Adaptive Target Tracking in Sensor Networks

Xingbo Yu, Koushik Niyogi, Sharad Mehrotra, Nalini Venkatasubramanian
University of California, Irvine
$\{xyu, kniyogi, sharad, nalini\}@ics.uci.edu$

Abstract

Recent advances in processor, memory and radio technology have enabled cheap nodes capable of sensing, communication and processing. Networks of distributed microsensors are rapidly emerging as a feasible solution to a wide range of data gathering applications. However, the key impediments to successful deployment of microsensor nodes are their energy and longevity constraints. We propose a quality aware information collection protocol in sensor networks for tracking applications. The protocol explores trade-off between energy and application quality to significantly reduce energy consumption in the sensor network thereby enhancing the lifetimes of sensor nodes. Simulation results over elementary movement patterns in a tracking application scenario strengthen the merits of our adaptive information collection framework.

Key Words: wireless sensor networks, mobile target tracking, quality of service

1 Introduction

With advances in computation, communication and sensing capabilities, large scale sensor-based distributed environments are emerging as a predominant mobile computing infrastructure. Sensor applications typically monitor real-world phenomena and utilize the accurate knowledge of current state to dynamically optimize application execution. Traditionally, a critical issue in enabling wide-spread use of mobile computing is that of effective power management to extend battery life. This is further exacerbated in sensor environments where it is often difficult or infeasible to replenish power supplies of wireless devices. Power supply in sensors is used for a variety of purposes: for basic sensing operations, for powering the memory and CPU, and for communication. While the specific rate of energy consumption for each of these operations is sensor and application specific, much of the existing literature indicates that communication constitutes a major source of power drain[4]. Modern sensors try to be power-aware, shutting down components (e.g., radio) when they are not needed and/or scaling the frequency/voltage of the processor depending upon the workload [3, 7].

While well engineered sensor technology can significantly conserve power, for a variety of applications much further gain can be accrued by exploiting the natural tradeoff between application quality and energy conservation. In this paper, we explore a framework that exploits resource versus quality tradeoff to reduce communication in the context of information collection for mobile target tracking in sensor environments. The proposed framework exploits the application tolerance by collecting data at only the accuracy levels needed to meet the application's requirements. The greater the application's tolerance to error in sensor values, the lower the communication overhead between sensors and server, and hence higher the resulting savings.

The rest of the paper is organized as follows. Section 2 introduces sensor nodes and target tracking using sensor networks. Section 3 describes adaptive protocols proposed in this paper. Section 4 illustrates the savings that result from our protocol by a set of simulations. Related work is presented in Section 5.

2 Quality Aware Information Collection and Tracking Applications

2.1 Sensor Nodes and Sensor Networks

The overall environment for adaptive information collection is illustrated in Figure 1, where applications employ information collection protocol to obtain the desired information from a distributed sensor network infrastructure.

