Battle Event Detection using Sensor Networks and Distributed Query Processing

Mira Yun[1], Danielle Bragg[1], Amrinder Arora[2], and Hyeong-Ah Choi[1]
[1]Department of Computer Science, George Washington University, Washington, DC
[2]Research and Development, NTELX, Vienna, VA
Email: {mirayun, hchoi}@gwu.edu, danielle.k.bragg@gmail.com, aarora@ntelx.com

***Abstract*—We consider the problem of identifying battlefield events using sensors deployed in the area. The goal is to alert centralized headquarters about the occurrence of significant events so that it can respond appropriately to the events. We propose a mechanism using which the sensors can exchange information using signatures of events instead of data to save on transmission costs. Further, we propose an algorithm that dynamically generates phases of information exchange based on the cost and selectivity of each filter. We present simulation results that compare the proposed algorithm to other alternatives. Our results show that the proposed algorithm detects events while minimizing the transmission and processing costs at sensors.**

I. Introduction

In wireless sensor networks, information processing is a critical problem due to the huge volume of real-time physical data collected through many diverse sensors [1]. As a powerful application domain of information processing, event detection in sensor networks has received much attention in the wireless research community. The literature provides various event detection schemes, as in [2], [3], and [4]. In these event detection projects, events are typically categorized into two groups: atomic events and compound events defined by [5] and [6]. An atomic event is an event characterized by a single type of physical data, while a compound event is one characterized by multiple types of physical data. Even detecting an atomic event in a constrained network is a significant challenge. Marticic et al. in [7] present a distributed event representation paradigm. This paradigm divides the detection area into cells, gathers data from all nodes in each cell, and computes a weighted average of all data gathered within the cell. Sensor nodes store copies of adjacent cell averages, and can detect an event by matching this submatrix of cell averages to an event signature. For compound events, Abadi et al. extend the TinyDB query processor in [8] to produce an event detection system for sensor networks called REED, presented in [9]. The system supports in-network joins between sensory data and static tables built outside the sensor network. Kumar et al. in [5] propose a distributed event detection framework by considering collaboration among sensor nodes. The framework forms a group of sensor nodes gathering all data types necessary to detect a compound event, and each group reports to the centralized headquarters if the aggregated data satisfies the given conditions.

In this paper, we consider the problem of detecting battle events. Previous work considering sensor networks in battle environments include [10], [11], and [12]. Motivated by the events of September 11, 2001, the Battle of the Water Sensor Networks (BWSN) was held in August 2006 [10], in which fifteen sensor network designs considering the following four design objectives were proposed: 1) minimization of the expected time of event detection, 2) minimization of the expected population affected prior to event detection, 3) minimization of the expected demand of contaminated water prior to event detection, and 4) maximazation of the detection likelihood.

In the case of battle event detection, sensors can be deployed in a battlefield, gathering several types of data. The sensors can take the form of smoke-detectors, noise-detectors, motion-detectors, as well as cameras. We would like to match data gathered around the same time and location to identify significant events that are characterized by specific sensory data and to alert a centralized headquarters about the occurrence of the significant events so that they can respond appropriately to the events. The type of events that we would like to identify can vary; for example, a significant event might be the explosion of a bomb or the deployment of a chemical agent. Each event is characterized by a particular set of sensory data, or "sensory signature". For example, an exploding bomb might be characterized by a bright light, loud sound, and hot temperature, and its sensory signature would reflect these properties.

The rest of the paper is organized as follows. In Section II , we outline the system model and define the problem. In Section III and Section IV, we describe our proposed protocols. Simulation results are shown in Section V, and conclusions and future directions are presented in Section VI.

II. System Model and Problem Formulation

A. Area as a Grid

We model the area that we would like to monitor as a grid of cells. Each cell encompasses a contiguous subsection of the area, every point in the area belongs to exactly one cell. We assume that sensor nodes are deployed throughout the area, and may be located anywhere within the grid. We also assume that each cell is monitored sufficiently by sensors to allow for event detection.

