

- Creating routing tree: any routing protocol can be used in this phase.
- Finding proper DAN: DAN can be one or more level higher than source nodes.
- Creating signatures phase. Child signature is used to avoid duplication problem.
- Data Collection phase.

We'd explain an example with two scenarios to present the basic idea. Let's suppose a routing protocol has built parent-child relations and specify level of each node in figure 1. For example if Root level be 0, therefore N_5 is parent of N_8 and N_9 and has been located in level 2. Nodes forward data to other nodes that are one hop lower and each node has one *Data Aggregation Node* (DAN). We want to aggregate N_9's data and the curve encircling it shows its radio range.

First scenario: Suppose N_9 and N_5 select N_2 as their DAN. N_9 broadcast its data packet and N_4, N_5, and N_6 receive that. Because N_5 is parent of N_9 and has same DAN with it, aggregates N_9's data with its own data (and maybe other nodes') and update its child signature. Besides N_5, N_4 and N_6 also receive N_9's data and if they have path to N_2 add it to their partial results. Suppose N_6 can hear N_2's packets but N_4 cant, so, if N_6's data packet would have enough space, merge N_9's data into the packet and broadcast it, but N_4 would drop this partial result because has no path to N_2. When N_2 receives data packet from N_5 looks at N_5's child signature first, for every bit that has been set, adds it (in this example N_9) to aggregated node table and aggregates N_5's data too. N_6's data packet contain partial result from N_9 too, however, this partial result will be dropped by N_2 because it has aggregated N_9's data before.

If N_5 can't get N_9 data packet or transmitted packet from N_5 would be lost in middle, N_2 adds N_9's partial result that has received from N_6 to partial table and finally aggregates it if does not receive any packet from N_5.

Second scenario: Suppose N_9 chooses one of the nodes that are separated by one hop (N_4, N_5 or N_6) as its DAN, in this condition no signature modification would be required and data will be aggregated at DAN without any overhead. In *A*, *B* and *C* we will illustrate FOMA protocol when DAN is two hops lower at most and generalized protocol will be introduced in *D*.

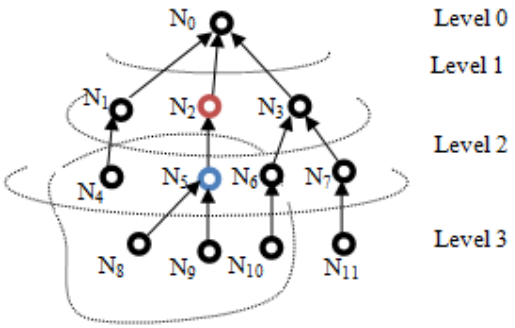

Figure 1. Parent child relationship between nodes

A. DAN Selection

We showed how a node aggregates other node's data or forward it to the sink by an example, and supposed N_9 has a DAN. In this section we explain how a node chooses its DAN. Choosing DAN in FOMA is close to that of OPAG, but with some differences.

Each node is responsible for choosing its DAN. Nodes try to select DAN in one hop distance if DAN can provide minimum user reliability otherwise DAN will be selected in lower levels. Suppose minimum reliability that user needs, is *R*, So, a node that has higher reliability than *R* is selected as DAN. If no node can provide this condition, a node with maximum reliability will be selected at lower layers.

DAN selection mechanism starts at root. Each node sends an announcement that could has few entries (its own data and lower level nodes that it can hear their announcement). An entry that is related to $node_i$, contains two reliability parameters, first, successful transmission probability of $node_i$ to sink (*p*) and second, successful transmission probability of this node ($node_j$) to $node_i$ (*pf*). For example suppose N_1 in figure 1 has received an announcement message from the root that both of its 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 (1), another entry 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.

Formula (1) shows how a node calculates its successful transmission probability to sink.

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

p(D) is successful transmission probability of DAN to sink (*p(D)* obtained from announcement message) and *pf* is successful transmission probability to DAN that is shown in (2). Where *F* belongs to forwarder list of *D* that announces *D*, and *l(F)* is the link reliability to *F*.

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

Finally if a node receives no announcement message or *p*(*D*) is lower than a specified threshold (for example 0.5), it chooses its parent as DAN.

B. Signiture construction

According to previous discussions data aggregation could be performed in parent or DAN. Each node creates child signature and broadcast its child to avoid duplication problem. We'd 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 data transmission phase a node sends one signature, but in this phase we create a signature for each child and maintain it. In contrast to DAN selection phase, signature creation starts at leaf nodes. Each node sends its own signature announcement message based on TAG scheduling algorithm [2]. The message contains *parent-id*, *DAN-id*, *child-no* and some *Cid/Did* pairs. Parent-id and DAN-id represent parent and DAN of sender node respectively, child-no specifies number of *Cid/Did* pairs. Each *Cid/Did* shows a child ID and its DAN.