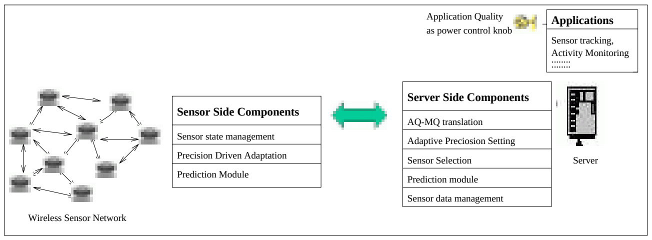


Figure 1. Information collection architecture

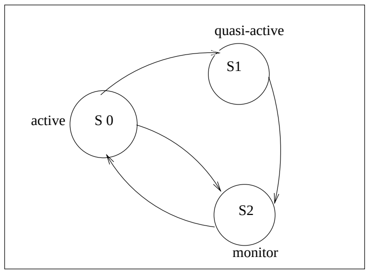


Figure 2. State transition diagram for sensor node in target tracking

A sensor node consists of the embedded sensor, a processor with some limited memory and the radio circuitry [3]. Each component is controlled by a micro operating system that decides which device to turn off and on. A power aware sensor node has 3 different states of operation (see Figure 2). While in state $\mathcal{S}0$(active state), a sensor node transmits and receives messages at every time instant. State $\mathcal{S}1$(quasi-active state) is a reduced energy state in which the sensor sends a message to the server at a reduced frequency as explained later. The most energy efficient state $\mathcal{S}2$ is the monitor state in which the sensor turns its radio off and senses the environment for signals. The state transitions of the sensor are dictated by the adaptive information collection protocols. A sensor senses the environment at periodicity t_{sense} and communicates its readings to the server at periodicity t_{send}. For simplicity we assume that t_{send} is the same as t_{sense}. We consider time to be discretized into units of length t_{send} each. A sensor reading at time unit τ is represented as $I_i(\tau)$.

The system architecture considered in this paper is logical. Physically, the server architecture may itself be centralized, distributed or hierarchical. Furthermore, a part of the server module might reside on a computationally superior sensor node, which may be responsible for both sensing as well as fusing the readings from other sensor nodes. In the latter case, the server (super sensor node) may host both the sensor module as well as the server module. Furthermore, the issue of wireless network routing of data between the sensors and servers is orthogonal to our work and is abstracted out. One may have a direct communication link between the server and each sensor node, or a multi-hop routing scheme from a sensor node to the server as discussed in [11].

2.2 Mobile Target Tracking

In our application scenario, each sensor has an unique identifier, coordinates, and visibility radius. For low cost and low energy consumption, we assume passive sensors that operate only on the received acoustic or seismic waveforms from non-cooperative sources. A sensor senses the environment and communicates its readings to the server periodically; the server triangulates the location of the object using the readings. The signal attenuation of the target source power P is governed by equation $I = P/4\Pi r^2$, where r is the Euclidean distance between the object and a sensor node. This relation is the basic equation used in triangulation.

Let $\{\langle o_x[1], o_y[1]\rangle, \cdots \langle o_x[t_s], o_y[t_s]\rangle\}$ be the real track of object O and $\{\langle o'_x[1], o'_y[1]\rangle, \cdots \langle o'_x[t_s], o'_y[t_s]\rangle\}$ be its approximate track perceived by the server. We formally define the notion of track quality ϵ_{track} and sensor measurement quality below:

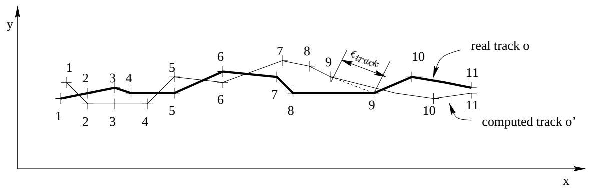


Figure 3. tracking quality

Track Quality: Track quality, ϵ_{track}, is defined as the maximum distance between the real track o of the target object and the approximate track o' generated at the user end(see figure 3):

$$\epsilon_{track} = max_{0 \leq \tau \leq n} |o[\tau] - o'[\tau]|.$$

Sensor Measurement Quality: If we consider the measurement at a sensor S as a time series $I[1:t]$ which is approximated by $I'[1:t]$, the *measurement quality* $|\Delta I|$ is the maximum divergence between $I[\tau]$ and $I'[\tau]$ at any time τ i.e :

$$|\Delta I| = max_{0 \leq \tau \leq t} |I[\tau] - I'[\tau]|$$

Given a user defined track quality parameter ϵ_{track}, the sensor network tracks an object with an error bound of ϵ_{track} with small communication overhead. We map track quality to measurement quality below, where we utilize a simple tracking approach of triangulating 3 sensor readings to determine the location of an object.