B. Sensor Nodes

We assume that each node deployed in our grid is equipped with specific sensor capabilities. In our work, we assume

978-1-4244-9920-5/11/$26.00 ©2011 IEEE

Fig. 1. Grid representing battleground with deployed sensors

that each node is "homogeneous," having a single sensing capability, as in [13]. Each type of sensor has a particular radius of detection. We also assume that each node has a particular radius of communication determined by the broadcasting power of the node. When the sensing radii of two sensor nodes overlaps, both nodes can capture data on events that occur in the overlapping area. More formally, each sensor node v_i is associated with the following information:

- location: the location within the area grid.
- detection capability: the sensor type located at the node.
- detection radii: the radii within which each sensor can detect data. For example, a temperature sensor is likely to have a much smaller radius of detection than a camera, since temperature is very localized whereas a picture can encapsulate data over a larger area.
- communication radius: the radius within which each sensor can broadcast data.

We assume that sensor nodes only capture data when they are triggered by a relevant stimulus. For example, a motion detector will only capture data when movement is observed within a certain proximity.

C. Cell Status as a Sensory Signature

The status of each cell in the grid is modeled by a "sensory signature" of bits, similar to the event hierarchy paradigm presented in [6]. Each bit indicates the data that sensors have gathered about a specific sensory property. For example, the first bit might indicate the presence of smoke.

Each type of battle event corresponds to a signature. These bits are relevant because the value of these bits determine whether or not the event has been perceived. For example, the bits in a sensory signature that corresponds to the explosion of a bomb might indicate the presence of smoke and noise above a particular decibel.

The bits in a sensory signature can take one of the following four values: Y, N, X, and ?. "Y" means that data has been collected indicating that the corresponding property is present. Conversely, "N" means that data has been collected indicating that the corresponding property is not present. "X" means that contradictory data has been collected, some of which indicate that the corresponding property is present and some of which indicate that the property is not present. Finally, "?" means that no data has yet been collected.

Caveat on the use of signature: While our idea of using signature bits is quite effective, and saves significant amounts of transmission power, it suffers from one drawback - we cannot distinguish between two sets of events in the same battle cell, if the two sets of events lead to the same composite signature. As an example, consider one set of two events with bit signatures "???YYY" and "YYY???", and another set with a single event with individual bit signature "YYYYYY". Both of these sets are recognized as a single compound event with bit signature "YYYYYY" using our signature based protocols. This drawback is not a limitation in the business problem being considered as multiple events are also of significant interest to a battlefield commander. However, we recognize that such signature based protocols may not be applicable in all scenarios.

D. Query Protocol

We assume that all event queries originate from a centralized headquarters node. This headquarters "asks" the network to look out for the occurrence of a particular event by specifying the corresponding sensory signature. For example, suppose the headquarters wishes to know if a bomb has exploded in the network area, and an explosion is characterized by smoke, noise above a particular decibel, and a flash of light. The headquarters specifies a signature with three bits set indicating sufficient presence of smoke, noise, and light, respectively. It then sends its query out to the network. Sensor nodes continue to monitor the landscape until a new query is received from the headquarters.

E. Problem Formulation

Given:

- an area grid with a set of sensor nodes $\{v_1, ..., v_n\}$
- set of filters k, where each sensor node can have one of these k filters
- the query signature of k 0/1 bits which corresponds to significant event
- the maximum latency for an event detection

Find: all events in the area grid that satisfy the sensory signature of the significant event within the maximum latency time. As we discuss in later sections, the event query and the latency can change dynamically at the discretion of the headquarters node.

F. Cost Model

We assume that the sensor nodes are equipped with the capacity to filter the data they collect. More specifically, a filter takes the data that has been captured in its original form, and converts it to a true or false value based on a test. For example, a filter for pictures of smoke would reduce a picture of the ashy air around a burning building to a true value, and a picture of a river to a false value. These true and false values

correspond to the "Y" and "N" values that can be placed in the sensory signature.

