A Survey on In-network Aggregation in Sensor Networks

Minkoo Seo

Division of Computer Science,
EECS Dept., KAIST, KOREA
{mkseo}@bulsai.kaist.ac.kr

Abstract

Sensor network is an active research area, because of its wide variety of applications. However, sensors used in in area have limited cpu, memory, battery, and wireless communications capability. Among them, battery life is the most critical factor because replacing batteries of sensors is not feasible in many situations.

Hence, the primary concern in this area is to devise an energy efficient algorithms that can be used for sending sensed values to the base station. *In-network aggregation*, which exploits routing tree and small but useful computation power of sensors, is a prominent answer to this problem in the literature. Therefore, in this paper, general architecture and its variations of in-network aggregation are explained. An overall architectural option matrix is also provided.

1 Introduction

Recent advances in wireless networks made it possible to distribute smart sensors in a specific region and monitors sensed values. This type of configuration is called *wireless sensor networks*, *sensor networks*, or *wireless ad-hoc sensor networks (WASN)*. Hereafter, I will simply use the term *Sensor Network*.

Sensor networks have a wide range of applications: monitoring endangered species [5][8], assessing structural integrity of bridges [8], or office monitoring [18]. Obviously, these applications depend on the ability of extracting sensed values from sensors and passing the values to the base station.

However, sensor nodes, used in sensor network, have four major physical resource constraints: (1) wireless only communication, (2) limited battery, (3) limited computation power/memory, and (4) unreliability in sensing and communication. These constraints complicate the process of sensing, extracting, and passing of values. Because of the limited battery, it's not possible to send values all the time. Limited computation power and memory hinder us from making a complex summarization and from just storing all values in sensor nodes. Because of unreliability, we can't be sure how long will it take before a node send some values to us.

To tackle this problem, *in-network aggregation* is proposed and gaining popularity. In-network means to sending partially aggregated values rather than raw values, thereby reducing power consumption[14]. And the aggregation means to send some representative values instead of sending entire values. For example, MIN, AVERAGE, MAX, and MEDIAN are representative aggregation operators. Also, to get a more specific data, GROUP BY and HAVING clauses can be used.

Therefore, in this paper, I will introduce the reason why in-network aggregations are needed, and how in-network aggregations are done.

The plan for the remained of this paper is as follows. In section 2, the characteristics of sensor nodes are explained. In section 3, sending raw data and in-network aggregation is compared, and the reason why in-network aggregation is advantageous is explained. After depicting general approach used in in-network aggregation in section 4, variations used in the scheme are explored in section 5. Section 6 shows overall architectural option matrix and concludes this paper.

2 Sensor Nodes

Sensor nodes are responsible for sensing data, say temperature, and sending them to the base station. In this section, four physical constraints of sensor nodes are explained using the example sensor node, MICA Mote [9], developed by researchers at UC Berkeley.

1. **Wireless Communication** The bandwidth of wireless communication is limited. Mote can send data at the rate of 10kb/s using 917Hz RFM radio. More importantly, wireless communication uses much more power than executing instructions in the node; transmitting a single bit of data is equivalent to 800 instructions. There's another interesting property in wireless communication. Generally, each packet is sent from a node to the other node using broadcasting. Thus each node within a certain range from the source can snoop the packet sent from the source.
2. **Limited Battery** In mote, power is supplied by AA battery pack or a coin-cell attached through the expansion. Thus Mote can operate for approximately one year in the idle state and for one week under full load [18]. Therefore, low power algorithm is most important criteria in sensor network.
3. **Computation Power and Memory** Sensor nodes does not have enough computation power and memory. MICA Mote has 4MHz Amtel microprocessor with 512 bytes of RAM and 8 kb of code space. Due to this, it is not always possible to perform complex algorithm, which might be useful if it could be done, like RSA, or DES.
4. **Unreliability** Sensor nodes are unreliable in two respects [18]. Firstly, communication link can be simply broken. Secondly, data read from sensor is uncertain inherently. Sensor usually covers a certain area, but the sensing happens exactly at one specific position in that area. Therefore, it's not sure whether the whole area would have the same value.