Consider three sensors whose locations $((x_1, y_1), (x_2, y_2), (x_3, y_3))$ are known. The intensity readings of the sensors at the time of triangulation are I_1, I_2 and I_3. We then have the following set of equations:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = \frac{p}{4\pi I_1} \\ (x - x_2)^2 + (y - y_2)^2 = \frac{p}{4\pi I_2} \\ (x - x_3)^2 + (y - y_3)^2 = \frac{p}{4\pi I_3} \end{cases}$$

Solving for x and y :

$$x = \frac{-b_2 c_1 + b_1 c_2}{b_2 a_1 - b_1 a_2} \quad (1)$$

$$y = \frac{-a_2 c_1 + a_1 c_2}{a_2 b_1 - a_1 b_2} \quad (2)$$

Where, $a_1 = 2(x_2 - x_1), b1 = 2(y_2 - y_1), c_1 = x_1^2 - x_2^2 + y_1^2 - y_2^2 - \frac{p}{4\pi}\left(\frac{1}{I_1} - \frac{1}{I_2}\right), a_2 = 2(x_3 - x_2), b2 = 2(y_3 - y_2)$, and $c_2 = x_2^2 - x_3^2 + y_2^2 - y_3^2 - \frac{p}{4\pi}\left(\frac{1}{I_2} - \frac{1}{I_3}\right)$.

For the above triangulation scheme, we can show that the tracking error tolerance ϵ_{track} is proportional to the ratio $\left|\frac{\Delta I}{I^2}\right|$ where I is the sensor measurement and ΔI is the deviation of sensor measurement (see [8] for derivation). Thus, we can compute $\left|\frac{\Delta I}{I^2}\right|$ from tracking quality ϵ_{track}. This provides, for each sensor S_i, a relative measurement error tolerance which is subsequently used in our adaptive protocols.

There are two issues with the above approach. Firstly, we observe that first_order approximations of Δx and Δy work well only when ΔI_i is small. As x and y are linear combinations of $\frac{1}{I_i}$'s, we can show that when $\Delta I_i < 0$, Δx is under-estimated and when $\Delta I_i > 0$, Δx is over-estimated. We overcome this problem by scaling $|\Delta I_i|$ as follows:

- If $\Delta I_i < 0$, then scale $|\Delta I_i|$ by $\frac{I_i}{I_i+\Delta I_i}$. Thus the new $|\Delta I_i|$ becomes $\frac{I_i^2\xi}{1+I_i\xi}$, where $\xi = \sqrt{\frac{\Delta d_{max}^2}{C}}$.
- If $\Delta I_i > 0$, then scale $|\Delta I_i|$ by $\frac{I_i}{I_i-\Delta I_i}$. Thus the new $|\Delta I_i|$ becomes $\frac{I_i^2\xi}{1-I_i\xi}$, where $\xi = \sqrt{\frac{\Delta d_{max}^2}{C}}$.

Secondly, in our tracking algorithm, the target source power was assumed to be constant. However, when the source power is unknown, tracking and error translation are more complicated. We discuss this issue in detail in [8].

3 Quality Aware Information Collection Protocols

As per the state transition diagram of a sensor node, at a particular instant of time some sensors will be active while the others will be in a *monitor* or *quasi-active* state. The sensor selection phase determines which sensors should be active at a particular time instant, and controls the state transition of the sensors from active to quasi active and monitor states. It also implements precision based measurement update. The following subsections describe the sensor selection process in detail at the server and the sensor side.

3.1 The Sensor Module

A sensor may be triggered by an external event to move from a monitor state $\mathcal{S}2$ to an active state $\mathcal{S}0$. An external event in a tracking application would be an object moving within the visible radius of a sensor node. In its active state, a sensor sends continuous updates at each time interval to the server module. An update packet consists of the sensor reading, timestamp and state of the sensor (e.g. energy statistics). The server module may decide to transit the sensor to a *quasi-active* state $\mathcal{S}1$ by sending the measurement tolerance ΔI corresponding to the desired application tolerance ϵ_{track}. The sensor now makes a transition to a *quasi-active* state, and sends an update to the server module only if its reading exceeds its previous reading by the measurement tolerance ΔI.

```
Sensor Module
Send Server Update
Input: $\tau$:time, $I_{(\tau)}$: reading at time $\tau$, $sensor\_state$: state of sensor node
       $I_{prev}$: previous reading, $\Delta I$:measurement error
Procedure Send_Server_Update($\tau, I_{(\tau)}, sensor\_state, I_{prev}, \Delta I$)
1. if((external event) and ($sensor\_state$=S0 or $sensor\_state$= S1)
2.         Send message to server to remove node from active list of server;
3.         $sensor\_state$=S2;
4. else if(external event)
5.         if($sensor\_state$=S2)
6.                 state=S0;
7.                 send_update_packet($sid, \tau, I_{(\tau)}, sensor\_state$);
8.         if($sensor\_state$=S1)
9.                 /*precision based adaptation */
10.                if($|I_\tau - I_{prev}| \geq \Delta I$)
11.                        send_update_packet($sid, \tau, I_{(\tau)}, sensor\_state$);
12.                        $I_{prev} = I_{(\tau)}$;
13.                else do nothing;
14. end procedure
```

