Reliable Data Aggregation for Real-time Queries in Wireless Sensor Systems

Kam-Yiu Lam[1], Henry C.W. Pang[1], Sang H. Son[2] and BiYu Liang[1]

Email: cskylam@cityu.edu.hk, henry@cs.cityu.edu.hk, son@cs.virginia.edu

Department of Computer Science[1]
City University of Hong Kong
83 Tat Chee Avenue, Kowloon
HONG KONG

Department of Computer Science[2]
University of Virginia
Charlotte, Virginia
USA

Abstract

In this paper, we study the reliability issue in aggregating data versions for execution of real-time queries in a wireless sensor network in which sensor nodes are distributed to monitor the events occurred in the environment. We extend the *Parallel Data Shipping with Priority Transmission (PAST)* scheme to be workload sensitive (the new algorithm is called *PAST with Workload Sensitivity (PAST-WS)*) in selecting the coordinator node and the paths for transmitting the data from the participating nodes to the coordinator node. PAST-WS considers the workload at each relay node to minimize the total cost and delay in data transmission. PAST-WS not only reduces the data aggregation cost significantly, but also distributes the aggregation workload more evenly among the nodes in the system. Both properties are very important for extending the lifetime of sensor networks since the energy consumption rate of the nodes highly depends on the data transmission workloads.

1 Introduction

In this paper, we study the use of in-networking processing approach [MFH03] for processing of *real-time queries* which access to sensor databases maintained by sensor nodes distributed in the system to generate timely responses if certain events are detected or emergency situations occur [SKH03, YG03]. If the communication workload is concentrated on some nodes, not only the energy consumption rate of the nodes will be heavy, the message loss problem will also be very serious due to high collision probability in data transmission. It is quite common that the sampled value for a data item may contain errors due to noises. Thus, the result generated from a query may contain error too if the data items accessed by the query contain error. In [LPSL04], a *parallel data shipping* scheme, called PAST, is proposed to gather the right versions of data items using the time-stamping method for a real-time query at a coordinator node so that they are relatively consistent with reduced data transmission cost. In this paper, we extend PAST by considering the data communication workload among the relay nodes in choosing the path for collecting sensor data for execution of a query. Our objective is to satisfy the constraints of the queries and at the same time to minimize the communication overhead and improve the reliability in data communication by evenly distribute the communication workload in the system.

2 System Model

A wireless sensor system consists of a *base station* (BS) and a collection of *sensor nodes* distributed in the system environment which is divided into a number of square grids with length of *r* as shown in Figure 1. It is assumed that the nodes within the same grid capture the same signals of their surrounding environment. The length *r* of a grid is defined such that a node can directly communicate with all the nodes in its neighboring grids. Each sensor node generates sensor data values following a pre-defined sampling period which is defined based on the dynamic property of the sampled entities. A real-time query T_i can formally be defined as a tuple: $\{D_i, Op_i, <_i, O_i, \Delta_i, R_i\}$. Op_i is the set of read operations with each operation access to a sensor data item (O_i). To simplify the discussion, it is assumed that the required data items of a query are defined at the *grid* level. The set of operations Op_i in a real-time query is associated with precedence constraints ($<_i$) on their execution orders. Due to the responsive nature of a real-time query and the dynamic nature of the system environment, it is important that the values of the data items accessed by a real-time query are representing the *current* information ("real-time status of the entities") in the environment. Each real-time query has a *currency* requirement (Δ_i) on its set of data items. Failing to meet the requirement implies that they are too "*old*" and not correctly describing the current situation. Since a timely response is critical to important events occurring in the environments, each real-time query is given a *deadline* on its completion time. In addition to meeting the deadline and currency requirement, another important issue is the reliability of the results generated from a query. As the query result generated from a set of data items may contain errors, it is important to provide multiple results by accessing multiple data versions of data items in processing a real-time query to improve the reliability and accuracy of the results. Therefore, a real-time query is associated with a result interval (R_i), which specifies the time interval of data items for generating the results.

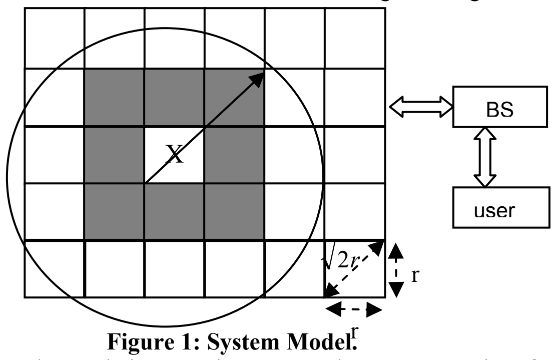

Figure 1: System Model.