3 Sending Raw Data versus In-network Aggregation

Sensor networks can be seen as a connected graph. In the graph, there is always a sink node to which sensed data should be delivered and the sources from where data should be read and sent. Then, the problem is that how and what values should be sent to the sink. For that purpose, a tree is built [1][9][10][14][16][17][18][15] where the root node represents the sink, and the others represents sensor nodes. Recently, a connected graph where tree and multi-path coexists is also proposed [12]. However, for the shake of exposition, we assume there exists a tree.

3.1 Sending Raw Data

In this scheme, all raw data are sent from the leaf to the root. After leaf nodes finishing sensing, values are passed to the parent. In turn, after parent nodes' finishing sensing, values are merged using a simple list or an array. Then, the value is passed to the grandparents. These procedure continues until reaching the root. It is also possible for internal nodes to just pass the received packet to its parent without merging [17].

This is a simple way of passing values, but has severe disadvantages. Firstly, it consumes a lot of energy than in-network aggregation [16]. Secondly, individual values does hold much value, so sending raw data is useless [15].

Therefore, it is common to compute aggregates and sending it along the routing tree [12].

3.2 In-network Aggregation

In-network aggregation, multiple data is aggregated. Then, the aggregated values are sent along the routing tree. For this purpose, aggregation operators should behave like the following.

Firstly, the structure of aggregation function should satisfy $\langle z \rangle = f(\langle x \rangle, \langle y \rangle)$ where f is an aggregation function and x, y, and z is a set of value, respectively [3]. This equation means that the aggregation value is computed from two or more values.

Secondly, it is desirable for f to satisfy $f(x \bigcup y) = g(f(x), f(y))$ which means that the same value should be produced whether f is applied to each values or to the entire values [9]. As an example, consider AVERAGE aggregation. Suppose that $x = \{1, 2\}$ and $y = \{2, 3\}$. If we compute $AVERAGE(x \bigcup y)$, the result will be 2. Now, if we compute $AVERAGE(AVERAGE(x), AVERAGE(y))$, then the result will be also 2.

Because it is less expensive to perform more computation than to send more data as mentioned earlier, in-network aggregation is widely being accepted as a common practice.

4 General In-Network Aggregation

In this section, a general architecture used in in-network aggregation is explained before showing variants of it. Henceforth, let G be a graph consists of sensor nodes. Let N_i represents a node whose ID is i. There also exists a node called base station, B, to which data should be sent. Let $P(N_i)$ be a parent node of the node N_i. f represents an aggregate function, and the administrator of this sensor network is interested in Q.

In in-network aggregation, using ad-hoc routing algorithm like [4][6], a graph G is built. Then, the query, Q, is sent to all sensor nodes. As an answer, sensed value of N_i is propagated to $P(N_i)$. Value of $P(N_i)$ is combined with the value of N_i using f, i.e, $z = f(\textit{value at } N_i,\ \textit{value at } P(N_i))$ is computed. Then the new value, z, is propagated to $P(P(N_i))$ until reaching B. Finally, administrator get the answer for Q.

In this procedure, interesting problems shown below are raised:

1. **Graph Topology**: Does G have to be a tree or a connected graph?
2. **Sending Queries**: How to send Q to all sensor nodes.
3. **Synchronization**: How can $P(N_i)$ knows whether N_i is trying to its value or simply the link is broken.
4. **Topology Optimization**: How to consider the health, say amount of battery remained or error rate, of N_i.
5. **Communication Frequency**: How often data should be sent to B? Continuously? or just from time to time?
6. **Aggregation Operators**: What kinds of aggregations can be offered?
7. **GROUP BY and HAVING**: How can GROUP BY and HAVING operator can be implemented?

In the following section, solutions from the literature will be explained and compared.

5 Variations in In-network Aggregations

5.1 Graph Topology

In TinyDB [9][10] and Cougar [17], G is a tree. Hence, each N_i sends its value to $P(N_i)$. This seems to be simple and effective.

However, recently, multi-path graph where multiple edges can exists between two nodes are is gaining more attention [2][12][13]. The reason is that the data sent from a sensor to the other is easily lost due to large communication error rate of wireless communication [13]. The rationale behind multi-path graph is as follows: if a node sends its value to the multiple parent, the possibility of losing the value decreases.

To exploit multi-path, however, a way of avoiding double counting the same value is quite important. For that purpose, [12][13] uses the *synopsis*. In synopsis, a path along which a value propagated is considered such that the value will not be counted more than once.