The performance of these filters is one factor that impacts the total cost of query processing in the network. So to evaluate the cost of our proposed solutions, we define the following terms:

- **filter cost** c_i: the computation cost of executing filter F_i. We normalize these costs to account for differences in the nodes' total power by dividing the computational power required to run the filter by the total power of the node to find c_i. We expect some filters to have a higher computation cost than others, depending on the complexity of the captured data. For example, a filter acting on a picture will typically be more expensive than another acting on a temperature reading.
- **filter selectivity** s_i: the fraction of sensory data that can be expected to pass the criteria of filter F_i [14]. For example, if we expect 2 out of 10 sound readings to measure sounds above 60 dB, a filter for sounds above 60 dB will have a selectivity of 20%. We assume that filters are independent, so that the selectivity of any given filter is not affected by the prior execution of other filters.
- **global broadcasting cost** B_G: the cost of broadcasting data records to all other nodes in the network. We assume a standard broadcasting protocol, where the time and cost of transmitting data throughout the network is a function of the network.

III. Proposed Solution

To improve overall network performance, we would like to discard data that is determined to be irrelevant to avoid unnecessary broadcasting and processing. However, it is difficult to ensure that we can discard data safely. If the detection radius of a node is larger than its communication radius, it may be able to detect events that are within the detection radius of another node, and still have no direct communication with that node. The data must be sent through intermediate nodes in this case.

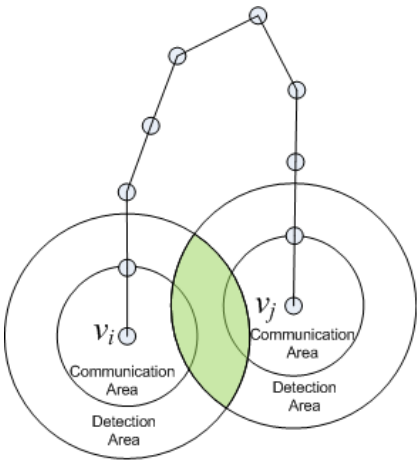

Fig. 2. Overlapping detection areas without communication

Figure 2 provides an example of a network where nodes v_i and v_j have overlapping detection areas. As a result of this overlap, data that originates at v_i might be relevant to data that originates at v_j, and vice versa. Intermediary nodes on the path from v_i to v_j might be tempted to discard data that has no relevance to the intermediary nodes, as the data is sent to nodes even further from the site of detection. However, as we observe, this data cannot be discarded safely, as it is relevant to data gathered at v_j.

A. Updating Sensory Signatures

We also must address the problem of updating the sensory signature of a cell to incorporate data gathered by several sensor nodes. Given two input signatures, we combine each bit using the $\oplus$ operator, defined as follows:

$\oplus$	Y	N	?	X
Y	Y	X	Y	X
N	X	N	N	X
?	Y	N	?	X
X	X	X	X	X

TABLE I
COMBINING BITS IN SENSORY SIGNATURES

In addition to handling the "joining" of two individual signatures, we must also handle joining streams of sensory signatures. When sensory nodes gather data, the time of data collection is recorded. To facilitate keeping track of the time and location of data collection, we define a data structure, which is a tuple consisting of the following information: $\ll timestamp, location, sensory\ signature \gg$.

A primary advantage of using such a data structure is the pairing of time with the gathered sensory data. In order to detect an event, we would intuitively like to consolidate records with the same time and location that provide data gathered at different nodes. However, we note that it can be difficult to pair sensory data with the appropriate time.