In this paper, we adopt relative consistency as the correct notion for ensuring the correctness of the results and meeting the currency requirement of a real-time query [LP04, SBLC03]. Each data version of *x* is assigned a time-stamp at its generation time to indicate the start time of the validity of the data version. It will become invalid when the next version is generated. We use a time bound, *upper valid time (UVT)* and *lower valid time (LVT)* to label the validity interval of a data version. The set of data items for execution of a real-time query are *relatively consistent* if they are temporally correlated to each other, i.e., representing the status of entities in the environment at the *same time point*.

Relative consistency: Given a set of data versions V from different data items, the versions in V are relatively consistent if $\bigcap\{VI(x_i) \mid x_i \in V\} \neq \Phi$, where $VI(x_i) = [\,LVT(x_i), UVT(x_i)]$.

To meeting relative consistency requirement, the deadline constraints and currency requirement, for query T_i, the validity of all its accessed data versions should not be earlier than ($D_i - \Delta_i$), where D_i and Δ_i are the deadline and the currency requirement of T_i, respectively, i.e. $(\bigcap\{VI(x_i) \mid x_i \in V\}) \bigcap [D_i - \Delta_i, D_i] \neq \Phi$. The time window ($D_i$ to ($D_i - \Delta_i$)) is called the *valid time window* for the set of valid results of the query.

3 A Parallel Data Shipping Scheme - PAST

In *Parallel Data Shipping with Priority Transmission (*PAST) [LPSL04], the participating nodes of a query submit data versions to a carefully selected coordinator node in a *parallel* and *synchronized* fashion. The submission of data versions from the participating nodes is synchronized depending on the farthest participating nodes from the coordinator node.

Once the base station receives a real-time query, it will determine: (1) which node (grid) should be assigned as the coordinator node such that the total transmission cost of the data versions from the participating nodes to the coordinator node is minimized, and (2) which data versions from each participating node should be sent to the coordinator node. The data transmission delay from a participating node to the coordinator node is measured in terms of the number of hops in communication between them as we assume that the data transmission delay for sending a data version through one hop is a constant t_d. Let $G_{all} = \{g_1, g_2, g_3, \ldots, g_n\}$ be the set of grids in the system and F_{ij} is defined as the distance (in number of hops) between grid i and grid j where $i, j = 1, \ldots, n$. Suppose T_i wants to access to u grids/nodes and its required nodes are in the grids set $G_i = \{g_{i1}, g_{i2}, g_{i3}, \ldots, g_{i,u}\}$ and $u = |G_i|$. Let F_{totalX} be the total transmission length defined in terms of hops for choosing grid X as the grid where the coordinator node is residing. Then, $F_{totalX} = \sum_{j \in G_i} F_{X,j}$. Let D_{max} be the maximum data transmission delay of all the participating nodes of a query measured in grid using the shortest distance. The set of data versions to be submitted from each participating node is those data versions which are valid within the interval from $(D_i - C_i - D_{max} - R_i)$ to $(D_i - C_i - D_{max})$. The maximum number of hops H_i of the participating nodes from the coordinator node is then: $H_i = D_{max} / t_d$. Let the coordinates of a grid k be (X_k, Y_k) and $S_{g_{ik}}(H_i)$ be the set of grids which can be reached by the data versions originated from g_{ik} with a distance of no more than H_i:

$$S_{g_{ik}}(H_i) = \left\{ g_{ij} \middle| F_{g_{ik},j} \le H_i, j \in G_{all} \right\} \qquad \text{eqn. (1).}$$

Eqn. (1) defines a square region with a participating node as the central point of the square and the boundary is H_i hop counts from the grid where the participating node is residing. Then, we calculate the coordinates of the coordinator node which is within the intersect regions of all the participating nodes to minimize the total hop counts is getting all the data items from the participating nodes.