5.2 Sending Queries

When generating a tree, it is possible to exploit a routing algorithm such as [4][6], or to flood a packet containing Q.

While doing flooding, it is also possible to omit some Q packet [9][10]. Because each packet is broadcasted in sensor networks, if a node $P(N_i)$ sends its sensed value to $P(P(N_i))$, N_i can snoop the packet. If N_i eavesdrops such packet, N_i will be able to figure out Q and send its sensed value accordingly. In this way, some packets required for sending Q can be omitted.

It is also possible to designate a leader node which is responsible for sending Q in its regions [17]. In that case, because the node is a leader, Q and the plan for processing Q are sent to the nodes by the leader.

5.3 Synchronization

Because wireless communication in sensor networks is unreliable, there's no trivial way for determining how long $P(N_i)$ should wait before N_i sends its value. This is where synchronization problem comes in. A trivial solution, *periodic simple* [16], is to set a fixed amount of time for a node to wait, but it is not flexible enough.

To solve this problem, TinyDB [9][10] proposed *pipelined aggregate*. In that scheme, queries are answered continuously. Because each value of N_i is sent to $P(N_i)$ continuously, missing some value of N_i does not matter.

Another problem is to make $P(N_i)$ time out no earlier than N_i. Otherwise, lots of data will be lost. Therefore, [16] proposed *cascading time out*. In the scheme, each node has different timeout value according to its position in G.

5.4 Topology Optimization

Because energy efficiency is one of primary concerns in sensor networks, a tree optimization algorithm which considers *health*, batteries remained in this case, of a node was proposed [7]. In the scheme, a tree is built using flooding. After then, the tree can change its shape based on the health of the neighbor nodes. For example, let N_1 be the parent of N_2 and N_3. Also, suppose that N_4 is the sibling of N_1. Each nodes informs other nodes of its health periodically. Based on that information, N_1 can determine whether to continue its role as the parent of N_2 and N_3, or to transfer the role to N_4. In the end, network life can be elongated.

Challenges in multi-path graph is different from that of tree. Multi-path approach targets minimizing errors. Hence, it is important to set accuracy as the parameter supplied by the administrator. In case of [12], G comprises of a tree region and a multi-path region. The ratio between the number of nodes participating each region is determined by comparing the error and the user supplied parameter.

5.5 Communication Frequency

Depending on applications, the answer for Q need to be delivered once or many times. Also, the frequency between successive answer must be adjusted according to the users' interest.

In case of TinyDB [9], answers are continuously delivered to the users to handle synchronization issue mentioned above. Thus TinyDB can not but suffer from unnecessary communications.

TiNA [14] tries to solve this problem by letting users set the *epoch duration* and *temporal coherency tolerance*. Epoch duration means interval at which data should be propagated, and temporal coherency tolerance sets the accepted error rate. Each node compare its previous value and the current value, and then determine whether to send new value based on temporal coherency tolerance.

TiNA also has another advantage in terms of synchronization. Because $P(N_i)$ knows the previous value of N_i, $P(N_i)$ can use the previous value of N_i if N_i fails. Thus error rate plays an important role in synchronization rather than explicit time out value.

5.6 Aggregation Operators

Simple aggregations include AVERAGE, SUM, and COUNT and implemented in [9][10][17]. However, there are other complicated aggregations like MEDIAN and Wavelet Histogram.

In [15], *q-digest* based MEDIAN operation is proposed. q-digest can be though of as a histogram whose buckets contains the number of occurrences of each values. However, sending entire histogram might be comparable to sending entire value if the size of bucket is not wisely managed. To solve this problem, q-digest assume that individual sensor values do not hold much importance and that extracting all values is not efficient. Based on these assumptions, q-digest merge two buckets if the number of occurrences in each bucket is smaller than the predefined threshold.

On the other hand, [3] explains wavelet histogram applied to TinyDB. In the scheme, a piecewise wavelet histogram building algorithm is proposed. In addition, to avoid floating point arithmetics, which is not desirable considering the limited computation power of sensors, they tried to apply integer wavelet wherever possible.