One possibility is to match sensory data with the time at which the data was gathered. However, such a pairing might be rendered ineffective when we combine data gathered at different locations and by different types of sensors. For example, if one sensor node is twice as close to an event as another sensor node, the data gathered at each note will possess different time stamps, even though they correspond to the same event. This problem is further complicated by the fact that different physical properties travel at varying speeds. For example, sound waves travel more slowly than light. As a result, even when sensors at the same sensory node gather data on sound and light pertaining to the same event, the time stamps will be different.

Furthermore, because time is a continuous variable, there are an infinite number of possible time values. For example, two data records that correspond to the same event might display times t and $t + \epsilon$. Yet they might not be consolidated because their times are different. One potential solution to this problem of continuity would be to create data records

with only standardized times. The set of potential times would be of the form $t + a\epsilon_{time}$, where t is the time when the system is put in place, a is an integer from 0 to ∞, and ϵ_{time} is a predetermined constant specifying the interval between consecutive potential times. Then each data record would be assigned the time of form $t+a\epsilon_{time}$ that is closest to the actual detection time.

One practical solution to this problem of consolidating sensory data by time is to make use of the nodes' memory. If a node receives a sensory signature, and has its own sensory data stored in recent memory, we can assume that these pieces of data match.

B. Distributed Query Processing Protocol

In this subsection, we present an innovative battle event detection protocol using distributed query processing with sensory signature.

We assume that the sensor nodes are aware of the query signature that is of interest to the head quarters. The algorithm divides the filters into several phases which depend on the cost and the selectivity of various filters as well as the query signature. In each phase, only a set of filters is active. The filters belonging to the first phase are active at all times. The filters belonging to the subsequent phases are active only when an event matching the signature of the previous filters has been received. As the sensors receive the event messages from their neighbors, they consolidate and maintain a partial signature for each event received. When the partial signature of some events in their sensing radius matches the signature of the prior phases, the sensor becomes active. The sensors that have filters belonging to the first phase do not perform the consolidation and maintenance steps. The steps of the algorithm are shown in Algorithm 1.

Algorithm 1 Distributed Query Processing Protocol

Compute and Broadcast Step
for each phase i distributed with the query **do**
- Run the ith class of filters on data pertaining to cells with partial sensory signatures satisfying the query specifications.
- Update the partial sensory signature for the monitored cells with the filter results.
- Broadcast updated signature records to all neighbors.
- Update signature records based on received records.

end for
Action Step The headquarters node receives all complete sensory signatures satisfying the query specifications, and takes appropriate action.

1) Algorithm for computing phases: In order to optimize the performance of Algorithm 1 in terms of total cost, the optimal number of phases must be found. This equates to determining the number of filter classes and which filters belong to each class. Since the central headquarter node generates event queries, it naturally follows that the headquarters should determine the grouping of filters into phases, and distribute this information along with the query.

In order to do this, we provide a dynamic programming algorithm. We first order the n filters necessary to process the meaningful bits specified by the event query, so that $\{F_1, ..., F_n\}$ where $\frac{c_1}{1-s_1} \leq \frac{c_2}{1-s_2} \leq ... \leq \frac{c_n}{1-s_n}$. This ranking is derived from the work done in [14]. We also define the cost function $c(F_x, F_y, k)$ to be the cost of running filters F_x through F_y using at most k phases.

In the following algorithm, k^* is the maximum number of phases possible that satisfy the maximum allowed latency for an event detection. For example, suppose it takes $t_{broadcast}$ time to broadcast data to all nodes in the network, $t_{filters}$ time to run all filters, and we want the headquarters to be alerted about events with a maximum latency of $t_{latency}$ time. Then the maximum number of phases possible is $k^* = \lfloor \frac{t_{latency} - t_{filters}}{t_{broadcast}} \rfloor + 1$, since $(k^* - 1)t_{broadcast} + t_{filters} \leq t_{latency}$. We further observe that k^* cannot exceed k, as there must be at least one filter in each phase.