Once the coordinator node and the set of data versions for transmission have been determined, the information together with result interval requirement R_i will be sent to the participating nodes. The transmission of data versions from the participating nodes to the coordinator node through the relay nodes are prioritized so that the arrival time of the data is close to the expected time. The priority of a data message M_i for query T_i at node N_j is calculated as: (D_i – Current time) / number of hops from N_j to the coordinator. A higher priority is assigned to a data message for transmission if the calculated value is smaller. The query will be processed at the coordinator node according to the order of the operations defined in the query and following the relative consistency requirement.

4 Workload Sensitive Data Aggregation – PAST-WS

In this section, we introduce the extension of PAST, *PAST with Workload Sensitivity (PAST-WS)*, with purpose to improve the reliability and to reduce the cost and delay in data transmission from the participating nodes to the coordinator node [IEGH02]. Although the total aggregation distance defined in hop counts from the participating nodes to the coordinator node is minimize in PAST, it has ignored the data loss problem in choosing the coordinator node and the aggregation paths. PAST-WS resolves this problem by calculating the mean number of data re-transmissions in choosing the coordinator node and the aggregation paths instead of using the physical hop counts. Another important benefit of the proposed scheme is that the data transmission workload will be more evenly distributed among the nodes in the system. Thus, the energy consumption rate of each node will remain similar over the network, enabling a longer system lifetime.

4.1 Error Modeling for Data Aggregation in Sensor Networks

After determining the coordinator node using PAST, the base station will determine the paths and the start times for the participating nodes to submit their data versions. Since the grids are in a square shape, the shortest path defined in terms of hop counts to the coordinator node can easily be calculated, i.e., it is the shortest line connecting the participating node and the coordinator node using a shortest path searching algorithm. However, this may not be the best one in terms of number of communication messages and total transmission time due to retransmissions. In particular, if multiple sensor nodes want to send messages to the same node *N* at the same time (or within its transmission time), the receiver *N* may not be able to receive all of them due to collisions. For the calculation of the error probability of message loss, the base station maintains an array indicating the current relay workload of each node in processing real-time queries in the system. The begin time and the end time of the activated queries are also recorded. When the end time of a query is expired, the array will be updated accordingly.

We model the probability of message loss at a node P_n as a function of the number of nodes (n) concurrently sending data to it. We assume that the transmission delay (S) and transmission period (P) are the same for all senders. For the case $n = 2$, i.e. there are two senders to the same receiver, the probability of having a conflict in transmission is $p2 = S/P$. This can be observed in Figure 2. If *node* 2 sends a message during the first conflict time interval (marked grey in Figure 2), the message from *node* 2 may be lost since the receiver is receiving a message from *node* 1. Within a period, if a message from *node* 1 is sent after the first conflict time interval, there will be no loss of *node* 2 message. So the message loss probability for the case of having two senders is S/P.

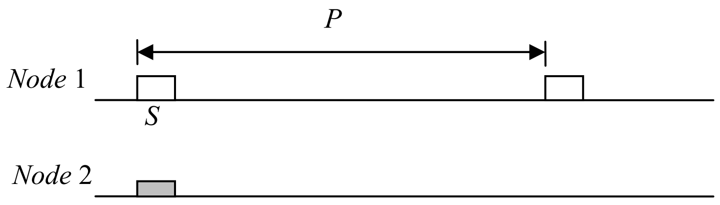

Figure 2: Data Collisions Probability for two senders.

In an interval of time P, at most $m = \lfloor P/S \rfloor$ messages can be sent, i.e. a period can be divided into m small time intervals in which only one message can be sent. Since there are n (suppose $n < m$) senders, there are m^n number of combinations for choosing message transmission intervals for these nodes. Among these m^n possibilities, there are $C_m^1 \cdot C_{m-1}^1 \cdots C_{m-(n-1)}^1 = m(m-1)\cdots(m-n+1) = \frac{m!}{(m-n)!}$ combinations that will not cause any message loss .Thus the probability of message loss from any node is:

$$1-(1-p_n)^n = 1-\frac{m!/(m-n)!}{m^n}$$

That is:

$$p_n = 1-\left(\frac{m!}{m^n \cdot (m-n)!}\right)^{1/n} \qquad \text{eqn. (2).}$$

We can solve equation (2) to get p_n for different values of n. After we do this for $n = 1, 2, \ldots, m$, we get the distribution of loss probability.