5.7 Group By and Having

Group by and Having operators are SQL standard, so it is widely used in in-network aggregation. It is obvious that it is needed to maintain aggregation for each group separately. However, just doing so will increase the number of packets, decreasing network life time. Also, the size of aggregated result might be larger than the memory of sensors, making it impossible to store the result within each sensor.

To avoid such situations, TinyDB [3] partition sensor nodes according to group, and then compute f within each group. If there is not enough memory, then some groups are evicted [9]. TiNA [14] maintains different groups and do not pay attention to this problem.

Works done by Beaver et al. [1] stands out in this problem where trees are built according to the group. In detail, a node broadcasts its group ID to the neighbors. A neighbor node become a child of the broadcasting node if its group ID coincides with the ID of broadcasting one.

6 Conclusion

In this paper, a general framework of in-network aggregation and its variations are explored. Table 1 shows overall architectural options that can be chosen when designing in-network aggregation in sensor networks. Note that options in each category is not disjoint; thus several of them can be applied together.

Table 1. Architecture options of in-network aggregations

Criteria Category	Options
Graph Topology	Tree
	Multi-Path
Sending Queries	Flooding
	Exploiting broadcasting characteristics of wireless network
	Leader and its region based approach
Synchronization	Periodic simple
	Pipelined aggregate
	Cascading timeout
Topology Optimizations	Battery based reorganization
	Adjusting the ratio between tree region and multi-path region
Communication Frequency	Only once
	Continuously
	Epoch duration with temporal coherency tolerance
Aggregate Operators	AVERAGE, MIN, MAX, q-digest, Wavelet Histogram
Group By and Having	Maintaining groups separately
	Evicting some groups if necessary
	Group based tree organization

References

1. J. Beaver, M. A. Sharaf, A. Labrinidis, and P. K. Chrysanthis, "Location Aware Routing for Data Aggregation for Sensor Networks", *In Post Proc. of Geo Sensor Networks Workshop*, 2003.

2. J. Considine, F. Li, G. Kollios, and J. Byers, “Approximate Aggregation Techniques for Sensor Databases”, *In Proc. of ICDE*, 2004.
3. J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek, “Beyond Average: Toward Sophisticated Sensing with Queries”, *In Proc. of IPSN*, 2003.
4. C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, “Impact of Network Denstity on Data Aggregation in Wireless Sensor Networks”, *In Proc. of ICDCS-22*, 2001.
5. P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, “Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet”, *In Proc. of ASPLOS*, 2002.
6. J. Kulik, W. Rabiner, and H. Balakrishnana, “Adaptive Protocols for Information Dissemination in Wireless Sensor Networks”, *In Proc. of Mobicom*, 1999.
7. R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. ramachandran, “DFuse: A Framework for Distributed Data Fusion”, *In Proc. of SenSys*, 2003.
8. C. Lin, C. Federspiel, and D. Auslander, “Multi-sensor Single-Actuator Control of HVAC Systems”, *In Proc. of Intl. Conf. for Enhanced Building Operations*, Oct. 2005.
9. S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler, “Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks”, *In Proc. of WMCSA*, 2002.
10. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks”, *In Proc. of OSDI*, 2002.
11. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, “Wireless Sensor Networks for Habitat Monitoring”, *In Proc. of ACM WSNA*, 2002.
12. A. Manjhi, S. Nath, and P. B. Gibbons, ‘Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams”, *SIGMOD*, 2005.
13. S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson, “Synopsis Diffusion for Robust Aggregation in Sensor Networks”, *In Proc. of SenSys*, 2004.
14. M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis, “TiNA: A Scheme for Temporal Coherency-Aware in -Network Aggregation”, *In Proc. of MobiDE Workshop*, 2003.
15. N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, “Medians and Beyond: New Aggregation Techniques for Sensor Networks”, *In Proc. of SenSys*, 2004.
16. , I. Solis and K. Obraczka, “In-entwork Aggregation Trade-offs for Data Collection in Wireless Sensor Networks”, *INRG Technical Report 102*, 2003.
17. Y. Yao and J. Gehrke, “Query Processing in Sensor Networks”, *In Proc. of CIDR*, 2003.
18. Y. Yao and J. Gehrke, “The Cougar Approach to In-Network Query Processing in Sensor Networks”, *SIGMOD Record*, Vol. 31, No. 3, Sep. 2002.