Algorithm 2 Dynamic Programming Algorithm to Determine the Optimal Number of Phases.

for $x = 1$ to n, and $y = x$ to n **do**
  $c(F_x, F_y, 0) = \sum_{i=x}^{y} c_i$ since this is simply the sum cost of running filters F_x through F_y.
end for
for $k_1 = 1$ to k^* **do**
  for $x = 1$ to n, and $y = x$ to n **do**
    $c(F_x, F_y, k_1) =$
    $$\min_{z=x}^{y} \left\{ c(F_x, F_z, k_1 - 1) + t_{broadcast} + \prod_{i=x}^{z} s_i c(F_{z+1}, F_y, 0)) \right\}$$
  end for
end for
return $\min_{k_1=1}^{k^*} c(F_1, F_n, k_1)$

Time Analysis: Algorithm 2 runs in $O(n^4k)$ time where n is the number of sensor nodes, and k is the number of filter types, since the second loop runs at most $O(n^2k)$ times, the min function inside must compare at most n values, and the inner product must multiply at most n elements. However, we can reduce this run time to $O(n^3k)$ by computing the possible product values $\prod_{i=x}^{z} s_i$ for all values of x and z in $O(n^2)$ time once before entering the loop.

IV. Localized Query Processing Protocol

Depending on the landscape and sensor types, the relation between the computation cost and communication cost can be different. The Distributed Query Processing Protocol presented in Section III focuses on minimizing the computation cost. However, if the communication cost dominates computation cost, then a different approach can be helpful.

In this section, we present a leader based protocol which focuses on reduce communication cost. Each cell has a leader node that gathers the partial sensory signatures from all cell members. In this localized protocol, all filters are active at all times. When a node runs a filter, it transmits the updated records to the local leader node. Nodes receive, process and forward the event signatures based on following two rules:

- A leader node only processes information from its local nodes or other leader nodes
- Non-leader nodes do not process or forward any information.

The steps of this algorithm are shown in Algorithm 3.

Algorithm 3 Localized Query Processing Protocol

Compute and Broadcast Step

1) Nodes process events that trigger their corresponding filters.
2) Nodes transmit the event signatures to their local leader nodes.
3) Leader nodes update signature records based on received records and transmit updated signature records to the other leader nodes.

Action Step The headquarters node receives all complete sensory signatures satisfying the query specifications, and takes appropriate action.

V. Simulation and Results

In our simulation, we deploy nodes with varying sensing capabilities and detection radii. Sensing capabilities of the nodes are coupled with filtering capabilities, which have associated costs. We assume that we have 10 basic types of data. The costs and detection radii of the corresponding filters are shown in Table II. Table II provides a mapping of these capabilities to real sensor node classifications, extrapolated from the sensor capabilities presented in [13]. We normalize the distribution of these sensor nodes so that the area is monitored by the same number of each type of sensor, as shown in Figure 3. Accordingly, the number of sensors deployed with a given detection radius is inversely proportional to the square of the radius. The communication radius of each sensor node is randomly selected from $[10, 100]$. In each simulation, we introduce 10 events, and a random number of these 10 have the matching event signature for our query.

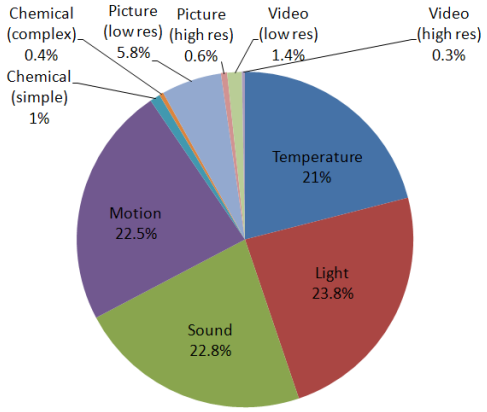

Fig. 3. Distribution of sensors

In each of our simulations, we compare three protocols: the distributed protocol and localized protocol described in this work, as well as a naive protocol. This naive protocol consists of filtering and broadcasting all data as soon as it is gathered at the sensors. These protocols are compared in terms of cost. Since power resources are limited and crucial for sensor networks, we model our cost function to represent power. This includes computational power to run filters and gather data, and communication power to transmit and receive data.