To calculate the average transmission cost for a single hop (measured in hop counts), we assume that the receiving node is the relay node of k nodes. Then, the loss probability of message sending to the receiving node is p_k. So the probability of successfully sending data to the receiving node by sending once is $1-p_k$. The probability that the sender needs to send twice (i.e. the first message sent is lost, and the second one is received) is $p_k(1-p_k)$. Similarly, the probability that the sender need to send K times is (i.e. the first $(K-1)$ transmissions are all lost, and the final one is received) is $p(K) = p_k^{K-1}(1 - p_k)$. So the expected message cost of sending one message through a single hop to the receiving node is:

$$C_{hop} = \sum_{K=1}^{N} K \cdot p(K) = \sum_{K=1}^{N} K \cdot p_k^{K-1}(1 - p_k) \qquad \text{eqn. (3).}$$

The cost of a path C_{path} is the sum of the costs of all the hops in the path. When the probability of message loss is considered in calculating the propagation cost, the message transmission delay of a path is no longer a constant proportional to number of hop counts. It is a random variable and larger than that of no message loss case. We can estimate the average number of times a message that has to be sent in one single hop. In order to calculate the mean delay of a path, we need to estimate the mean time length of intervals between consecutive message resend events. The probability distribution of the number of successive message loss is the same as the distribution of the number of resends, i.e. $p(K)$ discussed in eqn. (2). Thus the average time length between two consecutive retransmissions is:

$$\sum_{K=1}^{N} K \cdot P \cdot p(K) = P \cdot C_{hop} \qquad \text{eqn. (4)}$$

We get the expected delay D_{hop} of a single hop by multiplying it with the average number of retransmissions C_{hop}, i.e. $D_{hop} = P \cdot C_{hop}^2$.

4.2 Calculating the Aggregation Path and Coordinator Node

In choosing the path for data propagation, we need to ensure that the expected delay satisfies the currency requirement such that $(D_i - C_i - D_{hop}) > 0$. Algorithm 1 shows the steps of finding the coordinator node and the best path to forward the data versions to the coordinator node. If message loss is considered, the delay is larger than that of no message loss. The set of possible coordinators under the case of message loss is a subset of that of the case with no message loss. In this way, we exclude most of the impossible candidates for the coordinator node. Assuming a straight path (the shortest connection path between a participating node and the possible coordinator node), we find a coordinator node satisfying the currency requirement with the minimum cost. Finally, we find a feasible replacement of the maximum for each path; and for each replacement, we calculate the reduction in cost. We choose the replacement with the maximum cost reduction. The final step is repeated until there is no feasible replacement.

Objective: To find the coordinator node and the paths from the participating nodes of T_i with total minimum communication cost.
Inputs: The node status of all the participating nodes: n (number of receivers), S (mean message delay to send a data version) and P (mean data transmission period); $G_i = \{G_{i1}, G_{i2}, G_{i3}, \ldots, G_{iu}\}$, R_i
Outputs: The coordinator node and the set of paths from the participating nodes with minimum communication cost.

```
Call Algorithm 1 (PAST) to find the set of possible coordinator S;
    for each coordinator node c_node in S
    { /* exclude non-candidate coordinators from S */
        for each participating node p_node in $G_i$  {
            path = the straight path from c_node to p_node;
            $D = \sum_{H \in path} D_{hop}(H)$ ;
            if ($D > \Delta_i - C_i - R_i$) then    {
                $S = S - \{c\_node\}$;  /* cannot be a coordinator */
    break;  /* break and continue to check the next c_node */}
```

```
                }
        }
        if( $S$ ==$\Phi$) then abort;  /* no feasible solution */
        $C_{min}$ = infinity;
        for each node $c\_node$ in $S$ do
        { /* assuming a straight path (the shortest path), find the coordinator node with minimum cost */
            $C_{total}$ = 0;
            for each participating node $p\_node$ in $G_i$
              { $path$ = the straight path from $c\_node$ to $p\_node$;
                    $C_{path}$ = Sum of $C_{hop}$ of each hop of the $path$;
                    $C_{total} = C_{total} + C_{path}$;}
                    if ($C_{total} < C_{min}$)  {
                            $coordinator\_node = c\_node$;
                            $C_{min} = C_{total}$;}
        }
        $S_{path}$ = the set of straight path (the shortest path) from participating nodes to $coordinator\_node$;
        do
        { /* adjust the paths */
            $C_{max}$ = 0;
            $R_{path}$ = NULL;  /* path to be replaced */
                $R_{max}$ = NULL;  /* path which will replace $R_{path}$ */
                $F$ = false;
            for each $path$ in $S_{path}$
            {$C_R$ = 0;
                for each replacement $r$ of $path$ {
                    if($r$ satisfies delay constraint AND $C_{path} - C_r + \Delta C_{decrease} - \Delta C_{increase} > C_R$ ){
                    $C_R = C_{path} - C_r + \Delta C_{decrease} - \Delta C_{increase}$; /* cost reduction */
                    $R = r$;}
                }
              if($C_R > 0$ ) {
                            $F$ = true;
                if($C_R > C_{max}$)
                {   $R_{max} = R$;
                    $C_{max} = C_R$;
                    $R_{path} = path$;}
              }
            }
            $S_{path} = S_{path} + \{R_{max}\} - \{R_{path}\}$; /* replace the path $R_{path}$ with $R_{max}$ */
        } while($F$ == true);
        return $S_{path}$, $coordinator\_node$;
```
Algorithm 1: Finding the coordinator node and the path loss.