Suppose node *i* sends a child signature message to node *j*. when node *j* receives the message, checks parent field. If *j* is parent of *i* and *i* has same DAN with it, *j* adds *i* and its *DAN-id* to child list. Then the algorithm verifies *Cid/Did* pairs. If *j* has been selected as DAN by announced *Cid/Did*s, *j* adds them to the list of nodes that has selected *j* as its DAN that we call aggregation table. Beside Cid, *j* obtains signature of Cid in *i* and add it to Cid entry. We create Cid signature in *j* according to its position in message.

For example, in figure 1 suppose the signature size is one byte and N_9 has selected N_2 as its DAN. N_5 sends a signature announcement message to N_2 and N_9 has announced in second *Cid/Did* pair. When N_2 receives the message, sees N_9 has selected it as its DAN and N_9 is located in second position, so it assigns "01000000" to N_9's entry in aggregation table.

For Sending purpose, each parent node puts Cid/Did pairs in message according to its signature. In N_5 node, if N_9 has "10000000" signature, it would be first in message.

C. Data Collection

Here, we will illustrate how a node aggregates data and forward it through the network. Data collection is started just after signature construction because each node must have child signature to aggregate data.

Data message contains *aggregation signature* and *level* fields with one or more *partial result* data structure. The structure of a partial result is shown in figure 2. The *id* field refers to address of node that this partial belongs to. Its DAN address and level of DAN are shown with *Did* and *Dlevel*. Clearly the *payload* field refers to data that must be delivered to the sink.

$$\alpha + \beta = \chi. \tag{1}$$

id	Did	Dlevel	payload

Figure 2. Partial result structure

When node *j* receives a data message from node *i*, runs *receive* function that was shown in figure 3. First, It checks aggregation signature and performs bitwise AND operation between aggregation and child signature, if the result is not equal to zero adds it to *aggregated node* list. Then *j* executes *Process* function for every partial result in the message. It checks child list, if *i* is in child list, it aggregates data payload with its own data and updates aggregation signature. For this purpose *j* performs bitwise *OR* operation between aggregation signature and child signature.

```
1)  Periodically sense 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 = me.id)
11)     aggregaeBeforeSending(i , partial.payload)
12)   if (partial.Dlevel>= me.level and  )
13)     drop(partial)
14)   if (!isChild(i) and partial.Dlevel<me.level)
15)     addToPartialList()
16) }
```

Figure 3. Processing partials algorihm

If a partial result in node$_i$'s data message has selected *j* as its DAN, node *j* adds *id* of partial result and its data to the list of data that may be aggregated before sending task. Finally if *j* didn't meet above conditions and its level is lower than Dlevel adds partial result to partial list. The protocol drops partials that have bigger level than receiver.

D. Generalizaion

We show how FOMA works when a node selects its DAN in 2 hop distance at most. However our protocol can be generalized with few modifications. In that case each node can choose DAN in more than two hop distance. The generalized protocol creates a main problem: How DAN is informed from data aggregation in parent? We need to add some additional information to solve this problem.

DAN node must know the path of receiving aggregation signatures and maintains that. Path information contains middle parents that forward source data to the DAN. for example when N_9 selects N_0 as its DAN, N_0 needs to maintain N_2 and N_5 as forwarder parents. In data collection phase each parent puts its address and aggregation signature in the message and forwards data.

IV. FOMA-G

We'd introduce FOMA-G in this section. It is a modified version of FOMA that reduces transmitted partial results and energy consumption by *gossiping*. In gossiping source node broadcasts its data, each receiver forwards source data with specified probability. For example, suppose node$_i$ sends a packet and its packet is received by three nodes, if the reliability of all links is equal to one then each receiver forwards data of node$_i$ with probability 1/3.

FOMA forwards partial results in all of possible paths and doesn't control the number transmissions. FOMA-G control partial result transmission by gossiping. We want to aggregate a partial result in DAN. If DAN just has a copy of this partial result can aggregate it, so we have to guarantee that one packet can be received in DAN and drop extra ones. For this purpose when node *f* (a forwarder node) receives a partial result from *s* (a source node) forwards node s's partial result to DAN with probability *pg(f)* that is calculated in (3).

$$pg(f) = \frac{1}{k(1-e_{sf})(1-e_{fd})} \tag{3}$$

Where *k* is the number of forwarders of a source node to the DAN that is announced by node *s* in signature creation phase. The e_{sf} shows the link error rate between source and the forwarder *f*, and e_{fd} shows link error rate between forwarder and DAN. if *pg* has a value lower than one FOMA-G forwards the parial results with probability of *pg*, otherwise it behave such as FOMA and forward partial result.