Filter	Sample Processor	Cost	Detection Radius
Picture (low res)	Cyclops-ATmega128L MCU+CPLD	100	20
Picture (high res)	MeshEye-ARM7TDMI based on ATMEL	600	50
Temperature	Mica2Dot-ATmega128L 4MHz	1	10
Light	Mica2Dot-ATmega128L 4MHz	1	10
Sound	FireFly-ATmega128L 8MHz	5	10
Motion	Imote2-PXA271 XScale 13MHz	20	10
Chemical (simple)	MeshEye-ARM7TDMI based on ATMEL	600	50
Chemical (complex)	CITRIC-Intel XScale PXA270 CPU	3000	100
Video (low res)	MeshEye-ARM7TDMI based on ATMEL	600	50
Video (high res)	CITRIC-Intel XScale PXA270 CPU	3000	100

TABLE II
Sensor filter capabilities

A. Protocol Effects on Total Power

First, we analyzed the effect of the environment size on the total power cost of our algorithms. Figure 4 presents the simulation results reflecting the impact of the environment size, in terms of number of sensors, on the total cost, which includes the cost of transmission, reception, and computation. We see that both our distributed and localized protocols present significant improvements over the naive protocol. As the environment area grows, our protocols are increasingly beneficial, because they avoid processing and transmitting increasingly large amounts of data through an increasingly large area.

More specifically, the naive protocol transmits the original gathered data to the headquarters whenever that data satisfies the corresponding filter. We assume that the size of this original sensory data is proportional to the cost of the filter. Because our protocols reduce the original data to a single sensory signature, the amount of data that passes through the network is greatly reduced. Figure 4 demonstrates the scalability of our solutions, in comparison to the naive solution.

We note that the total power cost includes the cost of both computation and communication. Because communication costs contribute a large fraction of the total cost in sensor networks, our localized protocol outperforms the distributed protocol, since it transmits signatures to local leaders only.

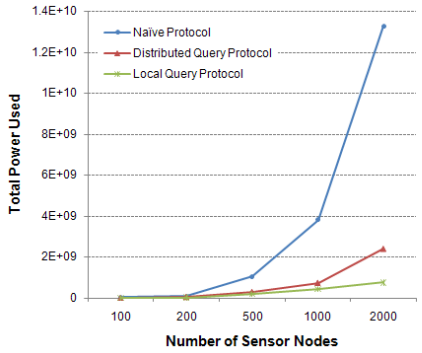

Fig. 4. Total Power Used by 3 protocols, as a function of network size.

B. Protocol Effects on Computational Power

Finally, we explore the effect of network size on the computational power used by the sensor nodes. As the computational capabilities of sensor nodes dramatically improve, the cost of computation at sensor nodes becomes increasingly important. Therefore, we isolate and explore computational performance.

Figure 5 presents our simulation results, using the same simulation settings described above. As shown in the graph, our distributed query protocol reduces the consumption of computational power through its use of phases. Since our protocol runs filters in groups, only running filters assigned to later phases for potential event sites, it avoids running these filters excessively. In our simulation settings, our protocol was optimized with two phases. Our results thus demonstrate that even dividing the filters into two phases saves a considerable amount of computational power.