5 Performance Results

Figures 3 through 6 show the results when we vary the size of a real-time query. As shown in Figure 3, increasing the query size (number of grids), the data transmission workload will be increased. Comparing with PAST, the data transmission workload of PAST-WS is consistently lower as shown in Figures 3 and 4. Figure 5 and Figure 6 show the distribution of data transmission workload of the nodes in the system. It can be seen that the workload is more evenly distributed in PAST-WS than in PAST. The numbers of heavy and medium loaded grids in PAST-WS are smaller than in PAST. In addition, we have measured the mean value and variance in workload of the nodes. Consistent with the results in Figures 5 and 6, both the mean and variance of PAST-WS are smaller than that of PAST.

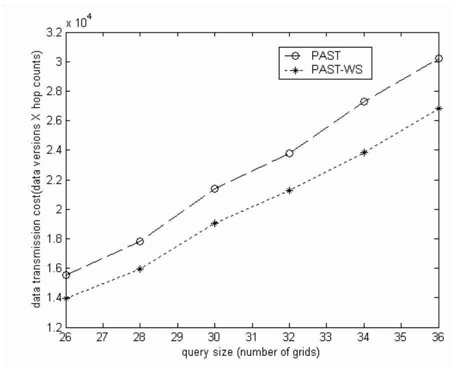

Figure 3: Query Size Vs. Data transmission cost

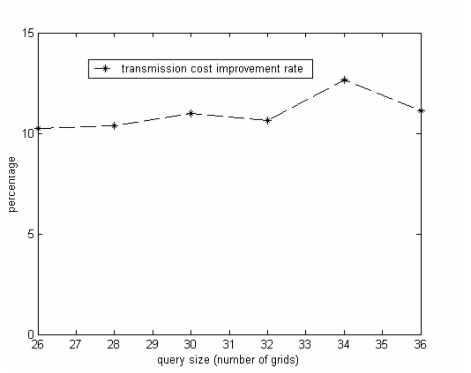

Figure 4: Percentage improvement of PAST-WS

Figures 7 and 8 show the results of PAST-WS and PAST respectively when we vary the currency requirement of a query. We can see that PAST-WS only not gives a smaller transmission cost, it can complete more queries successfully, i.e., meeting the deadline, currency and result requirements. In PAST, due to long aggregation time and heavy workload as a result of re-transmissions, a large number of queries can only be partially completed and some of them are even failed, i.e., no results are generated, especially when the currency requirement is tight. The situation is less serious in PAST-WS as shown in Figure 7 as its data transmission workload is lower after considering the workloads of the relay nodes in choosing the coordinator node and the relay nodes. We also have investigated the impact of varying the locality factor of a query to their performance. (Due to space limitation, we do not show the result figures.) Similar to the results discussed before, PAST-WS shows a better performance.