V. SIMULATION AND RESULTS

A. Experimental Setup

For the purpose of performance evaluation, we implement FOMA and FOMA-G as TinyOS modules that are represented in figure 4, and tested them on TOSSIM simulator. We also implement OPAG and TAG for comparison with same parameter. The simulated network consists of 36 nodes, located on a 6 * 6 grid. One of the nodes is configured as sink, all other nodes sense the environment and forward aggregated data to the sink. The physical distance between nodes is roughly 20m that enable reliable communication between neighbors. TOSSIM simulates Micaz motes with CC2420 radio stack that along with physical layer use 20 bytes header.

Our implementation is based on Collection Tree Protocol (CTP) inTinyOS-2.x [9].andhas three principal modules as shown in figure 4:

1- NetCtrl creates and updates first three phases of FOMA with control messages that was explained in secion III. This module is a modified version of CTPRouingEngineP.

2- DataCol performs collection and aggregation of data .

3- LinkEst is modified version of LinkEstimaorP module and estimates link qualities.

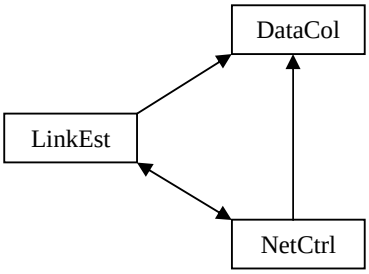

Figure 4. FOMA components in TinyOS

Each partial result contains 8 bytes data payload. TinyOS have limitation on size of messages (payload has at most 120 bytes length), so limited number of partials can be used in data message. Hop distance between source and its DAN at most is set two, for two reasons. First: control messages increase exponentially with increasing distance, second: created paths from source to sink often are *braid multipath*[10] and have common nodes in most cases.

We use five minutes to boot sensors and build network structure, then each source send its data every two minutes according to scheduling algorithm like as TAG. The Protocol also updates routing tree and data aggregation structure every 30 minutes.

B. Evaluation

We compare FOMA and FOMA-G with OPAG and TAG for performance evaluation. One of the most important parameters of WSN is energy consumption. Because message sending task consumes major energy, we'd use number of transmitted bytes to show energy consumption. Aggregation accuracy is another important parameter in data aggregation. Each source produces its own data and data value varies in time, so we have variable range of values in sink. We use Root Mean Square (RMS) to normalize final aggregated data.

$$RMS = \frac{1}{V}\sqrt{\sum_{t=1}^{N}(V_t - V)^2 / N}\,. \qquad (4)$$

Where V is expected value and V_t is aggregated value in sink at time *t*. By using (4), the obtained results would be scaled between [0,1] bound. We use another parameter that is called *Min Link Reliability*. This parameter denotes the minimum amount of link reliability that is expected for a link which all the active links have higher reliability than it.

C. Results

We evaluate FOMA in two aspects: energy consumption and accuracy. The values have been obtained in 30 minutes. (The protocol updates structure every 30 minutes).

1) Energy Consumption:

We compare FOMA's energy consumption with TAG and OPAG in figures 5 and 6. Figure 5 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 fixed value for different link reliabilities. OPAG selects a node with highest reliability as its DAN and almost chooses the same DAN in the most cases; therefore energy consumption does not decrease when the link reliability is high. Contrary to 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 DAN, if DAN satisfies user reliability threshold (user threshold is considered 0.9). lFOMA uses signature to decrease the number of partial results, thus it always sends fewer partial results than OPAG. FOMA-G send fewer partials because it try to deliver one copy of a partial result and drop extra ones.

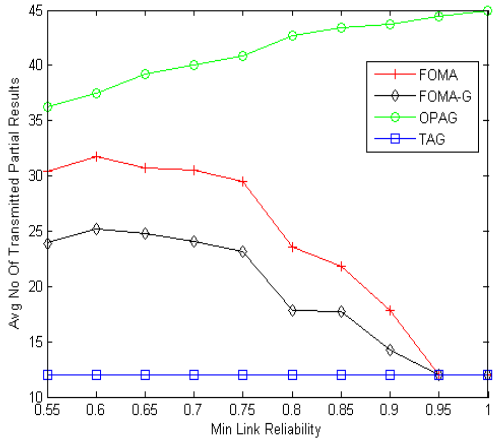

Figure 5. The effect of link reliability on the number of transmitted partial results

Figure 6 shows the total number of transmitted bytes. As expected before, it directly related to number of partial results. Total energy would be decreased if we decrease the number of partials.