Table 1. Sensor side algorithm for adaptation and state transition

The server on receiving the value update will make the corresponding update for the sensor measurement in its sensor database. A sensor may transit from the *active* or *quasi-active* states to the *monitor* state due to either an external event (e.g., sensing measurement falling below a threshold), or a message from the server. If the transition happens due to an external trigger, the sensor sends a corresponding message to the server. Table 1 shows the sensor module for precision based adaptation and state transition at a sensor node.

3.2 The Server Module

The server maintains an active list of sensor nodes which contains an array of sensors and a history of their readings over a time period. For the sake of simplicity, we bound the communication latency in the sensor network, i.e. the time taken for an intensity reading to reach the server by δt_l . Thus the network latency is less than δt_l for any node in the network. At time $t >= 0$, when the server receives an update from sensor S_i, it waits for δt_l (to receive all possible sensor readings at that time instant) and checks to see if the sensor is in mode $\mathcal{S}0$(active state). If so, it adds this sensor to its active list and sends an error update message containing the tolerable measurement error ΔI. However, if the sensor is in state $\mathcal{S}1$(quasi-active), the server looks up the active list to determine whether the sensor readings are already in use by applications at the server. If so, it adds the latest reading of sensor S_i along with its timestamp to the sensor's history list(see Table 2). Alternatively, if the sensor is not in the active list of the server (i.e., its value is not needed by the server), the server sends a message to the sensor to shift to the *monitor* state. On receipt of the message from the sensor that it has transited into *monitor* state, the server looks up its active list and removes the corresponding entry in the list.

The framework described above ensures that for each *active* sensor, the server maintains in its database an approximate measurement that is within the tolerable threshold of the actual sensor measurement. The reduced rate of communication between the sensor and the server in the *quasi-active* state and switching off of the radio in the *monitor* state result in energy conservation.

```
Server Module
Compute Error and Receive Sensor data
Input: $sid$: sensor id, $\tau$:time, $I_\tau$:sensor reading, $state$: sensor state
       $A_c$ :Active list of sensor nodes , $\epsilon_{track}$ : tracking quality
Procedure Recv_Data_Send_Error($sid, \tau, I_\tau$,state, $A_c, \epsilon_{quality}$)
1. if($state$=S0)
2.     $A_c$.add($sid$,$I_\tau$,$\tau$);
3.     Compute $\Delta I = f(\epsilon_{track}, I_{A_c})$;
4.     Send_error_update(sid, $\Delta I$);
5. else
6.     Look up $sid$ in $A_c$;
7.     if($A_c$.contains($sid$));
8.         Add $I_\tau$ to the corresponding entry in $A_c$;
9. end procedure
```

Table 2. Server side sensor selection protocol

A triangulation based tracking application program sits at the server above the proposed middleware framework. It takes n readings from the active list of the server module and pinpoints the location of the moving object with bounded tracking quality guarantees.

3.3 Prediction Models and Local Aggregation

The communication cost of the precision based approach can be further reduced by exploiting the predictability of readings of a sensor. If we consider a precision based approach, whenever the current location of the object deviates from its previous reported position by ϵ_{track}, the sensor has to communicate its readings to the server. If the sensor and the server can decide on the mobility model of the target object, they can predict its location in the near future and further communication savings can be achieved. In such a case, our approach can be modified such that a sensor reports its readings only if the observed value deviates from the expected value based on the prediction model by a pre-defined error bound ϵ_{pred}. Such prediction models may be integrated into the client and server modules of the information collection framework to gain further savings in energy. Many issues need to be resolved in incorporating prediction models, such as who determines the prediction model, the sensor or the server; the protocol by which a sensor or the server choose a model M; dynamic model switching when the prediction model needs to be changed. We address the above mentioned issues in [8].