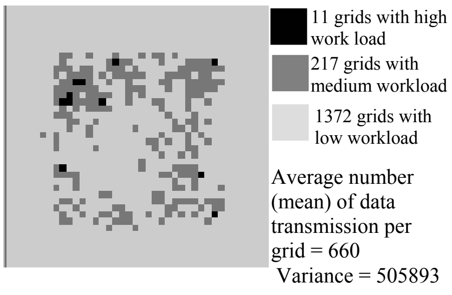

Figure 5: Distribution of transmission workload (PAST-WS)

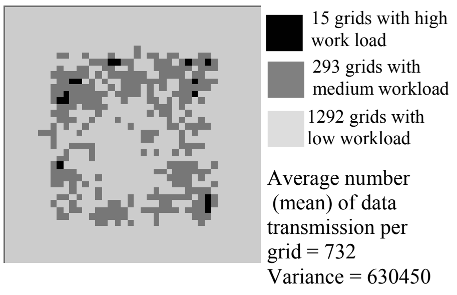

Figure 6: Distribution of transmission workload (PAST)

6 Conclusions

In this paper, we have studied how to improve the reliability in data aggregation for execution of real-time queries in a wireless sensor system. The real-time queries are associated with a deadline on their completion times and it is important to generate the results before the deadlines since it is mainly for generating responses to the events occurred in the system. To meet the query processing requirements with minimum data transmission cost, a parallel execution scheme, called PAST was proposed. However, the workload at the relay nodes was not taken into consideration in selecting the coordinator node and the aggregation paths. If the workload at the relay nodes is heavy, the data loss probability will be high and the consequence is either some data are lost or a lot of re-transmissions are required. A lot of re-transmission not only increases the energy consumption rate at the relay nodes, but also increases the data transmission workload in the system and the delay in gathering the data versions for processing of the queries. In this paper, we extend the PAST to include a workload sensitive scheme in selecting the coordinator node and the paths for data aggregation. The new algorithm is called PAST-WS. Simulation results have shown that PAST-WS can significantly reduce the aggregation workload and delay and at the same time can distribute the aggregation workload evenly in the system.

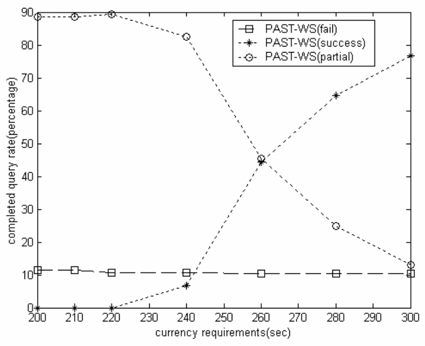

Figure 7: Currency Vs. Completed query percentage(PAST-WS)

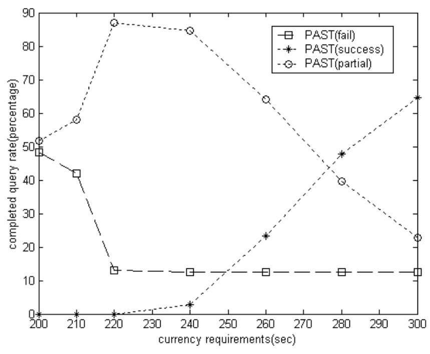

Figure 8: Currency Vs. Completed query percentage (PAST)

References

[IEGH02] C. Intanagonwiwat, D. Estrin, R. Govindan and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks", in *Proceedings of ICDCS'02*, Vienna, Austria, July 2002.

[LP04] Kam-Yiu Lam and Henry C.W. Pang, "Correct Execution of Continuous Monitoring Queries in Wireless Sensor Systems", in *Proceedings of the Second International Workshop on Mobile Distributed Computing (MDC'2004)*, Tokyo, Japan, March 2004.

[LPSL04] Kam-Yiu Lam, Henry C.W. Pang, Sang H. Son, BiYu Liang, "On Using Temporal Consistency for Parallel Execution of Real-time Queries in Wireless Sensor Systems", Technical Report, Department of Computer Science, City University of Hong Kong (www.cs.cityu.edu.hk/~henry).

[MFH03] S. Madden, M. J. Franklin and J.M. Hellerstein, "The Design of an Acquisitional Query Processor For Sensor Networks", in *Proceedings of SIGMOD 2003*, June 9-12, San Diego, CA.

[SBLC03] Mohamed A. Sharaf, Jonathan Beaver, Alexandros Labrinidis, Panos Chrysanthis, "TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation", in *Proceedings of 2003 International Workshop in Mobile Data Engineering.*

[SKH03] Narayanan Sadagopan, Bhaskar Krishnamachari and Ahmed Helmy, "Active Query Forwarding in Sensor Networks (ACQUIRE)", to appear in *Ad Hoc Networks*.

[YG03] Y. Yao and J. E. Gehrke, "Query Processing in Sensor Networks", in *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, California, January 2003.