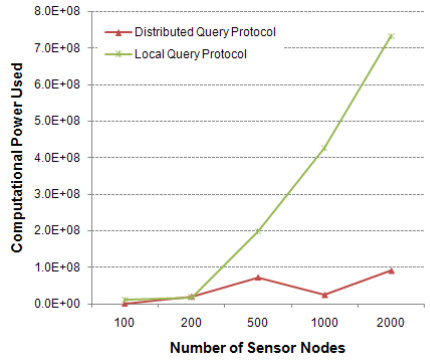

Fig. 5. Protocol Effects on Computational Power

VI. Conclusion

In this work, we considered the problem of identifying battlefield events using sensors deployed in the area. We presented two protocols that can reduce the computation cost and the communication cost for the sensor network, thereby extending the life of the network. The suggested protocols employ three main techniques to reduce these costs: (i) use of sensory signature to avoid sending raw or processed data, (ii) use of phases to avoid unnecessary computations, and (iii) use of leader nodes to avoid unnecessary communications.

Future Work: Future work in this field can consist of using sensor network to detect multiple types of events simultaneously. For example, we may want to detect bomb explosions, gunfire and chemical agent deployment simultaneously.

Future work can also consist of combining the helpful attributes of the two protocols suggested in this paper. Distributed query processing protocol reduces computation cost, and localized query processing protocol reduces communication cost. A protocol that combines aspects of these two protocols could possibly reduce both , and still detect events correctly.

References

[1] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach*, Morgan Kaufman, 2004.

[2] Dan Li, K.D. Wong, Yu Hen Hu, and A.M. Sayeed, "Detection, classification, and tracking of targets," *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 17 –29, Mar. 2002.

[3] Bhaskar Krishnamachari and Sitharama Iyengar, "Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 53, pp. 241–250, 2004.

[4] R.R. Brooks, P. Ramanathan, and A.M. Sayeed, "Distributed target classification and tracking in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163 – 1171, 2003.

[5] A V U Phani Kumar, Adi Mallikarjuna Reddy V, and D. Janakiram, "Distributed collaboration for event detection in wireless sensor networks," in *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, New York, NY, USA, 2005, MPAC '05, pp. 1–8, ACM.

[6] Shuoqi Li, Ying Lin, Sang H. Son, John A. Stankovic, and Yuan Wei, "Event detection services using data service middleware in distributed sensor networks," *Telecommunication Systems*, vol. 26, pp. 351–368, 2004, 10.1023/B:TELS.0000029046.79337.8f.

[7] Fernando Martincic and Loren Schwiebert, "Distributed event detection in sensor networks," *Systems and Networks Communication, International Conference on*, vol. 0, pp. 43, 2006.

[8] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, pp. 122–173, March 2005.

[9] Daniel J. Abadi, Samuel Madden, and Wolfgang Lindner, "Reed: robust, efficient filtering and event detection in sensor networks," in *Proceedings of the 31st international conference on Very large data bases*. 2005, VLDB '05, pp. 769–780, VLDB Endowment.

[10] Avi Ostfeld, James G Uber, Elad Salomons, Jonathan W Berry, William E Hart, Cindy A Phillips, Jean-Paul Watson, Gianluca Dorini, Philip Jonkergouw, Zoran Kapelan, and et al., "The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms," *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 556, 2008.

[11] Gyula Simon, Miklós Maróti, Ákos Lédeczi, György Balogh, Branislav Kusy, András Nádas, Gábor Pap, János Sallai, and Ken Frampton, "Sensor network-based countersniper system," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, New York, NY, USA, 2004, SenSys '04, pp. 1–12, ACM.

[12] Anthony D. Wood, John A. Stankovic, and Sang H. Son, "Jam: A jammed-area mapping service for sensor networks," *Real-Time Systems Symposium, IEEE International*, vol. 0, pp. 286, 2003.

[13] Islam T. Almalkawi, Manel Guerrero Zapata, Jamal N. Al-Karaki, and Julian Morillo-Pozo, "Wireless multimedia sensor networks: Current trends and future directions," *Sensors*, vol. 10, no. 7, pp. 6662–6717, 2010.

[14] U. Srivastava, K. Munagala, and J. Widom, "Operator placement for in-network stream query processing," in *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2005, pp. 250–258.