In order to further reduce the long range communication cost between sensors and the server, local aggregation of information within the sensor network can be exploited. The key advantage of local aggregation is that a partial result obtained at a localized point may only be communicated to the server only if the change in the result is significant enough to violate the quality requirement. In target tracking, the result is a location from early triangulation. The issues need to be addressed include group formation and group leader election. [8] provides detailed discussion.

$M1$	The target walks along the line $x + y = d$ with a constant velocity $\bar{v}$(t).
$M2$	Restricted random motion : The target starts at $(0, d)$ and moves from one node to another randomly chosen node, until it walks out of the grid . While moving from node to another, its velocity is assumed constant.

Table 3. Mobility models

4 Performance Analysis

4.1 Simulation

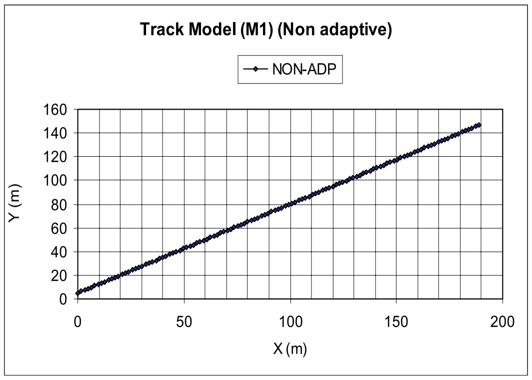
For our simulation experiments, the sensor network is modeled as a grid with nodes located at integer coordinate points. Time is modeled as a discrete quantity and the sensor nodes and the access points are assumed to be time synchronized with respect to each other. The server has superior computation and communication capabilities compared to the sensor nodes. Though we refer to a single grid and access point in our experiments and tracking protocols, the same set of protocols may be running on several such grids and access points to track a moving object in space and time. A grid based network topology was simulated to evaluate the performance of our adaptive tracking protocols with 400 sensor nodes with an internode distance d = 10m.

We study the performance of our algorithms in terms of the energy consumption at each individual sensor node and the entire grid over the period of time the object is tracked. For this purpose we use an energy model ($\mu AMPS$ project[3, 5]) where the energy dissipation for sensing and computation is negligible compared to that for communication overhead of the radio sub-system [4, 6, 2].

The performance is defined for the movement patterns for a target object. For this purpose we used two elementary and feasible movement patterns of a moving object (in line with the SenseIt signal processing tracking benchmarks, [13]) as shown in Table 3.

4.2 Results and Discussion

We use four separate protocols to compare the total energy consumption in the sensor network over the period of tracking. We use a non-adaptive algorithm **NON-ADP**, in which each sensor sends a reading at a fixed time



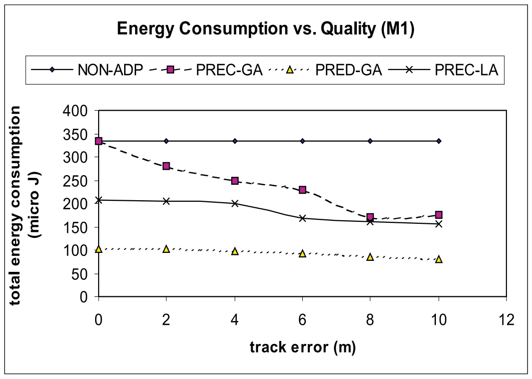
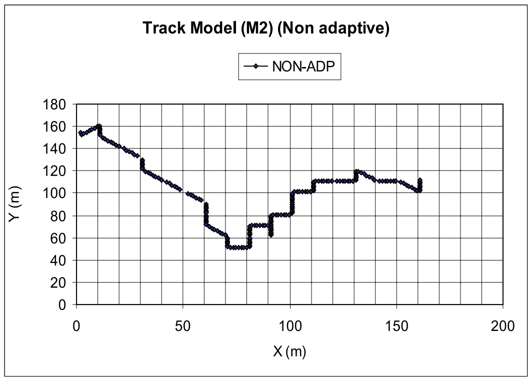