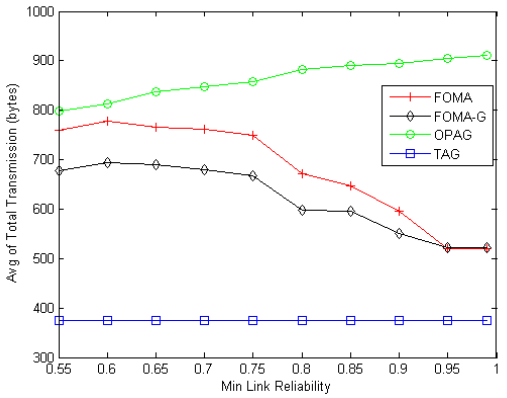

Figure 6. Total number of transmitted bytes.

2) Accuracy

Aggregation accuracy is shown in figure 7. The network has high accuracy if it has low RMS error. TAG transmits a few partials but has high RMS error when the link error rate is high. FOMA has lower RMS error than OPAG in different conditions but FOMA-G behave like OPAG and just is better than it in good environmental conditions. Our protocols reduces size of messages by using the signature but OPAG fills message space with partial results quickly and has to drop additional partials. So, it loses some partials and in the consequence has error in low error rate. This problem becomes more obvious in the dense networks definitely.

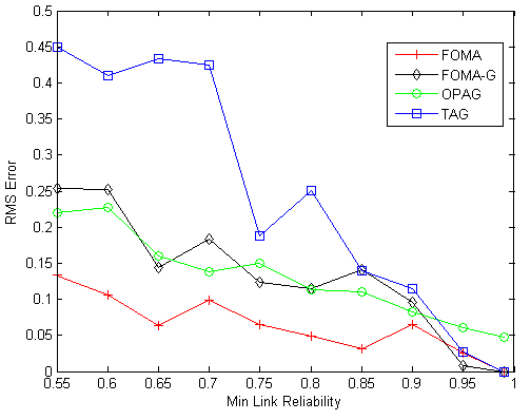

Figure 7. RMS error

VI. CONCLUSION

In this paper we present an overlay protocol for duplicate sensitive aggregation functions that aggregates partial results with no computational error. When the link error rate is low FOMA behaves like as spanning-tree and consumes a few amount of energy, otherwise it uses multipath to transmit data to the sink. Compared to previous works, our new protocol provides highly reliable data aggregation with lower energy consumption. We reduce energy consumption in FOMA-G because It guarantees that DAN just receives one copy of a partial result and drops extra ones.

VII. REFRENCES

[1] B. Krishnamachari, D. Estrin, and S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks,"*22nd International Conference on Distributed Computing Systems Workshops*, 2002, pp. 575-578.

[2] S. Madden, M.J. Franklin, J. Hellerstein, W. Hong, "TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks," *5th Symp. Operating Systems Design and Implementation (OSDI), 2002*.

[3] Z. Chen, K.G. Shin, and A. Arbor, "OPAG: Opportunistic Data Aggregation in Wireless Sensor Networks," *Real-Time Systems Symposium*, 2008, pp. 345-354.

[4] S. Nath, H. Yu, and P.B. Gibbons, "Synopsis Diffusion for Robust Aggregation in Sensor Networks," in *ACM SenSys*, Nov. 2004.

[5] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate Aggregation Techniques for Sensor Databases," *20th International Conference on Data Engineering*, 2004, pp. 449-460

[6] A. Manjhi, S. Nath and P.B. Gibbons, "Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams," in *ACM SIGMOD, Jun.* 2005.

[7] S. Gobriel, S. Khattab, D. Mosse, J. Brustoloni, and R. Melhem, "RideSharing: Fault Tolerant Aggregation in Sensor Networks Using Corrective Actions," in *SECON '06*, Reston, VA, Sept. 2006, pp. 595-604.

[8] S. Madden, R. Szewczyk, M. Franklin, and D. Culler. "Supporting aggregationg queries over sreaming sensor daa", In *ICDE*, 2002.

[9] "TinyOs" , http://www.tinyos.net

[10] D. Ganesan and S. Shenker, "Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, 2001, vol. 5, pp. 11-25

[11] R. Rajagopalan and P. Varshney, "Data Aggregation Techniques in Sensor Networks: A Survey," *Comm. Surveys & Tutorials, IEEE*, vol.8, 2006, pp. 48-63.

[12] J. Yick, B. Mukherjee and D.Ghosal, "Wireless sensor network survey," Computer Networks, vol 52, no 12, pp. 2292-2330, August 2008.