Figure 4. Precision and Prediction based experiments for mobility model M1



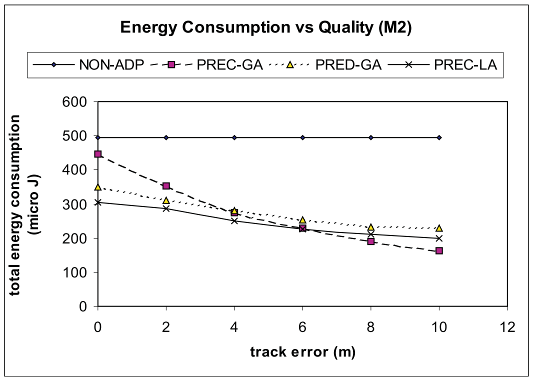


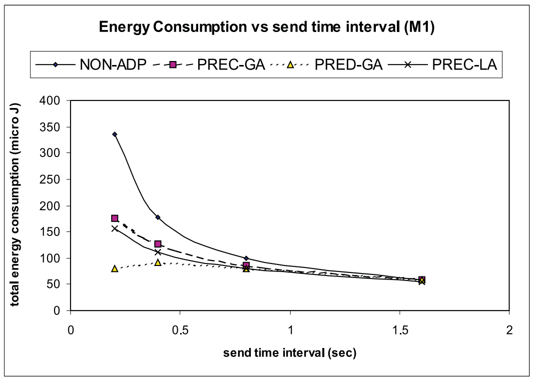
Figure 5. Precision and Prediction based experiments for mobility model M2

interval of t_{send} =0.2 seconds to the server. **PREC-GA** is a precision based adaptive protocol which has an user defined tolerance parameter ϵ associated with it. This means that the user specifies at the application level the quality at which he wants the object to be tracked. The application quality is mapped to the measurement quality, based on which a sensor sends an update to the server. In **PRED-GA**, we allow the sensor to fit a constant velocity based prediction model (see [8]) to track the object. The fourth algorithm **PREC-LA** implements the local aggregation mechanism where an leader node is elected among a group of sensors and each active sensor in the group sends its update based on its measurement error to the leader, who forwards it to the server only if the overall group quality is violated. Also, for the sake of simplicity we abstract our algorithm from the underlying routing algorithm which routes a packet from a sensor node to the leader.

We vary the error tolerance from 2 to 10 meters, and measure the energy consumption at each node and over the entire grid for the period of simulation. In each of these graphs the energy consumption is measured in terms of micro joules per bit. Figure 4 shows the results obtained for mobility model $M1$. The graphs show (a) the track of the object, (b) the variation of the total energy consumption per bit using the four algorithms described above. We get a significant savings (50-60%) in communication costs over the network (neglecting costs in switching from one active state to another). **PRED-GA** outperforms the precision based **PREC-GA** since the object follows a linear velocity model throughout its trajectory. However, at a certain track error, energy savings from prediction and precision based algorithms reach a saturation level as further optimizations are not feasible. **PREC-LA** outperforms **PRED-GA** since it performs lesser number of global server updates based on local computation at the leader node.

Figure 5 shows the same results for mobility model $M2$. We notice that using a prediction based algorithm **PRED-GA**, we quickly reach a saturation point and the precision based algorithm **PREC-GA** starts performing better at a certain value of track error. This is because the object now follows a more random path than in mobility model $M1$ which leads to model switching and more communication overhead at a sensor node when we use a prediction based approach.

Figure 6 shows the impact of send time interval and velocity of the moving object to energy savings at the sensor node. The first graph plots the energy savings with an increase in t_{send} or the time interval at which a node sends an update in the non-adaptive algorithm. We notice that at lower values of t_{send} we obtain significant energy efficiency. However, when t_{send} is sufficiently large, the quality of the true track as governed by t_{send} is not fine grained enough to be optimized. In the second graph, the velocity of the object was varied from 5 m/s to 15



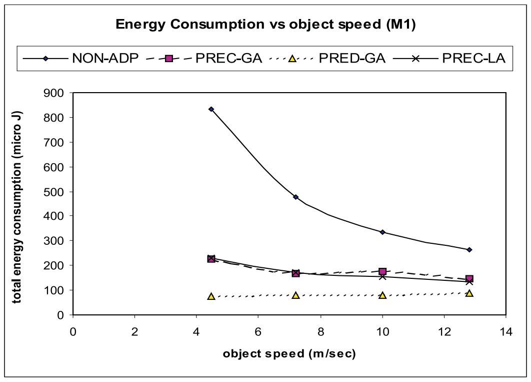


Figure 6. Effect of t_{send} and object velocity in mobility model $M1$

m/s while the track error was fixed at 10m and t_{send}=0.2 seconds for object model $M1$. At lower velocities, our energy savings are very high, since the sensor nodes in a grid cell send much fewer readings than in the non-adaptive case (both in case of **PREC-GA** and **PRED-GA**).

5 Related Work

The issue of quality and communication trade-off has been studied by Olston et al. [1] to reduce communication between a server and a single source. In [12], the authors extend their work to a best effort synchronization of source data objects and cached copies subject to bandwidth and other resource constraints at the source. The techniques did not take into account sensor state transition.

Several localized algorithms have been suggested in the literature [9, 10] to track an object using sensor collaboration. Zhao et al. [9] propose a distributed algorithm which routes a query related to the position of an object to a region of potential events, where a sensor node can answer it. The model of a sensor node there is different from the one we propose, since a sensor is able to figure out the position and the direction of a target object by itself and is in an active state throughout its lifetime. Estrin et al. [10] also describe a localized algorithm for coordination among sensor nodes, thereby electing a cluster-head among a group of sensor nodes which does the sensing and the triangulation. Our information collection framework is applicable to such localized algorithms too, since the server module would now reside on a sensor node or a cluster-head.

References

[1] C. Olston, B. T. Loo, and J. Widom. Adaptive precision setting for cached approximate values *ACM Sigmod, 2001*

[2] M. Bhardwaj, A. P Chandrakasan. Bounding the Lifetime of Sensor networks via optimal Role assignments *Proceedings of INFOCOM,2002*

[3] A. Sinha, A. Chandrakasan. Dynamic Power Management in Wireless Sensor networks *IEEE Design and Test of Computers,2001*

[4] G.J. Pottie and W.J. Kaiser. Wireless Integrated network sensors *Communications of the ACM,2000*

[5] R.Min *et al*. An architecture of a power aware distributed microsensor network *IEEE workshop on Signal Processing systems,2000*

[6] T. Rappaport. Wireless Communications:Principles and Practice *Prentice Hall Inc, 1996*

[7] S. Singh and C. S Raghavendra. PAMAS- Power Aware Mult-Access protocol with signaling for adhoc networks *SIGCOMM Computer Communication Review, July 1998*

[8] X. Yu, K. Niyogi, S. Mehrotra and N. Venkatasubramanian. An Adaptive Information Collection Middleware for Sensor Network Applications. *Technical Report TR-DB-02-08, ICS, UC Irvine, 2002*

[9] F. Zhao, J. Shin and J. Reich. Information driven Dynamic Sensor Collaboration for tracking applications *IEEE Signal processing magazine, 2002*

[10] D. Estrin, R. Govindan, J. Heidemann and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks *MobiCOM, 1999*

[11] W. R. Heinzelman, J. Kulik and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks *ACM /IEEE Mobicom Conference, 1999*

[12] C. Olston and J. Widom. Best-Effort Cache Synchronization with Source Cooperation *SIGMOD 2002*

[13] http://www.darpa.mil/ito/research/sensit/index.html